William Fitzgerald
Franklin Gracer
Robert Wolfe

GRIN: Interactive Graphics for Modeling Solids

This paper describes an experimental system for the original generation and subsequent modification of volume models of complex physical objects, using interactive computer graphics. The models are built up from primitive volumes, e.g., *cuboids, cylinders, swept surfaces, etc., entered by a mechanical engineer interacting with a two dimensional projection of the model on a graphic display screen. The primitives may be entered at any orientation in 3-space and combined to form a single polyhedral model. The central issue is the provision of an efficient, natural means for generating these models.*

Introduction

For over twenty years, computer aided design (CAD) has been increasing productivity in the electronics industry by providing designers with performance data before circuits are actually fabricated [1]. In recent years, CAD has come to the forefront with the tantalizing promise of techniques that may someday do for mechanical design what is already possible for electronic design.

CAD itself consists of two distinct disciplines, graphics and modeling, both of which are essential. In the electronics case, for instance, the interactive display of a logic diagram is graphics, but the program that simulates the operation of this circuit is modeling. In the mechanical case, a similar distinction may be made between the display of an engineering drawing of a part and the data base that enables a program to compute the moments of inertia of that part.

The experience to date in developing mechanical CAD systems indicates that mechanics is a more complex domain than electronics. The added difficulty is principally due to the fact that mechanics tends to deal with three dimensional (3D) parts rather than 2D circuit layouts. Since existing graphics displays are essentially all 2D, the added dimension requires extrapolation from a 2D display space to a 3D object space. In modeling, there are additional problems stemming from the fundamental laws of physics, *e.g.*, no two objects may occupy the same space at the same time.

At this laboratory, work on mechanical CAD systems began in 1975, using geometric modeling. Because this undertaking proved to be so challenging, the initial focus of the work was solely on the modeling domain and was limited to polyhedral representations. Specifically, a Geometric Design Processor (GDP) was built that provided data structures and programs for modeling complex mechanical objects from volume primitives (*e.g.*, cuboids and polyhedral approximations to cylinders) and for deriving many of their engineering properties [2, 3].

A model is represented as a hierarchical structure which retains primitives at the lowest level, as well as compositions of the subtended primitives at higher nodes. GDP has been used to model complex mechanical parts and assemblies from the computer, aircraft, heavy equipment, and architecture industries. It has also been used in the modeling of industrial robots and the parts, processes, and applications associated with them, *e.g.*, automated path planning [4] and model driven vision [5]. As in most existing geometric modeling systems, the user interface was a statement oriented procedural language [6]. Unfortunately, because GDP lacked an interactive graphic input facility, it was usable only by highly skilled programmers.

It was in this environment that the research reported here was undertaken. The work was intended to answer some fundamental questions regarding mechanical CAD

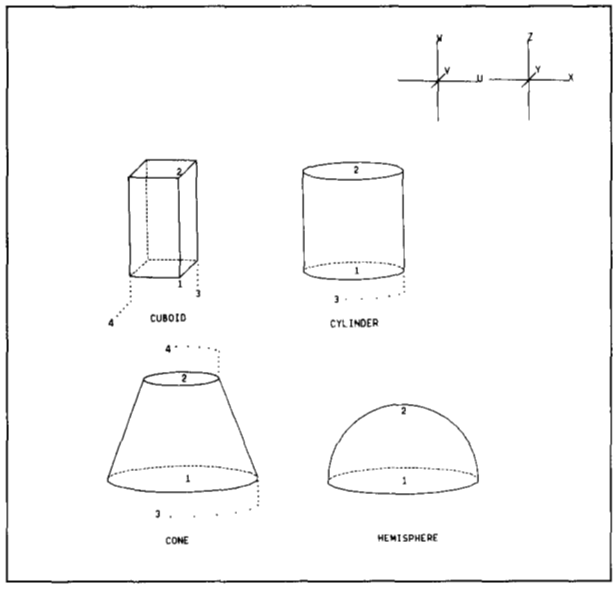

Figure 1 Primitive volumes and their entry points.

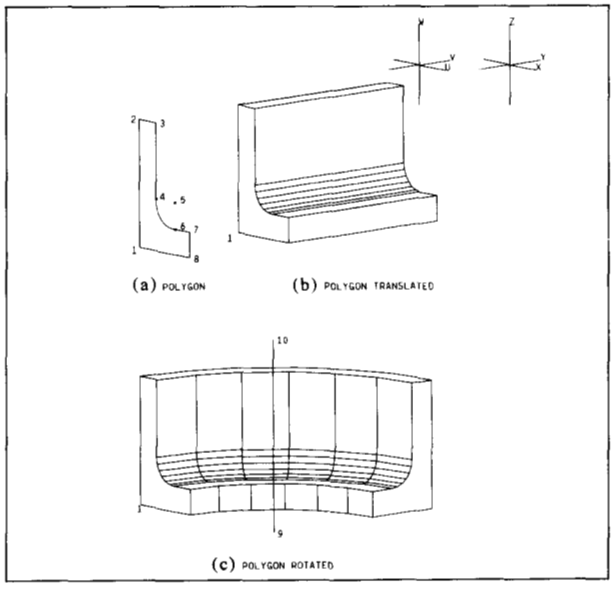

Figure 2 Translated and rotated polygons and their entry points.

systems: First, can computer graphics provide a simple means for specifying volume models of 3D objects? Specifically, can a simple protocol for a mechanical designer, who is not a programmer, result in generating the data required for a complex geometric model? The second question, raised by the mechanical community, is whether a mechanical designer, who is trained to design with points, lines, and arcs in 2D orthographic views (either on a drafting board or a graphic display), could think and work in terms of 3D volume primitives on a 2D graphic screen.

The approach taken was to add an interactive graphics input facility called GRIN (GRaphic INput subsystem) to GDP's modeling in order to build a complete experimental mechanical CAD system. An attempt was made to retain the power of the procedural language by mapping, as far as possible, the features of the language to the graphic interface. The resulting system was used by its developers and found to be effective in a wide variety of application demonstrations, several of which are described here. It is currently being used in a pilot production environment by a mechanical design group.

Prior art

The beginnings of mechanical CAD can be traced to 1963 in papers by Coons [7] on CAD requirements, Sutherland [8] on the Sketchpad system, and Roberts [9] on the now classical computer graphics problem of hidden line removal. One of the earliest production uses of computer graphics for mechanical CAD was in the aerospace industry, reported by Chasen [10] in 1965.

A number of CAD systems have been built in recent years for generating three dimensional computer models of physical objects and mechanisms [11–13]. One of the most interesting approaches has been the combination of many instances of a few simple volumes to produce models of complex parts. This approach is typified by the work of Braid [13] Baumgart [14], Voelcker [15], and Wesley *et al.* [2]. The primitive volumes are cuboids, cylinders, cones, swept polygons, etc., and the approach to manipulating the primitives is a statement oriented language allowing the user to translate, rotate, and scale the primitives and to combine them with the set operations union, difference, and intersection. A particularly powerful implementation of this procedural approach is that of Grossman [6] because the procedural language is embedded in PL/I and therefore has all the computational power and flexibility of a general programming language.

The promise of these systems to automate the mechanical design and manufacturing process has been very slow to materialize. Only in the last two or three years has the use of graphics systems for mechanical CAD begun to grow rapidly, and then primarily in the automation of two dimensional design/drafting procedures. A small number of three dimensional systems have been successful, nota-

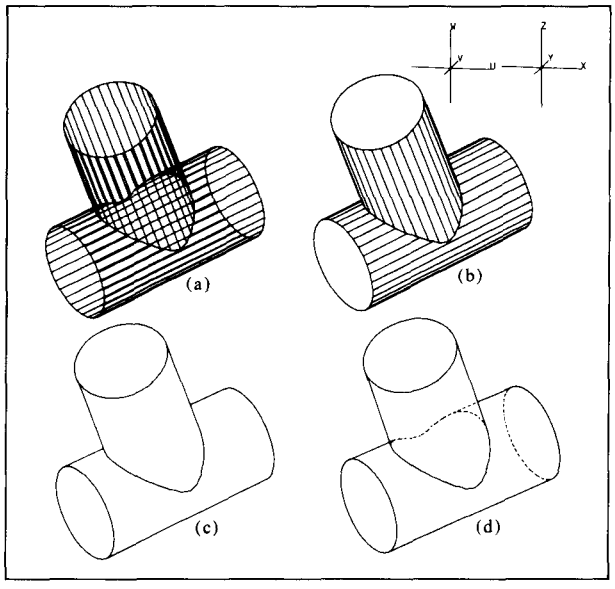

Figure 3 Methods for rendering visual displays of volume models.

