Digitally Mediated Design:

Using Computer Programming to Develop a Personal Design Process

by

Megan Yakeley

Diploma of Architecture, RIBA Part II, University of East London, UK, 1993

MSc, Architecture (Computing and Design), University of East London, UK, 1992

BA (Hons) Architectural Studies, RIBA Part I, Oxford Brookes University, UK, 1986

Submitted to the Department of Architecture
in Partial Fulfilment of the Requirements for the Degree of
Doctor of Philosophy in the field of
Architecture: Design and Computation

at the

Massachusetts Institute of Technology

June 2000

© 2000 Megan Yakeley. All Rights Reserved

The Author hereby grants to MIT permission to reproduce and to distribute publicly
paper and electronic copies of this thesis document in whole or in part

Signature of Author ____

Department of Architecture
9th March 2000

Certified by ____

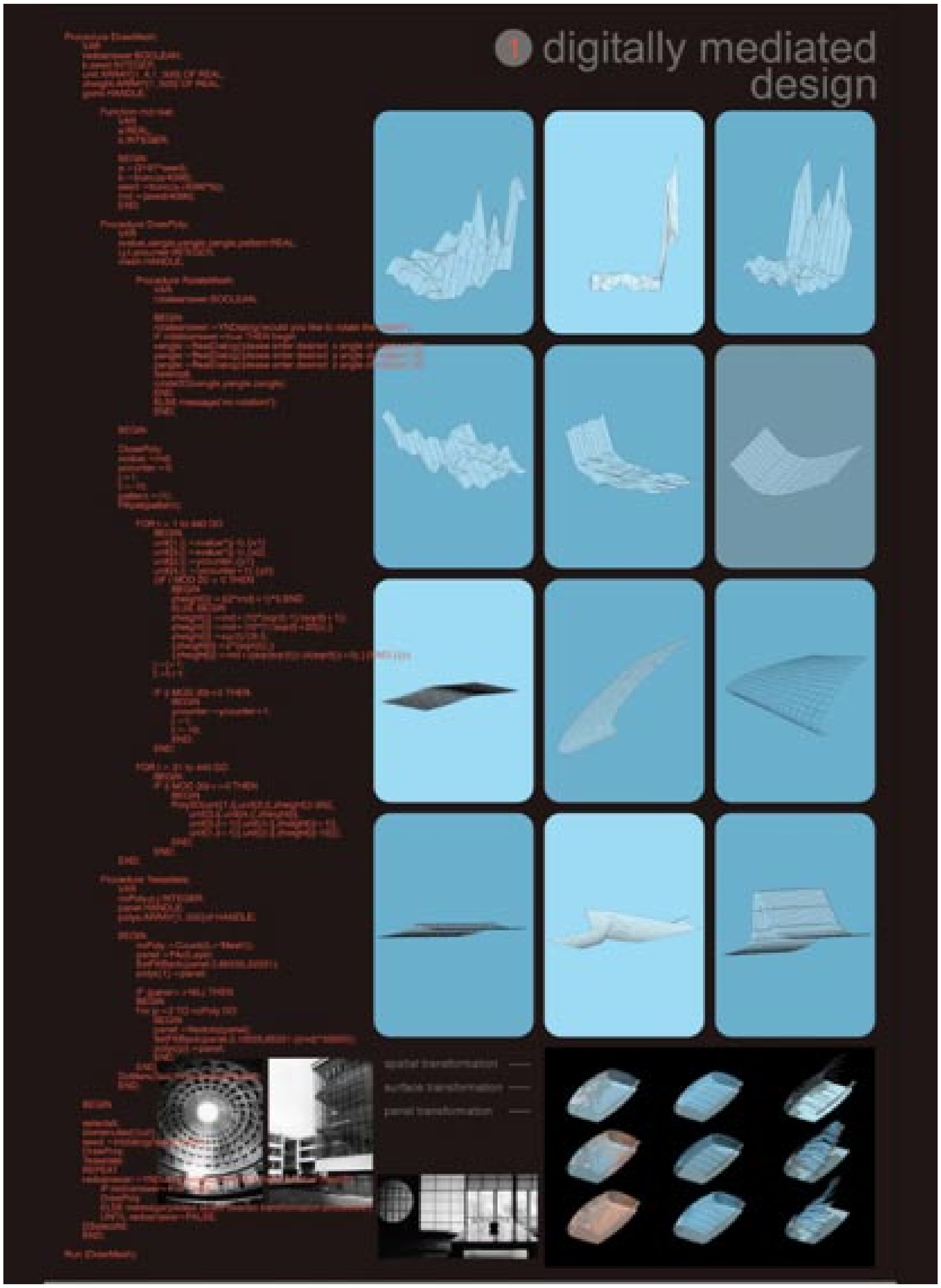
William J. Mitchell, Professor of Architecture and Media Arts and Sciences
Dean, School of Architecture and Planning
Chair, Dissertation Committee

Accepted by ____

Stanford Anderson, Professor of History and Architecture,
Head of the Department , Department of Architecture
Chair, Committee on Graduate Students

To the light shining on the path between
the duck pond and the swan lake
with gratitude beyond words

Digitally Mediated Design: Using Computer Programming to Develop a Personal Design Process



Advisory Committee

Committee Chair

William J. Mitchell, Dean, School of Architecture and Planning, MIT

Committee Members

Terry Knight, Associate Professor, Department of Architecture, MIT

Mitch Resnick, Associate Professor, Media Laboratory, MIT

Terry Tivnan, Professor, Harvard Graduate School of Education

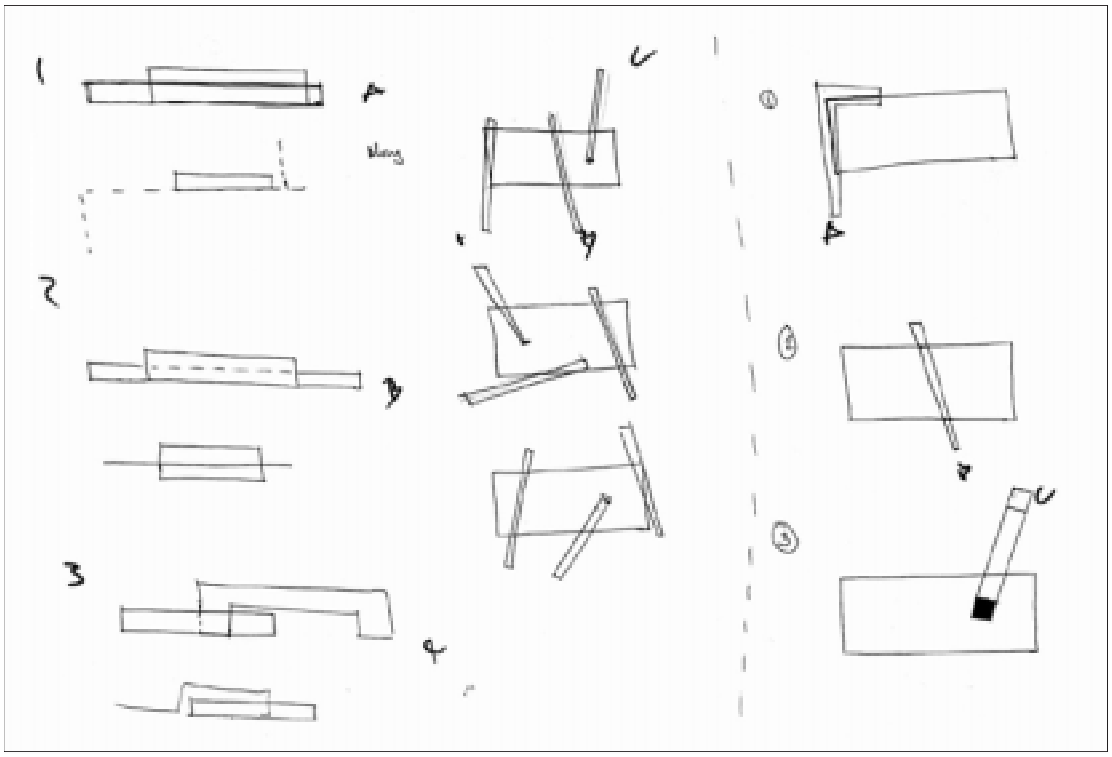


Figure 1. Anthony Guma was the only student to sketch during the interviews. Here he tries to explain to himself the rules for his project in graphical form, prior to verbal description and then writing in code.

Digitally Mediated Design:
Using Computer Programming to Develop a Personal Design Process

by

Megan Yakeley

Submitted to the Department of Architecture on 9th March 2000
in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy
in the field of Architecture: Design and Computation

Abstract

This thesis is based on the proposal that the current system of architectural design education confuses product and process. Students are assessed through, and therefore concentrate on, the former whilst the latter is left in many cases to chance. This thesis describes a new course taught by the author at MIT for the last three years whose aim is to teach the design process away from the complexities inherent in the studio system. This course draws a parallel between the design process and the Constructionist view of learning, and asserts that the design process is a constant learning activity. Therefore, learning about the design process necessarily involves learning the cognitive skills of this theoretical approach to education. These include concrete thinking and the creation of external artifacts to develop of ideas through iterative, experimental, incremental exploration. The course mimics the Constructionist model of using the computer programming environment LOGO to teach mathematics. It uses computer programming in a CAD environment, and specifically the development of a generative system, to teach the design process.

The efficacy of such an approach to architectural design education has been studied using methodologies from educational research. The research design used an emergent qualitative model, employing Maykut and Morehouse's "interpretive descriptive" approach (Maykut & Morehouse, 1994) and Glaser and Strauss's Constant Comparative Method of data analysis (Glaser & Strauss, 1967). Six students joined the course in the Spring 1999 semester. The experience of these students, what and how they learned, and whether this understanding was transferred to other areas of their educational process, were studied. The findings demonstrated that computer programming in a particular pedagogical framework, can help transform the way in which students understand the process of designing. The following changes were observed in the students during the course of the year:

Development of understanding of a personalized design process; move from using computer programming to solve quantifiable problems to using it to support qualitative design decisions; change in understanding of the paradigm for computers in the design process; awareness of the importance of intrapersonal and interpersonal communication skills; change in expectations of, their sense of control over, and appropriation of, the computer in the design process; evidence of transference of cognitive skills; change from a Behaviourist to a Constructionist model of learning

Thesis Supervisor: William J. Mitchell
Title: Professor of Architecture and Media Arts and Sciences, School of Architecture and Planning

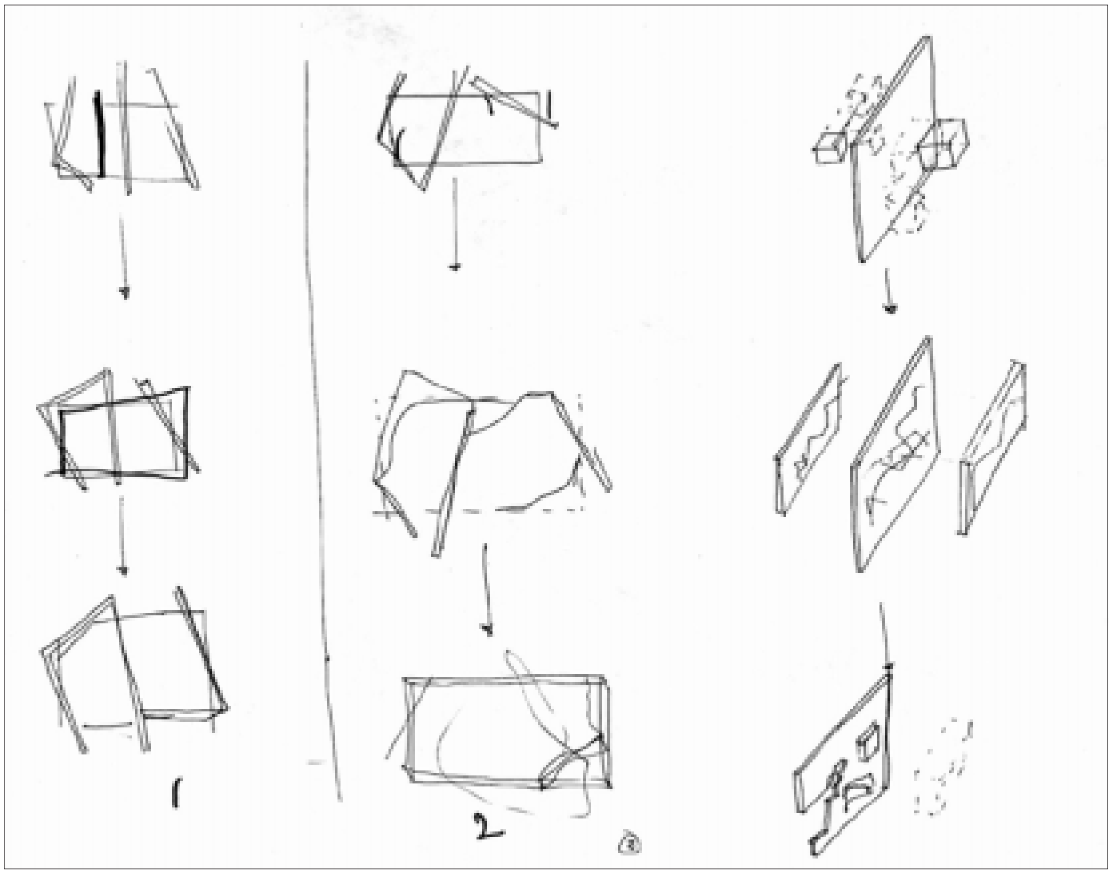


Figure 2. Anthony Guma was the only student to sketch during the interviews. Here he tries to explain to himself the rules for his project in graphical form, prior to verbal description and then writing in code.

Acknowledgements

There are many, many people I would like to thank for their support over the four years it took to complete this thesis. In chronological order, they include

Paul Coates, Head, Centre for Environment and Computing in Architecture, University of East London, for starting me on this journey ten years ago

Paul Richens, Head, Martin Centre, Department of Architecture, University of Cambridge, for recommending me to Bill, to Foster & Partners, and acting as a mentor throughout

William J. Mitchell, Dean, School of Architecture and Planning, MIT, for inviting me to MIT and giving me financial support, being my advisor, and introducing me to Elvis

Terry Knight, Professor, Department of Architecture, MIT, for being a critical member of my committee, for supporting me so extensively, and being a valuable mentor and friend

Mitch Resnick, Professor, Media Laboratory, MIT, for giving up his time to be on my committee and offer helpful advice throughout, despite his incredibly busy schedule

Terry Tivnan, Professor, Harvard Graduate School of Education, for overseeing the methodology of the research study and fitting me around his extensive teaching commitments

Stan Anderson, Head, Department of Architecture, MIT, for supporting my efforts in teaching the class and giving me the Instructor–G appointment to enable me to do so

Renée Caso, Department of Architecture, MIT, administrator for the PhD students and for scheduling classes (and a lot else), for her phenomenal organisation, help, and friendship

Tom Fitzgerald, Department of Architecture, MIT, network manager 'extraordinaire', for his efforts in creating the new digital classroom, running extensive equipment on minimal resources, and sense of humour

Esther, Gary, Hannibal, Jeremy, Josephine and **Raphael**, the students in the study (not their real names), who were so wonderful and gave me so much of their time so freely for the interviews

Talia Braude, the fantastic TA, without whom none of the students would have learned as much as they did, and none of the research would have gone so smoothly

Fernando Domeyko, and all the reviewers, who came to the class's juries and contributed so much towards creating a 'Knowledge Building Community'

Nancy Jones and Anne Simunovic, for their excellent work transcribing some of the interviews

I would also like to thank all of my family and friends who have given much needed support. In particular, I would like to single out: **All four of my parents**, for their love, support, and discrete financial aid in times of dire need, **Tess**, for giving Rotis such a wonderful foster home in my absence, **Jeff**, for always believing in me, and providing a foster home to my piano, **Birgul**, for being so supportive despite having her own PhD to do, and finally, **Roisin, Jack, Jim, Luke, Mia,** and **Ellie**, all born during the last four years, and whose photographs on my desk have reminded me that there is more to life than this.

Contents

Illustrations

For purposes of anonymity, all the students were given aliases when referred to in the text and when being quoted from the interviews. All the students were asked if they would like to be credited for their illustrations. Some requested their own names, whilst others requested an alias. The latter are different from those in the text. Therefore there is no link between the text and the images,
so it is not possible to determine which alias refers to which student through the work they produced. It is possible to read the captions to the illustrations as a separate story from the main text.

Anthony Guma	Figures 1 to 20
Christopher Mulvey	Figures 21 to 43
Debbie Kim	Figures 44 to 46
Hiroshi Okamoto	Figures 47 to 56
Henrietta Green	Figures 57 to 60
Josh Carver	Figures 61 to 64
Michele Auer	Figures 65 to 68
Talia Braude	Figures 69 to 72

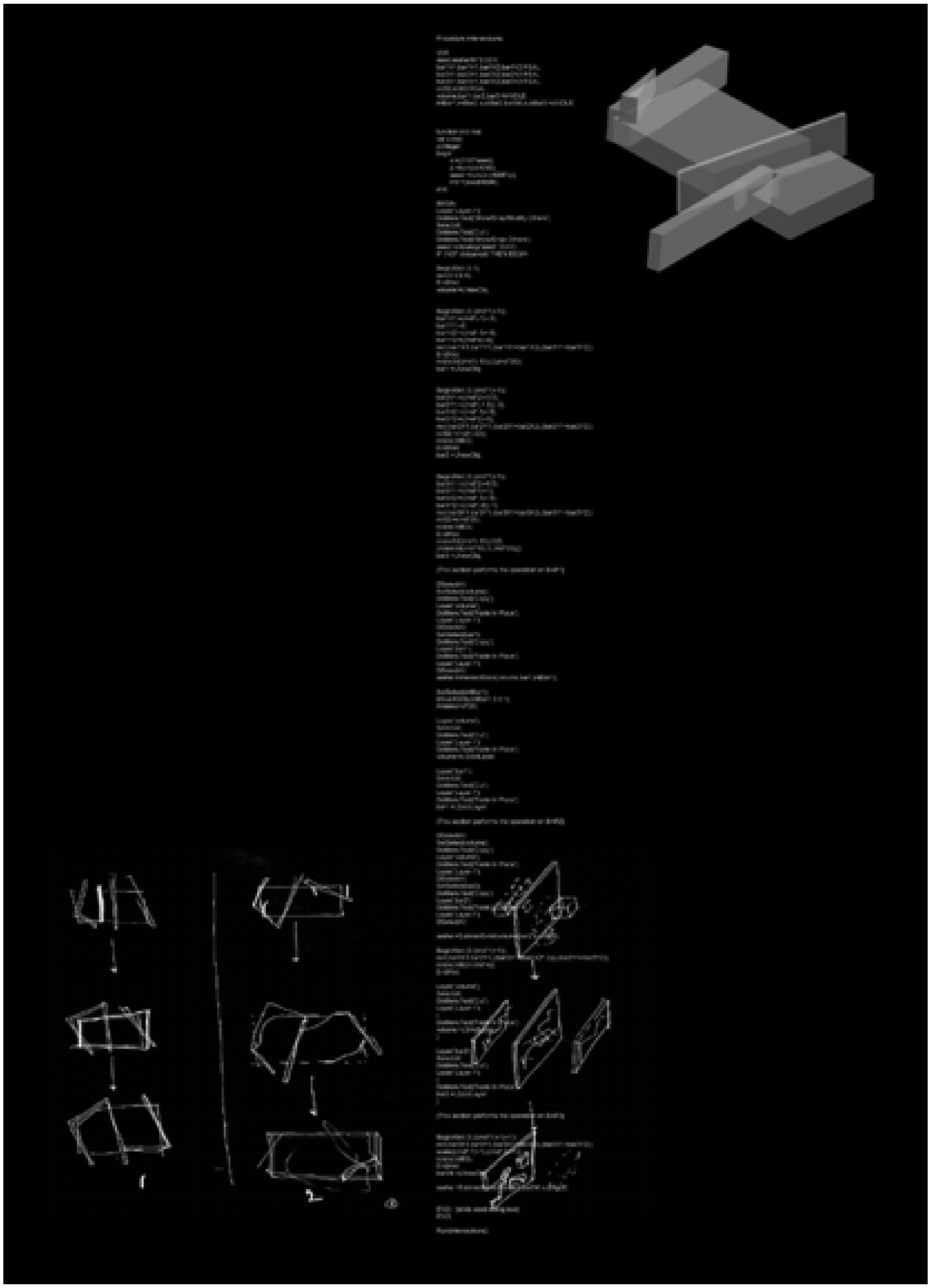


Figure 3. Anthony Guma's first poster, showing his sketches, the code, and an initial output of the code. Students almost always present in the first poster a combination of the languages they are using

Introduction

This thesis examines the potential for digital technology in the education of architects. Recent developments in the sophistication of computers in design have led to both an increased expectation from practice for architects trained to use computers in new ways, and new opportunities for the education of those architects. These new opportunities themselves offer the chance to reassess architectural education, in particular the teaching of design and the design process, as well as the opportunity to train architects to meet the demands of practice for the 21st century. This thesis has drawn parallels between design and education, and has looked towards recent developments in both the theory and methodology of educational research. There are two reasons for the link made in this thesis between education and design. Firstly, this thesis is examining the role of digital technology, and in particular the role of computer programming, in the teaching of design. It examines the efficacy of an intervention in the education process of architects. It therefore makes sense to draw upon educational theories of teaching and learning, and educational methodologies of research, and apply them to, and develop them for, architectural education (Grossman, 1990, Nadimi, 1996).

However, there is a second and more fundamental link between education and design. This thesis begins with the premise that design is a constant learning activity within a Constructionist framework. Therefore the most fundamental aspect of architectural design education must be to teach students, or to help them relearn, the Constructionist model of learning many of them used in early years of school. It is this model of learning, and more specifically the Constructionist techniques associated with it, that are so necessary for a successful design process. The specific techniques include the ability to use objects – the representations of designs in graphical form – to develop design ideas, with both internal and external images. They also include the ability to develop design ideas through progressive tinkering and experimentation with these objects, and a willingness to develop the design problem with its potential solution simultaneously. The links between these skills and those promoted by the Constructionist means that understanding the latter’s theories of education and learning will aid not only the design education process, but the design process itself for these students.

The architectural education system centres around the design studio. This model of learning is based on an old apprenticeship system, and thus can be viewed as a potentially Constructionist learning environment. It is a model for learning about design that has existed for a long time. But it has been more recently recognized that it is a model that is not without its problems. Students entering the design studio for the first time are often confused about what is expected of them.

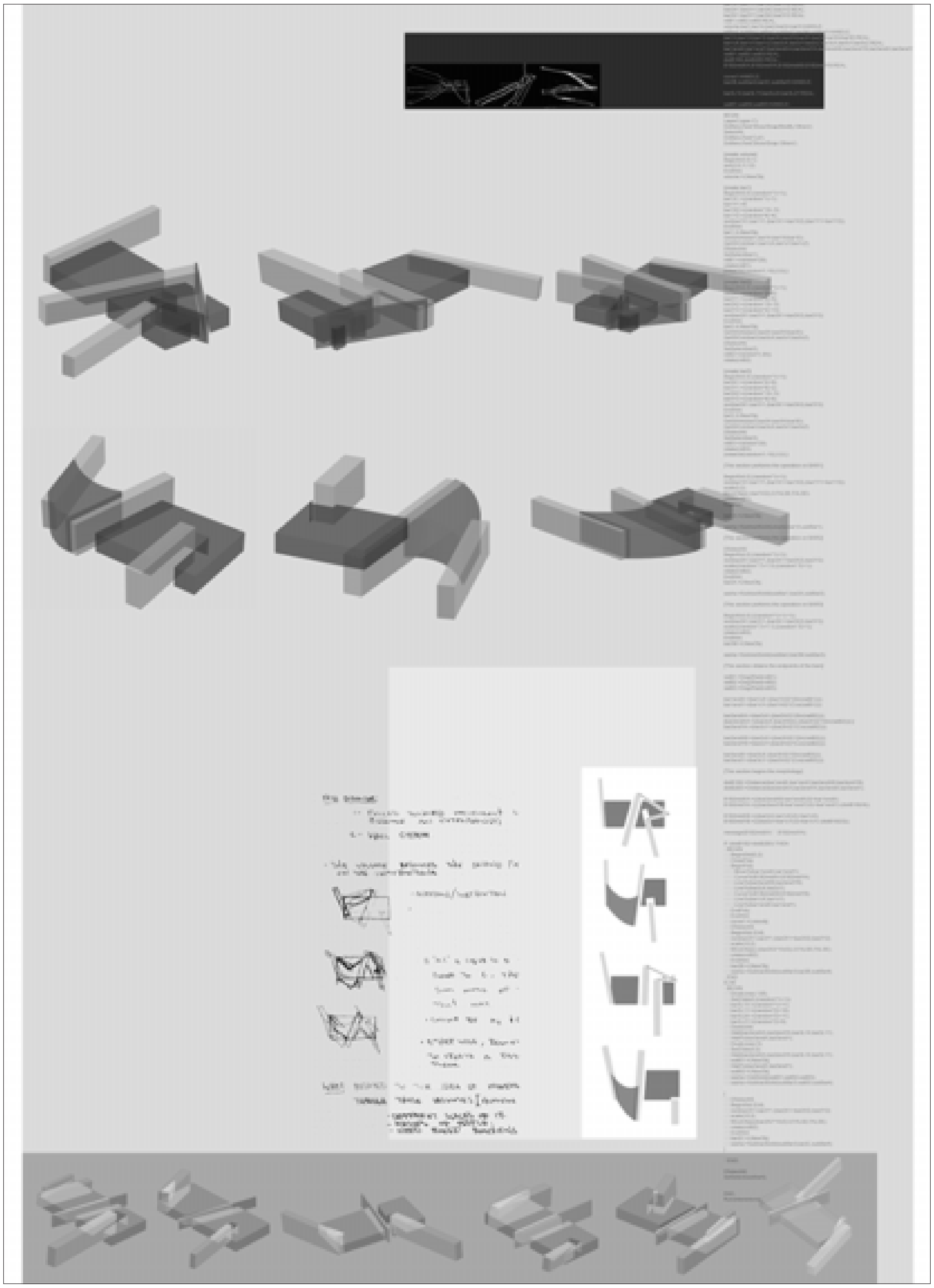


Figure 4. Anthony Guma's second poster, showing the development of the rules. The long elements intersect with the block and relationships between the former cause the latter to deform.

They do not work on real buildings, but rather representations of those buildings. The success of their designs is judged through these products. As such they quickly learn that the products of the studio have an importance that outweighs the process. Developing an understanding of the cognitive aspects of designing, that parallel those used in a Constructionist framework of learning is a by-product rather than the central aim of the design studio system.

Building upon the link between the cognitive aspects of designing and those required for the Constructionist learning process, a course was developed by the author and taught at MIT to enable students to learn about and develop their own design process away from the studio environment. This course uses computer programming within a CAD framework, in much the same way as the Constructionist Seymour Papert and some of his followers have used LOGO. The opportunities created by this course are twofold. Firstly the separation of process from product has significant pedagogical benefits. Secondly, because the computer invites a very personalised method of working ‘with the materials of the situation’, mimicking the cognitive skills required in the design studio, the students develop a closeness to the tool that permits them to use it as a tool to think with in the process of designing. This skill, a by-product of the course, is nevertheless a necessary skill for the 21st century architect, and is now a demand of practice.

The uptake of digital media in architectural design practice has largely been slow. It is only very recently with major improvements in both speed and interface of architectural software that many designers have begun to see real potential for their use in innovative design. The computer is beginning to be used by architects to permit them to do things not previously possible. Many of the computational functions used in these processes are not new, and have been widely available but largely unused. What has changed however, and is still in the process of changing, is the perception of the profession of what is possible, and of what digital media can and should be used for in design. This change in perception has been slow but is finally beginning to gather momentum. The change in speed is fuelled in part by the new, younger generation more aware of what might be possible than many of their employers. The most effective architectural practices, however, are grasping the nettle of this discrepancy in knowledge and understanding, and using it to push forward existing boundaries of technology in design.

Architectural education, on the other hand, lacks any major external impetus to follow suit and upgrade its approach to digital media in design. The result is an ever increasing ground swell of pressure from technologically astute students expressing desires to learn more about what is possible, against an often technophobic older academic body in the middle and upper levels

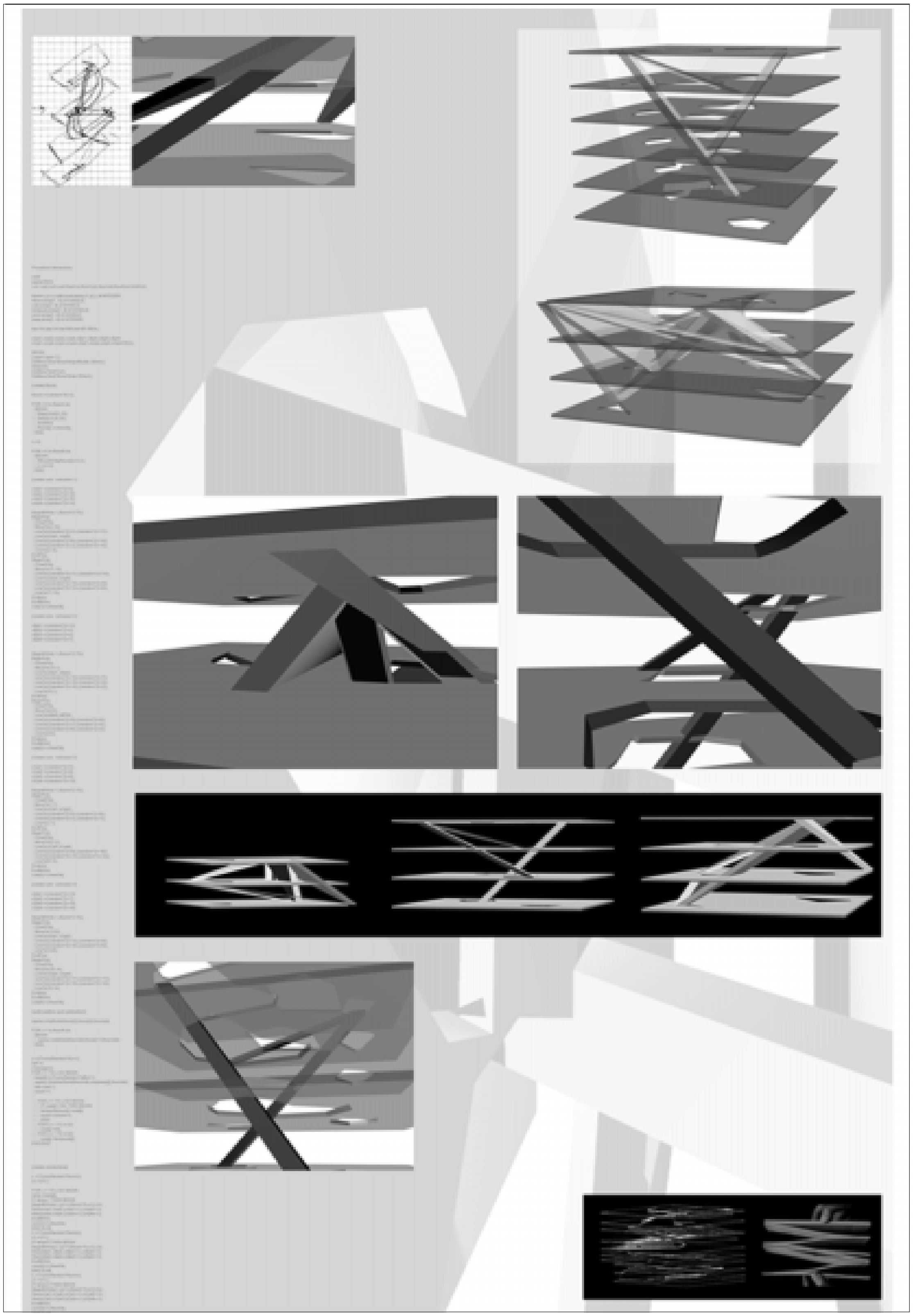


Figure 5. Anthony Guma's final poster, where he has taken the concept of intersection and applied it in three dimensions through planes that are beginning to represent floors

of architectural education institutions. The largely unmet demands of students, and the expectations of practice, are putting pressure on both ends of the architectural academic system. Architectural education must change, but it must do so with care. There needs to be better understanding of the differences between what architecture students currently learn, and how and why that learning will change as a result of the changes in the system so necessary to meet the demands of practice.

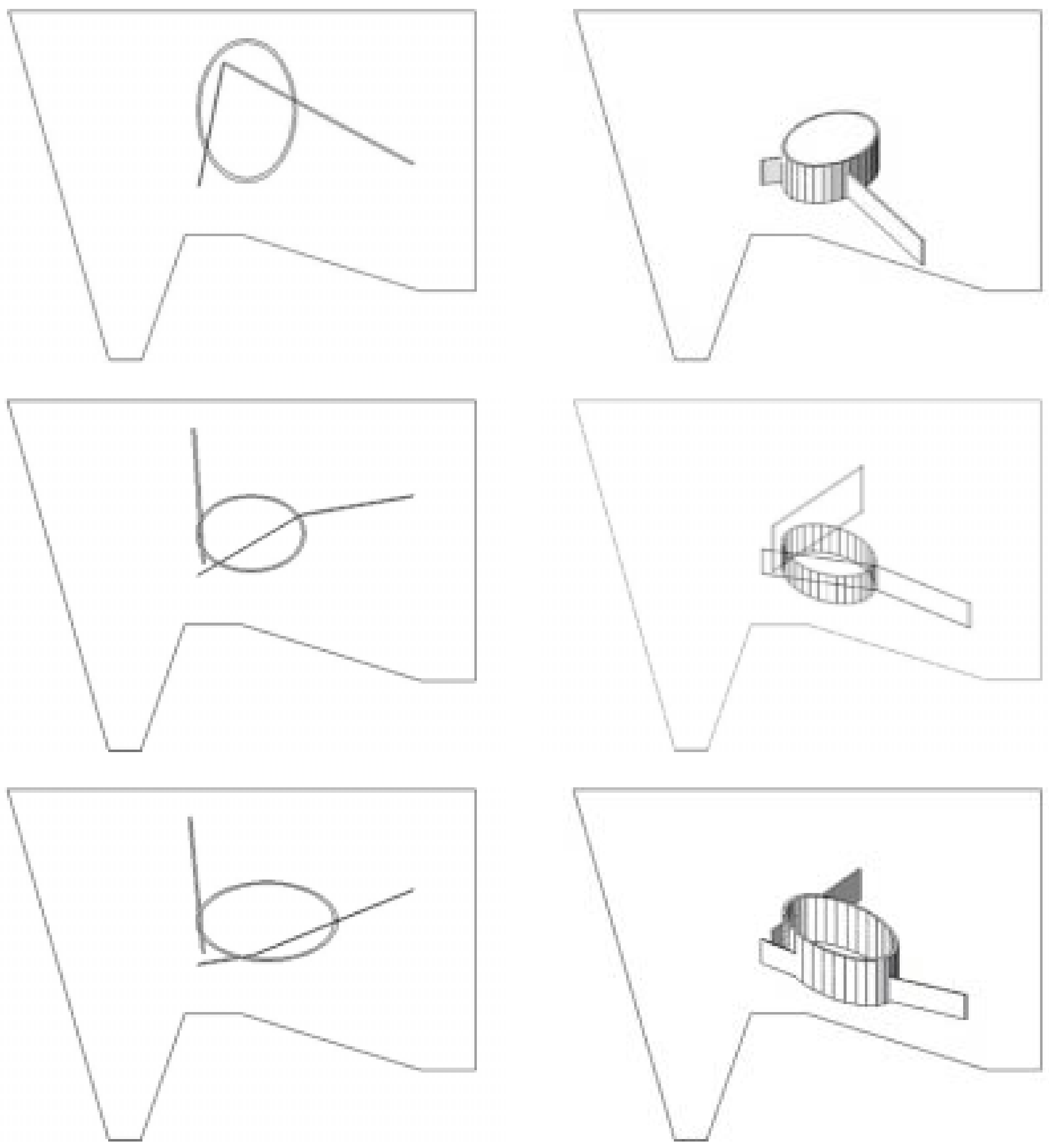


Figure 6. Anthony Guma was one of two students in the study to continue with the Fall class. Here he has written code to explore a particular form for an early, one week project looking at memorial representation

Challenging Design Education in Schools of Architecture

The teaching of design in schools of architecture has its history in an apprenticeship model. Students are grouped together to form "Units" or "Studios" under the direction of one more studio professors, echoing the atelier structure of earlier times. These studios become progressively more complex in their requirements as the student becomes more experienced (Schön, 1984a, Schön, 1985). Studios usually last either one or two semesters. Design projects are devised and supervised by the studio professor who conducts weekly tutorials or "desk crits", group discussions and activities. Student development in the studio is monitored through work presented in tutorials and at the more formal "crits" (also known as "pin ups" or "juries") that punctuate the semester's progression. A typical studio will have two or three major crits – one at the end of the semester, and one or more "mid-terms". In a crit the student presents her work to date through drawings, models, computer images and animations, or whatever medium she has been working with. The jury offer criticism, advice and suggestion of direction or work to be completed.

The model of studio teaching at MIT is for the student to be presented with a design brief at the start for which they must produce a final design by the end of the semester. Content and the level of detail of the brief vary from studio to studio, but most tell the student where the site is, and what the ultimate function of the building should be. The different studios a student must attend through her education process are taught by separate professors, who might individually teach related studios from one year to the next, but who do not often directly link with other professors to form a coherent learning environment for the student's degree. In this environment, the role of the brief and constraints of the individual projects plays an important part in the development of that semester's design.

The process described above is recognisable by anyone who is or has been involved in architectural design education. It is a model of learning that has a long history and is assumed to be appropriate and adequate. The means of assessment of the efficacy of such a system of learning is based on the quality of the representations created – the products of the system. Studies of the learning involved in design, of what the student is learning beyond the skills needed in the creation of these products, are rare. Assessment of the development of a successful design process, ways in which the cognitive development of the student as an independent and innovative architect might be assessed remain elusive. Whilst the ability to produce images and representations is an important aspect of architecture, the lack of criteria through which it is possible to assess quality means that it is difficult to gauge

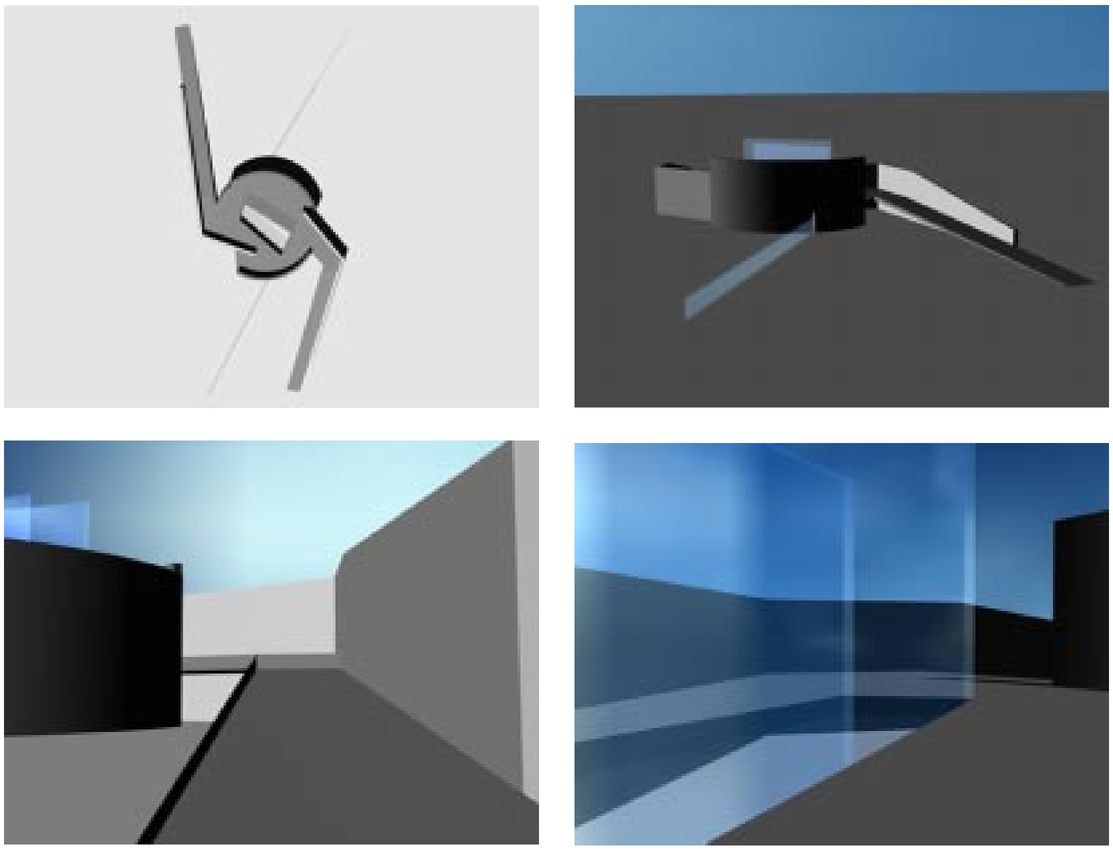


Figure 7. Anthony Guma’s proposal for the second, week long sketch project for the Fall semester.

whether this system of education is the best way of teaching design. However, its endurance, and the absence of any alternative ensures it is seen as the most appropriate solution.

This thesis is based on the premise that new possibilities of teaching the design process, and assessing the efficacy of such interventions, are now available. Both aspects, the development of alternative methods of teaching, and the assessment of these methods, have their roots not in the world of design, but in the field of educational theory and methodology. The Constructionist Seymour Papert suggested in his book, Mindstorms, that the best way to teach mathematics was perhaps not by teaching mathematics but through other means, something Solomon termed the “Papert Principle” (Solomon, 1986):

> “Two fundamental ideas run through this book. The first is that it is possible to design computers so that learning to communicate with them can be a natural process, more like learning French by living in France than like trying to learn it through the unnatural process of American Foreign-language instruction in classrooms. Second, learning to communicate with a computer may change the way other learning takes place. The computer can be a mathematics-speaking and an alphabetic-speaking entity. We are learning how to make computers with which children love to communicate. When this communication occurs, children learn mathematics as a living language. Moreover, mathematical communication and alphabetic communication are thereby both transformed from the alien and therefore difficult things they are for most children into natural and therefore easy ones. The idea of “talking mathematics” to a computer can be generalized to a view of learning mathematics in “Mathland”; that is to say, in a context which is to learning mathematics what living in France is to learning French.”
> (Papert, 1980 p.6)

Papert developed the computational environment LOGO to give children a framework within which they could create their own understanding of the abstract world of geometry and number. He stated that “children can learn to use computers in a masterful way, and that learning to use computers can change the way they learn everything else” (Papert, 1980 p.8). The course described in this thesis, developed and taught at MIT by the author, mimics this model through “talking design” to a computer, and is based on an adaptation of the same principle, namely that perhaps the best way to teach the design process is not through teaching design in studio. Advances in technology open up new possibilities for design pedagogy.

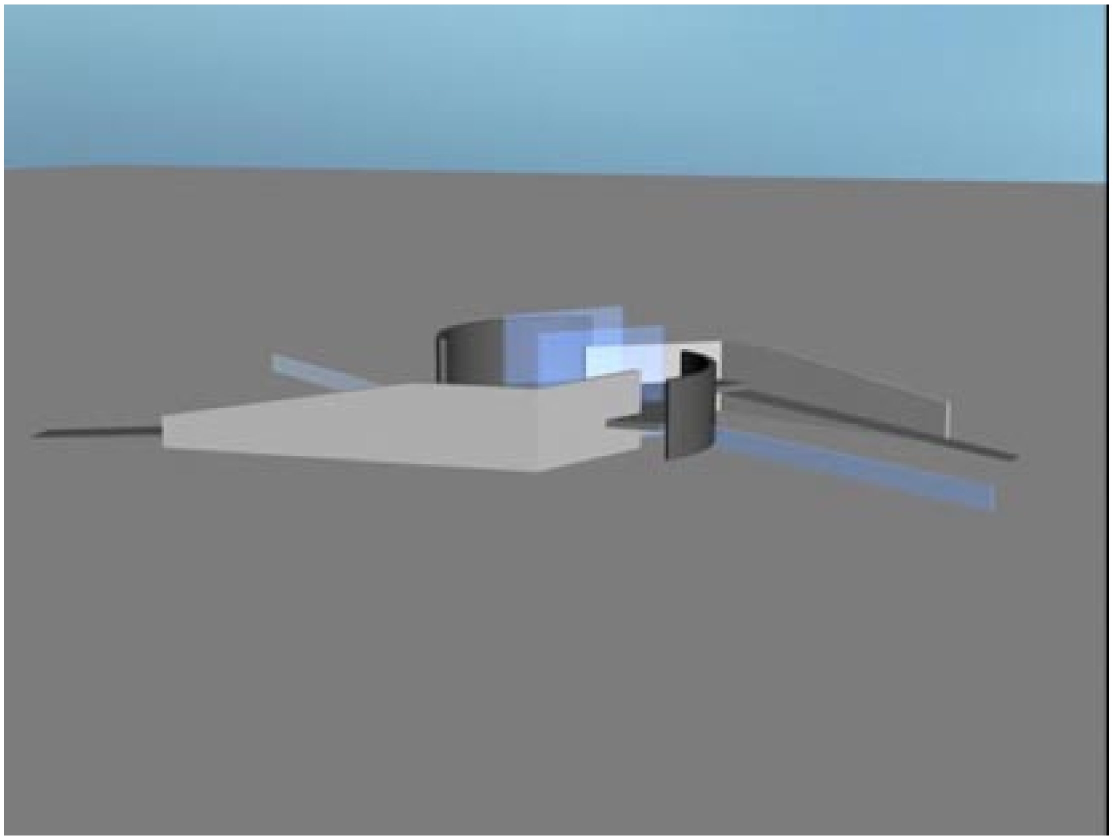


Figure 8. Anthony Guma’s proposal for the second, week long sketch project for the Fall semester.

Such new possibilities must be carefully examined to determine their effectiveness. Yet in architecture there are few studies of the education process, and consequently it is difficult to establish criteria by which an educational intervention might be assessed. Gero differentiates between product and process in architecture: "[there is an] approach [that] separates the identification of a design process from the evaluation of the outcomes of that process" (Gero & Purcell, 1993 p.253). This distinction is interesting, since architectural students are trained in a predominately Behaviourist environment that bases all judgments of their work on the evaluation of the outcomes – the products (Darke, 1979, Lloyd & Scott, 1995). The assessment of the successful development of their understanding of learning, and therefore designing is assumed to be reflected in the quality of these representations. Students are evaluated on the basis of the work they pin up in crits, and their accompanying verbal presentations. The risks involved in this highly public process are very great, since the potential for failure is also a potential for public humiliation. As with architecture as a profession, there is a concentration on product over process: architecture concerns itself with product, and most schools of architecture reflect this. But whilst in practice the product that is of interest is the final completed building, in schools, the product is the representation of a virtual building in the form of models, sketches and so on. When it is not possible to directly assess the success of the products in their built form, the jury members draw on their experience in translating and understanding representations to judge the success of the final design. They are, naturally, very influenced by the nature and quality of these representations. The best students are almost always those who have achieved the highest level of skills in creating such representations. Good performance in standardised assessment tests in primary and secondary Behaviourist education assumes an understanding of the material memorised. In a similar manner it is assumed that the creation of external representations in an architecture studio demonstrates an understanding of the process of design. It is not possible to assess, however, whether it demonstrates simply an understanding of the process of creating external representations (Oxman, 1999).

Better methods of assessment are required at two levels in architectural education. Firstly, a forum for the assessment of development of a personalised design process needs to be created, separated from assessment of the graphical skills necessary in the creation of the products of design. Secondly, better means by which the efficacy of interventions in the education process might be assessed need to be developed. The field of education has developed in recent years complex and sophisticated methods for assessment, particularly through qualitative methodology. There has been wider acceptance of the difficulties of developing adequate control groups to assess quantitative aspects of a process that necessarily involves so many uncontrollable factors

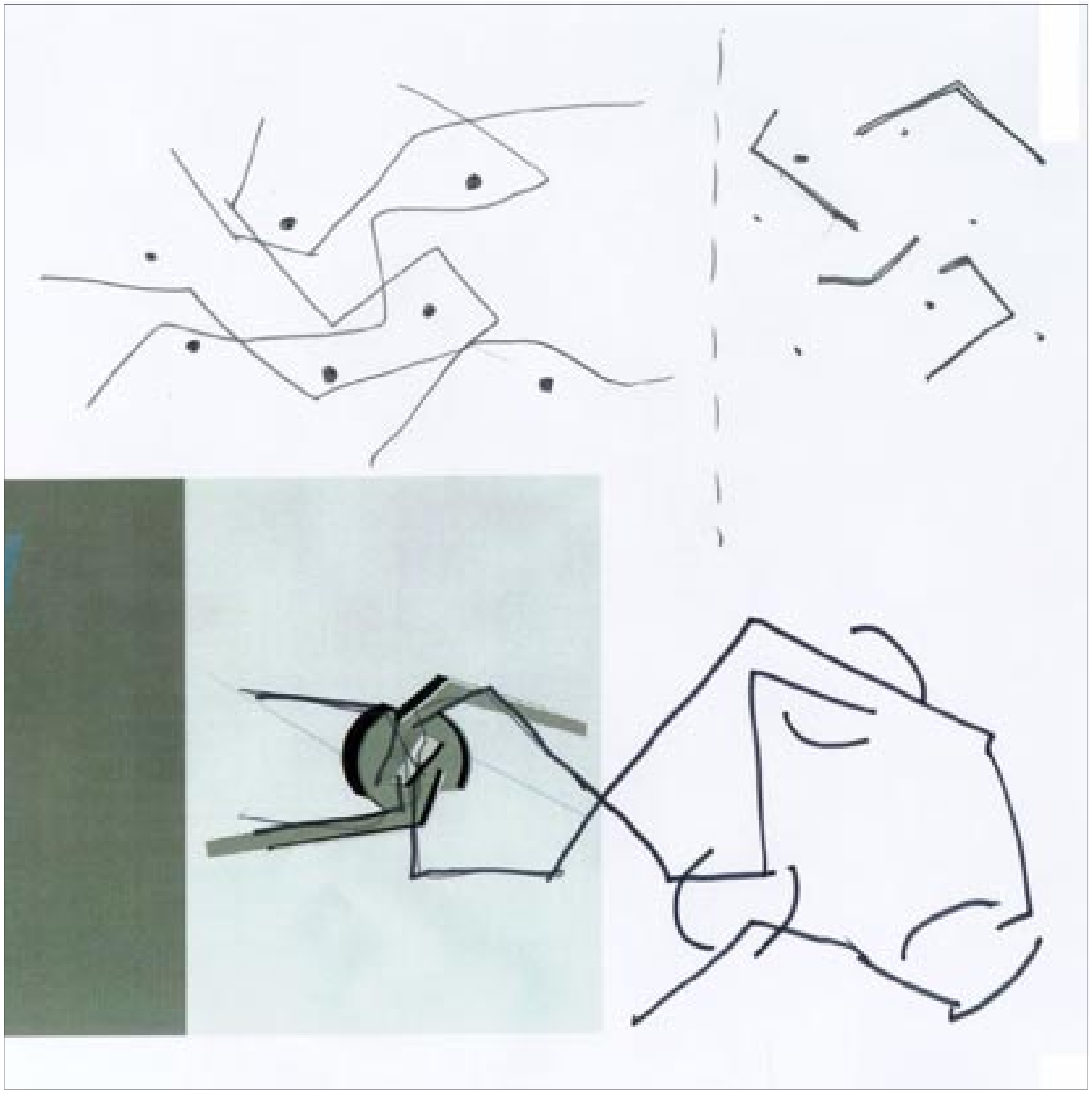


Figure 9. Moving from the early exercises to the main project to design a memorial for Martin Luther King, Anthony Guma builds on his earlier design, add sketches that represent aspects of the leader's life

in the education of individuals. Methods have been developed that concentrate on what individuals learn and details of *how and why* an intervention might succeed or fail, rather than more simply stating *whether* it has succeeded. In this thesis some of these theories are applied in the field of architectural education, with results that may influence the way in which architectural education is assessed in the future.

Aims and Objectives

There have been two parts to the work described in this thesis. First, the development of a significantly new course that challenges the role of computers in architectural design education, by demonstrating that computers, and more specifically computer programming techniques, can be used to develop the cognitive skills required for the design process. This course was called Digitally Mediated Design 1. Despite the legacy of Papert's proposals in mathematics education[1] the possibility that computer instruction might go beyond teaching skills of computer usage and instead use them as a vehicle to teach something not computationally based, has not previously been considered in the field of architecture. This is a revolutionary approach whose underlying Constructionist principle has implications beyond the confines of the course. The structure of the course is described here, and the illustrations used in the thesis are from its students.

Secondly, the thesis describes research conducted to assess the impact of the course on the educational experience of students who joined in the Spring of 1999. This research has looked towards the field of education for appropriate methods. The application of methodology from education research, rather than from more traditional methodology used in studies of architects and design process, is a novel but appropriate approach to the complex area of architectural design education assessment, particularly in the light of developments in qualitative educational research techniques. There were six students in this cohort, all of whom had chosen to join the course and all of whom agreed to be a part of the research study. The data for the research were gathered primarily through in-depth interviews that were recorded at regular intervals during the semester, with one final interview six months later. Secondary data sources included the formal course presentations and the students' work. All of the transcriptions of the interviews and presentations were analysed using the Constant Comparative Method. The primary limitations to the study were the small number of students, the impossibility of creating an identical control group, and the same person – the author – being instructor, course developer, and researcher simultaneously. The methodology used, the results of the study, and the conclusions drawn from the data gathered are all described in this thesis.

It is important at this stage to point out that the research looked at the impact of the class as a whole on the architectural educational experience of the students. It did not study the specific impact of digital media as a tool on the design processes of these students. All tools in design have a limited world within which their users must operate, and through which their designs are necessarily influenced (see Appendix H). All too often, particularly with novice designers, this influence is implicit, and limitations of the design world of the tool are incorporated into new designs without question.

This study did not examine the influence of the tool itself on the design processes of the students. However, as the results show, the impact of greater verbalisation and with it, greater understanding of the design process, resulted in students who were aware of both the possibilities and limitations of the tools used. This increased awareness is both a result of, and further support for, designers who are better able to take control of all the tools they use in design.

The implications of this thesis are therefore twofold. The first is the proposal that computers offer opportunities for new pedagogical techniques in many more areas of architectural education than are currently being explored. The second is that assessment of the efficacy of these interventions can and should draw on developments in educational research.

This thesis is based on the following: that computer programming used in a Constructionist pedagogical framework offers the opportunity to develop a better understanding of a personal design process; that this development is enhanced by the separation of process from product; and whilst the student is developing a better understanding of her own design process she is simultaneously developing her perception of the role of digital media in design.

Background Theory

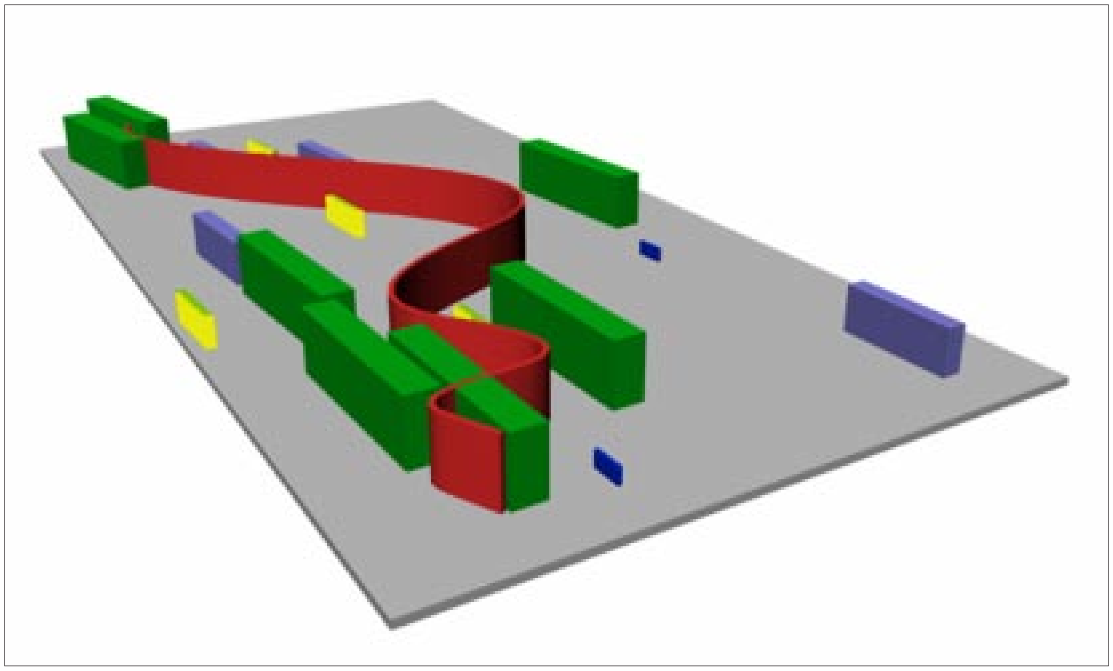


Figure 10. A three dimensional diagram of the four different types of events Anthony Guma identified surrounding Martin Luther King's life and work. These elements became objects in a museum style memorial

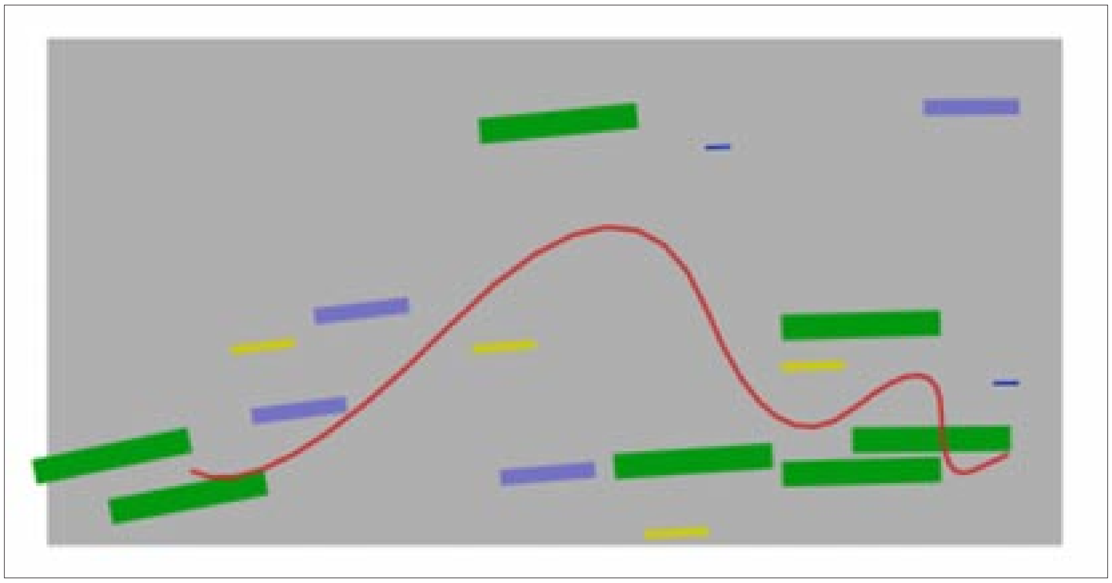


Figure 11. Plan of the four different types of events Anthony Guma identified surrounding Martin Luther King's life and work. These elements became objects in a museum style memorial

What is design?

To describe a manner in which computers can be used to teach the design process, it is first necessary to describe what is meant by design process and which aspects in particular are to be aided by the technology. This is not without considerable difficulty, however. The term “design” has a wide range of meanings as both a noun and a verb. Attempts to define it in either grammatical sense have proved difficult and concise definitions remain illusive. Many studies of design have been conducted. Many people have in turn offered their definition of design. “Design is the human power of imagining something that did not exist before” wrote Harvey in 1950 (Harvey, 1950 p.6). Rowe sees design as inquiry. “Design appears to be a fundamental means of inquiry ... [it is] a practical form of inquiry” (Rowe, 1987 p.1). Gero is more specific, stating that the activity of designing involves “the recognition of a need or opportunity, the specification of what needs to be produced, and the manifestations of that in some form, usually physical” (Gero & Purcell, 1993 p.253). Other definitions have offered similarly varying viewpoints, depending upon both the perspective of the author and the prevalent model or theory of the time (Mitchell, 1994).

It is not the intention of this thesis to add another definition of “design”, but it is important to define which specific cognitive skills of design as an activity are to be developed through learning the techniques described here. There have been many studies of design as an activity, looking at what designers do and attempting to understand how they do it (See for example Agabani, 1980, Akin, 1978, Branki, 1993, Cutkosky & Tenenbaum, 1990, Darke, 1979, Davies, 1987, Do *et al.*, 1999, Eckersley, 1988, Lawson, 1980, Roy, 1993, Tang & Leifer, 1991). Such studies continue to occupy researchers today (McFadzean, 2000).

Early attempts to study the architectural design process took their inspiration from similar studies in management science and cognitive psychology (Eastman, 1969). These studies were primarily concerned with describing the process in its most general form, and identifying the mechanisms and processes responsible for its development. They took as their model a human problem solving framework developed by Newell and Simon (Newell & Simon, 1972). No indication was given in the studies as to their underlying purpose, beyond describing or formalising an existing and therefore limiting process with no room for the inquisitional nature of design.

The primary conclusions drawn from this early work were that the design process has characteristics that are shared by other information processing activities and that as a result certain behaviours of designers can be described using models from these areas. In retrospect it is easy to see the limitations of this view. There is overlap in design with many different

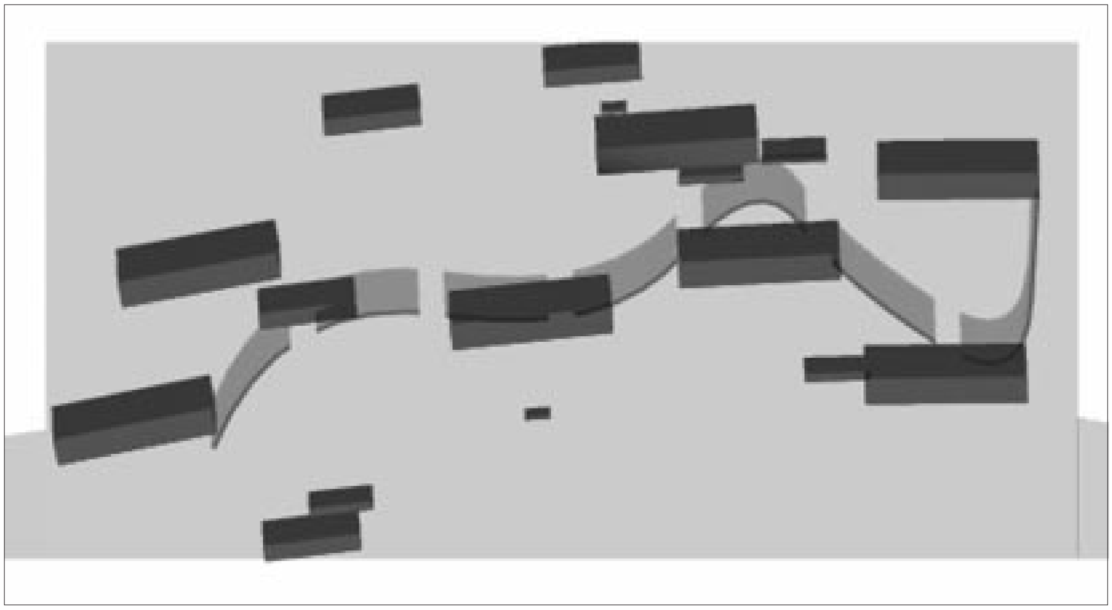


Figure 12. Anthony Guma wrote code to produce many variations of the placement of the elements in his open-air museum memorial. See figures 12 to 17

disciplines, but to describe design purely in terms of one of these is to place artificial limits on its possibilities. Perhaps the most significant conclusion of these studies is that the design process cannot be succinctly described: there are at least as many different styles of decision making as there are designers in the studies.

Studies of the design process continue, although often the purpose of such studies is less easily understood or documented. It appears from the research that those responsible are concerned with description and definition rather than practical application of their findings. Some researchers in architecture and artificial intelligence, and architecture and computer science, have used some of this information in the development of software tools for aiding or automating all or part of that design process (For example, Alexander *et al.*, 1966, Galle & Kovacs, 1992a, Mitchell *et al.*, 1992, Negroponte & Groisser, 1970, Whitehead & Eldars, 1964). Yet even today, studies of this kind tend to stand as models isolated from any practical use. This lack of purpose manifests itself in a continuing emphasis on describing methodologies rather than intentions and practical aims (Akin, 1993, Eckersley, 1988, Pohliman, 1982) and on the classification of previous studies of design through different frameworks (for example, Akin, 1993, Gero & Purcell, 1993, Lloyd & Scott, 1995, Mitchell, 1994, Sun, 1993).

