DRAFT *** DRAFT *** DRAFT
Users Manual for
BRL-CAD Graphics Editor
MGED

Keith A. Applin
Michael J. Muuss
Robert J. Reschly
US Army Ballistic Research Laboratory
Aberdeen Proving Ground, MD

The Applications Development Branch
Computer Services Division
US Army Foreign Science and Technology Center
Charlottesville, VA

Mike Gigante
Ian Overend
The Royal Melbourne Institute of Technology
Australia

6-October-1988

Contents

1 INTRODUCTION 12
1.1 Philosophy . . . 13
1.2 Displays Supported . . . 14
1.3 Portability . . . 15
1.4 Object-Oriented Design . . . 15

2 THE COMBINATORIAL GEOMETRY METHODOLOGY 17
2.1 Background . . . 17
2.2 Directed Acyclic Graph and Database Details . . . 18
2.3 Model Building Philosophy . . . 19

3 THE BASIC EDITING PROCESS 20
3.1 Interaction Forms . . . 20
3.2 The Faceplate . . . 20
3.3 The Screen Coordinate System . . . 23
3.4 Changing the View . . . 25
3.5 Model Navigation . . . 28
3.6 Editor States . . . 28
3.7 Model Units . . . 29

4 PERIPHERAL DEVICES 30
4.1 Joystick . . . 30
4.2 Button Box . . . 30
4.2.1 Vector General Buttons . . . 30
4.2.2 Megatek Buttons . . . 32
4.2.3 Silicon Graphics Buttons . . . 34
4.3 Knobs (Dials) . . . 34
4.3.1 Vector General Knobs . . . 34
4.3.2 Megatek Knobs . . . 34
4.3.3 Silicon Graphics Knobs . . . 34
4.4 Mouse or Data Tablet . . . 36
4.4.1 Vector General Data Tablet . . . 36

4.4.2 Megatek Data Tablet 37
4.4.3 Silicon Graphics Mouse 37
4.4.4 Sun Workstation Mouse 38
4.5 Keyboard 38

5 OPERATING INSTRUCTIONS 39
5.1 Entering the Graphics Editor 39
5.1.1 Running MGED on a Silicon Graphics 40
5.1.2 Running MGED on a Tektronix 40
5.2 The Pop-Up Button Menu 41
5.3 Starting Your Model 41

6 CREATING NEW OBJECTS 43
6.1 Creating New Leaves (Solids/Primitives) 43
6.2 Specific Cases 44
6.2.1 RPP (rectangular parallelepiped) 44
6.2.2 BOX (BOX) 46
6.2.3 ARB8: Arbitrary Convex Polyhedron, 8 Vertices 48
6.2.4 ARB4: Arbitrary Convex Polyhedron, 4 vertices 49
6.2.5 RCC (Right Circular Cylinder) 49
6.2.6 TRC (Truncated Right Cylinder) 50
6.2.7 RAW (Right Angle Wedge) 51
6.2.8 SPH (Sphere) 53
6.2.9 ELLG (General Ellipsoid) 53
6.2.10 TOR (Torus) 54
6.3 Creating New Combinations 55

7 VIEWING FUNCTIONS 56
7.1 Preset Views 56
7.2 View Translation 56
7.3 View Zooming 57
7.4 The Angle Distance Cursor (ADC) 57

8 MGED EDITING FEATURES 60
8.1 Combination Editing (OBJECT EDIT) 60
8.2 Selecting Objects For Editing 61
8.3 Object Edit State 62
8.4 Translate An Object 62
8.5 Rotate An Object 62
8.6 Scale An Object 62
8.6.1 Global Scale 62
8.6.2 Local Scale 63
8.7 Solid Editing 63
8.8 Selecting Solids For Editing 64

8.9 Solid Edit State 64
8.10 Rotate A Solid 65
8.11 Translate A Solid 65
8.12 Scale A Solid 66
8.13 Solid Parameter Editing 66
8.13.1 ARB Parameter Editing 66
8.13.2 Move ARB Edges 68
8.13.3 Move ARB Faces 68
8.13.4 Rotate ARB Faces 68
8.13.5 Truncated General Cone (TGC) Parameter Editing 75
8.13.6 Ellipsoid Parameter Editing 76
8.13.7 Torus Parameter Editing 76

9 KEYBOARD COMMANDS 79
9.1 Copy Object 79
9.2 Zap Screen 79
9.3 Drop objects from display screen 79
9.4 Move (rename) object 80
9.5 Set Local Working Units 80
9.6 Group objects 80
9.7 Create Region 80
9.8 Instance an object 81
9.9 Change Title of Database 81
9.10 Extrude 82
9.11 Remove members from Combination 82
9.12 List Object Information 82
9.13 Analyze Solid 83
9.14 Mirror Object 83
9.15 Create ARB8 83
9.16 Change Item (Ident) and Air codes of Region 84
9.17 Specify Material Properties 84
9.18 Edit Combination Record Info 84
9.19 Edit (Display) an object on the screen 84
9.20 Evaluated Display of Object on the screen 84
9.21 Zap screen and Display Object 84
9.22 Kill (delete) object from database 85
9.23 List Table of Contents 85
9.24 List Tree Tops 85
9.25 Make prototypical solid 85
9.26 Mirror ARB Face 85
9.27 Print Summary of Objects 85
9.28 Specify Numeric Parameter(s) 86
9.29 Release Current Display 86
9.30 Attach to Display Device 86

9.31 Numeric Object Rotation Edit 86
9.32 Scale Edited Object 87
9.33 Translate Edited Object 87
9.34 Fix Broken Hardware (sometimes) 87
9.35 Ray-Trace Current View 87
9.36 Emulate Knob Twist 87
9.37 Solid_Edit Named Solid 88
9.38 Illuminate Named Object 88
9.39 Rotate the View 88
9.40 Move Screen Center 88
9.41 Set View Size 89
9.42 Extended List of all Objects in Displaylist 89
9.43 Refresh Display 89
9.44 Print View Status 89
9.45 Simulate Button Press 89
9.46 Escape to the Shell 90
9.47 Get Short Help Listing 90
9.48 Get Long Help Listing 90
9.49 Exit (Quit) MGED 91
9.50 Copy and Translate TGC 91
9.51 Remove Object and All References 91
9.52 Remove Complete Tree 91
9.53 Add Color To Display 91
9.54 Edit Display Colors 92
9.55 Print Display Colors 92
9.56 Find Objects 92
9.57 Estimate Presented Area 92
9.58 Produce UNIX Plot 92
9.59 Text Edit Solid Parameters 93
9.60 Keyboard Input of Solid Parameters 93
9.61 Change View Azimuth, Elevation 93
9.62 Define Region Identifiers 93
9.63 Change Edge Direction 93
9.64 Prefix Object Names 93
9.65 Keep Objects in Another File 94
9.66 List Object Hierarchy 94
9.67 Create Inside Solid 94
9.68 Produce ASCII Summary of Solids 94
9.69 Produce ASCII Summary of Regions 95
9.70 Produce ASCII Summary of Idents 95
9.71 List Objects Paths 95
9.72 List Evaluated Path Solid 95
9.73 Evaluate Path and Copy Solid 96
9.74 Edit Objects Region Identifiers 96

9.75 List Object As Stored 96
9.76 List Regions With Given Ident 96
9.77 Create ARB Given 3 Points 97
9.78 Create ARB Given Point and Angles 97
9.79 Push Editing Down Paths 97
9.80 Check For Duplicate Names 97
9.81 Concat Files 98
9.82 Create Pseudo-Track 98
9.83 Define ARB Face 98
9.84 Define Plane Equation of ARB Face 98
9.85 Move (Rename) Everywhere 98
9.86 Miscellaneous Commands 99
9.87 UNIX-Plot Output 99
9.88 Ray-Tracing the Current View 99
9.89 Animation 100

10 TUTORIALS ON VIEWING AND STATES 104
10.1 States Within the Edit Process 105
10.2 Viewing State 105
10.3 Solid Pick State 109
10.4 Solid Edit State 112
10.5 Object Pick State 113
10.6 Object Path State 113
10.7 Object Edit State 116
10.8 Editing in the Plane of the Screen 118
10.8.1 Top View 119
10.8.2 Bottom View 119
10.8.3 Right View 119
10.8.4 Front View 125
10.8.5 35, 25 View 125

11 TUTORIALS ON EDITING SOLIDS 130
11.1 Solid Edit: A Six-Sided Polyhedron 130
11.1.1 Translate Operation 133
11.1.2 Rotate Operation 136
11.1.3 Scale Operation 137
11.1.4 Moving Edges 144
11.1.5 Extrude Command 144
11.2 Solid Edit: A Five-Sided Polyhedron 147
11.2.1 Translate Operation 151
11.2.2 Rotate Operation 152
11.2.3 Scale Operation 153
11.2.4 Move Edge Command 154
11.3 Solid Edit: Alter a Cylinder 158

11.3.1 Translate Operation 162
11.3.2 Rotate Operation 162
11.3.3 Scale Operation 167
11.3.4 Scale H Command 167
11.3.5 Scale A Command 170
11.3.6 Scale B Command 171
11.3.7 Scale C Command 172
11.3.8 Scale D Command 173
11.3.9 Move End H Command 174
11.3.10 Move End H (rt) Command 175
11.4 Solid Edit: Alter Ellipsoid 176
11.4.1 Translate Operation 178
11.4.2 Rotate Operation 180
11.4.3 Scale Operation 183
11.4.4 Scale A Command 183
11.4.5 Scale B Command 186
11.4.6 Scale C Command 187
11.5 Solid Edit: Alter Torus 188
11.5.1 Translate Operation 191
11.5.2 Rotate Operation 192
11.5.3 Scale Operation 195
11.5.4 Scale Radius 1 Command 196
11.5.5 Scale Radius 2 Command 197

12 TUTORIALS ON OBJECT EDITING 199
12.1 Object Editing a Six-Sided Polyhedron 199
12.1.1 Scale Operation 199
12.1.2 X, Y, and XY Move Operation 202
12.1.3 Rotate Operation 204
12.2 Object Editing an Ellipsoid 205
12.2.1 Scale Operation 205
12.2.2 Move Operations 205
12.3 Object Path and Object Edit 209
12.3.1 Organize the Primitives and Groups 209
12.3.2 Editing One Primitive 214
12.3.3 Editing a Group of Two Primitives 216
12.3.4 Editing Two Groups of Four Primitives 218

13 BUILDING A SET OF COORDINATE AXES 221

14 BUILDING A TIN WOODSMAN 224
14.1 Create Primitives 224
14.2 Copy Primitives and Set Up for Edit 226
14.3 Create Funnel Hat 226
14.4 Building the Head 231
14.5 Building the Collar 234
14.6 Building the Body 235
14.7 Building the Arms 237
14.8 Building the Legs 240
14.9 Building Regions 240

15 BUILDING A ROBOT ARM 245

16 RT MATERIAL TYPE, PROPERTIES, and COLOR 252

17 RAYTRACING YOUR CREATION 255

18 CONCLUSIONS 257

List of Figures

2.1 Basic Solid Types 17
2.2 Special-Case Solid Types 18

3.1 The MGED Editor Faceplate 21
3.2 The Pop-Up Button Menu 22
3.3 The Screen Coordinate System 24
3.4 An Engine Connecting Rod 25
3.5 Close-Up Connecting Rod, Showing Z-clipping 26

4.1 Vector General Button Assignments 31
4.2 Silicon Graphics Button Assignments 32
4.3 Megatek Buttons 33
4.4 Vector General Knob Assignments 35
4.5 Silicon Graphics Knob Assignments 35

6.1 Example RPP 45
6.2 Example BOX 46
6.3 Example ARB8 47
6.4 Example ARB4 48
6.5 Example Right Circular Cylinder 49
6.6 Example Truncated Right Cylinder 50
6.7 Example Right Angle Wedge 51
6.8 Example Sphere 52
6.9 Example General Ellipsoid 53
6.10 Example Torus 54

7.1 The Angle Distance Cursor 58

8.1 ARB Control Menu 67
8.2 Move Edge Menu for ARB8 69
8.3 Move Edge Menu for ARB4 70
8.4 Move Face Menu for ARB8 71
8.5 Move Face Menu for ARB4 72

8.6 Rotate Face Menu for ARB8 73
8.7 Rotate Face Menu for ARB4 74
8.8 Typical TGC During Parameter Editing 75
8.9 Ellipsoid Parameter Editing Menu 77
8.10 Torus Parameter Editing Menu 78

9.1 MGED Command Summary 1 101
9.2 MGED Command Summary 2 102
9.3 MGED Command Summary 3 103

10.1 "arb8" Top View 107
10.2 "arb8" Rotated View 108
10.3 MGED In Solid Pick State 110
10.4 MGED In Solid Pick with Two Solids 111
10.5 Solid Edit State 112
10.6 Object Pick State 114
10.7 Object Path Selection State 115
10.8 Object Edit State 117
10.9 A Top View of the Coordinate Axes 118
10.10 Translating from the Top View 120
10.11 A Bottom View of the Coordinate Axes 121
10.12 Translating from the Bottom View 122
10.13 A Right View of the Coordinate Axes 123
10.14 Translating from the Right View 124
10.15 A Front View of the Coordinate Axes 126
10.16 Translating from the Front View 127
10.17 An Oblique 35,25 View of the Coordinate Axes 128
10.18 Translating in the 35,25 View 129

11.1 Top View of ARB8 131
11.2 A Rotated View of the ARB8 132
11.3 An ARB8 in Solid Edit State 134
11.4 Translating ARB8 Point 1 to the Origin 135
11.5 Solid Edit Rotation of 45 Degrees about X 136
11.6 Solid Edit Rotation of 45 Degrees about Y 138
11.7 Solid Edit Rotation of 45 Degrees about Z 139
11.8 ARB8 Scale Increased by 2X 140
11.9 ARB8 Edge 15 Moved Through (9, -2, -2) 141
11.10 ARB8 Edge 12 Moved Through (2, 5, -2) 142
11.11 ARB8 Edge 14 Moved Through (2, -2, 7) 143
11.12 ARB8 Rear Face Extruded 5 Units in -Z 145
11.13 ARB8 Rear Face Extruded 3 Units in +Z 146
11.14 Top View of an ARB5 148
11.15 A Rotated View of the ARB5 149

11.16 The ARB5 in Solid Edit State 150
11.17 Translating an ARB5 151
11.18 ARB5 Solid Edit Rotation about X 152
11.19 ARB5 Scale Increased by 2X 153
11.20 ARB5 Edge 14 Moved Through (1, 1, 1) 154
11.21 ARB5 Point 5 Moved to (-1.5, 1, 1) 155
11.22 ARB5 Edge 45 Moved Through (-1.5 1 1) 156
11.23 ARB5 Edge 12 Moved Through (2, 1, 2) 157
11.24 Top View of a Cylinder 159
11.25 A Rotated View of the Cylinder 160
11.26 A Cylinder in Solid Edit State 161
11.27 Translating Cylinder Vertex to (1, 1, 1) 163
11.28 Solid Edit Rotation of 45 Degrees about X 164
11.29 Solid Edit Rotation of 45 Degrees about Y 165
11.30 Solid Edit Rotation of 45 Degrees about Z 166
11.31 Cylinder Scale Increased by 1.5X 168
11.32 Cylinder Scale H Vector 169
11.33 Cylinder Scale A Vector 170
11.34 Cylinder Scale B Vector 171
11.35 Cylinder Scale C Vector 172
11.36 Cylinder Scale D Vector 173
11.37 Cylinder Move End of H 174
11.38 Cylinder Move End of H & Rotate 175
11.39 Top View of an Ellipsoid 176
11.40 An Ellipsoid in Solid Edit State 177
11.41 Translating Ellipsoid to (-1, 1, 1) 179
11.42 Solid Edit Rotation of 45 Degrees about X 180
11.43 Solid Edit Rotation of 45 Degrees about Y 181
11.44 Solid Edit Rotation of 45 Degrees about Z 182
11.45 Ellipsoid Scale Decreased 184
11.46 Ellipsoid Scale A Vector 185
11.47 Ellipsoid Scale B Vector 186
11.48 Ellipsoid Scale C Vector 187
11.49 Top View of a Torus 189
11.50 The Torus in Solid Edit State 190
11.51 Translating a Torus 191
11.52 Torus Solid Edit Rotation about X 192
11.53 Torus Solid Edit Rotation about Y 193
11.54 Torus Solid Edit Rotation about Z 194
11.55 Torus Scale Increased 195
11.56 Scale Torus Radius 1 196
11.57 Scale Torus Radius 2 197

12.1 “arb8” Object Edit; Top View 200

12.2 "arb8" Object Edit; Scaled by 0.5 201
12.3 "arb8" Object Edit; Translated to (0.5, -2, 1.5) 202
12.4 XY Move 203
12.5 "arb8" Rotated by (30, 45, 60) 204
12.6 Object Edit; Ellipse viewed from 45,45 preset 206
12.7 Object Edit; Ellipse scaled up by 2 207
12.8 Object Edit; Ellipse Translated 208
12.9 Stacked Primatives 210
12.10Primatives After Translation 211
12.11Object Path With "tor.s" as Reference Solid 213
12.12Object Edit With "tor.s" as Reference Solid 214
12.13Object Edit Affecting Torus Only 215
12.14Torus and Ellipsoid Selected for Object Edit 216
12.15Torus and Ellipsoid Translated by (3, 1, 1) 217
12.16All Primitives Selected for Object Edit 218
12.17All Primitives Translated by (3, 2, 1) 219

13.1 The Model Axes Viewed from 35,25 222
13.2 Example Axes Viewed form 35,25 223

14.1 WoodsMan Primitives 225
14.2 Funnel Bowl Cylinder After Rotation 227
14.3 Funnel Bowl Cylinder After End Scaling 228
14.4 Funnel Bowl Cylinder After Moving 229
14.5 Funnel Tube Scaled and Positioned 230
14.6 Head Sphere 232
14.7 The Woodsman's Collar 233
14.8 The Woodsman's Body 234
14.9 An Upper Arm Prototype 235
14.10The Woodsman's Arms 236
14.11The First Leg 238
14.12The Tin Woodsman 239
14.13Evaluation of Funnel Hat Region 243

15.1 The RMIT Robot Arm 246

Chapter 1

INTRODUCTION

Computer graphics is one of the fastest growing fields in the computer industry. Computer graphics has applications in many diverse areas, from electronic games to medicine; from cartoons to the space industry. Just what is interactive computer graphics and why is it so versatile? Human visual perception is quite keen and communicating with a computer is generally faster and easier with images, rather than with numbers. Furthermore, by having the computer continuously updating a display, the display itself becomes the communications medium. The user converses with the computer through the display using devices such as light pens, mice, data tablets, buttons, and knobs. The response of the computer is immediately reflected on the display, providing a fast communication channel between person and machine. This technology is called interactive computer graphics.

As the Army's lead laboratory for vulnerability technology, the Ballistic Research Laboratory (BRL) constantly performs analyses for a wide variety of military systems. Three dimensional computer models of the physical characteristics of these systems are vital to these studies. Since the mid-1960's, BRL has used a solid modeling technique called Combinatorial Solid Geometry (CSG or COMGEOM) for representing these models. The COMGEOM technique uses Boolean logic operations to combine basic geometric shapes or primitives to produce complex three-dimensional objects. The COMGEOM geometric models are processed by the Geometric Information For Targets (GIFT) [BR75, KBR79] and LIBRT [Muu87] for use in follow-on engineering analysis.

Geometric models are large collections of numerical data which have traditionally been created and edited manually, and analyzed in a batch environment. The production and modification of geometric models has been a slow, labor-intensive process. In 1980, BRL initiated an effort to improve the response time of the geometric modeling process by applying interactive computer graphics techniques. As a result of this work, BRL developed the Multi-device Graphics EDitor (MGED), an interactive editor for solid models based on the COM-

GEOM technique. Using MGED, a designer can build, view, and modify model descriptions interactively by manipulating the graphical representation, receiving immediate visual feedback on a graphics display. MGED replaces the manual method for the production and modification of geometric models.

Before MGED was built, existing packages were evaluated with respect to their utility for the geometric modeling process. Quite an exhaustive search of commercially available systems was conducted and none were found which met the BRL requirements. A study was initiated to examine the feasibility of producing the required capability in-house; a preliminary version of MGED which quite convincingly demonstrated the feasibility of such an undertaking [WM80]. It was then decided to develop MGED into a full production code. Production MGED code has been used since January 1982 to build models interactively at BRL.

This report is intended to serve as a user manual for the MGED program. The process of viewing and editing a description using MGED is covered in detail. The internal data structure is also covered, as it is an important part in the overall design. All the commands will be discussed and a command summary table presented. Also, a section will be devoted to the hardware interfaces for each major class of workstations which MGED supports.

1.1 Philosophy

The role of CAD models at BRL differs somewhat from that of CAD models being built in the automobile and aerospace industries, resulting in some different design choices being made in the BRL CAD software. Because BRL's main use for these models is to conduct detailed performance and survivability analyses of large complex vehicles, it is required that the model of an entire vehicle be completely contained in a single database suitable for interrogation by the application codes. This places especially heavy demands on the database software. At the same time, these analysis codes require less detail than would be required if NC machining were the primary goal.

At BRL, there are only a small number of primary designers responsible for the design of a vehicle, and for the construction of the corresponding solid model. Together they decide upon and construct the overall structure of the model, then they perform the work of building substructures in parallel, constantly combining intermediate results into the full model database. Because of the need to produce rapid prototypes (often creating a full design within a few weeks), there is no time for a separate integration stage; subsystem integration must be an ongoing part of the design process.

Once an initial vehicle design is completed, there is usually the need for exploring many alternatives. Typically, between three and twelve variations of each design need to be produced, analyzed, and optimized before recommendations for the final design can be made. Also, there is a constantly changing definition of performance; new developments may necessitate rapidly re-evaluating all the designs of the past several years for trouble spots.

The user interface is designed to be powerful and "expert friendly" rather than foolproof for a novice to use. However, it only takes about two days for new users to start doing useful design work with MGED. True proficiency comes with a few months practice.

Finally, it is vitally important that the software offer the same capabilities and user interface across a wide variety of display and processor hardware. Government procurement regulations make single-vendor solutions difficult. The best way to combat this is with highly portable software.

1.2 Displays Supported

It is important for a CAD system to have a certain degree of independence from any single display device in order to provide longevity of the software and freedom from a single equipment supplier. The MGED editor supports serial use of multiple displays by way of an object-oriented programmatic interface between the editor proper and the display-specific code. All display-specific code for each type of hardware is isolated in a separate *display manager* module. High performance of the display manager was an important design goal. Existing graphics libraries were considered, but no well established standard existed with the necessary performance and 3-dimensional constructs. By having the display manager modules incorporated as a direct part of the MGED editor, the high rates of display update necessary to deliver true interactive response are possible, even when using CPUs of modest power.

An arbitrary number of display managers may be included in a copy of MGED, allowing the user to rapidly and conveniently move his editing session from display to display. This is useful for switching between several displays, each of which may have unique benefits: one might have color capability, and another might have depth cueing. The **release** command closes out MGED's use of the current display, and does an implicit attach to the "null" display manager. This can be useful to allow another user to briefly examine an image on the same display hardware without having to lose the state of the MGED editing session. The **attach** command is used to attach to a new display via it's appropriate display manager routines. If another display is already attached, it is released first. The null display manager also allows the MGED editor to be run from a normal alphanumeric terminal with no graphic display at all. This can be useful when the only tasks at hand involve viewing or changing database structures, or entering or adjusting geometry parameters in numerical form.

Creation of a new display manager module in the "**C**" language [RJLK78] generally takes an experienced programmer from one to three days. The uniform interface to the display manager provides two levels of interactive support. The first level of display support includes the Tektronix 4014, 4016, and compatible displays, including the Teletype 5620 bit-mapped displays. However, while storage-tube style display devices allow MGED to deliver the correct functionality, they lack the rate of screen refresh needed for productive interaction. The second level of support, including real-time interaction, is provided by the Vector General 3300 displays, the Megatek 7250 and 7255 displays, the Raster Technologies Model One/180 display, the Evans and Sutherland PS300 displays with either serial, parallel, or Ethernet attachment, the Sun workstations, and the Silicon Graphics IRIS workstation family.

1.3 Portability

Today, the half-life of computer technology is approximately two to three years. To realize proper longevity of the modeling software, it needs to be written in a portable language to allow the software to be moved readily from processor to processor without requiring the modeling software or users to change. Then, when it is desirable to take advantage of the constantly increasing processor capabilities and similarly increasing memory capacity by replacing the installed hardware base, there are a minimum of ancillary costs. Also, it may be desirable to connect together processors from a variety of vendors, with the workload judiciously allocated to the types of hardware that best support the requirements of each particular application program. This distribution of processing when coupled with the fact that users are spread out over multiple locations makes networking a vital ingredient as well.

BRL's strategy for achieving this high level of portability was to target all the software for the UNIX operating system, [RT78], with all the software written in the "**C**" programming language [RJLK78]. The entire BRL-CAD Package, including the MGED editor is currently running on all UNIX machines at BRL, under several versions of the UNIX operating system, including Berkeley 4.3 BSD UNIX, Berkeley 4.2 BSD UNIX, and AT&T System V UNIX.

The list of manufacturers and models of CPUs that support the UNIX operating system [Dei84] is much too lengthy to include here. However, BRL has experience using this software on DEC VAX 11/750, 11/780, 11/785 processors, Gould PN6000 and PN9000 processors, Alliant FX/8 and FX/80 processors (including systems with eight CPUs), Silicon Graphics IRIS 2400, 2400 Turbo, 3030, 4-D, and 4-D/GT workstations, the Cray X-MP, the Cray-2, and the ill-fated Denelcor HEP H-1000 parallel supercomputer.

1.4 Object-Oriented Design

The central editor code has four sets of object-oriented interfaces to various subsystems, including database access, geometry processing, display management, and command parser/human interface. In each case, a common interface has been defined for the set of functions that implement the subsystem; multiple instances of these function sets can exist. The routines in each instance of a subsystem are completely independent of all the routines in other functions sets, making it easy to add new instances of the subsystem. A new type of primitive geometry, a new display manager, a new database interface, or a new command processor can each be added simply by writing all the routines to implement a new subsystem. This approach greatly simplifies software maintenance, and allows different groups to have responsibility for the creation and enhancement of features within each of the subsystems.

Chapter 2

THE COMBINATORIAL GEOMETRY METHODOLOGY

2.1 Background

Since the MGED system is presently based on the COMGEOM solid modeling technique, a brief overview of the COMGEOM technique is required to effectively use MGED. For more detailed information on the COMGEOM technique see [BR75, KBR79].

The COMGEOM technique utilizes two basic entities - a solid and a region. A solid is defined as one of fifteen basic geometric shapes or primitives. Figure 2.1 lists the basic solid types, and Figure 2.2 lists special cases of the basic solid types for which support exists. The individual parameters of each solid define

Symbol	Name
ARS	Arbitrary Triangular Surfaced Polyhedron
ARB	Arbitrary Convex Polyhedron
ELLG	General Ellipsoid
POLY	Polygonal Faceted Solid
SPL	Non-Uniform Rational B-Spline (NURB)
TGC	Truncated General Cone
TOR	Torus
HALF	Half Space (Plane)

Figure 2.1: Basic Solid Types

Symbol	Name
RPP	Rectangular Parallelpiped
BOX	Box
RAW	Right Angle Wedge
SPH	Sphere
RCC	Right Circular Cylinder
REC	Right Elliptical Cylinder
TRC	Truncated Right Cylinder
TEC	Truncated Elliptical Cylinder

Figure 2.2: Special-Case Solid Types

the solid's location, size, and orientation. A region is a combination of one or more solids and is defined as the volume occupied by the resulting combination of solids. Solids are combined into regions using any of three logic operations: union (OR), intersection (+), or difference (-). The union of two solids is defined as the volume in either of the solids. The difference of two solids is defined as the volume of the first solid minus the volume of the second solid. The intersection of two solids is defined as the volume common to both solids.

Any number of solids may be combined to produce a region. As far as the COMGEOM technique is concerned, only a region can represent an actual component of the model. Regions are homogeneous; they are composed of a single material. Each region represents a single object in the model; the solids are only building blocks which are combined to define the *shape* of the regions. Since regions represent the components of the model, they are further identified by code numbers. These code numbers either identify the region as a model component (nonzero item code) or as air (nonzero air code). Any volume not defined as a region is assumed to be "universal air" and is given an air code of "01". If it is necessary to distinguish between universal "01" air and any other kind of air, then that volume must be defined as a region and given an air code other than "01". Normally, regions cannot occupy the same volume (overlap), but regions identified with air codes can overlap with any region identified as a component (i.e. one that has a nonzero item code). Regions identified with different air codes, however, can not overlap.

2.2 Directed Acyclic Graph and Database Details

One of the critical aspects of a graphics software package is its internal data structure. Since geometric models often result in very large volumes of data being generated, the importance of the data structure here is emphasized. Thus it is felt that a brief introduction to the organization of the MGED database is important for all users.

The database is stored as a single, binary, direct-access UNIX file for efficiency and cohesion, with fixed length records called database *granules*. Each object occupies one or more granules of storage. The user sees and manipulates the directed acyclic graphs like UNIX paths (e.g., car/chassis/door), but in a global namespace. There can be many independent or semi-independent directed acyclic graphs within the same database, each defining different models. The figure also makes heavy use of the *instancing* capability. As mentioned earlier, the *leaves* of the graph are the primitive solids.

Commands exist to import sub-trees from other databases and libraries, and to export sub-trees to other databases. Also, converters exist to dump databases in printable form for non-binary interchange.

2.3 Model Building Philosophy

The power of a full directed acyclic graph structure for representing the organization of the database gives a designer a great deal of flexibility in structuring a model. In order to prevent chaos, most designers at BRL choose to design the overall structure of their model in a top-down manner, selecting meaningful names for the major structures and sub-structures within the model. Actual construction of the details of the model generally proceeds in a bottom-up manner, where each sub-system is fabricated from component primitives.

The first sub-systems to be constructed are the chassis and skin of the vehicle, after which a set of analyses are run to validate the geometry, checking for unintentional gaps in the skin or for solids which overlap. The second stage of model construction is to build the features of the main compartments of the vehicle. If necessary for the analysis codes that will be used, the different types of air compartments within the model also need to be described. The final stage of model construction is to build the internal objects to the desired level of detail. This might include modeling engines, transmissions, radios, people, seats, etc. In this stage of modeling, the experienced designer will draw heavily on the parts-bin of model components and on pieces extracted from earlier models, modifying those existing structures to meet his particular requirements.

Throughout the model building process it is important for the model builder to choose part names carefully, as the MGED database currently has a global name space, with individual node names limited to 16 characters. In addition, BRL has defined conventions for naming the elements in the top three levels of database structure, allowing people to easily navigate within models prepared at different times by different designers. This naming convention facilitates the integration of design changes into existing models.

Chapter 3

THE BASIC EDITING PROCESS

3.1 Interaction Forms

Textual and numeric interaction with the MGED editor is the most precise editing paradigm because it allows exact manipulation of known configurations. This works well when the user is designing the model from an existing drawing, or when all dimensions are known (or are computable) in advance.

The use of a tablet or mouse, knob-box or dial-box, buttons, and a joystick are all simultaneously supported by MGED for analog inputs. Direct graphic interaction via a "point-push-pull" editing paradigm tends to be better for prototyping, developing arbitrary geometry, and fitting together poorly specified configurations. Having both types of interaction capability available at all times allows the user to select the style of interaction that best meets his immediate requirements.

3.2 The Faceplate

When the MGED program has a display device attached, it displays a border around the region of the screen being used along with some ancillary status information. Together, this information is termed the editor "faceplate". See Figure 3.1. In the upper left corner of the display is a small enclosed area which is used to display the current editor state; this is discussed further in the Editor States section, below.

Underneath the state display is a zone in which three "pop-up" menus may appear. The top menu is termed the "button menu," as it contains menu items which duplicate many of the functions assigned to the button box. Having these

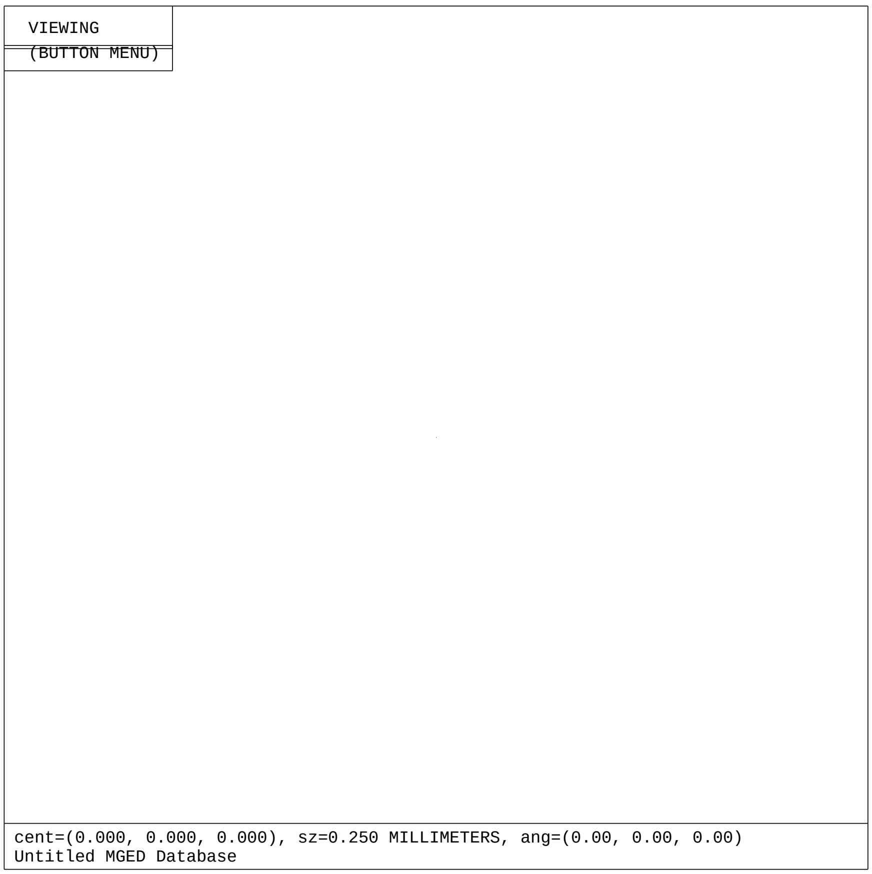

Figure 3.1: The MGED Editor Faceplate

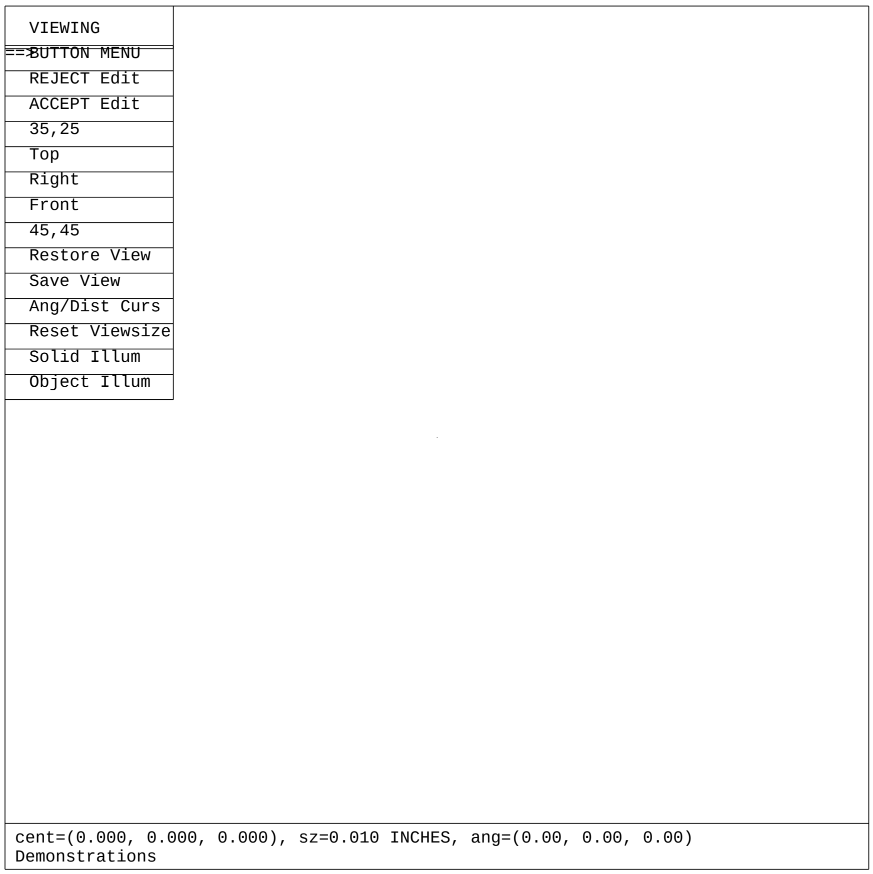

Figure 3.2: The Pop-Up Button Menu

frequently used functions available on a pop-up menu can greatly decrease the number of times that the user needs to remove his hand from the pointing device (either mouse or tablet puck) to reach for the buttons. An example of the faceplate and first level menu is shown in Figure 3.2. The second menu is used primarily for the various editing states, at which time it contains all the editing operations which are generic across all objects (scaling, rotation, and translation). The third menu contains selections for object-specific editing operations. The choices on these menus are detailed below.

It is important to note that while some display hardware that MGED runs on has inherent support for pop-up menus included, MGED does not presently take advantage of that support, preferring to depend on the portable menu system within MGED instead. It is not clear whether the slight increase in functionality that might accrue from using display-specific menu capabilities would offset the slight nuisance of a non-uniform user interface.

Running across the entire bottom of the faceplate is a thin rectangular display area which holds two lines of text. The first line always contains a numeric display of the model-space coordinates of the center of the view and the current size of the viewing cube, both in the currently selected editing units. The first line also contains the current rotation rates. The second line has several uses, depending on editor mode. Normally it displays the formal name of the database that is being edited, but in various editing states this second line will instead contain certain path selection information. When the angle/distance cursor function is activated, the second line will be used to display the current settings of the cursor.

It is important to mention that while the database records all data in terms of the fixed base unit of millimeters, all numeric interaction between the user and the editor are in terms of user-selected display [or local] units. The user may select from millimeters, centimeters, meters, inches, and feet, and the currently active display units are noted in the first display line.

The concept of the "viewing cube" is an important one. Objects drawn on the screen are clipped in X, Y, and Z, to the size indicated on the first status line. This feature allows extraneous wireframes which are positioned within view in X and Y, but quite far away in the Z direction to not be seen, keeping the display free from irrelevant objects when zooming in. Some display managers can selectively enable and disable Z axis clipping as a viewing aid.

3.3 The Screen Coordinate System

The MGED editor uses the standard right-handed screen coordinate system, as shown in Figure 3.3. The Z axis is perpendicular to the screen and the positive Z direction is out of the screen. The directions of positive (+) and negative (-) axis rotations are also indicated. For these rotations, the "Right Hand Rule" applies: Point the thumb of the right hand along the direction of +X axis and

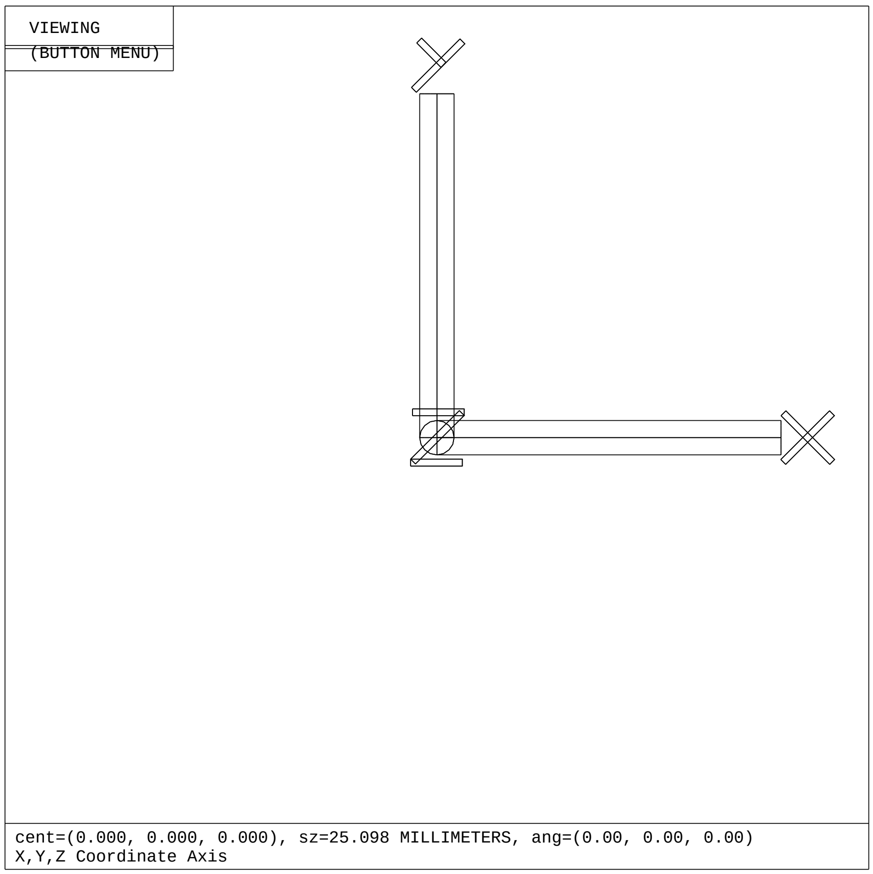

Figure 3.3: The Screen Coordinate System

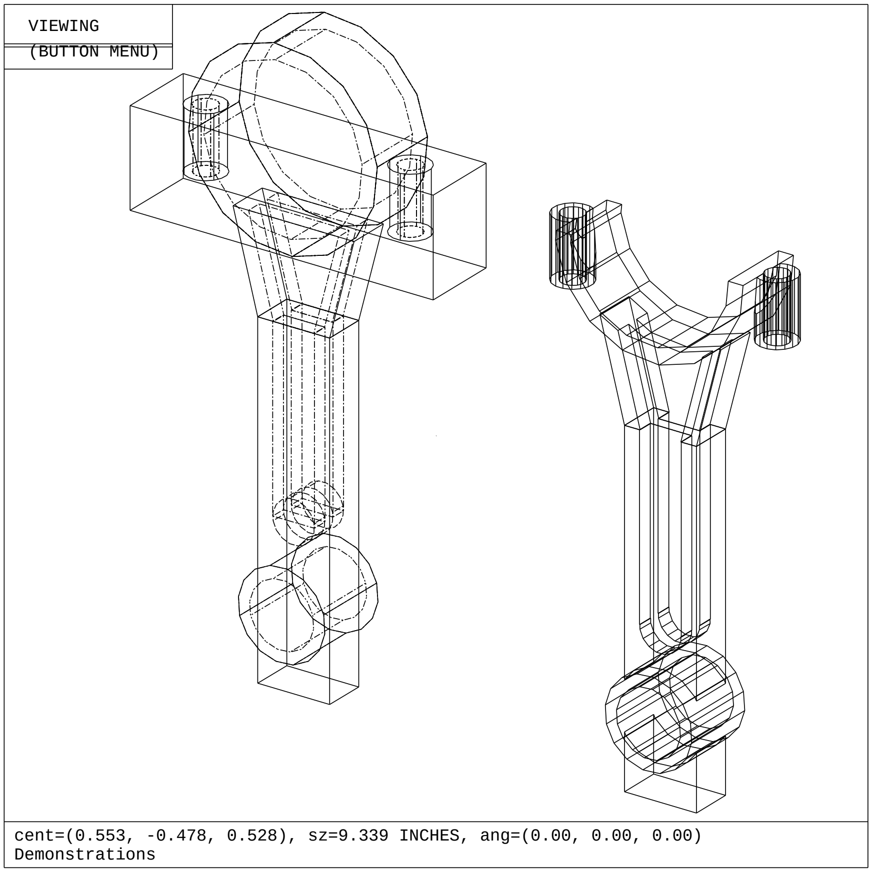

Figure 3.4: An Engine Connecting Rod

the other fingers will describe the sense of positive rotation.

3.4 Changing the View

At any time in an editing session, the user may add one or more subtrees to the active model space. If the viewing cube is suitably positioned, the newly added subtrees are drawn on the display. (The "reset" function can always be activated to get the entire active model space into view). The normal mode of operation is for users to work with wireframe displays of the unevaluated primitive solids. These wireframes can be created from the database very rapidly.

On demand, the user can request the calculation of approximate boundary wireframes that account for all of the boolean operations specified along the

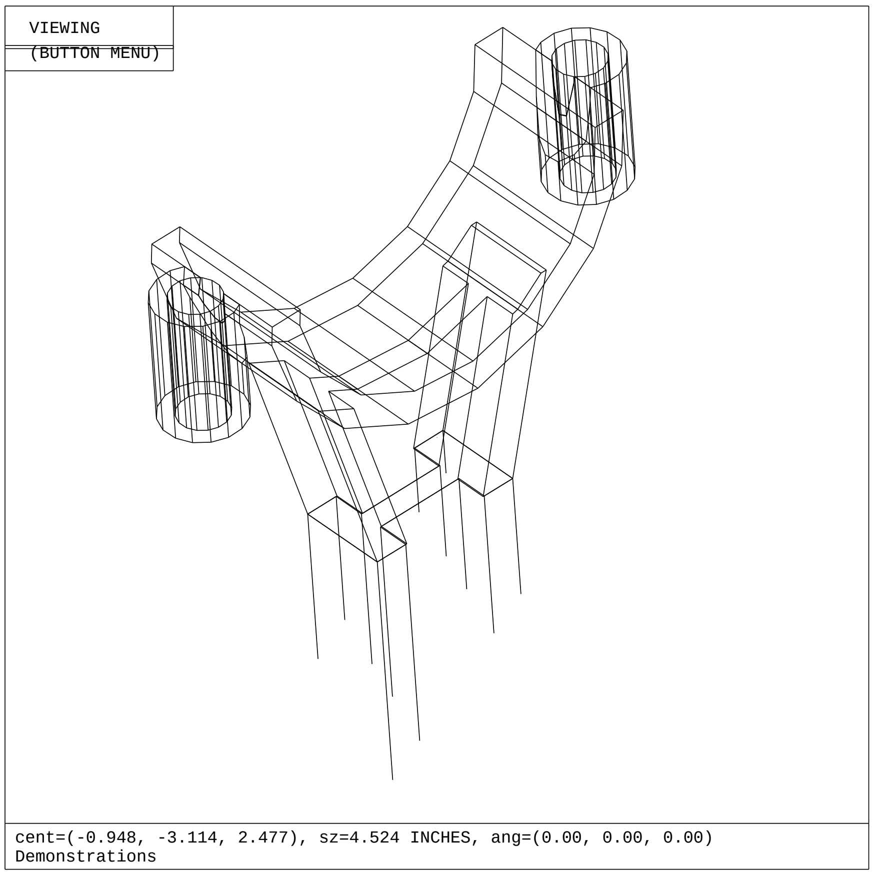

Figure 3.5: Close-Up Connecting Rod, Showing Z-clipping

arcs of the directed acyclic graph in the database. This is a somewhat time consuming process, so it is not used by default, but it is quite reasonable to use whenever the design has reached a plateau. Note that these boundary wireframes are not stored in the database, and are generally used as a visualization aid for the designer. Figure 3.4 shows an engine connecting rod. On the left side is the wireframe of the unevaluated primitives that the part is modeled with, and on the right side is the approximate boundary wireframe that results from evaluating the boolean expressions.

Also, at any time the user can cause any part of the active model space to be dropped from view. This is most useful when joining two complicated subsystems together; the first would be called up into the active model space, manipulated until ready, and then the second subsystem would also be called up as well. When any necessary adjustments had been made, perhaps to eliminate overlaps or to change positioning tolerances, one of the subassemblies could be dropped from view, and editing could proceed.

The position, size, and orientation of the viewing cube can be arbitrarily changed during an editing session. The simplest way to change the view is by selecting one of nine built in preset views, which can be accomplished by a simple keyboard command, or by way of a button press or first level menu selection. The view can be rotated and translated to any arbitrary position. The user is given the ability to execute a **save view** button/menu function that attaches the current view to a **restore view** button/menu function.

The rate of rotation around each of the X, Y, and Z axes can be selected by knob, joystick, or keyboard command. Because the rotation is specified as a rate, the view will continue to rotate about the view center until the rotation rate is returned to zero. (A future version of MGED will permit selection of rate or value operation of the knobs). Similarly, the zoom rate (in or out) can be set by keyboard command or by rotating a control dial. Also, displays with three or more mouse buttons have binary (2x) zoom functions assigned to two of the buttons. Finally, it is possible to set a slew rate to translate the view center along any axis in the current viewing space, selectable either by keyboard command or control dial. In VIEW state, the main mouse button translates the view center; the button is defined to cause the indicated point to become the center of the view.

The assignment of zoom and slew functions to the mouse buttons tends to make wandering around in a large model very straightforward. The user uses the binary zoom-out button to get an overall view, then moves the new area for inspection to the center of the view and uses the binary zoom-in button to obtain a "close up" view. Figure 3.5 shows such a close up view of the engine connecting rod. Notice how the wireframe is clipped in the Z viewing direction to fit within the viewing cube.

3.5 Model Navigation

In order to assist the user in creating and manipulating a complicated hierarchical model structure, there is a whole family of editor commands for examining and searching the database. In addition, on all keyboard commands, UNIX-style regular-expression pattern matching, such as "*axle*" or "wheel[abcd]", can be used. The simplest editor command (**t**) prints a table of contents, or directory, of the node names used in the model. If no parameters are specified, all names in the model are printed, otherwise only those specified are printed. The names of solids are printed unadorned, while the names of combination (non-terminal) nodes are printed with a slash ("/") appended to them.

If the user is interested in obtaining detailed information about the contents of a node, the list (**l**) command will provide it. For combination (non-terminal) nodes, the information about all departing arcs is printed, including the names of the nodes referenced, the boolean expressions being used, and an indication of any translations and rotations being applied. For leaf nodes, the primitive solid-specific "describe yourself" function is invoked, which provides a formatted display of the parameters of that solid.

The **tops** command is used to find the names of all nodes which are not referenced by any non-terminal nodes; such nodes are either unattached leaf nodes, or tree tops. To help visualize the tree structure of the database, the **tree** command exists to print an approximate representation of the database subtree below the named nodes. The **find** command can be used to find the names of all non-terminal nodes which reference the indicated node name(s). This can be very helpful when trying to decide how to modify an existing model. A related command (**paths**) finds the full tree path specifications which contain a specified graph fragment, such as "car/axle/wheel". In addition to these commands, several more commands exist to support specialized types of searching through the model database.

3.6 Editor States

The MGED editor operates in one of six states. Either of the two PICK states can be entered by button press, menu selection, or keyboard command. The selection of the desired object can be made either by using *illuminate mode*, or by keyboard entry of the name of the object.

Illuminate mode is arranged such that if there are **n** objects visible on the screen, then the screen is divided into **n** horizontal bands. By moving the cursor (via mouse or tablet) up and down through these bands, the user will cause each solid in turn to be highlighted on the screen, with the solid's name displayed in the faceplate. The center mouse button is pressed when the desired solid is located, causing a transition to the next state (Object Path, or Solid Edit).

Illuminate mode offers significant advantages over more conventional pointing methods when the desired object lies in a densely populated region of the screen. In such cases, pointing methods have a high chance of making an incorrect selection. However, in sparsely populated regions of the screen, a pointing paradigm would be more convenient, and future versions of MGED will support this.

3.7 Model Units

All databases start with an "ident" record which contains a title string that identifies the model, the current local units (eg, mm, cm or inches) of the model, and a database version identification number. As noted, all numerical information in the database is stored in the fixed base unit of millimeters, and all work (input and output) is done in a user-selected local unit. The user can change his local unit at any time by using the **units** command. This way of handling units was selected to free the user from worrying about units conversion when components are drawn from the "parts bin".

Chapter 4

PERIPHERAL DEVICES

Before we discuss the features of MGED, we will introduce the hardware devices used to implement them. These devices are the "tools of the trade" for the MGED user. We will discuss only basic operational characteristics here. Specific use of these devices will be covered in the later sections on the viewing and editing features of MGED.

4.1 Joystick

The joystick is a mechanical device used to do rotations in MGED. Any movement left or right rotates the display about the X-axis. Any movement up or down rotates the display about the Y-axis. When the joystick top is twisted in a clockwise or counterclockwise direction, the display rotates about the Z-axis. Any combination motion of the joystick will produce a "combined" rotation about the appropriate axes. As implemented on the Vector General hardware, all of these motions have a spring return to a null center position.

4.2 Button Box

The button box contains a collection of buttons. On each button is a light that can be lit under program control. Pressing a button sends a "press" event to MGED, and results in an action occurring, or a condition being set. The exact functions assigned to these buttons will be discussed in the sections on viewing the display and on editing.

4.2.1 Vector General Buttons

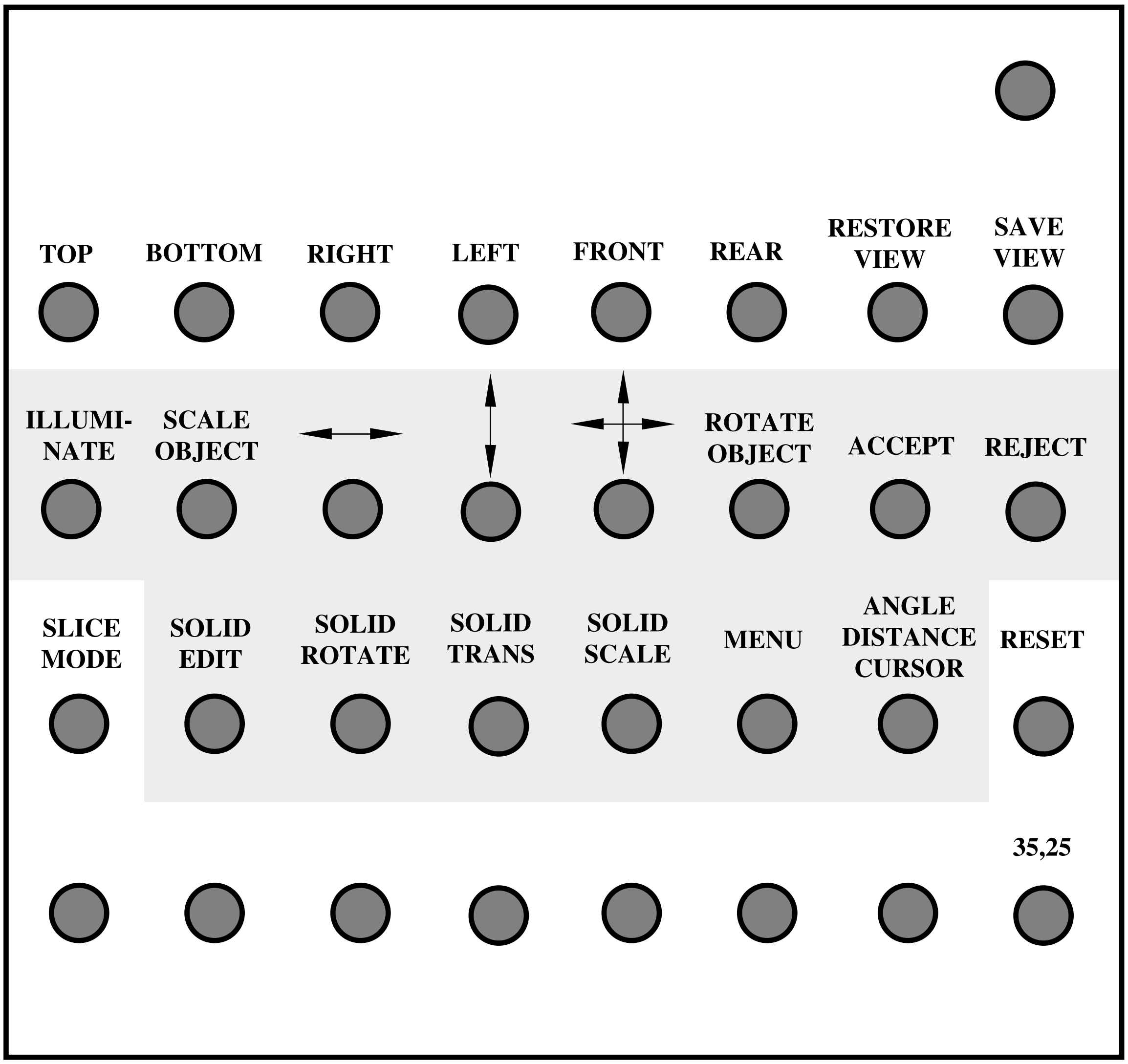

Figure 4.1: Vector General Button Assignments

	Help	ADC	Reset	Zero Knobs	
Obj Scale	Obj ScaleX	Obj ScaleY	Obj ScaleZ	Empty	Save View
Obj TransX	Obj TransY	Obj TransXY	Obj Rotate	Empty	Restore View
Solid Trans	Solid Rot	Solid Scale	Solid Menu	Obj Pick	Solid Pick
REJECT	Bottom	Top	Rear	90,90	ACCEPT
	Right	Front	Left	35,25	

Figure 4.2: Silicon Graphics Button Assignments

The Vector General has thirty-two buttons. Figure 4.1 depicts the functions programmed for each button. The buttons in the shaded area are used for editing while the rest are used for viewing the display.

4.2.2 Megatek Buttons

The Megatek button box is a general purpose input/output device that communicates with MGED through an intelligent control unit. The device has eight rotatable knobs and 16 buttons with lights. The "buttons" and "knobs" of the Megateks are located in the same box. There are not enough buttons to have just one assigned meaning, hence most buttons have dual functions. To toggle the functions of the buttons, use the upper right button (toggle button). When the light on this button is ON, the functions listed on the RIGHT above each button is the current function. When the light on the "toggle" button is OFF, the functions labeled on the LEFT are then in effect. The left/right meaning of these buttons is grouped generally according to viewing functions on the left and editing functions on the right.

Figure 4.3 summarizes the uses of the buttons. Depressing the button switches the light on and off. Many of these serve a dual role depending upon the selected mode - viewing or editing. The mode is selected by depressing but-

Button	Function
1	View Mode: Restores View Edit Mode: Translation in the Object-Edit mode
2	View Mode: Saves View Edit Mode: Translation in the Object-Edit mode
3	 Edit Mode: Saves the model being displayed on the screen
4	Off: Viewing mode On: Edit mode
5	View Mode: Resets View Edit Mode: Scaling in the Object-Edit mode
6	 Edit Mode: Rotation in the Object-Edit mode
7	View Mode: Angle/Distance Cursor Edit Mode: Translation in the Object-Edit mode
8	 Edit Mode: Rejects display and returns to Viewing display
9	View Mode: Bottom View Edit Mode: Scaling in the Solid-Edit mode
10	View Mode: Left View Edit Mode: Rotation in the Solid-Edit mode
11	View Mode: Rear View Edit Mode: Translation in the Solid-Edit mode
12	View Mode: 90, 90 View Edit Mode: Restores Edit mode menu
13	View Mode: Top View Edit Mode: Transfers from Viewing to Solid Pick
14	View Mode: Right View Edit Mode: Transfers from Viewing to Object Pick
15	View Mode: Front View
16	View Mode: 35/45 View

Figure 4.3: Megatek Buttons

ton 4. If light 4 is off, the system is performing in the viewing mode, and the commands shown in the top half of the table are executed. If light 4 is on, the system is performing in the edit mode, and the commands shown in the bottom half are executed.

4.2.3 Silicon Graphics Buttons

The button box layout for the SGI Iris is given in Figure 4.2. Note that the "right" button shows you the right side of the model, as if you were looking in from the left. To achieve the customary draftsman views, this function goes on the left.

The upper left button is the **help** key. If this button is held down, and any other button (or knob) is activated, a descriptive string is displayed in the eight character LED display on the button box. The upper right button is used to reset all the knobs to zero. This is useful to halt a runaway rotation or zoom operation.

4.3 Knobs (Dials)

The knobs (or control dials) are used to send digital information to the computer. As a knob is turned, a succession of numbers are available for use by the computer. The knobs can be used to rotate a displayed object about the x, y, or z axis, translate the object along the x or y axis, and change the size of the view. Action performed by these knobs is continuous and is initiated by turning the knob in the proper direction and terminated by turning the knob in the opposite direction.

4.3.1 Vector General Knobs

Figure 4.4 depicts the functions assigned to each of the ten knobs. The exact functions of each of these knobs will be discussed in the angle distance cursor section and in the viewing features section.

4.3.2 Megatek Knobs

The "buttons" and "knobs" of the Megateks are located in the same box. There are not enough knobs to have ONE assigned meaning, hence three knobs have dual functions. The second function of the first three knobs is only in effect when the angle-distance cursor (ADC) is on the screen.

4.3.3 Silicon Graphics Knobs

Figure 4.5 depicts the functions assigned to the eight knobs on the Silicon Graphics knob box. In normal operation, the left knobs provide rotations, and the

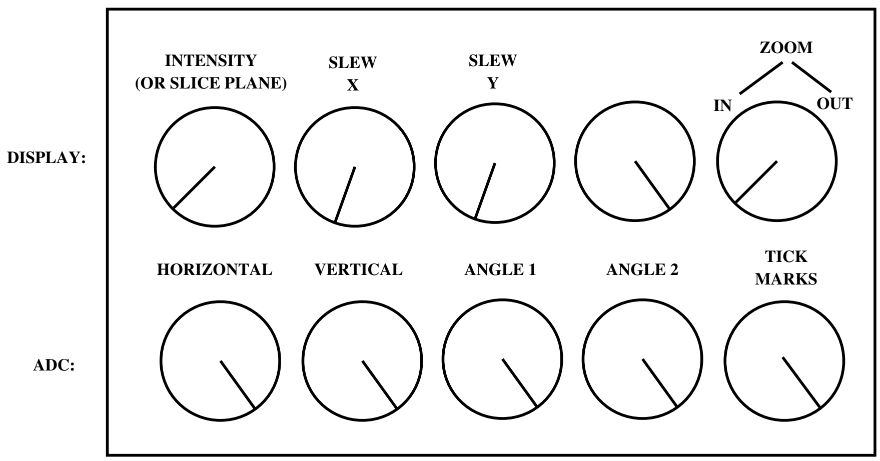

Figure 4.4: Vector General Knob Assignments

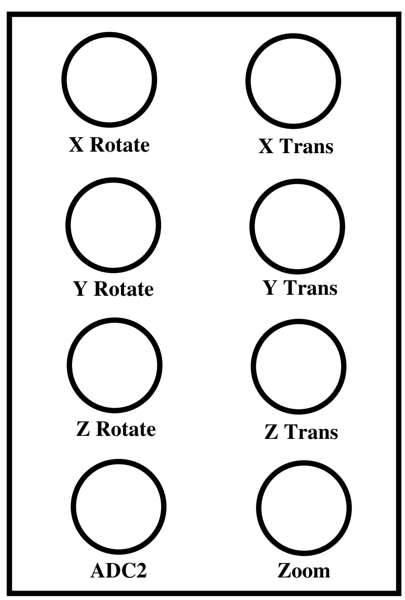

Figure 4.5: Silicon Graphics Knob Assignments

right knobs provide translations and zooming. When the angle/distance cursor is activated, some of the knobs are redefined.

4.4 Mouse or Data Tablet

Moving the mouse or the data tablet "pen" causes a cursor on the screen to move. The screen X-Y coordinates of the cursor can be sensed by MGED at any time. Clicking one of the mouse buttons, or depressing the tip of the pen, results in MGED receiving a special event notification. The meaning of this mouse event depends on the current editing mode and which portion of the display faceplate that the cursor is located in.

Below is a list of some of the functions the mouse is used for in MGED;

- Selecting editing menus, edit functions (move faces, move edges etc.) and viewing functions (selected from main edit menu); move pointer to appropriate edit function and press center mouse button.

- Pointing functions; interactively positioning solid primitive relative to other solids with positioning or size update being displayed at the same time, position pointer where required and click center mouse button.

- Scaling of view size; enlarge or reduce for a more detailed view of object, left button shrinks view size, right button enlarges view size.

- During the solid or object illuminate phase of editing, the screen is divided into invisible horizontal sections. The available selections are scanned by moving the mouse up and down.

- When MGED is is the viewing state, and a mouse event is received which is not in the menu area of the faceplate, the point at which the cursor is pointing at will be translated to become the center of the current view. By pointing and clicking the center mouse button, the center of the viewing cube can be moved to allow close-up viewing of different areas in your model.

4.4.1 Vector General Data Tablet

Position information is entered using a pen-like stylus. The distance this pen is from the tablet is important. If the pen tip is within one half inch of the tablet surface, the cursor location on the screen corresponds to the X,Y location of the pen on the tablet. This condition is called the "near" position. If the pen is more than one half inch from the tablet surface, the cursor remains located in the center of the screen. When the pen is pressed against the tablet surface, the pressure switch is activated and a "mouse" event is sent to MGED.

4.4.2 Megatek Data Tablet

Some Megatek systems enter position data on the data tablet using a pen-like stylus. If the tip of the stylus is within one-half inch of the surface of the tablet, a "star" corresponding to this location is displayed on the display screen. If the tip is moved more than one-half inch from the surface, the position of the star remains fixed. When the stylus is pressed against the tablet surface, the pressure switch is activated and a "mouse" event is sent MGED.

Other Megatek data tablets have a mouse instead of a pen. This mouse has four buttons on it. The yellow (top) button is used during illumination and editing just as the pen on the Vector General terminals. However, in the viewing mode, when pushed, the point which it was pointing at will be drawn at the center of the screen. The blue (bottom) button has this same function at ALL times and is used to "slew" the display during editing. The white (left) and the green (right) buttons on the mouse are used for zooming the display at a fixed rate. The white button will zoom out and the green button will zoom in.

4.4.3 Silicon Graphics Mouse

The left and right mouse buttons are used for binary (2x) zooming, and the center mouse button is used for all other MGED mouse functions.

On the Silicon Graphics 3-D workstations, MGED can be run directly, or it can be run under the window manager MEX. In both cases, MGED opens two windows, one outlined in white for all text interaction, and one outlined in yellow for all graphics display. When running MGED directly (without MEX), all mouse events are sent the MGED, regardless of where the mouse is pointing. In order to shift emphasis between the graphics and text windows, the smaller one can be enlarged by pointing the cursor within the boundaries of the smaller window, and pressing the center button. This enlarges that window, and reduces the size of the other window.

When MEX is running, it is necessary to follow the MEX convention of moving the cursor into the desired window, and clicking the right mouse button, to "attach" all input to that window. This has the unfortunate consequence of requiring a lot of extra mouse clicking, because the graphics window needs to be attached when using the buttons, knobs, and mouse, while the text window needs to be attached in order to enter keyboard commands.

On the Silicon Graphics 4-D workstations running 4Sight, mouse events are sent to MGED only when the cursor is within the boundaries of the MGED graphics window.

4.4.4 Sun Workstation Mouse

On the Sun workstation, MGED must be run in a **suntools** window. The main consequence of this is that mouse events are sent to MGED only when the cursor is within the boundaries of the MGED graphics window on the screen. The left and right mouse buttons are used for binary (2x) zooming, and the center mouse button is used for all other MGED mouse functions.

4.5 Keyboard

The keyboard is used to issue commands and supply parameters to MGED. It is also used to login and logout of the UNIX system, and to run other UNIX programs. All characters typed on the keyboard, with the exception of the user's password, are displayed (echoed) on the monitor. In this text, all input typed by the user is shown in *italics*, while all literal MGED output is shown in `typewriter font`. All entries are terminated by depressing the RETURN key. This action immediately precedes the execution of the directive. In most cases, lower case letters must be used. A space must be used between the command and its arguments. Embedded blanks are not allowed. Entering Control/H causes cursor to backspace and erase entered information. An MGED command is interrupted by entering Control/C. End-of-File is sent to MGED by entering Control/D. The graphics editor displays the prompt

```
mged>
```

on the display whenever it is ready to accept a command from the keyboard.