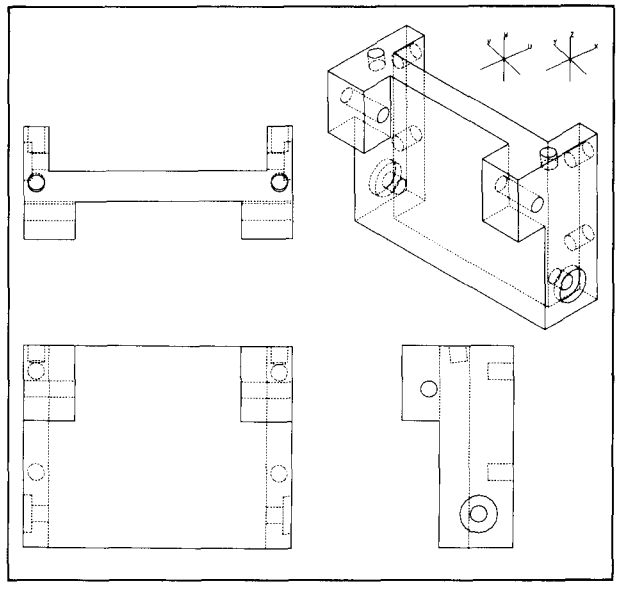

Figure 4 Automatic production of views for engineering drawings.

bly for the design of sculptured surfaces in the automobile and aerospace industries.

As late as 1978, in his review of graphical CAD systems, Elliott [11] commented on the use of computer graphics and a 3D internal computer model by a mechanical designer: ". . . there is a fascinating and open question as to how he will work from the essentially '3D' thought in his mind, through the 2D screen, to a computer model which simulates his 3D concept. . . . The question has attracted the attention of design methodologists, computer scientists, engineers, and psychologists from the time of the '3D sketching box' of Gregory [16] to a recent paper on the 3D wand and head mounted display [17]."

The question is especially relevant and fascinating when the designer uses volume primitives instead of lines and arcs to generate a computer model. The extensive review of geometric modeling done by Baer, Eastman, and Henrion in 1979 [18] does not treat the graphic user interface in any detail. Papers on geometric modeling have been concerned primarily with algorithms and not end users.

A recent paper by Johnson and Dewhirst [19] reports on a volumetric modeling system based on interactive computer graphics, but little detail of the graphic user interface is given.

User interface

• *Overview*

We believe the user interface should receive more attention than it has to date. It is a crucial factor in determining whether volume modeling systems will be accepted by mechanical designers.

Using GDP/GRIN, the mechanical designer sits at a computer graphics terminal interacting with 2D projections of the model. A model is built from the primitives shown in Fig. 1 and the swept polygons in Fig. 2. The user can view and interact from any angle with any four views displayed simultaneously. The system produces either perspective or parallel projections, although for reasons discussed below, parallel projections are used for all interactive work.

Figure 3 illustrates various methods of displaying the model of two intersecting cylinders. A routine called Merge creates a single polyhedron as the union, intersection, or difference of two arbitrary polyhedra, in this case two primitive cylinders. Figure 3(a) is the wire frame mode display of the merged cylinders. Note that the cylinders are solid and the portions of the surface of each cylinder internal to the other one have been eliminated. Figure 3(b) shows the results of standard hidden line removal. In Fig. 3(c), the facet lines of the cylinders have been removed to produce a figure more suitable for

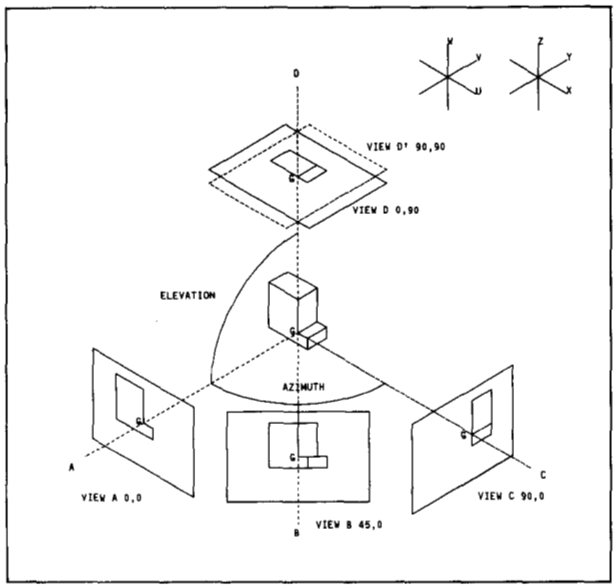

Figure 5 Defining views in GRIN.

inclusion in a publication. Figure 3(d) has the facet lines removed but includes hidden lines in dashed line type to illustrate that hidden features can be shown in a publication quality figure. This feature can be used in views at any angle, but its primary significance is that it enables the automatic production of correct orthographic views for engineering drawings, as shown in Fig. 4.

• *Viewing*

One of the requirements of a successful 3D design system is a quick, natural method of seeing the objects from different viewpoints. GRIN provides this convenience by (1) defining viewing angles and viewing translations independently, making them available to the user as separate commands, and (2) using orthographic projection.

The common projections for drawings are perspective, orthographic (including isometric, dimetric, and trimetric), and oblique [20]. The principal advantage of the last is that it is easiest to generate manually. Since the computer can produce any of the three and the first two appear more realistic to the eye, oblique projections are not used here. An orthographic projection is the limiting case of a perspective projection as the eyepoint is moved farther away from the object [21]. This reduces by one the number of viewing parameters (distance from the object) which must be specified by the user. When the eyepoint is close to the object, perspective projections tend to produce cluttered areas near the vanishing points. In addition, mechanical designers do not generally work with perspective projections. For these reasons orthographic projections are used.

The viewing angles are described for a simple notched block, as shown in the center of Fig. 5. If the eyepoint is at A looking at point G and the projection plane (screen of the display) is between these points and perpendicular to the line of sight, view A is produced. Similarly, as the eyepoint is changed to points B and C, still looking at G, their corresponding views are obtained. Point G is called the gaze point, and if azimuth and elevation are measured with respect to it, views can be identified by these two angles. A one-to-one relationship between the views and angle pairs is achieved by making the azimuth modulo 360 degrees and limiting the elevation between ±90 degrees. Note that at the north and south poles (elevation of ±90 degrees) different values of azimuth produce views D and D′. The viewing screen is wider than it is high, and it is the orientation of the screen with respect to the image (or vice versa) that is the difference between views D and D′. These two views are useful as top views for views A and C, respectively. The projection is displayed so that the image of G always appears at the center of the screen.

If the user wishes to change the point of interest, perhaps to part of the object which is off the screen, yet keep the same viewing angles, the image of G can be translated in the projection plane. This translation involves adding the same 2D vector to both the 3D eye and gaze points, which calculates a new gaze point without changing the azimuth or elevation.

The approach outlined above is embodied in a set of simple viewing commands which allow the user to change angle by rotating the eye (1) left, right, up, or down, (2) to an absolute azimuth or elevation, (3) to look along any coordinate system axis, or (4) to a standard isometric view. The eyepoint can be translated left, right, up, or down, or a new image center can be selected with the graphic cursor. In addition, the 2D image can be scaled.

• *Entry of models in three dimensional space*

As mentioned earlier, GRIN is the graphic input subsystem of a geometric modeling system that generates complex volumes by combining many instances of simple volumes. The major function of GRIN is to provide a quick, natural way to enter these simple volume primitives.

The model coordinate system is called the World Coordinate System (WCS). An axis indicator with six legs is displayed on the screen, indicating the orientation of the three mutually perpendicular axes (x, y, z) of the WCS and their negatives (see Fig. 1). The legs have the same

lengths in three dimensional space, but as the view point is changed, the orientations and lengths of the vectors in the axis indicator also change to reflect the new view point.

Because it may be convenient for the user to work at times in a coordinate system other than the WCS, a second coordinate system called the Rotated Coordinate System (RCS) can be defined interactively. Its directions (u, v, w) are shown on a second axis indicator (see Fig. 1). The RCS can be defined as a rotation about an axis of either coordinate system. It can also be specified with three points, the first two defining the origin and the u direction, the third defining the plane containing the u and v directions, with the w direction equal to the cross product of u and v.