However, through these studies common elements to the design process have emerged and a useful picture has been created of some of the cognitive functions and processes the designer calls upon, acts within and uses. Individuality comes in how and when each designer combines these elements. Design problems are widely recognised to be “wicked” or ill defined (Newell, 1970) requiring a particular methodological approach. Schön describes of the process of solving these design problems as experimental, where the action that is undertaken is both the hypothesis to be tested and the proposed solution (Beheshti, 1993, Gero *et al.*, 1991, Portillo & Dohr, 1994, Schön, 1985) although he does not elaborate on “experimental process” or what it might involve for different individuals. To work in this manner, the designer must continuously analyse and modify both her current solution and her understanding of the problem at every stage. Schön describes the cognitive techniques used as ‘reflection-in-action’ (Schön, 1983, Schön, 1984a), involving the non-linear exploration of conceptual design ideas through their representation in graphical form. Fish described these two different languages as conceptual or descriptive, and graphical or depictive (Fish, 1999). This cyclical process has been likened to a conversation that the designer has “with the materials of the situation” – a description not confined to the design process (Keller, 1983, Schön, 1987, Schön, 1992). This process of communication between the designer’s internal conceptual ideas,

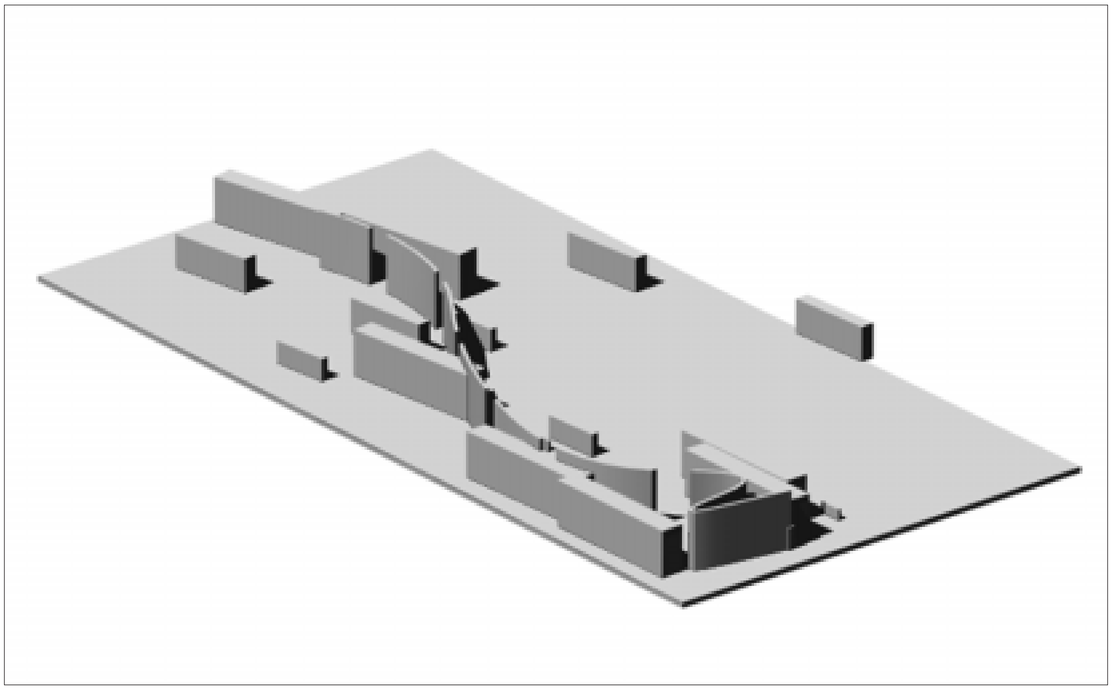


Figure 13. Anthony Guma wrote code to produce many variations of the placement of the elements in his open-air museum memorial. See figures 12 to 17

and the external representations of those ideas, requires familiarity with the different means of representation involved – verbal and graphical, and their associated architectural implications (Fish, 1999, Heylighen *et al.*, 1999, Ulusoy, 1999). The architect, when she is designing, is using the systems of notation involved with each of these means of representation as ‘objects to think with’ (Papert, 1980, Yakeley, 2000) – the ‘materials of the situation’ – representations of the design that at each stage are used as the basis of the next design thought to be explored (Fish, 1999). To do this effectively, she must interconnect the languages involved and move freely between their respective systems of notation. A good designer is able to express her ideas fluently in all these languages and maintain the many “parallel lines of thought” required to do so (Lawson, 1993). This process of simultaneous translation is a foundation of design thinking (Yakeley, 1999). It can be seen as a framework that may be applied to many designers with individual differences; a framework of ‘difference in similarity’ (Lloyd & Scott, 1995). The importance of this approach is the formal recognition, echoing that of earlier research, that the design process differs with each individual. Design involves the application of specific cognitive skills. Formulaic approaches have tended to concentrate on defining the method of application. What is required is a recognition that, whilst the application differs according to project and individual, the commonalities lie in the cognitive skills involved. With this interpretation it is possible to describe the individual differences that have been observed in more quantitative studies (Agabani, 1980, Eckersley, 1988).

The cognitive processes of design thinking involve a constant process of inquiry. It is not possible for the designer to hold all design knowledge prior to a design’s completion. Instead she must be willing to “step into” the problematic situation and “remain open to the situation’s back talk” (Bamberger, 1996 p.41). Yet this is not a process that is easily learned. In conversing with the materials of the situation, the designer becomes personally involved with the objects of her design. The student designer must learn to recognise her own creative output as the materials or objects with which to think. But to do this, she must face Schön’s Paradox, and begin to design in order to learn to design (Gero *et al.*, 1991, Schön, 1985). Often she is distracted by the syntactical difficulties of representation in graphical or verbal form. The difficulties of learning to translate easily between the two means of representation and their corresponding architectural meaning can be forgotten in a concentration on the creation of graphical products. In addition, the conversation of paper based designing described by Schön is more often a monologue, or at best a dialogue with something whose mode of expression requires considerable skills of interpretation. The designer must converse in the graphical language of the paper, and then infer the paper’s contribution to the discussion through looking at the sketches she has made in a new way (Berkeley, 1968). This dialogue is very hierarchical – the

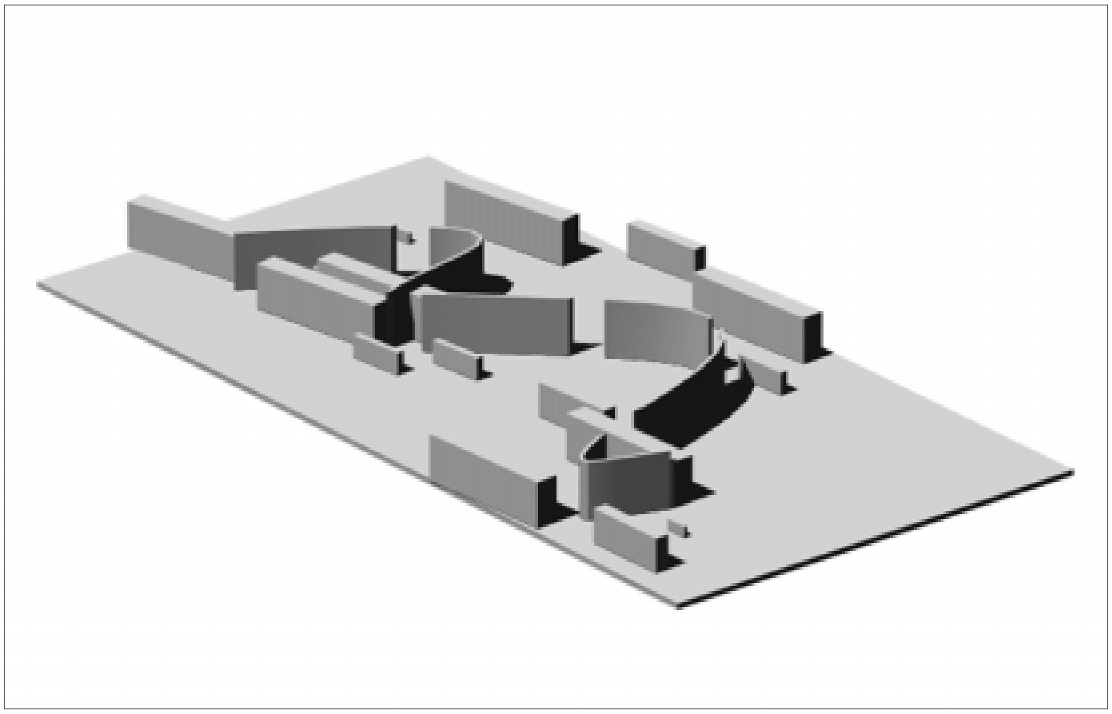


Figure 14. Anthony Guma wrote code to produce many variations of the placement of the elements in his open-air museum memorial. See figures 12 to 17

designer is in control and the sketchpad plays an entirely passive role – more so than in a teacher-pupil relationship. Scardamalia and Bereiter argue that hierarchical, Socratic dialogue, where the teacher asks questions designed to elicit particular responses and therefore understandings in the student, is unproductive. They see dialogue between individuals on a more equal basis as more useful in education. In the context of the design conversation, where the hierarchy is more extreme, there is potential for a more equal dialogue if the pencil sketches were replaced with a medium that might offer suggestions to the designer's ideas.

In this thesis, I am describing the design process as one involving similar cognitive skills to constant learning and inquiry (Gero & Purcell, 1993, Guarra, 1974). The designer must learn about each new design situation in order to develop a successful design solution, but it is only through actively designing that she is able to learn (Schön, 1985, Schön, 1988). It is this aspect of active exploration through representation that makes the design process a constant, Constructionist learning process. This link with the process of learning means that educational research into how we learn might have a greater significance into the way in which architectural students learn than might initially be expected. This active process was characterised succinctly by Schön's description of 'reflection-in-action', involving a 'conversation with the materials of the situation'. This description forms the basis of, and is supported by, the techniques taught in the course at MIT described in this thesis. The 'materials of the situation' are the means of communication with the computer, first conceptual, then computer programming language, and its resultant graphical output. The 'situation' remains an architectural problem, not a computer programming one. The paradigm for the role of the computer moves away from an electronic pencil, or a repository of data and information. The computer becomes an electronic partner able to contribute to the conversation of design, and to become an active voice of 'back talk'. The definition of the computer's role in the partnership is not predetermined, and is open to change (Chandrasekaran, 1989). In addition, the process of learning such techniques teaches the student, through explicit demonstration, the importance of listening to the back talk of design.

The view of designing that I am promoting in this course has been studied and proposed by others. It involves the development of an initial idea (Darke, 1979, Lawson, 1993) that is explored through a process that resembles a conversation between the designer's internal and external representations of that idea (Schön, 1983). Internal representations are often in the form of verbal thoughts and concepts. Each iteration of the process is expressed in external graphical form, and each of these representations is assessed in a manner that permits new interpretations to enter into both the proposed solution that has been externalised, and the

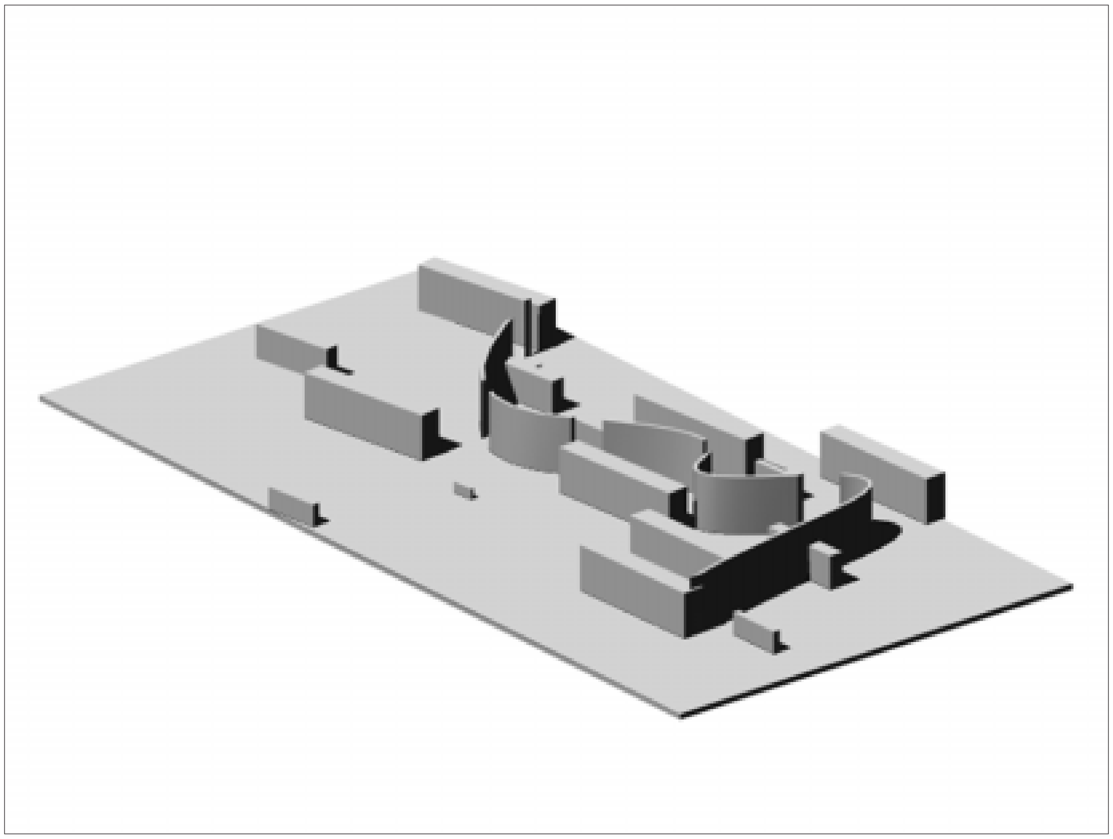


Figure 15. Anthony Guma wrote code to produce many variations of the placement of the elements in his open-air museum memorial. See figures 12 to 17

internal assumptions of what the original design problem involves (Akin, 1984, Eckersley, 1988, Lawson, 1980, Lloyd & Scott, 1995, Schön, 1988). The verbal and graphical notations together form the basis of the architectural design process. The mature designer understands the link between internal and external systems of representation, and is able to converse fluently in both. The process of design development involves the designer having a strong sense of direction dictated by the current project, and taking individual steps by sketching her ideas in two or three dimensions. These sketches become the 'objects to think with' of the process and form the basis for the next design decision. These decisions, or rules, have emerged as a part of the process. Such ""emergent rules"" are a direct result of the design process whatever medium is used.

Developing the cognitive skills involved in this process – constant inquiry, using external representations as cognitive props to develop the next design decision, understanding the significance of both verbal and graphical languages in representing architecture - is considered to be a by-product of the studio system of education. In the studio, the focus is more often on product rather than process. There is a need to redress the balance, and to provide a framework for focus on, and development of, design process. A demonstration of the potential for enhanced understanding in the design studio is also necessary.

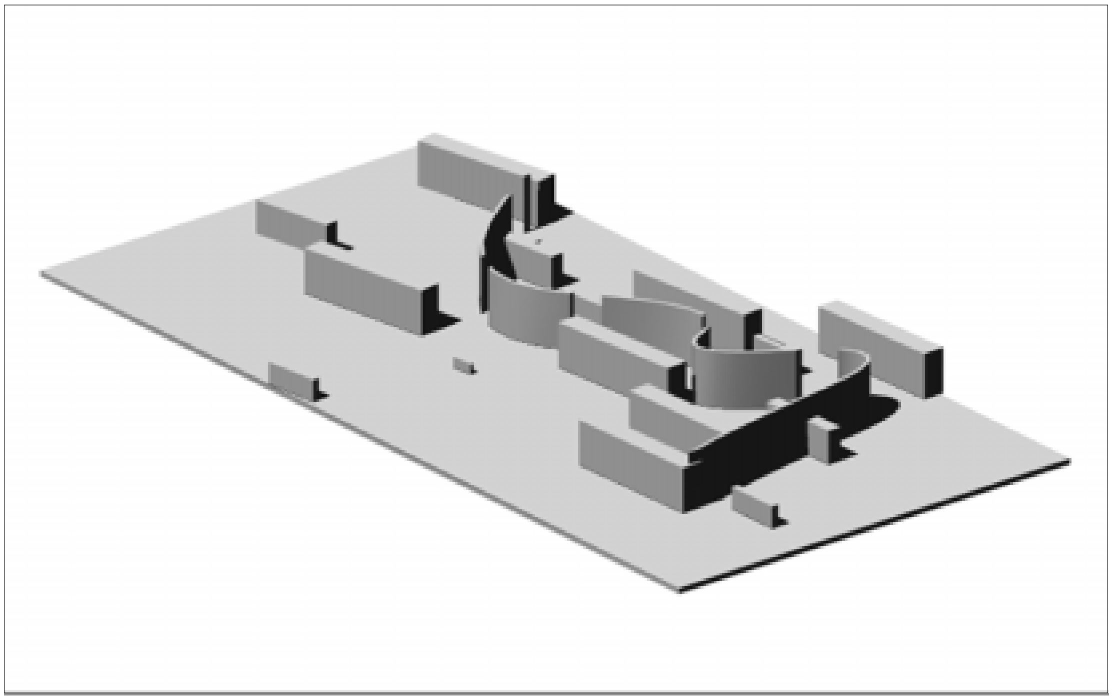


Figure 16. Anthony Guma wrote code to produce many variations of the placement of the elements in his open-air museum memorial. See figures 12 to 17

Process versus product

Although much architectural research has examined the process of designing, the practice of architecture is naturally predominantly concerned with the creation of products – the completion of physical buildings. In a similar tradition, schools of architecture tend also to focus on product, but in the academic context the products are representations rather than built form (Oxman, 1999). When there is a focus on process in a design studio it is often manifested in standardised techniques for designing. Courses teaching these techniques often result in products with minimal variation between students. It can be difficult for these processes to be transferred by the students into other courses and situations. Although valuable, it is not clear whether such pedagogical approaches enable the student to develop her own individual process of design thinking. Such an individual ability is generally assumed to be learned indirectly as a by-product of the studio system. Yet in the studio the student is often confused between design production and cognitive skills needed for the process of designing.

The 'products' used by architects in their design process, and the 'materials of the situation' described by Schön, are the external representations of the designers' internal ideas (Galle & Kovacs, 1992a). They act as "representation support structures" (Fish, 1999 p.II.20). It is generally assumed that cognitive skill of using external representations in the design process, in the form of rough sketches or models to be manipulated, internalised, and re-expressed in different forms (Rieber, 1995) will be learned as a side effect of the student's involvement in a design studio setting. It is a skill the architect must learn whilst studying all the other aspects that make designing a highly complex activity. But representations come in many forms and serve many functions. Those created by the designer during the design process are often intended for private use and not for public display. Representations for the latter have different characteristics of detail and are often intended to seduce the audience. They are usually created after the design has been completed.

In this thesis I am making a distinction between two functions for representation in design process – those for developing the design, for the designer's self-communication (Schön, 1992), and those for presentation and communication to others. The former are often working sketches, in a variety of media, and form part of the personal design conversation of the creative process (Schön, 1983, Schön, 1984a). The designer uses these external representations of her ideas to develop the next emergent design decision. Often very crude in appearance, they have no intrinsic value to the designer beyond the development of the next emergent rule. They are examples of Lévi-Strauss's

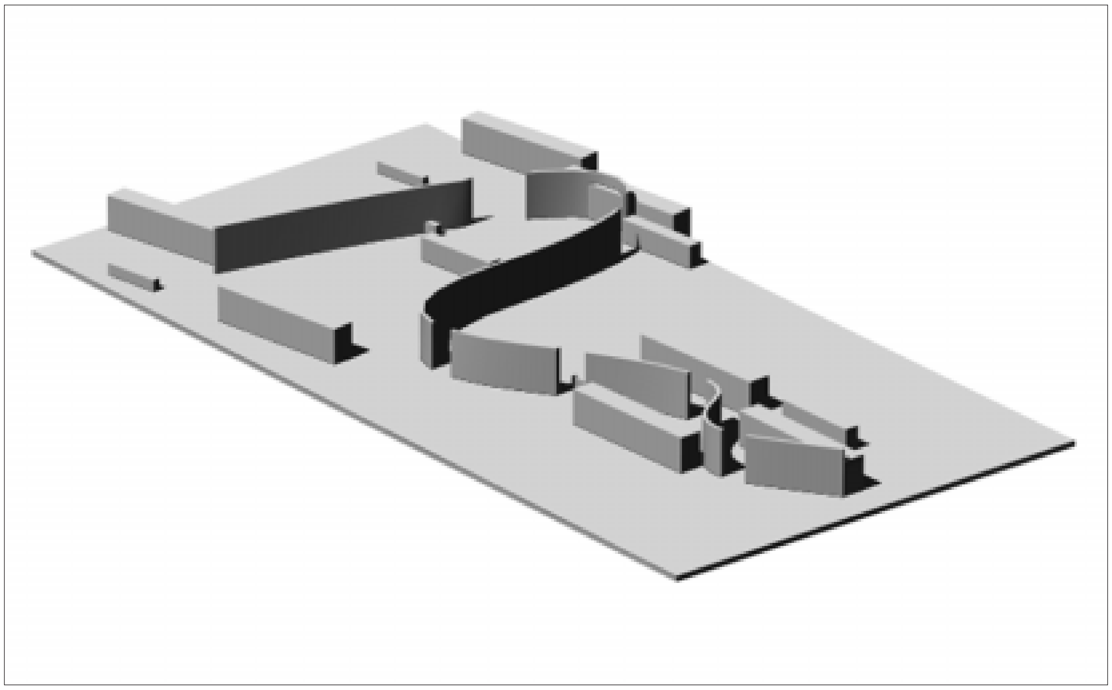


Figure 17. Anthony Guma wrote code to produce many variations of the placement of the elements in his open-air museum memorial. See figures 12 to 17

Bons-a-Penser (Lévi-Strauss, 1968, Turkle & Papert, 1990) or "objects to think with"[2]. This is in contrast to the images produced for presentation to others. Such iconographic representations can inhibit rather than promote the creative process: the designer becomes attached to the representation and is unable to move the design beyond the state captured in the image. They become *Bons-a-regarder*, or "objects to look at" (Yakeley, 2000).

In designing, sketches are used as a visual, external representation of internal idea. The transformation of these external representations supports the internal mental processes (Norman, 1993). The mode of representation is not as critical as its function in assisting in the cognitive aspects of the design process. In the same way, developing an idea in written form for a paper or journal article helps coalesce the idea and focus the writer's mind to examine the thoughts she has had in more detail. In design, the traditional media for representation of sketching is a soft, blunt pencil. Architects are trained to use it quickly and efficiently, to represent ideas in a flexible and ambiguous manner permitting multiple interpretations (Dorsey & McMillan, 1998, Fish, 1999). Yet, often in architectural education the emphasis rests on the representational object and the skills required to produce it, rather than the thought processes and design development resulting from their production (Oxman, 1999). Architecture students become adept in the creation of *Bons-a-regarder*.

Architecture is judged in the first instance solely through these representations – they are used by students to pass studio juries, and by architects to convince clients and developers to invest in building the proposed projects. Design work is thus judged through the quality of the imagery – the products. It is assumed that beautifully crafted drawings and models reflect, for an experienced architect, a comprehensive understanding of the design problem. For the student, it is additionally assumed they reflect a similarly highly developed understanding of the design process. As such they play a highly significant role in the profession (Oxman, 1999). The assumption that there is an understanding both of representation creation and the process of design is often true of the most able students. However, these students represent a relatively small percentage of high achievers who do very well in schools of architecture. For the majority of students the confusion remains, and the focus on design representation in the design studio does little to assist them in understanding the difference between product and process.

The confusion is in part a result of such a focus on design products inhibiting the use of these representations as objects to think with. In-depth exploration in any field depends on the ability to change, modify, and discard different ideas according to their significance in the solution of the current problem. An iconographic representation has a status that prohibits its rejection

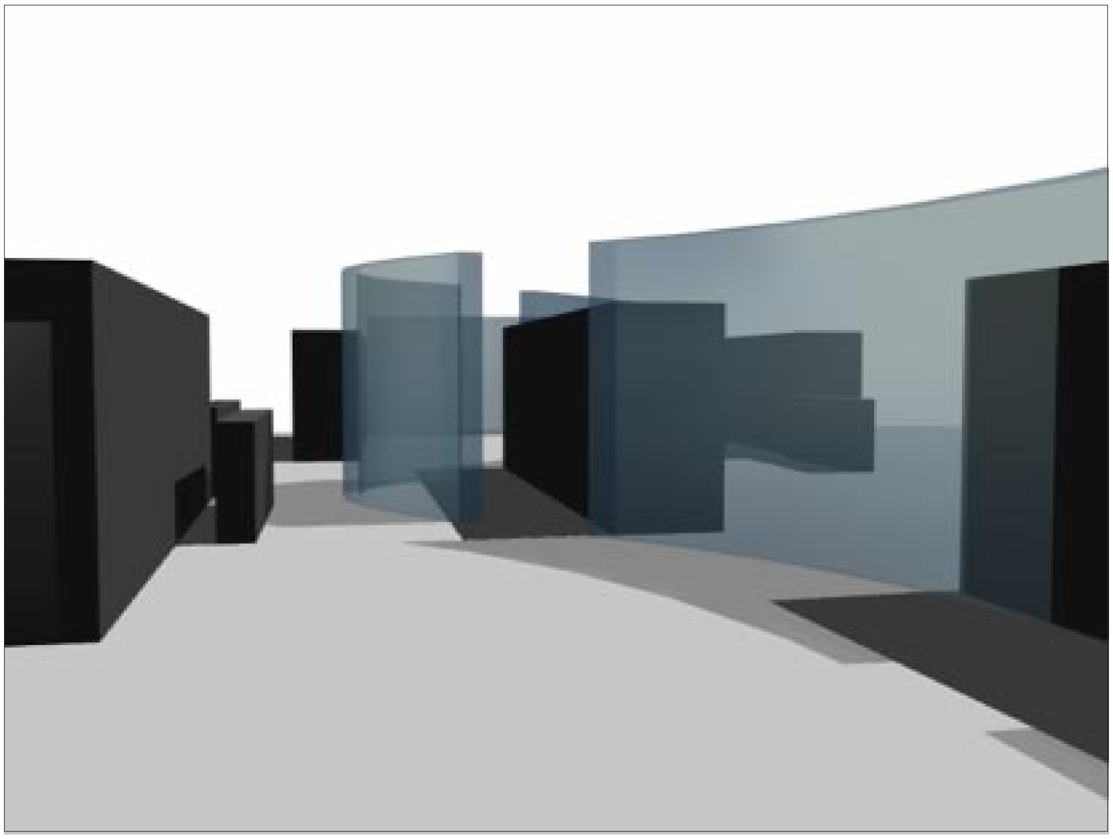


Figure 18. Interior view of Anthony Guma's final design for an open-air museum for Martin Luther King. Note that he has move directly from the diagrammatic output of the code to presenting this as real architecture

regardless of its relevance or suitability to the project. This problem can be seen in students who fixate on the same idea despite encouragement for broader exploration (Sachs, 1999). Where a representation has a value that is too high, it becomes an "object to look at" rather than an "object to think with" (Yakeley, 2000).

What is required is a learning environment which supports the development of a personal design process. However, teaching any subject necessarily involves the precise compilation and construction of the material to be taught. A proposal to teach the design process separate from the design studio must involve describing what the students will learn. Yet the role of the design studio as the most effective means of teaching design recognises the conclusions that can be drawn from design process studies: design as an activity is different for each individual. It is therefore difficult, if not impossible, to use descriptions of other people's design activities to teach students about design. Describing information and processes the students will learn and repeat in a Behaviourist manner is inappropriate. What is required is a description of a Constructionist framework of cognitive activities within which the student can construct her own understanding of what the design process means for her. This framework includes a cyclical, four stage view of the cognitive activities involved.

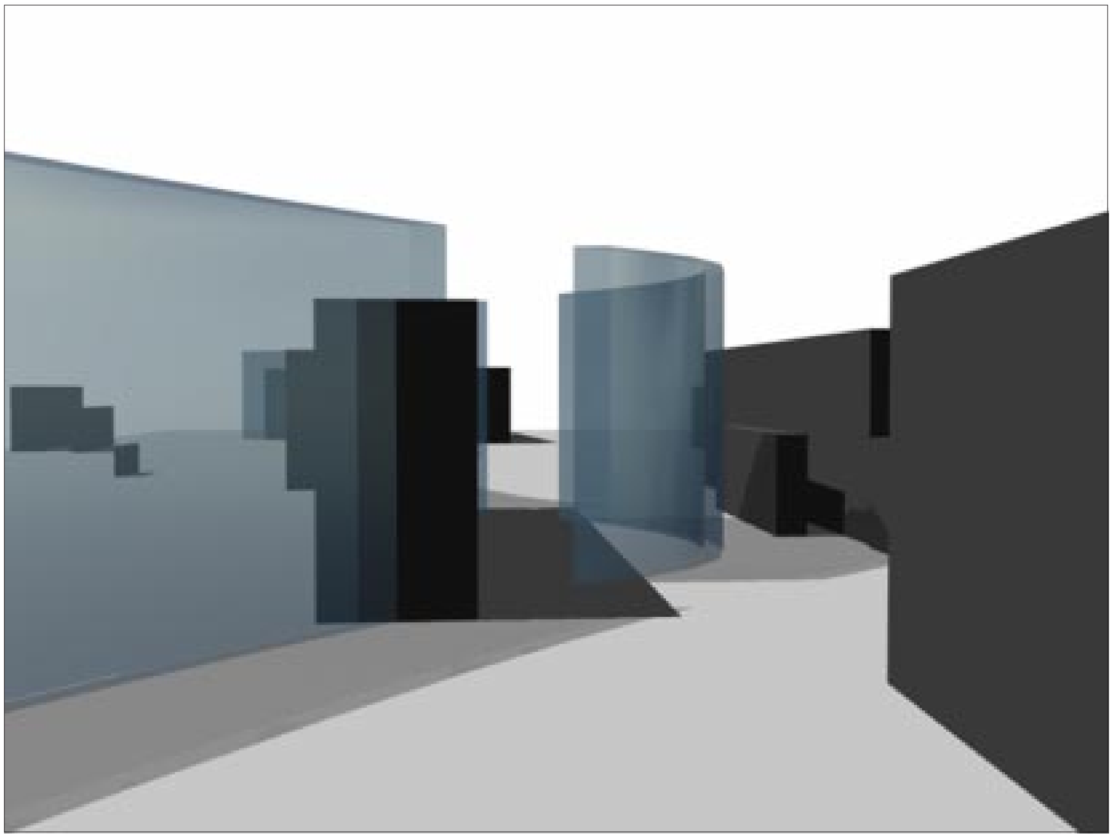


Figure 19. Interior view of Anthony Guma’s final design for an open-air museum for Martin Luther King. Note that he has move directly from the diagrammatic output of the code to presenting this as real architecture

Stages in the Cyclical Process of Design

The basis for the course developed for this research is the view that the process of designing has a cyclical nature and involves four stages: the statement of an initial concept in verbal form, the production of a graphical expression of the concept, the critical analysis of the graphical output, and the subsequent modification of the initial concept. These four steps represent one iteration of the process; the modified version of the initial concept must then be expressed verbally, and so the cycle repeats and the process continues. What each stage involves differs from individual to individual, and from one design project to the next. Nevertheless, each stage is a recognisable cognitive activity. It is proposed here that developing an understanding of each of the stages forms the framework of 'difference in similarity' described by Lloyd and Scott (Lloyd & Scott, 1995).

Step 1: Verbal expression of initial concept

Designing is a complex and multidimensional process in which it is easy to become distracted by the many restraints of creating a functional building. Consequently, it is easy to lose sight of the initial idea that is required to provide coherence to the building and the programmatic constraints. If the student is ambiguous about her original idea, she will quickly focus upon other more tangible attributes of the building, and the design will lose direction. Encouraging a student to clarify her initial concept is a difficult process; it relies heavily on the labour intensive teaching process of individual tutorials and "desk crits", and formal presentations to jury panels. But these focus predominantly on the process of communication of ideas to an external audience. Designing involves communicating ideas to oneself, and the process of learning to design must involve the development of an ability to clarify ideas without the need for an external critic.

Step 2: Graphical expression

Architectural students need to learn skills of representation in a variety of media, particularly the ability to express ideas quickly and easily in two dimensional sketches. The student must learn the difference between developing an idea through representation – using 'objects to think with' – and representing an idea for presentation – creating 'objects to look at' (Yakeley, 2000). Products of the former should be theoretically if not actually disposable as the architect moves from one idea to another. All too often, however, the student invests as much effort in the development of sketches as she does in the presentational ones, and the ideas embodied in them become correspondingly fixed and difficult to change.

Step 3: Assessment of output

Many novice students have a Behaviourist 'all or nothing' attitude to design – they assume that designing a building occurs in a single, huge leap of the imagination where the architect moves from not knowing, to knowing exactly the final building design in a single design thought. Unable to reproduce this perceived process, they tend to discard each attempt and start over. What is important to learn is the cognitive ability of gradually developing an idea through small steps, maintaining positive attributes whilst changing the more negative ones. Each step involves a careful examination of the previous step's output. This assessment must involve both the student's graphical and her conceptual output.

Step 4: Modification of idea

An important step beyond the assessment of output is the use of the knowledge gained to modify the original idea. Is the concept of the design still valid, or does the currently represented version modify the concept in some way, and if so, how? The process of architectural design is one in which the definition of the design problem progresses together with the proposed solution; only when the final design is complete is it possible to precisely define the original design problem.

This four step view of design describes a cyclical framework the student uses to develop her design. An experienced designer compresses the steps involved into more fluid process. But for the novice designer each step must be painfully learned. Describing this framework gives the student scaffolding to support their learning without prescribing any one specific methodology. Since design is a constant process of learning, ultimately what she is developing as she learns to design is a new process of learning in a Constructionist framework. Understanding both the student's original, Behaviourist understanding of learning, and the Constructionist model she must develop, is critical in understanding the process of architectural design education.

Educational theory

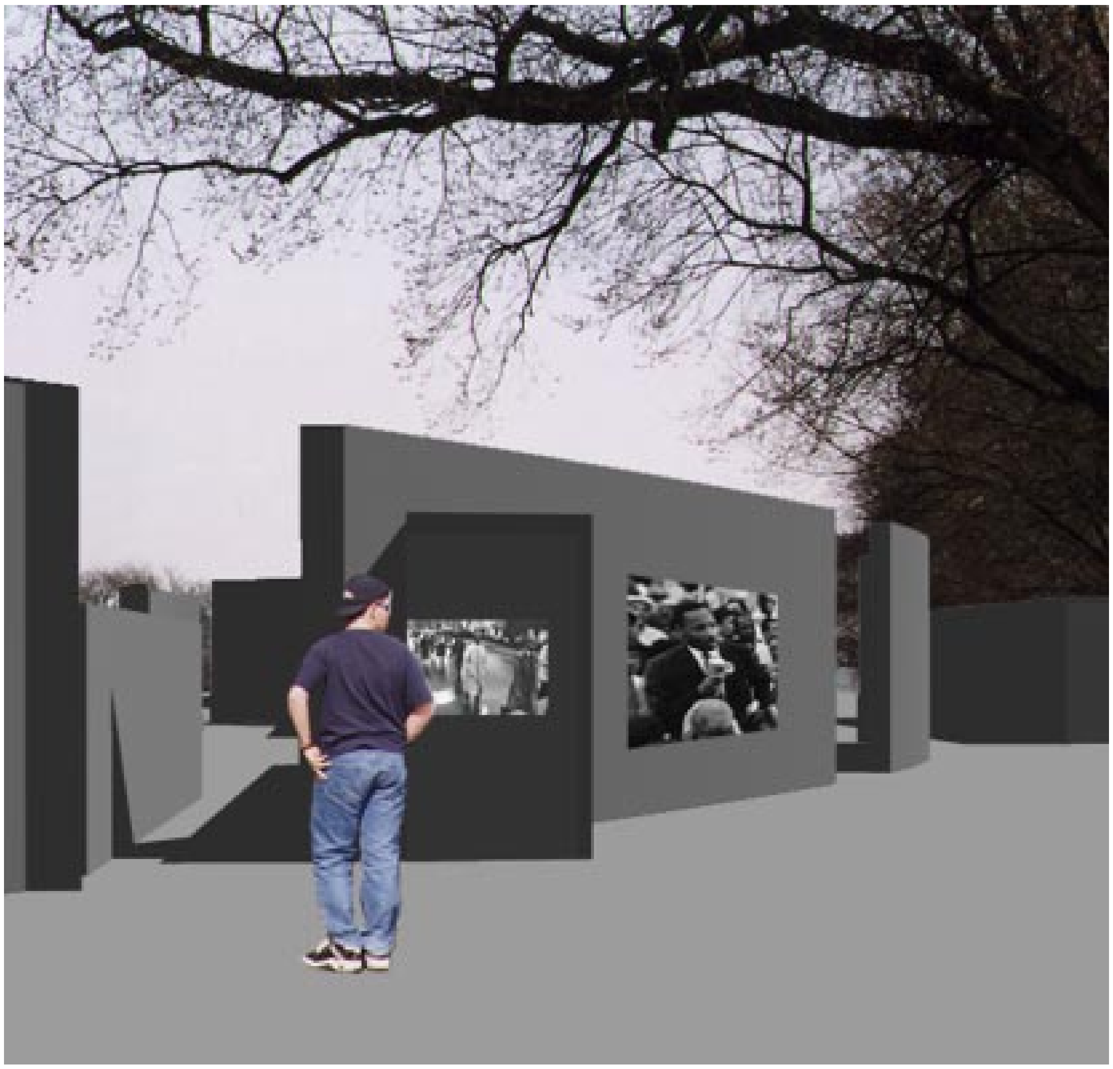


Figure 20. Interior view of Anthony Guma's final design for an open-air museum for Martin Luther King. Here the placement of some images makes the illustration appear real, but it is still a three dimensional diagram

Introduction

The basis of this thesis is the link between the Constructionist framework for learning, and the cognitive aspects required by designing. If it is possible to describe design as a constant learning activity (Guarra, 1974), then it is appropriate to use research in education theories as a framework from which to suggest ways of understanding the design process as well as ways of teaching it to students. To do so requires an understanding of the differences between the Constructionist and Socio-Cultural models of learning exemplified by the best architecture schools, and the Behaviourist model of learning that exists in most secondary schools, and with which many students arrive at university. Learning to design, for these students involves learning a new model of learning.

Behaviourism

The model of learning in many primary and secondary schools is an old but recognisable one. Children are empty vessels waiting to be filled with facts by the teacher, through a system of reaction and reward: the desired reaction is promoted by and rewarded with a suitable incentive. Learning is seen as “a measurable difference in displayed proficiency.” (Koschmann, 1996 p.6). This model was a pervasive view that controlled the direction of early research into human development. Researchers developed an understanding of the conditioned reflexes that can develop in laboratory animals when presented with a behaviour-reward system. Rats could learn to find food in a particular place in a maze, for example, after a simple process of ‘teaching’ them what behaviour would result in the edible reward. This model of learning is known as Behaviourism. Behaviourists believe we are born with certain behavioural traits and reflexes already ‘hard wired’ into our brains that cause us to respond to external stimuli in a predetermined manner. The brain is seen as a ‘black box’ with no need for further study or explanation; mental processing was not considered to be particularly important. In many secondary schools the Behaviourist model is still commonplace. In these schools, there is a three step approach to teaching: the teacher initiates, the student responds, and the teacher evaluates (Scardamalia & Bereiter, 1994). In such learning environments, there is an emphasis on a ‘mimetic’ method of teaching; the teacher demonstrates the desired performance which the student replicates (Gardner, 1991). There is a sense of ‘entity learning’ – you either ‘get it’ or you don’t, and there is no understanding of how an individual might go from ‘not getting it’ to ‘getting it’. This aspect of entity learning forms a potentially significant stumbling block for students of architecture just entering university. They often assume that the process of the design of a building is something that occurs in one giant step. The building is assumed to ‘appear’ in this mysterious single step, and when this fails to occur to them in their first design studio they often assume they can’t ‘do’ design.

A second aspect of a Behaviourist teaching in schools that inhibits an understanding of the design process is its tendency towards atomism. In Behaviourist teaching the complexities of our everyday experiences are broken down into much smaller processes that can be used as building blocks, much in the same way the human problem solving researchers divided a complex problem into sub problems (Alexander, 1966, Newell & Simon, 1972). The emphasis is on learning simple skills and facts in isolation that can be used some time later in more complex settings. However, in Behaviourist education there is little or no demonstration of how these isolated facts and figures can be recombined in new settings to solve new problems. The process of this recombination is treated in the same manner as the process of understanding each of the isolated facts; since behaviourism ignores mental processing and the appropriateness of thinking, it cannot teach either recombination nor the process of how an individual comes to understand the skills or facts being taught.

The predominant interest of Behaviourist educationalists is the retention of these isolated facts. The process of assessment in Behaviourist models of teaching is simple; the child is tested on their retention, and the outcome of the tests is used to determine their level of understanding. In the US, these tests are typically standardised and are used as an objective means of measuring performances. There is never any question of whether a child who scores well actually truly understands the work they have completed, an issue that has occupied researchers at major universities, particularly in physics (Gardner, 1991, White, 1981).

Behaviourism in Architectural Education

It can be assumed that the average student arriving at a school of architecture has come from a typical, general primary and secondary education system. In the US, as in many parts of the world, this often means a predominantly Behaviourist system. Many students expect there will be an expert – the studio professor – who will tell them what to do and how to do it. It is therefore possible to say that many students who come to schools of architecture arrive with a mental model of learning as one in which they receive information, remember it, and regurgitate it at the appropriate time. If they do this well, the results of testing them will prove that they have understood the material they have memorised. This view is largely unsubstantiated[3], but is suggested by views of the Behaviourist nature of some aspects of western education (Perkins, 1992, Perkins & Salomon, 1987, Perkins & Simmons, 1988). In recent years much has been done to overcome this (Papert, 1980, Papert, 1991, Resnick *et al.*, 1996, Resnick & Ocko, 1990) but students at university today may have benefited from this change to a much lesser degree than their younger siblings.

In many schools of architecture there is a danger that this Behaviourist model of learning is perpetuated, reinforcing the student's preconceived notion of learning. Architecture is seen as a very complex activity, and in a Behaviourist manner architectural education attempts to reduce this complexity by breaking it into constituent parts. As a result there are separate classes that cover, for example, the behaviour of structural aspects of buildings, and building services. These two are usually taught by separate people, so it is rare that the implications of one upon the other are discussed, despite the necessarily holistic nature of architecture. The projects set in design studios are seen as the location for such a discussion, but the focus of the studios tends to reach a point in the design that falls just short of consideration of such detail. Within the design studio, the level of complexity of design as a 'complete' activity is seen as achievable if the projects early in the students' academic careers involve very simple design solutions, and later ones deal with more complexity. As with primary and secondary Behaviourist education, there is a hierarchical progression through design school. Students rarely have the opportunity for supported reflection upon their combined studio projects, nor is it common for them to have direct encouragement of design ideas to be continued beyond the semester and across studios. This approach supports the inherently Behaviourist model of learning with which the student typically arrives: each studio is an unrelated hoop to be jumped through or box to be checked. Once enough boxes have been checked the student deems herself to have succeeded.

In keeping with the Behaviourist's view of the mind, both students and professors often see designing as a 'black box' activity which is mysterious and requires special, undefined abilities to achieve successfully (Coyne & Snodgrass, 1991). In his discussion of the design studio as a model for teaching the professions, Schön speaks of the confusion with which students view the process of designing (Schön, 1987). There is a discussion of the methods adopted by successful students, and those that inhibit the weaker ones, but he does not suggest how it is possible to transform the latter into the former. Design is seen by students and professors as a process that can either be 'done' or 'not done'. Students classify themselves fairly quickly as belonging to one or other, and once this conception is established it is difficult to remove. The incremental process of designing, where the architect tinkers with a design that gradually evolves over time is a process that most beginning students must learn. Yet it is rare for students of architecture to be directly taught these specific techniques that will help them learn to 'do' design.

Assessment of the process of learning the students do engage in, through the jury system, forms a similar, Behaviourist trap, despite appearing from the outside to be primarily a Constructionist or even Socio-Cultural activity, as described later in the thesis. Students pin up their work in the form of external representations of internal ideas, ready for discussion. Yet in practice many students, far from finding such a process constructive, instead learn to dread the procedure and see it as a highly alienating experience. In its worst form a jury panel consists of 'experts' from within the school – professors – and from outside – practising architects – who take the opportunity to voice their opinions about the students' work and criticise their representations. It is rare to hear from a jury questions aimed at eliciting either better understanding on the part of the professional, or by the student. Instead, it is a common occurrence for there to be monologues from the experts representing their opinions or experience, intended as information and knowledge the student would be advised to memorise (Frederickson, 1991). The subjective nature of the jury process is highly influenced by the quality of the representations the student uses to present her work. As with Behaviourist standardised tests in schools, it is possible to confuse an ability to create beautiful representations with an understanding of the design process. The latter is more difficult to assess than the former (Oxman, 1999). As Oxman suggests, it is easier to assess the quality of the representations as iconographic images, rather than the cognitive processes involved in the development of the design. Students with good representation skills will usually do better in architecture school than those whose skills are less well developed, although there is no proven correlation between good architecture and good drawing skills. Despite this, and in a Behaviourist atomistic model of education, the skills required for the creation of these representations are therefore taught in schools of architecture in specialist courses.

In schools of architecture that teach some form of digital manipulations – today this is most schools – such a Behaviourist approach is also commonplace in the CAD course. Many schools see the use of digital technology as a process as complex as design itself. As a result, and in keeping with the Behaviourist model of architectural education, classes teaching computer skills are held away from the context of their intended use – the design studio. They are divided into smaller chunks of skills that need to be learned and memorised. The production of imagery with ever increasing levels of sophistication is considered proof that the student can use digital media effectively in her design process.

In recent times educational researchers have looked towards the architectural studio as a Constructionist model of teaching that might be adapted to suit the needs of younger students (Shaffer, 1996, Shaffer, 1998). From the outside, the studio system does not readily reveal its Behaviourist tendencies. Instead it appears that the students are constructing their knowledge of architectural design at the same time as constructing representations of their ideas through models and drawings (Galle & Kovacs, 1992b). The process of designing has been described by Schön as one of ‘reflection-in-action’ (Schön, 1984a). Schön describes what he terms a ‘reflective practicum’ whose “main features are learning by doing, coaching rather than teaching, and a dialogue of reciprocal reflection-in-action between coach and student” (Schön, 1987 p.303). Schön is describing a Constructionist model of teaching, one that exists in the best schools of architecture. To be successful, architecture students need to learn to develop their designs in small but significant incremental steps, to understand the link between process and product, to bring together a wide range of knowledge and apply it to the studio project. In short, they need a Constructionist model of learning to help them learn about design, and ultimately to help them to *do* design. It is clear that there is a conflict between the Behaviourist model of learning with which the student arrives at university, and that required in the design studio. Yet as has already been described, there are many aspects of architectural education that are, or are in danger of becoming, Behaviourist, thus perpetuating the perceived model of learning. The lack of recognition of the mental model of learning with which the architecture student arrives at university compounds the problems facing the student (Gardner, 1991).

Constructivism

The process of thinking through a design idea, developing it through a 'conversation with the materials of the situation', is clearly an active process. The designer is involved in the creation of external representations that both represent and support the cognitive processes she is utilising. This recognition of cognitive processing was of no interest to the Behaviourists. But at the turn of this century, the understanding that thought might be an independent process, and judgment be considered an activity occupied the psychologists who formed the 'Würzburg School' (Piaget, 1963). In this model, it is believed that reactions to external stimuli are not determined prior to experience, but result from early processes of random trial and error. These processes become internalised as anticipations resulting from our awareness of the consequences of our actions. Knowledge is therefore not a predetermined activity as the Behaviourists believed, but the result of this collaboration between experience and deduction. Nothing is fixed or predetermined. In this model, our intellectual operations constitute genuine actions that can be seen as potential experiments on our surroundings (Piaget, 1963).

The Behaviourists believed that logic existed as a universal "truth" independent of our ability to think. In contrast, the Swiss developmental psychologist Piaget suggested that instead of seeing thought as a result of independent logic, we should instead perceive logic as the mirror of, or result of, thought. Piaget argued that we are neither born with knowledge, nor are we empty vessels waiting to be filled. He saw development and learning as a long and complex process where each individual constructs their own understandings of their world. Piaget is seen as the forefather of the Constructivist view of learning, where the learner plays an active role in constructing her own understanding. Since the world around us is constantly assimilated by us, it is possible to understand that every new thought we have must modify or contradict a previous one. Piaget saw this as a process of gradually constructing frameworks that permit the smooth incorporation of new elements of understanding. He considered the essential characteristic of thought to be a process of internally extending action. It is an act to think.

However, the type of concrete thinking used in the architectural design studio and involving the manipulation of external representations was considered by Piaget as a stage that was eventually superseded by a superior form of cognitive activity. The dominant belief of Western society is that logical, formal, abstract thinking is the highest, most advanced process a mind can attain. Piaget concentrated almost exclusively upon this formal thinking in his experiments. He demonstrated a progression of child development from egocentric beginnings, through concrete thinking, to a final "formal stage" where logic and abstraction predominate (Piaget, 1964, Piaget, 1926, Piaget, 1963, Piaget, 1977). Although Piaget recognised the importance of concrete, object based thinking as a stage of child development, he believed it was eventually superseded by formal, logical thinking when the child reached his teens.

This view of formal, logical thinking being superior to other models of cognitive activity has predominated in Western society for centuries. Action, as the process of doing or making something, has traditionally been viewed as secondary to intellectual activity. Nowhere is this view more true than in today's universities, where knowing and thinking are considered as immaterial, spiritual, and generally of a higher order than any practical or manual activity. As a result, architecture, concerned as it is with practical activity and creation, occupies a correspondingly the uneasy position within academia (Schön, 1987). The philosopher John Dewey asked why in this hierarchy between knowing and doing exists, and wondered what the effect might be if the division was instead one of equality. He argued that action and knowledge are inextricably linked. "Action, when directed by knowledge, is method and means, not an end. The aim and end is the securer, freer, and more widely shared embodiment of values in experience by means of that active control of objects which knowledge alone makes possible." (Dewey, 1929 p.30). Dewey's view is that knowledge is "the fruit of the undertakings that transform a problematic situation into a resolved one" (Dewey, 1929 p.193). In other words, it is through the process of actively constructing something that we develop our understandings of the world.

Constructionism

The belief that action forms a significant part of our understanding of the world was further developed by the Constructionist Seymour Papert, who argued that Piaget's approach overlooked the possibility that a process of concrete, object based thinking might be a valid alternative to formal thinking for some adults in some situations. Papert shared with Piaget a belief in the essential value of creating opportunities for children's learning through exploration, invention, and construction. They both viewed the child learner as actively engaged in formulating concepts and testing them out on the world around them. "Better learning will not come from finding better ways for the teacher to instruct but from giving the learner better opportunities to construct" (Papert, 1990 p.3). Piaget had challenged Behaviourist theories by questioning the assumption that knowledge is supplied by the teacher. In Piaget's view, knowledge itself is constructed by the learner. But whilst Piaget saw the process of concrete thinking as a childhood phase that was to be superseded by logical thinking later in life, Papert argued that the process of internalising what is outside and externalising what is inside is something we do at many stages of our lives and in many different circumstances. Papert also extended Piaget's theories by adding that one of the best ways to construct knowledge is through the construction not only of mental objects, but of physical, external artifacts that are meaningful to the learner and to those around her.

A significant aspect of classrooms that use a Constructionist framework is the focus on effort rather than ability, developing in the child an understanding of the significance of incremental, progressive effort in coming to understand and achieve things. "Comprehension is seen as an active process of construction rather than a simple form of reception" (Chan *et al.*, 1992 p.98). Learning occurs not through recording information that stems from a centralised source – the teacher – but by an active process of interpretation (Resnick, 1989). Learners understand through the creation and construction of mental models that represent their understanding. The learner works through her ideas with a great deal of autonomy. The teacher remains on hand, but not as the universal source of all knowledge. She takes instead the model of a coach and mentor, encouraging students to work through their own problems and being ready to guide where necessary (Nickerson *et al.*, 1985, Perkins, 1992).

The view of a teacher as coach or mentor has obvious links with the role of the studio professor and the history of the master-apprentice model behind the studio setting. Two other aspects of Constructionism are significant for the successful design studio as an educational environment, and for the successful development in each student of a model of the design process. The first is

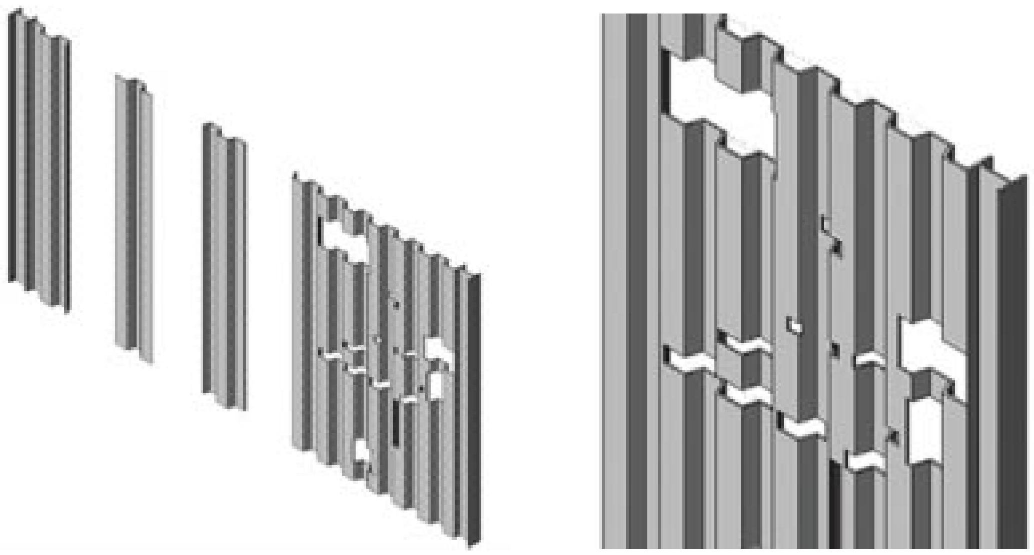


Figure 21. Christopher Mulvey's first exploration of VectorWorks revealed a concentration on the software as a tool that was to last for the entire semester. He explored the possibility of making holes in complex objects

the technique of using concrete representations or objects as a means by which to think about a problem. Papert is interested in “the process of invention of ‘objects-to-think-with’, objects in which there is an intersection of cultural presence, embedded knowledge, and the possibility for personal identification” (Papert, 1993p.11, Yakeley, 2000). The design process uses objects to think with in sketch design, but it is a cognitive skill that many students must relearn.

The second technique linking the Constructionist model of learning with the process of design, and learning to design, is the use of the principles of debugging in computer programming. Called ‘tinkering’ by Turkle and Papert, this involves working with something to get the required answer rather than entirely rejecting it each time and starting over, an antidote to the Behaviourists’ entity learning (Turkle & Papert, 1990). Papert’s example is from simple word processing – learning to compose as a process of tinkering with words, which is easier to do on a word processor than on paper. Architects learn to tinker with their sketches, becoming as they do so personally involved with the results. “Constructionism ... asserts that learning is an active process, in which people actively construct knowledge from their experiences in the world ... people construct new knowledge with particular effectiveness when they are engaged in constructing products that are personally meaningful” (Resnick, 1995 p.24). Architectural design involves the ability to develop an idea through the use of externally created objects that are successively modified in individual increments. This incremental process has been likened to a conversation “with the materials of the situation” (Schön, 1983) where the designer must ‘listen’ to the sketches and “remain open to the back talk of the situation” (Bamberger, 1991) to make the next change and thus develop her idea.

Tinkering with the design, remaining open to the back talk of the situation, object based thinking, are all techniques promoted in the design studio, with the studio professor as a coach and mentor. This suggests that the use of Constructionist principles as a model for learning in the design studio might be useful to students trying to develop their own understanding of what the design process means to them. The correlation between designing and Constructionist theories of education is more than superficial techniques, however. Constructionist learning paradigms are deeply embedded in the process of all designing, not just when the designer attends university. The process of designing is a process of constant learning and inquiry (Gero *et al.*, 1991, Rowe, 1987). The student designer must learn to learn in a Constructionist paradigm in order to learn to design. Ultimately the model shifts from a learning paradigm to a design paradigm, since all these principles apply to the design process itself. Papert’s example of tinkering with word processing, rather than constant rewriting of the entire composition, mimics this change in

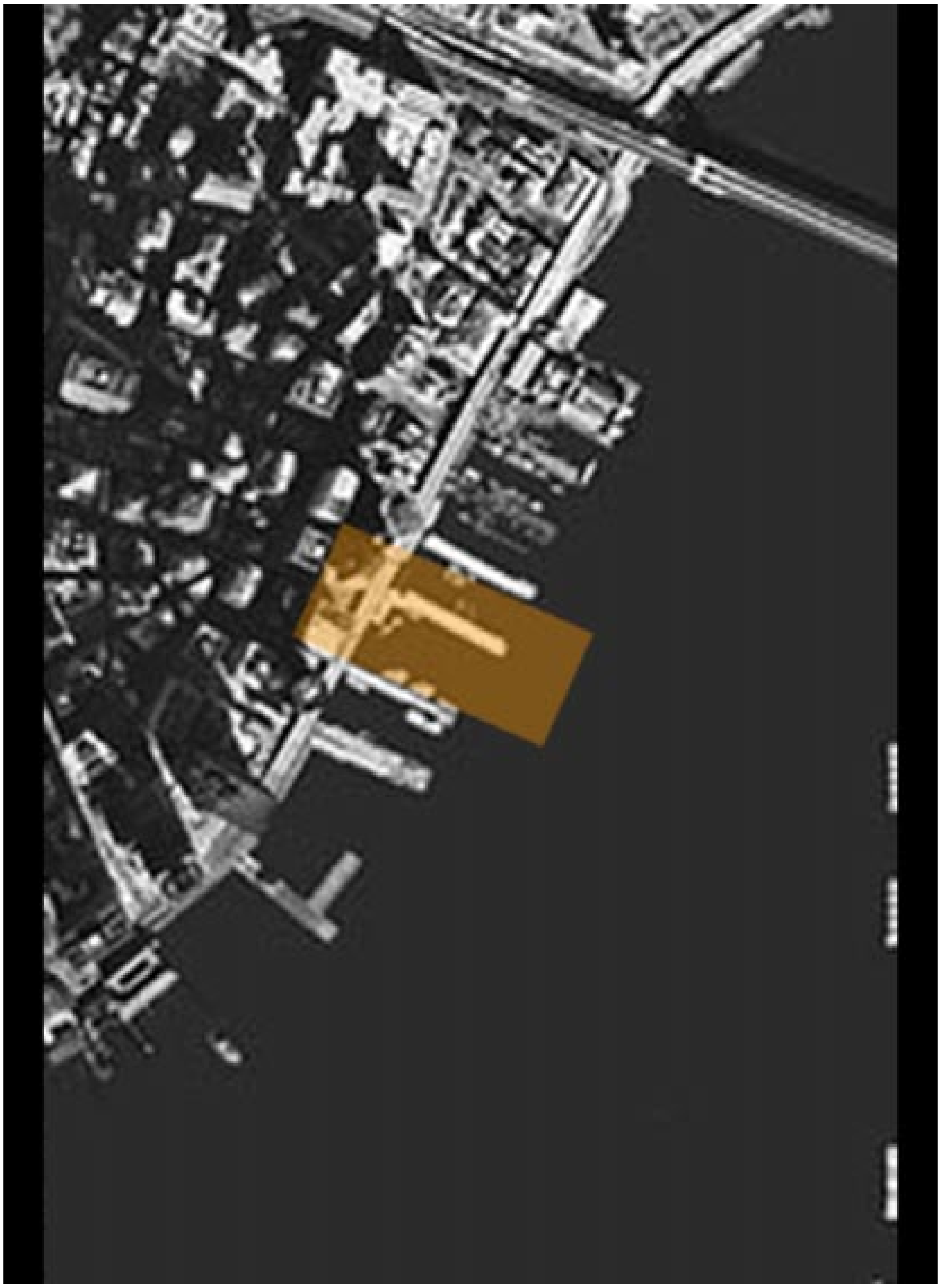


Figure 22. Christopher Mulvey's project for the semester was a design competition for the expansion of New York by developing the boundaries of Manhattan Island into the water

approach many novice design students must undertake. It is sometimes difficult at the start of a designer’s career to convince her to tinker with, rather than entirely reject, her proposed design each time it is criticised. The formal, ‘hard’ approach taught in high school must be rejected in favour of the ‘softer’ approach of earlier schooling. The rest of her design career will be spent applying these techniques of learning to each new design situation, inquiring into the nature of that situation in a Constructionist manner that permits solutions to be explored.

However, it is the Constructionist’s statement of equality between concrete and formal thinking styles that has most significance for architectural education. Architecture students arrive at schools of architecture fresh from an education system that has promoted learning through doing as a secondary activity to the ‘higher’ order of abstract thinking. They often come with portfolios exhibiting skills in artistic work, but rarely arrive with the ability to think and work using the language of those images to develop ideas. Encouraging them to return to the techniques of earlier years is difficult, since the traditions of Western thinking prevail. In Epistemological Pluralism, Turkle and Papert “isolate two approaches [to knowledge] which serve as ideal types, theoretical prisms through which to see simplified projections of more complex realities.” They question the “hegemony of the abstract, formal, and logical as the privileged canon in scientific thought.” Turkle and Papert term these two approaches as ‘soft’ and ‘hard’:

> “Soft is a good word for a flexible and non-hierarchical style, open to the experience of a close connection with the object of study. ... Observation of the soft approach ... provides examples of the validity and power of concrete thinking in situations that are traditionally assumed to demand the abstract. It ... suggests the need for closer investigation of the diversity of ways in which the mind can use objects rather than rules of logic to think with.
> “The ideal typical hard and soft approaches are each characterised by a cluster of attributes. Some involve organisation of work (the hards prefer abstract thinking and systematic planning, the softs prefer a negotiational approach and concrete forms of reasoning); other attributes concern the kind of relationship that the subject forms with computational objects. Hard mastery is characterised by a distanced stance; soft mastery by a closeness to objects.
> “Hard mastery is resonant with the logical and hierarchical elements of the traditional construction of “scientific method.” Soft mastery has always had its

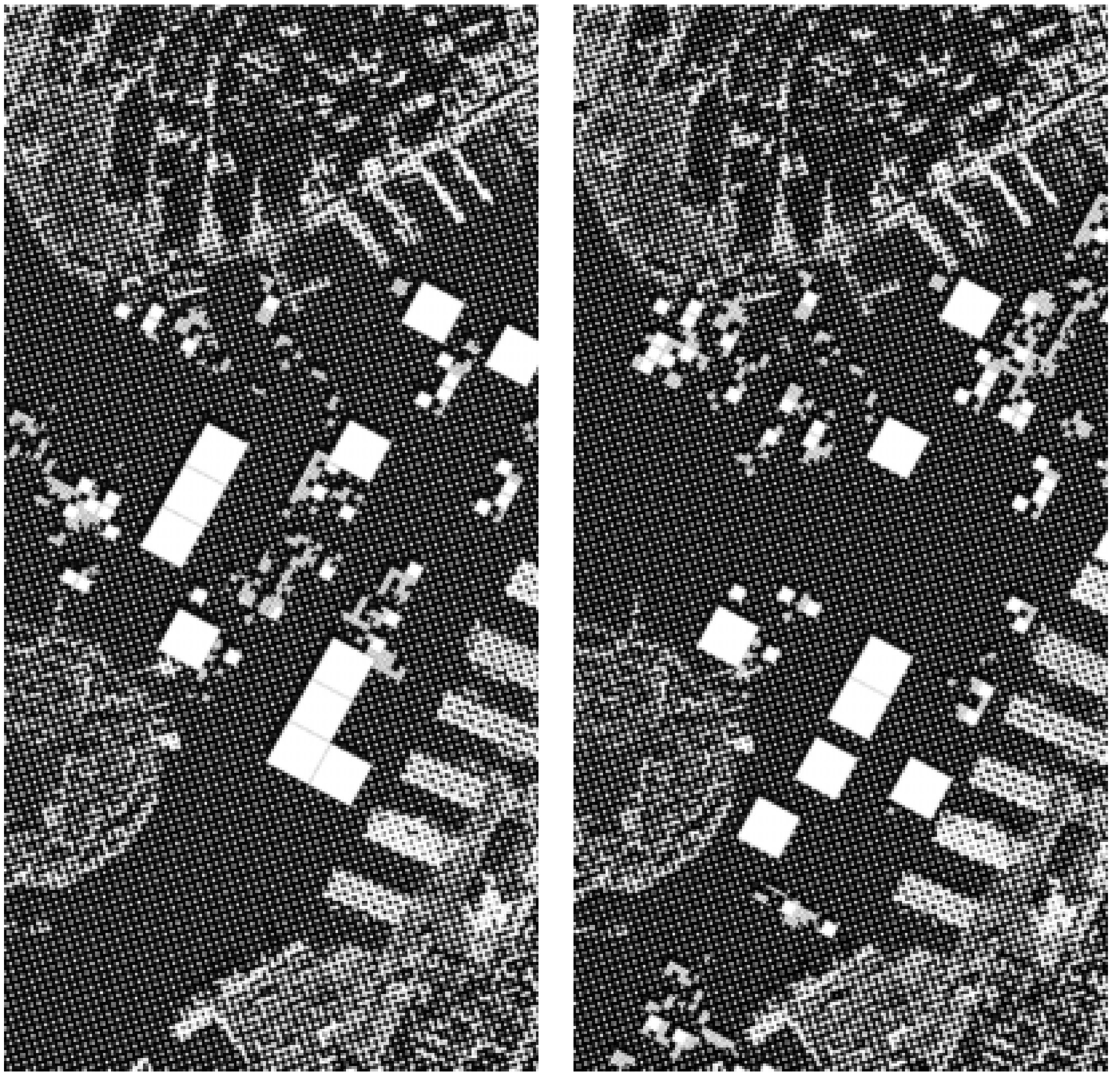


Figure 23. Christopher Mulvey wrote code based on the Mughul Garden designs (Knight, 1990, Stiny & Mitchell, 1980), dividing squares in a fractal manner, and placing them randomly on the water.

place in the discourse of the arts and has always been glimpsed in the autobiographical writings of scientists. Only recently has it gained academic recognition as an integral element of scientific practice" (Turkle & Papert, 1990 p.350).

