Interactive 3-D Modeling with Personal Computers

Robert W. Thornton and Gregory J. Glass
New York Institute of Technology
Computer Graphics Laboratory
Old Westbury, New York 11568

Abstract

An interactive three-dimensional modeling system, called *Facet*, is presented. *Facet* was developed for use during the design of physical artifacts: from studies of simple shapes, spaces, and forms to detailed modeling of complex physical systems. *Facet* provides a visual medium for enhancing perception of the model and for exploring various alternatives during the design process. In order to make this 3-D modeling medium available and affordable to any interested design professional *Facet* operates on commonly available IBM or compatible personal computers. In addition to an overall description of *Facet* and its interactive modeling techniques, several underlying issues in the development of a modeling system are discussed. These issues include: the interactive user interface; 3-D display techniques; and ways of dealing effectively with model complexity. A number of areas for continued development, based on experience with the current implementation, are discussed.

KEYWORDS: configurable, interactive, three-dimensional, modeling, computer graphics, personal computer, design.

1. Introduction to Facet

Facet is an interactive 3-D modeling system for design professionals that runs on widely available personal computers (PCs). It allows a user to develop and manipulate 3-D computer models of shapes, spaces, and forms for use during the design process. Models are displayed on the screen in a *wire-frame* form while the user is interactively building or modifying the model. Selected views (including perspectives) may subsequently be rendered by the computer with shaded surfaces or may be output with a pen plotter or dot matrix printer. These two-dimensional pictures can then be the basis for working drawings or other enhanced renderings to be used for communicating the design to others.

Facet is initially directed to spatial designers who need to study alternative forms and shapes during the design process and create visual representations of physical systems. However, it will be readily adapted for use in other diverse areas of 3-D modeling such as generating input for graphic arts illustration systems as an aid in producing perspective drawings and renderings; building models, sets, and backgrounds for use by a 3-D animation system, when the elements being designed are static and do not require the expensive hardware necessary for real-time dynamic manipulation; and generating charts, technical illustration, and other graphics to be incorporated with word processor output for document production.

A careful combination of features positions *Facet* as a readily available tool for 3-D modeling integral to the design process: a need that has not been adequately addressed affordabley in the past. Some of the key aspects of *Facet* which recognize this need follow.

- Perhaps the most important aspect of *Facet* is the quality of interaction. The emphasis is on providing a modeling aid for use during the initial design stages. This is accommodated to a large degree in the way the designer is able to effectively interact with this electronic modeling medium. For example, the designer is able to quickly develop models of shapes, spaces, and forms; examine the model from many points of view to establish an accurate mental perception; and easily make changes in exploring different alternatives.

- A second key aspect of *Facet* is that it can deal with complex models. Unlike some previous modelers, project complexity is not limited by the computer's address space. Features are provided that enable the designer to organize the model with structure, grouping, and symbolic naming to work with abstractions and deal effectively with complex or detailed project models.

- Third, the ability to generate high quality shaded surface renderings, after interactively constructing the model, is an integral feature of *Facet*. The designer is able to assign various attributes to elements of the model that allow *Facet* to exercise the full capabilities of a high quality display device, if available, yet still operate effectively when using less capable display hardware.

- Finally, low cost is a significant feature of *Facet* to make it feasible for use by any interested professional designer. The basic personal computer system to support *Facet* can be purchased today for as little as $2,000. (A much faster, full-featured system with the ability to generate full color renderings can be assembled for less than $10,000.) Marketing strategy and pricing for the *Facet* software have not been finalized.

To be available to the largest group of users *Facet* runs under the PC-DOS (MS-DOS) operating system on IBM (or compatible) PCs. The minimum hardware configuration includes the basic PC with 640K bytes of main memory, a math coprocessor, a hard disk, a supported graphics display

subsystem and a locator (e.g. mouse). MS-DOS computers with faster or more powerful processors than the standard PC, but not as compatible, are also supported by *Facet* as long as an appropriate graphic display and locator are used. Any such increase in power will apply directly to improving the speed of interaction, for example, when calculating and displaying a new view of the model.

