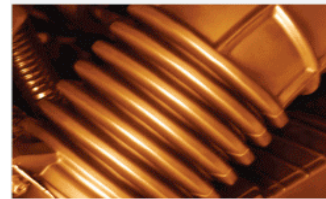


MEDUSA

VERSION 6.3

3D Design

GUIDE



All rights reserved. No part of this documentation may be reproduced in any manner (print, photocopy or other) without the written permission of CAD Schroer GmbH.

CAD Schroer GmbH has made its best effort to ensure that the information in this document is accurate and reliable, but cannot guarantee the accuracy, timeliness, reliability or completeness of any of the information contained herein. CAD Schroer GmbH will not make any warranty nor accept legal responsibility or liability of any kind for consequences resulting from errors or omissions.

Registered Trademarks of CAD Schroer GmbH:
MEDUSA, STHENO

Trademarks of CAD Schroer GmbH:
MEDUSA4, STHENO/PRO, MEDEA, MPDS4

Third-Party Products and Trademarks:
Creo, Pro/ENGINEER, Pro/DETAIL, Pro/TOOLKIT and Windchill are registered trademarks of PTC, Incorporated.

All other brand or product names are trademarks or registered trademarks of their respective owners.

September 2018

Copyright © CAD Schroer GmbH

Germany

CAD Schroer GmbH
Fritz-Peters-Str. 11
47447 Moers

Tel. +49 2841 91 84 - 0
Fax +49 2841 91 84 - 44
e-mail: info@cad-schroer.de
www.cad-schroer.de

TABLE OF CONTENTS

Introduction	11
Overview of the Modeling System	12
Accessing the 3D User Interface	14
The 3D Default Settings	15
Basics of MEDUSA4 3D	17
The Standard 3D Sheet	18
3D Commands	21
Viewboxes	22
Viewprims.	22
The Model Definition	24
The Modeling, Viewing and Reconstruction Process.	29
3D-Attributes	37
General	38
Sheet Level Attributes	39
Viewbox Attributes	46
Linkline Attributes	50
Basic Viewer Commands	53
Drawing and Sketching a View	54
Changing the Size of a View	55
Hidden Lines	56
Surface Detail.	59
Model Description Text	63
Introduction	64

Automatic Creation of Model Description Text File	64
Creating a Model Description Text on the Sheet	65
Manual Creation of Model Description Text File	66
Using Wildcards	67
Language Support	67
Configuration	68
Sweeps	69
General.	70
Slab Sweeps	71
Face Sweeps	74
Edge Sweeps	76
Basic Modeling Techniques	78
Summary of Element Types	94
Volumes of Revolution	95
Open and Closed Profiles	96
Simple Definitions.	97
Changing the Model Orientation.	106
Partial Volumes of Revolution	107
Summary of Element Types	109
Ruled Surfaces	111
General.	112
A Definition of a Simple Ruled Surface Model	113
Matching the Profiles	116
Creating Models Containing Holes.	119
Multiple Ruled Surfaces	121
Summary of Element Types	123
Slides	125
A Simple Definition	126
Defining the Start Point	129
Modeling Techniques	134
Summary of Element Types	142
Pipes	143
A Simple Definition	144

Specifying the Pipe Bend Radius.	147
Summary of Element Types.	149
Wires	151
A Simple Definition.	152
Wire Patterns in Groups.	154
Non-planar Wires.	156
Wire and Shell Models From Standard Definitions	163
Summary of Element Types.	166
Ducts	167
A Simple Model Definition	168
Smoothing the Model	171
Matching Points on Profiles	173
Controlling Model Shape with Profile Orientation.	175
Creating a Model Using Non-Planar Centerlines	176
Summary of Element Types.	179
Polygons	181
A Simple Model Definition	182
Locking a Link Line to a Profile	186
Summary of Element Types.	189
Meshed Surfaces	191
Defining a Meshed Surface Model.	192
A Meshed Surface Definition	197
The WIRE Command.	201
The MESHTOL Command	203
Surface Definition.	204
Summary of Meshed Surface Models	207
Summary of Element Types.	208
Boolean Operations	209
Boolean Operators.	210
Naming an Object	214
The MAKE Command	216
Naming a New Object	217
Examples of Simple Boolean Operations.	220

Changing the Order of Operation	226
Multiple Boolean Operations	229
Complex Boolean Operations	231
The BOOTOL Command	235
Boolean Operations on Shells	236
Summary of Element Types	240
Assemblies and Exploded Views	241
Overview	242
Instance Groups	244
Positioning and Scaling the Instanced Model	247
An Example of an Instanced Model	251
Naming Objects in Instanced Models	256
Creating an Assembly View With the Help of AVIEW	263
Exploded Views	271
Summary of Element Types	272
Example Camera	273
General Modeler Commands	281
General	282
Specifying the Chord Tolerance	283
Showing Patch Boundaries and Tile Edges	286
Naming Objects	287
Specifying the Detail Level	288
Sending Patch Data to a Model File	289
Entering Properties for Other Programs	290
Compressing a Model File	292
Viewing the Model	293
Introduction	294
Viewing Concepts	294
Changing the Type of Projection of a View	299
Sizing the Image and Setting the Degree of Perspective	300
Moving the Viewpoint	305
Moving the Aiming Point	309
Defining Oblique Views Using Viewprims	310

Defining Oblique Views Using the Creates an Oblique View Tools	313
Sectioning the Model	318
The VIEWTOL Command	334
Viewing Commands	335
The Model Viewer	343
The Model Viewer Dialog	344
Default Settings	347
Reconstruction Commands	349
General Notes	350
Specifying the Group Type	351
Specifying the Line Type and Layer	352
Specifying the Reconstructed Geometry Group Structure	355
Specifying the Reconstructed Geometry Identifier Text	356
Specifying Database Keys	357
Curve Reconstruction Accuracy	359
Specifying the Type of Reconstructed Arc	359
Controlling Line Compression	360
Highlighting the Sectioned Points on a Wire	360
The Model Validator	361
Running the Validator	362
Basic Commands	363
Validating Individual Objects	364
Syntax of Commands	365
Extracting Mass Properties	367
Introduction	368
Default Settings	369
Running the Mass Properties Program	370
Specifying Individual Objects	372
Properties	374
Black Boxes	375
Units	376
Sheet Output	377
User Defined Axes	379

Example of Mass Properties Program Output	380
Commands	382
The Shrinker	393
Running the Shrinker	394
Using the Shrinker Program in MEDUTIL	396
An Example of Shrinker Output	397
Shrinker Commands	399
Interfaces 1	401
Tools	402
Running the Converter Tools	403
Google Earth Export	405
MEDUTIL 3D Converter Programs	407
MEDVRML	408
STL	411
IGES	414
STEP	417
DXF and DWG	421
Interfaces 2	425
Tools	426
Running the Converter Tools	427
MEDUTIL 3D Converter Programs	428
MODASC, MODBIN, MODSMO - Overview	429
MODASC	429
MODBIN	431
MODSMO	433
MEDMERGE	435
The Model File Translator	437
The Structure of a Model File	438
The Structure of a MIF File	438
Description of MEDMIF	439
Running MEDMIF	441
Specification of the MIF File Format	446
MEDMIF Commands	452

The Terrain Modeler	455
Introduction	456
Constraints	463
The Model File	465
Creating the Model File using MEDUTIL	465
Creating the Model File within MEDUSA4	470
Viewing the Model	473
Smoothing the Model Surface	477
Specifying the Base Height	479
Creating Contour Lines and Cross-Sections	480
Gridding the Model Surface and Emphasizing Spot Heights	484
Naming an Object and Assigning It to a Detail Level	486
Manipulating the Model	487
Terrain Modeler Commands	489
Text Driven Modeling	493
Running the Interpolator	494
Using the Interpolator	495
Sweep Models	496
Sweep Examples	500
Meshed Surface Models	504
Meshed Surface Examples	508
Interpolator Commands	512
Model File Compression	517
Running MODCOMP	518
Using MODCOMP	518
MODCOMP Commands	520
Error Messages	521
Modeler Error Messages	522
Viewer Error Messages	536
Cross-section Views Error Messages	539
Model Validator Error Messages	541
Shrinker Error Messages	543
MODASC Error Messages	544
MODSMO Error Messages	545

Model File Translator Error Messages	546
Terrain Modeler Error Messages	547
Text Driven Modeling Error Messages	549
Model File Compression Error Messages	550
Glossary	551
Index	559

INTRODUCTION

MEDUSA4 3D is a product, which allows you to make 3D models from 2D data in a quick and easy way. This chapter introduces you into the MEDUSA4 3D modeling system.

- [Overview of the Modeling System](#) 12
- [Accessing the 3D User Interface](#) 14
- [The 3D Default Settings](#)..... 15

Overview of the Modeling System

The MEDUSA4 Modeling system creates models of real objects. A MEDUSA4 model can be created as any one of the following types:

- A faceted solid model with area, volume, and mass
- A faceted shell model with area only
- A wire model with length only

The choice of which model type to use depends on the object being modeled and the information required from the model, as well as considerations of generation speed. For example, the design for a streamlined locomotive body could initially be produced as a wire model to allow quick generation and modification, and then as a solid model for best visualization and extraction of mass properties.

The Model Generators

The system uses model generators to create a model. The model generators available create models known as:

- Sweeps (for details see [page 69](#))
- Volumes of Revolution (for details see [page 95](#))
- Ruled Surfaces (for details see [page 111](#))
- Slides (for details see [page 125](#))
- Pipes (for details see [page 143](#))
- Wires (for details see [page 151](#))
- Ducts (for details see [page 167](#))
- Polygons (for details see [page 181](#))
- Meshed Surfaces (for details see [page 191](#))

Most of the generators can create a model of any type (solid, shell, or wire). The exception is the *Wire Lobster* which only generates wire type models.

Each generator is designed to produce models of a particular form, so the choice of which one to use is governed by the characteristics of the object to be modeled. It is normal to use several generators to model complex objects completely.

The Model Definition Sheet

The model generators are driven by a two-dimensional (2D) definition drafted on a MEDUSA4 3D sheet and the model is generated when this sheet is processed. The form of the definition is similar for each generator, and consists of elements such as **profile lines**, **link lines**, **point functions**, and **prims**. The link line type defines which generator is to be used when the sheet is processed.

It is important to realize that the model definition always describes the real object exactly, while the resulting model is an approximation of it. This is because the model surface is composed of flat facets which cannot represent curved surfaces exactly, although an acceptable model is created by making the facets small enough in relation to the size of the model. This is an important aspect of modeling because as facets are reduced in size, they increase in number, and this means the model takes longer to generate. The best compromise between accuracy and processing time depends on what the model will be used for.

Details are described in "[Basics of MEDUSA4 3D](#)", "[The Model Definition](#)" on page 24.

Boolean Operations

Boolean operations can be added to the sheet to create more complex models from combinations of the generated models. The combinations possible are additions, subtractions, and intersections.

Details on this topic are described in "[Boolean Operators](#)" on page 210.

Viewing the Model

When a model has been generated it can be viewed either orthogonally, in true perspective, or in a trimetric projection. The system viewer is used for producing a permanent record of a particular view (or views) on the 3D sheet or for rapid production of alternative views on the screen using the Model Viewer tool.

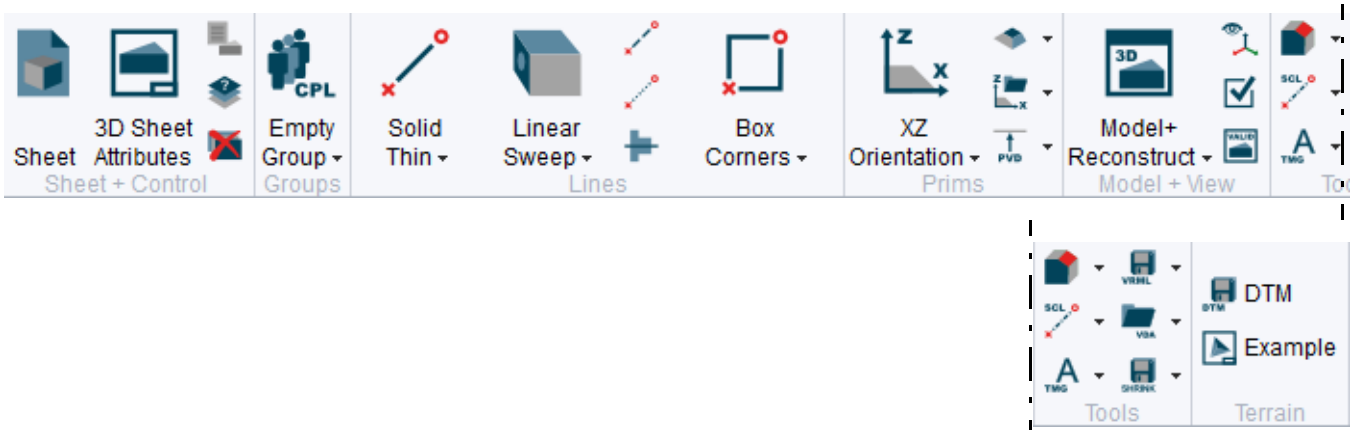
For details see "[Basic Viewer Commands](#)" on page 53 and "[The Model Viewer](#)" on page 343.

Accessing the 3D User Interface

In order to be able to use the MEDUSA4 3D Design tool you must have a license.

Choose File > Options > Licenses > Get license to enable 3D to activate the tools in the 3D tab.

Figure 1 3D-Tab



The 3D Default Settings

MEDUSA4 provides the possibility to define several default settings for the 3D product.

1. Open the Default Settings dialog via File > Default Settings.
2. Click on 3D Products to display the 3D Products as shown in the figure below.

Figure 2 3D Products Settings in the Default Settings Dialog

3D Products

Ducting Setup

Set Point Function to Ducting Profile On Off

Create Smooth Model

Set Point Funktion to Centre Line On Off

Reconstructor Setup

Delete or Keep View before Reconstructing Delete Keep

Model Viewer Setup

Shade Mode Wire Skeleton Hidden Flatoutline Flat Smooth

Toggle Visibility Axis Box Object Names

Mass Properties

Default Density

Length Units

Mass Units

Calculate for:

Model Description

Create Model Description Text Automatically

Text Style

Insert Text

The following settings are available:

Ducting Setup

Set Point Function to Ducting Profile

defines whether point functions are to be added automatically to Ducting Profile lines or not. If this function is On (default), the required point functions will be set automatically to the profile lines at the moment, when you draw the link line and you attach it to the center line. Additionally the Create Smooth Model option is enabled.

Create Smooth Model

When this option is activated, point functions 11 are created instead of 12 on the profile line, so a smooth model is created (see "Ducts", "Smoothing the Model" on page 171).

Set Point Function to Centre Line

If this function **On**, point function 12 is created automatically on the center line.

Reconstructor Setup

defines whether the current view is to be deleted before reconstruction or if it is kept.

The default setting is to **Delete** the current view.

If this function is set to **Keep**, the resulting new output of a modified model definition on the sheet overlays the previous one after running the modeler and viewer.

Model Viewer Setup

defines the default settings for the Model Viewer. For details see ["The Model Viewer"](#), ["Default Settings"](#) on page 347.

3D Mass Properties

defines the default settings for the 3D mass properties calculation. For details see ["Extracting Mass Properties"](#), ["Default Settings"](#) on page 369.

Model Description

defines the settings for creating the model description text files. For details see ["Model Description Text"](#), ["Automatic Creation of Model Description Text File"](#) on page 64.

At the bottom of the dialog you find following buttons:

OK

applies changed settings and closes the dialog.

Cancel

discards changed settings and closes the dialog.

Reset

resets the settings to the default after you have changed settings. The dialog remains open.

BASICS OF MEDUSA4 3D

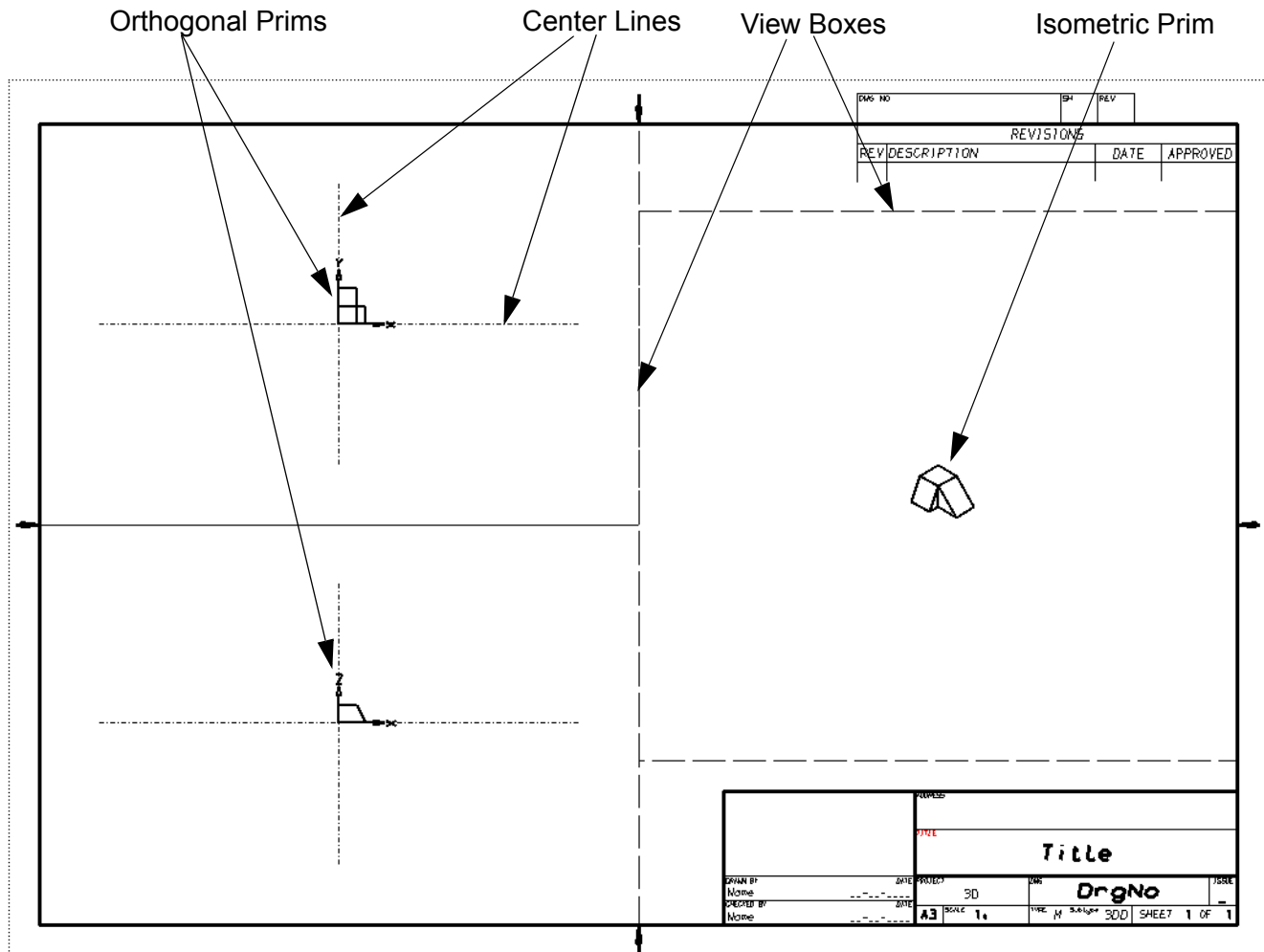
This chapter introduces you to the basic components and requirements of the 3D system.

- The Standard 3D Sheet 18
- 3D Commands 21
- Viewboxes..... 22
- Viewprims 22
- The Model Definition 24
- The Modeling, Viewing and Reconstruction Process..... 29

The Standard 3D Sheet

MEDUSA4 provides standard 3D sheets containing the required components for creating 3D models like viewboxes and viewprims, as well as a title and revision block and a sheet frame. **Figure 3** shows a standard 3D sheet of size A3, loaded using the procedure described in “[Loading a Standard 3D Sheet](#)” on page 19.

Figure 3 A Standard 3D Sheet



The Components of a 3D Sheet

A 3D sheet contains the following 3D specific components:

- 3D commands, details are described in “[3D Commands](#)” on page 21
- Viewboxes, details are described in “[Viewboxes](#)” on page 22
- Viewprims, details are described in “[Viewprims](#)” on page 22

The model definition, drawn on the 3D sheet, determines the type of model that is generated (solid, wire or shell) and its height respectively depth. The model definition consists at least of profile and link lines and point functions. For details on the model definition see [“The Model Definition” on page 24](#).

Loading a Standard 3D Sheet


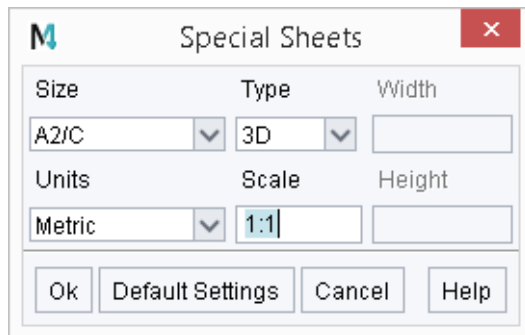
The Sheet tool  opens the Special Sheets dialog.

Figure 4 Creating a 3D Sheet Using the Default Settings



The settings shown in the figure are the default settings. In detail these are:

Size

The pulldown menu provides several standard sizes and the item *Custom*. If you select *Custom*, the input fields *Width* and *Height* are activated and you can define any values for the sheet. Additionally *Type* is deactivated, indicating that the sheet is completely empty and only its size is displayed by a dotted line.

Type

The sheet types 3D (default) and 2D are available.

Units

You can select between *Metric* or *Imperial* from a pulldown menu.

Scale

You can insert an arbitrary value for the scale of the sheet.

OK

loads the sheet and quits the dialog

Defaults

sets the entries back to the standard input. The dialog remains on the screen.

Cancel

quits the dialog.

Help

calls up the online help.

Default settings which you override are valid for all 3D sheets loaded during the rest of the session.

Designing Your Own 3D Sheet

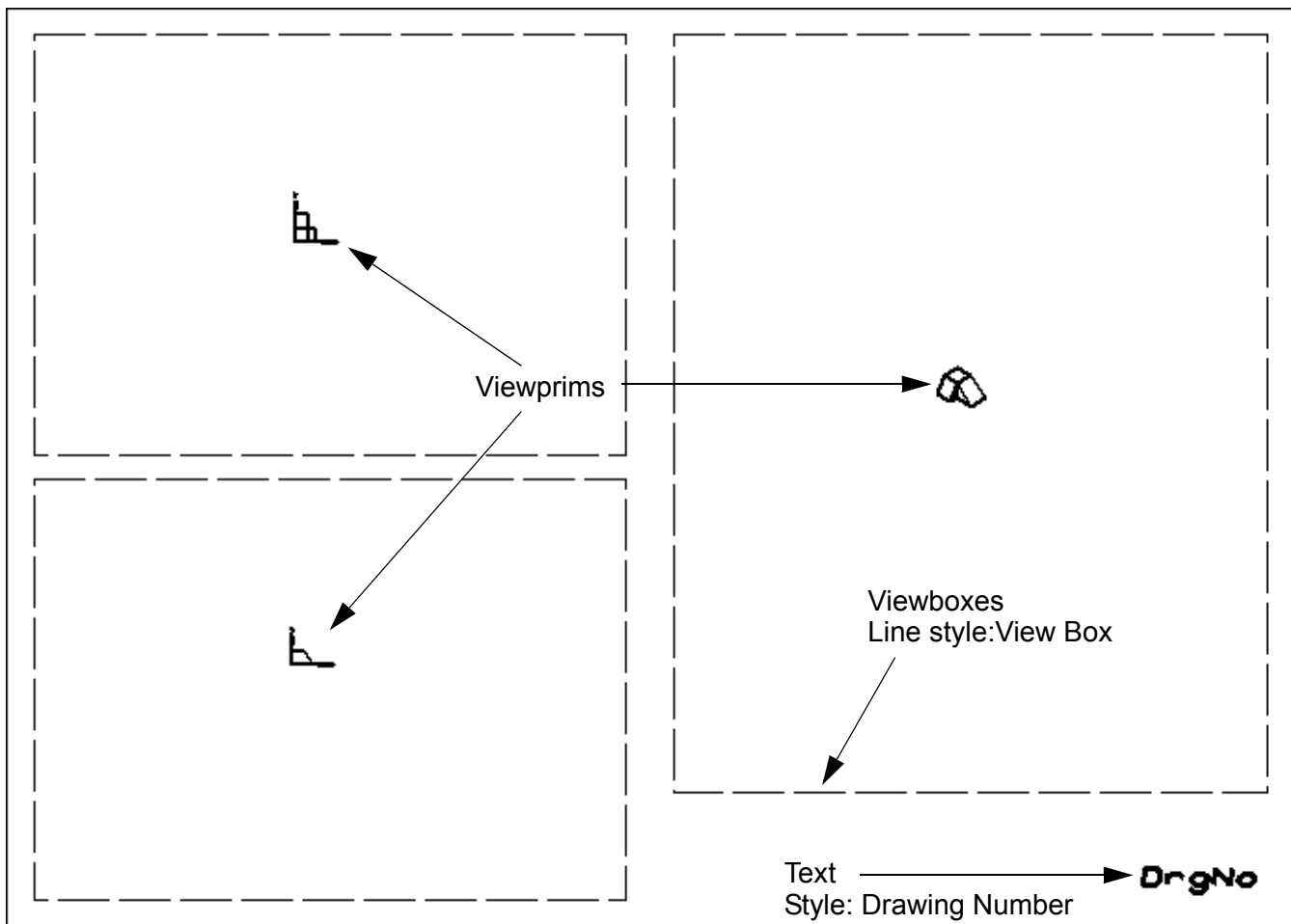
If you do not want to use one of the standard 3D sheets, you can design your own from a standard 3D sheet or even from a completely blank sheet.

Note that if you design your own 3D sheet it must contain the following components:

- Viewboxes, line style `View Box`, line type `3D View Box (LVB)`
- Viewprims (one in each viewbox)
- Text of style `Drawing Number (TSH)`

Figure 5 shows a user-defined 3D sheet.

Figure 5 A User-defined 3D Sheet



3D Commands

3D commands determine the generation and appearance of a model and can be defined as 3D attributes in the appropriate dialog or as texts on the sheet.

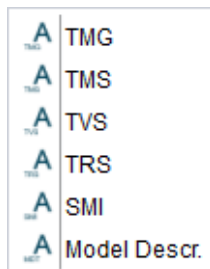
On the 3D standard sheets (see [“The Standard 3D Sheet” on page 18](#)) some basic 3D commands, like the chord tolerance CHOTOL for the modeller, are defined in the 3D Attributes dialog. The advantage is more clarity compared to a drawing which contains command texts. However the following examples use text so that it becomes clear, which commands have which effect. For detailed information on 3D attributes see chapter [“3D-Attributes” on page 37](#).

There are three kinds of 3D commands:

- Modeler commands described in [“General Modeler Commands” on page 281](#)
- Viewer commands described in [“Basic Viewer Commands” on page 53](#)
- Reconstruction commands described in [“Reconstruction Commands” on page 349](#)

For creating command texts on the sheet special tools are provided.

Figure 6 Toolset for Creating Command Texts



Each text tool uses its own style, in order that the texts can be recognized as commands by the modeller, viewer and reconstruction programs:

Tool	Style
TMG	Modeller Text ass. by Geometry
TMS	Model Specification Text for Modeler commands
TVS	View Specification Text for Viewer commands
TRS	Reconstructor Text for Reconstruction commands
SMI	3D Instance Model text
Model Descr.	3D Model Description text

3D command texts can be placed anywhere within the sheet boundaries.

Please note: If you place a command within a viewbox (see [“Viewboxes” on page 22](#)) it only applies to the view of the model shown in that viewbox. For further details, refer to [“Basic Viewer Commands” on page 53](#).

Viewboxes

Viewboxes are closed lines (normally rectangular) on the 3D sheet. They have two functions:

- Associating model definitions with viewprims and so define the position of the resulting model in the 3D coordinate system (see “Viewprims” on page 22)
- Viewing the model created in the particular view, e.g. XY-plane or isometric

In practice, viewboxes are often required to perform both functions at the same time. Each viewbox may enclose only one viewprim. All model definition items inside a viewbox, such as profiles and centerlines, are assumed to be associated with this viewprim.

The standard sheet shown in [Figure 3, “A Standard 3D Sheet” on page 18](#) contains three viewboxes but you are not limited to this number. The following points apply to viewbox construction:

- A viewbox is constructed from a single closed line of type `View Box (LVB)`.
- A maximum of 100 viewboxes can be drawn on a sheet.
- The system will determine the viewbox to be the smallest rectangle enclosing the viewbox if the viewbox is drawn as a non-rectangular shape.

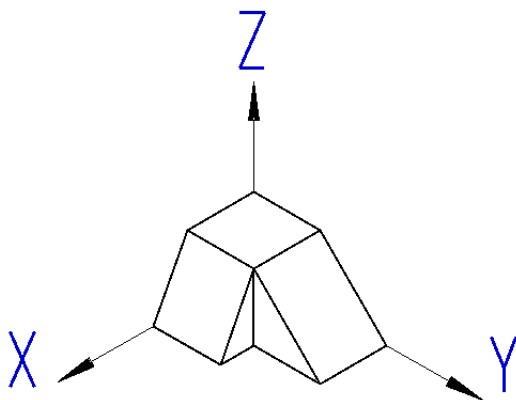
Viewprims

Viewprims are special MEDUSA4 prims predefined for use in the Modeler. Viewprims define the origin and orientation of a model profile or model view in the 3D coordinate system.

A model profile is oriented by including a viewprim of the required orientation in the same viewbox as the profile. A view of the model is automatically drawn in every viewbox in accordance with the viewprim orientation, unless suppressed by the command `NODRAW` in the viewbox.

Many viewprims are provided for use in the system, each one visually unique. However, it is important to realize that each viewprim only represents a different view of the same imaginary object known as the 3D prim. This prim is shown in [Figure 7](#).

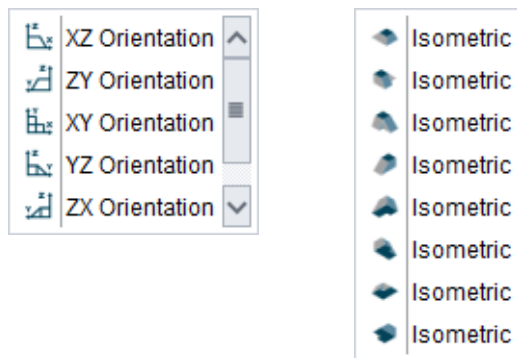
Figure 7 The 3D Prim



The 3D prim is assumed to exist at the origin of the 3D coordinate system and is oriented relative to the X, Y, and Z-axes as shown in [Figure 7](#). The corner of the prim is the datum point of the prim. The shape of the 3D prim is designed to give an immediate, unambiguous impression of model orientation from any viewing direction.

From the infinite number of possible viewing directions, fourteen are represented by viewprims in the MEDUSA4 3D system. These standard orthogonal and isometric viewprims are available in the 3D ribbon (see [Figure 1, “3D-Tab” on page 14](#)). There are six orthogonal directions and eight isometric directions shown in the following figure.

Figure 8 The Standard Orthogonal and Isometric Viewprims



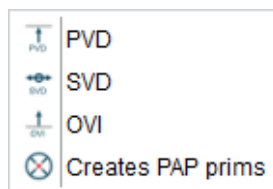
The orthogonal viewprims include arrows showing the major axis directions X, Y, and Z. This is done to emphasize that these viewprims give a view direction normal to the major coordinate planes. It does not mean that they are basically different to the isometric viewprims; all viewprims can be used for profile definition or viewing.

In 3D drafting, views of definition geometry (such as profiles and centerlines) are almost always orthogonal, with at least one major axis of the model aligned with a major axis of the 3D coordinate system. Orthogonal viewprims are therefore the natural choice for use with definition geometry in the 3D system.

The isometric viewprims are used for viewing only.

When oblique definition or viewing planes are needed it is best to use the oblique viewprims (PVD, SVD and OVI, see [Figure 9](#)). These viewprims allow model generation and viewing at any orientation desired when used in conjunction with orthogonal viewprims (see [“Defining Oblique Views Using Viewprims” on page 310](#)).

Figure 9 The Oblique Viewprims



The Model Definition

The definition of a 3D model can consist of the following elements:

- Profile Lines, see below
- Link Lines, see [page 26](#).
- Point Functions, see [page 27](#)
- Depth Text, see [page 28](#)
- instance groups reference models on other sheets and are described in [“Assemblies and Exploded Views” on page 241](#).

Profile Lines

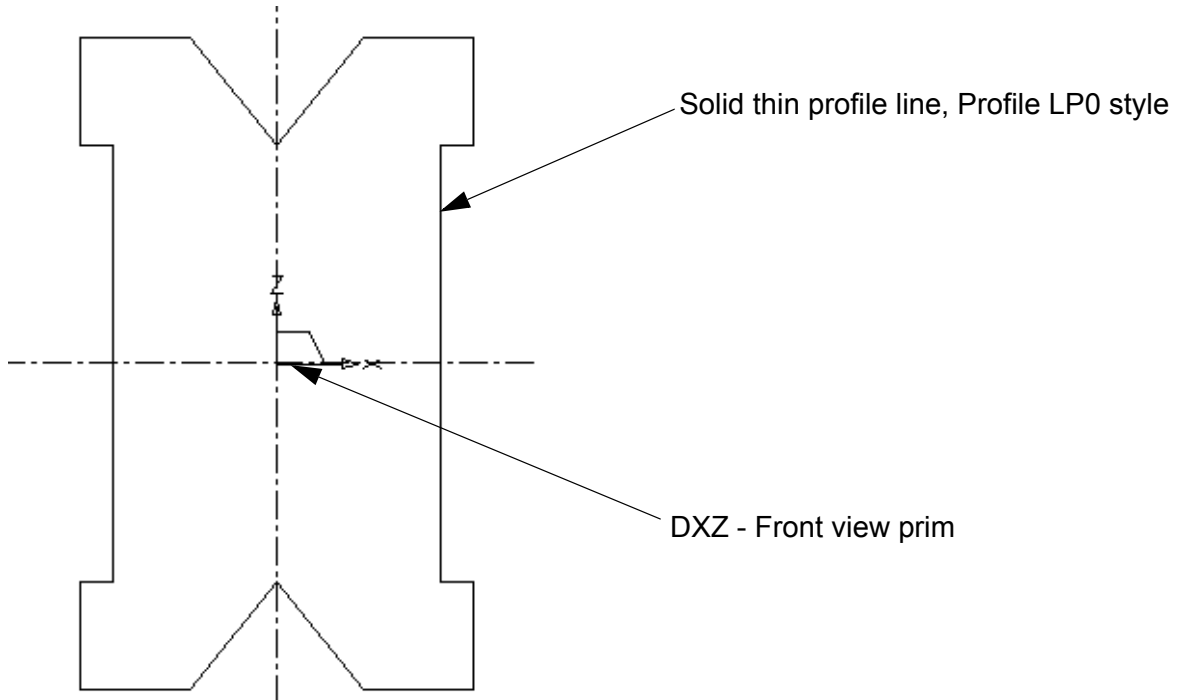
Profile lines represent the outline or cross-section of a model and are drawn in a viewbox. Profile lines are of style `Profile LP0` through `LP9` and can be chosen from the `Lines` tool group in the 3D tab.

Figure 10 The Creating Profile Lines Toolset





Figure 11 shows a profile drawn with the `Solid Thin` line tool (`Profile LP0` style).


Figure 11 Example for a Profile Line



Pipe Profile Line

The Pipe Profile tool  should be only used for the link line Pipe , otherwise when creating the model an error message like `Link point without a profile` is displayed.

3D Profile Line as Boundary Line

At the bottom of the profile line toolset (Figure 10, “The Creating Profile Lines Toolset” on page 24) the Change to profile  tool is provided used for creating a complex profile on the basis of several simple geometries.


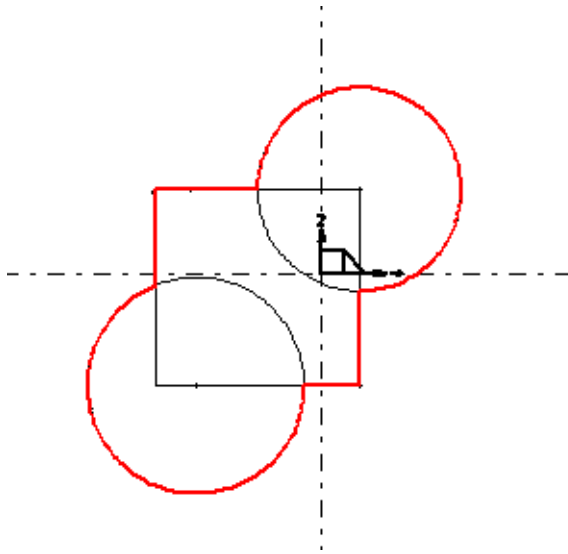
For example, if you create a geometry consisting of several simple elements like circles and rectangles and then select all elements, clicking on the Change to profile  tool creates a boundary line of style `Profile LP5`. If you then attach a link line to this line, a model is created relevant to the boundary line.

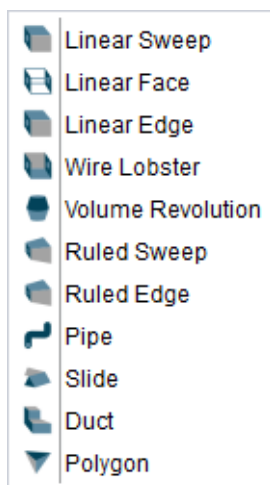
Figure 12 Example of a 3D Boundary Line



Link Lines

A link line is attached to a profile line. Its style defines which model generator will produce the model. The link line also defines the position and depth of the model in the third dimension. Typically this is done by extending the link line into another viewbox and placing point functions on it at the required positions (see “[Point Functions](#)” on page 27). But also a depth text (see “[Depth Text](#)” on page 28) can be attached to the link line. [Figure 13](#) shows the toolset for link lines.

Figure 13 The Create Link Lines Toolset




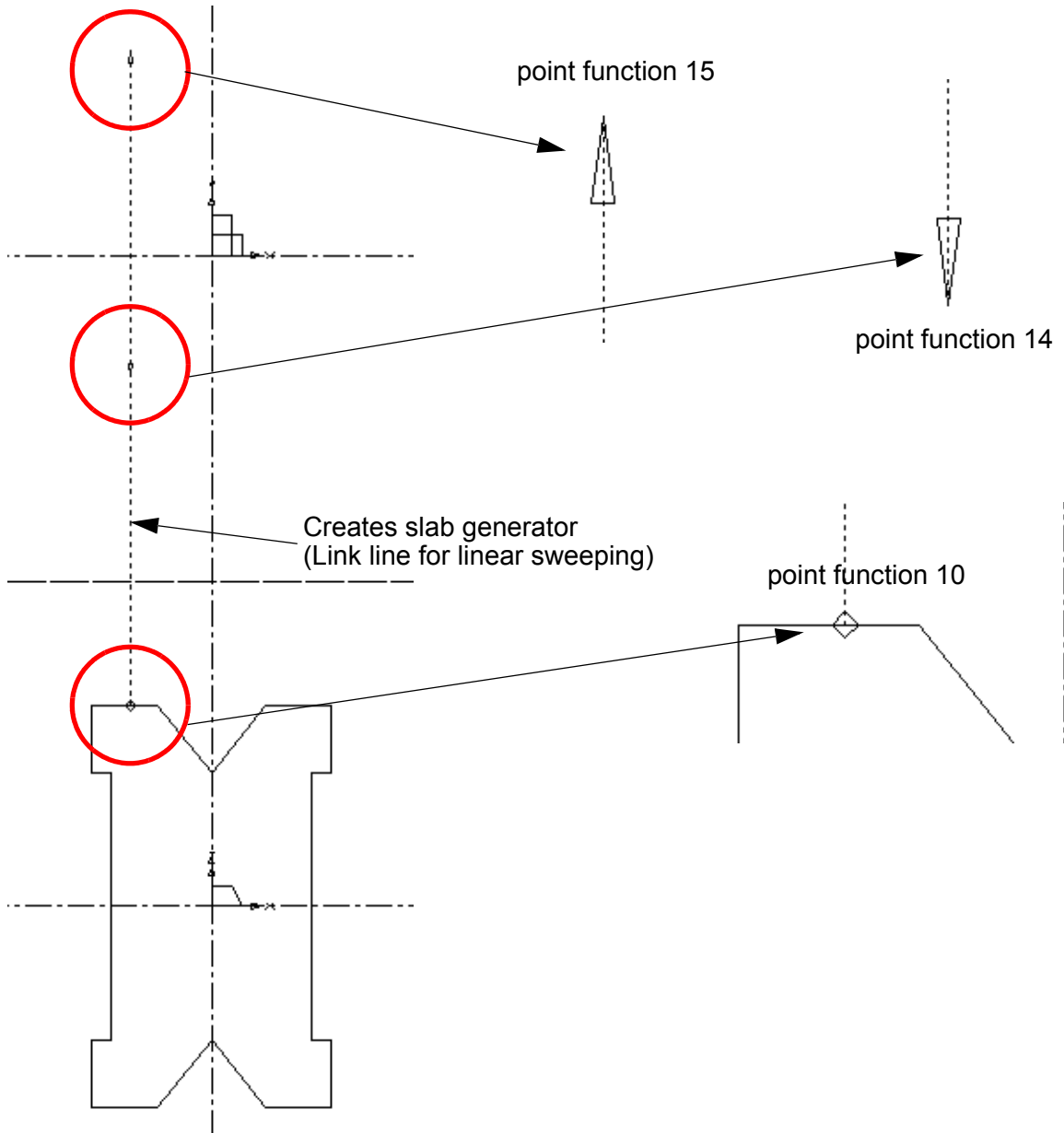
The following figure shows an example. The link line was made with the [Linear Sweep tool](#) . The point functions were added automatically to the link line while drawing it.

Figure 14 Attaching the Link Line To the Profile Line



Point Functions

Point functions are added to the link line to provide the Modeler with the following information about a model:

- A diamond point function (FUNV10) is added to the point where the link line attaches to the profile line. This defines the association between the profile line and the link line.