For Turkle and Papert, abstract or hard thinking is not superior to soft or concrete thinking; they are simply two different methods of working. Soft thinking and a "closeness to objects tends to support a concrete style of reasoning, a preference for using objects to think with, and a bias against the abstract formulae that maintain reason at a distance from its objects" (Turkle & Papert, 1990 p.350). Architecture's greatest difficulty in its uneasy relationship with the 'hard' world of academia is its insistence upon less formal, 'softer' styles of thinking (Schön, 1987). This acceptance of different styles of thinking is welcome news for architects, particularly when attempting to learn computing skills, including programming, outside of the CAD draughting and rendering traditionally offered to them. These skills are traditionally taught in a more formal, 'hard' manner that many architecture students find alien. Pedagogical approaches away from the design studio need to reflect the changes in learning the student has developed there, and with which she has come to feel more comfortable.

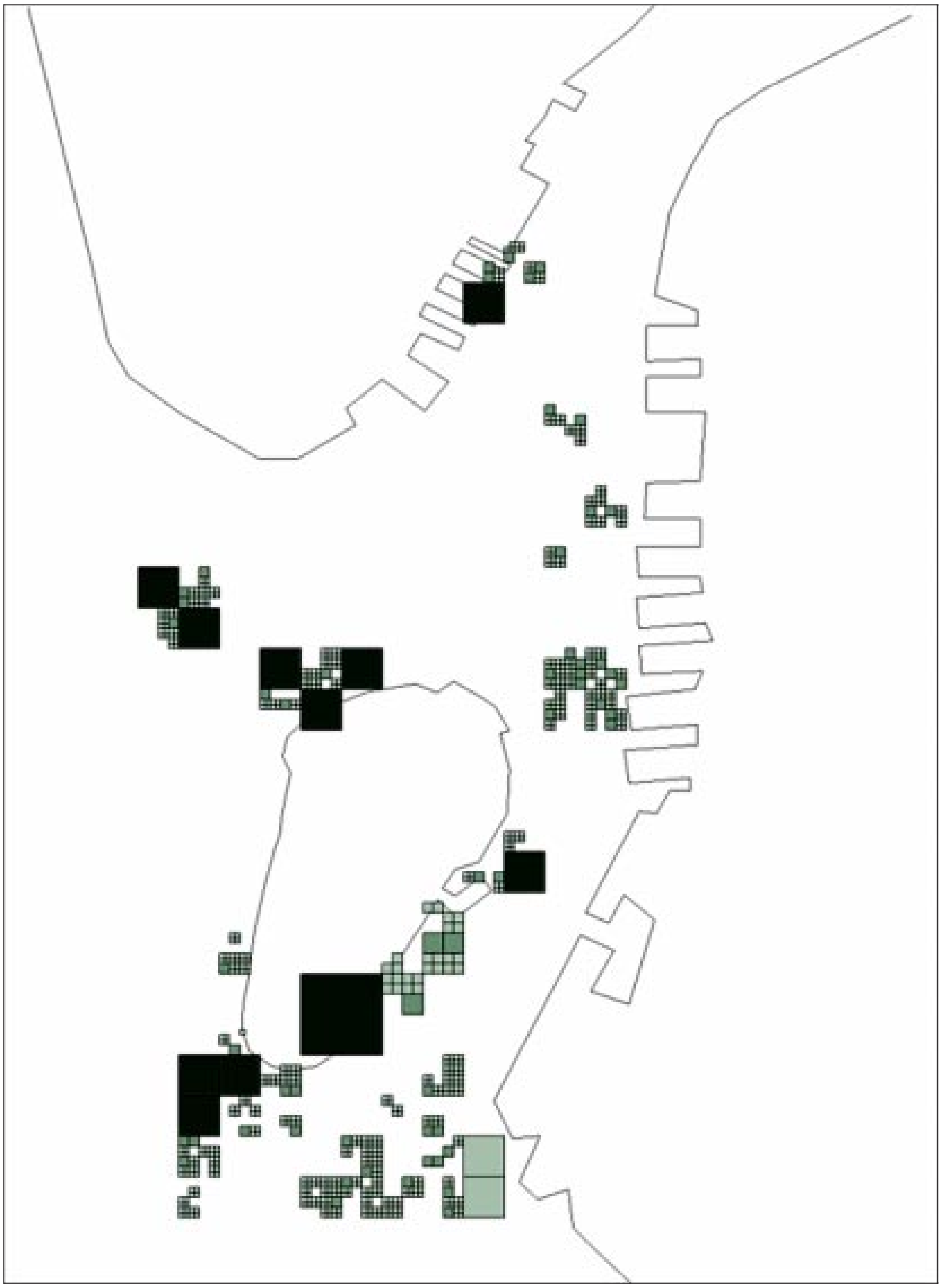


Figure 24. Christopher Mulvey wrote code based on the Mughul Garden designs (Knight, 1990, Stiny & Mitchell, 1980), dividing squares in a fractal manner, and placing them randomly on the water.

Socio-Cultural Learning

However, whilst it is clear that there needs to be a shift in understanding of learning from a Behaviourist model to a Constructionism one for the architecture student, the theories of Constructionism alone do not offer the whole solution to architectural education, nor recognise the significance in the development of understanding of the context within which learners are embedded. Constructionism accepts that the process of creating external representations of an internal cognitive understanding provides a common basis for discussion with others, but as a model for learning there is a concentration in Constructionism on the development of individuals. The traditional, Behaviourist theory of learning assumes that knowledge and skill are context free, and that they can be analysed into component parts that function in the same way no matter where they are used. Complexity is avoided; everything is divided into separate elements. The assumption is that these building blocks can become the basis of more complex studies. But research has demonstrated that knowledge is best retained when it is embedded in a predefined structure or context (Perkins 1992). “When learning is divorced from doing a meaningful task ... then learning becomes just another chore” (Soloway *et al.*, 1994 p.40). In this respect the educational philosopher Jean Bruner saw Piaget as “mind-” or “individual-centric” and criticised the Swiss psychologist’s lack of consideration two crucial aspects affecting our development and the way we learn (Bruner, 1966). Bruner believes that education is a social activity, especially in the early stages. He argues that other human beings play a part in our development, as do artifacts and inventions that exist in our everyday experiences. Socio-Cultural theories of development put forward a strong argument for the centrality of cultural factors in any consideration of human development. The context-dependent theories of Situated Learning, and with it Socio-Cultural Learning, suggest that the principles of Constructionism need to be combined with an awareness of the context of the learner – mental, physical, and social. “Students learn though an active social process of meaning construction; understanding is built up through the acts of conversing with others, constructing artifacts, and reflecting on those conversations and artifacts. ... Students need to actively engage in projects, and teachers need to act as mentors, coaches, managers” (Soloway *et al.*, 1994 p.41).

Researchers in many fields, including architecture, cognitive science, and education, have begun to look at the role interpersonal relations play in our everyday lives, and in particular the role of situated knowledge (Winograd & Flores, 1986). In schools and many other learning institutions, including schools of architecture, people are expected to learn and perform individually. Yet outside of those institutions we exist in a society where most mental processes take place in the context of some form of shared activity. In many cases cognitive processes are often distributed

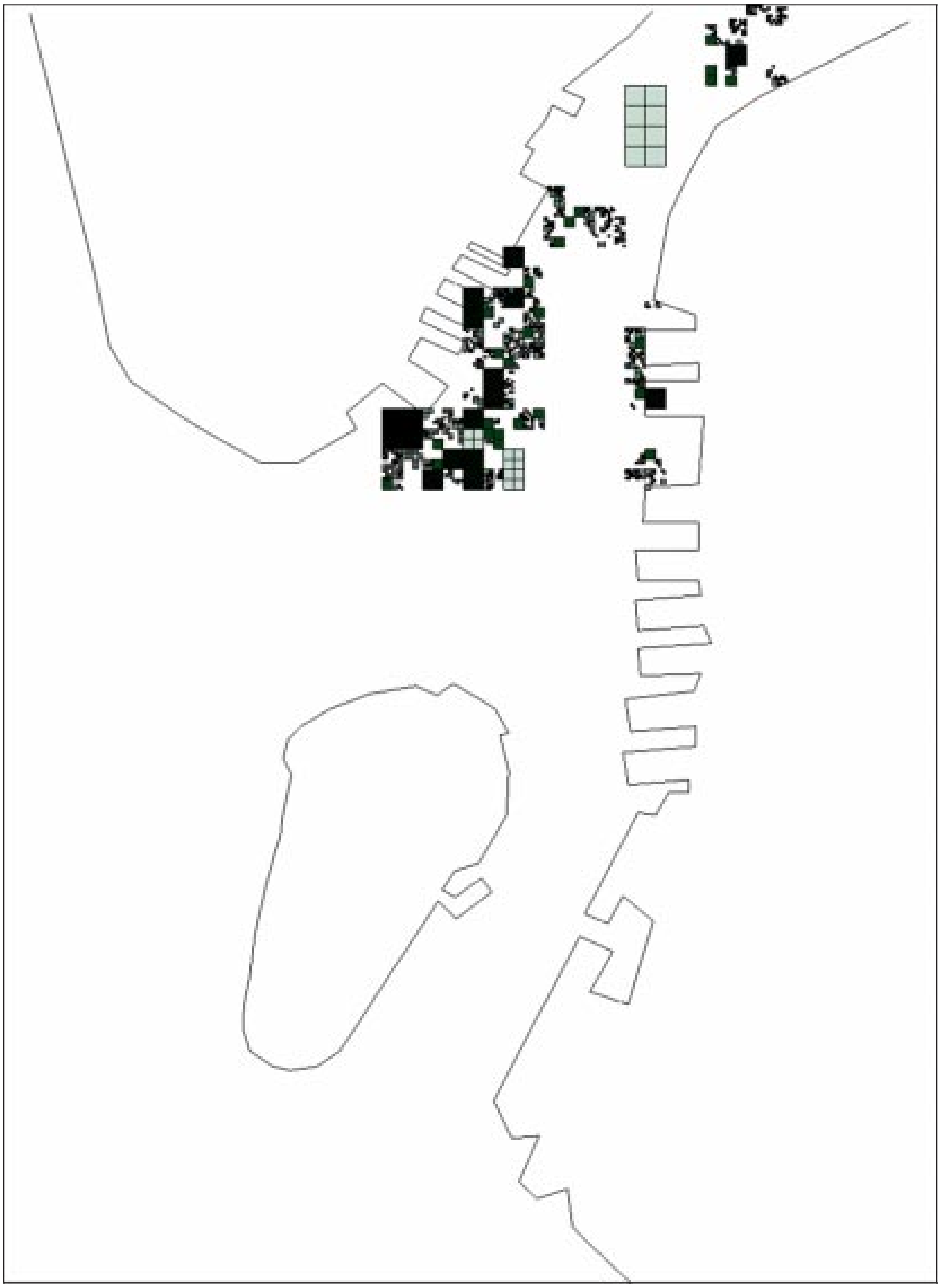


Figure 25. Christopher Mulvey wrote code based on the Mughul Garden designs (Knight, 1990, Stiny & Mitchell, 1980), dividing squares in a fractal manner, and placing them randomly on the water.

over several individuals. The terms situated knowledge, situated learning, or situated cognition have a variety of meanings in different disciplines, but the common theme is an awareness of the context of our actions. In these theories, learning is seen as a "process of entry into a community of practice" (Koschmann, 1996 p.13). Architecture is necessarily a cultural discipline on many different levels, not least of which is the need for the architect to interact with many other professionals during a building's development. Architects both work in teams and manage other teams, and the development of an awareness of the significance Socio-Cultural skills is critical.

Bruner followed in the footsteps of the then little published Soviet psychologist Lev Vygotsky. Vygotsky considered cognitive development to be strongly influenced by cultural, mainly interpersonal and communicational, factors. His general theory was that the development of a child is strongly influenced by her interactions with the world around her. Vygotsky saw language as the tool with which we interact in this manner. "Every function in the child's cultural development appears twice, on two levels. First on the social, and later on the psychological level; first between people as an interpsychological category, and then inside the child, as an intrapsychological category. This applies equally to voluntary attention, to logical memory, and to the formation of concepts. All the higher functions originate as actual relations between individuals" (Vygotsky, 1978 p.52). Skills and knowledge are dependent on context: mental, physical and social.

The mental context of a learner includes the mental models she may have already formed in her attempts to understand the world. Implicitly held mental models can interfere with learning; Gardner suggests that the mind of the five year old exists in all of us (Gardner, 1991) waiting to reappear when other models of understanding the world around us fail. He postulates there is a problem with the mismatch between the information we are taught at school, and the ideas we have already formed before the first day. Because there is no recognition of this disjuncture, there is no possibility for the child to resolve the differences, and so the ideas do not go away but instead go underground, ready to re-emerge when other theories do not match with her empirical understandings of the world (White, 1984). But suppression is not resolution.

The social context of our development can be seen in the desire for recognition and respect from our peers, the desire to have impact, and the desire to participate. These factors influence the actions we take to learn and understand at every level of development, including the rebellious teenage years, when being seen to be *not* learning often plays an important role in social acceptance. Knowledge and intelligence can be seen to be socially distributed. Harel talks of "collaboration-in-the-air", the process by which ideas become common knowledge (Harel &

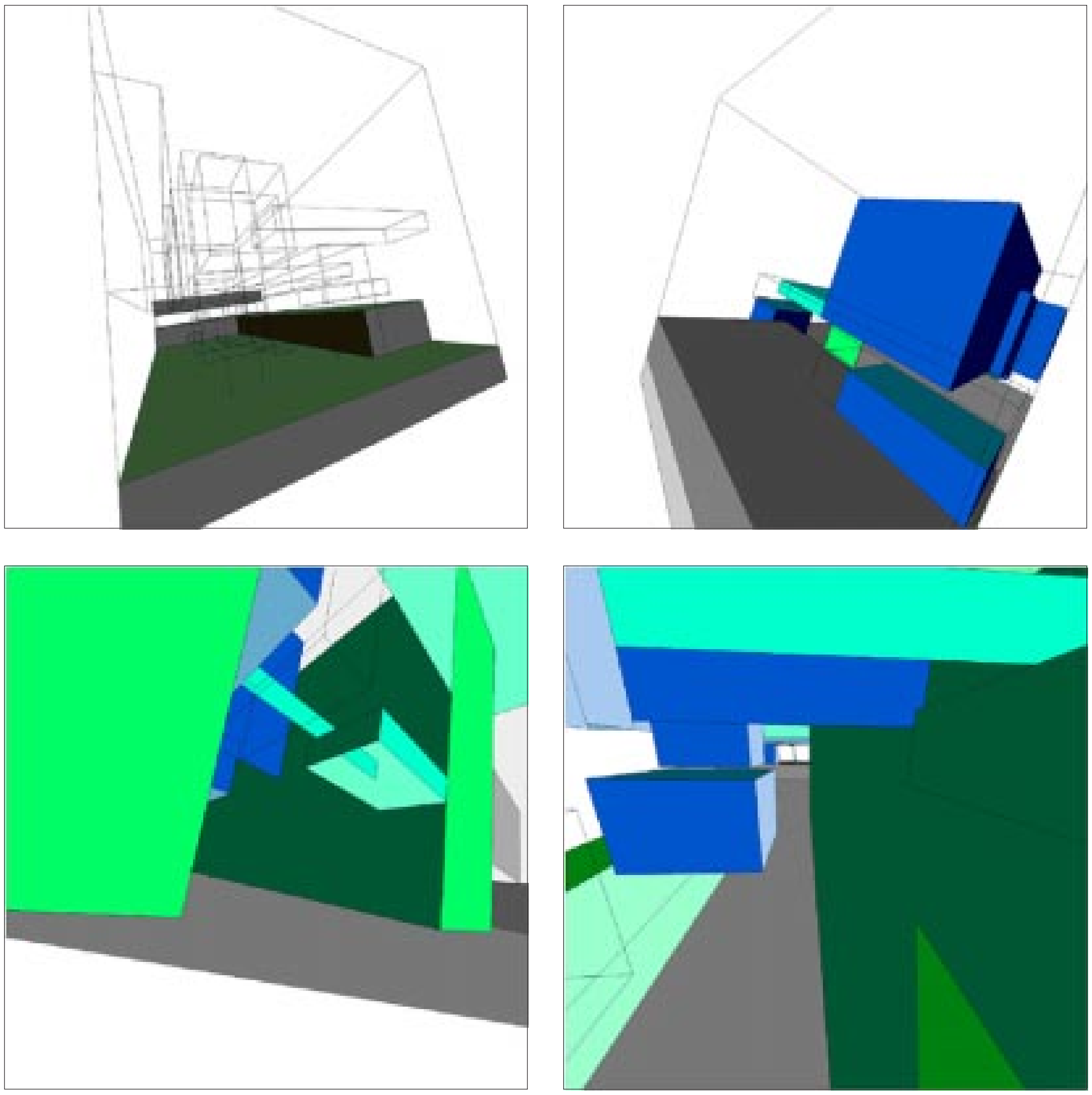


Figure 26. When Christopher Mulvey began to look at three dimensions for his project, he began to include less possibility for variation, coding only to produce known outcomes. These are all views of the same model

Papert, 1990). The MIT based annual Robotics competition exhibits similar collaboration – albeit unwittingly between the contestants, who keep one eye on their rivals whilst designing. Most of the resultant robots are very similar in their conception and final design. Despite all this, architectural education however does little to promote or develop social skills in the workplace, concentrating on the whole upon individual student development in the design studio. Unplanned collaboration – where students are not working in groups but help each other on individual projects – is actively discouraged by a jury system that values independence and novelty over collaboration and cooperation.

Decisions made in development and learning are affected by the physical as well as the mental and social contexts of the learner. Suchman refers to “situated actions” by which she means “actions taken in the context of particular, concrete circumstances” (Suchman, 1987 p.viii). In education, the traditional model for situated actions is the apprenticeship. In societies where this is still a common method of learning, apprenticeships are usually found in crafts and industries that permit the learning of skills by participation. In a true apprenticeship there is a premium on learning the required skills fast and well. Through participation the apprentices gain a greater understanding of the link between means and consequences (Dewey *et al.*, 1969). Clearly, by its nature apprenticeship is very contextualised. But true apprenticeships are no longer commonplace in today’s industrialised societies, where the cost of mistakes is too high.

The challenge for primary and secondary schools today is to link with the wider intellectual community, and to permit all aspects of the curriculum – practical as well as intellectual – to occur in a context that either is a part of, or reflects, their future use. Gardner claims the best chance for an education of understanding lies in linking certain aspects of apprenticeship with certain aspects of schools and other institutions. His position states that instruction needs to be linked to some level of sensorimotor stimulation, and that instruction should be contextualised (Gardner, 1991). Scardamalia and Bereiter argue for the development of “knowledge building communities” within schools, that interact with those in the wider world. They take as their model the collaboration apparent in research groups and corporate horizontal management structures. They see the major task of schools in today’s society as developing a “knowledge building discourse” where learners cooperate rather than compete to develop understandings (Bereiter & Scardamalia, 1993). Scardamalia and Bereiter take this one step further by suggesting that it is not until we change our perspective on education on a larger scale, and through doing so change our society, that we might reasonably consider our education system to meet the needs and requirements of a post industrial society. They have a vision of an expert society, where expert learning is considered normal (Bereiter & Scardamalia, 1993).

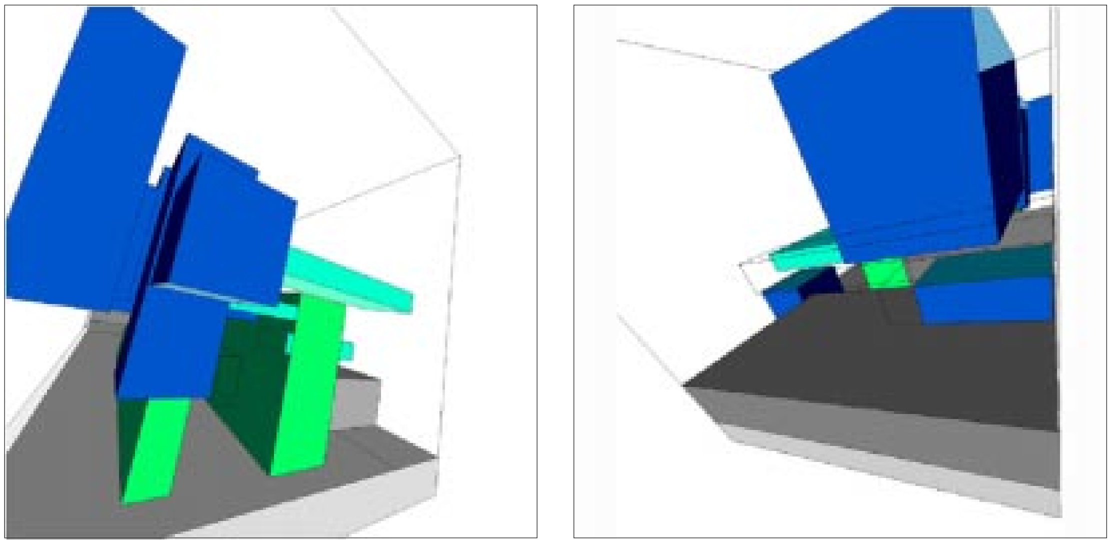


Figure 27. When Christopher Mulvey began to look at three dimensions for his project, he began to include less possibility for variation, coding only to produce known outcomes. These are views of the same model

The aims of architectural education

The application of a Behaviourist model to describe teaching in schools of architecture is perhaps extreme. Most schools are aware of the nature of the design process, and attempt to teach it in some form. In reality many schools strive to find a balance – albeit an unconscious one – between a tendency towards Behaviourist and the more desirable Constructionist teaching environments. It is assumed that the process of successfully completing many design studios in the course of a student's degree programme will automatically, through repetition, develop both an understanding of what this process involves, and how to go about doing it. Implicit in this assumption is a deeper one - that students, by default, will change from having a Behaviourist model of learning to having a Constructivist one. Many architecture professors are aware of the importance of bringing external professionals into the studio to develop the understanding of students and create a knowledge community, although they might not label these aims as such. As with other areas of tertiary education, there is an implicit attempt to develop in the student a sense of independence, and a sense of responsibility for her own learning.

Schools of architecture have more specifically design orientated aims in addition to developing these skills. Design is a process of exploration and discovery, and the best architecture schools aim to develop in their students an understanding of design as a choice between the products of those explorations, through the process of individual, one-to-one desk crits between professor and student. They aim to teach the link between the process of exploration and the further development of design ideas, and to develop in the student an ability to reflect upon her own design process. They aim to provide a collaborative environment where the risks of failure and the levels of competition are reduced, and where the construction of understanding of the process of design is possible. These are the aims of a school of architecture at its best, yet all of these aims are implicit and rarely labelled or discussed.

If there were a greater and explicit recognition of the differences between successful and unsuccessful teaching practices in architecture, including an awareness of the models of learning required and those with which students arrive at university, there would be a greater possibility of improving the less successful aspects of architectural education. The skills taught in the design studio, particularly the cognitive processes involved in designing, might similarly be improved if they were also explicitly defined and taught. The development of such metacognitive skills is assumed to come about through age, experience, and practice in the design studio. Forrest-Pressley describes metacognition as "a construct that refers, first, to what a person knows about his or her cognitive processes, and second, to the ability to control these cognitive processes"

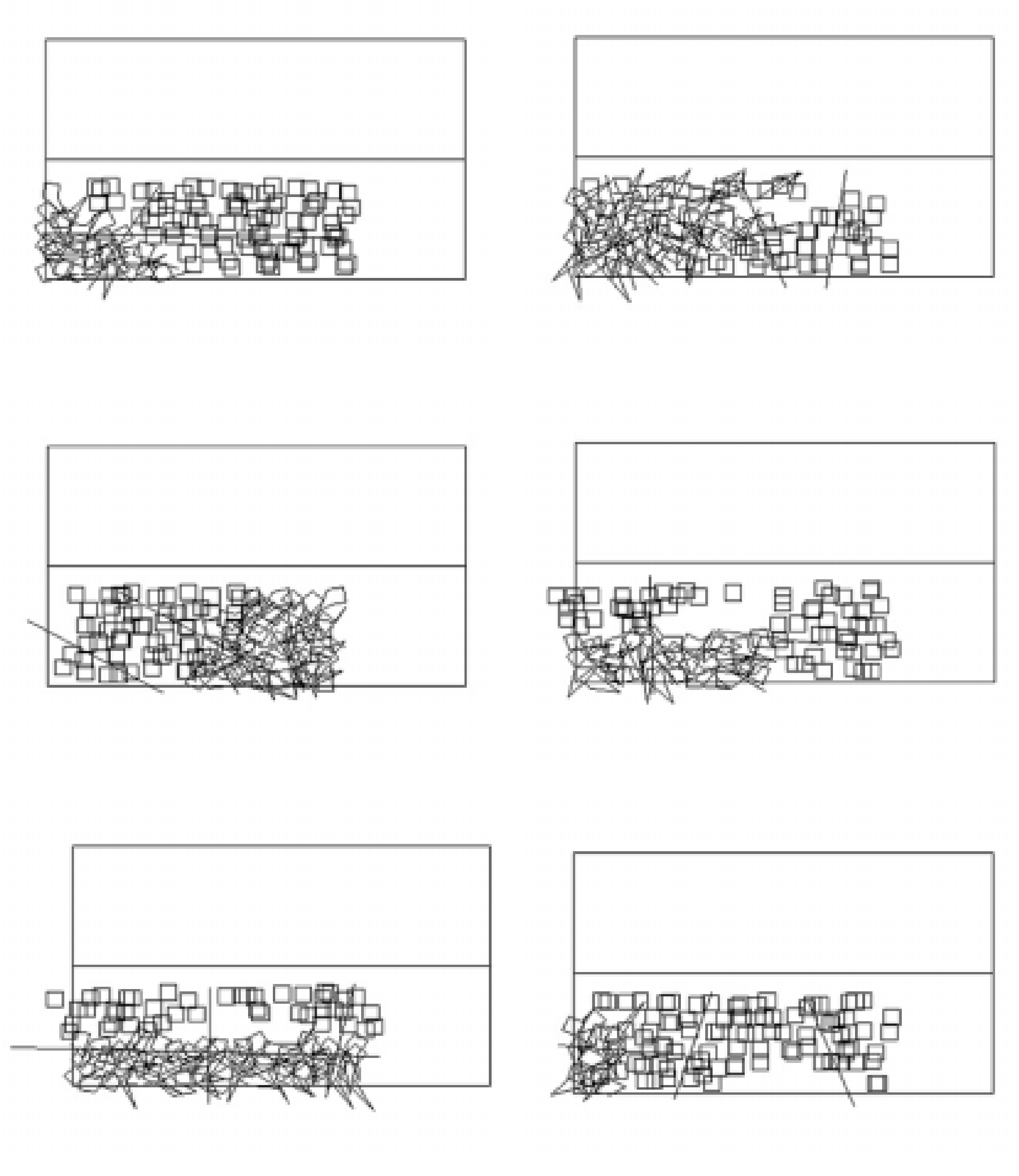


Figure 28. As with his approach to the Spring project, Christopher Mulvey coded predominantly what he already knew within very tight constraints, producing similar predetermined results with very little variation

(Forrest-Pressley & Waller, 1984 p.6). Developments in the field of education and learning have demonstrated that this process can be taught, and that digital media can play a role in such teaching (Perkins & al, (in press), Perkins & al., 1986, Salomon, 1988). Dwyer, Papert, Scardamalia, Bereiter, and others in the field of education theory have all seen the computer as a vehicle for the reform they see as necessary in the education of children. Dwyer believed that the best learning experiences come through learning to master the computer. He used the analogy of "solo flying" from his experience as a flight instructor to describe the successful development of metacognitive skills (Dwyer, 1971). Dwyer saw the computer as offering the possibility of going solo sooner than more traditional educational media.

The most significant and controversial aspect of Seymour Papert's book, Mindstorms, concerning primary mathematics education reform, was his proposal for a change in teaching content as well as teaching methods (Papert, 1980). In his view the constructs of arithmetic are dry and appear unrelated to the child's world. Therefore, he suggests, if you want to teach arithmetic to children, then arithmetic might not be the best way into the ideas behind the subject, something Solomon has termed the Papert Principle (Solomon, 1986). Instead, more exciting possibilities for Constructionist learning and exploration of this seemingly dry subject are offered though digital technology. Papert developed the computer language LOGO as an environment that linked a powerful, structured language with a dynamic graphic system as a means for providing solid support for diversity in a familiar framework. The successful use of LOGO in education was a result both of the qualities of the language and the vision of its author. Papert envisioned LOGO as part of a wider Constructionist approach to education. The basis of this thesis is that a similar approach of using digital media as a Constructionist framework to support the architecture students understanding of the metacognitive skills used in design is now possible. This use of digital media challenges two aspects of architectural education: the way in which design is traditionally taught, and how computer skills are taught and used in the design studio.

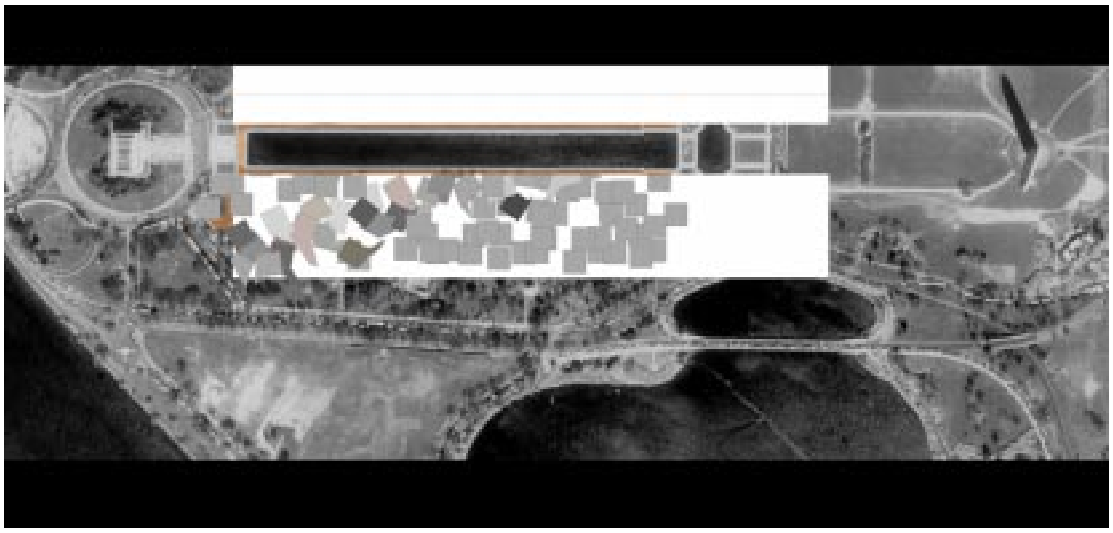
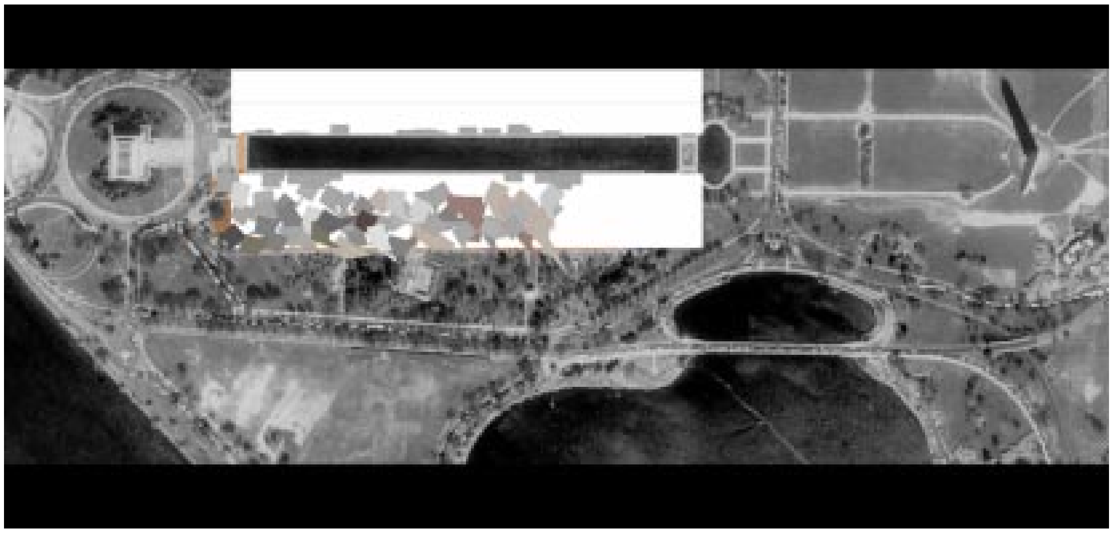



Figure 29. Here Christopher Mulvey places his diagrammatic squares on the site in the hope that they will transform from diagram to architecture. The scale of the drawing is very misleading

Computers in architecture

Computers have been a part of the architectural profession for almost half a century yet it is only in the last decade that they have become commonplace. Today's software has become easier to use and more transparent, and software and hardware are easily affordable. Computers today exist in virtually every office and every school of architecture, and are used primarily for draughting, working drawing production, and presentation. Most commercial software used in the architectural profession has been developed to aid in the production of visualisations and representations, from working drawing to photorealistic imagery. Some effort has been put into developing aids to support the management of information required in design, but despite this, computers are generally seen as aids in the creation of images – the products of architectural design. The computer is used today predominantly to simulate and represent finished designs. It is rarely used to stimulate the designer's imagination and design thinking (Resnick, 1995). Few designers believe the computer can help them to design or help them in the creative process.

The use of computers in the design studio in schools of architecture reflects a similar yet more intense focus on representation. Architecture students use highly sophisticated software with a focus on photorealism and extreme detail to present their ideas. Students often see the hours spent creating such representations as analogous to design thinking. Courses teaching computer skills further reinforce this idea, taught in an atomistic, highly Behaviourist manner away from the context of the design studio where they are needed, often taking as their basis the representation of an existing building that must be modelled in various software. Some schools of architecture teach computer programming in addition to modelling and renderings skills. Often the language used is one for commercial software development such as C++ or Java, and students are expected to learn to instruct a computer to draw graphical elements such as rectangles from first principles (Mitchell *et al.*, 1987). Usually the focus of such courses are the computational constructs, and the student is judged by her mastery of such constructs. These courses are usually taught as a Behaviourist process, the traditional, 'hard' approach of many computer science courses (Turkle & Papert, 1990), that is time consuming and has little in common with the design process. This approach requires a complete predefinition of the problem, broken down into sub problems that can each be solved with relative ease and subsequently combined. Often the sub problems are independently functioning pieces of code, but the product is not tested as a whole until all the parts are completed. The product of such courses is the final working code. This approach to computer programming is anathema to designers more familiar with problem definitions evolving with the design solution. None of these courses use the process of computer programming to teach design understanding and the fundamentals of cognitive strategies for designing.

There still exists in the architectural profession the remnants of an old prejudice of computer programming in design. Architects and students alike often assume that computer programming involves thinking about design from a purely quantitative standpoint. It is possible that this prejudice was formed when computers in the profession were new and their proponents dedicated to their use in quantitative rather than qualitative aspects of the profession (Teicholz, 1968). This use remains a central focus of digital intervention in design research (Purcell, 1988). Little research focused upon more qualitative qualities of the medium (Noll, 1967), and only recently has it become a more popular topic of research (Coyne & Snodgrass, 1993). This use of computers in schools of architecture purely as a means of creating architectural products has formed the basis of misunderstanding in the profession as a whole that computers cannot be used for assisting and fostering the design process.

The Research Problem

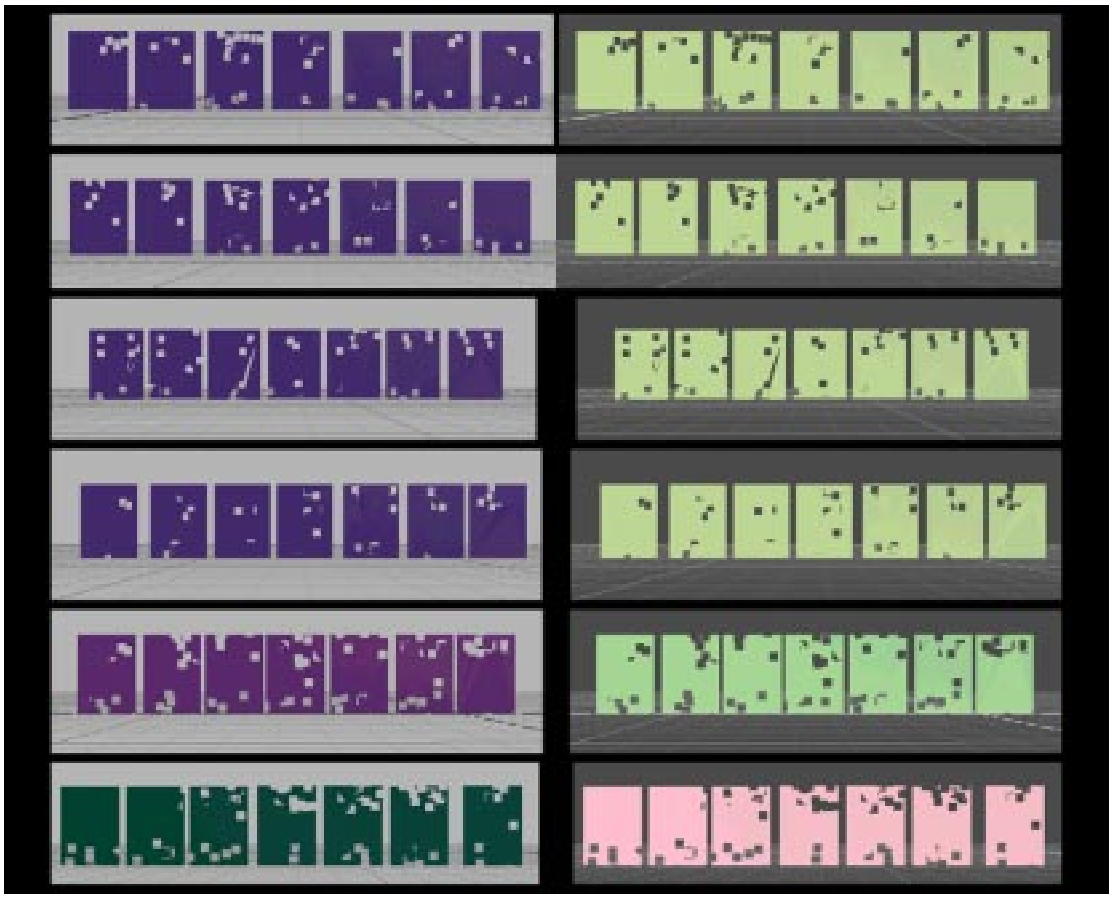


Figure 30. In his concentration on the tool rather than the architecture, Christopher Mulvey carried out a number of exercises ‘to see what would happen if’. Here his code cuts holes in a block, and then slices it

The Difficulties of Architectural Education

From the outside, architectural education exhibits many of the characteristics of a context-aware knowledge building community. Yet the awareness of the mental, physical and social contexts of the architecture student is not a significant factor in their education. The architectural studio system is based on an ancient model of apprenticeship, with an inherently Constructionist assumption of learning through doing. But whereas the model of apprenticeship included the atelier master demonstrating, and thereby continuing his commercial success, the design studio in a school of architecture consists of a teacher telling and students doing. It is, in fact, a demonstration of the highly Behaviourist three step approach to teaching described by Scardamalia: the teacher initiates with a design proposal, the student responds with an attempted solution, and the teacher evaluates in a desk crit or jury and offers more suggestions for improvement (Scardamalia & Bereiter, 1994). In design this initiation is often in the form of practical demonstration, but it is not to be confused with a genuine apprenticeship process of the designer demonstrating through her own design projects. The demonstration in a studio takes place in the context of the student's work, not the professor's. It is possible to continue with this pseudo apprenticeship model because, as Schön points out, architectural design occurs in a virtual world of representation, posing few risks for the novice. Schön saw the architecture student as operating in a low risk environment where she is free to hypothesise, test and evaluate. However, in schools of architecture there are other, much more immediate risks, which are particularly apparent in students who have had little experience of the world outside of education, and which Schön does not acknowledge. The atmosphere of most schools of architecture is highly competitive, and the risks of failing, or not doing as well as your peers, are very high. In such an atmosphere the student is encouraged to see the design brief as the single motivational force behind her work. Discrepancies between her project and those of her peers working on the same brief are dismissed as personal preferences. These risks play a strong part in the actions which architecture students are willing to take in their learning, and can impede any opportunity for true exploration. Schön's model fails to take into account the immediate social and cultural context of the architecture student.

The pressures of the risks involved in public failure are great. As a result, most students, and most architects stick rigidly to the one idea they have at the beginning, with minimal exploration (Darke, 1979, Lawson, 1993, Lloyd & Scott, 1995). They become attached to the physical manifestations of their ideas – the models and drawings – as a result of the high premium placed on this work in the assessment process, and the time involved in its creation. As a result, there is no forum for discussion of themes, underlying factors, and personal issues

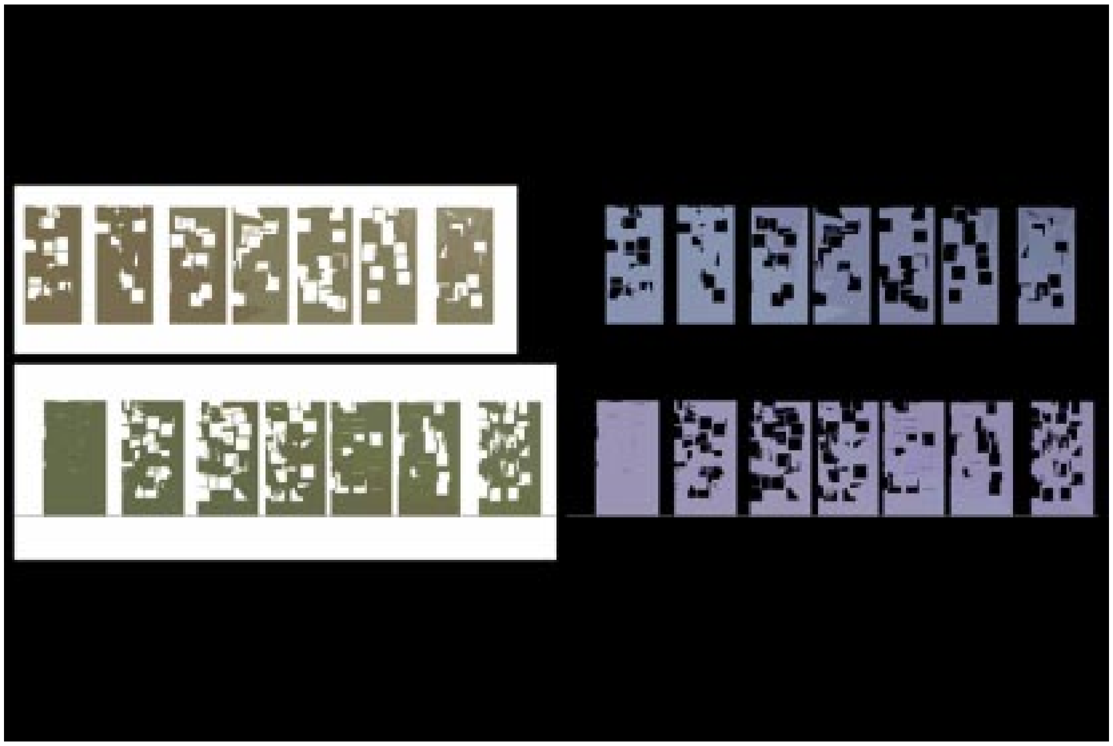


Figure 31. In his concentration on the tool rather than the architecture, Christopher Mulvey carried out a number of exercises 'to see what would happen if'. Here his code cuts holes in a block, and then slices it

that influence and shape the work of each student and architect, and that account for the personal stylistic differences in approach to the same design brief. Without such discussion, and with a predominantly Behaviourist background, students often have difficulty relating their perception of the professor's requirement of objective, 'right' design answers, with the reality of design as a highly personal activity. The novice student has difficulty in understanding that for every design brief they are given, and every individual design decision they make, there are no 'right' or 'wrong' answers, only choices made between ideas with varying degrees of appropriateness.

The Canadian educational researchers, Bereiter and Scardamalia, have studied the extent to which a preconception of knowledge and learning affects the extent to which students are willing to engage in the process of learning (Bereiter & Scardamalia, 1989). It is possible to say that such a preconception of learning as a Behaviourist activity plays a leading role in the extent to which architecture students are willing to participate in the design process of their studios, and therefore in the development of an understanding of designing. If such a mental model inhibits the student from participation, and if in most schools of architecture it is through participation that this Behaviourist to Constructivist shift occurs, it is possible to say that the process of studying architecture may not actually achieve such an understanding in some students.

In architectural education, the preconceived understandings the student arrives with are never addressed. Repeating her experience as a five year old, she learns to perform in the desired manner by producing the desired products, and her lack of understanding of the process of learning required for designing disappears underground, ready to emerge later in life when she faces design situations that are outside of the narrow confines of her academic studio model. Schön describes the reasons behind architecture's uneasy relationship with other academic departments in universities, but he does not acknowledge that the basis of their distrust is also the basic model of education that is the experience of students entering schools of architecture. Artistic endeavour and understanding through doing are used only in primary education, and are usually considered to be child's play. If the model of learning that architecture students arrive with is unacknowledged in architectural education, it is not possible to suggest ways in which the discrepancies between this approach and that expected in the design studio of learning through doing can be overcome (Gardner, 1991).

The social context of the crit might suggest that architectural education forms a model for Socio-Cultural learning. It is clear that the external representations used to develop designs and created for pin-ups and juries are created to form a common basis of conversation. During a studio project,

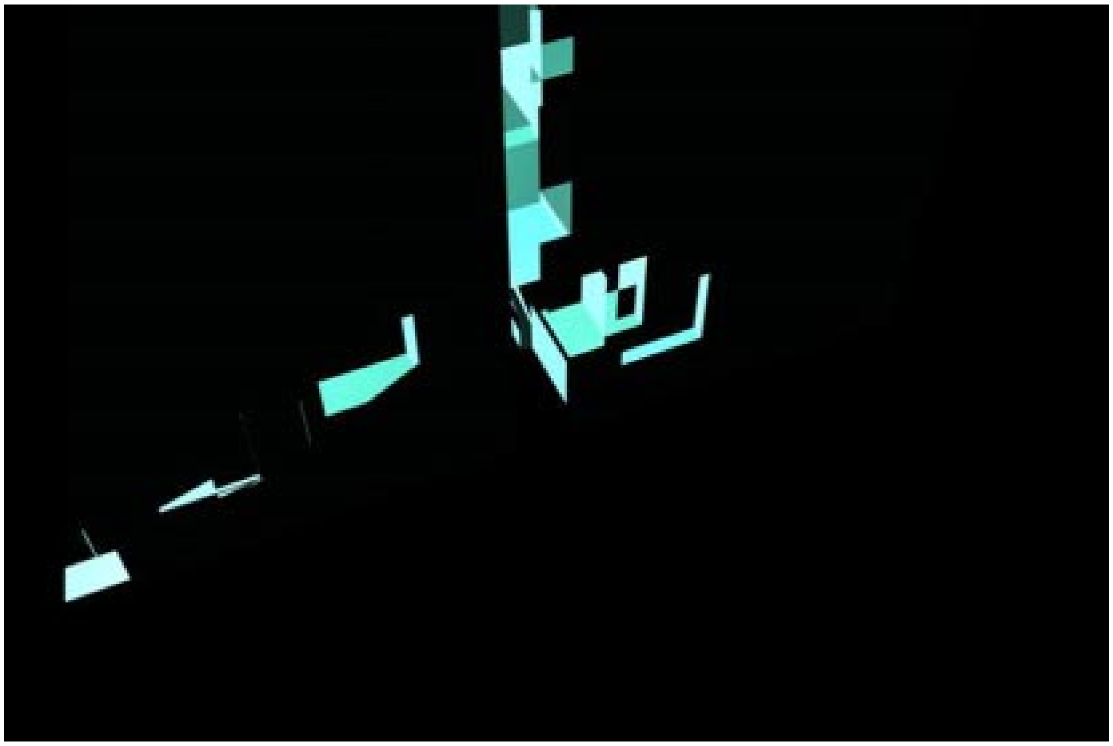


Figure 32. Christopher Mulvey again repeated his problems of the previous semester in moving from two dimensional diagram to architecture. Here he is playing with three dimensional shapes in section.

work is regularly pinned up for group discussion and development. Yet many students learn to dread such occasions, since the opportunity arises for the studio professor to lecture the student in a model much closer to Behaviourist teaching than Constructivist or even Socio-Cultural learning. Students rarely contribute to these discussions of other students' work, believing that such comments would be unwanted and would elicit similarly unwanted comments when it is their turn. In these pin-ups and their more formal counterpart of crits, the jury panel often includes members of the professional community. However, rather than coming to the school in the spirit of collaboration suggested by Scardamalia and Bereiter, these experts come to impart their wisdom and insight to the students, in a Behaviourist model of criticism and reward. It is rare for there to be any form of Scardamalia and Bereiter's knowledge building communities operating within, and linking outside of, schools of architecture.

Architectural education, particularly the teaching of design, faces many problems. Unlike more abstract and factual based subjects, architecture necessarily involves a design process that differs between individuals. Teaching the design process successfully requires that certain skills develop independently in each student. It might possible to teach architecture students about the studies that have looked at the designing activities of different individuals (Akin, 1984, Davies, 1987, Lloyd & Scott, 1994), but doing so only tells them about design for other people. These studies have concentrated on the external application of internal cognitive processes. They do not help the students construct an understanding of what design involves on a personal level, nor help them develop the personal cognitive skills required. Since individual designers conduct themselves differently during their design process, it is possible to say that every student will require a different description of designing as a formula. Teaching the design process is therefore not possible through the application of a formulaic prescribed process. It is instead necessary to engage the student in an activity that might promote and develop their own, personal method of designing. This activity needs to promote and support the cognitive skills that are common elements in every individual's design process. Each student must construct his or her own understanding of what the design process involves. This active, interpretive, constructive process of learning must be used by the student not only to learn design, but be applied to each design project in the future as a professional architect. To learn to design students need to learn to learn in a new paradigm. To learn this new method of learning, the student must throw herself in at the deep end and begin to design without knowing what is required. Schön's paradox appears unavoidable.

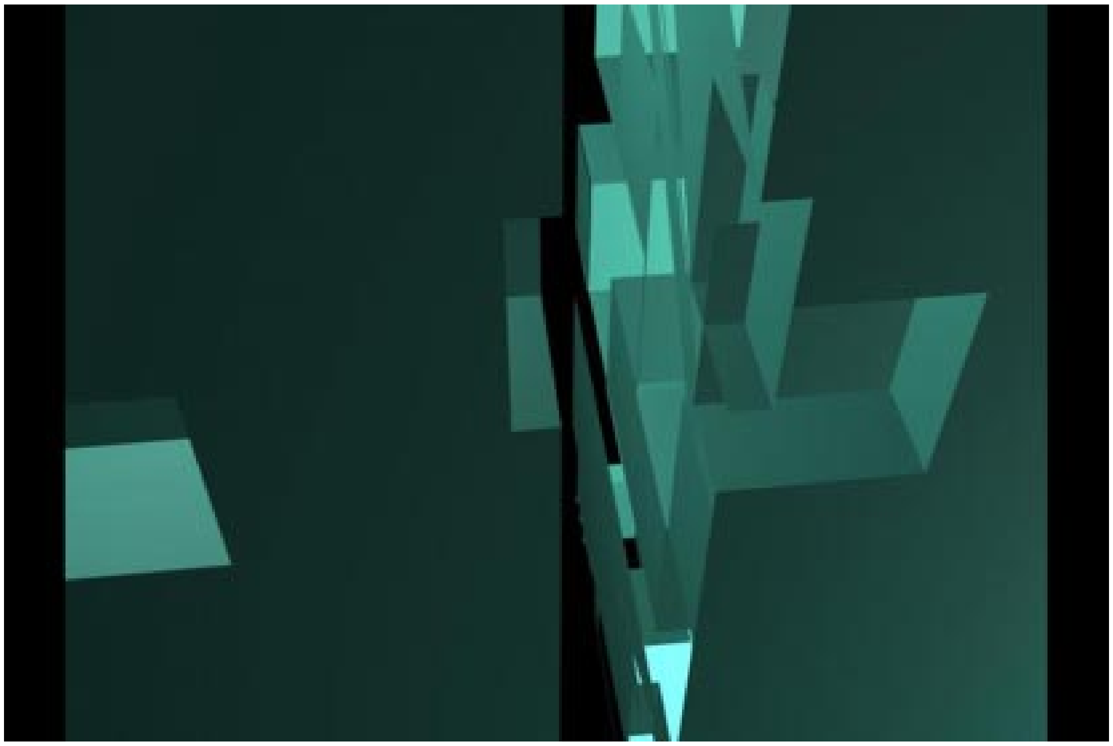


Figure 33. Christopher Mulvey again repeated his problems of the previous semester in moving from two dimensional diagram to architecture. Here he is playing with three dimensional shapes in section.

Architectural design requires the ability to simultaneously use two different languages – graphical and verbal – to explore possibilities of a third, architectural one. This process of exploration is an experimental conversation that evolves slowly and incrementally, and uses external objects as a means by which to think about a design problem. Extensive exploration is necessary, and designers must be willing to discard old ideas in favour of new, more appropriate ones. It requires the metacognitive ability of taking a wide range of knowledge and adapting it appropriately to the demands of the new design situation. It additionally requires the ability to work with and for a wide range of people both within and outside of the profession. These are definable skills studied by researchers and recognisable in mature architects. They are, however, a by-product rather than the main aim of a studio system that concentrates on product over process. Typically, a novice student will struggle with all aspects of the design process. She will have difficulty in formulating the verbal concepts to be expressed, her graphical skills may well be limited, and she may not be familiar with the process of critical analysis required by her at each stage of the iteration, where each representation must be seen in a new light so possibilities may emerge that will help the design process progress. In the studio system, there is little continuity, and rarely any attempt to discuss with the student the metadesign issues she is learning throughout her education: aspects of the design process that are transferable and exist on a higher level than the individual studio projects. Rarely are these techniques, used implicitly by mature architects, explained in any detail. Students concentrate on one project in each studio, and if they become consciously aware of their own design process it is often coincidental, and usually secondary to the particulars of the design project of the studio. Collaboration within the studio is rarely if ever encouraged. Group projects are occasionally set, but cooperation between individuals working on separate projects is discouraged. With so many things to think about it is no wonder that researchers have documented the confusion with which the novice designer views the process of design (Schön, 1984a, Schön, 1984b, Schön, 1985). Schön describes the mystery with which the novice architectural student sees the process of designing (Schön, 1985), a view often perpetuated rather than dispelled by design faculty and practitioners (Coyne & Snodgrass, 1991). He does not suggest positive actions that might overcome the difficulty.

In the absence of any more obvious clue to solving the mystery of the design process, the architecture student quickly learns that the assessment of her work is conducted through the images she produces, and that the higher quality of the images, the more likely she is to succeed. She learns to spend a great deal of time developing beautiful, iconographic imagery and often assumes that the time thus spent is time spent designing. She learns to rely heavily on the input of her studio professor, who will often tell her what the next step in her design should be, thus

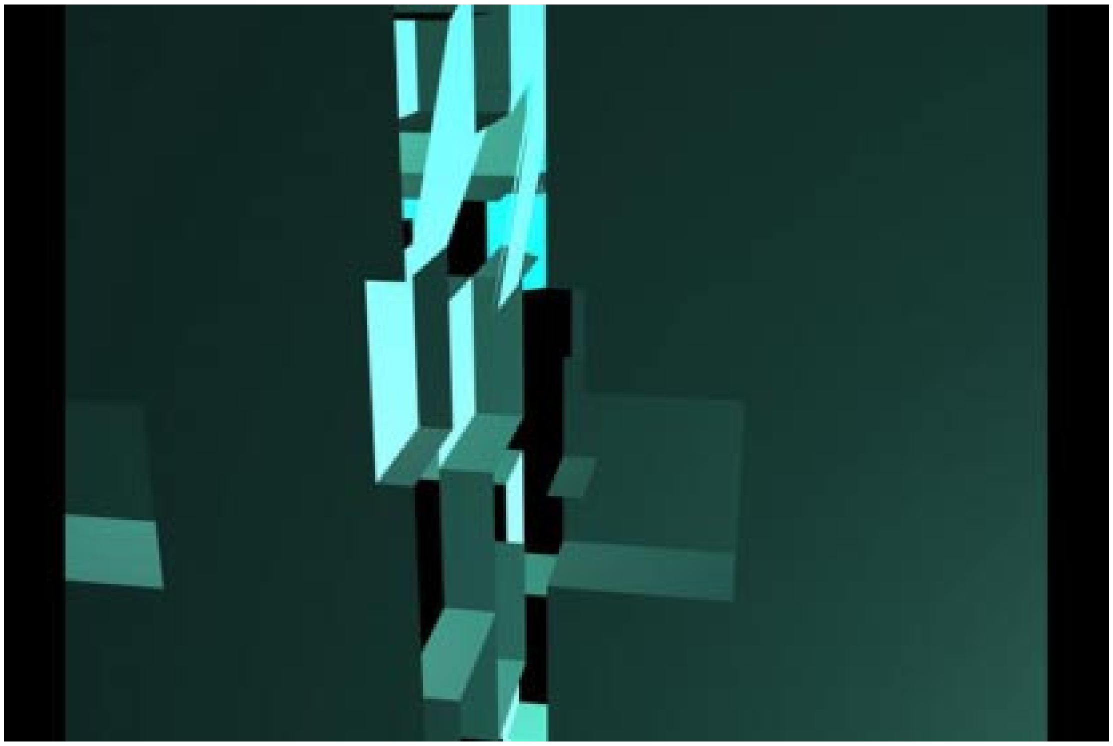


Figure 34. Christopher Mulvey again repeated his problems of the previous semester in moving from two dimensional diagram to architecture. Here he is playing with three dimensional shapes in section.

fulfilling her Behaviourist expectations of the role of the studio professor. Yet these definable skills required by mature architects are techniques that can be explicitly taught to students, to help them learn to "do" design. To do so successfully, architectural education needs to face the Behaviourist-Constructionist shift in learning required, and promote the latter in the context of a Socio-Cultural environment. The challenge to architectural education is to provide a collaborative environment where the risks of failure are reduced or removed, and to directly address the discrepancies between the model of thinking and working the student arrives with on the one hand, and the model with which they are supposed to leave on the other. Such a learning environment needs to address the lack of exploration of different ideas, and the predominantly hierarchical dialogue in traditional, paper based designing, and promote a greater understanding of the underlying factors that influence each individual student in their work. It is assumed that somewhere in their education the student will learn this process: those that don't are labelled as failures. The challenge is to develop tools and methods of using those tools to better assist the student in their learning and understanding of the process of design. In addition, the challenge to architectural educational research is to find new ways of assessing the impact of these tools on the students as they learn to learn in this new paradigm, and thus learn to design. The design studio favours the creation of beautiful imagery over the development of a successful personal design process. Advances in technology open up new possibilities for design pedagogy. Digital media now offer the opportunity to develop this understanding, away from the confusion of the design studio. Schön's paradox can be circumvented.

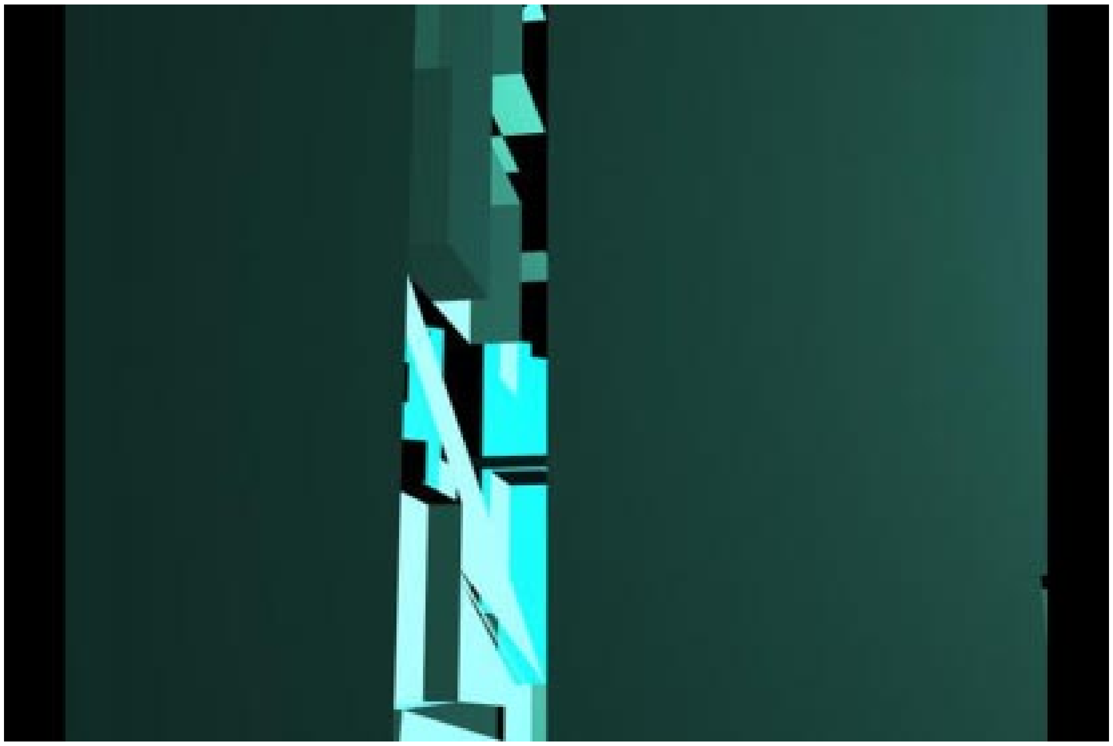


Figure 35. Christopher Mulvey again repeated his problems of the previous semester in moving from two dimensional diagram to architecture. Here he is playing with three dimensional shapes in section.

Hypotheses

It is proposed in this thesis that a separation of process from product in architectural pedagogy is of educational benefit to students. The separation of the architectural meaning of the student's work from the creation of the artifact traditionally used to express it, can give the student a unique chance to develop her own design process, freed from the constraints of studio brief requirements, jury restrictions, and model and drawing production. The limitations of architectural education described above suggest the best method of teaching the process of design may not occur in the academic design studio, as happens in most schools of architecture. The attachment the students develop to the products of that process, and the sense of competition that exists in the studio around them, prohibits true exploration of the nature exhibited by the best designers. It is proposed that Papert's Principle can be applied to the design studio – namely, that the process of design is not best taught in the design studio, and instead can be taught by analogy. Digital environments can be used to promote an understanding of the process of design. To achieve this successfully, they must be situated within the environment they are needed – the design process – not separated from it. Such environments need to be sophisticated enough to be used in the process of design: "attempting to innovate with supportive systems that don't begin to match the sophistication of the human learner should be viewed as a betrayal, not a consequence, of a humanistic approach to education" (Dwyer, 1971 p.100). They need to support but not dictate the efforts of students in understanding the process of design, and to encourage the student to think more deeply about the architectural implications of her work. It is proposed that being exposed to such an environment enables the student to explore architectural ideas more deeply both within the system and eventually when designing through more traditional means. More specifically, it is proposed that computer programming can be used to teach design understanding.

