DEVELOPING CRITERIA FOR PC-CADD EVALUATIONS: A CASE STUDY

by

Fadi J. Ariss

Bachelor of Science in Art and Design
Massachusetts Institute of Technology
Cambridge, Massachusetts
1980

SUBMITTED TO THE DEPARTMENT OF ARCHITECTURE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE
MASTER IN ARCHITECTURE AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 1987

Signature of the author [illegible]
Fadi J. Ariss
Department of Architecture
October 31, 1986

Certified by [illegible]
Patrick A. Purcell
Visiting Associate Professor of Computer Graphics
Thesis Advisor

Accepted by [illegible]
Judy Dayton Mitchell
Chairperson
Departmental Committee on Graduate Students

Developing Criteria for PC-CADD Evaluation: A Case Study

by
Fadi J. Ariss

Submitted to the Department of Architecture on October 31, 1986
in partial fulfillment of the requirements for the Degree of
Master in Architecture

ABSTRACT

While the usefulness of CADD in the production stages of the architectural design process is now widely acknowledged, its true benefits for earlier, schematic design phases remain unclear. Due to the very subjective nature of that process and the complexity of its overlap with other design stages, schematic design is a difficult task to model and emulate, therefore making the appraisal of CADD as a design aid a delicate problem. The last few years have witnessed a large increase of PC based CADD systems, most of which aspire to qualify as "true design tools"; however, outside of product release announcements and industry comparative checklists in the professional press, there is little work on the assessment of the role of CADD systems in preliminary design.
This thesis is an attempt to develop a strategy for the evaluation of CADD systems as a tool for design in its early conceptual stages. A specific system is selected and assimilated to a level of proficiency. Its main characteristics are then discussed and compared to the similar characteristics of another "standard" generic system, which is the most currently used system. They are analyzed in terms of their relevancy as effective design aids, based on my own observations of the system; they are also tested through a short design exercise.
The purpose of the study is to identify what constitutes valid parameters for the assessment of a system performance. Its main functions are prioritized and investigated as to whether they truly assist the user in his design process, with a particular emphasis on geometric modelling, visualisation and system interface. A set of performance criteria is derived along with their desirable attributes, so as to develop a comprehensive approach towards CADD evaluation.

Thesis Supervisor: Patrick A. Purcell
Title: Visiting Associate Professor of Computer Graphics

ACKNOWLEDGEMENTS

Thanks to Patrick, Earl, Karine, Bill, Bill.K., Paul, Eddie for constant support

Special thanks to my parents for their patience

TABLE OF CONTENTS

INTRODUCTION

"It is ironic that the word CAD has come to assume two very different and almost mutually exclusive terms, design and drafting."[Stoker 83]

This statement applies very well to most design and drafting disciplines. Design in itself is a complex process which demands a different set of tools than those required by drafting. These tools are used to manipulate abstract concepts rather than concrete geometric primitives. I find the argument only partly true, however, for architectural design and drafting; many aspects of the design/drafting cycle strongly differentiate it .

In most design processes, the draftsman is not an expert in the field of knowledge of the artefact he is drafting. The people who draft electronic circuit layouts know little about the circuit they are representing. Architectural draftsmen, on the other hand, mostly understand the drawing and what it represents. They are often architects themselves.

Architects like the tools of their trade; they like to draw. this partly explains their traditional suspicion towards any alternative tools , fearing it will substitute to pencil and paper.

In addition there is something about the architectural process which makes it difficult to delineate exactly where design ends and drafting starts. They are both intuitive and equally ambiguous.

There is a very intimate connection between the designer's thought process and the representation of the result of that process in a sketch or a diagram. Conceptual schematic and preliminary design are rather vague terms which overlap and are interpreted differently by different designers. Where does a bubble diagram fit in the design/drafting process (assuming one even starts with bubble diagrams)? When do the single wavy lines of an adjacency layout explode into double lines of walls with varying degrees of privacy? Even the careful hard line placement or dimensioning of an opening in a all, during design development, involves a minimum amount of decision-making by the draftsman which may qualify as a design decision. Sketch designing or whatever we chose to call it, is a process of browsing, doodling, and confirmation. The wobbliness of lines in a sketch has an important role in relation to the design solution which that sketch represents. The texture of graphite on paper, the different pressures applied on the line, the hand movements and hesitations, all express a range of completeness and certainty; some of the decisions on parts of the sketch have more permanency drawn into them than others. Some of these qualities will remain difficult or impossible for the machine to emulate.

On The Nature of Architectural Design

There are a number of reasons for which it is difficult to assess the performance of a CADD system in a design problem; many of them are directly related to both the process and the product of architectural design. Only in architecture is the visual representation of a design problem so linked to its solution. Whether we choose to call the product of that design process the actual building, or the set of documents which represent its description, that product is difficult to evaluate. Virtually all literature pertaining to CAD begins with a section dicussing the particularities of the architectural process, which differs in many ways from other design disciplines where CAD is already successfully implemented.

Design as problem solving

Most design processes can be described as a process of goal definition and problem solving. This is accomplished by

- stating the goal
- planning a strategy to reach that goal
- defining the constraints
- implementing the strategy.

The architectural design process is not dominated by the need to work through structured, well-defined problems. As s process of synthesis, it is not subject to explicit and

complete constraints. The validating of the problem solution may rely itself on conflicting criteria.
Another factor differentiating architectural design as problem solving is the need to generate the state of potential solutions before the design can be assessed. In addition, there may be (and there usually is) many different acceptable solutions.

There is a variety of ways in which architects view their design. Design objects are subject to a wide diversity of both expressions and perceptions. The physical artifacts resulting from these solutions are often embedded in some form of context.
Even in architectural drafting, there are less shared conventions and pictorial symbols to communicate the design product, than mechanical engineers for instance. There is virtually no common standard way to communicate the result of a preliminary design phase.

Design as an iterative process

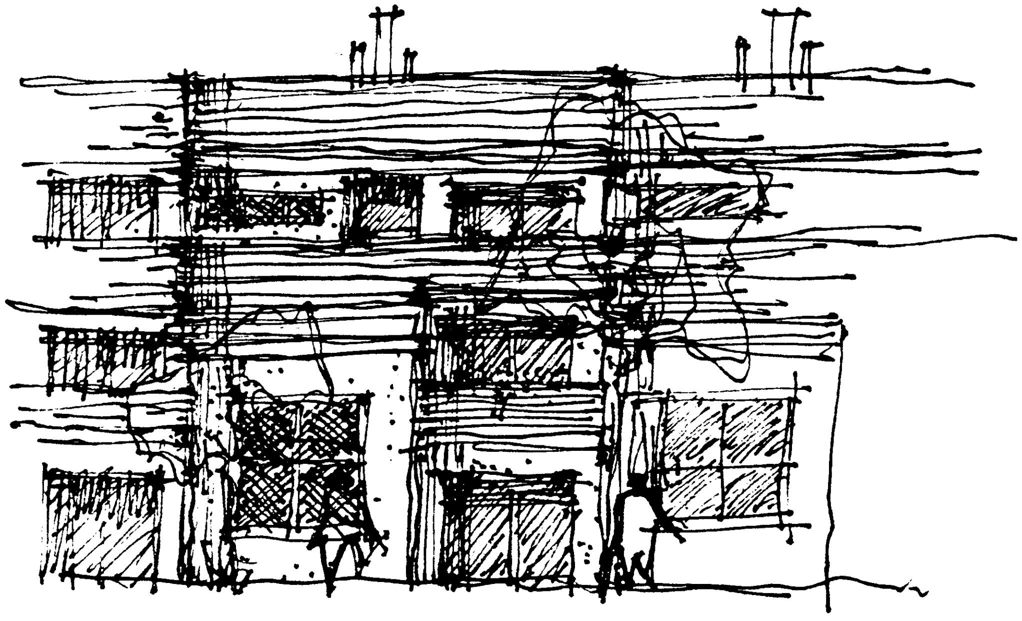
Since the problem solutions have to be generated before they can be assessed, parts of the design process may be seen as a process of "event exploration", in which partial responses lead to the constant redefinition of the goal.
The process is therefore an iterative cycle, where each iteration of solution/evaluation influences the next one [Greenberg 84]. It is an educated "trial and error" process

which relies heavily on knowledge and experience. This search through alternative solutions is similar to a loop through a learning process; the tighter the loop is between solution and evaluation, the more effective the final solution will be.

Data organisation

Architects rely on information which is by and large external to the problem. The knowledge which guides the design process can be described as abstract, ambiguous and often contradictory. Using this information implies the selection and prioritization of overlapping criteria; that process by itself is as intuitive as the nature of the knowledge. Intuitive knowledge refers to knowledge acquired by individual experience, learned inwardly, and subjectively assessed.
The organisation of data is very different in architecture from other design professions [Stoker 83]. As an example, an electronic engineer uses a few chips to develop an extensive circuit design. Since he uses many instances of a small number of elements, his data structure is narrow and deep. The architect, on the other hand, uses relatively few copies of a very large number of components, ranging in type or scale from a door knob to a precast wall panel; his body of data can be characterized as broad and shallow, and therefore more complex. There is no predefined organisation of his field of expertise; it becomes a kind of repositery for a vast array of information.
This does not necessarily imply that it is harder to

understand. The architect's desk, with its pile of yellow trace, sketches, cardboard models and notes, is another example of a broad database of random, ill-defined information; yet this data can be quickly understood and assessed by different individuals.



Current CADD Evaluations

Most current evaluations rely heavily on "benchmark" tests and derive comparative checklist charts. Benchmark tests are generalized assesment procedures, designed to test the system performance in carrying out typical user tasks. Such activity analysis can be derived for many disciplines. In architectural design, though, it may be very difficult to define what constitutes a typical user task, as working methods vary with the designer. The question of whether it is possible to design benchmarks for preliminary design is one worth asking.
Another problem with such reports is the fact that, due to the rapid changes in the CADD industry, these evaluations tend to become obsolete. Most sytems are regularly updated with the addition of new releases, which invariably offer some of the features other programs provide that they lacked. Often these changes take place as the program itself is being evaluated. There are even instances where, ironically, the evaluation itself starts dictating the changes between two successive versions of a software. Benchmark tests provide for a rigorous experimental evaluation of system effectiveness; they remain nevertheless only indicative and short lived.

An alternative method is to test the system for real life projects in a user environment, as opposed to testing it through specific benchmarks and user interviews. This evaluation strategy has the benefit of reflecting the needs of the user rather than those of the evaluation team. However, it also occurs in specific users organizations and is tainted with their design methods and philosophies, which other users may find objectionable. It is impossible to simulate and model all user environments; defining the evaluation context is therefore critical.

Another strategy for assessing CADD might be to evaluate the solution to a design problem and to compare it with the solution produced without the help of the system. In this case the difficulty lies in the assessment of the solution itself: there are always more than one acceptable solution to a design problem. How do we determine which of the two solutions, with or without CADD, is the better one ? Does better mean more efficient in terms of objective function, or does it include aesthetic form appraisal ? What constitutes a valid measure of a building "quality" ? There are all issues involved in this type of strategy.

One of the first steps of any evaluation is establishing the requirements of the user. The needs of an typical end user utilizing the system for preliminary design are frequently vague and sometimes contradictory. The key to a successful evaluation is a comprehensive analysis of the expectations of the user from the system; the measure of the performance will always be relative to these requirements. The list of needs can be ordered by priority and may be used later as a basic set of criteria.

The nature of the program for different applications often dictates different approaches; for example, the need for accuracy in architectural drafting and production is of little importance to the casual user who uses the system for preliminary geometrical modelling. A program suitable to a user or an organization may not be suitable to others. The evaluation should therefore examine the system performance against a set of attributes; it is up to the reader to weigh the relative successes and failures of the system, and determine the suitability of a particular program for a given individual or organisation. An evaluation strategy, therefore, is always based on the merits of its subject; it should be flexible and obey elementary rules, dictated by common sense. However, the flexibility of a system to lend itself to different users should not be confused with generality, where the system reduces the program to the solutioning of a general case [Bensasson 79]. There are two dangers for a system to seek the solving of the general case:

- The general case may be so trivial that the use of the computers for the solution may not be justified.
- The general case may be general in the sense that it covers the majority of instances of a problem in a specified way.This could lead to the adaptation of the CAD process to fit the capabilities of the program, which is not acceptable.

Some of the less subjective criteria for the assessment can be divided in two categories [Teicholz 86]:

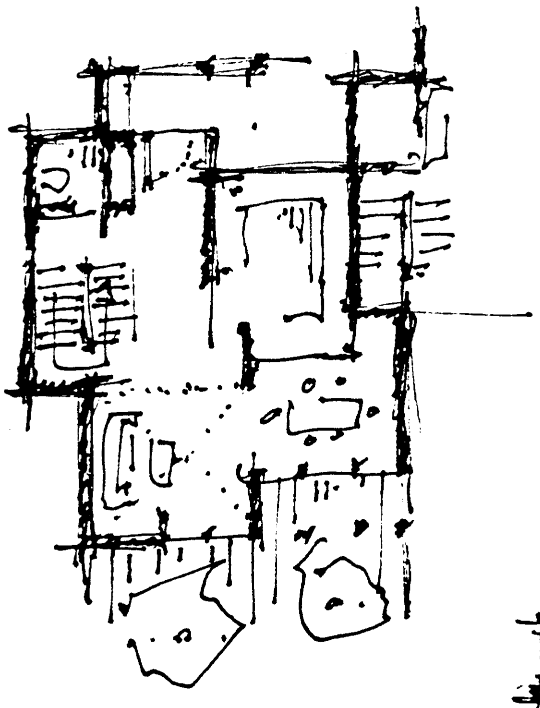
- Performance characteristics, which are related to the system efficiency and capacity for useful work, such as response time, throughput, potential for expansion, reliability, online data storage and communication (fast referencing of online graphic or textual data), potential for integration,...

- Utility characteristics, which describe what the system can do. The functional capabilities of a system describe the kind of operations the system handles, and the relative difficulty involved by the user in performing them. There is a difference in assessing "what" the system can do as opposed to "how" or "how well" it does it.

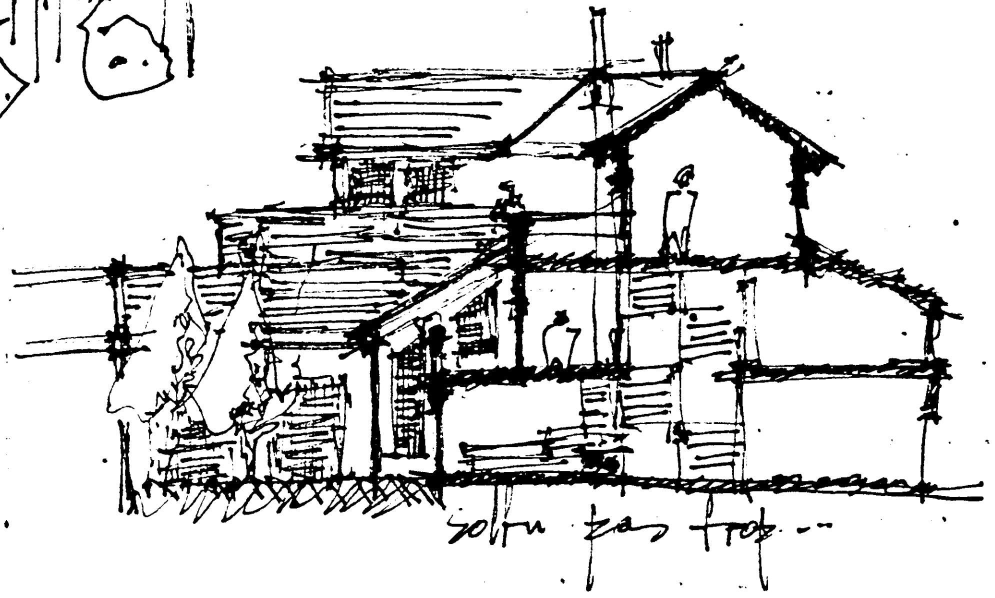
Ease of operation and ease of learning are directly linked to the qualitative concept of relative difficulty in getting the system to perform an operation. They are further discussed in the section on the interface. They remain. however, subjective criteria which are difficult to evaluate, for three reasons:

- They are a function of the user's skill.
- What is good for a "casual" operator is not necessarily the same for a full time operator.
- Long term users may have learned to live with the inconveniences of the system and work their way around them.

Other nonquantifiable factors an evaluation should take into account are more subjective and rely on individual perception of the system's potential role as a design assistant. There is nothing wrong with subjectivity of criteria, as long as the assumption of its relevancy remains valid. An evaluation which solely relies on definitive predetermined methods and rigid facts runs the risk of being restricted to trivial information. Such subjective criteria would attempt to answer some questions about the general feeling of the system, its limitations and possibilities from a design point of view, its value for architects, what is new about it and what it does best.