Since primitives are entered by specifying points, methods for entering these elements are discussed first. The concept of a 3D "current point" is used for moving about in 3-space. A diamond is always displayed on the screen at the current point, which can be positioned in absolute coordinates or by pointing to an existing point with the graphic cursor. The user must ensure the uniqueness of the 2D projection of the chosen point by changing the view if necessary. The current point can also be moved relatively along any of the six axes a specified distance or be limited by a point. For example, in Fig. 6, if the current point is at F, moving 2 units along the y axis will move it to H, but moving it along y limited by point D will move it to G. The current point may be moved any number of times until the user is satisfied with its position in 3-space. A user-initiated command then accepts the coordinates of the current point, *e.g.*, to specify a positioning point of a primitive.

• *Entry of primitive volumes*

The user issues a command specifying the primitive to be added to the model. The point entry procedures are used to enter the points required to define that primitive. The primitive is displayed automatically after the last point has been entered. If it is not correct, the graphic modification commands may be used or the primitive may be canceled. Any other command will accept the primitive and make it a permanent part of the model.

The positioning points required for specifying primitives are shown in Fig. 1. Points must be specified in the order indicated. The primitive is then built to match the entered points. For example, when two points are entered for a hemisphere, the center is placed at the first point, the distance between the points determines the radius, and the hemisphere is oriented so its pole is at the second point. A cuboid can be entered by specifying the location

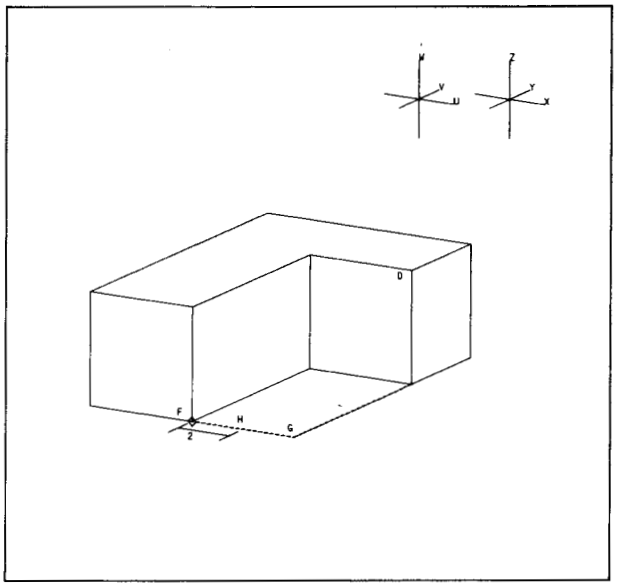

Figure 6 An illustration of moving the current point in 3-space.

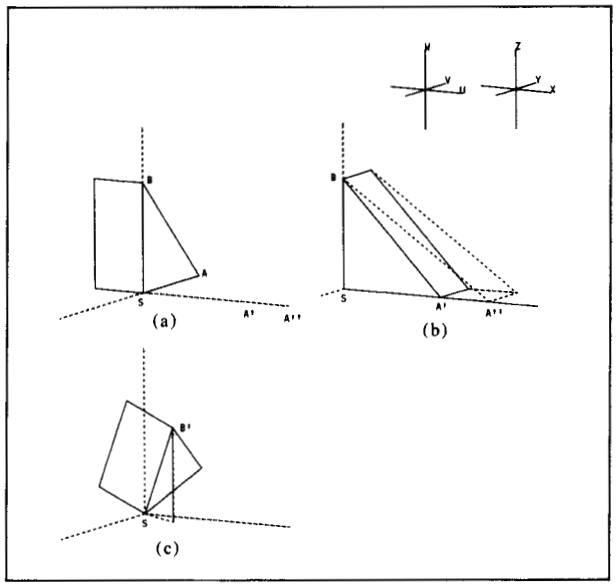

Figure 7 Defining the rotation of objects.

of one vertex and the three adjacent vertices which determine the height, width, length, and orientation.

GRIN frees the user from the requirement of maintaining perpendicularity among the specified edges. As shown in the cuboid of Fig. 1, point 3 can be anywhere on the unbounded line suggested by the dotted line through 3 and still provide the correct length and orientation for the associated edge. Point 4 need only be in the proper unbounded plane indicated by the two dotted lines con-

Table 1 Some GRIN commands.

Input commands	
CB	Enter a cuboid
CO	Enter a cone
CY	Enter a cylinder
HE	Enter a hemisphere
PT	Enter a translated polygon
PR	Enter a rotated polygon
Points and line segments (used within other commands)	
AB x y z	Enter absolute coordinates of a point
CU	Display cursor, to select an existing point
a ⟨n⟩	Move n units in the a direction from current point, where a is an axis (a = X, Y, Z, U, V, or W)
EN	Accept coordinates of current point
RJ	Reject last entered point or arc
ARC	Enter a circular arc in a polygon
Editing commands	
EO	Erase object
MO	Move object
ROX	Rotate object about an axis
ROP	Rotate object about a point
ROS	Rotate and scale object about a point
PO m	Reset polarity mode to m = H for hole, S for solid. For m = R, polarity of selected object is reversed, without changing mode.
F ⟨n⟩	Set the number of facets for new primitives. If defaulted, permits refaceting of a selected primitive.
RC a⟨-⟩n	Alter rotated coordinated system. Rotates n degrees ⟨counter⟩clockwise about the a axis (X, Y, Z, U, V, or W)
Display commands	
RL n, RR n, RU n, RD n	Rotate eyepoint left, right, up, or down n degrees
RI	Rotate to standard isometric angle
RX ⟨-⟩a	Rotate to axis a (X, Y, Z, U, V, or W)
RA n	Rotate to absolute azimuth of n degrees
RE n	Rotate to absolute elevation of n degrees
T	Translate so indicated point is centered on screen
TC	Automatically center picture and scale to fit screen
TL n, TR n, TU n, TD n	Translate eyepoint left, right, up, or down n units
DS ⟨n⟩	Scale smaller or larger by n. (Default is 2.)
DL ⟨n⟩	
DW	Set wire frame display mode
DH ⟨m⟩	Set hidden line display mode. For m = d, shows hidden lines dashed; for m = f, shows facet lines of curved surfaces; for m = df, shows both.
DR	Redraw display
DN ⟨m⟩	Display numeric values of coordinates of a point, or a variety of other data.
Miscellaneous	
CC	Cancel current command
IM	Merge new primitives
TX	Enter text on picture
MATHC	Enter desk calculator mode
FETCH n	Fetch a named model from storage
FILE n	File current model under specified name

necting that point to the volume in order to provide the correct length for the associated edge. If point 4 is specified on the side of the plane 123, which is the opposite of that shown in Fig. 1, the system will reflect the cuboid, so that the user need not concern himself with this reflection.

The first two points on the cylinder or cone specify its length and orientation. The third and fourth points specify radii. The dotted circular and straight lines through these points indicate that they can be anywhere on an unbounded cylindrical surface including the associated circle on the object.

• *Entry of swept polygons*

Volume primitives can also be created by sweeping polygons either in a straight line (as in an extrusion) or along the arc of a circle (to produce a solid of revolution) [14]. The polygon is defined by entering points as in Fig. 2(a). The arc of the polygon is entered by specifying its end points and center; the system approximates it with the straight lines indicated. To sweep in a straight line, a vector is entered, and a volume such as that shown in Fig. 2(b) is produced. To sweep along the arc of a circle, the axis of revolution and either a number of degrees or a starting and ending point are entered to produce a volume such as that of Fig. 2(c).

• *Graphic modification of volumes*

After a volume has been entered, it may be modified graphically. Modification is not limited to primitives, however. The graphical editing commands operate on any polyhedron at any node in the hierarchical tree of the model. If the editing of a polyhedron will invalidate higher level polyhedra, the user has the option of retaining or discarding the higher level polyhedra.

We use commands to move and rotate previously entered objects to illustrate modification. In these commands, the user first specifies the motion of the object(s). Motion for a move is specified by entering the end points of a vector in 3-space ("from" and "to" points). For rotation, the user specifies an axis of rotation and the amount of rotation about that axis, or rotation about a point may be specified by entering 3 points. For example, the polyhedron of Fig. 7(a) may be rotated to that of Fig. 7(b) by entering points S, A, and A′, or to that of Fig. 7(c) by S, B, and B′. Using the rotate and scale command, the polyhedron of Fig. 7(a) can be rotated and stretched to that of Fig. 7(b) by entering S, A, and A″. After the motion is specified, the first object to be modified is selected by pointing to it with the graphic cursor or typing its name if a previously assigned name is known. Subsequently selected objects will be moved or rotated through

the same relative motion as the first. Objects may also be modified to change their polarity to hole or solid and to change their names. Cylinders, cones, and hemispheres may be edited to change the number of facets used to approximate their curved surfaces.