It is further proposed here that a new paradigm for the role of technology in design is required if computers are to be used successfully in the design studio. Much research into computers in designing has focused on their role from one of two paradigms. The first is computer as sophisticated or augmented pencil, particularly in sketch design. The second is computer as library – a potential storage unit for information needed in the design process. It is proposed here that there is a third potential paradigm for digital media in design: the computer as student collaborator. The use of the computer in this model follows its use in Constructionist research, particularly in the development of LOGO, StarLOGO, Impromptu and MusicLOGO[4]. The paradigm for these programs is not the mimicking of existing methods of representation in their relevant domains, nor is it a repository of all information used in those domains. It is instead a learning

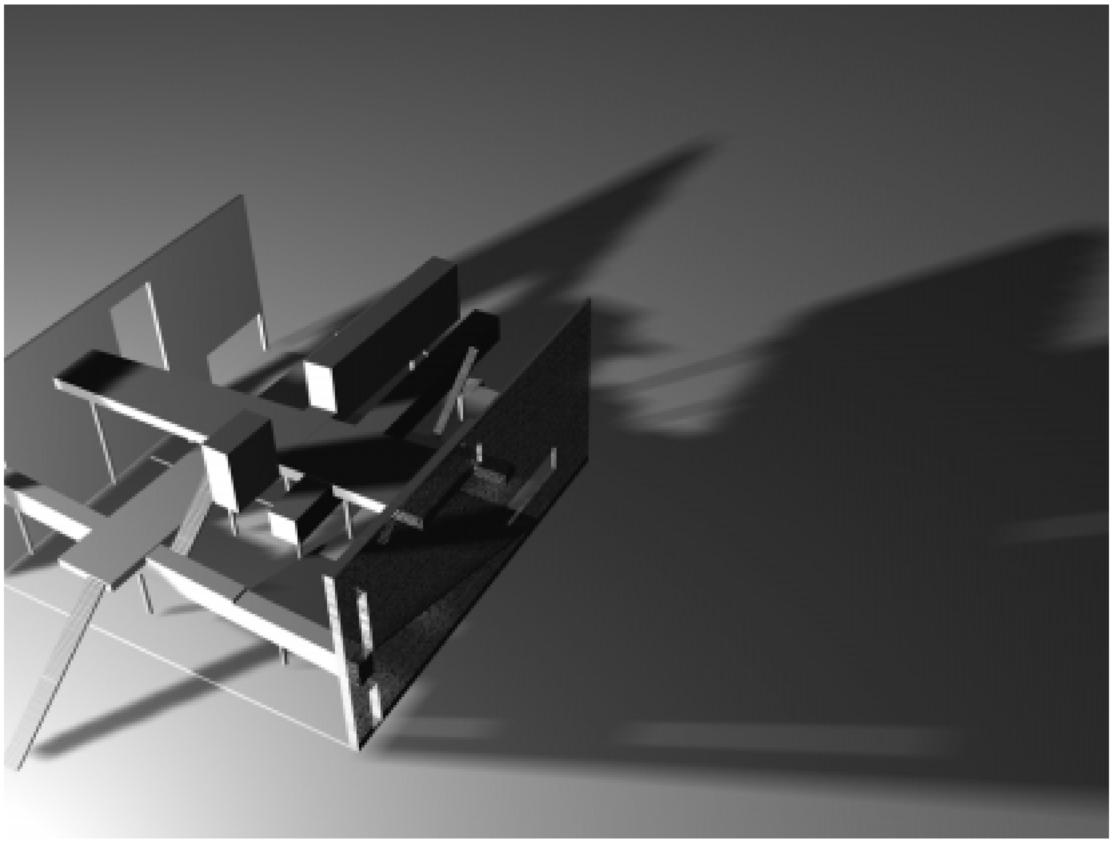


Figure 36. Christopher Mulvey did begin to face the architecture of his project towards the end. Here his code is producing sketch models for his final proposal, including placing columns to take the appropriate loading

paradigm – namely that the best way to learn something is to teach it, to communicate it clearly and unambiguously, to someone – or something – else. It is proposed in this thesis that design is predominantly a continuous process of learning, and that therefore the best way to design is to teach it – or communicate it – to someone or something. Many successful partnerships in design are based on this principle, as is Schön's metaphor of design as conversation (Schön, 1983). The act of communication clarifies to the designer her original intent. Communicating with 'the materials of the situation' of pencil and paper rely on the designer's ability to free associate, see possibilities, and look at things in a new way. Communicating to another designer has further benefits, since alternatives or possibilities not considered can be suggested. Communicating with a computer offers a middle ground between collaboration and individuality. The designer teaches the computer about her current design problem or thinking process. She states what is fixed and cannot be changed, and what she is unsure of, or has yet to decide. In so doing, she clarifies her descriptive design thoughts and ideas yet remains in control. In return, the computer contributes to the process of conversation by depicting those ideas as graphical alternatives that vary according to those aspects that the designer has stated are not fixed. The role of the computer in this process is not predetermined, nor is the information required in the design. The boundary in the division of labour in the design is flexible and depends upon need (Chandrasekaran, 1989). The product of the process is a greater understanding of the design problem. As with working sketches, the graphical output of the process is discarded once decisions have been made. The entire process is a vehicle through which the design is developed. At any stage the process can be integrated into a more traditional, paper based working design process, to be picked up or put down as needed. Ultimately, the final product of the process is the design itself. In addition, the computer provides a neutral vehicle for interpersonal communication whilst maintaining individual authorship.

It is additionally proposed that digital media, specifically computer programming within a CAD environment, offer the potential for collaboration between designer and computer that promotes the designer's communicative ability, whilst maintaining individual control. Development of communication skills aids the ability of working with others in design. It also permits greater exploration of the design idea if the designer is better able to explain her ideas to herself. It is proposed that the necessary limitations of communication with a computer actually promote rather than inhibit the development of this communication skill.

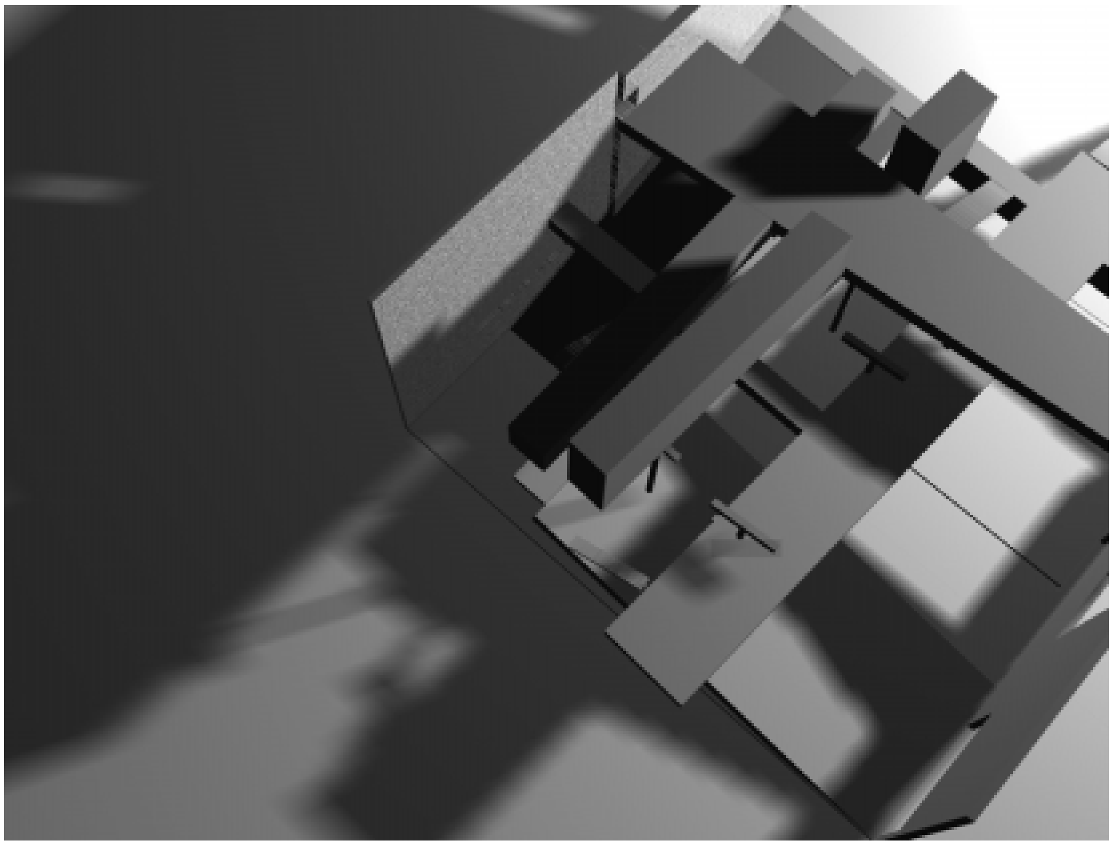


Figure 37. Aerial view of Christopher Mulvey's final model for the memorial for Martin Luther King.
Chris produced many images of the same model, reflecting an obsession with object rather than process.

Finally, it is proposed in this thesis that it is possible to use computers as a part of the creative process if designers learn to use them in a new way. In architecture, as in other disciplines, computers can provide a context for the development of ‘soft’, concrete thinking about a design problem. They can be used not only to represent abstract notions about the design process in real terms, but also to stimulate the imagination of the designer and represent concrete ideas in an abstract manner. They can also make the imagination concrete. “The computer is an expressive medium that different people can make their own in their own way” (Harel & Papert, 1990 p.3). “The computer stands betwixt and between the world of formal systems and physical things; it has the ability to make the abstract concrete. In the simplest case, an object moving on a computer screen might be defined by the most formal of rules and so be like a construct in pure mathematics; but at the same time it is visible, almost tangible, and allows a sense of direct manipulation” (Perkins, 1992 p.61). This thesis describes a course taught at MIT by the author that teaches students of architecture to use the computer as a tool to think with in the process of design.

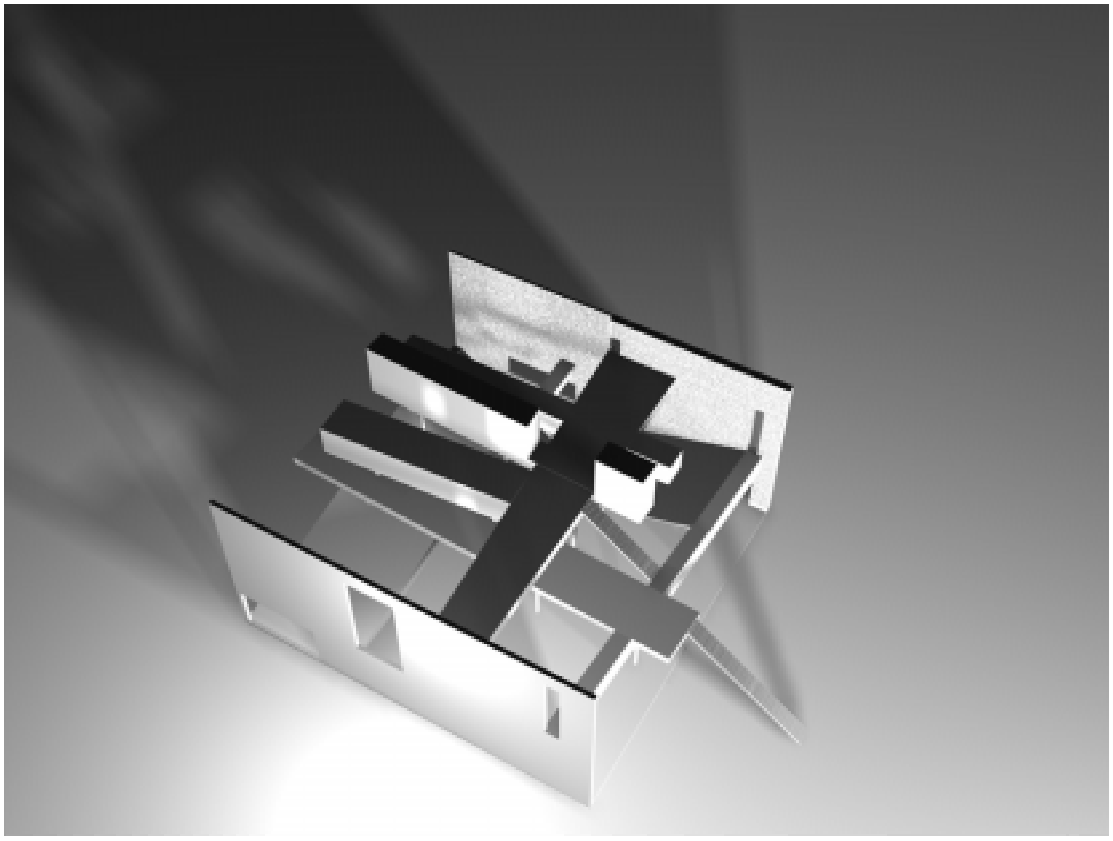


Figure 38. Aerial view of Christopher Mulvey's final model for the memorial for Martin Luther King.
Chris produced many images of the same model, reflecting an obsession with object rather than process.

Need for the Research

It has long been a complaint of the architectural profession that architectural education does not meet the requirements of the modern architect's practice. Practitioners complain that students need to be retrained when they enter the office. This approach has been tolerated through tradition – the apprenticeship model is extended from school into the workplace. Although this might seem ideal in educational terms, in practice there is less and less money in the profession to support such a model. Increasingly, architects are beginning to question the efficacy of the education system they themselves undertook, in the light of the demands of the modern office. This questioning includes the uses of digital media in practice. Increasingly it is practice and not academia that is at the forefront of advances in the use of technology, particularly in the integration of computational tools with other, more traditional means. Architecture firms such as Foster & Partners, or the office of Frank Gehry, are leading the field in the integration of technology in the design process, far preceeding research and techniques taught in the architectural academic world. Teaching students to use a computer through separate, Behaviourist courses away from the design studio is no longer enough. Introducing computers to studio settings is a start, but what is required is a method of teaching the student to use the computer as a tool to think with whilst she is designing. That this process also offers the chance to teach the student about the design process increases the significance of such a method within the architectural education system. However, as with all educational interventions, there is a need for the demonstration of the efficacy of this process. There have been few other documented studies of architectural students or the architectural education process (AA, 1967, Carver, 1964, Crowe & Hurtt, 1986, Frederickson, 1991, Knowles, 1993, Ledewitz, 1982, Schön, 1984a, Wingler, 1978), of digital interventions in architectural education (Aburawi, 1987, Bustinza Esparta, 1996, Prestamo, 1990), and fewer still of the effects of interventions in that process (Grossman, 1990). Very little information is available on suitable methodology for research into assessing the impact of architectural educational interventions (Grossman, 1990). These methods need to be developed if architectural education is to meet the needs of the twenty first century and change its approach to design and technology.

The possibility that aspects of digital media might support the process of designing and might aid in the development of a designer's creative process without dictating a particular methodology, has yet to be fully explored. Commercially available digital aids to design have been developed, but most support only the creation of the products of design – final, formal representations, not the working products of the process. Research into digital aids to process have concentrated on the creation of the working products – the sketches – but not on their purpose as 'objects to think with' in a Constructionist learning paradigm. There is a concentration in these studies upon the creation of marks on paper, the form of the marks, and their meaning in design (Do *et al.*, 1999). Very little is known about the effect of these tools on their intended users. Effective digital aids to the process of designing must support the Constructionist learning aspects of designing, where nothing is predetermined and the designer is in control of the decision making process. The course Digitally Mediated Design 1 described in this thesis, uses computer programming within a CAD framework as the Constructionist vehicle for development and understanding of the design process.

Solution to the problem

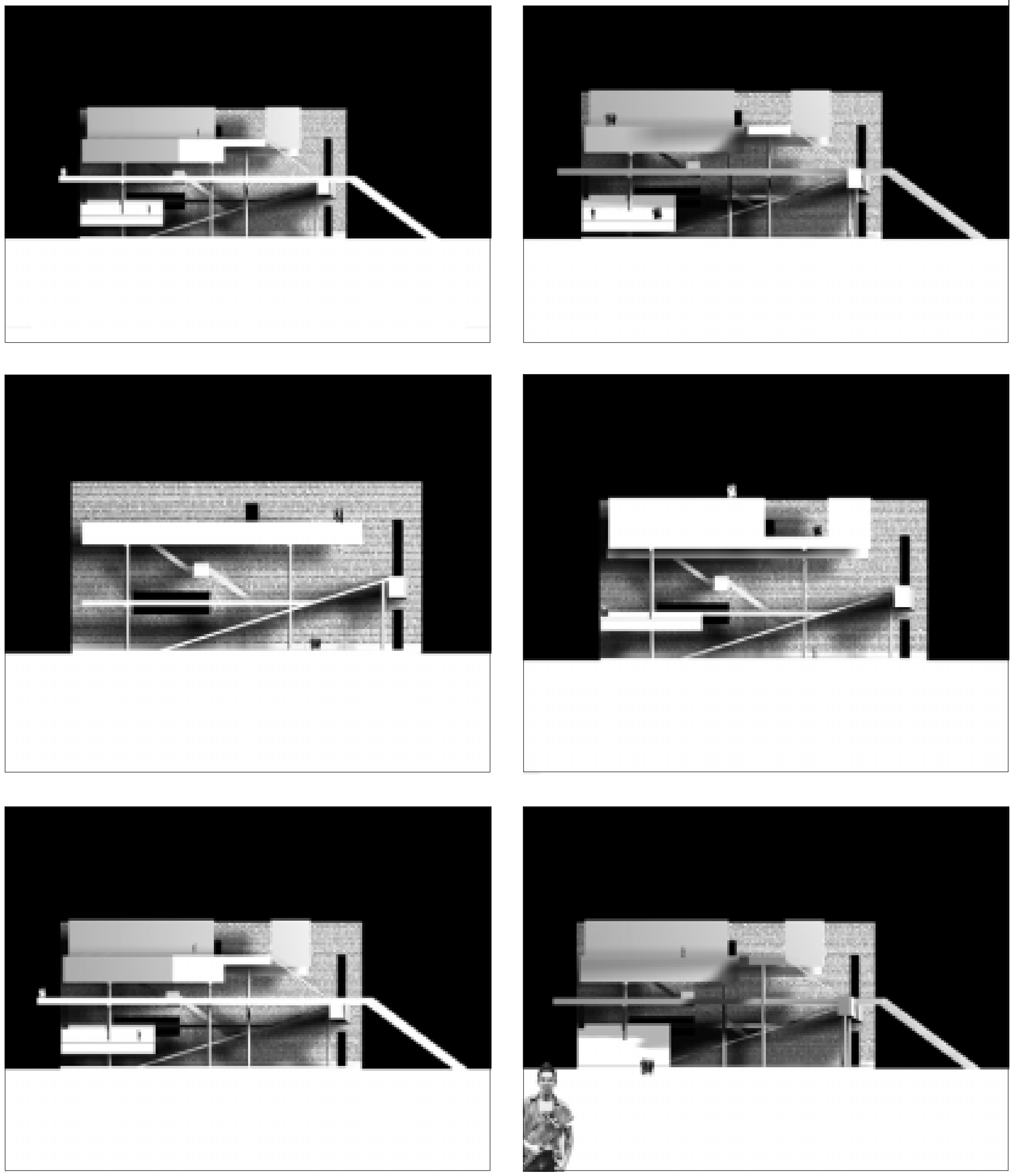


Figure 39. Sections through Christopher Mulvey's final model for the memorial for Martin Luther King. Chris produced many images of the same model, reflecting an obsession with object rather than process.

Introduction

To meet the requirements of the modern office, where use of the computer in the final stages only is no longer enough, a shift is required in the application of computers in design, away from the products of design, and towards supporting the process of designing. Practices are increasingly aware of the significance of hiring architects able to demonstrate using computers as a part of their design process and thinking. Current uses of computers in design permit but do not explicitly encourage the necessarily softer, tinkering style in their use, since they permit non-linear changes to be made. Yet as with Papert's example of word processing, the mere possibility of their use in this manner does not explicitly determine it. The shift in application away from products and towards process must be accompanied by a shift in perception of what computers are for. Designers need to rethink the computer's role in their design process, and must be encouraged to exploit more successfully the computer's support for tinkering. In doing so, the working products of the process, the objects used to think with, may change. So too may the process of their development. However, the final products, representations of an architectural idea and ultimately a successful final building, remain the ultimate goal of the process.

Background Theory to the Course: Developing New Ways of Design Communication

The course described here is both an environment and a pedagogical approach for architecture that focuses on the process, rather than the product, of design as a Constructionist activity, using current technology and methods of teaching developed by education researchers. It promotes an understanding in students of the role of exploration through the generation of design alternatives and permits a sense of detachment from the resultant product, in a manner that supports true design exploration. This exploration examines the appropriateness of interim solutions on a journey to find the most successful design product. The emphasis of such an explorative process is highly academic – it concentrates on process in an effort to teach the cognitive skills required for successful development of product. It uses the development of a generative system as a vehicle for the completion of this aim. It offers a supplement to the studio system by teaching the design process through the analogy between developing a computer program and designing. It is the integration of software tool and pedagogical approach that permits such an analogy. Turkle and Papert make a connection between a preference for using objects to think with in a concrete style of reasoning, attributes cultivated in architectural education, with a softer, tinkering style of computer programming. Here, a connection is made in the opposite direction: a tinkering style of programming, where the programmer "makes a simple working program and

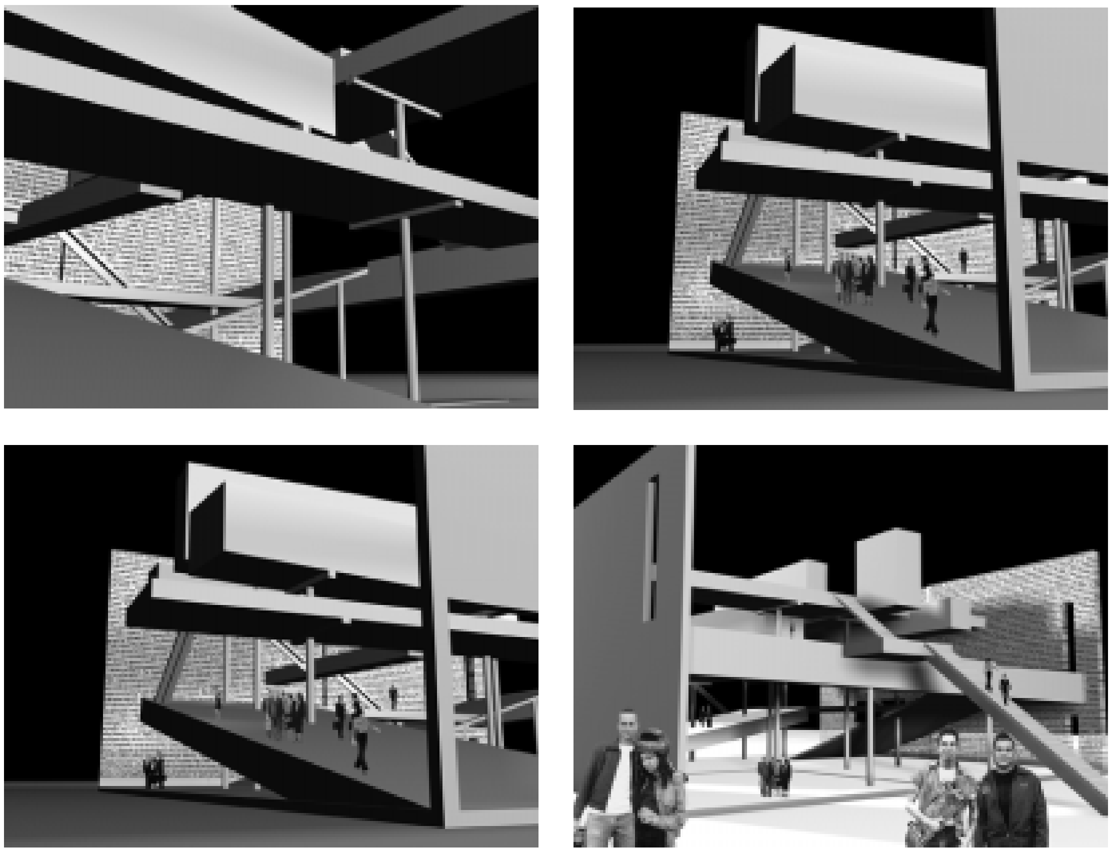


Figure 40. Exterior views of Christopher Mulvey's final model for the memorial for Martin Luther King. Chris produced many images of the same model, reflecting an obsession with object rather than process.

shapes it gradually by successive modifications" (Turkle & Papert, 1990 p.356) offers an alternative, analogous example to the student of the process of designing. In this context there is a parallel between programming and sketching in design. The significant difference between this style of programming and architectural designing, is that the computer is able to instantly verify the results of the code writing, whilst architectural judgments rely on the designer's knowledge and expertise, and cannot be verified. Yet the process of successive modification and reflection is an important lesson for the novice architect. There is no attempt to dictate the design process to the students. Instead, there is an acceptance of the individual differences in process for every designer, and the focus of the course is teaching to support a 'difference in similarity' framework.

This process, involving a graphical computer programming environment, is used in the Digitally Mediated Design 1 class at Massachusetts Institute of Technology in the Department of Architecture. The use of the computer in the course is similar to the use of the computer in Resnick's classes with children using his software StarLOGO (Resnick, 1995). The computer is on the one hand entirely necessary, since the program and therefore the course cannot run without it, but on the other hand it is entirely incidental to the process. Resnick's purpose was not to teach children how to use a computer, but to stimulate their imaginations when thinking about, and understanding the concepts of, decentralised processes such as flocking birds or traffic jams. The computer is merely the vehicle by which the students are learning. The process taught to the students in this course uses the computer in a similar manner, but the subject matter is architectural design. In its purest form, the actual output of the computer – what it draws on screen during the process – is irrelevant. So too is the final working code produced. It is the actual moment of communication, the point at which the designer expresses her ideas to the computer during the process, that is most relevant. The outcome is simply used to confirm the original intent and to form the basis of further thought. This follows a similar approach to Resnick's by the music theorist Jean Bamberger in her development of software to permit the exploration of musical structure. "Two very basic principles have guided the design of both MusicLOGO and Impromptu. First, computers should be used only to do things we cannot do better in some other way. Second ... [a] computer environment is valuable to the degree it causes its developers to rethink the structure of the relevant domain" (Bamberger, 1996 p.41). The focus of the course is on the process of designing rather than on the computational constructs involved.

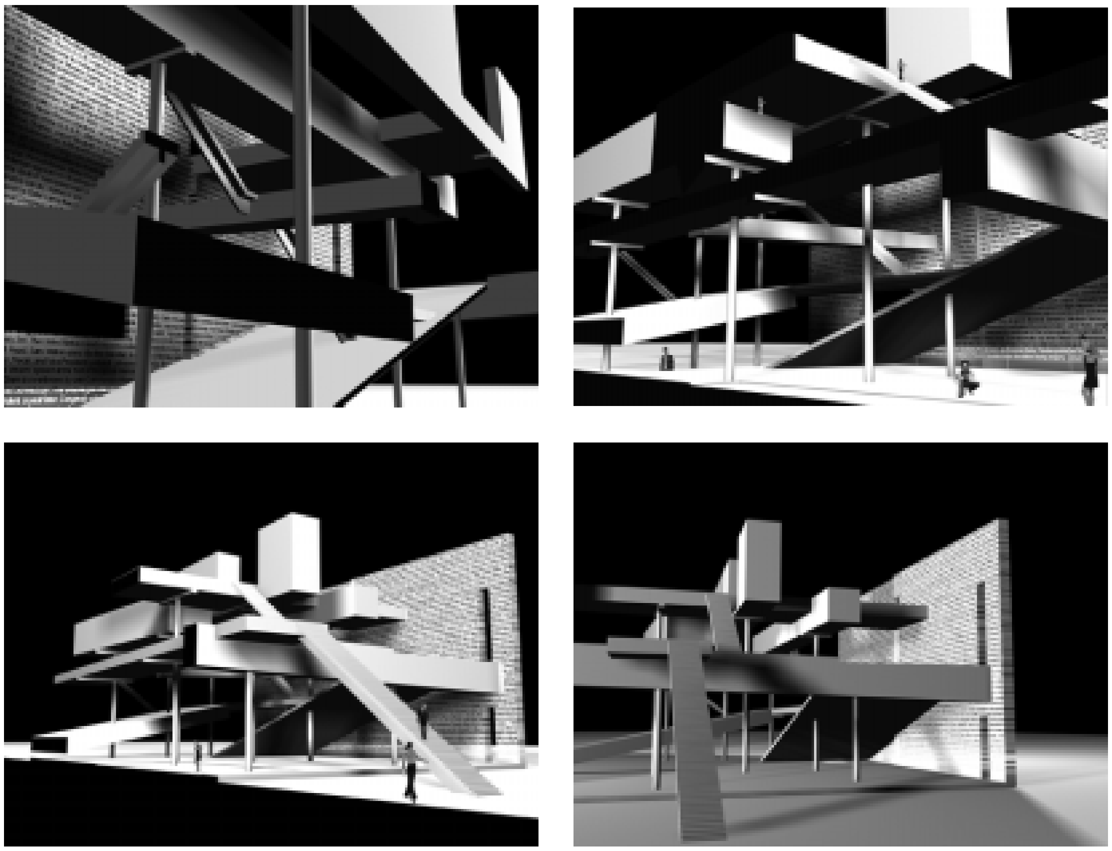


Figure 41. Exterior views of Christopher Mulvey's final model for the memorial for Martin Luther King. Chris produced many images of the same model, reflecting an obsession with object rather than process.

The motivational starting point of the process is the computer itself, recognised in education fields as playing a key role in the motivation of learning. The use of digital media in the course is simultaneously inconsequential, and yet significant: recent developments in digital media permit such a course, but the computer is of secondary importance to the understanding of design thinking.

The process of communicating with a computer in a manner that supports the design process involves the designer learning a computer programming language. This language needs to be sophisticated, and the graphical output more so, if it is to support the development of design ideas more actively than a paper based sketch book. A digital environment to support Constructionist learning must have a recognisable language base and an instant graphical output that can be related to the images a student produces in the course of her designing. For the software to form the bridge between verbal idea and graphical expression it is critical both that the written code is recognisable and builds upon the architecture student's existing knowledge of CAD software, and that the output is instantly graphical. The framework needs to support Schön's reflection-in-action process of designing. If it is to be used by novices, it needs to be adaptable to support the learner growing in expertise. The computer is used in its capacity of existing on the boundary between real and virtual worlds, to test and confirm the writing of a design algorithm through the creation of three dimensional, virtual form. In so doing, it makes concrete the conceptual ideas of the designer. The computer is capable of achieving this very fast, so results are seen quickly. Bruner defined three important aspects to encourage the exploration of alternatives in learning: activation, maintenance, and direction. The exploration process needs to have something to get it started, something to maintain it, and something to keep it from being a random process of trial and error. “Knowing is a process, not a product” (Bruner, 1966). In learning to use computers as a basis for designing, the direction is dictated by the design project, and the activation by the desire to learn. The maintenance, perhaps the most critical in this process that is difficult to learn, is the computer's inherent ability to repeat an action very quickly and as many times as desired. It occurs in a programming environment where the outcome of the algorithm is graphical, and the images produced are instantaneous, rewarding and exciting. Since the author of the algorithm is a designer, and the algorithm expresses thoughts about some aspect of the design process, the instant output is used to confirm the design thoughts, and form the basis of future design decisions, of the designer. If each time the action is repeated there is a possibility for variation, so that the resultant graphical product is slightly different, then the designer and novice programmer is encouraged to continue. The novice designer is assisted in the process of learning to design through the

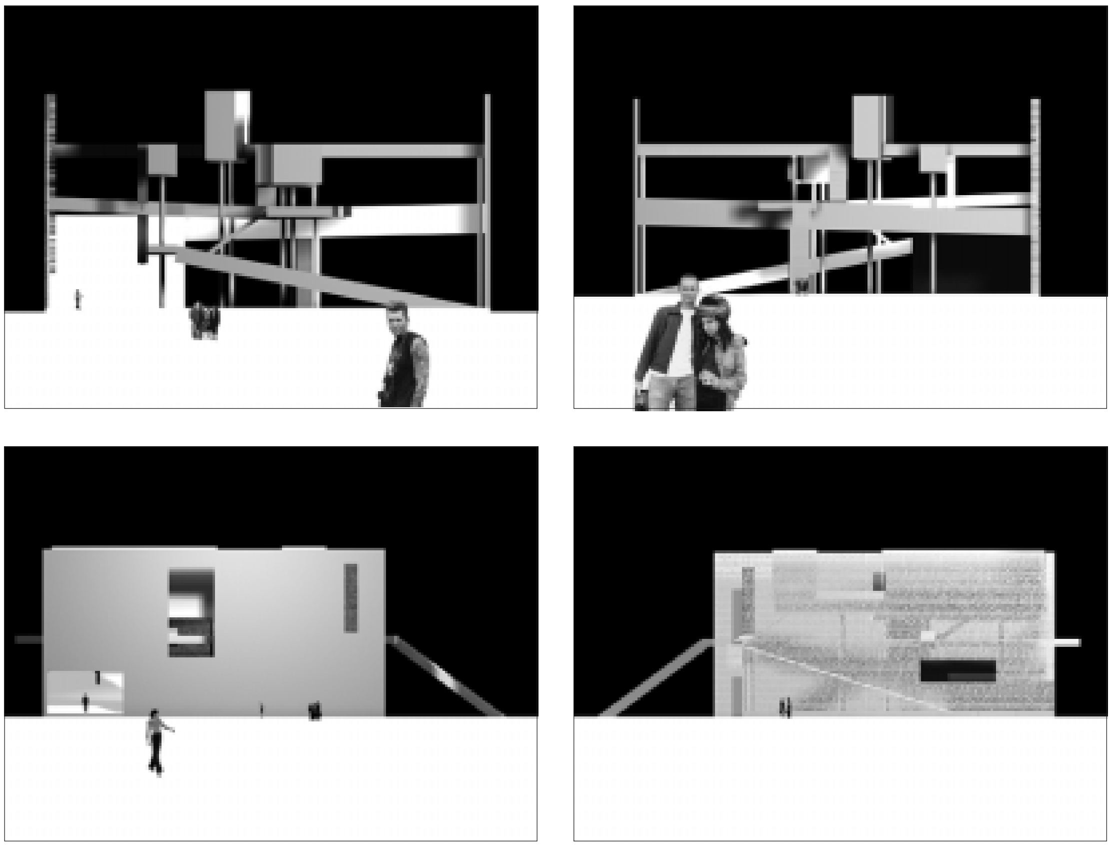


Figure 42. Exterior elevations of Christopher Mulvey's final model for the memorial for Martin Luther King. Chris produced many images of the same model, reflecting an obsession with object rather than process.

"scaffolding" (Vygotsky, 1978) provided by the more structured context than the infinite possibilities presented by paper and pencil. The limitations of the process taught in the course can act as temporary support mechanisms for novice students to learn to design. In this manner the computer is beginning to enter into the design process and actively become the 'back talk' of the conversation. This is digital sketching.

The action repeated by the computer, a precisely defined set of instructions, or algorithm, is an effective procedure, clear and unambiguous, applicable to a problem or set of problems, with a definite conclusion or stopping point. On the surface such a description would appear to better fit the 'hard' abstract processes of formal problem solving, and be alien to the 'softer' approaches of design. The processes taught to the students in the course do, however, demonstrate to them the benefits of using a computer to help them design. Note that it is their own code that is of benefit, not the code of others. Attempts in the past to predefine a set of algorithms for designing remain flawed. Since design is a constant process of inquiry the designer does not hold all relevant knowledge for all future projects. If it is not possible to predetermine what design knowledge will be required, it is impossible to predefine standard, universal algorithms applicable to every design situation (Winograd & Flores, 1986).

Yet shift the focus of attention away from the final written code and turn instead to the moment of its creation – the basis of Constructionist software use. Consider carefully what is involved in the writing of an algorithm. The instructions must be precise, unambiguous, and exact. There must be a clear starting point, and the algorithm must indicate when the process is complete. As any teacher will confirm, writing an exact set of instructions to achieve a particular task is only possible if the teacher herself fully understands both how to achieve the task, and also the consequences of achieving it in one particular manner rather than another. In addition, the process of formalising the knowledge held by the teacher itself clarifies her understanding. Thus to write a computer program, the software developer must have a strong understanding of the domain in which her product is created. The process of writing the algorithms further clarifies the developer's understanding. The computer becomes a student-collaborator. In designing, the designer holds the skills needed to seek out and use appropriate information at the time it is required. The role of the algorithm in this case is to help her express and explore that information – it is the process of writing the algorithm that is of use, not the running of someone else's code. Running the code here is used only to verify the design idea, and to produce variable graphical results that can be viewed with detachment and ambiguity, and eventually form the basis for the next emergent rule, in the development of a true dialogue of designing.

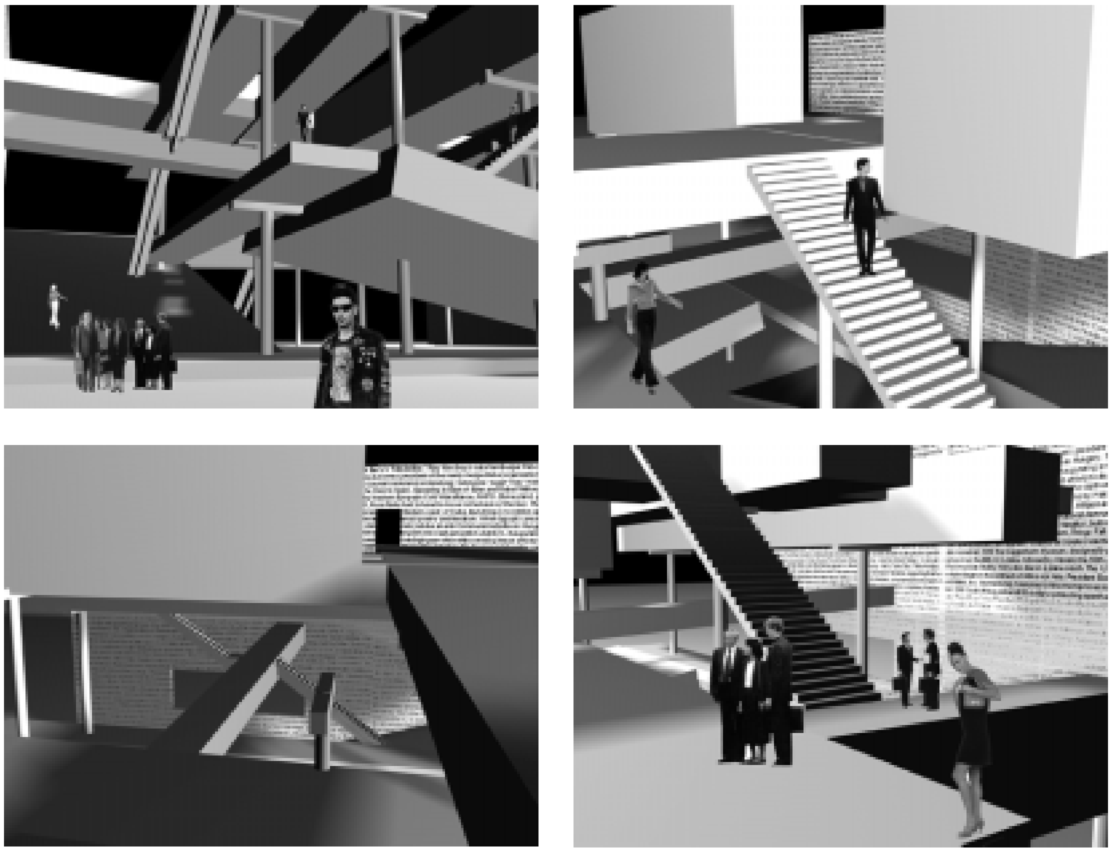


Figure 43. Interior views of Christopher Mulvey's final model for the memorial for Martin Luther King. Chris produced many images of the same model, reflecting an obsession with object rather than process.

Schön’s conversational description of the design process relies on the ability to reinterpret the graphical results of a design idea in a new manner. The sketch must be ambiguous for this to occur. However, algorithms are by definition unambiguous lists of sequential actions. Since computers run entirely on these unambiguous, predefined statements, it is perhaps difficult to imagine how a computer might be used in an ambiguous manner to become a successful partner in the design conversation. CAD drawings are necessarily precise, and the hard lines produced in these environments cannot simulate the ambiguity to be found in a soft pencil sketch (Stiny, 1990). Ambiguity can be simulated – software such as PhotoShop successfully masks the underlying unambiguous nature of its algorithmic structure; using a stylus and a pressure sensitive pad permits freehand sketching with exactly the same results as using a pencil and paper. Of course, Photoshop also permits far more than a pencil and paper; it is possible to change, modify, or edit marks with ease. PhotoShop permits the “dirty marks on paper” (Sutherland, 1975) to enter the exact realm of the computer. However, PhotoShop stores information about “lines” as pixels on the screen, which is useless for a CAD environment, where a line is required to have integrity.

It is this necessary integrity, where a line remains a line to the computer – and not a series of pixels – when it is moved, that makes the possibility of ambiguity, so necessary in early sketch design conversations, seem so remote in a CAD environment. To preserve integrity, the computer stores the end points of the line and connects them on the screen. Everything is precise and clearly defined. Yet despite this precision, it is possible to permit the appropriate levels of ambiguity required in design conversations to enter the CAD drawing if the role of ambiguity in design is more precisely defined, and a difference is made between the undesirable ambiguity of intention, and ambiguity of interpretation – the skill necessary for successful sketch design. The process of developing an idea, having a ‘conversation with the materials of the situation’ relies on the ability of the designer to interpret the graphical result of the last design decision in a new and innovative way, to see things in the image that had not previously been intended. This is a considerably skill, not least because it requires the designer to ignore her intentions of moments before and see something new. Ambiguity in graphical language is important here, since somewhat vague looking and ambiguous lines in a pencil sketch aid in permitting this new interpretation. Precise CAD lines created in the traditional manner do not permit this level of ambiguity. Each line or object in the CAD drawing has been created by the designer and remains as an exact and unambiguous reminder of her original intent.

In the process of writing code to generate a graphical output, it is the unambiguous nature of the code writing that aids in the clarification of the designer's ideas. A weak student can often confuse ambiguity of interpretation with ambiguity of intent – scruffy looking sketch lines could actually be masking her lack of understanding of what it is she is intending to design. Writing computer code does not permit this, and instead forces the designer to be precise and unambiguous about her intentions. The graphical result of the process of code writing, however, permits ambiguity of interpretation despite the unambiguous nature of the graphical language used. This ambiguity of interpretation is possible for two reasons. Firstly, unlike CAD drawings created in the traditional manner, each line in the drawing is a product of the running of the code, and has not been created by the designer. This detachment from the resultant image encourages a freer interpretation. Secondly, each time the code is run the output is slightly different, within limits set by the designer. When a number of different factors are permitted to vary, unexpected results will emerge, producing images the designer would not have created alone. This further supports ambiguity of interpretation. As with Constructionist software, the computer is being used not to simulate, but to stimulate the creative processes of the designer's mind (Resnick, 1995).

The algorithms that are written to produce these variable CAD drawings with their possibility for surprise in the design setting are called generative systems. Generative systems have existed in architectural design for a long time (Mitchell, 1975). A generative system can be loosely defined as a set of rules that act in some manner upon a set of elements to produce a set of potential solutions (Henke, 1990, Krishnamurti & Giraud, 1986). A digital generative system can be seen as an engine for the development and testing of a set of rules and a set of elements. Just as the development of a generative system is an engine for the testing of design ideas, so too is the computer an engine for the running of such a system. This pedagogical possibility has been created by technology – although generative systems have existing long before today's technology (Mitchell, 1975), the possibility of using it as a tool for teaching has been created by improvements in digital media. Computers are adept at repetitive iterations and predefined variations, both important in the successful implementation of a generative system.

The process of creation of a generative system in architectural software involves the definition of a set of ideas with common elements. The common elements are the rules of the system, and the graphical primitives of the software. The ideas are the interpretation of those primitives in the mind of the designer, into architectural artifacts. A paper based drawing can represent an architectural idea, although the paper itself has no 'understanding' of the marks upon it. In a generative system for designing, it is not always necessary to explicitly include the ideas as rules in the system or to have the system include some ‘understanding’ of the nature of an architectural rule, object, or element. It is more important for the designer or student to describe her ideas as a set of graphical objects that can be both interpreted by her and acted upon by the rules. The system need only understand the graphical objects as primitives. The output of the rules are represented on the screen in the graphical language of the software. These external representations form the basis for internal and external discussions, supporting an environment of collaboration and Socio-Cultural learning. These conversations in their turn play a part in the development of more rules, and more primitives. “The experimental process is event rather than goal driven” (Suchman, 1987 p.186).

It is the development of the rules and elements that represents a pedagogical possibility in a school of architecture. If design is seen as the generation of alternative ideas from which a suitable solution may be found (Guarra, 1974, Lloyd & Scott, 1995, Stiny, 1979) then the development of a generative system can be seen as both analogous to, and promoting, the design process. It is the development of such a system, rather than the system itself, that is of interest. The final system represents, for any person other than its author, a limitation on the process of designing, a criticism prohibiting the development of commercially successful generative systems. Universal rules for design cannot be predetermined.

In the development of a generative system as a part of the design process, in learning and applying previously alien computer programming techniques, the student gradually comes to view the computer in the framework of a new paradigm – not as a digital pencil, nor as a repository of well organised information. She learns to manipulate the computer as a stimulating but subservient pupil in a reflective design conversation. She learns to view the computer as a part of an active and creative process of her own making. She subconsciously becomes a teacher in a teacher-pupil relationship between herself and the computer. She teaches the computer – and clarifies for herself – her design thinking and personal design process. She herself learns through the creation of algorithms that express design intent the importance of using graphical objects to think with in design. She has a conversation with the digital environment that assists in the development of her design ideas. This conversation is similar to that described by Schön in his discussion of the process of designing - it is “a conversation with the materials of the situation” (Schön, 1983). Yet unlike the hierarchical, Socratic nature of the pencil and paper based conversation, when the sketchpad is transferred from paper to a digital generative system, the conversation becomes less hierarchical and the computer becomes a more equal contributor. As she develops her generative system, the student is constructing her understanding of the role of exploration in design. The concentration upon a single design project in learning to design, as occurs in the academic studio, encourages the belief that designing involves the development of a single idea, rather than the exploration of many ideas. In contrast, a concentration upon a set of rules in the development of a generative system, encourages the view of design as a process of exploration. The student becomes very attached to the actual system, but is considerably less attached to the results generated at any stage of development of her system. As a result, she is able to be more objective about its products, and thus better understand the role of the development of alternatives in the design process.

The process of creating rules can occur with quantitative ideas that appear ideally suited to such a system. Such ideas have formed the basis of attempts to develop commercial generative systems. But the true pedagogical possibility lies in the development of rules based on ideas that are not normally considered quantifiable. Architectural design is seen by most as an indefinable process that cannot be described in English, let alone in a language as pedantic as those used in computer programming. Yet the process of requiring a student to be more precise about her design intentions – to avoid ambiguity of intention - has enormous potential, both as a pedagogical technique for that moment, and as a metacognitive skill for the student to use in later projects. The representation of ill-formed ideas as concrete design possibilities is an important process in design. The development of a digital generative system permits – forces – this process of concretisation to occur more quickly.

Computer programming offers a structured environment in which the student focuses initially on the requirements of writing code. This focus becomes the scaffolding through which she is supported in her development of appreciation of many aspects of designing. The students are able to better separate and define in a more rigorous manner the stages in design, and the techniques used. The different systems of notation used in design thinking and representation are developed independently of one another, instead of together as in the traditional design studio. The separation is achieved through an additional notation system – the programming language VectorScript. The graphical results are not directly created by the designer. Her investment is in the script rather than its graphical output. The results are therefore open to greater degrees of interpretation and ambiguity. In a more traditional studio setting, the student invests time and effort into the graphical images used to represent her ideas. The graphical output of a VectorScript routine is produced for “free” after much hard work developing the code. It is entirely expendable, and has no iconographic value for the student. Its role is merely to confirm the validity of the rules, and to suggest the direction for the development of the next step. As a results, the architectural products of this design process have a value that is considerably less than the value of the thoughts they provoke. They are therefore ideal elements in a design conversation. This design conversation flows freely both with the designer herself, and between the students in the course.

The discussions that occur between the students centre around the development of the rules they are developing for their code, rather than the single design instance of the design studio. It is the concentration on rules rather than a single design solution that permits this conversation and with it the explorative process to be so efficient in developing the designer's ideas. The demands of programming a computer, inherent in its limitations and rigorous structure, force the student to search her intentions carefully, and to give deeper consideration to the link between verbal intention and graphical representation in architecture. As a result, exploration of the project is possible at a deeper level than with more traditional sketching media.

Aims of the Course

The primary aim of the course I have developed to teach digital sketching is to teach more effective cognitive design skills, metaskills in design that are transferable and exist independent of the project. This understanding is derived from the process of writing the code and not from its resultant product. In this aim, the course I am proposing has two distinct and equally important elements. One, already described, is a software package with particular, familiar characteristics. The other is the pedagogical framework within which the software is learned, developed, and utilised. This specifically Constructionist framework supports and reflects Schön's reflection-in-action description of the design process (Schön, 1983, Schön, 1984a, Schön, 1987), and is in stark contrast to the more traditional approach to teaching software development.

The course also aims to encourage the development of the skills required in listening to the "situation's back talk" (Bamberger, 1996 p.41), to develop an awareness of the process of design as a method of working and thinking that exists independent of any one particular project, and to promote understanding of the link between design generation and exploration. In so doing, the course promotes learning, and with it, design as a process of discovery, an inquisitional conversation. Communication skills are encouraged through both communication with the computer, and group discussion and collaboration surrounding the code the students are writing in their individual projects. Group support for computational constructs reduces the sense of risk of failure, and to support this, emphasis in the final crits is on general discussion around issues rather than instances of student work. The course also teaches students the role computer programming can play in the qualitative aspects of their design thinking. The course does not attempt to impose one model of the design process on the students, nor does it impose one model of learning. Instead, it provides the framework and a set of skills that support and promote the efforts of the students to construct, extend, and understand their own processes of learning and of designing. The framework is the digital intervention, and the skills are the abilities to translate between the verbal, graphical, and architectural languages needed to design, to use external representations as 'objects to think with', to incrementally develop a design idea through tinkering, and to remain open to the back talk of the situation. These skills are a part of the design process, and the students take them into their design studio.

In many senses this course circumnavigates Schön's paradox: that a student must begin to design in order to learn how to design. Schön observed this in the context of architectural design, but as students in this research study demonstrated, developing skills in computer programming offers an opportunity for the architecture student to understand the process of design in architecture. The course is not intended as an alternative to design studio education but as a supplement to it; the techniques used are not intended to replace but to augment other design practices. The course does not attempt to provide answers to the question of what constitutes design thinking. It takes as its starting point the cognitive strategies employed by experienced architects and designers, and asks how such strategies might be promoted and extended by digital media.

The process of designing promoted in the course follows the four stage model of design described earlier: verbal expression of the initial concept, graphical expression, assessment of the output, and modification of the original idea. Each stage is more clearly defined than in a more traditional design setting, helping to clarify the skills involved in designing for the novice designer. There is an important additional step between stages one and two: the process of writing code to express some aspect of the initial concept. This step is the single difference between this process and traditional designing, but hidden within it is the essence not only of the differences but also the significant advantages of this process as both a pedagogical tool and a means for design. It is the process of writing code that opens a window onto a world where greater possibilities are available, and where the nature of design itself is being reassessed. The novice student benefits in the early stages from the rest of the steps. The mature student, on the other hand, takes advantage of this one, new step in an already well developed personal process, and learns not only to exploit the possibilities but to come to understand the resulting effect on her own process of thinking. This process of code writing produces the graphical output to be analysed. The process of learning to write accurate code that results in the creation of graphical form on the screen is the starting point of the course. It is highly structured, the student has few expectations of herself, and is willing to learn from the beginning. Hiding behind this learning process is the concept of developing rules in design. The student is aware that she requires a starting point, and is encouraged to develop an idea within the world of knowledge she already exists – architecture. Her initial focus is on getting the code to work and removing all the bugs, rather than the developing the rules. But as each line of code is corrected, so each rule must be considered and developed. As the ability to write more accurate code is extended, so the focus of the student’s attention is moved gradually from code to content, from syntax to semantics.

Rule Development in Design: Emergent versus Imposed Rules

The word ‘rule’ should be applied with care in the context of design. The dictionary definition of “a prevailing custom or standard; the normal state of things” (OED, 1995) states that rules are created prior to a given situation in which they might be applied. The presence of prevailing customs in society is generally considered to be useful – the members of the culture understand the rules and choose to obey them and thus remain within the bounds of society. In architecture, however, there are few rules beyond the laws of gravity and physics that are applicable and appropriate in every design situation. The culture of today’s architecture in the western world promotes individualism over conformity, so those rules of design that existed in earlier times are now exploited and exploded until little unity remains. Despite potential for confusion, this lack of rules governing the design of buildings can result in architecture that is more appropriate for a given situation, client, or culture.

The term ‘rule’ is therefore rarely used in architectural design today. Yet the process of programming a computer necessarily involves the creation of algorithms, whose dictionary definition is “a process or set of rules used for calculation or problem solving, especially with a computer” (OED, 1995). Therefore if a parallel is to be drawn between programming a computer, and the process of designing, then the use of the word ‘rule’ must be re-examined.

The contradictions between the expected use of ‘rule’ and its use in the context of design lead to a possibly artificial dichotomy that despite obvious limitations nevertheless provides for useful comparisons, particularly for this research. The aim of the teaching in the class was for the students to develop algorithms in a manner that emulated the design process. Some students began with this approach, but others began with a more traditional view of rules and their use in computer programming. The transition from the latter to the former was a significant aspect of these students’ learning. It is therefore useful to categorise the characteristics of these rules and the manner of their use to aid understanding of the students’ development.

The first and perhaps most significant difference between these two types of rules, that are here labelled “emergent rules” and “imposed rules” is the significance of time. The act of designing involves making a number of decisions as the design progresses. As described earlier, each decision is made in the light of the current state of the design solution and the current state of the design problem. Each decision “emerges” from these current states, and both the problem and the solution are affected as a result. Even the very first design decision will emerge from the designer’s first reaction to the problem brief, site and situation, combined with her own experiences, beliefs and preferences. The significant characteristic of these decisions, or rules, is that they are applicable, in that form, in that instance only. They are time dependent, and are rarely if ever repeated in exactly the same manner. Repetitions of ideas are common in design, but the best designer evolves and adapts the ideas to suit changing circumstances and preferences.

In contrast, the more traditional view of rules is their universal applicability regardless of circumstance. They are time independent. These “imposed rules” are inflexible, and apply in every situation in the same manner. In a society this is useful, but in architecture the significance of context – social, cultural, physical – is ignored at a designer’s peril. Today context is considered to be highly significant for successful architecture, and such “imposed rules” are therefore prohibitive and constrictive factors on design development.

As with any dichotomy the divisions between the two opposites are in practice less clear cut than the theory would suggest, and most examples of rule development fall somewhere on a continuum between the two. However, the approaches of the students exhibited enough characteristics of one or other of the extremes. Students often become quickly enamoured with the concept of developing rules for design, and conceive of a large number of rules at the very start of the process. The absence of a design project in Digitally Mediated Design I, and the emphasis on students developing their own projects permitted some students to confuse the process of translating predetermined rules from natural to computer language, with the process of writing emergent rules – in a new language – to develop a design idea. Those students who focus on “imposed rules” eventually face the inherent problems and limitations of this approach. The number and complexity of the students’ rules, and the simple fact that they are usually based on known relationships means that little surprise can be afforded. Without the possibility of surprise there is little point in developing the code in the class, since the same results can be achieved through more conventional and considerably faster means by these students.

This process of rewriting predetermined, “imposed rules” bears little resemblance to the design process. “Emergent rules” are much closer in analogy, and therefore offer greater pedagogical possibilities. These are a natural part of the design process. The process of coding rules in this manner not only develops the design itself, but also the skills of critical analysis and synthesis of verbal concept and graphical representation, required for designing. Through the journey that must be taken to move from “imposed rule” writing to “emergent rule” creation the student sees the design process in a new light.

Techniques learned in this class require an investment of time in the writing of code, and therefore in the clarity of expression of concept. The graphical output is produced 'for free' as a result of this process, and since the student can reproduce it as many times as required by running the script, it becomes entirely disposable. Its role is to confirm the accuracy of the current rule set, and to suggest through new or ambiguous interpretations future rules to be added.

There is a tendency for many students used to a Behaviourist model of entity learning in the programming process - you either "get it" or you don't – to continually throw away a piece of code and begin again each time. However, with encouragement, based upon the effort required to make one section of code work, the student can learn to build upon what has been achieved. An important part of this process is building in the possibility of variation through the judicious use of random numbers within tightly constrained limits. The different outputs are a necessary part of finally breaking the investment in one graphical version, and permitting a comparison of possibilities to occur at a higher level than the development of a single instance of a design solution.

The techniques learned in this class promote this iterative process through exploitation of a basic limitation of writing any form of code: it is only possible to write one line, and therefore to concentrate on one rule, at any one time, particularly for novice programmers. This limitation decelerates the student's thinking process, thereby permitting an iterative and conscious modification process to become a part of the design thinking ability of the student.

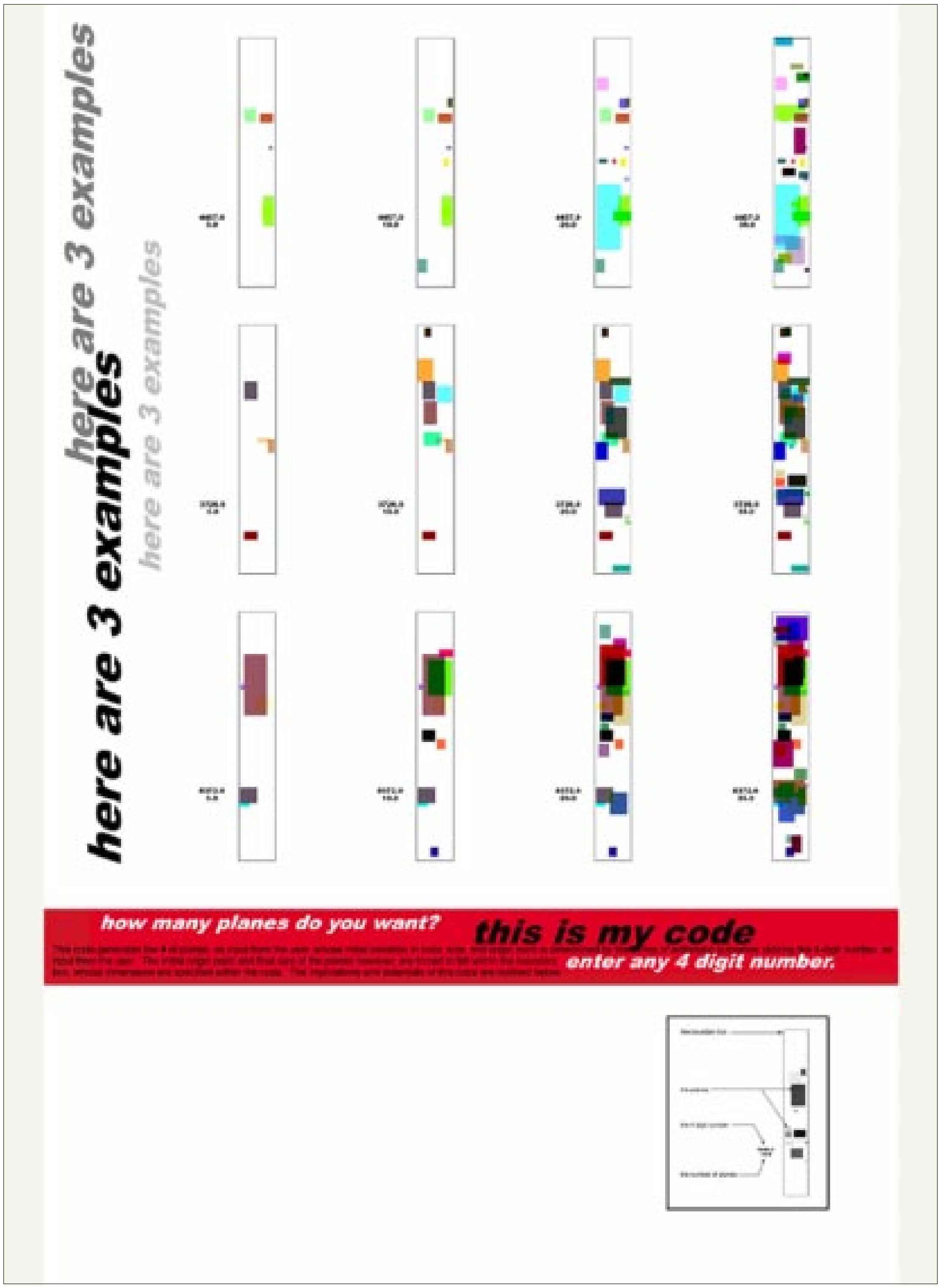


Figure 44. Debbie Kim's first poster, showing the output of her code – compositional diagrams for her studio project. The large rectangle represents a dry dock in the dockland area of Boston

VectorScript as a “Computational Construction Kit”

The software used in the course is the professionally orientated commercially available CAD package, VectorWorks. This CAD software is one of the leading packages for a wide range of design professionals and is usually employed in the production of working drawings. It is rarely used in schools of architecture where there is often greater interest in software with better three dimensional modelling and photo realistic rendering capabilities. VectorWorks (previously named MiniCad) was originally designed with architects in mind, and took as its starting point the two dimensional construction drawings that require precision tools and hatching patterns to communicate building information the known language of architectural plans. As computational capabilities of desk top computing extended during the 1980s and 1990s, VectorWorks expanded from two to three dimensions, and into lighting, rendering and so on. However, unlike many modelling programs, VectorWorks continues to maintain a strong sense of the ground plane upon which the building sits. It is possible to create different layers in the drawing to show or hide information, and these layers both relate to this notion of ground plane and are reminiscent of the layering of sheets of paper on an architect’s drawing board. The concept of the layers is implicitly connected both to the architect working predominantly in plan, and the hybrid two/three dimensional nature of the program.

VectorWorks can be used to create three dimensional objects, but it recognises that the notational system of two dimensional construction drawings is distinctly different from a three dimensional view of the same model in plan. Like any notation system, the characters are used to denote rather than represent. Therefore, VectorWorks divides its working into two and three dimensional modes. Each of these works extremely well individually. However it is in the crossover between the two modes that VectorWorks becomes clumsy, and cause it to be viewed as a hybrid program. For example, there is a tendency to view the layering system as analogous to floor planes, since the wall tool uses layering information to locate and scale wall elements in three dimensions. The extrude function is also implicitly tied to both the layering system and the user’s view. The latter is more often than not in plan, so extruded objects and walls are usually created at right angles to the ground plane. It is difficult initially but by no means impossible to produce sloping walls or complex curved surfaces. Therefore there is a tendency for students who are using it to sketch with to produce very rectilinear, block dominated designs (see figures 45 and 46).

Within VectorWorks is a procedural scripting language whose commands are predominantly and recognisably English, and whose output is screen-based graphical elements similar to those produced by a designer using VectorWorks in a more conventional manner. The procedural nature

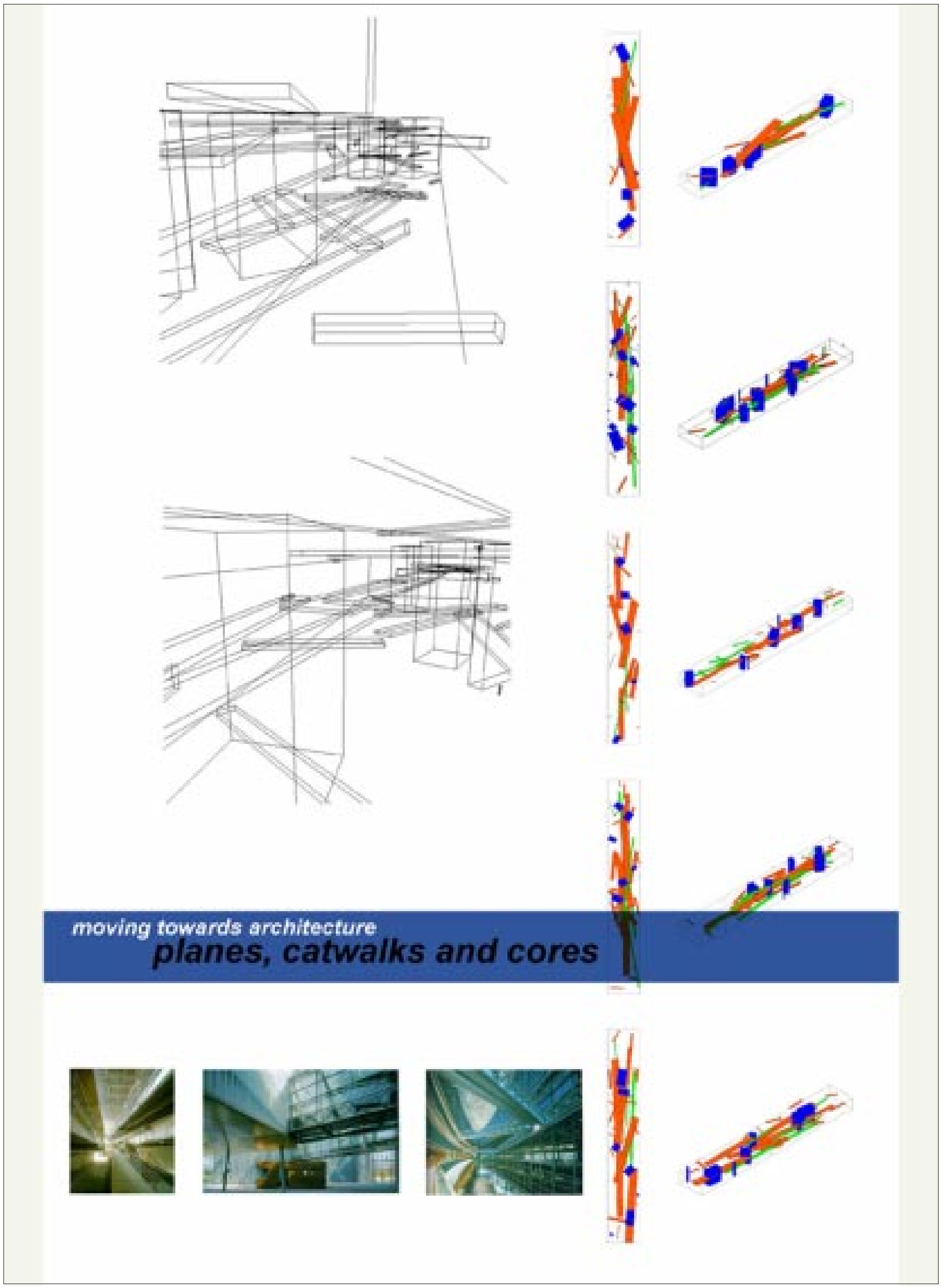


Figure 45. Debbie Kim began thinking about architecture from the start, so had little of the problems of other students in moving from diagram to form. Her integration of studio and workshop was very successful

of the language is significant, for it permits the important step of drawing the student's attention to the procedural nature of the design process. A significant aspect of what the students are learning lies in the slowing down of the process and the concentration of the student's attention on one small step at one time. This type of thinking is required for procedural programming, and structured teaching of the latter helps them develop the cognitive skills required for drawing analogies between it and design process. Object oriented programming requires cognitive skills to consider computational entities or objects that interact with each other and with their environment. These skills draw more attention to architectural objects or "products" than they do to process. Declarative programming requires, as its name suggests, declarations to be made prior to the process, which would support the use of "imposed rules" rather than the "emergent rules" process based activities the class is promoting.