Chapter 5

OPERATING INSTRUCTIONS

5.1 Entering the Graphics Editor

Type *mged filename*, e.g.:

```
mged s_axle.g
mged shaft.g
mged fred.g
```

where the filename is the name of the UNIX file in which your object description data is stored. It is conventional that the extension ".g" on the filename signifies a graphics file, and is a good practice, but is not required. If the named database does not already exist, MGED will ask if it should create a new database. MGED will ask:

```
% mged new.g
BRL-CAD Release 3.0 Graphics Editor (MGED)
    Tue Sep 6 02:52:55 EDT 1988
    mike@video:/cad/mged.4d
new.g:  No such file or directory
Create new database (y|n)[n]?  y
attach (nu|tek|tek4109|ps|plot|sgi)[nu]?  sgi
ATTACHING sgi (SGI 4d)
Untitled MGED Database (units=mm)
mged>
```

Here, the *italic* type indicates the user's response: *y* instructs MGED to create the new database, and *sgi* instructs MGED to attach to a window on the Silicon Graphics (SGI) workstation.

Directives to the graphics editor are made by

1. entering information from the keyboard, shown here in the text by the use of *italics*,
2. using the stylus to select items from the menu (select), and
3. pressing buttons and twisting knobs on the function control box (press, twist).

The prompt for a command is `mged>`.

5.1.1 Running MGED on a Silicon Graphics

When running MGED from the console of a Silicon Graphics workstation, MGED retains the text window from which it was started, and opens a second window for the graphics display. By default, the graphics window is quite large, and the text window is rather small.

On the SGI 3-d workstations, should you wish to have a large text window to scan printed output, move the mouse pointer into the text window and click the center mouse button. Use the reverse procedure to regain a large graphics window, i.e., move the mouse pointer into the graphics window and click the center mouse button.

5.1.2 Running MGED on a Tektronix

To run MGED on the tek4014 class of terminals one needs to have TWO terminals - the graphics terminal (4014 or one which emulates a 4014) and another terminal to enter commands on.

The procedure is as follows:

1. login on the graphics terminal
2. enter *tty* to find out which terminal number has been assigned
3. enter *sleep 32000* to put the graphics terminal in sleep mode
4. login on the other terminal
5. enter *mged file* to execute MGED
6. enter *tek* to select the tek4014 device processor
7. enter the tty value found in step 2.
8. perform editing
9. enter *q* to quit MGED

10. enter control-c on the graphics terminal to end the sleep mode
11. logout on both terminals

Since there are no knobs or buttons on the tek4014 class of terminals, one is forced to use the *press* and *knob* commands to emulate these peripherals. Other commands which can/should be used are:

ill	put up a desired path
center	slew the display
size	zoom the display
sed	solid edit immediately

The main force behind the writing of a driver for the tek4014 terminals was to allow the use of the Teletype 5620 terminals. These graphic terminals have an internal processor and different windows can be set up which represent different terminals. Hence two terminals are NOT necessary. The use of the Teletype 5620 terminals is then the same as the procedure outlined above, except each window represents a terminal.

5.2 The Pop-Up Button Menu

The default MGED faceplate is shown in Figure 3.1. If the BUTTON MENU area on the screen is selected with the mouse, then the pop-up button menu appears, as shown in Figure 3.2. This menu can be very useful in reducing the amount of hand motion between the mouse and the button box.

5.3 Starting Your Model

Modeling practices using MGED can be quite individual. The following is a suggested modeling method to start with; you may end up developing your own style as you become more familiar with MGED.

First of all, decide how you want to represent your model, including the amount of detail, types of solids and regions necessary. Have an accurate sketch or engineering drawing available, so that you can easily tranfer its information into the types of primitive solids necessary to create your model. Where possible it is recommended to start with a large block solid and "subtract" pieces from it. In this way you avoid errors with abutting faces of a collection of solids "unioned" together.

Next the solids are created using the *make*, *cp*, *mirror* or *in* commands. Depending on the complexity of the model, the solids may be created in the desired location or created at the origin and later translated to the desired location. Creation at the origin provides an opportunity to take advantage of possible symmetries in the geometry. Once all the solids are finished it is time to create the region[s], which will describe (to MGED) how to combine the solids to represent the model.

The region[s] are then given the desired item/air code (if this is necessary, otherwise leave it as the system default value), and material codes. The regions are then put onto a group, usually for functionality only. A group has no operations as such (like union [u], intersection [+] or difference [-]) and is just a collection of objects for convenient naming of a whole screen or collection of objects.

Chapter 6

CREATING NEW OBJECTS

6.1 Creating New Leaves (Solids/Primitives)

A family of commands exists to allow the user to add more actual solids (leaf nodes) to the model database. To obtain a precise duplicate of an existing solid (presumably to be changed by a subsequent editing command), the copy (*cp*) command can be used. It is important to note that the copy operation is different from creating an *instance* of an existing solid; there are occasions to use both operations. If the precise configuration of the solid desired is not important, the *make* command can be used to create a stock prototype solid of the desired type with the given name, which can then be edited to suit. The *mirror* command makes a duplicate of an existing solid reflected about one of the coordinate axes.

If the actual numeric parameters of a solid are known, then the *in* command can be used. In addition to prompting for the descriptions of the full generic primitive solids, this command also accepts abbreviated input formats. For example, a wedge or an RPP can be entered with a minimum of parameters, even though a database ARB8 is created. Similarly, the parameters for a right circular cylinder can be given, resulting in a truncated general cone (TGC) being stored. This is not a very sophisticated way to build solids, but it receives a surprising amount of use.

A number of commands also exist to create new solids with some higher level description. For example, the *inside* command creates a new solid inside an existing solid, separated from the existing solid by specified tolerances. This is quite useful for creating hollow objects such as fuel tanks. It is possible to create a plate with a specified azimuthal orientation and fallback angle, or to create an ARB8 (plate) by specifying three points and a thickness, or to create an ARB8 given one point, an azimuthal orientation, and a fallback angle.

6.2 Specific Cases

After having started MGED and created a new database, the next step is to use the *units* command to tell the system that you will be entering values in mm, cm, m, in or ft. For our example we will work in mm:

```
units mm
```

Now you may give your database a title using the *title* command as in:

```
title Mechanical Bracket
```

This title ("Mechanical Bracket") now appears at bottom left hand corner of graphics window. Further examples:

```
title six wheeled tank
title stub axle
```

At this point you can start creating your solid objects using the "arbs", "sph", "tor", *etc*, arguments to the *make* or *in* command. The *make* command gives you a solid to a default size, you then have to use the solid edit mode to interactively edit the solid to the desired size. *make* command is entered as below: Examples:

```
make box arb8
make cyl rcc
make ball sph
```

The first argument is the solid name, and the second argument is the primitive type.

The *in* command expects you to key in a set of parameters to describe your solid; the parameters can be the x y and z of a vertex (such as the corner of an ARP8), or the x y and z of a vector (such as the height or H vector of a BOX) or the radius (such as for a torus). Below is a list of primitives with their *in* commands as requested by MGED and sample input. Reading an *in* file into a MGED data file will be discussed later. Note how providing incomplete input to the *in* command will result in MGED repeating the prompt for the missing information.

6.2.1 RPP (rectangular parallelepiped)

```
mged> in name rpp
Enter XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX: 0 25
Enter YMIN, YMAX, ZMIN, ZMAX: 0 50
```

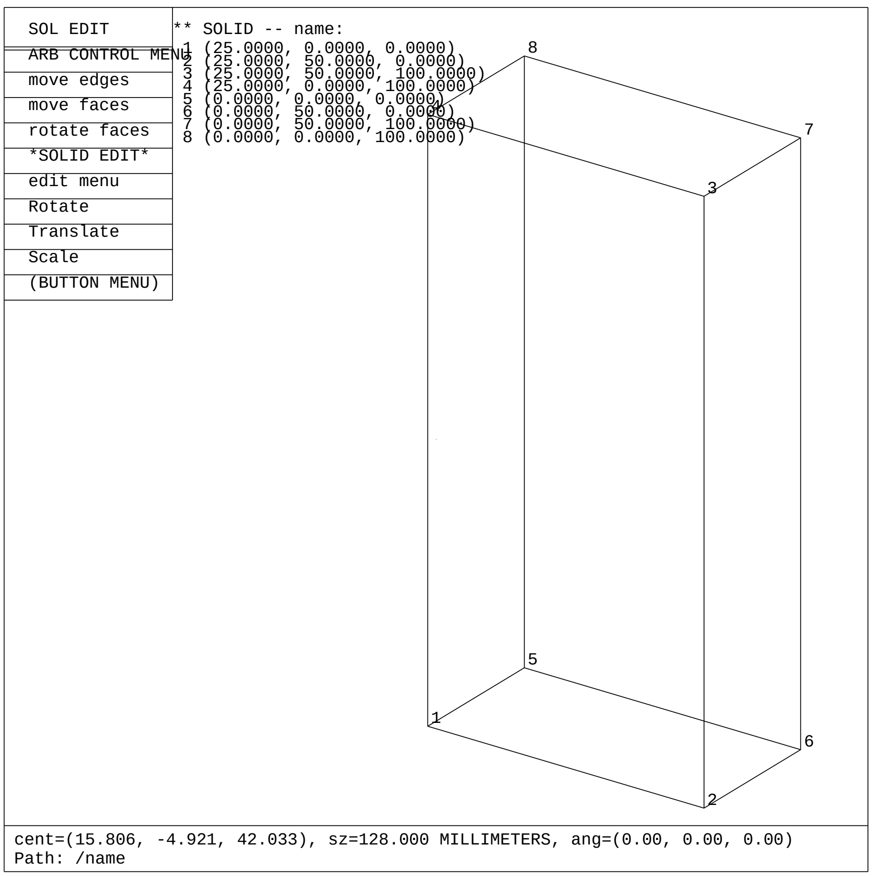

Figure 6.1: Example RPP

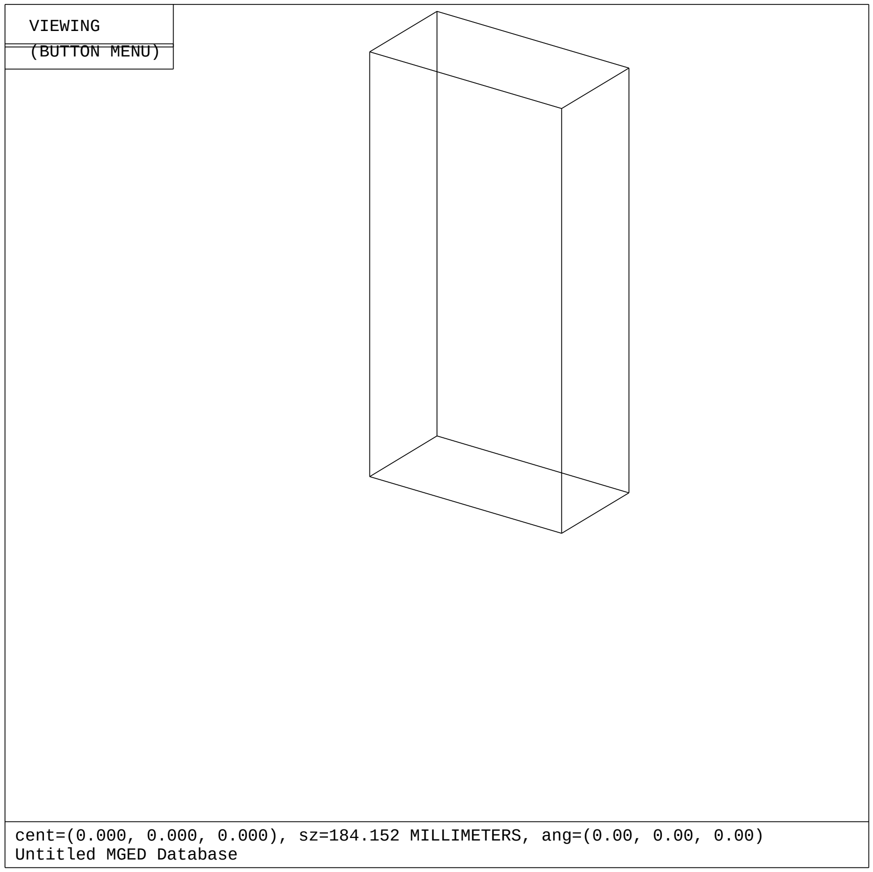

Figure 6.2: Example BOX

Enter ZMIN, ZMAX: *0 100*

This sequence produces the RPP shown in Figure 6.1.

6.2.2 BOX (BOX)

```
mged> in my box
Enter X, Y, Z of vertex:  0 0 0
Enter X, Y, Z of vector H: 25 0 0
Enter X, Y, Z of vector W: 0 50 0
Enter X, Y, Z of vector D: 0 0 100
```

This sequence produces the BOX shown in Figure 6.2.

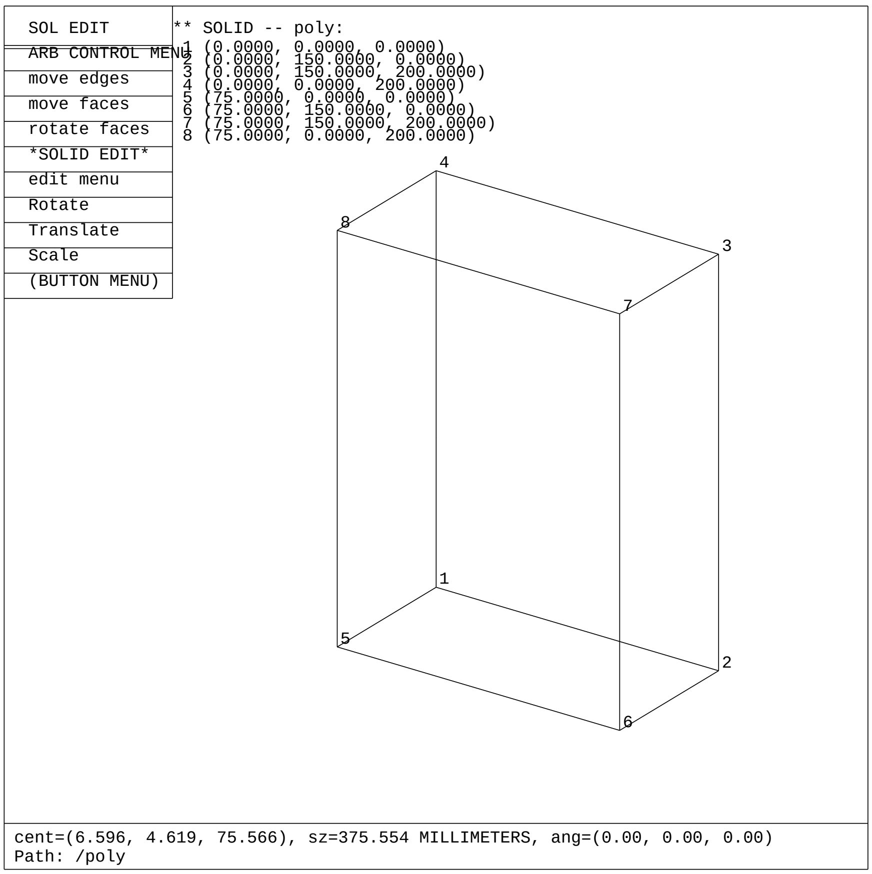

Figure 6.3: Example ARB8

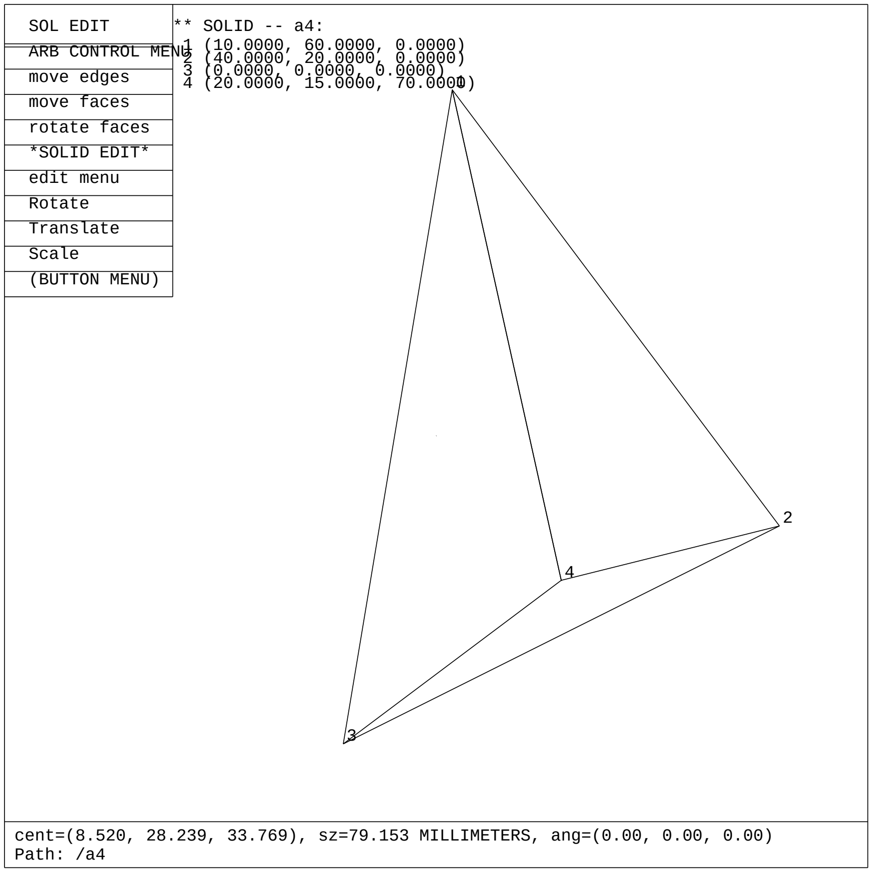

Figure 6.4: Example ARB4

6.2.3 ARB8: Arbitrary Convex Polyhedron, 8 Vertices

```
mged> in poly arb8
Enter X, Y, Z for point 1:  0 0 0
Enter X, Y, Z for point 2:  0 150 0
Enter X, Y, Z for point 3:  0 150 200
Enter X, Y, Z for point 4:  0 0 200
Enter X, Y, Z for point 5:  75 0 0
Enter X, Y, Z for point 6:  75 150 0
Enter X, Y, Z for point 7:  75 150 200
Enter X, Y, Z for point 8:  75 0 200
```

This sequence produces the ARB8 shown in Figure 6.3.

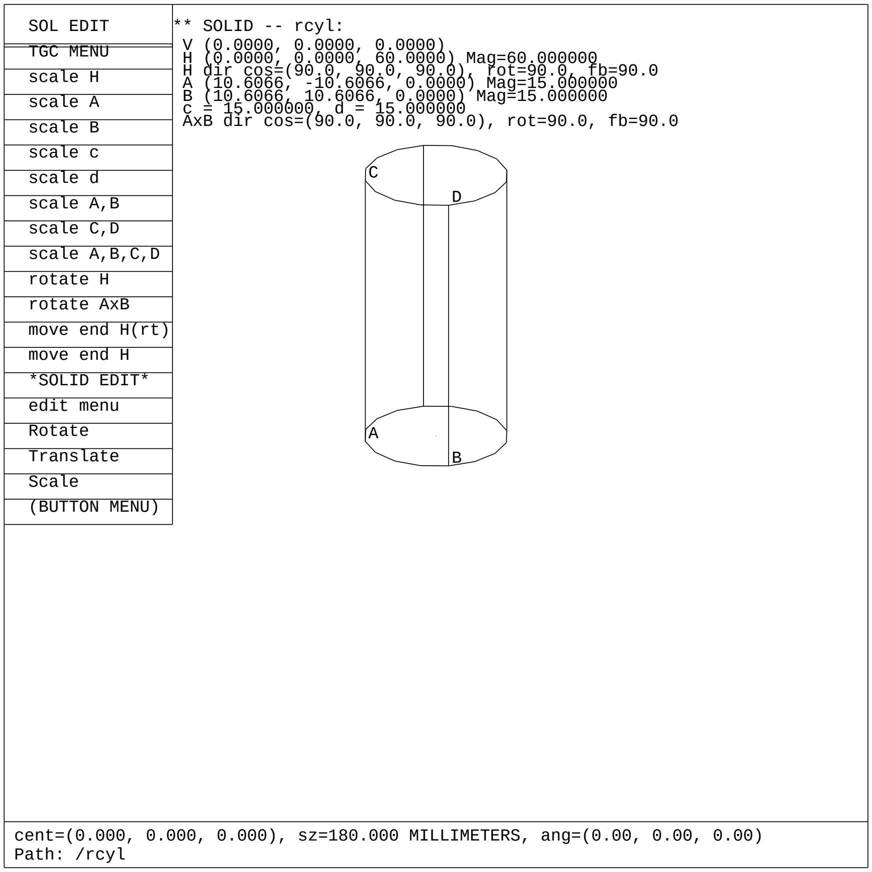

Figure 6.5: Example Right Circular Cylinder

6.2.4 ARB4: Arbitrary Convex Polyhedron, 4 vertices

`mged>` *in a4 arb4*
`Enter X, Y, Z for point 1:` *0 0 0*
`Enter X, Y, Z for point 2:` *10 60 0*
`Enter X, Y, Z for point 3:` *40 20 0*
`Enter X, Y, Z for point 4:` *20 15 70*
This sequence produces the ARB4 shown in Figure 6.4.

6.2.5 RCC (Right Circular Cylinder)

`mged>` *in rcyl rcc*
`Enter X, Y, Z of vertex:` *0 0 0*

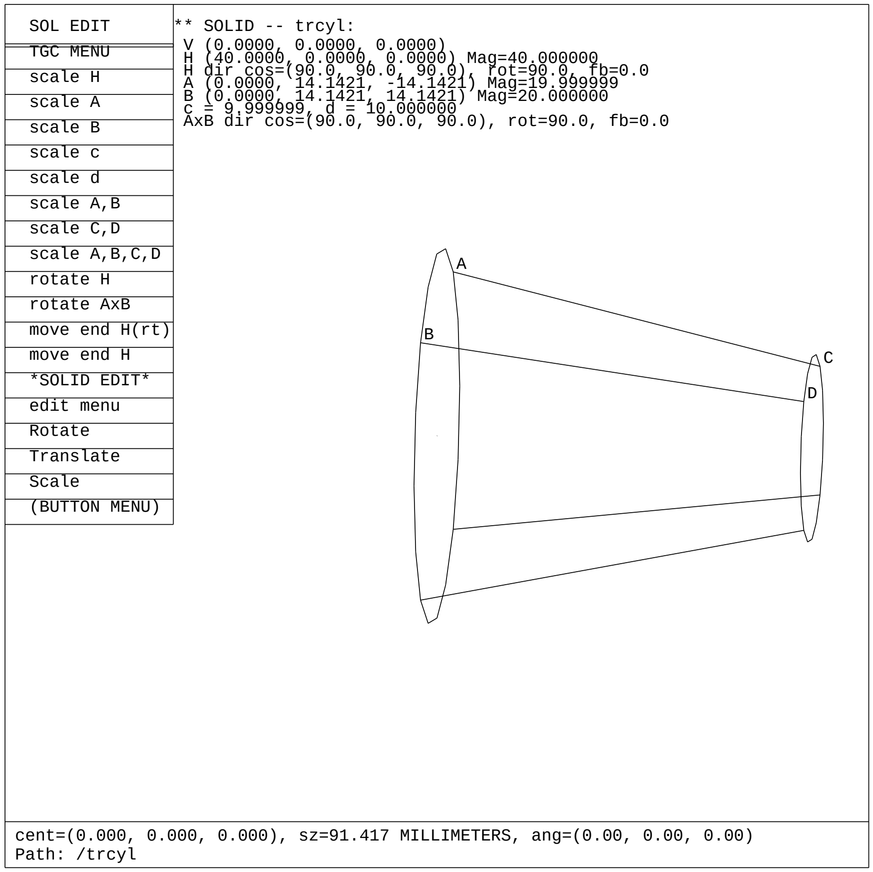

Figure 6.6: Example Truncated Right Cylinder

`Enter X, Y, Z of height (H) vector:` *0 0 60*
`Enter radius:` *15*
This sequence produces the RCC shown in Figure 6.5. Note that in this case, the A,B,C, and D vectors have magnitude which equal the radius, 15.

6.2.6 TRC (Truncated Right Cylinder)

`mged>` *in trcyl trc*
`Enter X, Y, Z of vertex:` *0 0 0*
`Enter X, Y, Z of height (H) vector:` *40 0 0*
`Enter radius of base:` *20*
`Enter radius of top:` *10*
This sequence produces the TRC shown in Figure 6.6. Note that the magnitude

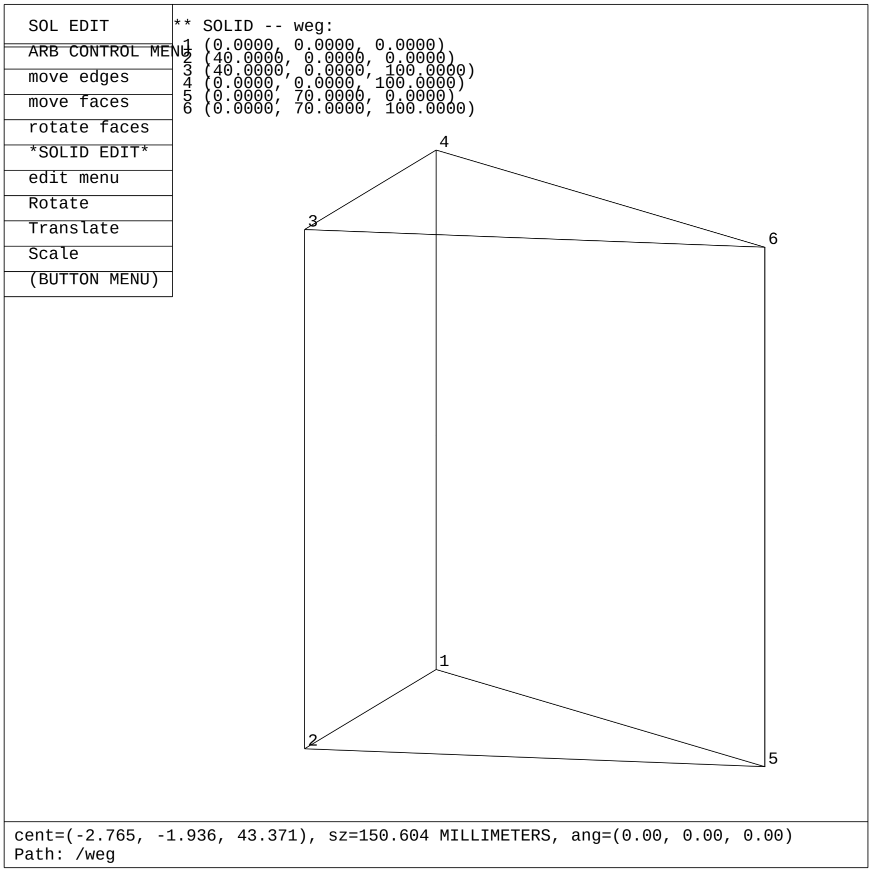

Figure 6.7: Example Right Angle Wedge

of A and B equal the base radius, 20,

6.2.7 RAW (Right Angle Wedge)

`mged>` *in weg raw*
`Enter X, Y, Z of vertex:` *0 0 0*
`Enter X, Y, Z of vector H:` *40 0 0*
`Enter X, Y, Z of vector W:` *0 70 0*
`Enter X, Y, Z of vector D:` *0 0 100*
This sequence produces the RAW shown in Figure 6.7.

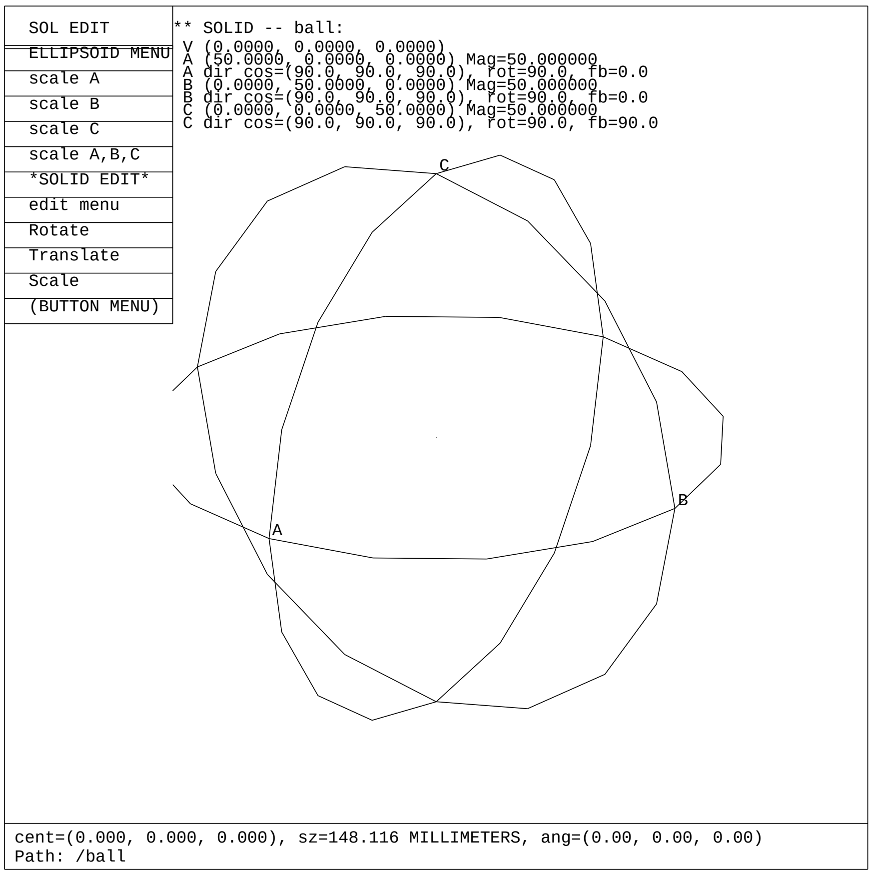

Figure 6.8: Example Sphere

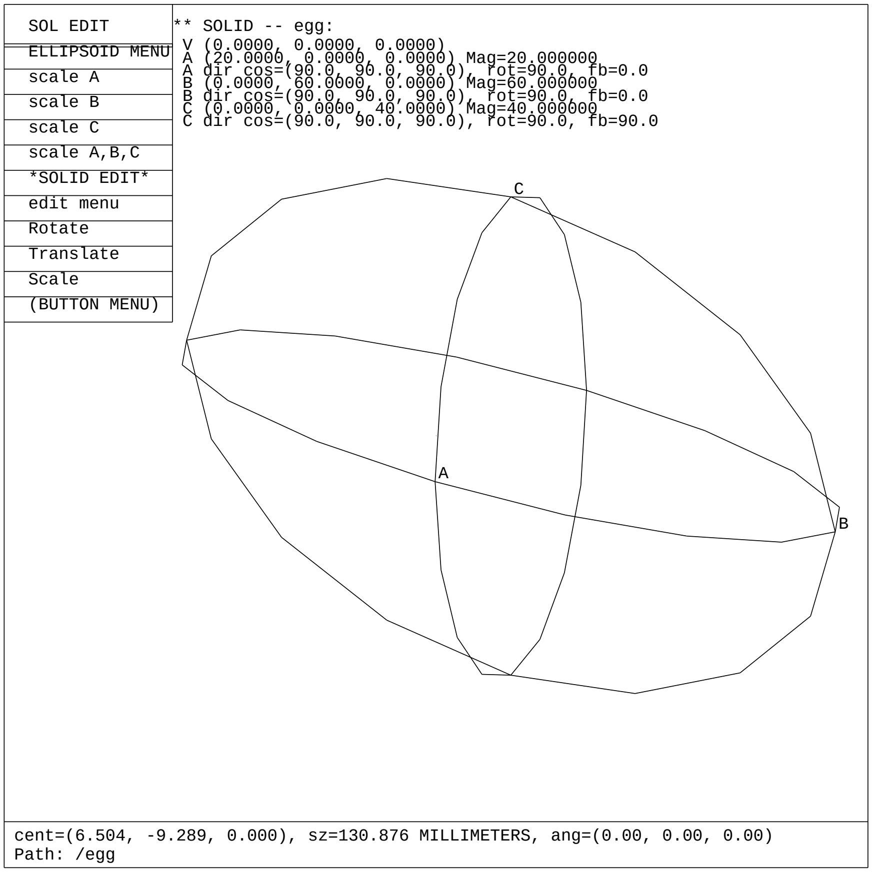

Figure 6.9: Example General Ellipsoid

6.2.8 SPH (Sphere)

mged> *in ball sph*
Enter X, Y, Z of vertex: *0 0 0*
Enter radius: *50*
This sequence produces the sphere shown in Figure 6.8. Note that the A, B, and C vectors all have magnitude equal to the radius, 50.

6.2.9 ELLG (General Ellipsoid)

mged> *in egg ellg*
Enter X, Y, Z of vertex: *0 0 0*
Enter X, Y, Z of vector A: *20 0 0*

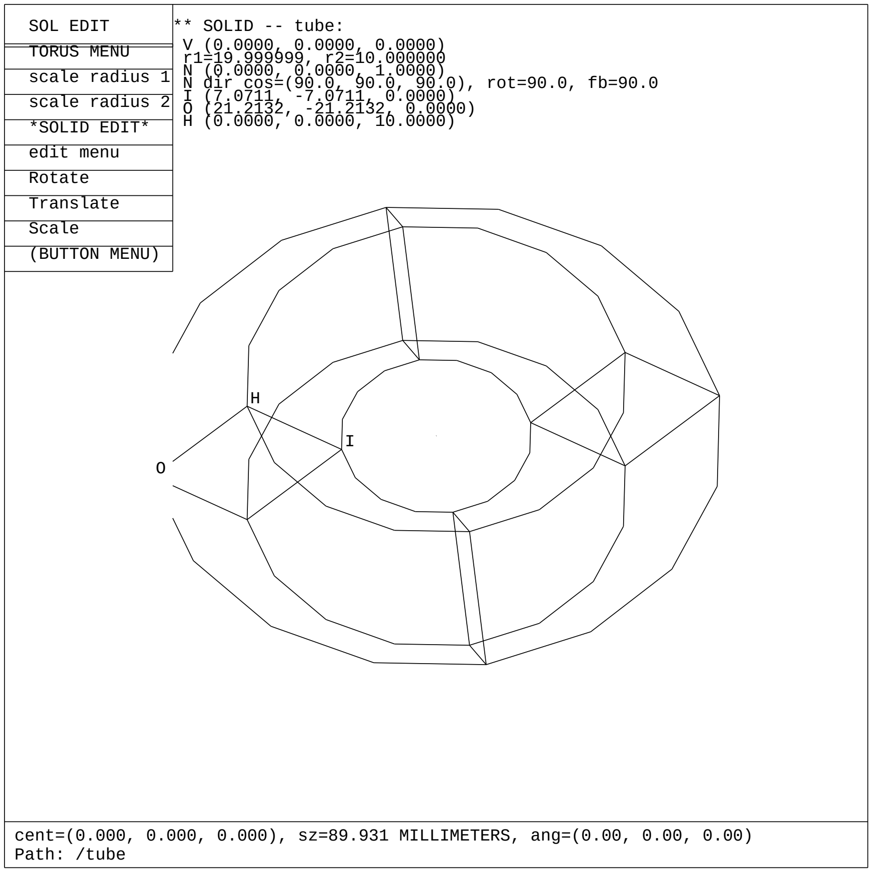

Figure 6.10: Example Torus

Enter X, Y, Z of vector B: *0 60 0*
Enter X, Y, Z of vector C: *0 0 40*
This sequence produces the ellipsoid shown in Figure 6.9.

6.2.10 TOR (Torus)

mged> *in tube tor*
Enter X, Y, Z of vertex: *0 0 0*
Enter X, Y, Z of normal vector: *0 0 50*
Enter radius 1: *20*
Enter radius 2: *10*
This sequence produces the torus shown in Figure 6.10.

6.3 Creating New Combinations

Non-terminal nodes in the directed acyclic graph stored in the database are also called *combinations*. It is possible to extend the definition of a non-terminal node by adding an instance of an existing node to the non-terminal node with an associated boolean operation of union; this is done by the *i* (instance) command. To start with, such an instance has an identity matrix stored in the arc; the user needs to separately edit the arc to move the instance to some other location. If the non-terminal node being extended does not exist, it is created first.

The instance command provides the simplest way to create a reference to another node. Instances of a whole list of nodes can be added to a non-terminal node by way of the group *g* command. If instances of a list of nodes with non-union boolean operations is to be added to a non-terminal node, the region *r* command accepts a list of (operation, name) pairs, where the single lower case character "u" indicates union, "–" indicates subtraction, and "+" indicates intersection. The first operation specified is not significant. An example of this command might be:

r non-terminal u node1 – node2 + node3

For historical reasons, there is no explicit grouping possible, occasionally forcing the user to create intermediate non-terminal nodes to allow the realization of the desired boolean formula. It is also important to note that for the same reasons there is an *implicit* grouping between union terms, i.e.

u n1 – n2 + n3 u n4 – n5

is evaluated as

(n1 – n2 + n3) union (n4 – n5)

rather than

((((n1 – n2) + n3) union n4) – n5)

Therefore, you can think of the solids on either side of the union operators as surrounded by parentheses. The order of the region members is critical, and must be scrutinized when members are added or deleted from a region. The order can be printed out using the *l* or *cat* commands.

Chapter 7

VIEWING FUNCTIONS

The MGED viewing features are designed to allow one to examine an object in close detail. Any of the viewing features can be invoked at any time. It should be noted, that these functions do not change the actual data, only the way the data is displayed.

7.1 Preset Views

Six standard views (front, rear, top, bottom, left, and right) and one oblique view (azimuth 35, elevation 25 isometric) are each assigned to the function buttons, views 35 25 (isometric), top, right, front, 45 45 are available from the screen editor menu. Hence, any of these views is immediately available at the press of the appropriate function button or mouse selection. The views available are not limited to these standard views however, as the display can be rotated to any view by using the dial box. By pressing the function button labeled "save view" or entering the keyboard *saveview* command, the present view as displayed display is saved (used for raytracing, producing colored pictures, which will be discussed later). At any time, the saved view can be immediately returned to the screen by pressing the "restore view" function button. The "restore view" button will be lit whenever a view has been saved. The function button labeled "reset", restores the display to the default view (front) when pressed.

7.2 View Translation

The displays can be panned or slewed on the screen in two ways – using the mouse pointer or by using the dial box knobs. When one is editing, the mouse functions are not available for slewing, hence one must use the dial box knobs to slew the display.

To slew the display using the control knobs, one uses the knobs labeled "slew x" or "slew y". The null positions on these knobs is in the center or straight up. If the "slew x" knob is turned clockwise of center, the display will move to the right. If it turned counterclockwise, the display will move to the left. For the "slew y" central knob, clockwise of the center moves the display up and counterclockwise moves the display down. The further these knobs are turned from center, the faster the display moves.

7.3 View Zooming

One can zoom the display by using the dial box knob labeled "zoom". Again the null position of this knob is center or straight up. Turning this knob clockwise of center causes the display to increase in size producing a zoom-in effect. Turning this knob counterclockwise of center causes the display to decrease in size or zoom-out. Again, the further the "zoom" knob is turned from center, the faster the zooming will occur.

7.4 The Angle Distance Cursor (ADC)

The angle distance cursor is a construction aid used to measure angles and distances. It should be noted that all measurements are made in the projected space of the screen, so one should measure only in a view normal to the surface where the measurement is to take place. The ADC is placed on (or removed from) the display by pushing the "ADC" button. The ADC consists of three cursors which cover the entire screen. Figure 7.1 depicts the ADC as it appears on the screen. All the cursors are centered at the same point and can be moved to any location on the screen. Two of these cursors rotate for angle measuring purposes. Angle cursor 1 is solid while angle cursor 2 is dashed. Angle cursor 1 has movable tic marks for measuring distances on the screen. The two angle cursors move with the horizontal and vertical lines of the main cursor. The resulting effect is the moving of the center point horizontally or vertically. The ADC is controlled by the bottom row of the (Megatek) knobs:

Knob	Function
6	moves the center in the horizontal direction
7	moves the center in the vertical direction
8	rotates angle cursor 1 (alpha)
9	rotates angle cursor 2 (beta)
10	moves the tic marks

Whenever the ADC is on the screen, there is a readout at the bottom of the screen listing pertinent information about the ADC. This information includes the angles that angle cursors 1 and 2 have been rotated (alpha and beta), the

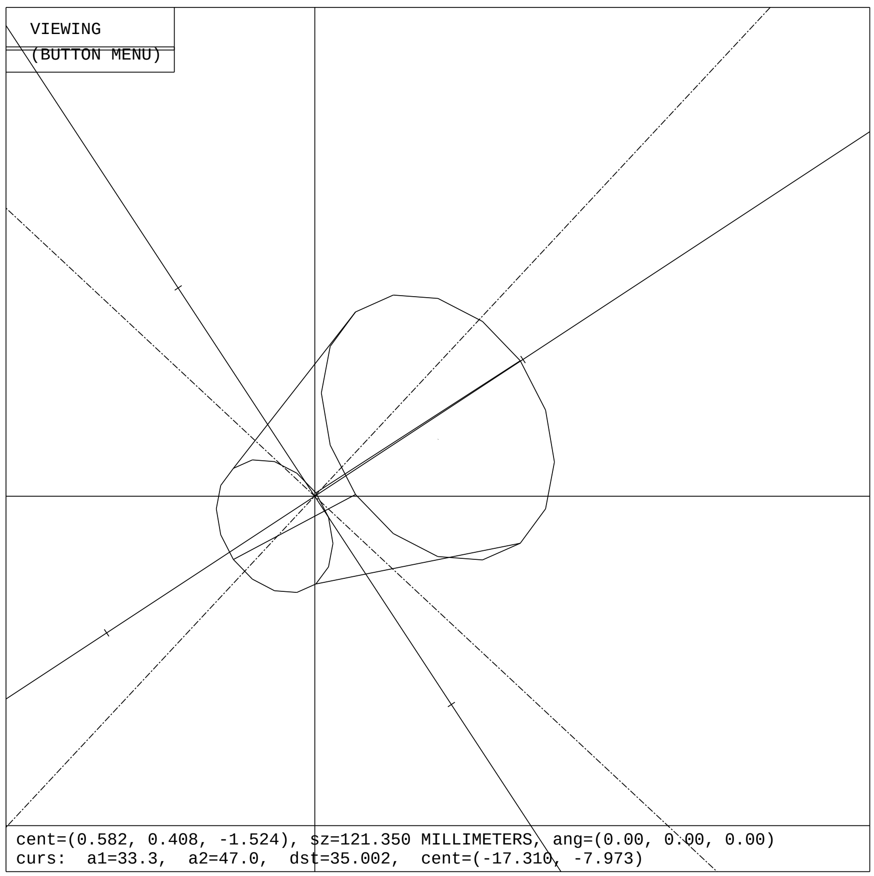

Figure 7.1: The Angle Distance Cursor

distance the tic marks are from the center of the ADC, and the location of the center of the ADC. This information is continually updated on the screen.

Chapter 8

MGED EDITING FEATURES

The heart of the MGED system is its editing features. The editing features are divided into two classes: object editing and solid editing. Object editing is designed to allow one to change the location, size, and orientation of an object. Recall that an object is defined as the basic data unit of the MGED system and includes both combinations and solids. In the case of a solid, one needs to change not only its location, size, and orientation, but also its "shape". Changing the shape of a solid means changing any of its individual parameters. Hence, solid editing is handled separately.

8.1 Combination Editing (OBJECT EDIT)

Before being able to enter the OBJECT EDIT state (i.e. edit non-terminal), it is necessary to pass through two intermediate states in which the full path of an object to be edited is specified, and the location of one arc along that path is designated for editing. It is possible to create a transformation matrix to be applied above the root of the tree, affecting everything in the path, or to apply the matrix between any pair of nodes. For example, if the full path /car/chassis/door is specified, the matrix could be applied above the node "car", between "car/chassis", or between "chassis/door".

The transformation matrix to be applied at the designated location can be created by the concatenation of operations, each specified through several types of user direction. Trees can be rotated around the center of the viewing cube; this rotation can be specified in degrees via keyboard command, or can be controlled by the rotation of a set of control dials or motions on a three-axis joystick. Translation of trees can be specified in terms of a precise new location via keyboard command, or by adjusting a set of control dials. Tree translation can also be accomplished by pointing and clicking with the mouse or tablet. Uniform and single-axis (affine) scaling of a tree can be controlled by a numeric scale factor via keyboard command, or by way of repeated analog scaling by pointing and clicking with the mouse or tablet. Before we discuss the editing features of MGED, we will discuss how one selects objects for editing.

8.2 Selecting Objects For Editing

To select a displayed object for editing, press the object illuminate button or select "Object illum" from the ***BUTTON MENU***. The object selection is a two step process.

Whenever an object is displayed (using the *e* command), all paths in the object's hierarchy are traversed recursively, accumulating the transformation matrices. When the bottom of the path (a solid) is encountered, the accumulated transformations are applied to the solid's parameters and the solid is drawn. Thus every solid displayed is really a path ending with that solid. If the object has been displayed using the *E* command, the same procedure is followed, but only until a region is encountered. Then all members of the region have the accumulated transformations applied and the region is then "evaluated" and drawn.

In the first step of the object selection process, the path is selected. Again, the data tablet is divided in as many horizontal sections as there are paths drawn. The path (solid or evaluated region) corresponding to the horizontal section the pen/mouse is located in will be illuminated (brighter on B/W displays and white on color displays). This complete path is also listed on the display. When the pen/mouse is pressed the illuminated path is selected.

In the second step, a member of the selected path is chosen. All editing will then be applied to this member. The tablet is divided into as many horizontal sections as there are path members. The word "[MATRIX]" is used to illuminate path members and will appear above the member corresponding to the location of the pen/mouse. Pressing the pen/mouse when the desired path member is "illuminated" will put MGED in the object edit state. The editing will be performed on the path member selected.

If a solid is located at the bottom of this path, it becomes the key solid and its vertex becomes the key point. If an evaluated region is at the bottom of the path, the center of this region becomes the key point. All object editing is done with respect to this key point.

The object editing features can be invoked in any order and at any time once an object has been selected for editing. During object editing, any of the viewing features, such as changing views, zooming, and slewing, can be used, and in fact, are usually quite useful. Again, the only way to exit the object editing mode is to "accept" or "reject" the editing. If the "reject" button is pressed (or selected from the edit menu), the object will return to its pre-edit state. If the "accept" button is pressed (or selected from the edit menu), the data base will be changed to reflect the object editing performed.

8.3 Object Edit State

When MGED enters the object edit state, the following occurs:

1. all the solids/evaluated regions of the edited object become illuminated
2. the key solid's parameters are labeled OR
   the center of the key evaluated region is marked
3. the key solid's parameters are listed and continually updated OR
   the key evaluated region's center is listed and continually updated
4. the ***OBJ EDIT*** menu is displayed

8.4 Translate An Object

There are three ways to translate an object: translate in the screen X direction only (X move), translate in the screen Y direction only (Y move) or just straight translation (XY move). In all cases, the complete object is translated so that the "key point" is positioned at the desired location. The *translate* command is used to enter a precise location (x,y,z) for the key point. Entering *translate x y z* will move the complete object so that the key point will be at coordinates (x,y,z).

8.5 Rotate An Object

Rotation of the object may be accomplished by selecting the "Rotate" menu item, or pressing the "Rotate" button. Turning the knobs results in the object being rotated. The *rotobj x y z* command can be used here, to specify a precise rotation in degrees. While in this edit state, the only way to rotate view is to use the *vrot x y z* command.

8.6 Scale An Object

8.6.1 Global Scale

To select global object scale, press the object scale button or select "Scale" from the ***OBJ EDIT*** menu. When the pen/mouse is pressed, the edited object is scaled about the key point by an amount proportional to the distance the pen/mouse is from the center of the screen. If the pen/mouse is above the center, the edited object will become larger. If it is below the center, the object will become smaller. The *scale* command can be used to enter precise scale factors. The value entered is applied to the object as it existed when object scale was entered. Hence entering *scale 1* will return the object to its size when the object scale session first started.

8.6.2 Local Scale

Local object scaling is allowed about any of the coordinate axes. To select local scaling, press one of the buttons (OBJ Scale X, OBJ Scale Y, or OBJ Scale Z) or select "Scale X", "Scale Y", or "Scale Z" from the ***OBJ EDIT*** menu. When the pen/mouse is pressed, the edited object is scaled in the selected coordinate axis only, about the key point. The amount of scaling is proportional to the distance the pen/mouse is from the center of the screen. If the pen/mouse is above the center, the edited object will become larger in the selected axis direction. If it is below the center, the object will become smaller in the selected axis direction. The *scale* command can be used to enter precise scale factors. The value entered is applied to the object as it existed when local object scale was entered. Hence entering *scale 1* will return the object (in the selected axis direction) to its size when the object scale session first started.

8.7 Solid Editing

There are two classes of editing operations that can be performed on leaf nodes, the primitive solids. The first class of operations are generic operations which can be applied to any type of solid, and the second class of operations are those operations which are specific to a particular type of solid. Generic operations which can be applied to all primitive solids are rotation, translation and scaling. Recall that primitives can be treated as any other object and "object edited" as detailed above. Each primitive solid also has a variety of editing operations available that are specific to the definition of that solid. These operations are detailed below.

The solid editing mode is necessary to perform to basic shapes of solids. Precise modifications of the shape are possible (using the *p* keyboard command) in the solid editing mode.

The solid editing feature allows the user to interactively translate, rotate, scale, and modify individual parameters of a solid. Whenever one is in the solid edit mode, the parameters of the solid being edited are listed and continually updated at the top of the screen. Certain parameters are also labeled on the solid being edited. Solid editing is generally used to "build" objects by producing solids of the desired shape and size in the correct orientation and position. Once the object is built, object editing is used to scale, orient, and position the object in the description. The general philosophy of solid editing is to first create a solid with the desired name and then to edit this solid. As an example, suppose one were to build an object called "BRACKET"; to produce the base of the object the primitive solid type ARB8 (see Figure 1) would be used along with either the *in* command or *make* command, so one would type:

in btm box 0 0 0 0 -90 0 40 0 0 0 0 6
make block arb8

A new solid called "btm" or "block" would be created and displayed on the screen. These solids would then be edited using solid editing to produce the solid parameters for the shape desired.

8.8 Selecting Solids For Editing

The procedure for solid editing is quite similar to that for object editing. First, solid edit state must be entered, by pressing the "solid illuminate" button, or selecting the "solid illum" menu item. Second, A solid is selected for editing using the illuminate mode, just as in object editing, by moving the cursor up and down, and choosing the desired solid. The solid data is listed at the top of the screen and a header depending on the solid type is written above the solid editing data. Third, select the appropriate function button or edit menu operations, and perform the editing desired. Finally, the solid editing mode is exited by either accepting or rejecting the editing performed.

A solid must be displayed before it can be picked for editing. To pick a displayed solid for editing, press the "solid illum" button or select "Solid Illum" from the ***BUTTON MENU***. The data tablet and pen/mouse are then used to pick the solid. The surface of the data tablet is divided into as many horizontal sections as there are solids displayed. The displayed solid corresponding to the horizontal section the pen/mouse is located in will be "illuminated" (it will become brighter on black and white devices and white on color devices). The complete hierarchical path to reach the solid is also listed on the display. When the pen/mouse is pressed, MGED enters the solid edit state with the illuminated solid as the solid to be edited. If the solid is not multiply referenced, entering *sed solidname* on the keyboard will immediately put MGED in the solid edit state with *solidname* as the edited solid.

8.9 Solid Edit State

When MGED enters the solid edit state, the following occurs:

1. the edited solid remains illuminated
2. the edited solid's parameters are labeled
3. the edited solid's parameters are listed

4. (and continually updated)

5. the ***SOLID EDIT*** menu is displayed

6. the parameter edit menu is initially displayed (default)

8.10 Rotate A Solid

Solid rotation allows the user to rotate the solid being edited to any desired orientation. The rotation is performed about the vertex of the solid. To select this option, one presses the function button labeled "solid rotate" or selects from the edit menu on screen. The rotation can be done using the dial box or one can input exact angles of rotation of the solid by using the *p* keyboard command. For example, typing:

p alpha beta gamma will rotate the solid *alpha* degrees about the x-axis, *beta* degrees about the y-axis and *gamma* degrees about the z-axis. Alpha, beta, and gamma are measured from the original "zero" orientation of the solid, defined when the "solid edit" function button was pressed. Hence, typing

p 0 0 0 will always return the solid to its original position (its position when the current solid editing session began) before accepting edit.

To select solid rotation, press the solid rotate button or select "Rotate" from the ***SOLID EDIT*** menu. The joy stick or appropriate rotation knobs then will rotate the edited solid about the coordinate axes. The solid is rotated about its vertex. The parameter (p) command can be used to make precise rotation changes. The values entered after the p are absolute – the rotations are applied to the solid as it existed when solid rotation was first selected. Thus entering *p 0 0 0* will "undo" any rotations performed since solid rotation was selected. The rotation about the z-axis is done first, then the y, then the x.

8.11 Translate A Solid

Solid translation allows the user to place the solid being edited anywhere in the description. To invoke this option, one presses the function button labeled "solid trans" or selects from the screen edit menu. To move the solid, use the mouse pointer to position the solid and click the center mouse button. Whenever the mouse button is pressed, the VERTEX of the solid moves to that location on the screen.

One can read the actual coordinates of the vertex on the top of the screen, along with other data. If the actual desired coordinates of the vertex are known, one can place the solid exactly using the *p* keyboard command. For example, to place a solid's vertex at the coordinates (x,y,z) one would type:

p40 20 10 The solid would then jump to this location.

To select solid translation, press the solid translate button or select "Translate" from the ***SOLID EDIT*** menu. When the pen/mouse is pressed, the vertex of the edited solid will move to that location. The parameter (p) command can be used to translate the solid to a precise location. Entering *p x y z* will place the vertex of the edited solid at (x, y, z).

8.12 Scale A Solid

The solid SCALE feature allows the user to scale the solid being to any desirable size. The scaling is done about the vertex of the solid, hence NO translation of the solid occurs. The scaling is performed using the mouse pointer and clicking the center mouse button, just as in object scaling. One can input an exact scale factor using the *p* keyboard command, in the form of. For example, typing

p factor will scale the solid by an amount equal to *factor*. The value of *factor* is absolute – the original solid is scaled. By setting *factor* equal to one (1), the original size solid will be displayed on the screen before accepting your edit.

To select solid scale, press the solid scale button or select "Scale" from the ***SOLID EDIT*** menu. When the pen/mouse is pressed, the edited solid is scaled by an amount proportional to the distance the pen/mouse is from the center of the screen. If the pen/mouse is above the center, the edited solid will become larger. If it is below the center, the solid will become smaller. The parameter (p) command can be used to enter precise scale factors. The value entered is applied to the solid as it existed when solid scale was entered. Hence entering *p 1* will return the solid to its size when solid scale session first started.

8.13 Solid Parameter Editing

To modify individual solid parameters, press the menu button or select "edit menu" from the ***SOLID EDIT*** menu. A menu listing what parameter editing is available for that particular solid type will be displayed. Using the pen/mouse select the desired item(s) from this menu. For most of the parameter editing, the *p* command can be used to make precise changes. Parameter editing is the default edit mode entered when MGED first enters the solid edit state. In the following paragraphs, we will discuss parameter editing for each of the MGED general types of solids.

8.13.1 ARB Parameter Editing

The GENERAL ARB class of solids represents all the convex polyhedrons (RPP, BOX, RAW, and ARBs). The ARBs comprise five classes of polyhedrons each with a characteristic number of vertices. These are the ARB8, ARB7, ARB6,

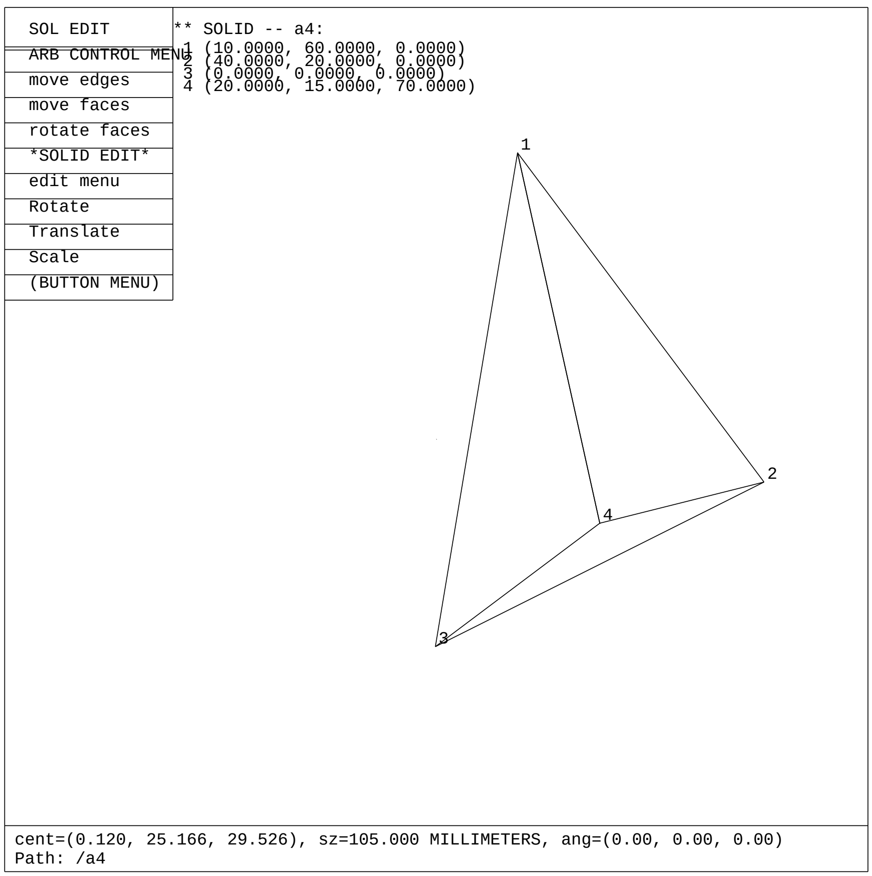

Figure 8.1: ARB Control Menu

ARB5, and ARB4, where the ARB8 has eight vertices, etc. During editing, all the vertices are labeled on the screen.

An ARB is defined by a fixed number of vertices where all faces must be planar. This fact means that during parameter editing, movement of individual vertices in faces containing four vertices is not allowed. There are three classes of parameter editing that can be done to ARBs: move edges, move faces, and rotate faces. There is an "ARB control menu" (see Figure 8.1) from which one selects the type of parameter editing to be done. A specific ARB edit menu will appear dependent on which parameter editing option was selected. The "return" entry on each of these specific menus will return the "ARB control" menu to the screen.

Note that there are several keyboard commands that apply only to ARB solids which are being edited in SOLID EDIT state. Once such command is *mirface*, which replaces a designated face of the ARB with a copy of an original face mirrored about the indicated axis. Another such command is *extrude*, which projects a designated face a given amount in the indicated direction.

8.13.2 Move ARB Edges

To move an ARB edge, select the desired edge from the "move edge" menu. For example, Figure 8.2 shows the menu for moving an edge of an ARB8, and Figure 8.3 shows the menu for moving an edge of an ARB4. A point is then "input" either through a pen press or through the *p* command. The line containing the selected edge is moved so that it goes through coordinate of the input point. Any affected faces are automatically adjusted to remain planar.

8.13.3 Move ARB Faces

To move an ARB face, select the desired face from the "move face" menu. A point is then "input" either through a pen press or through the *p* command. The plane containing the edited face is then moved so that it contains the input point. The new face is then calculated and the ARB is displayed. The move face menus for an ARB8 are shown in Figure 8.4, and the move face menus for an ARB4 are shown in Figure 8.5.

