21
Computational Design Instruction: Toward a Pedagogy

Ömer Akin

Department of Architecture
Carnegie-Mellon University

The computer offers enormous potential both in and out of the classroom that is realized only in limited ways through the applications available to us today. In the early days of the computer it was generally argued that it would replace the architect, When this idea became obsolete, the prevailing opinion of proponents and opponents alike shifted to the notion of the computer as merely adding to present design capabilities. This idea is so ingrained in our thinking that we still speak of aiding" design with computers. It is clear to those who grasp the real potential of this still new technology-as in the case of many other major technological innovations-that it continues to change the way we design, rather than to merely augment or replace human designers.

In the classroom the computer has the potential to radically change three fundamental ingredients: student, instruction, and instructor. It is obvious that changes of this kind spell out a commensurate change in design pedagogy. If *the computer is going to he more than a passive instrument in the design studio, then design pedagogy will have to he changed, fundamentally. While time practice of computing in the studio continues to he a significant aspect of architectural education, articulation of viable pedagogy for use in the design studio is truly rare*

In this paper the question of pedagogy in the CAD studio will be considered first. Then one particular design studio taught during Fall 1988 at Carnegie-Mellon University will he presented. Finally, we shall return to issues of change in the student, instruction, and instructor, as highlighted by this particular experience.

Paradigms for Computational Design Pedagogy

Representation-based Paradigms

Perhaps the clearest and earliest statement of pedagogic intent in this area can be found in William Mitchell's write-up in *Pioneers of CAD* (Mitchell, 1985). He advocates "awakening students to the insights that they can get by formulating design issues in computational terms" in lieu of learning the 'details of associated computer technologies." His pedagogy goes beyond the generality of this statement and argues for a specific approach to computational design. In this approach, students start by understanding the basic notions of computing, algorithms, and data structures. Then they learn, through the use of these, to generate the "primitives" of a language of graphics-points, lines, two-dimensional compositions-which correspond in some form to architectonic elements we use as designers. Next, students are introduced to elements of composition, proportion, repetition, rhythm, and geometric transformations. Each of these concepts of design correspond to equally basic functionalities in computation, such as mathematical expressions, loops, conditionals, and recursion. Mitchell goes on to show how other more complex issues-three-dimensional design, colors, textures-can be introduced using the same principle.

The basis of this pedagogy is the existing isomorphisms between representations we use in architecture and constructs used in computing. Mitchell's pedagogy capitalizes on this isomorphism, reinforcing, even justifying, the legitimacy of these constructs and representations by showing their mutual agreement.

Design Product-based Paradigms

Another category of approaches to pedagogy in the CAD studio is offered by Flemming and Schmitt (1986). Schmitt bases his pedagogy on the principle of complementing students evaluative abilities by providing a "work bench" of analysis tools, such as cost, heating load, cooling load, electrical load, structural feasibility, circulation efficiency, visual comfort, and contextual responsiveness. This allows students to objectively evaluate their designs once they are generated, by necessity on the computer, so that these tools can be applied to them. It is the computational power of the medium in analyzing properties of the designed object that gives rise to the pedagogic content of the studio.

Flemming's approach to CAD is generative rather than evaluative. He uses the parametric manipulation of basic architectonic elements-walls, openings, etc.-to mimic different architectural "languages-elementary wall architecture, mass architecture, layered transparent architecture, and

structure / infill architecture (Flemming, 1989). Starting from a theoretical understanding of the syntax of these languages, Flemming builds a link to computational capabilities in the computer, which in turn enables the student to explore systematically, and with relative ease and precision, the design instances in a given language category. Here, as in Schmitt's approach, the computational advantages of the medium are used to understand design in terms of the designed object.

Design Process-based Paradigms

The studio that I offer presents yet another paradigm for computational design instruction. Mitchell's approach outlines the way in which instruction stems from basic compositions or designs that in their making correspond to computational functions. Flemming's and Schmitt's approaches show how qualities of the object being produced form the basis of computational strategies. Here the aim is to consider the qualities of the design process and how the computer can be used to improve it.