• *GRIN commands*

Table 1 shows some of the GRIN commands. They are currently implemented by function keys on an alphanumeric keyboard and a joystick for graphic pointing, but are being converted to menu and light pen operation on the IBM 3250 Graphic Display Terminal.

A work session scenario

The experience of using an interactive 3D graphic system certainly cannot be conveyed as well with a written description as with a live demonstration. However, we use a series of figures in an attempt to give the reader a feeling for designing with computer graphics and volume modeling by illustrating the major steps performed in constructing a model of the complex part shown in Fig. 8.

Except for the appendage on the top, the part is symmetrical about the horizontal and vertical planes through the axis of the cavity in the part. Therefore, the creation of one quarter of the body is begun by constructing a four sided polygon in space and revolving it about the bottom edge to form one quarter of a truncated cone, as shown in Fig. 9. A fillet, blending the conical surface with a vertical planar surface (perpendicular to the y axis), is shown. It is entered by constructing a rectangle in a plane perpendicular to the z axis and revolving it about an axis parallel to the x axis. After the fillet material was created, it was translated and rotated in the vertical plane to give the best fit to the conical surface. Figure 9(b) is a view along the approximate line of tangency of the fillet and the cone. Figure 9(c) is a view along the axis of the part. The 3D viewing commands allow these views to be generated quickly and easily.

Negative volumes are shown in dashed line style. Fig. 10 is a top view of the merged primitives of Fig. 9, including a negative cuboid which is used to trim off the right end of the part. A negative translated polygon is used to trim off the left end and construct a fillet.

Figure 11 is the trimmed, filleted object of Fig. 10 with the addition of a quarter cylinder on the left and of a translated polygon which generates the ramp on the top of the part.

The negative volume of Fig. 12(a) is generated by rotating the polygon about the vertical edge at the left of the picture. Figure 12(b) results from removing the negative volume from the object of Fig. 11. This complex topology would be difficult for the designer to visualize and translate into a 2D engineering drawing.

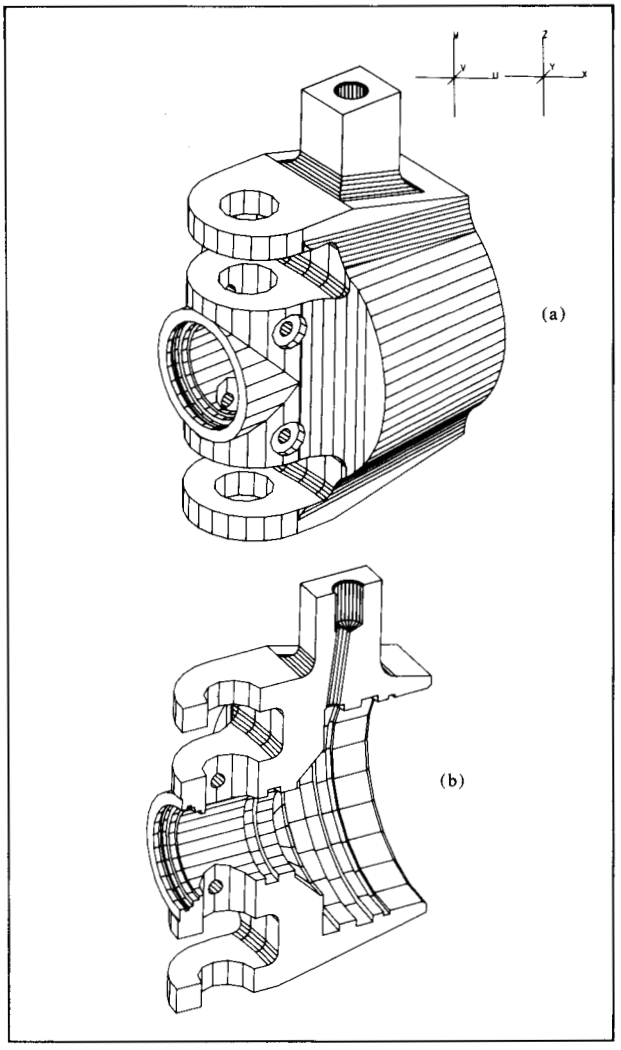

Figure 8 The completed part with section view.

The object of Fig. 13 was modeled as four instances of that shown in Fig. 12(b). A horizontal cylindrical cap has also been added to the vertical cylindrical surface.

Figure 14 shows the addition of a complex cavity to the part. Figure 14(a) is a side view of the negative volume defining the cavity. It was generated by constructing a complex polygon representing half the cross section of the cavity and revolving that polygon 360 degrees. Figure 14(b) is the result of removing the cavity from the part.

Figure 8 shows the completed part and a section view. It illustrates the complexity handled by the algorithms

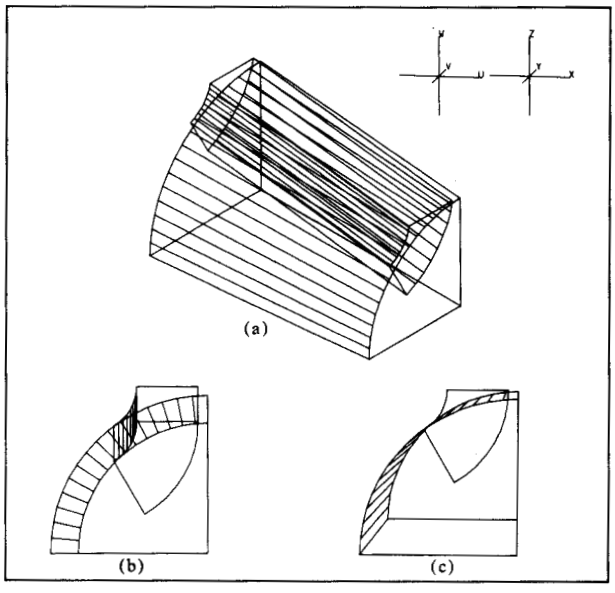

Figure 9 One quarter of a cone with fillet added.

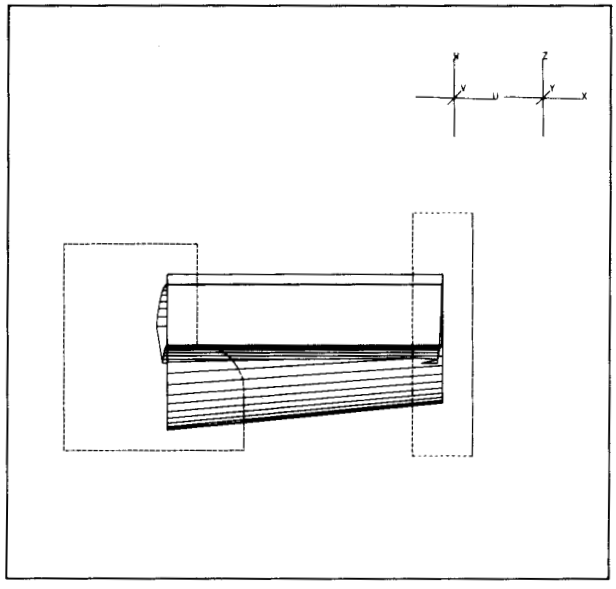

Figure 10 Trimming the object with translated polygons.

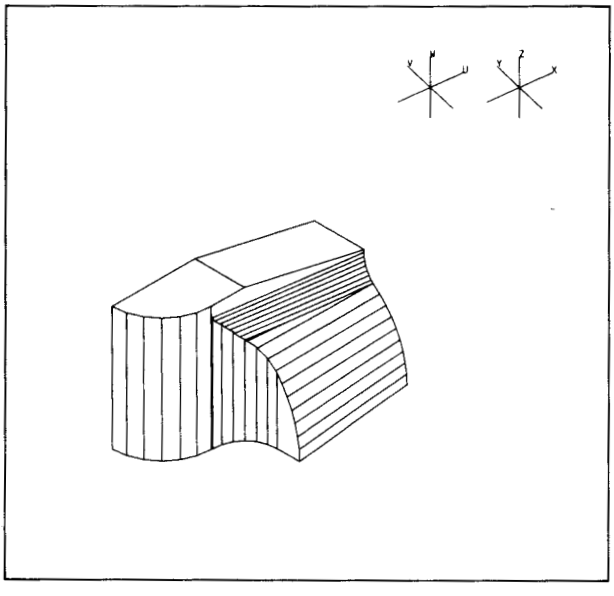

Figure 11 Ramp and vertical quarter cylinder added to the object.