8.13.4 Rotate ARB Faces

ARB faces may be rotated around any of the vertices comprising that face. First, select the desired face from the "rotate face" menu. You will then be asked to select the vertex number around which to rotate the face. The face can be rotated about the three coordinate axes. The knobs (Rotate X, Rotate Y, and Rotate Z) are used for this purpose. For precise rotations, use the *p* command. If three values are entered after the *p*, then they are interpreted as angles (absolute) of rotation about the X, Y, Z axes respectively. If only two

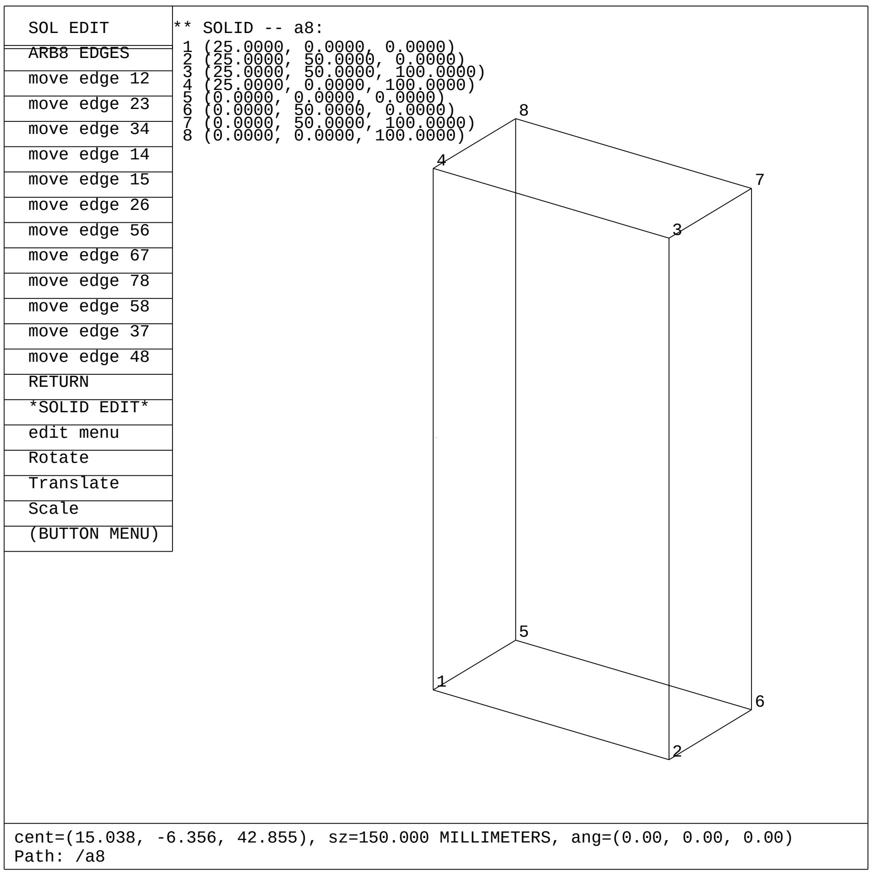

Figure 8.2: Move Edge Menu for ARB8

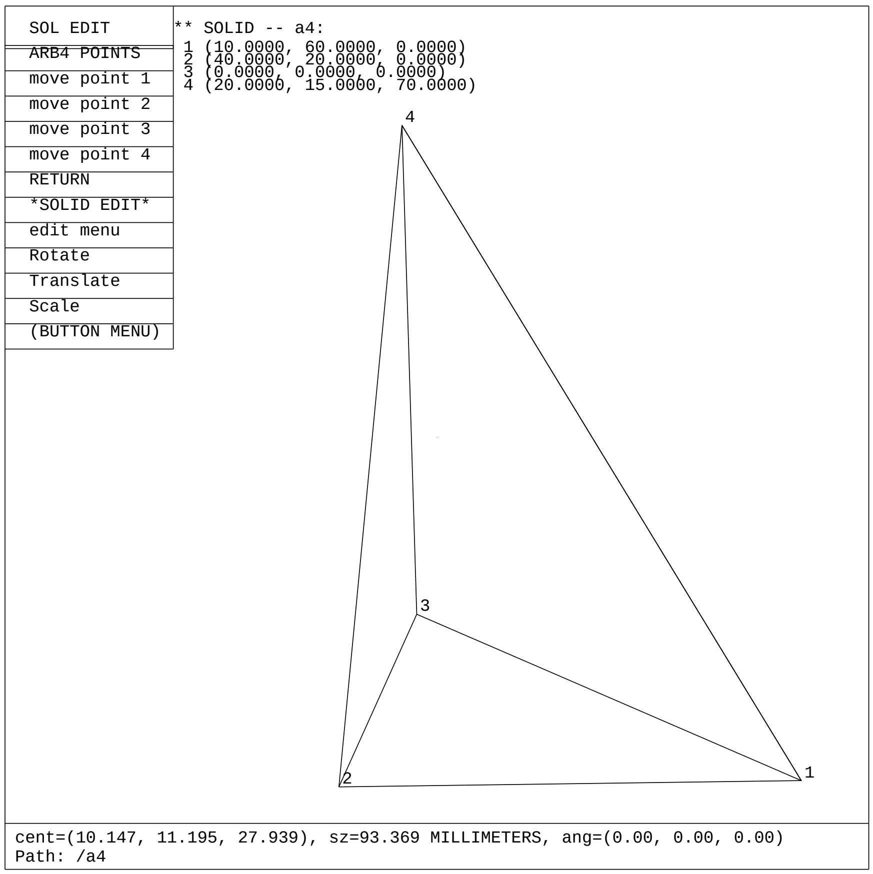

Figure 8.3: Move Edge Menu for ARB4

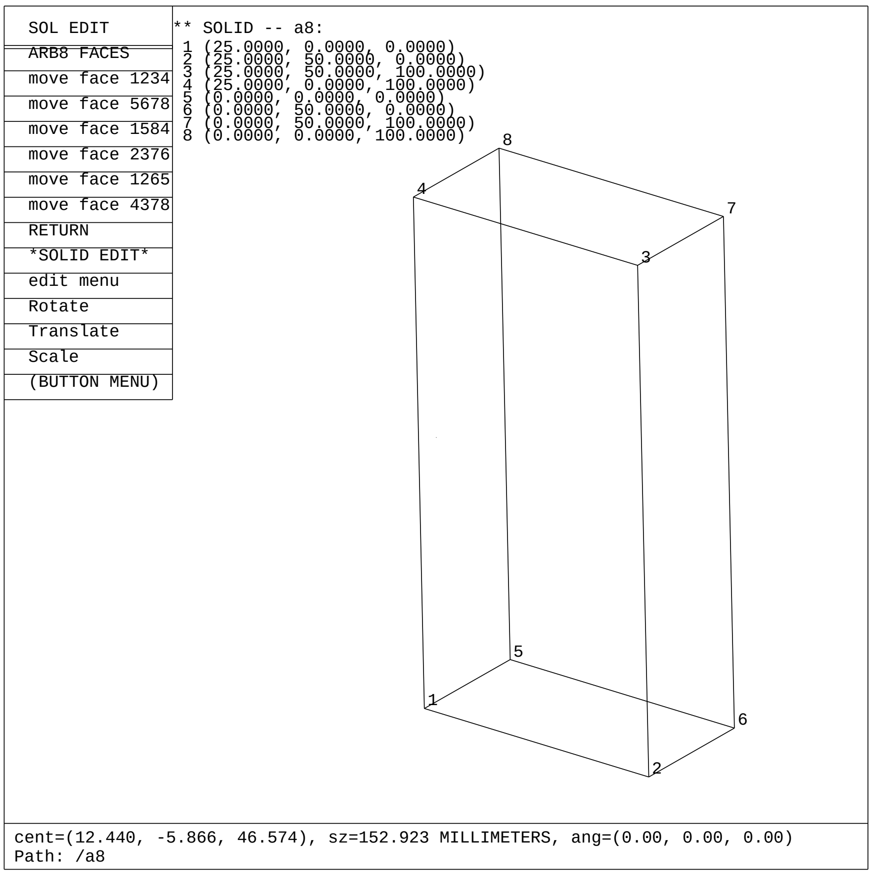

Figure 8.4: Move Face Menu for ARB8

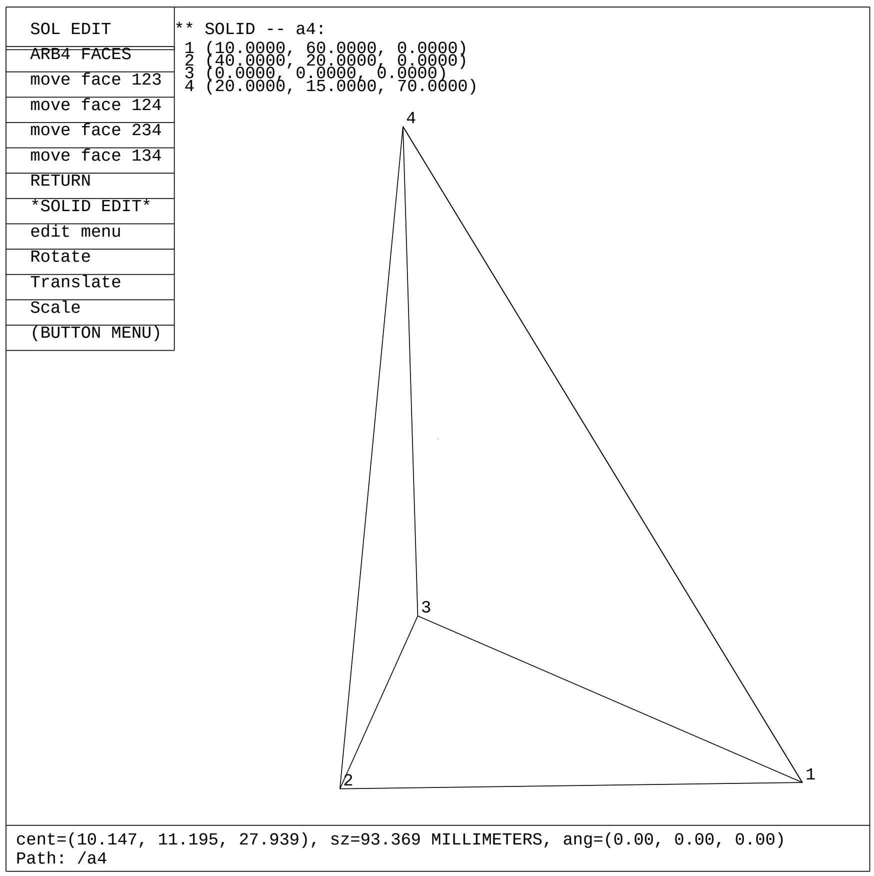

Figure 8.5: Move Face Menu for ARB4

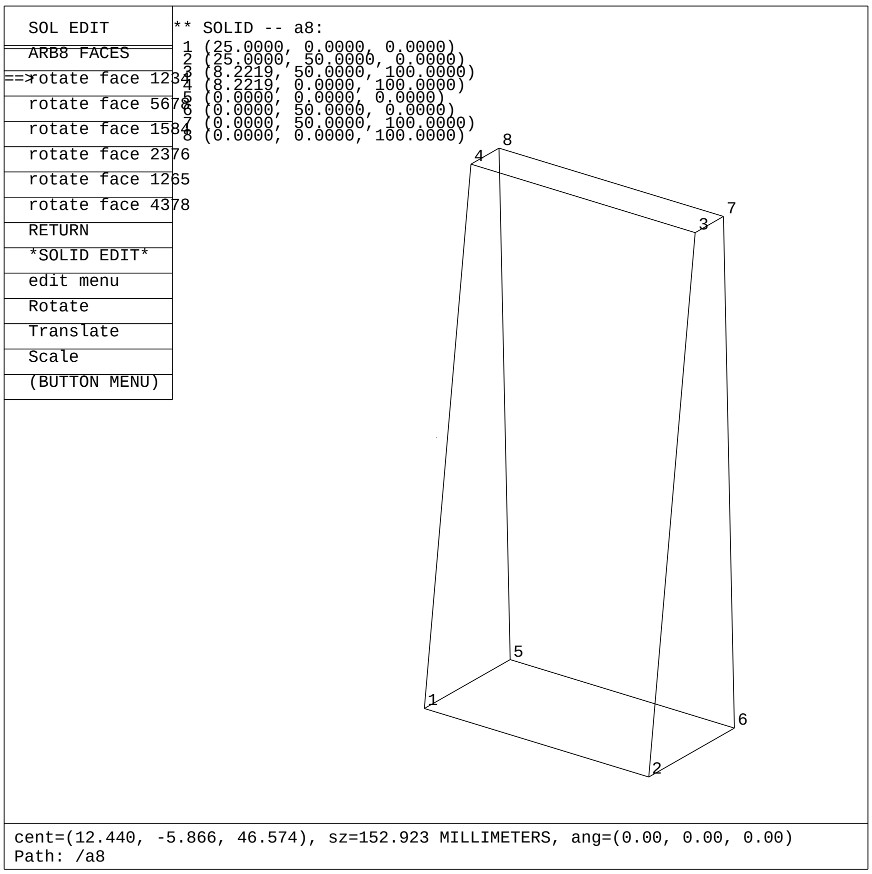

Figure 8.6: Rotate Face Menu for ARB8

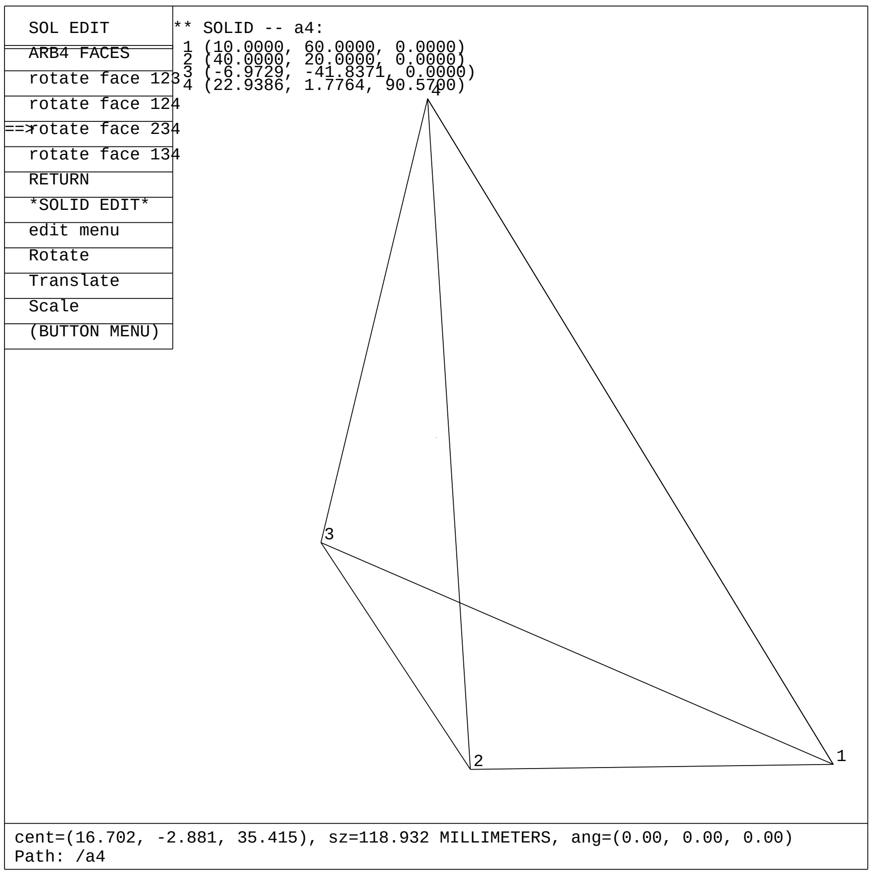

Figure 8.7: Rotate Face Menu for ARB4

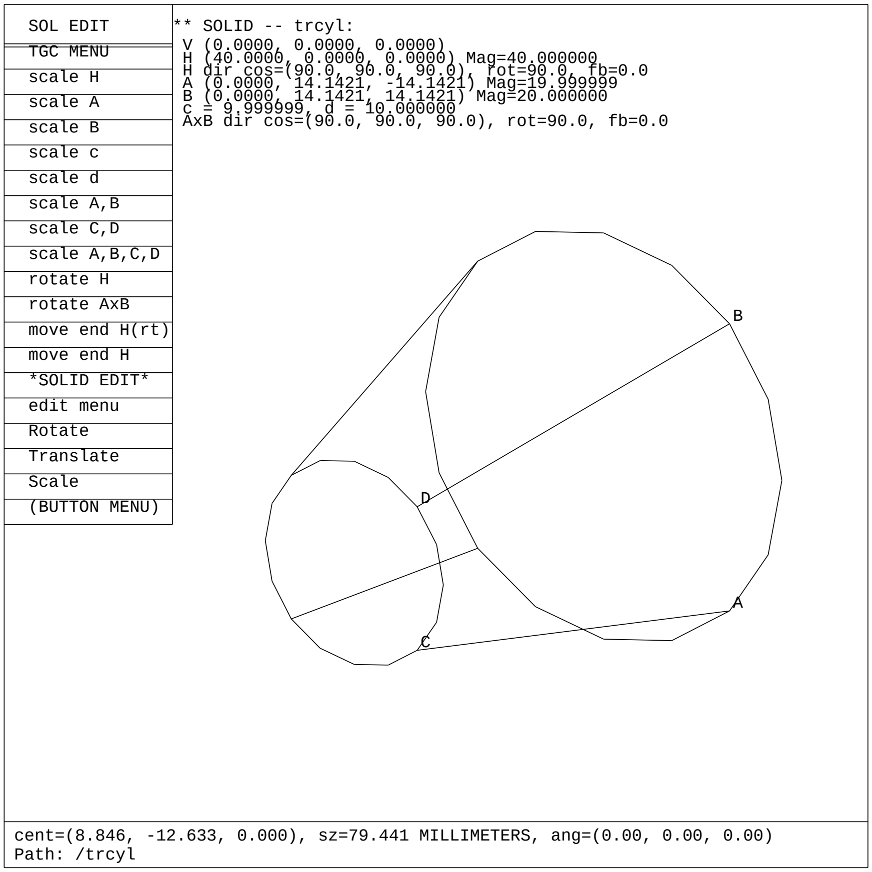

Figure 8.8: Typical TGC During Parameter Editing

values are entered, then they are considered as rotation and fallback angles for the normal to that face. The *eqn* command can also be used here to define the plane equation coefficients of the face being rotated. The rotate face menus for an ARB8 are shown in Figure 8.6, and the rotate face menus for an ARB4 are shown in Figure 8.7.

8.13.5 Truncated General Cone (TGC) Parameter Editing

The TGC general class of solids includes all the cylindrical COMGEOM solids. The defining parameters of the TGC are two base vectors (A and B), a height vector (H), two top vectors (C and D), and the vertex (V). The top vectors C and D are directed the same as the base vectors A and B respectively, hence the top vectors are defined only by their lengths (c and d). During solid editing, only vectors A and B are labeled on the display. Figure 8.8 depicts a typical TGC during parameter editing.

It is possible to change the length of the H, A, B, C, or D vectors, resulting in a change in height or eccentricity of the end plates. The overall size of the A,B or C,D end plates can be adjusted, or the size of both can be changed together, leaving only the H vector constant. The H vector or the base plate (AXB) can be rotated. Recall that vectors A & C and vectors B & D have like directions, hence rotating the base (AXC) will automatically rotate the top (BXD). Finally, one can move the end of the height vector H with the TGC becoming or remaining a right cylinder (move end H (rt)), or with the orientation of the base (and top) unchanged (move end H). Either the mouse/tablet or the *p* command can be used. These functions are selected from the menu which can be seen in Figure 8.8.

8.13.6 Ellipsoid Parameter Editing

The ELLG general class represents all the ellipsoidal solids, including spheres and ellipsoids of revolution. The defining parameters of the ELLG are three mutually perpendicular vectors (A, B, and C) and the vertex (V). When an ELLG is being edited, only vectors A and B are labeled on the display. Figure 8.9 depicts a typical ELLG during parameter editing.

The parameter editing of the ELLG consists of scaling the lengths of the individual vectors A, B, C. One may also scale all theses vectors together of equal length.

The scaling of these vectors is done using the data tablet/mouse in exactly the same manner as in object scaling. The *p* keyboard command again can be used to produce a vector of desired length.

8.13.7 Torus Parameter Editing

The TOR general class of solids contains only one type of torus, one with circular cross-sections. The defining parameters of the TOR are two radii (r1 and r2), a normal vector (N), and the vertex (V). The scalar r1 is the distance from the vertex to the midpoint of the circular cross section. The scalar r2 is the radius of the circular cross-section. The vector N is used to orient the torus. During solid editing, none of these parameters are labeled on the screen. Figure 8.10 depicts a typical torus during parameter editing.

The parameter editing of the TOR consists of scaling the radii, hence the menu contains only two members.

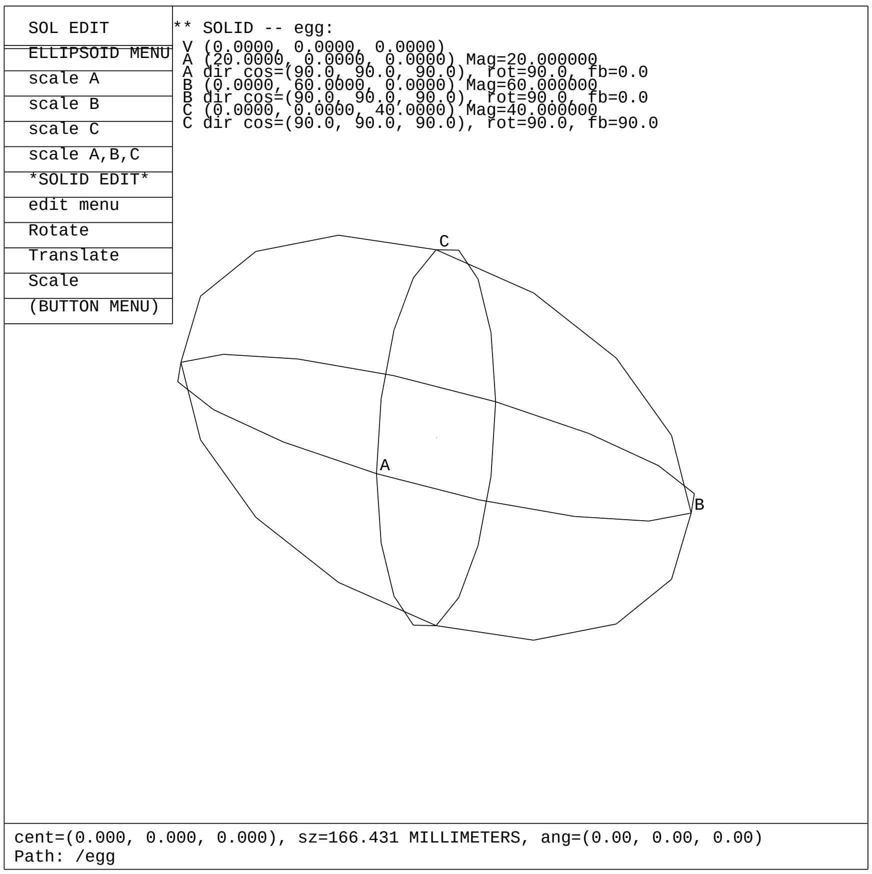

Figure 8.9: Ellipsoid Parameter Editing Menu

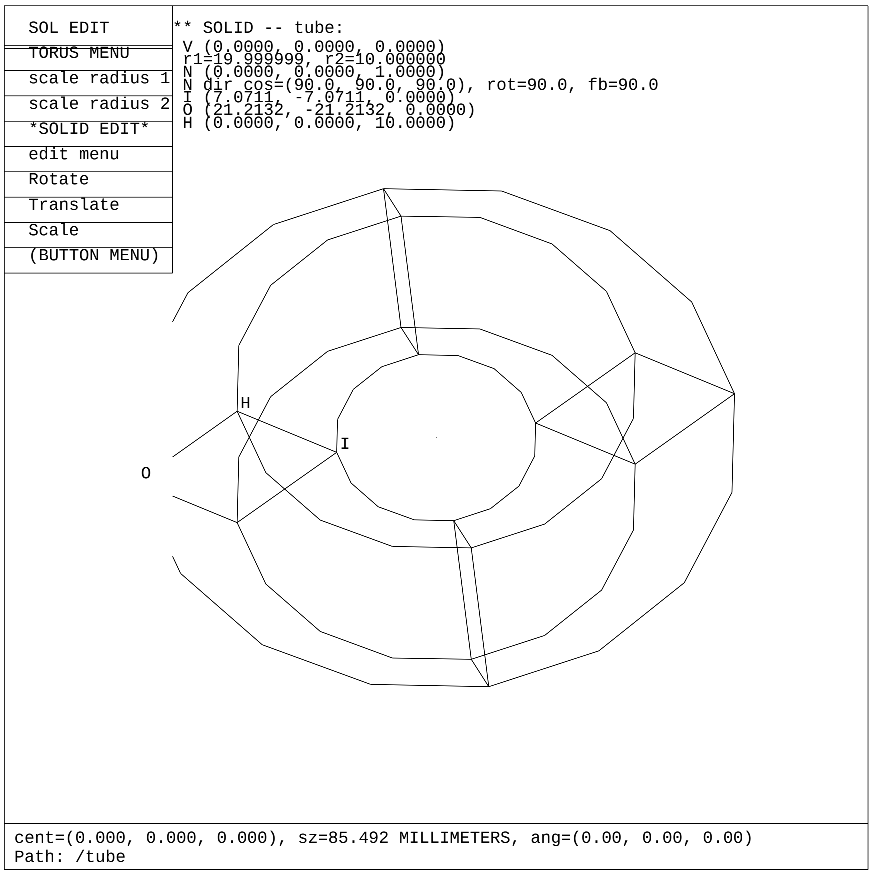

Figure 8.10: Torus Parameter Editing Menu

Chapter 9

KEYBOARD COMMANDS

The MGED keyboard commands are used to maintain overall control of the system and to perform general housekeeping functions. They are summarized in Figure 9.1.

9.1 Copy Object

cp oldobj newobj

This command is used to produce a copy of an object (solid or comb). In this case, the object "oldobj" will be copied into an object called "newobj".

Examples: *cp arb8 hullbot.s*
cp tgc wheelrim.s
cp torso.r driver_torso
cp proto.man driver

9.2 Zap Screen

Z

This is the Zap command. It clears all objects from the screen.

9.3 Drop objects from display screen

d obj1 obj2 ... objn

This command allows one to remove objects from the display screen. In this case "obj1" thru "objn" will be removed from the display.

9.4 Move (rename) object

mv old new

This command is used to rename objects in the data file. In this case, the object "old" will be renamed "new". A note of caution: the name is changed only in the object record itself, not in any member records. Thus if the object "old" appears as a member of any other object, the name will not be changed there. To rename all occurrences of an object, use the "mvall" command.

Examples: *mv test hull*
mv g00 air
mv g1 turret

9.5 Set Local Working Units

units ab

This command allows one to change the local or working units at ANY time. The only allowable values for "ab" are "mm", "cm", "m", "in", or "ft".

Examples: *units mm*
units in

9.6 Group objects

g group obj1 obj2 ..... objn

This command creates or appends to a combination record and is used to group objects together either for editing or displaying purposes. In this case, "obj1" through "objn" are added as members to the combination "group". If "group" does not exist, it is created and "obj1" through "objn" are added as members. NOTE: no checking to see if "obji" is already a member of "group".

Examples: *g shell hull turret*
g tank wheels engine crew shell
g tank track

9.7 Create Region

r region op1 sol1 op2 sol2 .... opn soln

This command is used to create regions or append to regions. If "region" exists, then solids "sol1" through "soln" are added as members with "op1" through "opn" as the defining operations. If "region" does not exist, then it is created and solids "sol1" through "soln" are added as members with "op1" through "opn" as the defining operations. A region is merely a combination record with a flag set and is distinguished from other combinations (groups) since it has meaning to the COMGEOM solid modeling system. Note that "+" or "u" must be the first operations in a region.

When a region is created, the item and air codes are set equal to default values. If the "regdef" command has been used, then those values will be used, otherwise the values "1000 0 100 1" will be used respectively. To change the item and air codes use the "item" command. The "edcodes" command is probably the easiest and fastest way to change these identifing codes. Note: In the past, all members of a region had to be solids, but recently combinations have been allowed as members of regions. Hence, the names "soli" can also be combinations (groups) now. Also, as in grouping, no checking for members already in a region.

Examples: *r hulltop.r + hulltop.s – hullleft.s – hullright.s*
r gun + gun.s – gunin.s
r gunair + gunin.s

9.8 Instance an object

i object combname

This command is used to make an instance of an object. An instance of an object is produced by creating a combination and making the object a member. In this case, an instance of "object" is made by creating the combination record "combname" (if "combname" does not already exist) and adding "object" as a member. If "combname" already exists, then "object" is added as the next member.

An instance is used to refer to an object, without making actual copies of the object. Instances are useful when one is adding a certain component to a target description many times. Furthermore, any modifications to an object which has been instanced need only be done in the original (prototype) object. These modifications will then be automatically reflected in all the instances of the object.

Examples: *i heround he1 .he1.*
i heround he2 .he2.
i heat heat1 .heat1.
i heat heat2 .heat2.

9.9 Change Title of Database

title newtitle

This command allows one to change the title of the model database at any time. The string "newtitle" will become the new title, and may contain blanks. The title is limited to 72 characters including blanks.

Examples: *title XM89A – New version of tank*
title M345 (groups are m345 and m345a)

9.10 Extrude

extrude #### distance

This command allows the user to project a face(####) of an arb being edited a normal distance to create a new arb. The value of "face" is 4 digits such as 1256. If the face is projected in the wrong direction use a negative "distance". One common use for this command is for producing armor plates of a desired thickness.

Examples: *extrude 1234 20*
extrude 2367 34.75
extrude 2367 -34.75

9.11 Remove members from Combination

rm comb mem1 mem2 .... memN

This command allows one to delete members from a combination (group or region) record. In this case, members "mem1" through "memn" will be deleted from the combination "comb".

Examples: *rm tank hull wheels*
rm region1 solid8 solid112
rm turtop.r tursidel.s tursider.s

9.12 List Object Information

l object

This command is used to list information about objects in the data file. The information listed depends on what type of record "object" is. If "object" is a combination record, then the members are listed. If "object" is a solid record, then the MGED general solid type and the parameters as presently in the data file are listed. Note: only the solid parameters as they exist in the solid record are listed, no transformation matrix is applied. Hence, if the solid was edited as a member of a combination, the "l" command will not reflect the editing in the listed parameters. To produce this type of listing, see the "listeval" command.

Examples: *l hull*
l turret
l turtop.s
l arb8

9.13 Analyze Solid

analyze solid

This command produces information about a solid (all except ARS). The information includes surface area(s) and volume. Also, in the case of ARBs, edge lengths and rot and fallback angles and plane equations are given for each face. If "solid" is present that solid name will be used and analyzed. If "solid" is absent, the solid at the bottom of the present path being edited will be analyzed.

9.14 Mirror Object

mirror oldobj newobj axis

This command is used to create a new object which is the mirror image of an existing object about an axis. The object may be either a solid or a combination. In this case, a mirror image of the object "oldobj" will created about the axis indicated by "axis" and the new object record will be called "newobj". The only acceptable values for the parameter "axis" are "x", "y", and "z".

Examples: *mirror tur.left.s tur.right.s y mirror tur.top.s tur.bot.s z mirror tur.front.s tur.back.s x mirror lt_gun rt_gun y*

9.15 Create ARB8

arb name rot fb

This command allows one to create an arb8 with the desired rotation and fallback angles. In this case, an arb8 with the name of "name" will be created with the desired rotation angle of "rot" degrees and the fallback angle of "fb" degrees.

Examples: *arb top1.s 0 90*
arb sidelt.s 90 0
arb upglacis.s 0 60

9.16 Change Item (Ident) and Air codes of Region

item region ident air

This command allows one to change the item or air code numbers of a region. If the air code ("air") is not included, a zero is assumed. To change the air code, a zero item code must be used (see second example below).

Examples: *item region1 105*
item region7 0 2
item region11 129 0

9.17 Specify Material Properties

mater comb [material]

This command is used to change the material properties specification for a combination.

9.18 Edit Combination Record Info

edcomb comb regionflag regionid air los GIFTmater

This command is used to change the material and the los percent for a region.

9.19 Edit (Display) an object on the screen

e obj1 obj2 ... objn

This command allows one to display objects on the screen. In this case, "obj1" thru "objn" will be displayed on the screen.

9.20 Evaluated Display of Object on the screen

E obj1 obj2 ... objn

This command is the same as the "e" command, except the regions will be evaluated before being displayed.

9.21 Zap screen and Display Object

B obj1 obj2 ... objn

This command is the same as the "e" command, except that the screen is cleared (Zap) before the objects are displayed.

9.22 Kill (delete) object from database

kill obj1 obj2 ... objn

This command allows one to remove objects from the file itself. Only the object records themselves are removed, any references made to these objects still will exist. To remove the references also, see the "killall" command.

9.23 List Table of Contents

t obj1 obj2 ... objn

This is the table of contents command. If arguments are present, a list of all objects in the file matching these names will be printed. If there are no arguments, then a listing of all objects will be printed.

9.24 List Tree Tops

tops

This command will search the target file hierarchy, and list all "top level" objects (objects which are not members of any other object). This command is useful to make sure objects have been grouped properly.

9.25 Make prototypical solid

make name type

This command will create a solid of a specified type. This solid will be named "name" and the solid type will be "type". The acceptable types are: arb8, arb7, arb6, arb5, arb4, tor, tgc, tec, rec, trc, rcc, ellg, ell, sph. This new solid will be drawn at the center of the screen. This command should be used to create solids for editing.

9.26 Mirror ARB Face

mirface #### axis

This command allows one to mirror a face of an edited arb about an axis. This command is quite useful for adding air to a "symmetric" target.

9.27 Print Summary of Objects

summary s—r—g

This command will produce a summary of objects in the target file. If the options s, r, or g are entered a listing of the solids, regions, or groups will also be presented.

9.28 Specify Numeric Parameter(s)

p dx [dy dz]

This is the parameter modification command and is used during solid editing to make precise changes. The actual meaning of the values typed after the "p", depend on what editing option is being performed. If one were translating a solid, then the values would be the x,y,z coordinates of the vertex of the solid.

9.29 Release Current Display

release This command releases the current display device, and attaches the null device.

9.30 Attach to Display Device

attach device

This command allows one to attach a display device (the present display device is released first). The present acceptable values for "device" are vg, mg, tek, rat, plot, tek4109, ir, sgi, and nu. The "plot" device will produce a UNIX-plot of the present view (including the faceplate) on a display device using a specific filter. You will be asked which filter to use. Sample filters include "tplot" and "plot-fb".

9.31 Numeric Object Rotation Edit

rotobj xdeg ydeg zdeg

This command allows one to make precise rotations of an object during object editing. MGED must be in the "object edit" state for this command to have effect. If "object rotation" is not in effect, MGED will select this option for you and perform the rotation. The object will be rotated "xdeg" about the x-axis, "ydeg" about the y-axis, and "zdeg" about the z-axis. The rotation is "absolute"....the total rotation since the beginning of object editing will be equal to the input values. The rotation is done about the "KEY" point for the object being edited.

9.32 Scale Edited Object

scale xxx.xx

This command allows one to make precise scaling changes to an object during object editing. MGED must be in the "object edit" state for this command to have effect. If one of the object scale options is not in effect, the "global scale" options will be selected. The object will be scaled by a TOTAL amount equal to the input value. If one of the local scale options is in effect, the object will be scaled in the selected axis direction by an amount equal to the input value. The scaling is done about the "KEY" point of the object being edited.

9.33 Translate Edited Object

translate xxx.xx yyy.yy zzz.zz

This command allows one to make precise translation changes to an object during object editing. MGED must be in the "object edit" state for this command to have effect. If the object translation option is not in effect, this option will be selected and the translation performed. The "KEY" point of the object being edited will move to the input coordinates.

9.34 Fix Broken Hardware (sometimes)

fix

This command will "fix" (restart) the display device after a hardware error.

9.35 Ray-Trace Current View

rt [-s#]

This command will run the **rt**(1) program to produce a color shaded iamge of objects on the currently selected framebuffer. The resolution of the iamge (number of rays) is equal to "#" from the "-s" (square view resolution) option. If the -s option is absent, 50x50 ray resolution will be used.

9.36 Emulate Knob Twist

knob id value

This command is used to emulate a "knob twist". Generally this command is used for display devices which have no actual knob peripherals (eg. tek). Any non-zero number entered for "value" is converted to 1 (if "value" is greater than zero) or is converted to -1 (if "value is less than zero). The user must enter the same command with "value" equal to zero to stop the action envoked by the knob twist.

The "id" defines which knob is to be "twisted":

x	rotates about x-axis
y	rotates about y-axis
z	rotates about z-axis
X	slew view in x direction
Y	slew view in y direction
Z	zoom the view

Examples: *knob x 1*
knob x 0
knob Z -1
knob Z 0

9.37 Solid_Edit Named Solid

sed name

This command allows one to immediately enter the solid edit mode with the solid "name" as the edited solid. Note that the solid must be displayed but not multiply referenced.

9.38 Illuminate Named Object

ill name

This command is used to illuminate an object ... a path containing this object ("name") will be illuminated. This command is primarily used with display devices which do not have a tablet to pick objects for editing.

9.39 Rotate the View

vrot xdeg ydeg zdeg

This command rotates the VIEW "xdeg" degrees about the screen x-axis, "ydeg" degrees about the screen y-axis, and "zdeg" degrees about the screen z-axis.

This command is useful when the precise rotation desired is known. It is also useful when in a rotation edit mode, and the viewing rotation needs to be changed, without affecting the current edit.

9.40 Move Screen Center

center xx.xx yy.yy zz.zz

This command moves the screen center to (xx.xx, yy.yy, zz.zz). Using this command is one way of slewing the view.

9.41 Set View Size

size xx.xx

This command sets the view size to xx.xx and is one way of zooming the display. Making the view size smaller has the effect of zooming in on the view.

9.42 Extended List of all Objects in Displaylist

x

This command produces a list of all objects displayed, listing the center of the object, its size, and if it is in the present view. It is intended primarily for software debugging.

9.43 Refresh Display

refresh

This command will send a new display list to the display device.

9.44 Print View Status

status

This is a debug command which prints the status of the current view, including all viewing and editing matrices.

9.45 Simulate Button Press

press button-label

This command allows one to emulate a button press and is generally used on display devices which do not have actual button peripherals. The following are the strings allowed for "button-label" and all produce the indicated view on the device screen: top, bottom, right, left, front, rear, 90,90, 35,25

The following is a listing of the remaining acceptable strings for "button-label" and the resulting action:

reset	reset the view
save	save the present view
restore	restor the saved view
adc	display the angle-distance cursor
oill	begin object illumination (pick)
sill	begin solid illumination (pick)
oscale	object scale
ox	object translat ion in x direction only
oy	object translation in y direction only
oxy	object translation
orot	object rotation
sedit	put up solid parameter menu
srot	solid rotation
sxy	solid translation
sscale	solid scale
accept	accept editing done
reject	reject editing done

Examples: *press 90,90*
press front
press oill
press orot
press reject

9.46 Escape to the Shell

This command allows one to escape to the shell to perform multiple commands without having to terminate the current MGED session. To return to mged, enter a control-d to the Shell. Note that the "!" escape at the beginning of a line can be used to send a single command to the shell.

9.47 Get Short Help Listing

?
This is the short form of the help command, that lists the names of all MGED commands.

9.48 Get Long Help Listing

help
This is the long form of the help command, that produces a listing of all the available commands and their arguments, and a one sentence summary of the

commands purpose.

9.49 Exit (Quit) MGED

q

Running the "q" command, or entering an End-Of-File (EOF) (typ. Control/D) is the normal way of exiting MGED.

9.50 Copy and Translate TGC

cpi oldtgc newtgc

This command is a specialized copy command and is designed to be used when one is "running wires" in a description. The object being copied must a cylindrical solid (TGC). The following occurs when cpi is used: first the cylinder ("oldtgc") is copied to "newtgc"; then "newtgc" is translated to the end of "oldtgc"; then "newtgc" is displayed; and finally, MGED is put in the SOLID EDIT state with "newtgc" as the edited solid.

9.51 Remove Object and All References

killall obj1 obj2 ... objn

This command will accomplish two things: first, the object[s] will be removed from the data file just as in the "kill" command; second, all references to the object[s] will also be removed.

9.52 Remove Complete Tree

killtree obj1 obj2 ... objn

This command will remove from the file complete trees originating with obj1, obj2, ..., objn. Every object in the designated paths will be removed from the file, hence CAUTION is urged. Make sure that "killtree" is what you want to do. Using the "paths" or "tree" command on an object before "killtree" will show what objects will be killed.

9.53 Add Color To Display

color low high r g b string

This command allows one to make color assignments to a range of item codes. The arguments "low" and "high" are the item ranges. The arguments "r", "g", and "b" are the red, green, and blue values respectively (the range of these numbers is generally 0-255). The argument "string" is a string describing this class of items. A blank is considered a terminator, so there can be no blanks in this string.

9.54 Edit Display Colors

edcolor

This command allows one to edit the existing color assignments (table). The changes are made using the user's "selected" text editor found in environment variable EDITOR. Note that this method of specifying object colors is obsolete, and has been replaced by the *mater* command.

9.55 Print Display Colors

prcolor

This command prints the color assignments as they presently exist.

9.56 Find Objects

find obj1 obj2 ... objn

This command will find ALL references of obj1 obj2 ... objn in the file.

9.57 Estimate Presented Area

area [tolerance]

This command finds an estimate of the presented area of all E'd objects in the present view from that aspect. The argument "tolerance" is the tolerance for the endpoints of line segments being "equal" and is optional.

9.58 Produce UNIX Plot

plot [-zclip] [-2d] [out-file] [— filter]

This command is used to produce a UNIX-plot hardcopy of the present view of the 'geometry' on the display device. The MGED faceplate will not be drawn. Some useful examples are:

plot-fb	LIBFB framebuffer (low res)
plot-fb -h	LIBFB framebuffer (high res)
tplot -Ti10	Imagen laser printer
tplot -Tmeg	Megatek 7250
tplot -T4014	Tek4014
tplot -Thpgl	HP7550A plotter

9.59 Text Edit Solid Parameters

ted

This command allows one to edit a solid's parameters using the text editor defined by the users's path. The solid being edited (solid edit mode) will be the one that will be "text edited".

9.60 Keyboard Input of Solid Parameters

in name type parameters

This command allows one to enter a new solid directly via the keyboard. The user will be prompted for any missing input and the solid will be displayed on the screen. WARNING: only minimal checking of parameters is done.

9.61 Change View Azimuth, Elevation

ae az elev

This command sets the display viewing angles using the input azimuth (az) and elevation (elev) angles.

9.62 Define Region Identifiers

regdef item [air los mat]

This command allows one to change the default codes which will be given to the next region created. If the "air" code is non-zero, then the "item" code will be set to zero.

9.63 Change Edge Direction

edgedir delta_x delta_y delta_z

This command allows one to define the direction of an ARB edge being moved. If only two arguments are input, the code assumes these to be the rotation and fallback angles for the edge. If three arguments are present, the code will use them as "deltas" to define the direction of the edge. Note that this command can be useful to find the intersection of a line (edge) with planes (faces).

9.64 Prefix Object Names

prefix string obj1 obj2 ... objn

This command will prefix obj1, obj2, ... objn with "string". All occurences of these names will be prefixed. String matching is allowed for the objects to prefix.

9.65 Keep Objects in Another File

keep file.g obj1 obj2 ... objn

This command allows one to keep the listed objects in the file "file.g". This command is useful for pulling out parts of a description.

9.66 List Object Hierarchy

tree obj1 obj2 ... objn

This command will print the tree structure for the objects listed.

9.67 Create Inside Solid

inside

This command is used to define a solid (inside solid) such that when it is subtracted from another certain solid (outside solid), the resulting region will have desired thicknesses. To invoke this option, enter "inside". You will then be prompted for the required data. If you are in the solid edit mode, the "outside solid" will default to the solid presently being edited. If you are in the object edit mode, the "outside solid" will default to the "key solid" of the object path being edited. If you are not in an edit mode, you will then be asked for the name of the "outside solid". Next, you will be asked what name you wish to call the new "inside solid" to be calculated. Finally, you will be asked to enter the thickness[es], depending on the solid type. The "inside solid" will then be displayed on the screen. If the thickness values input are negative, then the thickness will be directed to the "outside" of the solid.

9.68 Produce ASCII Summary of Solids

solids file obj1 obj2 ... objn

This command will produce an ascii summary of all solids involved with objects obj1, obj2, ..., objn. This summary will be written in 'file'. In this file, all regions and solids are numbered and will match the numbers of a COMGEOM deck produced by VDECK or GIFT if the objects are entered in the same order. The file 'file' will be overwritten if it already exists. String matching is allowed for the objects.

9.69 Produce ASCII Summary of Regions

regions file obj1 obj2 ... objn

This command will produce an ascii summary table of all regions involved with objects obj1, obj2,...,objn. This summary table will be written in 'file'. In this file, all regions and solids will be numbered and will match the numbers of a COMGEOM description produced by VDECK or GIFT if the objects are entered in the same order. This file will be identical to the "solids" command except the actual parameters are not listed. The file 'file' will be overwritten if it already exists. String matching is allowed for the object names.

9.70 Produce ASCII Summary of Idents

idents file obj1 obj2 ... objn

This command will produce an ascii summary table of all region idents involved with objects obj1, obj2,...,objn. This summary table will be written in 'file'. In this file, all regions will be numbered and will match the region numbers of a COMGEOM description produced by VDECK or GIFT if the objects are entered in the same order. At the end of this file will be the same region information, but ordered by ident number. The file 'file' will be overwritten if it already exists. String matching is allowed for the object names.

9.71 List Objects Paths

paths

This command will print ALL paths matching an "input" path. You will be asked to enter the path to match. The path members are entered on ONE line, seperated by spaces. The input path need not be a complete path, but must contain at least one object. All paths with the same first members as the input path will be listed. This command is useful for finding complete paths which begin with certain objects.

9.72 List Evaluated Path Solid

listeval

This command will "evaluate" and list ALL paths matching an input path, including the parameters of the solids at the bottom of each path. These parameters will reflect any editing contained in the path listed. You will be asked to enter the path to match. The path members are entered on ONE line, seperated by spaces. The input path need not be a complete path, but must contain at least one object. Note that since the solid parameters are printed, this could be a rather lengthy listing depending on the completeness of the input path.

9.73 Evaluate Path and Copy Solid

copyeval

This command allows one to copy an "evaluated solid", that is a complete path ending in a solid. You will be asked to enter a complete path. Again, this path is entered on ONE line with the members seperated by spaces. If you do not know the complete path, use the "paths" command above to find it. Next, you will be asked to enter the name of this copied solid. The input path will be traversed and the accumulated path transformations will be applied to the parameters of the bottom solid. These new parameters will then be the copied solid. Note: this command is useful for making "dummy solids" to subtract for overlaps between objects which have been "object" edited.

9.74 Edit Objects Region Identifiers

edcodes obj1 obj2 ... objn

This command provides for an easy way to modify the code numbers (item, air, material, and los) of ALL regions found in paths beginning with an object. For each object, all paths beginning with that object are traversed until a region is encountered...at which time the following is listed:

item air mat los /obj1/.../region.n

The cursor then jumps back to "item" at the beginning of the line. At this time the cursor can be advanced only by entering a new item code (only digits are allowed) or by hitting the "space bar" or "tab key". The space and tab will move the cursor to the next position in the line. Pressing the "backspace key" will move the cursor to the beginning of the next location to the left. Moving the cursor "past" the los location will return it to the beginning of the line. When the code numbers are as you desire, a RETURN will print the next line for editing. At any time, pressing the "R" key will restore the idents on the current line to the way they were originally. A Control/C or a "q" will abort the process at the current line. String matching is allowed for the object names.

9.75 List Object As Stored

tab obj1 obj2 ... objn

This command will produce a listing of objects as they are stored in the MGED object data file. String matching is allowed for the objects.

9.76 List Regions With Given Ident

whichid item1 item2 ... itemn

This command will list ALL regions in the data file which have certain item codes.

9.77 Create ARB Given 3 Points

3ptarb

This command will produce a "plate mode" arb8 given 3 points on one face, 2 coordinates of the 4th point on that face, and a thickness. The 3rd coordinate of the 4th point will be solved for and the "other" face will be a normal distance (== the desired thickness) away. You will be asked for all necessary input.

9.78 Create ARB Given Point and Angles

rfarb

This command will produce a "plate mode" arb8 given a point on one face, rotation and fallback angles for that face, 2 coordinates of the 3 remaining points on that face, and a thickness. The 3rd coordinates of the three points will be solved for, and the other face will be a normal distance equal to "thickness" away. You will be asked for all necessary input.

9.79 Push Editing Down Paths

push obj1 obj2 ... objn

This command will "push" an object's path transformations to the solid's parameters located at the bottom of each path. If a conflict is encountered, then an error message is printed and NOTHING is done. A conflict occurs when the same solid "ends" different paths but the transformations are different. Conflicts could occur when "instanced" or "copied" groups occur in an object's paths or when a solid is a member of 2 regions which have been edited separately. A complication of this command, which will not be listed as a conflict, is when a solid's parameters have been changed by a push, yet this solid is referenced by another object which was NOT included in the push. The user should beware of this situation. This command can be very useful when "adding" parts from another file. Once the "object editing" of these new parts is completed, the push command will put the editing done to the solid level. Since the added parts should have no cross-references with the existing objects, there should be no problems.

9.80 Check For Duplicate Names

dup file.g string

This command will compare the current object data file with another MGED file "file.g" and list any object names common to both files. If "string" is present, all object names in "file.g" will be prefixed with "string" before comparing for duplicate names. Generally, one uses "string" only when duplicate names are found without it.

9.81 Concat Files

concat file.g string

This command will concatenate another MGED file "file.g" onto the present object data file. If "string" is present, all names in "file.g" will be prefixed with this string. No objects from "file.g" will be added if the name already occurs in the current object file...the object names will be listed and skipped. However, this command should be used in conjunction with the "dup" command to eliminate any problems with duplicate names.

9.82 Create Pseudo-Track

track

This command adds track components to the data file to fit specified "wheel" data. Solids, regions, and a group containing the regions will be created and displayed on the screen. You will be prompted for all required data.

9.83 Define ARB Face

facedef ####

This command is used to define the face plane of an ARB that is being edited. The following is the option menu displayed when this command is used: a planar equation b 3 points c rot and fb angles + fixed point d same plane thru fixed point q quit To select any of these methods of defining a plane, enter the appropriate letter. You will then be asked for the desired input.

9.84 Define Plane Equation of ARB Face

eqn [A B C]

This command is used when one is rotating the face of an edited ARB and defines the coefficients of the face planar equation (Ax + By + Cz).

9.85 Move (Rename) Everywhere

mvall old new

This command is used to rename all occurrences of an object in the data file. In this case, the object "old" will be renamed "new" for every occurence.

9.86 Miscellaneous Commands

It is possible to view, specify, and text-edit information pertaining to the material type and color of various parts of the model tree. This is an interim capability intended to provide enough material properties information for current rendering and analysis purposes until the design of a full material properties database can be finalized.

In addition to a variety of usual database manipulation and status commands, there are commands to compare the current database for name overlap (conflicts) with another database, as well as commands to import and export subtrees to/from the current database. If name conflicts between the two databases do exist, there are commands to rename an individual node without changing any of the references to it ("mv"), or to rename a node and change all the references to it ("mvall"). Another command which is useful for preparing to move subtrees between databases is the "push" command, which adjusts the transformation matrices from the indicated point down to the leaves of the directed acyclic graph, leaving the higher level arcs with identity matrices.

9.87 UNIX-Plot Output

The "plot" command can store an exact image of the current (non-faceplate) display on the screen, either using the System V standard 2-D monochrome UNIX-Plot (*plot(4)*) format, or the BRL 3-D color extended-UNIX-Plot format. These plots can be sent to a disk file, or "piped" directly to a filter process. This can be useful for making hard copies of the current MGED view for showing to others, using a local pen plotter or laser printer.

9.88 Ray-Tracing the Current View

An important capability even beyond the ability to generate an evaluated boundary wireframe is the ability of MGED to initiate a quick ray-trace rendering of the current view on any nearby framebuffer! This is implemented by using the MGED "rt" command to fork off an instance of the RT program, and sending the RT program a description of the current view with any user-specified options. This allows the designer to use the power of MGED to select the desired view, and then to quickly verify the geometry and light source placement. A 50 by 50 pixel rendering of the current view can usually be done in less than a minute (on a DEC VAX-780 class processor), and allows for general verification before the designer uses the "saveview" command to submit a batch job for a high resolution ray-trace of the same view.

9.89 Animation

The MGED editor includes a number of features which are useful for developing animation tools and scripts. The full description of the current viewing transformation and eye position can be saved in a file, and such a previously saved view can be read back at any time, immediately changing the editor's view. In addition, the current viewing transformation and eye position can be appended to a file containing a collection of keyframes. Most importantly, a file full of keyframe information, either raw keyframe positions or smoothly interpolated keyframe sequences, can by "played" in real time using MGED, providing a powerful and fast animation preview capability.

As a separate animation capability intended for developing demonstrations and instructional material relating to the use of the MGED editor, all user interactions with the editor can be recorded in a file, along with an indication of the time elapsed between user actions. This file can be adjusted using a normal text editor to remove any errors, or to eliminate dead time where the user stopped to think. Once created, this session script can be replayed through the editor at any time, either to provide a smooth "canned" demonstration before a live audience, or to create a film or videotape.

Key	Argument[s]	Description
e	obj1* obj2* ... objn*	display objects on the screen
E	obj1* obj2* ... objn*	display objects evaluating regions
B	obj1* obj2* ... objn*	Zap screen, display objects
d	obj1* obj2* ... objn*	delete objects from screen
cp	oldobj newobj	copy 'oldobj' to 'newobj'
cpi	oldtgc newtgc	copy 'oldtgc' to 'newtgc' inverted
Z	-none-	Zap (clear) the screen
g	groupname obj1* obj2*....objn*	group objects
r	region op1 sol1....opn soln	create/modify a region
i	object instname	create instance of an object
mv	oldname newname	rename object
mvall	oldname newname	rename all occurences of an object
l	object*	list object information
kill	obj1* obj2* ... objn*	remove objects from the file
killall	obj1* obj2* ... objn*	remove object[s] + references from file
killtree	obj1* objn* ... objn*	remove complete paths **CAREFUL**
t	object*	table of contents
mirror	oldobj newobj axis	mirror image of an object
mirface	#### axis	mirror face #### about an axis
extrude	#### distance	extrude an arb face
item	region item air	change region item/air codes
mater	region material los	change region mat/los codes
rm	comb mem1* mem2*....memn*	delete members from combination
units	mm—cm—m—in—ft	change the units of the objectfile
title	new-title	change the title of the description
p	dx [dy dz]	precise commands for solid editing
rotobj	xdeg ydeg zdeg	rotate(absolute) an edited object
scale	factor	scale(absolute) an edited object
translate	x y z	translate an edited object
arb	name rot fb	make arb8 with rot and fb
analyze	solids	print much info about a solid
summary	s—r—g	solid/region/group summary
tops	-none-	list all top level objects
find	obj1* obj2* ... objn*	find all references to an object
area	[endpoint-tolerance]	find presented area of E'd objects

Figure 9.1: MGED Command Summary 1

Key	Argument[s]	Description
plot	[-zclip] [-2d] [out-file] [— filter]	make UNIX-plot of view
color	low high r g b str	assign color(r g b) to item range
edcolor	-none-	text edit the color/item assignments
prcolor	-none-	print the current color/item assignments
make	name type	create and display a primitive
fix	-none-	restart the display after hangup
rt	[options]	raytrace view onto framebuffer
release	-none-	release current display processor
attach	nu—tek—tek4109—plot—mg—vg—rat	attach new display processor
ae	az elev	rotate view w/azim and elev angles
regdef	item [air los mat]	set default codes for next region created
ted	-none-	text edit a solids parameters
vrot	xdeg ydeg zdeg	rotate view
ill	name	illuminate object
sed	solidname	solid edit the named solid
center	x y z	set view center
press	button-label	emulate button press
knob	id value	emulate knob twist
size	value	set view size
x	-none-	debug list of objects displayed
status	-none-	print view status
refresh	-none-	send new control list
edcomb	comb flag item air mat los	edit comb record info
edgedir	delta_x delta_y delta_z	define direction of an ARB edge being moved
in	name type parameters	type-in a new solid directly
prefix	string obj1* obj2* ... objn*	prefix objects with 'string'
keep	file.g obj1* obj2* ... objn*	keep objects in 'file.g'
tree	obj1* obj2* ... objn*	list tree for objects
inside	–prompted for input–	find inside solid

Figure 9.2: MGED Command Summary 2

Key	Argument[s]	Description
solids	file obj1* obj2* ... objn*	make ascii solid parameter summary in 'file'
regions	file obj1* obj2* ... objn*	make ascii region summary in 'file'
idents	file obj1* obj2* ... objn*	make ascii region ident summary in 'file'
edcodes	obj1* obj2* ... objn*	edit region ident codes
dup	file prefix	checks for dup names in 'file' & current file
cat	file prefix	cat's 'file' onto end of current file
track	–prompted for input–	builds track given appropriate 'wheel' data
3ptarb	–prompted for input–	makes arb8 given 3 pts, etc.
rfarb	–prompted for input–	makes arb8 given point, rot, fallback, etc
whichid	ident1 ident2 ... identn	list all regions with given ident
paths	–prompted for input–	lists all paths matching input path
listeval	-prompted for input–	gives 'evaluated' path summary
copyeval	–prompted for input–	copy an 'evaluated' path-solid
tab	obj1* obj2* ... objn*	list objects as stored in data file
push	obj1* obj2* ... objn*	push object transformations to solids
facedef	#### data	define plane of an edited ARB face
eqn	A B C	define plane coefficients of rotating ARB face
q	-none-	quit
?	-none-	help message

Figure 9.3: MGED Command Summary 3

Chapter 10

TUTORIALS ON VIEWING AND STATES

Tutorials with illustrations are provided to give the MGED user a step-by-step walk-through of the basic capabilities of the graphics editor. Standard UNIX login and logout procedures appropriate to each site should be followed prior to beginning and after ending the tutorials.

Each of the tutorials will use the solids contained the MGED database called "prim.g". These can be obtained by making a copy of "db/prim.g" from the BRL CAD Package distribution tree. It is important to make a copy of the database and work with that, rather than using the supplied one. Changes made during the editing process are written to the database when they are *accepted*.

The first tutorial shows a sample invocation dialogue. All other tutorials start at the first MGED prompt (`mged>`). If the user wishes to continue from one tutorial to the next without leaving MGED, issue the *press reject* and *press reset* commands before starting a new tutorial. User input will be shown in an *emphasized* font, and MGED output will appear in a `typewriter` font. If the user input is shown on the same line as a prompt, the input is literal. If the user input is shown on a line by itself, it is a directive, and is entered in an appropriate fashion.

The tutorials are self-contained, and if the user wishes to proceed to the next tutorial without exiting MGED, the RESET button should be pressed to return to the top view, where the model XYZ axes map to the screen XYZ axes.

The standard recovery procedure when in the middle of an editing operation is to select REJECT edit. Control is returned to the viewing state, and the user can restart with the last edit (e) command used in the tutorial.

10.1 States Within the Edit Process

In this tutorial, the user will invoke MGED on a file called "prim.g"; attach a *display manager*; explore the various MGED states; and finally, exit MGED. A MGED database has a treelike structure. The leaves are the individual solids, and the other nodes are groupings of those solids. The solid editing functions are concerned with defining and modifying the leaves, and the object editing functions operate on groups, which are Boolean combinations of solids. One useful mental model is to envision solid editing as operating directly on a leaf and object editing as operating on the arc connecting a pair of nodes. The object edit will affect everything below the selected arc (this is why there is an additional state transition when object editing).

10.2 Viewing State

The first task is to invoke MGED. This tutorial will assume the user has a copy of the "prim.g" database in the current directory.

```
$ mged prim.g
BRL-CAD Release 3.0 Graphics Editor (MGED) Compilation 82
Thu Sep 22 08:08:39 EDT 1988
mike@video.brl.mil:/cad/.mged.4d2

attach (nu|tek|tek4109|ps|plot|sgi)[nu]?  sgi
ATTACHING sgi (SGI 4d)
Primitive Objects (units=mm)
mged>
```

The first three lines give information about which version of MGED is running, when it was compiled, and who compiled it. The next line is the display manager attach prompt. This prompt provides a list of available display managers, then shows what the default will be (selected if the user answers with a carriage return). In this case, the Silicon Graphics 4d display manager was selected, as is noted by the following line. Next the title of the database and the unit of measurement used in the database are printed, and finally, the first prompt is issued. At this point MGED has loaded "prim.g"; attached the SGI display; and is awaiting commands. Attaching a display also causes what is known as the MGED *faceplate* to be drawn on the graphics display.

The faceplate has several features of interest. In the upper left corner of the display, is a box which always shows the current MGED *state*. This can be one of six states: **VIEWING**, **SOL PICK**, **SOL EDIT**, **OBJ PICK**, **OBJ PATH**, or **OBJ EDIT**.

Immediately below, is the menu area. The only menu item initially shown is one labeled **BUTTON MENU**. This menu item toggles the display of the button menu entries when *selected* (more on selection later).

At the bottom of the display are two status lines. The first line contains information about the current view. The entry labeled **cent=** gives the *model space* coordinates of the dot in the center of the display. The entry labeled **sz=** reflects the current size in model units of the *viewing cube*. The viewing cube is a mathematical construct centered on the the dot in the center of the display. The **ang=** display shows the current rate of rotation in each of the three axes. The bottom line is used for several kinds of information. In the **VIEWING** state, it displays the title of the database.

The MGED viewing features are designed to allow the user to examine models at different angles. Preset views can be invoked at anytime by using either the menu or the button box. Selecting a preset view does not change the coordinates of the primitives, but instead changes the angle from which these primitives are displayed. Five standard views (top, right, front, 35/25, and 45/45) can be obtained by using either the bottom menu on the display screen or the control box. Three additional views (button, left, and rear) can be obtained by using the button box, but not by using the menu.

The normal or default viewing state is the “top” orientation, with model +X pointing towards the right of the screen, model +Y pointing towards the top of the screen, and model +Z pointing out of the screen. In the “top” view, the model and screen axes are the same. The “reset” button and “Reset Viewsize” menu items also result in a “top” view.

The following table shows the angles of rotation to obtain the other views.

View	Angle of Rotation (from top)
Top	0, 0, 0
Bottom	180, 0, 0
Right	270, 0, 0
Left	270, 0, 180
Front	270, 0, 270
Rear	270, 0, 90
35, 25	295, 0, 235

```
mged> e arb8
vectorized in 0 sec
mged> size 12
mged>
```

The **e** command causes the named object(s) – a solid named “arb8” in this case – to be displayed, and the **size** command sets the size of the viewing cube. Figure 10.1 shows what the display currently looks like. In this view, the X-axis is to the right, the Y-axis points up, and the Z-axis is perpendicular to (poking out of) the screen.

Twist the **Y ROT** *knob clockwise and back.*
Twist the **X ROT** *knob counterclockwise and back.*

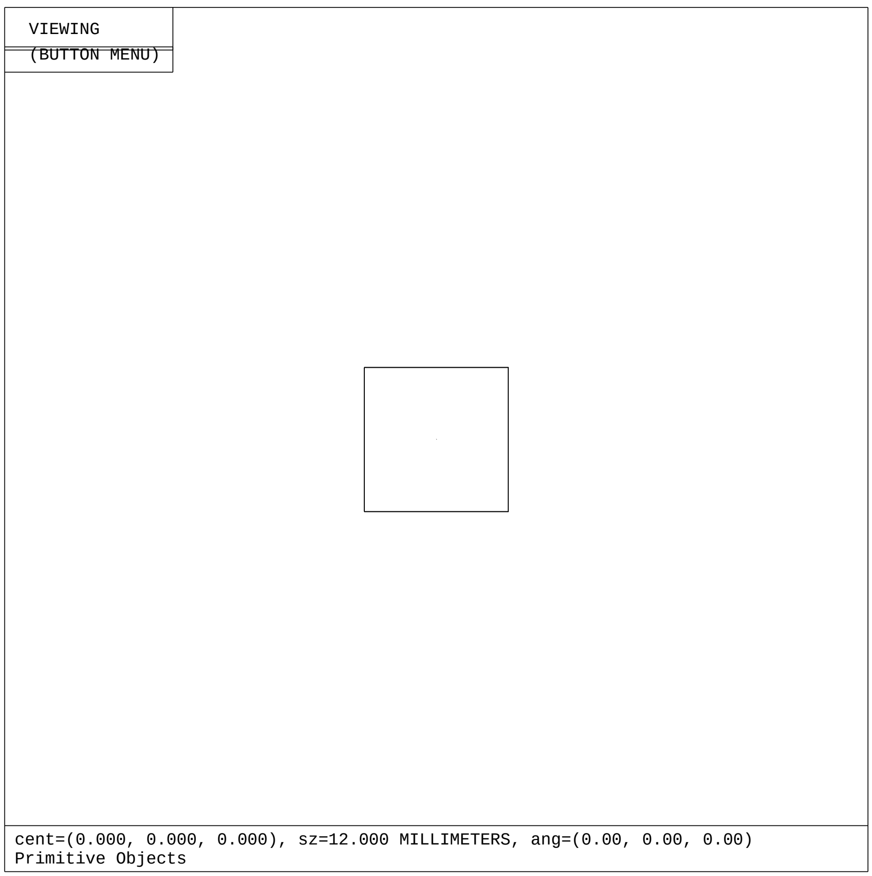

Figure 10.1: “arb8” Top View

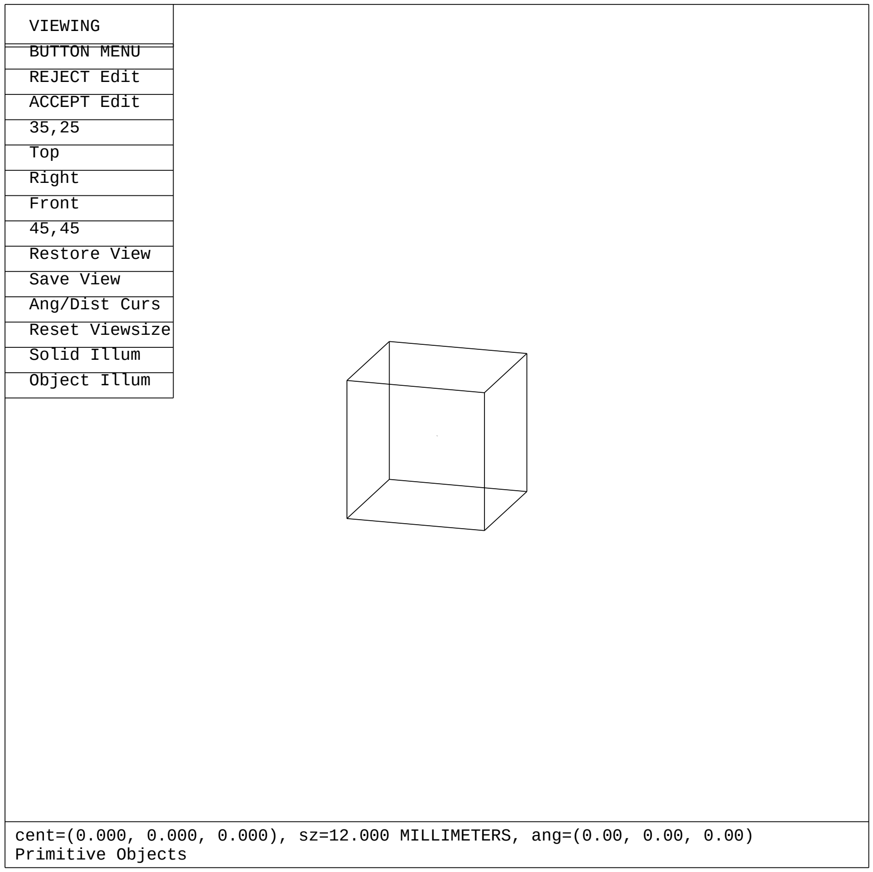

Figure 10.2: “arb8” Rotated View

These knobs, along with the **Z ROT** knob, rotate the viewing cube. Use of the rotation knobs allows the user to view the model from any orientation. Turning a knob clockwise causes a rotation in the positive direction, while turning a knob counterclockwise causes a negative rotation (right-hand rule). The knobs are rate based, not position based; once a rotation has been started, it will continue until the knob is returned to zero (or the **zeroknobs** button is pressed). Rotations are about the viewing cube (screen) axes, not the model axes. Systems without knobs can use the **knob** command.
Move the mouse (or pen) until the cursor is in the **BUTTON MENU** *block and then press the middle mouse button (depress the pen).*

Pressing the middle mouse button (or the pen) *selects* something. When

the cursor is inside the menu area, a selection causes the event described by the menu item to occur. Selecting **BUTTON MENU** causes the button menu to appear on the left side of the screen. The **BUTTON MENU** menu item is a toggle; subsequent selection of this item will cause the button menu to disappear. Figure 10.2 shows the new display.
Move the cursor from the menu area to a point near the upper left corner of the solid and select it (press the center mouse button).

In the **VIEWING** state, making a selection while outside of the menu area will move the selected point to the center of the display. Look carefully at the center of the display; the point just selected is now located at the center dot. Use the **center** command to reset any translations made with the mouse.
`mged>` *center 0 0 0*
`mged>`

From the **VIEWING** state, the user will normally transition to either the **SOL PICK** or **OBJ PICK** state. The **SOL PICK** state is selected by:

- Selecting the **Solid Illum** button menu entry, or,
- Pressing the **sill** button (this button may be labeled using some variation of "Solid Illum"), or,
- Typing **press sill**.

Similar entries (**Object Illum**) and (**oill**) exist for transitioning into the **OBJ PICK** state. In general, the **press** command is the basic mechanism (type **press help** for a list of available commands). Most of the press commands have been mapped onto a button box if it is available, and some of the most common are also mapped into the **BUTTON MENU** so they can accessed without letting go of the mouse.

10.3 Solid Pick State

Place MGED in the **SOL PICK** *state using one of the above mechanisms.*

Upon entering the **SOL PICK** state, the display will look similar to Figure 10.3. The **SOL PICK** state used to select which of the displayed solids is to be edited. Note that the color of the solid has changed from red to white. The screen is divided into as many horizontal zones as there are solids displayed, and each zone is assigned to one solid. As the mouse is moved vertically through each zone, the corresponding solid is highlighted ("illuminated") by drawing it in white. In this instance, there is only one solid being displayed, so this state is relatively uninteresting. If the system being used has no mouse, there is no reason to enter the **SOL PICK** state. The user will instead transition directly to the **SOL EDIT** state using the **sed** command.

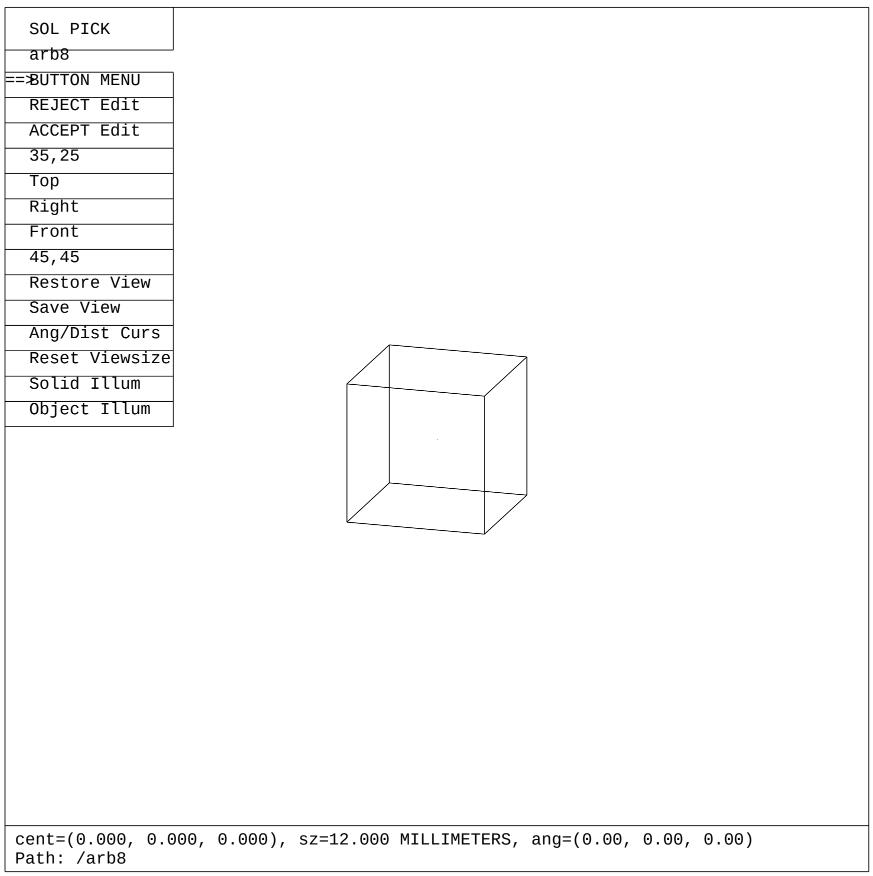

Figure 10.3: MGED In Solid Pick State

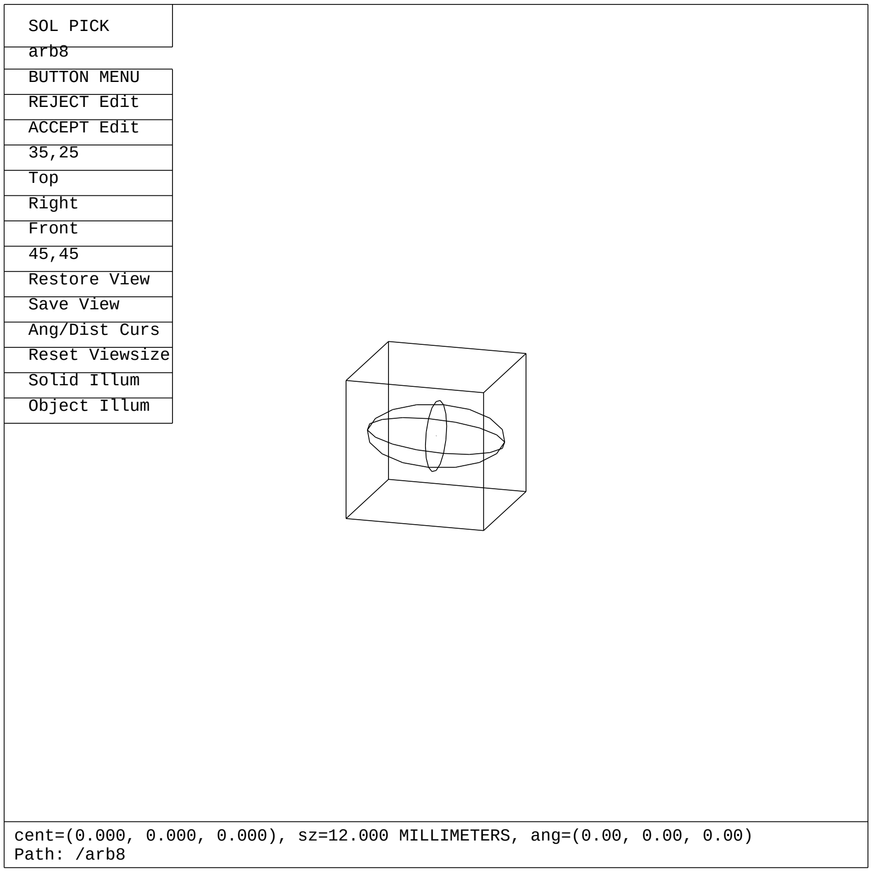

Figure 10.4: MGED In Solid Pick with Two Solids

`mged>` *press reject*
`mged>` *e ellg*
`mged>`
Press the **sill** *button*

Note that the first action taken was to *reject* the edit. Any time MGED is not in the **VIEWING** state, a *reject* command (via **press**, button, or mouse) discards all editing changes accumulated since the last transition out of the **VIEWING** state, and places MGED in the **VIEWING** state. The display should now look similar to Figure 10.4. Notice that one solid is white and the name of that solid is displayed in the upper left corner of the display, as well as in the bottom status line. The solid to be edited is selected by moving the mouse up and down until the zone corresponding to the desired solid is reached.