Preliminary sketches of a rowhouse



CADD Definitions and Descriptions

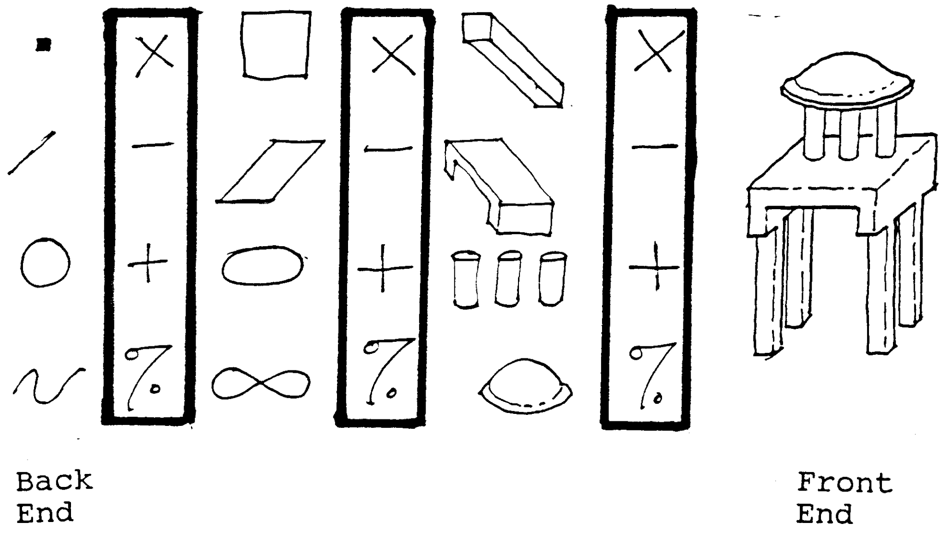
The basic function of a CADD system is the creation of individual entities of geometry and their manipulation through a drawing editor to describe the potential solution of a design problem. The graphic primitives may be points, lines, arcs, circles, ellipses, curves or polygons; they are inputted by means of a physical input device such as a digitizing tablet, and edited through a set of commands instructed to the system. A primitive can be moved, copied, modified, mirrored, rotated, scaled or combined with other primitives, so as to generate shapes and objects representing architectural elements which will later describe parts or the totality of a building.

Generic v/s Front End Systems

The process of going from the creation of the graphic primitives to the building representation involves many steps of combination of graphic elements, through different levels of abstraction of the building.The lowest level of abstraction is the one dealing with these basic geometric entities, the points, lines, circles, etc. These entities have little to do with any form of building representation; they are only the indissociable units required to start assembling an object. The highest level of abstraction is the one that deals directly with explicit descriptions of the building as a whole.

It is difficult for a CADD system to effectively handle both ends of the abstraction process, in both low and high level terms. In addition, since architects perceive and assess their design solutions in many different ways, the highest level of abstraction is also the most subjective. In other words it is hard for a system to deal with both the back end of the process (geometry creation and editing) and the front end (assembling elements to generate a building). CADD systems differ in their approach to this issue.

Generic CADD systems deal primarily with the lowest forms of representation, those of individual geometric entities. They are best at the creation and manipulation of lines, circles, etc, into various parts, and do not attempt to address how these parts describe a building. They are mostly drafting tools and concentrate on providing effective drawing editors.


Back End

Front End

Autocad, from Autodesk is an example of a generic CADD system. Its extensive drafting features have made it the de facto standard in the PC based market. It can accomodate a large number of all purpose drafting applications, from mechanical or architectural to electronic circuit design.

Front end systems, on the other hand, are "application specific". They focus on middle or high level of representation of the CADD process, by providing specialized functions geared to architectural purposes. The designer starts dealing with concepts and physical terms which form the tissue of his knowledge, such as walls, openings, volumes or planes. Since that phase of the CADD process of transformation and its product is also the most subject to interpretation, front end systems vary in the way they assemble primitives into elements that the architect understands. In addition to the drawing editor, they provide a syntax of commands, menus, templates and library of symbols. Current front end systems may be of two kinds:

- Auxiliary support programs are implemented as "add-on's" to an existing generic system. Their specialized functions are designed as a combination of basic commands which are assembled by means of a high level programming language, provided with the generic system. For example, a number of such systems currently use AutoLisp, a high level language similar to the Lisp language, such as A/E Cadd.
- Another solution consists in providing two distinct modules within one , handling the front and back ends of the process,

which usually communicate and share data. The two interfacing systems are sometimes called "Drafting" and "Design", or "2D" and "3D". However, it will quickly become clear that the equation "back end" = "drafting" = "2D" versus front end" = "design" = "3D" is not always valid and may prove to be a gross simplification.

Case Study: Personal Architect

There are currently about a dozen PC based architectural CADD systems used in the profession, covering a wide range of needs, costs and performances. The Personal Architect is a system recently introduced by Computervision Corporation, geared towards the high end of the low cost personal CADD market. There were specific reasons for the selection of this particular product:

- It is composed of two very different programs which are meant to be used alternatively.
- It exhibits some unusual features with strong implications about the architectural design process.
- A set of circumstances has led to exposure to it in a particular time and context of its development.

THE TWO MODULES

The Personal Architect is an architectural CADD system which attempts to link the phases of schematic design and design production through the transfer between two different modules,

labeled "Design" and "Drafting".
The Drafting module is a generic all-purpose tool for geometry creation, editing and modelling. Most systems which have the option between 2D and 3D, and use it to differentiate between "Drafting" and "Design", tend to rely heavily on the 2D program for most of the drafting and production tasks. In this case, however, the "Drafting" module happens to be a "true" 3D program with a strong emphasis on 3D geometrical modelling.
The "Design" module is also a 3D system which is closer to a typical front-end system and deals therefore with a higher level of architectural elements. The basic entity to work with is a six-sided rectangular volume with a top, a bottom, and four sides. The designer uses a combination of these volumes to generate a wireframe model of the building which is defined by its floors, roofs and walls. The two modules communicate through transfer programs which transform the models generated by one module into a format understandable by the other. While there are always many ways to use either module for a range of purposes, in this case the intended working method as suggested by the authors is the following :

- The user starts by using the Design module to construct the wireframe model of the building through the successive assembly and manipulation of volumes which are defined by their planar boundaries (floors, roofs and walls). This model may be quickly modified to generate alternative solutions based on the programmatic requirements of the project.
- Alternatively, or simultaneously, the "Drafting" module is used to build three-dimensional objects which may represent

any architectural element which is not a floor or a roof. These objects, which may vary from wall openings, furniture or equipment to columns, beams or partitions, are processed through a transfer program. This first transfer allows these elements to be inserted in three dimensions into the wireframe model created in the Design module, for further building definition.

- Once this model is complete and includes all the components of the building, it is processed through a serie of executable programs which contain user-defined and default information about the construction technology of the building (the technology file,described later). These programs perform two kinds of operations: they calculate the building database and they automatically generate a set of drawings describing the building in terms of standard views (plans, sections or elevations) or user-specified views such as axonometrics or three-point perspectives.

-These drawings, created in the Design format, are then transferred again to the Drafting module into two-dimensional drawings which can be edited and "cleaned-up" for final presentation or production drawings.

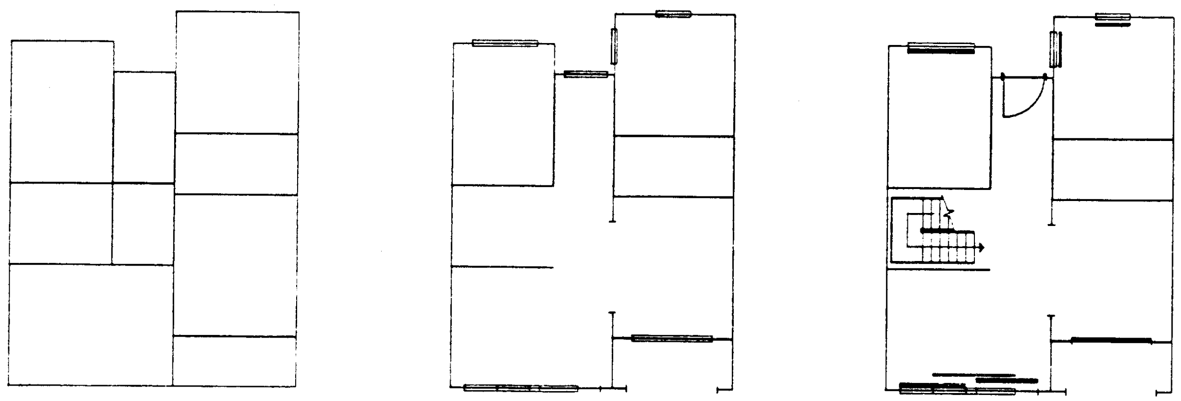
Since this second transfer involves the use of the drafting module as a simple drawing editor focusing on the later phases of the architectural process such as contract or client documents, it does not fall within the scope of this work; the first one, however, appears to be relevant in terms of supporting the Design module, and should be examined within a

design evaluation.
In addition, since the labelling "Design" and "Drafting" is, as we will see, rather simplistic at best, from now on in this work I will chose to use the "birthnames" of the two programs,Keops for the design module and Microcad for the drafting module (not to be confused with another PC system called Microcad, by Imagimedia Technologies).

THE TECHNOLOGY FILE

One of the features which differentiates this particular software from others, beside the concept of volumes, is the the "Technology File". The technology file is a set of construction parameters and design rules. It contains default information about the different components created in the wireframe model such as floors, roofs, walls, partitions, handrails,...It is structured as a relational database describing the technology of the project.
The user is guided through a hierarchy of menus and submenus prompting him for sizes of dimensional elements; his input is interpreted as a serie of if-then conditions which essentially define the design rules of the project. Once this array of information has been generated and saved, the user runs a sequence of programs which create the project database by merging the information of the technology file with the wireframe diagram. These programs perform various operations on the model based on the different values defined by the user. For example, the system will create a plan by exploding

the single lines of the wireframe into double lines with thicknesses reflecting the design rule of that specific wall. Similarly, it will cut a section through the building at a user-defined location, with corresponding floors and roof dimensions. All these values are related to initial parametric assignment of "space categories" such as "heated", "unheated", or "terrace". It is this database that the system uses for generating the hidden line images of the building.
Once the project has been processed through the technology file and the database programs, the architectural process may be described as having reached the the "design development" phase; the drawings are edited, completed and annotated for contract documents; three types of information have been calculated: dimensional information associated with all rooms and openings, quantity takeoffs of materials, and cost.


From volumes to spaces

DEVELOPING CRITERIA

Standard CADD Functions

Most systems perform a minimum of functions in a similar way; this core of functions is handled by virtually all systems which qualify as generic ;it is the "common denominator" of current systems. As an example, all systems offer a number of ways to draw a circle: with three points on the circumference, with two points defining the diameter, or with one center point and an explicit radius dimension, just ot mention a few . These types of functions are now taken for granted, and are therefore not addressed in this work .

Other functions are performed by most systems but are handled differently by each one of them. Different systems may perform dimensioning, for instance, much better than another.

Finally a few systems will perform some functions which others cannot do at all. Hidden line removal or perspective viewing, for instance, are provided by few PC systems.

We have to bear in mind, though, that these differences keep decreasing as every release of a software keeps leapfrogging the pack. Softwares, particularly generic ones, tend to look more and more alike.

If we look at standard industry evaluations as a starting point for organizing general CADD functions, we may classify them in the following categories:

* Geometry creation
* Display control
* Dimensioning
* Text
* 3D
* Drawing aids

First let us keep in mind that these are functions, not evaluation criteria. Other factors routinely mentioned in professional comparative evaluations, such as vendor friendliness, hardware or cost may constitute far more important criteria than those mentioned above. One cannot underestimate hardware or cost as critical elements of a selection process from a professional point of view.

From a design point of view, though, these categories are very broad and cover a range of functions from the indispensable to the unnecessary. They also vary a great deal in the extent of their relevancy to preliminary design. It is clear, for instance, that while text and labelling are two important components of construction drawings, they do not constitute a significant aspect of preliminary design. I chose therefore to disregard these "families" of functions as evaluation criteria.

LOCATIONAL CONSTRAINTS

CADD systems provide for ways of restricting the position of the cursor to specific points, either by referencing existing

Referencing existing geometry involves snapping the cursor to some key point of the closest entity such as end points, origin points, midpoints of lines, circles or arcs; all these are provided with most systems. While such "object snaps" greatly increase accuracy and speed, they have only limited value in perliminary design except for geometric modelling and complex object creation. A more helpful tool, however, is the ability to define a set of legal positions for the cursor, such as grids and snaps.

The simplest form of locational constraint is the orthogonal mode, which insures that all lines drawn are either vertical or horizontal, or , in some cases, at 45 degree angles. Occasionally a system will allow the user to specify the angular constraint. Locks are useful for designs with major angle changes.

Grids permit the user to display a lattice of dots at regularly spaced intervals which he can define. They can help the designer by giving an approximation of the dimension of an entity he is drawing. They allow him to assess the value of a given dimension, by acting as dimensional estimators. Grids may or may not restrict cursor positioning, depending on the "snap" function, which is the next step up in locational constraint.

The snap function restricts the crosshair to the nearest increment of a given value. That value may or may not be the same as the grid value; it is usually preferred to have them independent from each other, letting the grid act solely as a visual guide while the snap value controls the cursor

position. Some snaps allow the crosshair to move anywhere and only snap when digitising; others restrict the crosshair to rest only on legal points. In addition, grids and snaps are very often turned on and off; it is helpful to be able to do so from within a single command, without having to go back to the system prompt.
Grids and snaps vary mostly in the extent to which the user can edit them to reflect his own system of positioning constraints. The principle is the same as for other functions; the more user-definable grids and snaps can be, the more useful they are. This degree of freedom may range from different X and Y values to rotating grids at certain angles; some particularly handy features allow the user to define the grid origin or , better yet, to simultaneously display a major and a minor grid. In general, a grid is only as useful in design as it can be modified to suit a particular working session or project.

Keops, which utilizes grid lines rather than points, provides some unusual grid editing functions. It displays both grid and crosshair in a perspective mode; this allows for quick dimensional estimation in a 3D mode and makes up for the error factor due to perspective distortion. Another valuable feature in Keops is the ability to move, copy or erase every single line independently. In this case the grid becomes more than a modular constraint; grid lines can be used to identify major directions in the building structure or organisation, such as bearing walls, column lines or other directional fields.

PARTS AND BLOCKS

As mentioned earlier, the process of assembling low level graphics primitives into higher level explicit descriptions of architectural physical elements is an important part of the CADD design process. Most CADD systems provide in some way to allow the combination of geometric primitives into various types of groups. Operations may be performed on these groups by treating them as single entities. The elements of the group share some kind of property, and the relationship between them nay be graphical or non-graphical (such as attributes). The names for these groups vary with different systems, which call them objects, parts, elements, blocks, symbols, or simply shapes.
One of the most common arguments in favor of CADD is its ability to automate repetitive tasks; the use of predrawn shapes which can be recalled as a whole exemplifies this aspect of the CADD process. That is why the way a system handles symbols is considered essential for the effectiveness and productivity of the software.

One of the simplest examples of parts is the libraries of symbols, developed in user organizations, or supplied by third party vendors as part of front end templates. Symbols in this context refer to two things:

- A set of architectural pictorial symbols such as section markers, HVAC symbols, or trees and people.

- A catalogue of windows, doors, and furniture of standard sizes which can be inserted in a drawing.

The first type of symbols are used mostly for production and presentation drawings and amount to little more than "electronic Letrasets" of icons.The second type of symbols is analogous to manufacturer's catalogues of windows and furniture.

At some point, though, the user will need to create his own libraries of parts, to suit his working methods. Systems usually allow the combination of graphic primitives into other primitives , which are filed and recalled as single entities.

Autocad calls its elements blocks or write blocks. They are generated by selecting a group of entities according to whichever criteria the user needs. These blocks are filed with a name and stored for future reuse; when needed they are retrieved and inserted in another drawing.

Microcad also provides for the selection of a group of entities which may be filed as a whole. The command used for the creation of the element is "File Part." When reinserted, the element may be used in two ways:

- It may be inserted as a "symbol" (using the "Insert Symbol command), meaning that it will behave as one single entity; an operation performed on one of its entities affects the whole symbol.
- It may be inserted as a "part" (using the "Insert Part"

command) meaning that it remains a group of separate entities; the primitives which form the part may be individually edited. This differentiation is similar to the "block" versus "*block" in Autocad, where "block" refer to one entity, and "*block" to separate ones.

Both options have their advantages and disadvantages. A symbol treated as one entity is easier to manipulate when inserted. It also uses less memory than a part which retains its separate entities; for example, a room with ten identical chairs will use less memory if every chair is treated as a single entity. But if every chair is different,then there are no savings in memory, because all the information associated with the file of each symbol is inserted in the drawing as well.