- The depth and position of a model is shown on the link line by using two arrowhead point functions (FUNV14 or 15). The distance between these points in the third dimension defines the depth (or height).
Alternatively, you can use depth text instead of point functions to define the depth and position of the model. Depth text is covered in the next subsection.

Please note: The required point functions are added automatically to the link line during the drawing process.

Depth Text


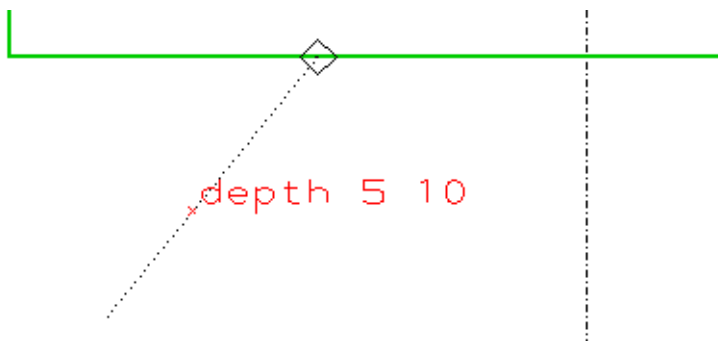
Depth text can be added to a link line instead of arrowhead point functions (FUNV14 or 15) to define the depth or height of a model. For example `DEPTH 5 10` or `HEIGHT 40 50`. The words `DEPTH` or `HEIGHT` can be used alternatively. The values are the absolute coordinates of the model in the third dimension. Depth texts are specified as a TMG-text of style `Modeller Text ass.` by `Geometry` added with the TMG tool . Figure 15 shows an example of the use of depth text.

Figure 15 Depth Text on the Link Line



Please note: If depth text is used, the link line does not need to extend into another viewbox to define the third dimension by depth text. In this case the link line is only required to specify which type of model generator is to be used.
If you extend the link line into another viewbox, the point function FUNV14 is automatically added to the endpoint of the link line. If you specify the depth by using a text, you have to replace this point function FUNV14 by the point function 0 manually.

The Modeling, Viewing and Reconstruction Process

When the model has been defined using a profile line, link line, and point functions or depth text, the final model is generated by passing the definition through the Modeler. The Viewer is then used to view the model on the 3D sheet. MEDUSA4 provides the following tools for this in the 3D tab.

Figure 16 The Modeling, Viewing and Reconstruction Process Tools

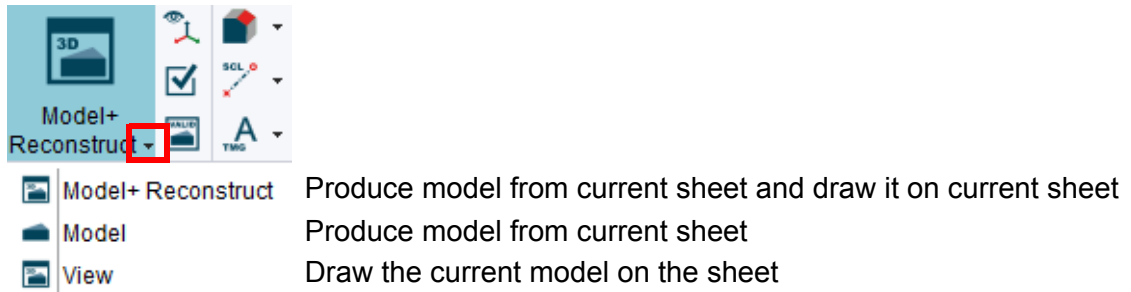
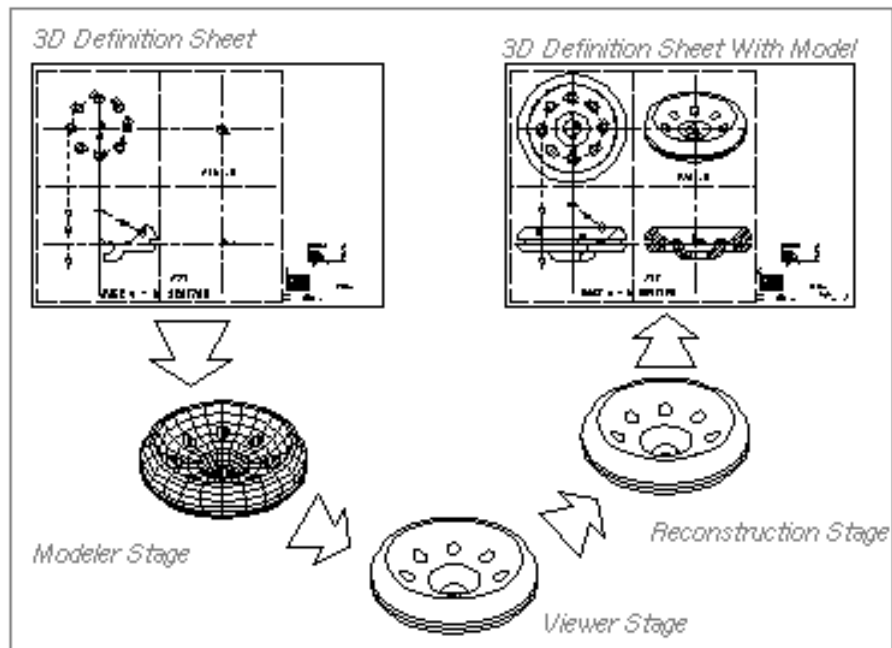



Figure 17 shows the relationship between the modeling, viewing, and reconstruction stages.

Figure 17 The Relationship between the Modeling, Viewing, and Reconstruction Stages



Example

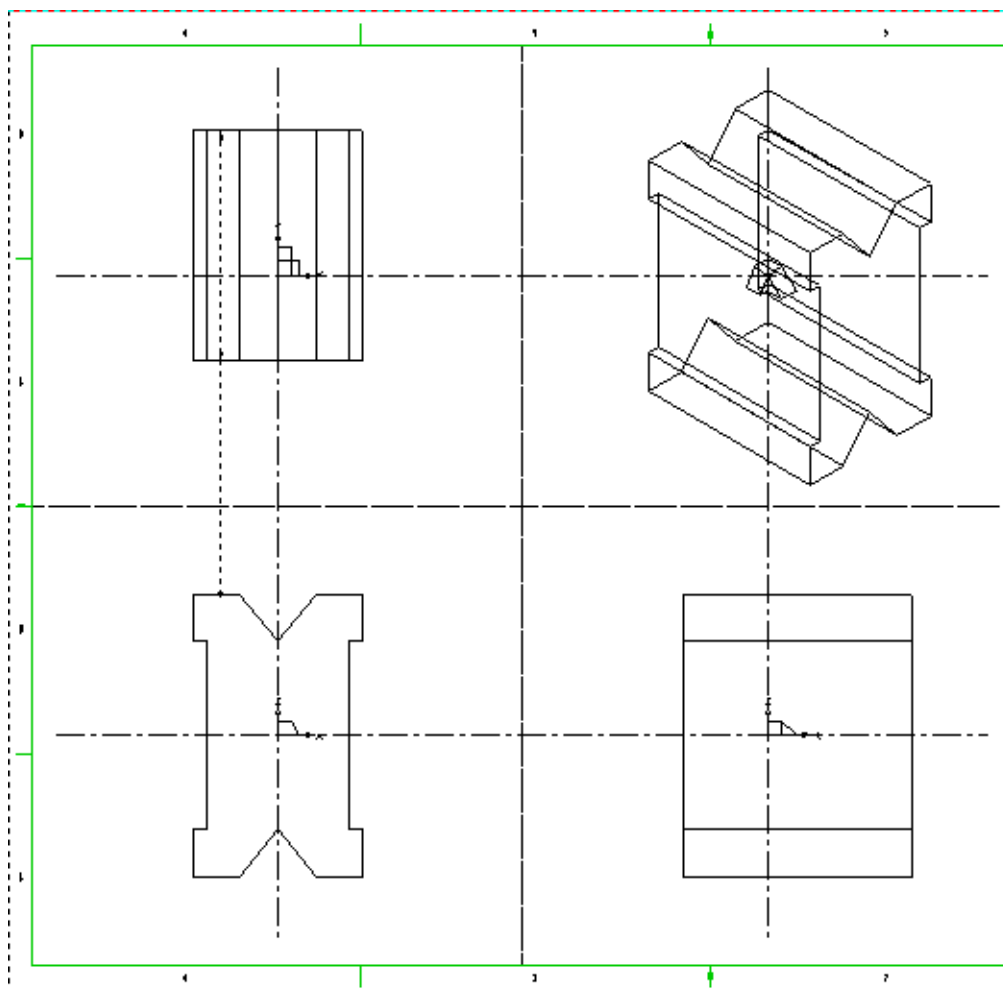
To generate **and** display the final model, select the Model + Reconstruct tool . This command executes the Modeler and Viewer in sequence, and no further input is required.

You can call up the console via File > Default Settings > General > Show Console. The console displays the following messages showing the status of the processes:

```
MEDUSA4 3D Modeling  
The level info is reset  
MEDUSA4 3D Viewer  
Returned from Viewer
```

Figure 18 shows the completed model from the definition.

Figure 18 The Completed Model



The Modeler

The Modeler takes the definition sheet and generates one 3D model. The model is stored as a collection of MEDUSA4 objects in a model file. One or more of these objects are created from every link line in the definition.

For example, a sheet with the title *pipe.she* generates the model file *pipe.mod*

The model name mentioned above is the default name, which is generated automatically. You can change it by modifying the *defaults.dat* file. The file can be found in the directory `<medusa4>\med3dui\m2d\src\defaults.dat`. By default the relevant section appears as follows:

```
-----  
-- Model Saving Expression  
-----  
-- To use a model saving expression "model_save" has to be set like this:  
-- model_save string "<drawing_number>_<sheet_issue_number>_<sheet_number>"  
-- == "@TTSH_@TTIS_@TTSN"  
-- To use the sheetname with extension .modset "use_filename",  
-- if "model_save" is not set, "use_filename" is used!  
model_save string "use_filename"
```

In this case the last line defines the model file name. If you want to change the definition of the model file name, you have to comment out or delete this line and activate for example one of the following lines; both have the same effect.


```
model_save string "<drawing_number>_<sheet_issue_number>_<sheet_number>"
```

or

```
model_save string "@TTSH_@TTIS_@TTSN"
```

Please see also in the “MEDUSA4 Administration Guide, chapter Administration, section Setting Default File Name“.

The precondition to view a 3D model is that you have already generated the model file of a definition sheet. To run the Modeler choose:

- the Model tool  or
- the Model + Reconstruct tool .

Using this tool causes the automatic generation of the model view after having finished the Modeler (see “[Example](#)” on page 30).

If you modify an element on the definition sheet, for example if you move a point of a profile line, and choose the Model tool again, the model file is overwritten, i.e. updated.

Please note: If you changed your geometry, it can happen that the contact between profile line and link line does not exist anymore. In this case adjust the link line accordingly.

If you have already generated a view of the 3D model by using the **Model + Reconstruct** and after this you modify an element on the definition sheet, the model file is updated. For default the previous view is deleted (see [“Reconstruction” on page 34](#)), however in case of changed default settings, the current view might be overlaid. Both views are visible now. To avoid this, delete the primary view first and then use the **Model + Reconstruct** tool.



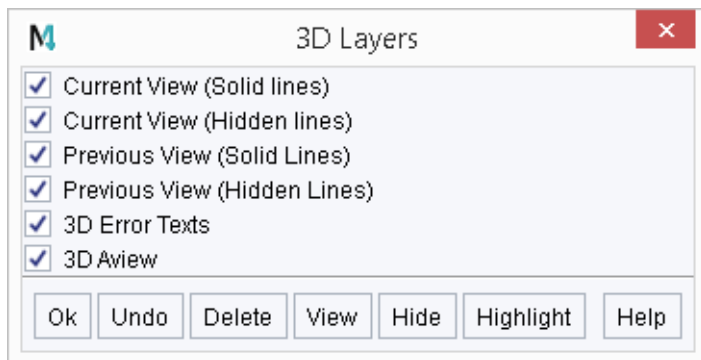


To delete views of your 3D model use the **Delete selected layers** tool . Which view elements to delete can be defined in the **3D Layer Controls** dialog. To open it, click on the **Controls 3D layers** button .

Figure 19 The 3D Layers Dialog





By default all options are activated. If you switch off all options and press **OK**, the dialog closes and the  button is deactivated. Until you activate any option in the **3D Layers** dialog again, the  button is re-activated.

The Viewer

The Modeler generates the model file, which is then passed through the Viewer which in turn generates views of the 3D model on the graphics screen. Viewer commands determine the appearance of the model. For example, hidden lines can be made visible, invisible, or dotted; the edges of facets can be made visible or invisible, and perspective views can be produced. For further details see [“Basic Viewer Commands” on page 53](#).

To run the Viewer you have following possibilities:

- The **Model** tool .
This tool assumes that a model file is already generated. The console (available via **File > Default Settings > General > Show Console**) shows following messages:

```
MEDUSA4 3D Viewer
Returned from Viewer
```
- The **Model + Reconstruct** tool .
This tool includes the automatic generation of a model file, i.e. you must not have generated a model file before (see [“Example” on page 30](#)).

The Viewer starts automatically when the Modeler has finished. The message shown above is displayed in the console:

When the Viewer has finished, the drafting system resumes.


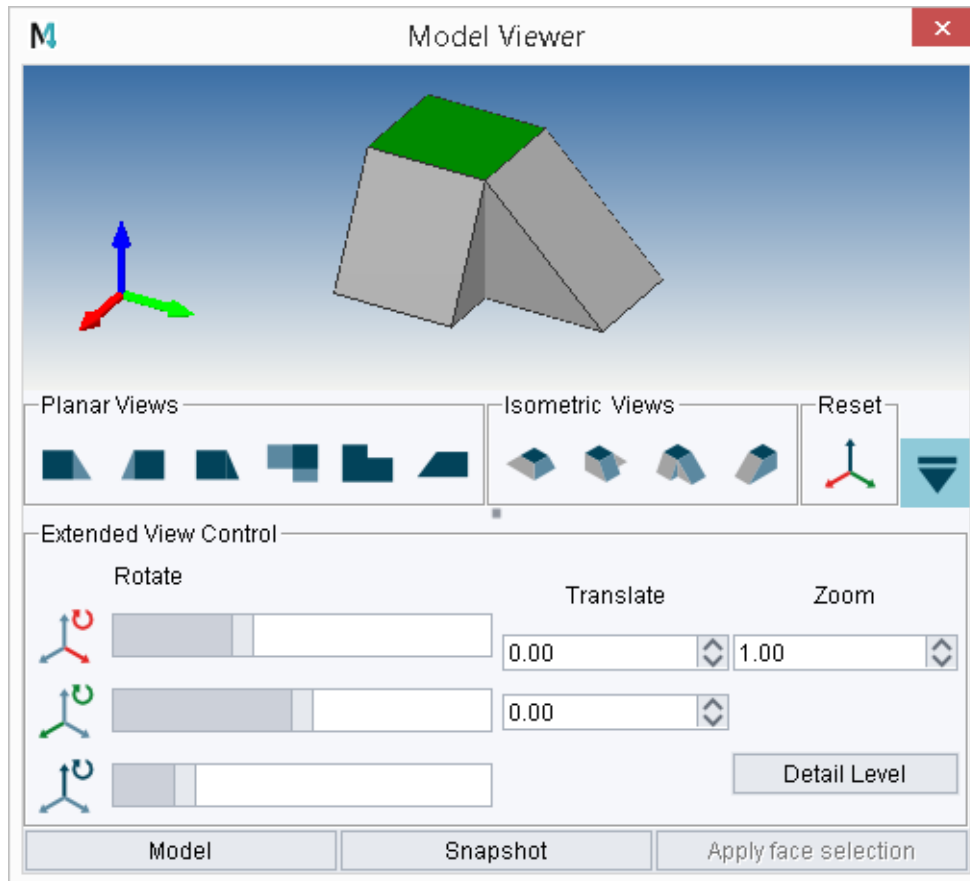
MEDUSA4 provides another tool for viewing your 3D model, the [Interactively view current model tool](#) . A click on the tool opens the [Model Viewer](#) dialog which displays an isometric view of the model.

Figure 20 **The Model Viewer Dialog**



This tool creates only a temporary view of the model which disappears, when you close the Model Viewer.

You can define and change the display of the model in the Viewer by using the [Model Viewer Setup](#) options in the [Default Settings](#) dialog (File > Default Settings > 3D Products). See also [“The 3D Default Settings”](#) on page 15.

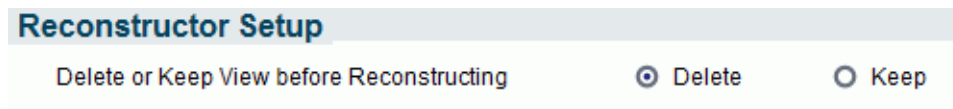
Detailed information about the use of the Model Viewer is described in chapter [“The Model Viewer”](#), [“The Model Viewer”](#) on page 343.

Reconstruction

Reconstruction starts automatically when the Viewer has finished. 3D views generated by the Viewer are translated into the data structure of the 2D drawing sheet. The sheet can then be saved to make a permanent record of the 3D views.

You can choose between two different reconstructor default settings by using the **Setup Reconstructor** options in the **Default Settings dialog** (File > Default Settings > 3D Products). See also [“The 3D Default Settings” on page 15](#).

Figure 21 Default Settings Dialog 3D Products - Reconstructor Setup

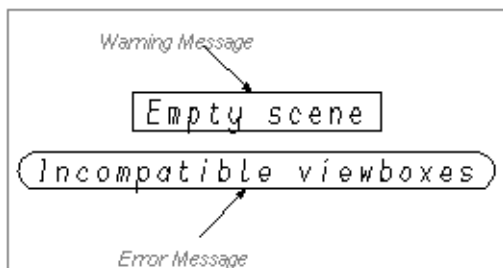


For further details on reconstruction see [“Reconstruction Commands” on page 349](#).

Error Messages

The 3D system generates error and warning messages if there is a fault with the modeling or viewing definitions. Refer to [“Error Messages” on page 521](#) for an explanation of these messages. [Figure 22](#) shows an example of each type.

Figure 22 Error and Warning Messages Generated by the 3D System



The messages are drawn on layer 99 of the sheet as text of style `3D Model Error Text`.

After correcting an error, you can delete the error messages.


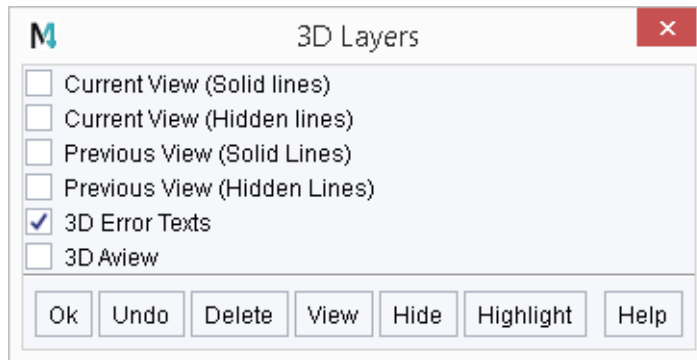
To delete all error messages on a sheet, click the **Delete Selected Layers** tool . For being able to do this, the option `3D Error Texts` in the **3D Layers** dialog has to be marked (see also section [“The Modeler” on page 31 et seqq](#)).

Figure 23 The 3D Layers Dialog

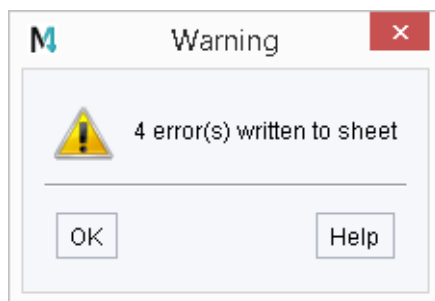


3D Syntax Checking

For saving time MEDUSA4 3D provides a program that scans the current 3D sheet for syntax errors before you generate a model. Choose the Check sheet for 3D syntax errors button to run the program.

If an error is found, an error message is written on the sheet at the location of that error. Additionally a dialog opens displaying the number of found errors.

Figure 24 Error Message



If no error was found, this is also displayed in an information dialog.



3D-ATTRIBUTES

This chapter informs you about the controlling of creation and viewing a model by adding 3D attributes.

- General 38
- Sheet Level Attributes 39
- Viewbox Attributes 46
- Linkline Attributes..... 50

General

3D attributes control the creation and viewing of a model in the same way as 3D commands (see [“3D Commands” on page 21](#)). The difference is that commands given as 3D attributes are not visible on the sheet and only displayed as properties in certain dialogs or the Dashboard.

There are three kinds of 3D attributes differing in its affects:

- Sheet Level attributes affect the whole sheet and are created and editing using a special dialog. For details see [“Sheet Level Attributes” on page 39](#).
- Viewbox attributes affect only the relevant viewbox and are displayed within the Dashboard and the properties dialog. They can be created and edited there. For details see [“Viewbox Attributes” on page 46](#).
- Linkline attributes affect the modeling and are displayed within the Dashboard and the properties dialog. They can be created and edited there. For details see [“Linkline Attributes” on page 50](#).

In the sections of this chapter the user interface regarding 3D attributes is introduced. The particular 3D commands represented by the 3D attributes are listed. The effect of these commands is not explained. This is done in the following chapters by means of detailed examples. In order to hit the appropriate texts in this book easily, behind the commands links are provided.

Sheet Level Attributes


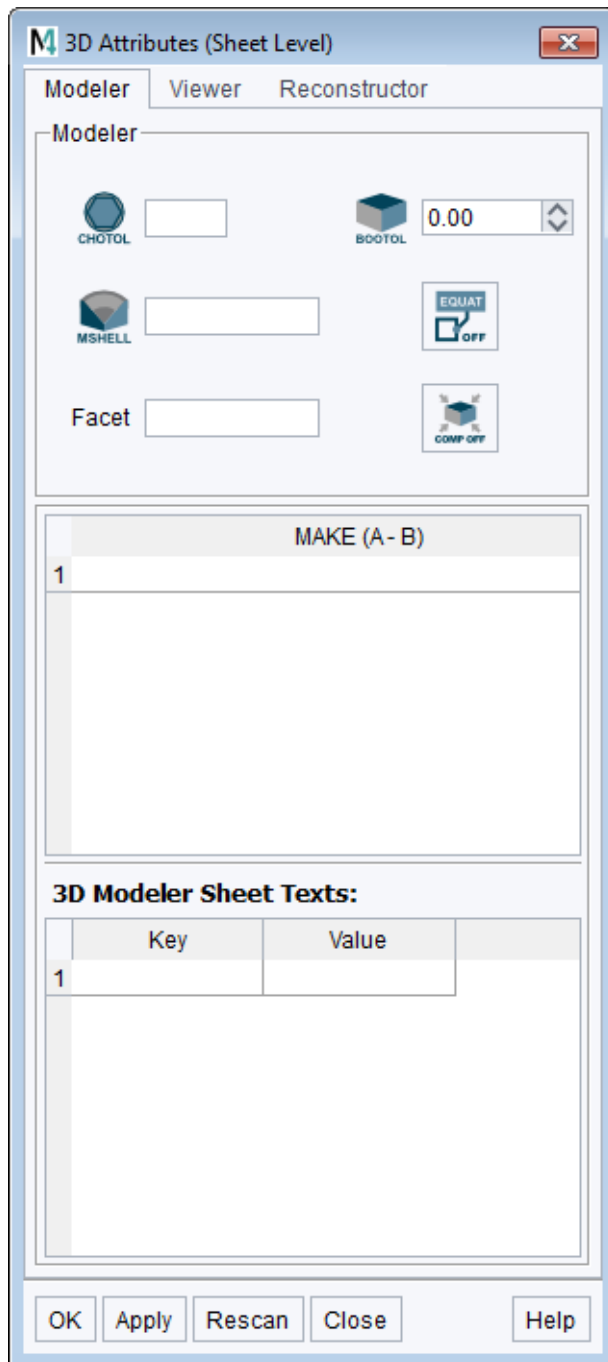

The 3D Sheet Attributes tool  opens the 3D-Attributes dialog. The button is located below the text creating tools within the 3D tab, Sheet + Control tool group (Figure 1, “3D-Tab” on page 14).

Figure 25 3D Attributes Dialog



The dialog contains three **tabs** for the creation of Modeler, Viewer and Reconstruction commands. These are described on the following pages.

The **switches** for 3D attribute commands differ according to their values which can be set. Here are some general things to consider:

- Particular commands require the setting of values in input fields, e.g. CHOTOL. If an input field is empty, the relevant attribute is not used.
- The command itself, e.g. CHOTOL, needs not to be entered.
- Some commands are toggle switches, if they can have only two values like ON/OFF or VIS/INV (visible/invisible), e.g. EQUATIONS. The button itself displays what happens when it will be pressed, so  will switch off EQUATIONS.
- Commands, which can be applied several times, are displayed in a list, for example the MAKE or TARGET command. In order to add a new line to the list move the cursor over the list field and click on the RMB to open a popup menu (see [Figure 25](#)). Insert Row adds a new line after the current line, Delete Row removes the currently selected line.

The tabs contain a **grid** which displays all 3D texts, it is read only.

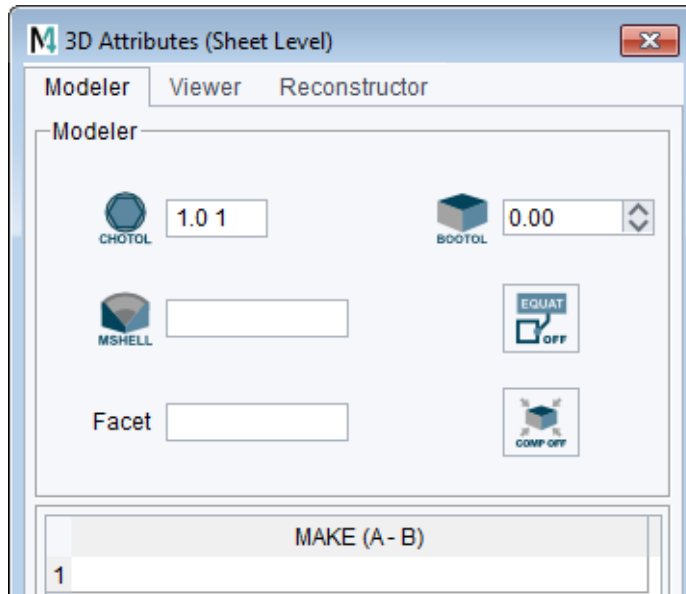
The **buttons** at the bottom work as usual. Apply and OK use the settings on the current tab, OK closes the dialog additionally. Close finishes the dialog without applying settings. Help opens the online documentation.

Please note: If you want to change the tab, press Apply before changing otherwise changed settings will be lost.

Modeler

The Modeler tab is used to specify the Modeler commands.

Figure 26 3D Attributes Dialog - Modeler



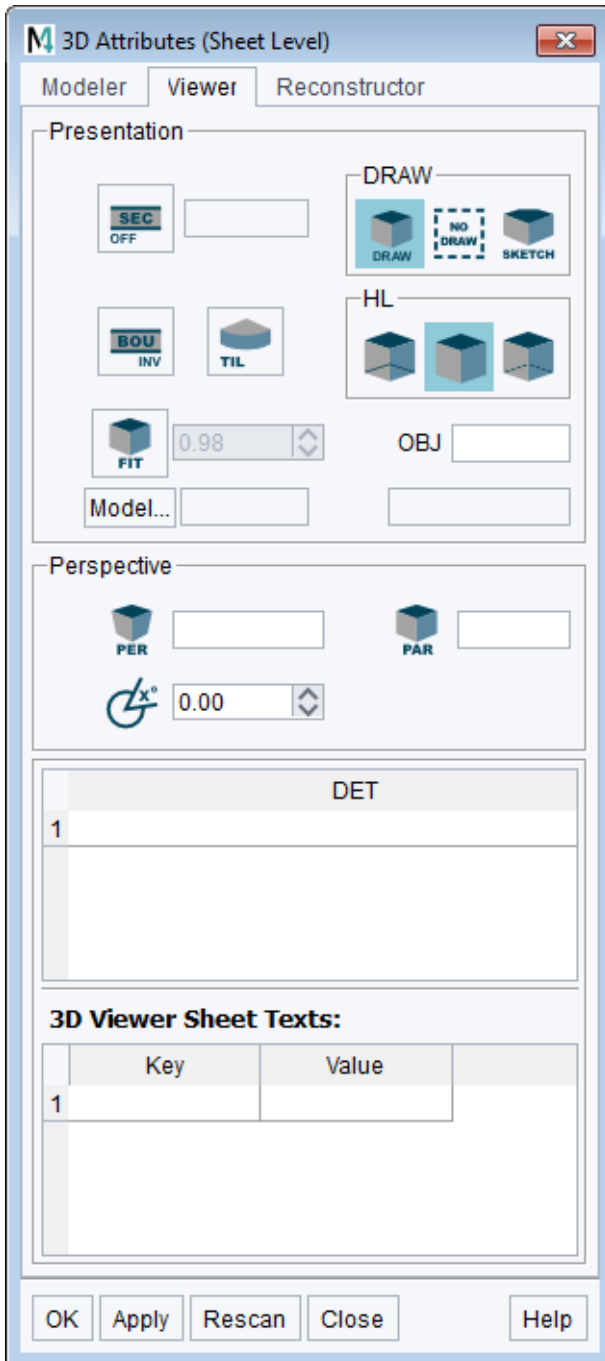
The following Modeler commands are provided:

- CHOTOL
(see [“Specifying the Chord Tolerance”](#) on page 283)
- BOOTOL
(see [“The BOOTOL Command”](#) on page 235)
- MSHELL
(see [“The MSHELL Command”](#) on page 239)
- EQUATIONS
(see [“Sending Patch Data to a Model File”](#) on page 289)
- Facette
(see [“The FACET Command”](#) on page 204)
- COMPRESSION
(see [“Compressing a Model File”](#) on page 292)
- MAKE
(see [“The MAKE Command”](#) on page 216)


Viewer

The **Viewer** tab is used to specify the Viewer commands.

Figure 27 3D Attributes Dialog - Viewer



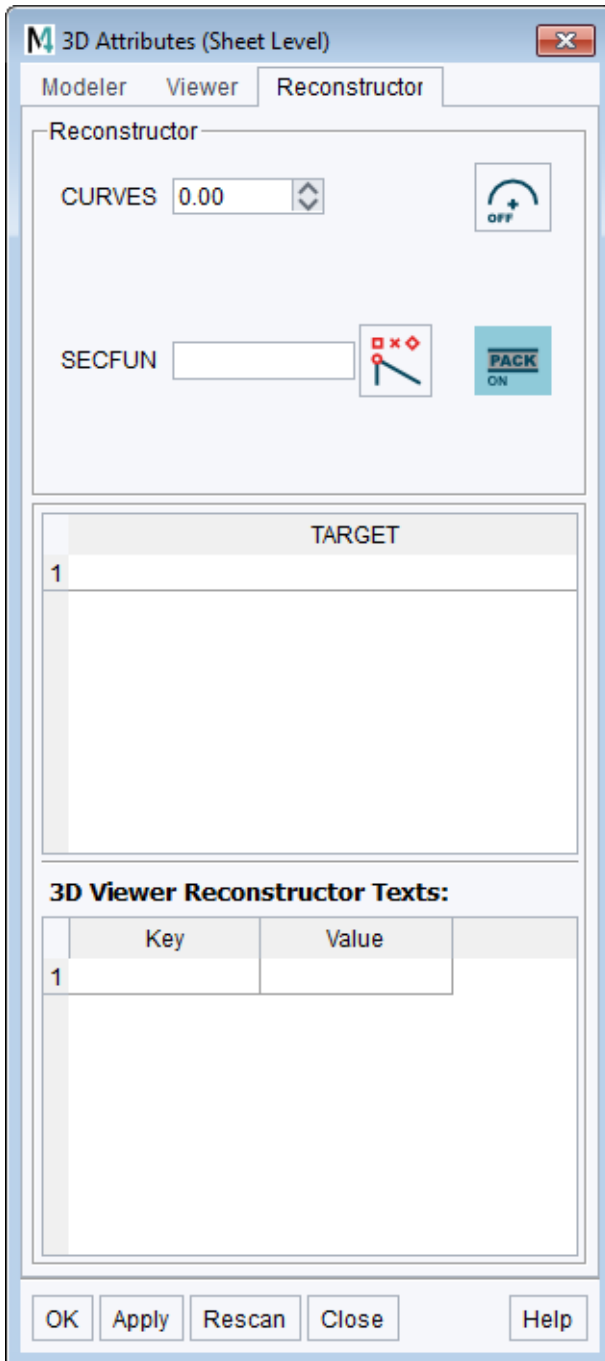
The following Viewer commands are provided:

- DRAW, NODRAW, SKETCH
(see “The Viewer Commands” on page 54 and page 336, page 337, page 340)
- BOU VIS, BOU INV
(see “Patch Boundaries” on page 59 and page 335)
- TIL VIS, TIL INV
(see “Surface Tiling” on page 60 and page 340)
- HL VIS, HL INV, HL DOT
(see “Hidden Lines” on page 56 and page 337)
- FIT
(see “Changing the Size of a View” on page 55 and page 336)
Pressing this button activates the input field and displays the default value 0.98.
- OBJ
(see “Specifying Individual Objects” on page 372 and page 338)
- Model
Once you press the Model button, the two input fields are activated. The Model command provides the following possibilities:
 - You can view an existing model file on an empty sheet.
Open a new 3D sheet.
Insert the model file name into the first field and the path to the file into the second one. Use the View tool  to view the model on the sheet.
 - You can generate the model file of a definition, which has been already created on a 3D sheet, under a specific name in a specific directory.
Insert the desired model name and path into the two input fields and press Apply, respectively OK. When you create the model file, it is created with the desired name under the inserted path.
- PER
(see “Trimetric Viewing” on page 300 and page 338)
- PAR
(see “Perspective Viewing” on page 301 and page 338)
- ANGLE
(see “Perspective Viewing” on page 301 and page 335)
- DET
(see “Validating Individual Objects” on page 364 and page 336)

Reconstructor

The Reconstruction commands are specified by using the Reconstructor tab.

Figure 28 3D Attributes Dialog - Reconstructor



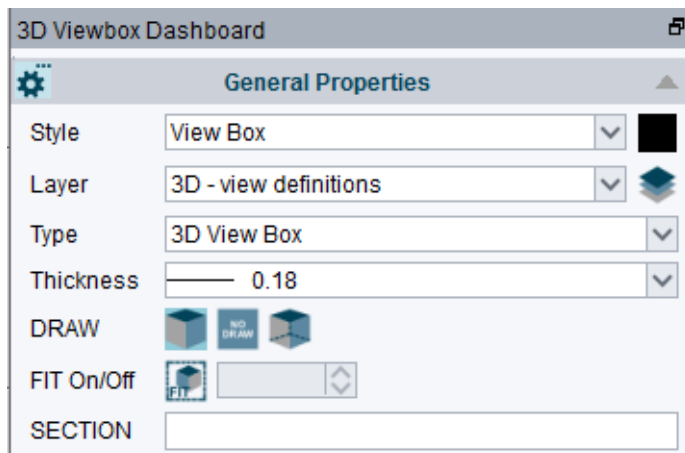
The following Reconstruction commands are provided:

- CURVES
(see “Curve Reconstruction Accuracy” on page 359)
- CIR
(see “Specifying the Type of Reconstructed Arc” on page 359)
- SECFUN
(see “Highlighting the Sectioned Points on a Wire” on page 360)
- PACK
(see “Controlling Line Compression” on page 360)
- TARGET
(see “Specifying the Line Type and Layer” on page 352).
In the text field you can enter either `name box1` or `box1`. In both cases the value of the 3D attribute name is created as `box1`.

Viewbox Attributes

Attributes associated with a viewbox are displayed in the Dashboard and in the Line Properties dialog. Click the LMB on a viewbox line, to which you want to add attributes. If it is selected, the attributes are displayed in the Dashboard.

Figure 29 Viewbox Dashboard

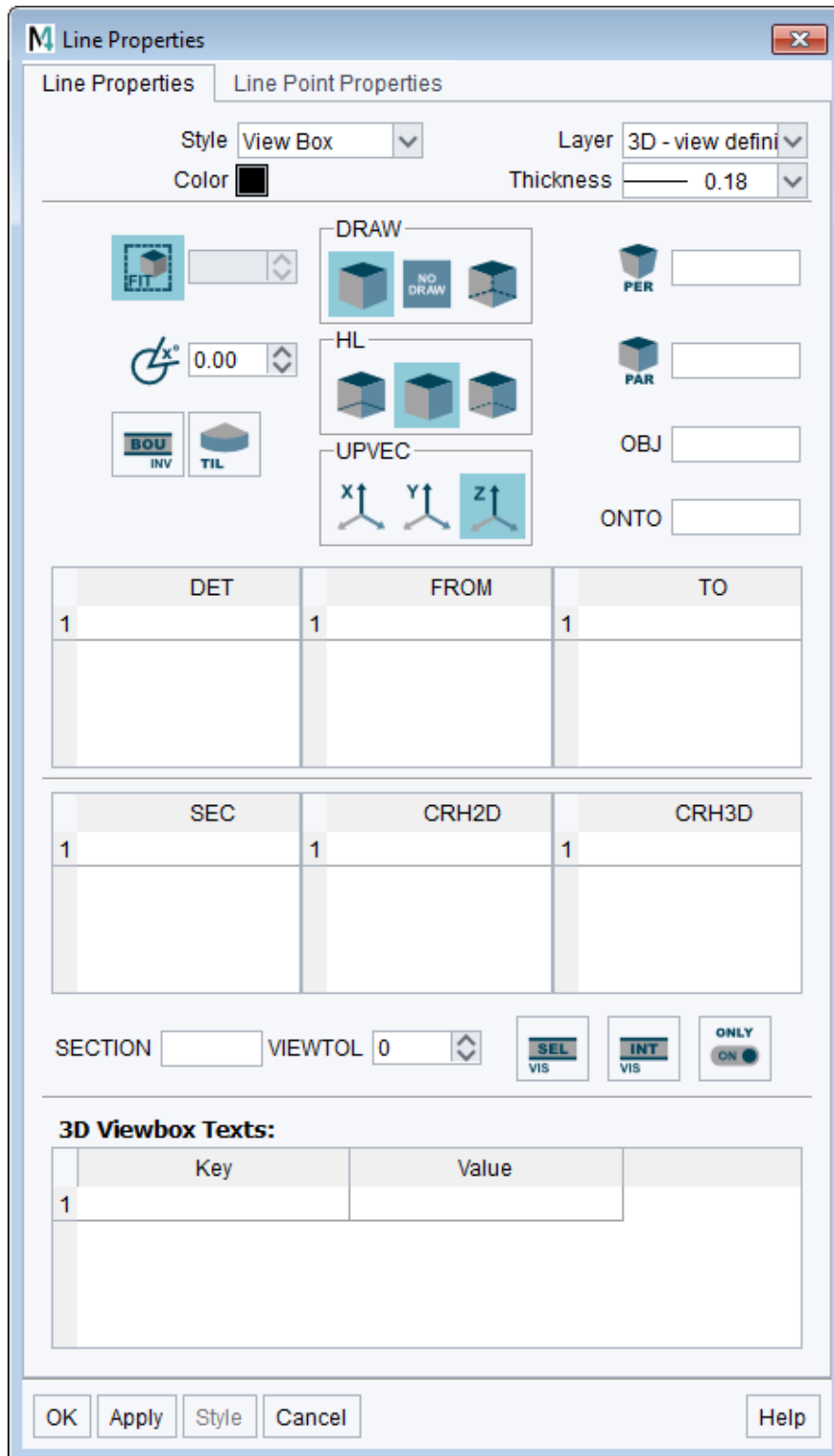


In addition to the common information like Style, Layer etc. the Dashboard provides the most frequently used 3D attributes, which are:

- DRAW, NODRAW, SKETCH
(see [“The Viewer Commands”](#) on page 54 and page 336)
- FIT
(see [“Changing the Size of a View”](#) on page 55 and page 336)
Pressing this button activates the input field and displays the default value 0.98.
- SECTION
(see [“SECTION Command”](#) on page 318 and page 340)

More attributes are available inside the Properties dialog which can be opened either by opening the popup menu and then choosing the entry Properties, or by clicking the LMB on the Properties button on the left hand side of the Dashboard heading General Properties. Both actions display the Properties dialog with the Line Properties tab.

Figure 30 Line Properties Dialog for a Viewbox Line



The following Viewer commands are provided as Viewbox attributes:

- FIT
(see “Changing the Size of a View” on page 55 and page 336)
Pressing this button activates the input field and displays the default value 0.98.
- ANGLE
(see “Rotating the Image” on page 298 and page 335)
- BOU
(see “Patch Boundaries” on page 59 and page 335)
- TIL
(see “Surface Tiling” on page 60 and page 340)
- DRAW, NODRAW, SKETCH
(see “The Viewer Commands” on page 54 and page 336, page 337, page 340)
- HL
(see “Hidden Lines” on page 56 and page 337)
- UPVEC
(see “The Upvector” on page 297 and page 341)
- PER
(see “Changing the Type of Projection of a View” on page 299 and page 338)
- PAR
(see “Trimetric Viewing” on page 300 and page 338)
- OBJ
(see “Validating Individual Objects” on page 364 and page 338)
- ONTO
(see “The ONTO Point” on page 297 and page 338)
- DET
(see “Validating Individual Objects” on page 364 and page 336)
- FROM
(see “Moving the Viewpoint” on page 305 and page 337)
- TO
(see “Moving the Aiming Point” on page 309 and page 341)

The following commands are provided as attributes specially for sections:

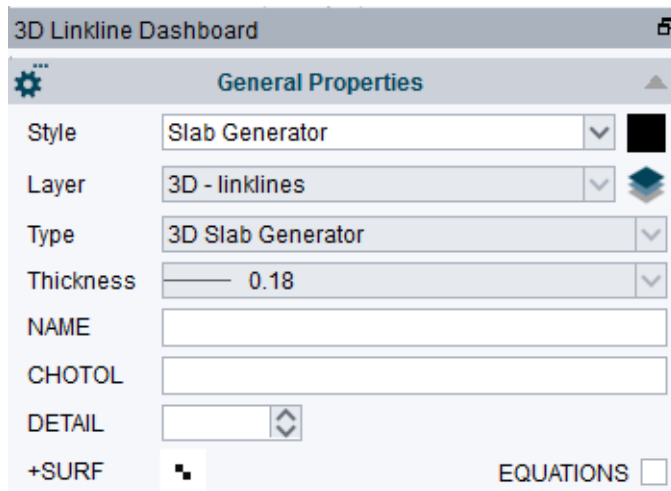
- SEC
(see “Controlling the Sectioning” on page 326 and page 339)
- SECTION
(see “SECTION Command” on page 318 and page 340)
- CRH2D
(see “Crosshatched Sections in 2D Space” on page 322 and page 335)
- CRH3D
(see “Crosshatched Sections in 3D Space” on page 323 and page 335)

- VIEWTOL
(see “The VIEWTOL Command” on page 334)
- SEL
(see “SEL” on page 340)
- INT
(see “Intersection Lines” on page 60 and page 337)
- SEC ONLY
(see “Controlling the Sectioning” on page 326 and the following and page 339)

Linkline Attributes

Attributes associated with a link line are displayed in the Dashboard and the Properties dialog. Click the LMB on a link line to select it in order to assign 3D attributes to it. Once the link line is selected the attributes are displayed in the Dashboard.