An important premise of this approach is that several attributes of the traditional design studio which arise from manual forms of representation cannot be supported by the computer. More specifically, the computational medium, unlike its manual counterpart, imposes no hierarchy of representations based on scale; is not bounded by representational conventions to any particular hierarchy of issues; requires considerable front-end input time before a critical mass of information, sufficient for design, is accumulated; and does not particularly support, in the levels of precision in representing objects, a top-down, hierarchical design strategy. (But see Mitchell, Liggett, and Tan 1989).

In the traditional design studio the process adopted is a top-down hierarchical one, defined by conventions of practice and by drawing (Ledewitz, 1985). The design development process and most design pedagogies advocate that design start with a concept of the whole and then progress toward greater detail through the formalized steps of preliminary design, design development, and working drawings. The central difficulty with this approach as a pedagogic tool is that while experienced designers are cognizant of the requirements of each phase of the design delivery process and thus can anticipate the "precautions" that need to be taken during early phases on behalf of issues that will come up during later phases (proactive design), students are usually unaware of such dependencies between design and do not know how to anticipate them (reactive design) (Akin, 1986). Consequently, students are given unnecessary opportunity to fail and to use time ineffectively.

A related but different difficulty arises from the continuity in the sizes of objects and scales of representations in the computer as opposed to those of manual drawings. Most design delivery practices are codified in the

conventions of particular drawing scales: engineering scales for the site, 32nd or 16th scale for preliminary design, 8th scale for design development, 8th scale and up for working drawings, and so on 2 Furthermore, specific decisions are considered only in certain drawings-exterior appearance in elevations, riser tread relations in sections, and so on-and there are generally understood dependencies between decisions taken while considering different drawings proportions of volumes relating sections and plan, and so on. On the other hand, sizes and scales of objects in the computer are continuous. While this permits the mimicking of the manual mode-i.e., fixing the scales in a computer application to correspond to those of the manual domain-it also permits the consideration of scaleless, or at best ambiguously scaled, objects. The distinctions established through the conventions of different manual representations get blurred in the computer. All decisions without any obvious hierarchic order can be considered on the nondiscriminating medium of the computer screen.

In approaching this studio, I considered the discrepencies between the manual and computational design modes a central pedagogic issue. The studio was structured through a series of sketch problems explicitly specifying the order and content of partial solutions to be generated. Students were allowed to use solutions generated for earlier sketch problems, but were restricted each time to certain patches of site-specified by a given, actual size-and a fixed set of design issues. During the second half of the studio the students were asked to return to a top-down design process. The results of this approach were:

1. Design commenced with a manageably small part of the site. This eliminated the consideration of the entire universe of design scales all at once and imposed a certain hierarchy of representations based on scale.
2. Each sketch problem introduced new issues along with scales in increments. This imposed a specific agenda of design issues and maintained the entire design problem at a manageable level.
3. The sequence of sketch problems allowed the incremental building of designs from scratch. This allowed the students to rely less on manual sketching and more on computer-generated designs.
4. Sketch problems constituted a bottom-up strategy, which is so often missing from the student's experience and which provides a counterpoint to the traditional top-down design strategy. This aided students in anticipating low-level concerns once they started their top-down design process.

Conduct of the Studio at CMU

Before we return to questions about the nature of computational futures for design, let me describe briefly the parameters of the design studio offered during Fall 1989 at Carnegie-Mellon University.

Objectives

The major factor determining the conduct of this studio was that it was in the mainstream of the "design sequence" of the department of architecture. That is, it was not differentiated in terms of its educational objectives from any of the other studios at the same level. It was placed in the fourth year of a five-year bachelor of architecture program.

The studio was designed to teach generation of alternative design solutions on the basis of geometric principles and programmatic requirements; use of basic structural and energy analysis packages to help evaluate and select solutions from alternatives;[3] presentation of work through electronically produced objects and the electronic media; and use of computers, in general, in the process of design and production of analytical and presentation documents.

Facilities

Seven Sun/60s were dedicated to the studio. Five of them were driven by independent disk drives. The remaining two were hosted by one of the five disk drives, the one dedicated to the instructor of the course. To facilitate file transfer and flexibility in saving large drawing files, all Suns were networked through the CMU campus network system.

An HP-plotter was dedicated to the studio. In addition a "jetstream" color printer was made available to the students to ease plot queues during submission times. Digitizer tablets and mouse input were not available. Site data was entered at the department's research lab and copied into the studio's file area over the campus-net. While being able to transfer files through the network was a great advantage, being on campus-net also made the studio's operating system overly complicated. We began to expect regular breakdowns of the network, and they became permanent toward the end.