On the other hand, symbols are difficult to reference since their only object snap point is the insertion point of the symbol. Parts, since they retain their different entities, may use the end points or origins of any of these entities when the user wants to reference existing geometry.

An example of this is the arched window elements shown in fig (3). While symbols were easy to manipulate as a whole, the difficulty arose when I wanted to create a double window element from two existing ones. If one window is inserted as a symbol, and the user wants to insert another one aligned with the first one, then the dimensions of the window have to be known in advance and explicit coordinates used for the second window to coincide and align. This is easier to do by inserting the element as a part in which the origins and endpoints of individual entities are referenced for correct alignment.

An alternative provided by the system is to insert the element as a symbol and manipulate it as such. When there is a need to use the individual entities, the symbol is broken down using the "Explode" command, which will divide it into separate components which may be referenced. However, it is not possible to reassemble exploded symbols into their original form, which is often desirable.

Another useful feature of parts and symbols is the ability to scale, stretch or rotate any part when inserting it. This allows for slight modification of the part being inserted without having to redefine it. Parametric variations, for example, a common exercise in CADD, consists in creating different versions of an architectural element through the scaling and rotating of its geometric parts. The corbels, for instance, were created by inserting the half lintels with a 90 degree rotation around the Z axis to make them perpendicular to the wall. Similarly, different sizes of openings were created by verticular and horizontal scaling of the jambs and sills, which were also filed as parts.

The capacity to edit parts, redefine them and file them as they are being used is always a good option, allowing more versatility in the manipulation of parts.

It is often desirable to create symbols that use other symbols, or "nested symbols." Nested symbols contain other

symbol references; they make the geometric modelling process an additive process with a hierarchical structure of elements. These elements can be combined and nested at many levels to generate larger and more complex objects in a "building block" fashion. There may or may not be a limit to the nesting complexity of symbols. The arched window elements use two levels of nesting, where one element is comprised of sills, jambs, and lintels which are themselves filed as parts.
Elements in CADD are stored in vector form, so that small complex objects use less memory than a large simple one. As a result, the physical limits in memory of a symbol are a function of its complexity rather than its dimensions.

Another clear advantage of using parts is that in a drawing which uses one symbol a large number of times, it is much easier to update the symbol once rather than modify each occurance. The system automatically updates the drawing by finding the new version of the symbol. This allows to quickly generate and store alternative design sketches by only modifying the basic part. For example, a few elevation studies may be created by only changing the basic opening. Incidentally, such a use of the computer illustrates a case where its ability to accomodate "grudgework" proves useful for the design process.

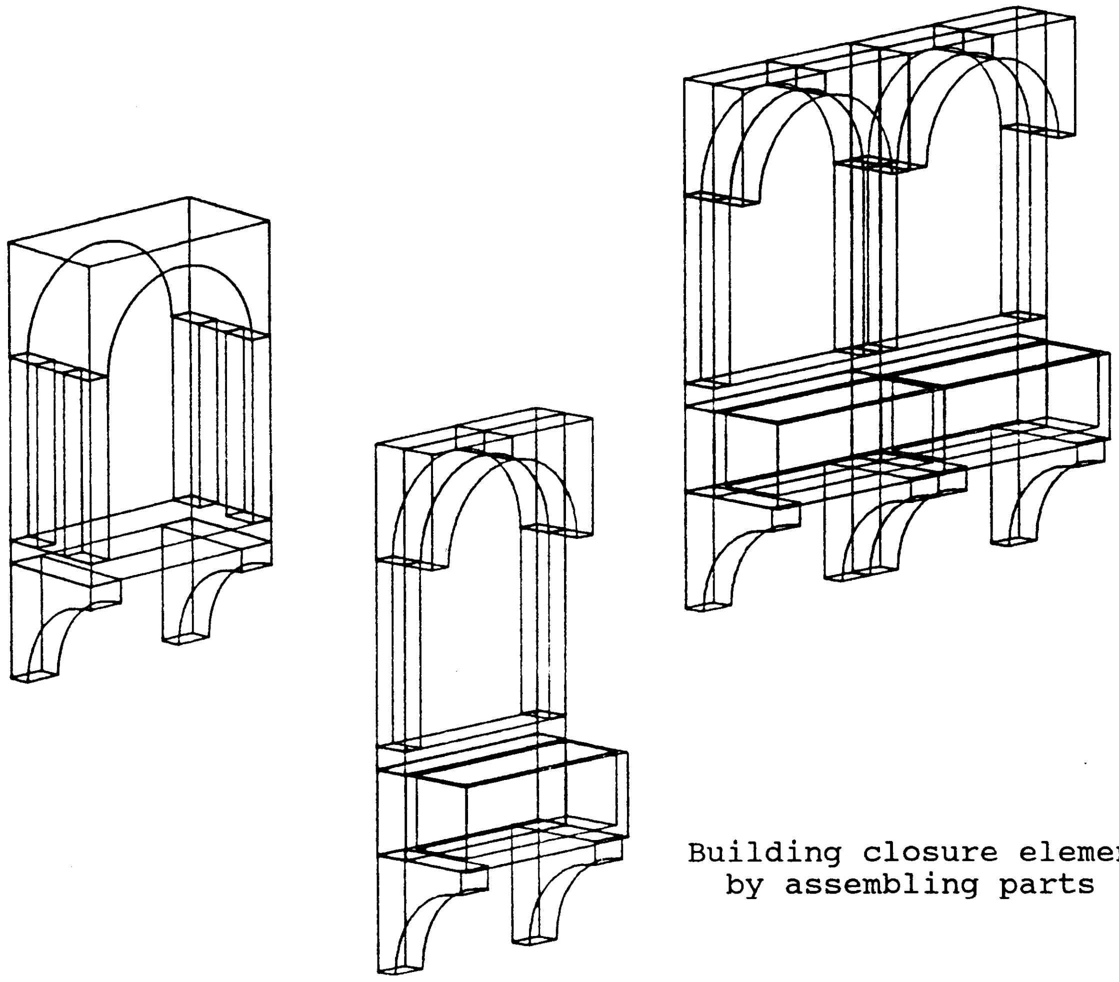
The drag mode, which allows dynamic displacement of the symbol as it is being inserted, is of great help when positioning the symbol. It is a good example of tight interactivity in the lexical output of the system.

Keops, on the other hand, does not allow the filing of part of a project and its insertion in another one. This limitation has two consequences:

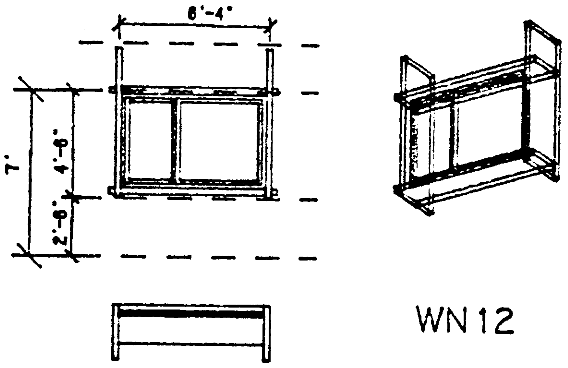
- Since it is impossible to store and retrieve one version of a model, every different element of that model has to be created within the project. In this case, for a cluster of three different units, every unit has to be modified in the cluster, as opposed to creating each one and inserting it in the cluster.
- While the system provides for some good symbol manipulation and relocation through mirrorring and rotation, symbols cannot be scaled or stretched as they are inserted. They have to be modified separately in the Drafting module, and processed through the tedious transferring program before they can be reinserted. This method completely lacks interactivity. As a result, Microcad becomes a more appropriate tool for elevation studies, as it permits to finalize opening dimensions before filing and transferring them.

In short, the efficiency of a system in handling symbol creation, insertion and manipulation is an important factor of the CADD process and constitutes a valid criterion in its assessment.

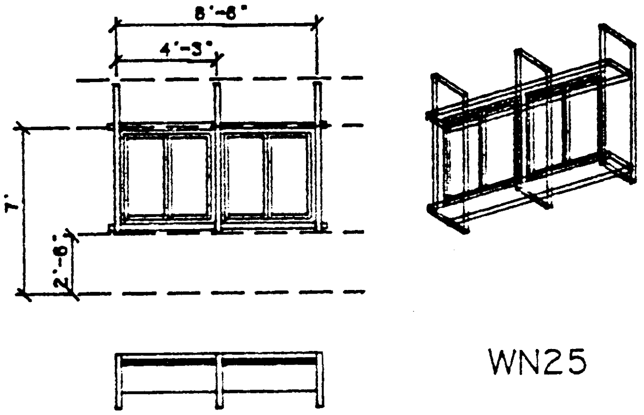
Just as it is helpful to group entities into geometric blocks and parts, the ability to assemble these blocks together according to nongeometrical criteria and define them as part of a group can be very handy. One can assign nongraphical according to nongeometrical criteria and define them as part of a group can be very handy. One can assign nongraphical properties, called attributes, to groups of objects. When attributes (labelled properties in Microcad) are attached to an object, it becomes possible to have classes of objects which share attribute values. Operations can then be performed on these classes of objects as a whole. It is often useful to refer to a component as part of a larger organizational system. For example, an object representing a wall panel may belong to a group called "infill", or to another called "prefabricated elements".