Figure 31 Link Line Dashboard



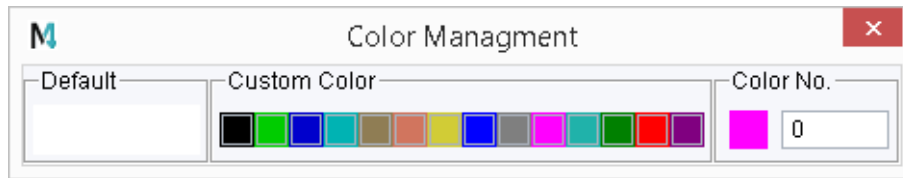
In addition to the common information as Style, Layer etc. the Dashboard provides the most frequently used 3D attributes, which are:

- NAME
(see [“Naming an Object”](#) on page 214 and [“Naming a New Object”](#) on page 217)
- CHOTOL
(see [“Specifying the Chord Tolerance”](#) on page 283)
- DETAIL
(see [“Specifying the Detail Level”](#) on page 288 and [“Assigning a Detail Level to an Object”](#) on page 486)
- EQUATIONS
(see [“Sending Patch Data to a Model File”](#) on page 289)
- +SURF
(see [“Entering Properties for Other Programs”](#) on page 290)

The +SURF button  opens the Color Management dialog. Two versions exist:

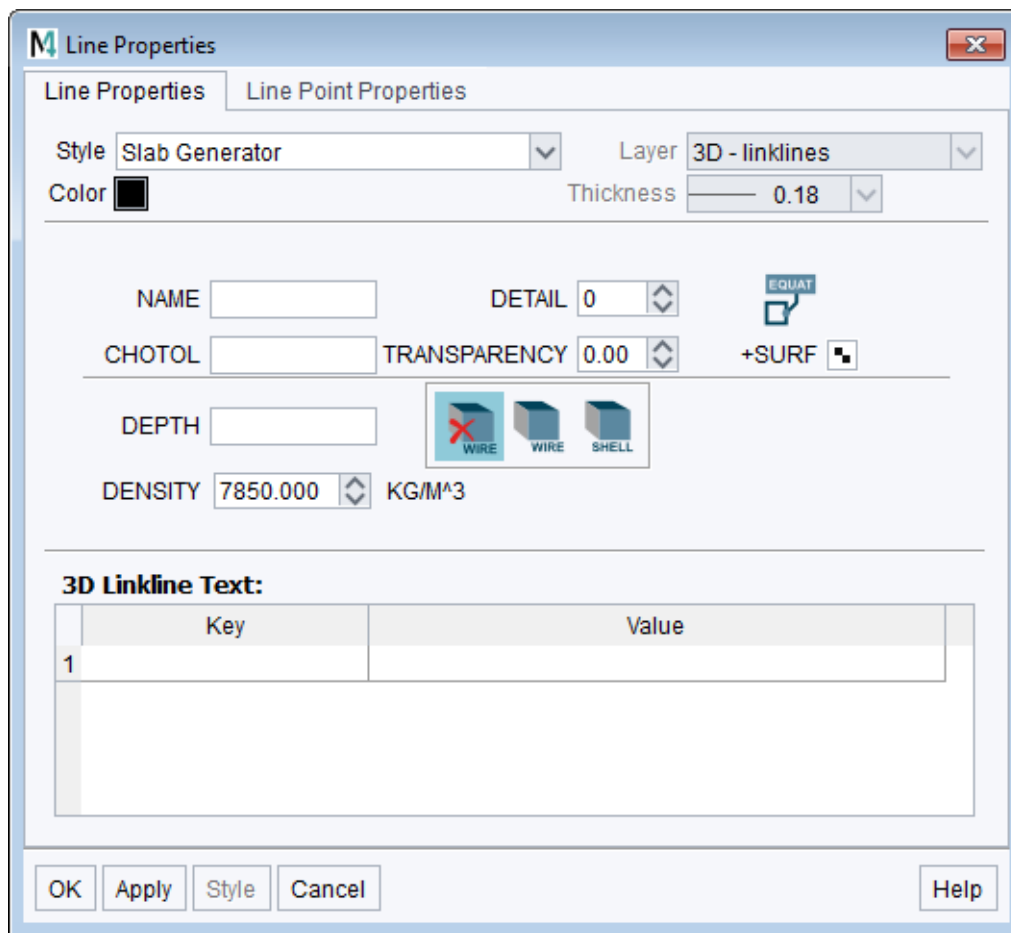
- For a standard project, e.g. the *master_project* in your installation path, the following dialog is shown:

Figure 32 Color Management Dialog - Standard Master Project



More attributes are available inside the Properties dialog which can be opened either by clicking the RMB to open the popup menu and then choosing the entry Properties, or by clicking the LMB on the Properties button on the left hand side of the Dashboard heading General Properties. Both actions display the Properties dialog with the Line Properties tab.

Figure 33 Line Properties Dialog of a Link Line



The dialog contains all attributes as explained for the dashboard. Additionally the following parameter is provided:

TRANSPARENCY

It sets the transparency for the model to a value between 0 and 1.0. 0 means no transparency and is default.

Please note: The transparency setting is not visible in the Model Viewer. This value is used for other applications only like MPDS4 Review or MPDS4.

The attributes of the lower section of the dialog differ depending on the style of the selected link line. [Figure 33](#) above shows the properties for a link line of style *Slab Generator*. The following figures show the attributes for a link line of style *Line Generator* and *Volume Revolution Generator*.

Figure 34 Line Properties Dialog - Link Line Style Line Generator

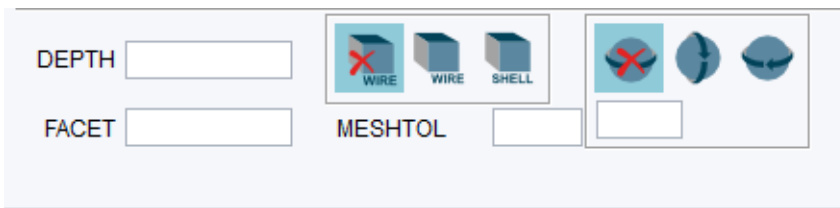


Figure 35 Line Properties Dialog - Link Line Style Volume Revolution Generator



In the lower section of the *Properties* dialog following commands are possible. Which of them are displayed depends on the selected link line.

- DEPTH
(see [“Depth Text”](#) on page 28)
- WIRE, SHELL
(see [“Wire and Shell Models From Standard Definitions”](#) on page 163)
- LON, LAT
(see [“Naming Latitudes and Longitudes”](#) on page 196)
- FACET
(see [“The FACET Command”](#) on page 204)
- MESHTOL
(see [“The MESHTOL Command”](#) on page 203)
- ANGLE
(see [“Changing the Model Orientation”](#) on page 106 and the following)
- DENSITY
(see [“Units”](#) on page 376)

BASIC VIEWER COMMANDS

This chapter introduces the basic Viewer commands that can be used to view a 3D model. It is helpful to be familiar with these basic commands before starting to use the Modeling system.

Viewer commands can be used in the Sheet Viewer. This chapter is only concerned with producing orthogonal and isometric views. The chapter [“Viewing the Model” on page 293](#) provides a full explanation of the Sheet Viewer commands.

Viewer commands are added as 3D attributes or specified by using a TVS-text of style `View Specification Text` placed on the sheet.

Commands placed anywhere within the boundaries of a sheet (outside viewboxes) affect all the model views on that sheet. Commands placed within a viewbox only affect the view in that viewbox.

Please note: Instead of the procedure described in the following in which commands are given by texts, you can give the same commands by adding attributes (see [“3D-Attributes” on page 37](#))

- [Drawing and Sketching a View](#) 54
- [Changing the Size of a View](#) 55
- [Hidden Lines](#) 56
- [Surface Detail](#) 59

Drawing and Sketching a View

The Viewer Commands

The following Viewer commands affect both the appearance of model views and the time needed by the Viewer to produce views:

- **SKETCH** (abbreviation **SKE**)

The **SKETCH** command produces a wireline view of a model. A **SKETCH** view is generated more quickly than a **DRAW** view but, as with all wireline representations, can be ambiguous. Use this command to verify a model quickly.

- **DRAW**

The **DRAW** command produces a more realistic view than **SKETCH** by giving control over how hidden lines are treated. These may be made invisible (**HL INV**); dotted (**HL DOT**); or visible (**HL VIS**). **HL INV** is the default mode.

- **NODRAW**

The **NODRAW** command suppresses the view of a model within a viewbox. This command is useful in a viewbox that contains a model definition where you do not want to obscure the definition by a view. The use of this command also speeds up the viewing process as it reduces the number of views that are produced.

Setting the Commands

You can add commands by using attributes (see [“3D-Attributes” on page 37](#)) or using the text tools (see [Figure 6, “Toolset for Creating Command Texts”](#)).

The Sheet Mode

Commands, added as 3D attributes on sheet level or as text on the sheet, affect any view of a model on the whole sheet.

The Viewbox Mode

Commands, added as 3D attributes to a viewbox or placed as text in a viewbox, affect the view of a model within this viewbox.

Modification of Commands

If you gave commands by adding attributes, you can change these commands, by modifying the attributes within the **3D Attributes** dialog (applies to sheet level attributes) or within the **Line Properties** dialog, respectively dashboard (applies to Viewbox attributes).

If you placed the commands as text on the sheet, you can edit these commands by selecting them and modifying the displayed text within the text input field.

Changing the Size of a View

The `FIT` command changes the normal size of a view by fitting it into a viewbox. It is, in effect, a controlled zoom. By placing this command within a viewbox, the view of a model is magnified to within 2% of the viewbox boundary; i.e. the default value of the viewer command is 0.98.

By adding a value to the `FIT` command, you can specify a magnification factor to vary the degree of fit.

For example, the command `FIT 0.75` magnifies the full viewbox fit by the factor 0,75 in each direction and so reduces the apparent size of the model.

Add the `FIT` command as attribute or as text to the viewbox.

There are many examples of the use of the `FIT` command in the illustrations in this manual.

Hidden Lines

Hidden lines are those lines that are normally obscured when viewing a solid object. When using the Viewer command `DRAW`, the hidden lines of an object can be specified to be:

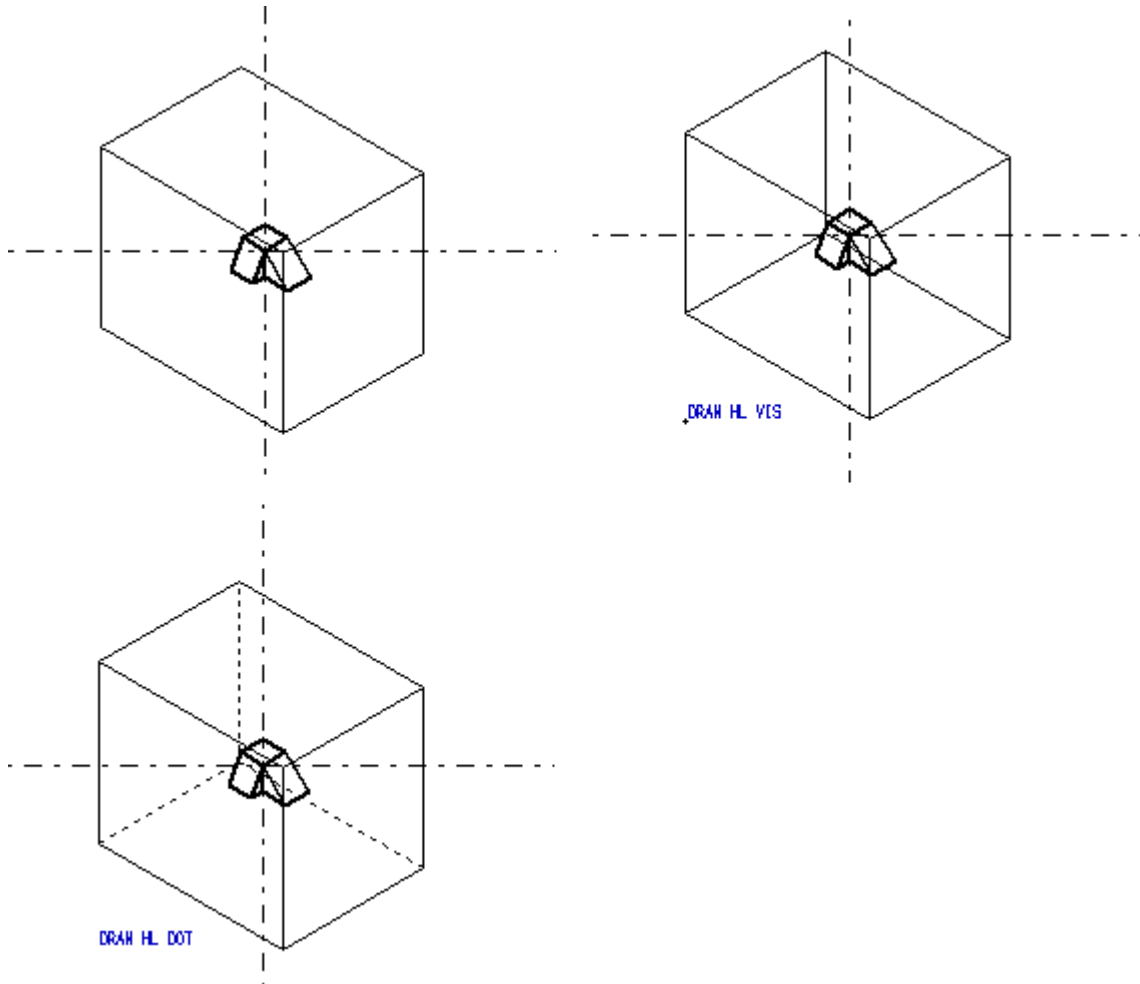
- Visible
To make hidden lines visible, use the command:
`HL VIS`
- Invisible (system default)
To make hidden lines invisible, use the command:
`HL INV`
- Dotted
To make hidden lines dotted, use the command:
`HL DOT`

Please note: Most of the example drawings shown in this User Guide are based on the default setting `HL VIS`.

Add the desired command to the `DRAW` command by editing within the 3D Text commands dialog as described before. For example: `DRAW HL VIS`

[Figure 36](#) gives an example of a view of a model with hidden lines visible, invisible, and dotted.

Figure 36 Hidden Lines in a View of a Solid Model



If a `DRAW` command already exists within a viewbox and a view has been created, you can edit the command directly in the sheet and generate a new model view.


1. Select the relevant model view and delete it.
2. Select the 3D command in the viewbox.
The Text Dashboard is displayed with the selected text.

Figure 37 Example of a Text Field with a Selected View Command



3. Edit the text as required, for example: `DRAW HL VIS`.

The text on the sheet is modified promptly.

4. Run the Viewer by using the View tool .

The new model view is created relevant to the specified DRAW command.

Surface Detail

Curved surfaces on a model are represented by a number of smaller curved surfaces called patches. Each patch is divided into smaller flat faces (planar polygons) called tiles.

Certain Viewer commands determine how much of this surface detail is shown on a model. This section describes the commands that determine the visibility of:

- Patch boundaries
- Surface tiling
- Intersection lines

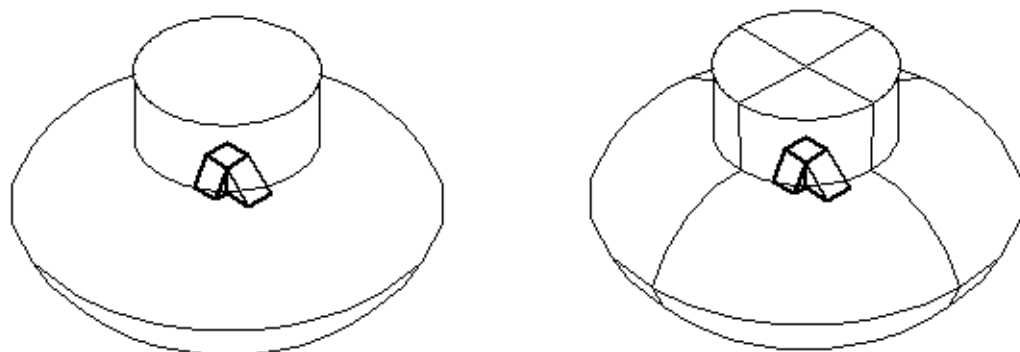
Please note: The Modeler command `FACET` can be used to control the number of tiles in a patch on the surface of a **meshed surface model**. See [“Meshed Surfaces” on page 191](#).

Patch Boundaries

The Viewer command `BOU VIS` specifies visible boundary lines on a model. The command `BOU INV` returns to the default of invisible boundary lines.

[Figure 38](#) shows a model with invisible and visible patch boundaries.

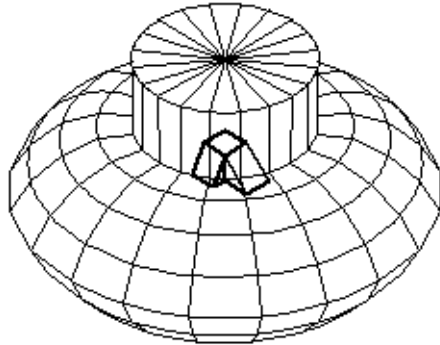
Figure 38 Patch Boundaries on a Model



Surface Tiling

The command `TIL VIS` makes the surface tiles of a model visible. The command `TIL INV` switches off visible surface tiling and is the default. [Figure 39](#) shows the result of making tiles visible on a model.

Figure 39 Surface Tiles On a Model



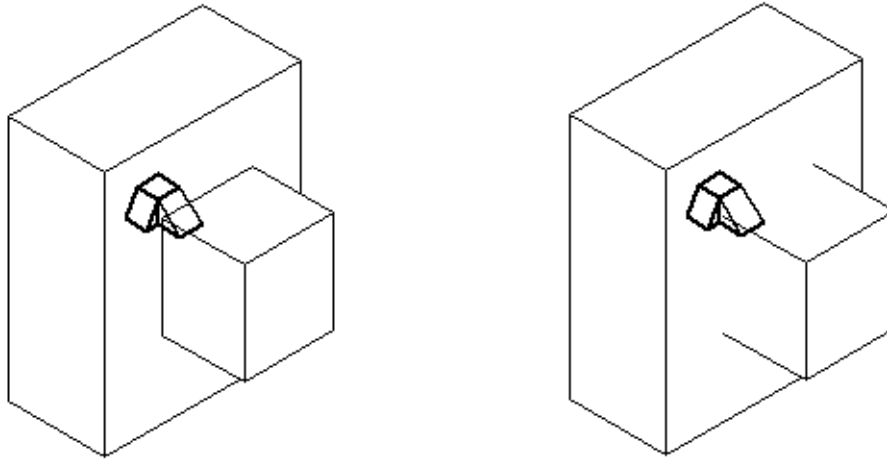
Intersection Lines

If a model contains objects that intersect each other, the intersection lines are visible (by default). The command `INT INV` makes the intersection lines invisible on a model. The command `INT VIS` returns to the system default.

Please note: Intersection lines are never shown if using the Viewer command `SKETCH`.

Figure 40 shows a model with visible and invisible intersection lines.

Figure 40 Intersection Lines On a Model



Please note: Intersection lines caused by Boolean operations are treated as edges and are not affected by this command. For further details of models created in this way “[Boolean Operations](#)” on page 209.

MODEL DESCRIPTION TEXT

- Introduction 64
- Automatic Creation of Model Description Text File 64
- Manual Creation of Model Description Text File 66
- Creating a Model Description Text on the Sheet..... 65
- Using Wildcards 67
- Language Support 67
- Configuration..... 68

Introduction

Model Description Text is used to add a description file to a model file. It gets the name *<modelfile>.txt*. If equipment will be loaded in MPDS4, the text is displayed in the Equipment Load Dialog.

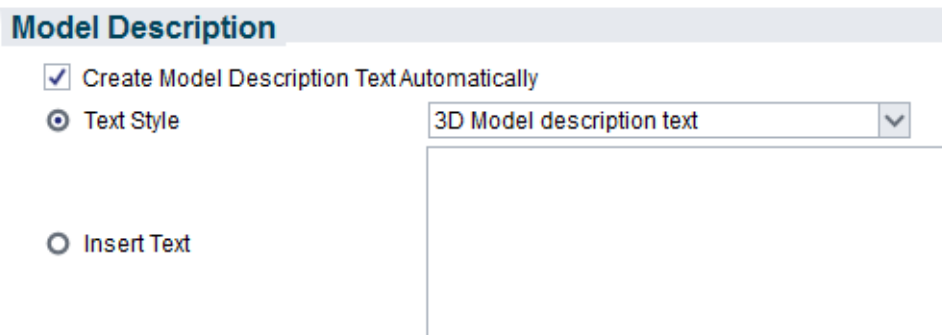
There are two possibilities to create Model Description Text.

- Automatic (see ["Automatic Creation of Model Description Text File"](#))
- Manual (see ["Manual Creation of Model Description Text File"](#) on page 66)

Automatic Creation of Model Description Text File

By default Model Description Text is created automatically when modeling the current sheet. You can see the current settings in the 3D Default Settings.

Figure 41 Dialog Default Settings > 3D Products > Model Description



Create Model Description Text automatically

is selected by default. In this case a model description file *<modelfile>.txt* will be created whenever a model file *<modelfile>.mod* is generated.

Text Style

is described in ["Option Text Style"](#) on page 65.

Insert Text

is described in ["Option Insert Text"](#) on page 65.

You can define the default for the Model Description Text in the file *defaults.dat* (see ["Configuration"](#) on page 68).

Option Text Style

If the option `Text Style` is selected, you can choose a text style with the help of the combo box. The `3D Model Description text` is the default style.

If you create a text with this style onto the 3D sheet and model the sheet, the text is read from the sheet and written to the model text file.

Please note: Only the first found text will be considered!

Option Insert Text

If the option `Insert Text` is selected, the text input field will be enabled. Whenever modeling a sheet the text added here will be used.

You can use special wildcards inside the text, e.g. for the content of the sheet title, see [“Using Wildcards” on page 67](#).

Creating a Model Description Text on the Sheet


For automatic creation of Model Description Texts at least one text has to exist on the sheet which has the style defined in the 3D Default Settings as Model Description Text.

To create a text with the special style `3D Model Description text` use the `Model Descr. tool`  from the text toolset inside the `Tools` tool group.

You can use special wildcards inside the text, e.g. for the content of the sheet title, see [“Using Wildcards” on page 67](#).

Please note: If you defined a different style in the 3D Default Settings, use another text tool for creating the Model Description Text.

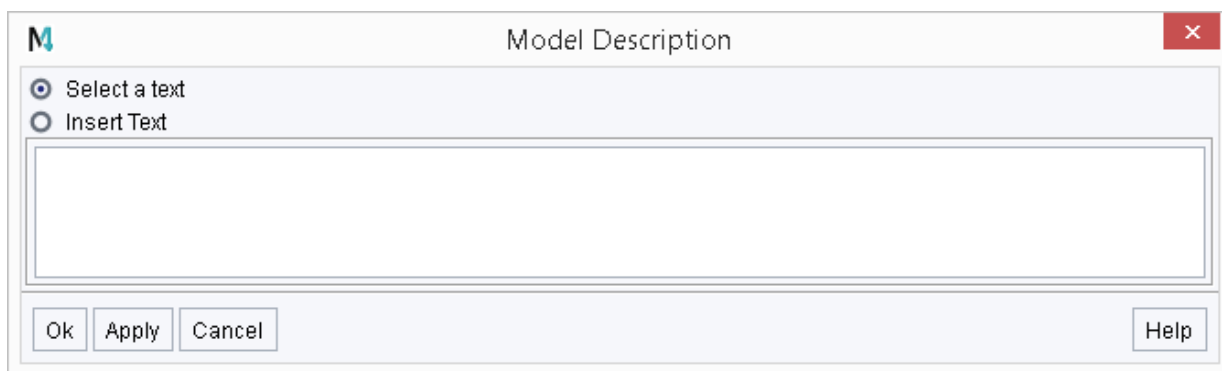
Manual Creation of Model Description Text File

Please note: In the Defaults Settings dialog > 3D Products > Model Description > Create Model Description Text automatically option needs to be deselected (see [“Automatic Creation of Model Description Text File”](#) on page 64), for activating the Model Descr. tool .

Manual creation of the Model Description Text file means that it is created without modeling the sheet.

Using the Model Descr. tool  opens the following dialog.

Figure 42 Dialog Model Description



You have the possibility to select one text on the sheet, or to insert a text into the dialog:

Select a text

If this option is selected (default), select any text on the sheet by clicking on the LMB on it.

The text will be written into the text field of the dialog immediately.

Only one text can be selected on the sheet.

The text can contain wildcards (see [“Using Wildcards”](#) on page 67).

Please note: As long as no text is selected and the text field of the dialog is empty, pressing Apply or OK results in an error message displaying that no file `<modelfile>.txt` was created.

Insert a text

Using this option you can insert any text into the text field. The use of wildcards is also supported. Inside the dialog the text will be always displayed with wildcards, in the text file they will be replaced.

OK

creates the `<modelfile>.txt` immediately and closes the dialog.

Apply

creates the `<modelfile>.txt` immediately. The dialog remains opened.

Using Wildcards

Inside the model description text you can use wildcards. The wildcard must be in-between doubled dollar signs, `$$xxxx$$`, to be identified as wildcard.

Inside the wildcard characters you can use the following terms:

- Text styles either with or without !, e.g.
`$$!model_descrip$$`
`$$model_descrip$$`
- Localized style labels, e.g. for the english environment
`$$3D model description text$$`
for the german environment
`$$3D Modell Beschreibungstext$$`
- Valid Can Codes, e.g.
`$$tsh$$`

It is searched on the sheet for the specified text. If the first one was found, the search is stopped and the wildcard is replaced by the found text string.

Please note: Only the first found text will be considered!

Language Support

Inside the Equipment Load Dialog of MPDS4 localization of the description text is considered. For this you need to enter an abbreviation in squared brackets as separator for the language. Following languages are supported:

- [eng] text followed will only be displayed in an English environment.
- [ger] text followed will only be displayed in an German environment.
- [fre] text followed will only be displayed in an French environment.
- [ita] text followed will only be displayed in an Italian environment.
- [jap] text followed will only be displayed in an Japanese environment.

So you can create a description in several languages but only the one that fits to the working environment will be displayed.

Example:

```
[eng]
This is a description
[ger]
Dies ist eine Beschreibung
```

Please note: In MEDUSA4 3D the complete text is displayed.

Configuration

You can control the settings of the model description text with the help of the 3D tab in the Options dialog or by editing the file *defaults.dat*:

```
-- model description text
model_descr_text  mdt_create, mdt_choice, mdt_text
                  true, !style, "!model_descrip"
```

mdt_create

If you want automatic creation `mdt_create` must be set to `true`. Only in this case the settings for `mdt_choice` and `mdt_text` will be considered.

mdt_choice

This variable can either have the values `!style` or `!text`.

!style: When `!style` is chosen, the `mdt_text` value needs to be the localized string of a valid text style or the text style itself. Examples:

- Style: `"!model_descrip"`
- Localized style name: `"3D Modell Beschreibungstext"`

The value has to be entered in quotation marks `" "`.

Please note: The use the localized style name only works in the environment that fits to the localization! If you want a configuration for any language insert the style here!

!text: When `!text` is selected, the value for `mdt_text` can be any text.

mdt_text

As described above, the value for `mdt_text` depends on the setting of `mdt_choice`.

Please note: The value needs to be inside quotation marks `" "`.

The use of wildcards for the text value is supported, see ["Using Wildcards" on page 67](#).

SWEEPS




This chapter describes how to use the Sweep generator to create models. It also covers some basic modeling techniques which can be used with the other model generators.

- General 70
- Slab Sweeps 71
- Face Sweeps 74
- Edge Sweeps 76
- Basic Modeling Techniques 78
- Summary of Element Types 94

General

The Sweep generator creates a model by sweeping a profile through space in a straight line for a specific distance. The sweep direction is always normal to the plane of the profile. The sweep start and end planes are defined in the sweep direction by a link line which is extended into another viewbox orthogonal to the profile viewbox.

There are three options available in the Sweep generator:

- The Linear Sweep tool  produces a solid model of the swept volume
- The Linear Face tool  produces a surface model consisting of one or more flat surfaces normal to the sweep direction
- The Linear Edge tool  produces a surface model consisting of the surfaces of the swept volume parallel to the sweep direction

The option which operates for a sweep is specified by the type of link line which you use for the definition. The link line styles are:

- Slab Generator
- Face Generator
- Edge Generator

Slab Sweeps

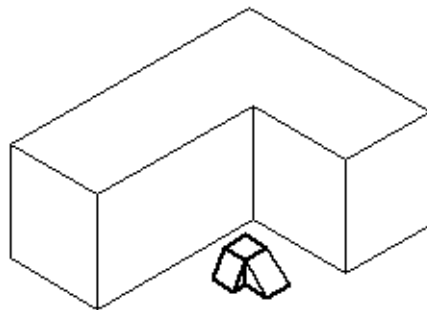
A slab sweep produces a slab model of the volume swept by a profile. The profile is drawn in one orthogonal viewbox and a link line for Slab generator (Linear Sweeping) is attached to the profile and extended into another viewbox, orthogonal to the first. The depth and position of the model in the third dimension can be defined in one of two ways:

- By arrowhead point functions (FUNV 14 or FUNV 15) on the link line at the start and finish planes, see [Figure 14, “Attaching the Link Line To the Profile Line” on page 27](#). (The point functions are automatically placed, when creating the link line.)
- By attaching a depth text of style `Modeller Text ass. by Geometry` to the link line which explicitly specifies the start and finish planes. In the second case, the values used in the depth text define the third dimension of the model. For example, if a profile is drawn in the XY plane, the command:
DEPTH 10 16.35 on the link line creates a model 6.35 deep between the planes Z=10 and Z=16.35, see [Figure 15, “Depth Text on the Link Line” on page 28](#) for an example of the use of depth text.

The resulting solid model has parallel faces of the same size and shape as the original profile, and edges (tiles) joining these faces. The edges are always perpendicular to the faces.

[Figure 43](#) shows an isometric view of a model produced with the solid sweep generator.

Figure 43 A Model Generated as a Solid Sweep



Example

Use the procedure described below to generate the model shown in [Figure 43](#).



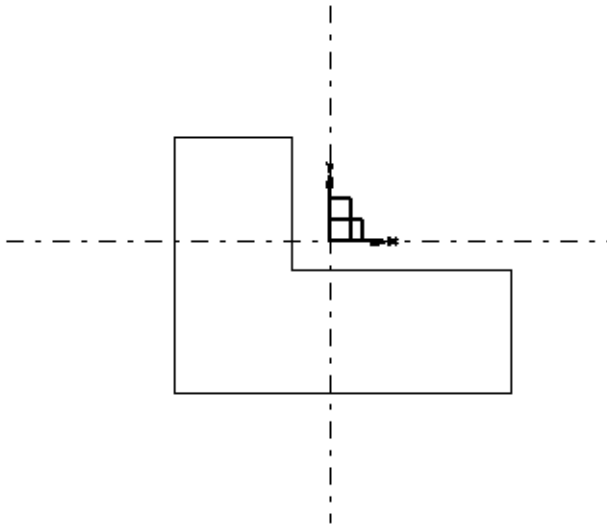



1. Load a MEDUSA4 3D sheet (for details see [“Loading a Standard 3D Sheet” on page 19](#)).
2. Select the Solid Thin tool  which creates a profile line of style `LP0`.
3. Draw the profile in a viewbox containing a DYX prim . The profile must be a closed line, as shown in [Figure 44](#).

Figure 44 The Profile Line

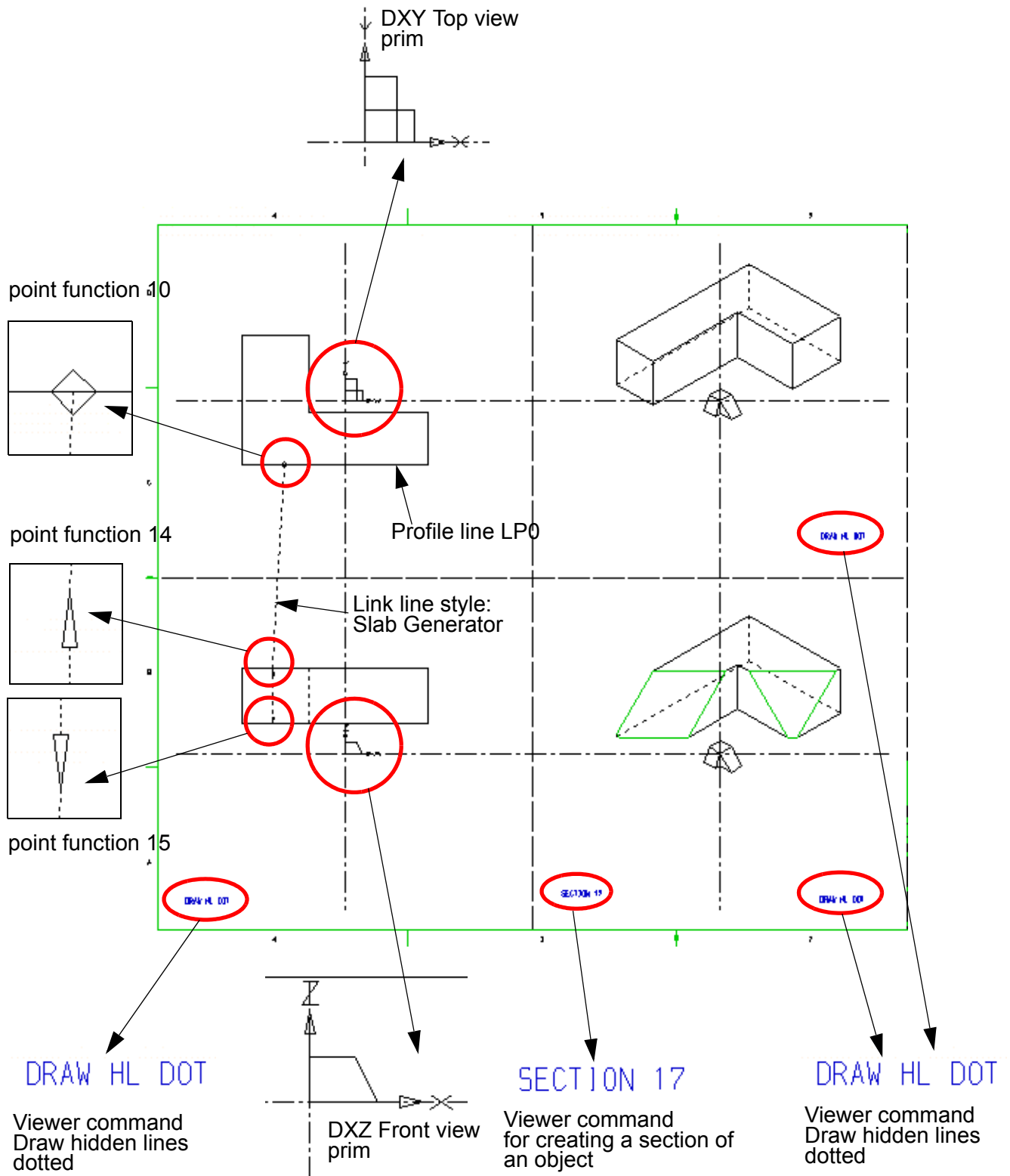


4. Select the Linear Sweep tool .
5. Attach the link line (style: Slab Generator) to the profile line using a segment probe as shown in [Figure 14, “Attaching the Link Line To the Profile Line” on page 27](#). The position of the diamond point function (FUNV10) on the profile line does not matter. Provided that the point function is on the line the Modeler will find it.
6. Define the depth of the model and its position in the Z-axis by extending the link line into a viewbox containing a DXZ prim . The link line contains automatically two arrowhead point functions (FUNV 14 or FUNV 15). The distance between these point functions in the Z direction defines the depth of the model.
7. Select the Model + Reconstruct tool  to generate and view the model.

Please note: The SECTION command is used to produce the view at the bottom right viewbox. This command sections an object and is used here to show that the model is solid.

The viewer commands used in [Figure 45, “The Completed Slab Sweep Definition and Model” on page 73](#) are described in [“Basic Viewer Commands” on page 53](#).


Figure 45 The Completed Slab Sweep Definition and Model



Face Sweeps

A face sweep produces a surface model consisting of a single flat surface (or multiple flat surfaces) defined by a profile. The profile is drawn in one orthogonal viewbox and a link line of style `Face Generator` is attached to the profile and extended into another viewbox, orthogonal to the first.

The position of the model in the third dimension is defined by the arrowhead point functions (FUNV 14 or FUNV 15) on the link line at each plane where a surface is to be generated.

To create a face model, follow the solid sweep procedure (see “[Slab Sweeps](#)” on page 71) but choose the `Linear Face` tool  to create a link line of style `Face Generator`.

[Figure 46](#) shows a face sweep definition using the same example as was used in the previous section. [Figure 47](#) shows the completed model.

Figure 46 A Face Sweep Definition

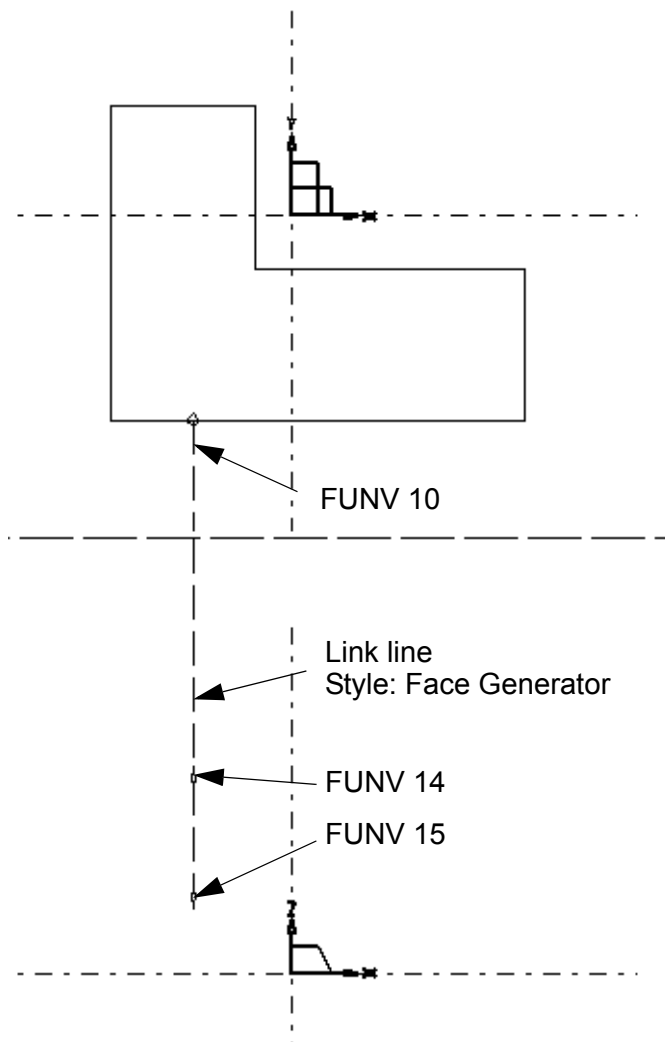
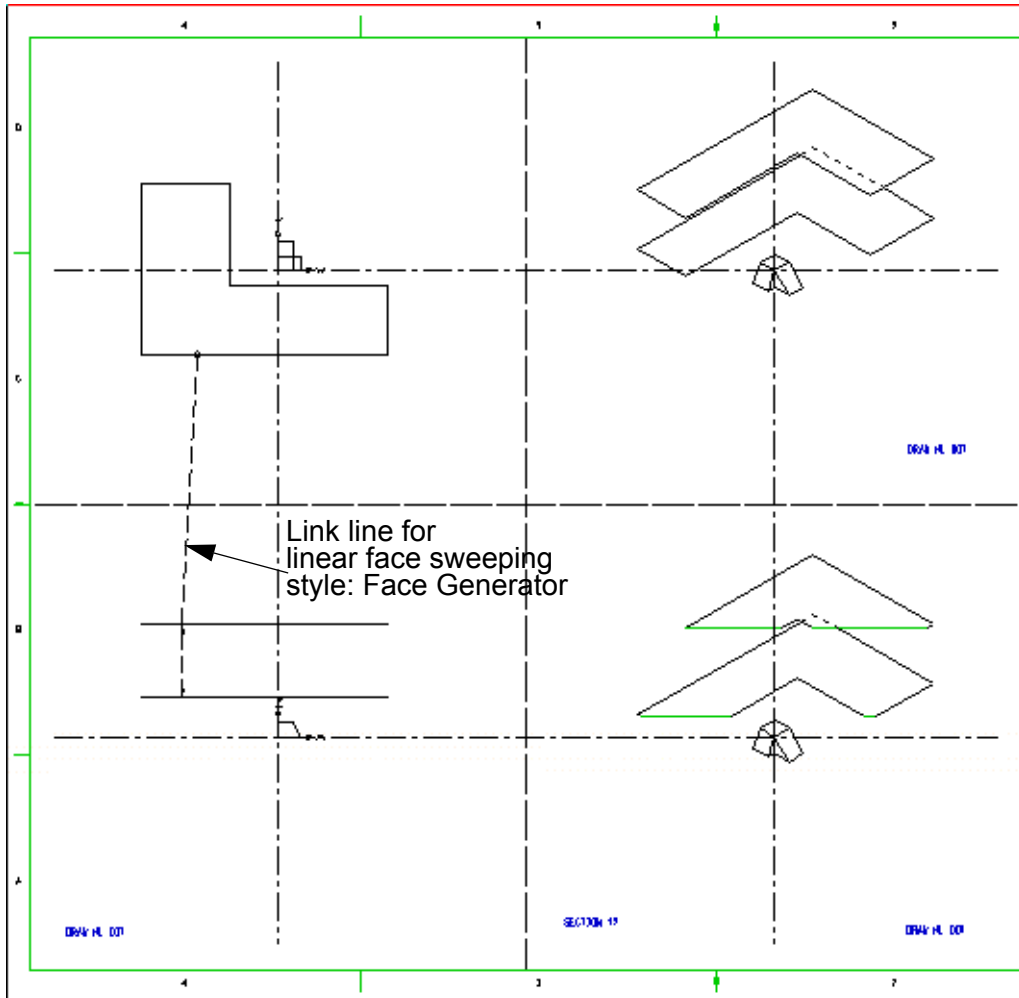



Figure 47 The Completed Face Sweep Definition and Model



Edge Sweeps

An edge sweep produces a surface model consisting of the edges of the volume swept by a profile. The profile is drawn in one orthogonal viewbox and a link line of style `Edge generator` is attached to the profile and extended into another viewbox, orthogonal to the first.