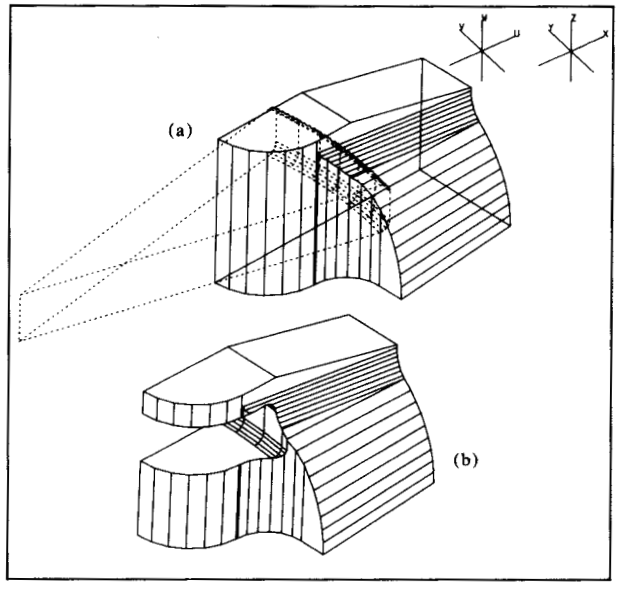

Figure 12 Removing material to produce a slot.

and demonstrates that a true volume model has been created and can be sliced to reveal inner material.

Key input algorithms

• *Algorithm for orienting and stretching cuboids*
The algorithm is described first for the cuboid, and then the variations are noted for other primitives. Figure 15(a) shows a cuboid of arbitrary size and orientation and a unit cube at the origin. Beginning with the unit cube as the master, the objective is to find the transformations necessary to convert a copy of the master into the orientation and size of the cuboid. As noted before, the user defines the cuboid by four points, denoted p', r', s' and t' in Fig. 15(a). The corresponding points on the master are unprimed.

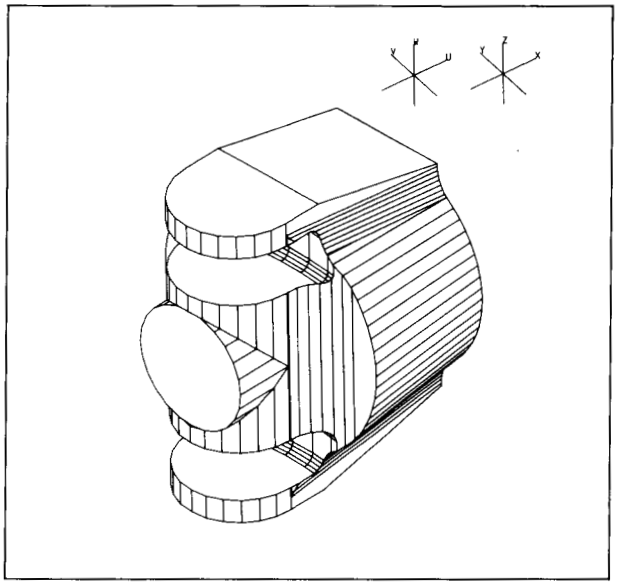

Figure 13 The result of reflecting and replicating objects in 3-space.

We first find the inverse transformation required to convert the primed points into the unprimed points and from these calculate the desired forward transformations. The translation of p' to $p = 0, 0, 0$ is just $-p'$, which can be represented [21] by a homogenous matrix $\mathbf{T}^{-1}$. Point r' is translated by $-p'$ to r'' ($r'' = r'\mathbf{T}^{-1}$) so that the next transformations take place with respect to the origin. Figure 15(b) shows that r' is rotated to the z axis to be consistent with r by first rotating $-a$ degrees about the z axis and then $-b$ degrees about the y axis. The rotation matrices $\mathbf{R}_a^{-1}$ and $\mathbf{R}_b^{-1}$ which represent these operations require the sines and cosines of these angles, which are

$$\sin a = \frac{r''_y}{f}, \cos a = \frac{r''_x}{f},$$

$$f = \sqrt{(r''_x)^2 + (r''_y)^2},$$

$$\sin b = \frac{f}{g}, \cos b = \frac{r''_z}{g},$$

$$g = \sqrt{(r''_x)^2 + (r''_y)^2 + (r''_z)^2}.$$

Next s' is put through these transformations to produce s'', which is then in a position consistent with the new positions of p' and r':

$$s'' = s'\mathbf{T}^{-1}\mathbf{R}_a^{-1}\mathbf{R}_b^{-1}.$$

Figure 15(c) shows that if s'' is rotated $-c$ degrees about the z axis, it will be consistent with s. This rotation matrix $\mathbf{R}_c^{-1}$ is the same as $\mathbf{R}_a^{-1}$ and is obtained by replacing r'' with s'' in the corresponding formulas.

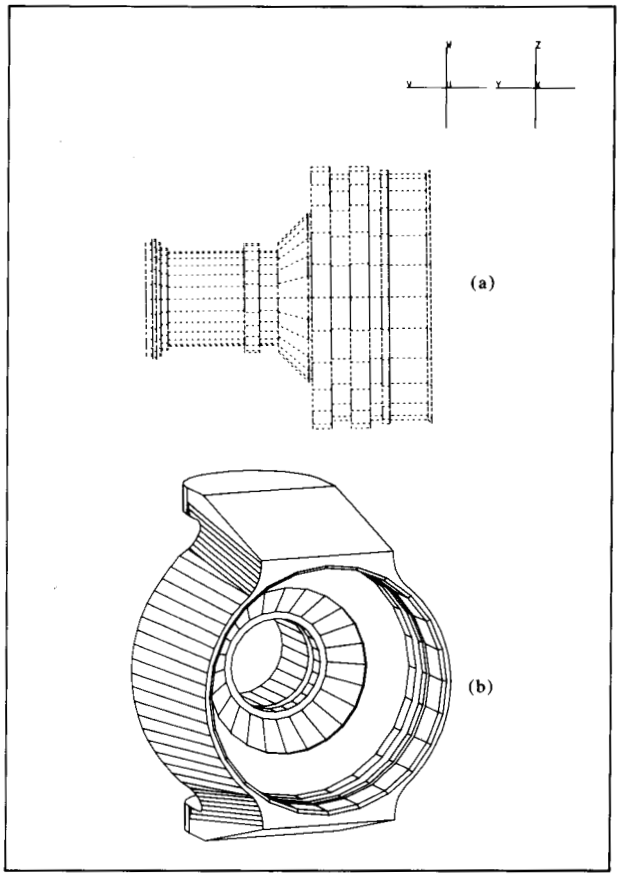

Figure 14 Removal of material to form a complex cavity in the part.

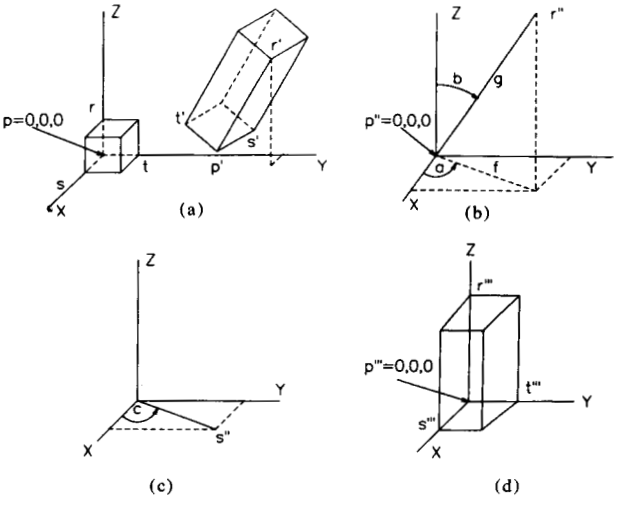

Figure 15 Orienting and stretching a cuboid.