Building closure elements by assembling parts

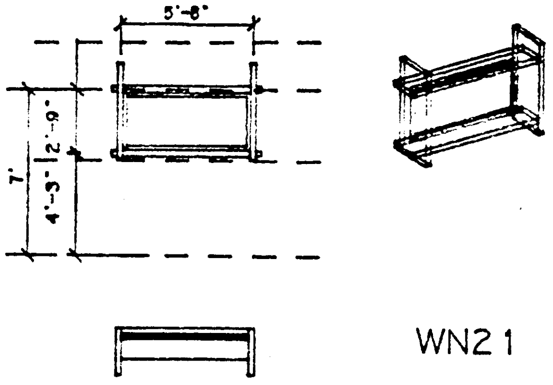
Developing a catalogue of parts



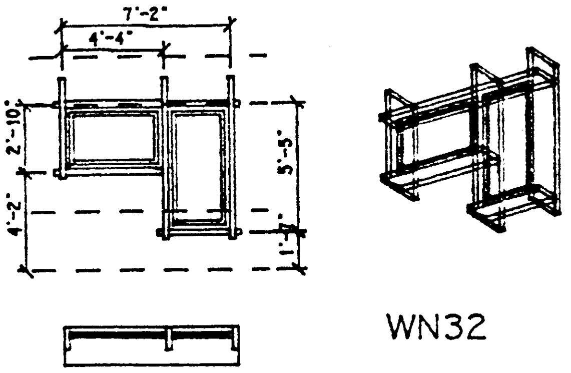
WN12



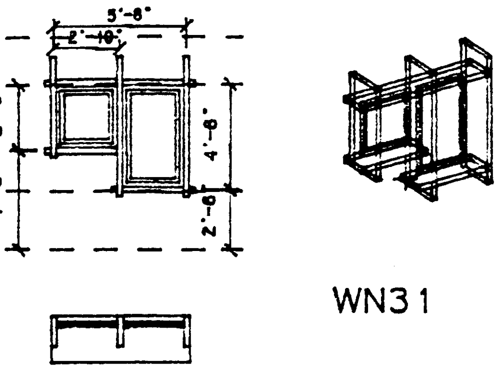
WN25



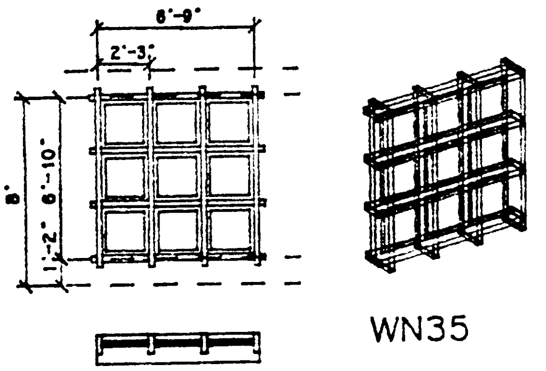
WN21



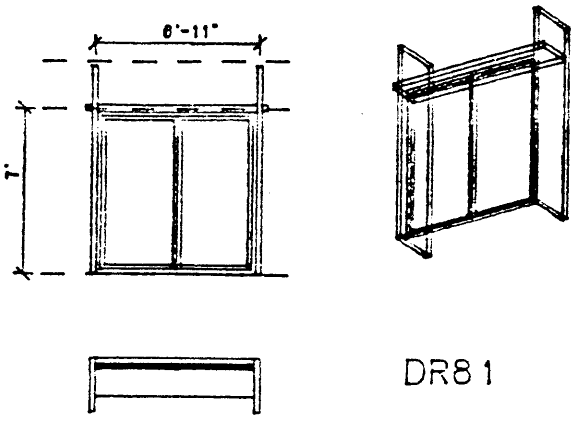
WN32



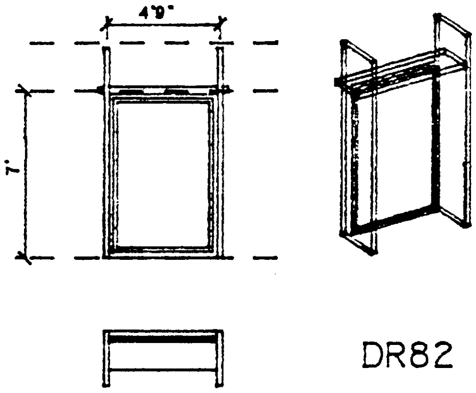
WN3 1



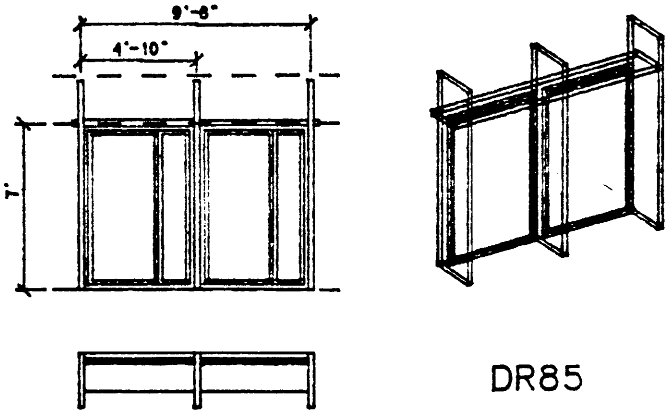
WN35



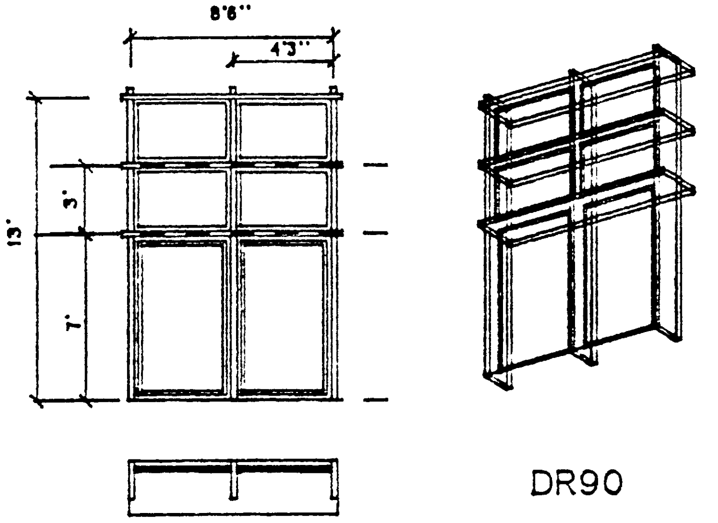
DR8 1



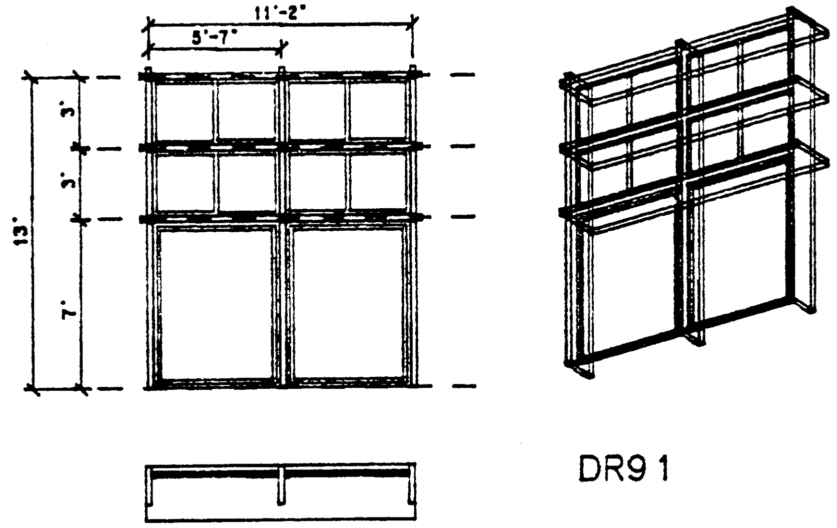
DR82



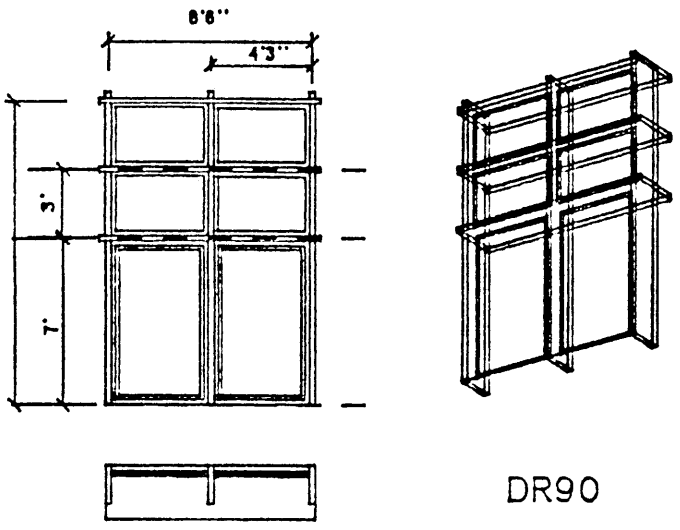
DR85



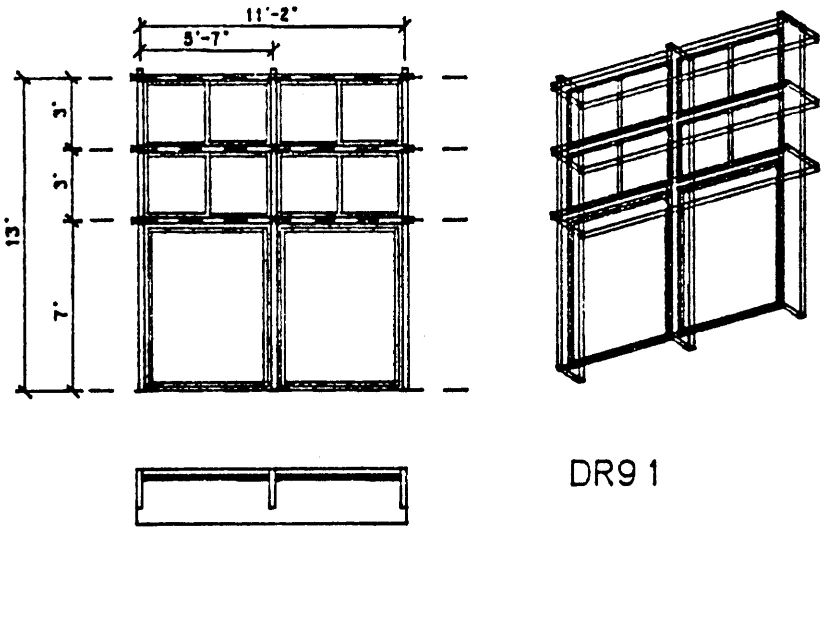
DR90



DR9 1



DR90



DR91

LAYERS

Layers allow the user to selectively organize and display information. They are similar to transparent overlays, which can be viewed on top of each other or separately.

It is ironic that the most common analogy used to describe layers is that of a tracing paper, when in reality the uses of layers are completely different from those of tracing paper. Layers are used to differentiate between classes of information related to one project. For instance, one layer may describe the structure, while another may include infill, or another yet dimensions or HVAC symbols. By selectively turning some layers on and off, the user may view and plot each one individually. Obviously, this use is very different from that of tracing paper. Perhaps a better analogy for layers would be a complete set of production drawings, including architectural, structural, mechanical and electrical drawings.

The use of tracing paper, on the other hand, is very intuitive one and may vary with different architects for preliminary design. Its primary quality is related to the fact that it is neither opaque nor transparent, but translucent. When a designer sketches an initial alternative and lays another sheet on top of it, the trace partially conveys the graphical information in an indicative way. His next iteration through the following sketch alternative will use the bottom trace as a guide, modifying some lines and retracing some others. The top trace "filters" the graphics of the bottom one, which acts as a "locational constraint", similar to a snap or a grid but less constraining. While a snap allows the crosshair to rest on specific legal points only, the trace is suggesting for the pencil to rest on points "in the vicinity" of such point or line.

Ideally, if layers were to be used as tracing paper, they would require its characteristics. The idea of a translucent layer is an appealing one. Another one would be to allow the geometrical entities on one layer to act as a "partial constraint", allowing the cursor to rest on points within some distance of the bottom ones. As we know it, though, it is difficult to input to a computer many-valued, non-discrete qualities such as "nearness" or "sidedness".

Another important characteristic of layers is their ability to allow the user to selectively discriminate what and how much information he wishes to view. There are two problems if a drawing on the screen has too much information: it loses its legibility, and the screen is very slow to regenerate. The issue of legibility is solved by turning off the unnecessary layers. As for screen regeneration, turning the layers off will only slightly decrease the search time, but not by very much. What will really shorten the repaint time, though, is the capability to "freeze" layers: when a layer is frozen, it is considered as nonexistant and does not slow the system processing time.

Autocad allows for an unfinite number of layers. These layers may also be named rather than numbered; it is much easier to understand and remember a layer labeled "Electrical" rather than "Layer 16". The system restricts, however, a layer to one color and one linetype. Layers, colors and linetypes are therefore completely related.

Microcad allows for 256 layers, and has no restrictions on the number of colors and linetypes per layer. Layers can only be numbered, and they cannot be frozen.

Keops,on the other hand, must be one of the very few CADD systems which does not allow layers. The system makes its own decisions about organizing the information. For example, one command will display the volumes, another will display dimensions, or furniture, text, etc. The user has no control whatsoever on how to structure all this information according to his own criterias. This leads, in turn, to problems in legibility particularly when viewing the wireframe in three dimensions. It is a very authoritative assumption by the system on behalf of the user.

GEOMETRIC MODELLING

Designers solve their problems visually. In no other design discipline is the end product so intimately related to its visual representation. It is no surprise, therefore, that an important aspects of evaluation criteria for designing with CADD involves its potential for geometric modelling.

A model is by no means a finished product; it is an approximation of that finished product, a physical prototype of the specific state of what a temporary design solution might look like. Once this prototype has been simulated, it is visualized, analyzed, evaluated, and modified to explore an alternative potential solution, before anything physically tangible is created.

The model is useful for communicating the design solution to other participants in the design process; these participants may analyze it and evaluate it according to different criteria. A model may be used for varied kinds of analysis such as structural analysis, energy calculations, daylight simulations, or cost estimates.

While these different forms of analysis may be performed at varied phases of the design process, we are concerned here with the use of geometric modelling in the early stages of sketch design, for functional and visual assessment. While many techniques of visual assessment are only initiated in the later stages of design, there is a definite need for the quick generation of models in the early conceptual phases, where most major visual design decisions are taken [Bridges 83].

2D V/S 3D SYSTEMS

During conceptual design, architects not only think visually, they also think in three dimensions. It is during these early stages that the processes of form making and visual appraisal dominate other kinds of assessment. This is illustrated by the extensive use of physical scaled down models (i.e. cardboard and others) in both academic studio environments and professional organizations. It should be no surprise, therefore, that three-dimensional geometric modelling ability is widely acknowledged to be an important and popular feature of current CADD systems.

Two dimensional systems, in general, are mostly production oriented and primarily aimed at efficiently creating architectural drawings, with little or no intelligence concerning the geometric characteristics of the drawing. This does not necessarily imply that 2D systems cannot be used for visual description of the building. They can be used, for example, to draw different isometric views of a building, without any definition of 3D coordinates. These views, however, are not related to each other; they do not share any information about the building representation. Similarly, some 2D systems allows the creation of automatic perspective views, through the projection of the plan on a picture plane, in a way similar to how architects manually construct their perspective views.

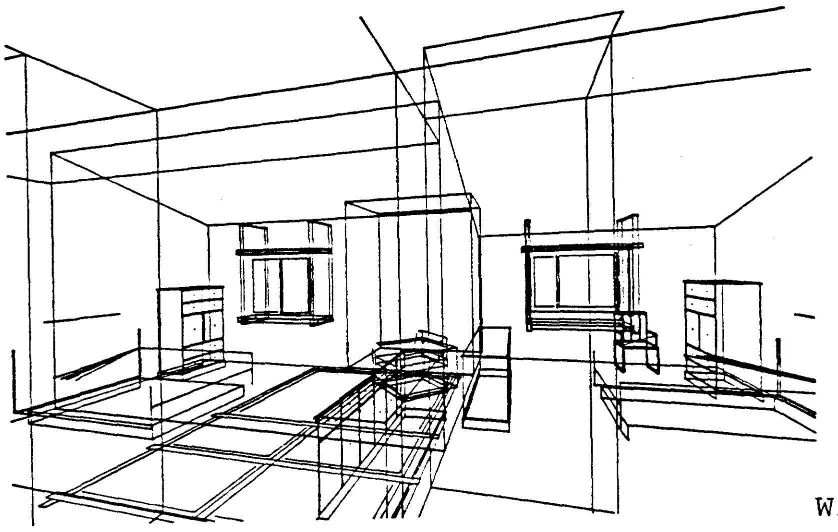
The next step up in geometric modeling involves systems which understand the Z axis. A three dimensional object may be generated by drawing a planar figure and extruding it along the Z axis. AutoCad is an example of such a system; objects may have a thickness and an elevation. These systems are referred to as "2D 1/2" or axonometric extrusion, since they do not have full 3D capability. Objects require a constant cross-section along the Z axis, and remain parallel to the XY plane. This precludes inclined planes such as sloped roofs, or primitive shapes such as spheres and domes.

"True" three dimensional systems, on the other hand, are characterized by the creation of a 3D "electronic" graphic database. All elements of construction have 3D definitions. Any line has end points with X, Y, and Z coordinates. Entities can be extruded or rotated along any of the three axes. As a result virtually any three-dimensional object may be generated and visualized from any point in space, with no limitations on its shape.
Perhaps the most important characteristic of a 3D system is the existence of one central electronic model database. All views, whether they are planar (plans or elevations), isometrics, or perspectives, are extracted from this single 3D model, as opposed to a set of nominally related 2D views; they

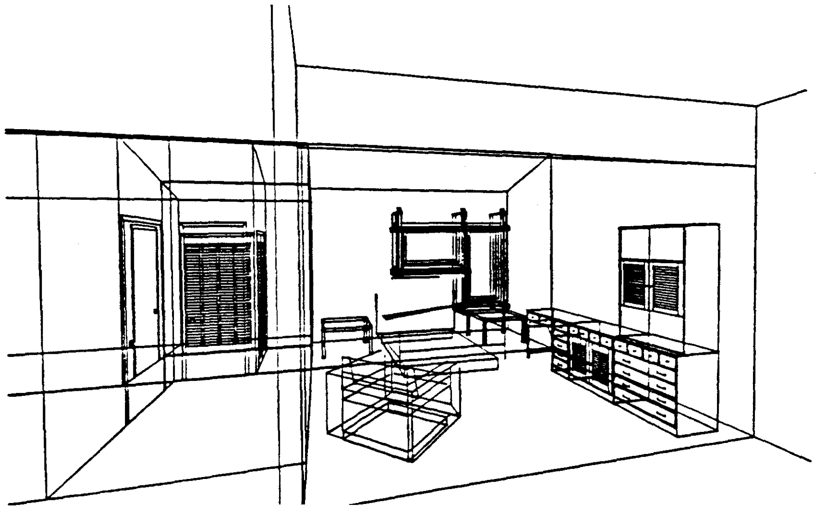
are said to be "model based" instead plan/elevation based. Whenever the model is modified or edited, all views are automatically updated. 3D systems allows the user to view the model at scale from any point in space and examine its interaction wiht other geometric elements.

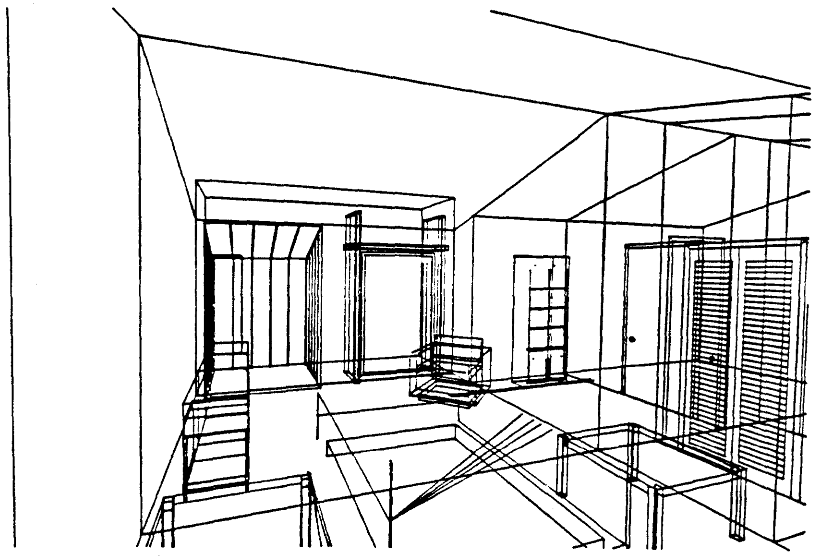
WIREFRAMES

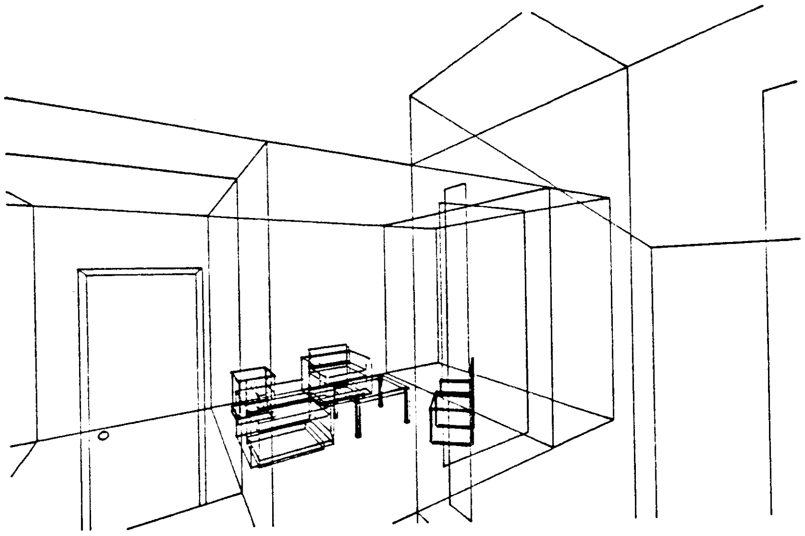
In most CAD systems the process of geometric modelling and its visualization starts with the construction of the wireframe model. "Wireframe" refers to a model in which all entities are visible from any point; when viewed the model looks like a sketch outline of an object, similar to an airplane model or a bird cage, where all points, lines, and vertices are visible, including the ones that would be obstructed by other planes from a given point of view. Viewing a wireframe model is similar to looking through the model; all the components of the model, internal as well as external, are simultaneously displayed. This allows for some interesting inside/outside viewing mechanisms which are not possible with physical models. While they can provide very valuable visualization and simulation tools for some situations, particularly for localized parts of a building, they remain, however, indicative and lack solidity.