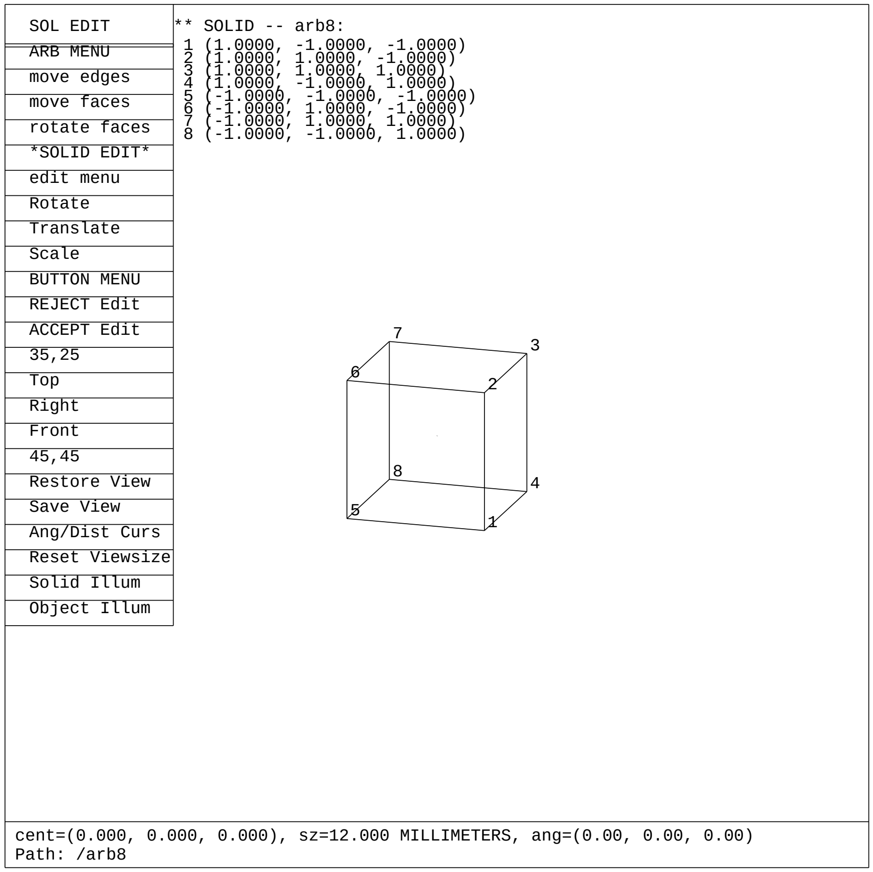

Figure 10.5: Solid Edit State

Once the appropriate zone is reached, select it. This selects a solid, and once a solid is selected, MGED enters the **SOL EDIT** state.

10.4 Solid Edit State

`mged>` *d ellg*
`mged>`
Select the solid called "arb8".

The **d** commmand removes something from the display. In this case, the solid "ellg" was removed to reduce clutter. The display should now look like Figure 10.5. When MGED enters the solid edit state, the following occurs:

- The solid selected for editing remains illuminated,
- The solid is labeled,
- The coordinates (or dimensions) associated with the labels, and other information is displayed to the right of the menu area,.
- If the solid is a member of one or more groups, a similar set of coordinates called the *PATH* is displayed immediately below the the first set of coordinates,
- The ***SOLID EDIT*** menu is displayed, and,
- A solid specific edit menu (in this case the **ARB MENU**) is displayed.

The ***SOLID EDIT*** menu provides access to generic operations (translation, rotation and scaling) common to all solids. The solid specific edit menu is a list of solid type specific editing operations. Selecting one of the solid specific edit menus causes a submenu with solid type specific choices to be displayed. To remove this submenu, select either the **RETURN** item in the submenu, or the **edit menu** item in the ***SOLID EDIT*** menu.

It is in this state that the solid is altered to meet the modeler's requirements. The shape, positioning, and orientation of the solid is changed using numeric keyboard input, positioning of the mouse, or by use of the knobs. Once the solid has been altered, the edit is either accepted or rejected. Accepting the edit causes all changes made to be written to the database; rejecting the edit "throws them away". Either operation will terminate the edit session and return MGED to the **VIEWING** state.
Reject the edit.

10.5 Object Pick State

Place MGED in the **OBJ PICK** *state.*

Figure 10.6 shows what the display looks like when in the **OBJ PICK** state. As with the **SOL PICK** state, a single solid is selected. This solid becomes the reference solid for the object edit. In the **OBJ PICK** state, the solid will be shown as a member of one or more objects. Less obvious is the fact that the local axes associated with the selected solid are the axes used for the entire object during the object edit.

10.6 Object Path State

Select "arb8".

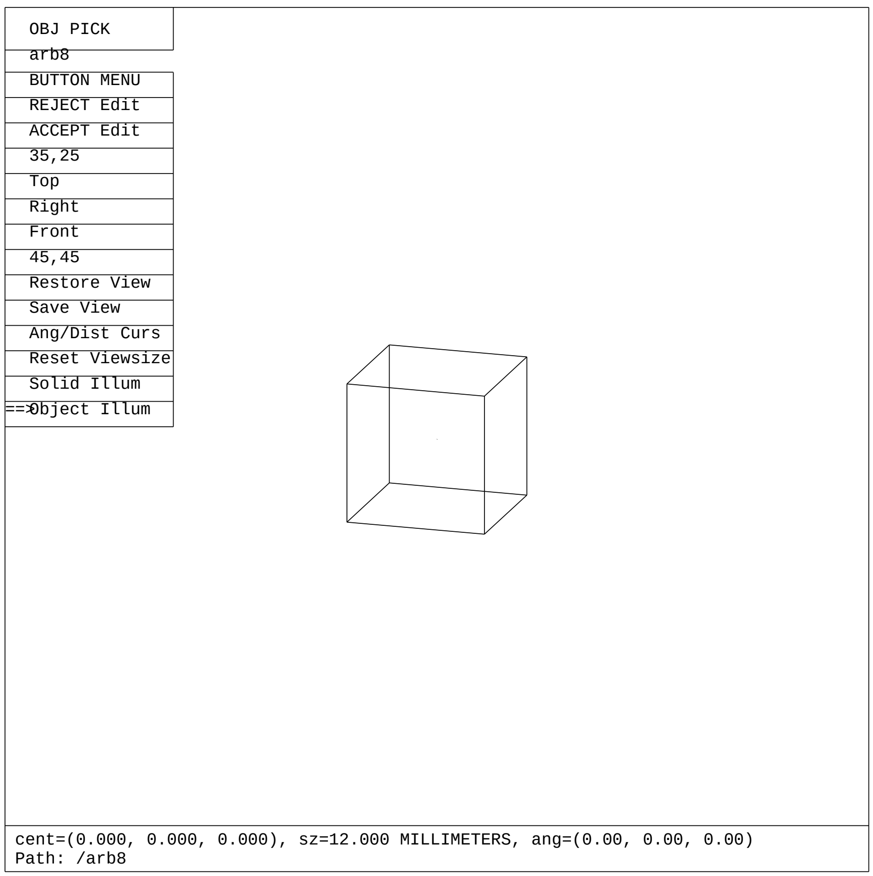

Figure 10.6: Object Pick State

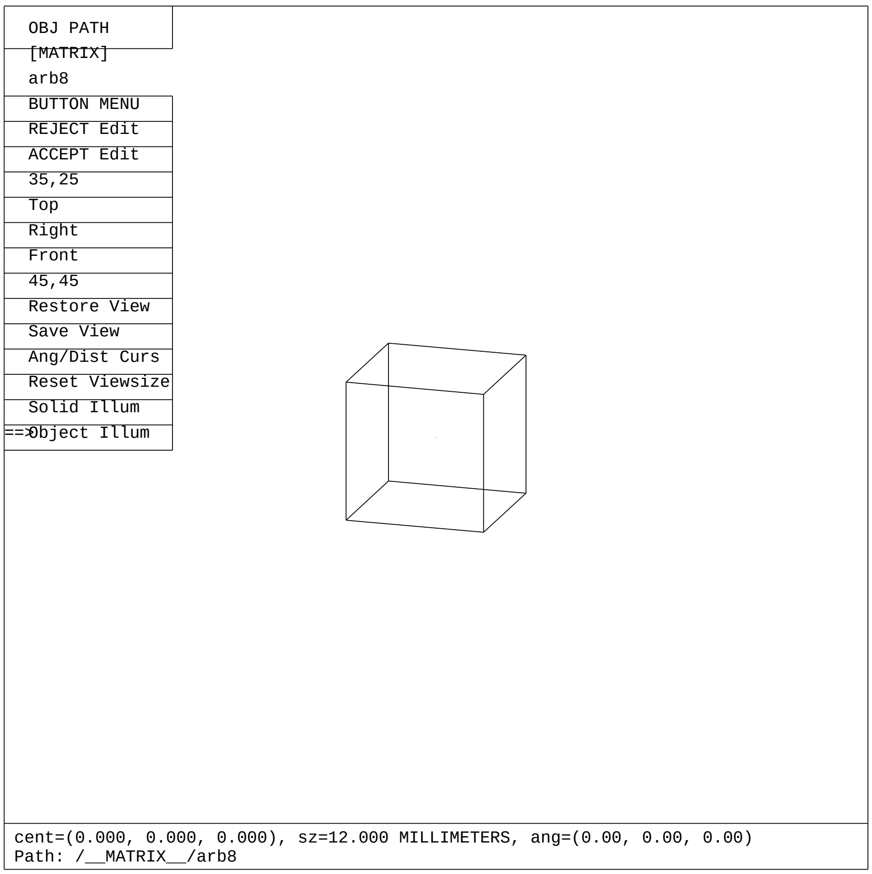

Figure 10.7: Object Path Selection State

MGED transitions into the **OBJ PATH** state once a solid has been picked from **OBJ PICK**. Figure 10.7 is the display in the **OBJ PATH** state. When in this state the extent of the editing operation is set. Everything below the editing point is affected by the edit. The editing point is shown by the *MATRIX* label in the display. It is shown as **[MATRIX]** in the upper left part of the display and as __**MATRIX**__ in the second status line. The editing point is chosen with the same mechanism used by **SOL PICK** and **OBJ PICK**. This time, there is one horizontal zone for each node in the path between the root and selected leaf. Moving the mouse up and down moves the editing point up and down in the tree. Once again, having a simple database and only one object in view makes for a relatively uninteresting situation.

10.7 Object Edit State

Select the editing point above "arb8".

MGED is now in the **OBJ EDIT** state and the display should look like Figure 10.8. When MGED enters the object edit state, the following occurs:

- The reference solid remains illuminated,
- The reference solid is labeled,
- The information associated with the labels is displayed to the right of the menu area, and
- The ***OBJ EDIT*** menu is displayed.

The **OBJ EDIT** state is used to modify the Homogeneous Transform Matrix selected during the **OBJ PATH** state. Permissible operations include uniform and affine scaling of the objects, as well as translation and rotation. As with the **SOL EDIT** state, MGED accepts changes entered using the keyboard, mouse or knobs.

This concludes the first tutorial. Examples of the appearance of MGED in each of the six states have been given, along with some idea of what each of the states is used for. All that remains is to reject the current edit, and exit MGED. Strictly speaking the **q** command could be entered directly, but doing so, can become a dangerous habit.

Select **REJECT Edit** *using the mouse.*
Press the **reject** *button.*
`mged>` *d arb8*
`mged>` *q*
`$`

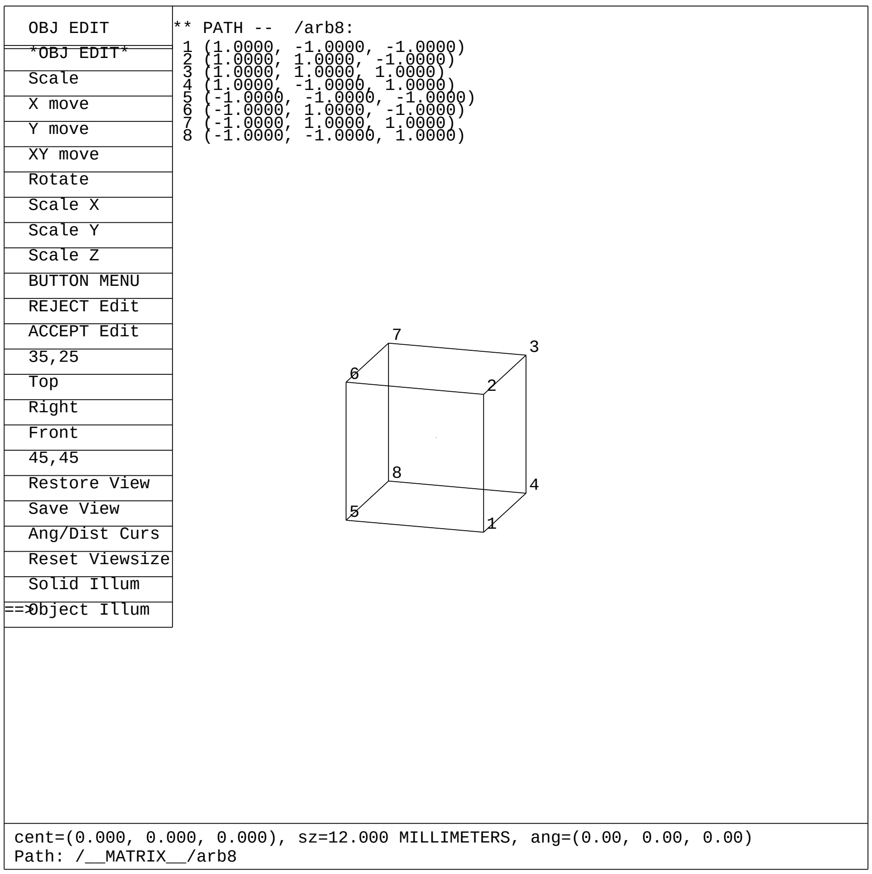

Figure 10.8: Object Edit State

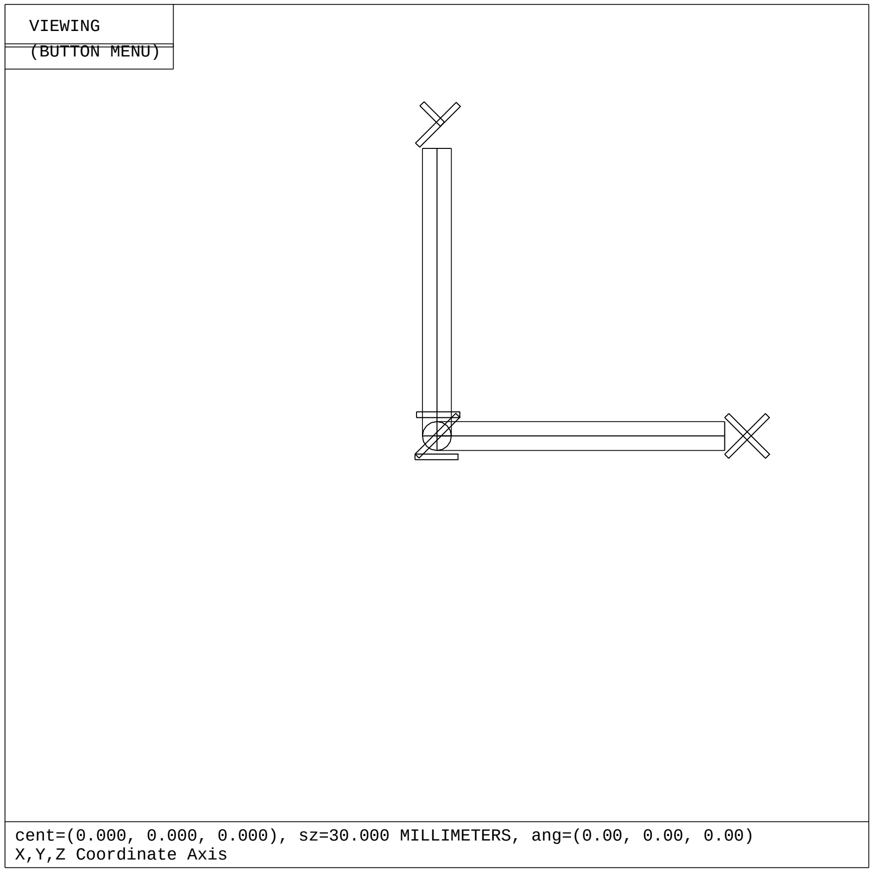

Figure 10.9: A Top View of the Coordinate Axes

10.8 Editing in the Plane of the Screen

When MGED is in a "translate" mode within an edit state, the plane of the mouse or data tablet is mapped to the plane of the screen, to permit moving objects in a controlled way in two of the three available dimensions. The orientation of the plane of the screen is determined by the currently selected view. In most circumstances, users will find that repositioning objects is easiest when the the plane of the screen is oriented in an axis-aligned view. This is most easily accomplished by utilizing one of the preset views. For this exercise, obtain a copy of the *axis.g* database, and run MGED, eg:

```
$ cp cad/db/axis.g .
$ mged axis.g
BRL-CAD Release 3.0 Graphics Editor (MGED) Compilation 82
```

```
Thu Sep 22 08:08:39 EDT 1988
mikel@video.br:/cad/.mged.4d2

attach (nu|tek|tek4109|ps|plot|sgi)[nu]?  sgi
ATTACHING sgi (SGI 4d)
X,Y,Z Coordinate Axis (units=none)
mged> e axis
vectorized in 0 sec
```
Select "Top" in the Button menu
`mged>`

10.8.1 Top View

The top view is the default view. The orientation of the axes is shown in Figure 10.9. The surface of the viewing screen and the graphics tablet is the XY plane. Edit changes using the graphics tablet will affect only the X and Y coordinates of the primitive.

`mged>` *sed x*
Select "Translate" in the Solid Edit menu
`mged>`

Select different points on the tablet with the mouse, each time pressing the middle mouse button. Notice how the X and Y coordinates of the V vector change, but the Z coordinate does not. An example of this is shown in Figure 10.10; compare the values of V with those in Figure 10.9.

Select "REJECT Edit" in the Button menu

10.8.2 Bottom View

`mged>` *press bottom*
`mged>` *sed x*
Select "Translate" in the Solid Edit menu
`mged>`

The *press bottom* command selects the bottom view of the model, and the new configuration of the axes can be seen in Figure 10.11. The surface of the viewing screen and the mouse or tablet are still in the XY plane. Edit changes using the graphics tablet will affect only the X and Y components of the solid. Select different points on the tablet with the mouse and notice the changes in the coordinates; compare the values of V with those in Figure 10.12.

Select "REJECT Edit" in the Button menu

10.8.3 Right View

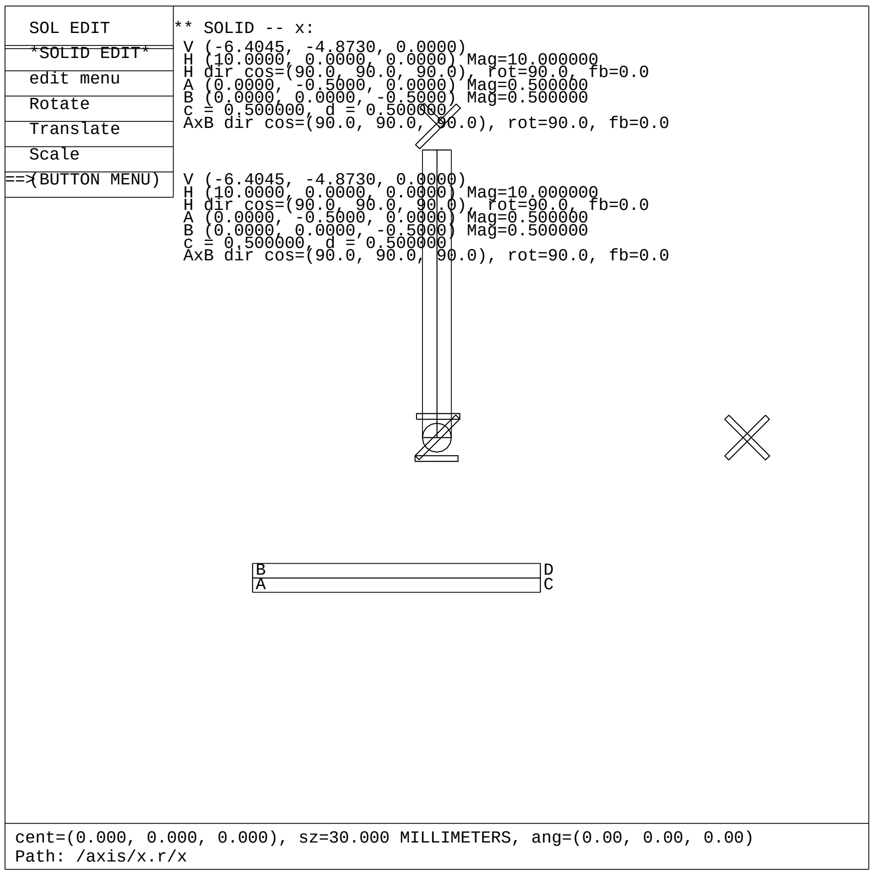

Figure 10.10: Translating from the Top View

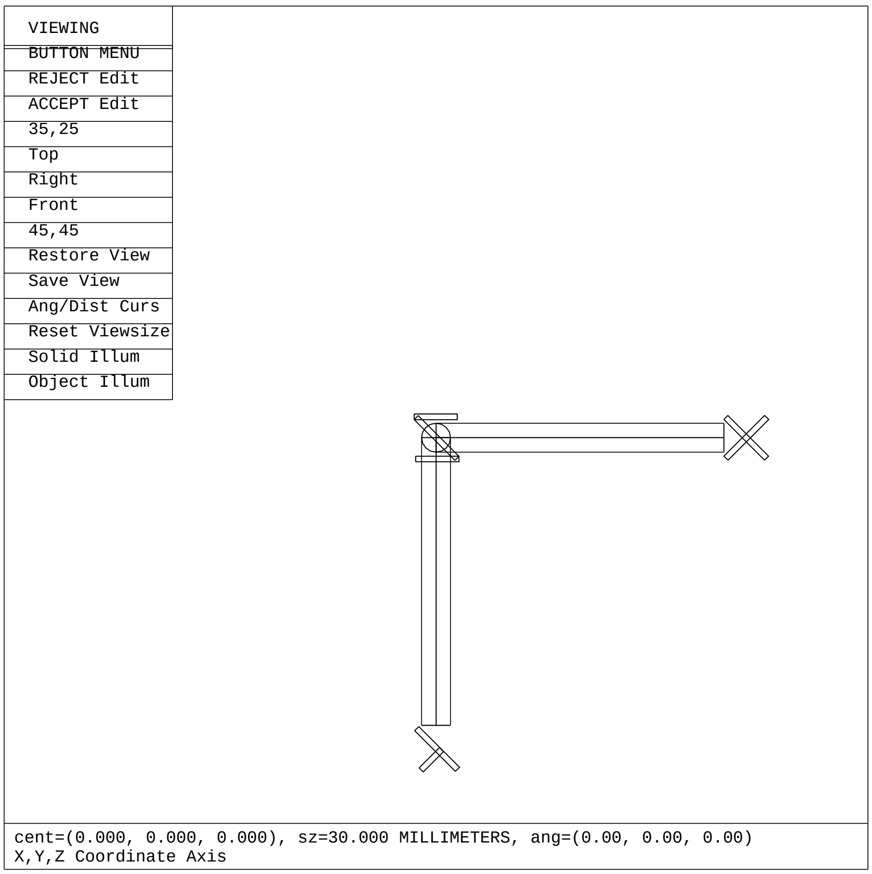

Figure 10.11: A Bottom View of the Coordinate Axes

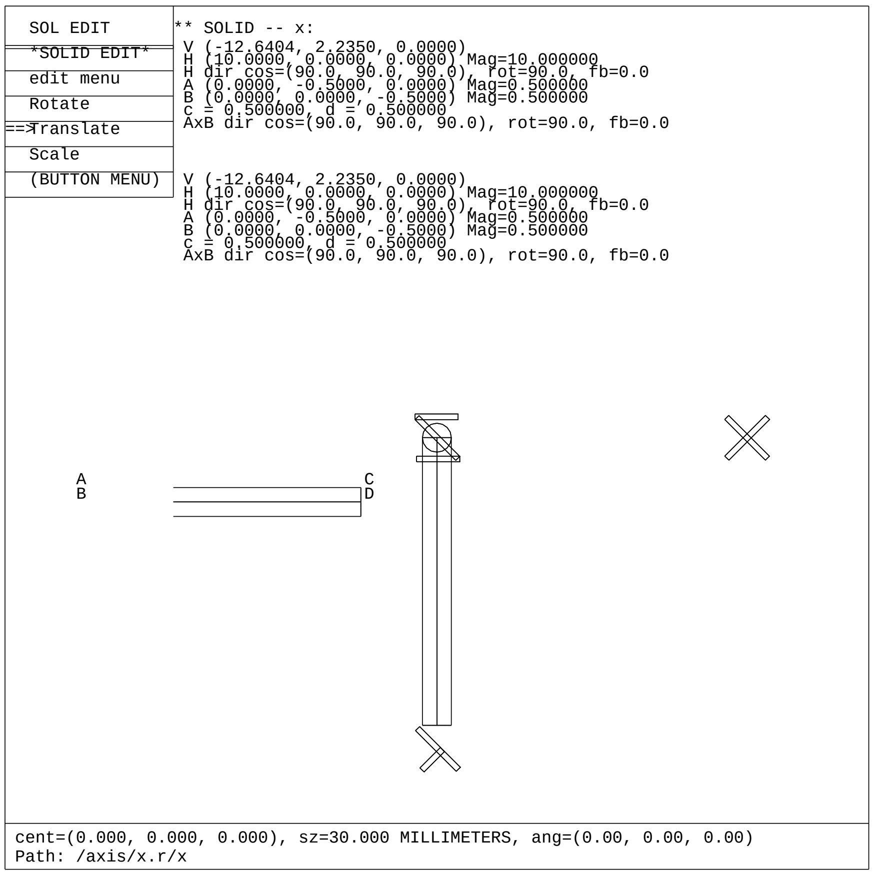

Figure 10.12: Translating from the Bottom View

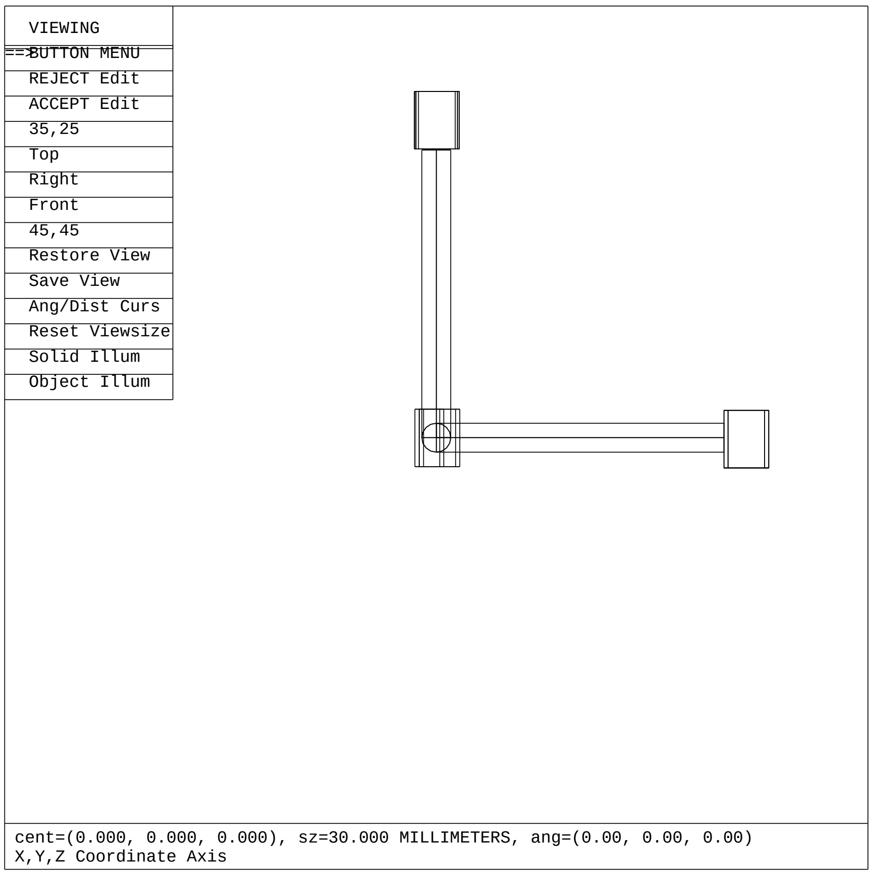

Figure 10.13: A Right View of the Coordinate Axes

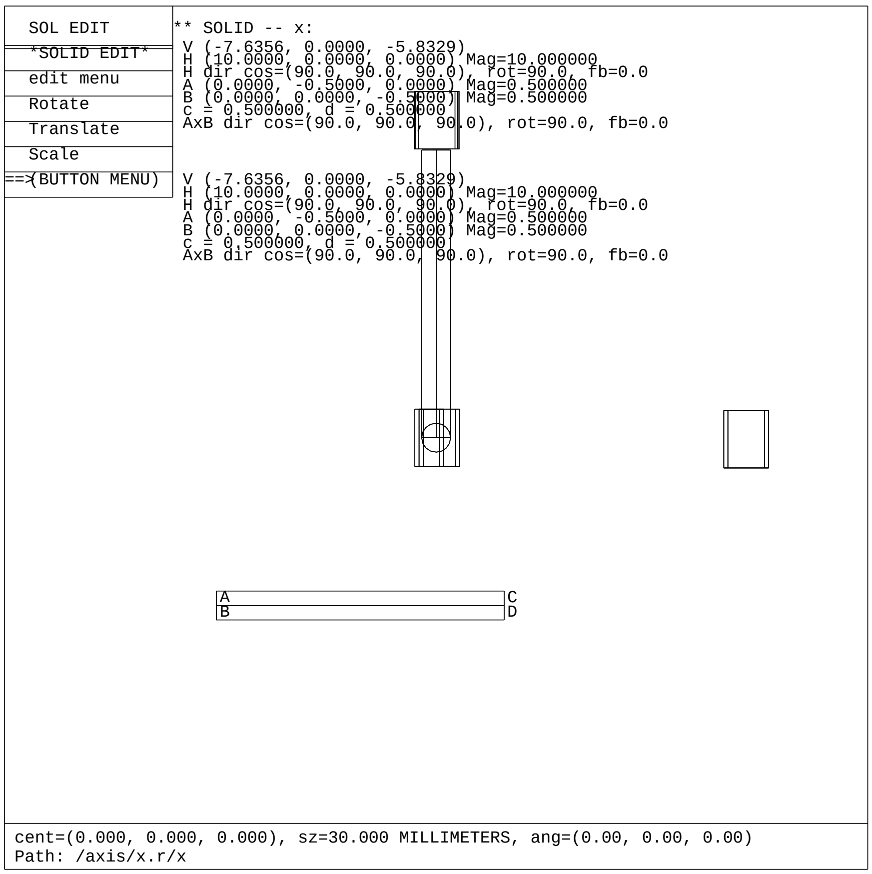

Figure 10.14: Translating from the Right View

Select "Right" in the Button menu
`mged>` *sed x*
Select "Translate" in the Solid Edit menu
`mged>`

The right hand view has been selected. Model +X still proceeds to the right, but now model +Z is at the top of the screen, and model +Y is pointing out of the screen. This new configuration is depicted in Figure 10.13. The surface of the viewing screen and the graphics tablet is the XZ plane. Edit changes using the graphics tablet will affect only the X and Z coordinates of the solid. Select different points on the tablet with the mouse and notice the changes in the V coordinates; only the X and Z components change, as in Figure 10.14.

Select "REJECT Edit" in the Button menu

10.8.4 Front View

Select "Right" in the Button menu
`mged>` *sed x*
Select "Translate" in the Solid Edit menu
`mged>`

The front view has been selected. Model +X points out of the screen, model +Y points to the right, and model +Z points towards the top of the screen, as shown in Figure 10.15, which has been slightly rotated off the preset view to improve the legibility of the axis labels. The surface of the viewing screen and the graphics tablet is the YZ plane. Edit changes will affect only the Y and Z coordinates of the primitive, as shown in Figure 10.16. Select different points on the tablet with the mouse and notice the changes in the coordinates.

Select "REJECT Edit" in the Button menu

10.8.5 35, 25 View

Select "35,25" in the Button menu
`mged>` *sed x*
Select "Translate" in the Solid Edit menu
`mged>`

Figure 10.17 is the 35,25 view of the axes model. The axes are no longer parallel or perpendicular to the viewing surface or to the graphics tablet. Edit changes using the graphics tablet will affect all of the coordinates of the solid, in a manner that is visually intuitive when the solid is moved around on the screen. Select different points on the tablet with the mouse and notice the changes in the coordinates, such as in Figure 10.18. Note how all three components of the V vector have changed.

Select "REJECT Edit" in the Button menu

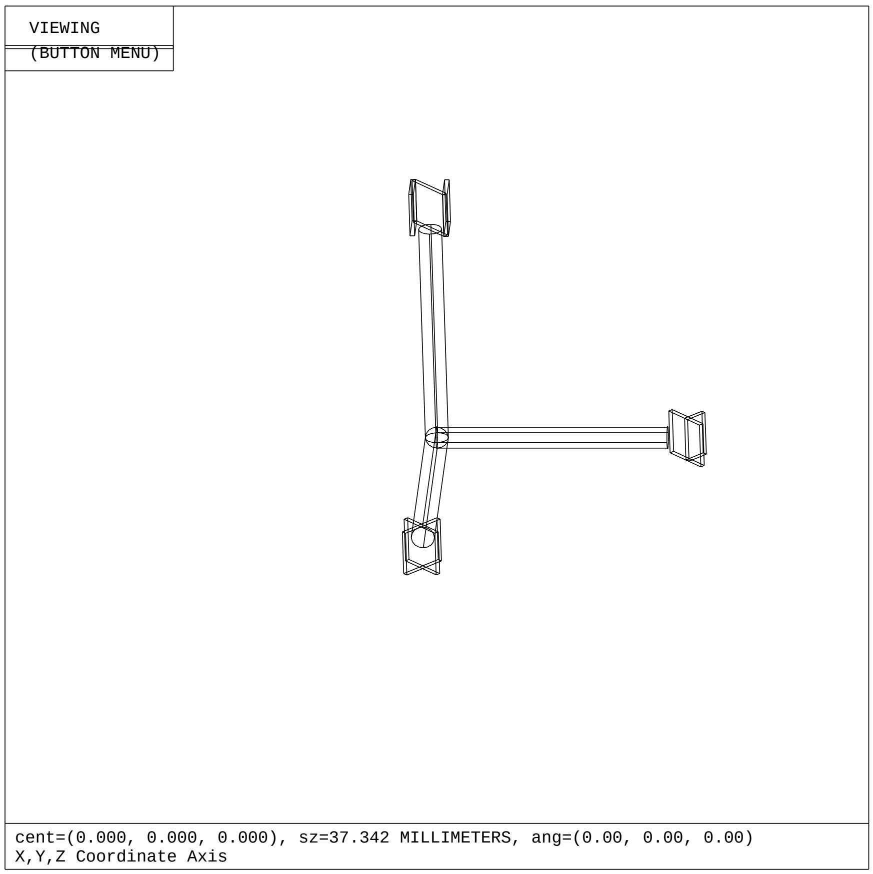

Figure 10.15: A Front View of the Coordinate Axes

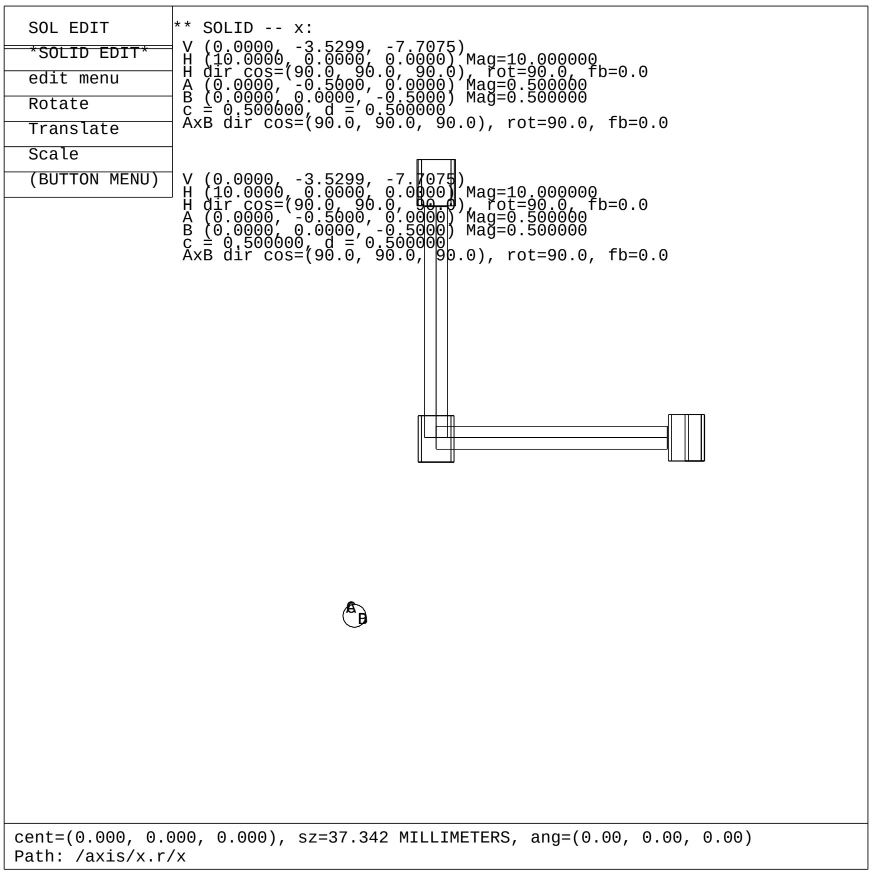

Figure 10.16: Translating from the Front View

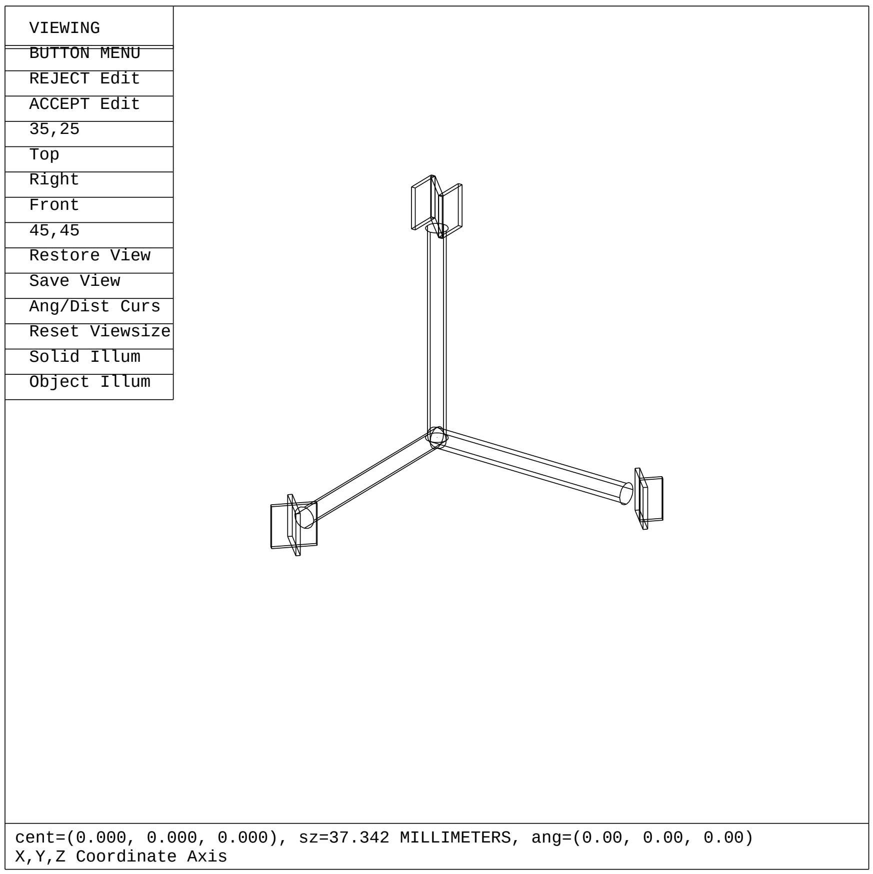

Figure 10.17: An Oblique 35,25 View of the Coordinate Axes

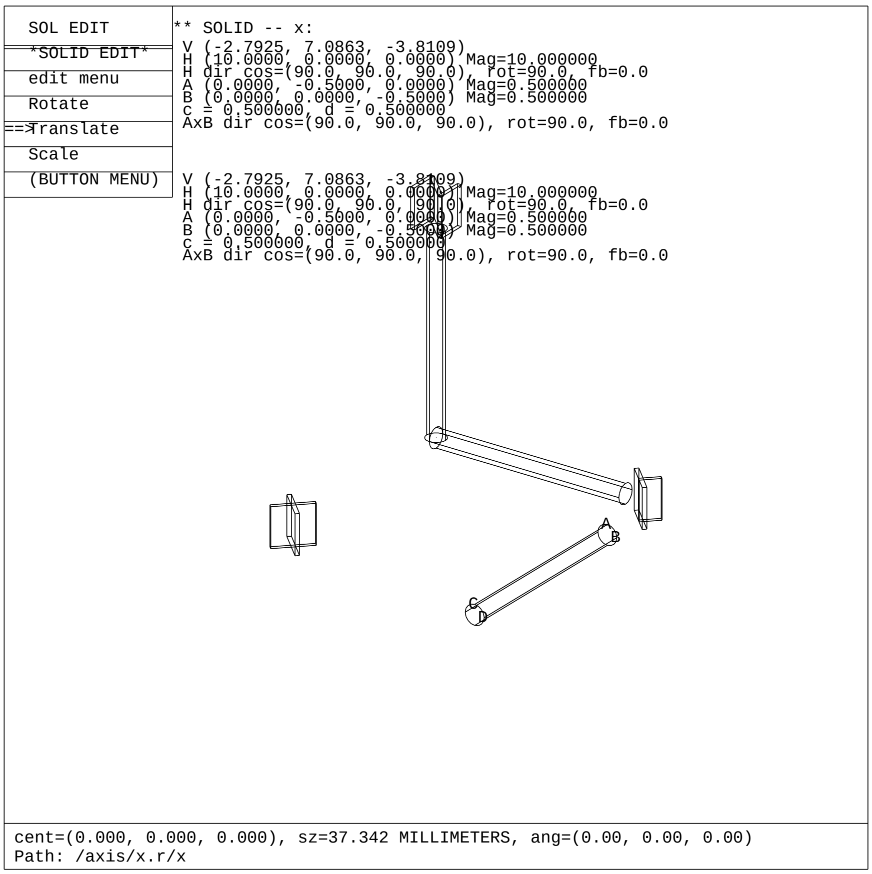

Figure 10.18: Translating in the 35,25 View

Chapter 11

TUTORIALS ON EDITING SOLIDS

The Solid Editing state of MGED is used to modify the fundamental parameters of an individual solid. Each solid must be modified individually.

11.1 Solid Edit: A Six-Sided Polyhedron

This section illustrates the use of commands while in SOL EDIT state to alter the shape of a polyhedron with six sides and 8 faces (ARB8).

`$` *mged es.g*

```
BRL-CAD Release 3.0 Graphics Editor (MGED) Compilation 82
Thu Sep 22 08:08:39 EDT 1988
mike@video.brl:/cad/.mged.4d2

es.g:  No such file or directory
Create new database (y|n)[n]?  y
attach (nu|tek|tek4109|ps|plot|sgi)[nu]?  sgi
ATTACHING sgi (SGI 4d)
Untitled MGED Database (units=mm)
mged> in arb8 rpp -1 1 -1 1 -1 1
mged> size 10
mged>
```

Figure 11.1 is a top view of the six-sided polyhedron. The Z-axis perpendicular to the viewing screen. Next, the view is rotated so that all sides can be seen.

```
mged> Twist ROTY knob clockwise and restore
mged> Twist ROTX knob counter-clockwise and restore
mged>
```

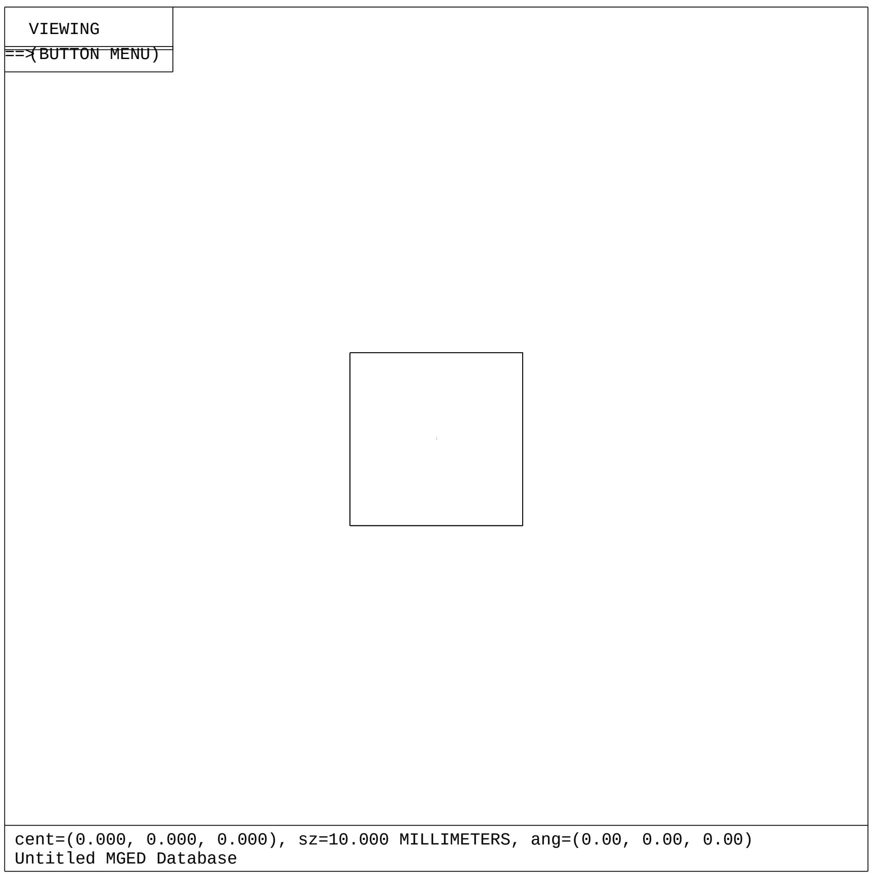

Figure 11.1: Top View of ARB8

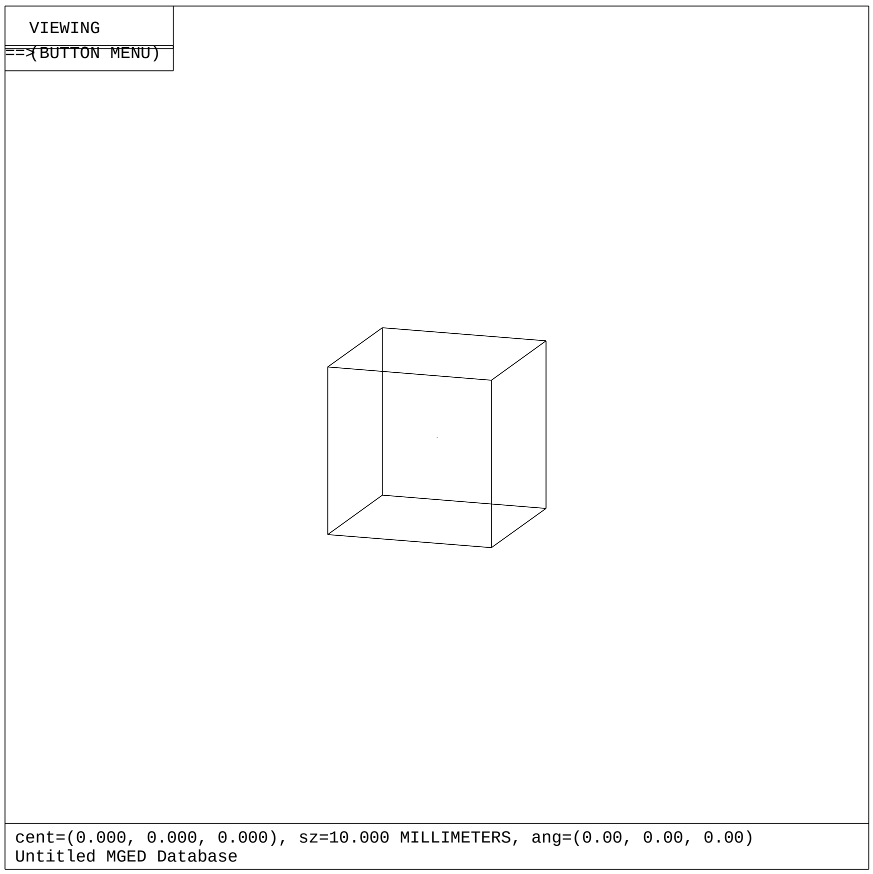

Figure 11.2: A Rotated View of the ARB8

Figure 11.2 shows a better perspective of the solid.

The next step in this tutorial is to transfer to the solid edit state. This can be accomplished in two ways: either by going through the SOL PICK state ("illuminate mode") or by direct transfer via keyboard command. Using illuminate mode is better when the name of the solid to be edited may not be known, while the keyboard command is generally preferred when the name of the solid is known.

`mged>` *Select the "Solid Illum" entry in the button menu*
`mged>` *Move the mouse out of the menu area*
`mged>` *Click the mouse to enter SOL EDIT state*
`mged>`

To perform a direct transfer from the viewing state to the solid edit state using a keyboard command, enter:

`mged>` *sed arb8*
`mged>`

Figure 11.3 corresponds to the view on the display. The ARB8 MENU is unique to the ARB primitive, and lists operations that can only be performed on an ARB solid. The items in the ARB8 MENU are selected by using the mouse. Each of the other types of solids have a similar unique menu. When one of these items is selected, the top level ARB8 MENU disappears, to be replaced with the indicated subordinate menu. The top-level menu reappears when either the "edit menu" item in the SOLID EDIT menu is selected, or the "RETURN" item in the subordinate menu is selected.

The SOLID EDIT menu applies to all solids when in the SOL EDIT state. The items in the SOLID EDIT menu are selected by either using the mouse or by depressing the appropriate button on the button box. When any of the SOLID EDIT menu items are selected (eg, "Rotate", "Translate", "Scale"), the solid-specific menu disappears. Th top-level solid-specific menu reappears when the "edit menu" item in the SOLID EDIT menu is selected.

The *p [params]* command is used to make precise changes, where the numeric value of the parameter being edited is know. Values for all parameters in the ARB8 and SOLID EDIT menus can be specified by using the *p* command, or by pointing and clicking with the mouse.

11.1.1 Translate Operation

`mged>` *Select the "Translate" entry in the solid edit menu*
`mged>` *p 0 0 0*
`mged>`

Point 1 of the primitive is moved to point 0 0 0, as shown in Figure 11.4.

The translate solid operation is selected by either picking "Translate" on the solid edit menu with the mouse, by depressing the solid edit button on the button box, or by entering the *press sed* command. Parameters to the translate solid operation are of the form *p a b c* where *a*, *b*, and *c* are the new coordinates

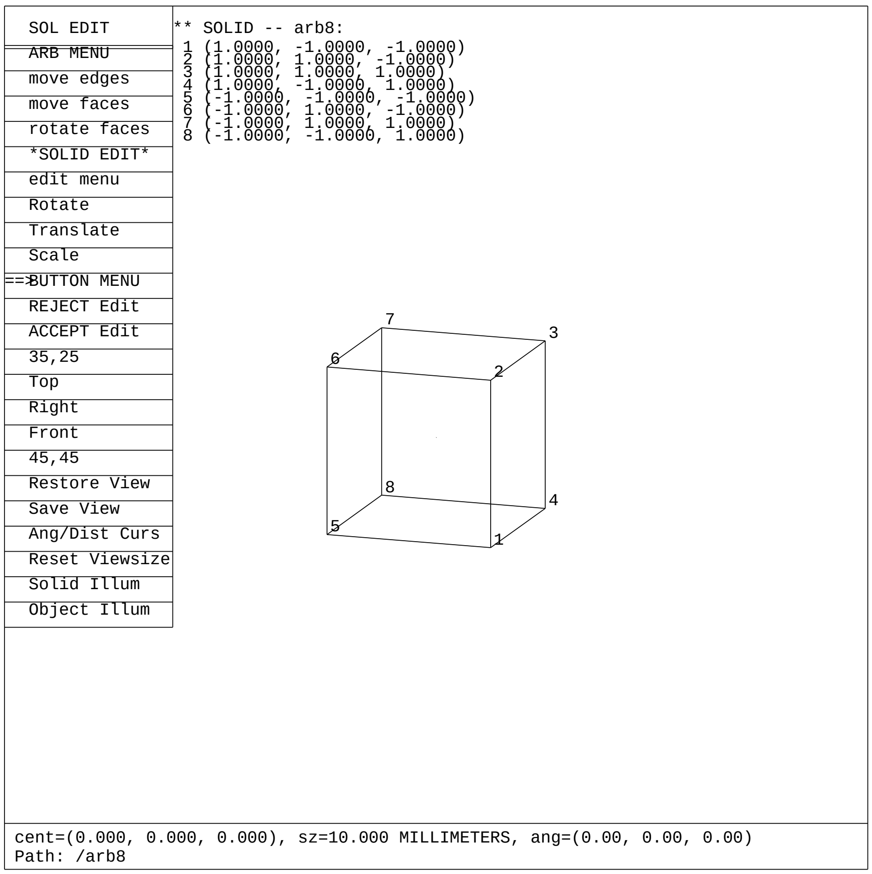

Figure 11.3: An ARB8 in Solid Edit State

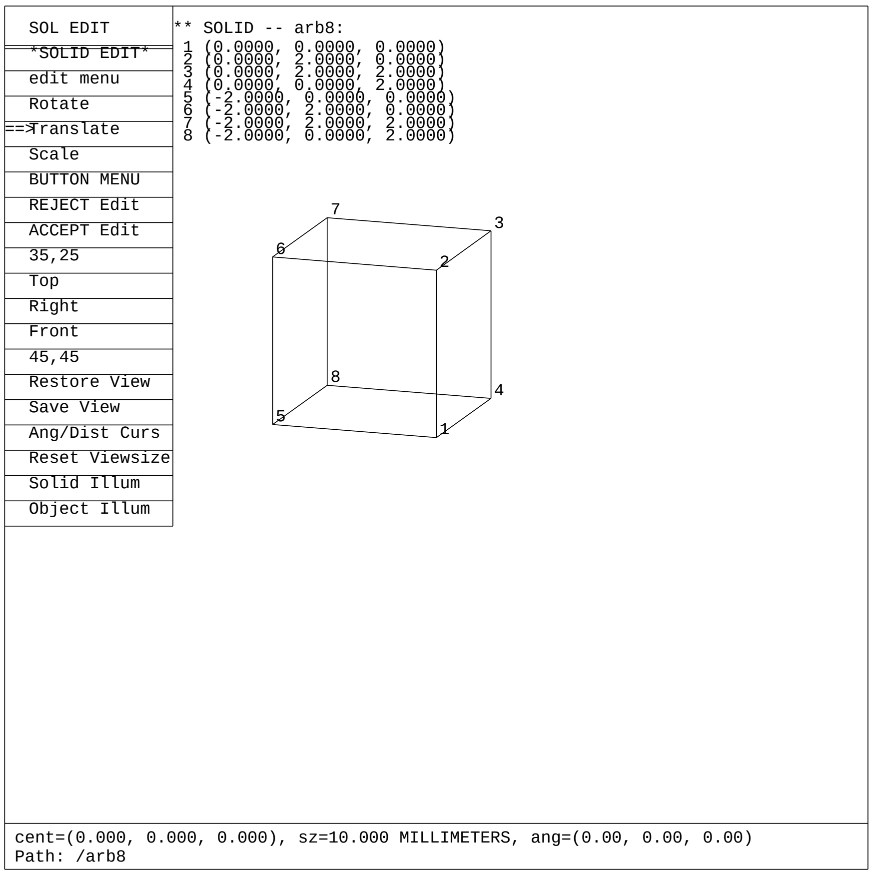

Figure 11.4: Translating ARB8 Point 1 to the Origin

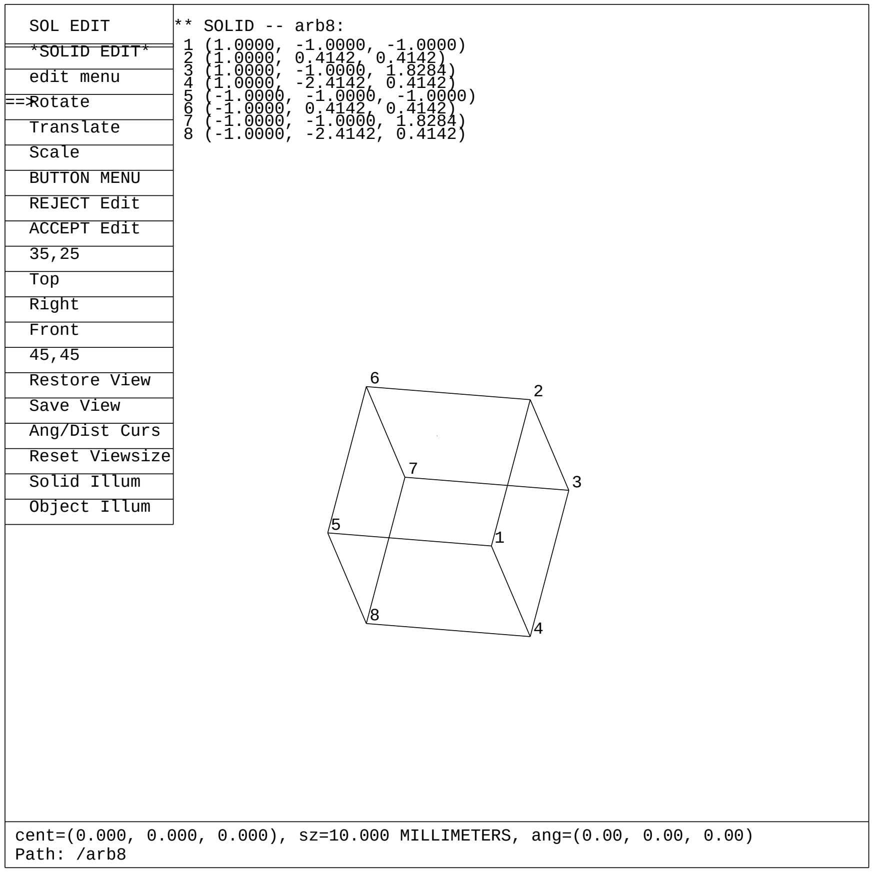

Figure 11.5: Solid Edit Rotation of 45 Degrees about X

of point 1 in the solid. The other points are transferred to keep the same position relative to point 1. The general form of the new coordinates for point is

`x ’ = x + a − x`

`y ’ = y + b − y`

`Z ’ = Z + c − Z`

The command

`mged>` *p 1 -1 -1*

`mged>`

can be used to restore the primitive to the original position.

11.1.2 Rotate Operation

The rotate operation is initiated by either selecting Rotate on the menu screen with the mouse, by depressing the Solid Rotate button on the button box, or by entering the *press srot* command on the keyboard.

`mged>` *Select the "Rotate" entry in the solid edit menu*
`mged>` *p 45 0 0*
`mged>`

The parameter *p* command is used to make precise rotation changes. The command is entered in the form *p a b c* where *a*, *b*, and *c* are the angles (in degrees) of rotation about the x, y, and z axes respectively. Point 1, the vertex, remains fixed, and the solid is rotated about this point. A positive angle of rotation is counter-clockwise when viewed in the positive direction along an axis.

The order of rotation is not commutative. Rotation takes place about the Z axis, Y axis, and X axis in that order. Figure 11.5 shows the rotation of 45 degrees about the X axis.

The values entered after the p are absolute - the rotations are applied to the primitive as it existed when solid rotation was first selected. Thus entering *p 0 0 0* will undo any rotations performed since solid rotation was begun.

`mged>` *p 0 45 0*
`mged>`

Figure 11.6 displays the solid after it has been rotated about the Y axis.

`mged>` *p 0 0 45*
`mged>`

Figure 11.7 displays the solid after it has been rotated about the Z axis.

`mged>` *p 0 0 0*
`mged>`

This restores the original orientation of the solid.

11.1.3 Scale Operation

`mged>` *Select the "Scale" entry in the solid edit menu*
`mged>` *p 2*
`mged>`

Figure 11.8 corresponds to the view that is shown on the display. The scale operation may be initiated by either selecting the Scale entry on the menu with the mouse, by depressing the Solid Scale button, or by entering *press sscale* on the keyboard. The parameter command *p n* is used to enter a precise scale factors, where *n* is the scale factor. The coordinates of point 1 remain the same. The distances from point 1 to the other points are multiplied by the scale value *n*. The general equations for the transformation from point p to p' are

```
x'[i] = x[i] + n (x[i] - x[1] )
y'[i] = y[i] + n (y[i] - y[1] )    i != 1
z'[i] = z[i] + n (z[i] - z[1] )
```

The size of the primitive may be changed by depressing the mouse at different positions. When the mouse is clicked, the edited primitive is scaled about point

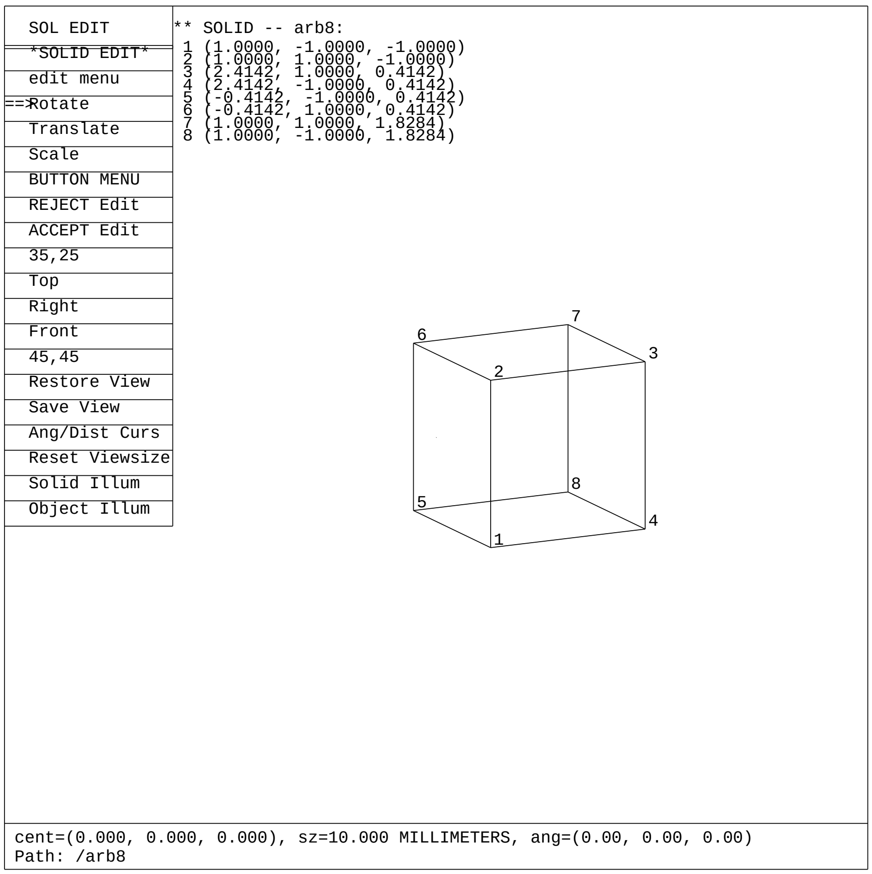

Figure 11.6: Solid Edit Rotation of 45 Degrees about Y

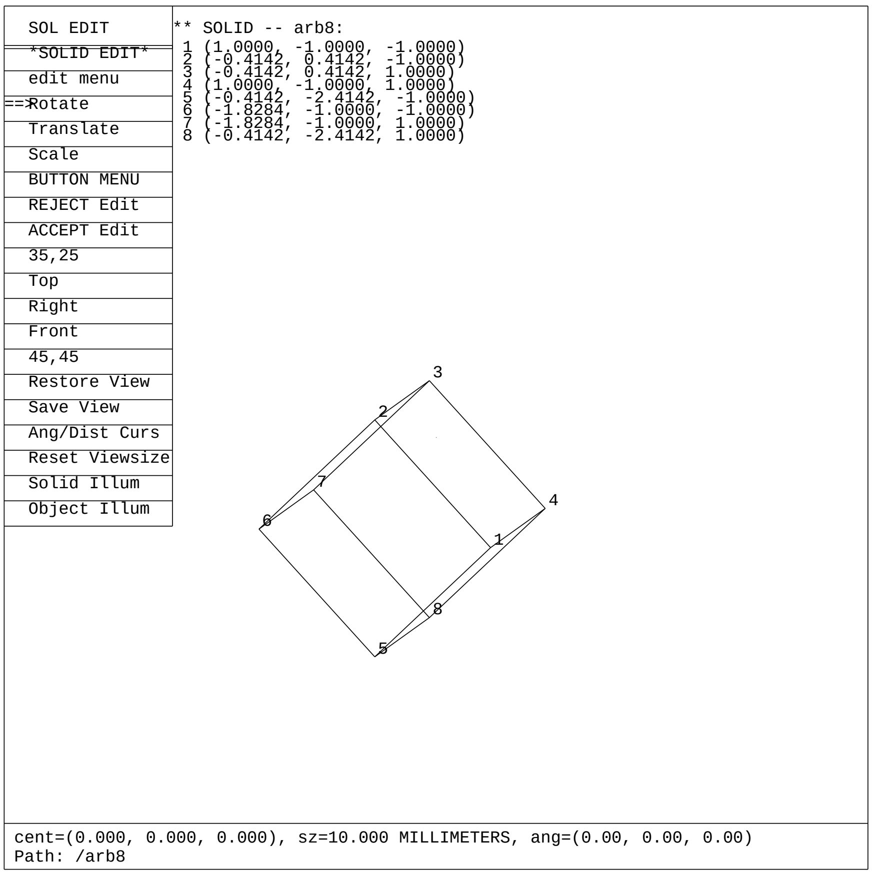

Figure 11.7: Solid Edit Rotation of 45 Degrees about Z

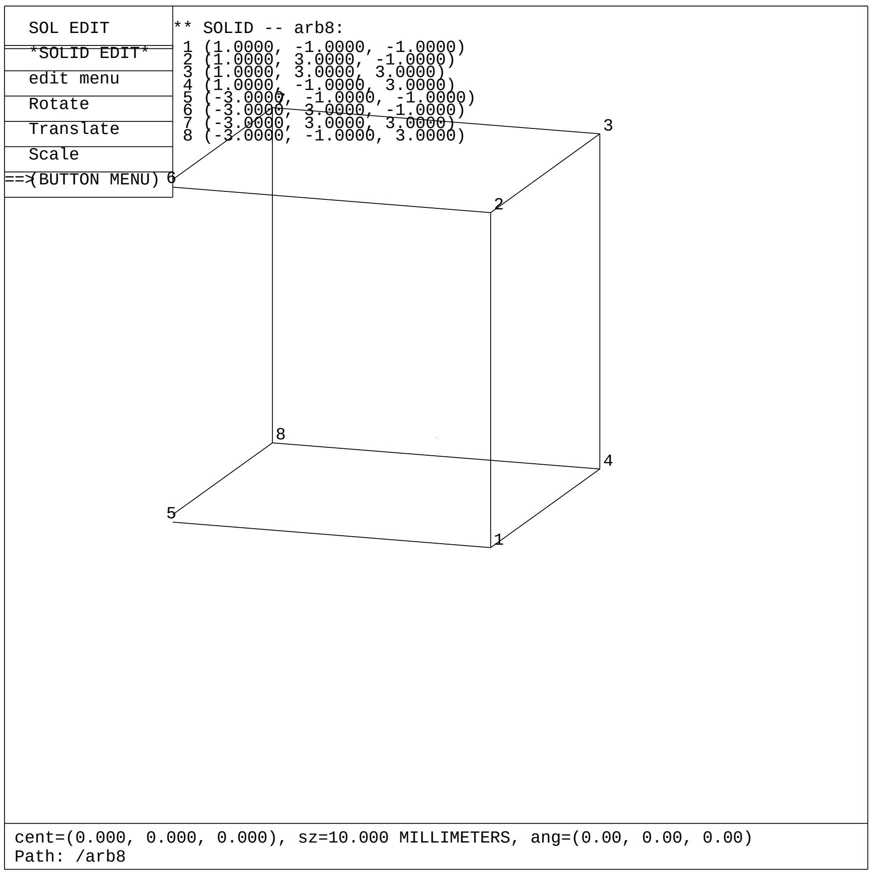

Figure 11.8: ARB8 Scale Increased by 2X

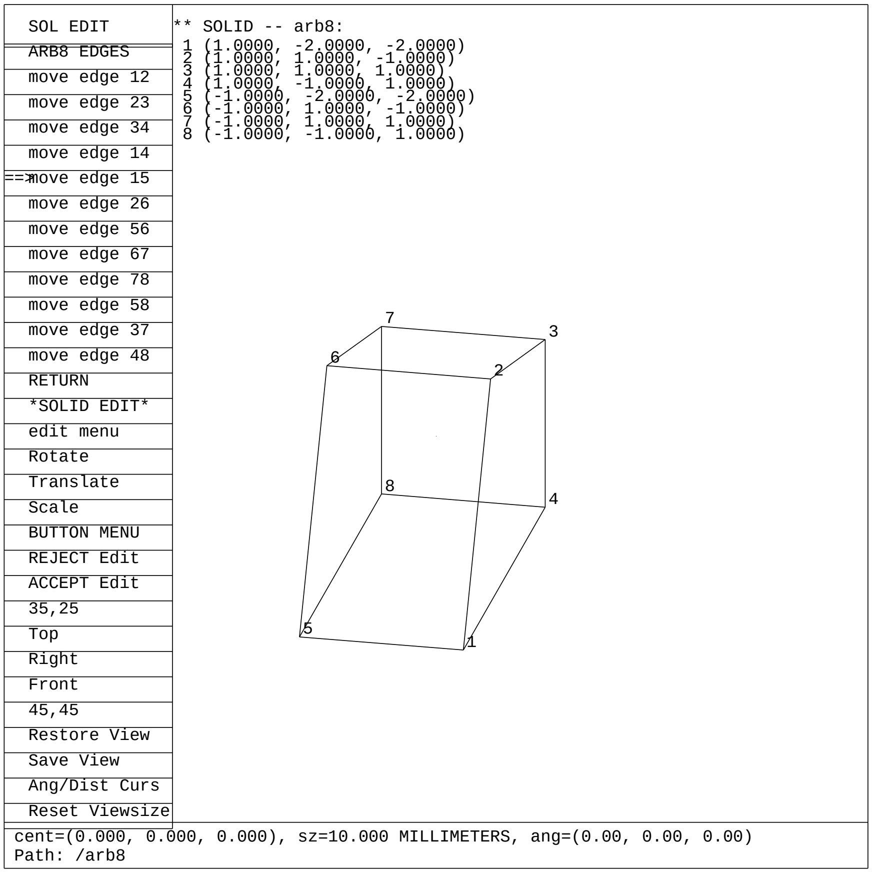

Figure 11.9: ARB8 Edge 15 Moved Through (9, -2, -2)

1 (the key point) by an amount proportional to the distance the mouse is from the center of the screen. If the mouse is above the center of the screen, the edited primitive will become larger. If the mouse is below the center, the primitive will become smaller.