Facet supports several commonly used graphics devices including IBM's Color Graphics Adaptor and Enhanced Graphics Adaptor and the Microsoft mouse. Full color display systems (such as Number Nine's 512 X 32) are supported for smooth shaded color renderings. A variety of hardcopy devices are supported, including pen plotters, dot matrix printers, and film recorders.

2. Model Editing Techniques

Models in *Facet* are represented geometrically as polylines (one or more connected line segments) and polygonal surfaces. In addition, polygons are often joined together (sharing common edges and vertices) to form polyhedral shapes. Although curved lines and surfaces are not represented specifically, they are readily accommodated with the use of approximating polylines and polyhedra. (In particular, built-in primitive shapes are provided to quickly model arcs, domes and other common objects.) There is a display attribute that may be specified for such approximating polyhedra that causes them to be smoothed together (using Gouraud or Phong shading) when subsequently rendered with shaded surfaces.

Although the representation of models in *Facet* is fully three-dimensional, we are limited by our display hardware to visible windows on the model that are 2-D planes of projection. Likewise, we are limited by our pointing hardware to show locations in a single plane at any given time or to select or *pick* existing elements of the model by pointing to their displayed location on the screen. *Facet* employs two primary concepts in its operations for constructing and modifying models that minimize these inherent limitations:

- The first is to provide a *current plane* in 3-D space on which operations may be performed. For example, detailed surface features may be drawn on the face of an object after positioning the *current plane* as required. This *current plane* is readily positioned and oriented, as desired, and is displayed graphically on the screen with the model. For most operations in the *current plane* the user will find it convenient to select the orthographic projection with the *current plane* as the picture plane: making the surface of the display screen congruent with the *current plane*.

- The second concept is to do operations relative to the position or shape of existing elements in the model. For example, an object may be rotated to align with an edge of another existing object; the existing edge is readily *picked* by pointing at it on the display.

2.1. Current Plane Editing

A common strategy for constructing models with *Facet* is to begin by defining lines and polygons in space by working in selected positions of the *current plane*. Elements may then be connected or combined with eachother by picking from the existing vertices to form additional lines or polygons. There is also a powerful *extrusion* operation with which the user can sweep existing lines and polygons through space to form polyhedra whose cross sections are defined by the lines and polygons being extruded.

Work in the *current plane* is normally done graphically by pointing at desired locations on the plane with the locator (e.g. mouse). To provide an intuitive capacity for drawing on the *current plane* an orthographic view, with the *current plane* as the picture plane, is normally used so that the *current plane* is congruent with the display screen. The user can zoom the display in or out and pan around to focus on desired areas of the *current plane*. There are some unique aids to quickly and accurately select desired locations in the plane. These include:

- multiple grids of coordinates to which points may be automatically latched (with graphic definition of, and graphic feedback on, the operation of the grid's gravity field). By using two grids the effect of a rotated grid may be obtained.

- sets of slopes to which lines may be automatically snapped; and

- *background* graphics (derived from other views of the current model, or other *Facet* projects) with which points may be automatically aligned.

The coordinate locations generated by any of these automatic latching aids are, of course, calculated to the full resolution (32 bits) of the model's world space and are not affected by the resolution or zoom factor of the display being used. These aids may be grouped together as an *environment* to be saved under a user specified name for later use in the same plane or other planes or even in different *Facet* projects.

Most of the model change operations, such as moving, rotating, and changing size or shape, may be performed with either absolute locations or measures (e.g. selected graphically in the *current plane*) or with locations or measures relative to existing elements of the model.

2.2. Extrusion

A flavor of how *Facet* works may be gained by following the steps of an extrusion operation. Extrusion takes groups of polygons or edges and sweeps them through space to form new polygons. The sweep proceeds in a specified direction and is terminated by a specified plane.