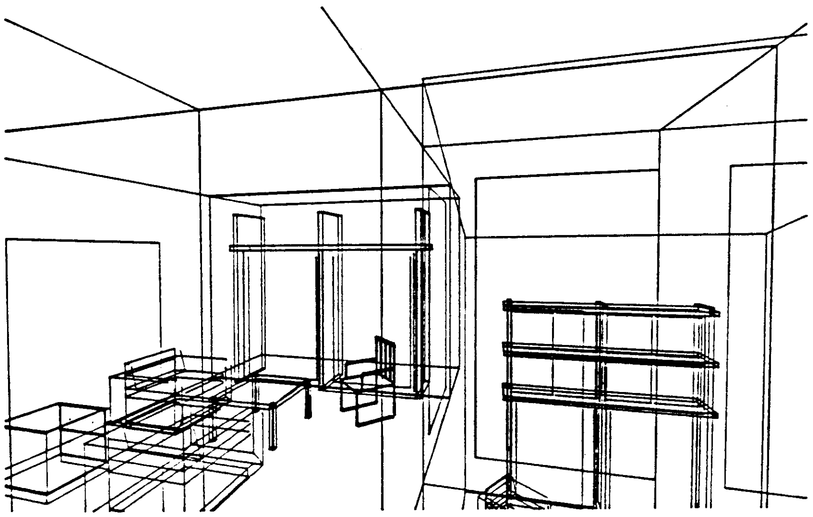


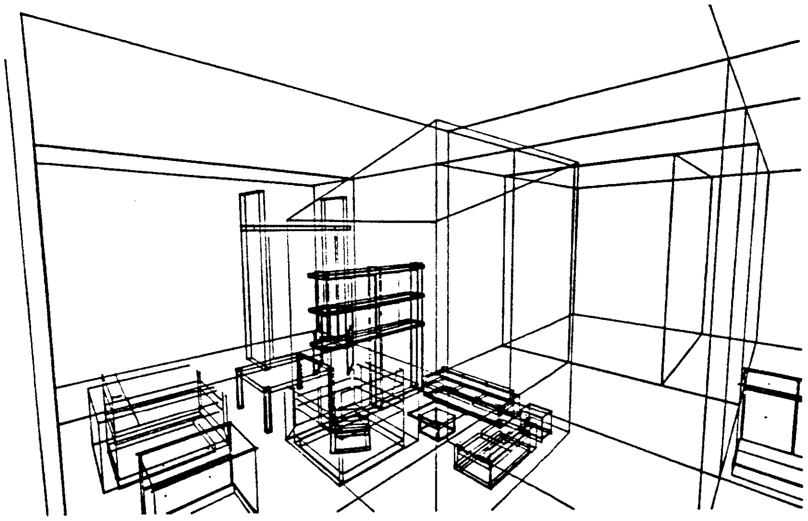
Wireframe views from within the model

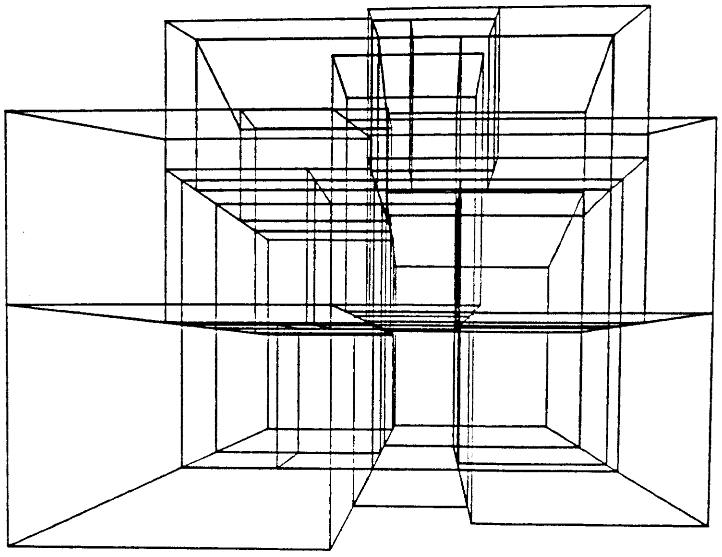
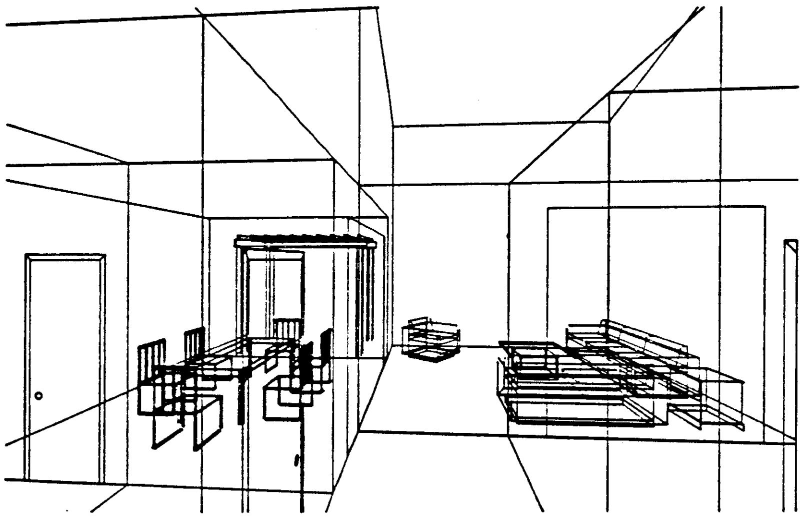
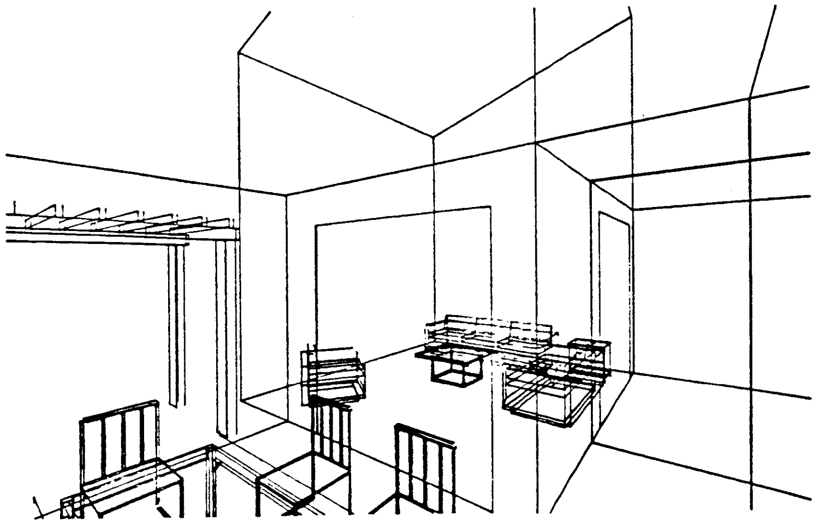










HIDDEN LINE REMOVAL

The removal of hidden lines on three dimensional wireframes is a subject of much research in the field of computer graphics. This technique is fairly standard on minicomputers and mainframes. Few PC based systems offer it, though, partly because it is a both a complex and computationally intensive process,particularly for PC's. Two methods prevail for basic hidden line removal [Witte 84]:

Surface Orientation This method imposes some restrictions on the construction of the model for a given view. For instance the drawing might need to be created in a counter-clockwise direction. Overlapping objects and planes are not treated. It necessitates therefore a certain amount of data preparation and input. On the other hand, the processing time required for calculating the resulting image is relatively short.

Surface Priority This method has less limitations on data input; it sorts objects, planes and polygons by distance to the observer and checks for intersections. As a result, the time required for image generation is considerably longer. It is not unusual to see systems that will take up to thirty minutes for a simple model.

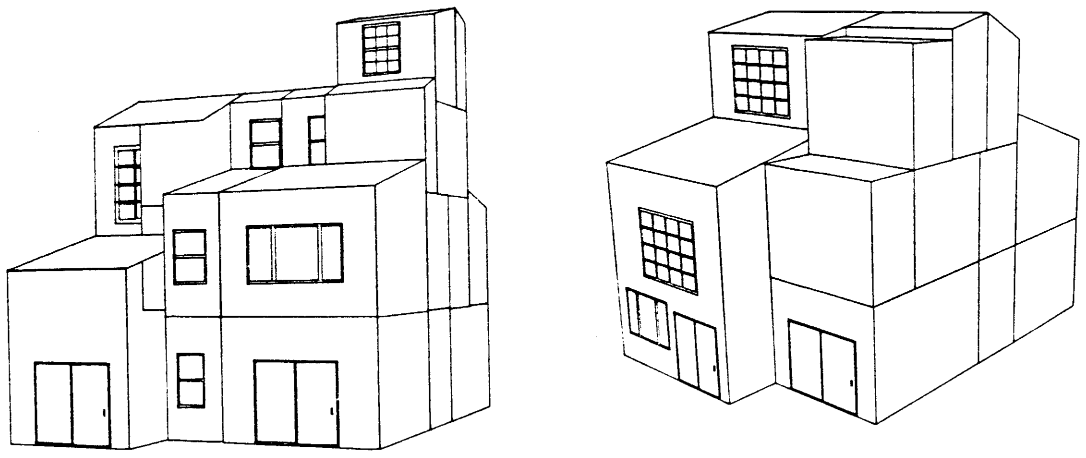
Given the complexity of both processes, many users argue that in most cases the removal of hidden lines is best done manually. One way to do this is by locating them and editing them on an image file. Another common way involves using large digitizers and a stylus to trace the visible lines from the hard plot of the wireframe view onto the screen .

ELECTRONIC V/S PHYSICAL MODELS

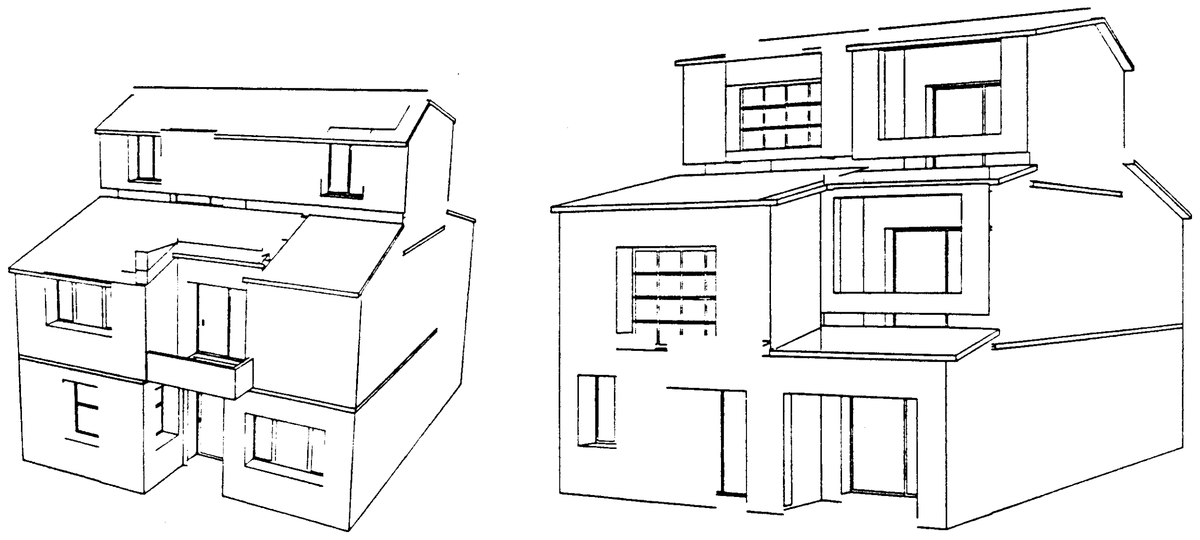
The process of three dimensional modelling involves a frequent succession of geometry construction and model visualisation. While the wireframe model remains independent from the viewing position, a hidden line removed image is directly connected to one particular view. Therefore, the quick visualisation part of the modeling process relies more on the wireframe views than on static finished images. In fact, hidden lines images are typically more used at the later phases of visual assessment or presentation drawings, as opposed to wireframes views which are a dynamic tool for the modeling process of early design sketches.

There is an inherent analogy between electronic models created in CADD systems and the physical scaled models which are commonly used in architectural education and practice. Wireframe views may be compared to what is referred to as "working models" in preliminary design. These models are usually partial descriptions of parts of the building in terms of its constructional elements, such as structure and infill. Hidden line images, on the other hand, are analogous to the presentation models which focus on the building envelope and are used for client presentation.

Obviously electronic models cannot compete with the dynamic viewing of a physical model and its "permanent state" of hidden lines removal (which is what animated "fly-through"



Hidden line removal before 3D database



Hidden line removal after 3D database

simulations try to achieve). One advantage, however of the electronic model is the ability to quickly generate design alternatives which can be stored for later comparison, and its potential as a tool for documenting the design process.

WIREFRAME LEGIBILITY

One of the arguments for CADD systems is the absence of a scale factor when creating a drawing: all dimensions are entered in real world coordinates. As a result a single model may include various levels of information, ranging in scales and sizes from a or a baseboard to a precast wall panel. A view of the wireframe showing all windows and doors of a building will look very different from one viewed from the same point, which would be limited to a volumetric description of major planes and masses.
For that reason wireframes may vary a lot in how well they can convey visual information. If the model has too many lines, or too many levels of information simultaneously displayed, visual clutter will result in poor legibility. Typically a wireframe image will be too dense in the center areas and most legible at its edges. If on the other hand, the model view is restricted to one type of information, or if the user can effectively control how much information he wishes to display, then the view of the wireframe remains legible. In other terms the ability for the user to discriminate the amount of information is a critical factor for the legibility of the wireframe and its use as a visualisation tool.

The issue of legibility of a wireframe is particularly relevant for Keops because of the specific way it represents the building.

Since the smallest entity is a four sided volume, the most indivisible unit of the wireframe is a parallepiped with six facets: four walls, floor and roof. The four vertices define the planar boundaries of the volume. To create a L shaped space in plan, one would simply delete the side shared by the two volumes. When in plan mode, the appropriate command will actually display the L space without the deleted side. In 3D viewing mode, however, the vertices which correspond to the ends of the deleted side are not removed; they remain visible in the wireframe, despite the fact that they do not really exist. In other terms when looking at a three dimensional view of the model, the wireframe will always display all vertices of all volumes, regardless of their prior deletion. The relationship which exists in plan between spaces and volumes does not exist in three dimensions.

A wireframe, therefore, can only be legible up to a certain number of volumes, since all their vertices are always displayed in the wireframe. As a result the physical limitation for manipulating the model is more determined by the number of volumes rather than the number of spaces or rooms. In order to effectively remove the unnecessary vertices, one would run the Hidden Line Removal program, and the user ends up relying more on static hidden line images for his visualisation, instead of using the more flexible wireframe viewing mechanisms.

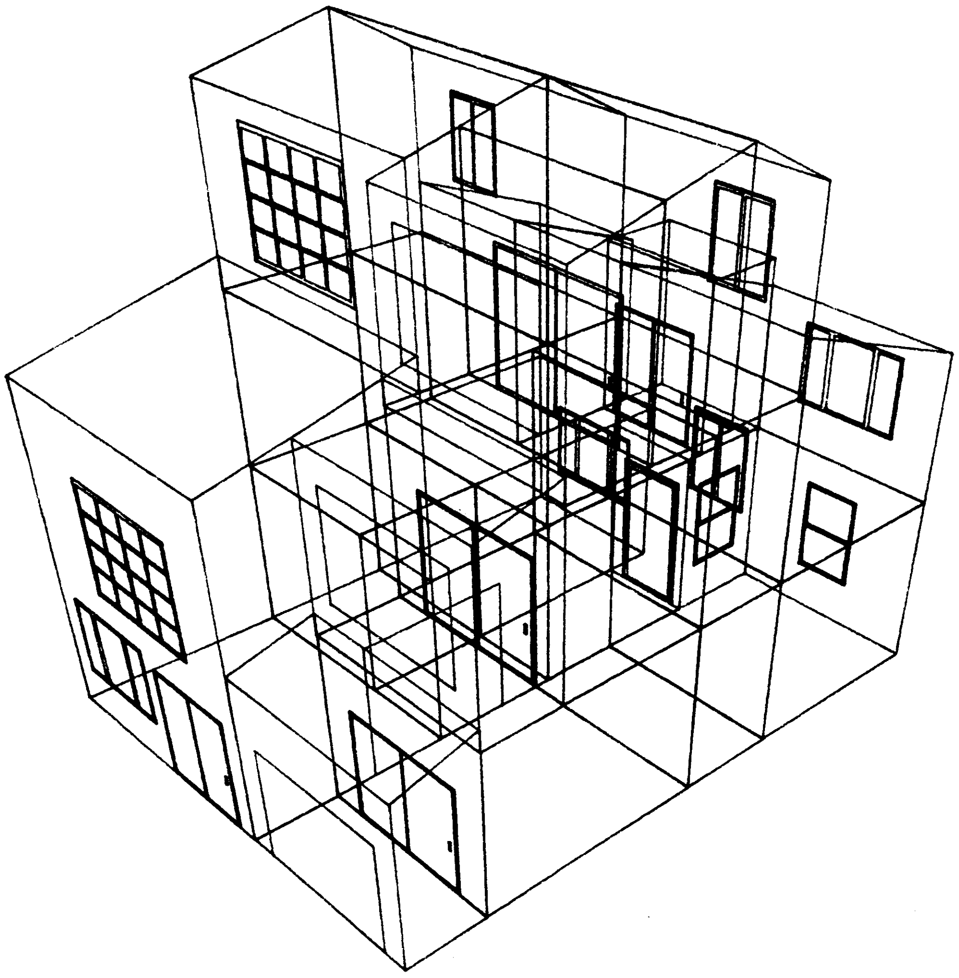
VISUALISATION

Visualizationrefers to the process involved in setting parameters for a particular view of the model. System vary in the way they allow the use to select his viewing mechanisms.
AutoCad uses an interesting solution. When in the elevation mode, a small compass is displayed on the upper right of the screen. The compass consists of two concentric circles; the model viewed being in the center. By moving the stylus around, the user can select his position relative to the model for the X and Y coordinates, and its height by getting closer or farther from the center. As the user does that, a X-Y-Z axes tripod is dynamically changing to represent the current position relative to the three axis origins. The user can therefore construct an unlimited number of axonometric views. It is an ingenous mechanism and relatively easy to use and control, though it is not very compatible and does not try to mimick the human process of an architect to view his building. In Microcad, the user selects a view by rotationg the model around its three axis. He inputs numerical values for the angles of rotation, until he has reached a satisfactory viewpoint; the view is then saved and assigned a number. It is the model which is rotated instead of the observer being displaced as in Autocad. In both Autocad and Microcad, the user cannot control the focus point; he is always looking at