The value of *n* entered is applied to the primitive as it existed when the solid scale state was entered.

Entering *p 1* will return the primitive to the size it had when the solid scale operation first started.

```
mged> p 1
mged>
```

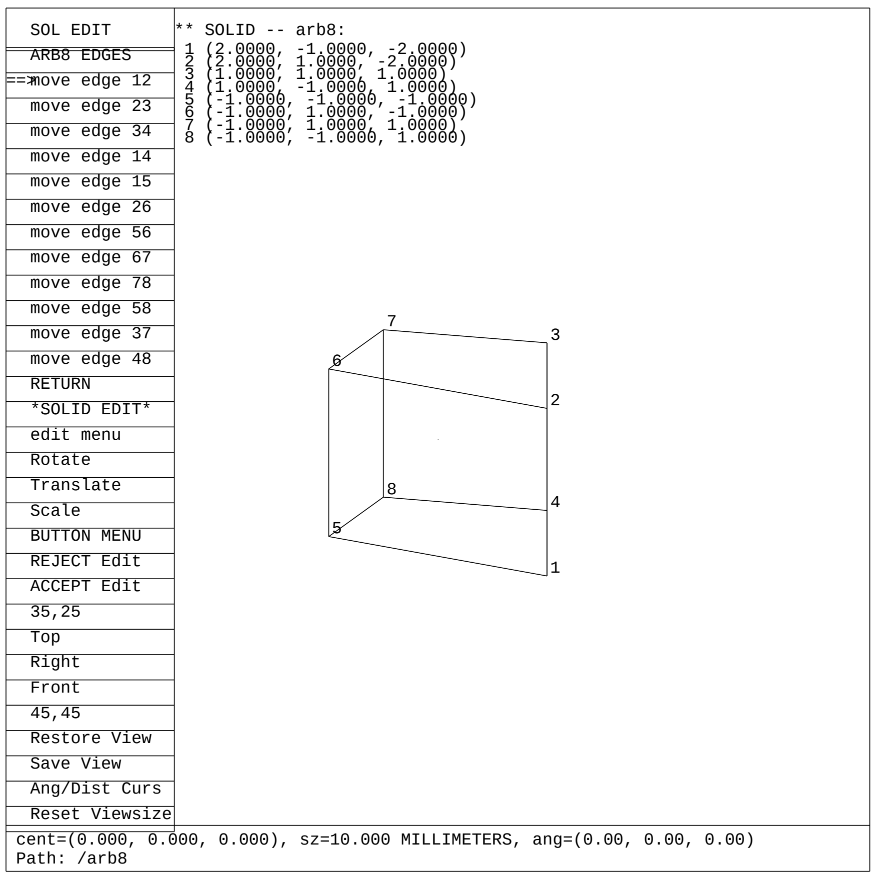

Figure 11.10: ARB8 Edge 12 Moved Through (2, 5, -2)

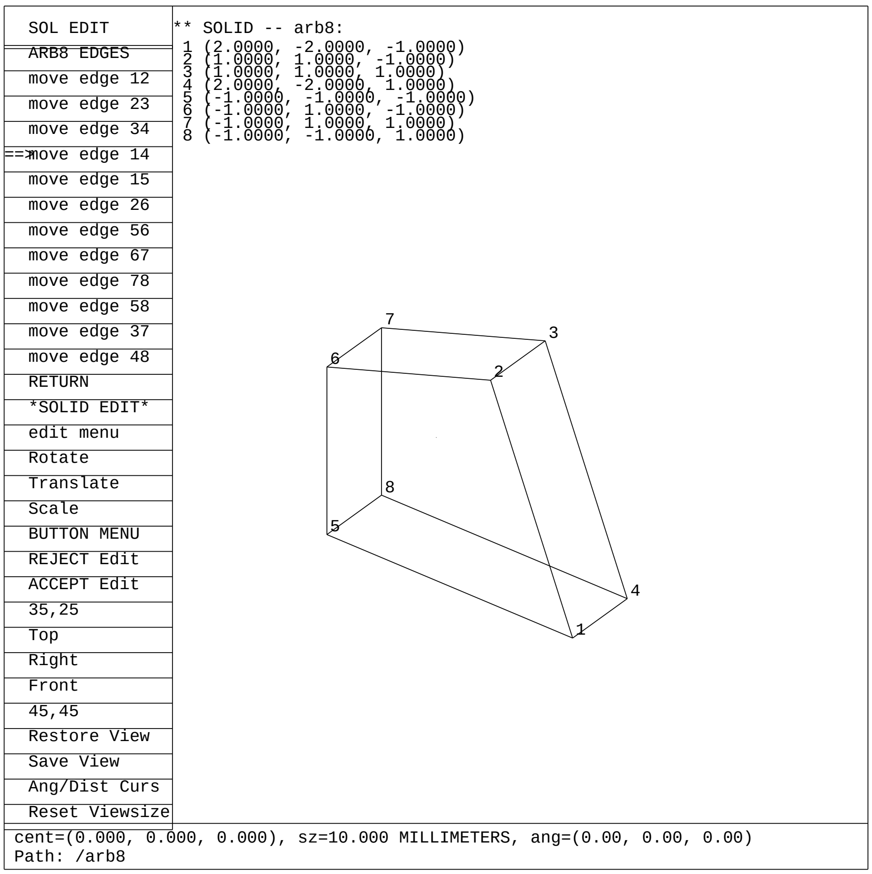

Figure 11.11: ARB8 Edge 14 Moved Through (2, -2, 7)

11.1.4 Moving Edges

The move edge command permits the moving of a line or edge so that the line passes through the selected point.

`mged>` *Select the "edit menu" entry in the solid edit menu*
`mged>` *Select the "move edges" entry in the ARB menu*
`mged>` *Select the "move edge 15" entry in the ARB8 edges menu*
`mged>` *p 9 -2 -2*
`mged>`

The edge 15 is moved so that it passes through the point (9, -2, -2). The coordinates of the new points 1 and 5 are the intersection of the new edge with the planes 234 and 678. Since both the old edge and new edge 15 are parallel to the X axis, the X coordinate of the point given by the *p* command has no meaning. The X coordinates for points 1 and 5 are not changed. See Figure 11.9.

`mged>` *p 9 -1 -1*
`mged>`

This restores the original shape. The choice of "9" for the X coordinate was arbitrary.

`mged>` *Select the "move edge 12" entry in the ARB8 edges menu*
`mged>` *p 2 5 -2*
`mged>`

The edge 12 is parallel to the Y axis. This command moves the points 1 and 2 so that their X and Z coordinates are 2 and -2. See Figure 11.10. The Y coordinates are not changed.

To restore the view, enter:

`mged>` *p 1 5 -1*
`mged>`

The choice of "5" for the Y coordinate was arbitrary.

`mged>` *Select the "move edge 14" entry in the ARB8 edges menu*
`mged>` *p 2 -2 7*
`mged>`

The edge 14 is parallel to the Z axis. This command moves the points 1 and 4 so that their X and Y coordinates are 2 and -2. See Figure 11.11. The Z coordinates are not changed.

`mged>` *p 1 -1 7*
`mged>`

This restores the original shape. The choice of "7" for the Z coordinate was arbitrary.

11.1.5 Extrude Command

The extrude command is used to move the opposite surface a distance from the specified surface or plate. This command can only be used when an ARB

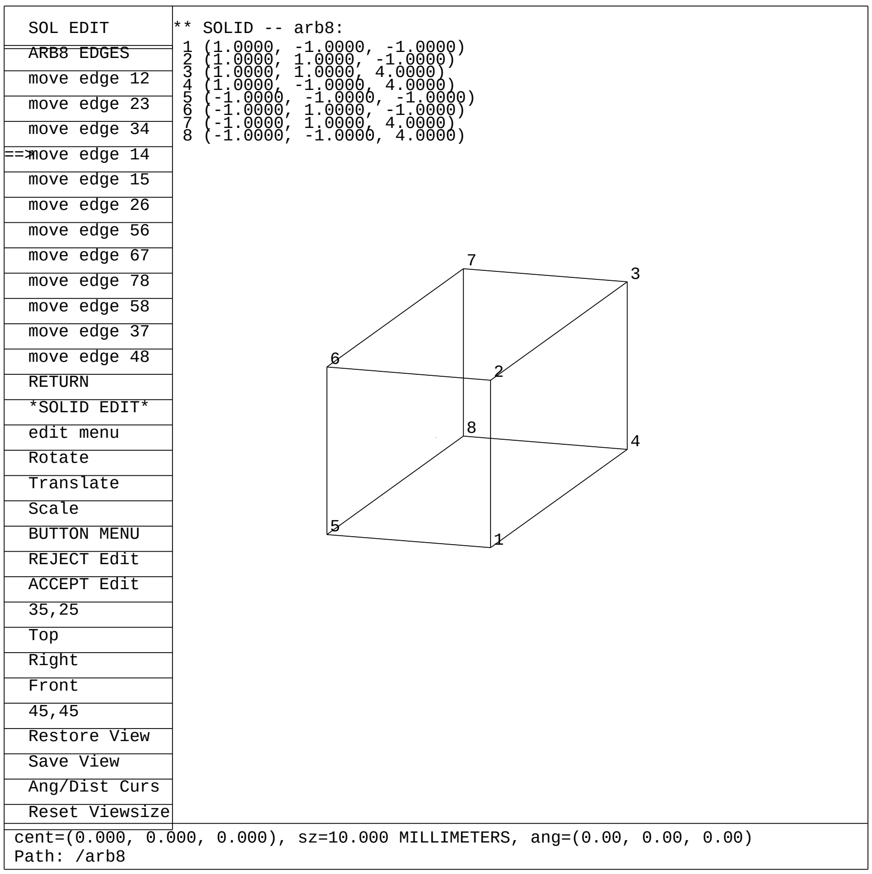

Figure 11.12: ARB8 Rear Face Extruded 5 Units in -Z

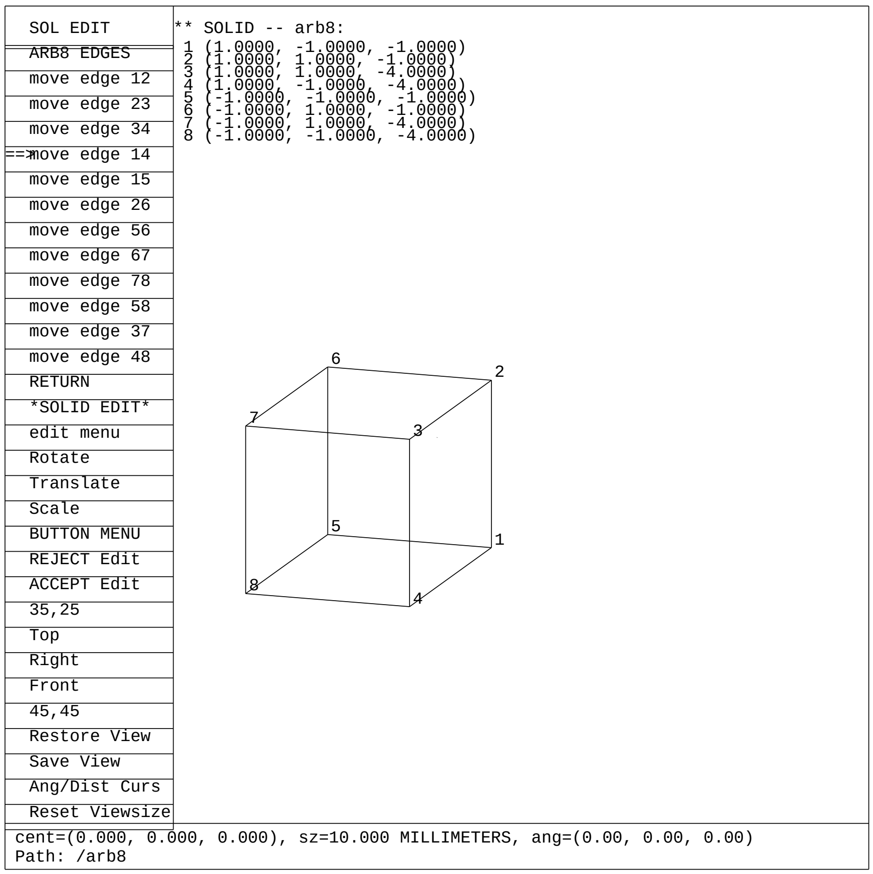

Figure 11.13: ARB8 Rear Face Extruded 3 Units in +Z

solid is in solid edit state.

```
mged> extrude 1265 5
mged>
```

In Figure 11.12, the plane opposite surface whose points are 1, 2, 6, and 5 is moved to a distance of 5 in the positive Z direction from plane 1265. Note that the points were selected counter-clockwise when viewed in the positive direction along the Z axis.

```
mged> extrude 1562 3
mged>
```

In Figure 11.13, the plane opposite surface 1562 is moved to a distance of 3 in the negative Z direction from 1562. Note that the points were selected clockwise when viewed in the positive direction along the Z axis.

```
mged> extrude 1265 2
mged>
```

This restores the original shape of this solid.

To return control to the VIEWING state, select the “REJECT Edit” item on the button menu, press the “reject” button on the button box, or enter the command *press reject* on the keyboard. Then, enter

```
mged> d arb8
mged>
```

to drop the ARB8 from view.

11.2 Solid Edit: A Five-Sided Polyhedron

This tutorial illustrates the application of the SOL EDIT state to the ARB5 solid. In this tutorial, the view is modified by using the rotation knobs so that all sides can be seen.

```
mged> size 6
mged> in arb5 arb5
Enter X, Y, Z for point 1:  0 0 0
Enter X, Y, Z for point 2:  0 0 1
Enter X, Y, Z for point 3:  0 1 1
Enter X, Y, Z for point 4:  0 1 0
Enter X, Y, Z for point 5:  -1 .5 .5
mged>
```

Figure 11.14 is the display of arb5 in the VIEWING state that is seen when the solid is first created. In this view, the Z axis is perpendicular to the viewing screen.

```
mged> Twist ROTY knob clockwise and restore
mged> Twist ROTX knob counter-clockwise and restore
mged>
```

These actions generate a view shown in Figure 11.15 that shows all sides.

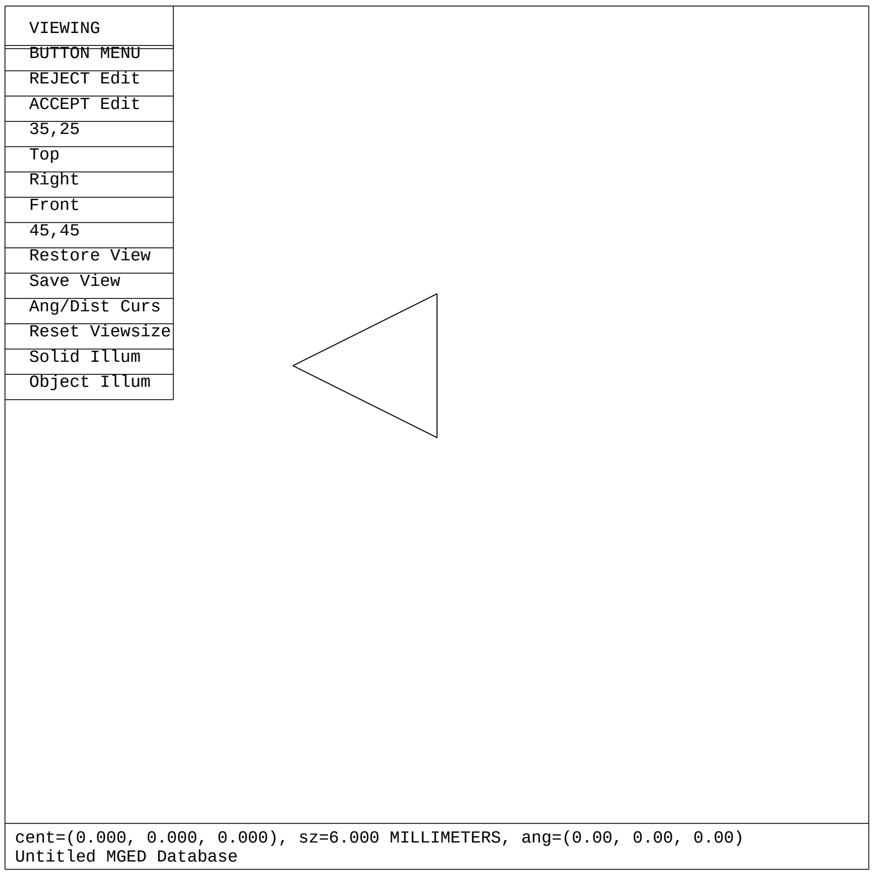

Figure 11.14: Top View of an ARB5

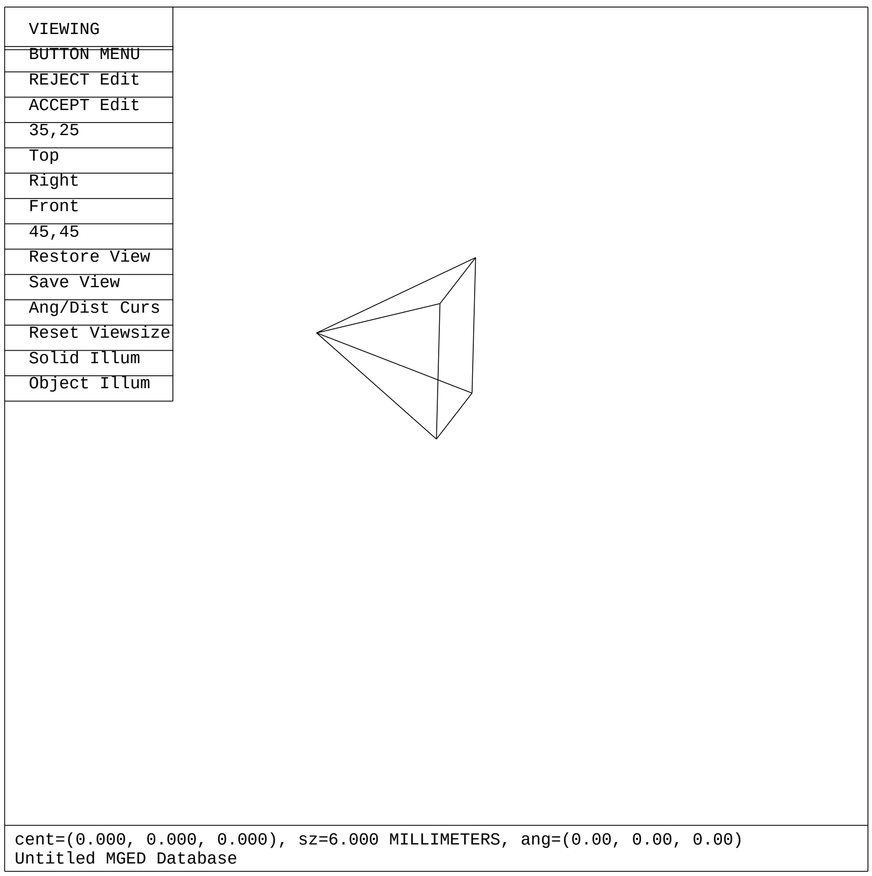

Figure 11.15: A Rotated View of the ARB5

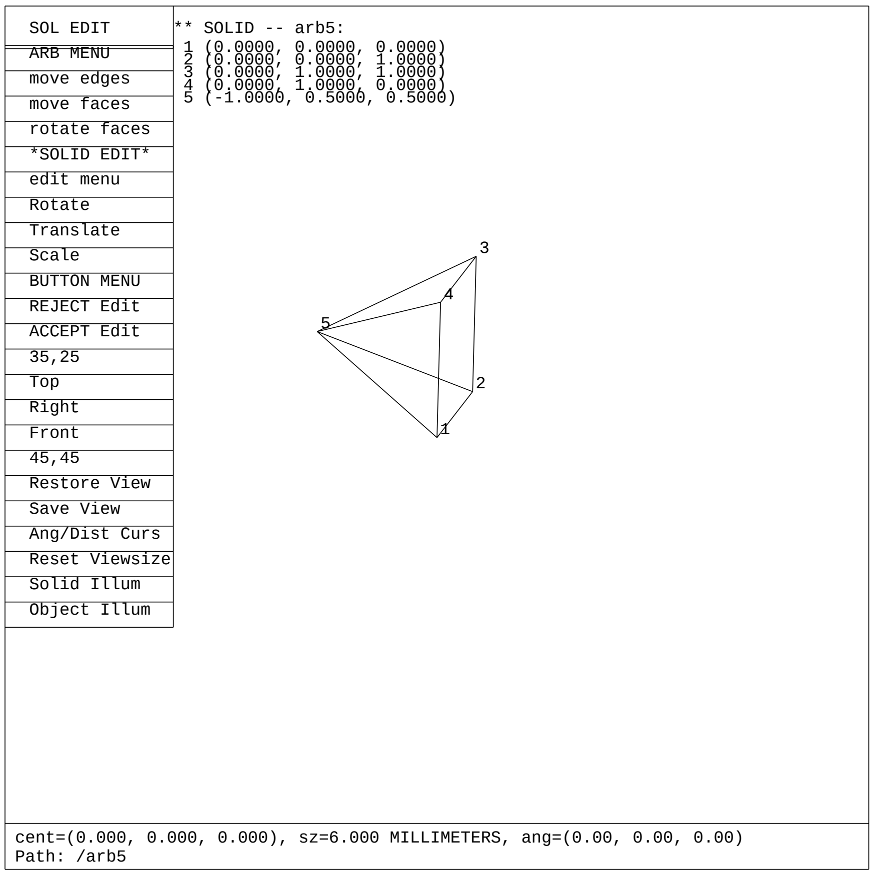

Figure 11.16: The ARB5 in Solid Edit State

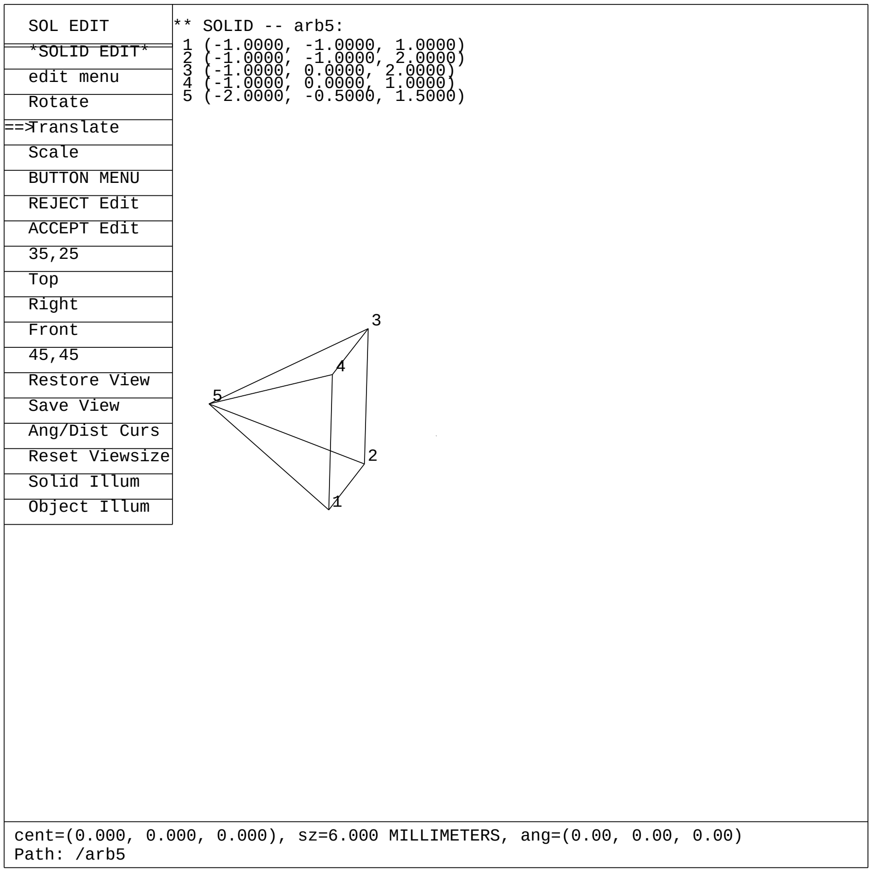

Figure 11.17: Translating an ARB5

mged> *Select the "Solid Illum" entry in the button menu*
mged> *Move the mouse out of the menu area*
mged> *Click the mouse to enter SOL EDIT state*
mged>

These actions will place MGED in the SOL EDIT state as shown in Figure 11.16.

11.2.1 Translate Operation

mged> *Select the "Translate" entry in the solid edit menu*
mged> *p -1 -1 1*
mged>

This command cause point 1 to be moved to coordinates (-1, -1, 1) and the

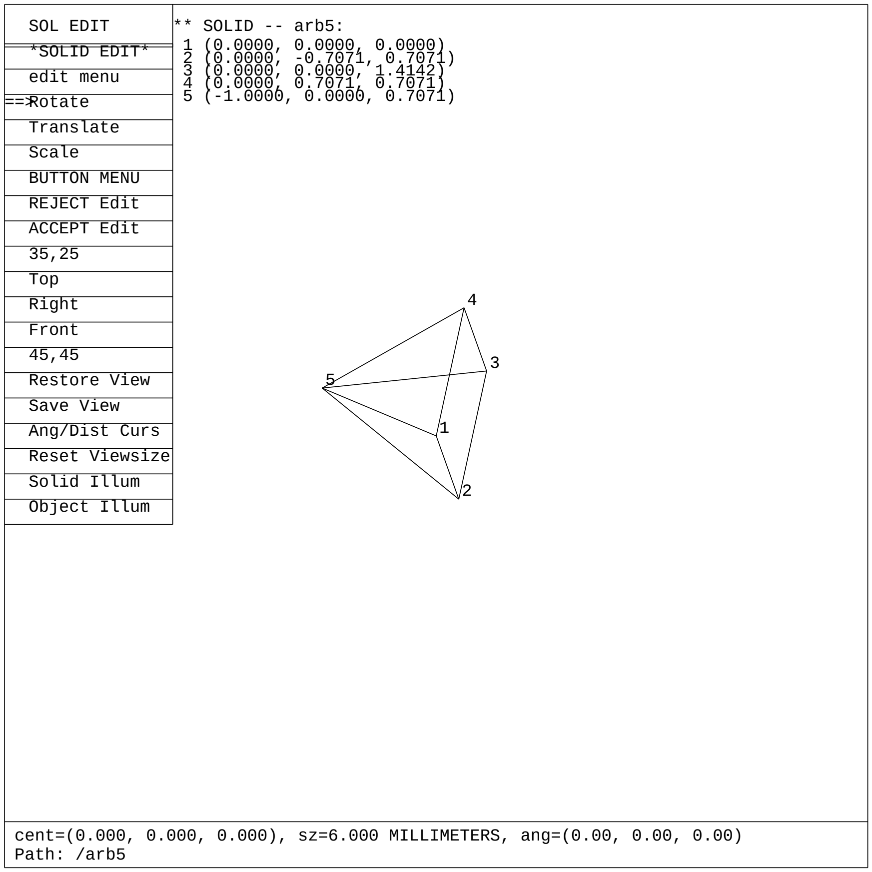

Figure 11.18: ARB5 Solid Edit Rotation about X

other points are moved so that they keep the same relative position to point 1. See Figure 11.17.

Enter this command to restore the solid to it's original location:

`mged>` *p 0 0 0*
`mged>`

11.2.2 Rotate Operation

`mged>` *Select the "Rotate" entry in the solid edit menu*
`mged>` *p 45 0 0*
`mged>`

Figure 11.18 shows a rotation of 45 degrees about an axis parallel to the X axis. The rotate command is entered in the form *p a b c* where *a*, *b*, and *c* are

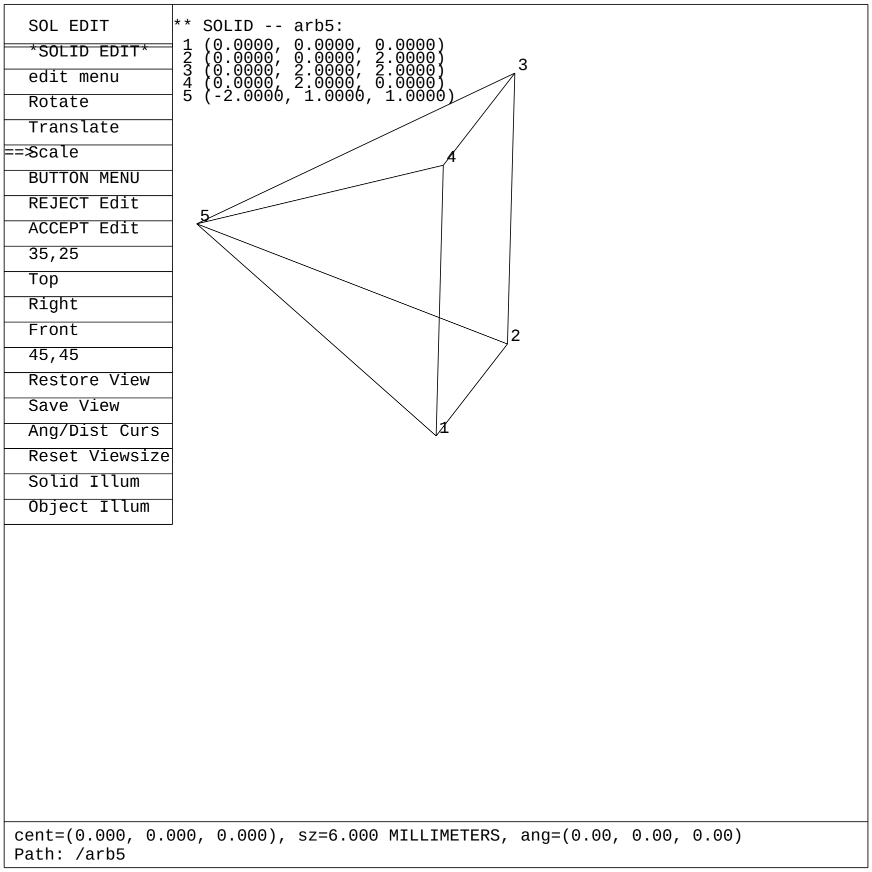

Figure 11.19: ARB5 Scale Increased by 2X

the angles (in degrees) of rotation about the x, y, and z axes and intersect at point 1. All rotation takes place about point 1.

`mged>` *p 0 0 0*
`mged>`

This restores the original orientation of the solid.

11.2.3 Scale Operation

`mged>` *Select the "Scale" entry in the solid edit menu*
`mged>` *p 2*
`mged>`

Figure 11.19 shows the change in the primitive. Point 1 remains the same and the distances of the other points from point 1 is multiplied by 2.

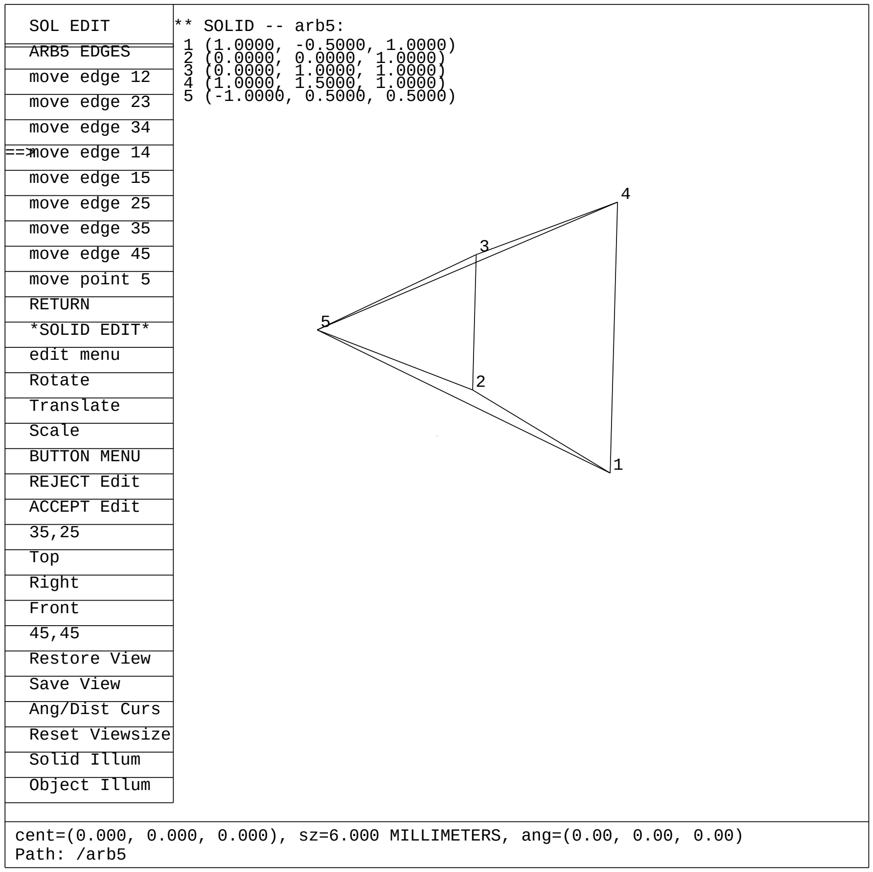

Figure 11.20: ARB5 Edge 14 Moved Through (1, 1, 1)

Entering *p 1* will return the primitive to the size it had when the solid scale operation first started.

`mged>` *p 1*
`mged>`

11.2.4 Move Edge Command

`mged>` *Select the "edit menu" entry in the solid edit menu*
`mged>` *Select the "move edges" entry in the ARB menu*
`mged>` *Select the "move edge 14" entry in the ARB8 edges menu*
`mged>` *p 1 1 1*
`mged>`

The edge 14 is moved so that it moves through the point (1, 1, 1). Note that

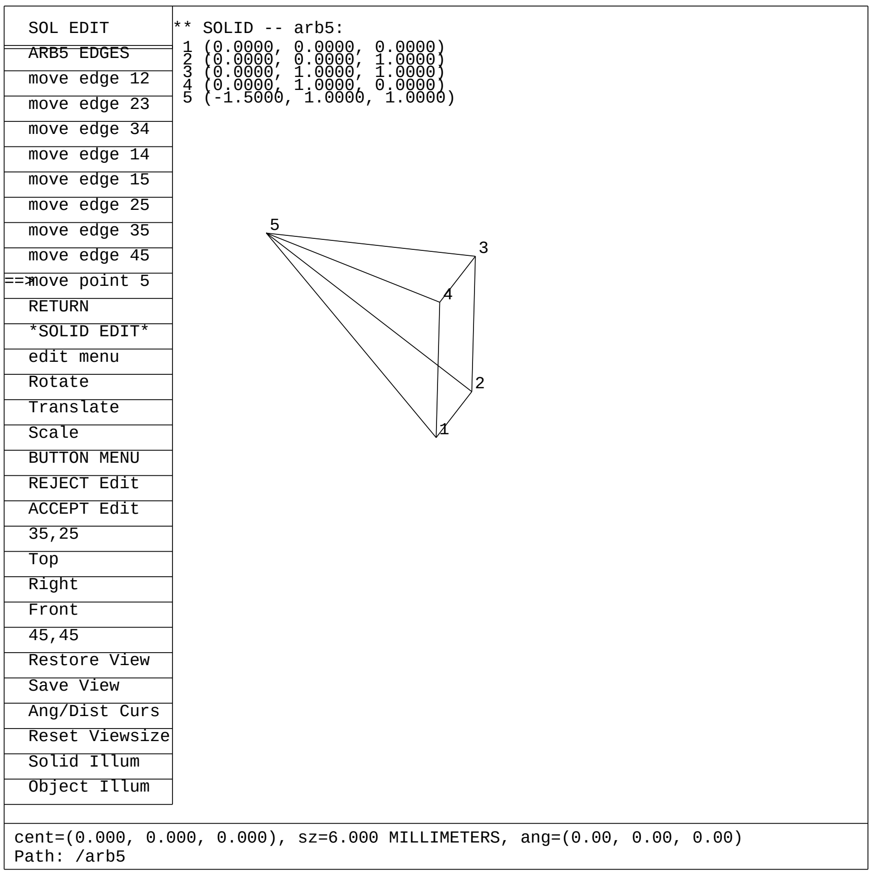

Figure 11.21: ARB5 Point 5 Moved to (-1.5, 1, 1)

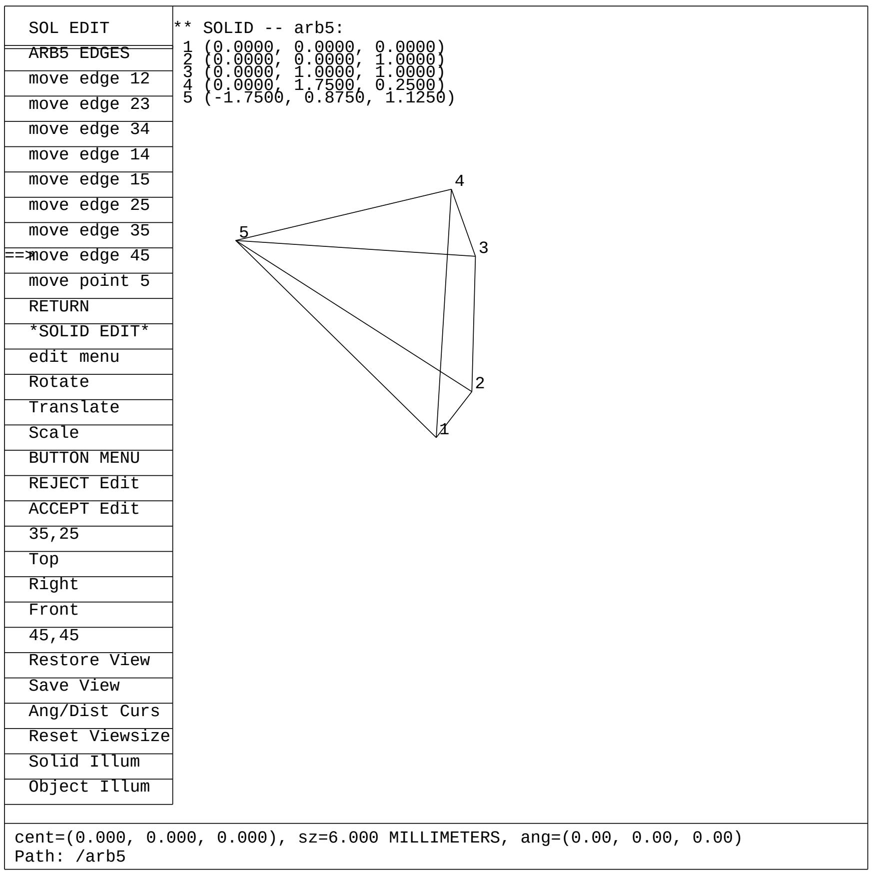

Figure 11.22: ARB5 Edge 45 Moved Through (-1.5 1 1)

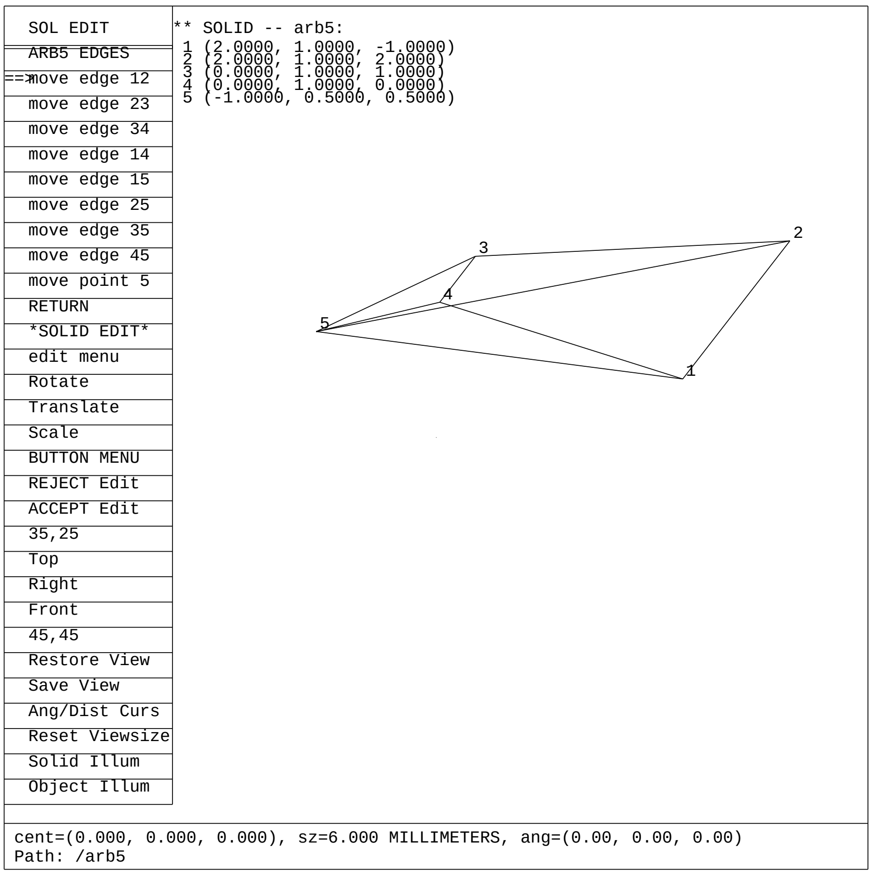

Figure 11.23: ARB5 Edge 12 Moved Through (2, 1, 2)

this point is the mid-point between points 1 and 4. See Figure 11.20.

`mged>` *p 0 2 0*
`mged>`

This restores the original shape.

`mged>` *Select the "move point 5" entry in the ARB5 edges menu*
`mged>` *p -1.5 1 1*
`mged>`

The point 5 is moved to location -1.5, 1, 1. See Figure 11.21.

`mged>` *p -1 .5 .5*
`mged>`

will restore the original shape.

`mged>` *Select the "move edge 45" entry in the ARB5 edges menu*
`mged>` *p -1.5 1 1*
`mged>`

In Figure 11.22, the edge 45 is moved so that it passes through the point (-1.5, 1, 1). Note that this point lies between the points 4 and 5.

`mged>` *p -1 .5 .5*
`mged>`

This restores the original shape.

`mged>` *Select the "move edge 12" entry in the ARB5 edges menu*
`mged>` *p 2 1 2*
`mged>`

In Figure 11.23, the edge 12 is moved so that it passes through the point (2, 1, 2). Note that the coordinates correspond to point 2.

The movement of the edges may yield unpredictable results when the edges are not parallel to one of the axes.

To return control to the VIEWING state, select the "REJECT Edit" item on the button menu, press the "reject" button on the button box, or enter the command *press reject* on the keyboard. Then, enter

`mged>` *d arb5*
`mged>`

to drop the ARB5 from view.

11.3 Solid Edit: Alter a Cylinder

This tutorial illustrates the application of the SOL EDIT state to cylinder solids.

`mged>` *size 12*
`mged>` *in cyl rcc*
`Enter X, Y, Z of vertex:` *0 0 0*
`Enter X, Y, Z of height (H) vector:` *2 0 0*
`Enter radius:` *1* `mged>`

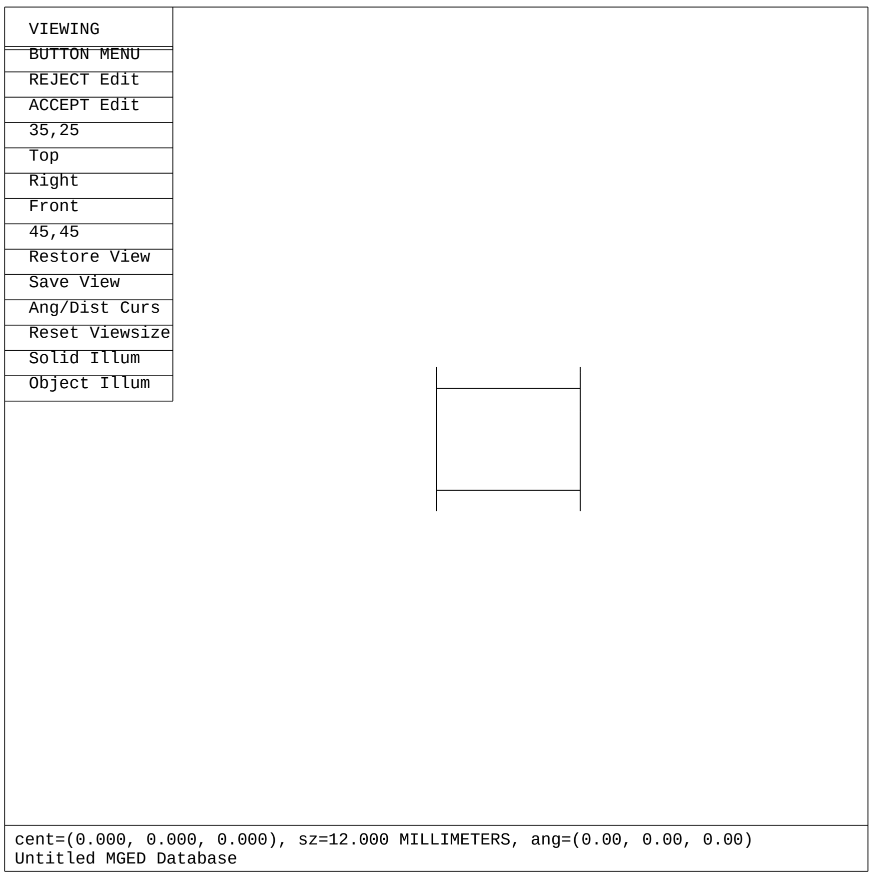

Figure 11.24: Top View of a Cylinder

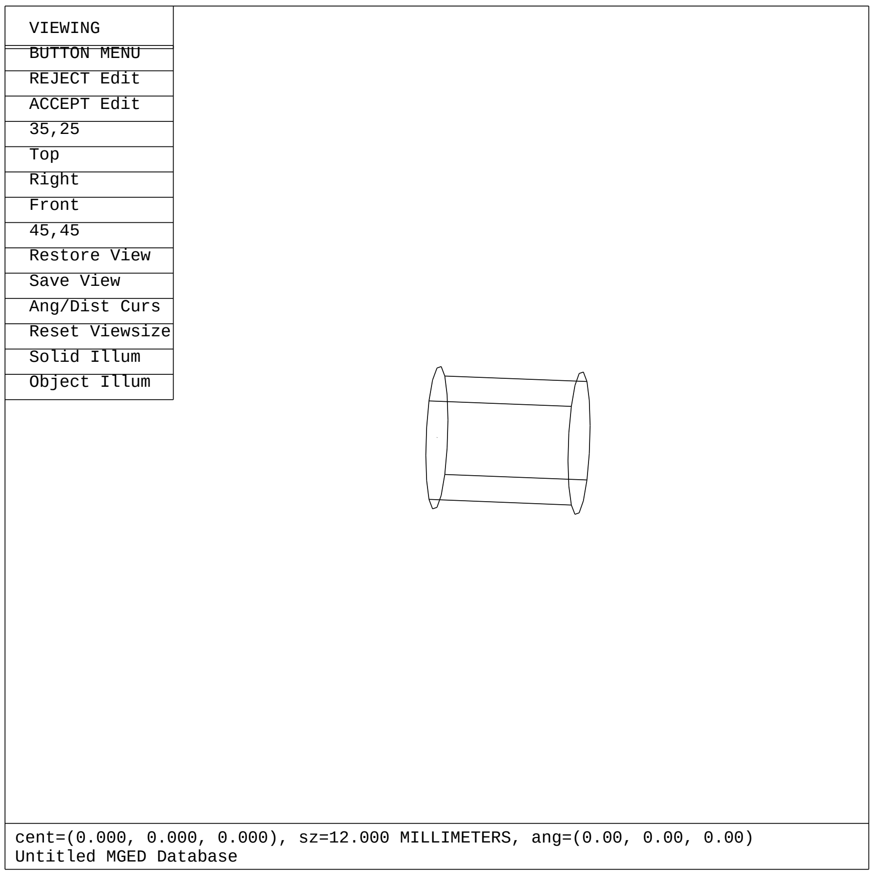

Figure 11.25: A Rotated View of the Cylinder

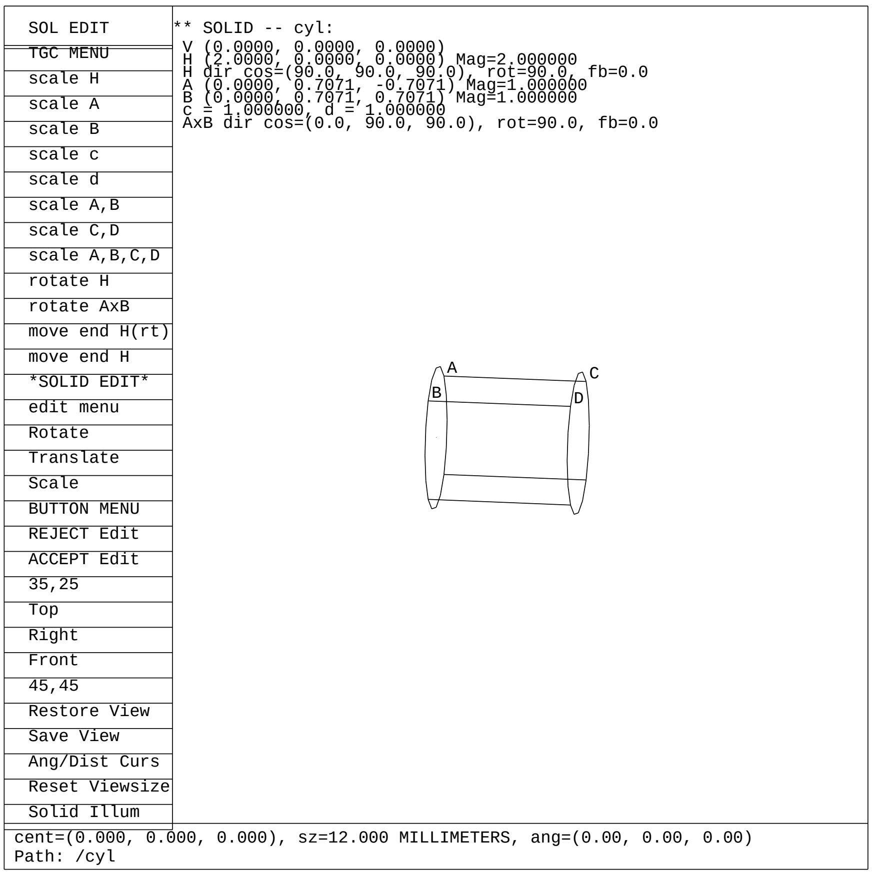

Figure 11.26: A Cylinder in Solid Edit State

Figure 11.24 is the display of the cylinder solid when viewed from the top. Since the Z axis is perpendicular to the viewing screen, a view of all sides cannot be seen.

`mged>` *Twist ROTY knob clockwise and restore*
`mged>` *Twist ROTX knob counter-clockwise and restore*
`mged>`

These actions generate a view, Figure 11.25, that shows all sides.

`mged>` *Select the "Solid Illum" entry in the button menu*
`mged>` *Move the mouse out of the menu area*
`mged>` *Click the mouse to enter SOL EDIT state*
`mged>`

Figure 11.26 is the view that displays the menu for the SOL EDIT state. The point V is at the origin (0,0,0) in this example and is in the middle of the circle that contains points A and B. H is the point of the center of the circle that contains points C and D. The coordinates of H are the coordinates of the vector from V to H and represent the relative position of H to V. Mag is the magnitude of these vectors and is represented by the formula

`Mag = sqrt( x + y + z )`

"H dir cos" are the direction cosines of the vector H which is perpendicular to plane of the points A, B, and V. The coordinates of A are the coordinates of the vector from V through A. Mag is the magnitude of the vectors from V to A. The coordinates of B are the coordinates of the vectors from V through B. Mag is the magnitude of the vector from V to B. The values for c and d are the magnitudes of the vectors from the tip of vector H to the points C and D respectively. "A x B dir cos" represents the direction cosines of the vector "A x B".

11.3.1 Translate Operation

`mged>` *Select the "Translate" entry in the solid edit menu*
`mged>` *p 1 1 1*
`mged>`

The location of the vertex point V is moved to (1, 1, 1). The locations of the other points relative to V remains the same. See Figure 11.27.

Move the mouse anywhere on the screen (outside the menu area), and click. Notice that the cylinder is moved so that V is placed at this location, and the coordinates of the other points remain the same relative to V.

`mged>` *p 0 0 0*
`mged>`

This restores the solid to the original location.

11.3.2 Rotate Operation

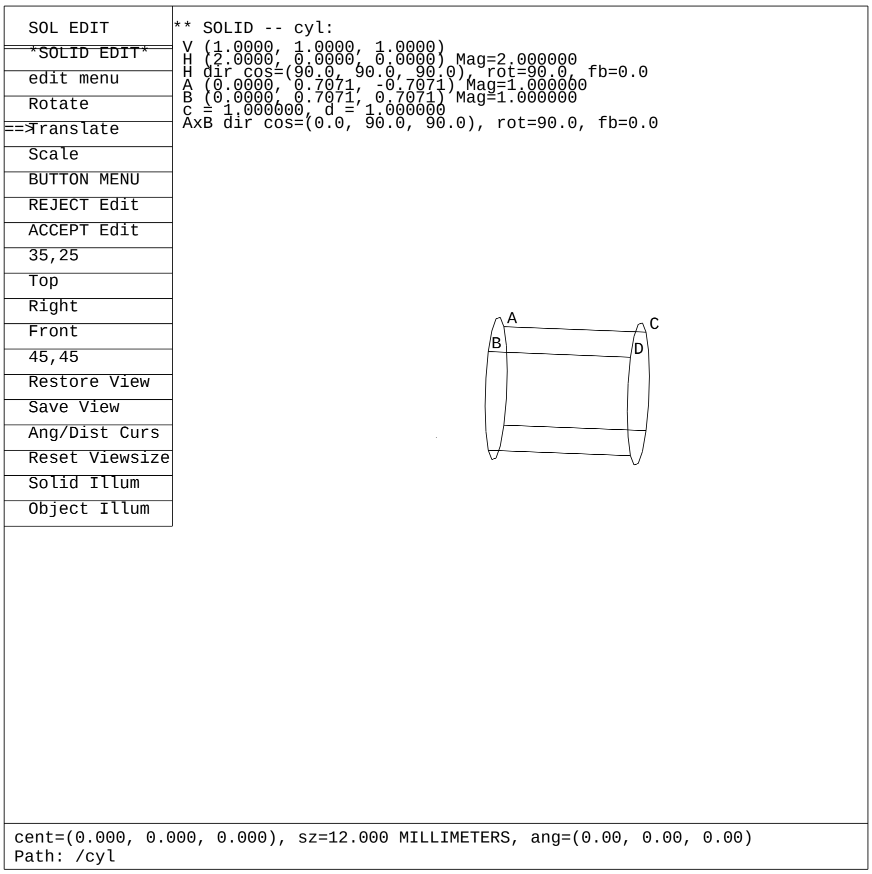

Figure 11.27: Translating Cylinder Vertex to (1, 1, 1)

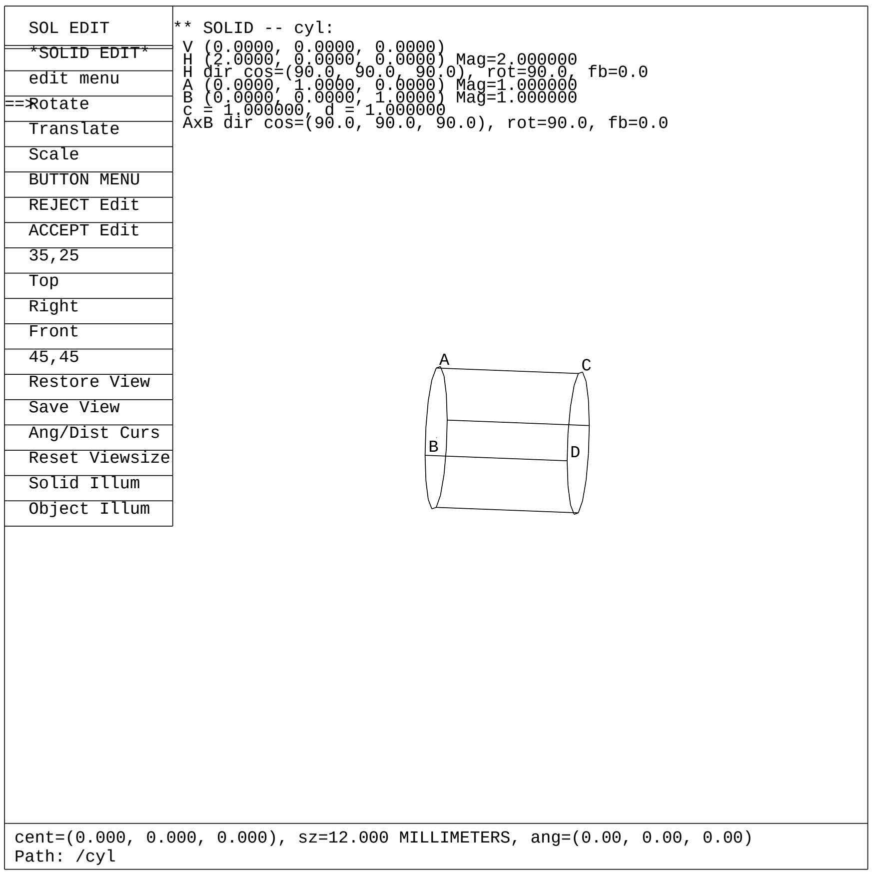

Figure 11.28: Solid Edit Rotation of 45 Degrees about X

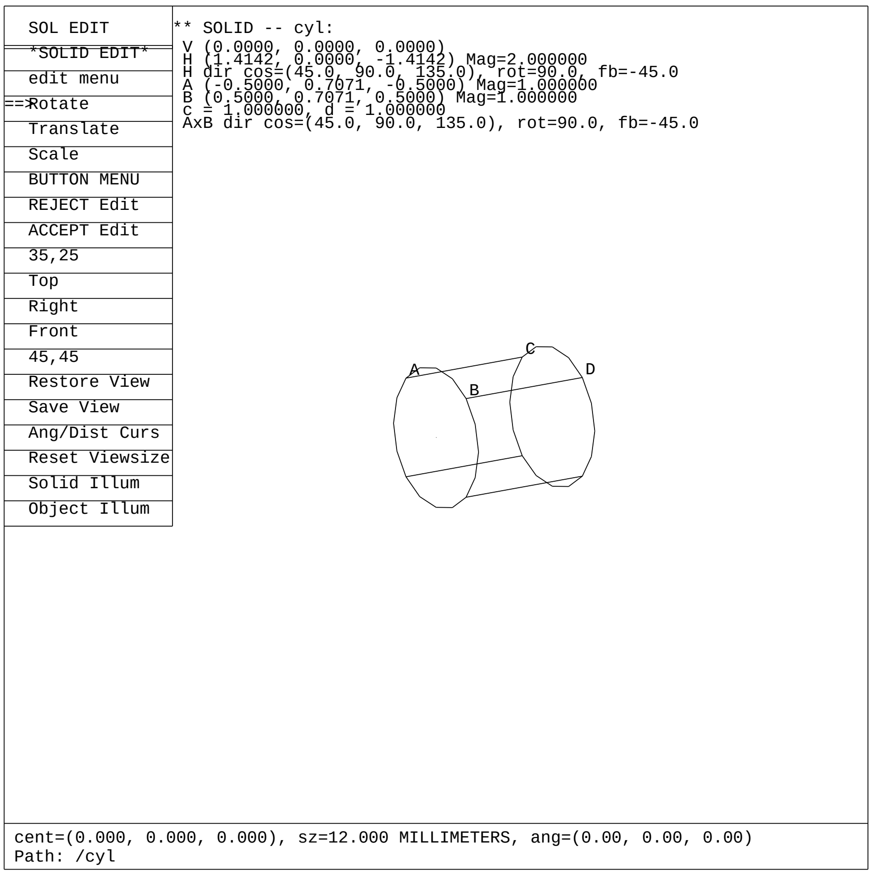

Figure 11.29: Solid Edit Rotation of 45 Degrees about Y

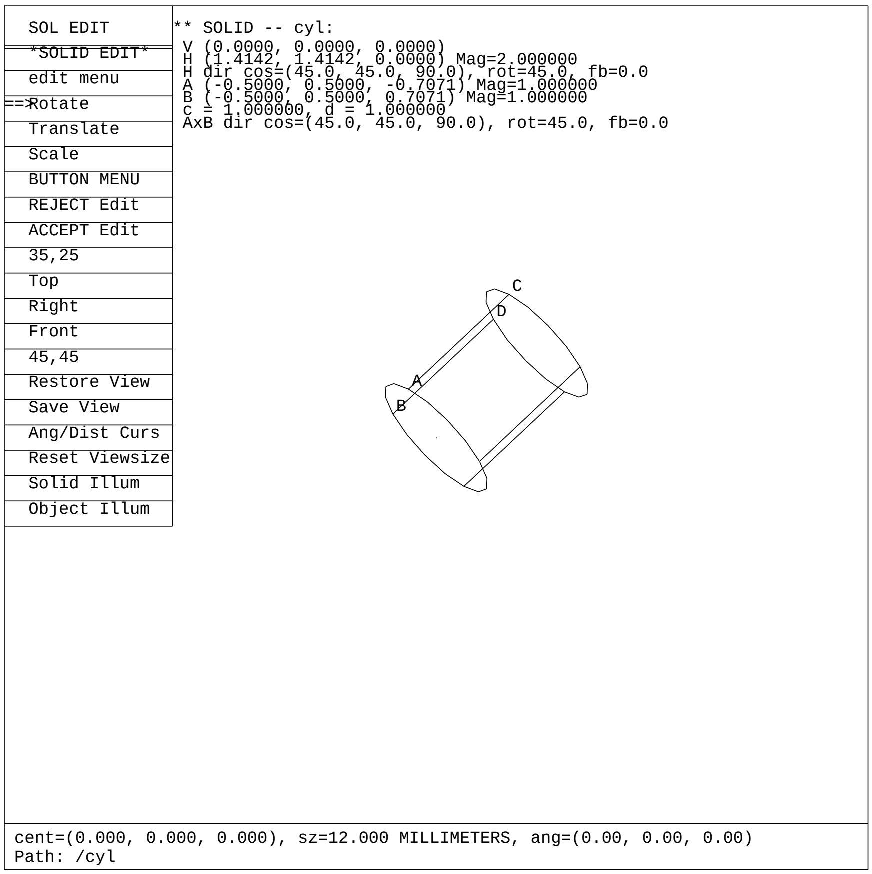

Figure 11.30: Solid Edit Rotation of 45 Degrees about Z

`mged>` *Select the "Rotate" entry in the solid edit menu*
`mged>` *p 45 0 0*
`mged>`

When viewing in the positive X direction, the cylinder is rotated counter-clockwise 45 degrees about an axis through point V parallel to the x axis. See Figure 11.28.

`mged>` *p 0 45 0*
`mged>`

When viewing in the positive Y direction, the cylinder is rotated counter-clockwise 45 degrees about an axis through point V parallel to the Y axis. See Figure 11.29.

`mged>` *p 0 0 45*
`mged>`

When viewing in the positive Z direction, the cylinder is rotated counter-clockwise 45 degrees about an axis through point V parallel to the Z axis. See Figure 11.30.

The command

`mged>` *p 0 0 0*
`mged>`

will restore the cylinder to the original orientation.

11.3.3 Scale Operation

`mged>` *Select the "Scale" entry in the solid edit menu*
`mged>` *p 1.5*
`mged>`

The point V remains fixed, the distance H between the two end-plate ellipses is multiplied by 1.5. See Figure 11.31.

The command

`mged>` *p 1*
`mged>`

restores the original scale.

11.3.4 Scale H Command

`mged>` *Select the "edit menu" entry in the solid edit menu*
`mged>` *Select the "scale H" entry in the TGC menu*
`mged>` *p 1*
`mged>`

The magnitude of the vector H is reduced from 2 to 1. See Figure 11.32. The command

`mged>` *p 2*
`mged>`

will restore the original shape.