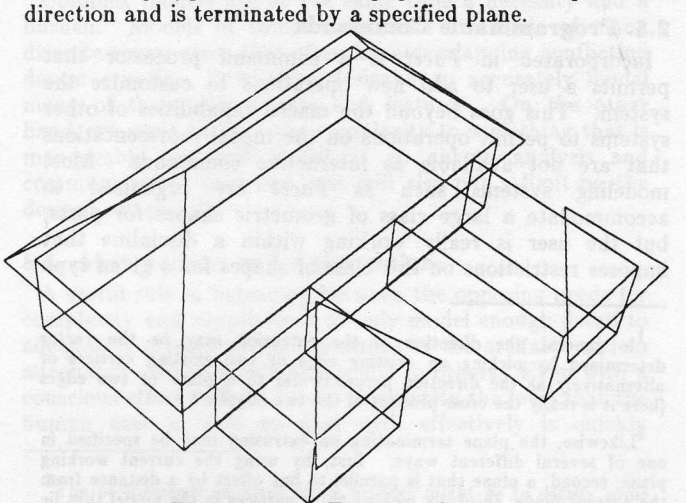

Figure 2-1: Extrusion Step 1

Figure 2-1 shows a simple building envelope being modeled in *Facet*; the pedements of the two projecting wings have been drawn by placing the current plane at an offset from the front of the wings. The user would like to extrude these triangles into the main roof to complete the form. First the direction of extrusion is specified. In this example the user picks an edge in the projecting wing to define the extrusion direction.[1] Second, the user defines the terminating plane. In the example the user picks three vertices that define the near slope of the existing roof over the main wing of the building.[2] Finally, the user selects the elements to be extruded. After the two pedements have been selected the extrusion takes place with results shown in figure 2-2.

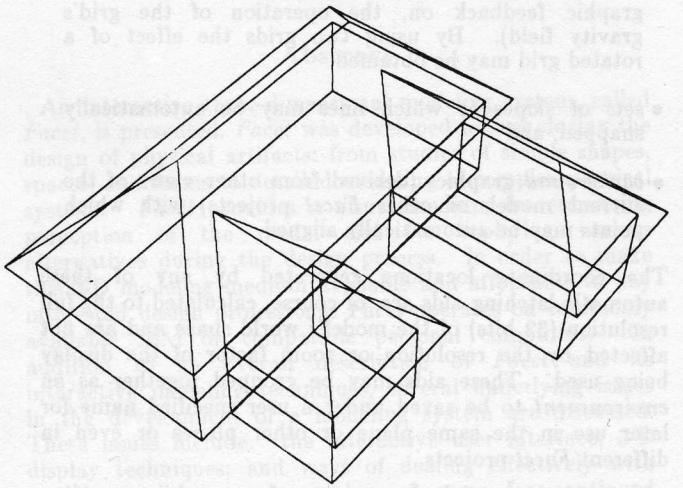

Figure 2-2: Extrusion Completion

The direction of an extrusion may also be specified by a polyline to create a multiple step extrusion. Here the terminating plane for the intermediate changes in direction at the polyline vertices is the average between the planes normal to the two adjacent edges of the polyline. An example of multiple extrusions is shown in figure 2-3 which shows a circle that has been extruded along the top of a building to form a pipe.

2.3. Programmable Commands

Incorporated in *Facet* is a command processor that permits a user to add new operations to customize the system. This goes beyond the macro capabilities of other systems to permit operations on the model representations that are not available as interactive commands. Most modeling systems such as *Facet* are organized to accommodate a large class of geometric shapes for parts, but the user is really working within a discipline that imposes restrictions on this class of shapes for a given type

[1] In general, the direction of the extrusion may be the vector determined by picking an existing edge or two existing vertices or alternatively as the direction perpendicular to a plane or two edges (here it is really the cross product of the two edges).

[2] Likewise, the plane terminating an extrusion may be specified in one of several different ways: first, by using the current working plane; second, a plane that is parallel to but offset by a distance from the current plane; third, by picking three vertices in the model that lie in the desired plane; and fourth, the plane that passes through a selected vertex and is perpendicular to a selected edge.