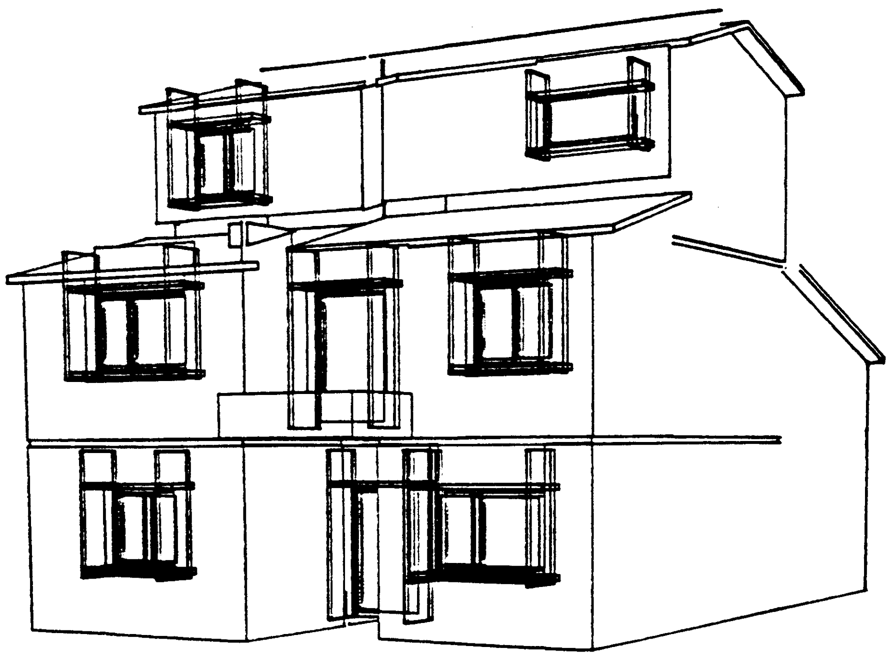
the center of the model.
Microcad also allows for perspective views, though the user has little control on its parameters; he just inputs an arbitrary number to determine the distance to the objects, thereby deciding on the amount of distortion.
In Keops, the viewing mechanism is the most complete and the closest to the natural process, as the user controls most of the view parameters. He starts by selecting the X-Y coordinates of the focus point, or the center of the cone of vision, on the drawing sheet; he also gives it an elevation. He then selects the observer position, also in terms of X-Y coordinates and height. The user therefore controls the degree of distortion in a three point perspective, the cone of vision, the distance between the observer and the object, and the pitch.

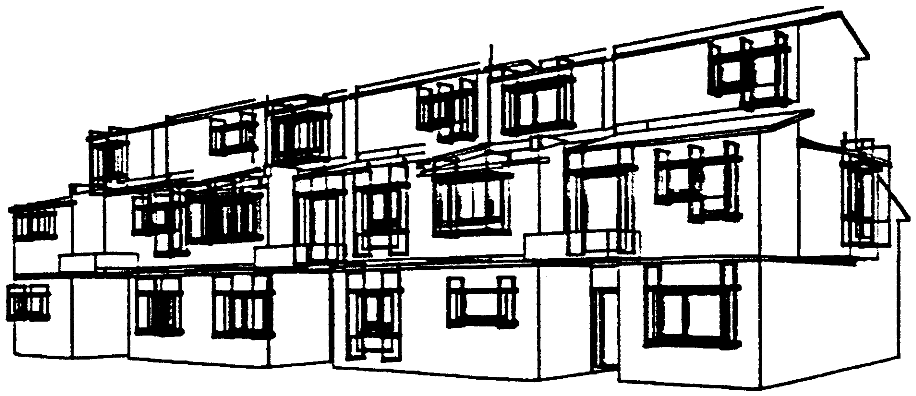
It is often important to save a specific viewing of a model instead of reinputting every parameter, where the view remains independent from the model modifications. In AutoCad, a specific view of the model can be saved as a "slide", an image which cannot be edited. Keops behaves the same way, and allows for the creation of images files containing all screen display; the image files can be saved and redisplayed. They are also non-editable.
In Microcad, on the other hand, the view parameters are angles of rotation of the model aroun the three axis; the resulting view, which is saved and numbered, is not related to the

model. The user can therefore save a specific viewing position and edit the model as seen from that position.

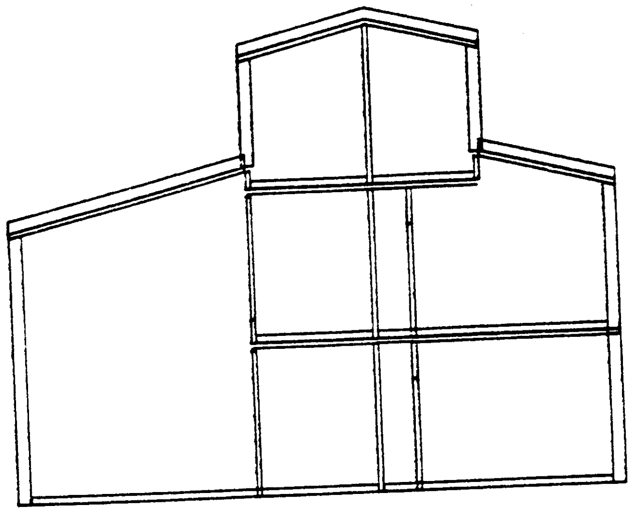


Visual clutter in the center area
of a wireframe

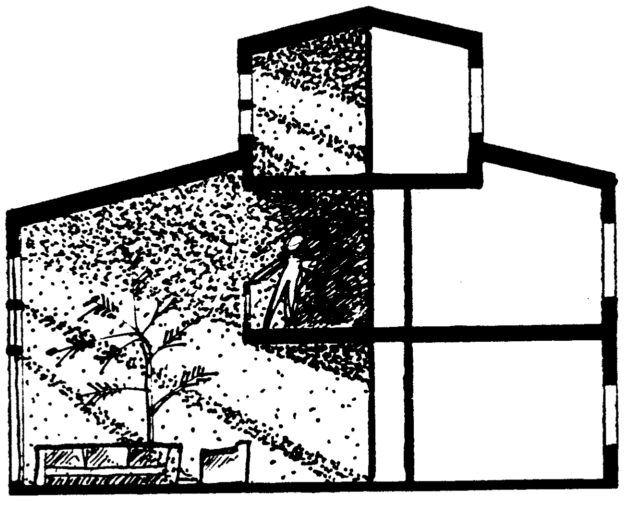




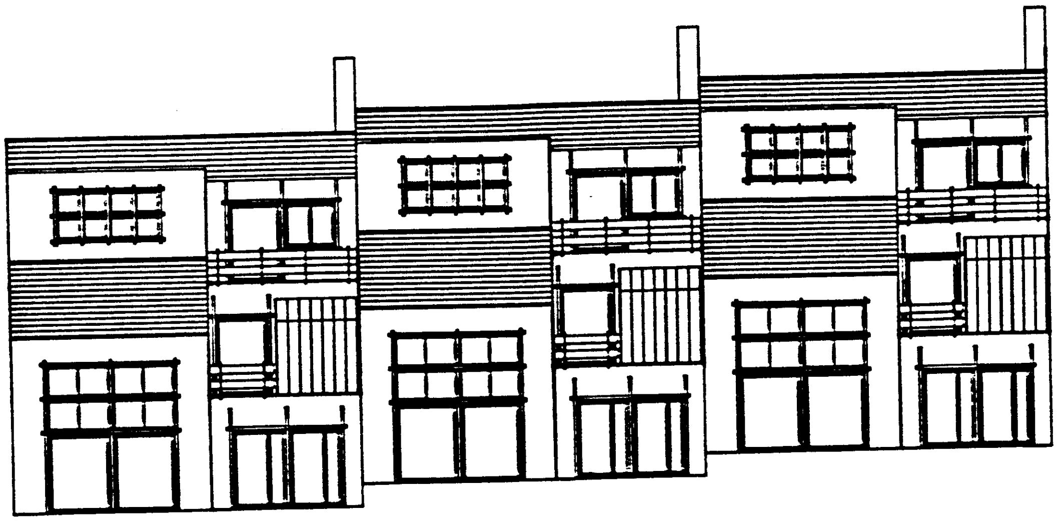
North perspective

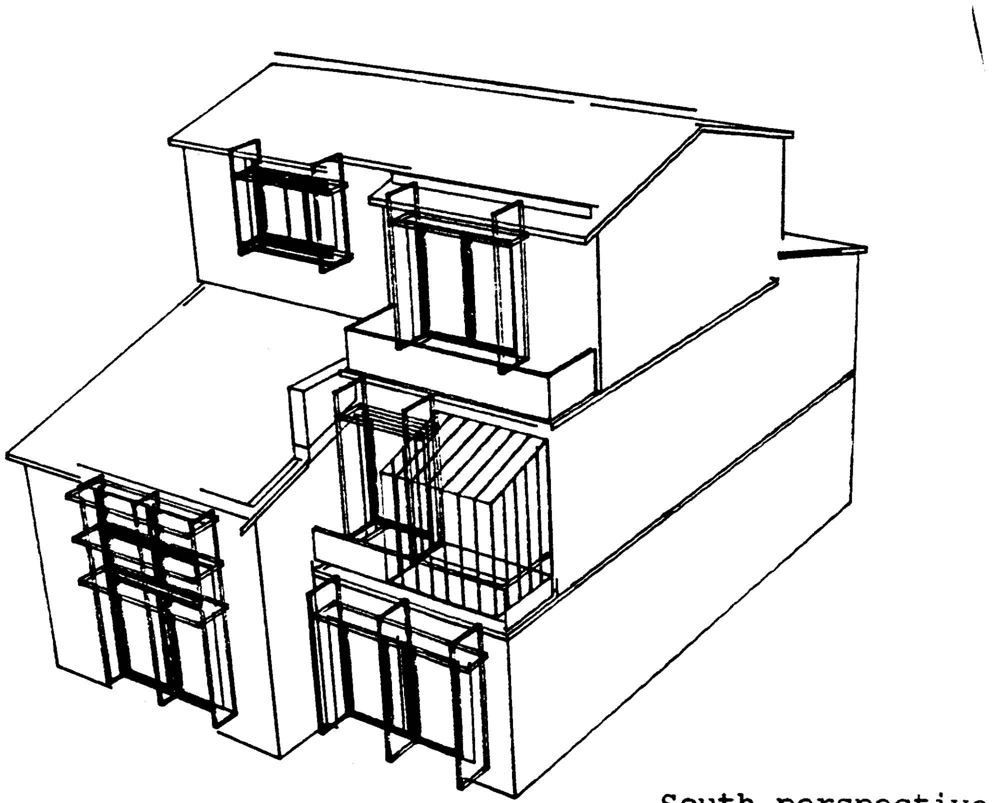


Automatic section cutaway



Manual section enhancement


South elevation


South perspective

System Interface

The quality of the interface may not seem to be a criterion for a design evaluation at first. There are, however, many reasons for the importance of user friendliness in CAD systems:

- The architect's traditional suspicion of alternative tools for his trade will discourage him from experimenting. He is more likely to demand a "better" interface sytem, one which will quickly alleviate his fears and feel natural to use.
- The performance of a CADD system depends on the user's skill and his ability to make the most of his system. That, in turn, is a function of the ease of use of the system and its "friendliness."
- The fact that the design process is ill-defined, and the need for many design alternatives, underscores the necessity for highly interactive systems.

Some systems are cryptic, require memorization and are alien to our design process; others are self-explanatory and responsive to our needs. As mentioned earlier, the complexity of the architectural design process makes it very difficult to establish a simple model for the interface to emulate. The connection between the pencil, the paper, and the brain is and will remain much more direct than that of a digitizer, a screen, and a brain.

The term interface loosely covers all instances in which the user interacts with the system. This includes input from the user to the system through physical devices, command sequences and syntax, and output from the system to the user through the screen display. The user friendliness of an interface refers to the ability of the system to bridge the gap between man and machine.

THE USER-COMPUTER DIALOGUE

The concept "man-machine dialogue" is often used to describe the interaction between the user and the system. The sequence of input-output commands is analogous to a conversation, where the user's language consists of words and commands, and the machine's language consists of images and pictures.

The analogy is a useful one because it allows us to establish natural language as a valid model for the design of an interface, calling for some of the same desirable attributes as a person-to-person conversation, such as consistency, flexibility, and simplicity.

When the user interacts with the system there are really two languages being used; one from the man to the machines, defined as input, and the other from the machine to the man, defined as output. Both of these languages occur at three levels [Foley 82].

LEVELS OF CONVERSATIONS

The semantic level of a language involves functionality; the meaning implied in a command and the amount of information needed to execute it. Examples of semantic input are commands which would be used to draw an entity, such as ADD, INSERT, CREATE, or DRAW. Examples of semantic output are prompting for additional information, displaying current status parameters, or displaying error messages.

The syntactic level of a languages specifies the rules by which sequences of actions are formed. It has to do with form, not meaning. For the user input, for instance, the syntax controls the combination of words into sentences and commands, very much as the grammar of a language controls the use of its vocabulary. The output syntax of a system includes the organization of the screen, such as positioning of prompts, error messages, and on-screen or tablet menus.

The lowest level of a language is lexical; it determines how primitives are entered or displayed to assemble sentences and commands. Lexical input deals with whatever physical input devices are available, such as keyboard, mouse or digitizer. The lexical level of output determines how geometric primitives such as line types, fonts, and colors combine to form symbols or visual codings.

FEEDBACK

Another important aspect of the interface involves feedback. In a natural conversation, one person's reaction to the other person's statement may be another statement, a facial expression, or a body gesture. This feedback is needed for the two persons to confirm what they are talking about, and acknowledge the understanding of their respective messages.

Similarly, in a CADD system feedback is an important component of interaction. Feedback occurs similarly to a language, on all three levels:

- The lowest level of feedback is lexical; the crosshair displacement as the stylus is moved, or the echoing of characters as they are entered on the keyboard. Rubberbanding and dragging are two quintessential examples of tight, interactive lexical output, cherished by interface designers and end users alike.

Dragging was mentioned in the sections on parts. It is a technique which allows for dynamic tracking of an object on the screen as the pen is moved. It is very useful for the positioning of blocks and parts such as furniture, windows, or columns. A similar technique is dynamic rotation or scaling of the object being inserted.

In rubberbanding, after digitizing the first point of aline, the line on the screen continuously follows the second endpoint as the cursor moves, always displaying the status of the line before the user finally selects a second point. Both these techniques make heavy use of immediate dynamic feedback on the lexical level.

- Syntactic feedback indicates whether the structure of the command is grammatically correct. It informs the user that both his word and his message are well formed. For example, an object can be highlighted when selected , before the command is executed, or the system can prompt for additional input in the middle of a command.An example of syntactic feedback is the fact that Microcad checks word by word the command line, and beeps if it encounters an error, as opposed to checking the entire command at its end. This makes error recovery considerably quicker.

- Semantic feedback indicates explicit acceptance or rejection of a request. It may tell the user that the command has been understood, that it is being executed, or that it has been completed by prompting for another command.

MENUS

Most CADD systems now use some form of menus, whether they are screen menus or tablet menus. A menu can be described as an organized list of options and commands, which can be displayed either on a portion of the screen or on a rigid overlay which fits on the digitizer tablet. Selecting an option from the menu through an input device is similar to invoking a command. The options corresponding to commands are listed in either character strings and explicit messages or through symbols and icons. Often the set of options is too large to fit in one menu. The options are grouped in submenus with a hierarchical tree structure, where the "root" or main menu may contain different modes such as "draw" or "edit." The user moves up and down the tree structure in order to select the appropriate option.

COMMAND DRIVEN V/S MENU DRIVEN SYSTEMS

Generally speaking, a user command structure may fall within two categories, menu driven or command driven.

- User computer dialogues which use menus and submenus are described as "computer initiated". The user is always faced with a number of options from which he has to select one. The software is therefore easier for beginners to use, because the system permanently displays a set of alternatives and prompts instructing the user for input; learning is done by recognition of familiar options rather than by memorization. However, menu driven systems are also described as "authoritative" for the same reasons; the computer makes more assumptions for the user, and guides him through the software. This is particularly true for hierarchical tree-structured menus if there is a need to go through many levels of submenus before selecting a command. While very valuable for beginners, submenu selection may be tedious and get in the way of experienced users.