The position and depth of the model in the third dimension is defined by the arrowhead point functions `FUNV 14` or `FUNV 15` on the link line (or depth text, see “[Slab Sweeps](#)” on page 71).

To create a model using the `Edge Generator`, follow the procedure described to create a slab sweep (see “[Slab Sweeps](#)” on page 71), but use the `Linear Edge` tool .

[Figure 48](#) shows an edge model definition using the same example as was used in the section “[Slab Sweeps](#)” on page 71. [Figure 49](#) shows the completed model.

Figure 48 An Edge Sweep Definition

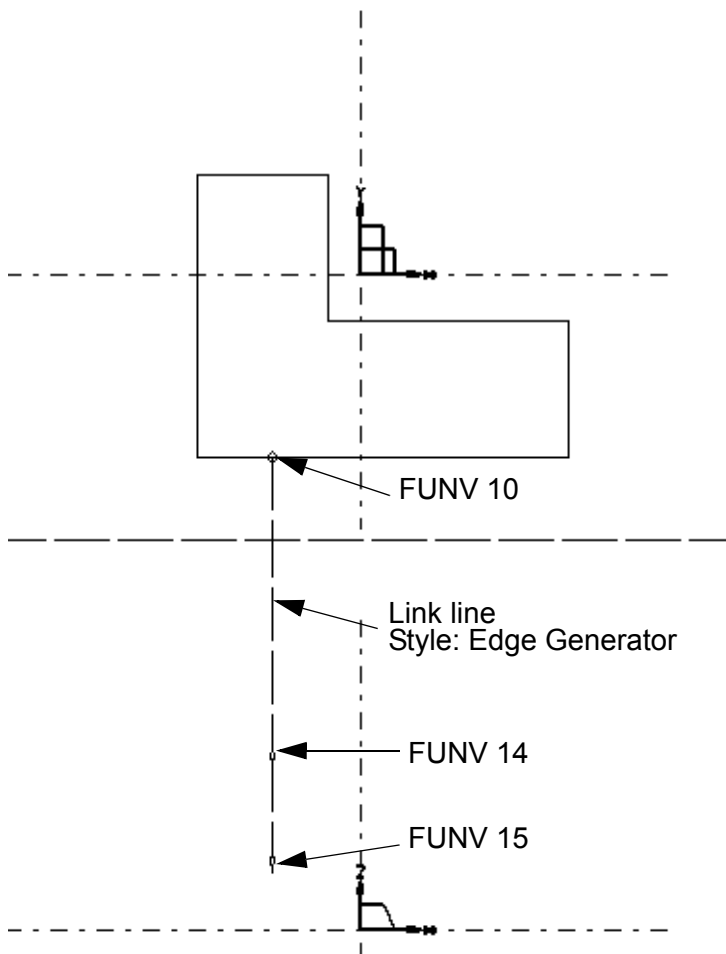
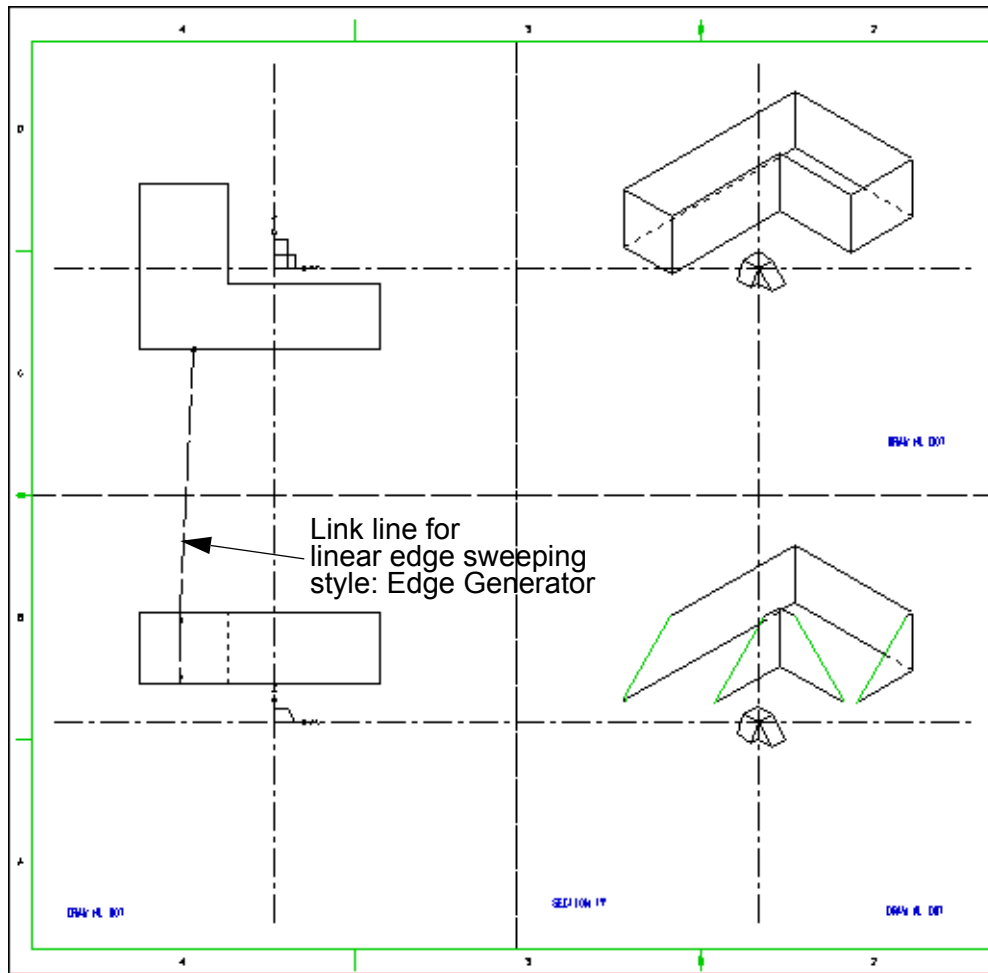


Figure 49 The Completed Edge Sweep definition and Model




Basic Modeling Techniques

This section discusses the basic modeling techniques used in the MEDUSA4 Modeling System.

Modeling Objects Containing a Single Hole

Objects which contain holes can be modeled. The basic principle is the same for solid sweeps, volumes of revolution, and all other model generators.

To model an object containing a hole, follow the procedure described below:

1. Draw a closed line representing the hole profile inside the outer profile line. Use a line of style `Profile LP0` through `Profile LP9` (type `LP0` through `LP9`) to draw the hole profile.
2. Pick up the hole profile line with the same link line that picks up the outer profile. Use a segment probe and a cross point function (FUNV 11) to attach the link line to the hole profile. It does not matter which profile the link line picks up first.
3. Extend the link line into another viewbox to define the depth and position of the object model. Use arrowhead point functions (FUNV14 or 15) or depth texts to define the depth on the link line as described previously in “Edge Sweeps” on page 76.
4. Select the Model + Reconstruct  tool to generate and view the model.

If the context is clear, the relevant point functions are added automatically to the link line. If the context is not clear, point functions have to be added manually or existing point functions have to be changed.

To assign the point functions to the link line set the line into the edit mode and use either:

- the Linepoint popup menu from the RMB popup menu (Figure 50) or
- the Line Point Properties dialog from the RMB popup menu (Figure 51) or
- the Point Functions dialog (Figure 52) from the Dashboard.

Figure 50 Line Point Popup Menu

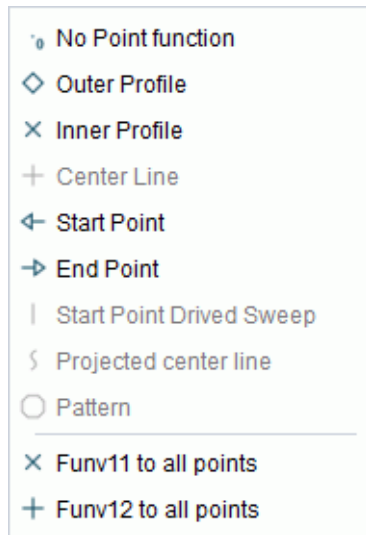


Figure 51 The Line Point Properties Dialog

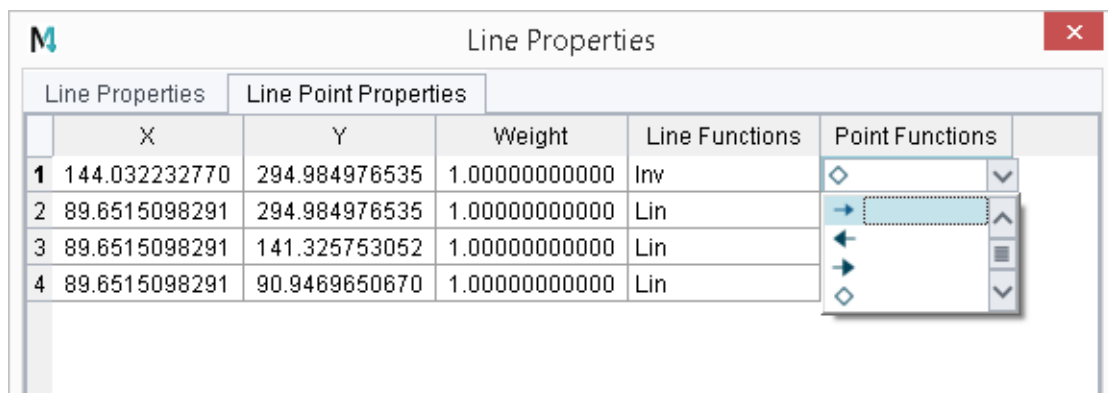
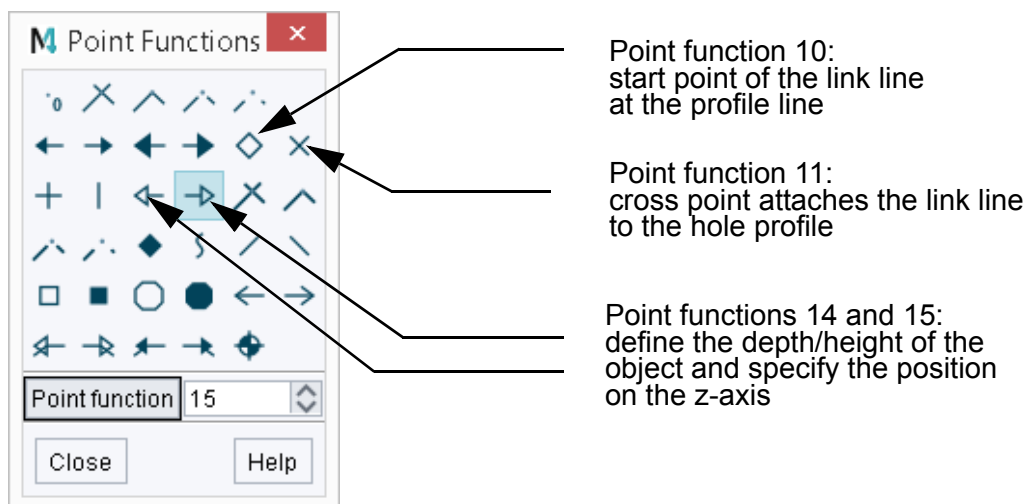


Figure 52 The Point Functions Dialog



Modeling Objects Containing Several Holes

To model an object containing several holes, the same principle is used, as in the previous section:

1. Draw as many hole profiles as required inside the profile line using a line of style `Profile LP0` through `Profile LP9` (type LP0 through LP9).
2. Attach the same link line to each new hole profile, using a cross point function (FUNV 11). The order in which you attach the link line to the holes does not matter.
3. Define the third dimension in another orthogonal viewbox as already described in [“Edge Sweeps” on page 76](#).

[Figure 53, “Definition of an Object Containing Holes” on page 81](#) shows a definition of an object which contains holes. [Figure 54, “The Completed Model” on page 82](#) shows the completed model generated from the definition sheet.

Please note: Inside the profile line as maximum 255 profiles can be drawn. If this number will be exceeded, the modeler aborts the process and an error message will be displayed on the drawing (see [“Error Messages” on page 521](#)).

Figure 53 Definition of an Object Containing Holes

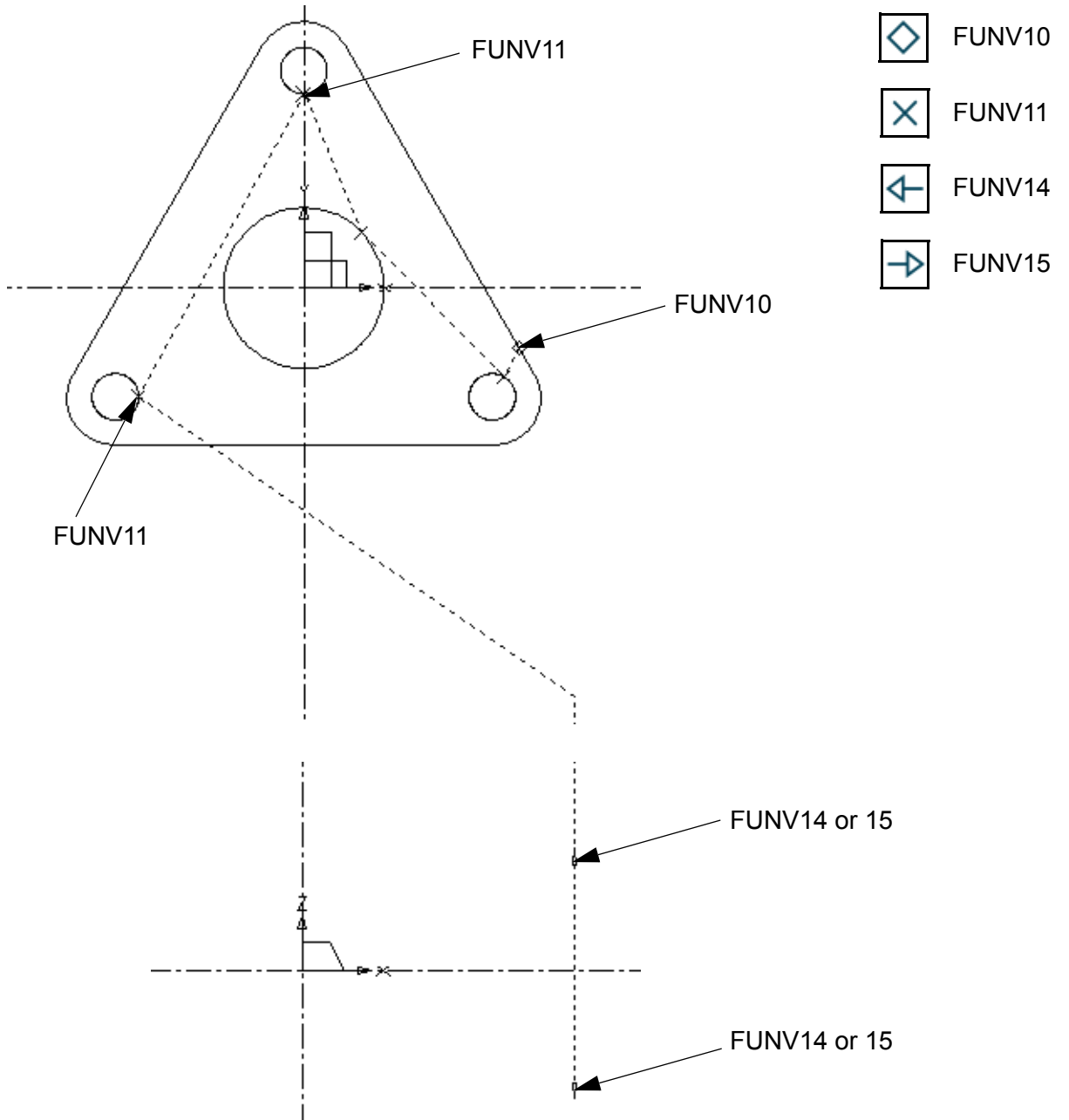
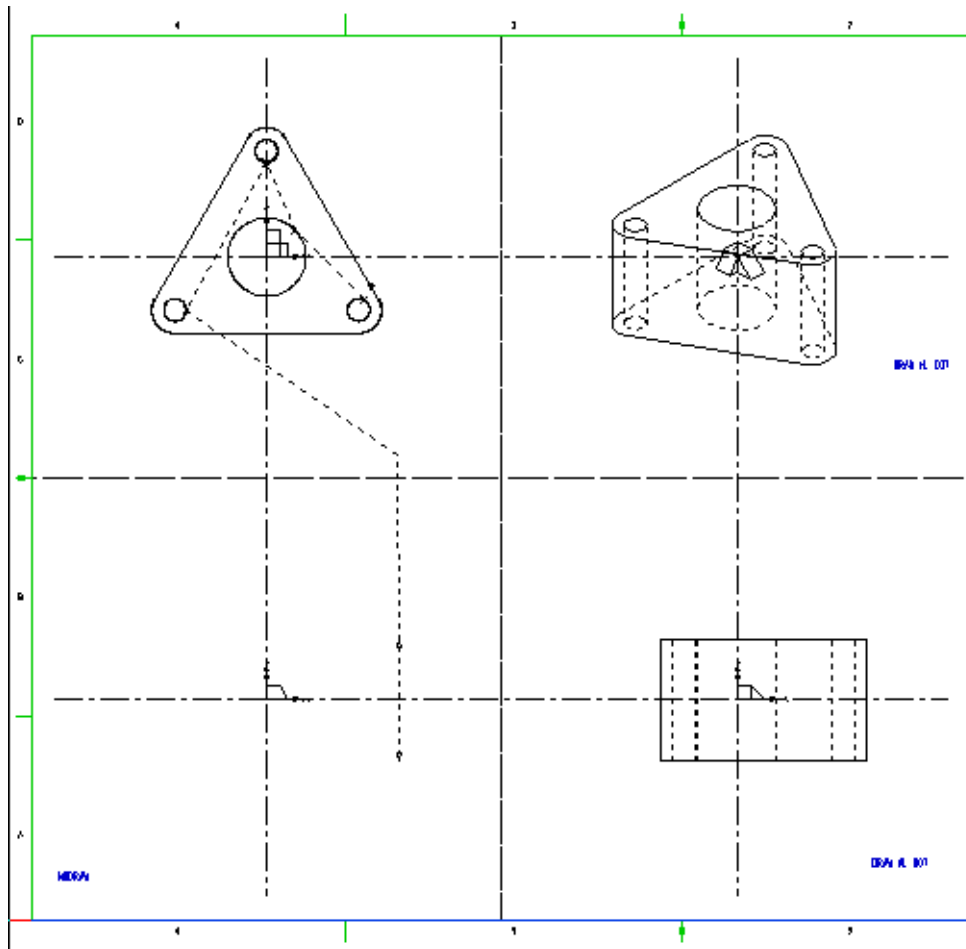


Figure 54 The Completed Model



Generating Multi-Object Models From a Single Profile

You can generate multi-object models simultaneously from a single profile. One of two methods can be used:

- Add extra point functions to the link line
- Add extra depth texts to the link line

For example, two pairs of arrowhead point functions (FUNV 14 or FUNV 15) applied to the example used in [Figure 54](#) produce a model consisting of two objects on different planes.

[Figure 55](#) shows a multi-object model definition. [Figure 56](#) shows the finished model.

Figure 55 Definition of a Multi-object Model from a Single Profile

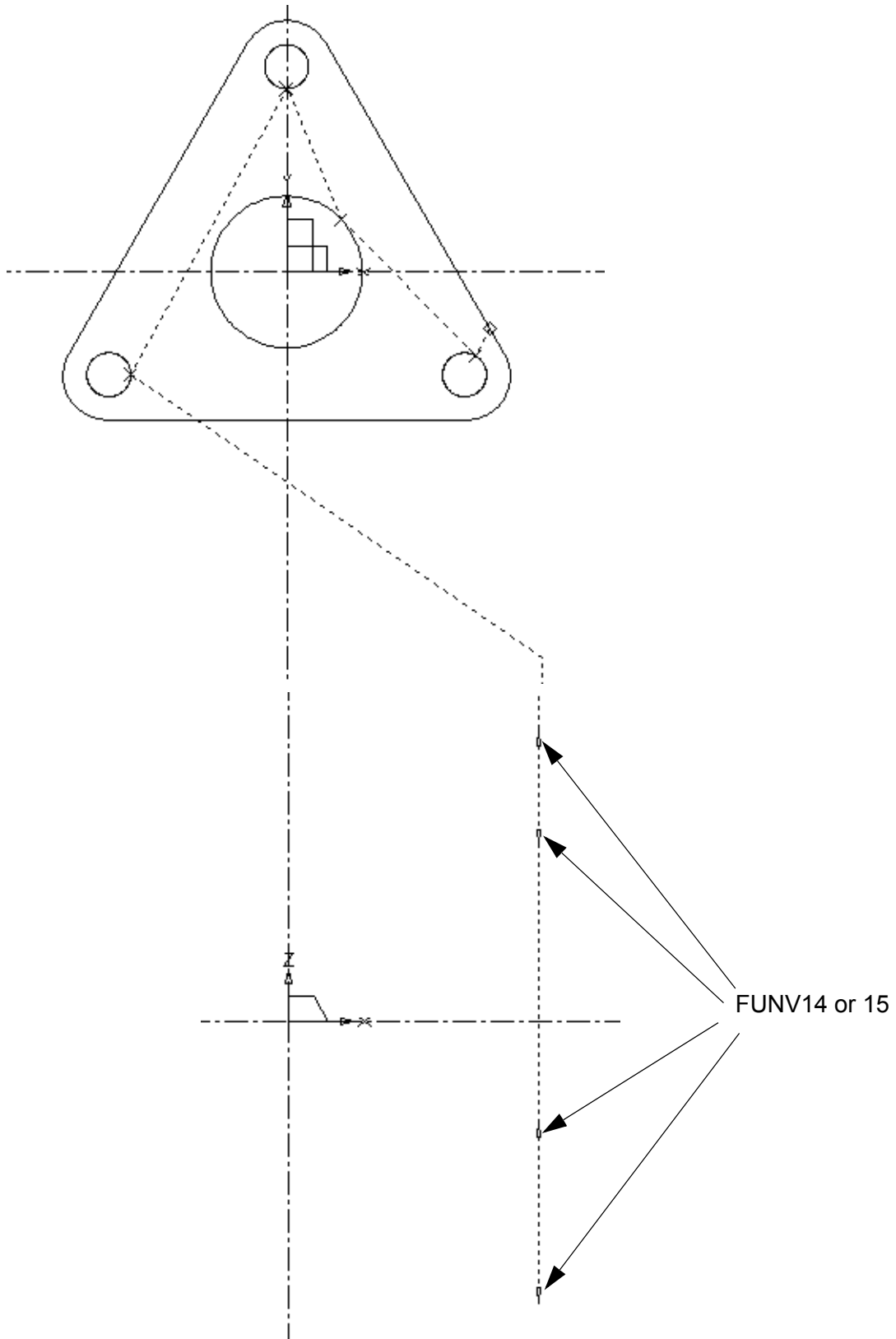
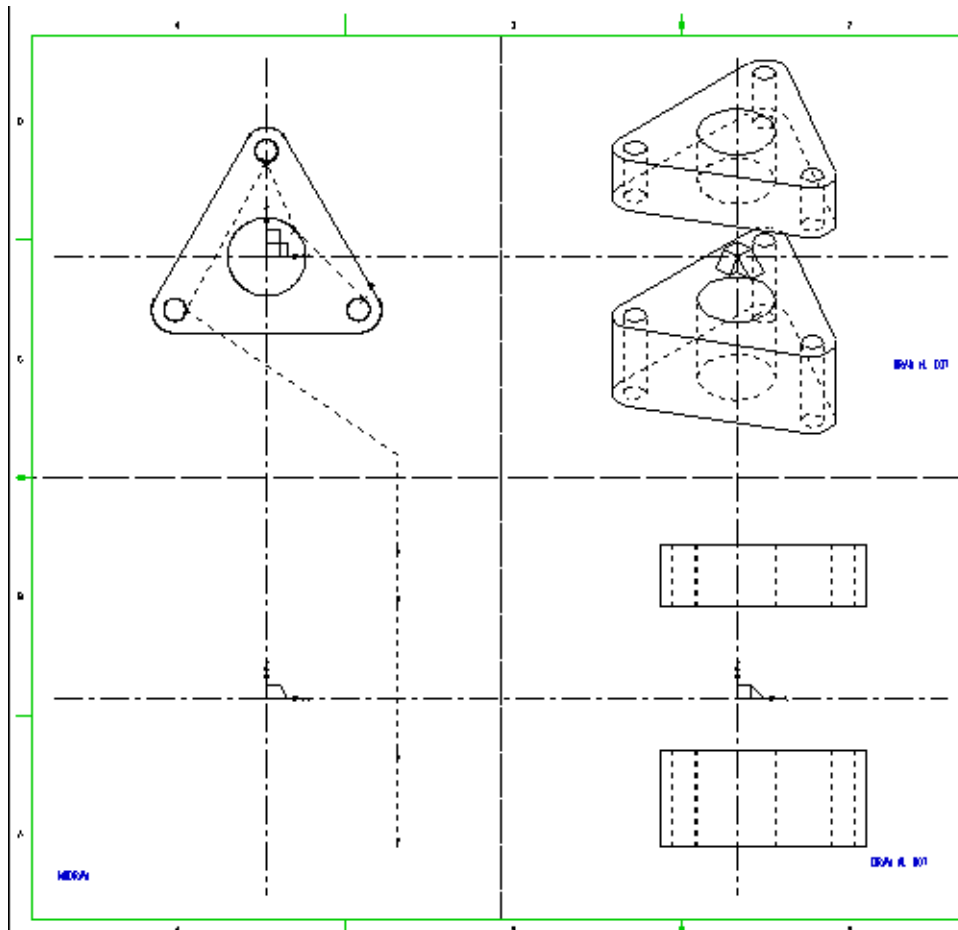


Figure 56 The Completed Model



Generating Multi-Object Models From Separate Profiles

You can generate multi-object models simultaneously from any number of profiles on the same plane.

Use the following procedure:

1. Draw the separate profiles of the objects in the same viewbox.
2. Pick up each profile with a diamond point function (FUNV 10) on a common link line.
3. Extend the link line into another orthogonal viewbox and define the depth of the model using a pair of arrowhead point functions or a depth text.

Figure 57 shows the definition for a two-object model. Figure 58 shows the finished model.

Figure 57 Definition of a Multi-object Model From Several Profiles

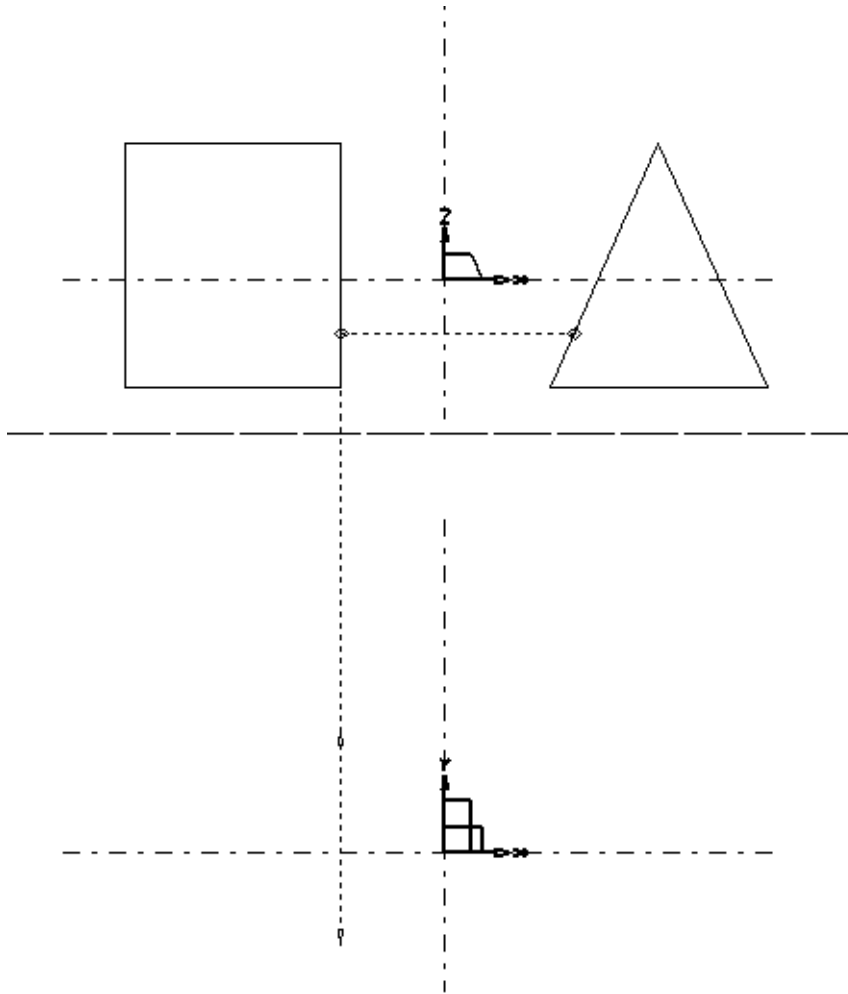
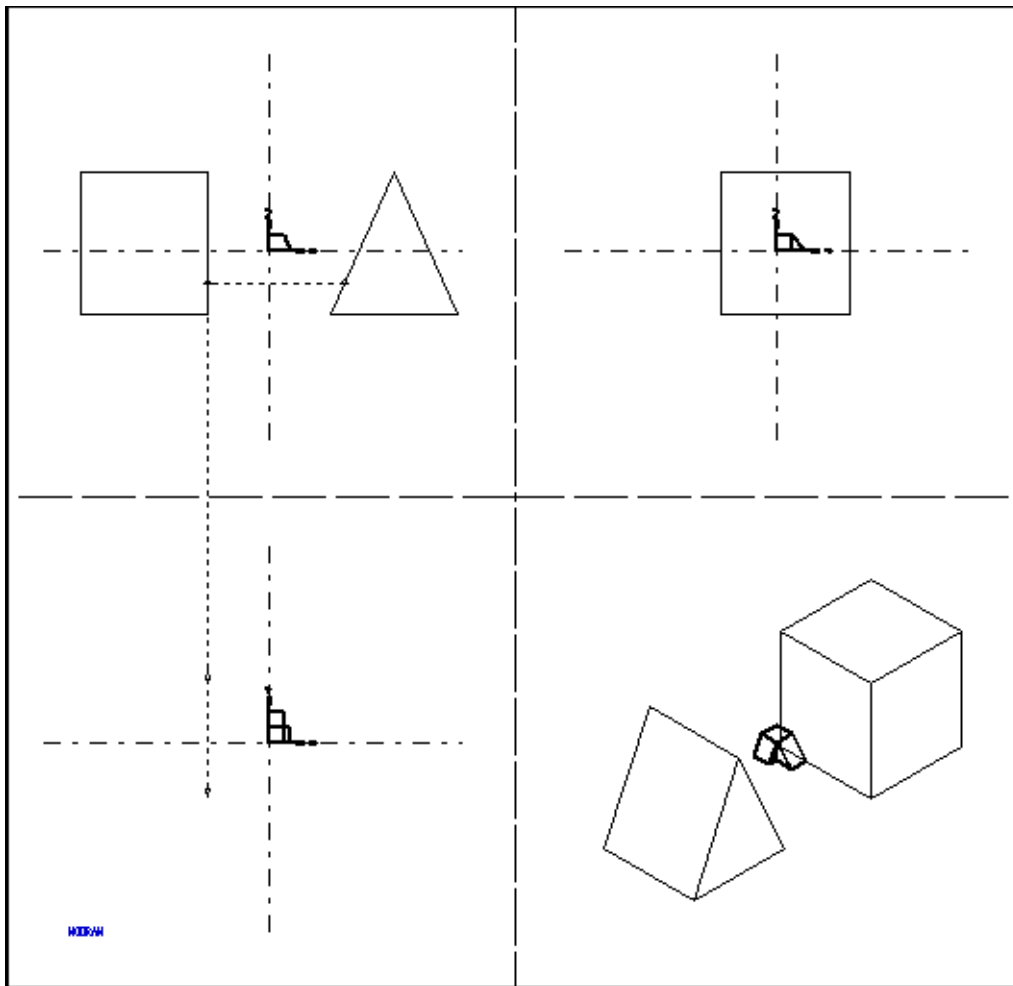


Figure 58 The Completed Model

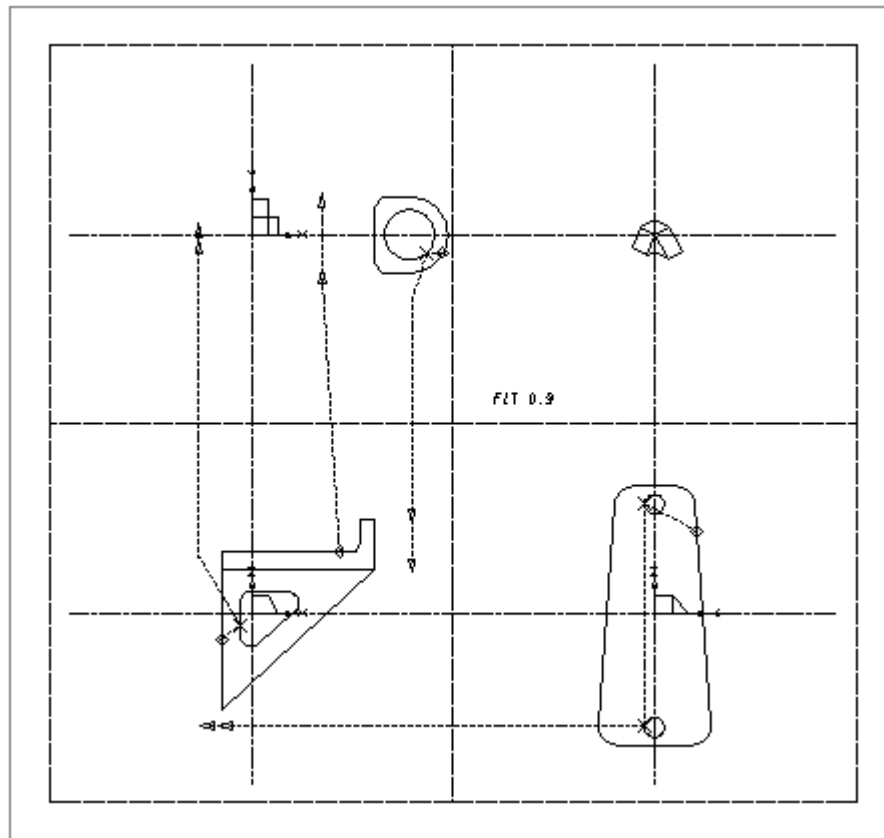


Generating Complex Models

In practice you may want to produce models of more complexity than shown in the previous examples. One way of doing this is to split the model into separate, simpler parts. This method is used in [Figure 59](#) and [Figure 60](#) which show a fabricated wall bracket modeled from four components.

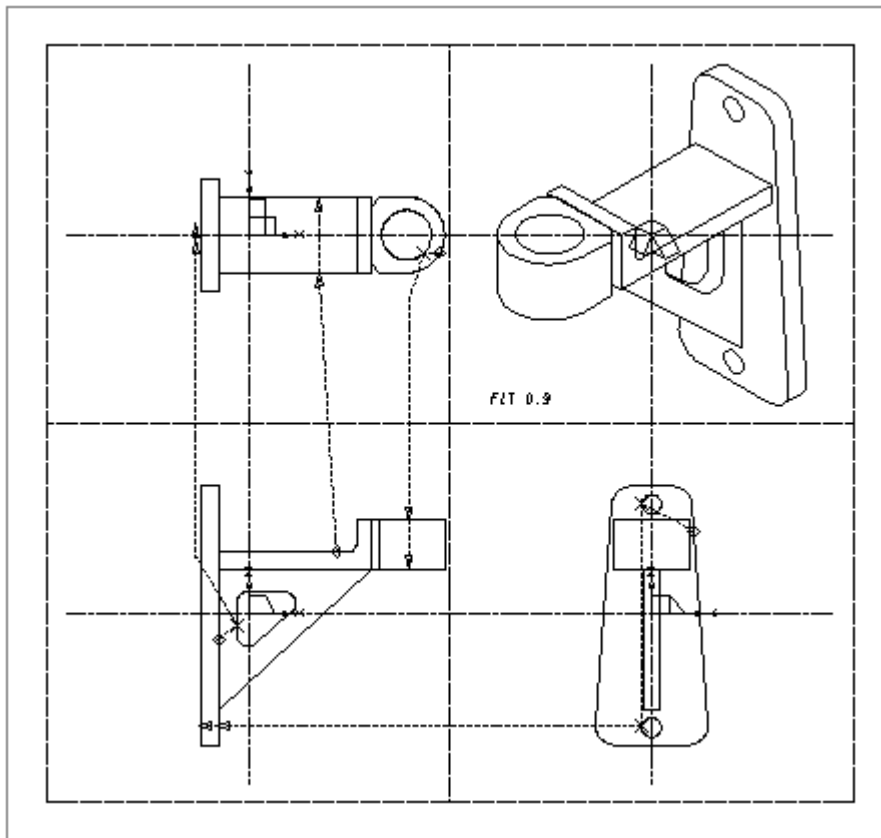
[Figure 59](#) shows the definition sheet. The object definitions are all solid sweeps.

Figure 59 Definition of Wall Bracket Components




Please note: The components have edges in common, they are drawn as separate closed profiles. [Figure 60](#) shows the finished model. Naturally, the spatial relationship between the profiles and the distances between point functions on the link lines must be correct. Construction lines can be used to help ensure that this is the case.

Figure 60 Completed Model of the Wall Bracket



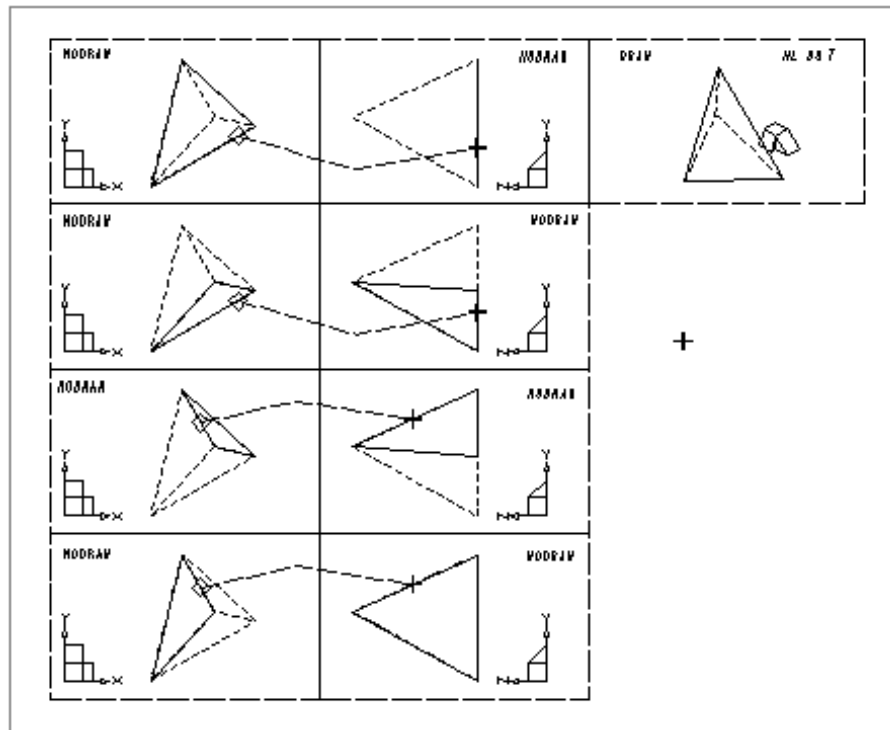
Defining a Shell Model as a Set of Polygonal Faces

A shell model may be generated by defining it as a set of polygonal faces. This feature may be used in the MEDUSA4 Sheet Metal Design system.

The method consists of drawing each polygonal face as a profile in two complementary views in the model definition. Each profile must contain the same number of corresponding points, which means that the first point in one profile must correspond with the first point in the other profile, and so on. The two profiles are then joined by a link line of style `Face Poly Generator` (tool `Polygon` ). One profile is picked up by a diamond point function (FUNV 10) and the other by a plus point function (FUNV 12).