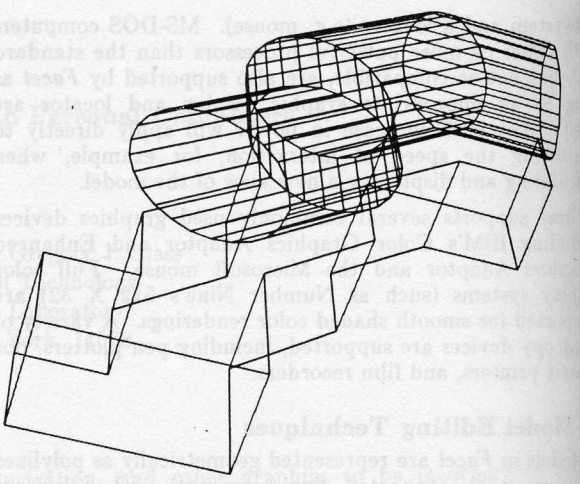

Figure 2-3: Extrusion along a Polyline

of part. Knowledge of such restrictions may be used to advantage in the definition of operations provided for the particular type of part. Operations restricted in this manner make it easier to construct valid examples of such parts. For example if a particular rectangular solid represents a wall, the different compositions that may be constructed by the user using that rectangular solid should be restricted to those that are possible with walls. Restriction of the possibilities that may be modeled during the design of an artifact makes the modeling task faster [4]. In order for a modeling system to speed the task of the user it should take advantage of any inherent restrictions and define the manipulation operations so they reflect those that may occur on the actual artifact.

3. Flexibility in User Interface

Perhaps the most important aspect of *Facet* is the quality of interaction. The emphasis is on providing a modeling aid to the designer for use during the initial design stages. This is accommodated to a large degree in the way the designer is able to effectively interact with this electronic modeling medium. For example, the designer is able to quickly develop models of shapes, spaces, and forms; examine the model from many points of view to establish an accurate mental perception; and easily make changes in exploring many alternatives.

The user's interface to *Facet*'s operations is at the same time easy to learn and effective in use for the expert. All operations may be accessed through mouse activated menus (which may be restructured by the designer to fit any particular desires). In addition, operations may be bound to any of the keys of the keyboard for invocation at the push of a button. Any operation may also be accessed by explicitly typing in a unique abbreviation of the operation's name or selecting it from a scrollable window containing descriptions of all the operations.

The style of menuing used is *pull-down* or *drop-down*, similar to those used in several recent integrated operating environments (such as Apple's MacIntosh [7]). The operations are grouped into several small menus according to some similarity of use and each menu has a name. Small groups of these menus are clustered together to be accessible to the user simultaneously. Each operation is represented in the menus by a descriptive phrase (i.e. menu button). Normally, the display screen is filled by the

window displaying the 3-D model except for a single line of text at the top containing the names of the menus available in the currently active cluster. By pointing to a menu name, the user *pulls down* the menu whereupon the list of operations available in that menu appear immediately on the screen. The user drags to the desired operation to select it and, on doing so, the model display window is immediately restored and the selected operation begins. Within the individual operations, additional input requested of the user is performed graphically, whenever appropriate, by pointing to locations in the *current plane*, by *picking* existing elements of the model, or by answering a question from a multiple choice *dialogue menu*. In addition to activating a particular operation, selecting an item from a menu can also be used to effect a change to a different active menu cluster or a change in the contents of a pull-down menu.

There has been a strong commitment, beginning with the initial designs for *Facet*, to a configurable user interface. This is primarily embodied in the ability of the users to define their own set of command menus. In the first place, the users may define the names used in the menus to represent the commands. Second, they may define the grouping of commands to form menus and the names of the menus. Third, they may define the clustering of the menus. Finally, the dynamics of when the clusters and menus change may be fully customized. Configurable interaction is provided in order that the designers may tailor the interface according to their specific needs and desires since each designer might evolve an uncommon approach to design using this modeling medium. In addition, an individual designer will find it useful to organize the interaction differently for varying types of projects.