If the vertices of the primed cuboid of Fig. 15(a) are represented by v' and each is put through these transformations

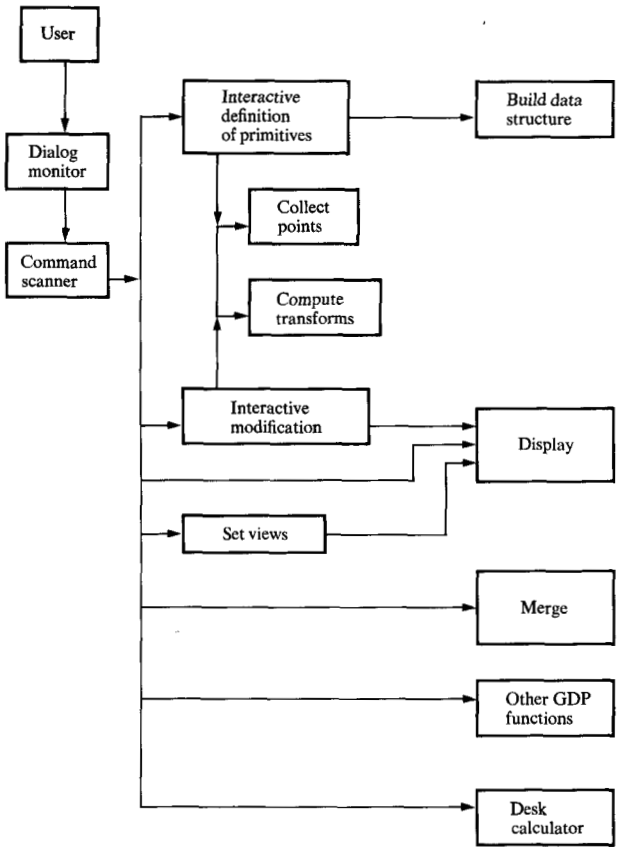

Figure 16 GRIN system architecture.

$$v''' = v'\mathbf{T}^{-1}\mathbf{R}_a^{-1}\mathbf{R}_b^{-1}\mathbf{R}_c^{-1},$$

the resulting orientation would be as shown in Fig. 15(d). The cuboid now only differs from the master in its size. The three dimensions are given by s_x''', t_y''', and r_z'''. These values are sent to a subroutine which generates a cuboid of this size. The vertices of that cuboid, v''', are then subjected to the forward transforms in the opposite order from that used above,

$$v' = v'''\mathbf{R}_c\mathbf{R}_b\mathbf{R}_a\mathbf{T},$$

to orient the cuboid as specified by the primed points in Fig. 15(a). If $t_y''' < 0$, the cuboid is to be reflected about the $y = 0$ plane by the matrix $\mathbf{F}$ and so

$$v' = v'''\mathbf{F}\mathbf{R}_c\mathbf{R}_b\mathbf{R}_a\mathbf{T}$$

is used instead.

It is not necessary for the user to specify all points on the corners of a cuboid. A rigid object has three degrees of freedom of translation and three degrees of freedom of rotation. The cuboid has in addition three degrees of freedom in its length, width, and height, a total of nine. Each of the four points p', r', s', and t' represents three constraints, totaling twelve. Three of these constraints are not used by the algorithm and can be used to ease the user's job. The coordinates s_z''', t_x''', and t_z''' are the ones not used. This accounts for the dotted lines associated with the cuboid shown in Fig. 1 and discussed earlier.

• *Algorithm for orienting and stretching other primitives*
The algorithm described for the cuboid is used for the other primitives with minor modifications. Reflection is not used for other primitives. In each case where they are used in Fig. 1, the pointings 1–4 determine p', r', s', and t', respectively. After 1 and 2 are put through the inverse transformation of the algorithm, the value of the z coordinate of 2 determines the height of the cylinder and cone and the radius of the hemisphere. Similarly, the value of the inverse transformed x coordinate of 3 [compare s_x''' in Fig. 15(d)] determines the radius of the cylinder and the radius of the base of the truncated cone. The magnitude of the inverse transformed x and y coordinates of 4 determines the other radius of the cone.

In the case of the translated polygon, the plane of the polygon is determined by its first point [1 on Fig. 2(a)] and the translation vector, which is normal to the plane of the polygon. The other points defining the polygon need not be in this plane but will be projected along the translation vector onto the plane before the primitive is constructed. Point 1 is used as p', and the other end of the translation vector is used as r'. This inverse transforms the plane of the polygon into the $z = 0$ plane, where the primitive is constructed. It is then transformed forward into the orientation specified by the user.

For the rotated polygon the first point of the polygon and the axis, points 1, 9, and 10 of Fig. 2(c), determine the plane of the polygon. Other points defining the polygon need not be on this plane, but will be rotated about the axis into the plane before the primitive is constructed. Points 9, 10, and 1 are used as p', r', and s', respectively, so that the polygon is inverse transformed into the $y = 0$ plane, where the primitive is constructed. The completed primitive is then transformed forward to the orientation specified by the user.

Parameterized objects and movement

In order to retain as much as possible of the power of a procedural language, interactive facilities have been included to (a) save and replay user input sequences with specific values substituted for symbolic names, (b) use symbolic names in place of constant data, and (c) define mathematical functions to be included in graphic commands. The commands are collected as they are entered and executed, so that the user sees graphically the consequences of the command being collected. Wherever

the user would normally enter a numeric value while generating a graphic model (*e.g.*, a coordinate, distance, angle, etc.), a symbolic name can be substituted. An interpretive facility has been included which permits the user to enter FORTRAN assignment statements, evaluate them as part of a command, and assign their value to a symbolic name. This facility allows parameterization, branching, and iteration (looping). Using these interactive facilities, a completely new graphical function, such as finding the intersection of two straight lines, could be added to the system without program recompilation. Examples of these facilities are described in the section on application feasibility. A similar facility implemented in APL for a 2D graphic system has been reported by Bleher *et al.* [22].

System architecture

The GDP/GRIN system runs under the VM/CMS Operating System. The user interacts with a dialog monitor (see Fig. 16), which collects user input and invokes a GRIN command scanner.

Of major importance are the GRIN commands to generate primitive volumes. These commands collect user input and call routines that generate the data structure and display the resulting object. Other commands perform modification of existing objects (move, rotate, refacet, erase, merge, etc.) or change the viewing parameters (scale, view type, viewing angle). Where a GRIN command requires function that already exists in GDP, the corresponding GDP routine is used where practicable. This is done for the generation and merging of polyhedra and for display with hidden lines removed. These functions were adapted, where necessary, for interactive rather than batch operation.

The hardware configuration consists of a standard IBM 3270 alphanumeric display equipped with the Graphic Attachment feature, which allows a storage tube display such as a 19-inch Tektronix 618 to be attached. This results in a dual screen display station with graphic commands transmitted to the display head at a very high rate. The Graphic Attachment feature includes a graphic cursor controlled by a joystick. The display station is attached to an IBM System/370 Model 168.

Application feasibility demonstrations

Several applications in mechanical design have been examined, and the feasibility of implementing them with computer graphics, volume primitives, and a geometric modeling system has been demonstrated.

The bulldozer of Fig. 17(a) is an example of the use of volume modeling for mechanism design. The object was to find the range of lift and tilt that could be applied to the blade without causing interference between members. This range determines the extensions required of the piston-cylinder pairs, which cause the blade movement.

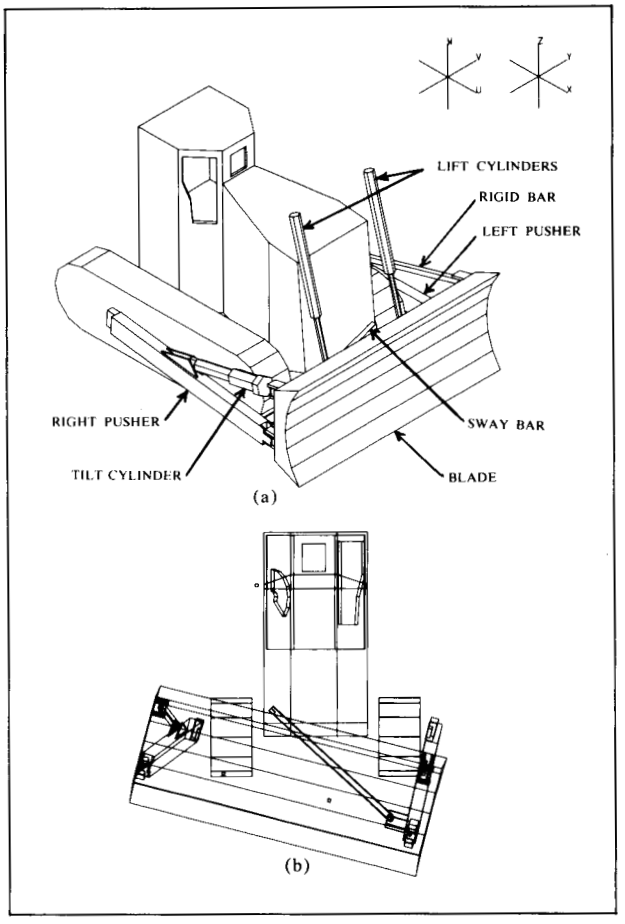

Figure 17 (a) A bulldozer blade mechanism design problem. (b) Results of blade lift and tilt motion.

The end of the sway bar visible in Fig. 17(a) is attached to the chassis, and the other end is attached to the right pusher. This complicates the motion by causing the blade to move horizontally as it is driven vertically by the lift cylinders.