- On the other hand, "user initiated" dialogues refer to system interfaces where the user himself enters, without being presented with a set of options, a complete command sentence with the appropriate syntax. These systems are also called "command driven," or "language driven," becuase the selection of a command requires an initiative from the user and an understanding of the correct software syntax, with no indication of a choice between alternatives.

As a result, command driven systems are harder to learn, but not necessarily to use; they may frustrate new users because they imply the memorization of some of the vocabulary and its syntax. However, they are also more flexible and better suited for the experienced user, for whom the formulation of a complete command is less of an obstacle than the rigid structure of different levels of submenus.

Autocad is an example of a menu driven system. The right part of the display area is reserved for listing the current active menu options, as well as the root menu. The screen menu allows the user to focus the attention to the screen only. The user does not need to know ahead of time the command he wishes to use; the options are listed, he only has to select one. Using the screen menu, however, implies moving through different levels of submenus until the desired complete command is reached.
AutoCad, though, is sometimes described as both menu driven and command driven, because one may use the keyboard to input a command at any moment, thereby short-circuiting the different levels of menu selection.

The general structure of a command does not vary much and usually consists of verbs, nouns and modifiers.
The verb describes what operation the user is instructing the system to perform. For example, a system may use INSERT, ADD, DRAW, or CREATE to indicate a drawing mode.
The noun refers to the object on which the operation is to be

performed. In generic systems, nouns represent low level geometric entities such as LINE, ARC or CIRCLE. In higher level systems, they describe a building component or concept, such as WALL, OPENING or VOLUME.
The modifier specifies, when required, how the operation is to be performed by further defining it. For example, it might specify to draw a line tangent to a circle or at a certain angle from an existing one (TANTO or ANG).
The meaning of words of such a complete command represents the semantics, whereas the syntax determines how they may be combined together to formulate a correct sentence, understandable by the system.

Microcad, in comparison, best exemplifies a software that is extremely command driven. The words which constitute its vocabulary are globally similar to those of most generic CADD systems; the vocabulary deals with low level entities such as lines, points, and routine editing procedures such as copy, move, etc...
The grammar, however, which dictates the rules according to which these words may be assembled, though, would give nightmares to a student in Latin. The syntax of Microcad takes the concept of dialogue with the machine quite literally. It is more analogous to a written dialogue, and relies on a sophisticated and highly specific set of punctuations which control the conversational loop. Every colon, semicolon or period has a meaning at various parts of the command line.

The language of Keops resembles that of Microcad, with verbs, nouns and modifiers, without being dictated by punctuation. The verbs used remain the same and are mostly editing commands. It differs semantically, though, reflecting its purpose as a front end system. The nouns describe physical elements in high level terms; the user manipulates volumes, walls, roofs, floors, openings.
There is no puntuation controlling the way these verbs, nouns and modifiers assemble; every command is different and must be memorised as such. As an example, instead of having a command "Move entity" , we have "Move volume", "Move opening", "Move side", "Move symbol", etc... The result is a staggering amount of linear commands, with no hierarchical structure.

Paradoxically, an example of a very strong menu structure can be found in the technology file of Keops.In this case the user is driven through a rigid tree-structured set of options prompting him for constructional information and dimensional values.
Furthermore, in this case the menus occupy the whole screen area. As a result, the connection between the building components being defined and their graphic depictions is nonexistent; there is no visual association whatsoever between constructional elements and the model they are part of.

EASE OF LEARNING

The learning curve of a system describes the amount of time needed for the user to become productive, or to accomplish a minimum of tasks. Like other factors, it is difficult to quantify. There are generally two tendencies in the learning of a system, reflected once again in Autocad and Microcad. As mentioned earlier, menu driven systems such as Autocad are easier to learn in the initial period of acquaintance. They provide operational guidance of the user through the software. As a result Autocad beginner's learning curves climb quickly to a certain level.

By contrast, Microcad is slow and tedious to learn, as the beginner slowly understands the intricacies of its syntax and its punctuation. The learning curve is slow, particularly at first.

Very quickly though, because of the flexibility of the language driven Microcad syntax, the user makes a slow but steady increase of his abilities with the system. He progressively masters the in's and out's of his program . Meanwhile, the Autocad beginners reach a "plateau". After learning quickly to do some amount of work, they find that in order to move further into the program, a whole additional learning process is required, which may involve scripts, macro's or other sophisticated aspects of the system.

In short, the learning curve for Microcad - or for most command driven systems - is slow at first but steadily increasing; whereas in AutoCad and menu driven systems, it quickly attains a certain level which it difficult to bypass. Many users claim their preference for a learning curve that is constantly growing, not because of system is cryptic but because of its depth.

This difference underscores an important distinction to be kept in mind, that between ease of use and ease of learning. One does not necessarily imply the other. Ease of learning depends on a range of other factors such as on line help and tutorials; this may affect very little the day to day ease of operation of a proficient user.
This point is further illustrated in the command structure of Keops, whose difficulty (due to quantity and not complexity) was described above. Every single Keops command can be replaced with two or three letter mnemonics, typed on the keyboard. Obviously it is suggested that, at least for beginners, the explicit commands be selected from the tablet menu. Nevertheless, after a certain time, entering short mnemonics from the keyboard proves to be faster and easier than digitizing the menu. This is partly due to the fact that several commands can be entered in a row and stored in the keyboard buffer, freeing the user to await their execution. This is more convenient than digitizing a tablet slot and waiting for that command to be completed before another one is activated. Mnemonics are by no means easier to learn (they do not even correspond to any form of abbreviation...) but they are certainly easier and faster to use.

PHYSICAL INPUT DEVICES

The lexical level of input involves the use of physical devices for coordinates and command entry, ranging in compatibility from keyboards to touch sensitive screens, including tracker balls, thumbwheels, rotary knobs or joysticks. They vary in the extent to which they attempt to emulate the designer's drafting process. For reasons of cost, obsolesence and interactivity, most PC based now use keyboards, mice and digitizers for input.

The keyboard

Input devices are either discrete or continuous. The keyboard is a discrete device; it sends input information in discrete packs of data, with a clear beginning and an end. It is not very interactive, since the user repeatedly adjusts input. Keypads, and to a lesser extent button cursors, are other discrete device.

Though the keyboard seems to hardly be compatible and the least natural of devices, it is not likely to disappear as an essential part of any interface system. It is used for certain tasks within the larger CADD environment such as operating system functions, file manipulation, or commands to peripheral devices such as plotters, or film cameras.

In addition, some elements of the designer's thought process are alphanumerical as opposed to geometrical, and keyboards deals with them better than mice or digitizers. They include:

- Dimensioning, notes, or other alphanumeric data.
- Entering system parameters or status commands which remain active for a session (i.e. "toggles"): linetypes, pen thicknesses, color,...
- Naming and specifying: it is always preferable to have the option of naming a group or a class of objects rather than assigning numerical values to them. For example, it is better to name a layer "Structure" than to call it "Layer 11".

The mouse

The major other two input devices, mouse and digitizer, are referred to as continuous devices, because they allow for a range of input values on an arbitrary scale, continuously fed back to the user. Both are used to input two-dimensional movements on a work plane. They differ mostly in their use of relative versus absolute coordinates.

The mouse looks like a little box, slightly smaller than a pack of cigarettes. It fits in the palm of the hand and is rolled over the work plane, accordingly moving the cursor on the screen. When the cursor reaches the desired position, the user activates a switch by depressing a push button on top of the mouse, to inform the system that the cursor is at the selected position.

The mouse is a "relative" device, also called "unrestricted". It is not associated to an absolute range of values. When it is picked up and set on another location of the work surface, the cursor remains stationary; its x-y movement on the screen

corresponds to the movement of the mouse relative to itself, not to its location on the work surface. The cursor displacement is made with respect to the body axis of the mouse, as opposed to absolute screen coordinates.
One of the advantages of a relative device such as the mouse is that it does not require a specialized surface; it may be used on any plane, even on our laps when reclining on a chair. Another advantage is that if the mouse is accidentally picked up, the cursor on the screen remains fixed at its last position. The user does not have to relocate it in order to resume cursor movement from that position; he can start anywhere, on any surface.
On the other hand, orientation of the mouse with regard to its own body axis is a factor in the cursor displacement.If the "nose" of the mouse is pointed forwards, and the mouse is moved laterally, an oblique line will result. Therefore the angular orientation of the mouse is a variable which the user must control.

The mouse was very popular when introduced, and is a big step up in interface friendliness from rotary knobs, thumbwheels or joysticks, particularly for computer-illiterate users. Some users argue that the CADD drafting process should not even attempt to emulate pencil and paper, and that the mouse is the best device for input. As a relative device, however, it is not very good for drawing lines if no snap mode is active. The mouse works best as a pointer for the selection of commands or icons from on-screen menus, where orientation is not a factor.

The digitizer

A digitizer is a form of electronic drafting board. It consists of a stylus and a tablet. The stylus looks very much like a pen, and is connected to the tablet with a wire. The user controls cursor movement by moving the stylus on the tablet, and selects a position by slightly pressing the pen on the tablet, which activates a small switch at its tip.

The digitizer is a "restricted", because it produces values from a finite range of values. It is also called an "absolute" device and requires a specialized working surface. When the user selects a stylus position on the tablet, the cursor moves to a corresponding location on the screen. The stylus is indicating an absolute location, with X and Y coordinates from a specific origin. As a result, it may be used to trace existing drawings or sketches.

The digitizing pad is often covered with plastic or cardboard overlays which usually consist of two areas. One of them is used to input coordinates to the cursor, and another is reserved for tablet menu. The tablet menu is a set of commands divided over a grid. By digitizing one slot of the grid, the user activates a command as if it was typed from the keyboard. Larger digitizers are used for the inputting of large existing drawings. They usually substitute a puck instead to a stylus. A puck looks similar to a mouse (though it remains an absolute device). A fine crosshair is set within a small window on the puck, allowing for higher accuracy in positioning the cursor.

The puck usually has two or three buttons on top, one of which is used to activate a point location; the others allow for the quick selection of commands which are often used, such as a repaint. The buttons may also be user defined.
Digitizers are currently by and large the most common means of physical input devices. They feel the most natural to people accustomed to pens and pencils, and as such are considered the most "user friendly" of physical interfaces.

In general ,though, despite the rapid developments of ergonomics in the field, CADD system interfaces remain far from being natural, human based structures. Even mouse and tablet remain relatively contrived solutions to the problem of simulating a sketching process.

CUSTOMIZATION

Since we are working under the assumption that the architectural design process varies with individuals, then the flexibility of a CADD system to adapt to different working method is an important factor in its effectiveness as a tool. A system should lend itself to customisation on two levels: it should allow flexibility of approaches for a wide range of project types, and it should accomodate the needs of different users. A modular system with an open architecture will encourage modification and adaptation by different designners and for various tasks.
There are many ways that a system may provide for

customisation by the user, ranging in complexity and usefulness. Perhaps the simplest example of customisation is the ability to create one's own library of symbols, as discussed earlier. Other forms of customisation include:

- Editing tablet menus: Most systems provide for some means for the modification of part or all of the tablet menu. They vary in the extent of freedom they allow: Autocad allows for the complete recalibration of the tablet; the user may redefine the number and sizes of command slots on three areas of the tablet. Microcad just assigns a given area (with constant key sizes) which can be edited to activate one or several actions, using the command "Edit Key".

- Editing screen menus: Autocad's screen menus are stored in text files which can be modified or recreated using a word processor. This allows the user to generate his own custom tree-structured menus and subordinate submenus. These menus may be attached to a drawing or used by more than one person. One may also write his own custom help text. Systems with onscreen icon menus sometimes allow the redefinition of these icons.

- Script and execute files: Sometimes the user needs to activate more than one command at once. Autocad provides a way of using the text editor to write sequence of commands and actions, called script files. Microcad uses a command called "Select Journal On/Off" which records every command entered until turned off, and creates an "execute" file which is stored for future recall. Alternatively, execute files may be created using the text editor. Scripts and execute files are useful for repetitive tasks which are often routinely needed, such as setting up a grid, a snap mode, or the limits and borders of a drawing before starting a session. Some of these routines are simply executed while others may prompt the user for input.

- Macro's: The most complex form of customization involves the use of macro's. They are similar to short programs, written in some high level language provided with the software (such as Autolisp for Autocad).They perform sequences of actions which accept input variables, conditional statements, and other programming characteristics. While beginners may not recognize the usefulness of macro's at first, they are an invaluable tool for the experienced user who can customize them for specific purposes. Macro's are good at performing grudgework such as calculating and drafting a stair, lay out a column grid, draw parallel lines or cleaning up unnecessary lines at wall intersections. Microcad currently offers a language (called User Programming Language- UPL) with the Drafting module. UPL, though, in its present form is not very friendly (it is Fortran based)and not geared for architects. Other kinds of customization more related to presentation and production, include the creation of custom made line types, text fonts, hatching and other pattern fills.

IDENTIFYING LIMITATIONS

Although that may seem like a rather negative approach at first, an important aspect of evaluating a system involves identifying what kind of limitations it exhibits. Some of these limtations are physical, such as size or number of entities; these are quantifiable, and designated as "topological" limitations. Others are more qualitative in nature, and referred to as "conceptual" limitations. They will be discussed under those headings.

Another useful way to prioritize limitations can be according to their degree of inflexibility:

- Some operations simply cannot be performed by the system.
- Other operations are not designed for the system, but the user can find a strategy for getting around it.
- Other operations yet were not designed for the system; and the extra effort involved in overcoming the problem is not worth it.

TOPOLOGICAL LIMITS

Microcad has few limitations other than the size of the drawing. Keops, on the other hand, has a number of limitations and restrictions on the types of projects it can understand.

* The maximum number of volumes which can exist in a single project file is 500 volumes. We should keep in mind, though,

that these are volumes and not rooms; a single volume may vary in size from a closet to a whole floor of a building. The relationship between volumes and rooms, as it is discussed later, is what really determines the size of the project. In addition, one should bear in mind that if these numbers are the physical limits as provided by the manufacturer's, project files nearing this size tend to because slow and awkward to manipulate.

- A single volume cannot have more than 10 openings or wall symbols.
- The walls of a volume cannot be inclined, neither can floors; only roofs may be sloped.
- Volumes cannot overlap.

All the limitations mentioned above are real, physical, and cannot be overcomed in any way. The following ones have usually a way around them, usually with extra steps for the user.

- A single volume can only have four sides. This, however, is barely a limitation since the system itself provides for its solution; by deleting volume sides one can model a multisided volume.
- A volume cannot be triangular. The way around this problem as suggested by the authors is to have one volume side extremely small. This method works fine for the wireframe manipulation; it becomes unnecessarily cumbersome when working with the technology file and the three dimensional database.
- There can only be one plane for the roof slope of a volume.

In other words, a single space covered by a gable roof requires two volumes with roofs sloping in opposite directions. A hip roof, similarly, will require three quasi-triangular volumes.The relationship between volumes and roofs is further discussed in the next section. Generally speaking, modelling roofs of average complexity such as dormer or shed roofs involves an intricate process requiring unusual volumes layout and corner height modifications.

- The system does not understand arcs, circles or any kind of curves. This causes problems at two levels:

  - Volumes cannot have a curved side. The way to bypass this as suggested in the manual , is to build a series of adjacent volumes, the sides of which make up the small rectilinear segments of a curve. <FIG> Given the complexity of modelling the volume sides in a regular way, and the additional size of these volumes, I found that limitation definitely not worth the extra effort involved in bypassing it.

  - Objects which are modelled in Microcad and transferred cannot include curves, such as arched windows. If they do, these curves must be exploded before the transfer, into individual line segments. This results in a very slow repaint of the object, as the system displays one segment at a time. Positioning a toilet or circular table symbol, for instance, ends up being a painfully slow process.

The problem can be partially alleviated by a powerful Microcad function which allows the user to control chord tolerance (the number of segments into which an are can be divided) thereby reducing repaint time; the process nevertheless remains a tedious one.

CONCEPTUAL LIMITS

Some of the limitations of a system may be less quantifiable than the physical constraints such as the topological limits on the volumes. They are more difficult to detect, because they deal with a higher level of abstraction, and they are derived from the assumptions that the system makes for the user; they may be described as more intellectual. They are also limitations around which a way can be found; the question becomes whether they are worth the extra effort involved in bypassing the problem. A good example of this type of limitation in our case study would be the programmatic labelling of the spaces into one of four categories (heated, unheated, terrace, mechanical).

Generally speaking, there are two slightly different approaches involved when describing a building in terms of spaces versus boundaries:

- One approach would be to deal with objects which describe the edges of adjacent spaces. The designer defines the limits between different zones and is concerned primarily with the boundaries. The spatial qualities are derived from these boundaries; they are a consequence of the edge conditions.
- In the second approach, the designer defines the spaces

themselves; their edge conditions result directly from the descriptions of the spaces. The boundaries between two adjacent spaces are derived from the qualities of these spaces. This is the approach that Keops follows.