Brown [1] has defined a methodology that might be used to specify the structure of a menu network. His approach is not different from the one used in *Facet*. Both approaches define the structure using a predefined syntax but *Facet* does not provide any mechanism for explicitly controlling the complexity of the menu structure. We did not wish to impose, at the time of implementing the menu package, any untested assumptions about what the desirable properties of a menu structure would be for our applications. Because of this uncertainty about what constitutes "good" structuring we provide one low-level structuring tool that in Brown's terms is the GOTO statement.

If the user is going to be permitted to repartition his menus and order them to his liking there must be some discipline imposed on the structure of the application program. This may be done through the syntax of the menu structure definition. It is well know that these problems of structure definition exist and many people have proposed solutions [6, 8]. *Facet* is implemented using yet another solution which will be described in more detail in a future paper.

4. Model Display Techniques

As a physical system modeling tool (which might often be used during the design process) it is important for *Facet* to provide the user with a strong perception of the shapes and forms being modeled. This visual perception is readily gained through the ability to look at the 3-D model from any desired point of view. *Facet* displays selected parts of the model (the *working set*, as presented in the following section) on the screen as a *wire frame* (i.e. with the edges of each polygon drawn as lines) while the user is interactively constructing or modifying the model. The user is able to select a view graphically by pointing to the desired location of both the viewing position and the center of interest. Alternately, orthographic views may be selected with the picture plane placed in the *current plane* (which may be readily positioned to any desired location in 3-D space).

Selected views may be saved for later use in *view files*. Although the specifications stored in such files represent a 2-D projection of the 3-D model, they are not specific to the particular display screen in use when they were saved in order that the views may be subsequently displayed on other devices including pen plotters and dot matrix printers. In addition to the location of the elements in the 2-D view, the *view files* also contain depth information so they can be displayed with polygons rendered as shaded surfaces with any hidden surfaces removed. Also, *view files* can be automatically regenerated, on request, to update a previously saved view after the model has been modified.

5. Effectively Dealing with Model Complexity

Facet is able to accommodate large complex models. However, complexity by itself is of no advantage: it degrades the speed of interacting with the model and leads to visual clutter, logical confusion, and chaos. In order for the user to deal effectively with complex models, *Facet* provides several mechanisms for structuring and organizing the elements of a modeling project. *Facet* allows the user to develop abstractions and aggregations to work with selected aspects of the model in isolation from other aspects. In *Facet* there are several basic mechanisms for organizing the model: *the family tree*, *collections*, *symbolic naming*, and *the working set*. Some CAD systems, usually those based on automated drafting and used primarily to generate 2-D drawings, provide a simple method of partitioning a project (i.e. drawing) into several separate entities (usually called *layers* or *overlays*), a technique that is derived from traditional drafting methods. In order for a 3-D modeling tool, such as *Facet*, to be used effectively with complex projects, richer ways of organizing the modeling medium are required.

Complex models are at the same time a necessity and a burden. Models of complex artifacts are made of many discrete components that often have overlapping conflicting design criteria. It is often desirable to accurately model many of these components and features. On the other hand, we want to limit the complexity to something that is manageable and still useful as an aid to analysis and communication; time and cost will also put a limit on the desired complexity.

5.1. Abstractions and Aggregation

A useful rule in balancing between the opposing needs for complexity and simplicity is to only model enough detail to adequately predict the *performance* of the artifact[3] and to adequately communicate the model to others. Even with a conscious effort to limit model complexity the level that the human user is able to deal with effectively is quickly

[3]The performance required of an artifact might be to satisfy some form of aesthetic evaluation as well as structural, functional, or other types of more readily calculated performance.

exceeded. We must accommodate this overload by learning to make abstractions and aggregations.