An example of this modeling technique is shown in [Figure 61](#). For clarity, the face definitions are shown in separate viewbox pairs, whereas in practice they would all be drawn in the same pair as in [Figure 62](#).

Figure 61 A Tetrahedron Modeled as Four Polygonal Faces




Locking a Link Line to a Profile

A means of locking each link line to a specific profile is provided by the CPL group. This is a 3D system group which accepts profile lines and link lines as members.

This locking capability is needed because sometimes it is necessary (or convenient) to draw profile lines superimposed on each other in a model definition. This can occur when defining model faces as polygons, as shown in [Figure 62](#) for example, using just one pair of viewboxes. In such a case, there is often no part of a profile which is not superimposed on another, and therefore that profile cannot be unambiguously picked up. Thus it is likely that the Modeler will associate a profile with the wrong link line.

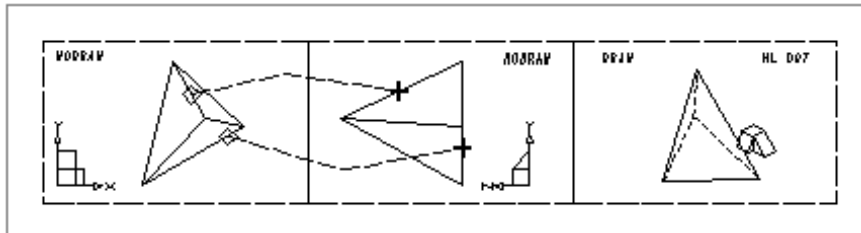
This is the procedure to follow when you need to lock a link line to a profile:

1. Choose the Empty group tool  to open a new CPL group.
2. Draw the profile line.
3. Select and attach the link line to the profile line, and then complete the link line.
4. Exit tool and with it you close the group.

As an example, [Figure 62](#) shows the shell model definition of [Figure 61](#) reworked using the CPL group method.

Please note: There are still four link lines in this definition (one for each CPL group) but two of them are now superimposed on the other two.

Figure 62 The CPL Group Method Used On the Tetrahedron Definition





Generating Line Patterns on Faces

Patterns of lines can be generated so that they appear fixed on one face of an object. This is done by creating them as wire model profile lines in a CPL group on the object definition sheet.

This technique is useful for:

- Applying text patterns to a face
- Stippling a face

As an example, to apply a text pattern to the face of an object, follow the procedure described below:

1. Place the required text string inside the face profile on the object definition using one of the standard MEDUSA4 text (e.g. *Large*).
2. Select a 3D profile line (for example, of style `Profile LP5`). Carefully trace over each of the characters in the text string.
3. Unload the pattern of profile lines as a symbol and delete the original text and the profile lines from the sheet.
4. Open a new group of type CPL.
5. Load the symbol containing the pattern of lines at the required position in the definition viewbox. The symbol is now part of the CPL group.
6. Choose the *Wire Lobster* tool .
7. Attach the link line to one of the pattern profile lines using a circle point function (FUNV 26).
8. Extend the link line into the viewbox containing the link line for the object definition. End the link line at the point where the object link line ends. Place a single arrowhead point function (FUNV 14) at the end of the link line.
9. Close the CPL group.
10. Select the *Model + Reconstruct* tool  to produce the completed model.

By placing the end of the link line at the same position as the end of the object link line, the pattern will be in the same plane as the object surface and thus will appear attached to the surface.

Where two arrowhead point functions define the depth of an object, for example in a solid sweep, end the LL link line on the same plane as one of them. The one you choose will depend on which face of the modeled object you want the text pattern to appear.

The procedure for stippling a face is almost identical to the above. In this case it is the stippling, of line types $LP0$ through $LP9$ covering the object profile, which is unloaded as a symbol.

Modeling Oblique Features

Features of an object which are obliquely aligned to the main model can be modeled by using the oblique viewprims (types OVI, PVD, and SVD).

These viewprims work in conjunction with the orthogonal viewprims to define an oblique viewing direction and an oblique plane which is normal to that direction. The oblique viewprims are available in the *Prims* tool group shown in [Figure 64](#).

Figure 63 The Oblique Viewprims

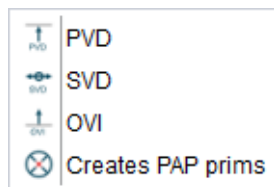
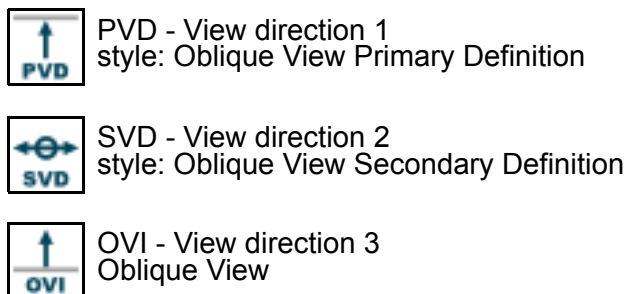


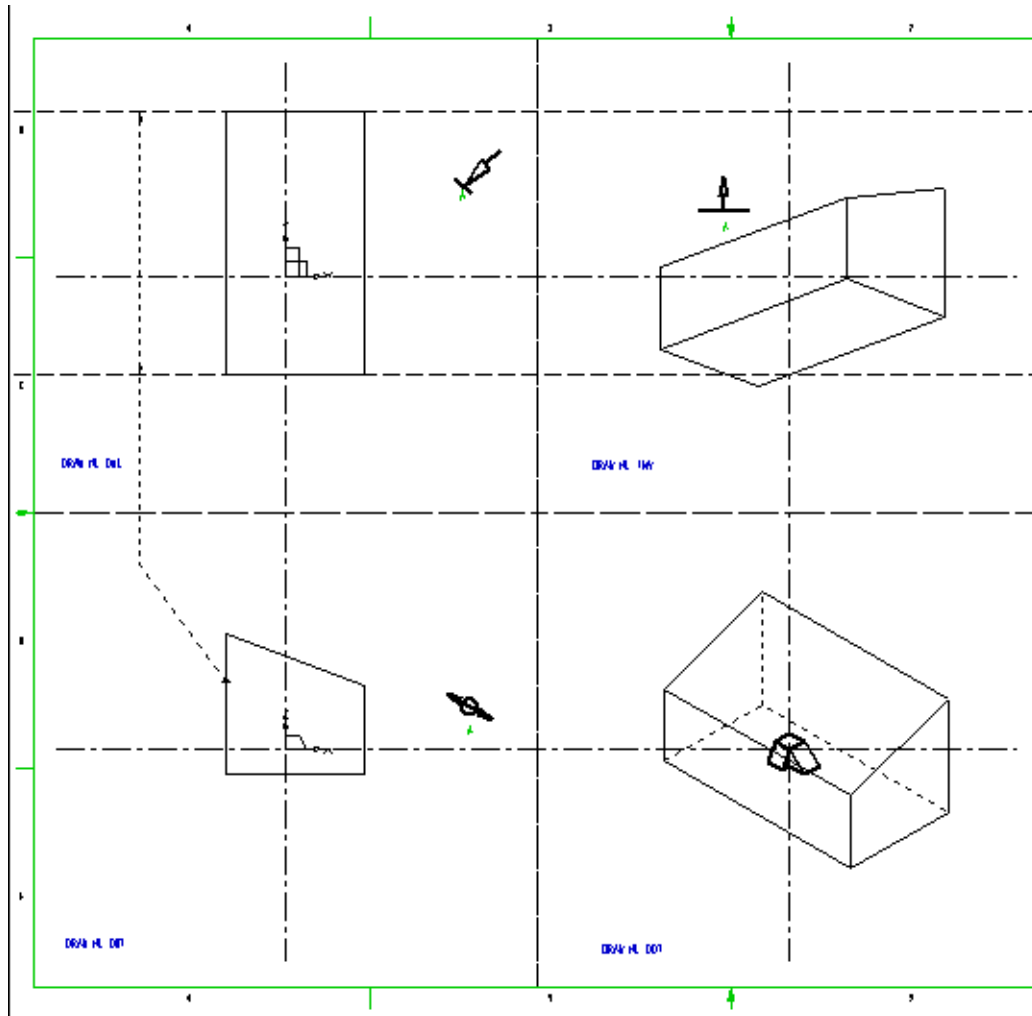
Figure 64 The Tools for Oblique Viewprims



Any model type may be generated using oblique viewing directions and planes. In the case of sweeps, the oblique profile to be swept is drawn in a viewbox containing an OVI viewprim which represents the oblique plane. The link line is drawn to another viewbox containing a PVD viewprim which, in conjunction with the orthogonal viewprim in that viewbox, defines the primary view direction. Arrowhead point functions define the length of sweep in the primary view direction. An SVD viewprim is used in another orthogonal viewbox to define a secondary view direction if a compound oblique view is required.

An example of oblique modeling is shown in [Figure 65](#) and [Figure 66](#), which show how an oblique sweep is used to cut a T-slot through a block at a compound angle.

Figure 65 The Oblique View Definition

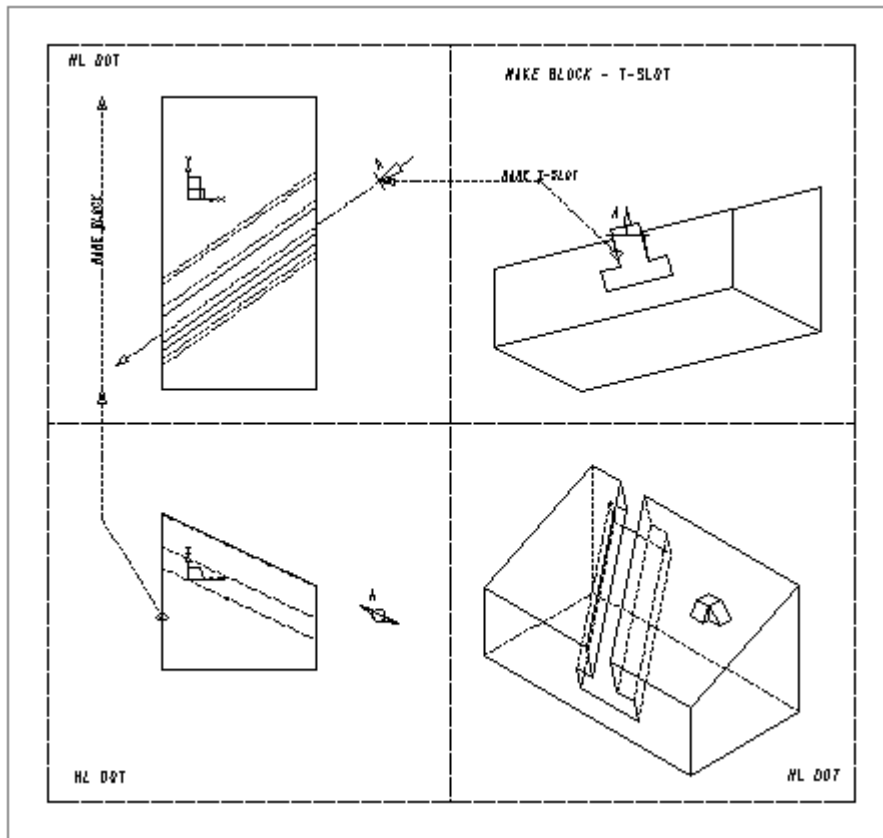


The definition sheet for the block, including the oblique view definition required, is shown in [Figure 65](#). The block is modeled as a solid sweep.

Please note: The oblique viewprims are associated with each other using a unique text label, **A** in this case, in each prim group. A dummy text (which appears as `lbl` when the viewprim is first loaded) of style `3D Oblique View Label` is provided for this purpose. Simply edit this text for each viewprim as required.

The complete definition and model is shown in [Figure 66](#), which shows the T-slot defined as a solid sweep and then subtracted from the block in a Boolean operation (“[Boolean Operations](#)” on page 209).

Figure 66 The Completed Model



See “[Modeling Oblique Features](#)” on page 91 for details of how to use oblique viewprims purely for oblique viewing.

Summary of Element Types

The following table gives a summary of the line styles, text styles and point functions used in this chapter.

Element Description		Element Style and Point Functions
Profile Lines		
	All Sweep profiles	Profile LP0 - Profile LP9
Link Lines		
	Slab sweep	Slab Generator
	Face sweep	Face Generator
	Edge sweep	Edge Generator
Point Functions		
	Pick up outer profile line	FUNV 10
	Pick up hole profile line	FUNV 11
	Define start and finish planes for sweep	FUNV 14 / 15
	Pick up pattern profile lines in CPL group	FUNV 26
Text		
	Depth or height text	Modeller Text ass. by Geometry (TMG)
Groups		
	Lock link line to profile line	3D group (Profiles+Linklines) (CPL)

VOLUMES OF REVOLUTION

The Volume of Revolution generator creates a solid model by rotating a profile through a full (or partial) revolution around a centerline in the profile viewbox so as to enclose a volume. The profile is connected to the centerline by a link line for rotational sweeping of style `Volume Revolution Generator`. This line is then extended into another viewbox (orthogonal to the profile viewbox) to define the position of the axis of rotation in the third dimension.

- [Open and Closed Profiles](#) 96
- [Simple Definitions](#) 97
- [Changing the Model Orientation](#) 106
- [Partial Volumes of Revolution](#) 107
- [Summary of Element Types](#) 109

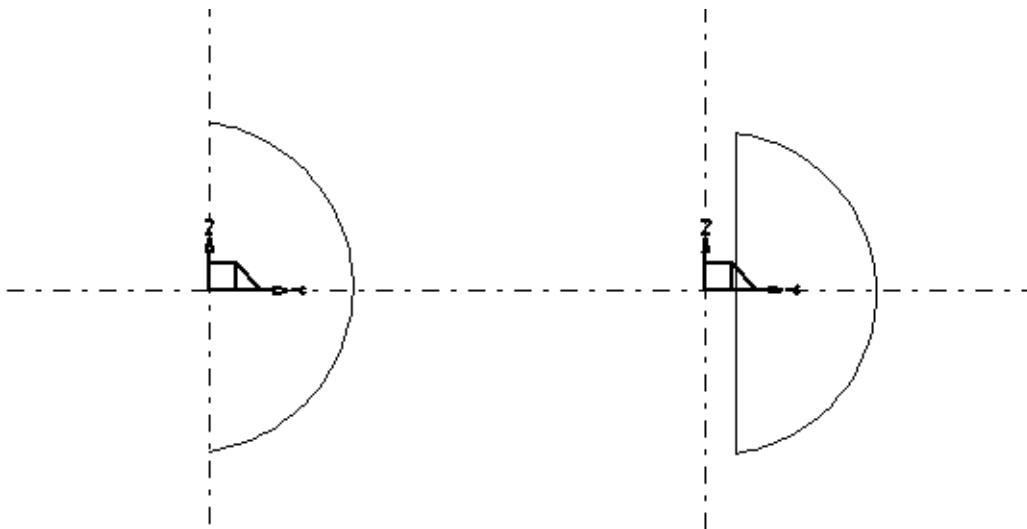
Open and Closed Profiles

Two types of profile can be used to generate a model as a volume of revolution:

- An open profile joined to the centerline at two points
- A closed profile which always generates an annular type model

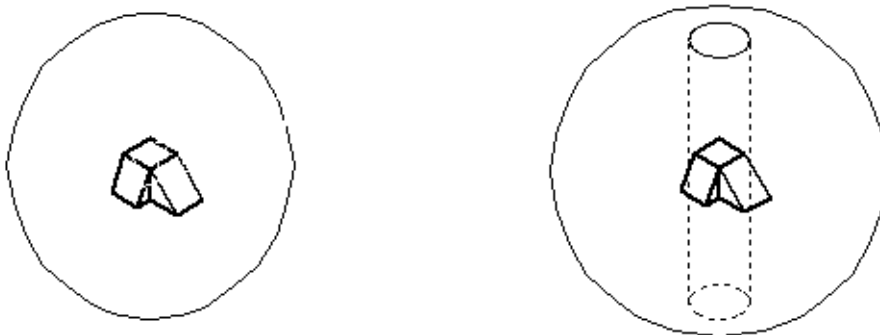
Figure 67 shows open and closed profiles. Figure 68 shows the models generated from these profiles.

Figure 67 Open and Closed Profiles



Please note: When drawing an open profile, the profile should be attached to the centerline at the two points using segment probes. If the profile line cuts through the centerline, the resulting model will intersect itself and will therefore be invalid.

Figure 68 Completed Models Using Open and Closed Profiles



Simple Definitions

This section shows how to create simple models using the Volume Revolution Generator. It covers:

- "Defining a Model Using an Open Profile"
- "Defining a Model Using a Closed Profile" on page 100
- "Defining a Hole in a Volume of Revolution" on page 103

Step-by-step procedures and examples of the techniques are provided.

Defining a Model Using an Open Profile

Use the following procedure to model a piston using an open profile. [Figure 69](#) and [Figure 73](#) show the example definition. [Figure 71](#) shows the completed model.


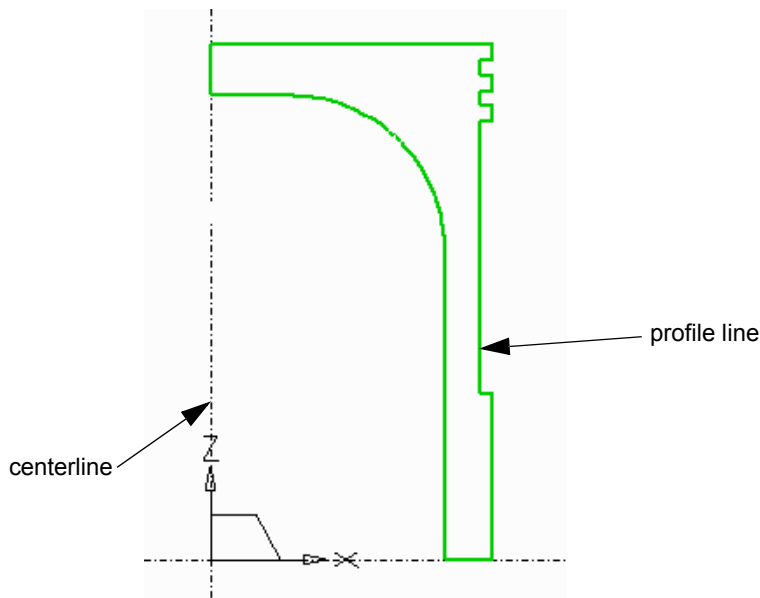

1. Choose the Solid Thin tool  (or any other profile line tool) from the Lines tool group to draw the profile line of the piston.
Draw it in the same way as shown in [Figure 69](#) because the profile of the piston will be rotated about the vertical centerline provided in the viewbox.

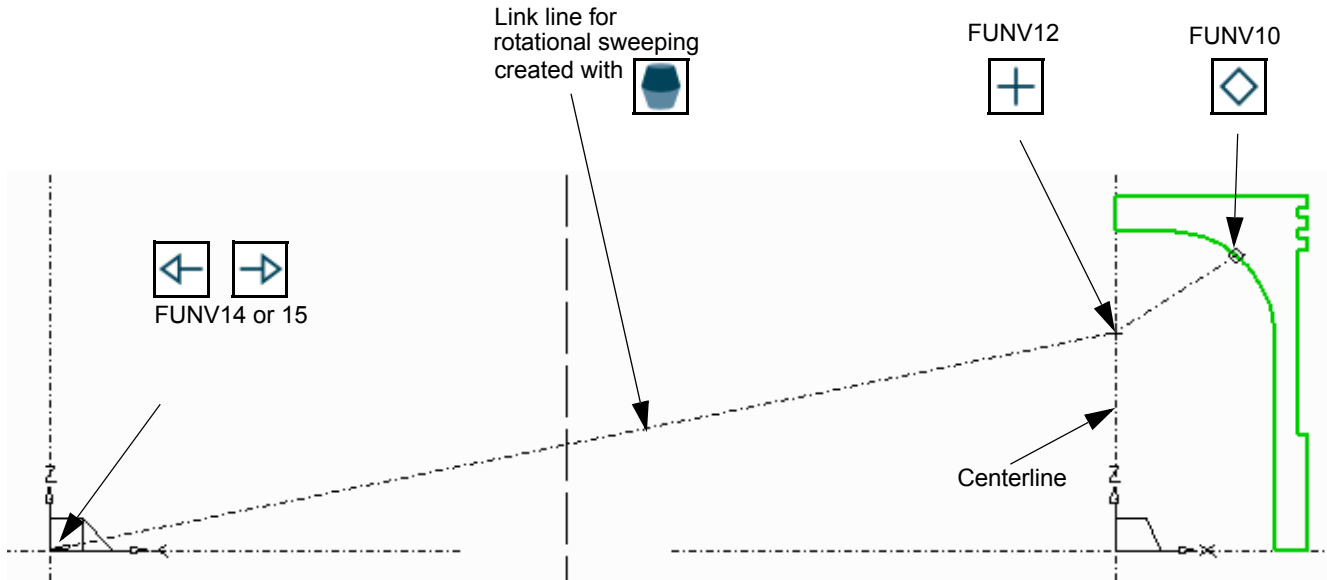
Figure 69 Profile of the Piston



2. Select the Volume Revolution tool  from the link lines toolset in the Lines tool group.
3. Click the LMB on the profile line to attach the link line to it.
4. Click the LMB on the center line to attach the link line also here.
5. Finish the link line by extending it into another viewbox with an orthogonal viewprim and then selecting Exit Tool from the popup menu.

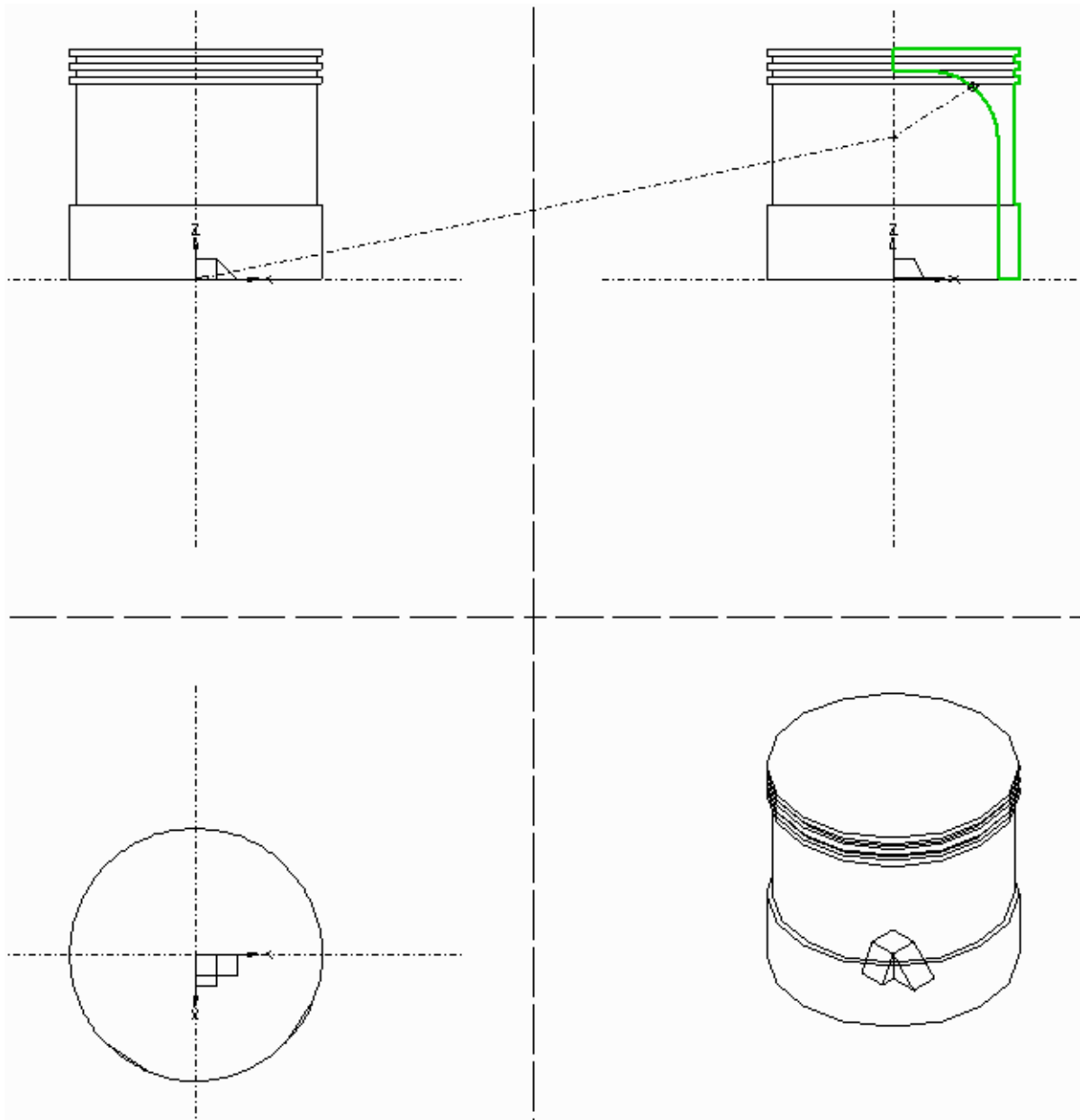
While drawing the link line the required point functions are added automatically.
Figure 70 shows the completed link line.

Figure 70 Adding the Link Line



6. Select the Model + Reconstruct tool  to generate and view the model.
Figure 71 shows the completed definition and model.


Figure 71 The Completed Model



Defining a Model Using a Closed Profile

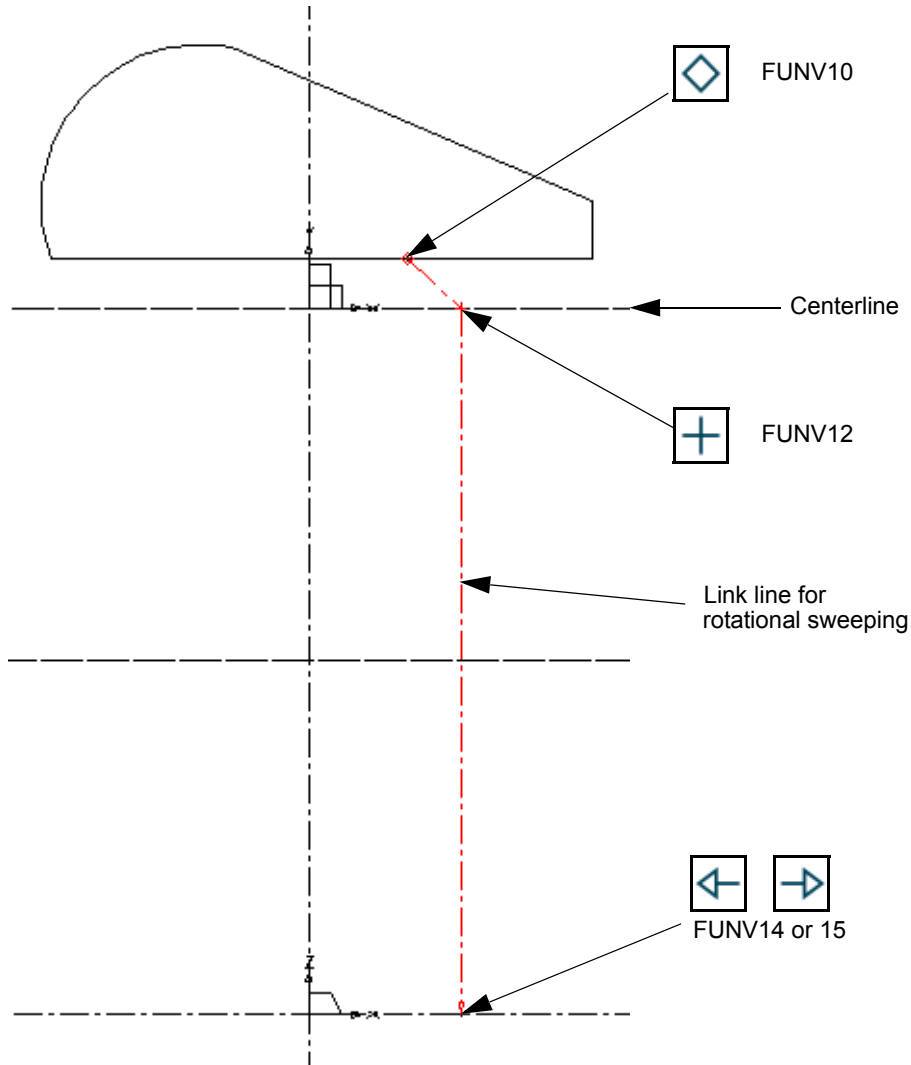
A closed profile creates an annular type model. [Figure 72](#) shows the definition for a model of this type. [Figure 73](#) shows the completed model.

To create the model, follow the procedure described below:

1. Draw a closed profile as shown in [Figure 72](#).
2. Select the Volume Revolution tool  from the link lines toolset in the Lines tool group.
3. Click the LMB on the profile line to attach the link line to it.
4. Click the LMB on the center line to attach the link line also here.
5. Finish the link line by extending it into another viewbox with an orthogonal viewprim and then selecting Exit Tool from the popup menu.

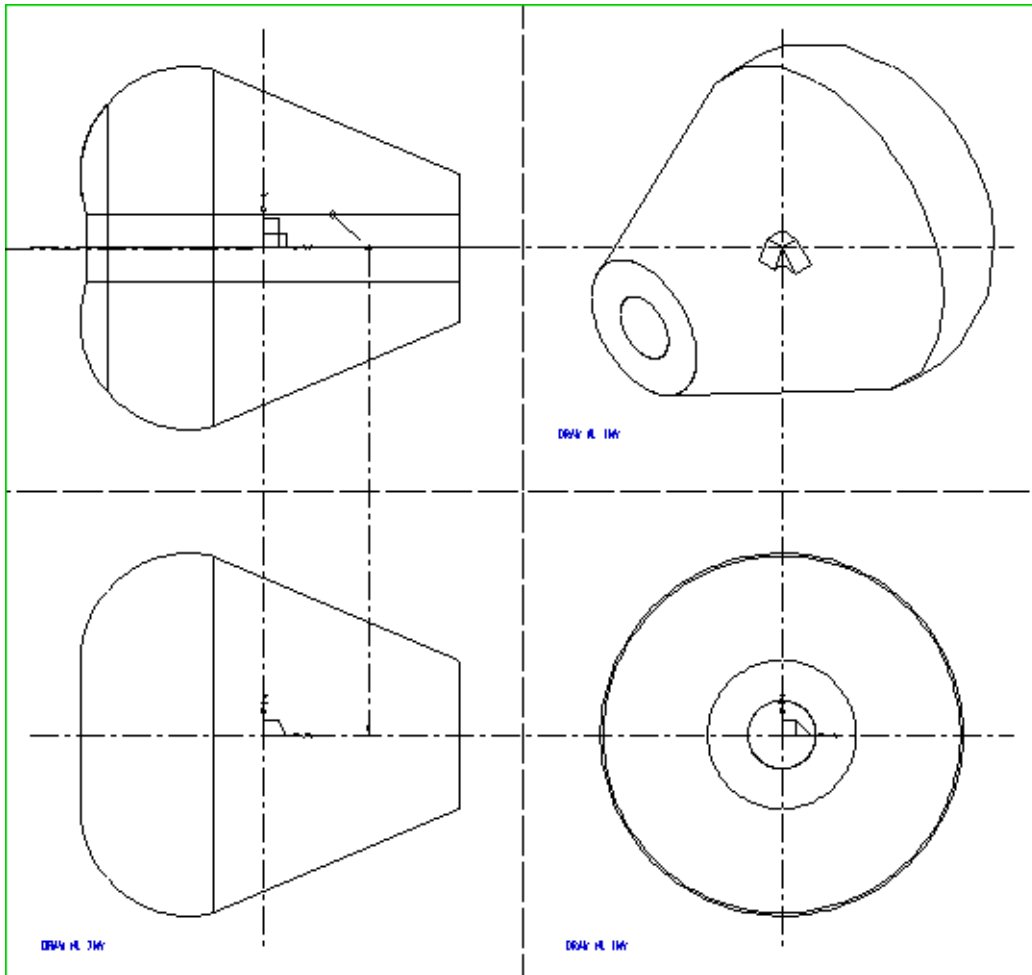
While drawing the link line the required point functions are added automatically. [Figure 72](#) shows the completed link line.

Figure 72 Definition of a Model Using a Closed Profile



6. Select the Model + Reconstruct tool  to generate and view the model. [Figure 73](#) shows the completed definition and model.

Figure 73 The Completed Model



Defining a Hole in a Volume of Revolution

It is possible to create a model of an object which has a hole or cavity around the centerline. This is done by defining a second profile within the outer profile. Although this technique produces an unrealistic model, such a model can be useful when modeling real objects using Boolean operations (see “[Boolean Operations](#)” on page 209). [Figure 74](#) shows an example of a definition of this type.

Use the following procedure to create this model:


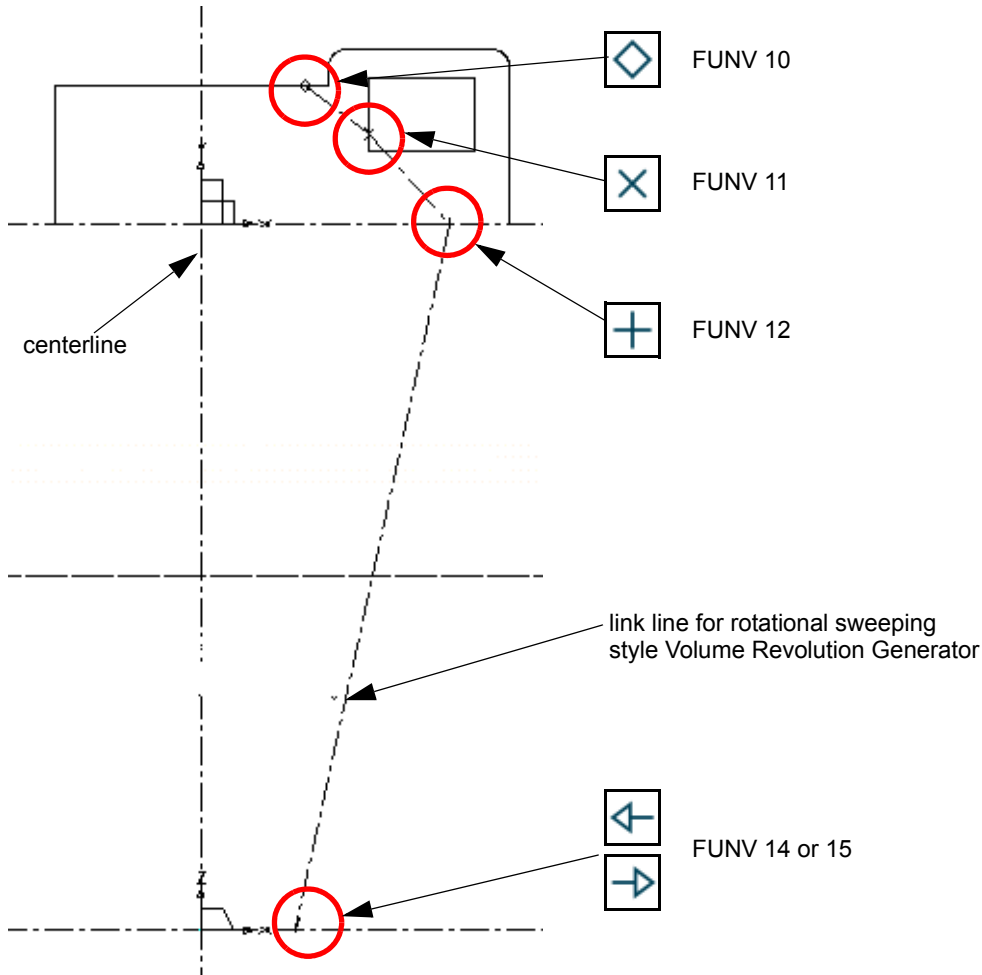
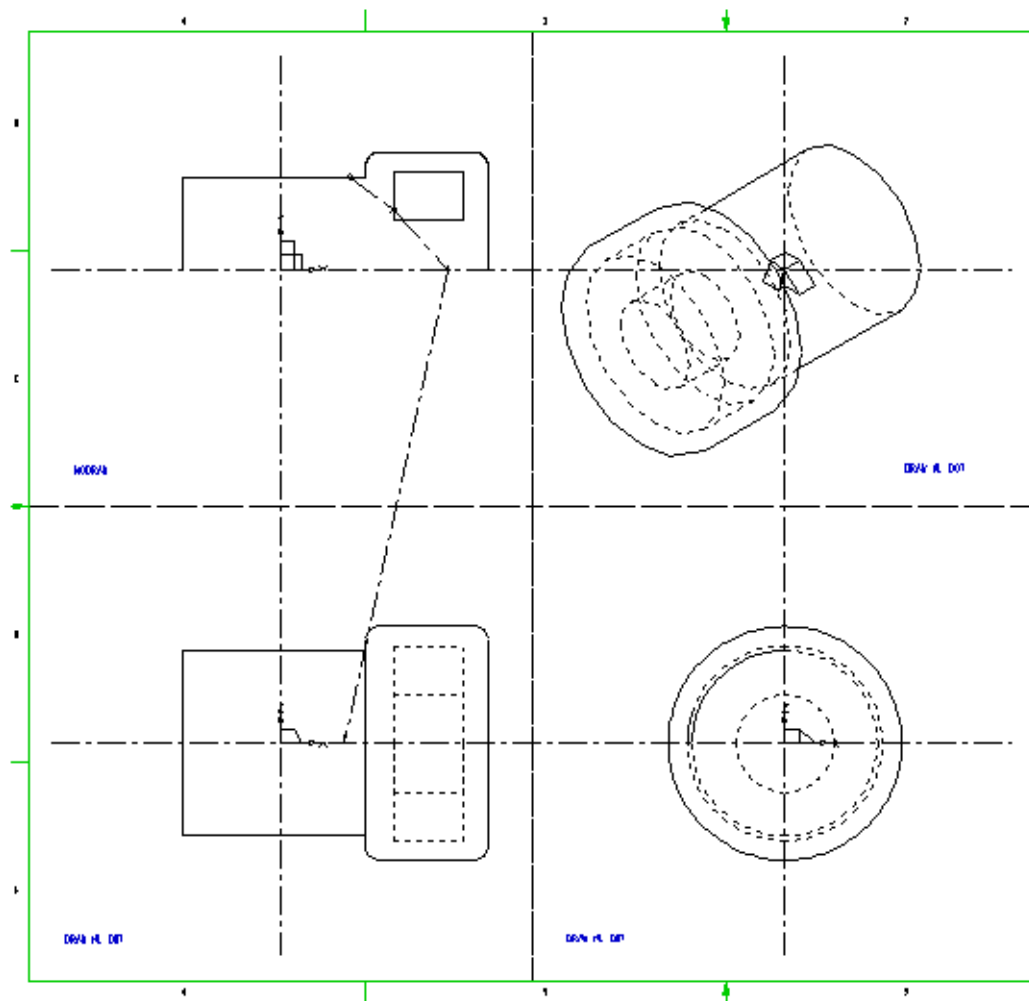
1. Draw the outer profile of the object as shown in [Figure 74](#).
2. Draw the profile of the hole inside the outer profile.
The line type does not need to be the same as the line type used to draw the outer profile.
3. Select the Volume Revolution tool  from the link lines toolset in the Lines tool group.
4. Click the LMB on the profile line to attach the link line to it.
5. Click the LMB on the center line to attach the link line also here.
6. Finish the link line by extending it into another viewbox with an orthogonal viewprim and then selecting `Exit Tool` from the popup menu.
While drawing the link line the required point functions are added automatically.
[Figure 74](#) shows the completed link line.

Figure 74 Defining a Hole Around the Centerline



7. Select the Model + Reconstruct tool  to generate and view the model. [Figure 75](#) shows the completed definition and model.

Figure 75 The Completed Model

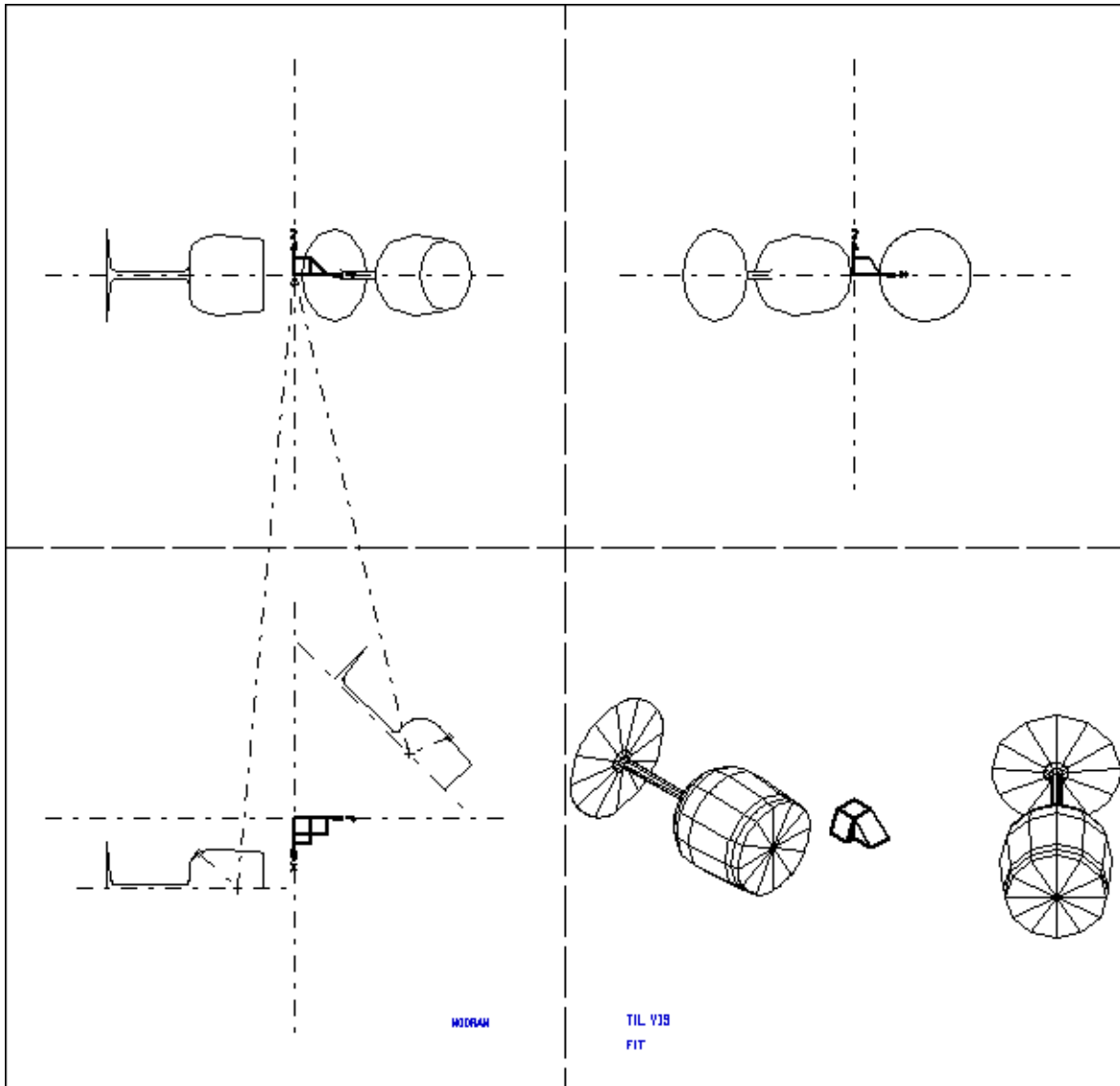


Please note: The Viewer command `DRAW HL DOT` is used to show hidden lines on the model.