Software

The software used was Autocad , version-9. Although there were plans to obtain additional software, particularly Autocad version-10, Autoshade , and HVAC and structural-analysis packages compatible with Autocad a last-minute cut in the studio's budget prevented their purchase.

Setup

It was intended that software and hardware acquisition and installation be complete by June 1988, in preparation for the Fall 1988 semester. Due to ubiquitous delays in delivery and setup, these installations were not in place until the third week of the Fall semester. This made preparation of special software, algorithms for generation of standard shapes, analysis packages, and general-skill development exercises impossible. In addition, much of the system unreliability resulted from late and hasty setup procedures.

Staffing

The studio was taught by a full-time faculty member experienced in both CAD and design instruction. Two part-time teaching assistants, one a senior in the Bachelor of Architecture program who had taken a similar studio in the past and the other a graduate student with considerable Autocad experience, were assigned to help the studio with Autocad instruction.

Departmental priority was established to make sure that the facility was kept in working order twenty-four hours a day. Nevertheless, hardware problems caused considerable disruption of work. At the end of the studio, when the final presentation work was being done, the manager of the departmental lab left his position. This was a considerable hardship for the studio and negatively influenced the final documentation of work.

Design Problem

The studio was structured around a primary design problem selected to emphasize two key issues: geometry and designing with programmatic requirements that are both repetitive and diverse. The program used was a regional Islamic Center complex. The Islamic tradition of using geometry both as applied pattern and as generator of form in most public buildings was intended to set up the former objective. The selection of the functions of the Islamic complex to include a variety of room sizes and span conditions with many repeated functions-i.e., mosque, school, library, performance center, exhibition hall, administration-was intended to set up the latter objective.

The studio was structured in two parts. In the first part, which lasted seven weeks, students were given short sketch problems. Each one focused on an aspect of the problem in a bottom-up order. The first sketch problem required students to design an Islamic niche at l/2” = I´ - 0” scale. The second covered a larger area of the site at 1/4”= 1´ - 0” scale and focused on a garden or building containing the niche. The third through sixth sketch problems focused on increasingly larger parts of the site and dealt with solar, structural, circulation, and site organization issues, respectively.

In the second part of the studio, which was eight weeks long, a top-down approach was taken, starting with site plan and working down to building and interior design, as in the traditional manner. At this stage the students were expected to achieve better integration between the issues germane to different scales and the top-down versus bottom-up approaches to design.

Students

Students[4] were introduced to computing through two previous courses: an introductory modeling course and a Pascal programming course. This provided some degree of familiarity with the computer systems on campus and basic programming skills. However, as both courses are required in the first years of the undergraduate program, most students did not show a great deal of retention of the material at the time they entered the studio.

By and large, the students who were enrolled in this course elected to be in it. This meant that many of them were motivated to learn to use the computer, and the others were already knowledgeable about computers. This in many respects made up for the operational difficulties cited earlier.

No more than ten students were allowed to enroll in this studio because of the additional instruction time needed to cover technical aspects of using the computer and of the limited number of workstations that could be dedicated to the students.

Projects

Eight of the ten students enrolled in the course completed their projects on the computer. (The remaining two decided to drop the studio from their course load). Figures 1-7 show a selection of student drawings . Below is a synopsis of each students performance.

S1. A strong design student, who accepted parameters of the studio readily and grasped computer's capabilities and limitations. Used both traditional and computer media simultaneously, keeping a top-down as well as a bottom-up strategy alive throughout. Adopted a strict convention of scales and design issues. Final design shows the cumulative effect of incremental design decisions. Produced a strong design.

S2. A strong design student, enthusiastic about CAD. Used the computer exclusively to design. Became the studio's in-house technical expert, which was a burden on him. Worked well with each problem but was unable to integrate the top-down and bottom-up strategies. Could not manage to keep the different scales of design separate. Was unable to develop a solution in the site scale. Produced a weak design.