All models that fall short of embodying the emulated artifact itself are in one way or another abstractions. When abstractions eliminate the need to deal with details that are irrelevant at the moment, the user is free to concentrate on a few related criteria at a time. For example, two simple boxes of appropriate proportions might be used as an abstraction of the World Trade Center to do site positioning studies. Often, it is helpful to abstract an abstraction and so on, establishing many layers or a hierarchy of abstraction.

With aggregation we can group items similar in feature, purpose, or design, allowing us to work with the group efficiently, in isolation from other items and groups, and apply operations to the entire group in a like fashion. For example, advantageous groupings in a building might include the structural, electrical, or mechanical components; spaces serving a similar function, such as stairways; or different components from the same manufacturer.

5.2. Model Organization in *Facet*

The previous requirements drove the development of the four organizational mechanisms provided by *Facet*. These are the *family tree*, *collections*, *symbolic naming*, and the *working set*.

The *family tree* allows a hierarchical organization of the elements of a model. The nodes of the tree are called *members* with each having a *parent* member, a set of zero or more *sibling* members (all having the same parent), and a set of zero or more *child* members. At the base of the family tree is the *root* member that does not have a parent, is an ancestor of all other members in the family tree, and is provided automatically in a new project. The family tree is inherently a spatial organizational mechanism. In the first place, all parts, or spatial elements, of the model exist within the hierarchy of the family tree. In addition, the content of each member in the family tree (other than the hierarchical structuring information) is a location, orientation, and scale in 3-D space that is applied to an optionally referenced part description (called a *prototype* in *Facet*). Each prototype may be referred to by one or more members allowing many discrete instances of similar parts with only a single description being stored. Finally, the location/orientation/scale of a particular member is not defined in *absolute* terms within the 3-D world, but is *relative* to that of its parent in the family tree. Thus, an operation performed on a particular member may also affect all descendants of that member. For example, moving a member to a new location may be done in a fashion that effectively moves all descendants in a like way so that the relative positioning of the member and its descendants remains the same.

A second mechanism for grouping members, without any implied spatial connectivity, is provided with *collections*. Collections offer a means of assembling aggregations of members for any desired purpose in a manner orthogonal to their structure within the family tree. A member may be included within several different collections or, on the other hand, is not required to be in any collection. Other collections may be included in a collection, allowing hierarchies of collections.

The third mechanism, *symbolic naming*, allows the user to associate a name (any string of up to 60 characters) with a particular member or collection. Selection of such a component may be done by specifying its name. *Facet* will also, in certain situations, identify components to the user by displaying their names.

The last organizational mechanism is a dynamic sub-set of the project model, called the *working set*. This is the set of components that the user has selected to work with together at a particular point in a modeling session. All the components in the working set are displayed on the screen and are readily accessible by the user through the *Facet* operations. By the same token, components not in the working set are not displayed and are not generally accessible without placing them in the working set. There is a set of operations in *Facet*, collectively called the *project browser*, that allows the user to scan through the entire project model and move components into and out of the working set. The project browser permits the user to search through any accessible project and copy parts of other projects into the current one.

5.3. Modeling Medium Limitations

In addition to addressing the conceptual difficulties of effectively working with complex models, we must also deal with the physical limitations of our modeling medium. For systems with an electronic computer performing an essential role, such as *Facet*, the principal limitations are the memory size and processing speed. The size of memory regulates the amount of model information that may be displayed and manipulated simultaneously. The processing speed affects the smoothness of interaction, e.g. when redisplaying the model from another viewpoint, selecting an item, or performing a spatial modification. The amount of processing required for such operations is directly proportional to the complexity of the model on display.