Changing the Model Orientation


The orientation at which a model is created depends on the angle of the centerline in the view-box. [Figure 76](#) shows the definition and model of two glasses, one of which has its centerline drawn at 45° to the X axis.

Figure 76 Effect of Changing the Centerline Angle



Partial Volumes of Revolution

Partial volumes of revolution can be created by rotating the profile by less than a complete revolution. The use of this technique can avoid the need to use Boolean subtraction operations on a complete volume of revolution.

A partial volume of revolution is generated by defining the angle of rotation using a, TMG text  on the link line in the form:

```
ANGLE <start_angle> <finish_angle>
```

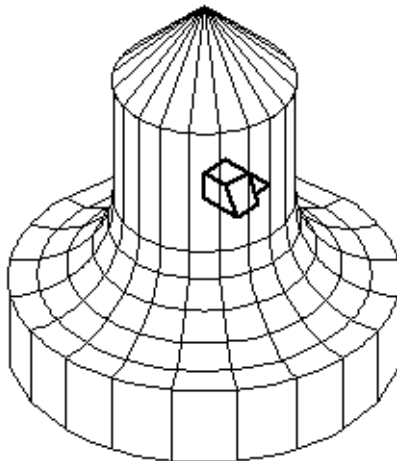
This has the effect that the profile is rotated around the centerline in a partial revolution that starts at the `start_angle` specified and ends at the `finish_angle`.

The datum plane for these angles is the half-plane onto which the profile is projected, and the sense of rotation is clockwise. The values of `start_angle` and `finish_angle` are taken as degrees and can be positive or negative. The TMG-text of style `Modeller Text ass. by Geometry` should be attached to the link line using a segment probe.

Please note: The half-plane is defined as that half of the plane to the right or left of the axis of rotation onto which the profile is projected. The start and finish angles are measured relative to this half-plane.

Figure 77 shows a model created from a complete revolution.

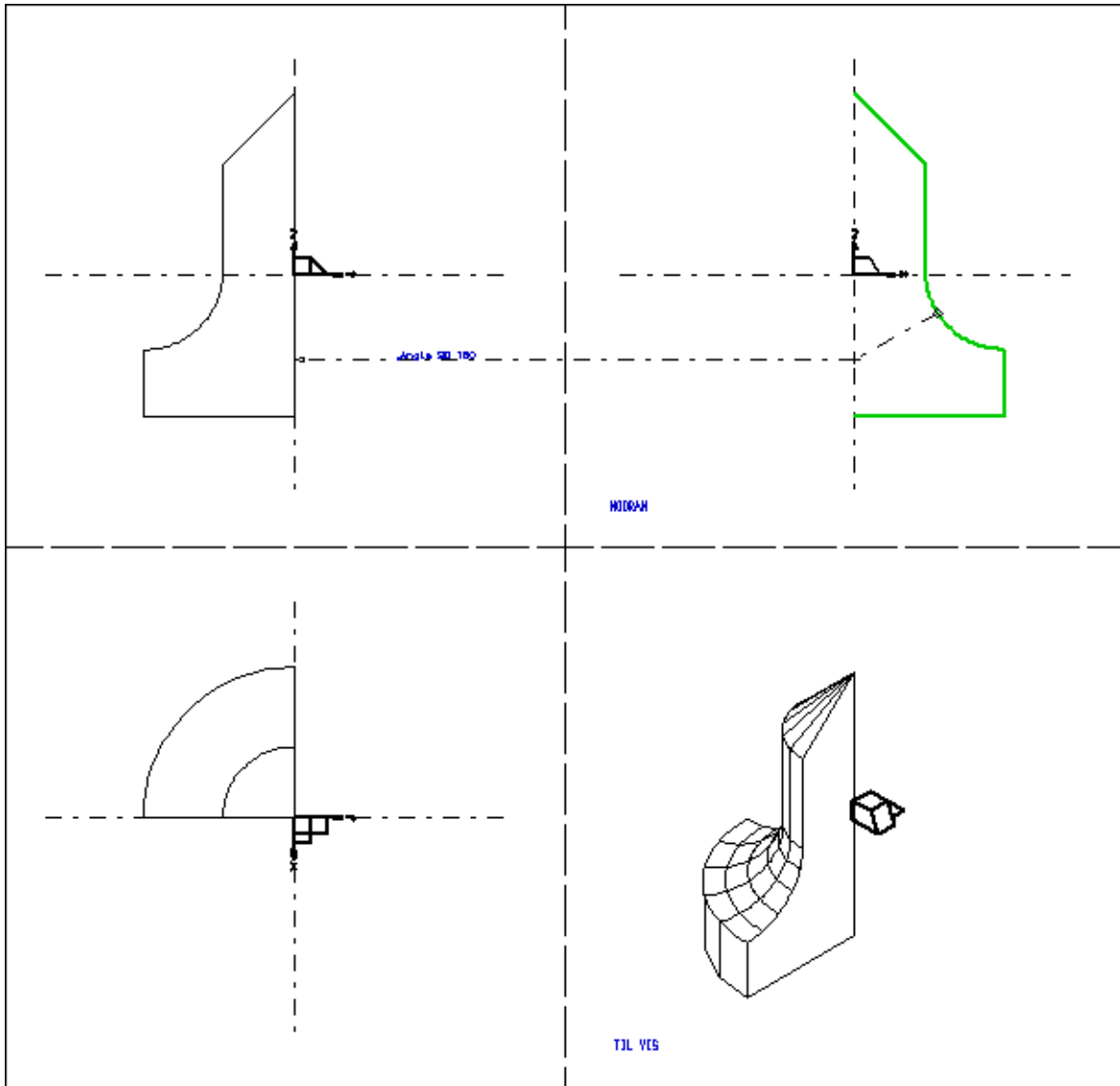
Figure 77 Complete Volume of Revolution



TIL V3S

Figure 78 shows a model created by rotating the same profile in a partial revolution from 90° to 180° . The datum plane in this case is an XZ-plane.

Figure 78 Definition and Model of a Partial Volume of Revolution



Please note: The half-plane in this example is that half of the XZ-plane to the right of the Z-axis (that is, in the positive X-direction).

Summary of Element Types

The following table gives a summary of the line types, text styles, and point functions used to create volumes of revolution.

Element Description		Element Style and Point Functions
Line Types		
	Profile lines	Profile LP0 - Profile LP9
	Link lines	Volume Revolution Generator
	Centerlines	center line
Point Functions		
	Pick up profile line	FUNV 10
	Pick up centerline	FUNV 12
	Pick up hole profile	FUNV 11
	Indicate third dimension on the link line	FUNV 14 or 15
Text Types		
	ANGLE command	Modeller Text ass. by Geometry (TMG)
	Depth text	Modeller Text ass. by Geometry (TMG)





RULED SURFACES

- General 112
- A Definition of a Simple Ruled Surface Model 113
- Matching the Profiles 116
- Creating Models Containing Holes 119
- Multiple Ruled Surfaces 121
- Summary of Element Types 123

General

A ruled surface model is created by linking two profiles with straight lines in that way that they enclose a volume. The profiles are drawn in one orthogonal viewbox connected together by a link line, which is extended into another orthogonal viewbox to define the position and height respectively depth of the model in the third dimension. The resulting model is built by the two parallel profile faces with tiles joining these faces at whatever angle a straight line can be ruled between them.

For creating ruled surface models two tools are available:

- The Ruled Sweep tool  creates a solid model defined by the two profile lines and the joining faces created automatically.
- The Ruled Edge tool  creates a model consisting of the joining faces only. The profile lines are not displayed.

The styles of the link lines are:

- Ruled Solid Generator
- Ruled Edge Generator

A Definition of a Simple Ruled Surface Model

To create a simple ruled surface model, follow the procedure described below.



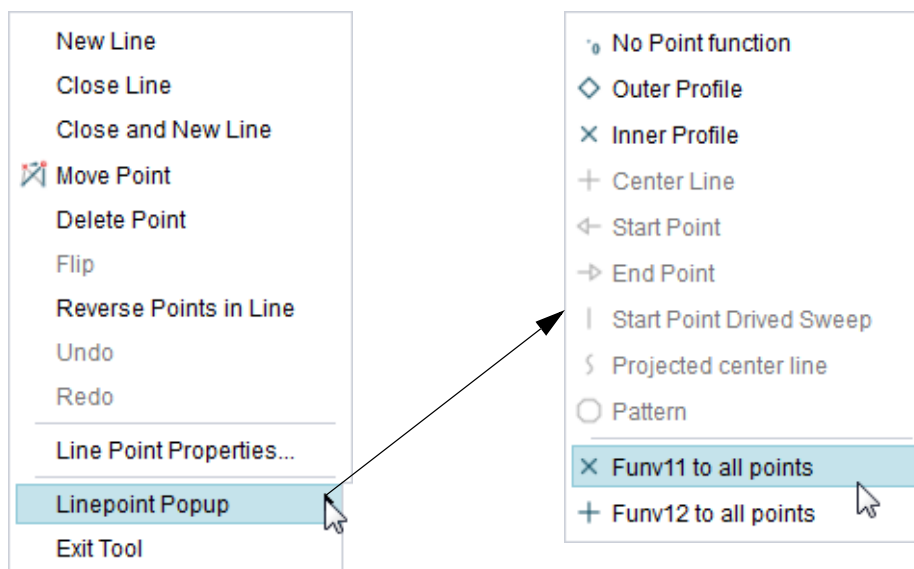

1. Draw two rectangular profiles as shown in [Figure 80, “Definition of a Simple Ruled Surface Model” on page 114](#).
 You can create the rectangles by using one of the profile line tools  setting point by point and close the line or you can choose the *Creates boxes* tool . Make sure that the rectangles consists of profile lines of style `Profile LP0` through `LP9` (see *Style in Dashboard*).
2. Assign point function 11 to all points of the profile lines by executing the following procedure (for the reason see [“Matching the Profiles” on page 116](#)):
 - a. Click twice on the profile line to set it into the edit mode.
 - b. Open the popup menu (RMB, see [Figure 79](#)).
 - c. Click on *Line Point Popup*.
 A second popup opens.
 - d. Choose `Funv11` to all points.
 The desired point function is promptly assigned to each point of the selected profile line.

Figure 79 The Right Mouse Popup Menu and the Line Point Popup Menu



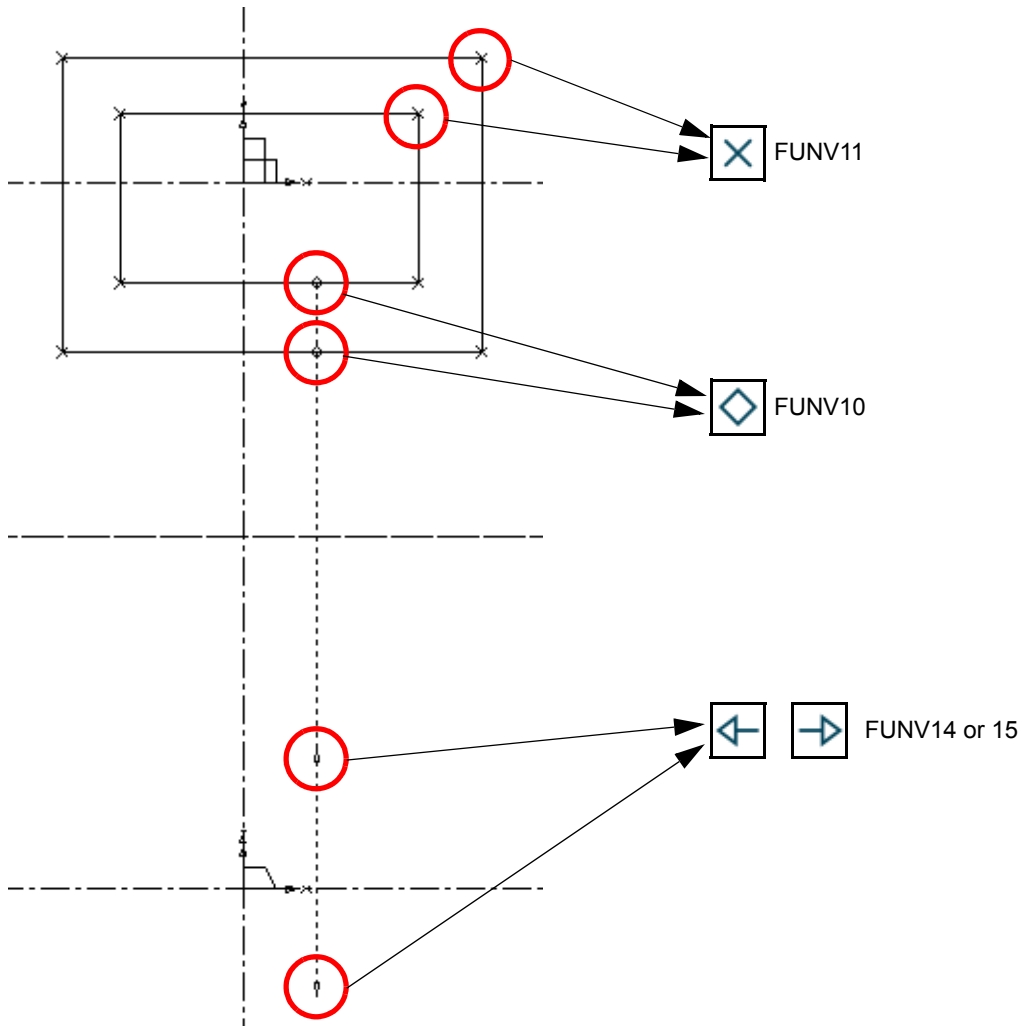
- e. Close the popup menu with `Exit Tool`.
3. Select the *Ruled Sweep* tool  and draw the link line as follows:
 - a. Click on the first profile line.
 - b. Click on the second profile line.
 - c. Move the mouse pointer into another view box.
 - d. Click a third and fourth time, to define the positions of the profile lines in the third dimension.

The link line is finished.

At each point the necessary point function was created automatically. At both profiles a diamond point function (FUNV 10) exists and the point functions 14 and 15 are at the end of the link line defining the positions of the profiles in the third dimension.

Figure 80 shows the result.

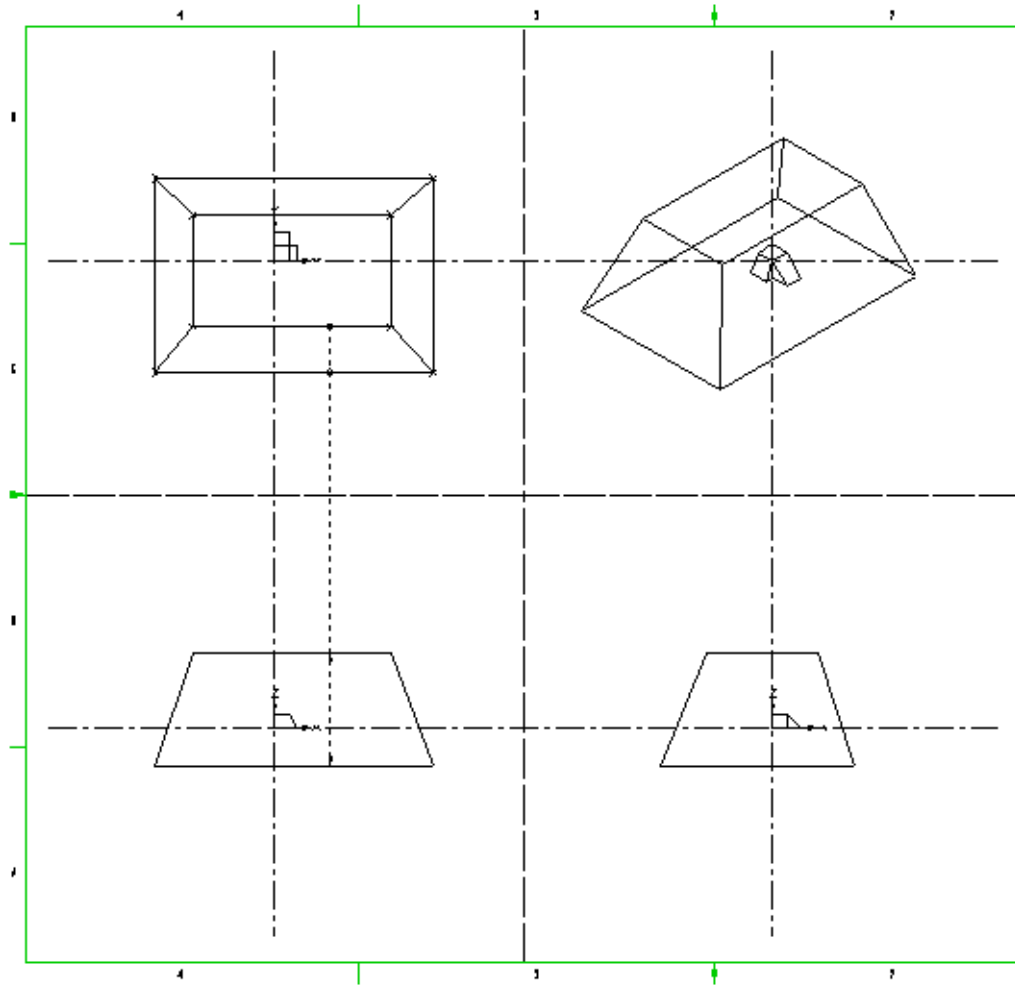
Figure 80 Definition of a Simple Ruled Surface Model



4. Choose the Model + Reconstruct tool  to generate and view the model.

Figure 81 shows the completed model.

Figure 81 The Completed Ruled Surface Model



Matching the Profiles

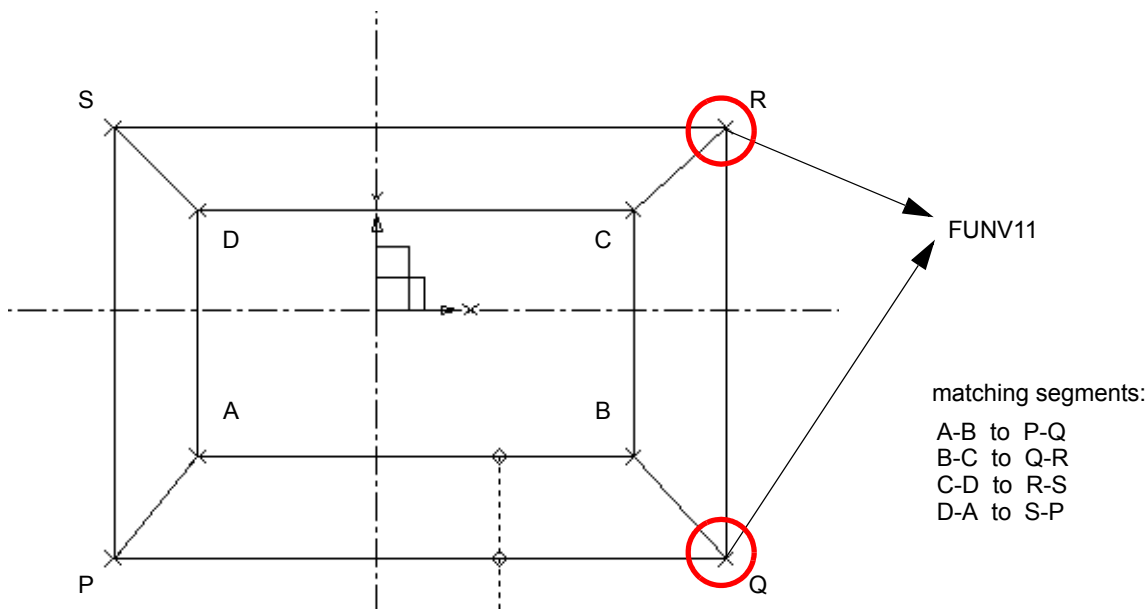
The Modeler creates a ruled surface model by drawing lines between the profiles to create a model. To do this, the length of the profile segments is calculated. Then the profiles are assigned to each other according to the ratio of the lengths. This method of matching the profiles can sometimes lead to unexpected results, therefore it is possible to assign points before modeling. following cases are differed:

- A point of one profile matches exactly a point of the other profile. Examples are provided in "Unique Point Assignment - FUNV 11".
- A point of one profile must be assigned to several points of the other profile. An example is provided in "Shared Points - FUNV 10" on page 117.

Unique Point Assignment - FUNV 11

When this method is used, the Modeler matches the profile segments between the fixed point functions in order, beginning with the two segments that are attached to the link line with diamond point functions (FUNV 10). [Figure 82](#) shows how the segments are matched. Segment AB is matched to segment PQ, BC to QR, and so on.

Figure 82 Matching Order for Segments



As a further example, [Figure 83](#) shows two profiles which differ in shape. The outer profile is a rectangle containing four corner points. The inner profile also has four corner points plus an arc (defined by three points) on one segment. To control matching of the profiles, cross point functions (FUNV 11) are added to the corner points of each profile. In this case the Modeler matches the profiles at each corner and creates three new points on the outer profile to match the three points that make up the arc on the inner profile. [Figure 84](#) shows how the arc blends into the outer profile.

Figure 83 Matching Points

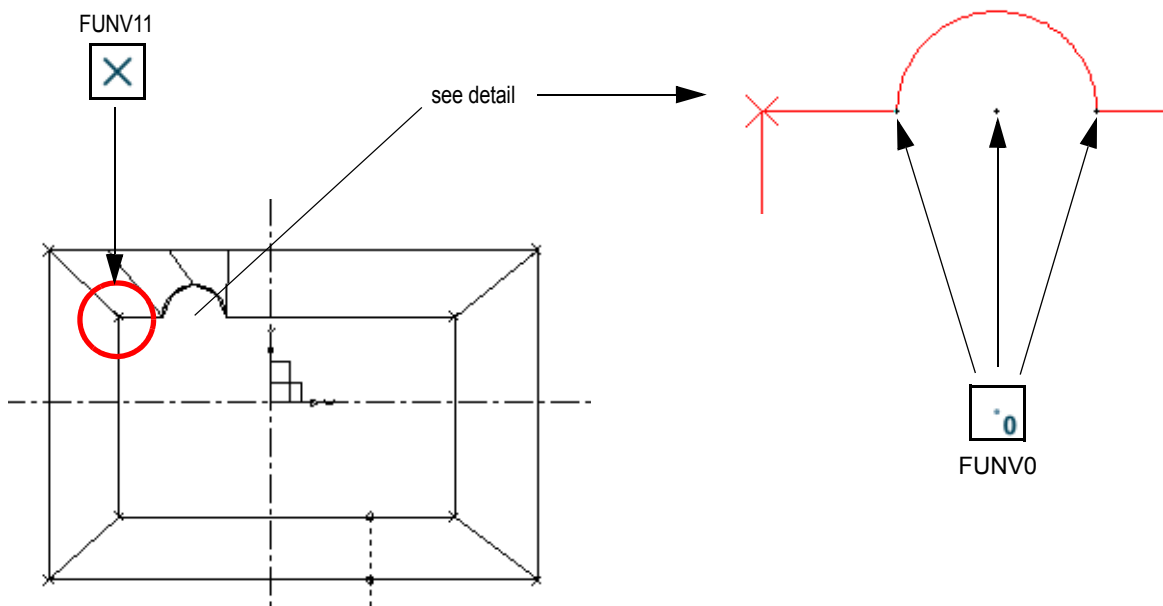
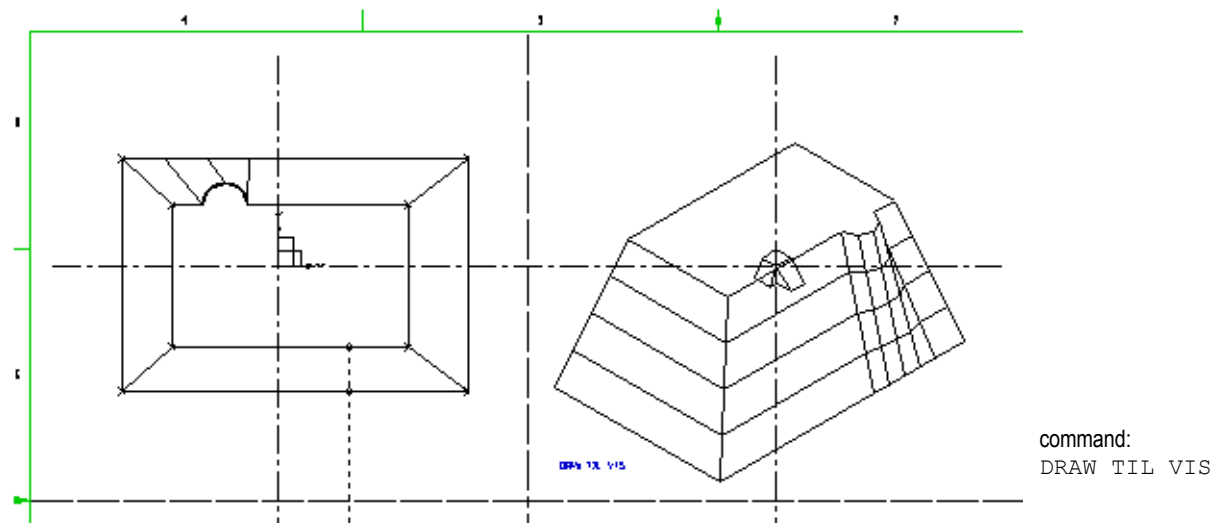


Figure 84 The Completed Model



Shared Points - FUNV 10

Another method of profile matching is the use of shared points. Shared points can be defined on profiles of differing shape so that they can be successfully matched. For example, [Figure 85](#) shows an inner profile drawn with three line segments and an outer profile of four segments. To match these profiles, point C of the inner profile can be shared by points 3 and 4 on the outer profile. You can do this by changing the point function at C to FUNV 10. [Figure 85](#) shows this.

Figure 85 Defining a Shared Point

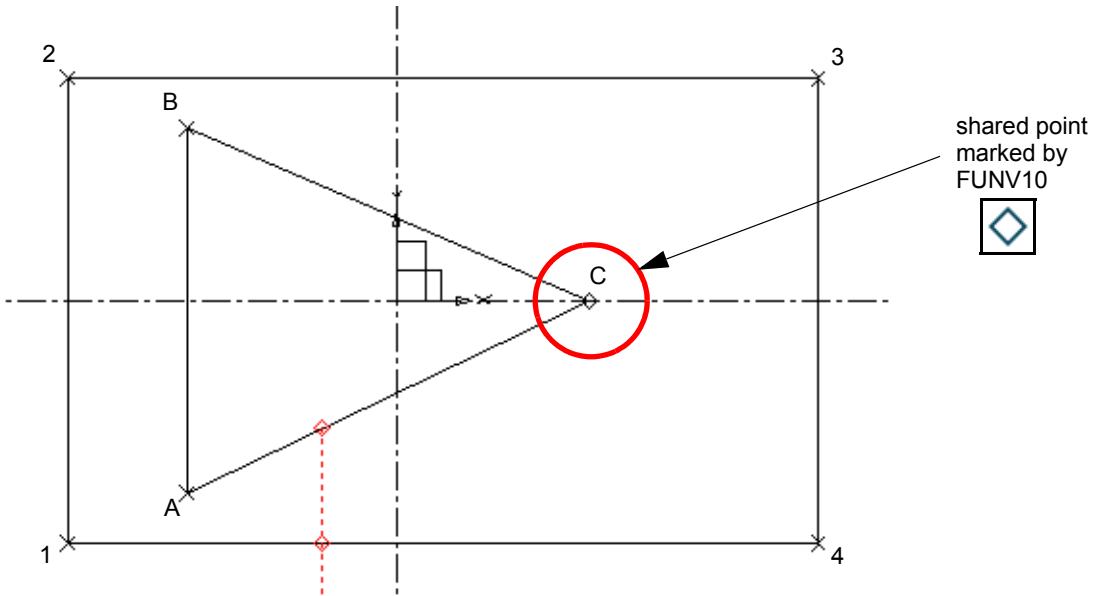
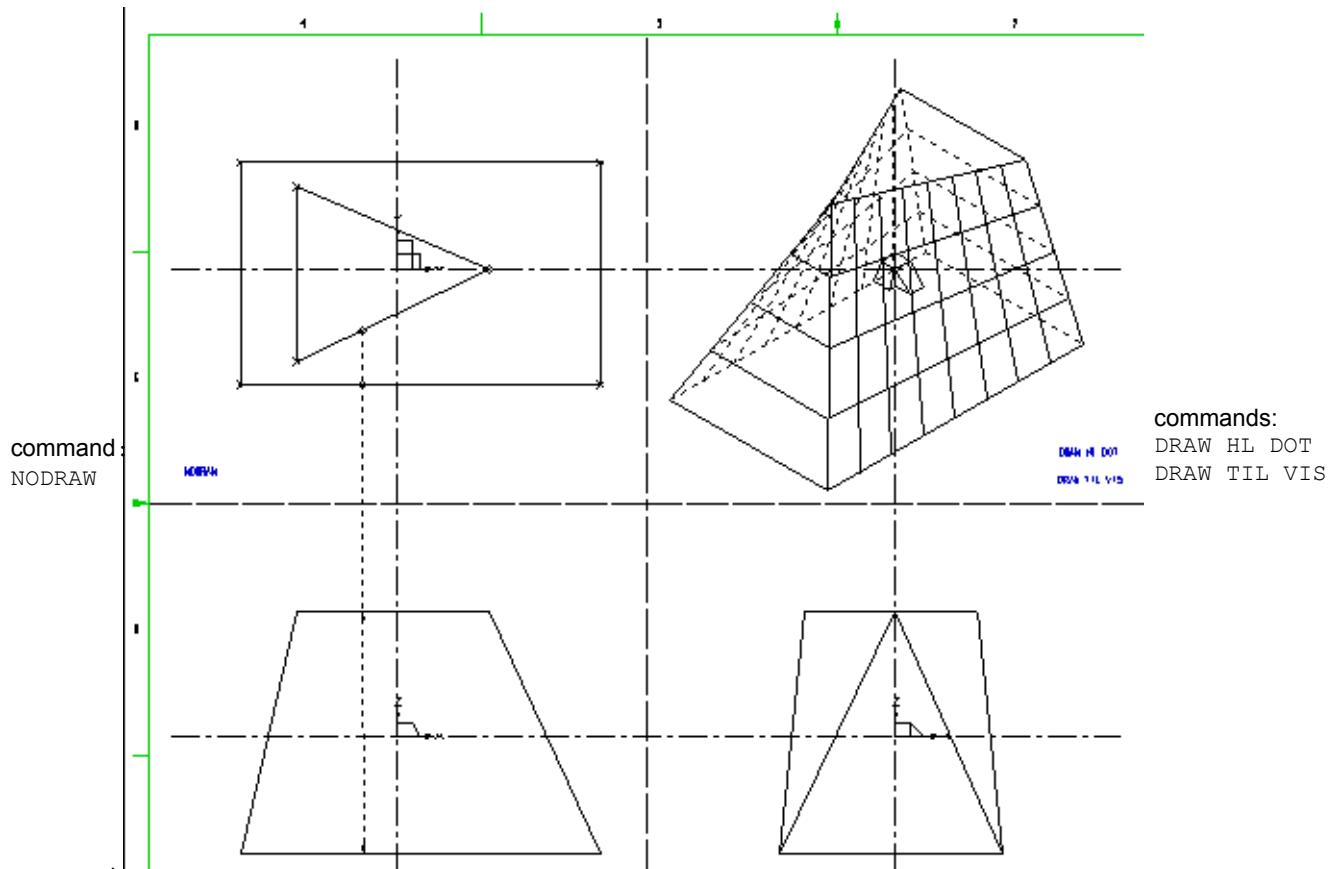


Figure 86 shows the model produced using this definition.

Figure 86 The Completed Ruled Surface Model



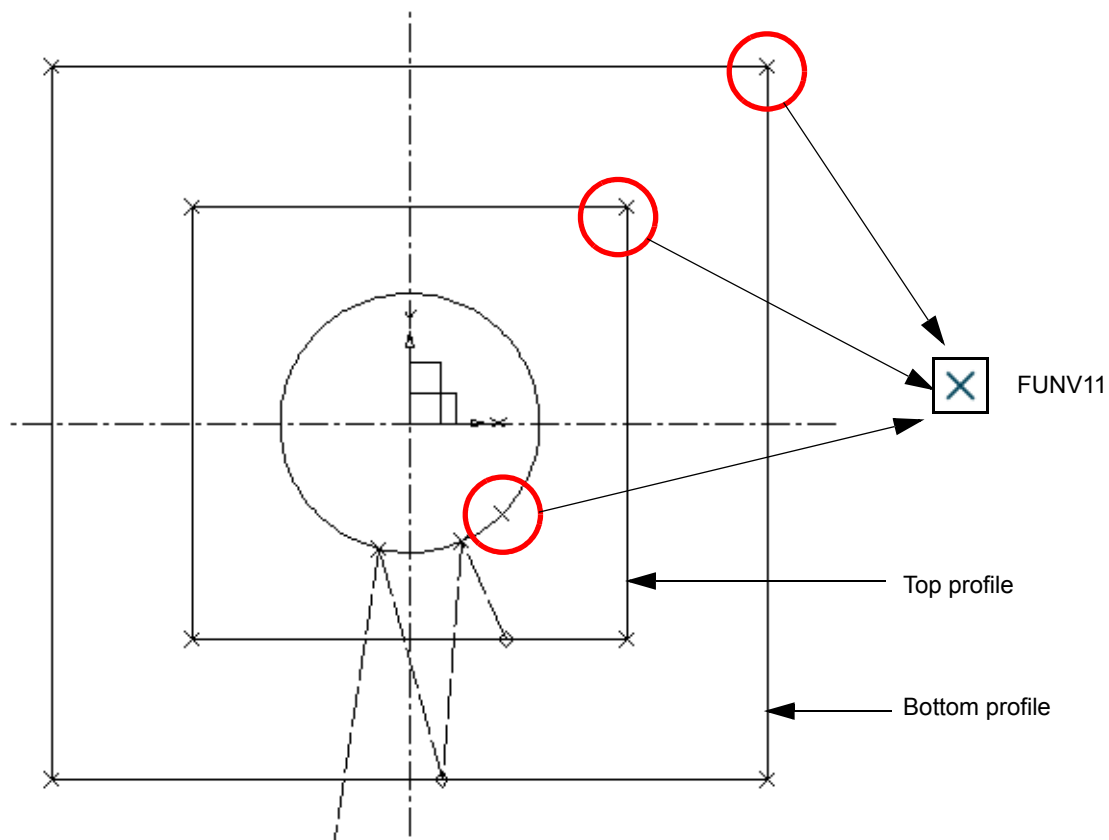
Creating Models Containing Holes

Ruled surface models can contain holes. When you define a hole in a ruled surface model you need to define the depth at which it starts and finishes and the corresponding profile. For this there are two possibilities:

- Drawing two separate profiles for the top and bottom of the hole
- Attaching the link line twice to a single hole profile

The procedure described below constructs the definition for a model containing a hole using the second method. [Figure 87](#) shows the definition.

Figure 87 Defining a Hole Using the Same Profile Twice



1. First draw the rectangular profiles of the model using a profile line type of LP0 through LP9.
2. Add a cross point function (FUNV 11) to each corner of the profiles for assigning the vertices explicitly.
3. Draw a single hole profile using a line type of LP0 through LP9.
4. Add a cross point function (FUNV 11) at the circumference point of the circle.
With this you define that the lines which create the hole face are parallel from start (first profile) to the end (second profile).



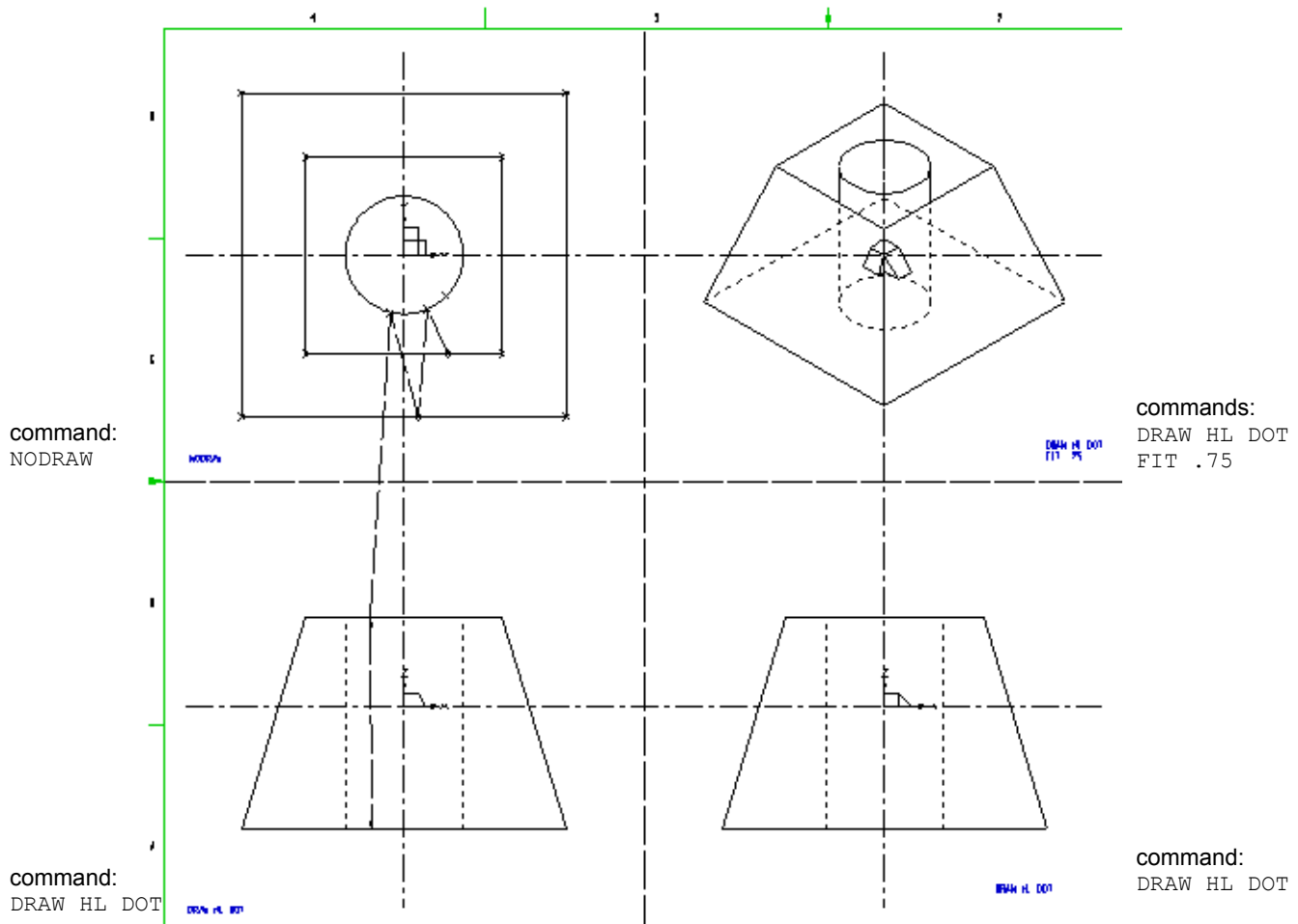
5. Choose the Ruled Sweep tool .
6. Draw the link line as follows:
 - a. Click on the small rectangle to define the top profile.
 - b. Click on the circle.
 - c. Choose the entry Linepoint Popup from the popup menu.
 - d. Choose Inner Profile from the popup menu, to define the circle as hole of the first profile.
 - e. Click on the big rectangle to define the lower profile.
 - f. Click on the circle to define it as hole of the second profile.
The point function Inner Profile (FUNV 10) will be set automatically.
 - g. Move the mouse cursor into another view box.
 - h. Click twice there to define the positions of upper and lower profile.
Also here the point functions (FUNV14 and 15) are set automatically.
7. Finish drawing the profile line with Exit Tool.
8. Choose the Model + Reconstruct tool  to generate and view the model (see [Figure 88](#)).