All of the basic functions of VectorScript are founded upon the capabilities of the software's toolbox. As such, it resembles in many ways an architectural version of the Constructionist programming environment, LOGO. VectorScript, the scripting language, is being used in this context as a "computational construction kit" (Resnick *et al.*, 1996). It is adaptable for use by novice and experts alike. For the novice programmer it offers automatic generation of simple statements at the click of a button. With more experience, the novice can script simple lists of commands that execute one function without needing the constructs normally expected in a full programming language. For the expert programmer, it offers the full range of facilities of any computer language, including inheritance, object creation and other constructs from object orientated languages. It achieves all of this in a language whose basic commands are recognisable English, and whose output is instantly graphical. It is very easy to create a graphical image to test a design idea. However, as with LOGO, the success of VectorScript is dependent on the pedagogical environment within which it is used.

The students use VectorScript to create scripts that generate three dimensional form, corresponding to the student's design intent at any one given moment in her design process. The potential set of elements in this context is the set of graphical primitives and their derivatives possible in VectorWorks. The rules are defined by the designer and emerge from her design process. They are expressed first as concepts or ideas, then translated by her into the scripting language. In the development of a digital generative system, the computer as a processing engine offers both the activation and the motivation. The direction comes from the students' intellectual efforts in producing something that is meaningful for them (Bruner, 1966). The possibility for variation is built into the generative system through the selective use of random numbers to encourage the process of exploration (Guarra, 1974). Elements are given form, but the exact size, location, or other physical attributes can vary.

The focus of attention for the student in the creation of her generative system is on the process of development of rules and elements, not on the final code. She is the only person for whom the script is being written. This is in contrast to the development of generative systems in other contexts, particularly digital ones, where the focus has been on the development of complete systems for use by others (Henke, 1990, Krishnamurti & Giraud, 1986). In the Digitally Mediated Design 1 course the digital generative system is merely the engine for the development of the design through the creation and testing of descriptive decisions and the depictive elements they are applied to. The development of these rules and elements is incorporated into, extends, and ultimately becomes, the design process. At every stage the next step to be taken is the creation of graphical output that expresses the student designer's latest design decision. The techniques taught in this course offer the designer additional advantages not available with more traditional sketching tools. In developing a set of rules, and creating a generative system that graphically illustrates the application of those rules, the designer is sketching not a single design instance, but an entire design typology of her own making. She teaches the computer the theme of the design. The computer contributes to the conversation by presenting variations upon that theme. At every stage she can make comparisons between these variations to establish the validity of the rules, and develop the next design decision, the next Emergent Rule, and thus the next step in the script.

As already mentioned, the existence of a suitable scripting language used to teach this process does not alone ensure the student learns about the design process. Computer programming courses for architects are not new, but the significance of this course, and its appeal to those students who find programming difficult and alien lies in its Constructionist pedagogical approach. It uses the less formal, 'soft' pedagogical approach described by Turkle and Papert involving a process that many designers find familiar. The student begins with an idea of what is needed, and produces from the outset software that works as a whole but whose functionality and sophistication is refined and developed in successive stages through gradual 'tinkering' with the code. "Each step is a small modification to a working program that she has in hand. If a change does not work, she undoes it with another small change ... At each stage of the process, she has a fully working program, not a part but a version of the final product" (Turkle & Papert, 1990 p.356). This process is the software development version of architectural designing. The developer has no definitive description of the final outcome, only a strong sense of the direction in which the code should be developed and the next step that should be taken to promote the direction. At every stage there is a working piece of code that can be tested and subsequently used as an 'object to think with' for the next design decision. The code plays the role of the sketch in the design process. The product of this approach are the cognitive demands of such a process and the consequent development of the design project. This description can easily be applied to an architect working on part of a design. Each mark made on the paper is a modification to the drawing and to the design ideas in her head. It is considered within the context of the design brief, and modified accordingly.

It is this hands on, learning by doing, pedagogical style that is familiar to an architecture student, and is the one that permits a cognitive link to be made between the process taught in the course, and the processes she is struggling to understand in the design studio. She is encouraged to develop her generative systems through this bottom-up tinkering and in so doing to develop her design ideas through a process of experimentation. Such a process forces the student to confront her past iterations, and from them to infer the rules with which she made her decisions. It develops in the student an understanding of the episodic, cyclical nature of the design process, and of the unique, inquisitional conversation skill she must utilise in her design work. She must be constantly inquiring and learning, re-evaluating what is known in the light of what might be interpreted from the latest output.

Teaching a course in this manner requires additional demands of the professor not often required in the more traditional, 'hard' pedagogical approach to teaching computer programming. As the students progress with their scripting, they require different computational constructs at different times, often before these have been introduced in the lectures that accompany the project work. The professor must spend time helping each student with her code. The pedagogical approach required thus mimics that used in the design studio, with a combination of individual tutorials, lectures, and student presentations. Since the student begins her project work considerably before she has learned much of the fundamentals of computer programming, her initial attempts are often poorly constructed and inefficient by more traditional, computational standards. Yet as with pencil sketching in design, the skills required in this process develop over time and are refined within the context of the design studio setting. This context based environment, where the student is involved in a design project that has personal significance to her, insures she learns the computational skills involved more effectively than when they are learned out of context.

The course

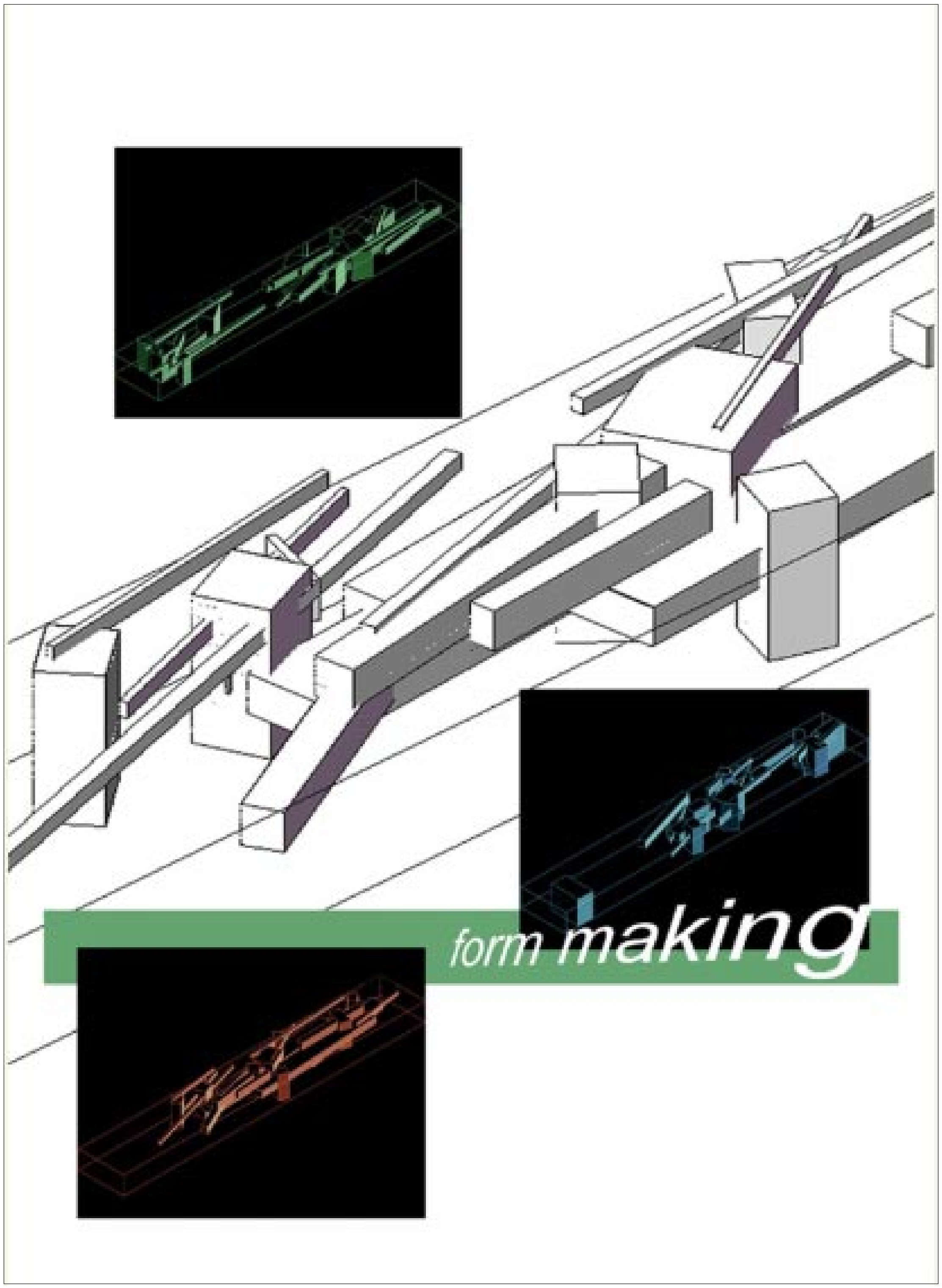


Figure 46. Debbie Kim's final poster, showing some of the final outputs of her code. She used these models to further develop her studio project, due for completion several weeks after this

Introduction

For the last three years, students of architecture at MIT have enrolled in a course that teaches them techniques of computer programming as a tool to think with in their design process. 'Digitally Mediated Design' is a course that is open to students new to architecture who are beginning to understand what is required in designing. The course runs over two semesters. During the first semester students learn the techniques involved through a design project of their choosing, whilst in the second semester the techniques are applied to a design project set by the professor. It uses an existing, widely available CAD software package that contains within it a programming environment exhibiting many of the fundamentals of Constructionist software. The course is not intended solely for the beginning design student, however, and has much to offer the more advanced architecture student. It is also open to graduate architecture students wishing to learn how computers can assist in their design thinking process. The course is presented at the start of the semester in the 'Show and Tell' presentation along with all the other Department of Architecture studio and workshop professors. It is a difficult class to present accurately, since much of what is learned involves the development of a personal design process that is not always immediately obvious from the graphic work that can be shown. It is described by me as a class that looks at ways in which computers can be used to do more than create sophisticated static imagery, and how digital media might play a role in the design process. There is an attempt to play down the significance of the role of computer programming, since the course is aimed at primarily design oriented students who can view programming with some suspicion, and like most students assume it is for purely quantifiable aspects of design.

The aim of the course is to teach students of architecture to use computer programming as a way of understanding their own design process. However, the course is not simply the introduction of new computational techniques. The pedagogical approach to the course is as important a factor in the students learning as the subject matter. This approach is highly flexible, and permits the students to develop at their own pace in a structured environment, through project work and extensive one to one coaching. Collaboration and shared knowledge are promoted as much as possible. This collaborative spirit helps reduce the competition that exists in the design studio, as the students are encouraged to help rather than compete with each other. Juries and class presentations are promoted as times of group discussion of general issues rather than criticism of specific students' work. All of these factors contributed to the students learning. It must also be remembered that the interviews themselves may unwittingly have had an influence in the learning of the students, and therefore should be considered carefully in any review of the techniques used to teach the students.

Course Description

The group studied for the research participated in the Spring 1999 semester. The course was scheduled to meet twice a week, on Wednesdays from 4:00 p.m. to 6:00 p.m. for a lecture, and on Mondays from 2:00 p.m. to 6:00 p.m. for recitation. The two hour lecture period was usually divided in two, one half a lecture by me on computational constructs, the other by a guest speaker in the field of design and computation. These external speakers were intended to introduce to the students a broad range of uses of technology in design and to raise subjects for discussion. All of the guest speakers talked about their own research work, developing the basis for a ‘knowledge building community’ and emphasising the importance of discussion over criticism in the course.

During the recitations the students initially worked on practical exercises, then later in the semester they worked on their own projects with help from myself and the TA. The first exercise to be completed by the students in the first week was to use VectorWorks in a conventional manner for drawing, introducing them to the available tools in the drawing environment that the language utilises. Every procedure and function in VectorScript relates in some way to a tool or command in the drawing part of VectorWorks. Since every CAD package is based on a slightly different functional model, it is important that the students begin to understand in the first week the mental model required for this one. Usually, students with prior CAD experience will repeat methods of working from known packages regardless of applicability in the new one. An example can be seen in the way in which VectorWorks shortens a line. In AutoCAD, the software package most of the students have previously used, shortening a line involves marking the point at which it should end, and then cutting it and deleting the excess. In VectorWorks, shortening a line involves using the interactive cursor to click and drag on one end to make it the required length. VectorWorks does have a Trim function, so it is possible to mimic AutoCAD, but at this is a less efficient approach.

Prior experience of other CAD packages influenced both the type of drawing completed as well as the manner in which the tools were used. Raphael was the only student to have had prior experience of VectorWorks, but his entire use had been in two dimensions and so the drawing he completed in class did not look at any of the three-dimensional functions of the software. Esther’s previous CAD experience was almost exclusively AutoCAD, and although she produced a complex three-dimensional model of the back of the chair, she transferred all of her AutoCAD skills to VectorWorks without regard for their appropriateness. The drawing she produced was significantly inefficient in computer terms, since had she used VectorWorks in a manner that better reflected its potential, the file size of the document would have been substantially reduced.

The exercise was completed in the second week of class, and presented by the students in the recitation of Week Three. Most students are familiar with a pedagogical approach in the computer studio that requires them to mindlessly model or draw an existing object or building. The focus of the students during the second week therefore was on the drawings produced and the expectation for that first presentation was that there would be a critique of the resulting products. However, they were required in the presentation not to talk about their drawings, but to give a critique of VectorWorks, and to describe which of the tools and functions they used and what they thought of them. The audience for their presentations was the other students in the group, rather than me as the professor. The emphasis was on collaboration and shared knowledge; in one week it is not possible to discover everything about a computer program, but if everybody shares experiences of using the software it is possible to learn more than working alone. The criteria for success in this presentation became how well each student could help the other members of the group, and how useful the information they provided might be. There was a common goal towards which everyone was working: the enhanced understanding of the group as a whole, rather than individuals in competition with each other.

At the end of the presentation, the next assignment was given. Each student was to take one particular aspect of the drawing they had done, such as the creation of a three-dimensional pitched roof, and write detailed, step by step instructions of how to achieve this. In the recitation of Week Four, the students presented these written instructions to each other; everyone had a copy of everyone else's instructions. Once again the emphasis was on sharing knowledge. This time, however, there was a premium placed on each individual making their instructions as clear and unequivocal as possible. Individual competition returned, since each student wanted to make their instructions the clearest, but it did so within the context of collaboration and shared learning. Additional impetus for clarity came from the knowledge that confusing instructions would result in students, wishing to follow them, asking the author to explain herself more clearly.

There were two further and obvious pedagogical aims in the writing of these instructions. Firstly, keeping such a record of a newly acquired skill helps the author coalesce in her mind the steps required. Secondly, the instructions promoted discussion about the definition and creation of algorithms, the precursor to understanding what is involved in communicating with a computer. In the lecture of the previous week, the students had been given an overview of the language and introduced to the most basic form of using VectorScript: writing single line statements without the need for a more traditional programming structure. (See Appendix B, Class Notes). In the recitation of Week Four they were divided into three teams and given a choice between solving two different problems. Either they were to create a shape on the screen that then moved in a linear direction, or they were to make a rectangle change progressively and gradually into a circle. They were given hints and helpful suggestions by me as to how they should begin, and they showed considerable creativity in attempting to solve these with minimal knowledge. Each of these exercises involves repetitive action by the computer. In moving the shape across the screen, the simplest method is to repeatedly draw new versions of the same shape that are located progressively further along a line. Changing a rectangle into a circle involves using the intermediate shape of the rounded rectangle and progressively changing the radius of the curve on the corners. In both cases the students laboriously wrote out the changes in a long list of repetitive single line commands. This inefficiency paved the way in the subsequent lecture for a discussion about repeat structures, introducing basic computational constructs such as Begin and End, and leading towards the concept of variables.

In the following recitation the students rewrote their code in a more efficient manner to utilise these structures. In subsequent lectures they were gradually introduced to reserved words, constants, expressions and operators, expression statements, control structures, conditional statements, and so on. Appendix B, Class Notes, gives full details, and the order in which the information was given. Sections of these notes were given to the students at the end of every lecture as the material was covered. The students also had photocopies of the first six chapters of the VectorScript manual that came with the software, giving them another source of information. In addition, the on-line help contained all of the procedures and functions grouped according to subject, with explanations and examples of use for each. The emphasis was that the information concerning the software and its functions was shared knowledge in the general domain, and that the only thing prohibiting learning was difficulty in memorising its use, rather than any lack of intelligence. It was emphasised that the photocopies, handouts, and on-line help were all aids to memory, not to stupidity. Asking for help in remembering how to use a particular procedural function was encouraged, and in the first weeks there is usually much noisy discussion as the students try to help each other whilst simultaneously sitting at their own machines. It was made clear to the students that they should work together to solve each other's problems. Students appeared to enjoy this, since the competition they are used to exists in the course only in the initial decision for project development, not in the programming details. Individual authorship of design decisions is maintained in an atmosphere of collaboration over syntax. Students in the course are offered chunks of code from previous students, emphasising a lack of ownership to code writing. By the third or fourth week there is usually much reciprocal sharing of ideas and debugging – everyone is teaching and learning, there is “collaboration in the air” (Berger & Luckmann, 1966, Harel & Papert, 1990).

In the fifth week, after discussions with me, the students began to spend the recitation periods working on their own projects. Initially they were confused about how to begin, and their preconceptions about what they might achieve, and what computer programming is for, had a strong influence on them. As described in the results section of this thesis, the initial assumption for the students is that the most important aspect of the process is the code they are writing, and their initial attempts therefore focus on writing commands they believe will be useful to someone else in the future. It is often several weeks before they realise that the emphasis is not the code but the design they develop. This change in understanding requires a change in perception of what code to write. They move from thinking about what computers are for towards thinking about the representation of a design problem. This move, from function to representation is critical. When, in the first few weeks, the students ask how they should begin, the answer is always for them to consider the way in which they would begin to represent the same problem in their sketch books. The emphasis is on design, not computation.

Very quickly, with the scaffolding provided by me, their colleagues, and the programming language itself, the students were in Dwyer’s terms ‘flying solo’ well before all of the programming constructs have been covered in the lectures. At this point, between Weeks Six and Ten, the students were proceeding at markedly different rates requiring different constructs at different times. My role as instructor was to remain flexible and to offer individually tailored scaffolding as each student became fluent in this new language. Every conversation with the students emphasised that the important aspect of their thinking should be that they have a specific design goal in mind, and the only inhibition is their lack of knowledge about the programming language. This approach required flexibility in teaching, since in more rigid pedagogical style of each student learning the same constructs at the same time emphasises the computer and reduces the importance of the individual design project. Creativity and individual authorship, and emphasis on design are more important than rote learning of computational constructs.

Every week the students gave brief presentations to each other about the work they were doing. During the 10th, 13th, and final weeks of the semester these presentations were more formal. They were required to make one poster for each of these, 24 by 36 inches in size to present their ideas. The purpose of this exercise was threefold. Firstly, the creation of the poster acted as a punctuation mark for the students to look back on their achievements and assess the direction of their projects. Secondly, at the time of the first and second presentations, these visually oriented students had been involved with the rather dry process of code writing for so long that the process of creating a visually pleasing record of their achievements was important in boosting their morale. Finally, the significance of a comparative record demonstrating progression through the semester cannot be underestimated. Each week throughout the semester the students were required to hand in the latest version of the code. They were encouraged to use the “Save As” command as often as Save, to create copies of their work rather than continuously rewriting the same file. Yet unlike the design process, where progression is demonstrated through visual record of sketch books and models, progression of code writing is difficult to see through this file saving process alone. In a Constructionist learning environment, effort is rewarded over ability, and students are often encouraged keep diaries or work books that demonstrate their achievements over time to themselves as well as others. The three posters made for presentations served this function, as well as providing a visually pleasing basis for discussion by the group and the jury panel.

All of the computational constructs were introduced and explained to the students by the end of Week Ten, and in the final five weeks the students concentrated on their projects. Each student met with me at least once a week during the recitation to discuss progress and development of ideas. In addition, the TA offered support in writing code to the students outside of class hours. Each student was required to meet with the TA at least once a week, although most of them met with her more often.

In the final presentation there were five guest critics, two of whom were from the design faculty. The other three specialised in some way in computational design. In addition, an open invitation was circulated within the Department, and other members of the community came to find out what the students had been doing and contribute to the discussions. These were broad ranging and involved not only the students’ work but more general issues such as the best method of communication of rules, the role of programming in design, the effect of programming on design process, and so on. Students were asked what they felt they had learned in the course, and the data gathered at the presentation formed an important part of the research study.

In the subsequent semester a second class is offered to students who have either taken the first class, or who have taken one of the courses on shape or colour grammars offered in the Department by Professor Knight. The emphasis in this second semester is on the practical application of theoretical techniques to learned. In the Fall semester of 1999, five students joined this class, three of whom had previously taken the Digitally Mediated Design course, two of whom were in the research group of Spring 1999. They developed a design for the Memorial for Martin Luther King, an international competition based in Washington D.C.

Early results of the course: Pointers for Developing Research

In the first two years of teaching the course, as with the group of students who formed the basis for the research in the third year, there was a wide range of starting levels and abilities. Each student appeared to gain from the course in a different manner, but a common theme was strengthening of weaknesses in the students' abilities and understandings of design thinking and representation. For example, one student from the second year, skilled in graphical representation but poor in verbal description, improved the latter through her project. Verbal skills are important not just for public presentation – they are a vital part of an architect's abilities to communicate with herself and thus develop the design. Another student used the course to develop aspects of her studio project at that time that required an alternative approach in thinking and representation. Other students developed in other ways. The specific areas of change anecdotally observed in the students of early courses were changes in the student's sense of control over the computer, changes in expectation of digital media, changes in attitude towards the role of digital media in design, changes in levels of confidence in inter-group working, and, most importantly, changes in understanding of the design process. This change in understanding appeared in the early years of teaching the course to come from the normally implicit steps in the design process being decelerated and made explicit.

Other aspects of the studio setting for the course appeared in early years to play an important part in the students' understanding. The formal presentation process became an important tool in bridging the gap almost all students perceive in their expectations and use of programming in design. Since the students who join the course are self-selected, it must be assumed they are naturally predisposed to such a process, but they must present their work to an often skeptical audience in the crits. This forces them to examine more closely the inherent links in the project work between design and programming. The links exist, but many students do not immediately see them and require some time to understand the possibilities.

Research Study

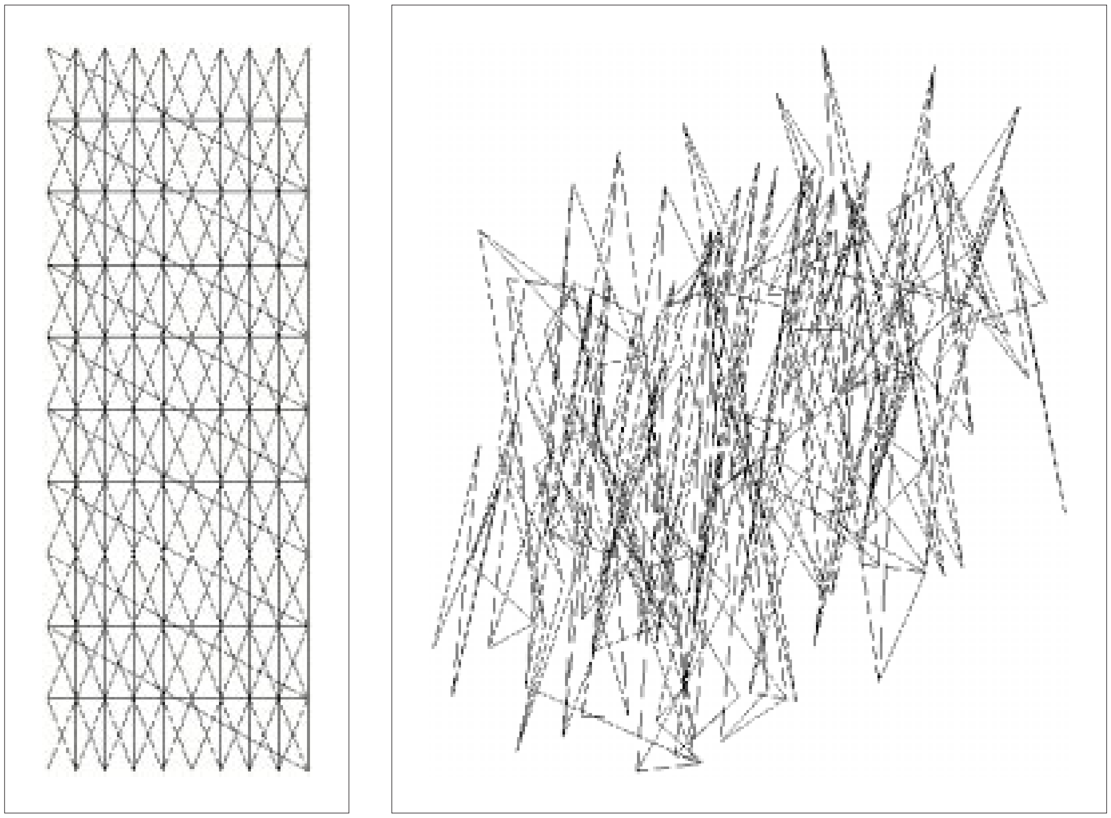


Figure 47. Hiroshi Okamoto wrote code to develop a three dimensional mesh that deformed according to the placement of objects underneath. Here, early output of his code appears ordered in plan (left) but not in 3D

Need for Assessment

Digitally Mediated Design 1 is an innovative approach to the design education of architects. Yet any intervention of an educational process, particularly controversial ones using new pedagogical techniques and technologies, requires an in-depth understanding of what it is the students are learning and how the course affects their educational process. Mere description of the course does not alone constitute adequate assessment of this impact. Results of earlier years of teaching the course, described in the previous section, indicated but did not formally support the claims made for its efficacy. Therefore, the second and equally important part of the work for this thesis was the development, application, and results of research into the significance of this intervention and its impact on the students in the Spring 1999 course. This research specifically examined the effects of the techniques the students learned on their design process, their understanding of design, and the transfer of these cognitive skills to other aspects of their design education. The most significant questions to be answered by the research are:

What is the benefit of teaching design as a process in schools of architecture, rather than assuming such an understanding will come as a result of teaching design through copying the practices of the architectural profession?
What value might there be in teaching students about the process of how to get to their desired products rather than assuming, as happens in many design studios, that they will learn this process merely as a consequence of getting the product 'right'?
Specifically, what is the effect of teaching design by analogy using the development of digitally mediated generative modelling to students of architecture?

These questions are based on the premise that the potential role for digital media in design and design education has yet to be fully explored. The basis of this thesis is the proposal that the process of design can be taught in a way that was not previously possible as a result of today's technology. The research examined in detail a group of self selected architecture students as they learned to learn in a new paradigm. It recorded, through in-depth interviews at regular intervals what the students learned, how they learned it, and specifically whether what they learned was transferred away from the confines of the course and into other areas of their work.

The question of the impact of teaching the design process away from the design studio was explored and partially answered by the research results. The question of what value there might be in addressing the discrepancies between the novice student's assumptions of learning, the process of learning they are supposed to be involved with in architecture school, and the consequently learned process of designing they are supposed to leave with proved more difficult

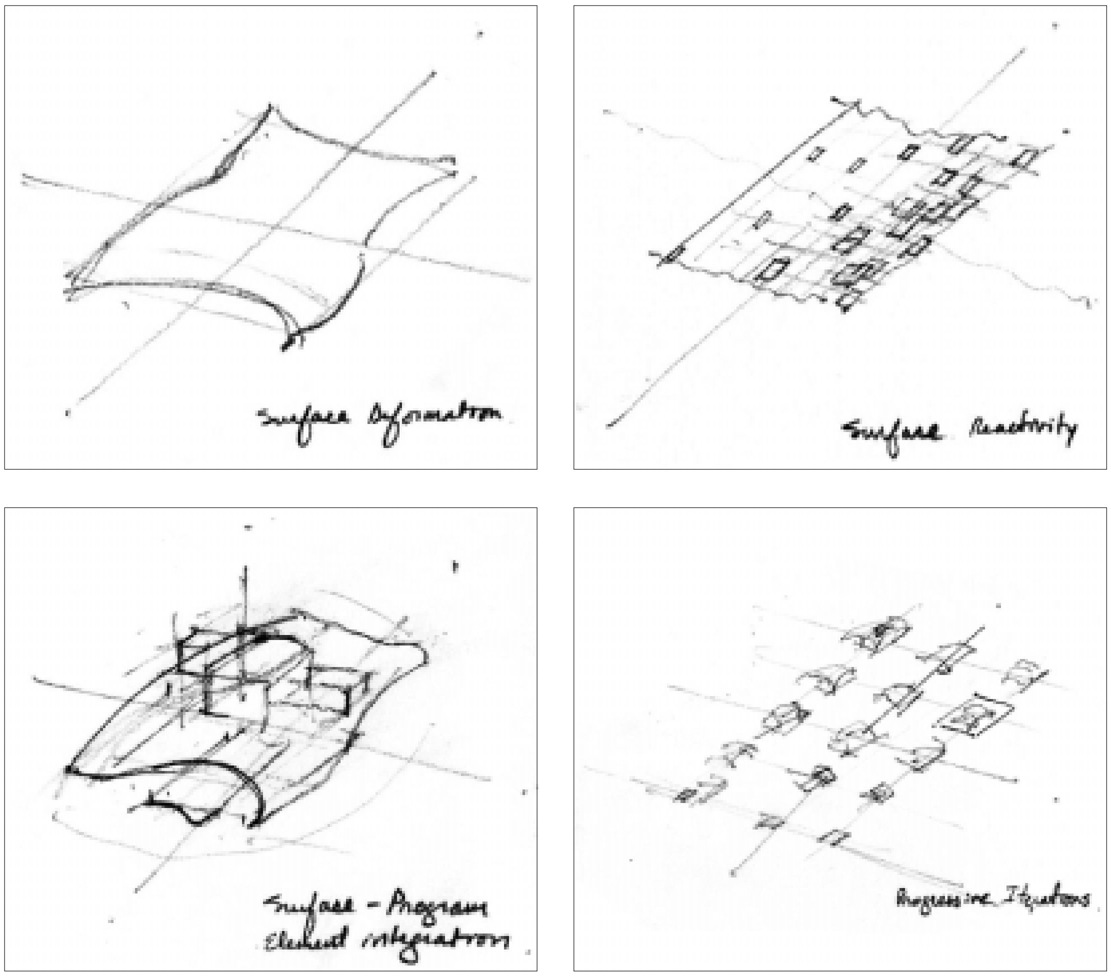


Figure 48. Like Anthony Guma, Hiroshi Okamoto used diagrams and text to help express his rules to himself before coding

to answer. None of the students in the study were novice, and this question therefore still remains to be more full explored. The research also asked, if architectural education must concern itself with changing the model of learning with which the student arrives at the school of architecture, what role might digital media play in this process? Changing the way in which students see the process of learning and therefore of designing – indeed, developing an understanding of the two in the students to begin with – is a question that is as difficult to answer as asking whether or not such an impact produces better architects. But never the less it is an important aspect of architectural education and should continue to be explored.

Research Questions

The initial questions of the study looked at the students' current thinking about the link between design and digital media. The research examined the effect of the course on the students' understanding of design, digital media, and the relationship between them. It aimed to demonstrate changes in these understandings, and to induce theories about the changes as a result.

This thesis answers the following research question:

> What awareness did the students have of their own design process and how did this change?

The research study was a qualitative, value laden one. The design of the research was emergent (Lincoln & Guba, 1985), in that questions for each of the interview schedules were developed as a result of the analysis of the previous interview (See Appendixes C – F). As a result, and as with many qualitative studies, the initial questions that emerged from anecdotal evidence of previous years of teaching the course – and the reason behind interest in a more formal analysis – did not entirely predict what was to emerge. The nature of qualitative inquiry is to gather data that may be studied and from which will emerge both questions and answers. Qualitative inquiry is based on the researcher entering the field relatively uncluttered by previously developed hypotheses and questions. However, it is important particularly for a beginning researcher to have some initial questions that may guide the study, provided those questions can themselves be modified rejected or added to as a result of analysis of the data. Therefore, the following questions, formed from previous years of teaching the class, were also used as a starting point for the study.

> What is the impact of the course upon the students' metacognitive abilities of reflection (Nickerson *et al.*, 1985, Perkins, 1992) upon their own design process?
> Are these metacognitive skills transferable?
>
> What did the students learn?
>
> How did the students view and use computers in the design process and how did this change?
>
> What evidence demonstrates change in the students' perception of digital media in design?

> What model of learning did the students have at the start and end of the course?
>
> How might collaborative skills be developed in architectural education without the loss of independence or authorship?

These questions evolved over time, and were developed and extended as the data emerged. Topics thought to be simple to demonstrate in the research, such as improved collaboration, proved in the study to be more elusive than had previously been expected. Some areas of interest took on increased significance, such as the model of learning, whilst others reduced in significance as the data unfolded. It is not possible to begin a research project with an entirely blank mind, but in this type of research it is important to recognise preconceptions and be open to change.

Structure of the Research Study

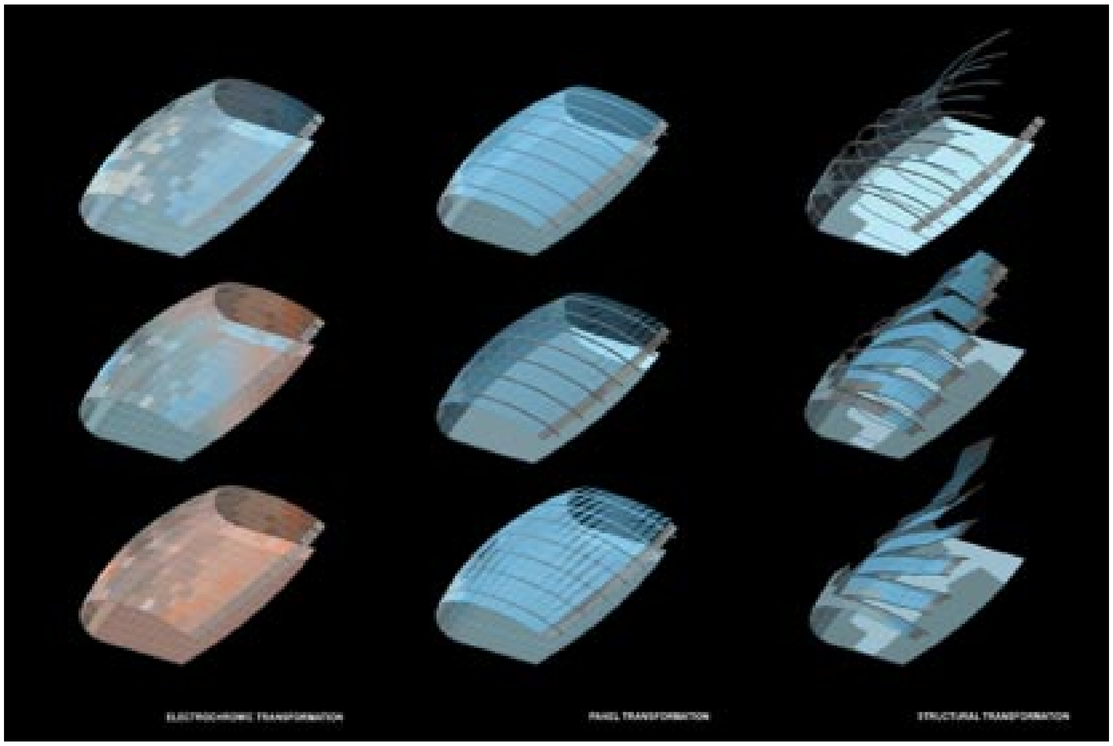


Figure 49. Hiroshi Okamoto used other software to help coalesce his ideas concerning the deformation of the glass mesh he was designing with code in the class

Introduction

Research into the design process has typically utilised protocol analysis methodologies (Adelson, 1989, Agabani, 1980, Akin, 1984, Akin, 1993, Akin *et al.*, 1992, Atman *et al.*, 1999, Do *et al.*, 1999, Eckersley, 1988, Ericsson & Simon, 1980, Galle & Kovacs, 1992b, Hayes, 1981, Lawson, 1993, McFadzean, 2000). This technique relies on removing the designer from her normal workplace and studying her working processes in a controlled environment. It is a technique that has produced some valuable research into how architects design, but it is not a technique that lends itself to studying an educational intervention. Studies of the education process need to occur in the pedagogical setting rather than away from it. It is also recognised that developing a control group who are not party to the educational intervention is difficult, if not impossible, in a real world setting. As with much research in education, it is not possible to isolate factors influencing change in understanding, since the students are not operating in isolated conditions and are subject to many other factors in their daily lives. Equivalence between the groups is impossible. Research requiring a control group assumes the intervention as a 'treatment' exists as a single influence and comparisons are made between having and not having the course. Results are studied with a view to proving whether or not the intervention was successful. The intervention is seen as a 'black box'. *How* it was successful is not of interest – a Behaviourist approach to research.

The research conducted into the course described here took a different approach. It did not have a control group, and did not look at students in the course as a group in comparison to a group of equivalent students who did not take the course. Instead the study looked at how and why the course was and was not successful - the black box of the intervention became transparent and the processes within it were studied. It was a qualitative research study that took as its model the theory based approach of recent general education research (Chen & Rossi, 1983, Weiss, 1997, Weiss, 1998). The model for the research is one in which factors within the course were taken into consideration. The research was developed to predict, examine, and propose reasons for different aspects within the development of the course. Such an emphasis on grounding the theoretical aspects of the course had many advantages. In particular it supported the evolving nature of the research design, and permitted interim results to be analysed and reported to a much greater extent than the more traditional all-or-nothing research design. Such evaluation aims to explain how and why effects in a course occur, and to give a better sense of whether the course was the cause of the desired outcomes. Since this course was not a controlled 'intervention' with a corresponding control group, but rather a real course with real students engaged in work that was directly relevant to their degree programs, such a model of inquiry was appropriate. It was

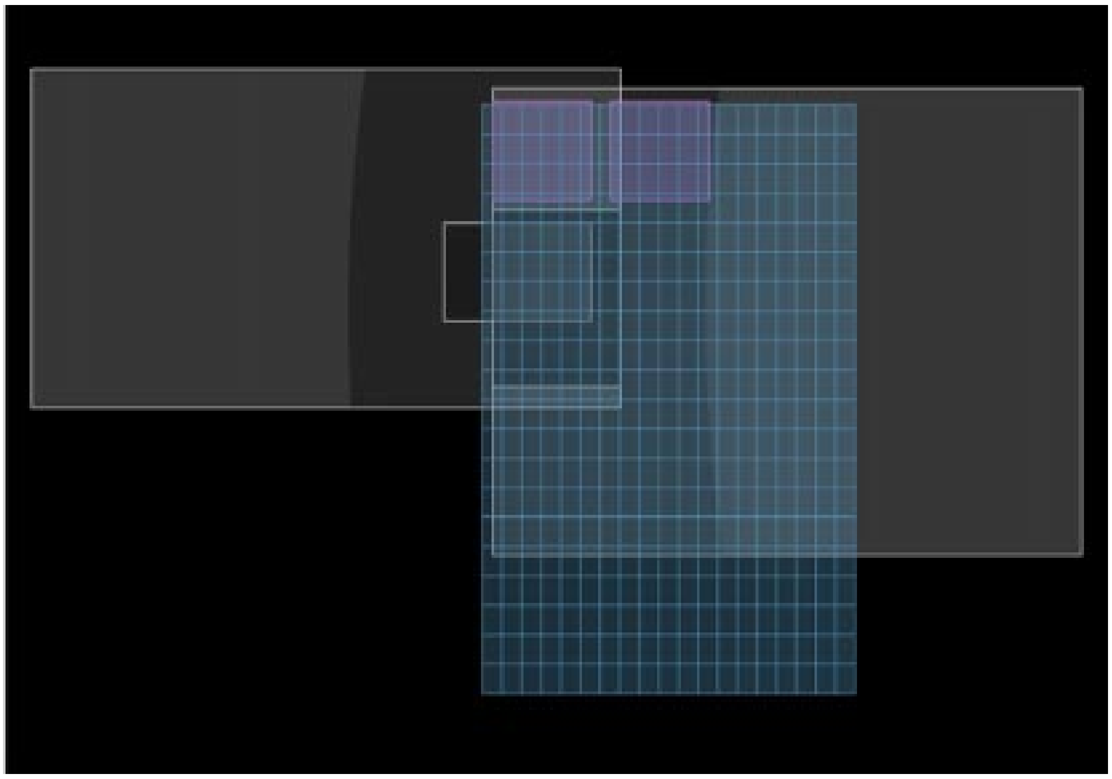


Figure 50. Plan view of the final output of Hiroshi Okamoto's code, showing the mesh roof and underlying forms

the effect of the processes learned in the course, rather than the results or products of the course, that were of interest in the study, just as it is the process rather than the product of design that is supported by the techniques taught. It was not appropriate to ask in this instance whether these techniques 'worked' but rather what it was that made them succeed or fail.

Population

With any study involving human subjects, the potential population from which the participants were chosen must be defined. In this study, however, all of the students were self selected. The course is open to all architecture students within MIT, and six graduate students entered the course in Spring 1999. The total population of students of architecture at MIT is small and there is a wide range of non-compulsory courses in the Department that are offered to the students who choose them based on personal interests and educational aims. It is therefore assumed that all the students in the study were predisposed to gaining something from the course and aiming to succeed within the course as a part of their degree program. All the students except one had specific, positive reasons for joining the course. Making such a course compulsory, and randomly choosing students to be in the study would presumably have had different results. No student dropped out of the course during the semester, nor out of the research study during the year.

Data collection

The aim of the research was to establish changes in understanding of the design process and to assess the impact of the course as a transformative educational experience. Attempting to establish the development of, and changes in, metacognitive skills cannot be done by asking simple questions. If it is the questions themselves that are introducing metacognitive concepts, and not the educational intervention, the research is invalid. Instead, the development of the skills must be examined through changes in words used by the students as they talk about their work. To established this, the students were interviewed three times in a 10 month period. These in-depth interviews were conducted at the beginning and middle of the first semester, and once in the middle of the subsequent semester. Three formal presentations during the semester also formed a part of the data gathered, along with the products of the process – the students' work.

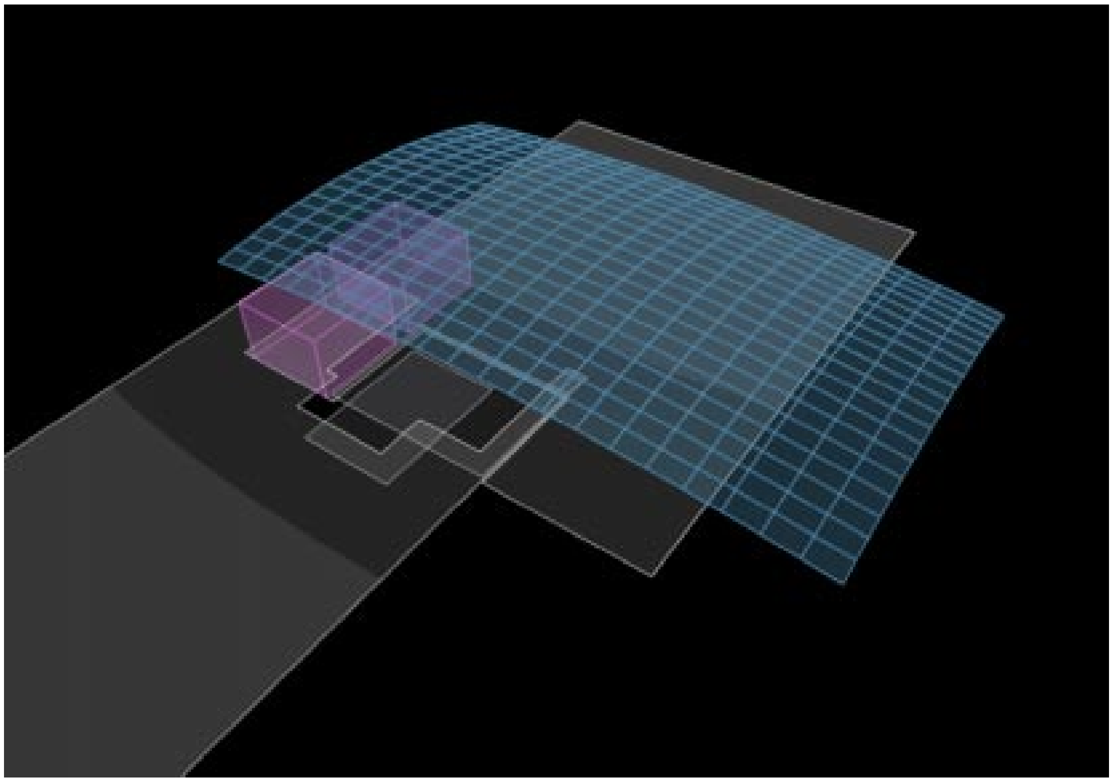


Figure 51. Aerial view of the final output of Hiroshi Okamoto's code, showing the mesh roof and underlying forms

The first interviews were conducted during the second and third weeks of the Spring semester. The aim of this first interview was to establish a basis of understanding – or external descriptions of the students' understandings, since it is impossible to determine actual degrees of understanding in another individual – for a number of different issues. These included the students' approaches to education, their awareness of a personal design process, their views on, and uses of, computers in design, their reasons for choosing the course, and their preference for working in groups.

During analysis of data from the second interview it became apparent that the conflict between my both teaching and researching the course was having an unexpected but retrospectively obvious impact on the data collected. Students were reluctant to talk about their work in the interview in any depth because subconsciously they knew I was well aware of what they were doing through teaching the course. As a result, perhaps exacerbated by the close proximity to the first interview, data from the second interview offered little additional material. To address this difficulty, the proposed third interview planned for the end of the semester was substituted by an enhanced end of semester crit, in which the crit panel had a list of topics to cover in their questioning of the students (See Appendix E). Subsequent analysis of that interview proved much more profitable than the second interview.

The final interview took place in the middle of November 1999, six months after the end of the course. The expectation, based on experience from previous years of teaching the course, was that significant changes in understanding of the design process do not occur until after the end of the semester, when the student has had a chance to reflect upon the techniques taught. The ultimate goal of the course is to develop metacognitive skills in design that are transferable from one situation to another – in effect to transform the way in which the students think about their design process and the role of computers in that process. Such a transformative educational experience is a complex one to describe, and can only be fully appreciated through direct experience of the techniques. The long and steep learning curve required in the course, whilst pedagogical necessary, inhibits the understanding of the analogous aspects of the process. It is not until the student is comfortable with the software that she is able to raise her thinking to a higher plane and reflect upon the experience in its totality. Thus typically this understanding of the relationship between the techniques in the course and the process of designing is not fully understood until late in the semester, and is best explored in projects conducted in the following semester. The timing of the fourth interview therefore permitted this understanding to develop more thoroughly prior to the data collection. Two students from the Spring 1998 course chose

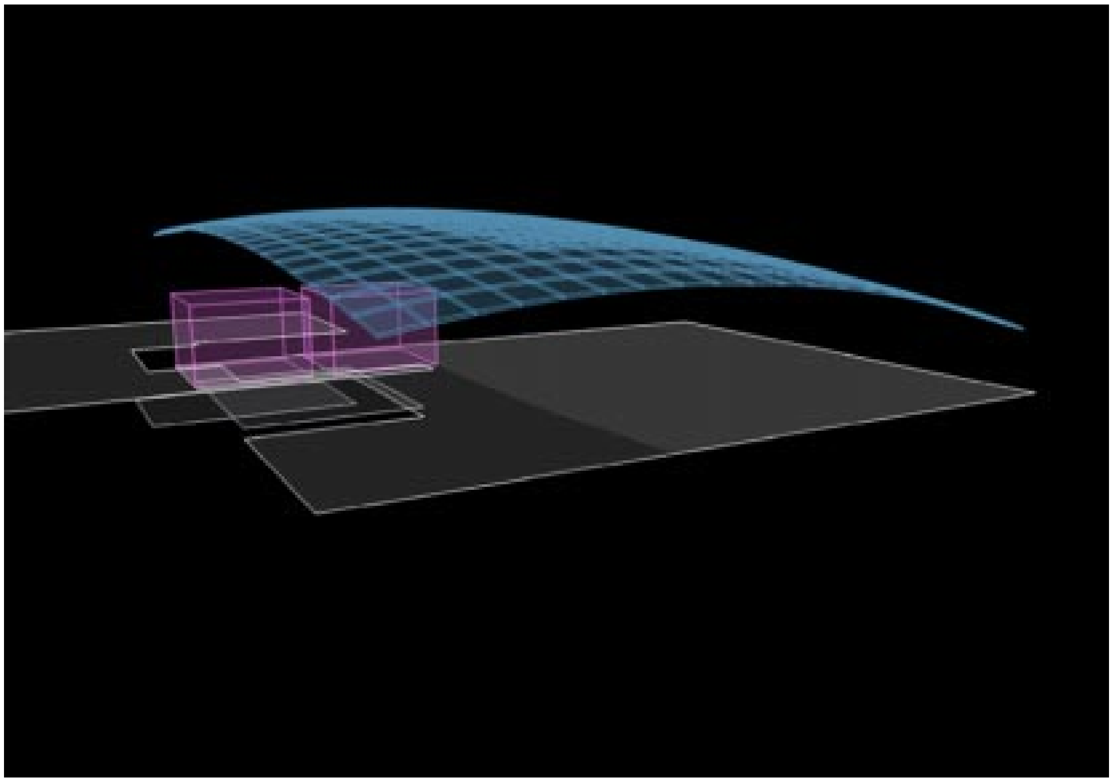


Figure 52. Aerial view of the final output of Hiroshi Okamoto's code, showing the mesh roof and underlying forms

to continue working in this manner through an Independent Study during the Fall 1998 semester, giving an indication of the time taken for this understanding to occur, and suggesting the best time for the fourth interview. Two students from the Spring 1999 course continued with the program in the Fall 1999 semester, along with a third who had taken Digitally Mediated Design I the previous year. A third student from the Spring 1999 group third joined the course as a Listener but due to other commitments was unable to participate more fully.

Between the in-depth interviews and three formal presentations – all tape recorded and precisely transcribed – an extensive range of data were collected. In addition, all of the conversations and lectures that occurred within the course were recorded. The course structure is a two hour lecture per week, with a four hour recitation during which time there is much free conversation between the students and myself. These conversations occurred in front of the student's work and formed a third source of information. However, it was not possible in the time available to transcribe and analyse every four hour recitation to the same degree as the formal interviews and presentations, which must be regarded as the primary and secondary sources of data. In addition to the lecture and recitation periods, students were required to work alone for a further six hours per week.

The research's aim was to use this data to demonstrate a development in design thinking as a result of taking the course. However, the interviews themselves are interventions in the students understanding – interviews measuring development in cognitive ability may cause cognitive activity. Although unavoidable, great care was taken with the early interviews that the questions did not introduce ideas the students had not themselves previously thought about. In the final interview, however, there was a chance to ask more provocative questions probing more deeply into the understanding the students had gained (see Appendix F). This provocative nature was clearly demonstrated by all of the students at one or more times either significantly pausing before answering, requesting the question be restated, or in several cases stating they had never thought of these things before and found the questions difficult to answer. However, having used the question to provoke thought, the students were then able to answer in a manner that appeared to be based on their experiences in the course. But another research question arises from this that should be studied in the future: what would the results be of asking similar questions to a group of students who had not taken the course? Could the students in the study have answered with such insight without the education of the previous year?

Quality and Credibility

The two halves of this thesis – the development of a novel and innovative approach to using digital media in architectural education, and the development of a research study into the effects of that educational intervention – are to a certain extent in conflict. The biggest single problem affecting credibility of this study is the issue of my gathering and analysing data on an intervention that I myself both created and conducted. Since the research study evolved, and the content of the course itself fluctuated in response to the students' requirements, it is possible to level the accusation of the initial findings influencing subsequent direction in the course. It is possible to imagine a course that would simply teach the required subject areas in a manner that would influence their understanding of these areas, and thus document an obvious change. However, it should not be assumed from this that the study is automatically unreliable. As Denzin points out, "Value-free interpretive research is impossible. This is the case because every researcher brings preconceptions and interpretations to the problem being studied" regardless of the methods used (Denzin, 1989 p.23). In addition, the course itself was not monitored. Rather, the students were taught a skill that might be used by them as a tool for design. It was the effectiveness and use of the tool, and the impact of its use on the students' understanding of design, rather than the course, that was being studied. The course focused on the students' project work, teaching them the programming skills required to complete the project, and then threw them in the 'deep end' of presenting such work to an audience predominantly comprised of architects and designers with little programming background. There was a careful and conscious avoidance by me of some key aspects of the discussion that had preoccupied previous students. This avoidance was referred to in three separate incidents. At the end of each semester students are invited to evaluate the courses they have just participated in through an anonymous evaluation procedure conducted by the Department of Architecture. On one form, one of the students wrote that he or she thought that not all information pertaining to the theoretical basis of the course had been given:

> "I would suggest more informal "talks" or reviews within the class covering each class members' projects as well as theoretical talk about the nature of the subject of the course re programming and its integration to the architectural design process, and digitally mediated design in general. This should happen as often [as possible] and perhaps in place of / in conjunction with, the pin-up reviews. Perhaps this is a little unfair since the method of Megan's research is to take students who are not familiar with programming or fields integrating computationally mediated design, yet for the student, it maybe helpful for Megan to talk over with them precedence, and

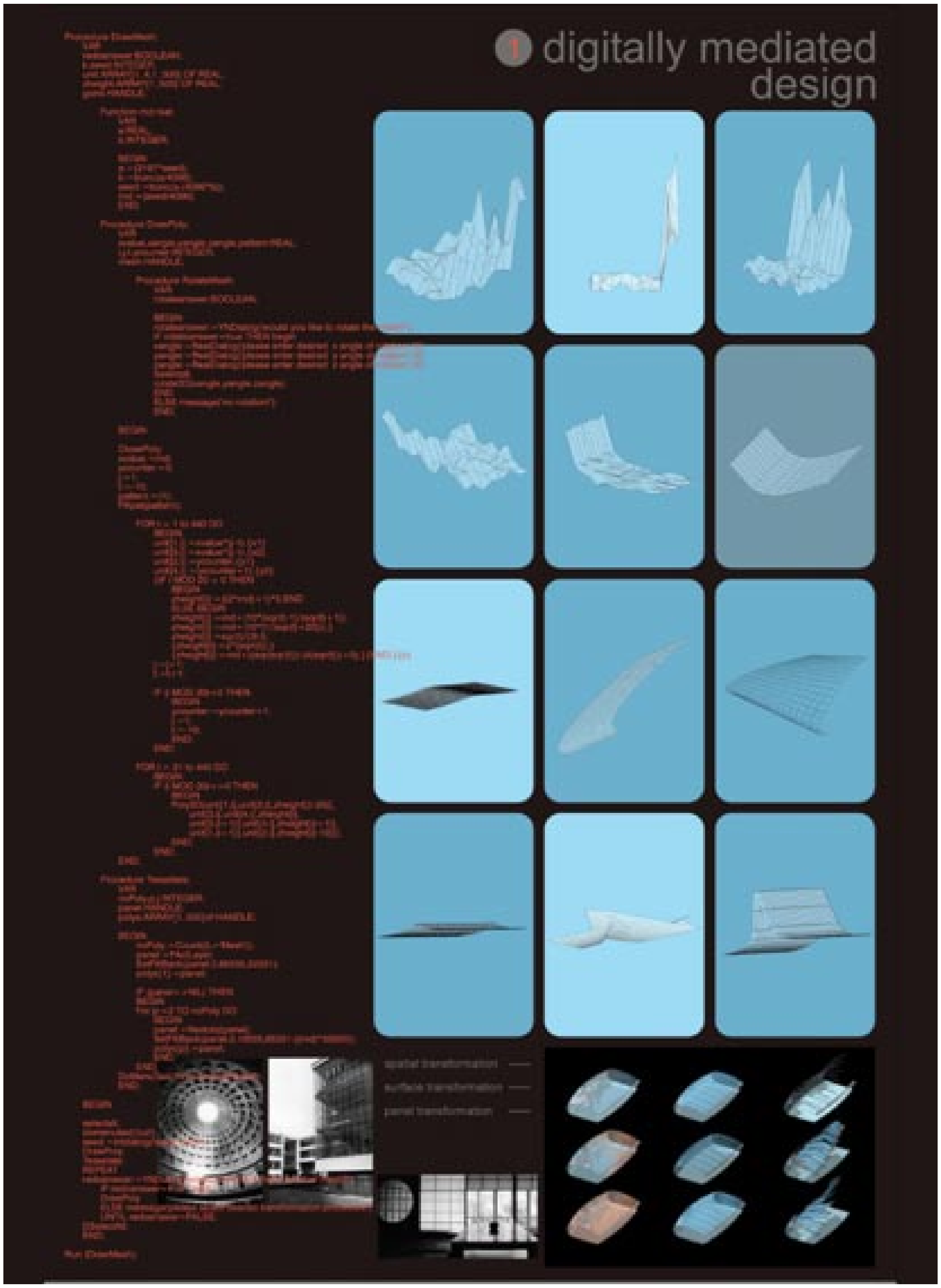


Figure 53. Hiroshi Okamoto's first poster, showing early versions of the mesh, the code, and architectural examples of the sort of glass skin design he was exploring

> theory from the beginning of the semester, immersing them with some background into a new attitude towards digitally mediated design. Overall through, I believe her patience and flexibility with each students' project was a big part of the success of each students' development." (See Appendix G)

During the second semester, following a crit of the second course, before the final interview, in which three of the original six students participated, a reviewer queried whether it was possible to teach the students more about the processes being discussed.

> "I think there is an opportunity for elevating the pedagogical techniques to be able to address these fundamental questions regarding the media during the class. Is there a way to frame the discussion such that the student is required to formulate a level of precision in describing the opportunities inherent in the use of the media? Maybe they are already doing this but I did not hear very much of it yesterday."[5]

Finally, in the week just prior to the final interview, an article about my course written by one of the students from a previous year appeared in the departmental student magazine, in which she refers to concepts in the course that had not been discussed with the Spring 1999 group.

> Taking Megan's class for me was a real eye opener. It was a test of how computers could be used in capacities far exceeding their traditional roles as representational and visualisation tools, and also an opportunity to try and understand my own design process. The point was not to become an expert programmer, but to use coding to explore the formal and thematic preoccupations that drive each of our particular architectural inquiries. Megan often likened the process to a kind of digital "sketching". The class was also invaluable as a primer in understanding the role of "rules", "grammars", or "systems" in architectural design. It was useful to see how a rule influenced a formal outcome, but far more interesting to learn that while the computer's ability to rapidly produce multiple variations was a powerful asset to the designer, real value only emerged from the process when the designer (that's me!) re-entered as the decision maker. In other words, you can code till the cows come home, but it doesn't make a better barn, unless you're able to stand back and critically reassess each barn from the standpoint of wanting to make the Best

> Possible Barn. I have used what I learned in my classes with Megan in studios since then, and I think her teaching is a valuable basis upon which to found an understanding of the rapidly changing future of design. (Auer, 1999)

In that final interview, all the students were asked if they had read the article – none had – and when raised, they expressed interest in the novelty of these topics. Although anecdotal, all these incidents support the success of the separation between the actual content of the course, and the higher order metacognitive skills studied by the research.

Another concern of the research was the students' willingness to participate fully in interviews conducted by the person - myself – who was additionally both teaching the course and more importantly evaluating their performances at the end. This concern in part was unfounded – students were reasonably unconcerned about their final grade, and were happy to participate in the interviews. They appeared to enjoy, and in several cases specifically stated their enjoyment of, the chance to talk about themselves and their work. The process of interviewing – being tape recorded – affected at least one student in the interviews, who expressed a nervousness of the process that lessened with each interview. But the fact that no students were concerned with a link between their participation in the interview process and their work in the course might indicate further support for the separation between what was being taught in the course and what was being researched in the interviews.

Data analysis

Each of the interviews and crits were transcribed word for word, a laborious and expensive but necessary process. Evaluating a transformation in understanding as a result of an educational experience can only be done adequately through an analysis of the words the subjects use in their descriptions of understanding. The way in which they use language gives the clues to the mental models they have developed to support their understanding of the processes being researched. Subtle changes in language can show how these models are changing. These transcripts were then analysed. Strauss and Corbin (Strauss & Corbin, 1990) describe three approaches to data analysis that vary along a continuum from a low level of interpretation, to a much higher level required for theory building. This study took Strauss and Corbin's second approach, with the recognition that some interpretation of the data was necessary, but where the primary aim was to describe the events of the study in as accurate and clear manner possible, using quotes from the participants own words. It is an "interpretive descriptive" approach (Maykut & Morehouse, 1994, p.123).

The actual analysis of the data used Glaser and Strauss's Constant Comparative Method (Glaser & Strauss, 1967). As its name implies, this method of data analysis involves the categorisation of the data into individual units of meaning, and the comparison of each of these units with all of the categories and other units. This search for meaning began with the words of the students, and used the framework of the questions outlined above. After the interviews were transcribed, these units of meaning were identified. A possible definition of a unit of meaning is that each unit must be able to be understood in isolation, and its essence should be describable in a word or short sentence. Each unit was then separated from the data, coded, labelled, and grouped. As each new unit was analysed, it was compared with all the other units and grouped with those that were similar. When there was nothing similar, a new category was formed. The comparisons are made both within and between categories to ensure internal homogeneity, and inter-group heterogeneity. Lincoln and Guba describe the process: "The essential tasks of categorising are to bring together into provisional categories those [units of meaning] that apparently relate to the same content; to devise rules that describe category properties and that can, ultimately, be used to justify the inclusion of each card that remains to be assigned to the category as well as to provide a basis for the later tests of replicability; and to render the category internally consistent" (Lincoln & Guba, 1985 p.347). It is a distinctly inductive method. Although my analysis of the data was framed by the questions formed from the previous years of teaching, I did not generate formal hypotheses a priori, nor did I develop predetermined categories to group the data. The subsequently created groupings then became the tool of analysis. The aim was for the categories to suggest and support meanings and ultimately hypotheses to explain the phenomena observed.