Some modeling systems choose the route of limiting a project's size so that it will fit into the computer's main memory(examples of this are Design Board Professional by Mega CADD Inc. [3] and Polycad/10 by Cubicomp Corp. [2]). These might be called part modelers as opposed to assembly modelers. Others use a simple form of virtual or paged memory, pretending that the main memory is larger than it really is and storing it instead on the slower disk memory(examples of this include Microcad by Imagimedia Technologies Inc. [5]). As in any system that saves project models from one session to the next, *Facet* stores its models on disk. *Facet* does not limit the project by the size of main memory. It uses the organizational mechanisms described above to allow the user to select parts of the model to be operated on together at any given point (perhaps a set of abstractions or aggregations). This *working set* is then resident in main memory and does not incur the slowdown of continually reading and writing the data on disk as in a demand paged memory scheme, that is not able to take advantage of the user's logical organization of the project data and working patterns. The user will normally keep the working set as small as possible, including only those components relevant to or providing context for the current operations, to minimize the processing time for interactive operations. *Facet* includes a *project browser* with that the user may alter the working set (by adding or removing items) at any time during a modeling session.

6. Future Directions for Facet

Although *Facet* is now a complete system, the area of 3-D modeling is very wide ranging and we have several additional enhancements in progress and planned for development over the next several years. In the area of interactive model construction and editing techniques developments that are planned include:

- Suites of application specific modeling operations and primitive objects (ultimately user defined primitive objects provided through application of a programming facility).
- Additional general purpose modeling operations, e.g. patterned replication, polygonal meshes, spatial reconstruction (digitizing 3-D shapes from multiple 2-D views).
- Use of image digitization from hardcopy, film, video tape, or video camera, for use as surface texture on 3-D model elements or as backdrops for the model where the synthesized perspective projection of the project model is matched to that of the scanned in scene.
- Interactive rendering enhancement with *paint system* type of techniques and 2-D images as backdrops.
- Solid modeling - using a winged-edge type of polyhedral boundary representation currently running under Unix (with robust spatial set operations).

In the area of general interactive control and menuing, we will be experimenting with ways of providing a displayable map of the entire menu structure, with the active menu cluster highlighted, to provide an effective aid to navigating a complex network.

The next step would be to provide an interactive menu structure editor that operated directly on the displayed graphic representation.

Continuing developments are planned in dealing with the graphic display and access of *Facet* model data. Several enhancements to the shaded surface rendering are under way. To transfer graphic data to drafting systems, paint systems, and other picture enhancement programs, output will be provided in IGES and widely used proprietary formats. (Currently a library of routines is provided for use by client programs that allow direct access to the *Facet Disk Data Structure.*) The current simple form of interactive animation set up provided to interpolate camera views through key poses will be enhanced to allow more flexibility of movement. With the use of new high performance hardware display processors (both on the PC and other hosts for *Facet*) the playback of these animated sequences will be speeded up to near real-time; interactive operations will also benefit from increased dynamic motion and speed of redisplay.

7. Conclusions

In *Facet* we attempted to provide a modeling tool that is accessible to many designers during the conceptualization of a project. We have done this by presenting operators that make generation and change of models fast and simple. Use of the system has shown that it has the depth and flexibility to easily adapt, extending the habits and design processes of an individual designer.

References

[1] Brown, J.W.
Controlling the Complexity of Menu Networks.
Communication of the ACM , July, 1982.

[2] *Polycad/10 User's Manual - Version 2.0*
Cubicomp Corporation, Berkeley, Ca, 1985.

[3] Engelke, D.J., Heitzman, F.E., Voosen, J.C.
3-D Program Reviews.
Architectural Technology :39-42, January/February, 1986.

[4] Glass, G. J.
Automated Part Location in the Design of Assemblies.
Technical Report CSL-83-4, Institute of Building Science, Carnegie-Mellon Univ., January, 1983.

[5] Hart, G.
Complex CAD: Software for the PC.
PC Magazine 5(5):111-148, March, 1986.

[6] Kasik, D.J.
A User Interface Management System.
Computer Graphics 16(3), July, 1982.

[7] *Macintosh*
Apple Computer, Inc., 20525 Mariani Ave., Cupertinc CA 95014, 1984.

[8] Price, L.A.
Design of Command Menus for CAD Systems.
In *Proceedings of the 19th Design Automation Conference.* Las Vegas, June, 1982.