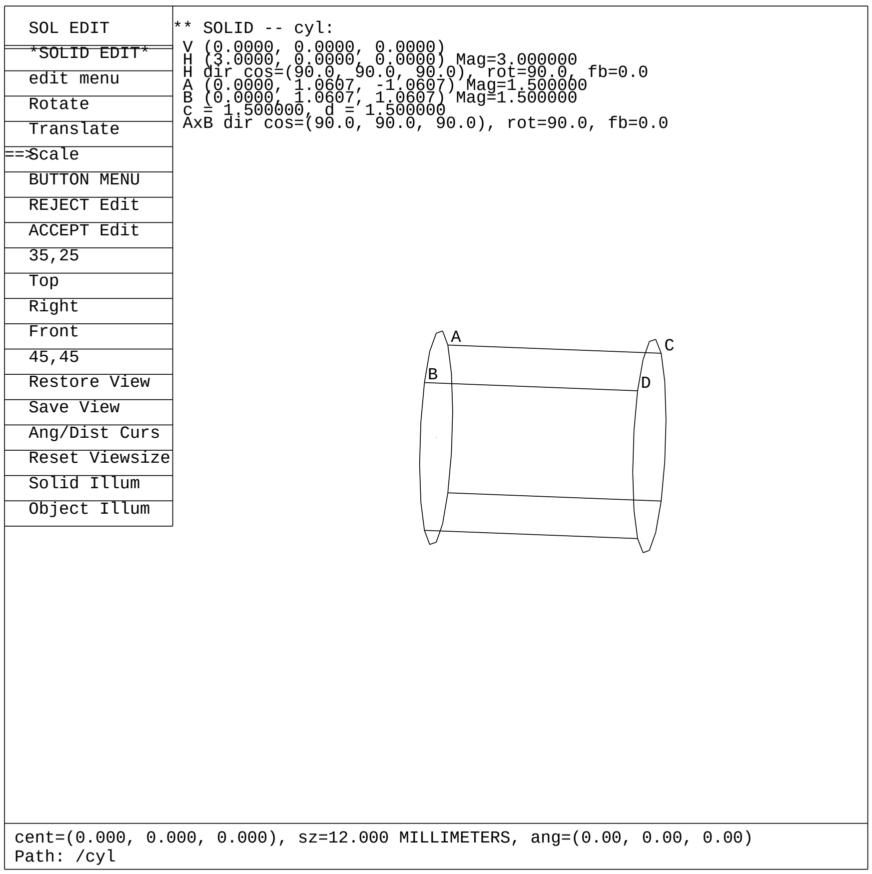

Figure 11.31: Cylinder Scale Increased by 1.5X

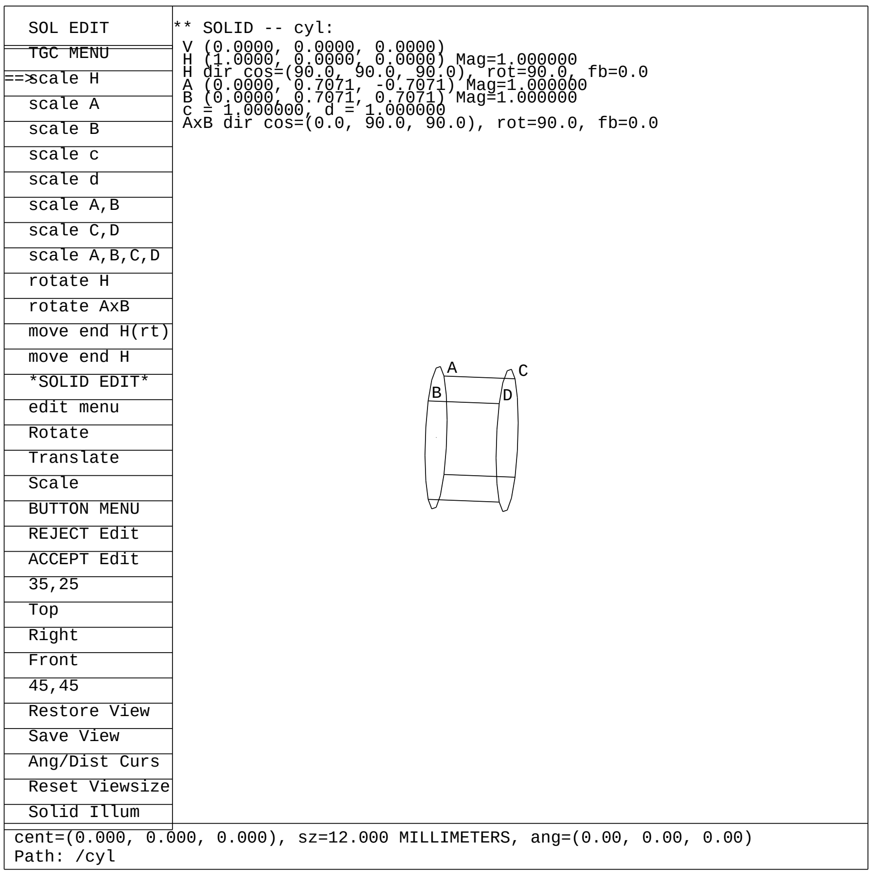

Figure 11.32: Cylinder Scale H Vector

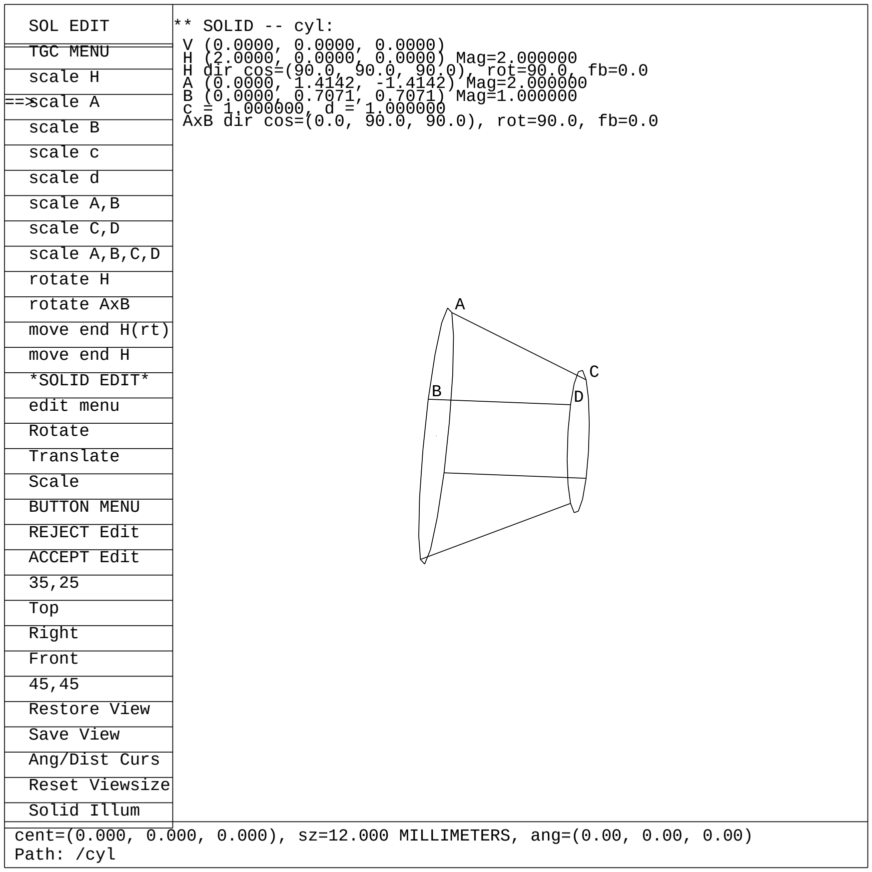

Figure 11.33: Cylinder Scale A Vector

11.3.5 Scale A Command

mged> *Select the "scale A" entry in the TGC menu*
mged> *p 2*
mged>

The magnitude of the vector through point A is increased to 2, ie, the length of the axis of the ellipse through point A is set equal to p. See Figure 11.33. The command

mged> *p 1*
mged>

will restore the original shape.

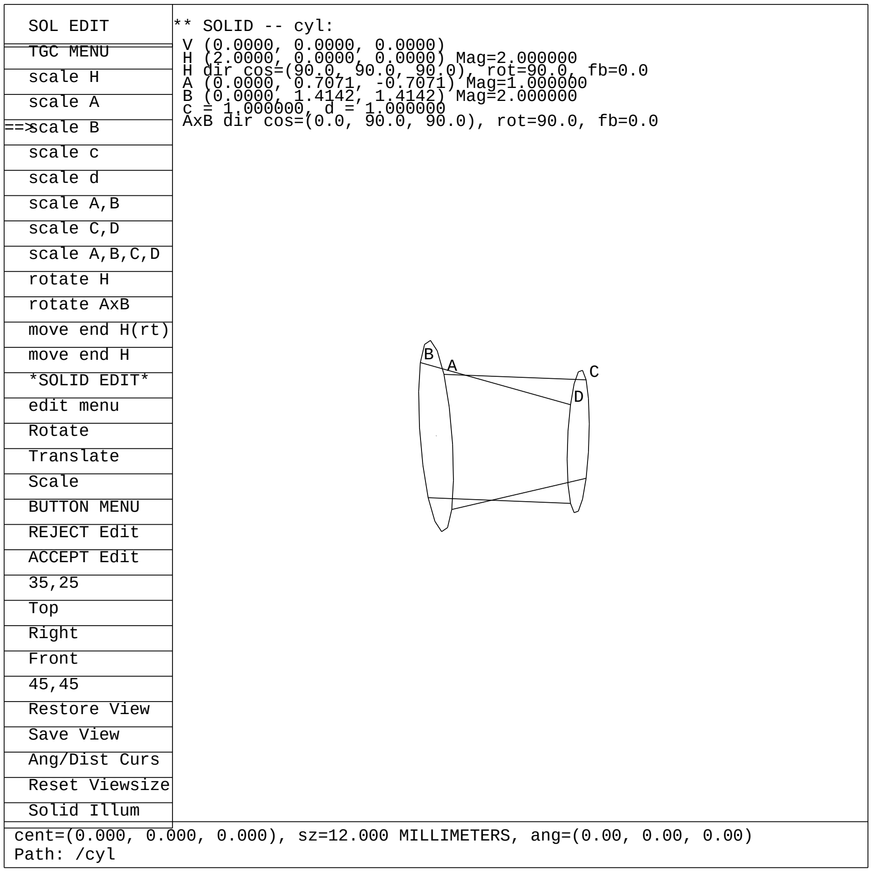

Figure 11.34: Cylinder Scale B Vector

11.3.6 Scale B Command

mged> *Select the "scale B" entry in the TGC menu*
mged> *p 2*
mged>

The magnitude of the vector through point B is increased to 2, ie, the length of the axis of the ellipse through point B is set equal to p. See Figure 11.34. The command

mged> *p 1*
mged>

will restore the original shape.

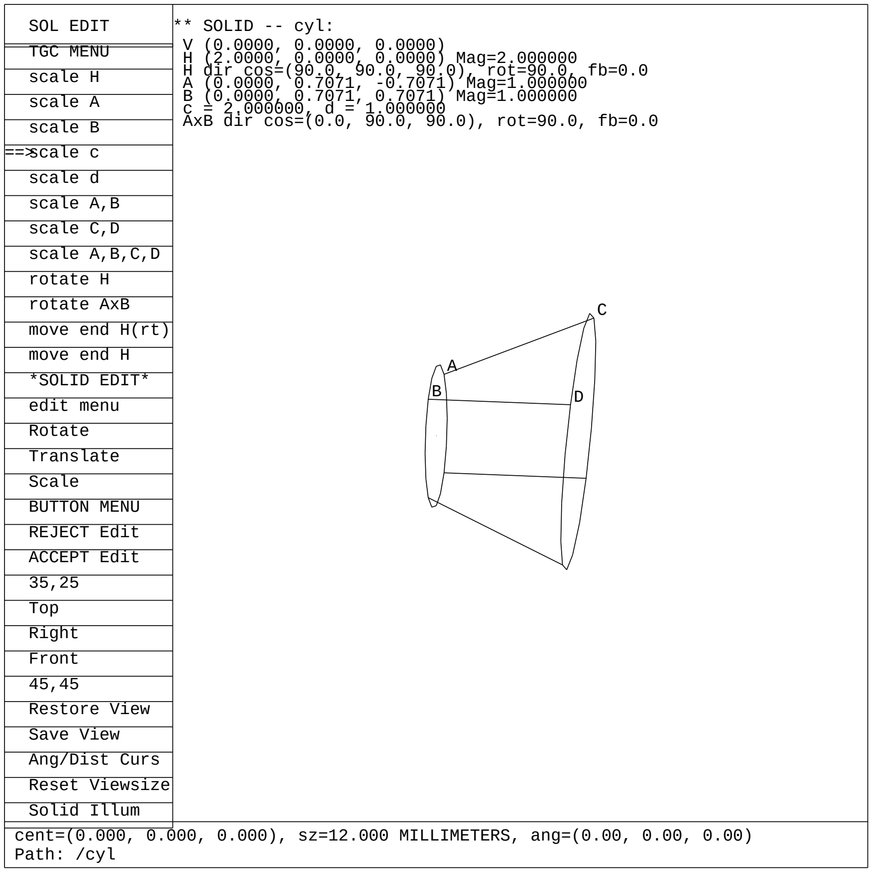

Figure 11.35: Cylinder Scale C Vector

11.3.7 Scale C Command

`mged>` *Select the "scale C" entry in the TGC menu*
`mged>` *p 2*
`mged>`

The magnitude of the vector through point c is increased to the value of p. The length of the axis of the ellipse through point c is set equal to the value of p. See Figure 11.35. The command

`mged>` *p 1*
`mged>`

will restore the original shape.

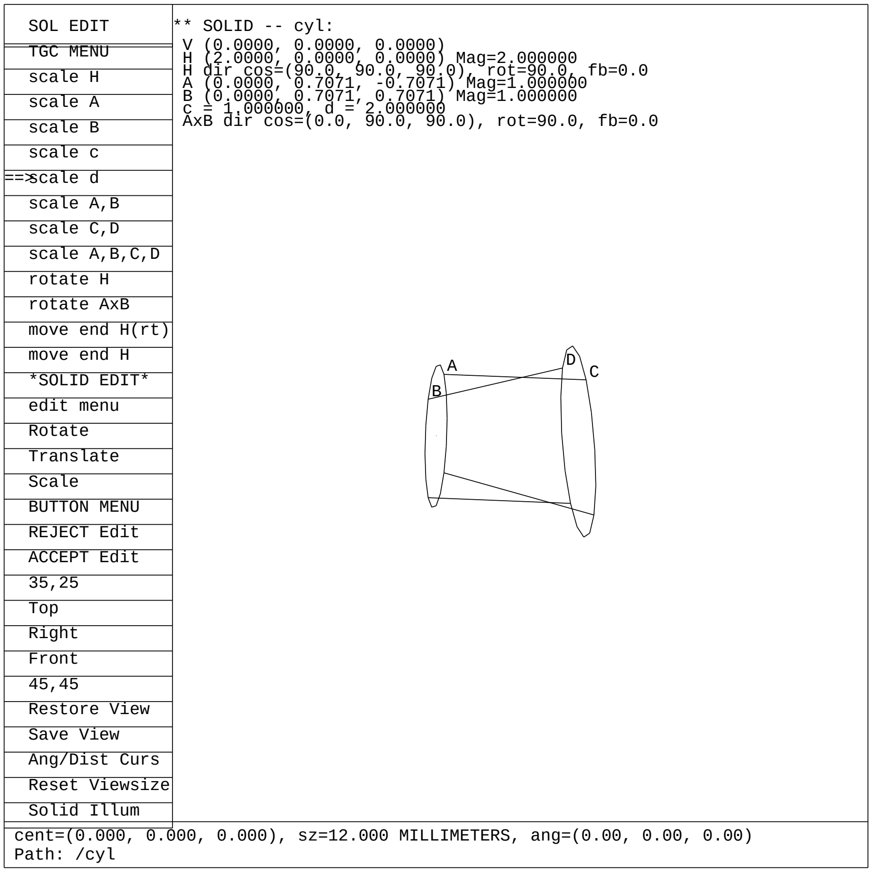

Figure 11.36: Cylinder Scale D Vector

11.3.8 Scale D Command

`mged>` *Select the "scale D" entry in the TGC menu*
`mged>` *p 2*
`mged>`

The magnitude of the vector through point D is changed to the value of p. The length of the axis of the ellipse through point D is set equal to the value of p. See Figure 11.36. The command

`mged>` *p 1*
`mged>`

will restore the original shape.

The scale H, A, B, C, and D commands provide for setting the magnitude equal to the value entered by the *p* command. The solid edit **scale** operation

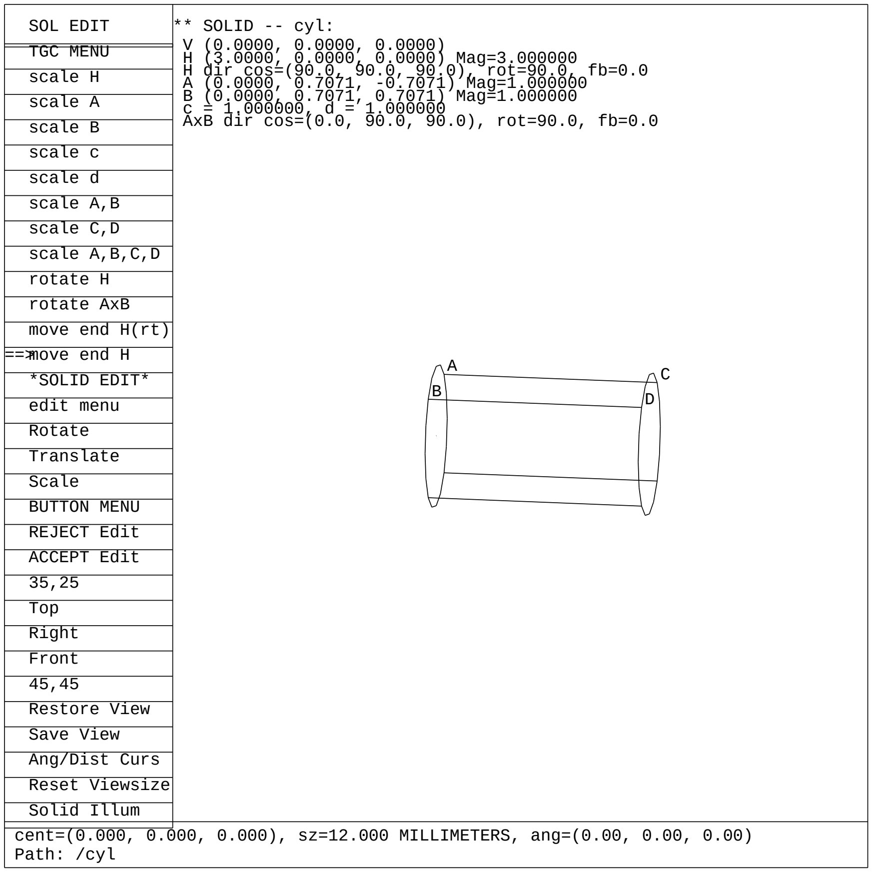

Figure 11.37: Cylinder Move End of H

provides for multiplying **all** the vectors by the value entered by the *p* command.

11.3.9 Move End H Command

mged> *Select the "move end H" entry in the TGC menu*
mged> *p 3*
mged>

The length of the vector H is changed to the value of p. See Figure 11.37. The command

mged> *p 2*
mged>

will restore the original shape.

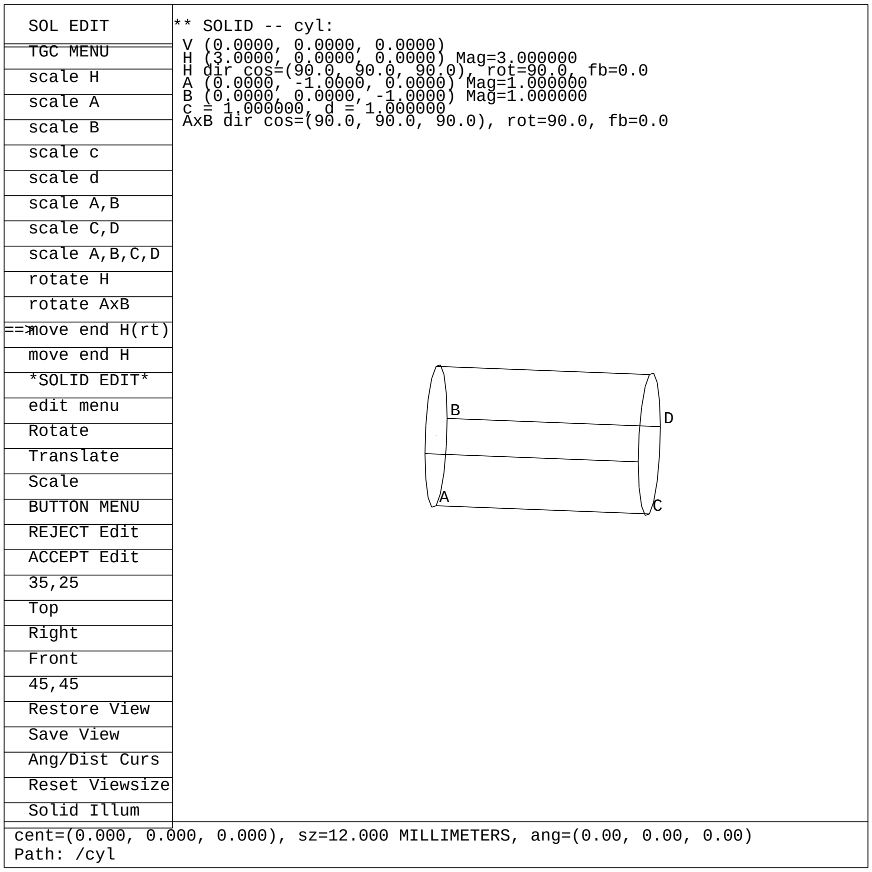

Figure 11.38: Cylinder Move End of H & Rotate

11.3.10 Move End H (rt) Command

mged> *Select the "move end H(rt)" entry in the TGC menu*
mged> *p 3*
mged>

This command is similar to the "move end H" command except the vector through point A is rotated so its direction is in the -Y direction. See Figure 11.38. The command

mged> *p 2*
mged>

will restore the original shape, but not the original orientation.

To return control to the VIEWING state, select the "REJECT Edit" item on the button menu, press the "reject" button on the button box, or enter the

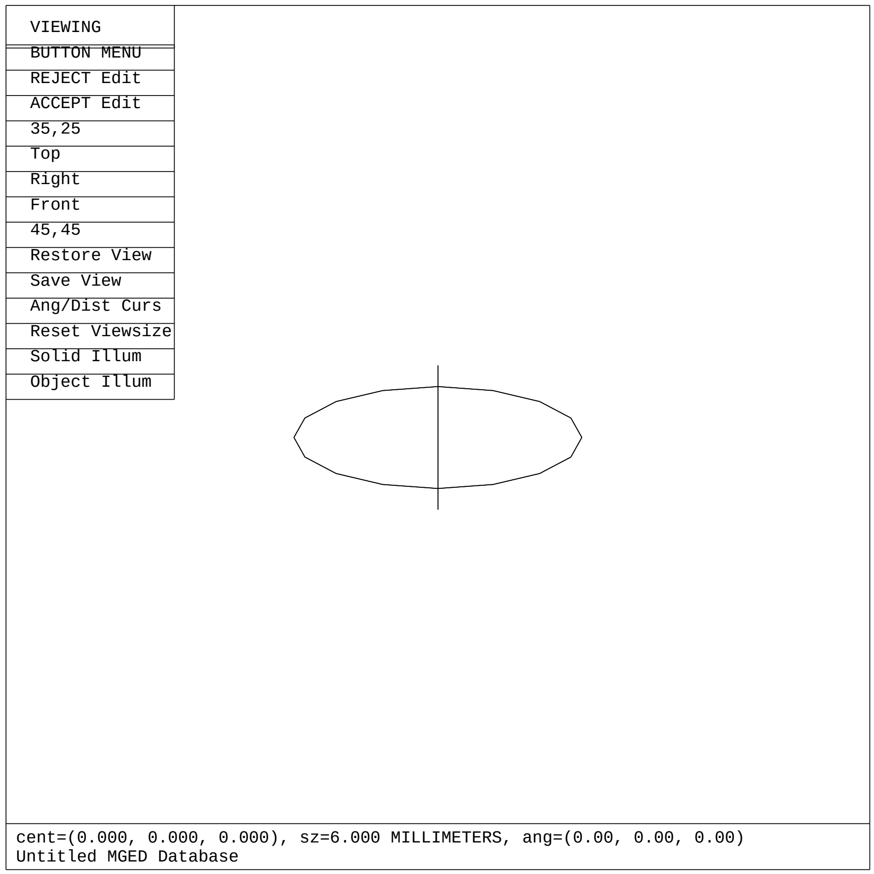

Figure 11.39: Top View of an Ellipsoid

command *press reject* on the keyboard. Then, enter

mged> *d cyl*
mged>

to drop the cylinder from view.

11.4 Solid Edit: Alter Ellipsoid

This tutorial illustrates the application of the SOL EDIT state to the ellipsoid primitive.

mged> press reset mged> *size 6*
mged> *in ell ellg*
Enter X, Y, Z of vertex: *0 0 0*

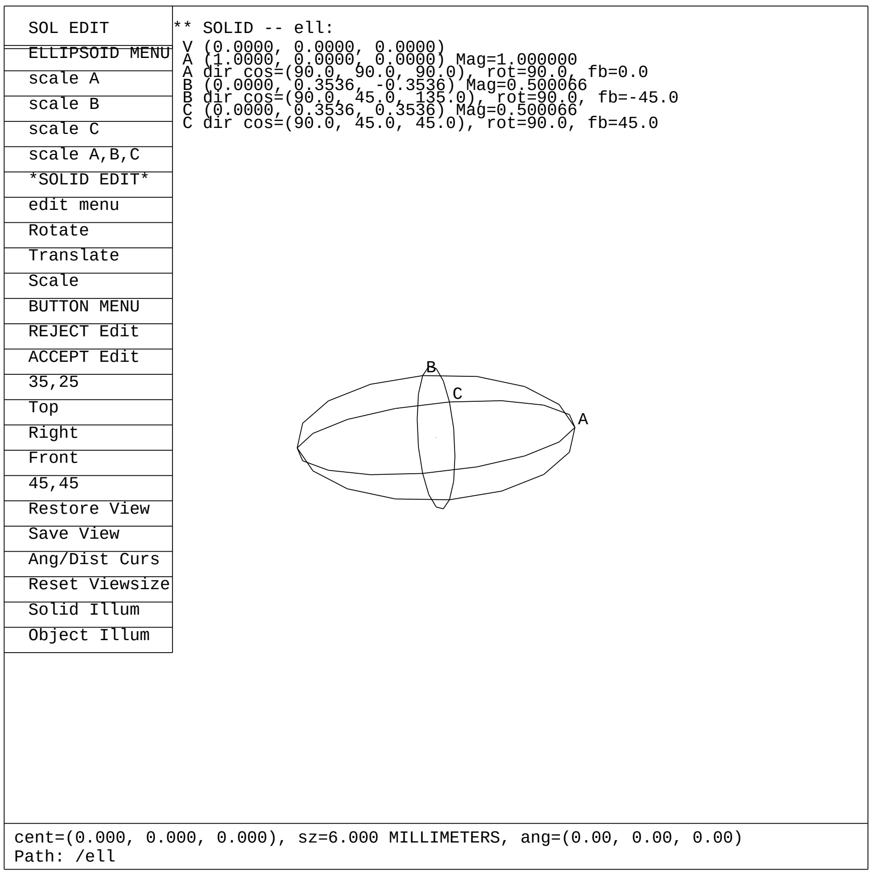

Figure 11.40: An Ellipsoid in Solid Edit State

`Enter X, Y, Z of vector A:` *1 0 0*
`Enter X, Y, Z of vector B:` *0 .3536 -0.3536*
`Enter X, Y, Z of vector C:` *0 .3536 0.3536*
`mged>`

Figure 11.39 is the display of the primitive in the viewing state. Since the Z axis is perpendicular to the viewing screen, a view of all sides cannot be seen.

`mged>` *Twist ROTY knob clockwise and restore*
`mged>` *Twist ROTX knob counter-clockwise and restore*
`mged>`

These actions generate a view that shows all sides.

`mged>` *Select the "Solid Illum" entry in the button menu*
`mged>` *Move the mouse out of the menu area*
`mged>` *Click the mouse to enter SOL EDIT state*
`mged>`

The display will be changed from the VIEWING MODE through the SOL PICK to the SOL EDIT state. Figure 11.40 is the view that is displayed.

The coordinates of the points A, B, C, are given by the product of the magnitude of the vector and the cosine of X, Y, and Z direction cosines. In the display, the coordinates are:

```
A = (1, 0, 0)
B = (0, 0.3536, -0.3536)
C = (0, 0.3536,  0.3536)
```

or

```
A = ( 1* cos  0,  1* cos 90,   1* cos 90 )
B = (.5* cos 90, .5* cos 45, -.5* cos 45 )
C = (.5* cos 90, .5* cos 45,  .5* cos 45 )
```

11.4.1 Translate Operation

`mged>` *Select the "Translate" entry in the solid edit menu*
`mged>` *p -1 1 1*
`mged>`

The key point V is moved to (-1, 1, 1) and the ellipsoid maintains its relative position to V. See Figure 11.41.

While in the SOL EDIT state, the solid may be translated by using the mouse. These changes are not numerically exact, but they can be useful to visually position a solid with respect to other solids. Move the mouse to a position outside the menu area on the screen. Click the mouse. The center point (V) of the ellipsoid will be translated to that point. Note that only the value of the coordinates of V are changed. The command

`mged>` *p 0 0 0*
`mged>`

will restore the original position.

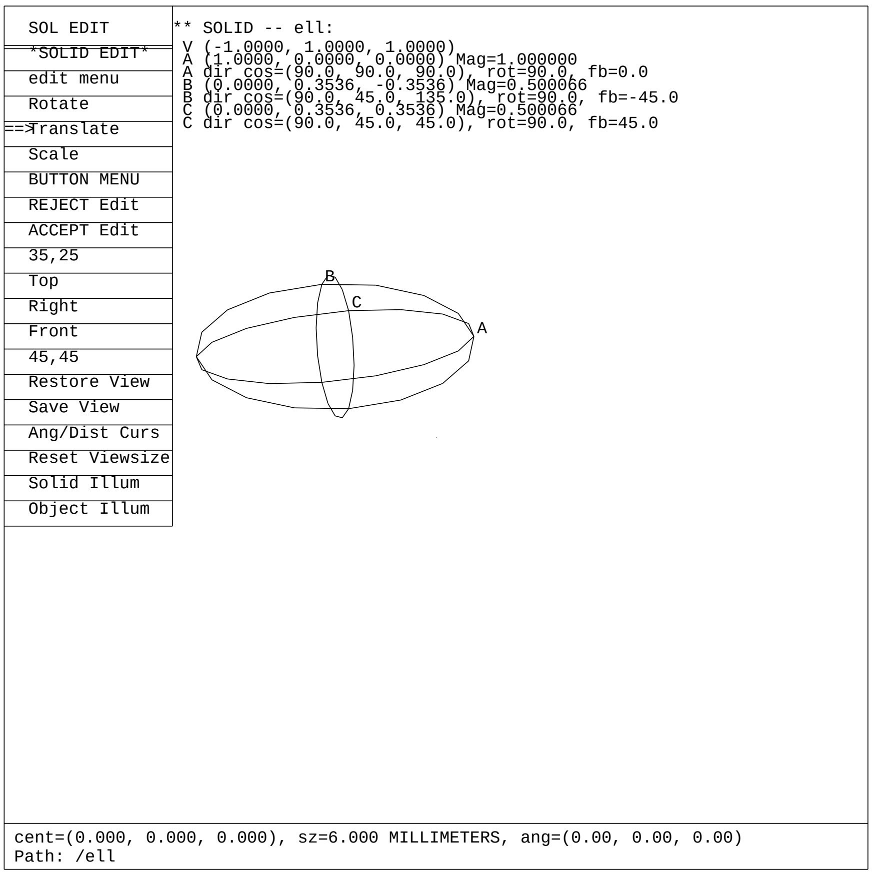

Figure 11.41: Translating Ellipsoid to (-1, 1, 1)

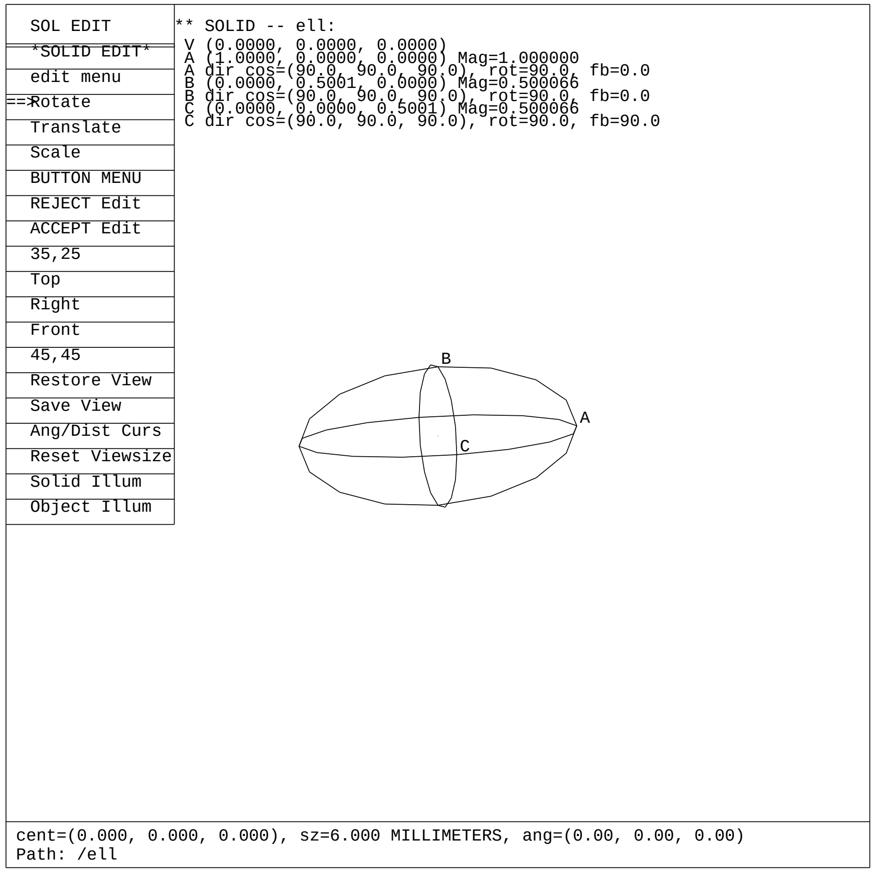

Figure 11.42: Solid Edit Rotation of 45 Degrees about X

11.4.2 Rotate Operation

The rotate operation is initiated by either selecting Rotate on the menu screen with the mouse, by depressing the Solid Rotate button on the button box, or by entering the *press srot* command on the keyboard.

mged> *Select the "Rotate" entry in the solid edit menu*
mged> *p 45 0 0*
mged>

Figure 11.42 shows the rotation of the ellipsoid about its X axis. The angle of rotation is counter-clockwise when viewed in the positive X direction. The direction cosines of vectors VB and VC are changed by 45 .

mged> *Select the "Rotate" entry in the solid edit menu*
mged> *p 0 45 0*

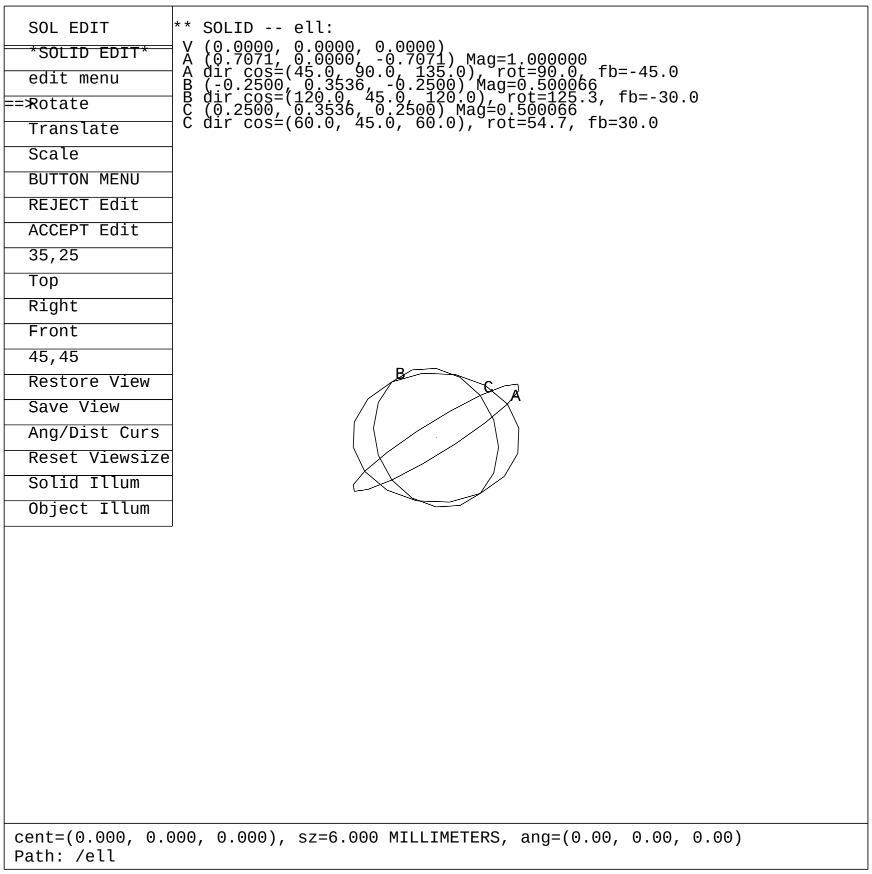

Figure 11.43: Solid Edit Rotation of 45 Degrees about Y

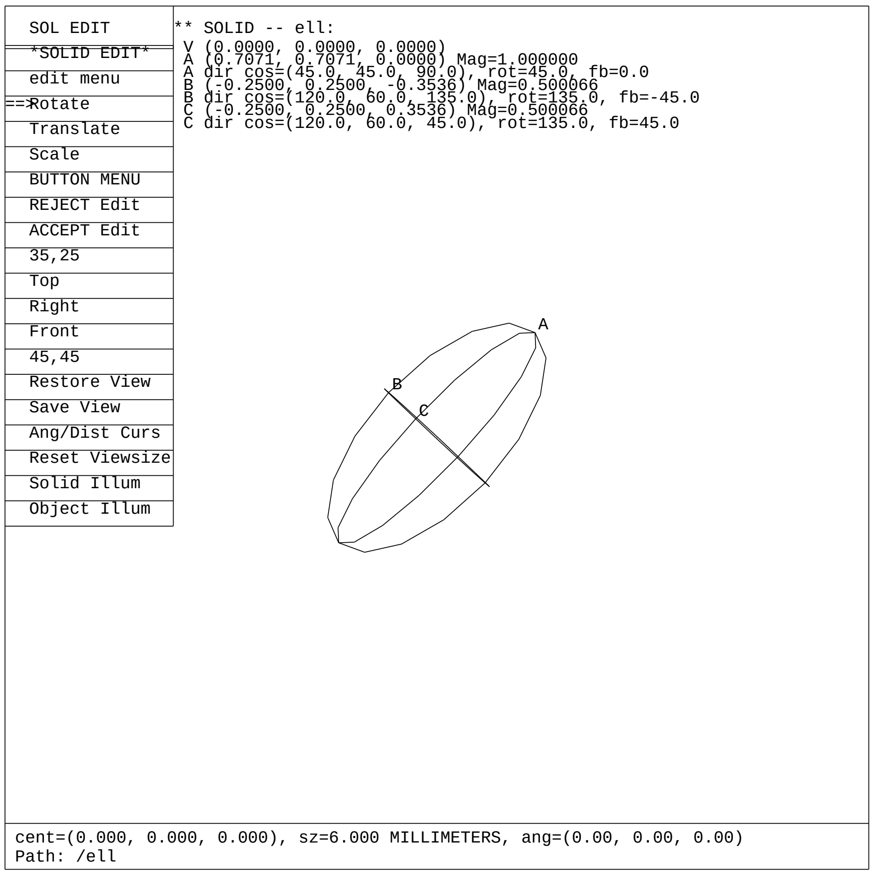

Figure 11.44: Solid Edit Rotation of 45 Degrees about Z

`mged>`

Figure 11.43 shows the rotation of the ellipsoid about its Y axis. The angle of rotation is counter-clockwise when viewed in the positive Y direction. The rotation is made from the original view, and the restoration of the view is not necessary.

`mged>` *Select the "Rotate" entry in the solid edit menu*
`mged>` *p 0 0 45*
`mged>`

Figure 11.44 shows the rotation of the ellipsoid about its Z axis. The axis of rotation is counter-clockwise when viewed in the positive Z direction. The command

`mged>` *p 0 0 0*
`mged>`

restores the original orientation of the solid.

11.4.3 Scale Operation

`mged>` *Select the "Scale" entry in the solid edit menu*
`mged>` *p .5*
`mged>`

Point V is not changed, but the distance from V to the surface of the ellipsoid is multiplied by 0.5, because the magnitude of the vectors are multiplied by the value of 0.5. See Figure 11.45.

Move the mouse to a position outside the menu area and above the X axis, and click the mouse. Notice that the size of the ellipsoid has grown, ie, the magnitude of the vectors have increased. Move the mouse to a position below the X axis, and click the mouse. Notice that the size of the ellipsoid has increased.

The command

`mged>` *p 1*
`mged>`

will restore the original scale.

NOTE: The use of the scale operation from the Solid Edit menu will result in the values of all the vectors being multiplied by the value of the scale. Use of the scale operaton from the Ellipsoid menu with a particular vector A, B, or C changes the magnitude of that vector to the value of the scale.

11.4.4 Scale A Command

`mged>` *Select the "edit menu" entry in the solid edit menu*
`mged>` *Select the "scale A" entry in the ellipsoid menu*
`mged>` *p 1.5*
`mged>`

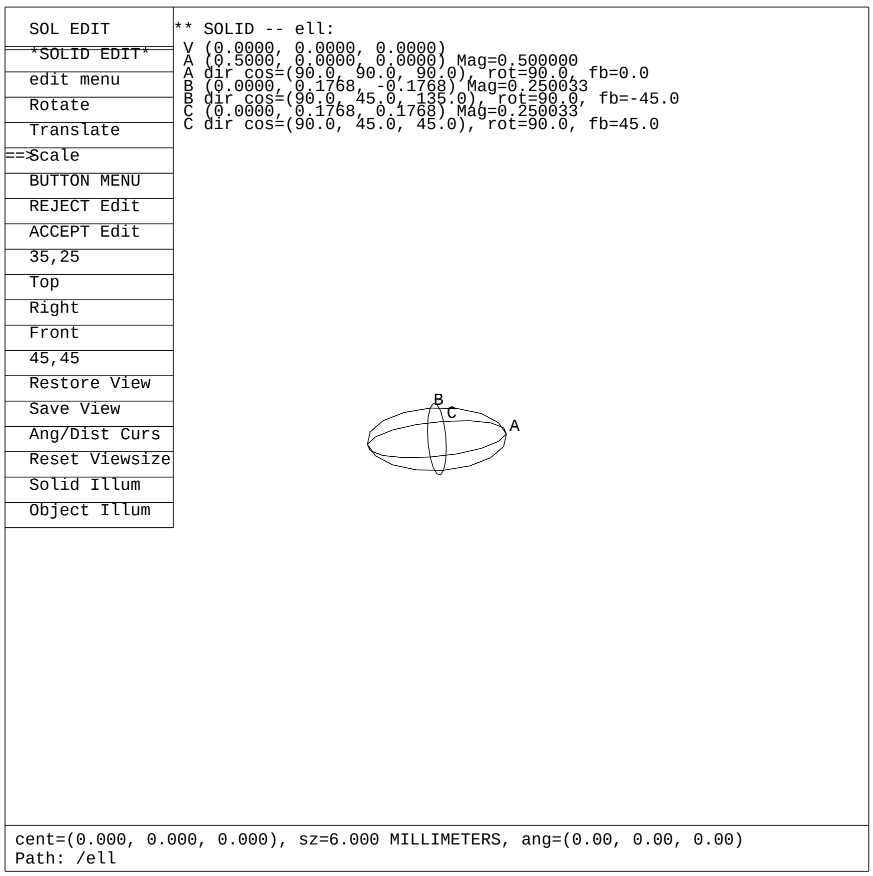

Figure 11.45: Ellipsoid Scale Decreased

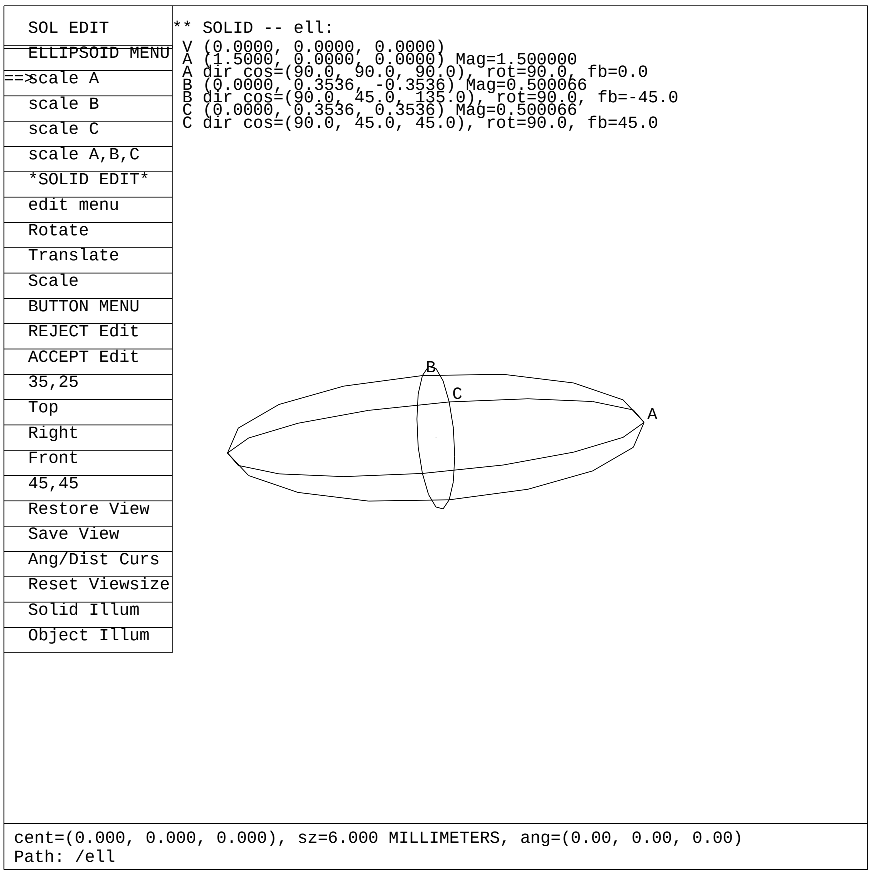

Figure 11.46: Ellipsoid Scale A Vector

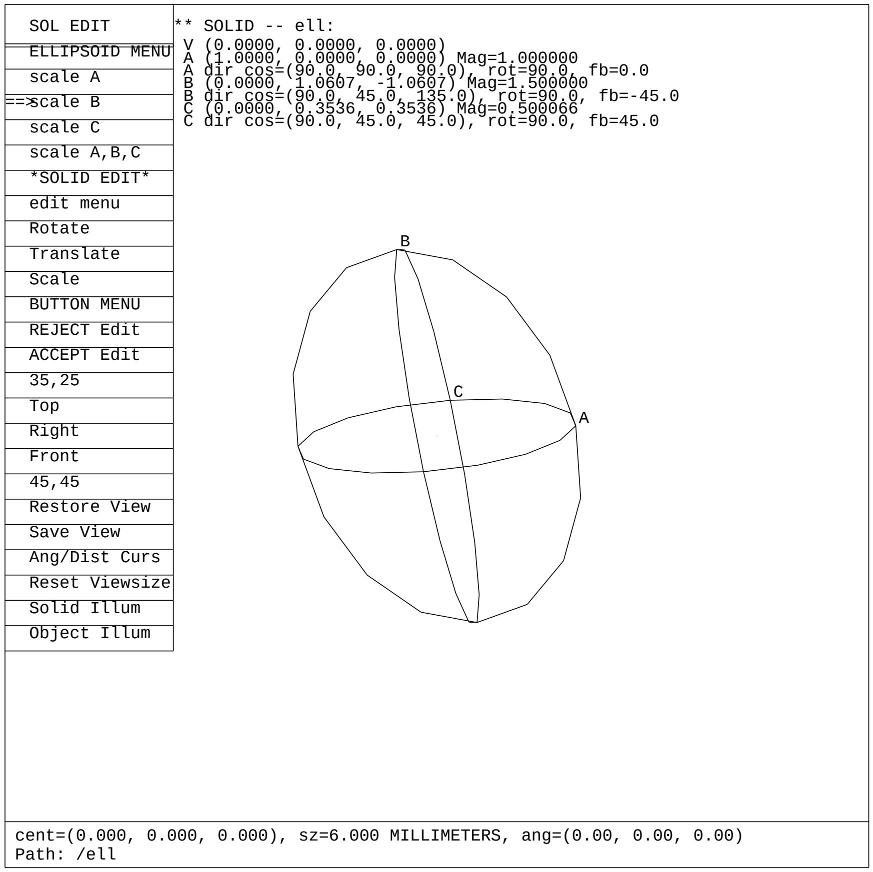

Figure 11.47: Ellipsoid Scale B Vector

The magnitude of the vector to point A is set equal to the value of p (eg 1.5). The components of the vector are (1.5, 0, 0) since the vector was parallel to the X axis. See Figure 11.46. The command

mged> *Select the "scale A" entry in the TGC menu*
mged> *p 1*
mged>

will restore the original shape.

11.4.5 Scale B Command

mged> *Select the "scale B" entry in the Ellipsoid menu*
mged> *p 1.5*
mged>

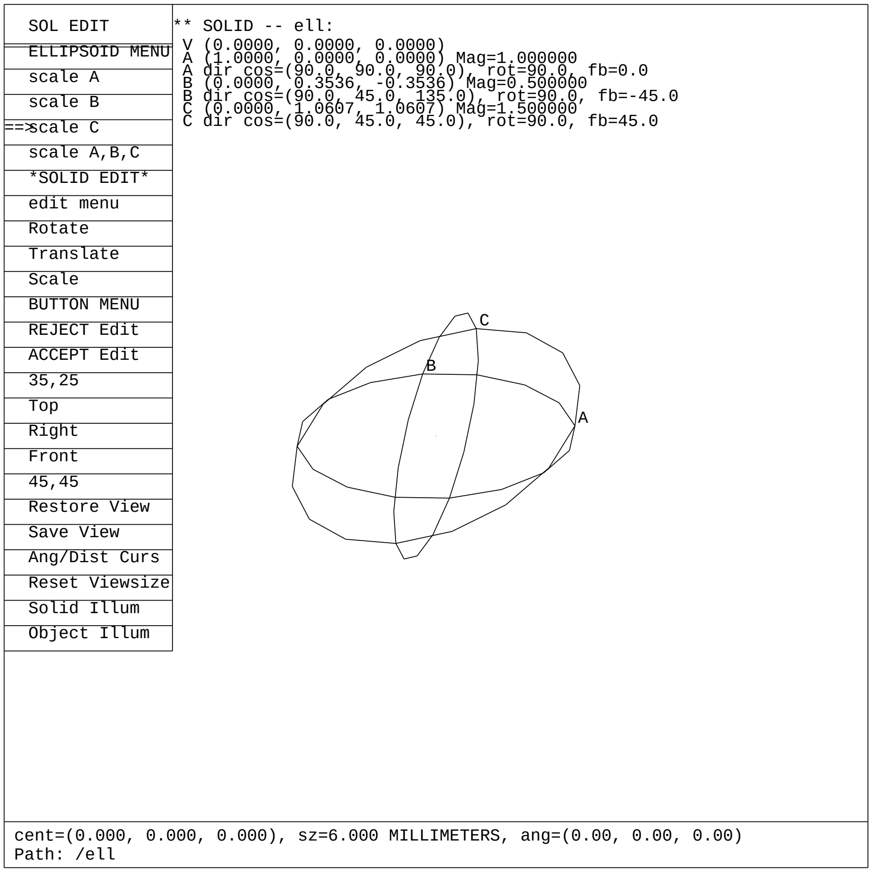

Figure 11.48: Ellipsoid Scale C Vector

The magnitude of the vector to point B is set equal to the value of p (eg 1.5). The coordinates of the vector are the product of p and the direction cosines of B. See Figure 11.47. The command

`mged>` *p 0.5*

`mged>`

will restore the original shape.

11.4.6 Scale C Command

`mged>` *Select the "scale C" entry in the Ellipsoid menu*

`mged>` *p 1.5*

`mged>`

The magnitude of the vector to point C is set equal to the value of p (ie, 1.5).

The coordinates of the vector are the product of p and the direction cosines of C. See Figure 11.48. The command

mged> *p 0.5*
mged>

will restore the original shape.

To return control to the VIEWING state, select the "REJECT Edit" item on the button menu, press the "reject" button on the button box, or enter the command *press reject* on the keyboard. Then, enter

mged> *d ell*
mged>

to drop the ellipsoid from view.

11.5 Solid Edit: Alter Torus

This tutorial illustrates the application of the SOL EDIT state to the torus solid.

mged> *size 6*
mged> *in tor tor*
Enter X, Y, Z of vertex: *0 0 0*
Enter X, Y, Z of normal vector: *0 1 0*
Enter radius 1: *1*
Enter radius 2: *0.2*
mged>

Figure 11.49 is the display of the torus solid in viewing state. Since the Z-axis is perpendicular to the viewing screen, a view of all sides cannot be seen.

mged> *Twist ROTY knob clockwise and restore*
mged> *Twist ROTX knob counter-clockwise and restore*
mged>

These actions generate a view of the torus that shows all sides, as shown in Figure 11.50.

mged> *Select the "Solid Illum" entry in the button menu*
mged> *Move the mouse out of the menu area*
mged> *Click the mouse to enter SOL EDIT state*
mged>

The torus is a ring whose cross-section is a circle. The distance from the vertex to the center of the cross-section is r1 and r2 is the radius of the circular cross section.

Let the points I and O be the intersection of the line x=-z and the torus. Then,

```
I = (-(r2-r1) cos 45, 0, (r2-r1) cos 45 )
O = (-(r2+r1) cos 45, 0, (r2+r1) cos 45 )
```

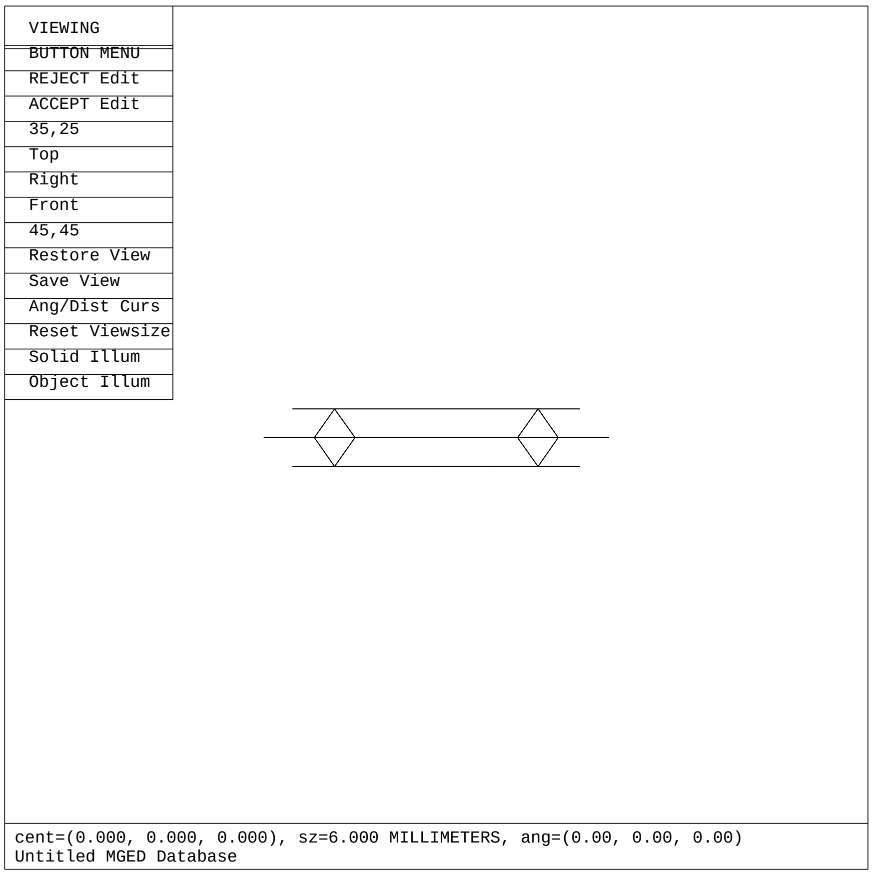

Figure 11.49: Top View of a Torus

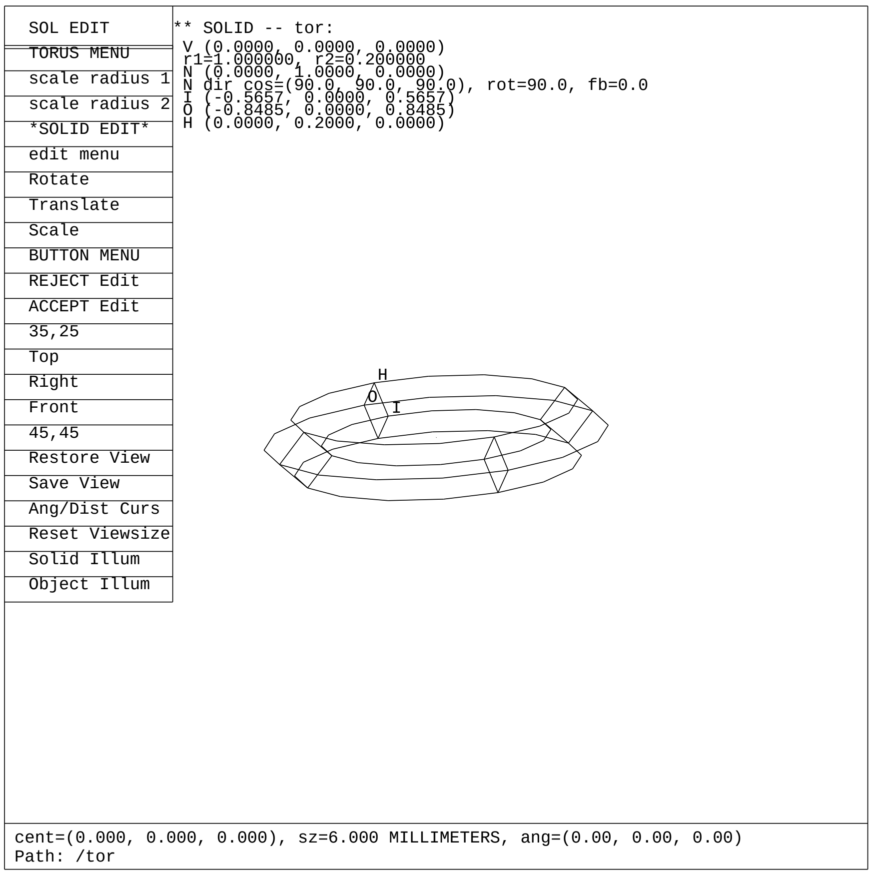

Figure 11.50: The Torus in Solid Edit State

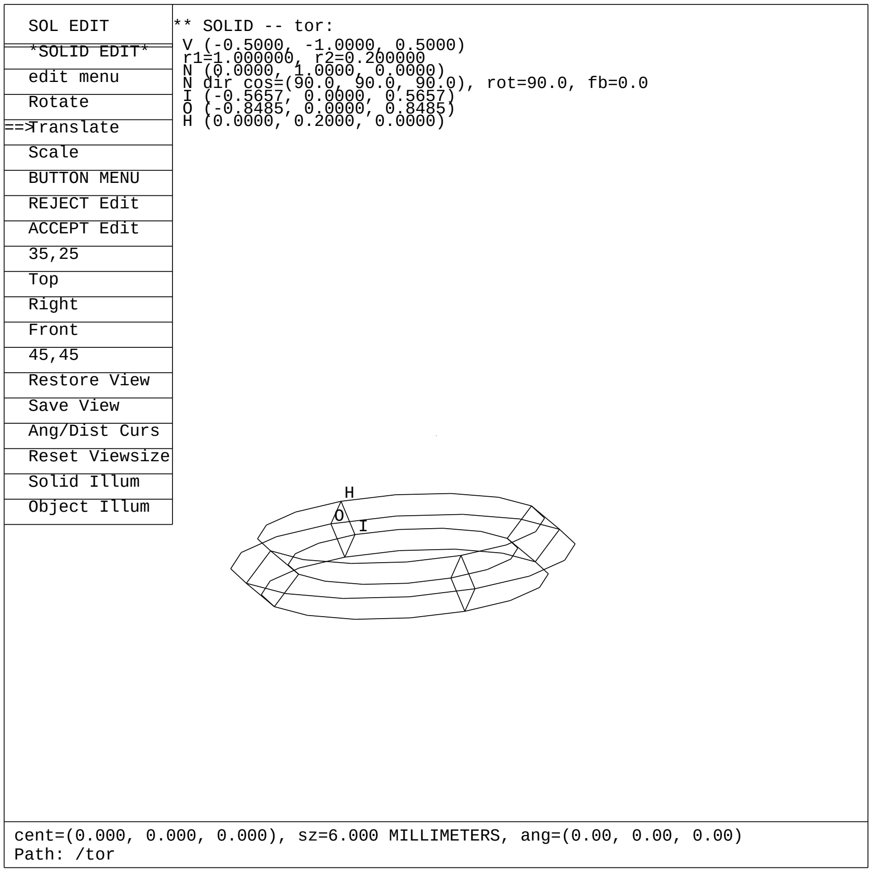

Figure 11.51: Translating a Torus

11.5.1 Translate Operation

mged> *Select the "Translate" entry in the solid edit menu*
mged> *p -.5 -1 .5*
mged>

The vertex V of the torus is moved to (-.5, -1, .5). See figure 11.51. The coordinates of the other points remain the same, relative to the vertex. The command

mged> *p 0 0 0*
mged>

will restore the original position.