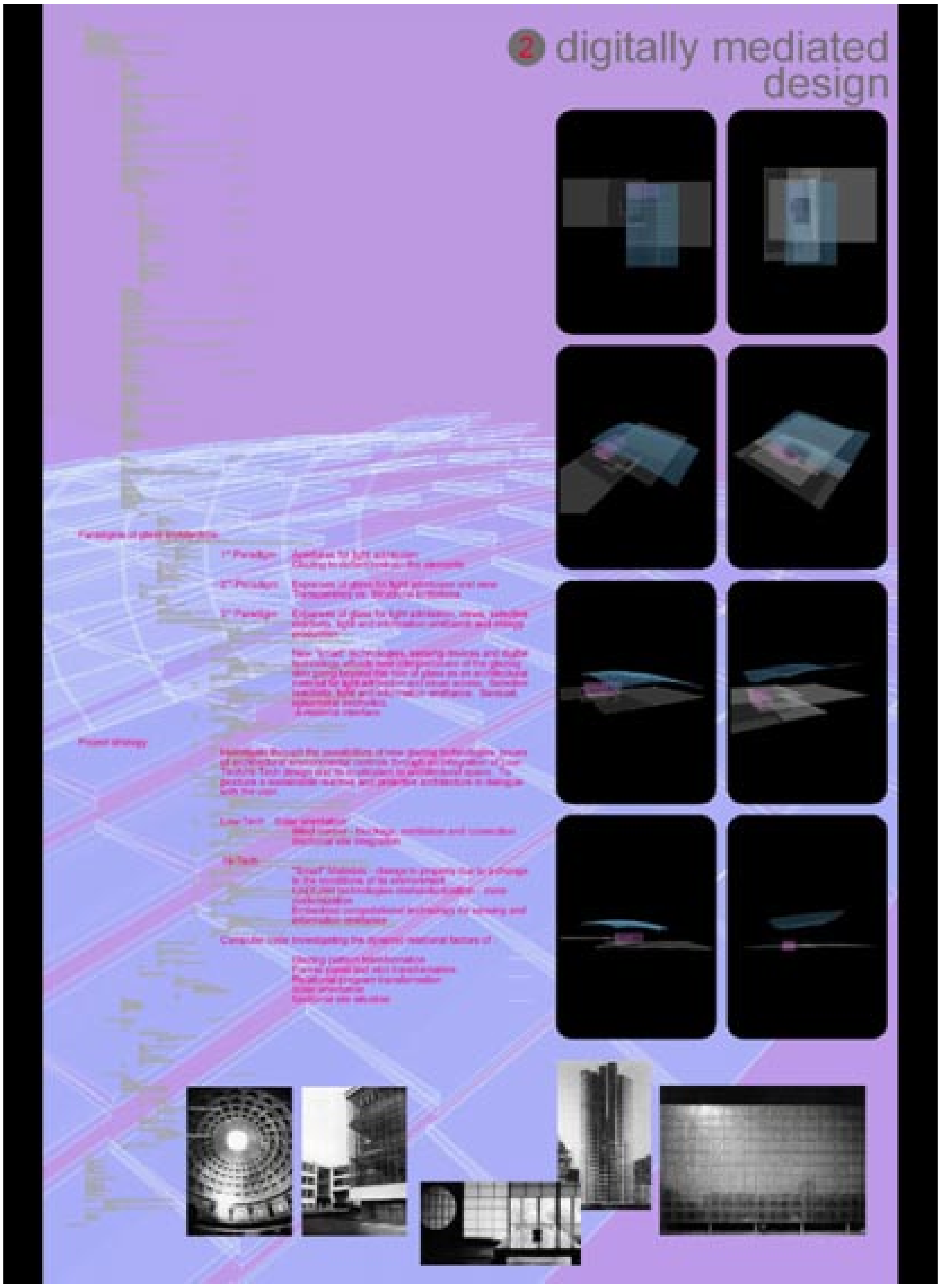


Figure 54. Hiroshi Okamoto's second poster, showing the final design, the code, and more architectural examples of the sort of glass skin design he was exploring

Results of Research Study

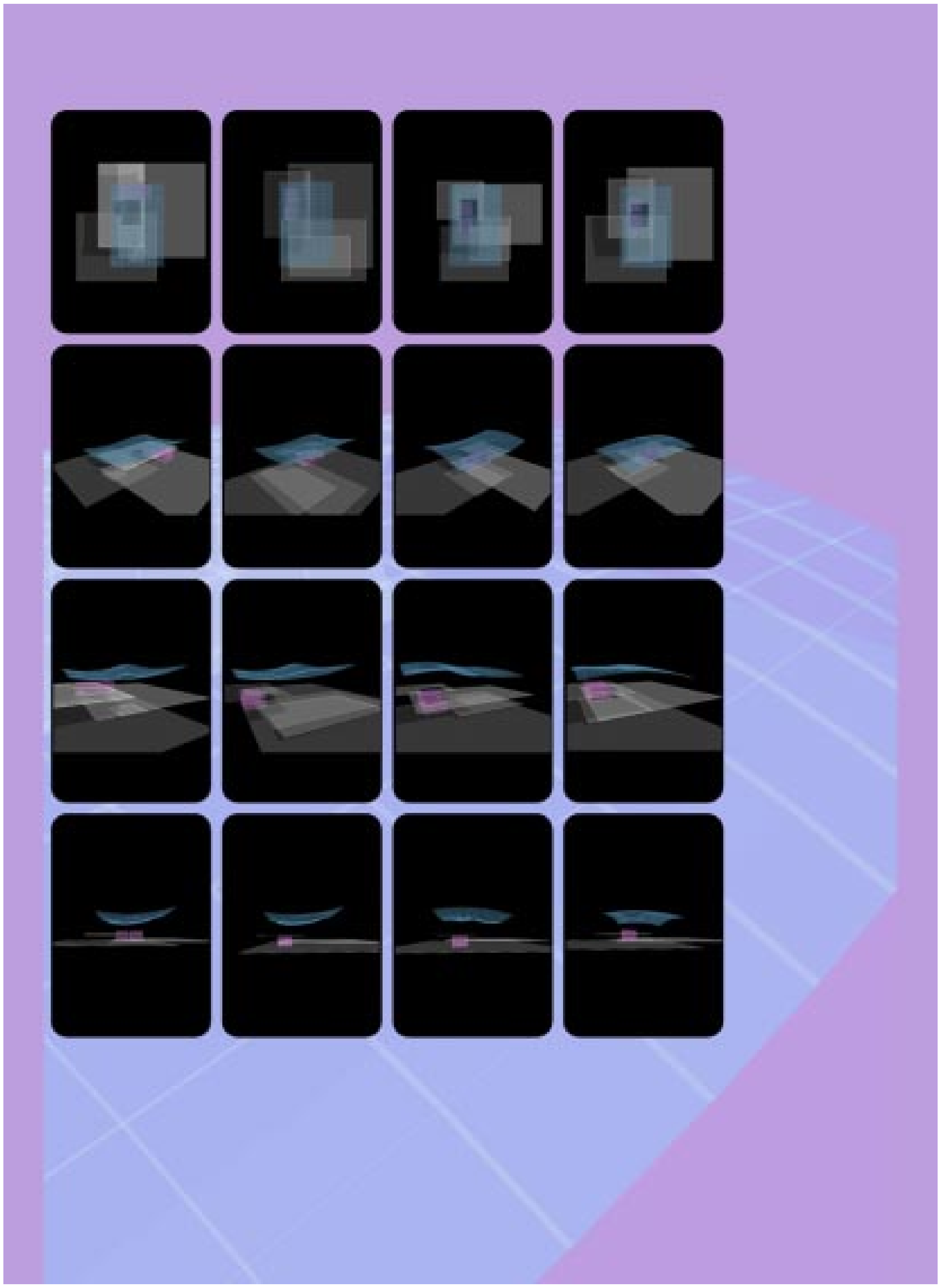


Figure 55. Hiroshi Okamoto's final poster, showing the variations produced by the code

Development of a design process

One of the main aims of the research for this thesis was to demonstrate how the course developed in the students an awareness of the cognitive skills required for a successful design process. These cognitive skills are utilised by the four step cyclical process described at the beginning of this thesis. The ability to create and use external representations of the design idea at any stage, and use them as objects to think with for the next emergent rule, is a skill mature designers do unconsciously, but one that must be learned by the novice.

Level of Understanding at Start of Study

The first stage in the research was to establish what understanding of the design process the students already had. Since there were no undergraduates in the course, and all of the students had experience of being in more than one design studio, it is perhaps not surprising that most of them were able to refer in some form to an awareness of a design process. Gary had the clearest description of his own process at the start of the semester, what happened and why, what at the major influences were, and what helped it to progress. He was also the only student to persistently use a pencil and paper to explain what he was talking about throughout the interviews. Yet as with the other students, Gary's ability to describe his personal design process in abstract terms, away from describing the decision making process in any one particular design studio, was limited. He was able to talk in general terms: "you know, you analyse something, you break, you find the oppositions, and then you play with the possibilities, ... and the similarities and differences." (1a.InterviewGary18.2.99/US/p.4/l31-p.5/l1) but when pushed could only describe an individual project in terms of the series of steps involved. This was true of all the other students; their descriptions of designing invariably centred around one particular project and took the form of a list of sequential moves.

> [When I] started off it was a drawing, [I was] thinking about the notion of time and waiting, and things like that, and trying to map graphically ... these two different elements, this ... rhythmic time of the clock or the train schedule, ... [and] interactions of yourself waiting for the train ... [So] it got broken up into two, let's see, do I have any paper here? ...[T]he diagrams look ... like this [drawing whilst talking]. And with one tick, real time and not so real time. ... And, so the actual ... bench became two different things with two different elements. The person sits here ... there's sort of three of [still drawing the whole time] these systems ... graphically it's kind of displeasing the relationships between the time analysis and what actually I did. (1a.InterviewGary18.2.99/US/p.4/l7-24)

Where they were able to reflect at a slightly higher level on their process, it was in terms of describing similarities in this sequential list between different projects. Both Gary and Hannibal expressed awareness of a process that existed independently of any one individual project. Both referred to time taken to consider the similarities and differences between different design studio projects, a sophisticated understanding they are unlikely to have had any structured support in developing. It must be remembered, however, that the high quality of students at MIT gives an available population of highly motivated students.

> The only … reason why I can talk about this so clearly is because I've spent a whole month sort of thinking about this … since the beginning of this last semester. I mean if we had this conversation two months ago I probably wouldn't even talk about it in these terms. But, after the first semester of graduate school, reflecting on the five years prior to this, and, you know, and then this (1a.InterviewGary18.2.99/US/p.5/l36-p.6/l2)

Raphael was able to refer to his own process but purely in terms of the external structure. At no stage did he describe his internal cognitive activity or how an idea developed from one stage to the next.

> I think that … design is problem solving in a sense, and that's a top down kind of attitude, but, you always have to have an idea of what you're trying to do, I think and it doesn't necessarily have to constrict you in terms of how you go about it, and so there's a period when you always have to test your ideas, I think, and throw away certain things, and take up other things, and refine it, so, yes I think in one sense I'll get an idea, I'll look at, hopefully I think, I like to throw out all my thoughts and kind of leave them on the table, and it doesn't necessarily mean I'll always use those thoughts in a certain pattern, but at least I'll have them there, so, there's more facets to the testing, and, or the ideas, and then it might be a process of creating, I don't want to say creating tools, because this is what I've been talking about, but it's, it's about, creating ideas, and testing those ideas, and, it could be it's really, that's the design process, I think. (4a.InterviewRaphael19.2.99Div/US/p.9/l1-13)

Raphael did have the most sophisticated view of a general "process" and what it might involve, but spoke very little on his own method of working, beyond describing its as a process of "testing" and realizing why things are "not working". Like Raphael, Jeremy was also clear about the external steps taken in his design process but gave no description at all of what he was thinking about or where his ideas were generated from. At his previous University he had been consistently exposed to a series of studios with a strongly dictated methodological approach.

> I mean, it's really teacher run at [my previous University].... I mean, like the, the teachers have a lot of say. I don't know, I think it's, that's the bad thing about [my previous University], I think, and the good thing. That's very, you can see the teachers very much through the students' project work. [The work that's produced visually] very similar. I mean there's the [University's] Style which I think anyone outside [my previous University] can see in about five seconds. [laughing] And it basically drives you insane by the time you get to kind of the third year of, you don't want to see [it] ever again (6a.InterviewJeremy23.2.99Div/US/p.5/l18-29)

This may be the influence behind his emphasis on external steps in his descriptions, since it is clear he has not been encouraged to develop a more personal approach. More strongly than any other student, Jeremy placed little value at the start of the semester on the cognitive skills developed in learning to design. For Jeremy, the most significant aspect of his architectural education at the start of the course was the acquisition of useful facts and knowledge, a Behaviourist understanding of education.

> [Y]ou come out of university, and we don't really know anything still. And that's kind of a problem. You get to practice, and you're useless to anyone until another year and a half when you start to pick up some knowledge, and you're actually useful to a client. And ... I think architects just need to get out of this, ... it's really quite annoying. And if we actually had some useful information, like everyone else seems to manage, some bit of information, I mean we come out with ... drafting skills, which is good, and now ... computer skills, but we need to come out with things that we can help people with and tell people about.
> (6a.InterviewJeremy23.2.99Div/US/p.9/l20-29)

Josephine had a similar exposure as Jeremy to strong methodology in studio, but at her undergraduate school this took the form of a heavy emphasis on a somewhat abstract notion of "Process" as being a black box skill that needed to be developed (Coyne & Snodgrass, 1991), although there did not appear to be one particular method promoted for acquiring such a skill.

> Yeah, because [at my previous university] it was very much about the process. It was very much like, alright, you go home and come back with some kind of generative abstract mapping or whatever, you know. And it ... couldn't be something that was ...

> literally [taking] the building form. It had to be something that … you would assign symbols to or something. Like, my long line represents emotion and my dashed line represents, you know what I mean? And these are the layers of the site, and so there was always this, what is your next rule that you set up for yourself, and the next thing? But here I haven't felt like that at all. … No, I actually ever been a process. Maybe they told me what to do in terms of oh, you should make a model next. (3a.InterviewJosephine21.2.99Div/US/p.11/l6-16)

It appeared at the start of the course as though this was something that had always remained external to Josephine; she had not appropriated the term "process" for anything that might be in any way personal or under her control. Listening to her words in the first interview it is possible to hear all of the relevant information – she knew about process, about the tools of process, about its role in architecture – but she appeared unaware of her own knowledge or at least unable to put it all together and take advantage of it.

Esther was the most reticent on the subject of personal process, although at the beginning of her semester her English was poor and it was not clear whether this reticence was a lack of understanding of the question or ability to express an appropriate answer than any difficulties talking about process. She described projects she has done for clients in the past in very general terms, and mentions some external influences but did not elaborate. Since she chose not to talk about her undergraduate experience, and since her two studios at MIT have been radically different enough both from each other and from anything she has done in the past to make them unique, there was little subject matter in the first interview upon which to draw her to give any indication of what more general process she was aware of in her project work. Yet it was Esther who offered the clearest example of the benefits of the course in developing an understanding of the design process.

Understanding the Design Process: Five Stages of Development

The understanding of the process of designing that Esther and all of the students eventually achieved, occurred with the exception of Jeremy after the end of the semester. Although Jeremy expressed his awareness earlier, a period of time for reflection was required for the rest of the students. From the very similar process of development of the students, it is possible to see that the development of metacognitive abilities of reflection on the design process has several earlier stages.

Stage 1: Imagined Future User of Final Code Product

In the first stage, the aim in the students mind is to develop software that will be functional for another person. In this stage the development of rules is something that occurs a priori to the coding process. For many designers, the term ‘rules’ implies constricting factors, dictated by external forces, limiting options and inhibiting design development. These are “imposed rules”, and unlike ““emergent rules”” neither promote nor support the design process. Much of the early discussions in the course focused on either the rules as separate entities, or an imagined third party future user. One student, Raphael, began the semester by trying to program functions into the code such as the ability to rotate a three-dimensional object, replicating what was already available to him in the CAD environment. Another, Gary, started out by linking his project for the course with his studio project. The studio’s emphasis was on the exploration of rule based design, but they were strictly Imposed Rules. A couple of weeks into the semester, however, Gary had a crisis where he realised the limitations of his approach but didn’t understand how to change. He did, however, successfully make the transition to a project with a less predetermined outcome. In the second interview, he reflected on this change of direction:

> [Originally] I made this thing that made a whole bunch of boxes, and that was really it. And ... there were bits and pieces of this stuff that were in it. You saw the thing that made this, I mean [VectorWorks] made these things ... so there was just a whole bunch of little exercises loosely related to this, probably more related than I like to admit at the time, because in my own head they weren’t quite so related, but they were. And ... so originally it was just all these sort of bits and pieces that didn’t do anything for each other, and didn’t do anything, ... you, you watched me struggle through this. ... And, and, this being like, well this doesn’t really go anywhere. This isn’t really doing anything, ... it was just a machine ... but as far as being, getting at why, something else, it wasn’t really so much. And, you know and that came out [when] we talked about it a bit and then ... I actually realised that’s true and because

> to me I was writing this code and it did this cool stuff by itself, and to me that was being successful to some degree, but, as a, on a larger picture it wasn't really getting at anything too, too great. (1b.InterviewGary4.4.99/A4/p.15/l5-29)

Much of the initial enthusiasm of this first stage surrounding the desire to create software that will be useful for another person, is based on naive and unrealistic expectations of what a designer turned novice programmer can achieve. As the reality of the difficulties involved dawns, the student begins to see herself as the future, but still not current, user, transitioning to the next stage.

Stage 2: Student Sees Herself as Future user of Final Code Product

The product in this second stage is still seen to be the computer program. In the second interview, Raphael had begun to reflect on the starting point for his project, and in so doing revealed his position between this second stage and the critical third stage of understanding that a more significant focus of the process is the design project rather than a computer program.

> I had it to a point where I thought of the tool as being the end project, the end code would be the tool, but it's not really the end code, I think now more and more. I think it's more about, well of course there's the end code, but you have to get to that code, by finessing your code in between, so I think it's more about testing certain possibilities. I think ... more and more I'm realising that, a lot of the times you can start programming to test quickly certain ideas, and use that and go back to your design and use another tool, to develop that, and not being enamoured with the whole tool ... the whole tool, code. (4b.InterviewRaphael2.4.99Div/A4/p13/l5-12)

This realization of Raphael's indicates the beginning of change in attitude and understanding. He is recognizing that the role of programming is to express his ideas, examine the results, and make further design decisions from them – to form ""emergent rules"" for the next coding process. Most students assume that computer programming can only be used to make quantifiable decisions; the understanding of the significance of "emergent rules" is supported by an understanding that it can also help in the development of qualifiable design decisions

> I've put my finger down a little more. It's forced me, because you have to know what you want to do, because you have to code it, to think about OK, it's not just intuitive, I can pen some concrete things about what my intuition tends to do, so I can chart it more. Whereas before, it's almost like ... keeping a diary of my design habits. ... I thought all along that programming equalled analytic, quantitative, something really,

> just alien, and I never thought of it as more a qualitative tool, more a thing that I could just tweak a little, and it didn't have to be about being an exact quantifiable thing. (3c.FinalPresJosephine10.5.99Div/US/p.11/l12-22)

Stage 3: Focus of Process is Design not Code; Emergent rather than Imposed Rules

Understanding the limitations of Imposed Rules and appreciating the generative qualities of "emergent rules" is key in the third stage of development. In this stage the significant product is the design project, not the code being written. In the second interview, Hannibal articulated the benefits of the programmer also being the user.

> So then I think the further removed you get from the programming language the worse off you'll probably end up being. You'll use it, you use the program and you use the end result to do something, but you're more or less using someone else's rules and someone else's ideas and not understanding how those ideas or how those rules were formulated, or in what context, or how they're used. So you'll have almost no interaction then with the program. It's just, I mean it's a video game, it's you interacting with the end result to do certain things, and it doesn't progress much more than that. So I think if it results in programs, it's probably not beneficial. I think if it results in people programming, I think it's much more beneficial. (2b.InterviewHannibal5.4.99Div/US/l28-36)

Stage 4: Recognition of Parallels Between Coding and Designing Processes

As the student begins to focus on the design project, an ability that results from increased skill and level of comfort in writing code, so she moves towards the fourth and penultimate stage in understanding her own design process. With the increased ability of coding, comes a reduction in focusing on individual commands and an ability to view the constructs involved from a higher level of abstraction. She becomes aware of the cyclical nature of the coding process and simultaneously aware that she is developing a new way of thinking. The awareness of this cyclical process, and the individual steps involved in it, results from the need in programming to explicitly and unequivocally state every intention. Jeremy was the first to express an understanding of the significance of the rigour involved, and in that the final review said:

> so the question about what I've learned from doing this, I'd say that ... even though these are things, ... if I probably sat down and tried to work out what was processing and going on, I could think it, but by writing it in this way you have to be so rigorous

> about what you're doing and what steps, you suddenly realise that you can't have this happening first, you've got to have that happening first. And it's basically that rigour which I think is very useful, that's actually, I find useful for taking into other classes. It's that rigour of, even though it's something you do intuitively, there's so much because we learned it like ten years ago that it's just become part of our background. It's quite good to get that rigour back and see what see what steps you need. (FinalPres10.5.99Div/US/p.34/l8-16)

This awareness was already formed before the end of the semester; by the second interview Jeremy had a sophisticated understanding of the aims of the course:

> I think one of the main things which is really good about suddenly coding is that you have to be really rigorous and logical about steps you've taken and where that links back to other steps, and I think, I don't know, it's actually affected the way I actually do things like write papers, and studio. ... Like, it makes, it makes you look really rigorously at every step, and how that step connects with that step, which it's so easy to be not rigorous, because there's a sort of looseness in things like writing text, or designing. And I think it's really good at making you very aware of the language that you're using, and, about how each step is kind of connected to something else. Basically reading things, reading the language much more rigorously. (6b.InterviewJeremy28.4.99/US/p.14/l1-14)

> I think it's quite good training, especially in someone who's not computationally minded, that it's quite good to force people to get a subject which they have to be rigorous about, they have to suddenly question themselves. They have to continuously question why they're doing it. Because there's so many things that's so down the line, that you've got to continuously question why you're going forward. Which is really healthy in, and that practice is just is so useful for just the way you do everything, the way you read, anything. ... It's really healthy way of questioning yourself all the time, and making sure each structure makes sense with the next (6b.InterviewJeremy28.4.99/US/p.15/l22-32)

This "healthy way of questioning yourself all the time" is a critical skill in the design process that must be learned, since design is a constant process of inquiry. The course supports the development of this skill by slowing down and explicitly expressing the stages involved, through the 'soft' tinkering style of programming taught. In the first interview, Hannibal saw this need to slow the process of designing down as the primary role of a professor and a significant aspect of learning to design:

> whether that's the professor, or whether that's another student, they bring out some of the latent ideas that you can see, and then you can start to explore those which otherwise you may have just done, and you know that you're doing for a certain reason, and you really don't think about it too much, it's just part of the process, but if they can pick out those certain things, that just gives you those avenues in which to explore things. It may sound bad, but I think their role should be almost to slow down the process, and to try and make you go the different areas, and to look at different things, instead of push you along, you know, towards the goal, towards the final presentation. (2a.InterviewHannibal15.2.99/US/p.9/l1-8)

Stage 5: Transference of Knowledge and Understanding to Design Studio and Beyond

The understanding that the real focus of the course is the development of a new way of thinking is the necessary precursor to the fifth and final stage of development: the ability to transfer this understanding into the design studio and use it to understand the student's own design process. Understanding and therefore controlling these steps gives the student greater power in utilizing her knowledge – the definition of metacognition. Raphael, an experienced student with a well developed but largely implicit sense of his own design process, recognized the significance of what he learned in the course and expressed it in the fourth interview:

> And you think about setting about even just simple relations that can start to generate unbelievable possibilities. And I think it's the amount of control that you can set in terms of controlling yourself and controlling your program. I think it becomes a powerful tool but I also think you need to be a – even a better designer, in a sense ... if you're going to use it.
> (InterviewRaphael12.11.99/A4/p.9/l1-7)
>
> He continues: "And I think it's good because it slows down the process in a sense, [so] that you also have time to reflect along the way." (InterviewRaphael12.11.99/A4/p.9/l29-30).

For Raphael as with other experienced students, this reflection was an opportunity to understand an existing process of thinking rather than developing a new one. Here, in the final interview, he recalls his original aim of learning about digital tools, and reflects on the realisation that he hasn't just learned about them, he's learned about himself as a result of learning to use them:

> I wanted to, before I came here I wanted to look at some of the new digital tools that are coming out, and seeing how I can use that for designing, in terms of architectural artifacts. And I'm not necessarily interested just in that, but I noticed the way I used to design before was so regimented, in a sense. Not regimented, but it was – I mean, people have worked just in sketching, you know, and I'm not saying that that is regimented in a sense, but I also feel that there are new tools that you don't necessarily apply the way you design to it, but you also adjust the way you design. And in that sense it was a revelation to me. Now I can learn something where I can hopefully well use and not trying to fit the way I work but to really adjust to ... the possibilities of other things. That's why I think I learned more about how I designed. (InterviewRaphael12.11.99/A4/p.13/l25-35)

For mature designers, the course creates a framework with which to examine their individual design processes. It is a 'framework of difference of similarity' and requires metacognitive abilities of reflection. In the fourth interview, Hannibal describes the change:

> [Now] I probably have a better understanding about the way I'm structuring the beginning part of the project and where it's going to lead it towards, so again, ... I understand the fact that doing certain things at the beginning is going to limit certain things as I go through it towards the end (InterviewHannibal11.11.99/US/p.12/l1-3)

Hannibal attributed this newly found ability of reflection and understanding to the techniques of coding he had learned. He was able to transfer the knowledge implicit in the technical skills learned back to his design work:

> So I think just because of the way you have to understand how the actual structure of the code worked – as to this had to be determined first, and then this worked back upon that, and this was back upon, and all the loops and how that structure actually worked within the overall structure. I think that just was ... made very explicit to me ... So I think, again, it probably was just a re-evaluation of the design process, and as – yeah, I think that's good. (InterviewHannibal11.11.99/US/p.12/l3-14)

All of the students developed this understanding and reflection, but the time taken to do so varied considerably. The most extreme example was Esther, who like Gary chose to link the project she did in the course with her studio project. Like Gary, she was also in the design studio exploring Imposed Rules in design. But unlike Gary, Esther was unable to see the limitations of this approach during the semester. Her project was concerned with housing layout. The rules she chose were on the whole concerned with adjacency – which spaces had to be next to which other spaces – and the more rules she created the smaller became the available options. Despite continuous encouragement during the semester, Esther was unable to develop an understanding of the importance of "emergent rules" in the design process until the fourth interview, six months after the end of the semester. She forms an interesting case study.

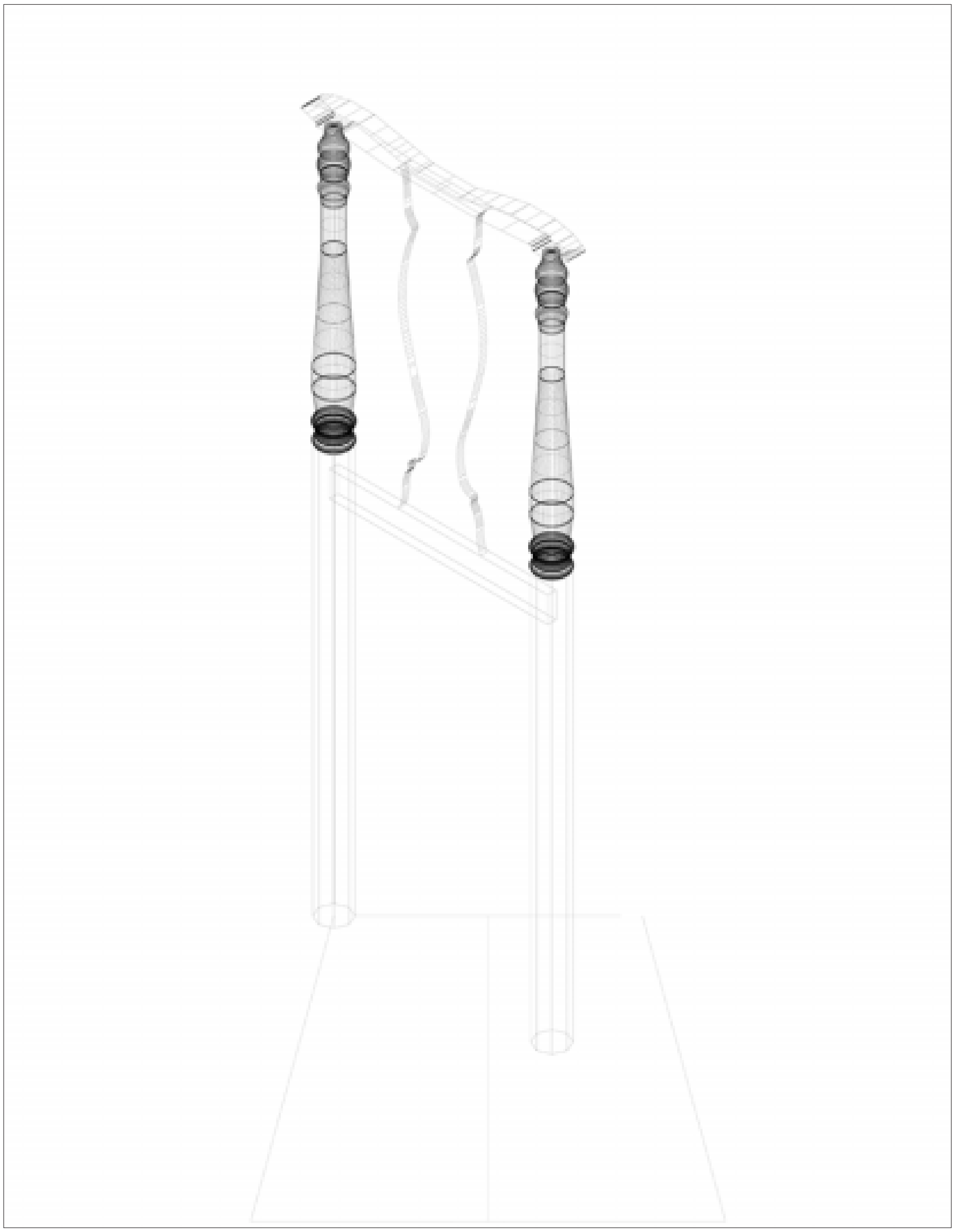


Figure 56. Henrietta Green's previous experience with AutoCAD strongly influenced her approach to using VectorWorks to repeat the procedures of the former in the latter. The resultant file size was very large

Case Study: Developing a Design Process

At the beginning of the semester, it was very difficult to ascertain Esther's view of her own the process of designing. When asked about how she worked before coming to MIT, it was clear that the primary generator for the decision making process were factual quantitative sources of information.

> But when I work, basically the client give you a site, and ... it was always like an emergency or something. They wanted it rapidly, very soon, just give the site and give me a piece of paper saying what kind of function they want, and how many square meters of what room, or what room, or what room. ... And basically I just use the site, and analyse the context, and arrange these function to the site very quickly. Maybe they only give me one week or two week to do that.
> (5a.InterviewEsther16.2.99Div/US/p5/l9-16)

When asked about where at the physical form of the buildings she was designing was coming from, she spoke of this in much the same manner of a external decisions applied to the form.

> [Developing the physical form] has always been my problem. Because I don't believe in just, you know, function, I believe a building could be any form. So, every time when I have to design about a form, I really am never sure. So actually I did some experiments ... Once I ... just for fun designed a house for three ... artists. It begin with forms and fit the function into this form. So, [I] just want to approach a house from another way. And another time when I was doing a design in a housing project I use ... different layers. Design different layers independently. And hopefully these layers try to find something interesting. Because these different layers interact.
> (5a.InterviewEsther16.2.99Div/US/p5/l20-30)

This somewhat random nature of her external ideas, taken and applied without recourse to earlier decisions, was repeated in her approach at the beginning of the semester in her design studio, where they were beginning to study the concept of emergence in complex systems.

> We just use a natural phenomena, as a example, and try to extract simple rules from this phenomena, and view this rule, this very simple elements to create some complicated 2D or 3D pattern. That's what we are doing right now.
> (5a.InterviewEsther16.2.99Div/US/p1/l24-26)

Having chosen coral reef growth as her first idea Esther then rejected it through lack of clearly defined rules.

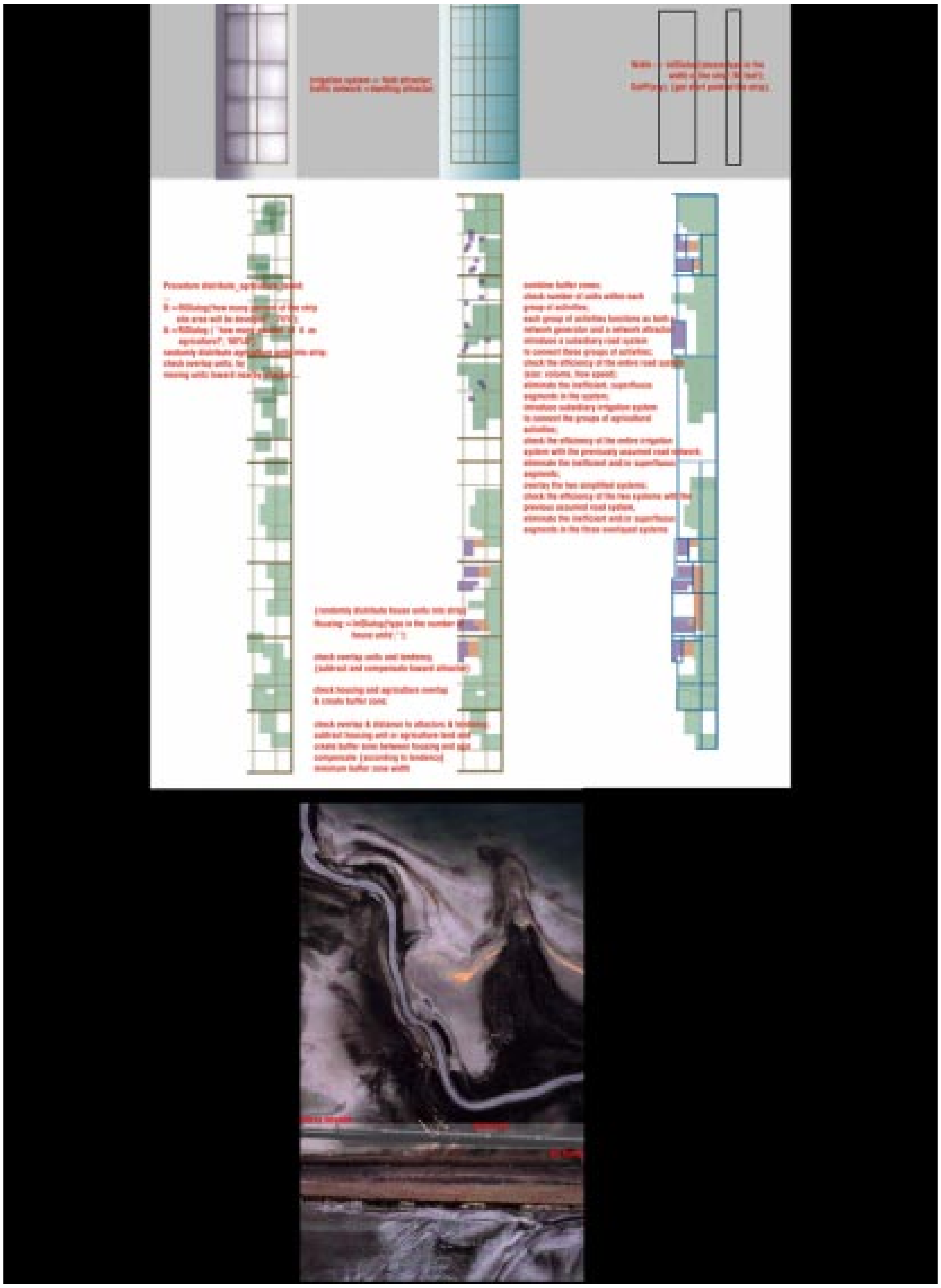


Figure 57. Henrietta Green combined the work she did in the Digitally Mediated Design course with her studio project. She was in the Rule Based Design studio, and her exclusively Imposed Rules reflected this

> [M]any many small coral polyps form this whole coral, a small coral tree, or coral reef. The individual is very small and very simple. But the society, the coral society is very complicated. ... With other reef, complex form. But first is very difficult to abstract very simple rule from that because it grow in the sea, the ocean, there's many factors will effect the form of this coral tree. ... And so I try to find something, with a very clear rule, and now I'm using colour blind as a example, ... colour blind people. Because about the chromosome, is very that's a very clear rule. If people have a X chromosome, [this] will determine whether he or she is a colour blind people. (5a.InterviewEsther16.2.99Div/US/p.2/l17-30)

Esther's desire was to find a 'rule' that could readily be modelled computationally to produce a largely predetermined outcome. The consistent theme with Esther's description of designing is a somewhat random search for an external source of ideas to support her decision making process. Conversations with her in the early part of the semester centred on this search. It was not possible to gain any deeper understanding of Esther's decision making process, and Esther herself appeared largely unaware of her own cognitive activity. In jumping from one source of ideas to another, she exhibited a highly Behaviourist approach to education – an 'all or nothing' model. She was aware that architectural design is more than simply a sum of its factual parts, but remained firmly entrenched in the belief that the answer lay somewhere outside of herself. Twice during the first interview she mentioned being curious about choosing forms for architecture.

> [My studio professor] pointed out to me that, he ask me, how do you choose this form at the very beginning? ... I said, I don't know. But I told him, shape grammar, at the class last semester, and I was wondering the whole time, you know, you create a lot of forms, how do you choose which one you use? And well how to really put these functions into these selected forms? (5a.InterviewEsther16.2.99Div/US/p8/l10-17)

There did not appear to be any possibility in her mind that an architectural idea might evolve over time, appearing instead to view the decisions as fixed and predetermined. This firmly held belief inhibited Esther throughout the semester in her development in the course. It also made her susceptible to, and was firmly supported by, the processes involved in her design studio. This studio had a particularly strong computational basis, and students in the studio were exploring the nature of rules in architectural design. Yet unlike the course where the emphasis is on the development of "emergent rules" to support the design process, this studio was exclusively concerned with the development of externally defined and predetermined, Imposed Rules.

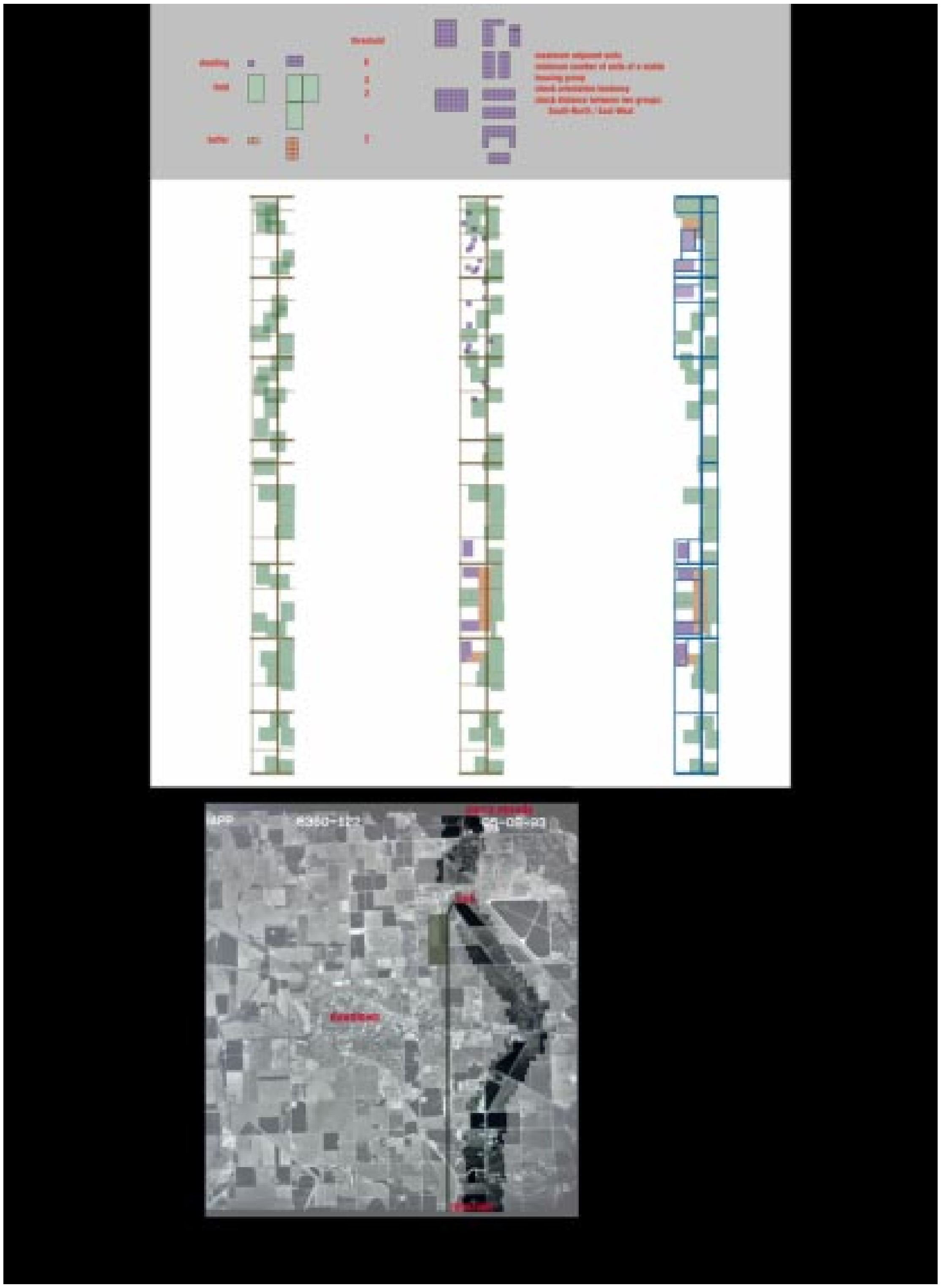


Figure 58. The rules Henrietta Green was using looked at adjacency of spaces in dwellings. As with all examples of Imposed Rules, more rules means a reduction in the possible number of solutions

> Here this analysis [is about] how these elements react to the site and the natural conditions, and also ... this site, these elements react to other elements. This is accessibility, and at the moment, analyses. These elements need to be more accessible ... this is more private so is not need direct connection to the site entrance .... I think the site entrance is sort of like a generator... The garage is very constrained because we need a road from this site entrance towards, into this garage, so this went first. And the living room, dining room, the second, and etcetera until the most private. There's rules such as none of the other elements should in between the entrance and the garage, and the kitchen or, bathrooms which not required to many sunlight, it should not be in a position where the sun light is very good. It should not be in a better light condition, location than that of these kind of room.
> (5b.InterviewEsther4.4.99Div/US/p.3/l1-17)

Esther's predisposition to this approach of using predominantly Imposed Rules, combined with the significance any design studio has in a students life, made it impossible for her to develop any understanding of the role of "emergent rules" in design, and therefore her own design process. Esther's final presentation in the Digitally Mediated Design course was particularly antagonistic. She was asked several times by the crit panel why she had not used her own work as the basis for the next design decision; why each step was not emerging from the products of the previous one.

> Fernando: My question is how you are programming that that decision? ...
> I think I established this decision before I did the code. I was just thinking how many different spaces in [a house], what they are, whether they are more public views or private views, or what they, so, based on that analysis I abstract them into just these three elements ...
> Megan: ... Can you give me an example of a rule that has resulted from an earlier output of previous code?
> Mmm, [pause] well, this for example, well, the combination, I just use the three elements, and the combination here is different from here, that was because of the pattern it generated is not what I wanted. So, I change to this.
> Megan: But the thing that you wanted existed independently of the code, is that right? You were looking for a specific thing?
> Yeah, that's the goal there, no matter what space come out
> (5c.FinalPresEsther10.5.99/US/p.5/l19-p.6/l8)

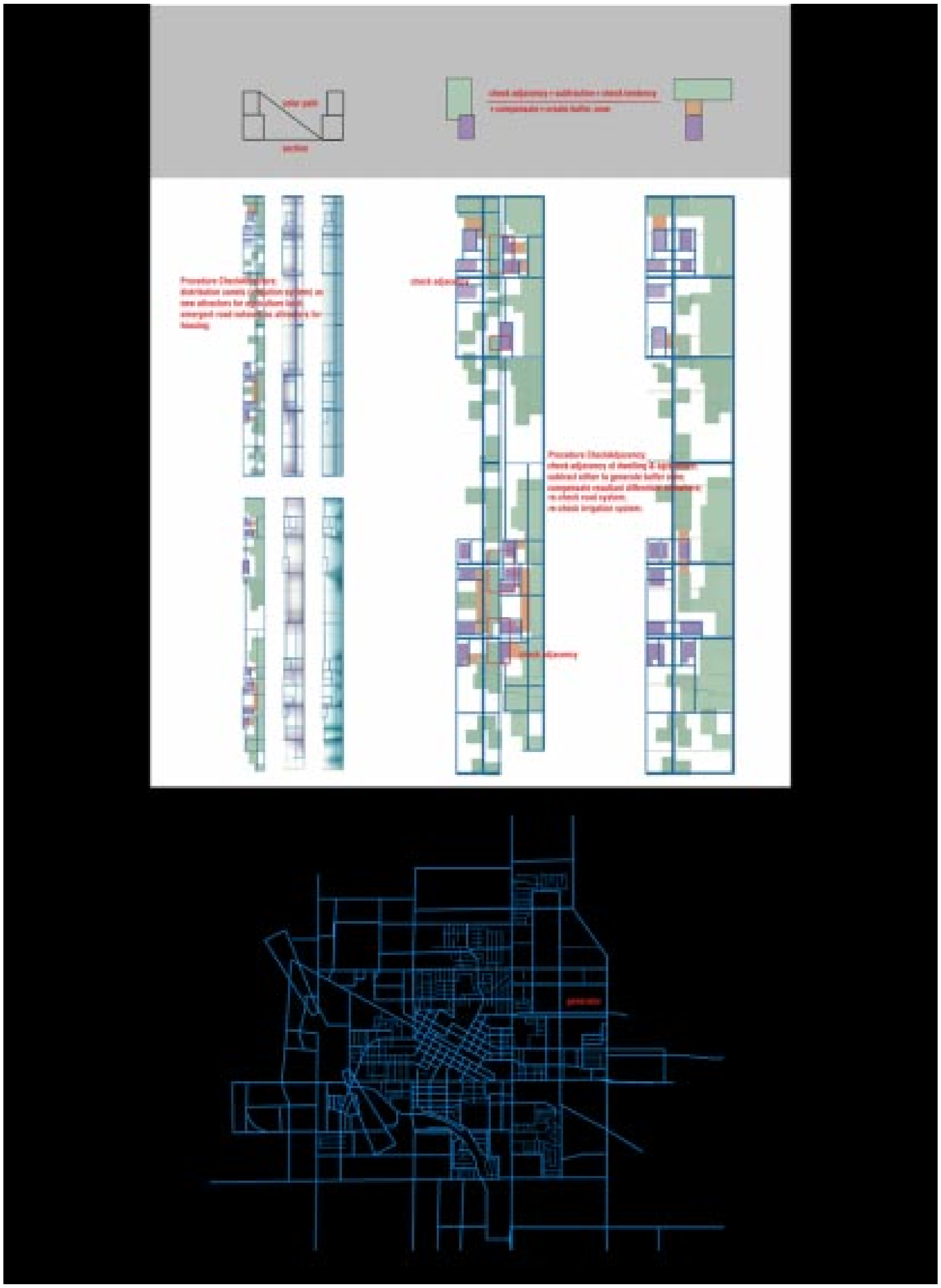

solar path
section
check adjacency + subtraction + check tendency
+ compensate + create buffer zone
check adjacency
check adjacency

Figure 59. Henrietta Green's three posters were actually done at the same time, and show a Behaviourist single moment of achievement rather than recording progression in a more Constructionist manner

At the end of the semester, I had assumed that for whatever reason, Esther had been unable to break out of her existing and limiting model of designing. It was therefore an enormous surprise, in the fourth and final interview six months after the end of the semester, to hear Esther speak in entirely different terms. She spoke of a desire to work in her current design studio outside of her previous constraints of form following function, trying to understand the process of developing one to support the other, although she confessed it wasn't always easy.

> And also I'm trying to avoid introduce ... function or program into my study.
> MY: What are you trying to avoid that?
>
> I [am] trying to get a degree of some kind of word I think. I will be like go back and forth. Sometimes I feel like function is too [much of] a burden and constrain me too much, and sometimes I feel like if I don't have function, I don't have a start point. (InterviewEsther13.11.99/A4/p.3/l23-29)

When asked how her approach to design had changed over the course of the last year, she acknowledged this change and attributed it to several different sources.

> [How has my approach to design has changed] in the last year? Well that's hard. ... I changed some because of the building technology class I took and I changed some because of the [Digitally Mediated Design] workshop I take and the studio I take, but ... well probably one [way it has changed is] before I came here, the way I design was much like function generated form, like totally modernism way to approach it. ... And ... something like Esmond or Tschumi did, I totally don't understand it, and I try to understand it totally from my perspective. But you, you can understand the problem [with what] used to be my perspective, I totally couldn't understand it – and now, gradually ... I started to [realise] everything is possible. I think it's not so [much] any more, not so strongly [that way]. Architecture has to serve some kind of a function. And in terms of the way to design – I started to look other ... aspect of architectural design. Probably not just architecture ... for use, but architecture as technique and the way you work it, but not ... the thing you produced. Probably that is wrong, but you know (InterviewEsther13.11.99/A4/p.12/l28-p.13/l13)

By "architecture as technique and the way you work it, but not the thing you produce", Esther is distinguishing between process and product in her design education. More significantly, she attributed to the Digitally Mediated Design course her newly discovered understanding of feedback and interpretation in designing.

> MY: What was it about the workshop that made you think that?
> Oh, make me to think how to you know, if you use computer or program to help you to design, whether – if you still use a sketchbook, it's like you still got some predefined idea and you let the computer to do it for you, and it totally obey your will. And by doing this actually you, reduced or neglect the potential of get feedback, or get some inspiration, or get suggestion from this design tool. So ... now I think more important is the interpretation process. Like what you can get from soft- from design tool and how to interpret it, and to what extent you can change it, you can control it, how to compromise and how to take the feedback from, or the output of this design tool. And so that's how people should open [their] minds to accept the output from design tool. (InterviewEsther13.11.99/A4/p.6/l7-17)

This knowledge, at the time of this final interview, was firmly framed in the concept of using a computer as a design tool. Esther had not yet made the move from stage four to stage five of understanding the design process. Yet the distance she had overcome in moving from stage one to stage four between the third and fourth interviews was far greater than the final step of viewing this as a description of designing in general. And when Esther was asked to make a link between programming and sketching, she replied with a very clear description of the cyclical nature of the design process.

> MY: So what would you say was the link between programming and sketching?
>
> I think it's sort of like a spiral. You start with something, I don't think it matters start with programming or start from sketch. But you start from something and if say you start from program, it's output and you back, go back to sketch, probably in this sketch you're trying to understand this output. And probably you're trying to say I want to do this, I wanted to do that. Next step. But this, this goal is based on what you get from this design tool, [the] program.... And then, you go back from the sketch you go back to program again and test it and, to simulate, and probably you get something right, and probably you get something totally different, and probably the design tool refuse to form what you wanted it to do (laughs) and generates something totally different. But from this sort of output or from feedback from this design tool, you have to start here. And the second phase, the next phase of this [is] based on this output and so it's like a spiral. (InterviewEsther13.11.99/A4/p.6/l18-30)

Esther recognized her own limitations in the course and admitted a recognition of her need for control in the course and the realization that she should let the design tool do more for her. Most important of all, she explicitly stated a recognition for a change in attitude towards using digital media in design that is so critical.

> I didn't really let it to help me to think during last semester ... because I ... too much wanted to control everything (laughs)
> MY: What has made you change your view since the end of the semester?
>
> Because I was so frustrated during that workshop and ... [I was] thinking about how to use design tool. That's the thing to trigger me to think about this, and now I was thinking probably ... I should let design tool to do more. But ... I just don't know how to let it do [more, I] don't know the method, and don't know the attitude actually. And because I was thinking ... it's a gradual thing, gradually realize that you need to change attitude, and how to change this attitude, and how to work with design tool. (InterviewEsther13.11.99/A4/p.9/l34-p.10/l4)

This change in Esther's ability to describe the design process so clearly is considerable and cannot be underestimated. It represents the most extreme example of a change that all the students in the course experienced to some degree. It is a clear demonstration, in Esther's own words, of the significance of the Digitally Mediated Design course in developing an understanding of the design process. As such it presents a strong case for affirming the main hypothesis behind this study: that computer programming used in a Constructionist pedagogical framework offers the opportunity to develop a better understanding of a personal design process.

Appropriation of Digital Media in Design

"The computer is an expressive medium that different people make their own in their own way" (Turkle & Papert, 1990 p.348)

As has already been stated, the process of learning to design involves the development of a concrete style of reasoning, and the ability to use objects to think with. In a traditional design process, these objects are normally sketches or sketch models, roughly created and necessarily disposable. The process of learning to design involves becoming close to these objects and personally involved with them. This closeness must cease as soon as the object has served its function of helping to develop the next Emergent Rule. Mature designers are involved deeply with the creation of these objects, as they are physical representations of mental constructs. However, the trap that many novice designers fall into is one in which the personal attachment to the objects remains beyond their immediate use. Developing the ability to become involved with these objects, yet discard them as soon as they are no longer useful, is an important aspect of learning to design. Designers with this developed skill have usually found the more traditional, formal and logical approaches to learning to use computers counter intuitive and unhelpful for their design process. The emphasis on the images created further exacerbates the difficulties of using the computer whilst designing, since they cease to be 'objects to think with' and become 'objects to look at'. For computers to be useful in the process of designing they must both support and enhance a closeness to the objects of design, encouraging a detachment from the resultant product that enables the next Emergent Rule to develop more freely. All too often digital media create rather than remove a distance between the designer and the objects of her designing.

Reasons for Joining the course

All of the students stated in the first interview that their reasons for joining the course were primarily focused on a desire to learn more about the computer as a tool. This was expressed in one of two ways. Either the students declared that they wanted to know about the computer's role in the design process, or they stated that they wanted to know more about computer programming. Although the focus of these two statements differs, the outcome is the same; the student is primarily interested in the medium. No student said they had joined the course to extend their abilities of designing. None of the students considered the possibility that the course's aim was to teach them cognitive skills for designing, not to teach them to program a computer, nor even to teach them about using the computer as a tool for design. The computer was a tool, but in the context of design this has an enhanced meaning. The focus of the course was the development of their design ability. The aim of the course was to get the students to be able to use the computer as a medium for design: to use it as a tool for design. To be successful for designing, any tool used in the process, whether pencil and paper, sketch models, or digital media, must support the designer's ability to become personally involved with the objects she is working with and help her to develop them to support her concrete thinking processes. Whatever the medium, design necessarily involves this soft style of thinking and a strong personal investment in her the materials of the situation. Turkle and Papert summarised characteristics of the style of thinking so critical for designers: "a closeness to objects tends to support a concrete style of reasoning, a preference for using objects to think with, and a bias against the abstract formulae that maintain reason at a distance from its objects." (Turkle & Papert, 1990 p.350).

If the process of learning to design involves learning to use this soft approach to thinking, and learning to use objects rather than rules of logic to think with, it might seem odd at first glance to use the process of writing logical rules for a computer program to teach the process of designing. As Turkle and Papert point out, the prevailing image of the computer is that of a logical machine, and programming is usually seen as a technical and mathematical activity. This view is dominant within the architectural profession and was visible in all the students to some degree at the start of the course. Their choice to join the course – the fact that the students were self selected – to some extent indicates an openness to the possibility of using the computer in a more creative way. But all of them indicated at the start of the course some extent of assuming the computer was to be primarily used to solve quantitative design decisions.

Teaching for Digital Appropriation

Contrary to the view of computer programming as a technical activity, supporting only logical modes of cognitive activity, Turkle and Papert point out that the computer's position as the interface between real and virtual worlds means that it can actually provide a context for the development of concrete thinking. The rules of logic, when used to produce a graphical image on a computer screen, are re-expressed in a tangible way that is helpful in supporting softer styles of thinking. Turkle and Papert refer to computer science students who preferred this style as having difficulty in coming to terms with the more traditional, hard approach to the programming courses they were attending. To cope with the difference in approach, Turkle and Papert described the students as distancing themselves from the problems they were trying to program. "One of its symptoms [of this distancing process] is the language with which they neutralise the computer as they deny the possibility of using it creatively" as they dismiss it as "*just* a tool" (Turkle & Papert, 1990 p.349). The Digitally Mediated Design course described in this thesis aims to do the exact opposite of a traditional programming course. Rather than teaching a distanced stance the students are taught to become more involved with the situation and closer to the objects. The success with which this was achieved can be measured by the extent to which the students focused on their designs and not on the computer as a tool. Learning to use it as a real, transparent tool for design, permitting and even enhancing the development of concrete thinking, and therefore of designing, occurred to varying degrees amongst the students. However, there was never any denial of the possibility of using the computer creatively. All the students in the study were interested in using the computer for creative acts. The problem was the manner in which the computer was used within the creative act and the distance the students kept from the designs they were creating that permitted some of them to refer to it as just a tool.

The successful development of this closeness to the design process relies on careful nurturing through both the pedagogical style used in the course, and the programming environment utilised, combined with constant and consistent encouragement to focus the student's concentration through the computational constructs, so that they become transparent, and on to the designs she is developing. The formal, logical rules must be re-expressed by the computer as tangible objects in a graphical language with which she is familiar. The pedagogical benefits of expressing design ideas in formal rules have already been described. This chapter examines the extent to which the students appropriated the techniques of the course to become more involved with their designs through an examination of the extent to which they focused on the computer or on their design work when describing their projects. A concentration in the interviews on the computer as an entity, the development of the rules and computational constructs, a focus on Imposed Rules and writing code to produce an already known outcome with no room for surprise, combined with reference to the computer as "just a tool" implies a maintained distance from the design process. Reference to the computer as a "tool" alone does not imply this distance, however, because all successful design tools imply a closeness. A focus on the design development, descriptions of the design process in the first rather than third person, variation permitting surprise designs to emerge, development of "emergent rules", and references to the computer only as secondary to the design imply the successful appropriation of the computer as "just a tool" in the specific context of successful design tool usage.

The success with which the students became involved with their designs in the course varied enormously. The most extreme example of remaining distant was Hannibal, who consciously chose to not commit himself in this way by focusing on the computer as a tool for the entire semester. In the final interview he said he was not sure whether he wanted to use programming to design with in the future, or simply "to investigate the idea of using it" again.

> And that's ... going to be independent of where I go after I leave here – whether or not I'll be able to continue to use it ... as much as I would like to or actually continue just to investigate the idea of using it, let alone actually using it again.
> (InterviewHannibal11.11.99/US/p.7/l23-26)

As with all the other students, Hannibal stated in the first interview that he wanted to see what role the computer could play in the design process and to learn about using it as a design tool and possibly as a teaching tool.

> I would like to understand how it can be used in the design process, if it would be helpful to me personally, but also if it would be helping as a teaching tool, because I'm very interested in teaching after I leave. ... I find myself analysing certain classes, what I'm going to take or what I'm going to look at in terms of is this going to be really of any use to me outside education. I mean, aside from furthering thinking is it going to do, asides from scholastics is it going to be anything that I can use either to better my design process or to teach design. ... So I think the main thing is I would like to see if it can be used as a design tool, ... [and] also as a teaching tool.
> (2aInterviewHannibal15.2.99/US/p.22/l22-33)

However, what was significant about Hannibal's approach to the course in the first interview, was his answer to questions concerning his initial ideas for a project for the course.

> [The language we're using is] [VectorScript], correct? Because I don't know the program at all, or what the program can actually do, I really don't want to get too set in what I'm going to look at until I know not the limits, until I know, you know, what it can be used for. ... Because I don't want ... to take the program and apply it to this. I want to try and exploit the program somehow. And the vehicle I use as long as it's something that's going to keep me interesting, it really doesn't matter as much to me right now. (2aInterviewHannibal15.2.99/US/p.23/l9-16)

Hannibal began the semester with the firm belief that his design ideas should be dictated by the constraints of the software. The focus on tools continued through the semester. In the second interview he repeated that the programming constructs were leading the project

> I'd like to look at other things [in the class], I think, just for the programming language ... I would ... really like to start to see how, I'm not even sure how things go, start to move with the vectors yet, in either in graphic 2D or, so I'd like to start to look at that in three dimensions to see some of the relationships that I could, this could start to make. (2b.InterviewHannibal5.4.99Div/US/p.3/l1-6)

He became stuck in design terms in his project and resorted to attempts to rewrite the code rather than deal with the more difficult design issues he was facing, referring to the code in the third person:

> So, I guess that's what I was saying, was to start to look at a ... smaller scale of the water. Maybe I do that but extend the boundaries back slightly, to the edge, and then start to use an existing building block, or it takes an original object, ... or a building object, and then can start to perform the same type of operations, but, three dimensionally, I guess, where it can make a cube instead of a square, and then it divides the cube into certain ways ... Well, maybe now since I can input more of the rules as it does this, so [there will be] some reason to why it's going to break it down into four squares as opposed to six, or as opposed to nine.
> (2b.InterviewHannibal5.4.99Div/US/p.3/l27-37)

In the fall semester Hannibal continued with the course but again, despite being set a purely design project, focused on the code when he had difficulties with the design

> I just have to be very conscious of what I'm actually starting to use the code for. So I think it worked to begin with but now I think I'm having the trouble again of thinking about what to use the code or this point, or at this step in the design (InterviewHannibal11.11.99/US/p.7/l59-p.8/l2)

He remained very lucid about the benefits of coding and using the computer as a tool in design without actually using it himself in that way

> I guess most people would think that probably ... drawings would probably be more about phenomenology, and the computer might be used more for the hard information. But I think – I mean from an analytical standpoint it'd almost be just the opposite. (InterviewHannibal11.11.99/US/p.8/l56-p.9/l1)

So detached was Hannibal from his design work that he expressed an interest in continuing with his project as two separate parallel investigations using the same design element but different tools, unaware that this would in fact be developing one project, not two

> I think what is pretty interesting is how you can start a project off, ... and then branch it, but then continue both projects simultaneously, and probably do the same amount of work as if you were continually working on one intensively. But start to continually branch that off in different avenues, and to see how one starting point can branch different ways, by changing the code a little bit or by doing this, or by altering things. I think you start to see the possibilities of ... so much more different variations on one theme. (2b.InterviewHannibal5.4.99Div/US/p.21/l21-28)

His detachment was also expressed in the way in which he used the code. He stuck rigidly to writing code that would produce a graphical outcome that he already knew, and despite continuous encouragement had little interest in permitting variation to create unexpected designs

> I think it was extremely helpful in using the coding process to try and set up a series of, of certain aspects that I wanted, and certain aspects that were very important to the project, which needed to occur a hundred percent of the time, or certain things I thought were interesting but may or may not play an important role in the project. I think it helped me clarify certain intentions, very much clarify intentions I had in the project (2c.FinalPresHannibal10.5.99Div/p.16/l22-27).

Interestingly, in the final presentation of the semester Hannibal did make some effort to present his work to the design faculty in more involved terms; referring to the work in the first person on occasion, and even correcting himself mid sentence.

> I guess there's two ideas that, there was a series of different parts of the code. One was it generated the site boundaries, which it generated an overall volume which became this, and it used that as the initial restriction, but it was, it inputted, I inputted, or it, it looked for the two piers, and then went over a random distance between this pier but the, this volume between this edge of the pier and this edge of the site had to be large enough for the islands to come in. (2c.FinalPresHannibal10.5.99Div/p.6/l6-11).

Like all the students in the study, Hannibal is highly intelligent and listening to him in the final interview describing the benefits of coding, comparing it with the process of sketching, and demonstrating his understanding of its role in designing, it would be easy to think that Hannibal had successfully achieved this himself. But his interest is purely theoretical and as such highly detached with no personal investment and no involvement. Hannibal did not actually use the tool to design, and did not become involved with anything in the process. He remained distanced and as theoretical as possible. Hannibal's SMArchS thesis further supports this detachment since it is an investigation of tools used in design.