The traditional solution to this problem involves writing and solving the simultaneous equations which represent the constraints on the system. Our approach frees the mechanical designer from this task by using a powerful new "graphic programming by example" technique. A volume model was built with the labeled members in Fig. 17(a) as subparts. The system of mechanical constraints could then be modeled by a sequence of GRIN commands to rotate the appropriate subparts. Since the extent of the lift and tilt pistons was to be an output of the design, the

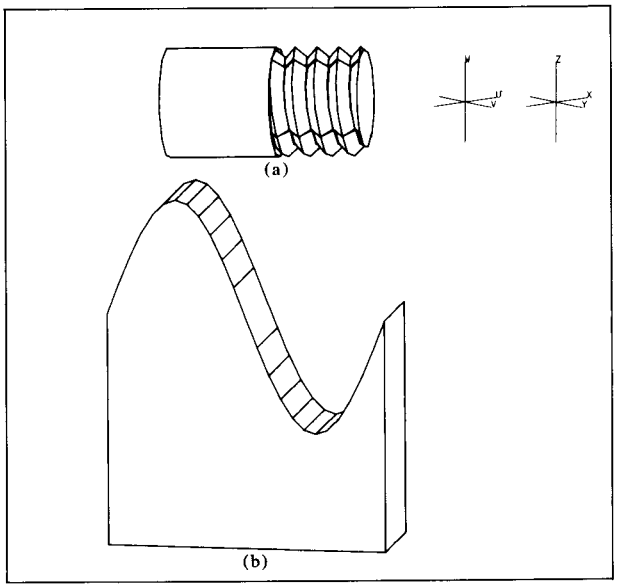

Figure 18 Objects produced from parameterized graphic dialog: (a) screw, (b) sinusoid.

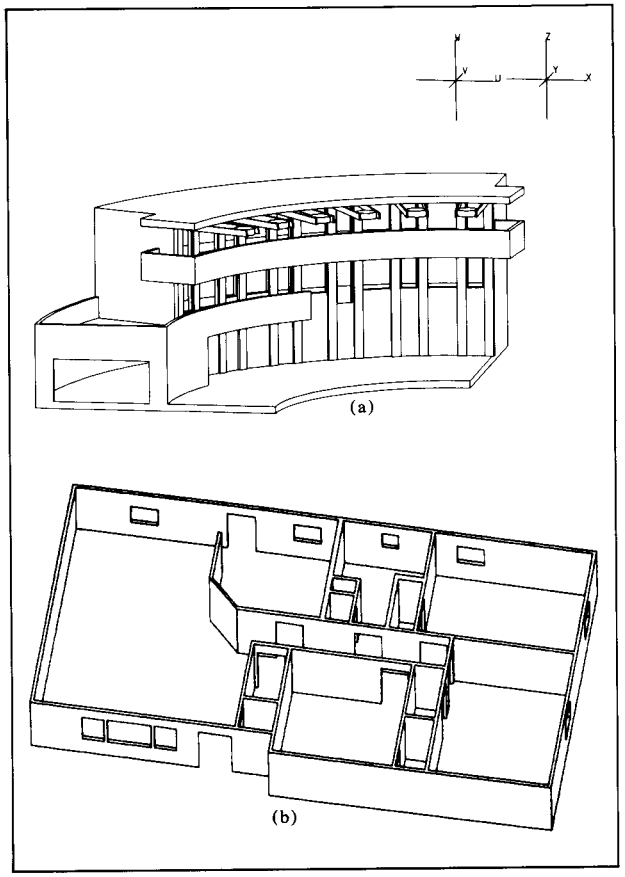

Figure 19 Examples of architectural applications using GDP/GRIN.

cylinders and associated pistons were not initially part of the model, providing two degrees of freedom in the mechanism (lift and tilt angles). For connected parts which must be rotated about different axes, iteration is required in order to preserve their connection. Iteration must continue until a solution of desired accuracy is obtained.

The sequence of rotation commands entered by the user to lift the blade and its connected members a specific number of degrees was collected as the commands were executed, so that the collection could be re-executed as a single command. Iteration and parameterization were subsequently introduced into the collected sequence of commands, producing a procedure for lifting an arbitrary number of degrees. A similar procedure for tilt was generated.

After rotation, analyses of various kinds can be performed on the model. The interference routine can be invoked to find that there is interference between the right pusher and track in the position shown in Fig. 17(b). At any position, the length to be spanned by each piston-cylinder combination can be measured digitally, completing the design process.

The volume model allows the designer to activate the mechanism as though it were a physical model, which helps him to visualize the consequences of changes, while providing him with the accuracy of a digital computer representation.

Figure 18 shows examples of parameterized objects containing a sinusoidal surface and a thread. A specific model of each type was built using GRIN and CMS interpreted commands. The commands were collected during execution and parameters added to generalize them. The user can now build a customized sinusoidal surface by invoking the resulting program by name and specifying the number of degrees per step and the number of steps. This causes the collected commands to be re-executed and the volume model to be built. The model can then be scaled independently in each axis, translated, and rotated as desired. To build the basic thread the user invokes the collection of commands by name and supplies as parameters the OD, pitch, length, and number of facets. The bottom of the thread has been cut off and a piece of cylinder added to the top to produce the model shown in Fig. 18(a).

Examples of the use of GDP/GRIN for architectural applications are shown in Fig. 19. Using the GRIN viewing commands, the user could take a simulated walk through the houses.

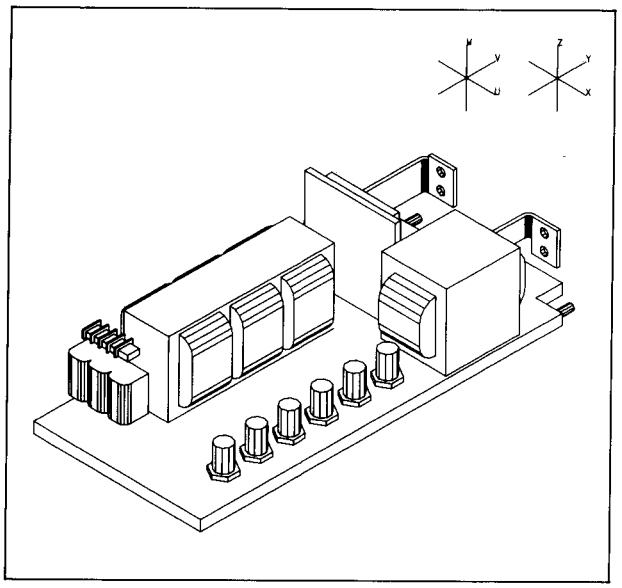

Figure 20 An example of power supply mechanical design using GDP/GRIN.

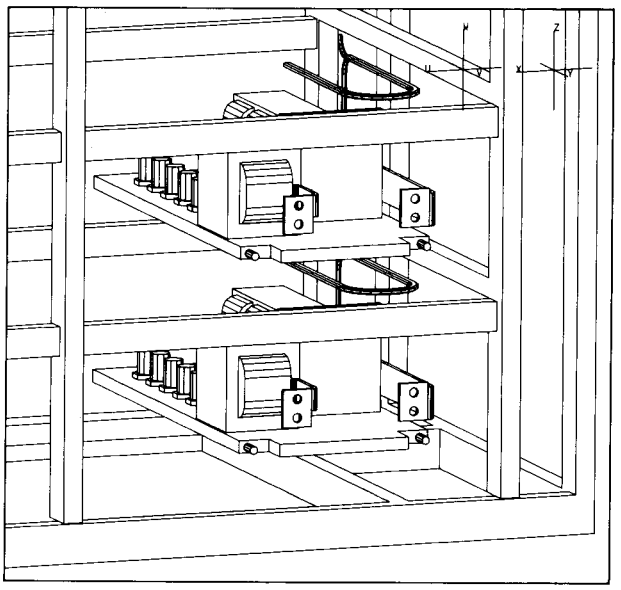

Figure 21 Three dimensional cable design using GDP/GRIN.

Within IBM, the use of geometric modeling for the design of computer hardware is being investigated. Figures 20 and 21 show the results of feasibility demonstrations in power supply mechanical design and three dimensional cable routing. Figure 22 illustrates that the system is able to produce color graphics output in checking for interference among a welded frame, cable (in green on the display), and piping (in blue on the display) for water cooled power frames.

Conclusions

Several questions posed in the introduction can now be addressed. The first asks whether a simple protocol between a designer and a computer graphics system can generate the data required by a geometric modeling system. The answer has been shown to be positive for some very complex models.