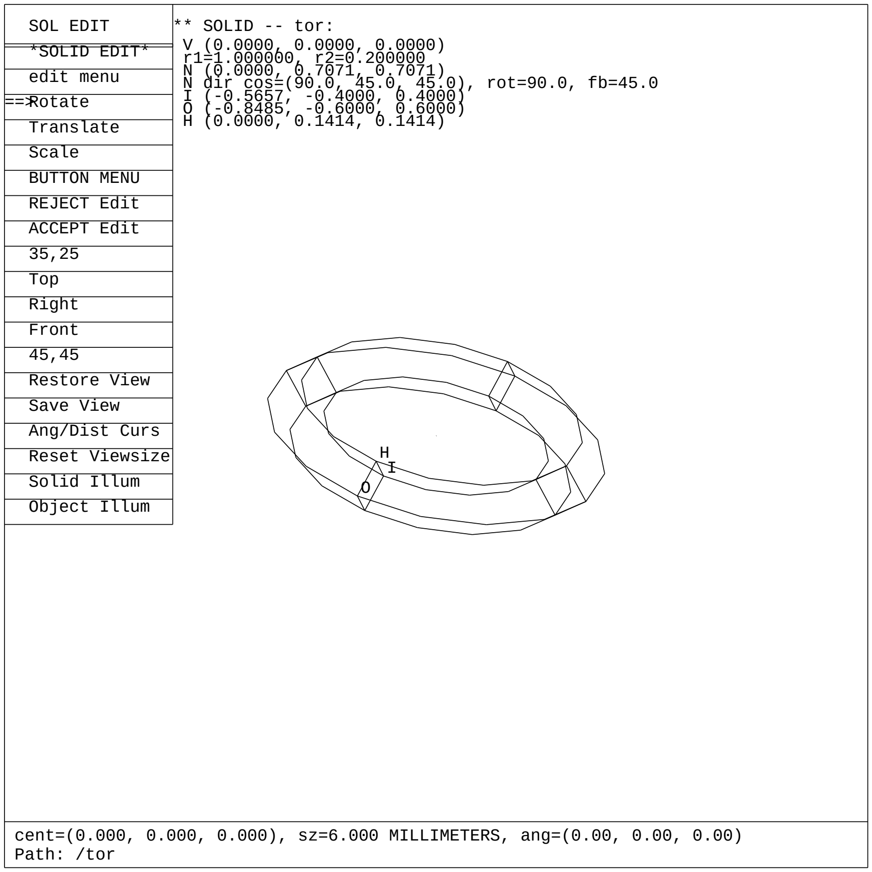

Figure 11.52: Torus Solid Edit Rotation about X

11.5.2 Rotate Operation

mged> *Select the "Rotate" entry in the solid edit menu*
mged> *p 45 0 0*
mged>

The torus is rotated 45 degrees counter-clockwise about the positive X axis. The coordinates of the points I and H are transformed using the following matrix:

```
[x'] [1    0       0   ] [x]
[y']=[0  .7071 -.7071] [y]
[z'] [0  .7071  .7071] [z]
```

See Figure 11.52.

mged> *p 0 45 0*

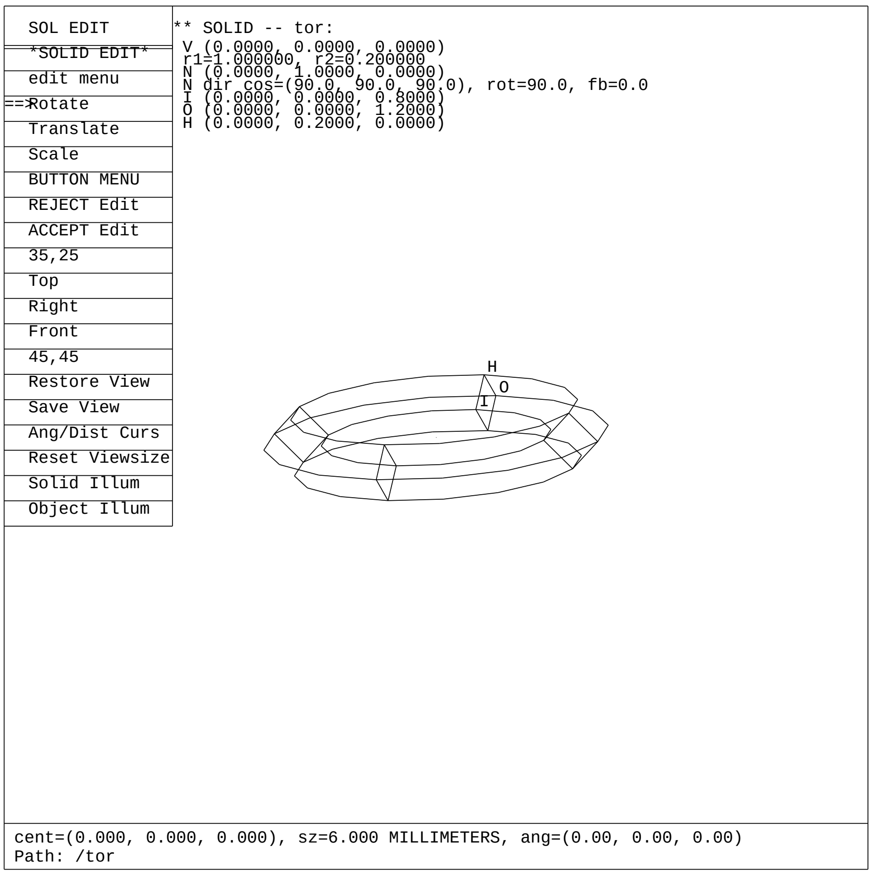

Figure 11.53: Torus Solid Edit Rotation about Y

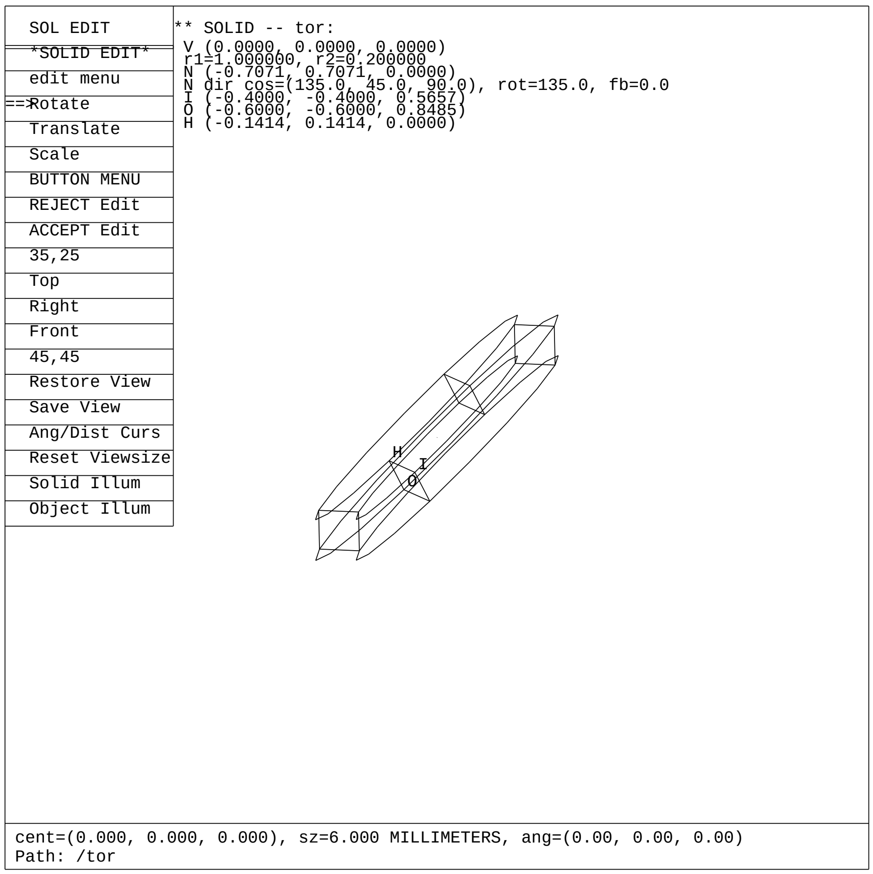

Figure 11.54: Torus Solid Edit Rotation about Z

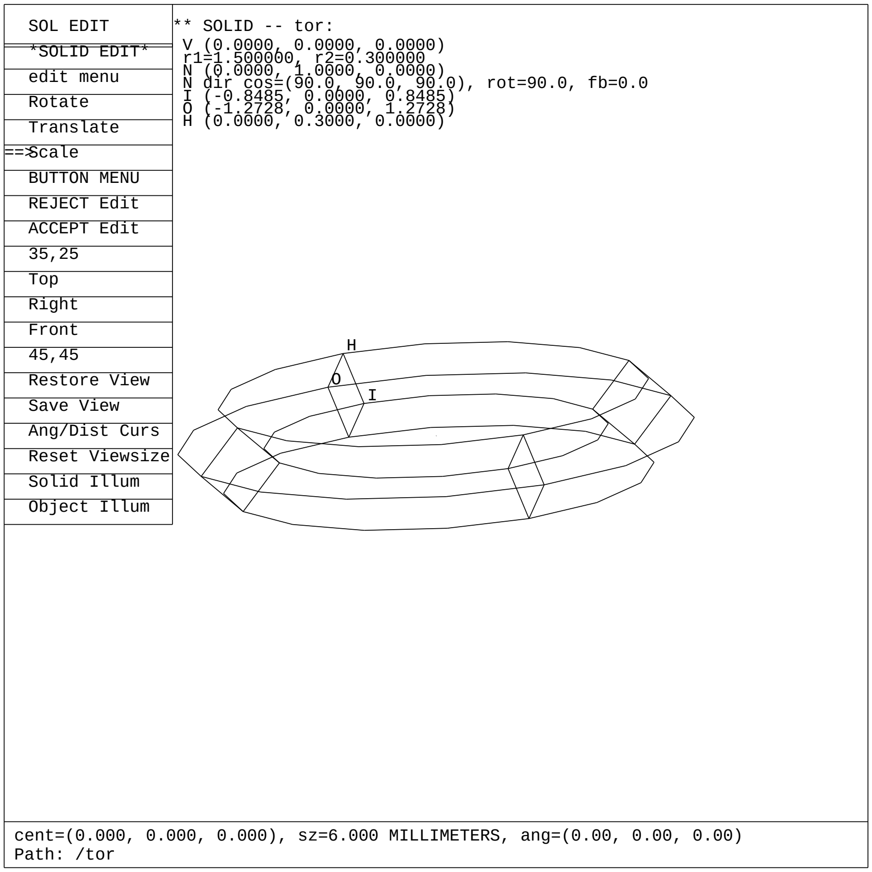

Figure 11.55: Torus Scale Increased

mged>

The torus is rotated 45 degrees counter-clockwise about the positive Y axis. See Figure 11.53.

mged> *p 0 0 45*

mged>

The torus is rotated 45 degrees counter-clockwise about the positive Z axis. See Figure 11.54. The original orientation is restored by entering

mged> *p 0 0 0*

mged>

11.5.3 Scale Operation

mged> *Select the "Scale" entry in the solid edit menu*

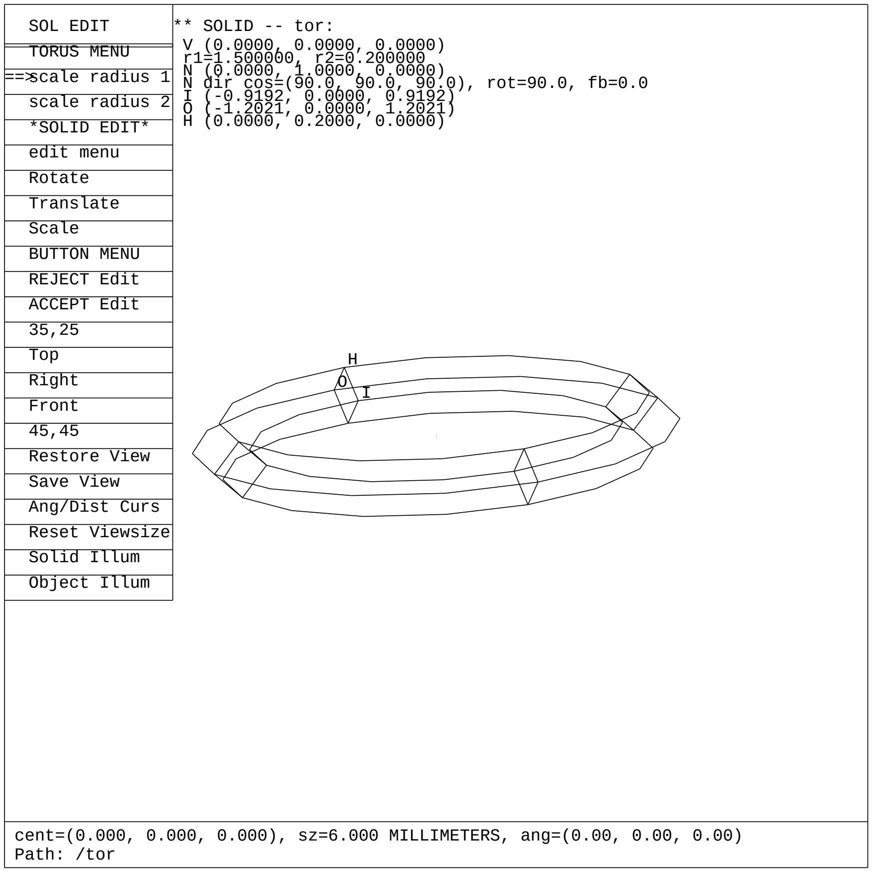

Figure 11.56: Scale Torus Radius 1

`mged>` *p 1.5*
`mged>`

The vertex remains the same and all distances from the vertex are multiplied by 1.5, the value entered with p. See Figure 11.55. To return to the original scale, enter

`mged>` *p 1*
`mged>`

11.5.4 Scale Radius 1 Command

`mged>` *Select the "edit menu" entry in the solid edit menu*
`mged>` *Select the "scale radius 1" entry in the TORUS menu*
`mged>` *p 1.5*

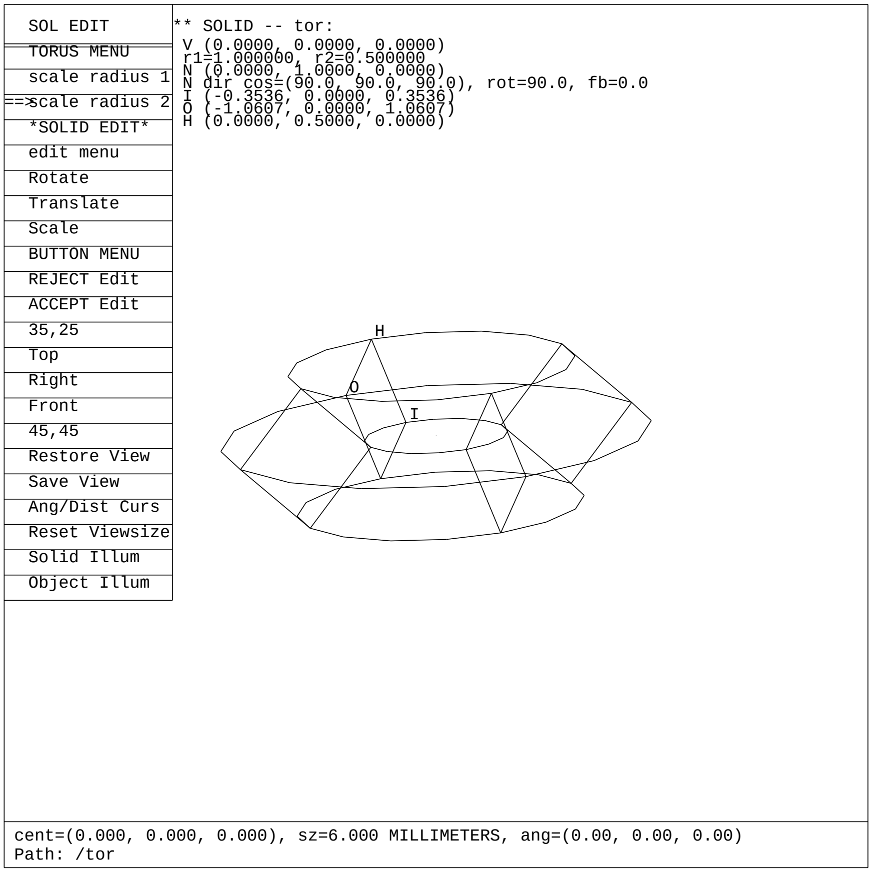

Figure 11.57: Scale Torus Radius 2

`mged>`

The distance from the vertex to the center of the cross-section of the ring is set equal to the values given with *p*, eg, 1.5. See Figure 11.56. The original scale can be restored with

`mged>` *p 1*

`mged>`

11.5.5 Scale Radius 2 Command

`mged>` *Select the "edit menu" entry in the solid edit menu*

`mged>` *Select the "scale radius 2" entry in the TORUS menu*

`mged>` *p 0.5*

`mged>`

The distance from the center of the cross-section of the ring is set equal to the value given with *p*, eg 0.5. This value must remain less than the value for r1. See Figure 11.57. The command

`mged>` *p 0.2*
`mged>`

will restore the original shape.

To return control to the VIEWING state, select the "REJECT Edit" item on the button menu, press the "reject" button on the button box, or enter the command *press reject* on the keyboard. Then, enter

`mged>` *d tor*
`mged>`

to drop the torus from view.

Chapter 12

TUTORIALS ON OBJECT EDITING

12.1 Object Editing a Six-Sided Polyhedron

This tutorial illustrates the application of the **OBJ EDIT** state to the "arb8" primitive. The editing of the arb8 illustrates the absolute movement of the points.

`mged>` *e arb8*
`vectorized in 0 seconds`
`mged>` *size 8*
`mged>`
Select the **BUTTON MENU** *if not already displayed.*
Select the **Object Illum** *menu entry.*
Move the mouse away from the menu area and select twice.

These operations select an object for editing. Control is passed through the **OBJ PICK** and **OBJ PATH** states to the **OBJ EDIT** state. The display should look similar to Figure 12.1.

12.1.1 Scale Operation

`mged>` *scale 0.5*
`mged>`

As always, the selected operation operates with respect to the key vertex – point 1 remains the same, and the distances from point 1 to the other points are multiplied by the scale factor. See Figure 12.2. What will happen if another **scale 0.5** command is given? The **scale** operator is an absolute operator. It

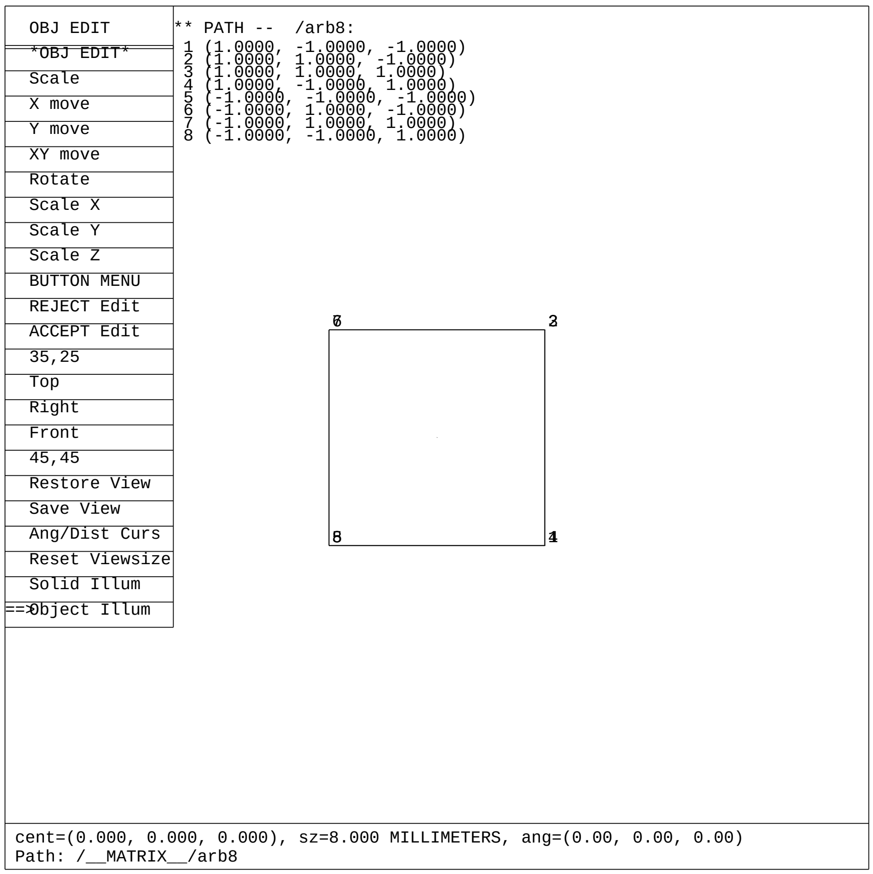

Figure 12.1: "arb8" Object Edit; Top View

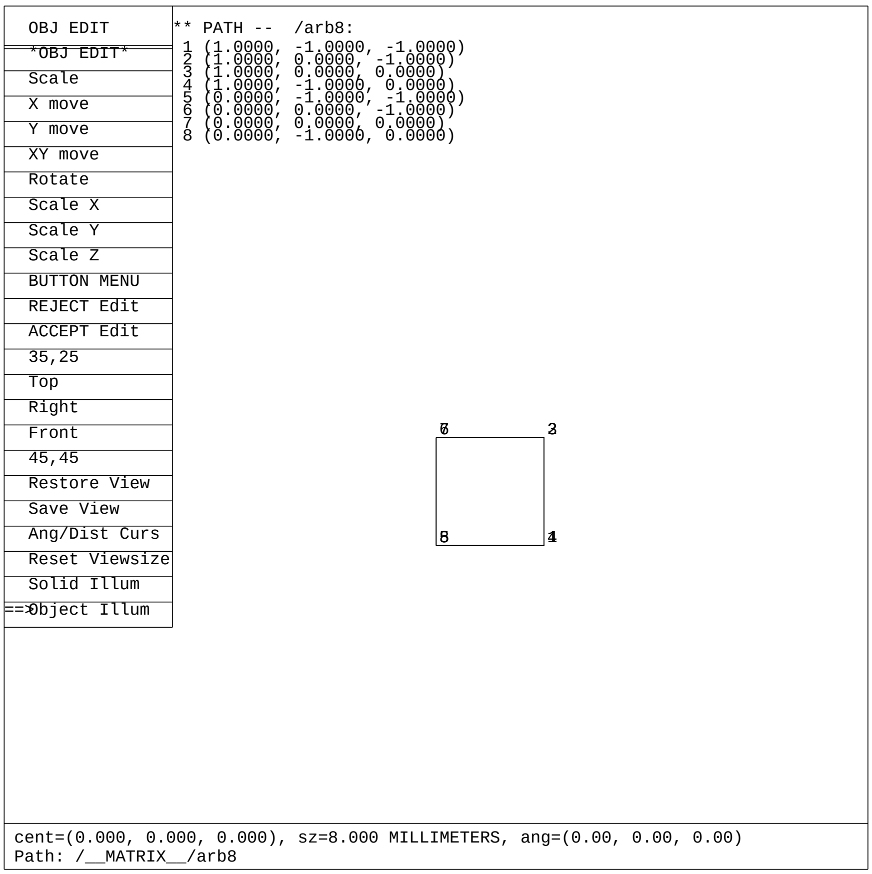

Figure 12.2: "arb8" Object Edit; Scaled by 0.5

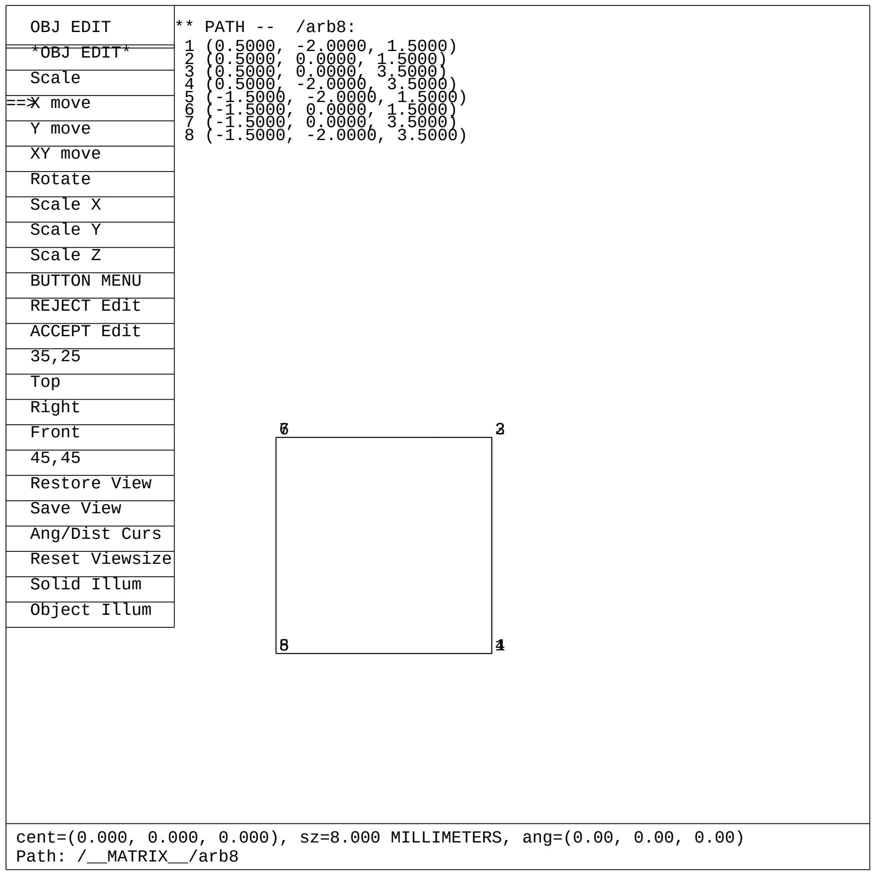

Figure 12.3: "arb8" Object Edit; Translated to (0.5, -2, 1.5)

sets the scale factor associated with a particular transformation matrix. It does not multiply the current transformation matrix scale factor by the new scale factor.

12.1.2 X, Y, and XY Move Operation

`mged>` *scale 1*
`mged>` *translate .5 -2 1.5*
`mged>`
Select the **X move** *menu entry.*

The first two commands undo the effects of the previous scale operation, and translate the key point to (0.5, -2, 1.5). The coordinates of the other points are

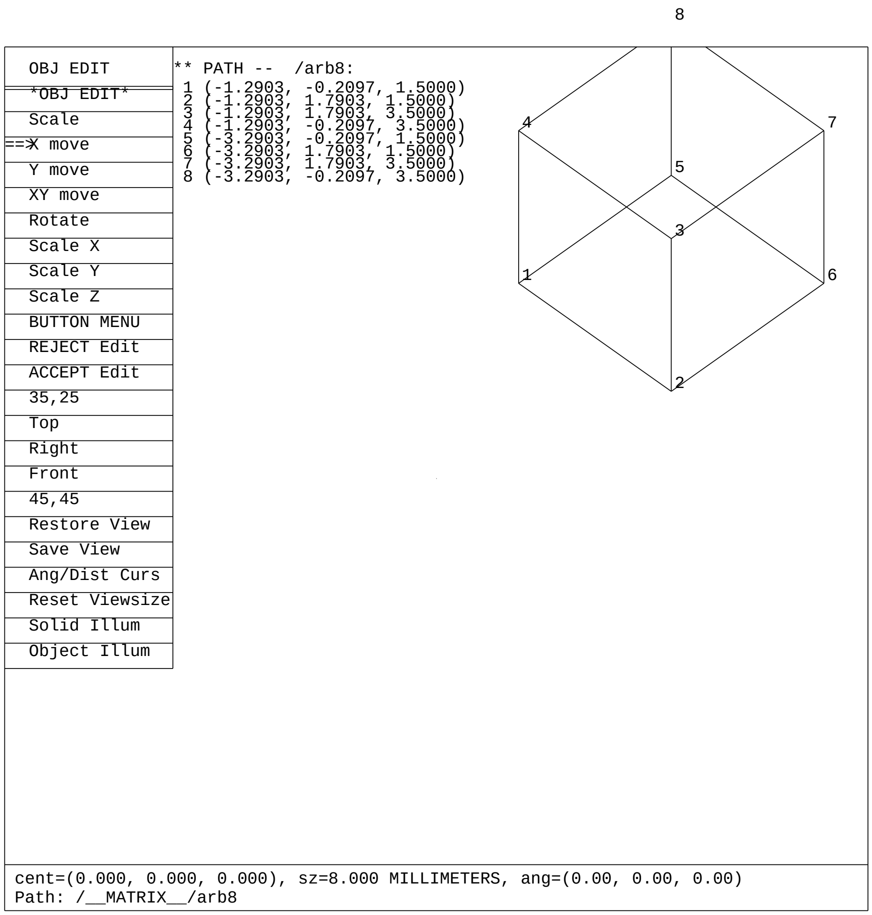

Figure 12.4: XY Move

changed accordingly; preserving their distances relative to point 1 (see Figure 12.3). The last operation above placed MGED into a state where the key point of the object will track the X component of successive selects, but not the Y component. Note that on some displays point 1 may not be directly visible. It is actually behind point 4. Watch the area listing the information concerning "arb8" as selections are made.

Do several selects while moving the cursor slowly in a circle.

Observe that only the X axis information changes. Similarly, the **Y move** tracks only changes in the Y axis, and **XY move** tracks changes in both axes.

`mged>` *press 45,45*

`mged>`

Do several more selects while moving the cursor slowly in a circle.

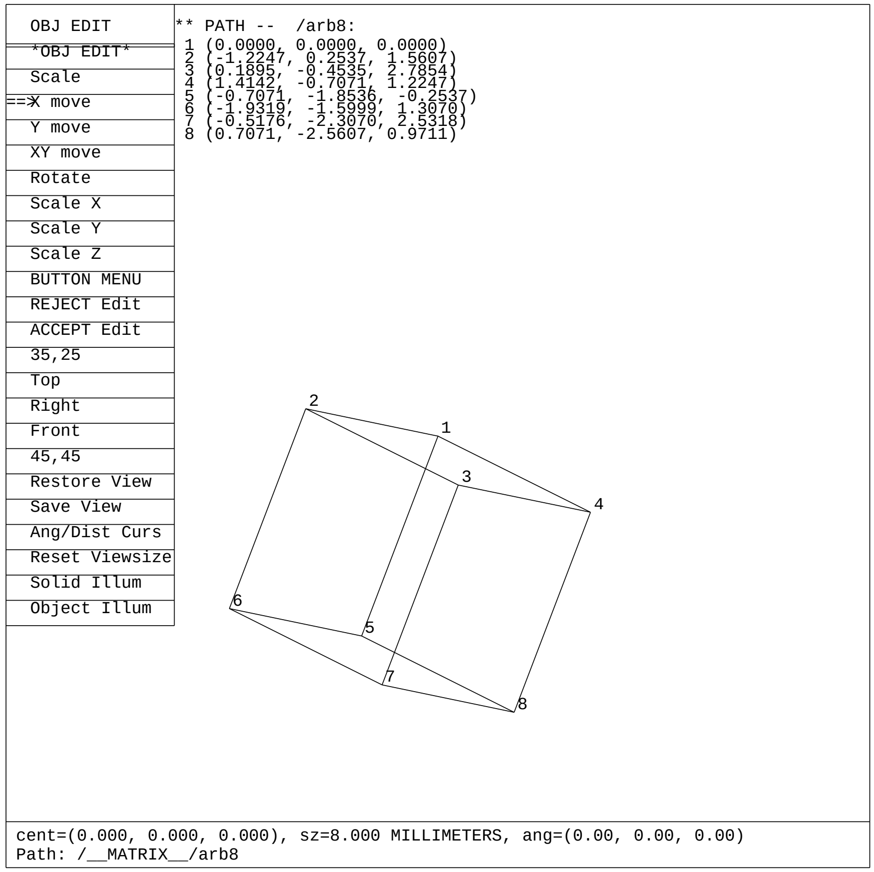

Figure 12.5: "arb8" Rotated by (30, 45, 60)

This time, moving point 1 modifies the model x and y axes (observe the changes in the list of vertices). If **Y move** is selected, all three sets of axes are modified. These operators work in screen space, not model space, so using them with an oblique view moves the model in more than one axis (Figure 12.4 for example)..

12.1.3 Rotate Operation

```
mged> press reset
mged> translate 0 0 0
mged> rotobj 30 45 60
mged>
```

The primitive is rotated 60, 45, and 30 about the z, y, and x axes in that order as shown in Figure 12.5. Note that the coordinates of the points are changed when scaling, translation, and rotation are performed.

`mged>` *press reset*
`mged>` *d arb8*
`mged>`

If an object which is being edited is deleted from view, MGED transitions back into the **VIEWING** state.

12.2 Object Editing an Ellipsoid

`mged>` *e ellg*
`mged>`
Select the **45,45** *button menu entry.*
Select the **Object Illum** *button menu entry.*
Move the cursor outside the menu area and select twice.

Control is again passed from the **OBJ PICK** state through the **OBJ PATH** state to the **OBJ EDIT** state. When editing a cylinder, ellipsoid, or torus, the coordinates of the primitives are set relative to the center of the primitive, and are not changed by any translation of the primitive. Figure 12.6 represents the display.

12.2.1 Scale Operation

`mged>` *scale 2*
`mged>`

The magnitudes of the vectors from the center V to the points A, B, and C are multiplied by the scale factor 2. The location of the center V is unchanged. See Figure 12.7.

12.2.2 Move Operations

`mged>` *scale 1*
`mged>`
Select the **XY move** *buttom menu entry.*
Move the cursor to some location away from the menu area and select.

The as with the arb8 in the previous section, the ellipsoid is moved in the model space plane parallel to the plane of the screen so the key point (vertex V)

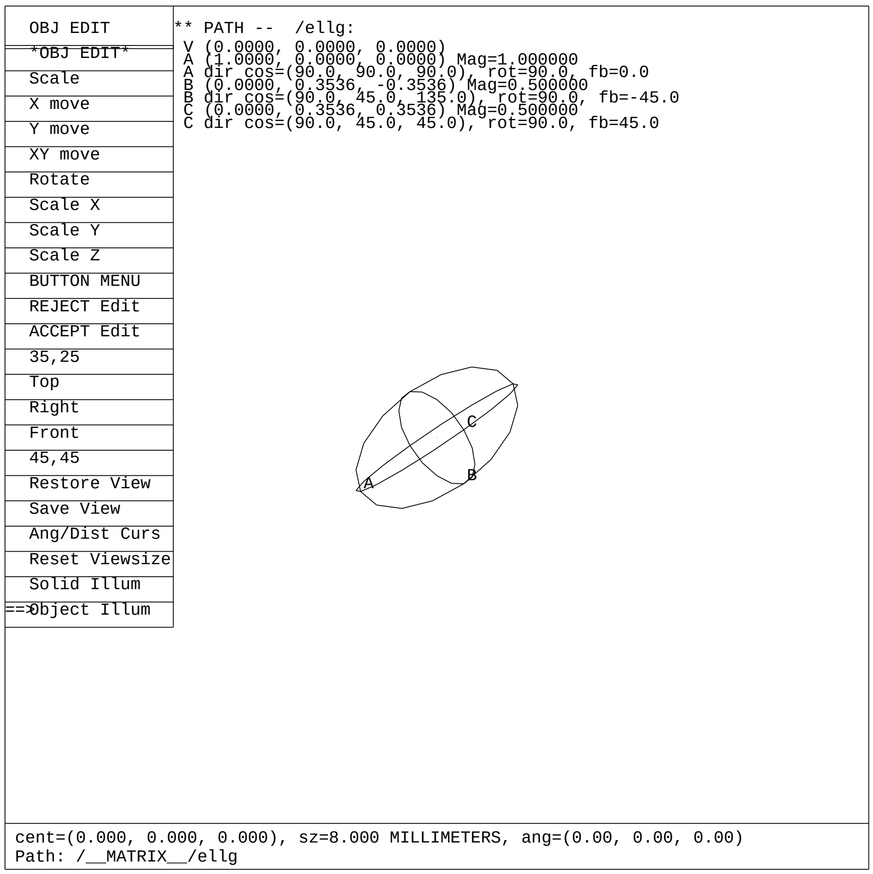

Figure 12.6: Object Edit; Ellipse viewed from 45,45 preset

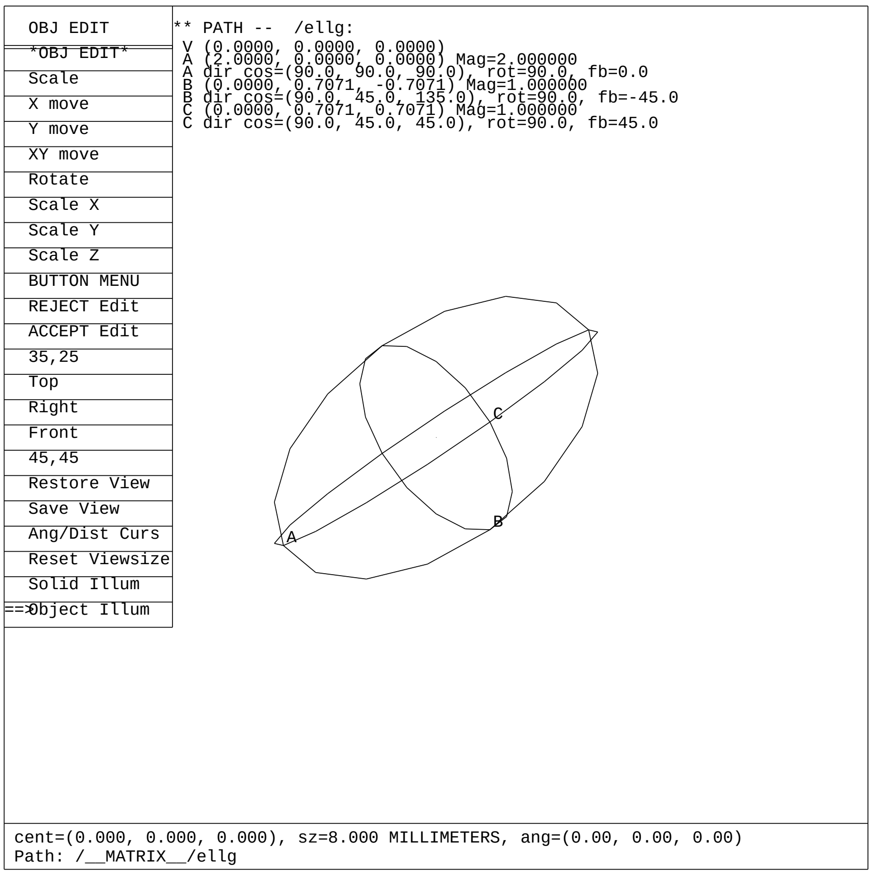

Figure 12.7: Object Edit; Ellipse scaled up by 2

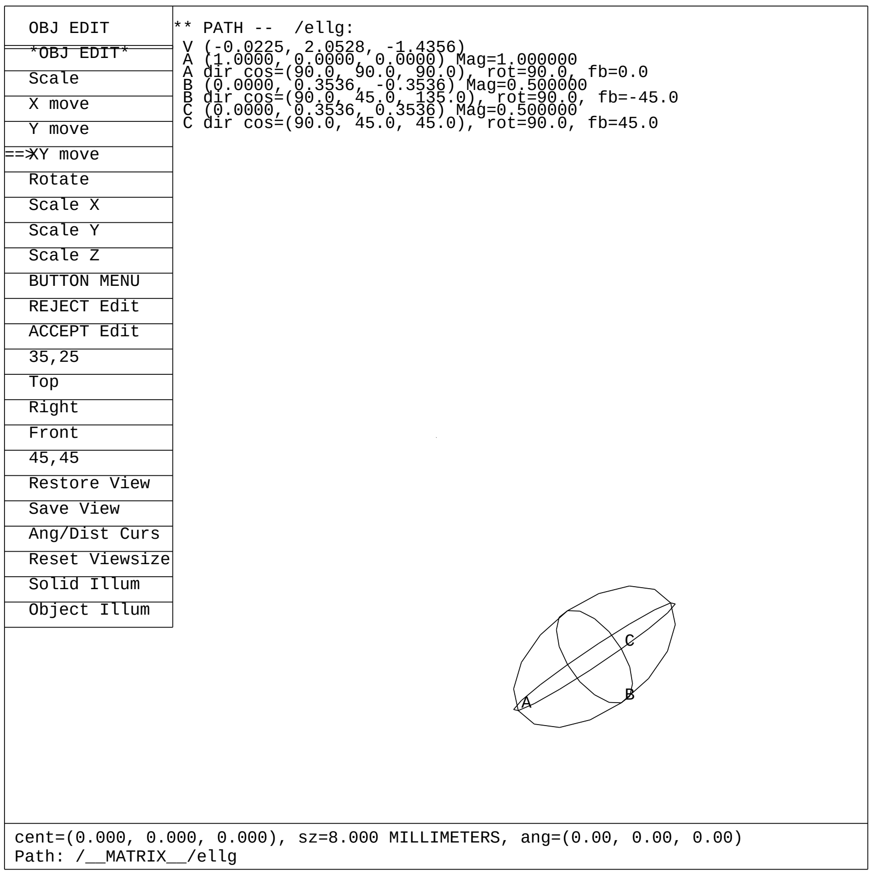

Figure 12.8: Object Edit; Ellipse Translated

is "placed at" the point corresponding to the cursor location. Note that there is no explicit control over the location of the point with respect to screen Z (depth in the viewing cube); the selected point has the same screen Z as the original point. The screen should look similar to Figure 12.8.

```
mged> d ellg
mged>
```

12.3 Object Path and Object Edit

This section illustrates the use of the **OBJ PATH** state to select the number of objects that are affected by one edit command. In the previous sections the user was shown how to manipulate only one primitive. A group of primitives may also be edited as a single entity. This section shows how an entire group may be edited without addressing each individual primitive.

MGED generally has several ways to achieve a particular result. In this section, keyboard commands are used instead of the control buttons and the display menu. The creation and saving of special primitives is illustrated.

In the database, the original primitives are centered around the origin. Copies of these primitives will be made, translated away from the origin and saved for future editing.

12.3.1 Organize the Primitives and Groups

```
mged> cp arb8 arb8.s
mged> cp ellg ellg.s
mged> cp tgc tgc.s
mged> cp tor tor.s
mged>
```

Figure 12.9 shows the four primitives; arb8.s, ellg.s, tgc.s, and tor.s. One convention frequently used by experienced modelers is to tack an identifying suffix on the names of the various primitives and objects. Often, a ".s" suffix denotes a *solid*, a ".r" denotes a *region* and a ".g" denotes a *group* (note that this is a different ".g" from the ".g" suffix used with filenames.).

```
mged> size 16
mged> sed arb8.s
mged> press sxy
mged> p 3 -3 1
mged> press accept
mged>
```

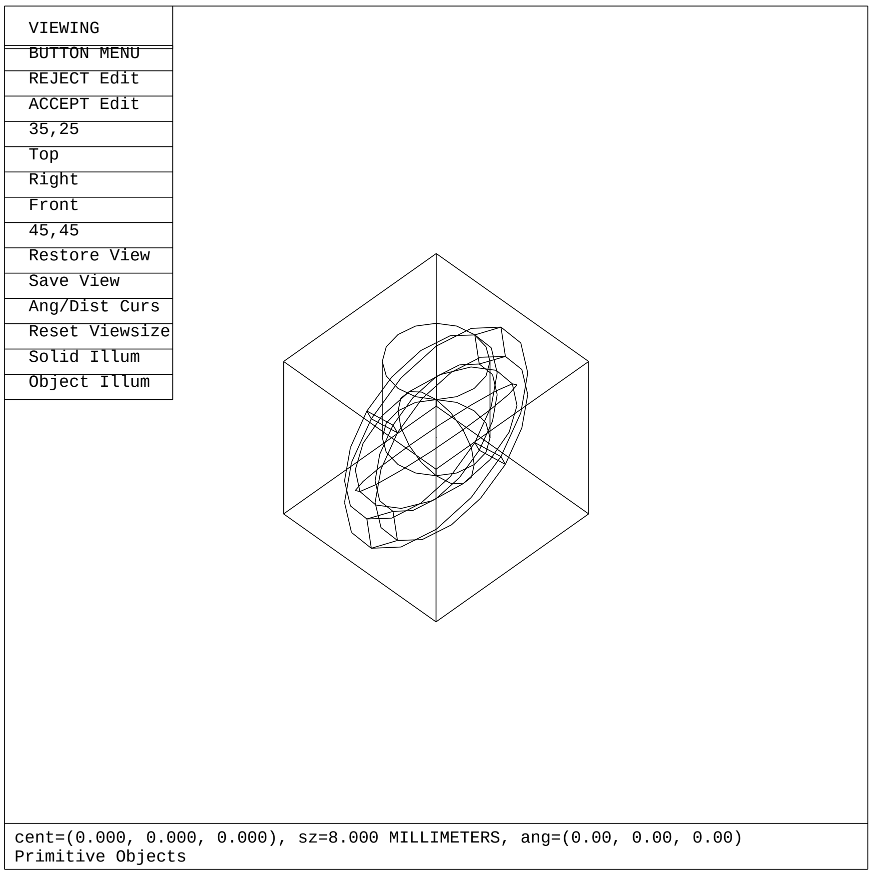

Figure 12.9: Stacked Primatives

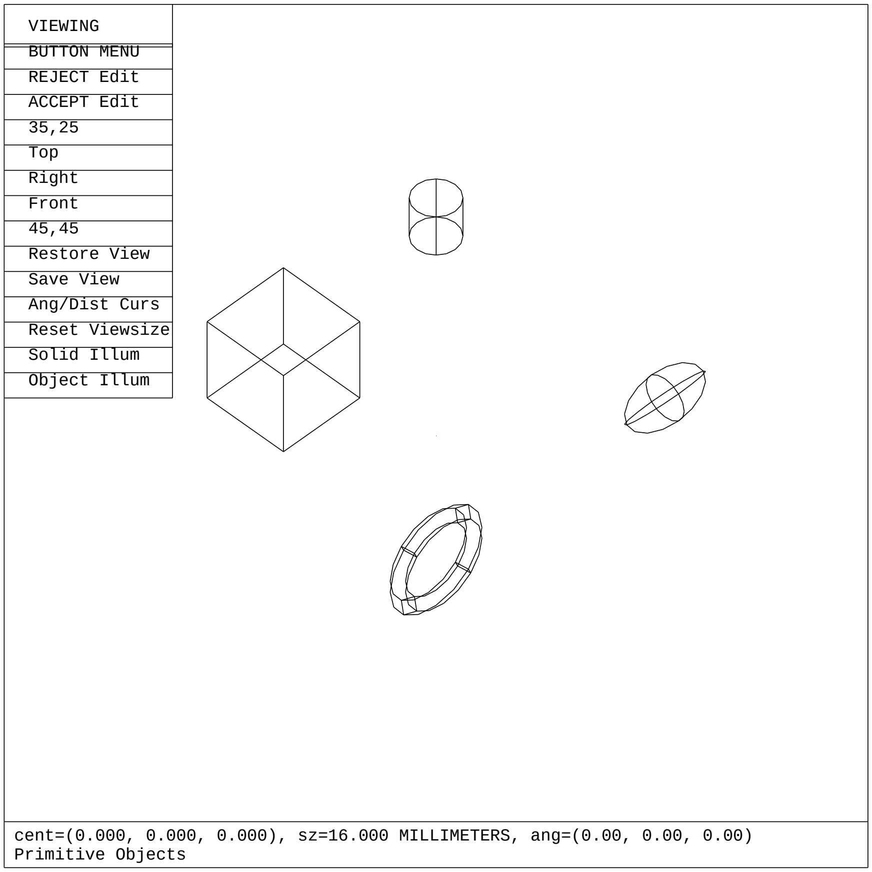

Figure 12.10: Primatives After Translation

Several things happened in the above sequence. The net result is that the solid arb8.s was "unstacked" using a Solid Edit so it is more visible. The same sequence of operations will be performed with the other objects to move them to other locations.

mged> *sed ellg.s*
mged> *press sxy*
mged> *p -3 3 1*
mged> *press accept*
mged> *sed tgc.s*
mged> *press sxy*
mged> *p -3 -3 1*
mged> *press accept*
mged> *sed tor.s*

```
mged> press sxy
mged> p 3 3 1
mged> press accept
mged>
```

The screen should now look like Figure 12.10. The next step is to group the primitives

```
mged> g a.g tgc.s arb8.s
mged> g b.g ellg.s tor.s
mged> g c.g a.g b.g
mged> B c.g
vectorized in 0 sec
mged> tree c.g
| c.g_____________| a.g_____________| tgc.s
                                    | arb8.s
                  | b.g_____________| ellg.s
                                    | tor.s
mged>
```

The *group* operator (**g**) generates an object, named by the first argument, which is the union of all objects named in succeeding arguments. Therefore, the object "a.g" is composed of the union of "tgc.s" and "arb8.s". Likewise, the object "c.g" is the union of "a.g" and "b.g". The next command in the above sequence is called the *blast* command. It is effectively a *zap* (**Z**) followed by an *edit* (**e**).

The final command is the *tree* command. It is intended to give the user some idea of the hierarchical structure of an object. It presents a tree laid on it's side. The root is at the left, and the leaves are at the right. A vertical bar denotes a connection at a given level, with the proviso that a vertical bar having a line of underscores coming in from the left represents the start of a particular subtree when read from top down ("ellg.s" and "arb8.s" do not have a common parent).

```
mged> press oill
mged>
```

Move the cursor up and down the screen until the primitive "tor.s" is illuminated, then select

Selecting the solid "tor.s" transitions MGED into the **OBJ PATH** state, and establishes "tor.s" as the reference solid for any future editing operations. Note that the name "tor.s" is shown in the upper left corner of the display, and on the second status line at the bottom of the display (Figure 12.11).

The **OBJ PATH** state has little meaning unless there is more than one path or group in the display. One of the following paths may be selected:

c.g/b.g/_MATRIX_/tor.s

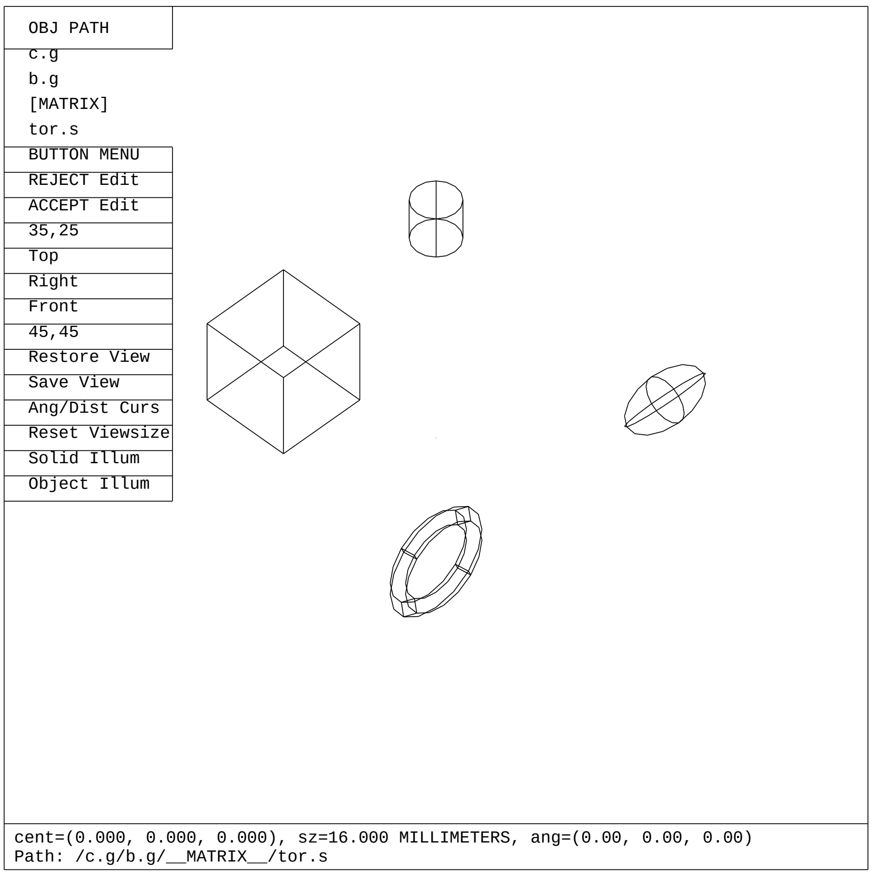

Figure 12.11: Object Path With "tor.s" as Reference Solid

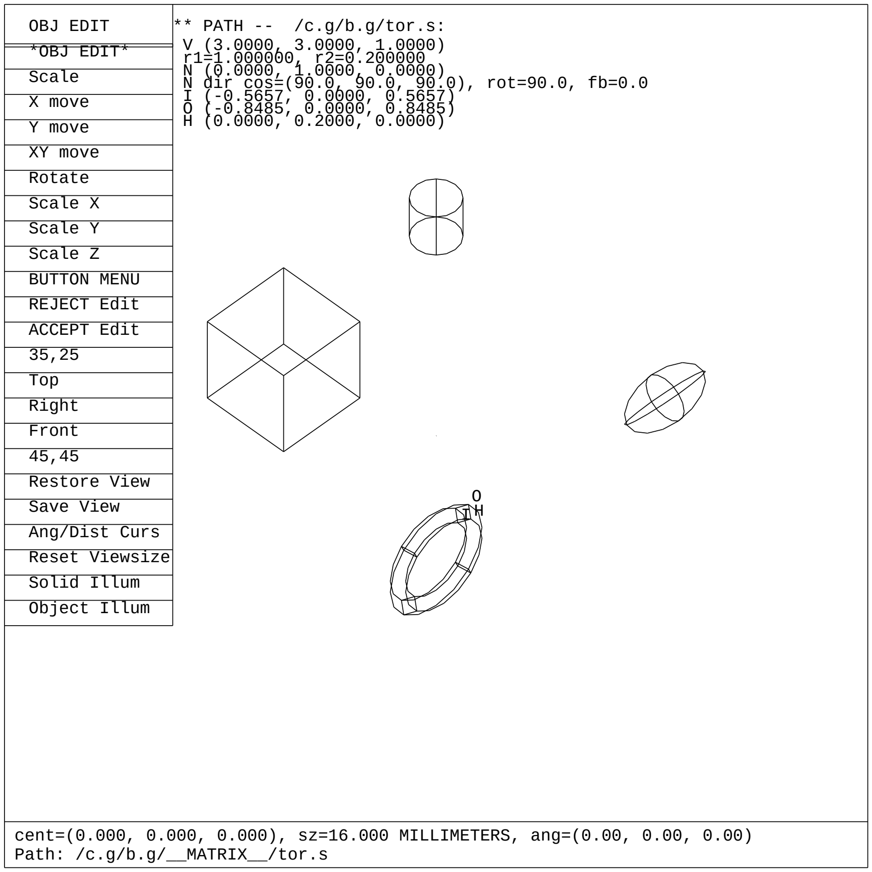

Figure 12.12: Object Edit With "tor.s" as Reference Solid

c.g/_MATRIX_/b.g/tor.s

MATRIX/c.g/b.g/tor.s

Although the torus primitive has been selected as the reference solid, the position of **_MATRIX_** determines the extent of the effects of the edit. The first choice affects only the torus. The second choice affects everything under the group "b.g" (the torus and ellipsoid). The third choice affects all of the primitives. Remember though, that in all cases, what is being edited is the Homogeneous Transformation Matrix (thought of as the arc connecting objects), not the underlying solid.

12.3.2 Editing One Primitive

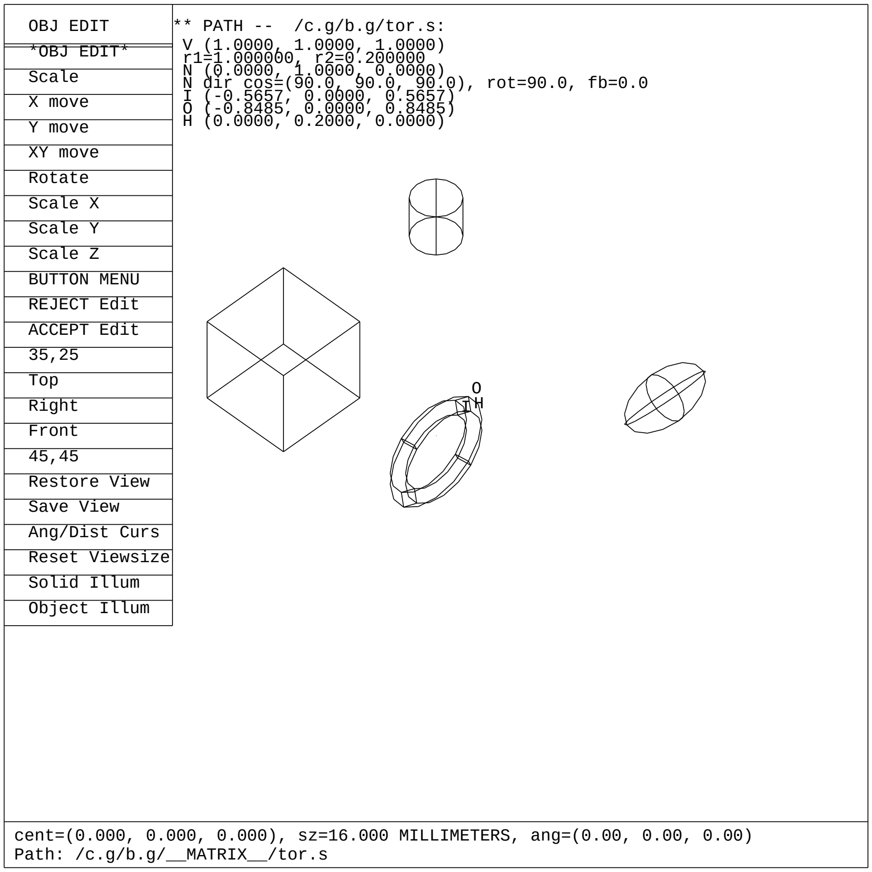

Figure 12.13: Object Edit Affecting Torus Only

Move the cursor up and down until `/c.g/b.g/_MATRIX_/tor.s` *appears in the bottom status line, then select.*

Figure 12.12 is the new display. Note that the torus is illuminated. The **OBJ EDIT** state has been reached.

`mged>` *translate 1 1 1*
`mged>`

The key point of the torus is moved to 1, 1, 1. The other primitives are not moved. See Figure 12.13.

`mged>` *press reject*
`mged>`

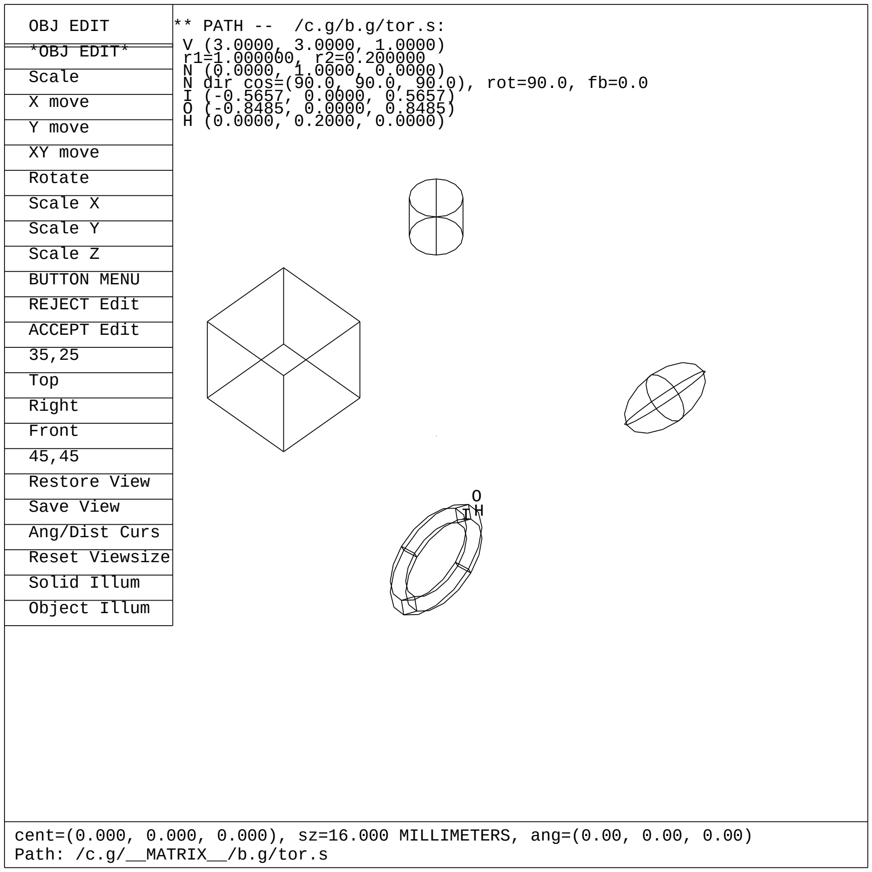

Figure 12.14: Torus and Ellipsoid Selected for Object Edit

12.3.3 Editing a Group of Two Primitives

`mged>` *press oill*
`mged>`
Move the cursor up and down the screen until the primitive "tor.s" is illuminated, then select.
Move the cursor up and down until `/c.g/_MATRIX_/b.g/tor.s` *appears in the bottom status line, then select.*

Control has again been passed to the **OBJ EDIT** state. Notice that both the torus and ellipsoid are illuminated (Figure 12.14. Although only the parameters for the torus will be changed in the display, the ellipsoid in group "b.g" will be affected by the edit.

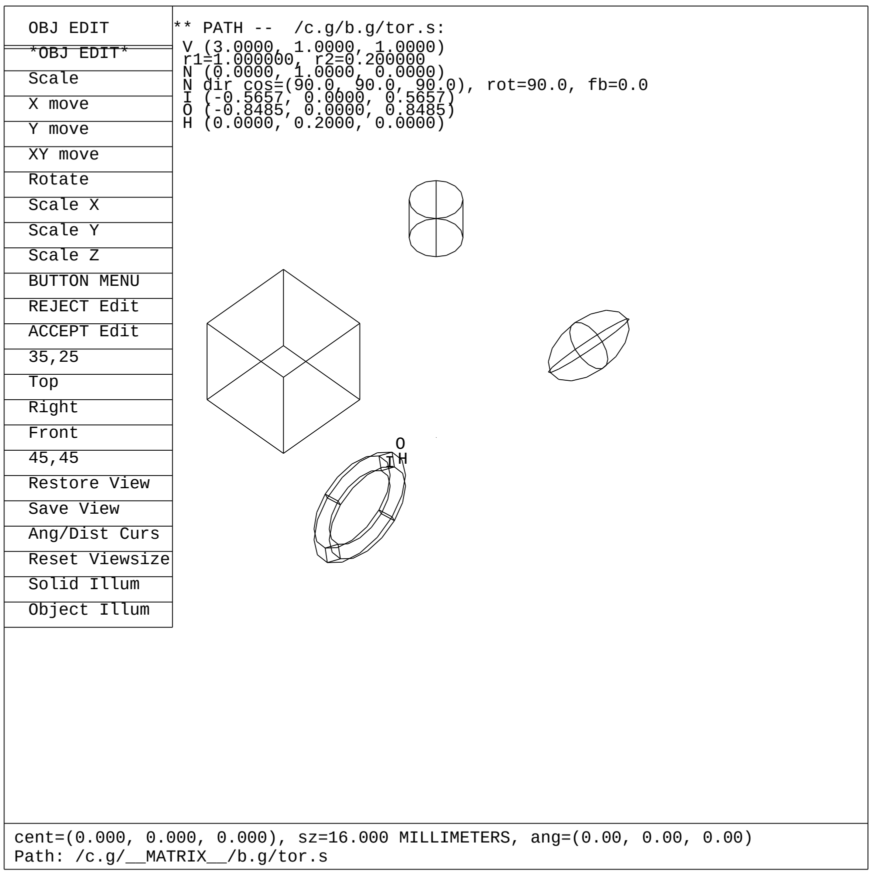

Figure 12.15: Torus and Ellipsoid Translated by (3, 1, 1)

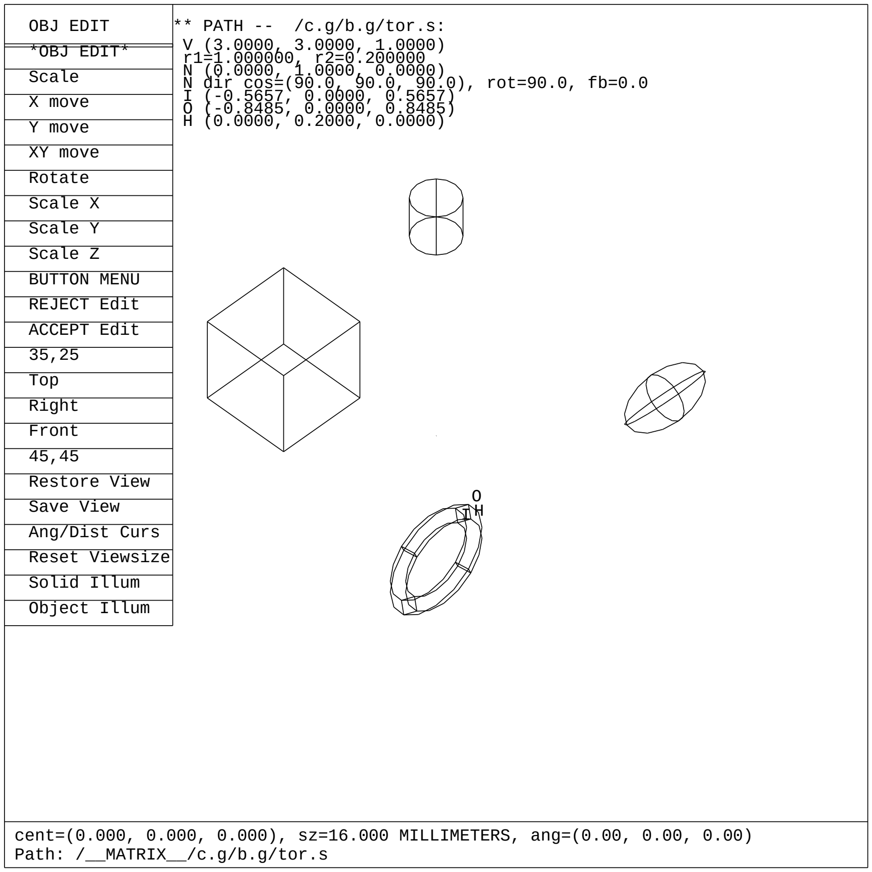

Figure 12.16: All Primitives Selected for Object Edit

mged> *translate 3 1 1*
mged>

The key point of the torus is moved to 3, 1, 1. The ellipsoid is moved by the same amount. See Figure 12.15.

mged> *press reject*
mged>

12.3.4 Editing Two Groups of Four Primitives

mged> *press oill*
mged>

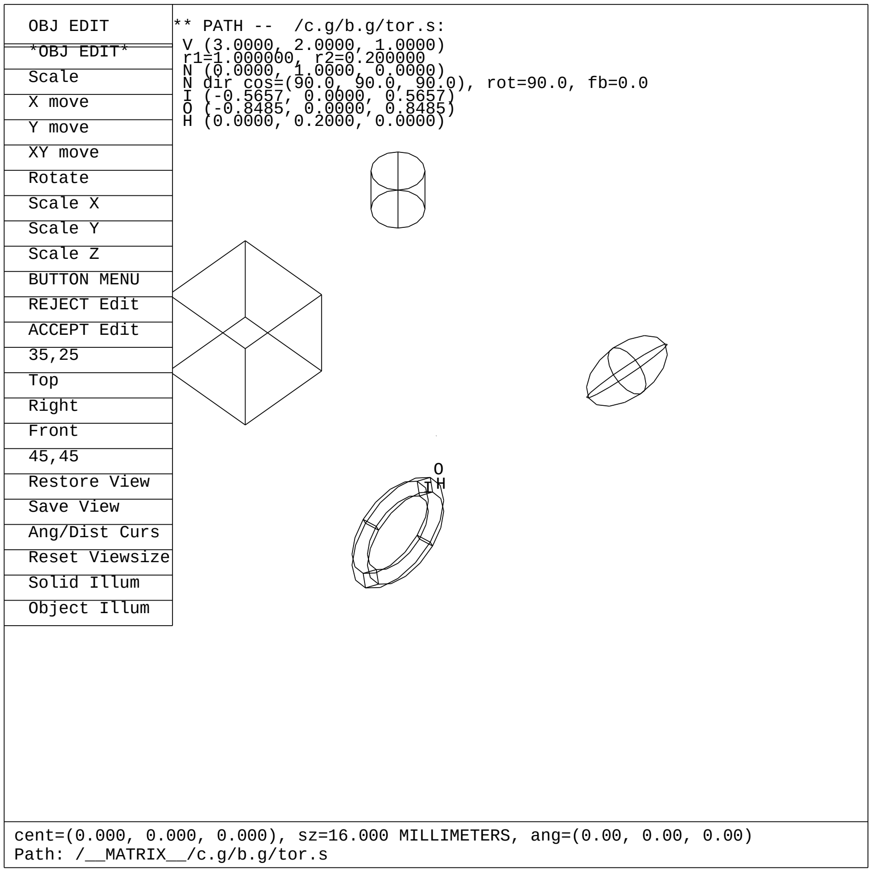

Figure 12.17: All Primitives Translated by (3, 2, 1)

Move the cursor up and down the screen until the primitive "tor.s" is illuminated, then select.
Move the cursor up and down until `/_MATRIX_/c.g/b.g/tor.s` *appears in the bottom status line, then select.*

Control has again been passed to the **OBJ EDIT** state. Notice that all of the primitives are illuminated (Figure 12.16). Although only the parameters for the torus will be changed in the display, the ellipsoid in group "b.g" will be affected by the edit.

`mged>` *translate 3 2 1*
`mged>`

The key point of the torus is moved to 3, 2, 1. All other primitives are moved

by the same amount. See Figure 12.17.
`mged>` *press reject*
`mged>`

Control has now returned to the VIEWING state.

Chapter 13

BUILDING A SET OF COORDINATE AXES

MGED does not display a set of XYZ co-ordinate axes on the screen. When you are in the 35,25 (isometric) viewing state the axes are positioned as in Figure 13.1. This database can be found in cad/db/axis.g.

If you would like a set of coordinate axes to assist in model building, the easiest thing to do is to construct three axes using "rcc" cylinder primitives via the "in" command;

`mged>` *in x rcc 0 0 0 50 0 0 1*
`mged>` *in y rcc 0 0 0 0 100 0 1*
`mged>` *in z rcc 0 0 0 0 0 150 1*
`mged>`

with the short leg as the "x" axis, next longer leg the "y" axis and longest leg the "z" axis, as in Figure 13.2. Now, at any stage through construction of the model, the 'solid' or 'object illuminate' mode can be used to identify which axis cylinder is going where; they will have the solid names of "x", "y", and "z". The name of the solid will also be displayed in the top left hand corner of the graphics window and at the bottom of this window.

Before going on to create a model, construct the three axes cylinders with the "in" commands mentioned above. Select the "button menu" in the upper left corner of the graphics window to enable the button menu, and select 35, 25 from this menu. Your axis will be displayed as shown in Figure 13.2.

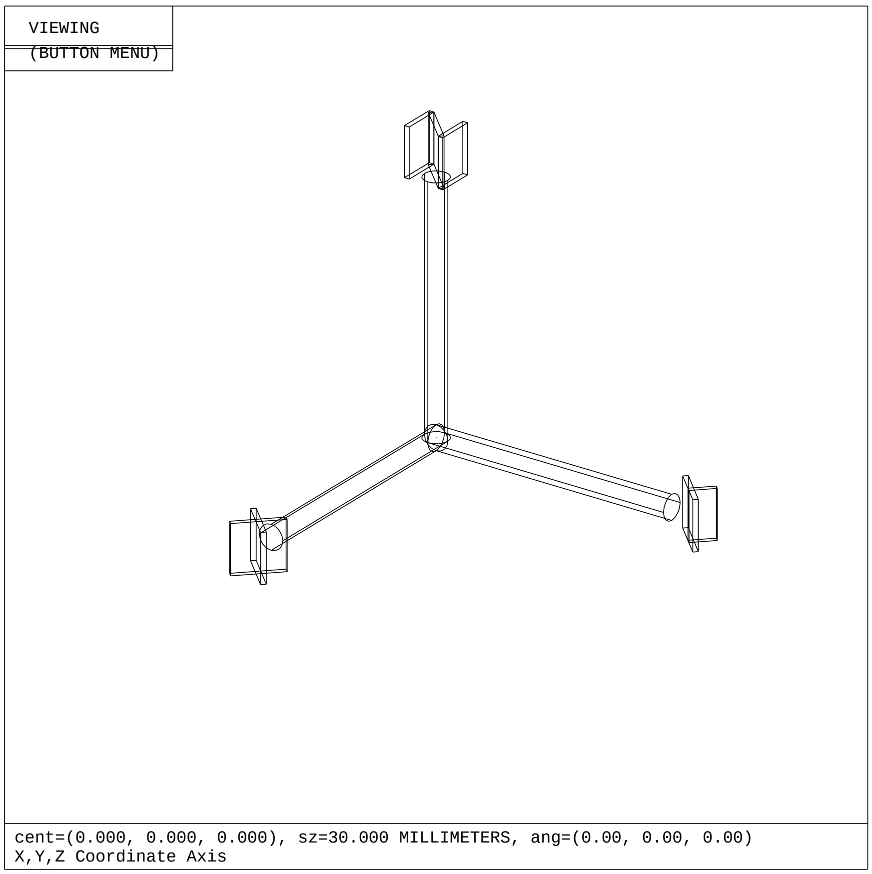

Figure 13.1: The Model Axes Viewed from 35,25

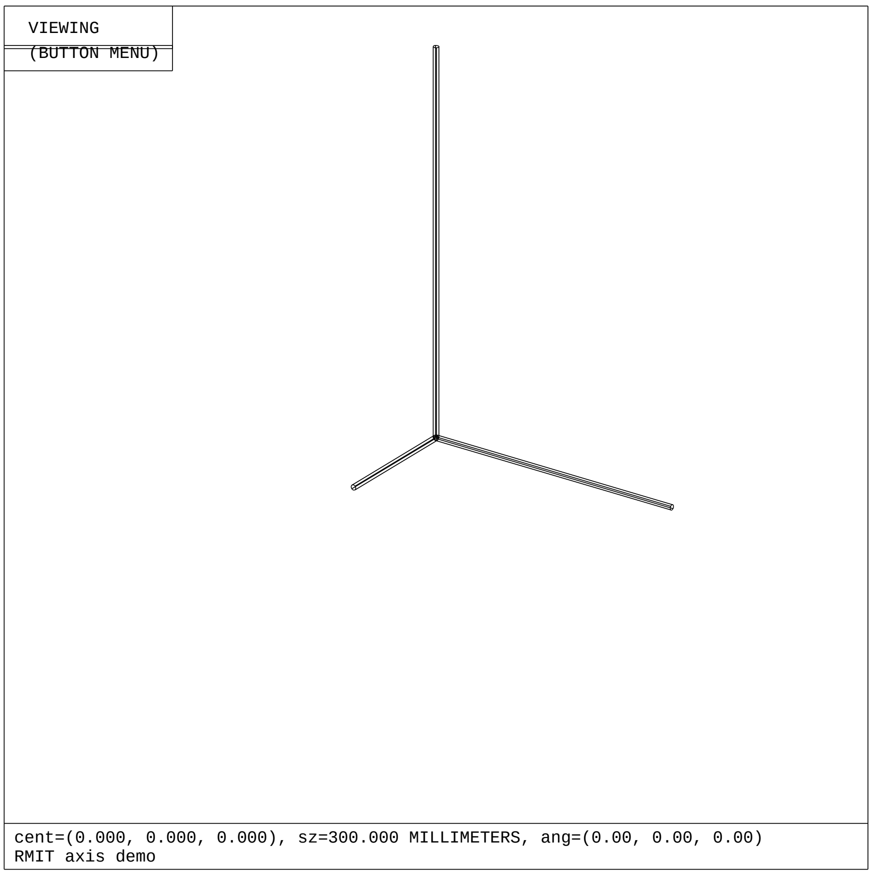

Figure 13.2: Example Axes Viewed form 35,25

Chapter 14

BUILDING A TIN WOODSMAN

The purpose of this tutorial is to demonstrate how to build a model using a few basic primitives. The model to be constructed is a tin woodsman. The four primitives used in the construction of the tin woodsman are an ARB8, a cylinder, an ellipsoid, and a torus. These four primitives will be duplicated several times, and each copy will be modified using solid editing, to obtain the required shapes. The finished version of this database can be found in the BRL-CAD Package file "db/woodsman.g".