> [R]ight now [my thesis is] half of using it as a tool for studio, and you know just traditional type design, and the other half is using the perspective, and, using the perspective to try and understand the mode of operation and the mode of representation, if those two can be the same. Because I mean if we work in plan we sort of think three dimensions, so if we can start to adjust the two, that could work together. (2b.InterviewHannibal5.4.99Div/US/p.4/l7-12)

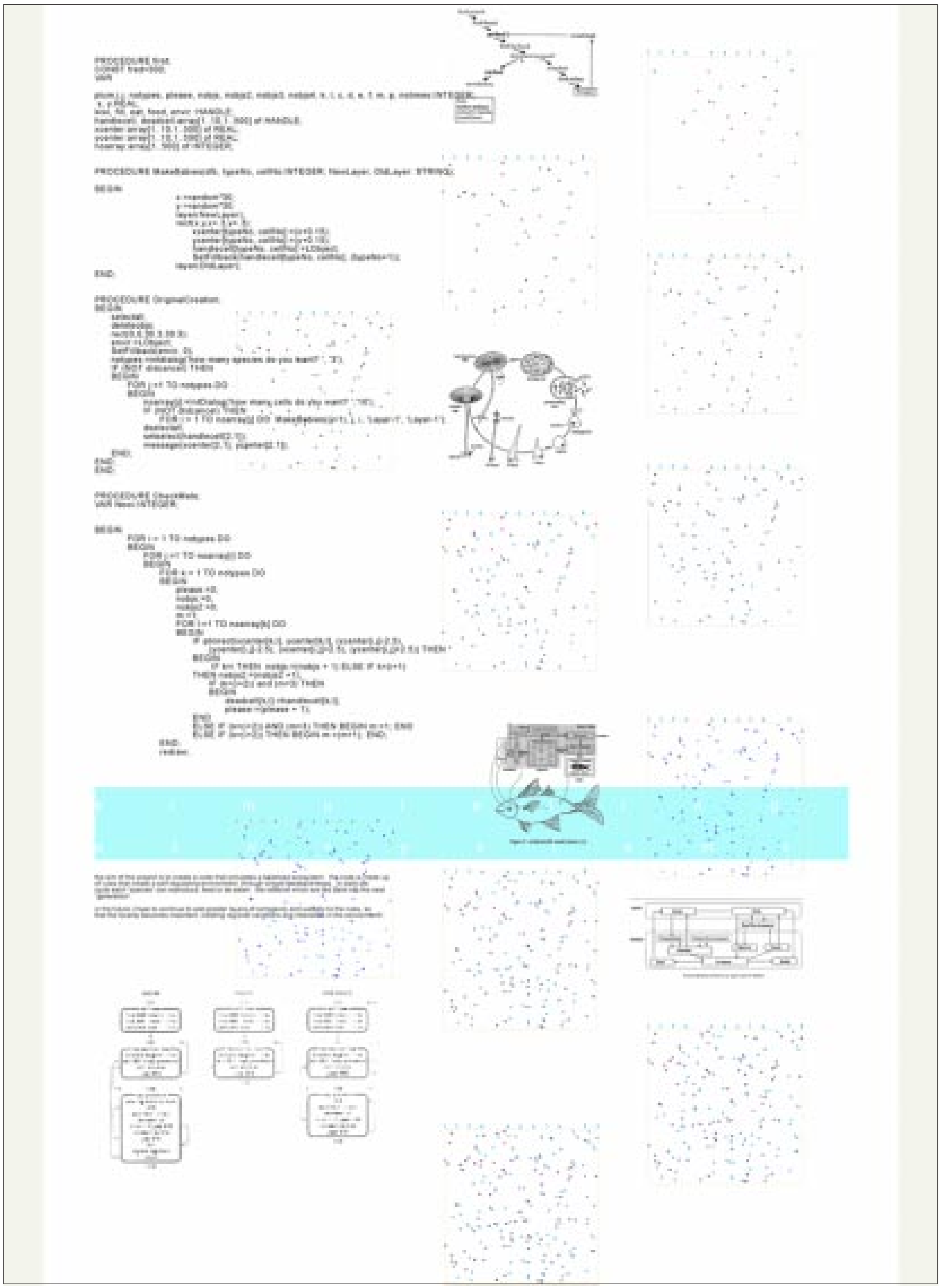


Figure 60. Josh Carver's project was entirely computationally based, modelling the Game of Life, but he was the first in the group to realise the cognitive significance of what he was learning

At the other end of the spectrum from Hannibal was Josephine, whose intense computer phobia, developed during a bad summer camp experience as a child appeared to actually support her largely successful desire to focus exclusively on a design project.

> And I went to a computer camp when I was in sixth grade, but it was totally a waste because there were these kids that knew so much more than I did and I had no idea what was going on. I went there and I sat in these classes and I couldn't understand what they were saying. It was like they were talking Greek, you know. ... It stunk, because I would go to lab and not do a thing ... So (a) I like had no idea what was going on, (b) like you know all the kids that did know what was going on banded together as well, and ... it totally stunk, because I had no idea what was going on. (3a.InterviewJosephine21.2.99Div/US/p.16/l1-29)

She expressed feeling out of control of the computer and scared by it, and found VectorScript difficult to learn and use.

> Oh, it's a pain in the butt [learning the language VectorScript]. I hate it, I hate it so much ... if I don't think too hard about it then it's fine, but as soon as I start thinking too hard about it I get so confused, ... as soon as I start thinking too hard about handle and array or any of those terms, I get really like lost, but then if I step back and not think about it, you know, and I just sort of type in .... Yeah, but it is a headache. Like, sometimes it's something as stupid as I know I want to program this and I know I probably could, but, I'll start it and I'll just go the wrong way, and like I'll sit there for like half an hour because I don't want to bother anyone, and then all it takes is like, you know, five minutes, five seconds of [the TA] (3b.InterviewJosephine5.4.99Div/p.17/l18-p.18/l6)

> I hate programming! It's like, I really, really don't like it ... you called it logical; I completely hate it. I completely hate the language of it, and having to get it. (3c.FinalPresJosephine10.5.99Div/p.10/l1-8)

Yet despite this – or perhaps because of it – she was able to see the point of programming and be excited by the opportunities it presented for her designs.

> It's hard to say what kind of studio project I would do, or how I would use [programming in the future]. Because even at this, I mean I think it could be useful at most scales because you know even, I mean, I could have used programming to

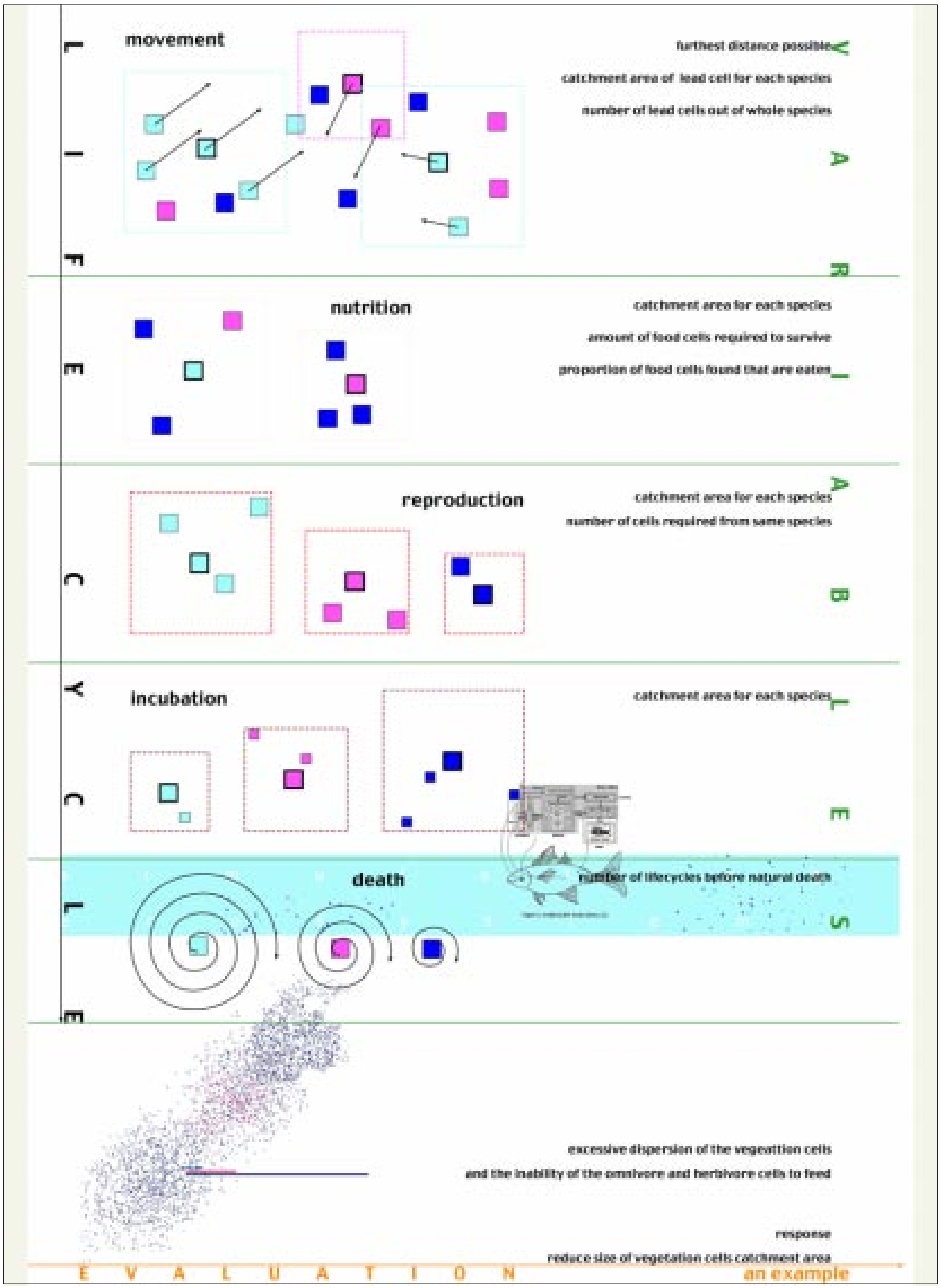

L I F E C Y C L E
movement
furthest distance possible
catchment area of lead cell for each species
number of lead cells out of whole species
nutrition
catchment area for each species
amount of food cells required to survive
proportion of food cells found that are eaten
reproduction
catchment area for each species
number of cells required from same species
incubation
catchment area for each species
death
number of lifecycles before natural death
V A R I A B L E S
excessive dispersion of the vegeattion cells
and the inability of the omnivore and herbivore cells to feed
response
reduce size of vegetation cells catchment area
E V A L U A T I O N
an example

Figure 61. Josh Carver's second poster shows a development in the project, but more importantly for Josh was a development in attempts to document process

> detail a smaller problem about structure, just dealing with the kind of you know shape of the dry dock, you know. But my dealing with it would [be] generated out of the fact that I design very intuitively, and I tend to like just take my pencil and you know kind of do these shapes and move things around, and I literally cut pieces of chip board out and move them around and that's, and I just play all day. So, like the advantage of [the programming] was it's really exciting because these are, I can make thousands of these in ten minutes, and you know, I can't do that all drawing by hand. Even to see things three dimensional is very exciting, because to build a model like that would take for ever, and to maybe even think about, you know, like some things that can float because it's digital and it's not physical, you know and to see that, and just to feel that sense I couldn't with just making a model
> (3c.FinalPresJosephine10.5.99Div/p.10/l17-33)

The project Josephine developed during the semester was based on, influenced by, and had an influence on, the studio project she was working on at the same time. She was the first to express the understanding that the computer could assist her in qualitative design decisions.

> I thought all along that like programming equalled analytic, quantitative, something really, just alien, and I never thought of it as more a qualitative tool, more, you know, a thing that I could just tweak a little, and it didn't have to be about being an exact quantifiable thing, and … I had a problem … even when I was taking the class [in] getting over that hump, because I was just like oh it's just numbers, and code, but it really is just a vehicle, so, and you get there, and so, that was the biggest realisation I made
> (3c.FinalPresJosephine10.5.99Div/p.11/l19-26)

In the first interview she made a comparison between computer and pencil working:

> I actually plan to be very digital this semester, because … [my studio professor] insisted that we do our drawings digitally, or posters or whatever you want to call it. … I feel like once you start designing on the computer, you can't really go back to, it's hard to go back to the drawing board because I don't know, it's not the same. … I started to draw with a pencil and MayLine. And it wasn't the same at all, the same kind of control … and there's something really like absolutely clean about the computer line (3a.InterviewJosephine21.2.99Div/US/p.18/l20-34)

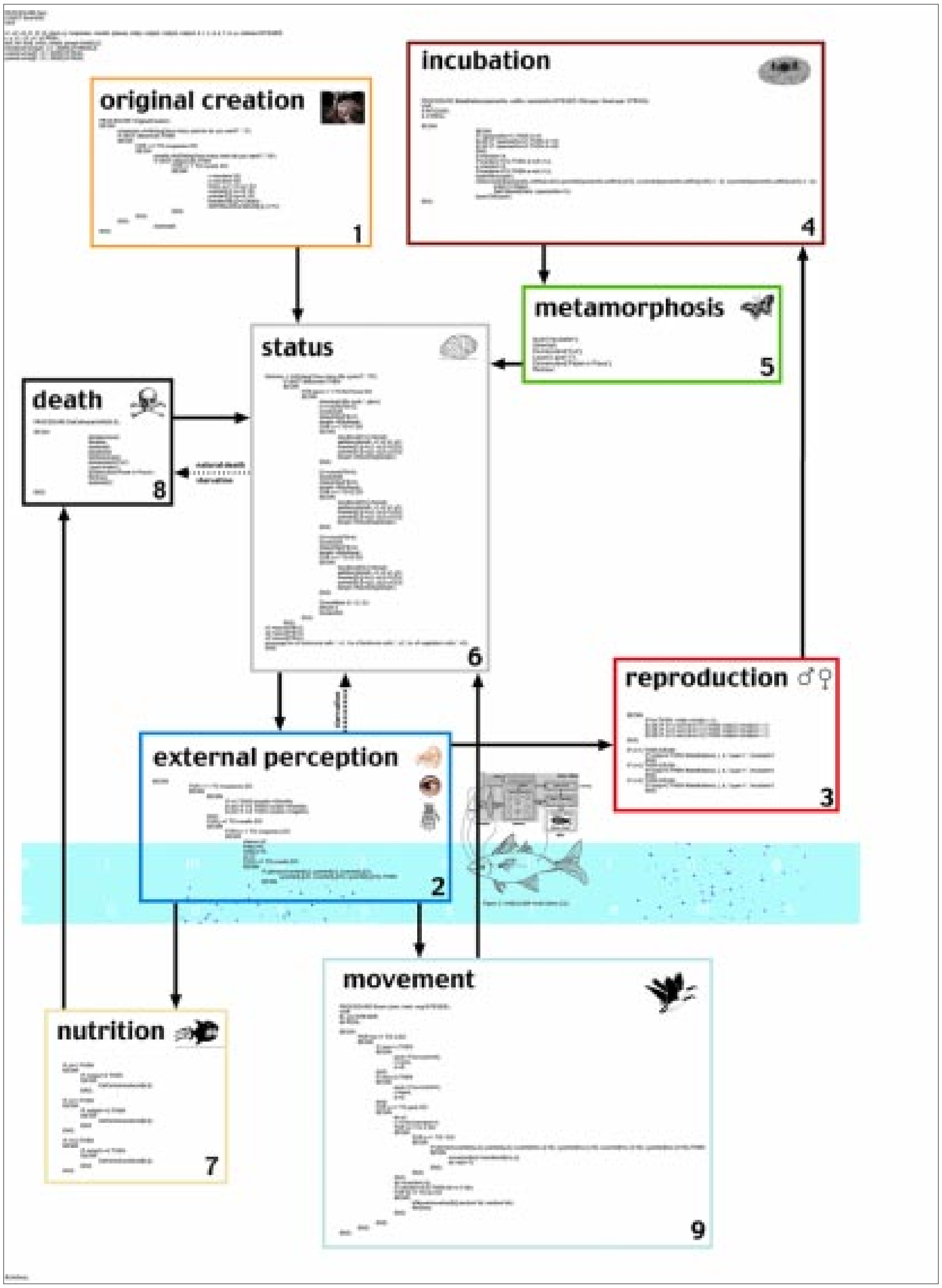


Figure 62. Josh Carver’s third poster shows both extension of the capabilities of the software, and a further effort to better describe the cognitive process of understanding Josh has been through

This largely implicit knowledge of the links between the two became explicit by the fourth and final interview. She describes the point of programming as being tinkering with design ideas

> [I realised] that the programming isn't like this absolute control thing. It's, it's this thing you use to tweak things so you can sharpened your preferences more for what your thing is doing. And then, also, just to see it changing more and more and having lot of [feedback] is really exciting because then you start to feel like you're really doing something. (InterviewJosephine12.11.99/A4/p.9/l4-8)

and could see through her experience that when compared to sketching the benefits of coding included being forced to ask yourself 'why' more with the latter.

> [T]hat's what I notice from programming in your class was that, you know, why you're going to do the next thing. You know you always think Why? right? Well that's something with the sketching. Sometimes you don't think Why? you just do it, 'cause Oh it's pretty and your hand likes to move this way, or something. So – um – which is good and bad, you know (InterviewJosephine12.11.99/A4/p.6/l20-29)

She was very aware of the point of the Emergent Rule process, and realised that it wasn't the programming that was answering her questions but her own manipulations of it.

> Obviously I see a direct link between programming and designing because well like in your workshop it was like, OK your code is doing this now. Now some of the outcomes are doing this and you don't like that. And you like these kind of outcomes so how are we going to like change the programming or, you know, edit it so that it'll just do that and not this. So I mean it was a direct link, I thought, ... Well it's a different concept I guess of designing, right? Because just from my limited experience of the chunk of project that I designed, the design part I was using was about finding the right almost like composition, three-dimensional composition that I liked. And sometimes the parameters for why I liked it, they didn't have to be defined. They could be totally based on subjectiveness, you know. Like I like it, like why you like blue versus yellow. So in that way programming doesn't really answer some of the questions of why you like what you do, or why you're doing what you're doing. I think that's where – most people think it is doing, but it's not. (InterviewJosephine12.11.99/A4/p.7/l14-27)

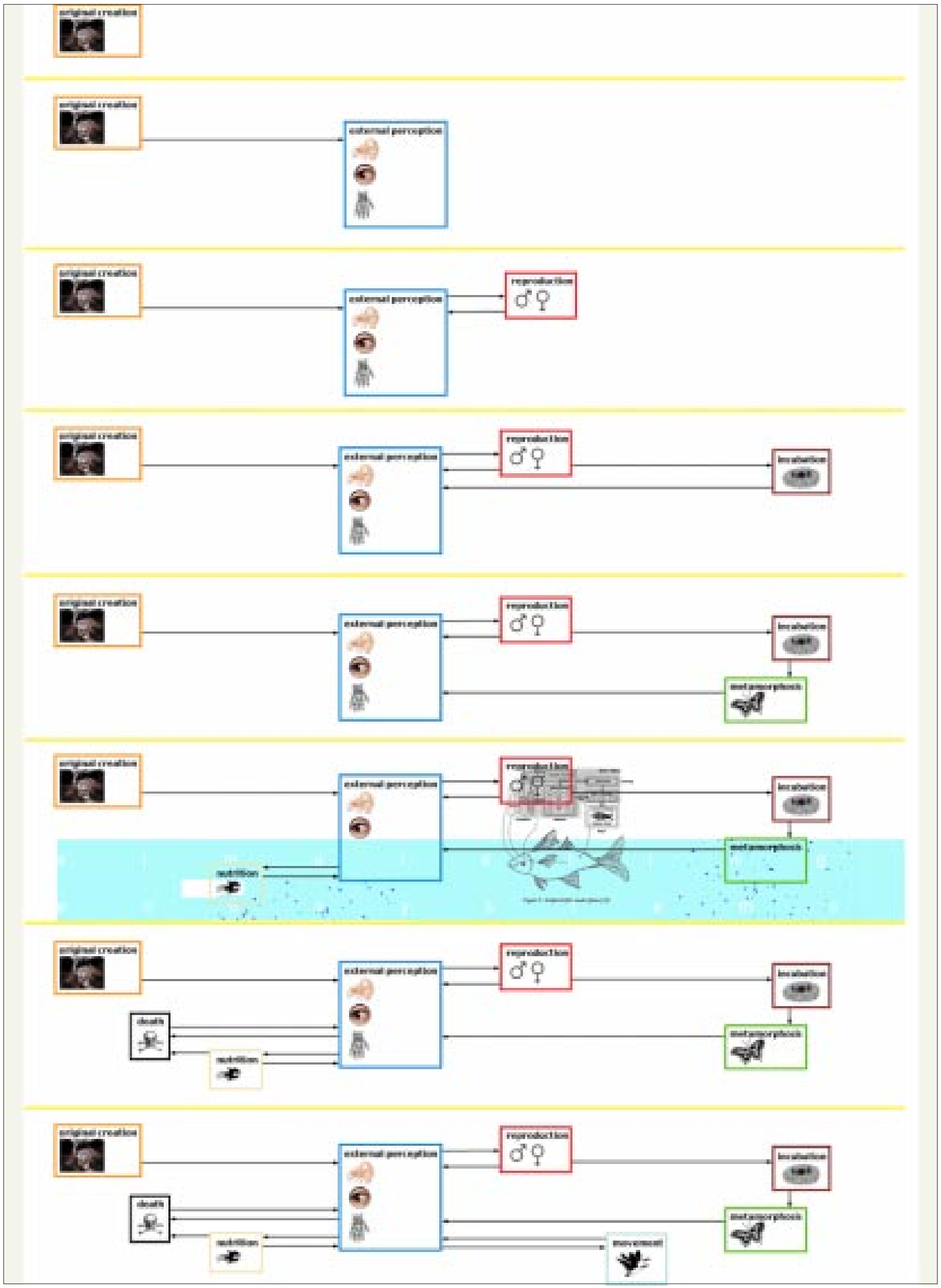


Figure 63. In Josh Carver's final poster, he has incrementally shown the development of the process he went through, showing clearly the evolution of Emergent Rules

Although she continued to express a dislike of computers in general, and programming in particular, she said the point of designing on a computer was that it gave you more control.

> You could be sitting at your desk playing with a bunch of models, but on the computer you know exactly what you program is going to be doing, your code is doing. So it's like you're, you *feel* in control. Whereas when you're almost like at your desk making models, you're in control but I don't think you feel as much in control as when you're like – designing on a computer.
> (InterviewJosephine12.11.99/A4/p.7/l41-p.8/l2)

Note that Josephine is talking exclusively about feeling in control of the design project, since she remained feeling somewhat out of control of the computer itself.

The example of Josephine offers an interesting insight into the possibilities of teaching the course to architecture students who might not necessarily see themselves as being interested in computers. The predominant view of architectural academia, despite significant advances in technology, remains that to take a course with any computational element, you must be predominantly interested in the computer as an entity. Josephine joined the course almost by mistake: although she had seen what previous students had achieved, she joined only because there were no other courses available to her that fitted with her timetable. She has an intense dislike of computers and is only interested in using them if she can see clear benefits for her primary interest in design. She uses them to become closer and more involved with her design decisions. In this regard, she represents the most successful product of the course. It is interesting to note that Josephine is the third person in the course's short history to have been taught LOGO in grade school. The previous two students, from the Spring 1998 semester were, using the same criteria of appropriation and focus on the design project, similarly the most successful in their course. They both chose to continue the following semester with self directed design studies that equally successful.

All of the other students in the research study fell between Hannibal and Josephine on the spectrum of computer appropriation for designing. All four remaining students during the Spring semester were either directly involved in the rule based design studio mentioned earlier, or had been in it the previous Fall. The emphasis of this studio over the last two years has been, and continues to be, an exploration of the computer as a design tool. Detachment is encouraged, as students moved from one piece of software to another with the express purpose of exploring their potential.

> [I'm currently in the Rule Based Design] studio. We are using, we have softwares as design tool, so we have to choose one of them and define our own area of study. Some people choose material and construction process, and some people choose to experiment more with this design tool and explore its potential and possible way to interpret it with this software as well as the output of this design tool. So I chose the second direction, I chose MOSS (InterviewEsther13.11.99/A4/p.1/l8-29)

This exploration, however, largely falls short of actually designing. With any new and complex tool in the design process some attention must be given to the tool itself at the beginning before it can be appropriated successfully for designing. The rule based studio focuses only on this beginning period of learning to use the tools, and although there is some semblance of a design project around which these explorations occur, the emphasis is on tool exploration and not design development. This view both stems from and further supports the predominant view of computers in design. It is therefore not surprising that the remaining four students exhibited signs of detachment from their designs throughout the semester. Raphael – like Hannibal in the SMArchS program – decided not to join in the second semester course in the Fall because he felt there were too many other options open to him and other roads he wanted to explore

> I'm thinking of the Java class [for next semester], and also, because I only have two years here, it's a big problem. Last time I took 6 classes, it's just, it's too much. This time I'm taking four, but that TA-ship is taking more time than a half TA, so, so it's tough. ... I look at the course book and I'm like, oh, this sounds interesting, well this sounds interesting, but when you get down to it it may not be that interesting, and so you, it took a while at the beginning to kind of go through what was being offered, and ... to know what's going on (4b.InterviewRaphael2.4.99Div/US/p.26/l11-20)

MIT offers an enormous range of courses that are available and of interest to architecture students, which combined with those available at Harvard, mean that many students find it difficult to focus on and commit themselves to one particular aspect of architecture in their studies. Like Hannibal, at the start of the semester Raphael viewed the computer in design as a distant, remote tool to be manipulated and controlled, referring to the sense of remoteness with a Japanese term:

> It's, you've just got to take advantage of, you know, I think to me the computer's a tool. I mean I take advantage of what that tool can do, and ..., with the computer there's a lot that computer can do, and it depends on how much you know how to

> manipulate that tool. And that's true with anything I think, but in Japanese you call it oku, which, like there's a depth, ... there's something back there, you know, and oku is always something you can't obtain, there's kind of a space time relationship, so there's always more, you know. That's how I kind of feel with the computer right now. (4a.InterviewRaphael19.2.99Div/US/p.28/l1-9)

Yet, unlike Hannibal, Raphael's predominant interest is in design and finding ways to become a better designer.

> I definitely want to learn more [about computers]. ... I want to learn more because I want to become a better designer, and not necessarily to learn [just] how to [use them]. (InterviewRaphael12.11.99/A4/p.19/l31-32)

This predominant interest in design helped begin the process of appropriating the computer. By the second interview, Raphael had begun to realize that programming can be used to test ideas – although still not designing, this is beginning to move in that direction.

> I think you ... more and more I'm realising that a lot of the times you can start programming to test quickly certain ideas, and use that and go back to your design and use another tool, to develop that, and not [be] enamoured with the whole tool, per se, the whole ... code per se (4b.InterviewRaphael2.4.99Div/US/p.19/l16-20)

He began to realize the difference between programming as tool creation and programming as tool usage, stepping closer to the concept of programming as designing.

> What's the point of looking at these ideas with this tool? ... I think it's ... using a design project to investigate how to use this tool, design this tool, ... for future applications for myself. (4b.InterviewRaphael2.4.99Div/US/p.18/l2-8)

By the final presentation, Raphael had tacit knowledge of the links between programming and sketching,

> Even if it's random, you control the randomness of it ... I gave it certain tendency but I don't control all of it. And I think that's the key to programming, is why would you want to program if you controlled everything? ... I don't know, it's more fluid to sketch, because you're probably more used to [sketching], and you're literate in

> terms of that media. But I think when you program there are certain ... advantages to generate, I mean it's computational, it has all this power ... it creates these possibilities that you can evaluate (FinalPresentation10.5.99/A4/p.43/l32-p.44/l4)

and using programming to make qualitative decisions

> It's that, there's, the one thing that I wanted to control [in my project], within a range, was that there is a certain spatial quality of the layers of site, that have a formal aesthetic quality (FinalPresentation10.5.99/A4/p.44/l29-31)

However, Raphael had difficulty in becoming aware of this tacit knowledge in the fourth interview until the language used in the questions was changed to include the phrase digital sketching.

> MY: I'm asking you about digital sketching.
>
> Oh, I see. I think, OK now that I know what you're getting at... um (pause) ... I guess you're trying to say Is it an explorative, a tool for exploration? ... I think, though, the one problem is just the actual amount of time it takes, maybe, to set up the initial [program] and then on the other hand you have this unbelievable amount of generation [of] possibilities. And then you can start to play around with your settings, so that's also part of the design right there. So in that sense, I think yeah, you could use it as a tool to, a sketching tool. (InterviewRaphael12.11.99/A4/p.11/l2-13)

At first he put sketching and programming on the same plane by saying they are just different tools within the design process, a comparison the makes programming for him as personally meaningful as the process of sketching,

> MY: What do you see as the link between sketching and programming?
>
> ... I don't know if, I don't think one is the same. I think they're all different ways of designing, and they're all different tools within the design process. (InterviewRaphael12.11.99/A4/p.8/l19-24)

but when pushed he began to recognize in the interview itself the improvements of programming over sketching.

> And programming to me, as opposed to sketching? I think um, (pause) there's the power of programming in terms of ... it's just pure computational power. And you

> think about setting about even just simple relations that can start to generate unbelievable possibilities. And I think it's the amount of control that you can set in terms of controlling yourself and controlling your program, yes. And I think it becomes a powerful tool, but I also think you need to be even a better designer, in a sense (InterviewRaphael12.11.99/A4/p.8/l36-p.9/l5)

His reasons for this thinking are vague to begin with but he is eventually able to articulate why it is important to be a better designer to be able to take full advantage of the programming process.

> I'm interested in how, within the process, you have deliberations of which way to go. But I'm also interested in how that can lead to a good design. And I feel that it's such a powerful tool that it makes the deliberation sometimes harder. That may not necessarily lead to the end product that's a great, a good design ... but I feel for myself I even need to be able to become a better designer ... probably because ... there's a level of mediation between the code as is, like the text in the coding, and the amount of control you have over that. And I think it's good because it slows down the process in a sense, that you also have time to reflect along the way. But you also want to speed up the process sometimes. And the ... amount of time it takes, [particularly] for a novice who's not a coder [it's difficult to get] the amount of iterations in terms of speed ... and I think in that sense maybe being a good designer also means being a better coder or a better programmer. (InterviewRaphael12.11.99/A4/p.9/l19-34)

He recognized that in designing, a tool is something you become personally involved with.

> But ... each method [of programming or sketching is] like a sketch within itself. [With a] sketch you're actually manipulating on paper or pencil. I don't know [whether programming is] any more ... controlled I think in terms of your hand. I don't know how to say – they're just two different things I think. I'm not, I'm not saying one is more expressive than the other. I'm just saying they're expressive in different ways. And I think you can use both. (InterviewRaphael12.11.99/A4/p.11/l18-23)

But Raphael's development in the course did not go as far as Josephine's, and even in the final interview he continued to remain somewhat detached in his descriptions both of using the computer in designing and of using sketching. His final project in the course was an exploration that was inextricably linked to a process enabling him to remain detached.

> [My final project] was very much kind of a mesh fluid form, but trying to see how an envelope, a building envelope could be reactive to certain stimuli, I guess, which could be either the sun, the climactic conditions or the preferences of the user or events within, or the program of building elements within it. It's very much talking to an envelope that can be reactive to different ranges of criteria. And I wanted to try and manipulate the form to those criteria, or be able to control those forms. And what happened was that I looked at two things. One was the programmatic elements and how the form can react to that, and the other was the sun positioning, climactic. And to see how to generate as many possibilities that I could (InterviewRaphael12.11.99/A4/p.12/l15-23)

He also described the significant aspect of sketching as being the drawings, the products, not the cognitive strategies involved, providing an analog example of remaining detached and not becoming personally involved with the design process.

> I don't see like too many nice sketches any more. I don't know, maybe I'm not around the studios or – it seems like people aren't being taught how to sketch any more. (InterviewRaphael12.11.99/A4/p.11/l23-26)

In the Fall he had decided to continue exploring tools by taking a Java course. By the fourth interview however, he had become frustrated with Java and was realizing the benefits of an integrated language like VectorScript in enabling him to design.

> I was thinking that Java could do this [in my] thesis next year, which I want to integrate a design part to. But it's just becoming so hard to even get like a shape on the screen that reacts to something that I might switch back to VectorScript, in the sense that looking – even if its a reactive wall, you know, ... the programs are relations of people or events, and seeing how configurations can evolve. I mean there's tons of possibilities I thought that could be applied. (InterviewRaphael12.11.99/A4/p.17/l35-41)

It is clear that learning Java was too removed from designing and too exclusively focused on computational constructs for Raphael's design preferences.

The remaining students all exhibited some appropriation and some distance. Jeremy presented an interesting case because of all the students he became most closely involved with his project, and was the first to realize the benefits of the process on his other areas of work, but the subject of his project was the highly theoretical and purely computational recreation of the Game of Life, and it was unclear during the semester how he might transfer this to a design project.

> Well, hopefully the point would, it would get to the point where each was informing each other. That would probably be the ultimate, ... where you were sketching and that's informing the fact that you're doing computer, and what you're doing on computer is affecting what you're drawing. There's continuous reciprocity. (InterviewJeremy4.11.99/A4/p.18/l23-36)

> [P]rogramming opens up another way [and] the more ways you open up your brain to be able to do something and break down the pattern which you have which tells you "Work like this" I think the better. I think the more skills you have, the more tools you have ... the more flexible you are to different problems, different [sorts of] situations. ... Because you know before I took computer there were so many things which I would have just done like that because that's how I know how to do it, and that's, that's just the way you do it. And then since [your class] I can suddenly see things Oh I can actually do it like this. And by doing it that way, you suddenly see that certain problems [are better seen from a] different perspective, and so the more perspectives you have the better. (InterviewJeremy4.11.99/A4/p.19/l6-15)

Gary and Esther, both in the rule based design studio during the Spring semester, were highly influenced by the latter although Gary fought with himself to permit a better appropriation of the computer for his designing. Here he describes the transition he made from trying to link the course with his studio work, and realising that there had to be more to the process in design:

> OK, well, it's doing this, [indicating images infront of him] but ... it was just a machine. My team members [in studio] did this all on paper with spread sheets and something they did with AutoCAD ... [pasting] it all onto the command line ... and that's all I was doing, but ... a lot faster and a lot easier ... but as far as being ... something else, it wasn't really so much. And ... [you and I] talked about it a bit and then ... I actually realised that's true and because to me I was writing this code and it did this cool stuff by itself, and to me that was being successful to some degree, but ... on a larger picture it wasn't really getting at anything too great. And ... so then I tried to throw it all away, and come up with something from somewhere ... from texts and reading and so on and ... ultimately it got back to ... the impetus for this investigation that dealt with something last semester. And ... that's sort of the journey and now it's at here. (1b.InterviewGary4.4.99/A4/p.15/l22-p.16/l2)

Note that when Gary refers to “just a machine” he isn’t referring to the computer; he is referring to the code that he wrote as being an automated machine that is separated from him. His project in the course reflected his desire to find a way to make the process become more personal and a part of his designing. Interestingly, this desire put him at odds with the same studio professor at the beginning of the following Fall semester when his instincts were to use the latest software he was given to design with, and the professor’s desires were for him to merely explore what the software’s limitations and possibilities were. Gary wanted to focus on design; the professor wanted to focus on the software. By the end of the previous Spring semester, Gary had successfully appropriated the computer. He had moved from referring to the work he did involving the computer in exclusively the third person to differentiating between the studio work and the course work by referring to the work in the former in the third person and work in the latter in the first person, indicating the level to which he felt involved with the projects in each. But by the final interview he was referring again to everything the computer did as something separate from himself, reflecting both the studio’s strong influence over two semesters and the prevailing view of computers in design within academia and the profession.

This view affected the jury panel in the final presentation even more than the students. In Gary’s presentation, there was confusion surrounding the term ‘designing’ where the crit panel were referring to developing code and Gary was referring to the architectural design decisions he had made for his project. In Hannibal’s review there was confusion over ‘process’ where Hannibal referred to the process of coding as an activity which, although not designing, was one step closer to the crit panel’s actual meaning of processes or functions within the code itself. Questions were asked by several of the crit panel about the students’ rules and how they might be best described. Only one of the crit panel, the most design oriented faculty present, consistently framed his questions and comments in predominantly design terms. It is fair to say that the majority of the crit panel members had a predominant interest in computational design processes rather than architectural design, so perhaps this is not surprising. However it is extremely difficult to find jurors who are receptive to the possibilities within the course, yet still focus on design as the main objective. Within MIT there is antagonism between the design faculty group and the computation faculty group that prohibits the former from becoming involved with anything seen to be belonging to the latter. That said, it is possible to view the course as a bridge between the two groups, and the increasing number of design faculty attending reviews in the most recent semester would support this, but there is a long way to go both within MIT and in the profession as a whole if computer programming is to become an accepted means of designing.

Changes in Understanding of Learning

In an earlier section of this thesis it was stated that the process of designing is an activity of constant learning within a Constructionist framework. Therefore architectural education needs to teach students this Constructionist view of learning, often at odds with the predominantly Behaviourist view of learning taught to them through their high-school years. The intention for the course is to specifically address this Behaviourist-Constructionist shift required for novice architectural students. However, all of the students in the study had had prior architectural education, and therefore it was not surprising to find that five of the six students had a distinctly Constructionist view of the education process at the start of the semester. All of these five students expressed in the first interview a view of the professor as a coach, the need for the professor to help promote the students' own explorative process, and an awareness of the self motivation required within the framework of the design studio.

> I think the ideal studio professor should you know, come up with a strong framework for the studio to work within. I think a bit more than, well I picked a building, I picked a project and I've picked a site, and go with it and I'll respond. (1a.InterviewGary18.2.99/US/p.12/l7-9)

Their education process had involved this Behaviourist-Constructionist shift through a highly structured educational framework in the beginning that loosened as the student progressed. Hannibal's view of his undergraduate educational progression was that

> ... it was definitely student-professor in the first few years. But then the student I think came more up to the level of the professor, from the student's point of view and probably from the professor's too. It became, instead of being more of an independent, I mean I guess you could say independent, but you could also say either a colleague or working in groups or something, just as much probably. (2a.InterviewHannibal15.2.99/US/p.11/l23-27)
>
> But it became much more of a dialogue than a monologue, really I think, in the last few years. (2a.InterviewHannibal15.2.99/US/p.10/l37-38)

Since this view of the development of independence, and largely self motivated exploration is the key Constructionist approach required for successful designing, and that promoted in the course, it is therefore not surprising that the five students who already had this awareness

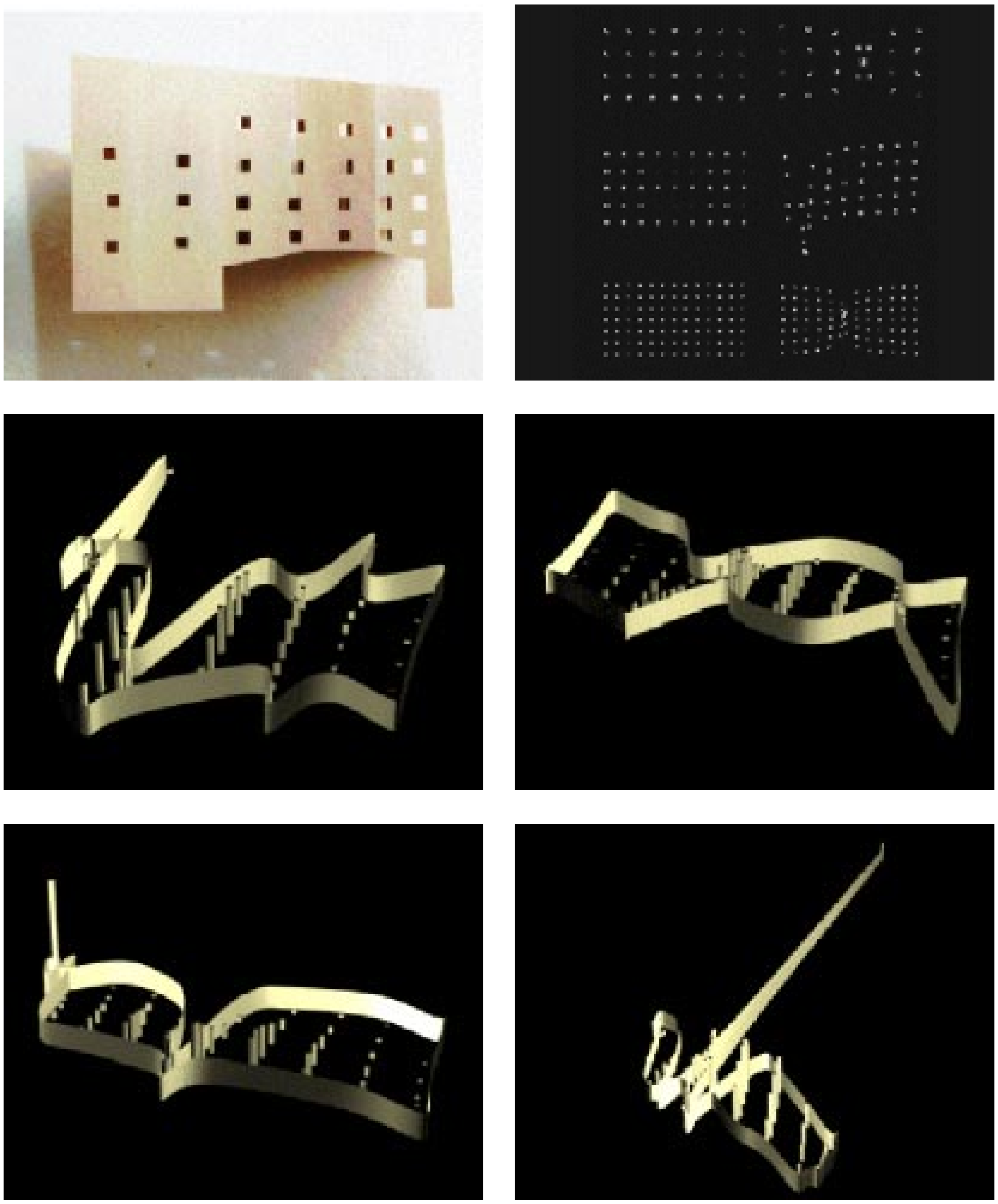


Figure 64. Michele Auer was one of two previous students to (a) be exposed to LOGO in grade school, and (b) continue with her studies into the second semester.

showed little change during the research study. One of these five students, Jeremy, had slight Behaviourist assumptions at the beginning of the course that the primary role of architectural education should be about teaching facts and knowledge.

> I think ... in architectural education there needs to be much more [of an] information base, rather than design teaching base, especially later on. ... I think there's a lack of information passed, knowledge, ... because you come out of university, and we don't really know anything still. And that's kind of a problem. You get to practice, and you're useless to anyone until another year and a half when you start to pick up some knowledge, and you're actually useful to a client.
> (6a.InterviewJeremy23.2.99Div/US/p.9/l15-23)

This assumption changed by the end of the study through his realisation that the most significant aspect of architectural education was the ability to question himself and his assumptions when designing – becoming more appreciative of the cognitive problem-solving abilities he had learned.

> I just think this, this way of training your mind to think rigorously about when you're doing that step to when you're doing this step, I mean I just found it really useful for writing a paper. Just because you suddenly realise whether that's an assumption you've just assumed and then you kind of realise, you have a chance to look back, and it really just gets my mind thinking in a way that I think is quite useful.
> (FinalPres10.5.99Div/US/p.36/l19-23)

However, it is the sixth student, Josephine, who demonstrated most clearly the effects of the course on someone with a highly Behaviourist approach to their education process. In the first interview, Josephine repeatedly stated an awareness of needing someone – a professor – tell her what to do and what next step to take, to define guidelines and set goals for her in her design studio. By the end of the study, six months after the end of the semester, this Behaviourist approach to education had changed dramatically, and Josephine herself attributed the change to this course.

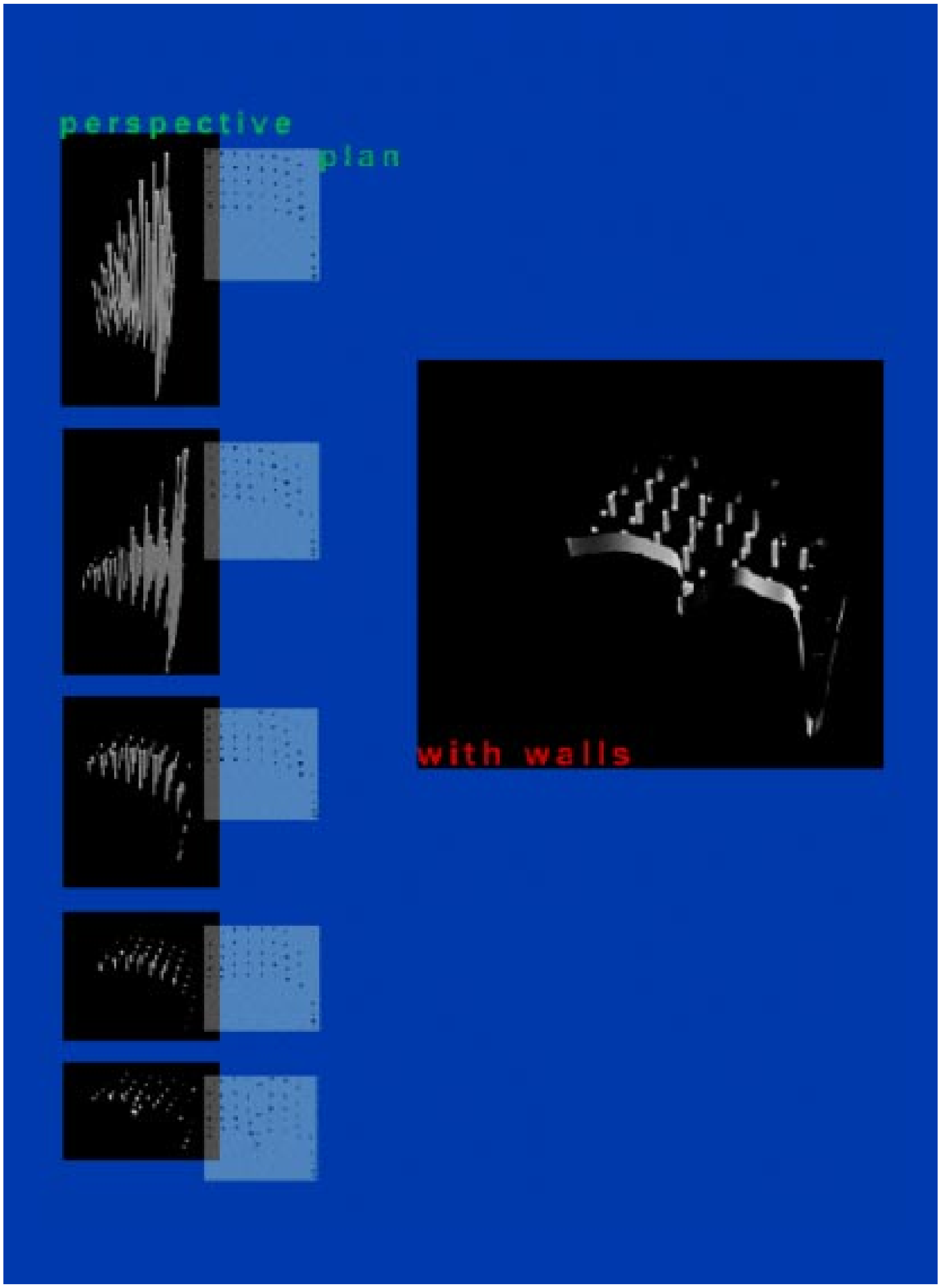


Figure 65. Like Anthony Guma, Michele Auer used the first semester to explore an aspect of her studio project, developing a simple two dimensional grid deformation into an interactive, 3D digital 'clay'

Case Study: Development of Learning

Josephine originally studied architecture at undergraduate level, and her view of the education she received was primarily a description of the studio professors giving her explicit instructions of what steps to take next in her design project.

> And I had no idea what to do [in this undergraduate studio], and I felt like, I didn't like the assignment at all, but you know, he had to basically tell me everything to do, because I just couldn't generate the ideas. So he told me, why don't you document movement, and look at photographs of [this]? So I did, and then I did these stick figures, and then I did these graphical things, and reduced them to something that you could look in plan and section. Whatever. And like, I'm, you know, and he had to push me through all this stuff.
> (3a.InterviewJosephine21.2.99Div/US/p.8/l7-13)

Although Josephine appeared to have the knowledge and skills to progress on her own in the design studio, she had very little self-confidence and believed strongly that other people know more than she did.

> I don't want to just constantly go through like MIT being told by who happens to be my professor what I should do next. Like [one professor I had] was always drawing a Corbusian diagram ... I mean, I know some people, like they just know a lot for themselves, they do a lot of research for themselves, and they educate themselves a lot, but, I don't know.
> (3a.InterviewJosephine21.2.99Div/US/p.25/l31-36)

Her view of the role of the studio professor was one in which the professor should work with her at her level, adapt himself to her needs and tell her what to do

> I think it's really important for a studio professor to get down into the level of your mind, and work with you. (3a.InterviewJosephine21.2.99Div/US/p.7/l4-5)

Josephine did express an awareness of, and some concern over, her lack of self motivation

> I need to have guidelines, you know, and goals. Because then I can just wander around, and focus on like a little stupid detail and, and not have enough sense of like what else I should be doing. So, I still can't structure things for myself.
> (3a.InterviewJosephine21.2.99Div/US/p.6/l8-11)

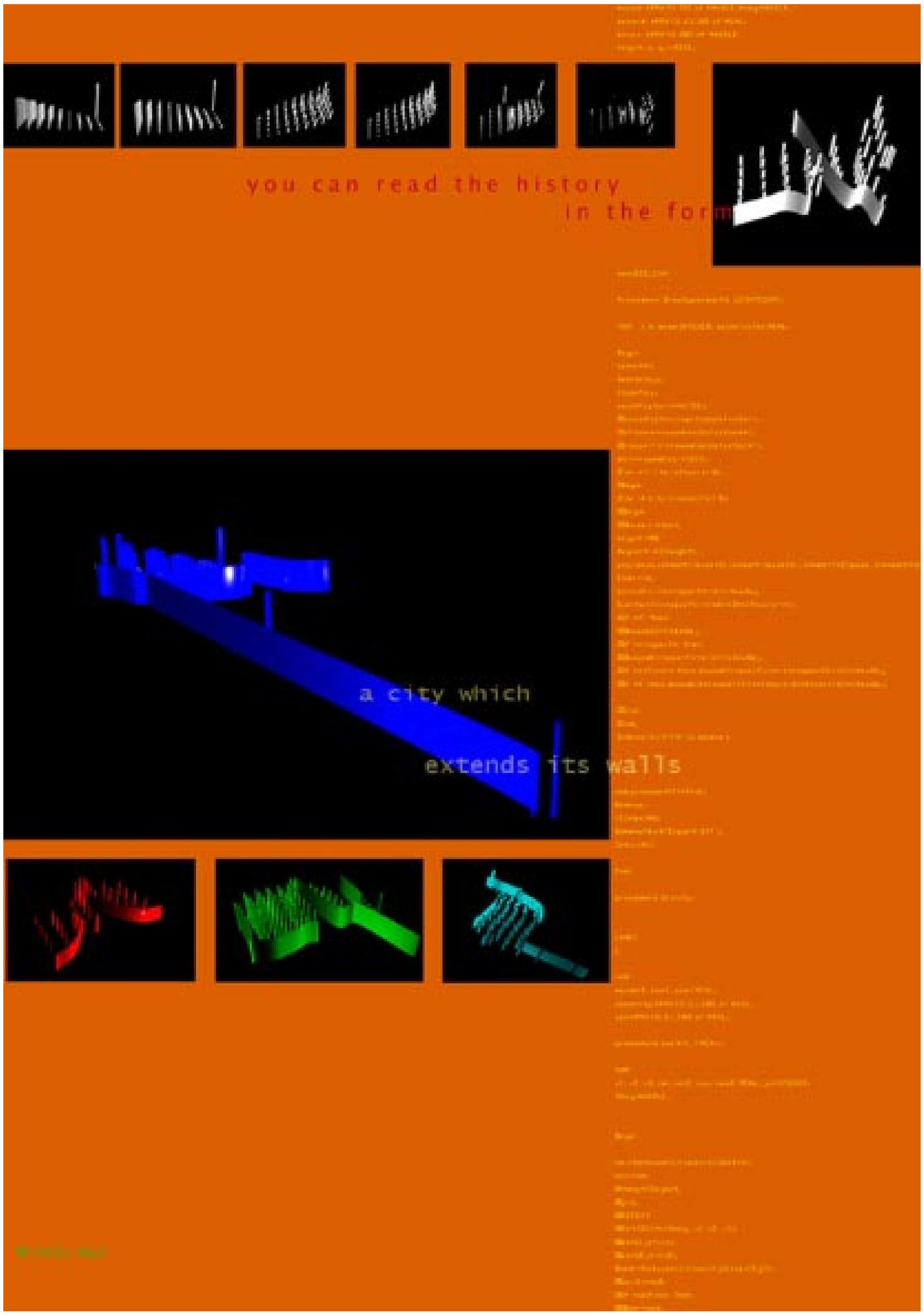


Figure 66. Like Anthony Guma, Michele Auer used the first semester to explore an aspect of her studio project, developing a simple two dimensional grid deformation into an interactive, 3D digital 'clay'

> That's been my biggest fear in school though, is like, one of my more successful studios was when I was really pushed and told, and kind of steered in which direction to go, which was like my senior year. And I, I was really happy with the work I did, but I felt like a lot of it was because I was really pushed into certain directions by my teacher. And, that's still something I'm really struggling with, in terms of how to move from one thing to the next.
> (3a.InterviewJosephine21.2.99Div/US/p.4/l13-18)

She was still searching, largely unsuccessfully, at the time of the first interview for a mentor at MIT to perform this role, something she recognised in the final interview.

> [My studio professor at my previous University] thought of [it] for me, so, ... [it] helped, and ... pushed me into other realms. And that's something, like no one here has really done because I mean, I feel like a lot of people work closely with you here, but depending on how much they like you
> (3a.InterviewJosephine21.2.99Div/US/p.5/l5-9)

> I thought at a small school [like MIT] I would find this mentor kind of person that would be really there for me and watch over me those three years and tell me – like see my progress and – you know almost like when I took like piano lessons when I was younger. And it's this private thing and you really see your progress and they really talk to you one-to-one. But it's not like that, you know. This is your studio, that's your professor, that's it, goodbye like. No one cares, no one fills out the evaluations at the end, so you have no idea what they meant when they gave you a grade, which doesn't mean anything anyway.
> (InterviewJosephine12.11.99/A4/p.14/l7-14)

It was clear through talking to Josephine that she felt out of control of her own education process. When asked about a personal process of design, she answered not about developing her own, but stating she had not yet found a professor who was interested in helping her do that.

> I don't even know if any professor I've ever had has even been ever a thought of being concerned about my personal process or cultivating that for me. In fact, I, it doesn't occur to me, ever since I've been here [at MIT], that anyone has been concerned with ... my own individual process versus thinking along their lines of thinking. [Pause] Maybe they have and I just didn't realise, I never really thought about my own process. (3a.InterviewJosephine21.2.99Div/US/p.10/l31-36)

When describing collaboration with another student, she clearly expressed feeling out of control of, and no responsibility for, her own work.

> It stinks. I hate [working in groups], because ... it's like always someone has control, you know what I mean? And if that person is in control and you're not then you kind of feel like, alright, let's just do what they say, and you feel kind of stupid and you don't feel like you learn anything. And I always end up in these like science classes with like the smart girl that has no friends but decides to make me her friend. And, like, I have no idea what's going on, but she writes the lab report, and I get an A, which is great, but like I have, I didn't learn a thing
> (3a.InterviewJosephine21.2.99Div/US/p.22/l23-30)

By the second interview, Josephine had begun to change her view of her own role in her education. She was experiencing difficulties with her studio professor and had decided to continue with her ideas somewhat independent of him.

> I didn't get anything out of [my last meeting with my studio professor]. I feel like I'm basically working on my own, and, I think he's just, he doesn't really feel like I, he has anything, like to talk to me about.
> (3b.InterviewJosephine5.4.99Div/p.10/l15-17)

Although this on its own may be seen merely as defiance, it can be seen as the beginnings of a Constructionist viewpoint when framed in the context of other comments she made. She had begun to be aware of her own responsibility in nurturing design ideas.

> I don't know, it's much more, you know, you don't, it's much more of my responsibility to ... nurture it, this ... idea, as far as it can go, in this project
> (3b.InterviewJosephine5.4.99Div/p.10/l27-29)

In the course she was beginning to realise that development was not about learning new code – learning facts in a Behaviourist model – but about asking questions of herself and her own ideas, becoming aware as she did so that she should try and evolve these ideas more than continuously beginning again.

> And ... one of the big hurdles was this whole idea of ... how you make progress in code, not by learning new code, but by learning to question why you do what you do and what to do next and why. Like all the answers to the whys. ... So like, just that

> whole answer to the why's and why not, and, or why should I make a new program .... I guess learning to make things evolve more than starting, trying to start new things constantly. (3b.InterviewJosephine5.4.99Div/p.13/l13-24)

This 'all or nothing' model of learning is a Behaviourist view that contrasts with the Constructionist's tinkering view of evolution of ideas. Although Josephine admitted in the final presentation remaining dependent on asking others for help with writing the syntax, and that she preferred to learn through other people telling her what to do, she also expressed her realisation that it was up to her, that the computer was offering scaffolding for her to find a way to continue when stuck with her design ideas

> I don't like figuring stuff out, I like people to show me, like I like to learn by people like just telling, or showing me and then I can see that way, but you know I realise, but you know and I really wasn't excited about it for a long time, because I was stuck, but after that, you know I said, it's not so bad because I can do things. (3c.FinalPresJosephine10.5.99Div/US/p.10/l12-16)

She also expressed a desire to be more in control of the process.

> I want to be able to program using, you don't have to learn the code and stuff, because it will only empower me in like efficiency and time, you know. I mean, it's always better to know how to drive the car than be a passenger, you know. It's just easier. (3b.InterviewJosephine5.4.99Div/p.21/l25-28)

She had begun to view the course as a process of Design Inquiry, and made a significant differentiation between 'design' and 'designing':

> I guess I would say that like, that, you know, it's it's a whole different way of like kind of ... Design Inquiry, ... inquiry into designing. I mean, I don't even know if I would say design, but – yeah, designing (3b.InterviewJosephine5.4.99Div/p.14/l14-18)

She stated that the coding process was about learning to design in pure form,

> Too much of the time in studio people talk too much about all these other issues, and different, the most important issue [of design process is] skirted around. It's easy to skirt around it, because like I think it's the most painful and the hardest one. And, and, you know, it's so easy not to be completely raw about it, in studio. Whereas like, you know, maybe ... in your programming, it's ... the pure code, it's ... what you're

> going to do with it and how you evolve it is, is like, you know, is your process, so, it's nothing more, it's like kind of naked like that. You know, whereas in studio it's so easy to, ... shroud things, and then not have any seed ... it's almost too easy, actually. It's too easy to make pretty drawings and ... models, but ... be inconsistent. (3b.InterviewJosephine5.4.99Div/p.15/l8-22)

and said her recommendations for the course would be influenced by how interested the other party was in learning more about the design process.

> I guess [I would recommend taking the class, depending] on how much they were really willing to, come, how concerned they were about the design process, I mean, because if they were the kind of person that like didn't really care, and they really were just interested in issues and stuff, well, you know, I don't think that's that great, but if that's what they care about then ... I can't tell them to take this class, because then they're just going be ... either and totally blown away and enlightened ... or just they won't get it. (3b.InterviewJosephine5.4.99Div/p.15/l8-22)

By the final interview, Josephine was showing significant signs of a Constructionist approach to learning, attributing to the course the realisation that it is up to her to help herself.

> [I realised] if I don't help myself I'm only going to have more problems, you know. I have to get to the root of the problem, and help it. (Pause) So I mean obviously like you know the class was interesting cause it really made me think about [it], even if it's just a little process, with little stuff, just you know – that was nice. (InterviewJosephine12.11.99/A4/p.13/l39-43)

She was still fearful of the process of writing code, but admitted both that there was distinct potential in the process for exploring her own ideas,

> I learned that learning program for me is very slow and hard, but I mean it definitely has its rewards. Also that just ... my skepticism was so strong, you know. So I don't know, it's important not to be overswayed by that sometimes too (InterviewJosephine12.11.99/A4/p.13/l39-43)

and that she felt capable of being able to have done an Independent Study this semester to link the programming processes with her current design studio.

> [My current studio is] very ideal for programming in a lot of ways. Of course, but the thing is I'm not going to do it unless I have someone's hand to hold during the

> programming, you know, like really helping me. ... But ... maybe I could have done an Independent Study you know. That could have been interesting too. ... In fact, it would be, I think, very good for this site.
> (InterviewJosephine12.11.99/A4/p.12/l11-20)

She said that the most significant thing she learned was that programming was not about absolute control, but about tinkering with the code to get the desired results.

> And then I realised that the programming isn't like this absolute control thing. It's this thing you use to tweak things so you can sharpened your preferences more for what your thing is doing. (InterviewJosephine12.11.99/A4/p.9/l4-6)

Later in the interview, she made the same statement about her approach to design, seeing her previous work as being rigid and controlled, and realising that her approach had changed to focus more on the process of development.

> I think I'm a lot looser now. I'm like – I change things a lot ... I think I try to [keep] as little as rigid as possible. I think I used to immediately start designing in plan with door swings and everything. Getting all my dimensions right; it had to be this absolute thing from the beginning, so I could move pieces around. But I try to stay a little looser now. (InterviewJosephine12.11.99/A4/p.12/l28-32)

She attributed to the course a change in attitude from caring about what other people were expecting, to concentrating on discovering what was important for her.

> I care less about the school requirements now and it's more about what's good for me. I, well, that's been the big change, right? Especially after you know your class
> (InterviewJosephine12.11.99/A4/p.13/l30-32)

Finally, she realised that the course taught her the significance of documentation in process, and that it had influenced her to look at architecture in different ways and to find different ways of designing.

> I wanted to use your workshop as one of my concentration [classes, looking at] new ways into thinking about designing. So it's influenced me in terms of that. ... It's influenced me in terms of seeking more paths, and looking at architecture and different ways (pause) of designing. (InterviewJosephine12.11.99/A4/p.11/l23-28)

Although it was hoped that the course would influence, or would be a part of, a more gradual change from a Behaviourist understanding of learning to a Constructionist, self motivated process of exploration, the strength of change in Josephine's approach, and the extent to which she attributes this change to the course is quite surprising although certainly not unwelcome. She began the semester wanting to be told what to do by someone she thought knew better than her, and ended the study nine months later by expressing a realisation that it was up to her to develop her own design process through tinkering with ideas and documenting the progressive results. Her parallel statements about relinquishing control over both programming and designing, and realising that progress in both was a process of tinkering and documentation of development, suggests success in the research's original statement that perhaps the best way to teach the design process is not through the design studio, an application of the Papert Principle. Josephine's attribution of this progress to taking the course offers interesting, if speculative, insight into the potential for teaching this course to beginning design students.

Transformative Educational Experience

One of the significant aspects of Josephine's development during the course of the research study was the change she made in her assumptions about what the course would teach her. She was not alone in this change. All of the students began the semester by stating in the first interview that the reasons for choosing to take the course were to learn more about computer programming as a process or tool to be used in design:

> hopefully with this class, on a pragmatic level I want to be able to understand [and] get a better view of what programming can do. The possibilities of programming, and some programming language. I also want to be able to use that as another set of ideas and tools that I can use in conjunction with other things. At the end of the class I'd like to ... be able to go through a process where I, as much as I'm learning this pragmatic sense I develop, a message or an idea or a process and use, and can be able to refer back to that. And not, not just emulate that process, but refer back to that process and be able to implement that.
> (4aInterviewRaphael19.2.99Div/US/p.33/l1-10)

When asked, however, in the final interview what they considered to be the most significant aspect of learning in the course, all of the students said in some form that it was a new understanding of their own design process. This represents a transformative educational experience for the students. They started the course with the assumption that computer programming was a tool to be learned. They ended by realizing that the programming techniques learned in the course had transformed the way in which they understood their own design process. This realization was stronger in some students than others. Hannibal made the least progression, talking in abstract terms about understanding his design process, but only implicitly comparing coding and sketching, and showing few concrete examples of transferring this understanding to his design thinking, despite talking about the possibilities of doing so in abstract terms:

> this semester, the use of the code and the use of my drawings and the use of the models have been all, more or less three different explorations: one that was about image ... one was about a sketch and one was about the realization of the other two, so I think in that sense they were all very very different. And the fact that they were different but they all influenced each other, I think that's probably, I guess with any

> design process [the thing] to learn is to understand not this, this and this, but probably the space between all two or all three, and to understand what the spaces, or if you could talk about it by the boundaries as opposed to the spaces, its probably much more interesting or it's much more beneficial.
> (InterviewHannibal11.11.99/A4/p.10/l28-36)

This is perhaps not surprising bearing in mind Hannibal's extensive focus on programming as a tool, and consequent difficulties in true appropriation for designing. Esther made a significant discovery for herself when she realized that the computer could be used as a design tool to help her think, although this knowledge had yet to be internalized to affect her design process.

> [The most significant thing I learned from taking the class] was the experience of dealing with programming, and begin to, making me think from, the look of design from a different way ... you can use design tool to help you to think. Although I didn't really let it to help me to think during last semester but because I was too much wanted to control everything (InterviewEsther13.11.99/p.9/l32-36)

Raphael stated he had learned more about his own design process as a result of taking the course,

> I think that the one great thing that I got out of it I think was that (pause) I started to learn not just about programming but the way I design, and how I can use programming within the way I design, so – Or how I can design with it.
> (InterviewRaphael12.11.99/A4/p.13/l6-8)

changing his view of what he had intended to get out of coming to MIT:

> before I came here [to MIT] I wanted to look at some of the new digital tools that are coming out, and seeing how I can use that for designing, in terms of architectural artifacts. And I'm not necessarily interested just in that. But I noticed the way I used to design before was so regimented, in a sense. Not regimented, but it was – I mean, people have worked just in sketching, you know, and I'm not saying that that is regimented in a sense, but I also feel that there are new tools that you don't necessarily apply the way you design to it, but you also adjust the way you design. And in that sense it was a revelation to me. Now I can learn something where I can hopefully well use and not trying to fit the way I work but to really adjust to the way, the possibilities of, of other things. That's why I think I learned more about how I designed. (InterviewRaphael12.11.99/A4/p.13/l25-35)

Josephine also expressed an awareness of changes in her design process as a result. However, it was Jeremy and Gary who made the most significant advances by internalizing the processes and stating they had learned a new way of thinking. In the final review, Gary said:

> it's a way of thinking, it's been said already, but ... if you translate a spatial idea or an architectural idea into ... another form that you can tell the machine, and then it can produce something for you, [then] throw the computer way – that sort of movement, that sort of reassessment of the thinking of your ideas is one of the biggest learning things that you have. (FinalPres10.5.99Div/p.62/l15-20)

Jeremy also said that his new way of thinking was independent of using a computer and that he had already applied it to other areas of his work.