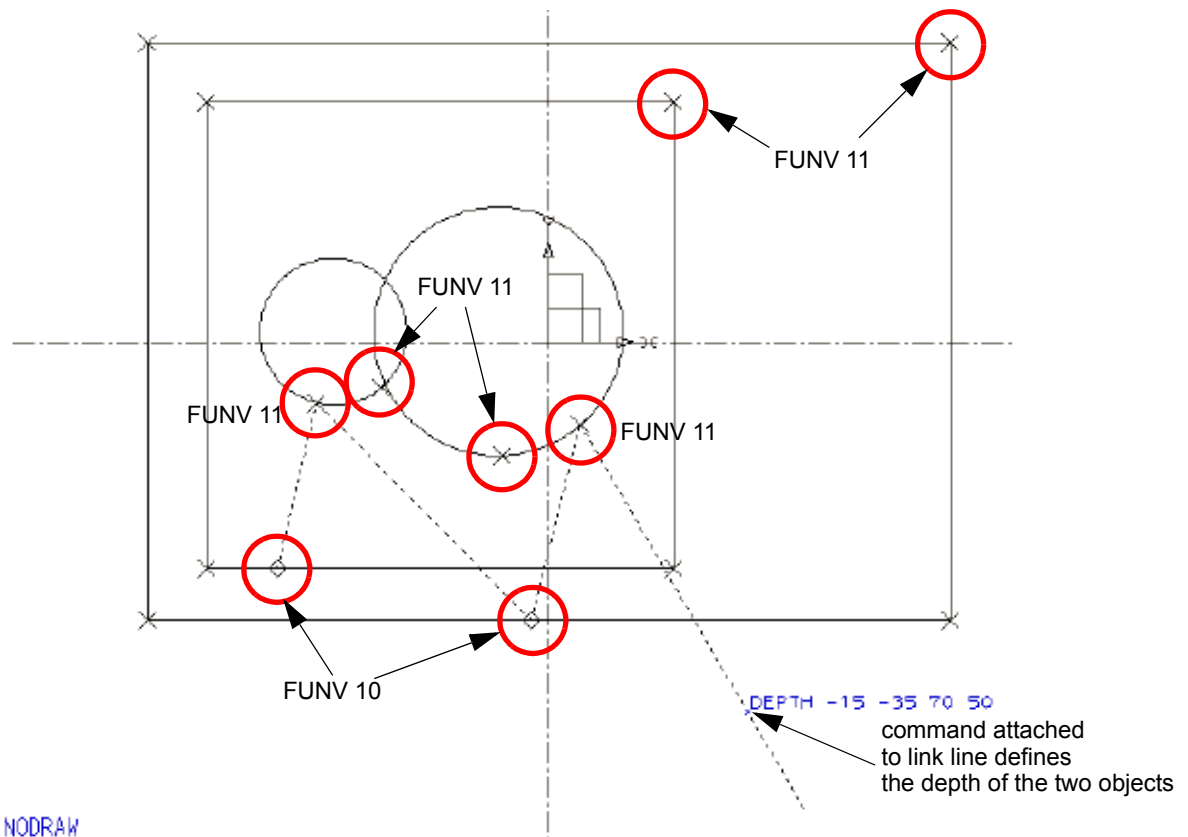
Figure 88 The Completed Model



Multiple Ruled Surfaces


You can create multiple ruled surface models using either additional arrowhead point functions or depth texts on the link line. [Figure 89](#) shows the definition of a multiple ruled surface by depth text.

Figure 89 Definition of a Multiple Ruled Surface Model



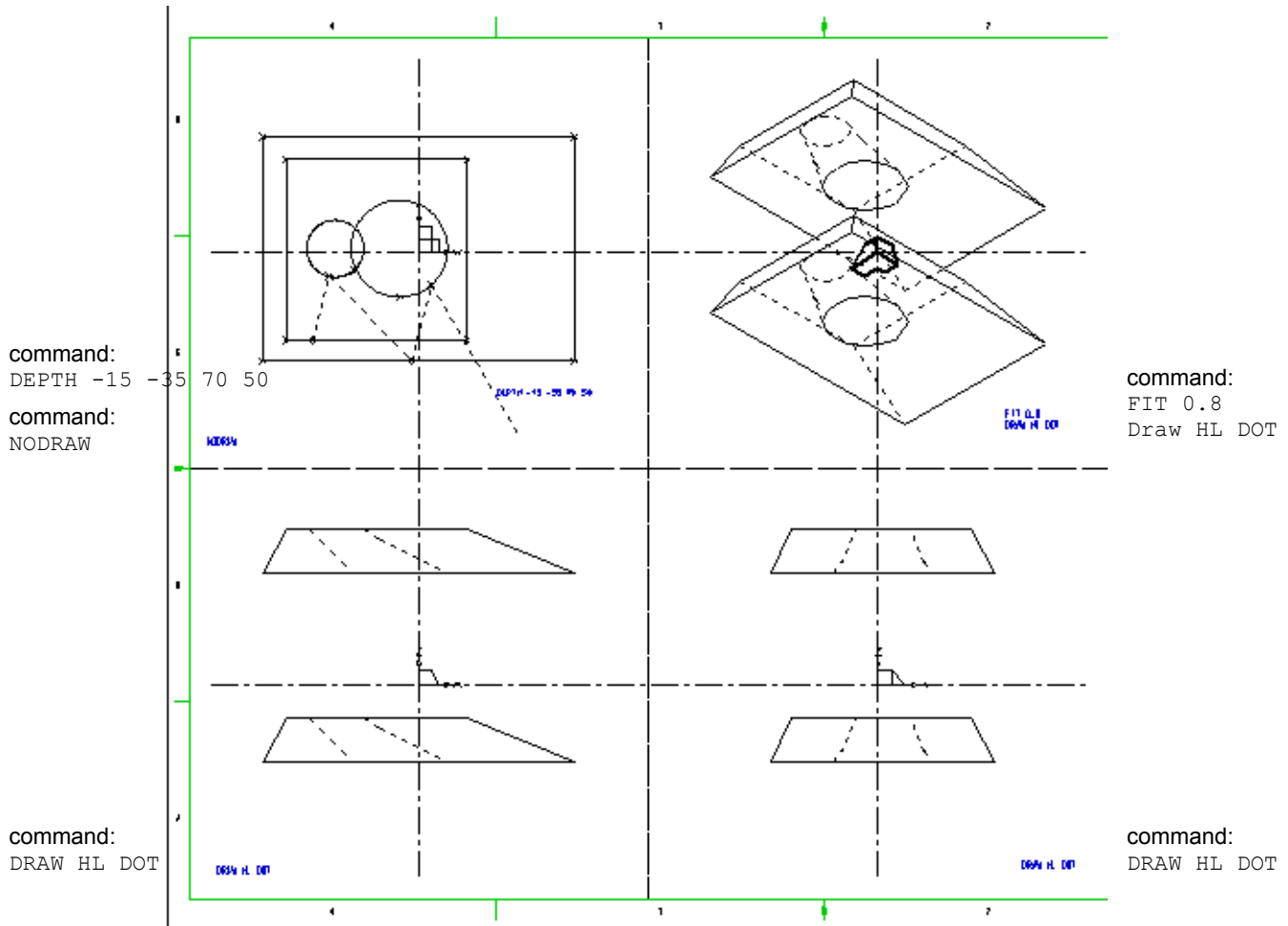
Both the vertices of the two rectangles and the two circles at the circumference point of the circle have cross point function (FUNV 11).

The link line is created with the Ruled Sweep  tool. The link line ends in the same view box which contains the profile lines and its endpoint has the point function 0 . This is possible because the position of the profiles is defined by depth text.

The position of the profiles is specified by a depth text of style Modeller Text ass. by Geometry (tool Create TMG text ) which is attached to the link line. Two ruled surface models will be created, the first two values define the position of one and the second two values the position of the other object.

[Figure 90](#) shows the resulting solid model of multiple objects generated using depth text.

Figure 90 A Multiple Ruled Surface Model



Summary of Element Types

The following table gives a summary of the elements with styles and point functions used to create ruled surface models.

Element Description	Element Style and Point Function
Line Types	
Profile lines	Profile LP0 - Profile LP9
Link Lines	
for Solid Models	Ruled Solid Generator
for Ruled Surface Models	Ruled Edge Generator
Point Functions	
Pick up profile lines	FUNV 10
Pick up hole profiles	FUNV 11
Indicate third dimension on the link line	FUNV 14 / FUNV 15
Match profile line segments	FUNV 11
Shared points of profile lines	FUNV 10
Text Types	
Depth text	Modeller Text ass. by Geometry (TMG)



SLIDES

The Slide generator creates a solid model by sliding a profile around a path which is defined in the same way as for wire models (see “Wires” on page 151). The slide path is defined by one or two centerlines (of style `centre line`, type `LCL`) which can be open or closed. The profile defines a cross-section of the model at a user-defined point on the path.

The profile is drawn in a separate viewbox to that of the path and a link line (of style `Slide Generator`, type `LSL`) is used to fix the position of the model in the third dimension.


There is a point in the plane of the profile which is considered to be attached to the path during the slide. This point is always the datum point of the viewprim in the profile viewbox.

- [A Simple Definition.....](#) 126
- [Defining the Start Point](#) 129
- [Modeling Techniques](#) 134
- [Summary of Element Types.....](#) 142


A Simple Definition

The simple definition below is an example of how a solid model is created using the Slide generator. [Figure 91](#) shows the definition. [Figure 92](#) shows the definition and completed model.

This is the procedure to follow:

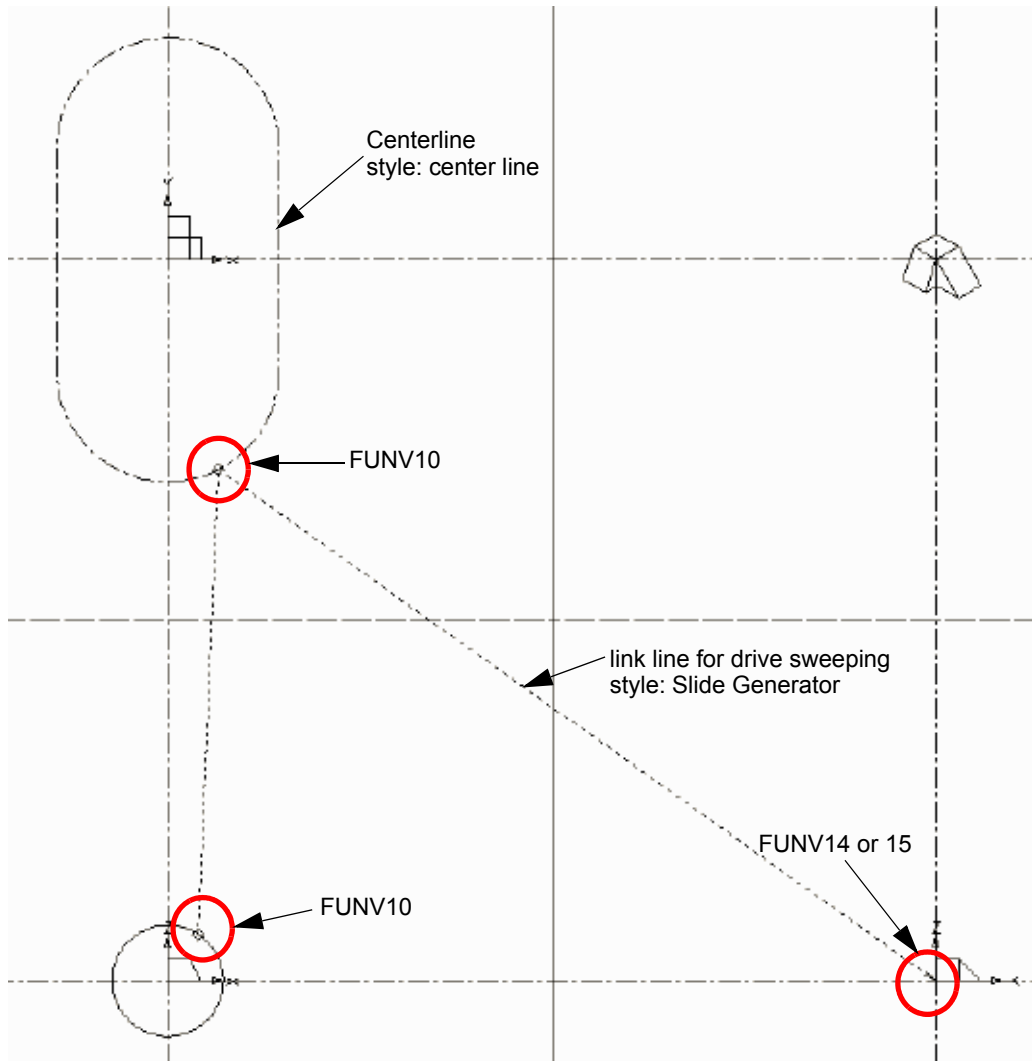
1. Draw a centerline using the Create Center Lines tool  to define the slide path as shown in the figure below.

Please note: Curved centerlines must be constructed using **tangent-point arcs**.

2. Draw the profile in another orthogonal viewbox, using any line of type LP0 through LP9. Note the shape of the slide profile and its position relative to the viewprim as shown in the following figure.
3. Select the Slide tool 
4. Click the LMB on the profile line.
5. Click the LMB on the center line.
6. Click the LMB in another viewbox with an orthogonal viewprim to define the position of the model in the third dimension.

The link line consists of three points now, see [Figure 91](#). The points of the profile and center line have the diamond point function (FUNV 10). The last point has the simple arrow point function (FUNV 14 or FUNV 15).

Figure 91 A Simple Definition




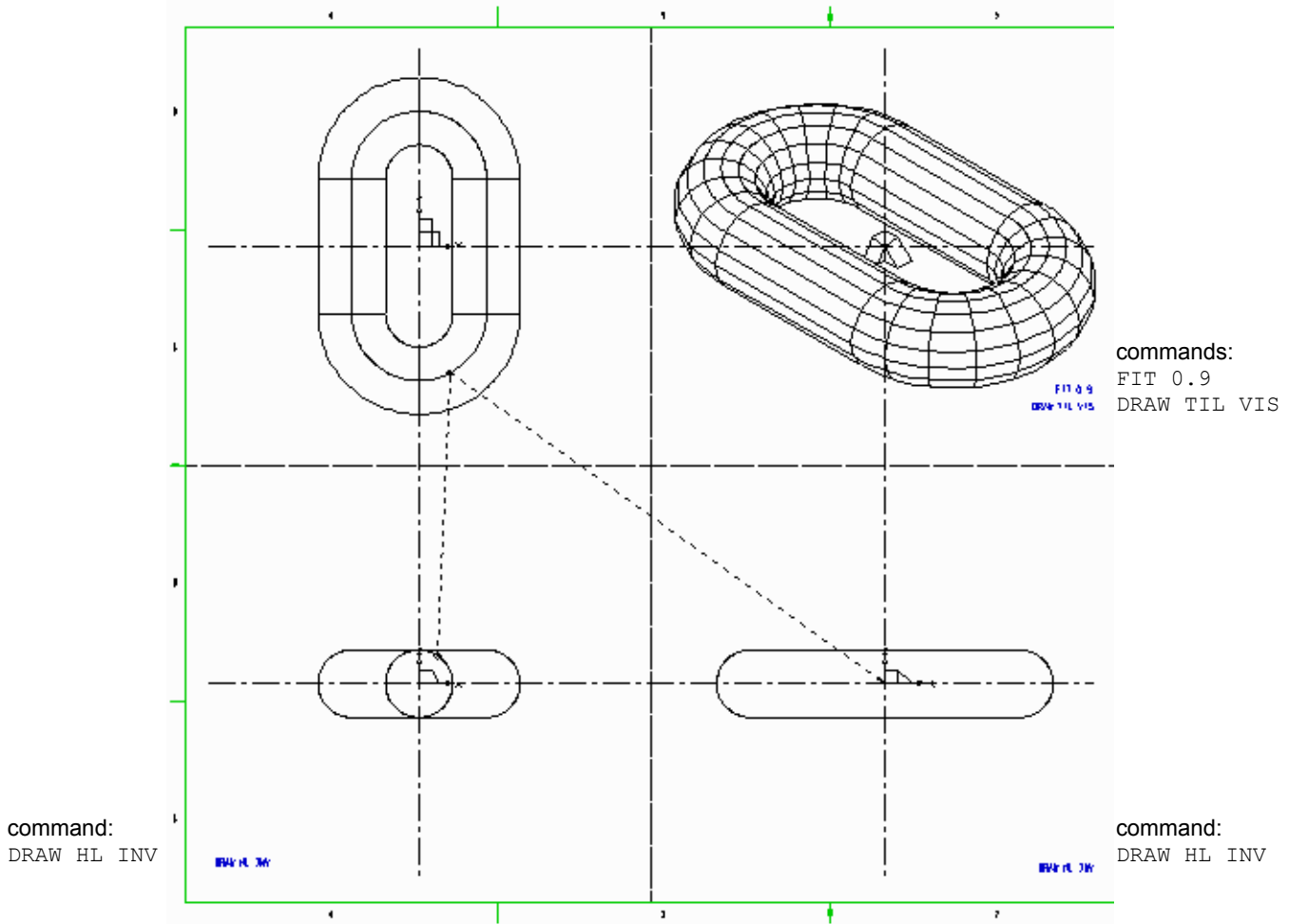
7. Select the Model + Reconstruct tool  to produce the completed model as shown in Figure 92.

Figure 92 The Completed Model



Please note: When creating a model with the Slide generator, the size of the profile in relation to the curvature of the centerline is important. If the profile is too large the resulting model will intersect itself, and this warning message will be displayed:

Warning: object intersects itself

This means that the resulting model is not valid in the strictest sense, although it may pass through the Model Validator.

Defining the Start Point

When the Slide generator produces a model, it loads the profile at the start point on the path and slides the profile around the path. By default, the slide start point is the first point that you create when drawing the centerline.

However, you can override the default start point by adding a tick point function (FUNV 13) to a point on the centerline. The profile will then be forced to start sliding from this point.

Please note: The tick point function does not have to be added to an existing point on the centerline. A new point can be added to the line for this purpose.

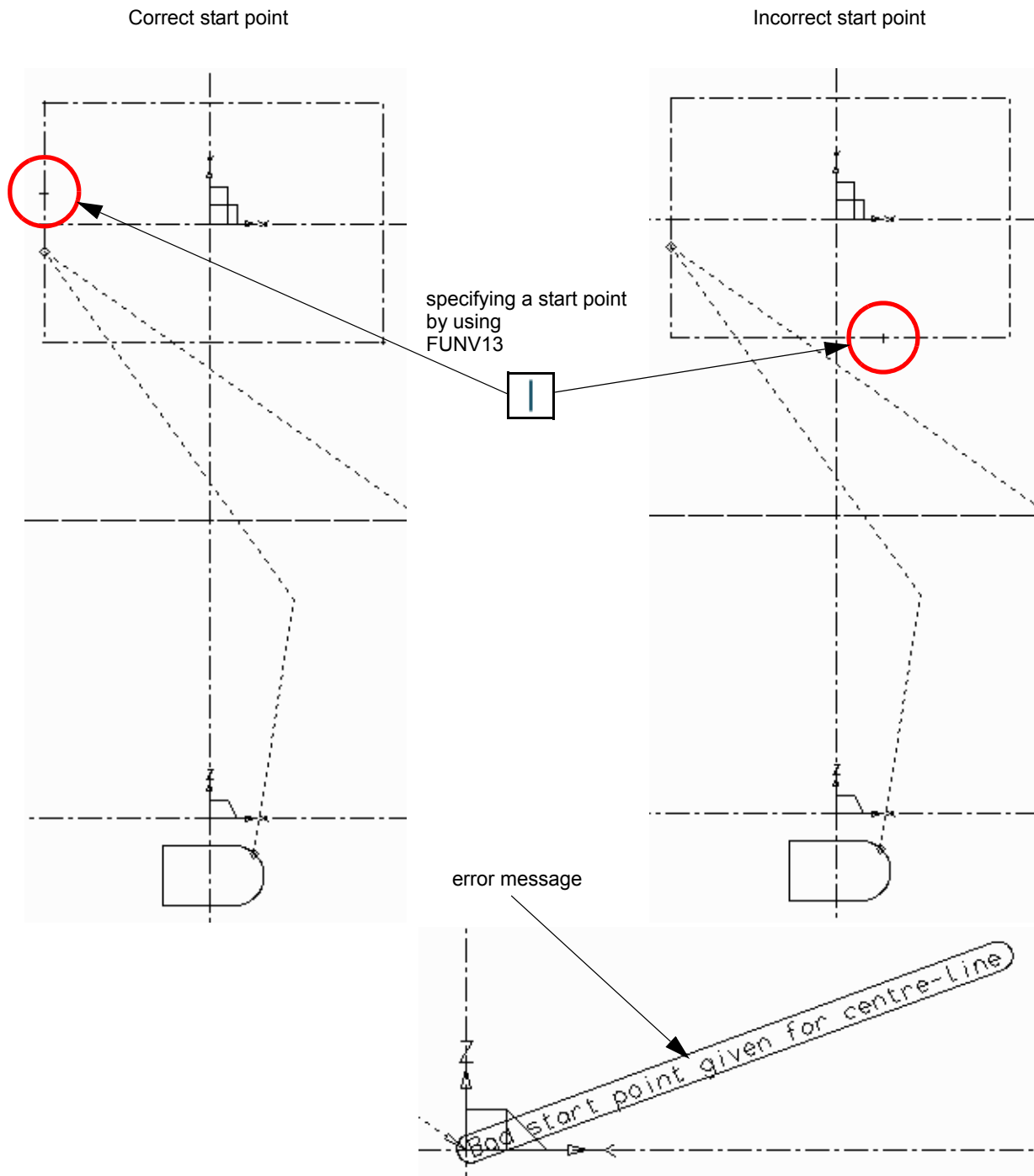
The position of the start point is important for the following reasons:

- The profile determines the cross-section of the model at the start point, and therefore the cross-section will (in general) vary with the position of the start point because the orientation of the profile is fixed.
- If the profile is attached so that it is parallel with the slide path, the Modeller cannot slide the profile. If this happens the following error message will be displayed:

```
Bad start point given for centerline
```

For example, [Figure 93](#) shows an example of a correct and an incorrect start point.

Figure 93 A Correct and an Incorrect Start Point



If the profile is loaded at the incorrect start point the Modeller displays the error message. This is because the profile defined in the DZX viewbox would create a section parallel to the center-line in the DXY viewbox.

Figure 94 shows the model generated from the correct definition.

Figure 94 The Completed Model

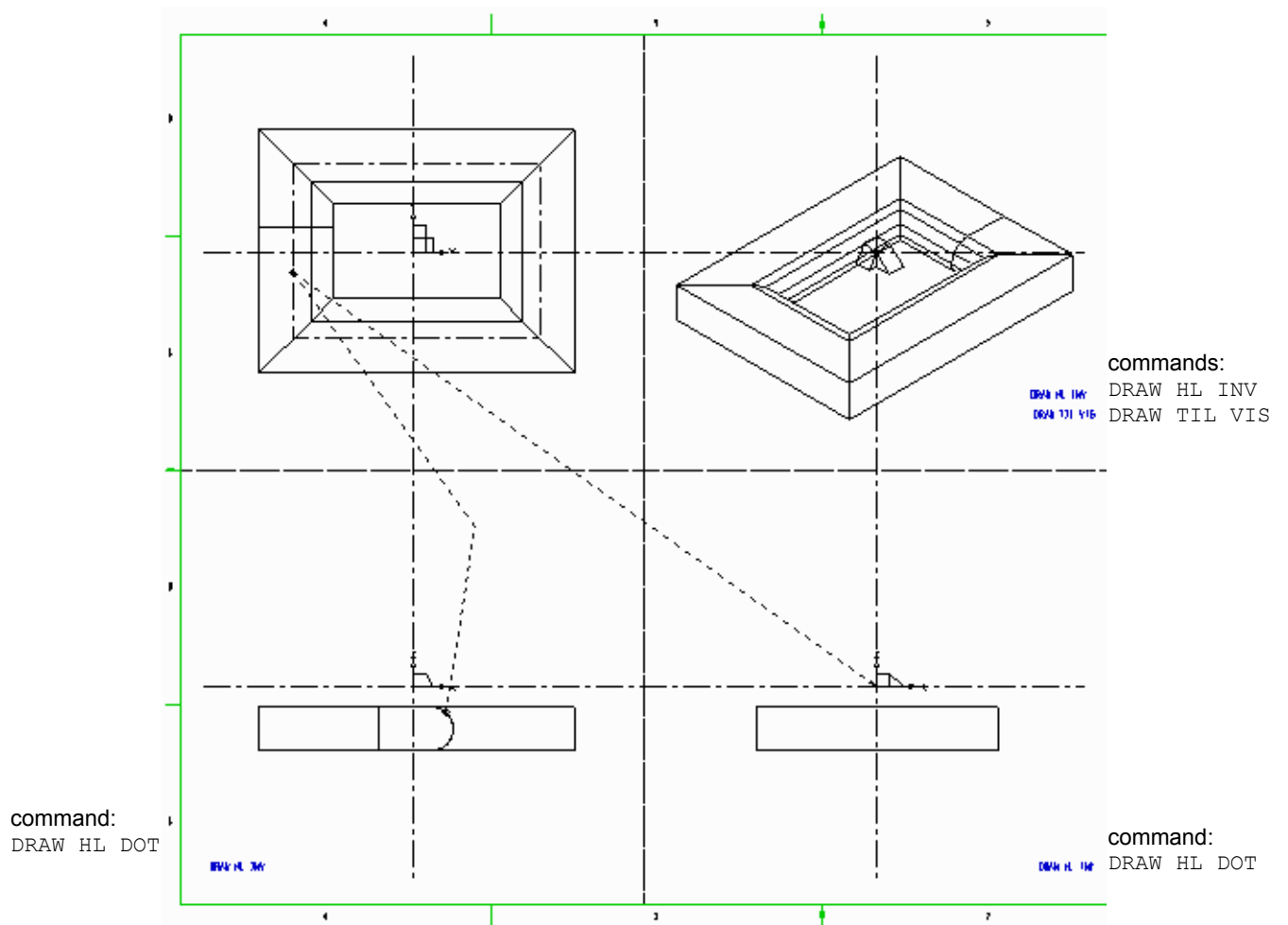


Figure 96 shows another correct position for the slide start point and the model generated from this definition.

Figure 95 **Another Example for a Correct Start Point**

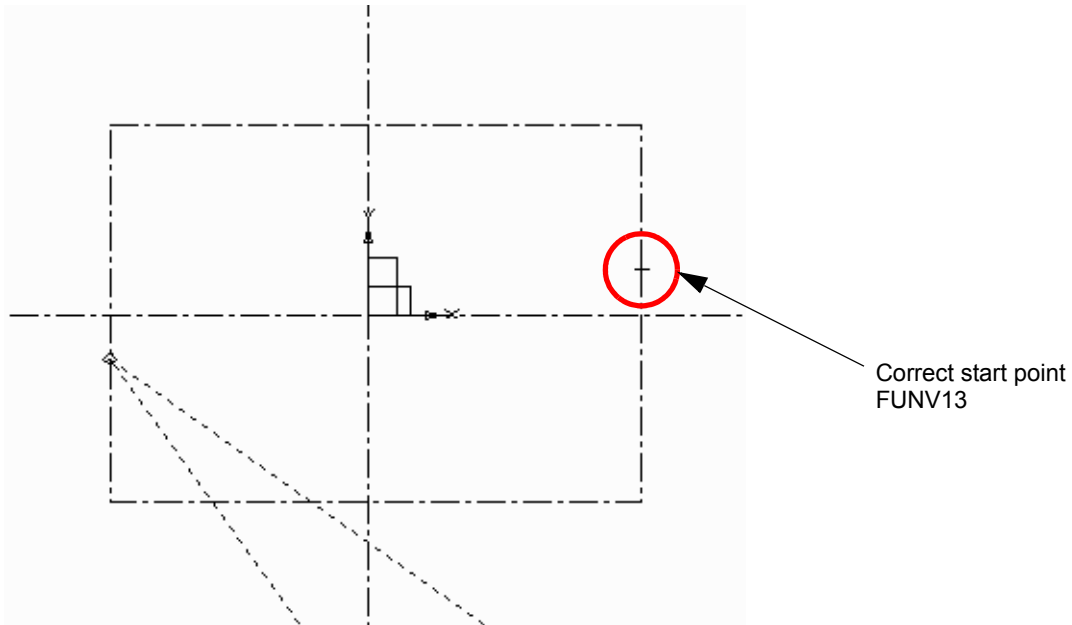
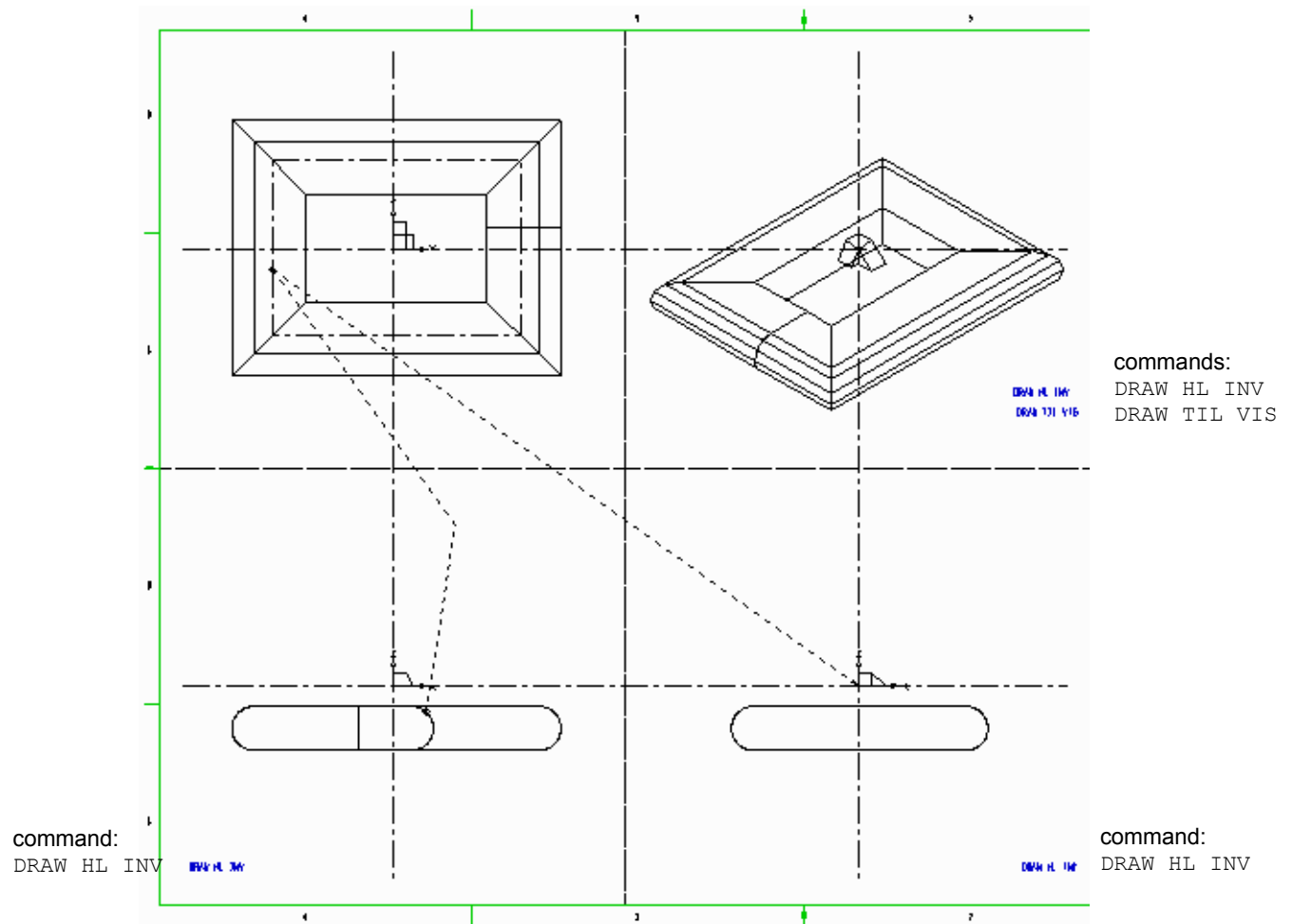


Figure 96 Repositioned Start Point and the Resultant Model



Note the difference between this model and the model shown in [Figure 94](#). In [Figure 94](#) the model was generated with the rounded edge of the profile turned inwards, but by changing the start point, as shown in [Figure 95](#), the model is generated with the rounded edge turned outwards (see [Figure 96](#)).

Modeling Techniques

This section shows how to create slides using the following techniques:

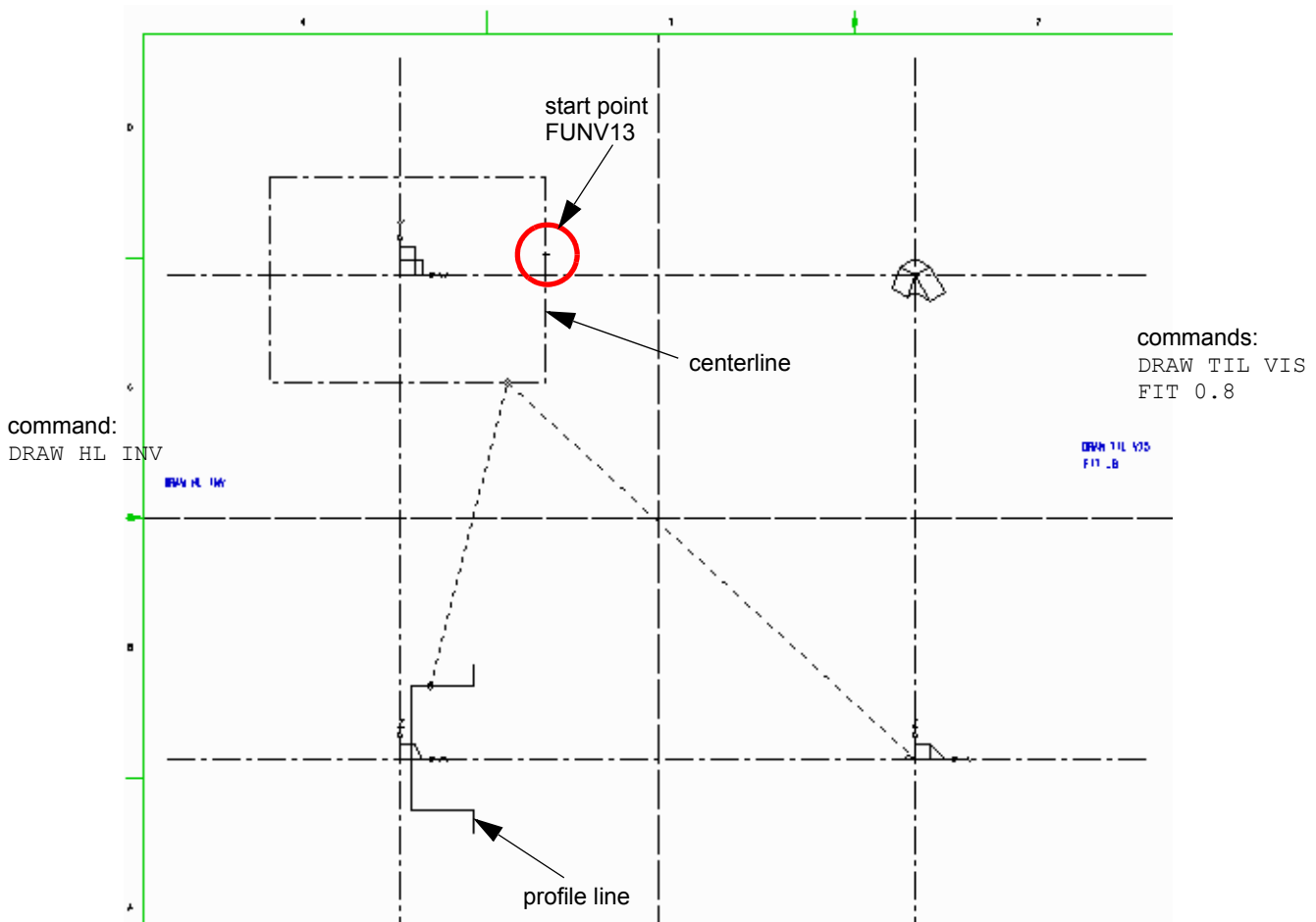
- "Open Profiles"
- "Open Centerlines"
- "Non-planar Paths"
- "Primary Centerlines and Projection Surfaces"

Open Profiles

An open profile can be used to generate a solid model with the slide generator, but only in conjunction with a closed, planar centerline.

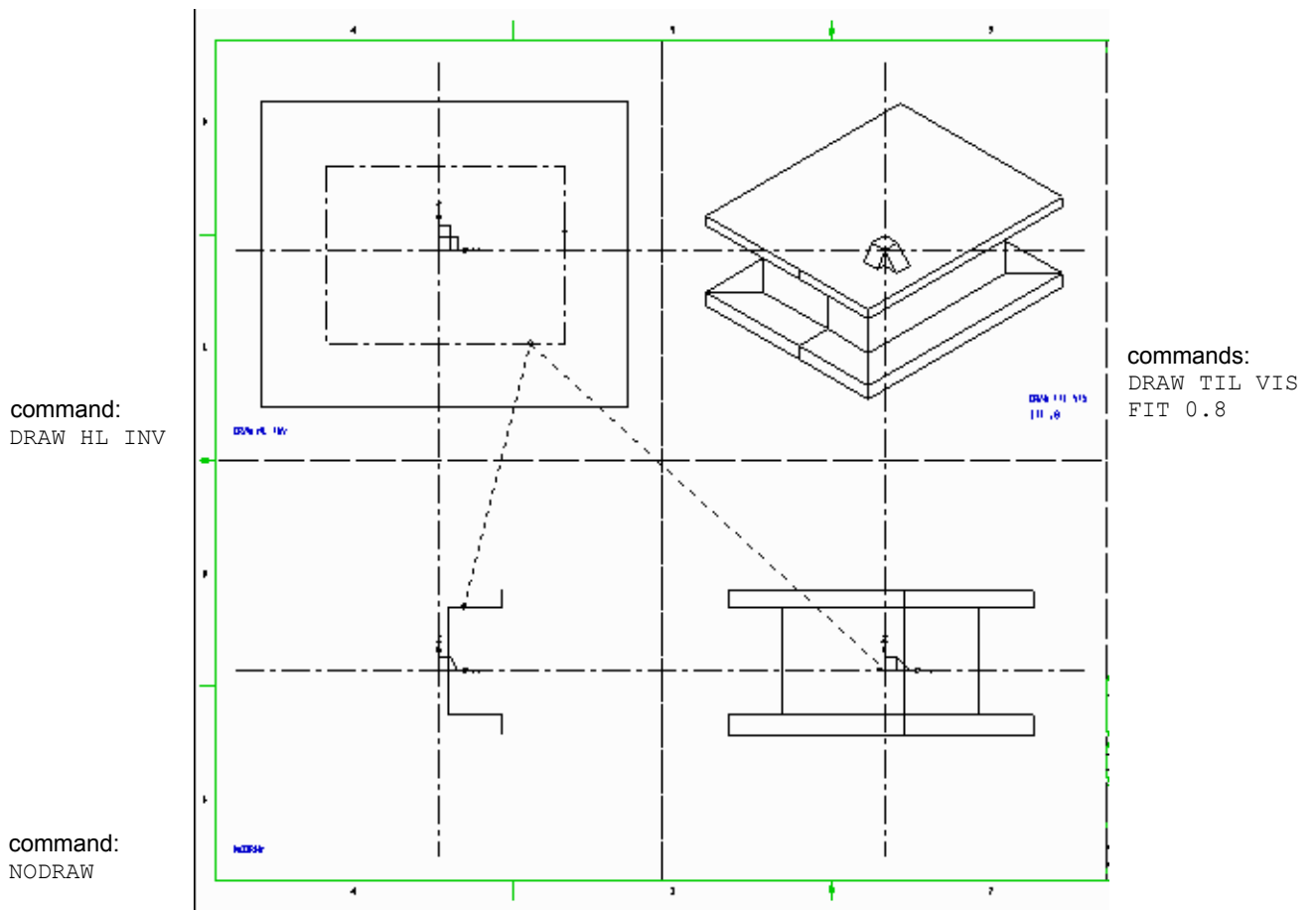
Figure 97 shows an example of an open profile definition.

Figure 97 A Model Definition Using an Open Profile and a Closed Centerline



The definition above generates a solid model even though the top and bottom surfaces are not specified explicitly. This is because the Modeler creates the undefined surfaces by bridging across the ends of the open profile. **Figure 98** shows the completed model.

Figure 98 The Completed Model



Open Centerlines

Figure 99 shows an example of an open centerline definition. Note that the profile is made small (relative to the span of the centerline) to prevent the model intersecting itself.

The link line picks up the hole profile using a cross point function (FUNV 11).