14.1 Create Primitives

```
$ mged woodsman.g
BRL-CAD Release 3.0 Graphics Editor (MGED) Compilation 82
Thu Sep 22 08:08:39 EDT 1988
mikel@video.br:/cad/.mged.4d2

woodsman.g:  No such file or directory
Create new database (y|n)[n]?  y
attach (nu|tek|tek4109|ps|plot|sgi)[nu]?  sgi
ATTACHING sgi (SGI 4d)
Untitled MGED Database (units=mm)
mged> size 20
mged> title A Tin Woodsman
mged> in solid8 rpp
Enter XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX: -1 1 -1 1 -1 1
mged> in torus tor
Enter X, Y, Z of vertex:  0 0 0
```

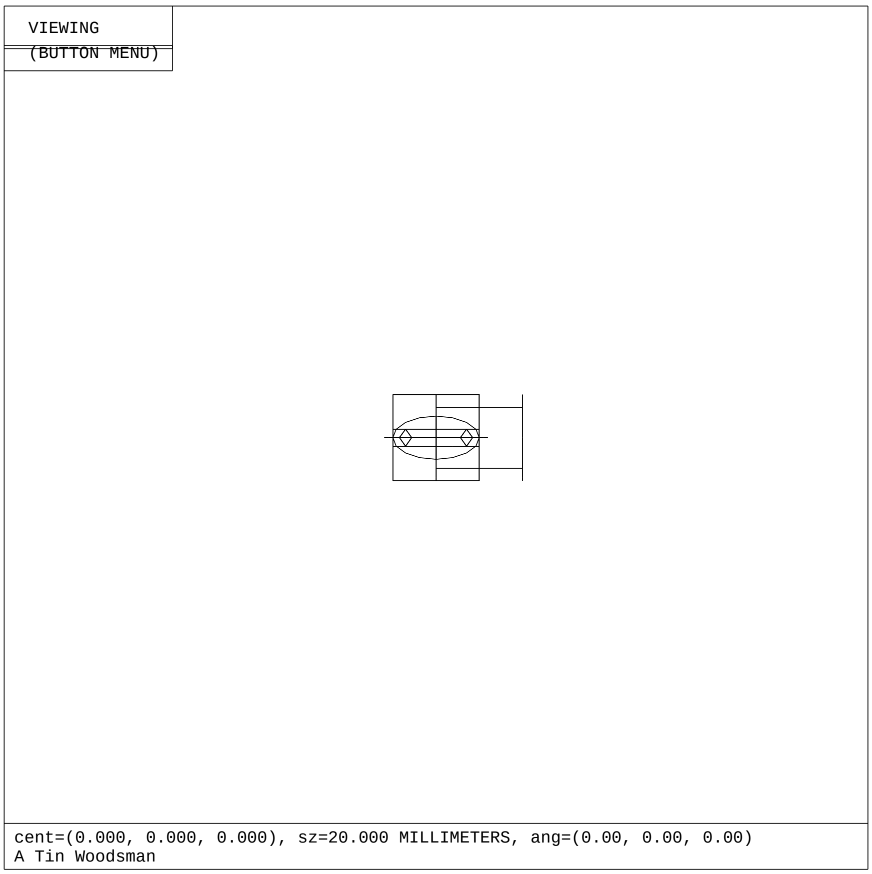

Figure 14.1: WoodsMan Primitives

```
Enter X, Y, Z of normal vector:  0 1 0
Enter radius 1:  1
Enter radius 2:  0.2
mged> in ellipsoid ellg
Enter X, Y, Z of vertex:  0 0 0
Enter X, Y, Z of vector A: 1 0 0
Enter X, Y, Z of vector B: 0 0.5 0
Enter X, Y, Z of vector C: 0 0 0.5
mged> in cylinder rcc
Enter X, Y, Z of vertex:  0 0 0
Enter X, Y, Z of height (H) vector:  2 0 0
Enter radius:  1
mged>
```

At this point, the screen should look like Figure 14.1.

14.2 Copy Primitives and Set Up for Edit

Although eight copies of the ellipsoid and two copies of the cylinder shall be used in the final solid, fewer copies are made initially since there is replication in the editing of these primitives.

```
mged> cp ellipsoid e.2
mged> cp ellipsoid e.6
mged> cp cylinder c.1
mged> cp solid8 s.1
mged> cp torus t.1
mged> Z
mged> e e.* c.1 t.1 s.1
vectorized in 0 sec
mged>
```

14.3 Create Funnel Hat

The solid "cylinder" has a height vector ("H") which is 2mm long. This will be used to good advantage, to make the Tin Woodsman's funnel hat, with the bowl of the funnel being 2mm high, and the tube of the funnel being 2mm long. The tube of the funnel will point straight up the +Y axis.

```
mged> sed c.1
mged> Select the "Rotate" entry in the solid edit menu
mged> p 0 45 90
mged>
```

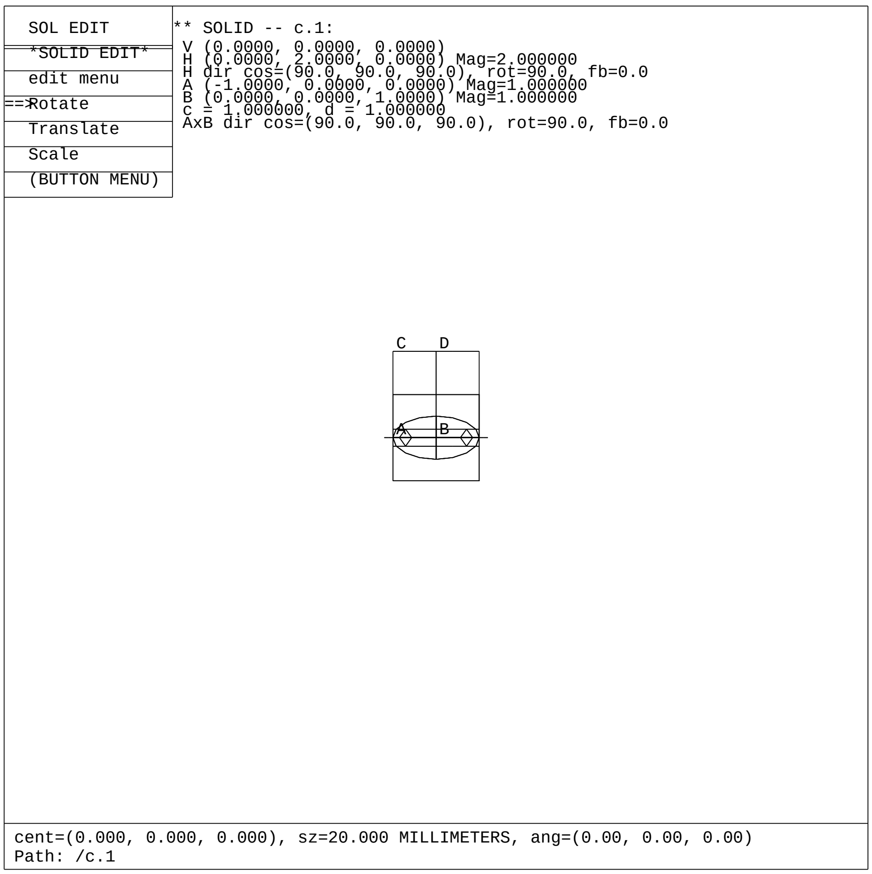

Figure 14.2: Funnel Bowl Cylinder After Rotation

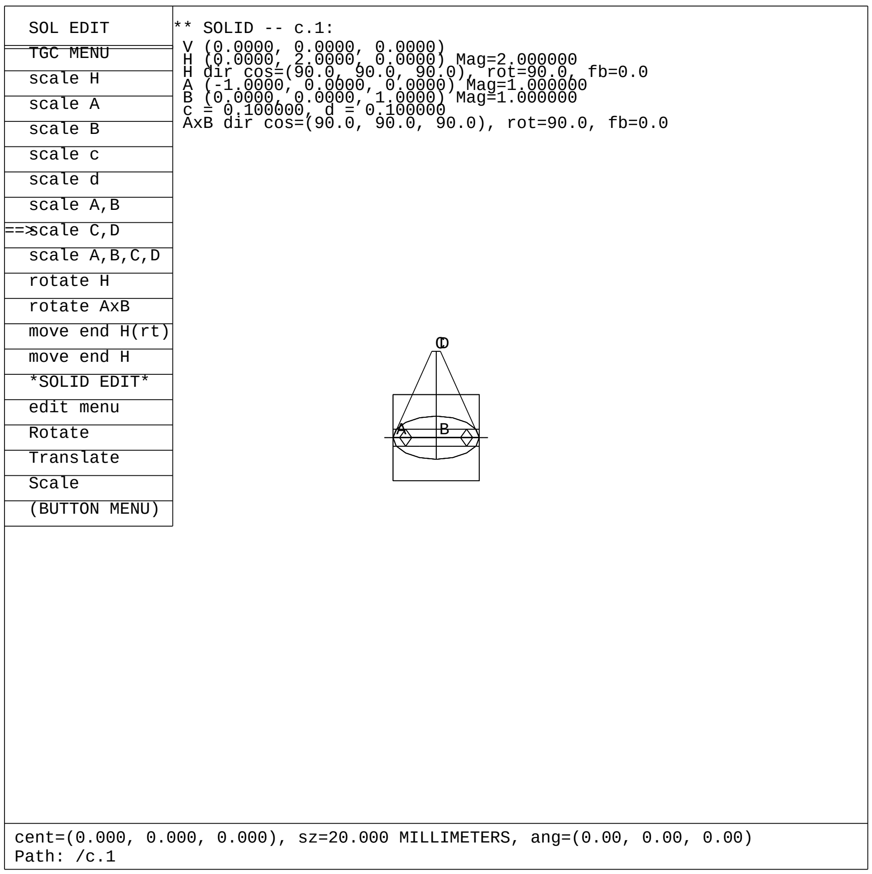

Figure 14.3: Funnel Bowl Cylinder After End Scaling

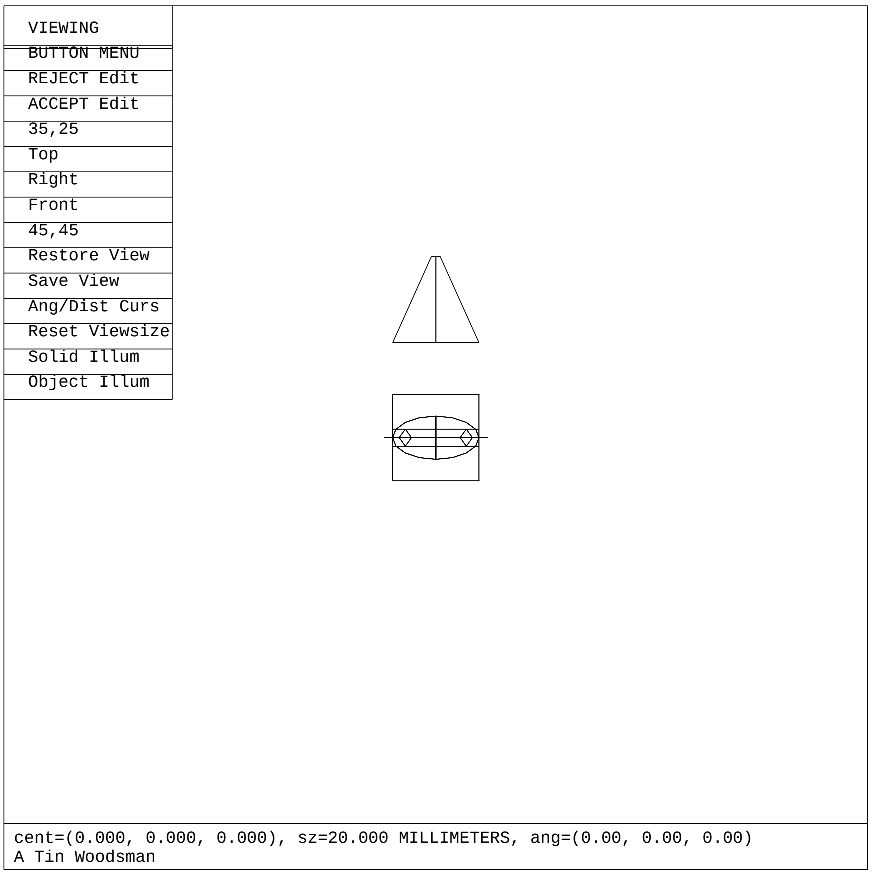

Figure 14.4: Funnel Bowl Cylinder After Moving

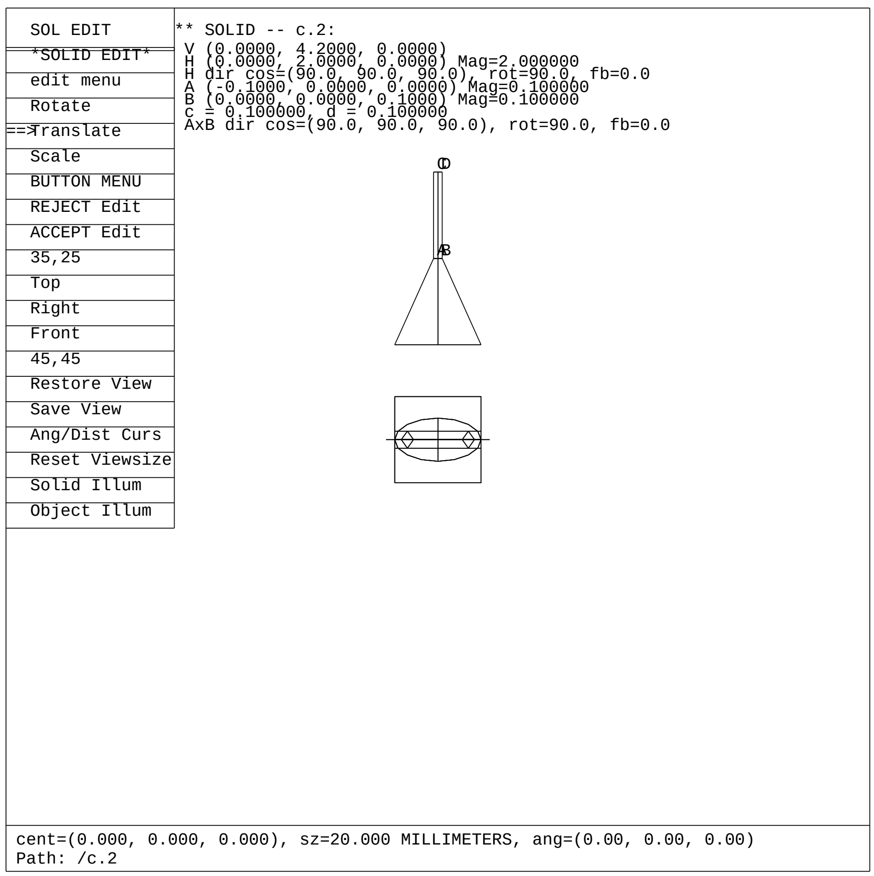

Figure 14.5: Funnel Tube Scaled and Positioned

This places the cylinder so that the lines BD and AC are at the outer ends of the cylinder. See Figure 14.2.

Next, the cylinder is shaped to look like the top of a funnel. The vectors c and d are scaled.

Select the "edit menu" entry in the solid edit menu
Select the "scale c" entry in the TGC menu
mged> *p .1*
Select the "scale d" entry in the TGC menu
mged> *p .1*
mged>

Figure 14.3 is the new shape of the cone. Note how the on-screen display records the new lengths of the "c" and "d" vectors. This cone must be moved to the planned locations for the top of the head.

Select the "Translate" entry in the solid edit menu
mged> *p 0 2.2 0*
Select the "ACCEPT Edit" entry in the button menu
mged>

The bottom of the hat is now properly shaped and positioned. The new version of the solid "c.1" has been saved in the model database. The editor returns to the viewing state. See Figure 14.4.

A copy of the saved "c.1" cone is made. A byproduct of the *cp* command is to display the new solid, as if the *cp c.1 c.2* command had been immediately followed by an *e c.2* command. This new solid will be edited to make the neck of the funnel, which is the top of the hat. The "c.2" copy of the cone must be scaled down to become a tube and the tube must be placed on top of the cone "c.1".

mged> *cp c.1 c.2*
mged> *sed c.2*
Select "scale A,B" in the TGC menu
mged> *p 0.1*
Select "Translate" in the Solid Edit menu
mged> *p 0 4.2 0*
Select "ACCEPT Edit" in the Button menu
mged>

Figure 14.5 is the new shape of the funnel tube. The woodsman's hat is comprised of solids c.1 and c.2.

14.4 Building the Head

The head of our Tin Woodsman is perfectly spherical, and will be located at coordinates (0, 2, 0). While it would be possible to duplicate the ellipsoid

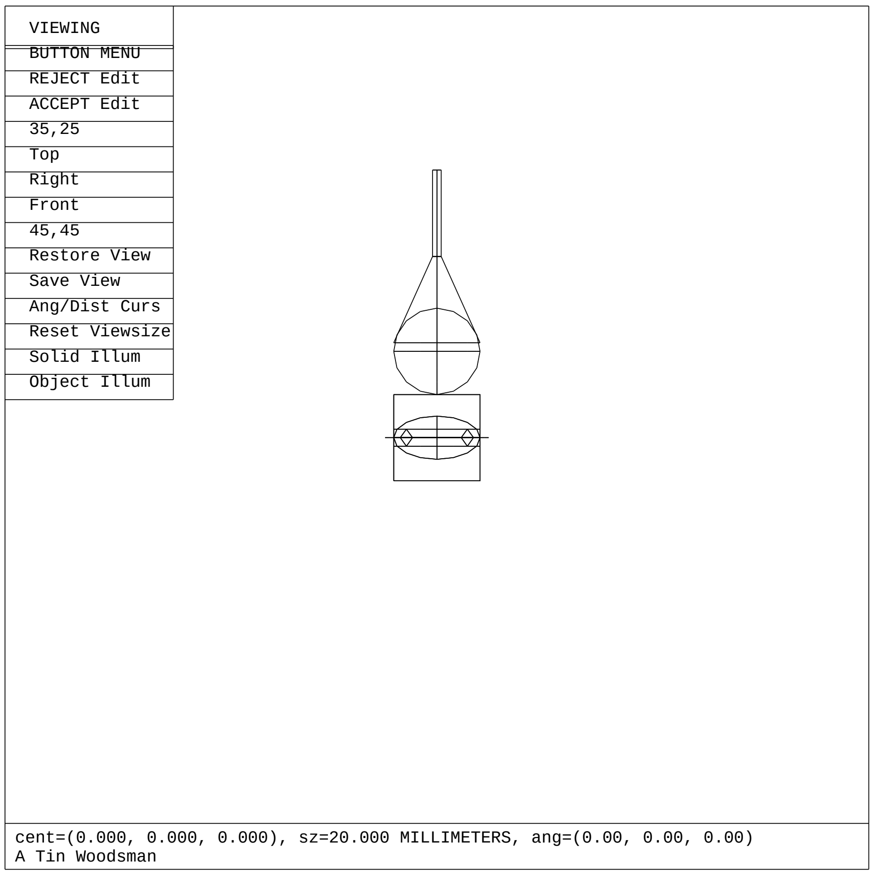

Figure 14.6: Head Sphere

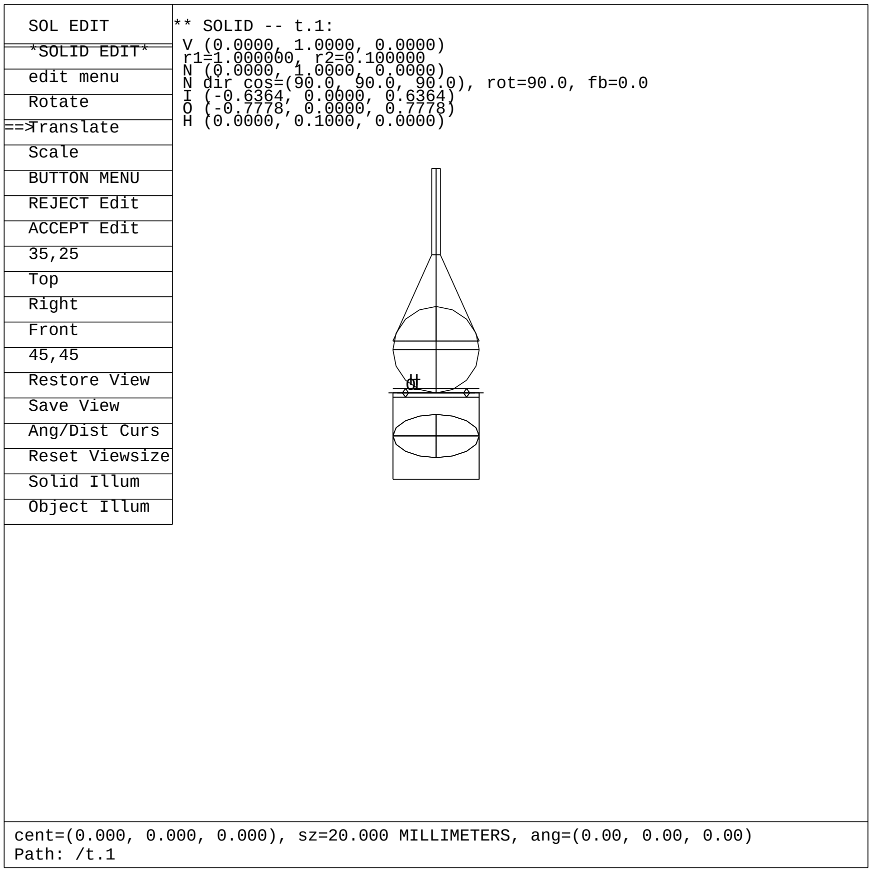

Figure 14.7: The Woodsman's Collar

solid created above, and modifiy it to produce the desired sphere, since all the parameters of the head sphere are known, it is more economical simply to use the *in* command to construct it directly. Figure 14.6 shows the results of this operation.

```
mged> in e.1
Enter solid type:  sph
Enter X, Y, Z of vertex:  0 2 0
Enter radius:  1
mged>
```

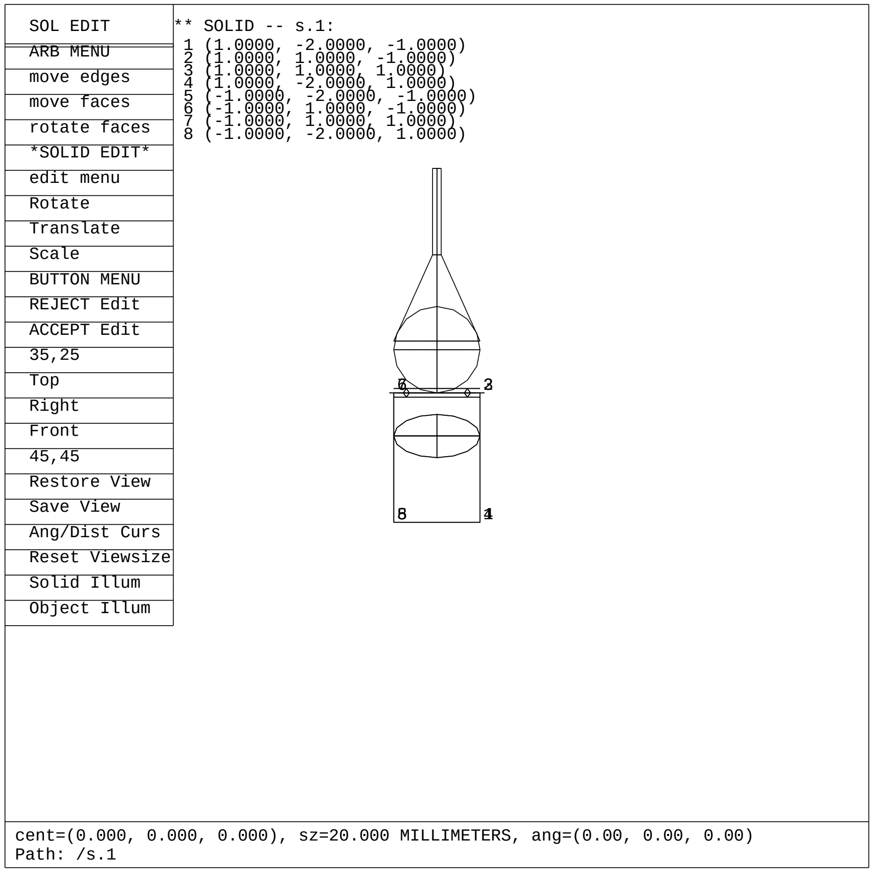

Figure 14.8: The Woodsman's Body

14.5 Building the Collar

The torus (primitive t.1) is used to build a collar between the head and the body. The ring of the collar is scaled to 0.1 of its original size, and repositioned at the base of the head. The results of this step are shown in Figure 14.7.

`mged>` *sed t.1*
Select "scale radius 2" in the TORUS menu
`mged>` *p 0.1*
Select "Translate" in the Solid Edit menu
`mged>` *p 0 1 0*
Select "ACCEPT Edit" in the Button menu
`mged>`

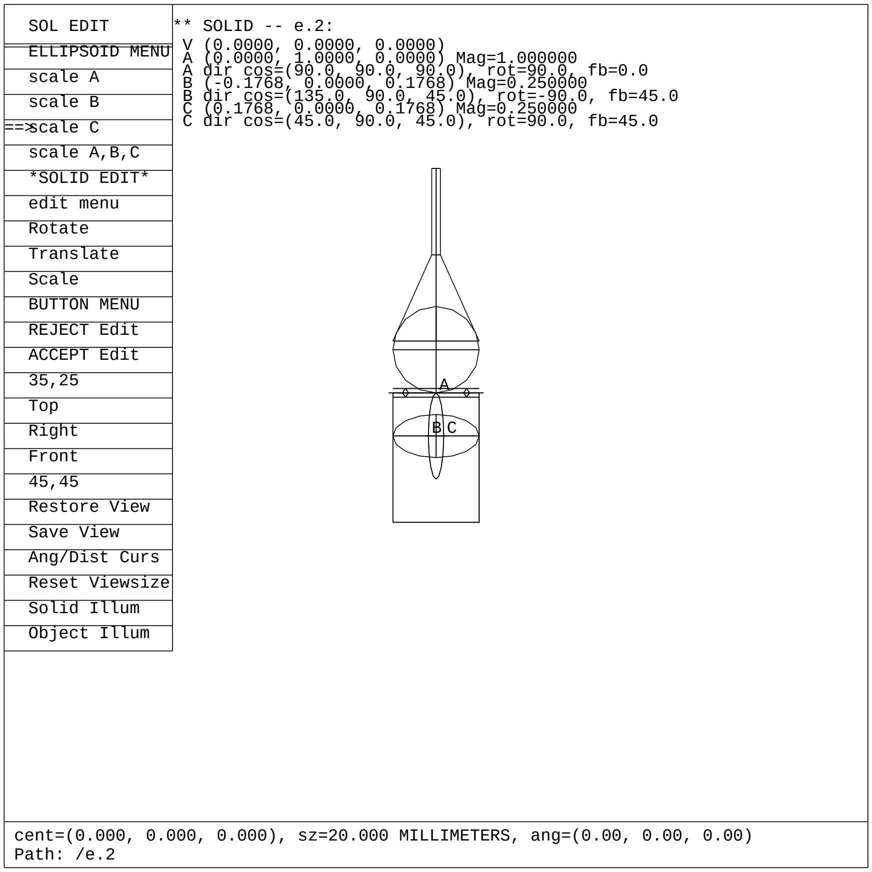

Figure 14.9: An Upper Arm Prototype

14.6 Building the Body

The ARB8 (primitive s.1) is used to build the body. The original height of s.1 is only 2mm; the required length of the body is 3mm, so the extrusion command is used to adjust the position of the lower (-Y) face of the solid. The result is shown in Figure 14.8.

`mged>` *sed s.1*
`mged>` *extrude 2367 3*
Select "ACCEPT Edit" in the Button menu
`mged>`

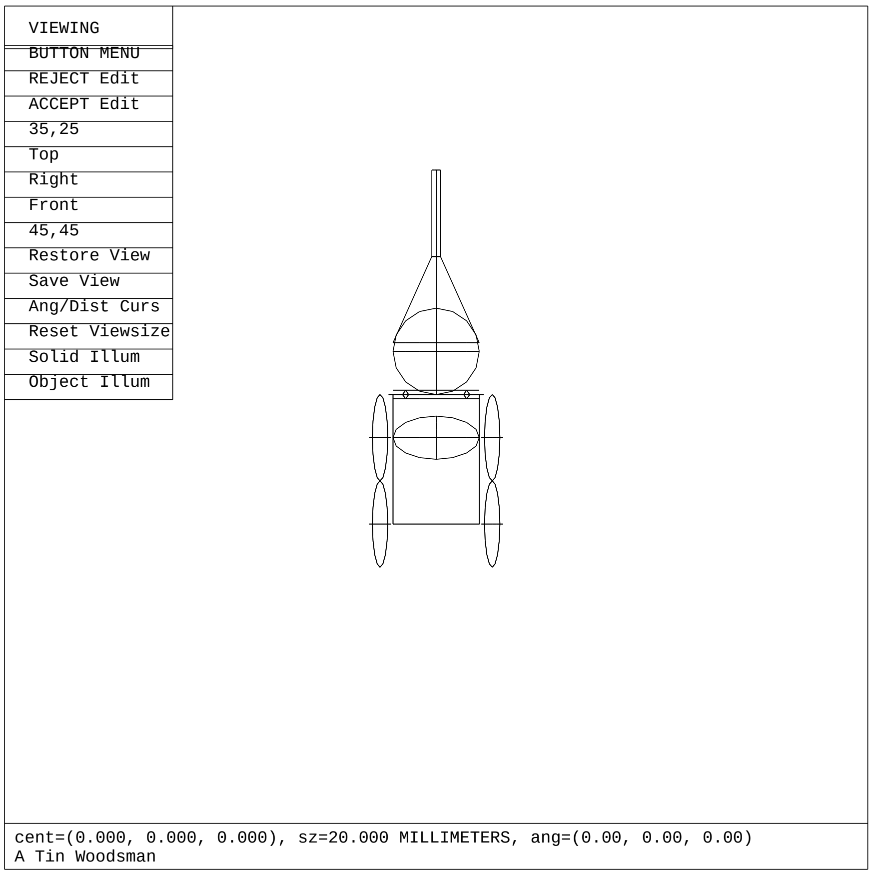

Figure 14.10: The Woodsman's Arms

14.7 Building the Arms

The ellipsoid primitive (e.2) is used to build the upper and lower parts of the left and right arms. The original solid "e.2" is oriented with the major axis of the ellipse oriented along the X axis. The arms need to have the major axis oriented along the Y axis, so first the solid is rotated. A more graceful arm is obtained by decreasing the length of the B and C vectors, and the resulting upper arm solid can be seen in Figure 14.9.

```
mged> sed e.2
Select "Rotate" in the Solid Edit menu
mged> p 0 45 90
Select "edit menu" in the Solid Edit menu
Select "scale B" in the ELLIPSOID menu
mged> p 0.25
Select "scale C" in the ELLIPSOID menu
mged> p 0.25
mged>
```

This e.2 solid will now be moved into final position as the upper left arm. Then it will be duplicated three times to make the rest of the arm parts. Finally, each new arm part will be translated into the proper position, as seen in Figure 14.10.

```
Select "Translate" in the Solid Edit menu
mged> p -1.3 0 0
Select "ACCEPT Edit" in the Button menu – This is the upper left arm
mged> cp e.2 e.3
mged> cp e.2 e.4
mged> cp e.2 e.5
mged> sed e.3
Select "Translate" in the Solid Edit menu
mged> p -1.3 -2 0
Select "ACCEPT Edit" in the Button menu – This is the lower left arm
mged> sed e.4
Select "Translate" in the Solid Edit menu
mged> p 1.3 0 0
Select "ACCEPT Edit" in the Button menu – This is the upper right arm
mged> sed e.5
Select "Translate" in the Solid Edit menu
mged> p 1.3 -2 0
Select "ACCEPT Edit" in the Button menu – This is the lower right arm
mged>
```

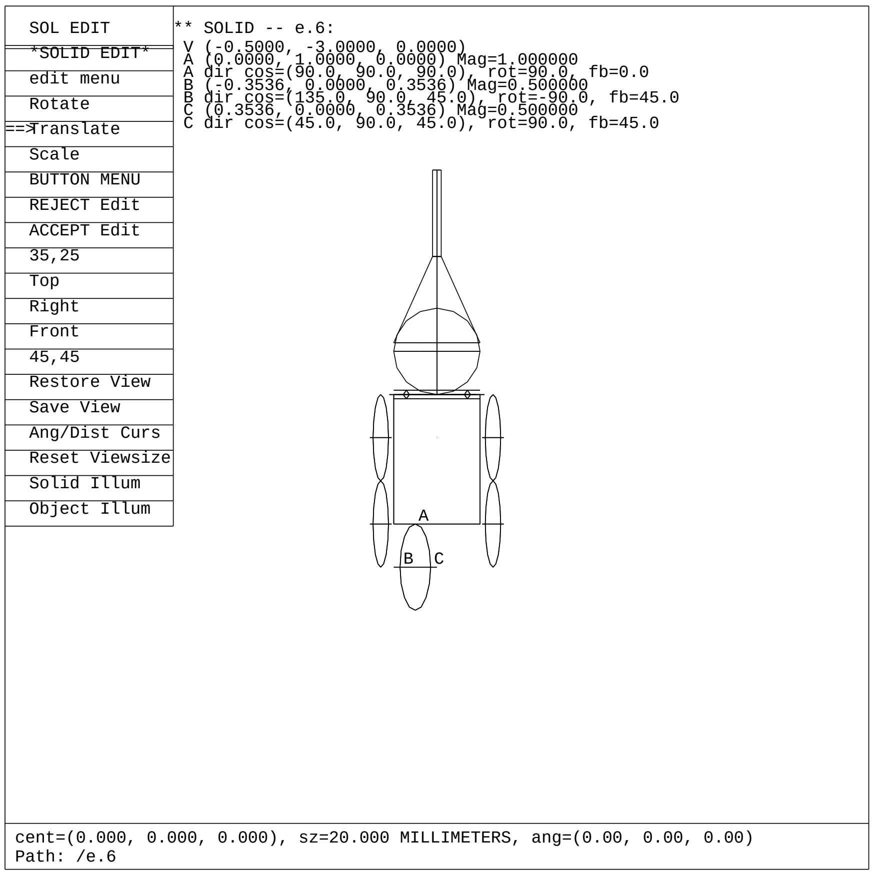

Figure 14.11: The First Leg

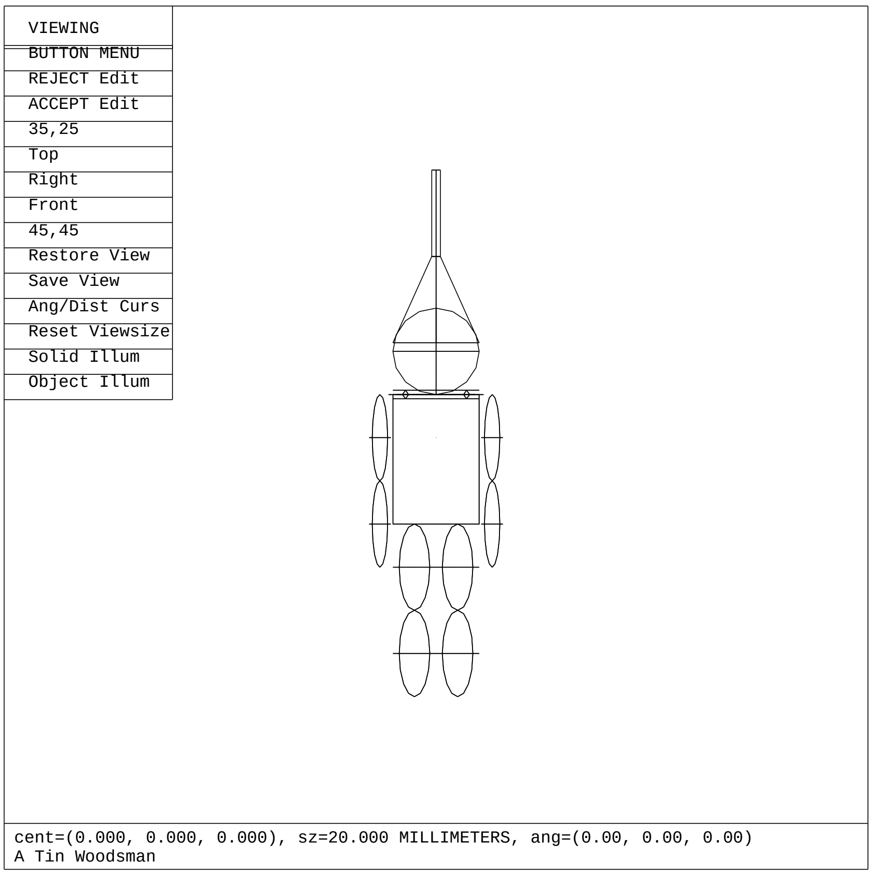

Figure 14.12: The Tin Woodsman

14.8 Building the Legs

The ellipsoid primitive (e.6) is used as a prototype to build the upper and lower parts of both legs from. The primitive e.6 is scaled, rotated, and translated into position as the upper left leg, as seen in Figure 14.11. Then, copies are made and translated to the remaining positions, just like the arms were.

`mged>` *sed e.6*
Select "Rotate" in the Solid Edit menu
`mged>` *p 0 45 90*
Select "Translate" in the Solid Edit menu
`mged>` *p -0.5 -3 0*
Select "ACCEPT Edit" in the Button menu – This is the upper left leg
`mged>` *cp e.6 e.7*
`mged>` *cp e.6 e.8*
`mged>` *cp e.6 e.9*
`mged>` *sed e.7*
Select "Translate" in the Solid Edit menu
`mged>` *p -0.5 -5 0*
Select "ACCEPT Edit" in the Button menu – This is the lower left leg
`mged>` *sed e.8*
Select "Translate" in the Solid Edit menu
`mged>` *p 0.5 -3 0*
Select "ACCEPT Edit" in the Button menu – This is the upper right leg
`mged>` *sed e.9*
Select "Translate" in the Solid Edit menu
`mged>` *p 0.5 -5 0*
Select "ACCEPT Edit" in the Button menu – This is the lower right leg

Figure 14.12 is the view on the screen, the Tin Woodsman. Take a moment to use the rotation knobs to view the model from various angles.

14.9 Building Regions

So far, this example has concentrated on describing the basic shapes involved in making the Tin Woodsman, without concern for establishing a proper hierarchical structure. To illustrate this point, the various solids will be grouped by purpose and composition. First, a region will be constructed to contain the torso, and the color of "cadet blue" will be assigned:

`mged>` *r torso.r u s.1*
```
Defaulting item number to 1001
Creating region id=1000, air=0, los=100, GIFTmaterial=1
```
`mged>` *mater torso.r*
```
Material =
```

```
Material?  (CR to skip) plastic
Param =
Parameter string?  (CR to skip) [RETURN]
Color = (No color specified)
Color R G B (0..255)?  (CR to skip) 95 159 159
Inherit = 0:  lower nodes (towards leaves) override
Inheritance (0|1)?  (CR to skip) [RETURN]
mged>
```

Second, a region will be constructed to contain the collar, which will be colored red:

```
mged> r collar.r u t.1
Defaulting item number to 1003
Creating region id=1002, air=0, los=100, GIFTmaterial=1
mged> mater collar.r
Material =
Material?  (CR to skip) plastic
Param =
Parameter string?  (CR to skip) [RETURN]
Color = (No color specified)
Color R G B (0..255)?  (CR to skip) 255 127 0
Inherit = 0:  lower nodes (towards leaves) override
Inheritance (0|1)?  (CR to skip) [RETURN]
mged>
```

Third, a region will be constructed to contain all the limbs, and a flesh color will be assigned. Even though none of the limbs touch each other, note how they are combined with the UNION operation, to create a single object of uniform composition and color.

```
mged> r limbs.r u e.2 u e.3 u e.4 u e.5 u e.6 u e.7 u e.8 u e.9
Defaulting item number to 1001
Creating region id=1000, air=0, los=100, GIFTmaterial=1
mged> mater limbs.r
Material =
Material?  (CR to skip) plastic
Param =
Parameter string?  (CR to skip) [RETURN]
Color = 0 0 0
Color R G B (0..255)?  (CR to skip) 255 200 160
Inherit = 0:  lower nodes (towards leaves) override
Inheritance (0|1)?  (CR to skip) [RETURN]
mged>
```

Next, the funnel needs to be placed in a region. For the sake of simplicity, the funnel will be solid, rather than having a hollow center. Note that the interior of the funnel overlaps with the top of the Woodsman's head. The funnel can be made "form fitting" by subtracting out the overlap zone:

```
mged> r funnel.r u c.1 - e.1 u c.2 - e.1
Defaulting item number to 1004
Creating region id=1003, air=0, los=100, GIFTmaterial=1
mged> mater funnel.r
Material =
Material?  (CR to skip) plastic
Param =
Parameter string?  (CR to skip) sh=100
Color = (No color specified)
Color R G B (0..255)?  (CR to skip) 35 107 142
Inherit = 0:  lower nodes (towards leaves) override
Inheritance (0|1)?  (CR to skip) mged> l funnel.r
funnel.r (len 4) REGION id=1003 (air=0, los=100, GIFTmater=1) --
Material 'plastic' 'sh=100'
Color 35 107 142
u c.1
- e.1
u c.2
- e.1
mged>
```

Note how the boolean expression was written. The concept that we need to express here is the combination of all the funnel parts, minus the portion of the head that overlaps with the inside of the funnel. The natural way to write this is

(c.1 union c.2) - e.1

but note that there are no grouping operations permitted in the *r* command. Furthermore, for historic reasons, union operations bind more loosely than intersection and subtraction, ie, there are implied groups between union operations. Thus, the expression above needs to be rewritten as the the formula:

(c.1 - e.1) union (c.2 - e.1)

which with the binding precedence can be expressed as:

c.1 - e.1 union c.2 - e.1

which is what was entered in the sequence above. To see the effect that this command had on the shape of "funnel.r", run these commands, the effect of which is shown in Figure 14.13:

```
mged> Z
mged> E funnel.r
vectorized in 1 sec
mged>
```

These regions should be grouped together into a group, for convenience in referencing. This can be done with these commands:

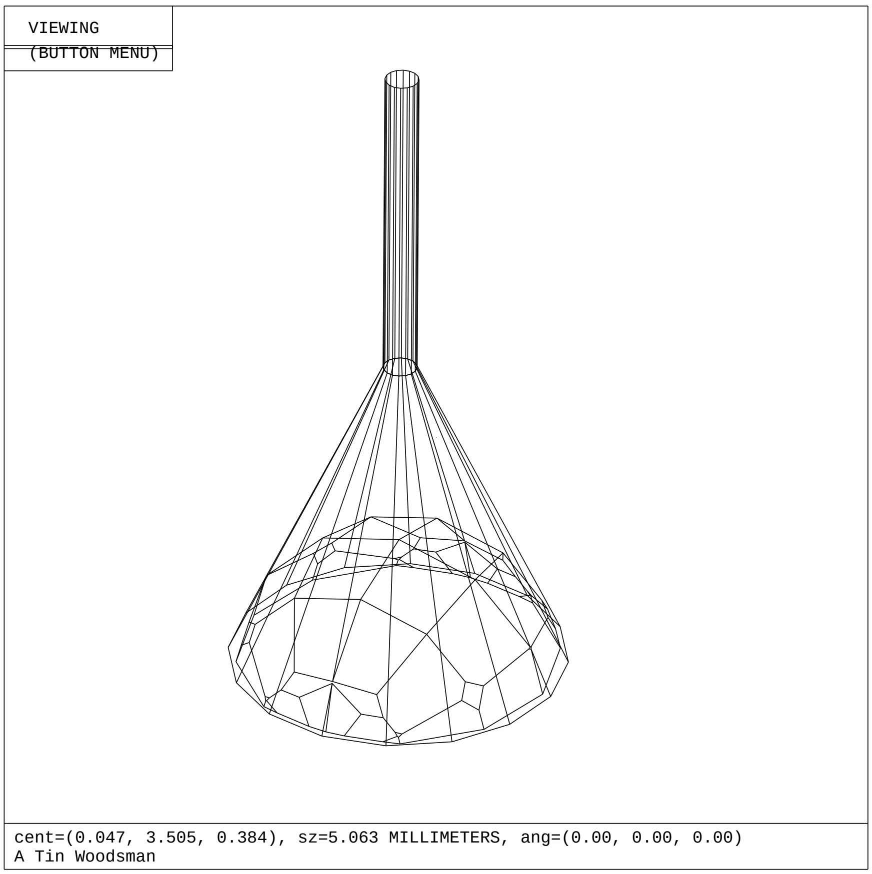

Figure 14.13: Evaluation of Funnel Hat Region

```
mged> g man.g collar.r funnel.r limbs.r torso.r
mged> Z
mged> e man.g
vectorized in 1 sec
mged>
```

The grouping *g* command combined the regions, the Zap command *Z* cleared the screen, and the edit *e man.g* command drew the whole object. As an exercise, run the database structure printing command *tree man.g* to obtain a simple depiction of the tree structure that has been created. For the final step of this example, the model will be ray-traced. Run the command:

```
mged> rt -s128
rt -s50 -M -s128 woodsman.g man.g
db title:  A Tin Woodsman
Buffering single scanlines
initial dynamic memory use=35152.
Interpreting command stream in old format
GETTREE: 0.01 CPU secs in 1 elapsed secs (1%)

...................Frame 0...................
PREP: 0.01 CPU secs in 0.01 elapsed secs (100%)
shooting at 13 solids in 4 regions
model X(-2,2), Y(-6,7), Z(-2,2)
Beam radius=0.078125 mm, divergance=0 mm/1mm

SHOT: 3.73 CPU secs in 6 elapsed secs (62.1667%)
Additional dynamic memory used=29728.  bytes
3515 solid/ray intersections:  1005 hits + 2510 miss
pruned 28.6%:  13647 model RPP, 8197 dups, 10740 RPP
Frame 0:  16384 pixels in 3.73 sec = 4392.49 pixels/sec
Frame 0:  16384 rays in 3.73 sec = 4392.49 rays/sec (RTFM)

Press RETURN to reattach
[RETURN]
mged>
```

Chapter 15

BUILDING A ROBOT ARM

The model shown in Figure 15.1 will be described in a step by step instructions on how to build and display this model.

This is the MGED input file:

```
in btm box 0 0 0    0 -90 0      40 0 0   0 0 6
in btm1 box 0 -90 0  0 -61.549 0  40 0 0   0 0 6
in rad rcc 20 -150 0   0 0 6   8
in cyl rcc 20 -45 6    0 0 30  20
in cyl1 rcc 20 -45 0 0 0 36 15.5
in cyl2 rcc 20 -45 0 0 0 36 12.5
in hole rcc 8 -8 0   0 0 6   3
in hole1 rcc 32 -8 0  0 0 6   3
cp hole1 hole2
in gus raw 21.5 -25.3 6  0 0 30  0 25.3 0  -3 0 0
in cnr box 0 0 0 6 6 0 6 0 0 0 0 6
in cnr1 box 34 0 0 0 -6 0 6 0 0 0 0 6
cp cnr cnr2
cp cnr1 cnr3
in rad1 rcc 6 -6 0 0 0 6 6
in rad2 rcc 34 -6 0 0 0 6 6
in head rcc 20 -45 36 0 0 30 18
in shaft rcc 20 -45 36 0 0 -50 12.5
in han rcc 20 -45 51 0 120 0 6
in ball sph 20 75 51  15
in cut box 20 -45 0 0 50 0 25 0 0 0 0 40
in squ box 12 -53 -14   0 16 0   16 0 0   0 0 -30
r handle u squ u shaft u han - ball
```

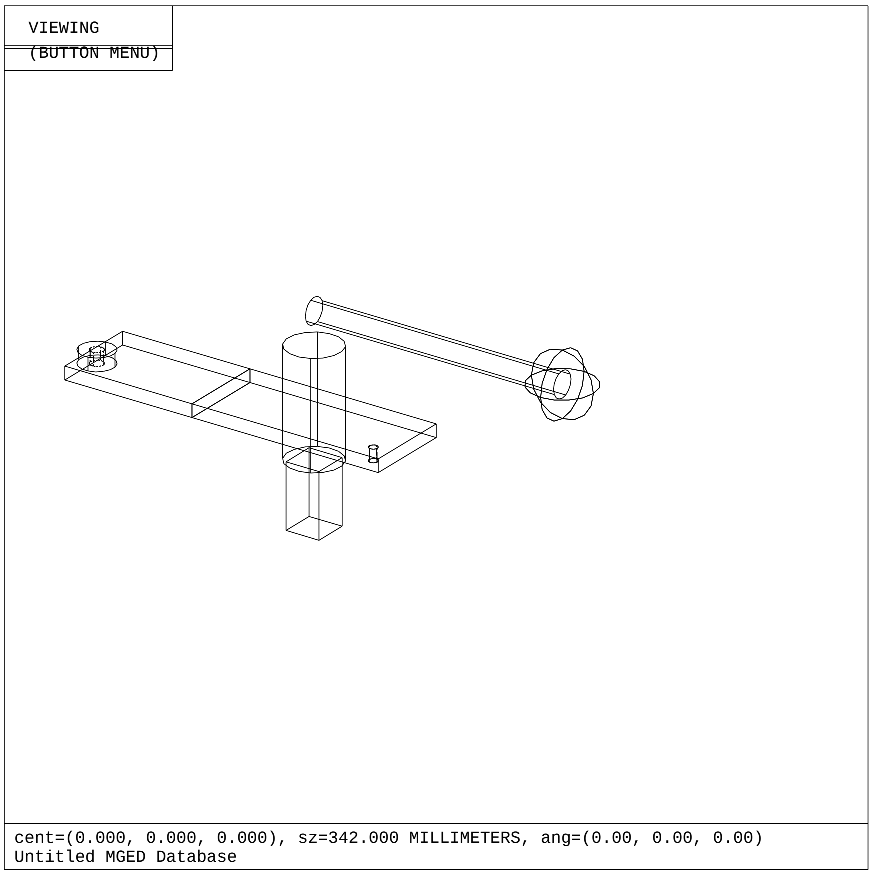

Figure 15.1: The RMIT Robot Arm

```
r knob u ball
r cor u cnr2 + rad1
r cor1 u cnr3 + rad2
in hole4 rcc 20 -150 0   0 0 6   3
cp hole2 hole3
r base u btm u btm1 - hole2 - hole3 - hole4 u rad - hole4
g all base handle knob
size 300
e all
```

This is the MGED dialog:

```
mged mark
BRL Graphics Editor (MGED) Version 2.31
  Sat Oct 17 20:33:05 PDT 1987
  mg\@godzilla:/usr/staff/mg/brlcad/mged

mark: No such file or directory
Crete new database (y/n)[n]? y
attach (nu|tek|plot|ir) [nu]? nu
ATTACHING nu (Null Display)
Untitled MGED Database (units=mm)
mged> in btm box 0 0 0 0 -90 0 40 0 0 0 0 6
mged> in btm1 box
Enter X, Y, Z of vertex:  0 -90 0
Enter X, Y, Z of vector H:  0 -61.549 0
Enter X, Y, Z of vector W:  50 0 0   40 0 0
Enter X, Y, Z of vector D: 0 0 6
mged> in rad rcc 20 -150 0 0 0 6 8
mged> in cyl rcc
Enter X, Y, Z of vertex:  20 -45 6
Enter X, Y, Z of height (H) vector: 0 0 30
Enter radius:  20
mged> in cyl1 rcc 20 -45 0 0 0 36 15.5
mged> in cyl2 rcc 20 -45 0 0 0 36 12.5
mged> in hole rcc 8 -8 0 0 0 6 3
mged> in hl ole 1 rcc 2 32 -8 0 0 0 6 2 3
mged> cp hole1 hole2
mged> in gus raw
Enter X, Y, Z of vertex: 21.5 -25.3 6
Enter X, Y, Z of vector H: 0 0 30
Enter X, Y, Z of vector W: 0 25.3 6
Enter X, Y, Z of vector D: -3 0 0
mged> in cnr box 0 0 0 06 6 0 6 0 0 0 0 6
mged> incnr1 box 34 0 0 0 -6 0 6 0 0 0 0 6
```

```
incnr1: no such command, type ? for help
mged> in cnr1 box 34 0 0 0 -6 0 6 0 0 0 0 6
mged> cp cnf r cnr2
mged> in cp cnr1 cnr3
mged> in rad1 rcc 6 -6 0 0 0 6 6
mged> in rad2 rcc 34 -6 0 0 0 6 6
mged> in shaft rcc 20 -45 36 0 0 30 18
mged> in shaft rcc 20 -45 36 0 0 -50 12.5
mged> in han rcc 20 -45 51 0 120 0 6
mged> in ball sph
Enter X, Y, Z of vertex: 20 75 51
Enter radius: 15
mged> in cut box 20 -45 0 0 50 0 25 0 0 0 040
Enter Z: 03  NOTE: error again
mged> killall cut
mged> in cut box 20 -45 0 0 50 0 25 0 0 0 0 40
mged> in squ box 12 -53 -14 0 16 0 16 0 0 0 0- -30
mged> r handle + squ shaft u han u ball
Defaulting item number to 1001
Creating region id=1000, air=0, los=100, GIFT material=1
mged> r knob + ball
Defaulting item number to 1002
Creating region id=1001, air=0, los=100, GIFT material=1
mged> r cor + cnr2 + rad1
Defaulting item number to 1003
Creating region id=1002, air=0, los=100, GIFT material=1
mged> r cor1 + cnr3 + rad2
Defaulting item number to 1004
Creating region id=1003, air=0, los=100, GIFT material=1
mged> mater knob plastic
Was
Parameter string? n
Override material color (y/n)[n]? y
R G B (0..255)? 255 0 0  NOTE:  This is color RED
mged> mater handle plastic
mged> Was
Parameter string? n
Override material color (y/n)[n]? y
R G B (0..255)? 219 147 112   NOTE:  This is color TAN
mged> r base + btm u btm1 u gus cyl - cyl1 m1 - hole2 -hole3 -hole4 u rad-
hole4
mged> error in number of args!   NOTE: Typing errors
mged> r base + btm u btm1 - hole2 - hole3 - hole4 u rad - hole4
Defaulting item number to 1005
```

```
Creating region id=1004, air=0, los=100, GIFTmaterial=1
dir_lookup:  could not find "hole3"
skipping hole3
dir_lookup:  could not find "hole4"
skipping hole4
dir_lookup:  could not find "hole4"
skipping hole4
mged> t
ball     cnr3     gus     knob/
base/     cor/     han     rad
btm     cor1/     handle     rad1
btm1     cut     head     rad2
cnr     cyl     hole     shaft
cnr1     cyl1     hole1     squ
cnr2     cyl2     hole2
mged> in hole 4 rcc 20 -150 0 0 0 6 3
mged> cp hole2 hole3
mged> killall base NOTE:  Redo "base" region
mged> r base + btm u btm1 - hole2 - hole3 - hole4 u rad - hole4
Defaulting item number to 1006
Creating region id=1005, air=0, los=100, GIFTmaterial=1
mged> g all base handle knob
mged> tree all
| all_________________| base_________| btm
                         | btm1
                         | hole2
                         | hole 3
                         | hole4
                         | rad
                         | hole4
                         | handle______________| squ
                                     | shaft
                                     | han
                                     | ball
                         | knob________________| ball
            | handle_______________| squ
                        | shaft
                        | han
                        | ball
            | knob_________________| ball
mged> l base
base (len 9) REGION id=1005 (air=0, los=100, GIFTmater=1)--
 + btm
 u btm1
```

```
 - hole2
 - hole3
 - hole4
 u rad
 - hole4
 u handle
 u knob
mged> l gus
gus:  ARB8 (ARB6)
1 (21.5000, -25.3000, 6.0000)
2 (21.5000, 0.0000, 6.0000)
3 (21.5000, 0.0000, 6.0000)
4 (21.5000, -25.3000, 36.0000)
5 (18.5000, -25.3000, 6.0000)
6 (18.5000, 0.0000, 6.0000)
7 (18.5000, 0.0000, 6.0000)
8 (18.5000, -25.3000, 36.0000)
mged> l ball
ball:  ELL
V (20.0000, 75.0000, 51.0000)
A (15.0000, 0.0000, 0.0000) Mag=15.000000
A dir cos=(0.0, 90.0, 90.0), rot=0.0, fb=0.0
B (0.0000, 15.0000, 0.0000) Mag=15.000000
B dir cos=(90.0, 0.0, 90.0) rot=90.0, fb=0.0
C (0.0000, 0.0000, 15.0000) Mag=15.000000
C dir cos=(90.0, 90.0, 0.0) rot=90.0, fb=90.0
mged> l knob
knob (len 1) REGION id=1001 (air=0, los=100, GIFTmater=1)--
Material "plastic"
Color 255 0 0
 + ball
mged> l handle
handle (len 4) REGION id=1000 (air=0, los=100, GIFT MATER=1)--
Material "plastic" "n
Color 219 147 112
 + squ
 u shaft
 u han
 u ball
mged> canter-0-75 0
mged> size 300
mged> tops
all/     cor1/     cyl2      hole1
cnr      cut     gus
```

```
cnr1     cyl     head
cor/     cyl1     hole
mged> analyze cyl
cyl:  TGC
V (20.0000, -45.0000, 6.0000)
H (0.0000, 0.0000, 30.0000) Mag=30.000000
H dir cos=(90.0, 90.0, 0.0), rot=90.0, fb=90.0
A (-17.5032, -9.6767, 0.0000) Mag=20.000000
B (9.6767,-17.5032, 0.0000) Mag=20.000002
c=20.000000, d=20.000002
AxB dir cos=(90.0, 90.0, 0.0), rot=90.0,fb=90.0
Surface Areas:  base(AxB)=1256.6371
  top(CxD)=1256.6371 side=3769.9114
Total Surface Area=6283.1855
   Volume=37699.1132 (0.0100 gal)
mged> q
```

Chapter 16

RT MATERIAL TYPE, PROPERTIES, and COLOR

First the solids must be formed into a "region", eg:

r ball u torus u tube-hole

To change material type, properties and color use the "mater" command:

```
mged> mater base
Material =
Material?  (CR to skip) plastic
Param =
Parameter string?  (CR to skip) sh=10 dl=0.2 sp=0.8 re=0.75
Color = (No color specified)
Color R G B (0..255)?  (CR to skip) 112 219 147
Inherit = 0:  lower nodes (towards leaves) override
Inheritance (0|1)?  (CR to skip) 0
mged>
```

For the values in Parameter String for material "Plastic", you can enter such things as: "shinyness (sh)", "specular lighting fraction (sp)", "diffuse lighting fraction (di)", "transmission fraction (tr)", "reflection fraction (re)", and "refractive index (ri)". Two formulas must hold to keep the material "physical": sp + di=1.0, and tr + re=1.0.

Suggested values for these properties are listed below:

sh=10, dl=0.2, sp=0.8, re=0.75

NOTE: Not all of these fields need to be input, you can use the system defaults for the rest.

To display objects in different colors on the screen, each object must be a region with its own material properties and colors. All regions must be displayed on screen before a ray tracing can be performed (region objects can have cutouts to display other parts).

R	G	B	COLOR
112	219	147	aquamarine
50	204	153	med aquamarine
0	0	0	black
0	0	255	blue
95	159	159	cadet blue
66	66	111	corn flower blue
107	35	142	dk slate blue
191	216	216	light blue
143	143	188	light steel blue
50	50	204	medium blue
127	0	255	medium slate blue
47	47	79	midnight blue
35	35	142	navy blue
50	153	204	sky blue
0	127	255	slate blue
35	107	142	steel blue
255	127	0	coral
0	255	255	cyan
142	35	35	firebrick
204	127	50	gold
219	219	112	golden rod
234	234	173	med goldenrod
0	255	0	green
47	79	47	dark green
79	79	47	dk olive green
35	142	35	forest green
50	204	50	lime green
107	142	35	med forest green
66	111	66	medium sea green
127	255	0	med spring green
143	188	143	pale green
35	142	107	sea green
0	255	127	spring green
153	204	50	yellow green
47	79	79	dk slate grey
84	84	84	dim grey
168	168	168	light grey

R	G	B	COLOR
159	159	95	khaki
255	0	255	magenta
142	35	107	maroon
204	50	50	orange
219	112	219	orchid
153	50	204	dark orchid
147	112	219	medium orchid
188	143	143	pink
234	173	234	plum
255	0	0	red
79	47	47	indian red
219	112	147	medium violet
255	0	127	orange red
204	50	153	violet red
111	66	66	salmon
142	107	35	sienna
219	147	112	tan
216	191	216	thistle
173	234	234	turquoise
112	147	219	dk turquoise
112	219	219	med turquoise
79	47	79	violet
159	95	159	blue violet
216	216	191	wheat
252	252	252	white
255	255	0	yellow
147	219	112	green yellow

material types are: plastic mirror glass texture

Shinyness (ie: sh=16)

Refractive index for: crown glass = 1.52 Flint glass = 1.65 Rock salt = 1.54 Water = 1.33 Diamond = 2.42

Transmission fraction for a mirror: re=1.0 (tr=0)

Chapter 17

RAYTRACING YOUR CREATION

Once you have finished creating all your solids, positioned them in their correct relationships to each other, formed all your regions (forming your finished object), created groups (if required), you can now do a ray- tracing of the view displayed on the screen.

Note! If you want to display solids or objects (collection of solids regioned together) of different colors, each of the solids or objects must be separate regions so you can give them a specific color.

The raytracing command is

rt [-s#]

This command produces a color shaded image of the solids or objects on the display. This color shaded image will appear on a frame buffer display. The resolution of the image (number of rays) is equal to "#" from the "-s" option. If the "-s" option is absent, 50x50 ray solution will be used (very course raytrace). The higher the "-s" option the better the raytracing, but it takes longer to display. Recommended optimun value of "-s" option for picture quality and speed of display is 256!. Some examples follow:

rt -s128

rt

rt -s256

When the rt command is given the text and graphic window will appear, then the frame buffer starts to appear (the picture window). The first scaned display will be what was previously stored in it, it will then over write it with your picture; sometimes two buffer scans are displayed before yours.

The default background color is blue with steel grey colored solids and objects. The terminal will beep when the scanned picture is finished; press return to get back to the "text and graphic" window.

With the blue background it is sometimes hard to visualize the raytraced picture; two things you can do to improve the situation is: (a) Make separate regions for all solids and objects, so that you can assign a specific color to each region; this can be a time consuming task if you have a lot of solids and objects. (b) Construct as a separate region, three thin flat plates to form two walls with a bottom, as shown below; using "make name arb8", then solid editing this arb8, using move faces to the required thickness, then use command "cp" (copy command) to make two more copies which you can rotate to their respective relationships, then translate all three into the correct positions relative to each other and the solids and objects you are displaying.

The advantage of doing this is to give the light source something to reflect off, giving back lighting; improving contrast considerably. With the three plates formed into their own region you can delete them from the screen with the "d" command, rotate your creation then re-display your plates (walls) with the "e" command to do another rt, the walls need to be deleted from the display when you rotate your objects, otherwise everything will rotate together.

Figure

A bonus of having constructed these three walls is that you can quickly change the material type to "mirror" so that you can get reflections of the three hidden faces.

Chapter 18

CONCLUSIONS

MGED performs two basic functions: viewing and editing. The standard viewing capabilities of zooming, slewing, slicing, and rotation are available. Likewise, all the standard editing features are also available. The user easily traverses the hierarchical data structure, applying the editing functions of rotation, translation, and scaling to any position in the hierarchy. The hierarchical structure can be modified and regrouped and regions created and modified. Specific parameter editing can also be applied to the solids to produce any shape solid desired.

For several decades, the production and modification of geometric models suitable for sophisticated engineering analysis has been a slow, labor-intensive procedure. In an effort to improve the response time of geometric models, the Ballistic Research Laboratory (BRL) has developed an interactive model editor for their combinatorial solid geometry modeling system (The BRL-CAD Package). The user interface to the geometry of these models is a program called the Multi-device Graphics Editor (MGED) that is designed to replace the traditional manual method for producing and modifying model databases. Using MGED, the geometric models are interactively viewed, modified, and constructed with immediate visual feedback at each step. When desired, the MGED editor can be operated without the need for explicit numerical input and opens a new dimension in the model building process. MGED has made great gains in reducing the bottleneck in the creation of high resolution geometric models.

Bibliography

[BR75] L. W. Bain Jr. and M.M. Reisinger. *The GIFT Code User Manual; Volume I, Introduction and Input Requirements (U).* Technical Report BRL Report No. 1802, AD #B0060371, U.S. Army Ballistic Research Laboratory, APG, MD, July 1975.

[Dei84] P. H. Deitz. Modern computer-aided tools for high-resolution weapons system engineering. DoD Manufacturing Technology Advisory Group MTAG-84 Conference, Seattle WA, November 1984.

[KBR79] G. G. Kuehl, L. W. Bain Jr., and M. M. Reisinger. *The GIFT Code User Manual; Volume II, The Output Options (U).* Technical Report ARBRL-TR-02189, AD #A078364, U.S. Army Ballistic Research Laboratory, APG, MD, September 1979.

[Muu87] M. J. Muuss. Understanding the preparation and analysis of solid models. In D. F. Rogers and R. A. Earnshaw, editors, *Techniques for Computer Graphics*, Springer-Verlag, 1987.

[RJLK78] D. M. Ritchie, S. C. Johnson, M. E. Lesk, and B. W. Kernighan. The c programming language. *The Bell System Technical Journal*, 57(6):1991–2019, 1978.

[RT78] D. M. Ritchie and K. Thompson. The unix time-sharing system. *The Bell System Technical Journal*, 57(6):1905–1929, 1978.

[WM80] E. P. Weaver and M. J. Muuss. Interactive construction of com-geom targets. In *Proceedings of the BRL Spring Technical Conference*, U.S. Army Ballistic Research Laboratory, APG, MD, 1980.