For example, let us suppose the user designates adjacent volumes A and B as being respectively "heated" and "unheated". By assigning a space category (which is more a value than a spatial quality) to these volumes, the user is instructing the system to accordingly define the boundary between them. The design rule formulated in the technology file stated that "all inside-outside walls will be 12" thick" and that "all window frames on that wall are flush with the inside wall". Thus the user, by defining the zone, lets the system extrapolate what the boundaries will be like. It interprets the user's definition of a space in order to generate the edge conditions resulting from the volume category. As a result, both space and edge are very strongly defined as being from one type or another.

However, there are many situations in design where it is preferred to decrease the definition of adjacent spaces. There may be a need for one or many spaces with intermediate qualities or categories, if a more gradual transition from one to the other is sought. A designer will want to think in more conceptual terms than "inside" and "outside". The concept of volume labelling may not seem at first to accomodate the range of qualities which may be desirable at a specific edge condition. Similarly, the edges of a space may be more than two-dimensional planes with a given thickness. A boundary may be defined with a combination of different architectural elements, some of which may be three-dimensional.

In the case of the Personal Architect, the potential solution to the problem resides within the interface between the two modules. The three-dimensional modelling of complex objects in Microcad, their transfer to Keops for insertion in the wireframe appears to be a critical factor for the assessment of the system performance in overcoming this conceptual limitation. These objects can represent the physical elements which, when combined and inserted in the volumes, will further define the edge conditions and help to describe a range of intermediate qualities of spaces.

These objects may be partial openings, screens, non-load bearing partitions, parts of a structural framework such as columns and beams, or any kind of "semipermanent" furniture. They are the physical components which describe, more effectively than a wall plane, the different territorial conditions at the volume edges.

Similarly, there is often a need to think of a window or a door as more than just an aperture in a wall plane. Microcad can be used to model three-dimensional openings which may have a usable depth dimension, thereby defining a transition zone between inside and outside. Bay-windows, planters, shading devices, portals, are all examples of architectural elements which can physically define such a zone.

About volumes:

As discussed earlier, there are a number of ways by which a building may be conceptualized and represented. It can be thought off as collection of physical elements such as columns, beams, assembled with specific constructional relationships between them. It can be defined as an assemblage of planes, surfaces and masses, with a set of geometric relationships. Or it can be looked at as a set of rectangular volumes with planar boundaries, sharing organisational relationships.

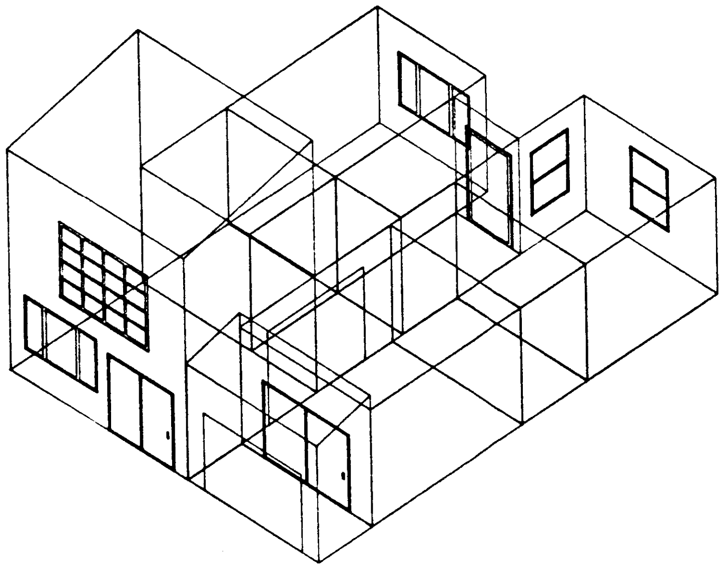
The use of the volume as the basic entity to work with in Keops is partially intended to keep the user a few steps ahead in both 3D geometric modelling and constructional technology, by making him think early in the design process in terms of 3D volumes with associated floors, roofs and walls. When the user inserts an orthogonal volume, his action may be thought off as a sequence of low level generic commands producing a high level result. By digitizing the opposite corners of a volume, the user is instructing the system to :

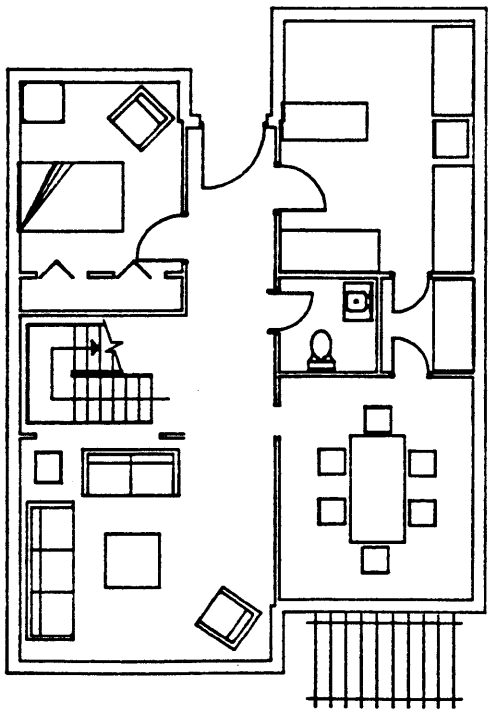
- define two corners of a space;
- draw orthogonal perpendicular lines from each point oriented towards the other point;
- trim the intersection of these lines;
- project the resulting rectangle along the vertical Z axis by some default value;
- designate the upper face of the volume as a roof, its base as a floor and its sides as walls;
- assign default values to these elements;

all in one command. The resulting volume may be a bay window or a building, it remains the smallest indivisible entity of which the building is made.

The first analogy a designer would make is to associate the concept of volume to a rectangular room, partly because of the topological limitation of four sides to a volume. By selectively deleting the sides of adjacent volumes, one can start generating spaces with more than four sides, such as L or U shaped spaces. Since more than one volume is required to define a space with more than four sides, the volume now becomes a part of the space. The more complex and multisided the space is, the larger the number of volumes which are required is, and the less the volume may be graphically associated to a room.
The same process of decomposing a space into a number of volumes occurs when sloping the roof of that space. The topological limits of the system restricts the roof to a single plane. This means that in order to create a single space with a gable roof, that space must be divided into two volumes with roofs sloping in opposite directions, and the side between them deleted. Similarly, in order to roof a space with a hip roof, the space has to be divided into three volumes, one of which is almost a triangle with two corners very close to each other (since the system does not allow triangular volumes). The consequence of this roofing process further decreases the association between volume configuration

and room layout. Another noticable result is the tendency, when working on the layout of upper floors (those with the sloped roofs), to design in terms of ceiling plan instead of space configuration, since the roofing configuration must be known before the layout of the volumes. In fact, upper floor plans read as ceiling plans.





CONCLUSION

The Personal Architect concept of two interfacing modules which communicate with each other is an innovative one. It is ironic, however, that the so called "Drafting" module imposes far less restrictions on building form than the "Design" module. I perceive it as a far more powerful tool for geometric modelling than its "Design" counterpart. There is something bothersome about a system which takes a line drawing (the wireframe model) and returns a complete set of dimensioned working drawings. My best description of Keops is yet another variation on the theme of CADD abbrevations, where CADD here would mean "Computer Assisted Design Development".

When choosing any tool it is difficult to imagine a selection process in which the user is not at least partially familiar with the system being investigated. In this case the evaluation was the result of a year long exposure to the system. Benchmark tests may be easy to devise for a drafting system, whereby the answer to a particular question from a checklist is either yes or no; it is entirely different if not impossible to define benchmark tests for a design methodology. There are no standards whatsoever which constitute typical design tasks, and it would be useless to even attempt to formalize the architectural design process in such a way that it could catagorized within any given mechanical process, no matter how "intelligent".

designer describe a building in whatever terms he deems appropriate; planes, shapes, privacies, masses or volumes. Mythical systems would allow the user to input many-valued, continuous qualities such as "finished" or "unfinished", "empty" or "full", "nearness" or "farness". Mythical systems would understand the user's concepts, his ambiguities or his hints (i.e . Negroponte's idiosyncratic systems). Mythical systems would behave as intelligent assistants rather than perfect slaves. Mythical systems would expand the designer's representation of a physical environment rather than restrict it. Mythical systems...

CUMULATIVE LISTING OF FUNCTIONAL CHARACTERISTICS

<u>Locational constraints:</u>

- Orthogonal lock
- Angular lock
- User-defined angular lock
- Grids display
- Different X and Y grids
- Different X and Y snaps
- Different grid from snap values
- Grid off while snap still active
- User-defined grid origin
- Simultaneous major and minor grids
- Rotation of grid
- Grid viewing in isometric
- Grid viewing in perspective
- Three dimensional grids (X,Y and Z)
- Editing and moving of single rows or lines
- Grid functions accessible from within the command

<u>Parts and blocks:</u>

- Ability to create and file parts
- Option on entities organisation (parts v/s symbols, blocks v/s write-blocks)
- Exploding of symbol
- Regrouping of symbol
- Stretching part on insertion
- Scaling part on insertion
- Rotating part on insertion
- Nested parts
- Placing symbols on multiple layers
- Automatic part updating
- Editing and redefining parts during use
- Automatic insertion of doors and windows
- Symbol library supplied
- Dynamic symbol dragging
- Showing symbol outline only

<u>Layers</u>

- Layers ability
- Limit on number of layers
- Different colors per layer
- Different linetypes per layers
- Freezing and thawing layers
- Moving object from layer to layer
- Copying object from layer to layer
- Naming of layers

Customization:

- User-defined library of symbols
- Tablet editing:
  - Designated keys only
  - Complete recalibration
- User-defined screen menus
- User-defined screen icons
- Execute and script files:
  - Without variable input
  - With variable input
- Macro's
- User-defined lines, patterns and fonts

Three dimensional modelling and visualisation:

- Geometric modelling:
  - Vertical axonometric extrusion (2 1/2 D)
  - Axonometric extrusion on any axis
  - True 3D; extrusion and rotation around any axis
  - 3D electronic wireframe model
  - Complex curves and surfaces
  - Option of volumes/planes/shapes
  - Construction planes with local 3D coordinates
  - Automatic section cut-away through the model
  - Automatic plan cut-away through the model
  - Automatic shading
  - 3D solid modelling
- Construction technology:
  - Automatic double lines wall thicknesses
  - Removal of unnecessary wall lines ("Clean-up" of intersecting L's and T's)
  - Creating automatic floors, walls and roofs
  - Inserting automatic 3D doors, windows and openings
  - Quick, interactive deleting and relocating of doors and windows
- Visualisation:
  - Isometric view
  - User-defined angle of isometric
  - 2 points perspective
  - 3 point perspective
  - Selecting viewing parameters (station point, focal point)
  - Saving viewing parameters
  - Modifying cone of vision
  - Viewing from within the model
  - Real-time dynamic viewing
  - Simultaneous multiple viewing windows on screen
  - Automatic hidden line removal
    - Surface priority
    - Surface orientation
  - Naming/numbering views
  - Saving views
    - Editable image files
    - Non-editable views
  - Automatic shading and surfacing
  - Cursor and grids viewing in 3D

BIBLIOGRAPHY

- Aish, R., "3D Input for CAAD systems", Computer Aided Design, March 1979.

- Barto, Michael, "Computervision Reveals Micro CAD/CAM Expert AEC System".

- Bensasson, Simon, "Evaluation of Computer Programs in the Building Industry: A European Approach?", International Conference on the Application of Computers in Architecture, Building Design, and Urban Planning, PARC 1979.

- Bensasson, Simon, "User Evaluation of Programs in Computer-Aided Building Design", Third International Conference on Computers in Engineering and Building Design, Computer Aided Design, March 1978.

- Berman, M., Cepeda, H., Gros, J., Volpe, E., "Visualisation automatique en ville nouvelle", International Conference on the Application of Computers in Architecture, Building Design, and Urban Planning, PARC 1979.

- Bijl, Aart, "A CAD Logic Modelling Environment", Architecture Science Review, December 85.

- Billon/Rocca, "Manipulation de MCA", Paris.

- Blake, Tyler, "Evaluating Interactive Graphic Interfaces: Human Factors Concepts for Real World Systems.", Proceedings of the Seventh Annual Conference, Computer Graphics National Computer Graphics Association, Anaheim, CA, May 1986.

- Bridges, A.H., "Case Studies in Computer Aided Visual Impact Analysis", Proceedings of the International Conference on Computers in Architecture, PARC 1983.

- Cross, Nigel, "Assessing Computer-Aided Architectural Design Systems", Third International Conference on Computers in Engineering and Building Design, Computer Aided Design, March 1978.

- Darrow, Barbara, "PC Based CAD Comes of Age", Design News, August 1985.

- Dill, Jonn C., Pittman, Jon H.,"Evaluate your Options.", Architectural Technology, November 85.

- Driay, J.J., Azema, J.M., Chapron, E., Mahe, P., Senechal, R., "CAO: de la conception a la construction, tout un programme", Techniques et Architecture, Paris, 1983.

- Finkel, James I., "Buying a PC CAD System", CAE, July 1985.

- Foley, J.D., Vandam, A., "Fundamentals of Interactive Computer Graphics", Addison-Wesley, Reading, MA, 1982.

- Goumain, Pierre, Mallen, George,"Interface Design and the Future of Interactive Building Design Systems: A Study Report", Third International Conference on Computers in Engineering and Building Design, Computer Aided Design, March 1978.

- Greenberg, Donald, "The Coming Breakthrough of Computers as a True Design Tool", Architectural Record, September 1984.

- Grezes, Denis, "Micro ou mini: comment choisir", Techniques et Architecture, Mai 1971.

- Hammond, B.G., Leifer, D., "A Graphics Interface to Complement Traditional Techniques", Proceedings of the International Conference on Computers in Architecture, PARC 1983.

- Harrison, Steve, "Coordinating A/E/C CAD", Computer Graphics World, November 1983.

- Hart, Glenn, "CAD support: an Embarassment of Riches", PC Magasine, March 86.

- Hart, Glenn, "CAD: the Big Picture for Micros", PC Magasine, March 86.

- Hart, Glenn, "Complex CAD software for the IBM PC", PC Magasine, March 86.

- Kalay, Yehuda K., "Redefining the Role of Computers in Architecture: from Drafting/Modeling tools to Knowledge Based Design Assistants", Computer Aided Design, January 1985?.

- Little, Steve, "The Organisational Implications of CAAD", Department of Design Research, Royal College of Art, London.

- Marcus, Aaron. "Graphic Design of User Interfaces.", Proceedings of the Seventh Annual Conference, Computer Graphics National Computer Graphics Association, Anaheim, CA, May 1986.

- Miller, Frank., "The Personal Computer CADD Revolution: Capabilities and Trends", Proceedings of the Seventh Annual Conference, Computer Graphics National Computer Graphics Association, Anaheim, CA, May 1986.

- Mitchell, William J., "Computer-Aided Architectural Design", Van Nostrand Reinhold Company, NY 1977.

- Montalvo, Fanya S., "Diagram Understanding: The Intersection of Computer Vision and Graphics", AI memo, November 1985.

- Negroponte, Nicholas, "On Idiosyncratic Systems", Technical report, March 77.

- Radford, A.D., Stevens, G., "Style in 1984: Computers and Building Form", Proceedings of the International Conference on Computers in Architecture, PARC 1983.

- Reynolds, R.A., "A Comparison of Graphical Information Handling Techniques in Architectural Practice", Third International Conference on Computers in Engineering and Building Design, Computer Aided Design, March 1978.

- Rogers, Gary, "Dynamic 3D Modeling for Architectural Design", International Conference on the Application of Computers in Architecture, Building Design, and Urban Planning, PARC 1979.

- Rufle, Simon, "A Better Man-Machine Interface to Replace the Keyboard",Proceedings of the International Conference on Computers in Architecture, PARC 1983.

- Sannie, Antoine Quentin, "MacIntosh: La table a dessin de l'architecte", Decision Informatique, December 1985.

- Schley, Michael, "CAD Buyer's Checklist.", Architectural Technology, September 85.

- Stoker, Douglas F., Weingarten, Nicholas H., "Computers: CAD v/s CAD", Architectural Record, September 1984.

- Swinson, Peter S.G., "Prolog: a Prelude to a New Generation of CAAD ", Computer Aided Design, November 1983.

- Teicholz, Eric., O'Connor, Timothy C. "Computer Graphics for Design and Drafting."

- Witte, Oliver R., "Affordable CAD.", Architectural Technology, December 84.

- Witte, Oliver R., "Affordable CAD: Taking the Plunge", Architectural Technology, January/February 86.

- Wright, Victor E., "Drafting by Design", PC Tech Journal, January 1986.

- Zeitoun, Jean, "Panorama et evolution des approches de la CAO", Techniques et Architecture, Mai 1971.