Another question asked whether mechanical designers would be able to think and work in terms of volume primitives on a 2D graphic screen. The section on application feasibility reports several nontrivial applications which have been demonstrated. The system has had a number of users other than the developers. Although GDP/GRIN is an experimental research tool, it is now being used in a pilot production system for mechanical designers in one of IBM's operating divisions. Their reactions to date have been positive.

The success of the applications described in the preceding section seems sufficient to encourage further development and testing to determine the economics and extent of penetration of this approach into mechanical design. Regardless of these results, it seems clear that computer graphics is an important, probably indispensable, component of a geometric modeling system for mechanical design.

Acknowledgments

We would like to acknowledge the contributions of the members of the Automation Research group at the Thomas J. Watson Research Center for the use of GDP and many graphic programs upon which GRIN was built. We are also indebted to Arthur Stein and Stephen Rosenthal of the Graphic Systems group for the device-independent basic graphic support which allows us to run on different types of vector graphics displays and the color display system. We wish to thank Ivan Kijac and Robert Zwissler for building the models shown in Figs. 19(a) and 19(b), respectively. We are indebted to David Grossman and Michael Wesley for suggestions which resulted in substantial improvements in the content and organization of this paper. Finally, we would like to thank C. K. Chow, whose continuous support over the past three years made this work possible.

References

1. J. P. Roth, *Computer Logic, Testing and Verification*, Computer Science Press, Potomac, MD, 1980.
2. M. A. Wesley, T. Lozano-Perez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, "A Geometric Modeling System for Automated Mechanical Assembly," *IBM J. Res. Develop.* **24**, 64–74 (1980).
3. M. A. Wesley, "Construction and Use of Geometric Models," *Computer Aided Design: Lecture Notes in Computer Science No. 89*, Chapter 2, J. Encarnacao, Ed., Springer-Verlag, New York, 1980.

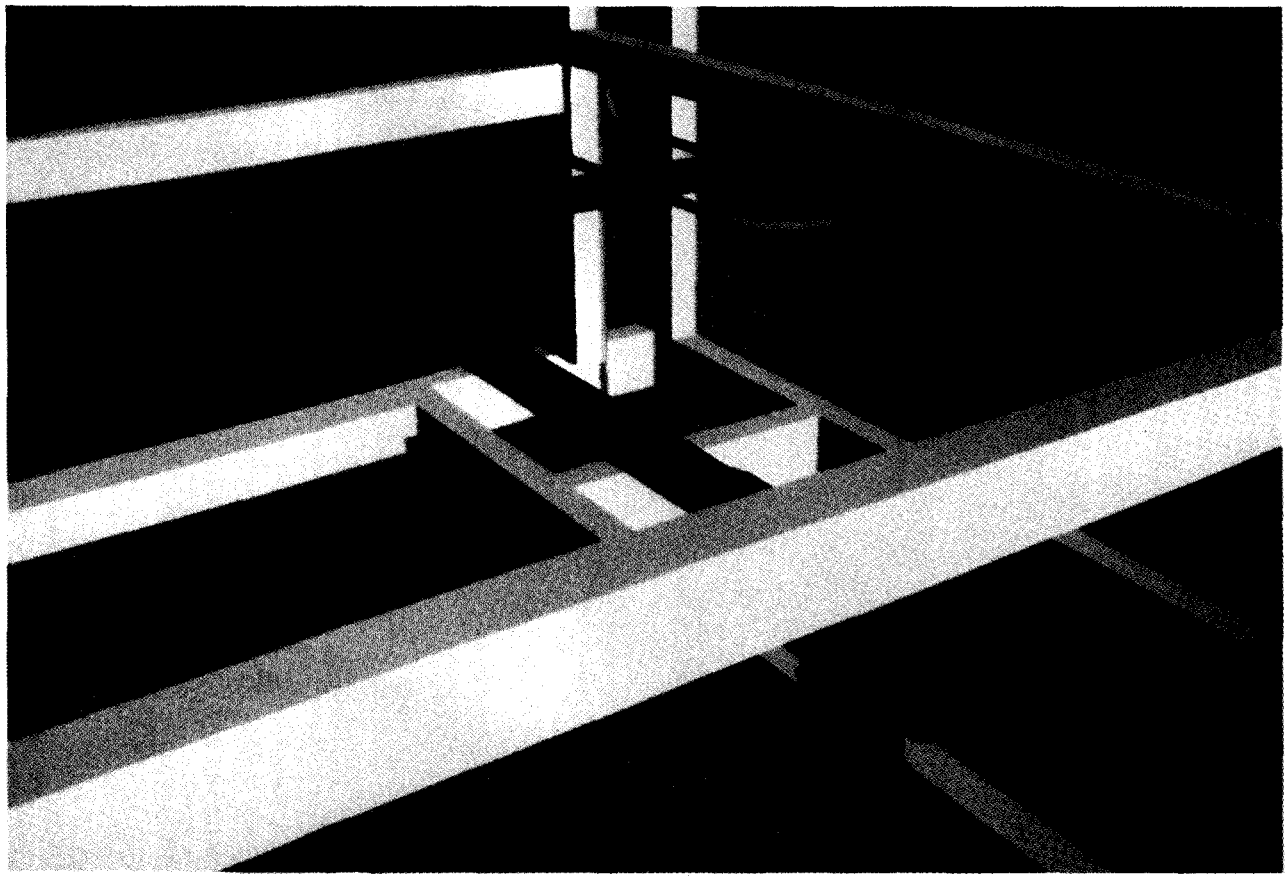

Figure 22 An example of colored, shaded output on a raster TV display showing the integration of cabling and piping in a computer frame.

4. T. Lozano-Perez and M. A. Wesley, "An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles," *Commun. ACM* **22,** 560–570 (1979).
5. L. I. Lieberman, "Model Driven Vision for Industrial Automation," presented at the IBM International Symposium on Advances in Digital Image Processing, Bad Neuenahr, W. Germany, September 26–28, 1978.
6. D. D. Grossman, "Procedural Representation of Three-dimensional Objects," *IBM J. Res. Develop.* **20,** 582–589 (1976).
7. S. A. Coons, "An Outline of the Requirements for a Computer Aided Design System," *Proc. Spring Joint Comp. Conf.*, Spartan Books, Baltimore, MD, 1963, p. 299.
8. I. E. Sutherland, "Sketchpad: A Man-Machine Graphical Communication System," *Proc. Spring Joint Comp. Conf.*, Spartan Books, Baltimore, MD, 1963, p. 329.
9. L. G. Roberts, "Machine Perception of Three Dimensional Solids," *MIT Lincoln Laboratory TR315*, Massachusetts Institute of Technology, Cambridge, MA, May 1963.
10. S. H. Chasen, "The Introduction of Man Computer Graphics into the Aerospace Industry," *Proc. Fall Joint Comp. Conf.*, Spartan Books, Washington, DC, 1965, p. 883.
11. W. S. Elliott, "Interactive Graphical CAD in Mechanical Engineering," *Computer Aided Design* **10,** 91–99 (1978).
12. T. C. Woo, "'Progess in Shape Modeling," *Computer*, 40–46 (December 1977).
13. I. C. Braid, *Designing With Volumes*, Cantab Pess, Cambridge, England, 1974.
14. B. G. Baumgart, "Geometric Modelling for Computer Vision," *Stanford Artificial Intelligence Laboratory Rep. STAN-CS 74-463*, Stanford, CA, 1974.
15. H. B. Voelcker and A. A. G. Requicha, "Geometric Modeling of Mechanical Parts and Processes," *Computer*, 48–57 (December 1977).
16. R. L. Gregory, "Techniques and Apparatus for the Study of Visual Perception," *Sci. Prog. Oxf.* **58,** 358–378 (1970).
17. J. H. Clark, "Designing Surfaces in 3-D," *Commun. ACM* **19,** 454–460 (1976).
18. A. Baer, C. Eastman, and M. Henrion, "Geometric Modelling: A Survey," *Computer Aided Design* **11,** 253–272 (1979).
19. R. H. Johnson and D. L. Dewhirst, "Machine Layout with Volumetric Models," *SAE International Congress and Exposition*, Detroit, MI, February 1981.
20. W. J. Luzadder, *Basic Graphics*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1968.
21. W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill Book Co., Inc., New York, 1979.
22. J. H. Bleher, P. G. Caspers, H. H. Henn, and K. Maerker, "A Graphic Interactive Application Monitor," *IBM Syst. J.* **19,** 382–402 (1980).

Received September 12, 1980; revised February 18, 1981

The authors are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.