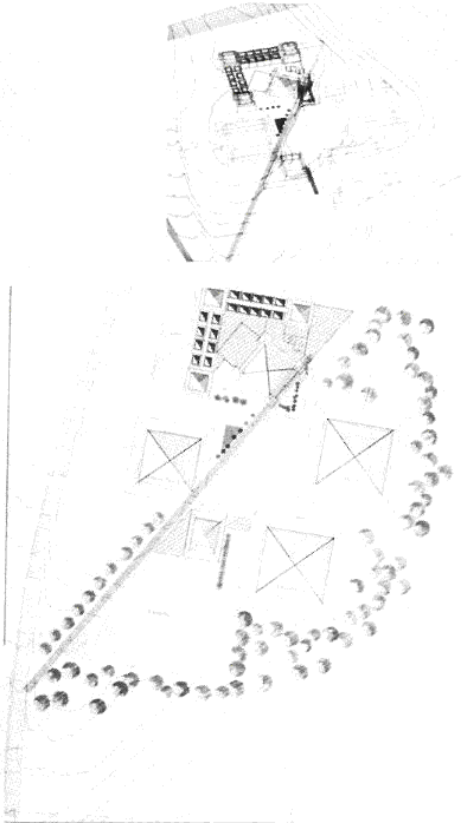

Figure 1 Drawing by Student-I. Generated using AutoCAD and rendered by band. Plan view and axonometric of 2 1/2 D model of site plan.

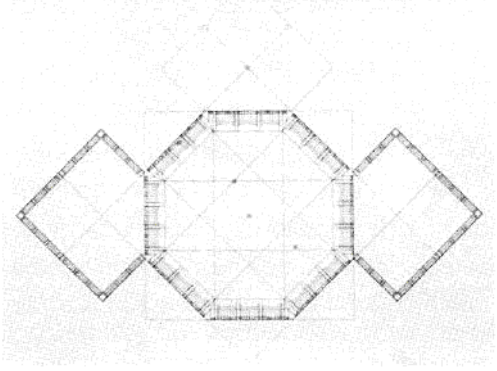

Figure 2 Drawing by Student-3. Generated using AutoCAD - Plan view of partial 21/2 D model of mosque.

S3. A competent design student, enthusiastic about CAD. Did not work very hard and did not gain enough momentum with the new medium. Could not bridge the bottom-up and top-down approaches. Did not succeed in integrating partial solutions into the final design. His files were lost during a system crash just before final submission deadline. Submitted an incomplete and late project.
S4. A competent design student, who kept a manual process going in parallel with the computer. Did most design manually. Was unable to bridge the top-down and bottom-up processes. Developed some parts of the project with great skill. Kept a clear decomposition of the problem by scale. Produced an inconsistent design.
S5. A competent design student, who developed a redundant process duplicating manual and computer output. Was unable to bridge the top-down and bottom-up processes. Developed some parts of the project with great skill. Kept a clear decomposition of the problem by scale. Produced an inconsistent design.
S6. A weak design student, unable to control scale of the design both on and off the computer. The geometric and repetitive capabilities of the computer proved to be overwhelming. Produced a poor design.
S7. A weak design student, who gained computer skills quickly and utilized its geometric and repetitive capabilities well. Kept a strict scalar decomposition and integrated top-down and bottom-up strategies. Was able to integrate partial solutions cumulatively in the final design. Produced a strong design.
S8. A competent design student, who adapted to the medium perfectly. Maintained a clear scalar decomposition. Built design incrementally by integrating new partial solutions into the overall design. Was able to benefit from the top-down / bottom-up strategies. Produced a strong design.
S9. A competent design student (worked with Sb). Was unable to adapt to the studio and dropped it.
S10. A strong design student (worked with S9). Was unable to adapt to the studio and dropped it.

Many, if not all, of the students were disappointed by the constant inconvenience caused by hardware and software problems in the studio. This in general contributed a great deal to the performance slump in the studio, particularly toward the end of the semester.

Conclusions: Expectations for Computational Design

Let us now return to some of the issues outlined in the introduction and examine those characteristics of the student, the instruction, and the instructor which are expected to change through the use of the computer in the design studio.