> I think that [this] way of thinking has already affected the way that I do projects, as I, just to make sure that I'm rigorous about what I've chosen before, just the structure of how I'm working. And in a way I don't think, in a way the computer is not totally necessary for that (FinalPres10.5.99Div/p.40/l19-22)

This change, from deciding to take the course in order to learn about computer programming, to admitting that the most significant thing learned was a development of a personal design process, and the fact that it occurred in all six students, supports the most significant aim of the course; that the development of a personal design process is perhaps best taught away from the design studio and in a course with a Constructionist pedagogical approach, and using software with similar Constructionist characteristics to LOGO. It also supports the recognition that the course is best understood by those who have taken it. As Josephine said in her final interview:

> [Taking your class] just really changed my perception of what your class was really about because I still believe no one can know until they do it, and I really totally understand now why you and [the TA] say that everyone says that. I really believe that you can't just explain it to someone, you know. It's really an experience to go through I think, especially if ... you have certain tendencies that you need to [explore]
> (InterviewJosephine12.11.99/A4/p.10/l26-31)

As Raphael said:

> there's a myth around programming where like, I don't know, ... a lot of people don't know what it really can do. And that was me also. I don't think anyone really does until they start seeing what programming is about and how you can go about using it. And some of the limits also and possibilities.
> (InterviewRaphael12.11.99/A4/p.13/l2-6)

Collaboration: Intrapersonal versus Interpersonal Skill Development

Despite growing evidence that the abilities to both work within, and manage, large groups of people are critical for success in the architectural profession, architectural education continues to emphasise the importance of the individual, particularly in the design studio. Collaborative and interactive skills are occasionally encouraged, but usually not specifically taught. It appears to be assumed that these critical skills, like so much else in the architectural profession, will be learned in practice after the student has completed her education.

In the field of education, it is becoming recognised that the importance of the social, mental, and physical contexts within which a student is learning cannot be underestimated. Indeed, it is also being recognised that there are gains to be had in education terms if more attention is paid to the effect and benefits of these contexts. In particular, the theories of Socio-Cultural learning imply the benefits to be found in developing a social setting within which students can learn from each other and from a broader community.

There is therefore a twofold need for architectural education to recognise the importance of Socio-Cultural learning, both for the interpersonal skills required, and for the educational benefits of collaboration. All of the students in the research study recognised the importance of teamwork in the profession.

> I think of course collaboration is becoming even more important, there's more opportunity, I think, to collaborate. Things are in one sense in the architecture profession there's a lot of specialisation, or there's, there's a need to [organise] now in terms of [working with] certain engineers, certain even, people who do signs, signage, graphic, industrial design, coordinating things.
> (4a.InterviewRaphael19.2.99Div/US/p.30/l7-11)

> I think probably [collaboration in] the office is a different thing, the workplace is different, much different in terms of [collaboration] and I think ... it works ... much better [than] the studio environment sometimes
> (1a.InterviewGary18.2.99/US/p.23/l27-29)

With the exception of Gary, all of the students had had experience of such collaboration in practice, finding the environment conducive to further learning and development.

> [W]hen I had my job in New York, and I [worked] in a team, there was three of us ... I liked my team a lot, it was just, I really liked the people, they made me feel like I was

> equal to them even though I was half their age, you know. And they gave me as much responsibility as they had, and that was really nice, and I learned a lot. And they treated me like a normal person, not like a kid.
> (3a.InterviewJosephine21.2.99Div/US/p.23/l30-36)

However, they recognised the differences between collaboration in the design studio in architectural school and in practice.

> [I]n a company, ... people work together because ... they just do this project. They concentrate on this project. They put their whole energy on this project. [All] of them, you know, just do whatever they can. And it's [a] very pleasant thing. But at MIT everybody [takes] a lot of classes, and everybody is busy, ... [so] some people just don't want to do this work, that were in this group. If this person didn't finish ... his work, so somebody else will be affected
> (5a.InterviewEsther16.2.99Div/US/p.17/l4-11)

All of the students had either had experience of, or were aware of, the difficulties of collaborating in the design studio. Often conflicting schedules made getting together difficult, different members of the group contributed different amounts, or placed differing importance on the work done, and the prevailing culture of the individual made competition rather than collaboration the predominant flavour.

> It was very tough [working with someone else], just because, and it's unfair of me, but I probably expect as much out of them as I expect out of myself. Which, if I push myself too hard then I would expect them, that of them. Which is unfair to do. So it's tough to work because working styles are inherently different
> (2a.InterviewHannibal15.2.99/US/p.20/l3-5)

This contrasted with the students' experience of team work in practice where the emphasis was success for the project and not for the individual, the structured hierarchy of the office was clear, team work was valued, and there was a commonly shared goal – the development of the design project – towards which all the team members were working.

It is considered important in education to monitor the progress of individuals. Faculty regulations of institutions such as MIT specify that in situations of group working, clear information should be given to the students about the way in which the individuals of each group will be monitored. Architectural education recognises the difficulties involved in assessing the contributions made by individuals in open ended, qualifiable projects such as those in the design studio, but has done

little to address the conflict between this and the desire for collaborative skills in practice.
In previous years of teaching the course, there was extensive collaboration between students whilst working on individual projects. This collaboration has been anecdotally noticed by others in computer studios elsewhere. At a recent conference on Digital Creativity I attended in London, there was general agreement amongst those present that this mutually beneficial collaboration was a natural and automatic result of the computer room environment. However, the unexpected results of the Spring 1999 course studied in this research demonstrated that, contrary to all previous experience, this collaboration is not automatic and should not be taken for granted.

> [In our class] everyone's so serious, and staring at their monitor, and I kind of wish, you know, either, it didn't have to be like a show and tell session where you told everyone to pin up, ... but naturally kind of hang out, ... it feels really kind of serious, ... not as chatty, you know, chatty in a good way, you know.... Like going up to someone and asking them how their thing is going, and just asking to look at it ... takes people off guard sometimes. (3b.InterviewJosephine5.4.99Div/US/p.16/l6-15)

Reasons for the lack of collaboration between the students in the Spring 1999 course are unclear, but it is possible to suggest potential reasons through looking at the difference between this course and those of previous years. In the first few weeks of the semester, prior to the students beginning their individual projects, they worked in teams to solve programming exercises set in class. Team spirit was developed by encouraging competition between teams, usually through seeing which team could solve the problems first. As with previous years, this approach broke down interpersonal barriers, and through much laughter it was assumed that a precedent would be set for later team work. Also as before, teams ended up helping other teams. In previous years, this helpful attitude prevailed in the course as students helped each other out with the difficulties of writing code at times of day when I was generally unavailable. The detachment of investment from the design product through the intermediate stage of writing code encouraged this process and the students appeared to learn valuable and enjoyable interpersonal skills. They valued each other's input, and helped with the common goal of writing more successful code.

It is possible that the Spring 1999 class simply did not have the semi mystical 'chemistry' for group working so many of the students referred to as being important for successful collaboration in the design studio.

> I thought that [those two students] worked well together, but that's chemistry too, and ability, so it's, that was a lucky team. But I've never found someone I could really work with. (3a.InterviewJosephine21.2.99Div/US/p.23/l26-28)

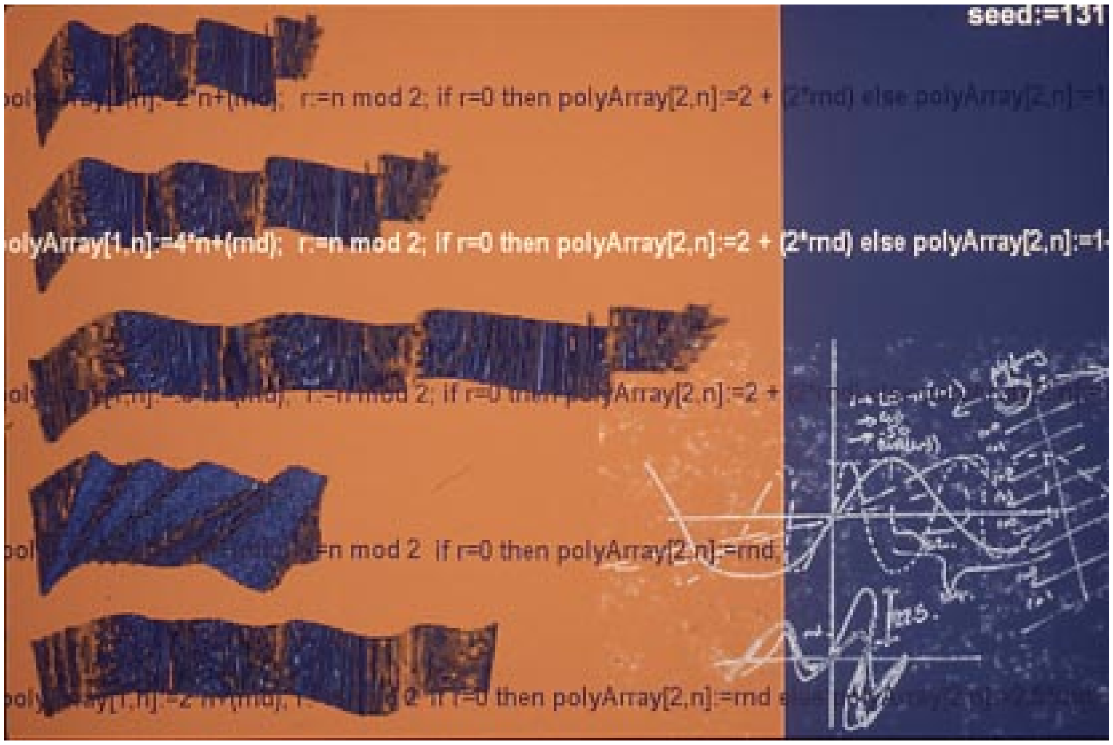

seed:=131
r:=n mod 2; if r=0 then polyArray[2,n]:=2 + (2*rnd) else polyArray[2,n]:=1
olyArray[1,n]:=4*n+(rnd); r:=n mod 2; if r=0 then polyArray[2,n]:=2 + (2*rnd) else polyArray[2,n]:=1
Array[2,n]:=2 +
n mod 2 if r=0 then polyArray[2,n]:=rnd
then polyArray[2,n]:=rnd

Figure 67. Talia Braude was the second student to have both used LOGO in grade school and develop her ideas in the second semester.

> I think, [successful collaboration] depends on the people. Like ... if you respect the person, that you're working with, and they respect you, and, you ... produce something, that's the optimal kind of ... collaboration, I think ... and you bring different qualities to that collaboration.
> (4a.InterviewRaphael19.2.99Div/US/p.32/l2-6)

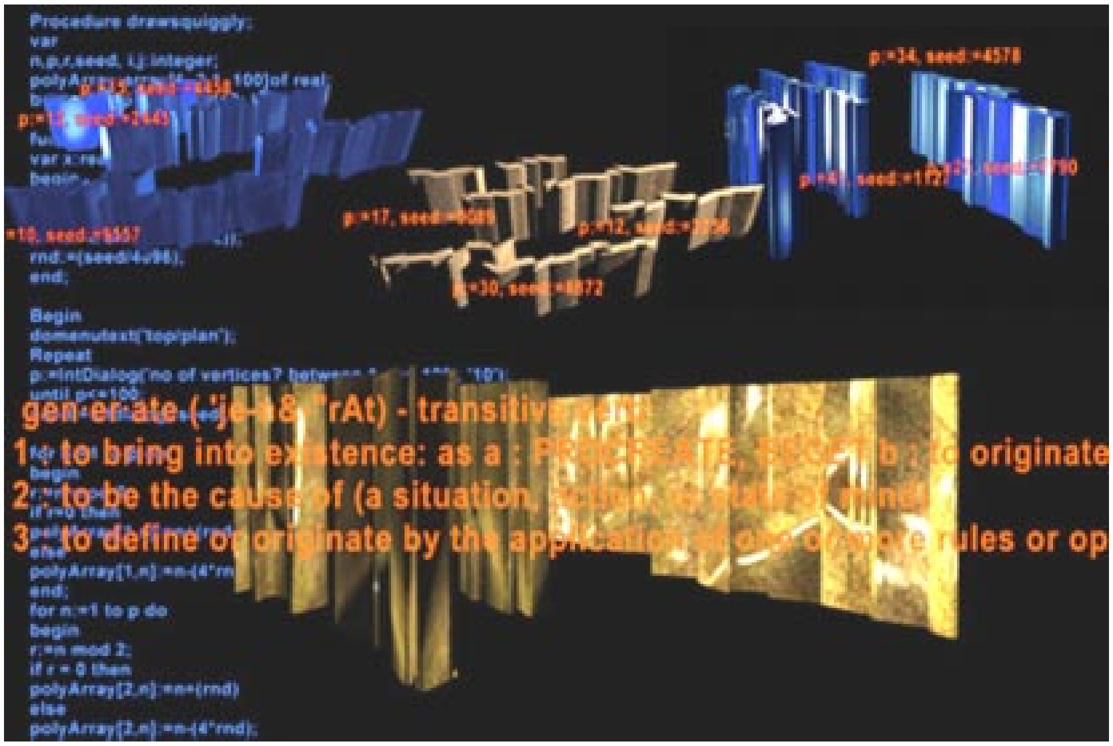
It is not possible from this study to establish the exact cause of the lack of collaboration, but there is nothing to indicate that specific group dynamics is the sole reason. Although the students were aware of potential difficulties in collaboration at school, all of them saw benefits in doing so, and did not indicate any desire to not collaborate at any time.

> I think we should [work in groups] much more, because [I] actually got so much out of it ... different points of view, ... if you're working with people you're suddenly forced directly, you're actually talking with them, you can't just go, no it's wrong. You've actually got to express why, so therefore you get into a conversation, therefore you suddenly realise, oh, wait a minute, this is actually much better ... and different ways of representation which suddenly you can see why they were drawing something in a certain way, or doing something in a certain way, which you were reading as different. And suddenly you see, oh really, that's useful. And it's also quite fun. ... it's less intense, and you can talk about it. I mean I think there should be much more (6a.InterviewJeremy23.2.99Div/US/p.23/l1-14)

> I think this [collaboration is] also a learning process, because, ... when I look at it in the end it's something that if you can say well, you know, this turned out pretty nicely, and I don't think I could have done this, so who's to say that that was, that wasn't a better way, or if there was a better way.
> (4a.InterviewRaphael19.2.99Div/p.32/l10-15)

One student, Josephine, complained that this course did not involve as much collaboration as she understood had occurred in the previous year's course.

> Someone was telling me from [the class] last year that it was really fun, like that everyone really talked a lot about what they were doing, and I just I don't know, it's not that I don't like our class, I like everyone in our class a lot, but I don't feel like we talk enough about what each [other is doing].
> (3b.InterviewJosephine5.4.99Div/US/p.16/l2-6)



```
Procedure drawsquiggly;
var
n,p,r,seed, i,j:integer;
polyArray:array[1..2,1..100]of real;
[illegible]
fu[illegible]
var x:re[illegible]
begin
[illegible]
rnd:=(seed/4096);
end;

Begin
domenutext('top/plan');
Repeat
p:=IntDialog('no of vertices? between [illegible]', '10');
until p<=100;
[illegible]
for [illegible]
begin
r:=[illegible]
if r=0 then
[illegible]
else
polyArray[1,n]:=n-(4*rn[illegible]
end;
for n:=1 to p do
begin
r:=n mod 2;
if r = 0 then
polyArray[2,n]:=n+(rnd)
else
polyArray[2,n]:=n-(4*rnd);
```

p:=34, seed:=4578
p:=13, seed:=2443
p:=41, seed:=1127
p:=17, seed:=6086
p:=12, seed:=2856
=10, seed:=6557
p:=30, seed:=6872
gen·er·ate (ˈje-nə-ˌrAt) - transitive
1 : to bring into existence: as a : PROCREATE, BEGET b : to originate
2 : to be the cause of (a situation, action, or state of mind)
3 : to define or originate by the application of one or more rules or op

Figure 68. Like Debbie Kim, Talia Braude linked her Digitally Mediated Design project with her current studio, where she was exploring curvilinear walls as a means of enclosing space

She said she had very much enjoyed the course pin-ups as a time for beneficial group discussion of other people's work.

> I remember at the beginning we were all kind of complaining because we didn't know what each other was doing. And then we'd have to literally just physically bring ourselves to someone else's computer and see what their code did. But even then on the computer screen it's not the same as when you print it out. So, I really liked that a lot. Class pinups were great. (InterviewJosephine12.11.99/A4/p.6/l6-11)

It is my belief that the lack of collaboration in the group, that had occurred in previous years as the students helped each other with syntax difficulties, is not solely the result of coincidental group dynamics, but rather the unexpected consequence of having a particularly helpful, conscientious and hard working teaching assistant. The Spring 1999 semester was the first time the course had had a TA to help the students with their code outside of course hours. There is no doubt that the efforts of this TA resulted in students who were able to get further with their projects because she was on hand in the studio for the majority of the day – and much of the night – to help when the students experienced syntax errors. This effort lessen the gradient of the learning curve involved, and enabled the students to more quickly develop their design ideas. Yet at the same time, it is now clear that the provision of a freely available expert permitted the students to resort to a Behaviourist model of expert and novice, and to avoid overcoming the initial inhibitions many students have when asking for help from their peers.

If the assumption that continuous and freely available expert advice inhibits the collaborative atmosphere observed by many of my colleagues in the computer studio is correct, then it is perhaps true to say that this collaboration is not an automatic response to the digital environment, but rather to a shared desire to learn in the absence of expert help. And if, as has been stated in this thesis, design is a constant process of learning, and in practice there is no expert to dictate how a design team should develop the design, then there are useful parallels to be drawn from the two environments of successful collaboration in learning and designing in the absence of expert assistance.

This potential poses a dilemma for the future of this course. The provision of a good TA results in more developed projects and greater success in the course's original aim of developing an understanding of a personal design process. The lack of TA appears to encourage the development of collaboration and interpersonal skills whilst inhibiting personal project development. The course provides a rare time in which the students feel comfortable in asking each other for help, yet the provision of a TA eases the extensive burden on the professor teaching alone. The success of the future of the course depends on finding a balance between intrapersonal and interpersonal development.

Conclusions

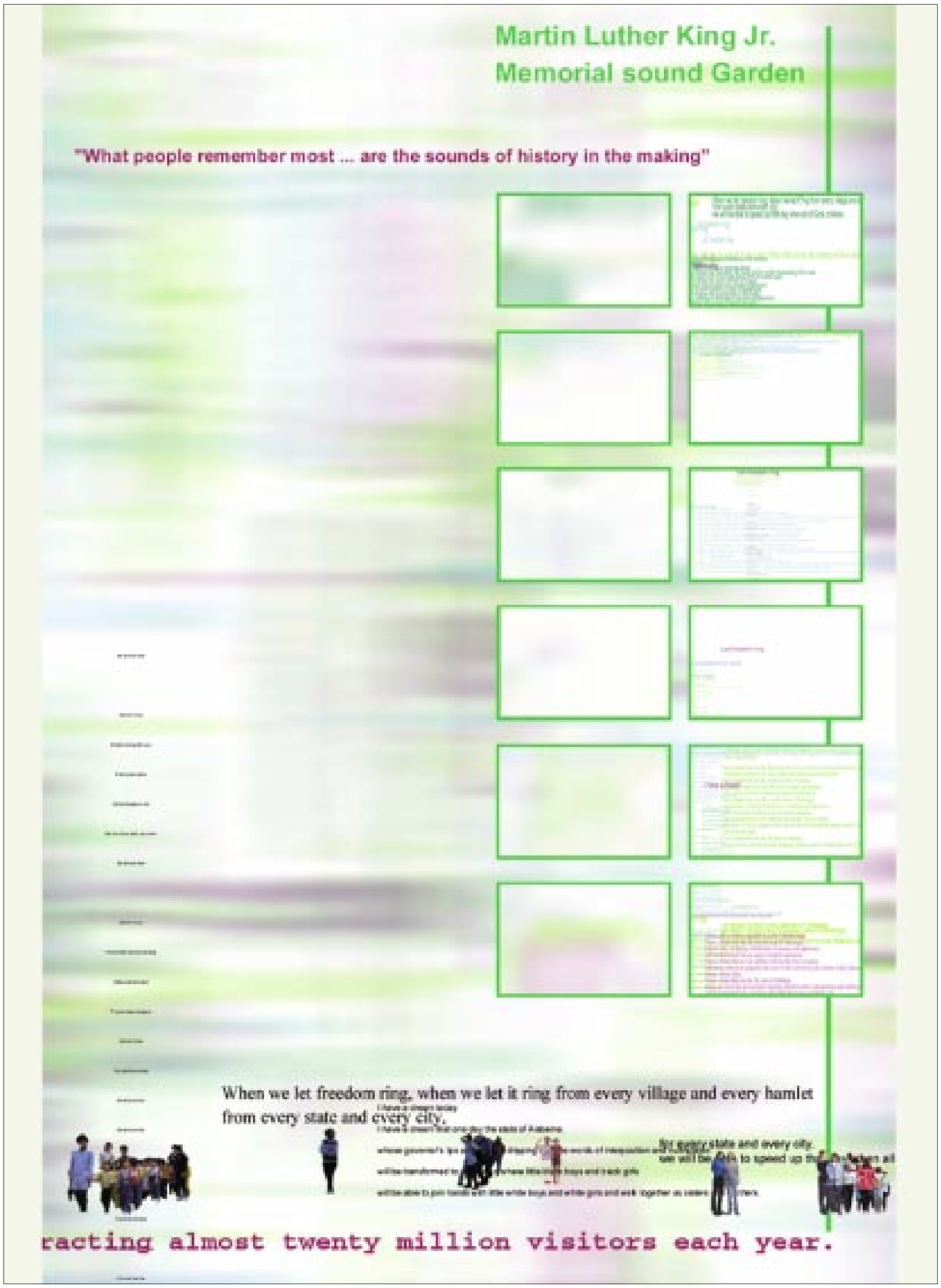

Martin Luther King Jr.
Memorial sound Garden
"What people remember most ... are the sounds of history in the making"
When we let freedom ring, when we let it ring from every village and every hamlet
from every state and every city,
racting almost twenty million visitors each year.

Figure 69. Talia Braude joined the Fall 1999 semester class, her fourth semester of looking at the concepts. It is therefore not surprising that she showed the most appropriation of both the tools and techniques

Conclusions

Although some of the results of this research study were to some extent predicted from previous years of teaching the course, there were new findings, and the extent of some of the expected results was somewhat surprising. Five stages of development of understanding of design process were seen in all of the students, although the speed with which they progressed through the stages varied enormously. These five stages of development were also five stages of computer appropriation, since learning to design in this way necessarily involves breaking through the barrier of focusing on the computer as a tool, enabling a closeness to develop with the objects of design. In traditional designing these objects are sketch models and drawings, but in the context of the course the objects become both the computer code and its resultant graphical output.

At the beginning of the semester the students' understanding of their design processes was largely described by them in terms of descriptions of specific design projects, describing the step by step, linear process through a description of the concrete moves they made. None of them was able to describe a more abstract process in any detail. This had changed for some students by the end of the semester, and for others by the time of the final interview six months later. All of them stated they had learned about their own design process, and several were able to describe it in abstract terms of tinkering with ideas through representation, linking it to their programming skills learned in the course.

The course was a transformative educational experience for the students. They began with an assumption about computer programming as a tool, and ended with the realization that through learning techniques in the course they had transformed their own understanding of their personal design processes. In most of the students, the metacognitive skills they developed were transferred to other areas of their education, including the design studio. This supports the reinterpretation of the Papert Principle proposed in this thesis, that the development of a personal design process is perhaps best taught away from the design studio and in a course with a Constructionist pedagogical approach, using software with similar Constructionist characteristics to LOGO.

The course also transformed the paradigm with which the students both viewed and were using digital media in the design studio. Their use of digital media in design at the start of the semester indicated an assumption that computers were predominantly for representation, although most of the students had joined the course because they wanted to challenge this. The paradigm for the computer was either an electronic pencil or a digital library. It changed the students preconceptions about computer programming, namely developing the understanding that it is a process that can be used to solve qualifiable design decisions, as well as the more standard quantifiable processes. The paradigm for the computer became one of a student-collaborator, able to contribute to the design dialogue and became an active voice of 'back talk'. The definition of the computer's role in the dialogue was not predetermined, and remained open to change. The process reaffirmed in these mature students, through explicit demonstration, the importance of listening to the back talk of design. In addition, the students were required by the process to clarify their concepts in verbal language in order to be able to communicate them to the computer. Designing necessarily involves communicating ideas to oneself, and this process formed part of the student's development of understanding of her own designing. These ideas, expressed verbally and then in the computer language produced a graphical version of the student's thought for her to assess. This reinforced the understanding of the incremental process of tinkering with ideas to develop a design project. As the students saw the results, particularly in the beginning stages when they were somewhat unexpected, and then later when this unexpected nature was explicitly specified, the importance of using surprising results to reassess the original idea became apparent. The inclusion of variation to give unexpected results additionally supported the understanding that design is in part a choice between products of explorations. In offering the variations upon the theme developed by the student, the computer contributes more to the dialogue than the pencil sketch. The process of architectural design is one in which the definition of the design problem progresses together with the proposed solution. Only when the final design is complete is it possible to precisely define the original design problem. The same was true for the students' projects in the course, reflected in the different descriptions they gave of their projects at the beginning and end of the research study.

The work that the students created provided a vehicle for their understanding of their own design process. This awareness permitted the students to raise the subject in the presentations and offer it as a topic for discussion. The concentration of the students on the code writing, which in a traditional architectural jury system is incomprehensible in visual terms, and the poor graphical quality of the resultant output help to remind both the students and the review panel that discussions concerning design process development are as important as those concerning the design product. It is rare in a traditional jury for either students or jury members to raise the subject of design process understanding. The concentration of juries in a studio setting is on product, not process. The juries of this course provided a forum for discussion of both.

A relationship could be seen in some of the students between the extent to which they focused on the computer as a tool, the computational constructs they were using, and the extent to which they were unable to develop their design ideas. As with all tools used in design, true appropriation is critical for success. Any concentration on the tool, whether watercolour, pencil, model making, or digital media, distracts the designer from focusing on the design. A period of learning is necessary with any new tool, but the sooner it can become transparent the better. There is a distinct danger that an excessive concentration on the tool can convince the student that she is actually designing. The same danger can be seen with students creating excessively detailed digital models, or with any process that inhibits the use of objects created as a external aids to thinking.

An unexpected surprises was the strength of change in learning model in the only student with distinct Behaviourist tendencies at the start of the semester. None of the students were novice, and further work is required to understand both the model of learning novice students arrive with, and the extent of change of this model through their exposure to architectural education. Students need to learn that they have responsibility for their own development of their own design process, through tinkering with their ideas and documenting the progressive results in studio. They need to learn to break their investment with the graphical products they produce, and view these as only being important to help develop the next idea. They need to learn the skill of using objects to think with in the process of designing. In short, since the process of designing is a process of constant learning and both share similar cognitive strategies, they need to move from a Behaviourist view of designing and learning to a Constructionist one. This success of this with this one student, who gained realization of these issues and attributed for her understanding to having taken the course, further indicates success in the research's reinterpretation of the Papert Principle. However, more research needs to be done on the effect of the course on beginning design students.

The unexpected result of minimal collaboration in the group studied contrasted with experience of teaching the course in previous years. It was clear that the students were not adverse to talking with each other, and that the highly competitive spirit of the design studios had largely been removed. The final presentation, with its emphasis on general discussion of the issues as much as individual student work emphasized the removal of the risks of failure normally associated with this highly competitive process. The contexts within which the students were operating were considered to be important, and this emphasis on discussion with many people contributing, and the extent to which the students enjoyed it, along with their enjoyment of the guest lecturers served to support the success of the beginnings of a knowledge building community surrounding the concepts of the role of digital media in design. However, the lack of collaboration in the Spring 1999 course outside of normal class hours, contrasts also with experience of other computer professionals in teaching in architectural computer studios. There is an assumption that this collaboration will naturally occur in these settings. However, from the results of this research it is possible to suggest firstly, if it is assumed that the lack of collaboration in the Spring 1999 semester is a result of a helpful TA, then normally observed collaboration in computer studios is a result of a common desire to learn skills in the absence of the expert help. Secondly, that collaboration in a computer studio is a result of a separation between skill learning and designing, and is therefore a negative indicator of computer appropriation. Therefore, the provision of such assistance enhances individual development of computer appropriation at the expense of collaboration, and as such represents a potential dilemma for the future of the course. Communication skills were enhanced through the student learning to communicate with the computer, but the hope is that a balance might be found in the future between the intrapersonal and interpersonal development.

References

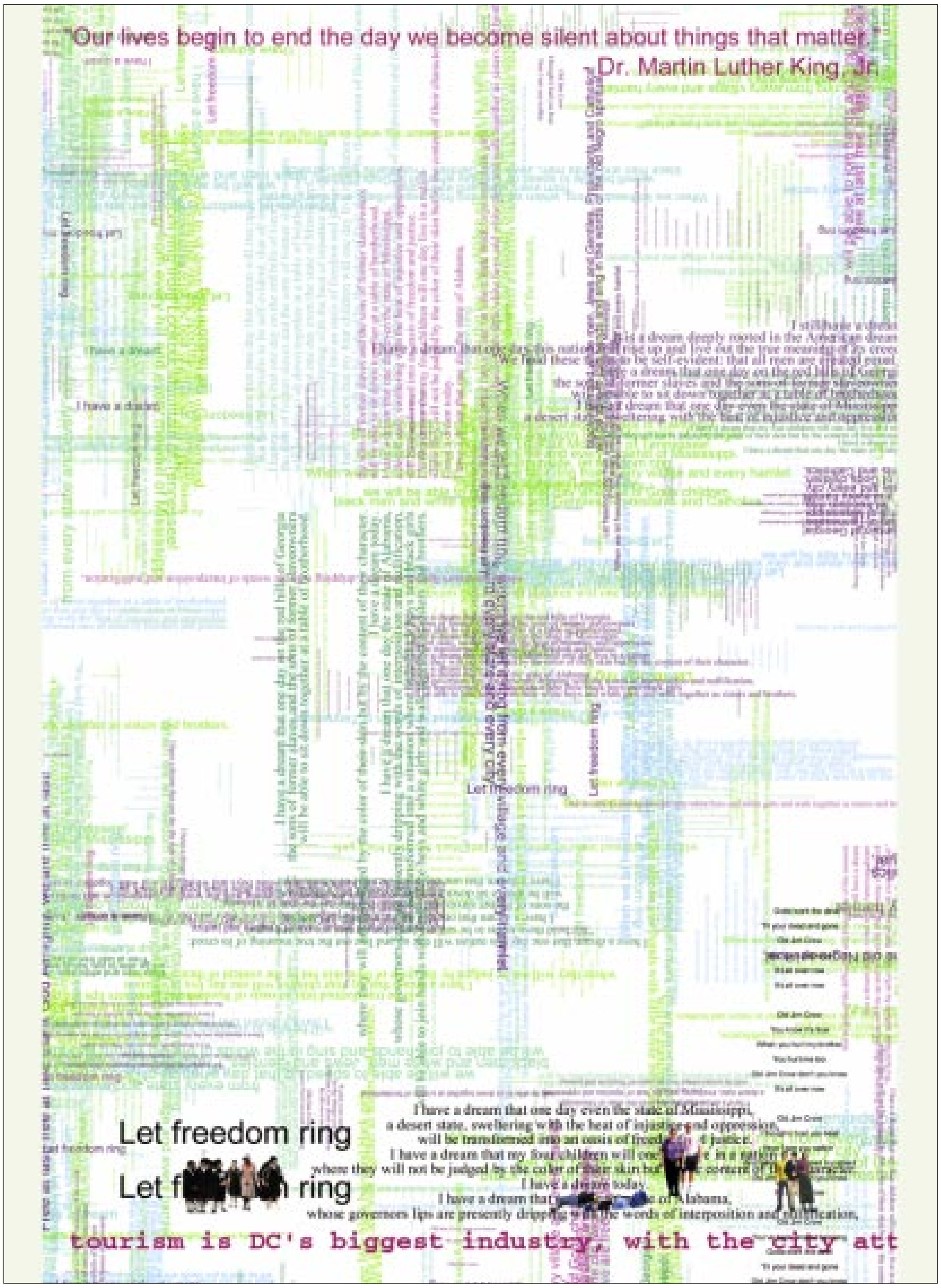

"Our lives begin to end the day we become silent about things that matter."
-Dr. Martin Luther King, Jr.
Let freedom ring
Let freedom ring
I have a dream that one day even the state of Mississippi, a desert state, sweltering with the heat of injustice and oppression, will be transformed into an oasis of freedom and justice.
I have a dream that my four children will one day live in a nation where they will not be judged by the color of their skin but by the content of their character.
I have a dream today.
I have a dream that one day the state of Alabama, whose governors lips are presently dripping with the words of interposition and nullification,
tourism is DC's biggest industry, with the city att

Figure 70. Talia Braude's final design for the competition for a memorial to Martin Luther King was a 'Sound Garden'. Sensors determined the number of people in an area and played his speeches to an appropriate volume

AA (1967) *Reporting Back Attingham Park: The teaching of Design – Design Method in Architecture* (London, Architectural Association).

Aburawi, M.M. (1987) Integration of Computers within the Architectural Design Process and Design Education Studios, *PhD., Department of Architecture* (Tampa, Florida, University of Florida).

Adelson, B. (1989) Cognitive Research: Uncovering How Designers Design; Cognitive Modeling: Explaining and Predicting How Designers Design, *Research in Engineering Design*, 1, pp. 35 - 42.

Agabani, F.A. (1980) Cognitive Aspects of Architectural Design Problem Solving, *PhD, Architecture* (Sheffield, University of Sheffield, UK).

Akin, Ö. (1978) How do architects design?, in: J.-C. Latombe (Ed) *Artificial Intelligence and Pattern Recognition in Computer Aided Design* (New York, North Holland).

Akin, Ö. (1984) An exploration of the design process, in: N. Cross (Ed) *Developments in Design Methodology* (Chichester, John Wiley).

Akin, Ö. (1993) Architects' reasoning with structures and functions, *Environment and Planning B*, 20, pp. 273 – 294.

Akin, Ö., Dave, B. & Pithavadian, S. (1992) Heuristic generation of layouts: based on a paradigm for problem structuring, *Environment and Planning B*, 19, pp. 33 – 59.

Alexander, C. (1966) *The atoms of environmental structure* (Berkeley, Center for Planning and Development Research, University of California).

Alexander, C., King, V.M., Ishakawa, S., Baker, M. & Hyslop, P. (1966) Relational complexes in architecture, *Architectural Record*, 140(Sept), pp. 185-190.

Atman, C.J., Chimka, J.R., Bursic, K.M. & Nachtmann, H.L. (1999) A comparison of freshman and senior engineering design processes, *Design Studies*, 20(2 March), pp. 131 – 152.

Auer, M. (1999) Getting in Touch with the Programmer Inside, *PinUp (MIT School of Architecture Student Magazine)*, 4(Fall), pp. 7.

Bamberger, J. (1991) *The Mind behind the Musical Ear* (Cambridge MA, Harvard University Press).

Bamberger, J. (1996) Turning Music Theory on Its Ear, *International Journal of Computers for Mathematical Learning*, 1(1), pp. 33-55.

Beheshti, R. (1993) Design decisions and uncertainty, *Design Studies*, 14(1 Jan), pp. 85-95.

Bereiter, C. & Scardamalia, M. (1993) *Surpassing Ourselves: An Inquiry into the Nature and Implications of Expertise* (Chicago, Open Court).

Berger, P.& Luckmann, T. (1966) *The Social Construction of Reality* (New York, Irvington).

Berkeley, E.P. (1968) Computers for design and design for the computer, *Architectural Forum*, 128(Mar), pp. 60-65.

Branki, N. (1993) A study of socially shared cognition in design, *Environment and planning B*, 20(3), pp. 295-306.

Bruner, J. (1966) *Toward a Theory of Instruction* (Cambridge, MA, Harvard University Press).

Bustinza Esparta, J. (1996) A Methodological Proposal for Hypermedia Systems Integration in Architectural Education (Hipercon), *DR., Department of Architecture* (Pamplona, Spain, Universidad de Navarra (Spain)).

Carver, W. (1964) Creativity in architectural design: The ACSA reports, *Journal of Architectural Education*, 19, pp. 19-22.

Chan, C., Burtis, P., Scardamalia, M. & Bereiter, C. (1992) Constructive Activity in Learning From Text, *American educational research journal*, 29(1), pp. 97-118.

Chandrasekaran, B. (1989) A Framework for Design Problem-solving, *Research in Engineering Design*, 1, pp. 75 – 186.

Chen, H.-T. & Rossi, P.H. (1983) Evaluating with Sense: the Theory Driven Approach, *Evaluation Review*, 7(3, June), pp. 283 - 302.

Coyne, R. & Snodgrass, A. (1991) Is Designing Mysterious? Challenging the dual knowledge thesis, *Design Studies*, 12(3), pp. 124 – 131.

Coyne, R. & Snodgrass, A. (1993) Rescuing CAD from rationalism, *Design Studies*, 14(2 April), pp. 100-123.

Crowe, N.A. & Hurtt, S.W. (1986) Visual notes and the acquisition of architectural knowledge, *Journal of Architectural Education*, 39(3), pp. 6-16.

Cutkosky, M.R. & Tenenbaum, J.M. (1990) Research in Computational Design at Stanford, *Research in Engineering Design*, 2, pp. 53 - 59.

Darke, J. (1979) The primary generator and the design process, *Design Studies*, 1(1), pp. 36 – 44.

Davies, R. (1987) Experiencing ideas: identity, insight and the imago, *Design Studies*, 8(1), pp. 17-25.

Denzin, N.K. (1989) *Interpretive Interactionism* (Newbury Park, CA, Sage).

Dewey, J. (1929) *The Quest for Certainty: a study of the Relation of Knowledge and Action* (Chicago, Southern Illinois University Press).

Dewey, J., Barnes, A.C., Buermeyer, L., Mullen, M. & de Mazia, V. (Eds.) (1969) *Art and Education* (Merion, Pa., Barnes Foundation Press).

Do, E.Y.-L., Gross, M. & Zimring, C. (1999) Drawing and Design Intentions – An investigation of Freehand Drawing Conventions in Design in: G. Goldschmidt & W.L. Porter (Eds) *The Fourth Design Thinking Research Symposium* (Massachusetts Institute of Technology, School of Architecture, MIT).

Dorsey, J. & McMillan, L. (1998) Computer Graphics and Architecture: State of the Art and Outlook for the Future, *Computer Graphics*, 32(1 (February)), pp. 45 – 48.

Dwyer, T. (1971) On the importance of complexity in supportive systems for educational computing, *Interface*, 5(3), pp. 99 – 105.

Eastman, C.M. (1969) Cognitive processes and ill-defined problems: a case study from design in: D.E. Walker & L.M. Norton (Eds) *Joint International conference on Artificial Intelligence* (Washington, D.C.)

Eckersley, M. (1988) The form of design process: a protocol analysis study, *Design studies*, 9(2), pp. 86 – 94.

Ericsson, K.A. & Simon, H.A. (1980) *Protocol Analysis: verbal reposts as data* (Cambs. MA, MIT Press).

Fish, J. (1999) From Description to Depiction and Back: (Uncertainty and the Mental Translation of Representational type as a Function of Designer's Sketches) in: G. Goldschmidt & W.L. Porter (Eds) *The Fourth Design Thinking Research Symposium* (Massachusetts Institute of Technology, School of Architecture, MIT).

Forrest-Pressley, D.-L. & Waller, T.G. (1984) *Cognition, Metacognition, and Reading* (New York, Springer-Verlag).

Frederickson, M. (1991) Design juries: a study in lines of communication (interpersonal communications), *PhD Thesis, School of Architecture and Urban Planning* (Los Angeles, CA, University of Los Angeles).

Galle, P.& Kovacs, L. (1992a) Introspective observations of sketch design, *Design Studies*, 13(3 July), pp. 229-272.

Galle, P.& Kovacs, L. (1992b) The logic of worms: a study in architectural knowledge representation, *Environment and Planning B*, 19, pp. 5 – 31.

Gardner, H. (1991) *Unschooled Mind: How children think and how schools should teach* (New York, Basic Books).

Gero, J., Coyne, R. & Maher, M.L. (1991) Laboratory Report: Design Computing Unit: University of Sydney, *Research in Engineering Design*, 2, pp. 109 - 115.

Gero, J. & Purcell, T. (1993) Guest Editorial: Cognitive modelling in design, *Environment and Planning B*, 20, pp. 253 – 256.

Glaser, B.G. & Strauss, A.L. (1967) *The Discovery of Grounded Theory* (Chicago, IL, Aldine).

Grossman, M.L. (1990) Theory and Practice in Introductory Level Landscape Architectural Design Education, *M.L.A., Department of Landscape Architecture*, University of Guelph (Canada)).

Guarra, F. (1974) The use of Random in Architectural Design, *Bulletin of Computer Aided Architectural Design*, 14(January).

Harel, I. & Papert, S. (1990) Software design as a learning environment, in: E. Soloway (Ed) *Interactive Learning Environments*, Vol. 1 (NJ, Ablex).

Harvey, J. (1950) *The Gothic World* (London, Batsford).

Hayes, J.R. (1981) *The complete problem solver* (Philadelphia, Franklin Institute Press).

Henke, S.L. (1990) Systematic Generation of Architectural Compositions, *Environment and Planning B: Planning and Design*, 17, pp. 149 – 165.

Heylighen, A., Neuckermans, H. & Bouwen, J.E. (1999) Walking on a thin line – between passive knowledge and active knowing of components and concepts in architectural design, *Design Studies*, 20(2 March), pp. 211 - 235.

Keller, E.F. (1983) *A Feeling for the Organism: The Life and Work of Barbara McClintock* (San Francisco, W. H. Freeman).

Knight, T. (1990) Mughul Gardens Revisited, *Environment and Planning B*, 17, pp. 73 – 84.

Knowles, E.E. (1993) A Study to Determine the Rationale for Using Jury Critiques in Architectural Education (Studio Instruction), *PhD, Architecture* (Oklahoma, University of Oklahoma).

Koschmann, T. (Ed.) (1996) *CSCL: Theory and Practice of an Emerging Paradigm* (Mawah, NJ, Lawrence Erlbaum Associates).

Krishnamurti, R. & Giraud, C. (1986) Towards a shape editor: the implementation of a shape generation system, *Environment and Planning B: Planning and Design*, pp. 391 – 404.

Lawson, B. (1980) *How Designers Think* (London, The Architectural Press).

Lawson, B. (1993) Parallel lines of thought, *Languages of Design*, 1, pp. 321 – 331.

Ledewitz, S. (1982) Models of design in studio teaching, *Journal of Architectural Education*, ?(?), pp. ?

Lévi-Strauss, C. (1968) *The Savage Mind* (Chicago, University of Chicago Press).

Lincoln, Y.S. & Guba, E.G. (1985) *Naturalistic Inquiry* (Beverly Hills, CA, Sage).

Lloyd, P.& Scott, P. (1966) Discovering the design problem, *Design Studies*, 15(2 April), pp. 125-140.

Lloyd, P.& Scott, P. (1966) Difference in similarity: interpreting the architectural design process, *Environment and Planning B*, 22, pp. 383 – 406.

Maykut, P.& Morehouse, R. (1994) *Beginning Qualitative Research* (London, The Falmer Press).

McFadzean, J. (2000) Analysing the Creative Process in Architectural Conceptual Design in: C. Teeling (Ed) *Greenwich2000 International Symposium on Digital Creativity: Architecture, Landscape, Design* (The University of Greenwich, London, The University of Greenwich).

Mitchell, W.J. (1975) The Theoretical Foundation of Computer-Aided Architectural Design, *Environment and Planning B*, 2, pp. 127 – 150.

Mitchell, W.J. (1994) Three paradigms for computer-aided design, *Automation in Construction*, 3, pp. 239 – 245.

Mitchell, W.J., Liggett, R. & Tan, M. (1987) *The Art of Computer Graphics Programming* (New York, Van Nostrand Reinhold).

Mitchell, W.J., Liggett, R. & Tan, M. (1992) Multi-level Analysis and optimization of Designs, in: E.K. Yehuda (Ed)

Nadimi, H. (1996) Conceptualizing a Framework for Integrity in Architectural Education with Some References to Iran, *DPhil.,* (York, UK, University of York).

Negroponte, N. & Groisser, L. (1970) URBAN5: A machine that discusses urban design, in: G.T. Moore (Ed) *Emerging Methods in Environmental Design and Planning* (Cambs, MA, MIT Press).

Newell, A. (1970) Heuristic Programming: ill structured problems, in: J.A. Arnofsky (Ed) *Progress in Operations Research*, Vol. 3 (New York, John Wiley).

Newell, A. & Simon, H.A. (1972) *Human Problem Solving* (Englewood Cliffs, NJ, Prentice Hall).

Nickerson, R.S., Perkins, D.N. & Smith, E.E. (1985) *The Teaching of Thinking* (Hillsdale, N.J., L. Erlbaum Associates).

Noll, A.M. (1967) The Digital Computer as a Creative Medium, *IEEE Spectrum*, 4(October), pp. 89 – 95.

Norman, D.A. (1993) *The Power of Representation: Things that make us smart: defending human attributes in the age of the machine* (Reading, MA, Addison-Wesley).

OED (1995) *The Concise Oxford Dictionary of Current English* (Oxford, Clarendon Press).

Oxman, R. (1999) Educating the designerly thinker, *Design Studies*, 20(2), pp. 105 – 122.

Papert, S. (1980) *Mindstorms: children, computers, and powerful ideas* (New York, Basic Books).

Papert, S. (1990) Introduction, in: I. Harel (Ed) *Constructionist learning : a 5th anniversary collection of papers reflecting research reports, projects in progress, and essays* (Cambridge, MA., The Media Laboratory, Massachusetts Institute of Technology).

Papert, S. (1991) Situating Constructionism, in: I. Harel & S. Papert (Eds) *Constructionism* (Norwood, NJ, Ablex Publishing).

Papert, S. (1993) *The Children's Machine: rethinking school in the age of the computer* (New York, Basic books).

Perkins, D. (1992) *Smart Schools: From training memories to educating minds* (New York, The Free Press).

Perkins, D. & al., e. (1986) Conditions of Learning in Novice Programmers, *Journal of Educational Computing Research*, 2(1), pp. 37 – 56.

Perkins, D. & al., e. (Eds.) (in press) *Teaching for Understanding in the Age of Technology*

Perkins, D. & Salomon, G. (1987) Rocky Roads to Transfer: Rethinking Mechanisms of a Neglected Phenomenon, in: J. Lockhead & D. Perkins (Eds) *Thinking* (Hillsdale, NJ, Erlbaum).

Perkins, D. & Simmons, R. (1988) Patterns of Misunderstanding: An integrative model for Science, Math and Programming, *Review of Educational Research*, 58, pp. 303 – 326.

Piaget (1977) *The Psychology of the Child*

Piaget, J. (1926) *The Language and Thought of Children* (New York, Harcourt, Brace, World).

Piaget, J. (1963) *The psychology of intelligence* (Paterson, N. J, Littlefield, Adams).

Piaget, J. (1964) *The child's conception of the world* (London, Routledge & Kegan Paul).

Pohliman, R.W. (1982) A system for recording behaviour and Occupying Design, in: Ö. Akin & E.F. Weinel (Eds) *Representation in Architecture* (Silver Springs, MD., Information Dynamics).

Portillo, M. & Dohr, J. (1994) Bridging process and structure through criteria, *Design Studies*, 15(4 Oct), pp. 403-415.

Prestamo, F.J. (1990) Architectural Education in Postindustrial America: an Application of the Tyler Model to the Development of a Curriculum (Curriculum Development), *Ed.D,* (Tampa, Florida, University of Florida).

Purcell, P.A. (1988) Computer Environments for Design and Designers, *Design Studies*, 9(3), pp. 144 – 149.

Resnick, L. (Ed.) (1989) *Knowing, learning, and instruction* (Hillsdale NJ, Lawrence Erlbaum Associates).

Resnick, M. (1995) *Turtles, Termites, and Traffic Jams* (Cambridge, Mass., MIT Press).

Resnick, M., Bruckman, A. & Martin, F. (1996) Pianos Not Stereos, *Interactions*, 111(5), pp. 40 – 50.

Resnick, M. & Ocko, S. (1990) Lego/LOGO: Learning through and about design, in: I. Harel (Ed) *Constructionist learning : a 5th anniversary collection of papers reflecting research reports, projects in progress, and essays* (Cambridge, MA., The Media Laboratory, Massachusetts Institute of Technology).

Rieber, L., P. (1965) A Historical review of visualisation in human cognition, *Educational Technology Research and Development*, 43(1), pp. 45 – 56.

Rowe, P.G. (1987) *Design Thinking* (Cambridge, MA, MIT Press).

Roy, R. (1993) Case studies of creativity in innovative product development, *Design Studies*, 14(Oct), pp. 423-443.

Sachs, A. (1999) 'Stuckness' in the design studio, *Design Studies*, 20(2 March), pp. 195 – 209.

Salomon, G. (1988) AI in reverse: Computer tools that turn cognitive, *Journal of Educational Computer Research*, 4, pp. 123–139.

Scardamalia, M. & Bereiter, C. (1994) Computer Support for Knowledge-Building Communities, *The Journal of the Learning Sciences*, 3(3), pp. 265 – 283.

Schön, D.A. (1983) *The Reflective Practitioner: How professionals think in action* (New York, Basic Books Inc.).

Schön, D.A. (1984a) The Architectural Studio as an Exemplar of Education for Reflection-in-Action, *Journal of Architectural Education*, 38(1), pp. 2 – 9.

Schön, D.A. (1984b) Problems, Frames and Perspectives on Designing, *Design Studies*, 5(3), pp. 132 – 136.

Schön, D.A. (1985) *The Design Studio; an exploration of its traditions and potentials* (London, RIBA Publications for RIBA Building Trust).

Schön, D.A. (1987) *Educating the Reflective Practitioner: toward a new design for teaching and learning in the professions* (San Francisco, Jossey-Bass).

Schön, D.A. (1988) Designing: rules, types and worlds, *Design Studies*, 9(3), pp. 181-190.

Schön, D.A. (1992) Designing as Reflective Conversation with the Materials of a Design Situation, *Research in Engineering Design*, 3, pp. 131 – 147.

Solomon, C. (1986) *Computer environments for children: a reflection on theories of learning and education* (Cambridge, Mass., MIT Press).

Soloway, E., Guzdial, M. & Hay, K. (1994) Learner-Centered Design: The challenge for HCI in the 21st century, *Interactions*, (April), pp. 36 – 48.

Stiny, G. (1979) A generative approach to composition and style in architecture, PARC *79*

Stiny, G. (1990) What Designers Do that Computers Should, in: M. McCullough, W. Mitchell & P. Purcell (Eds) *The Electronic Design Studio* (Cambridge, MA, MIT Press).

Stiny, G. & Mitchell, W. (1980) The grammar of paradise: on the generation of Mughul gardens, *Environment and planning B*, 7, pp. 209–226.

Strauss, A. & Corbin, J. (1990) *Basics of Qualitative Research: Grounded Theory Procedures and Techniques* (Newbury Park, CA, Sage).

Suchman, L. (1987) *Plans and Situated Actions: The problem of Human machine communication* (New York, Cambridge University Press).

Sun, D. (1993) Memory, design, and the role of computers, *Environment and Planning B*, 20, pp. 125 – 143.

Sutherland, I.E. (1975) Structure in Drawings and the Hidden-Surface Problem, in: N. Negroponte (Ed) *Reflections on Computer Aids to Design and Architecture* (New York, Petrocelli/ Charter).

Tang, J.C. & Leifer, L.J. (1991) An Observational Methodology for Studying Group Design Activity, *Research in Engineering Design*, 2, pp. 209 – 219.

Teicholz, E. (1968) Architecture and the computer, *Architectural Forum*, (Sept), pp. 58-61.

Turkle, S. & Papert, S. (1990) Epistemological Pluralism: Styles and voices within the computer culture, in: I. Harel (Ed) *Constructionist learning : a 5th anniversary collection of papers reflecting research reports, projects in progress, and essays* (Cambridge, Mass., The Media Laboratory, Massachusetts Institute of Technology).

Ulusoy, Z. (1999) To design versus to understand design: the role of graphic representations and verbal expressions, *Design Studies*, 20(2), pp. 123 – 130.

Vygotsky, L. (1978) *Mind in Society* (Cambridge, MA, Harvard University Press).

Weiss, C. (1997) How can Theory-Based Evaluation make greater headway?, *Evaluation Review*, 21(4, August), pp. 501 – 524.

Weiss, C. (1998) *Evaluation: Methods for Studying Programs and Policies* (New Jersey, Prentice Hall).

White, B.Y. (1981) Designing computer games to facilitate learning, *PhD, Artificial Intelligence Laboratory* (Cambridge, Mass., Massachusetts Institute of Technology).

White, B.Y. (1984) Designing computer games to help physics students understand Newton's laws of motion, *Cognition and Instruction*, 1, pp. 69-108.

Whitehead, B. & Eldars, M.Z. (1964) An approach to the optimum layout of single-storey buildings, *Architect's Journal*, 139(25), pp. 1373 – 1380.

Wingler, H.M. (1978) *The Bauhaus: Weimar, Dessau, Berlin, Chicago* (Cambridge, MA, MIT Press).

Winograd, T. & Flores, F. (1986) *Understanding Computers and Cognition: A new foundation for design* (Norwood, NJ, Ablex).

Yakeley, M.W. (1999) Simultaneous Translation in Design: the Role of Computer Programming in Architectural Education in: A. Brown, M. Knight & P.Berridge (Eds) *eCAADe* (Liverpool, UK, The University of Liverpool).

Yakeley, M.W. (2000) Bons a Penser ou Bons a Regarder? Using a Computer to Aid Creativity in Design in: C. Teeling (Ed) *Greenwich2000 International Symposium on Digital Creativity: Architecture, Landscape, Design* (London, UK, The University of Greenwich).

End Notes

1. Several of the American educated students in the course's history were taught LOGO in grade school, including one in the research study. All were highly successful in their approach to the course and the extent to which they appropriated the computer for designing

2. Bons-a-Penser is a clever play on words in French that does not easily translate into English: it can mean both "goods (objects) to think with" and "good (useful) to think with"

3. The extent to which the students in the study believed in this model of education was difficult to gauge and varied. None of the students were novice and it was therefore impossible to tell what model of learning they had arrived with. More research needs to be done with beginning design students

4. All software developed at MIT to support the Constructionist research of Seymour Papert and colleagues

5. Personal communication from John Fernandez, Visiting Assistant Professor, MIT Department of Architecture, to the author, November 1999

Appendix A

Student profile in study

Undergraduates

The school of architecture at MIT currently has four levels of degree programs, within which there are a number of different specialities or research interests. The undergraduate program is a four year Bachelor of Science in Art and Design, with the possibility of a concentration in one of four areas: design, visual arts, building technology, or history, theory and criticism. In the Spring 1999 semester there were no undergraduate students in the course, although there has been in previous years.

Students in MArch

Students who take MIT's BSAD are eligible for entry into the second degree program, Master of Architecture (MArch) at Level II. Those who take a similar general design degree elsewhere enter at Level I, as are students with Bachelor degrees in other fields. The MArch program is therefore either three and a half years or two and a half years in length, depending on level of entry. However, some schools offer a four year Bachelor of Architecture program that permits them to enter the MArch program at Level II. Of the six students in the course, four were in the MArch program, three of whom were in Level II. Only one of these three entered in Level I; the other two completed BArch degrees elsewhere and therefore were eligible to enter in Level II.

Exchange student

The fourth MArch student was in Level III, and was a special student at MIT for a year as part of the exchange program with the Department of Architecture at the University of Cambridge. In the UK the system is very different, and this student has studied architecture for a shorter but much more intensive period of time than the other students.

MArch Concentration

MArch students are required to declare a concentration during the second semester of Level II that is intended to provide a self oriented framework around which the student can structure her own education and begin to think about a thesis project for the final semester. At the time of the first interview, none of the students in the course in Level II had declared their concentration, but all joined the course with the intent of considering it to be a part of this process.

Students in SMArchS

After the MArch preprofessional degree, students are able to enter the world of professional practice, but many choose to continue to study and take the post professional masters program, the Master of Science in Architecture Studies (SMArchS). This is a four semester program of study in one of four research areas: design technology, architecture and urbanism, architecture and technology, and history, theory and criticism. Two students in the course are in the SMArchS program. Interestingly, these two students are also the youngest and oldest students in the course, aged 23 and 30 respectively. One came to MIT direct from his five year architecture program at Roger Williams University, and is consequently one of the youngest in the SMArchS program. The other has worked for many years in Japan as an architect, and has chosen to return to education much later.

Cultural background of students

MIT is a very multicultural university, and this diversity is reflected in the course. Three of the students were born, raised, and educated entirely in the United States, one of whom is the first generation of Asian immigrants. A fourth arrived from Japan at the age of five but had extensive Japanese schooling throughout his childhood in addition to his US public education. Two students are from abroad, having been raised and educated entirely in their home countries, England and China respectively. Experience of studying and practising architecture range from five to eleven years.

Appendix B

Class Notes

Appendix C

First Interview Schedule

Section 1: Understanding of the process of design

Get the student talking about their current studio project

1. Description of current studio project:

    What is it

    where did/how has student started on it

    what avenues are being explored

    what has been main influence(s) so far.

2. Intentions of student in developing studio project: Get student to talk about how they work in general at the start of a project, and then what happens later in the semester

    What avenues will be pursued next?

    What are the main influences in deciding how to proceed?

3. Summation: can the student summarise what has just been described

    Is there a general way in which project is approached?

    Or does it differ every time, and if so, what does it depend on?

Section 2: Methods of Teaching

4. Prescriptive design:

    What are experiences of studios with a strong or weak methodology?

    If yes: What were they like?

    Opinion of processes used in these studios?

    What else is known of prescriptive design?

5. Role of studio professor:

What is view of role of studio professor?

When good, how was it good? Or bad?

What is opinion of teaching methodologies used in studios?

What methods in particular were used that helped explain how to design?

6. Learning to design:

Describe the process of learning architecture as though to a prospective student

What would they learn in studio? How would they go about learning it?

Should they be encouraged to apply? Why?

Section 3: Understanding of Digital Media

7. Design or analysis?

Does student have a more artistic temperament or a scientific one?

Why does he/she think so?

8. Previous experience:

What was very first experience with computers? What was it like?

What other experience, and in what areas?

What are courses taken at MIT in which digital media play a strong role, e.g. 4.203?

What were they like?

What aspects of digital media does he/she want to know more about or experience?

Intentions of taking any more classes here that use computers in some way?

9. Digital media in own design:

How did the courses taken help with design work?

What role does the computer play in own designing?

What is it used for, primarily?

What software is used, what is used most often?

Examples of work?

10. Digital media in the profession:

What are the main uses of computers in architecture?

What is the future of computers in the building industry?

Section 4: Group Working

11. Collaboration:

Previous experience of group working in classes?

Group working in studio?

If yes: What was it like?

If no: Desire to collaborate with others in studio?

What benefits or difficulties seen in doing so?

Opinion of why people choose to work in a group?

Section 5: The Class

12. Expectations:

Why class was chosen?

What is hoped to be achieved?

What are views of aims of the class in general?

What are initial ideas for projects?

Section 6: Demographics

13. Background information:

What degree program?

Which studio?

Age?

How many years have studied architecture? Practised?

Where was undergraduate degree? In what?

Country of main education process (K-12)?

Appendix D

Second Interview Schedule

March 29th to April 4th 1999

Section 1: The Design Process

1. Current Studio Project

    1a. Description of project (NB Remember specifics of description for previous interview to prompt)

    1b. How has project developed since the last interview?

    1c. What have been the main influences in defining the direction?

    1d. What is being currently worked upon?

2. Means of Representation

    2a. What models, sketches or other representations have been produced in the last couple of weeks?

    2b. What is being thought about with these models? Describe ideas

    2c. Is this a typical manner in which problems and ideas are being explored? If yes, examples of other projects? If no, what would be a more typical method of working and how is it different?

    2d. What is the favourite medium for working? Why?

3. Role of Professor

    3a. Describe the most recent tutorial/desk crit

    3b. What was said by student? What was said by professor?

    3c. What ideas emerged? In what way were these ideas communicated?

    3d. What suggestions were made by the professor?

    3e. Was this a typical or atypical meeting with this professor?

Section 2: Work done in Class

4. Class project current: (NB remember this is interview not tutorial so no comments)

    4a. Describe project in its current state

    4b. What is the next step to be taken?

5. Class project history

    5a. Where did the project begin?

    5b. Describe the journey taken to reach this point

    5c. What has been learned from doing this project? Computationally, subject matter, etc.

    5d. If you were asked by another student in the school what the point of taking the class was, what would you say? Would you recommend that they take the class?

6. Opinion of tools used

    6a. Opinion of VectorWorks? Comparison of VectorWorks with other CAD software already known?

    6b. What are the positive and negative aspects of VectorScript as a programming language?

    6c. Now that you have a better understanding of the language and its capabilities, what role do you see such a language playing in architecture? What might it be used for?

7. Class, future

    7a. Intentions of joining 4.184 next year?

    7b. Intentions of using these techniques in other classes?

Appendix E

Final Review Interview Schedule

May 10th 1999

E-mail sent to reviewers

May 7th 1999

Friends

Many thanks in advance for agreeing to come to some or all of Monday's review. Just to let you know we will be offering a late lunch at 1.30, and then afternoon tea at 3.45. (NB Neither of these will be MIT food -- only outside catering here!) You are most welcome to come to both meals even if you are only able to attend part of the afternoon.

Most of you know that my research is looking at what happens in the class. I would like to ask you to help me in this regard on Monday by asking some particular questions of each student in addition to the normal crit process. Below are the areas in which I am interested for Monday: it may not be appropriate/possible for these to be asked, but if it is at all possible I would be very grateful for the assistance. In other cases the student may volunteer the information anyway. The wording is not important, only the subject area.

1. Context:

> What is the broader context within which the project sits? (If there is one).
>
> Some students may be good at explaining this, but others may well be somewhat reticent about why they did what they did in the broadest sense!

2. Experience:

> What does the student feel she/he has learned through doing her/his project?
>
> I have asked them this in the second official interview, and the answers are quite wide ranging. I'm hoping they will have had time to contemplate this a little more.

3. Reflection:

> What are the student's views of the role of computers/computer programming in the design process? In what ways has the class influenced this?

4. Future:

Will the project be taken further? Will the techniques learned be used again?

Let me know if you have any questions, otherwise I will look forward to seeing you all for lunch

Thanks again, Megan

Appendix F

Final Interview Schedule

November 4th to 10th 1999

Introduction: One of the problems of being both the professor and the researcher is that you know that I know a great deal about your work, and therefore there is a tendency to not bother to spell things out in the interviews, because it seems as though you are repeating yourself unnecessarily. BUT I need as much information and detail in your answers as possible, even when it seems as though you are telling me things I should already know.

Section 1: The Design Process

1. Current Studio Project

    Description of project

    How did the project start?

    What have been the main influences in defining the direction?

    What is being currently worked upon?

    What is the role of your professor in your current studio?

2. Design project from last semester – studio project/competition.

    Description of project.

    How do you think it turned out?

    How was it received by critics? How successful was the final crit?

    What about the approach of using only plans and sections with no perspectives?

    Work (preferably in digital form) that I can have copies of?

3. Representation in design working

In what way do you use sketches when you are designing?

What other types of representation do you use in designing, and what do you use them for?

What media do you use, and how regularly?

Have you learned any new software packages since last semester?

What software are you currently using? What are you doing with it?

What role do you see programming playing as a means of representation of ideas?

What do you see as the link between sketching and programming?

How do you view the link between programming and designing?

Section 2: Work done in Class

3. The project

Describe, with as much detail as possible, the final project completed for 4.183 as though you are explaining it to someone who didn't get the chance to see it

Now that you've had time to reflect, how would you summarise the process involved in the development of your project from the start of the semester?

What do you see as being the most important thing you learned by taking the class?

What else was learned from the class?

How do you view the final crit process? Was it successful, and if so, why?

If you were to continue with the project now, what would the next step be?

4. Effect of class on other areas of work

In what ways have the processes learned in 4.183 influenced the way in which you approach a project? What about later on in the project?

In what ways have the processes affected other work, other projects?

How does the subject/content of the 4.183 class influence your designing now? (As opposed to the techniques)

Are you using any of the programming techniques today, and if so, what for?

(If yes:) Now that you are more familiar with the coding process, how easy is it to express your ideas in code? In what ways do you benefit from the process?

(If no:) If you were to use programming in your design work now, what aspect of your current design studio would you choose? And why? How useful do you think it would be?

5. Class, future

In what ways has your approach to your design work changed in the course of the last year? Why do you think it has changed in that way?

Do you intend to continue with the programming in the future? If so, what for?

In the previous interview, you were asked about what classes you intended to take with a strong technical emphasis, or what things you intended to learn that had a computational basis. What of these plans have you carried out? What things do you still intend to learn? Anything new you've recently decided you want to know more about?

Appendix G

Anonymous End of Semester Class Review Form

The quote on page 159 is taken from the following form. All the students fill in the forms, and the results are collated for all the classes, studios and workshops offered by the department for each semester