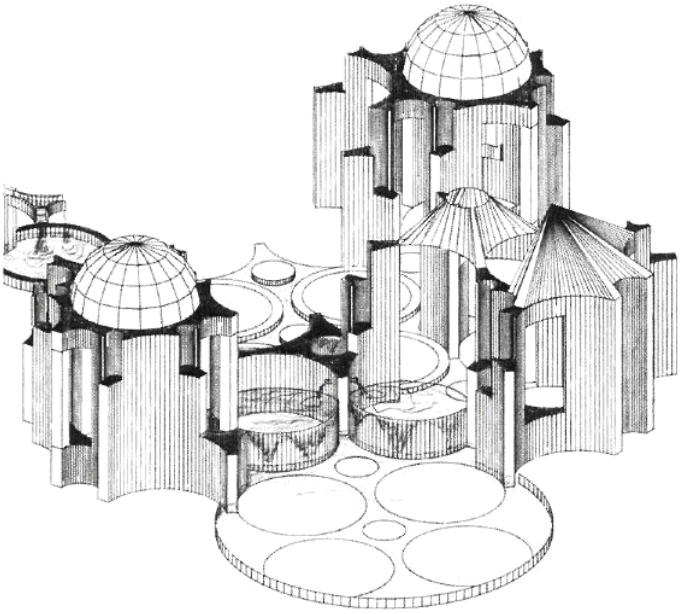

Figure 3 Drawing by Student-4. Generated using AutoCAD and rendered by hand. Axonometric view of 2 1/2 D model of graveyard and mausolea. Hidden lines removed.

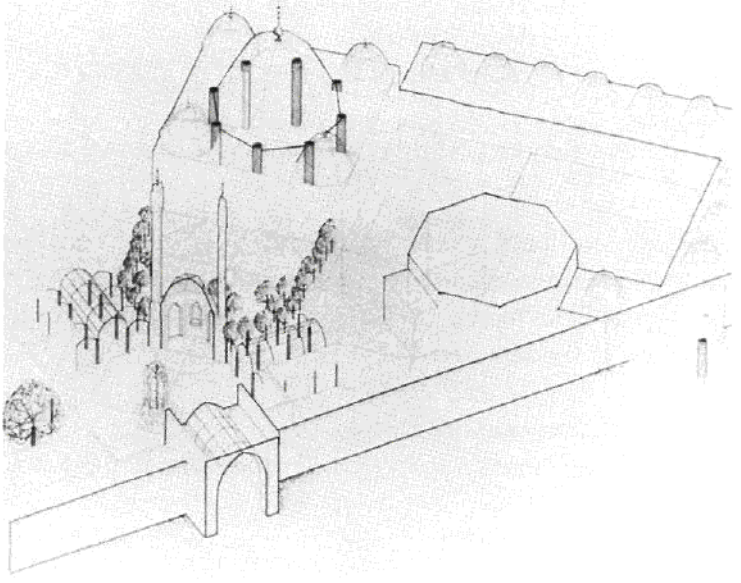

Figure 4 Drawing by Student-5. Generated using AutoCAD and rendered by hand. Axonometric view of partial 2 1/2 D model of mosque and entry court. Hidden lines removed.

Student in the Computational Design Mode

Several changes for the student in the design mode can be predicted: *self-reliance, process consciousness,* and *technological sophistication.*

The student is likely to become more *self-reliant,* able to explore and verify ideas independently, within the new medium, which is technically more robust and consistent than any other medium he/she knows, particularly in terms of geometric manipulations and permutations of form. In these areas the computer automatically extends the skills and knowledge of the designer.

In the CMU studio this manifested itself primarily in the form of an overriding presence of geometric manipulation of forms. This, however, should not be construed as across-the-board sophistication with geometry. In some cases, such as S7, it indeed meant that the entire design became a geometric idea. A sense of Islamic design was achieved through the use of geometry. In another instance (56), geometry was a thin veil behind which chaos reigned. A false sense of credibility was induced by the computer, particularly in this case, lulling the student to inaction. Ultimately, geometry became a crutch for a weak design.

As discussed earlier, there are fundamental differences between the design processes normally used in the studio and those compatible with a computing environment. If they are to benefit from this medium, students are expected to become more self-conscious about their *process* and to adapt it to the offerings of the computer. The results in the studio were mixed in this respect as well. 58 made the best adjustment to the medium and used the computer to generate formal ideas, as well as to examine and manipulate them in the computer. S1, also in control of his process, did not modify it, however, other than to multiply his effort and use the computer only as a representational and evaluative medium. S9 and S10, on the other hand, tried desperately, and without success, to integrate the computer into their process. This caused them to drop the studio.

The computer, regardless of the friendliness of the software and the abundance of user consultants and maintenance crews-which was not the case in this studio-requires some *technological sophistication* from the user, particularly in the case of nontrivial networking setups and computation environments. Often the student is required to troubleshoot and perform basic maintenance tasks. In fact this sort of relationship with the medium is desirable, particularly in "creative" fields such as design. No one can predict the day when craft" will leave the practice of architecture 6 As any craftsman, the designer using the computer should have the ability to manipulate with skill his/her tools as well as his/her products.

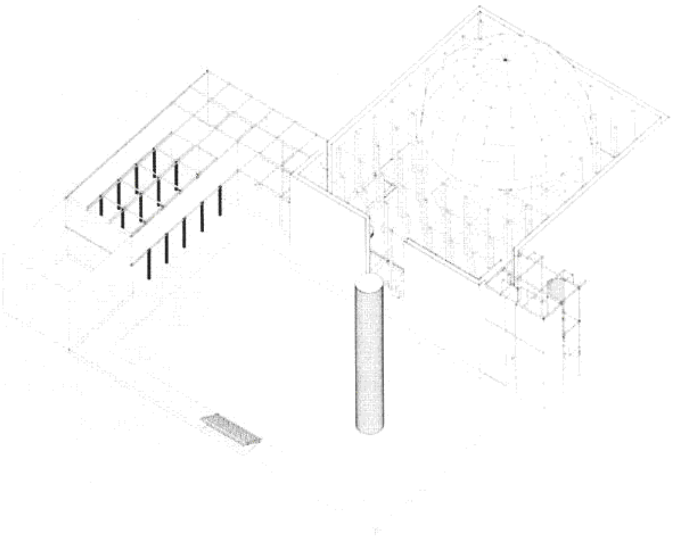

Figure 5 Drawing by Student-6. Generated using AutoCAD and rendered by hand. Axonometric of 2 1/2 D model of mosque. Hidden lines removed.

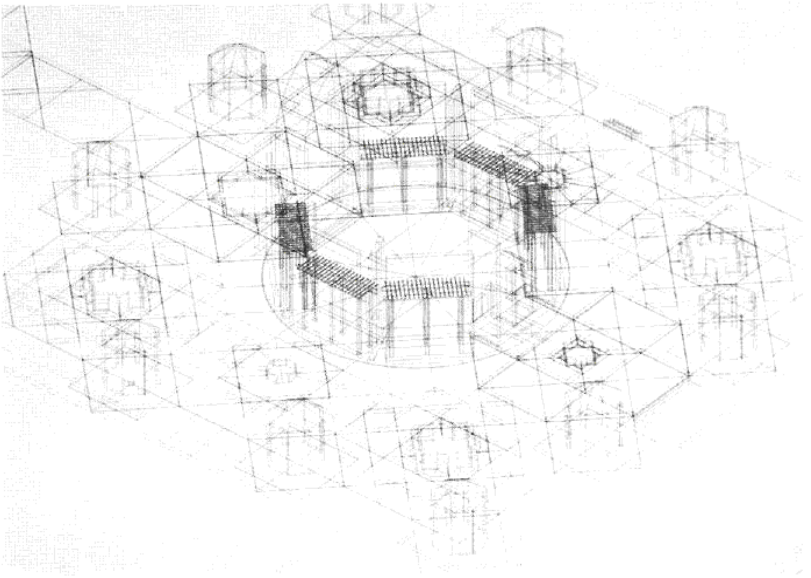

Figure 6 Drawing by Student-7. Generated using AutoCAD and rendered by hand. Axonometric view of partial 2 1/2 D model *of* outdoor mosque, graveyard and mausolea.

One student in the studio ventured in this direction (S2). He was a talented designer, and his instincts compelled him to try to master the medium. However, a lack of technical support in the studio and demands for technical help from his peers created problems. This student eventually became the in-house expert and spent most of his time helping his peers by searching for files, transferring files, setting up peripheral devices, and discovering new ways of modeling form in the limited 2-l/2-d world of Autocad. As a result his design suffered.

Instruction in the Computational Design Mode

Instruction is also bound to undergo fundamental changes in the computer environment. Primarily, issues of *design process will* become more explicit. *Design problems* that capitalize on the advantages of computing will be preferred over others. *Standard information* about codes, details, and quantifiable computations will readily be assumed to exist as part of the designers' capabilities. Time will have to be allocated to the instruction of operations specific to the *medium* .

The contention here is that by virtue of the new medium the *design process* will become radically different for the designer who strives to employ the full potential of the computer. Earlier I outlined the difference the computer makes in the design process. I argued that in the computational design studio, unlike its manual counterpart, design should commence with a manageably small part of the site; each subsequent step should introduce a new issue commensurate with the scale used; the sequence of design steps should help the accumulation of sufficient information to make designing on the computer as palatable and as quick as possible; and a bottom-up strategy should be induced to provide a counterpoint to the traditional top-down design strategy.

The studio showed that the first three recommendations work reasonably well. Working with the computer, students can gain sufficient momentum through a sequence of carefully selected subproblems and gain both confidence and insight. The primary difficulty was the bottom-up approach, which most of the students had never used before and had a hard time accepting.

The type of *design problem* suitable for the computational studio context is also an important issue. It is essential that the problem type offers the computer medium some advantage. In this studio, geometry and geometric manipulation were the principal criteria, which led to the decision to use a program, an Islamic Center complex, which traditionally is derived from geometric concepts. Other options include complex institutional buildings for the building performance analysis approach used by Schmitt, the syntactic analysis of landmark buildings by famous architects used by Flemming, and basic design exercises used by Mitchell.

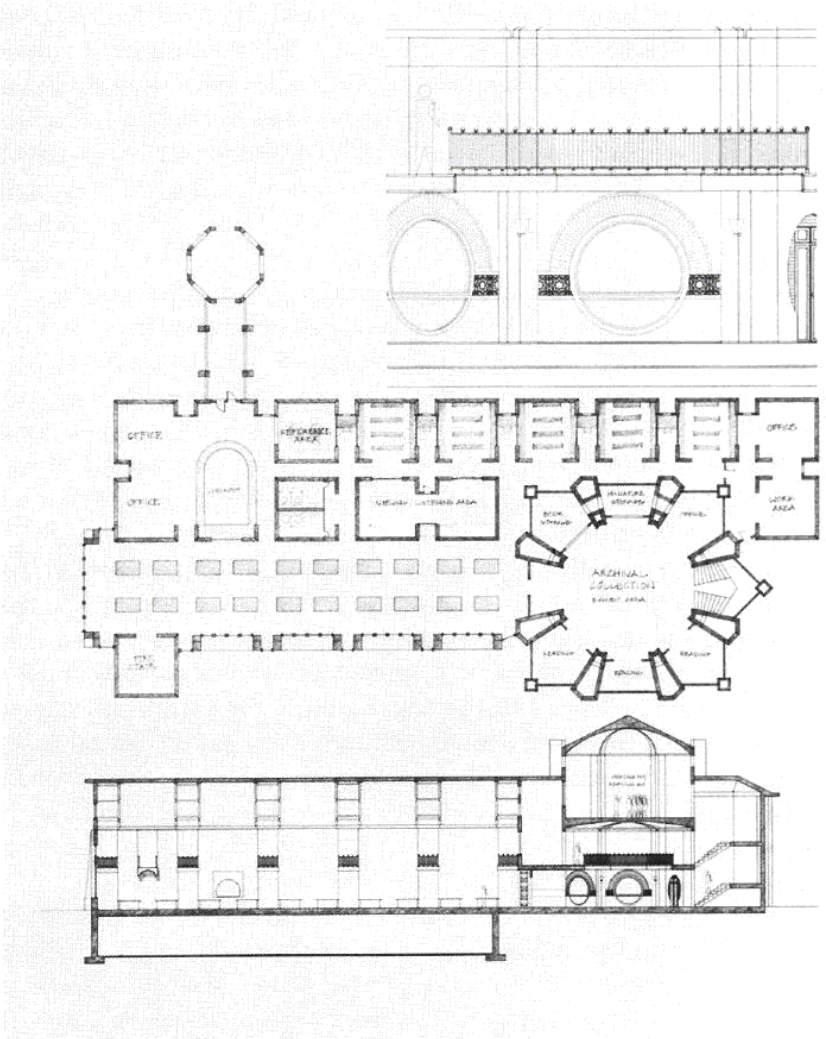

Figure 7 Drawing by Student-S. Generated using AutoCAD . Plan, section, and detail elevation of library.

There are other important design criteria for the computational design studio, which, for circumstantial reasons, we were not able to use to full advantage. One of these is the incorporation of standard information-codes, details, and other analysis algorithms-in a central database for all students to share. In our case there was not enough time to set up such an environment. Some of this spontaneously developed in the studio through shared, standard shapes, including both structural and cosmetic elements such as domes, arches, vaults, human figures, trees, figures,

and so on. Inclusion of standard analysis packages in a central database can improve the students' ability to conduct fast, accurate analysis of structures, HVAC, and the like. This would have necessarily modified the expectations from students and increased the number of design criteria to be satisfied by a design.

Finally, an important factor in any computational design instruction context is the time invested in learning about the *medium*. In our case one week was allocated to refreshing students' memory of Autocad commands and introducing them to version-9. This was adequate but would not have been so had we been also tackled issues of solids modeling and building performance analysis in this studio.

Instructor in the Computational Design Mode

Perhaps the most critical change in the educational context is needed in this category. Yet there is very little to say about it, specifically. The instructor by default has to deal with all of the above issues: he/she needs to design the course, establish its pedagogic objectives, configure its computational milieu, monitor the students' progress through the course, and be the judge of how well he/she is doing as an instructor.

More pointedly, the instructor is expected to learn about the functionalities of the computer and design operations compatible with computation. He/she must be able to define processes of design that encourage productive use of the computer during both design and presentation. Also, the instructor will have to acquire new skills or aids to supplement his/her knowledge about technical functionalities of the computer-such as, geometric manipulations of form, technical analysis of buildings, and general competence with the use of the medium of representation. This is a nontrivial body of knowledge.

References

Akin, Omer. *1986. Psychology of Architectural Design.* London: Pion Ltd.,

Flemming, Ulrich. *1989.* "Syntactic Structures in Architecture". This volume.

Hemming, Ulrich, and Gerhard Schmitt. *1986.* "The Computer in the Design Studio. Ideas and Exercises that Go Beyond Automated Drafting." *ACADIA Proceedings. '86.*

Ledewitz, Stefani. *1985.* 'Models of Design in Studio Teaching." *Journal of Architectural Education 38, pp. 2-7.*

Mitchell, William J. 1985, "A Computational Approach to Basic Design." In Pioneers *of* CAD in Architecture. A. Kemper (ed.), Pacifica, CA: Hurland/Swenson Publishers.

Mitchell, William, Robin S Liggett, and Milton Tan. 1989. "Top-Down Knowledge-Based Design". This volume.

Notes

1. At the time Pioneers *of* CAD in Architecture was published in 1985, ten schools reported their pedagogic activities: UCLA, Carnegie-Mellon University, Harvard University, University of Houston, Iowa State University, University of Michigan, SUNY Buffalo, North Carolina State University, Ohio State University, Rensselaer Polytechnic Institute, and Virginia Polytechnic Institute. It is safe to assume that this is at best a conservative estimate of the total number of schools teaching CAD. It does not include the token CAD instruction that extraneous demands sometimes elicit, such as NAAB requirements, public relations, and so on. Even so, and not taking into account the growth since 1985, which could double or quadruple the earlier number, it is fair to say that a significant number of accredited schools of architecture in the nation have a serious commitment to CAD.

2. The specific scales may differ from project to project; however, their relative values remain fairly constant.

3. This was de-emphasized later because of the unavailability of funds to obtain needed software.

4 In this study students will be treated as subjects taking part in surveys and empirical experiments. Their identity is suppressed to protect them from possible unwarranted or undesirable exposure. Thus, from here on they are referred to as S1 (Student-1), S2, S3, S4, S5, S6, S7, S8, S9, and S10. To acknowledge their work, however, I wish to credit them by name: Azizan Abdul Aziz, Dan Cohen, Mathew Carter, Catherine McCall, Steven Park, Glynnis Patterson, Steven Shanley, Andrew Grossi, Henry Altman, and Al DeSantis.

5. All of these figures are replicated from hard-line drawings produced on the HP

plotter; some are augmented by renderings by hand.

6. Even in the case of computer applications, we see the effect of manual dexterity, particularly in presentation drawings.

7. These include the basic command structure of a given software to create operations that correspond to manual design acts, such as drafting, symbolic manipulation, simulating, manipulating, evaluating, selecting, etc.