Figure 99 Open Centerline and Closed Profile

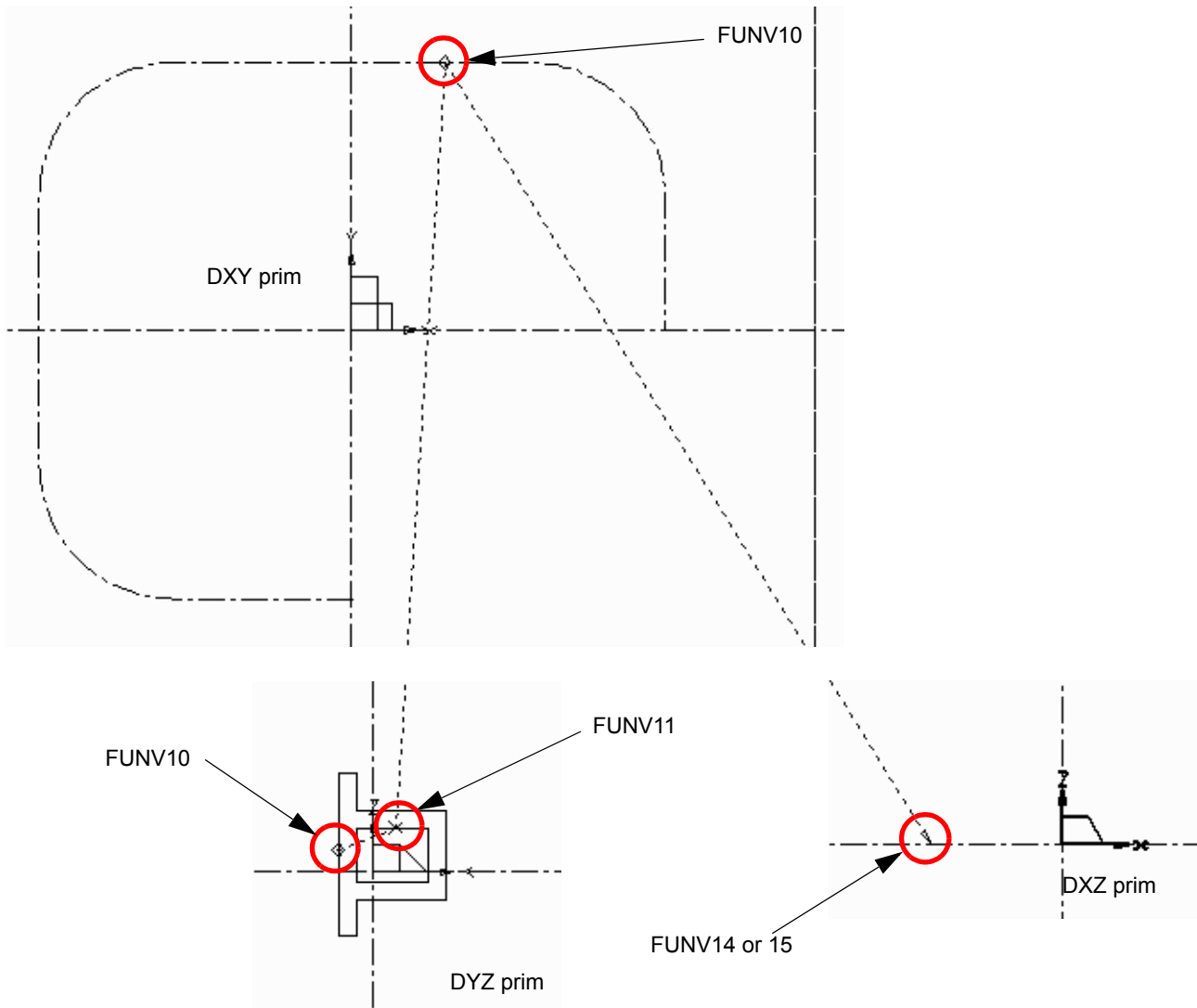
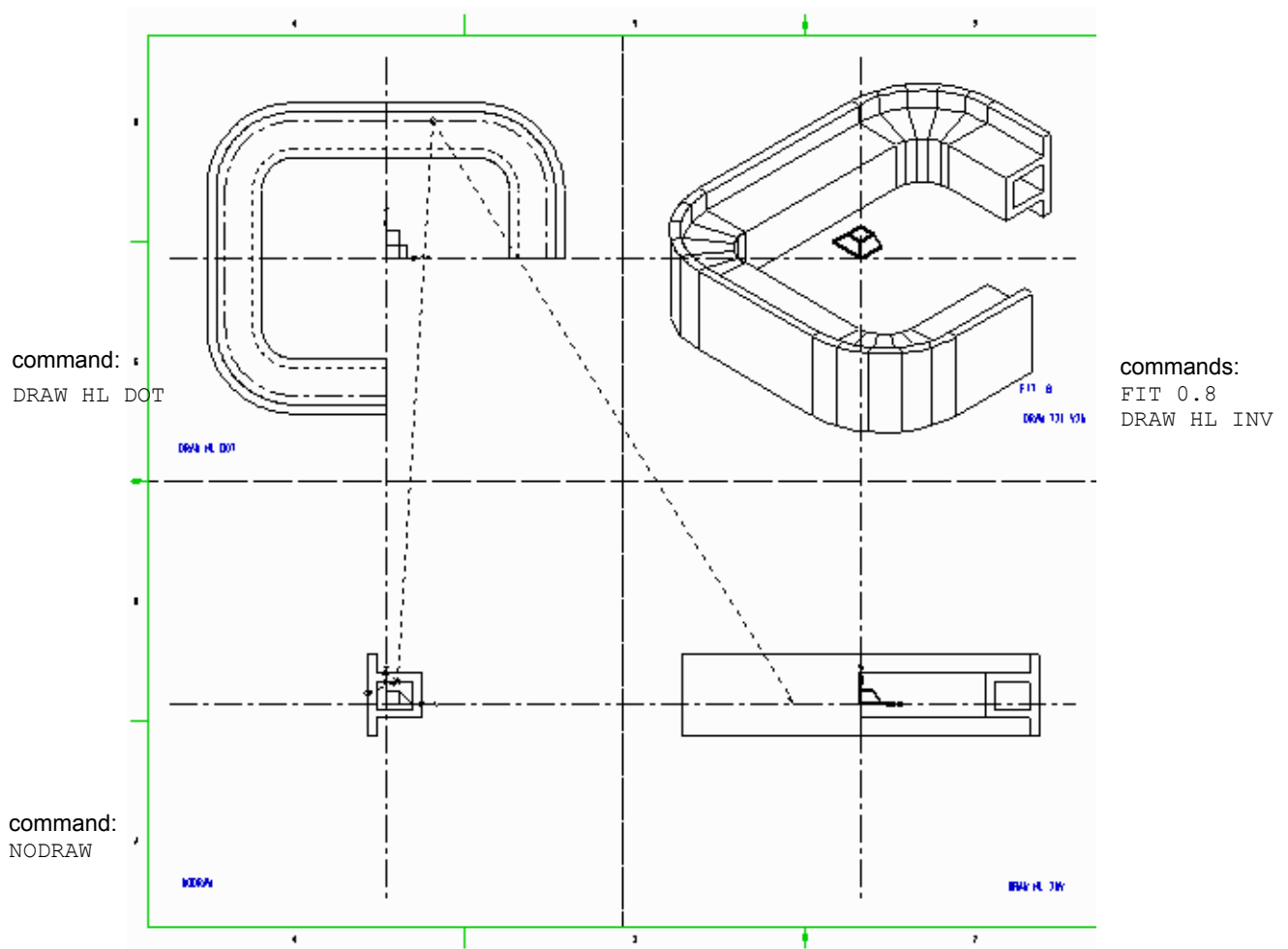


Figure 100 shows the completed model.

Figure 100 The Completed Model



Non-planar Paths

A model can be produced by sliding the profile around a non-planar (3D) path. In this case two centerlines are used; a primary and a secondary centerline, each drawn in a separate viewbox to define the complete slide path.

This technique requires that:

- Each centerline starts from the same corresponding point.
- Corresponding points in the centerlines match spatially and sequentially (point 2 to point 2, point 3 to point 3, and so on).
- There are the same number of points in each centerline.

- The link line be attached to corresponding points (or line segments) on both centerlines.
- The link line be attached to the secondary centerline by a plus point function (FUNV 12).

In the following example a solid model is created by the Slide generator using this technique. The figure below shows the two centerlines required and the order of points on these lines.

Please note: Curved centerlines must be constructed using **tangent-point arcs!**

Figure 101 Matching the Centerlines

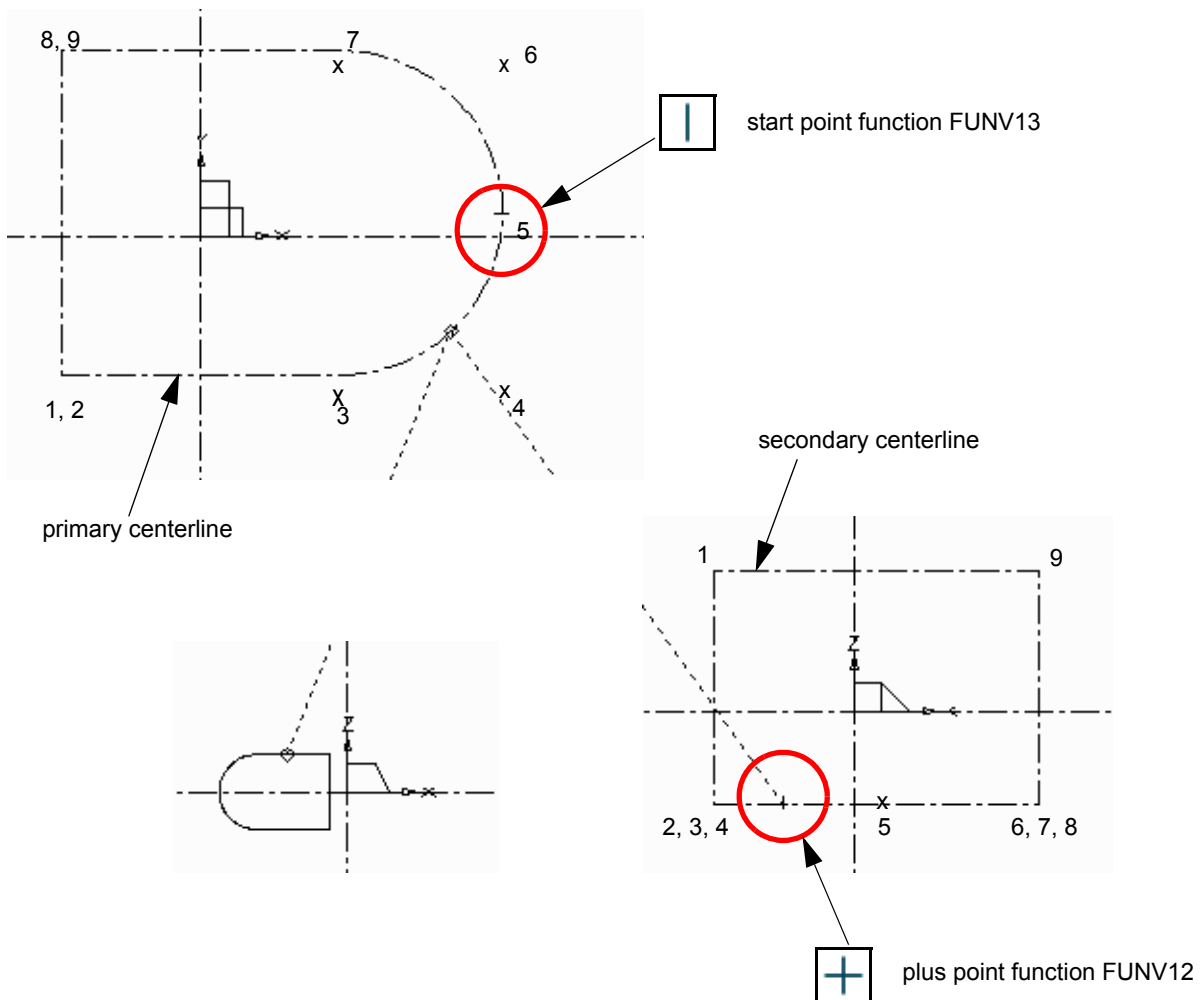
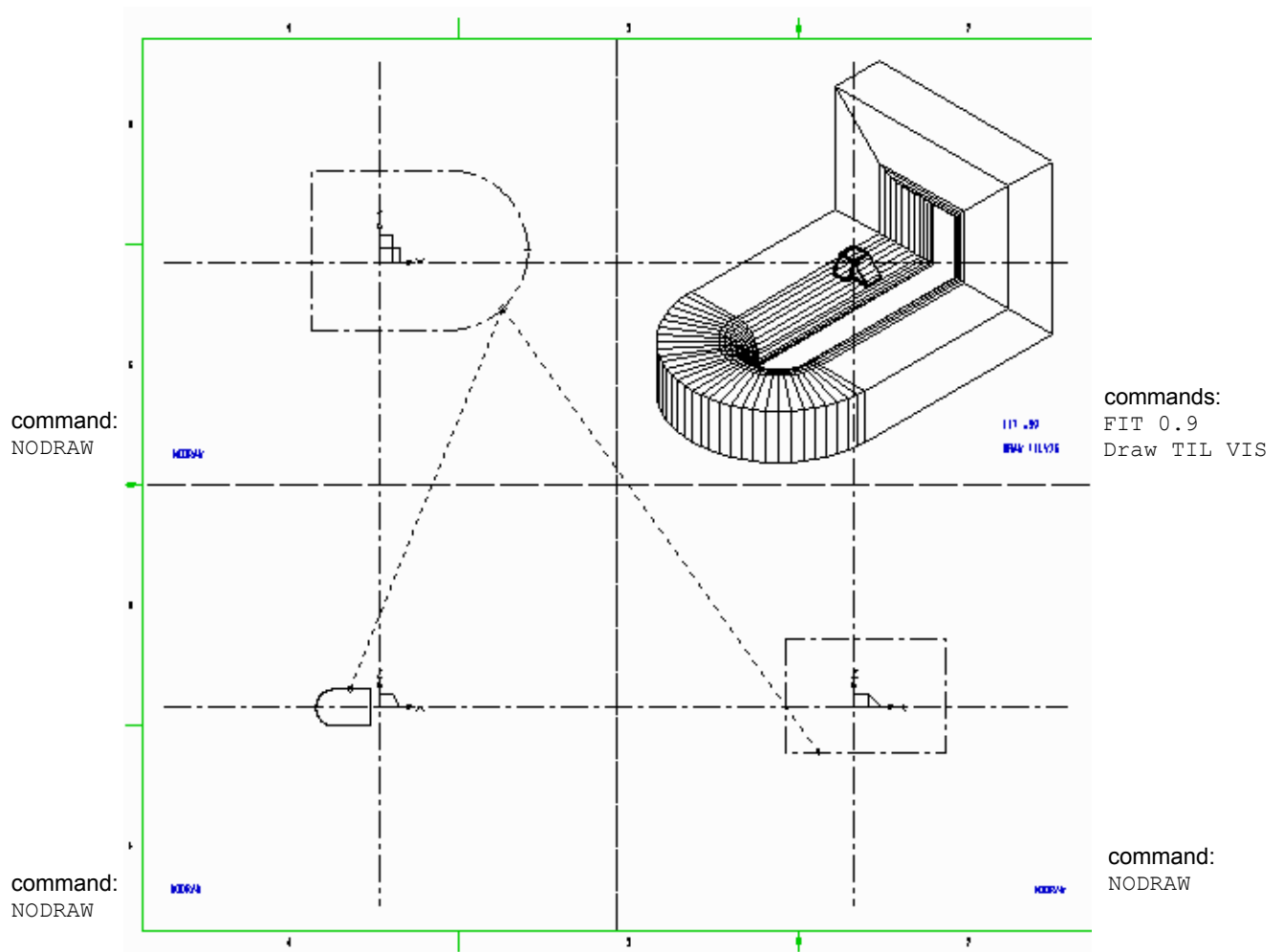


Figure 102 shows the completed model.

Figure 102 The Completed Model



Primary Centerlines and Projection Surfaces

A model can also be produced by sliding the profile around a non-planar (3D) path formed by the projection of a primary centerline onto an imaginary surface. This surface is defined by an infinite linear sweep of a secondary centerline.

With this technique the Modeler does not match the centerlines. Therefore the points in each centerline do not need to correspond in any way. The position of the link line on the projected centerline is also unimportant.

This technique requires that:

- The projection surface is **wide enough** for the primary centerline to be projected onto it.
- The link line be attached to the secondary centerline by a tilde point function (FUNV 21).

Figure 103 shows a model definition of this type.

Figure 103 A Projected Centerline Model Definition

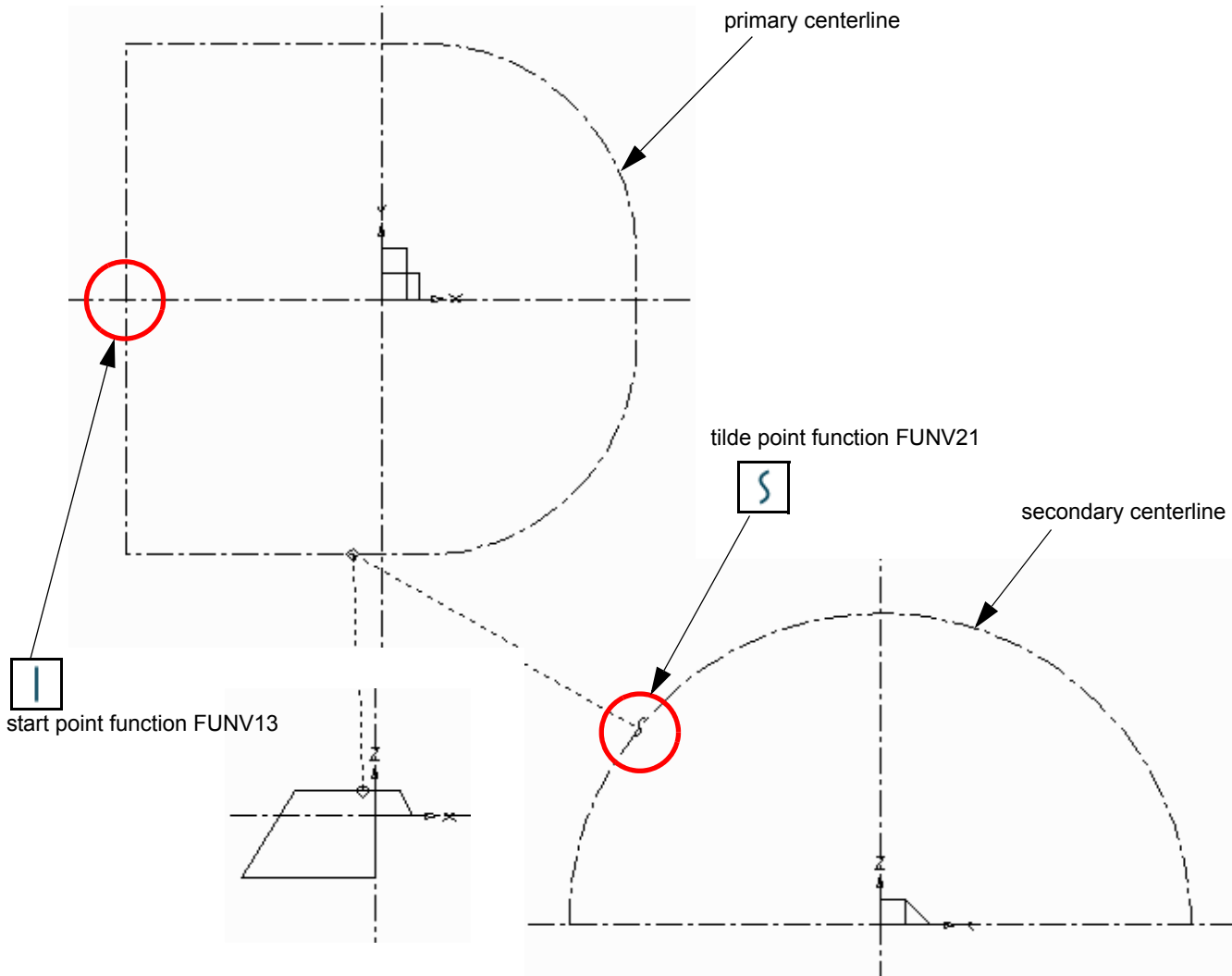
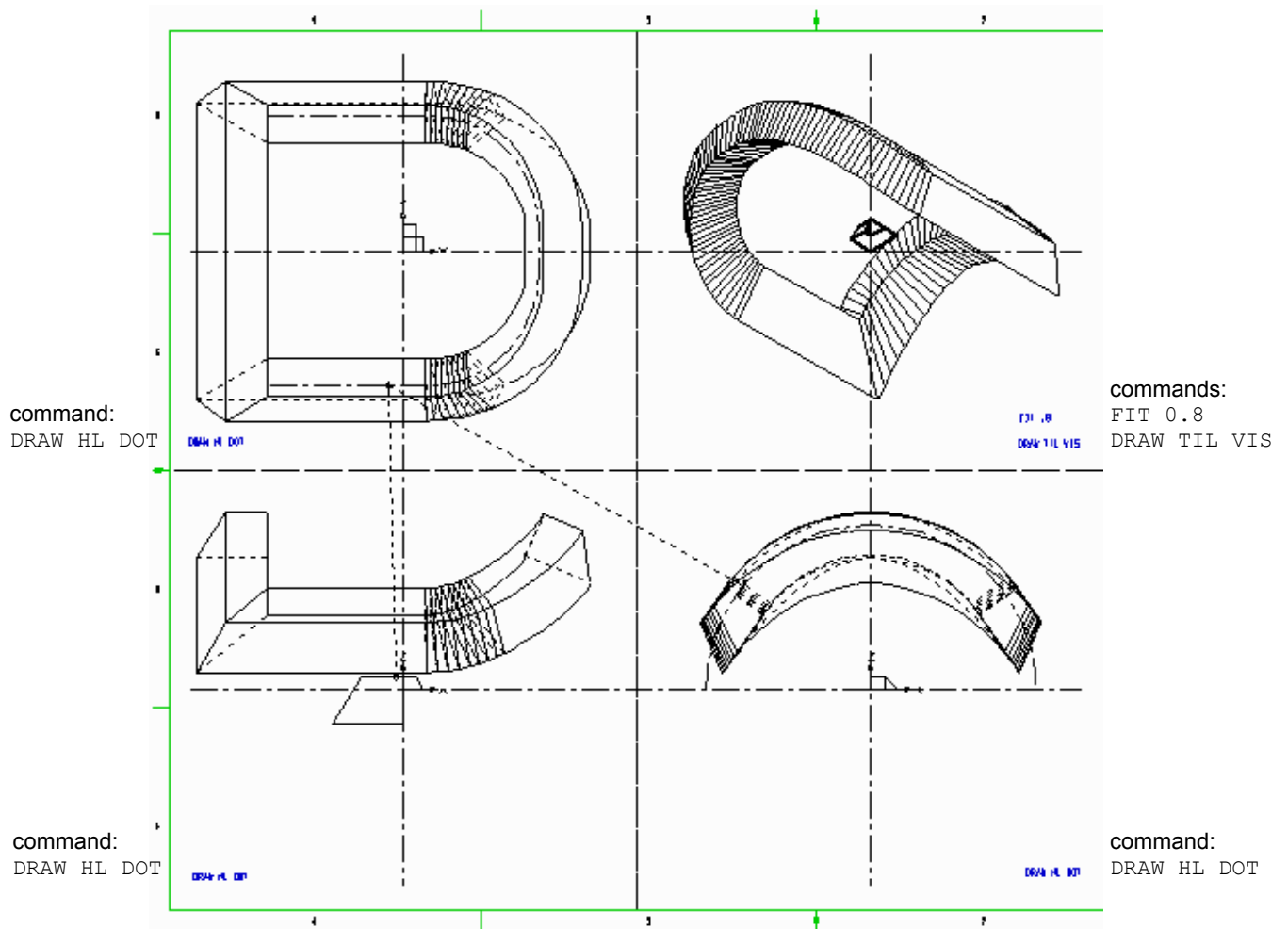


Figure 104 shows the completed model.

Figure 104 The Completed Model



Summary of Element Types

The following table is a summary of the line types and point functions used to produce models with the Slide generator.

Element Description	Element Style and Point Functions
Line Types	
Profile lines	Profile LP0 - LP9
Link line	Slide Generator
Centerline	center line
Point Functions	
Pick up profile line	FUNV 10
Pick up primary centerline	FUNV 10
Pick up hole profiles	FUNV 11
Fix position of model in the third dimension	FUNV 14 / FUNV 15
Override default start point	FUNV 13
Pick up secondary centerline	FUNV 12
Pick up projected centerline	FUNV 21
Text Types	
Depth text	Modeller Text ass. by Geometry (TMG)


PIPES

The Pipe generator creates models of pipes and pipe-like structures. The main use of this generator is for modeling small-scale pipework in assemblies. Large-scale pipework is handled more efficiently by the MEDUSA4 Plant Design System.

Multiple pipes are created using the techniques discussed in “[Sweeps](#)” on page 69. Non-planar (3D) pipes are created using techniques identical to those used to create non-planar wires (“[Wires](#)” on page 151), except that the link line is style Pipe Generator.

- [A Simple Definition](#)..... 144
- [Specifying the Pipe Bend Radius](#)..... 147
- [Summary of Element Types](#)..... 149

A Simple Definition

A pipe is drawn using a special profile line tool, Pipe profile . A width must be specified for this line type, which defines the diameter of the pipe. [Figure 105, “Definition of a Simple Pipe” on page 145](#) shows a simple pipe definition.

Follow the procedure described below to create a model of a pipe.




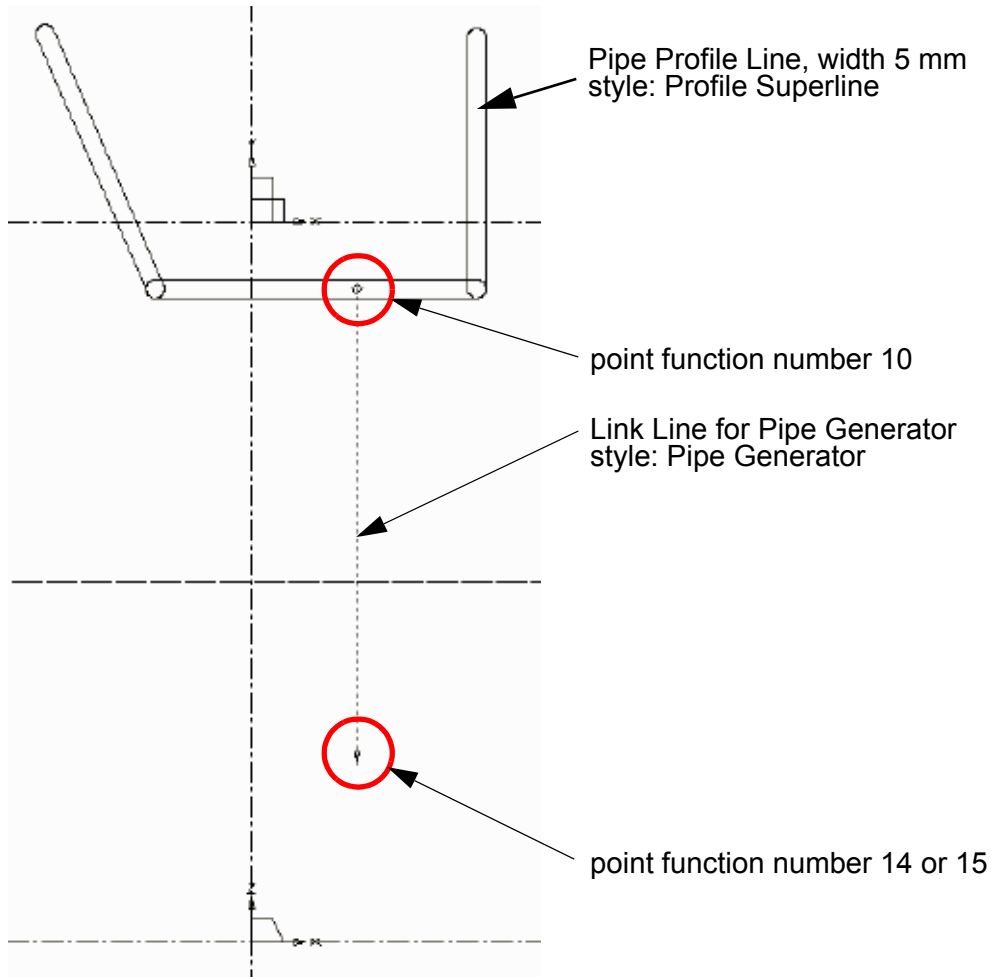
1. Select the Pipe profile tool  from the profile lines toolset.
In the dashboard you can specify the width of the line. The default setting is 5 mm.
Keep the default value.
2. Draw the pipe profile in one of the orthogonal viewboxes as shown in [Figure 105](#).
3. Choose the Pipe tool  from the link lines toolset.
4. Click the LMB on the pipe profile to place the starting point of the link line.
5. Extend the link line into another orthogonal viewbox and an click into the drawing area to place the second point of the link line.
Immediately the starting point of the line gets the diamond point function (FUNV 10).
The second point automatically gets the arrow point function (FUNV 14 or 15).
The point functions can be changed either with the Line Point Popup or the Line Point Properties menu (see [Figure 50, “Line Point Popup Menu” on page 79](#)). You also can use the Point Functions dialog ([Figure 52, “The Point Functions Dialog” on page 79](#)) which can be opened from the Dashboard.
6. Exit the Pipe tool .

Figure 105 Definition of a Simple Pipe




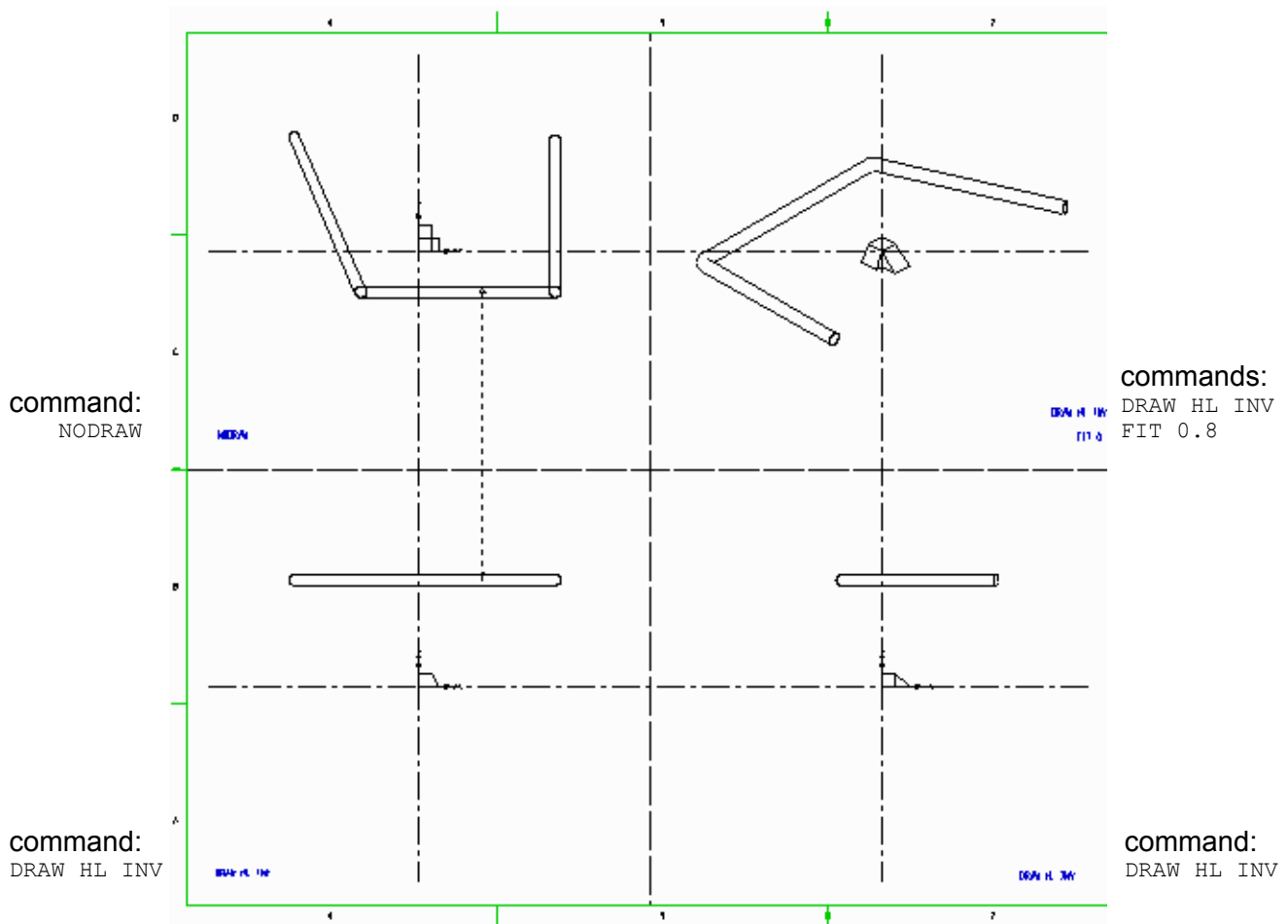
7. Select the Model + Reconstruct tool  to generate and view the completed model. [Figure 106](#) shows the completed model of a pipe generated from the definition shown in [Figure 105](#).

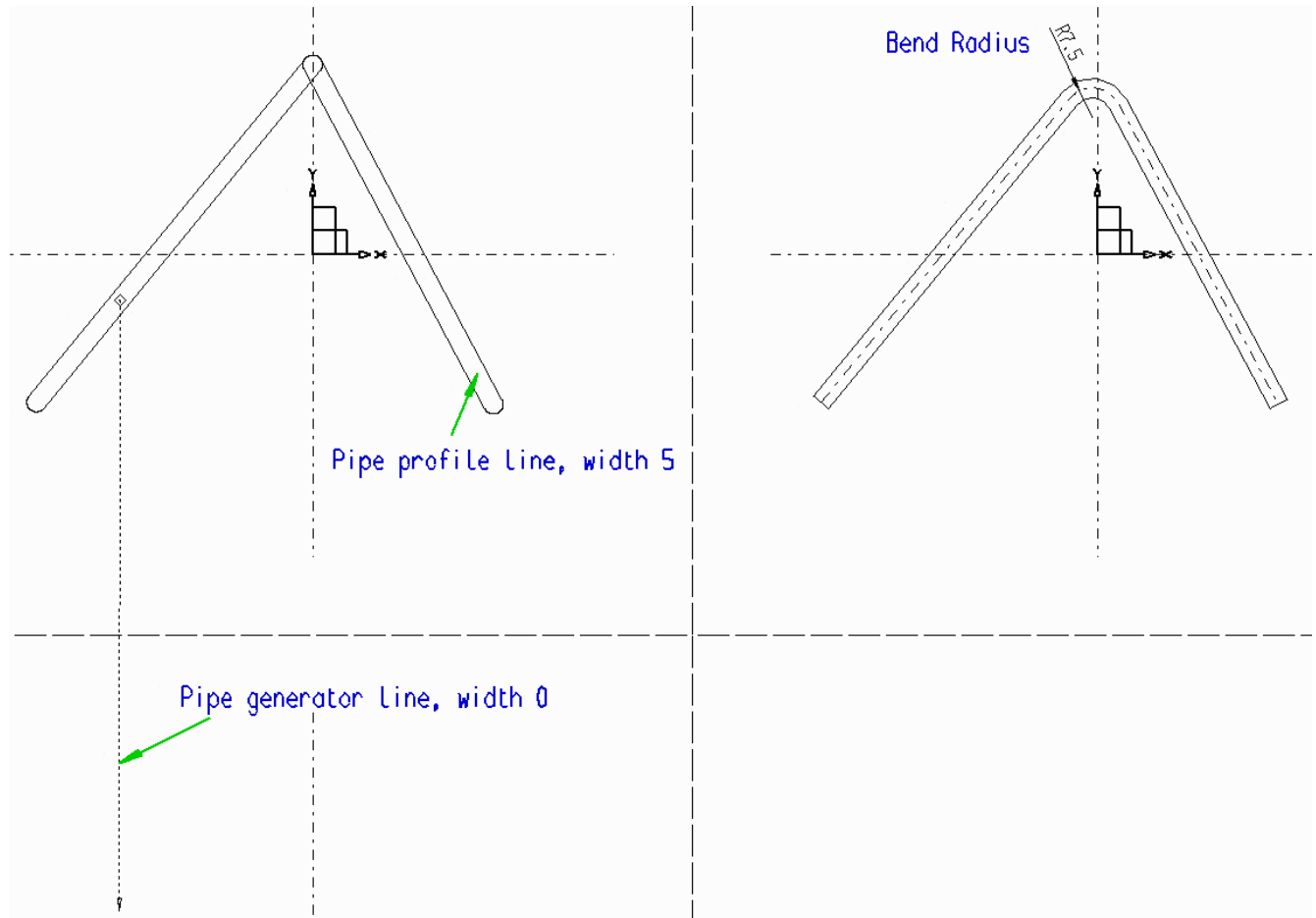
Figure 106 The Completed Model



Specifying the Pipe Bend Radius

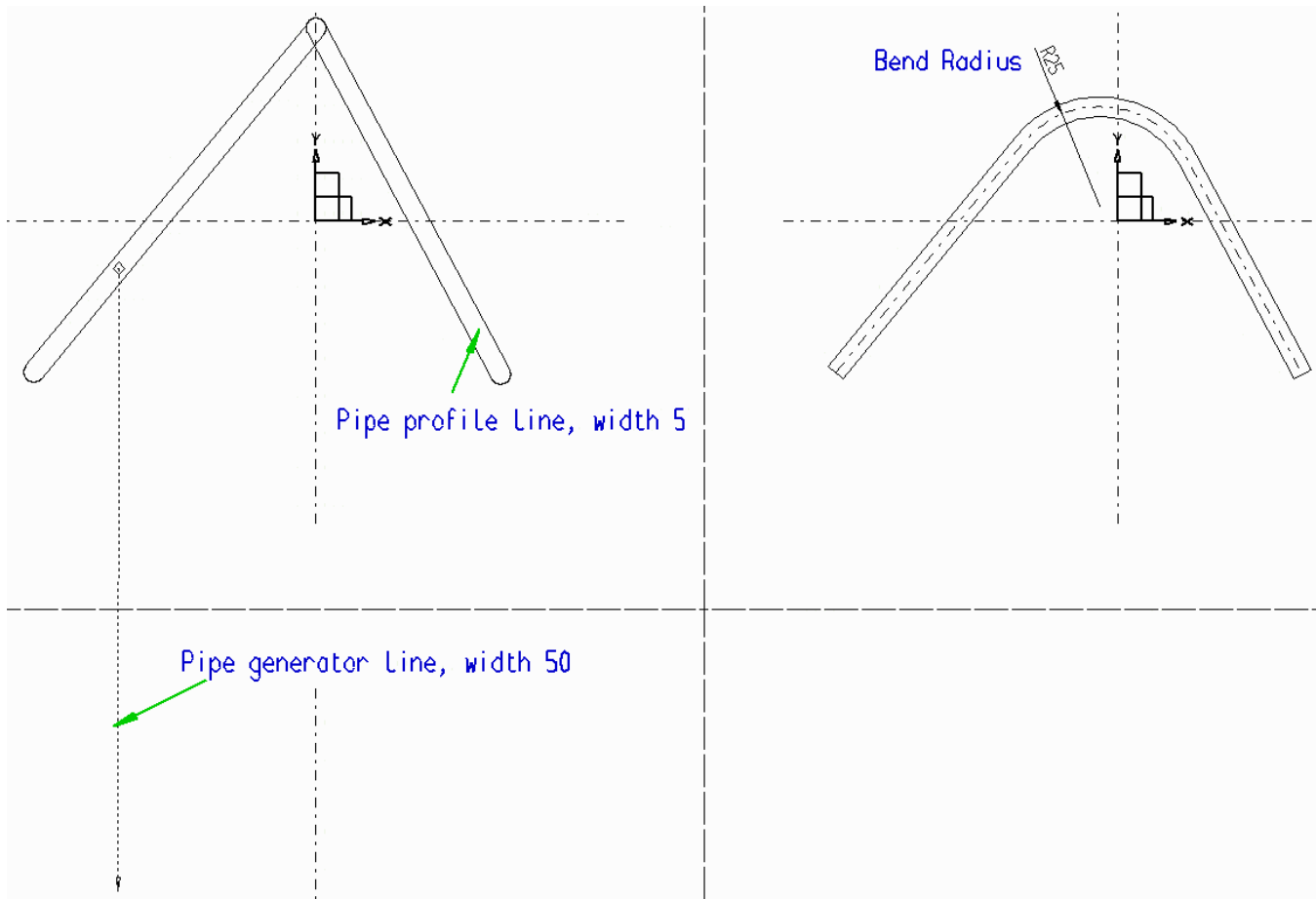
The Modeler blends each vertex in the pipe centerline with a bend radius. The bend radius value is the same for all vertices and is defined by the width of the link line. If you do not specify a width for the link line, the system will use a default bend radius based on the width of the profile line. The default bend radius is 1.5 times the width of the profile line. For example, a pipe drawn using a line width of 5 mm will have a bend radius of 7.5 mm, as shown in the following figure.

Figure 107 Definition and Model Using the Default Bend Radius



You can override the default bend radius by changing the width of the pipe generator line. To do this, select the pipe generator line and change the width in the properties dialog. The bend radius will be equal to half the value you specify for the width of the pipe generator line. For example, if you set the width to 50mm, the bend radius of the pipe centerline will be 25mm. [Figure 108](#) illustrates this principle. To reset to the default width, enter a pipe generator line width of zero.

Figure 108 Definition and Model Using a Specified Bend Radius



Please note: The physical appearance of the pipe generator line does not change when its width is changed. If you try to use a bend radius that cannot be fitted to the pipe you have drawn, the Modeler displays an error message (e.g. `WARNING - object intersects itself`) and an invalid model is generated.

Summary of Element Types

The following table gives a summary of the elements with its styles and point functions used to produce pipe models.

Element Description	Element Style and Point Functions
Line Types	
Profile lines	Profile Superline
Link line	Pipe Generator
Point Functions	
Pick up profile line	FUNV 10
Indicate third dimension on the link line	FUNV 14 / FUNV 15
Text Types	
Depth texts	Modeller Text ass. by Geometry (TMG)



WIRES

The Wire generator creates a wire model from one or two profile lines, and a link line of style `Line Generator`, type `LL`. A wire model looks like a line, has no thickness, and can be one, two, or three-dimensional. It has the property of length only.

An understanding of the techniques of wire model generation is an important part of creating models using the Meshed Surface generator, see [“Meshed Surfaces” on page 191](#).

Multiple wire models can be created using the techniques discussed in chapter [“Sweeps”](#), [“Basic Modeling Techniques” on page 78](#).

- [A Simple Definition..... 152](#)
- [Wire Patterns in Groups..... 154](#)
- [Non-planar Wires 156](#)
- [Wire and Shell Models From Standard Definitions 163](#)
- [Summary of Element Types 166](#)

A Simple Definition

Use the procedure described below to create a simple wire model. [Figure 109](#) shows an example definition. [Figure 110](#) shows the completed model.



1. Draw the profile of the wire object using a line of type LP0. [Figure 109](#) shows the shape of the profile line.
2. Select the Wire Lobster tool  from the link line toolset.
3. Click on the profile line.
4. Click into another orthogonal viewbox to define the position in the third dimension.
The link line is finished. A diamond point function (FUNV 10) is automatically added to its start point. The second point of the link line gets a single arrowhead point function (FUNV 14 or FUNV 15).
5. Select the Model + Reconstruct tool  to generate and view the finished model.

Figure 109 Definition of a Wire Model

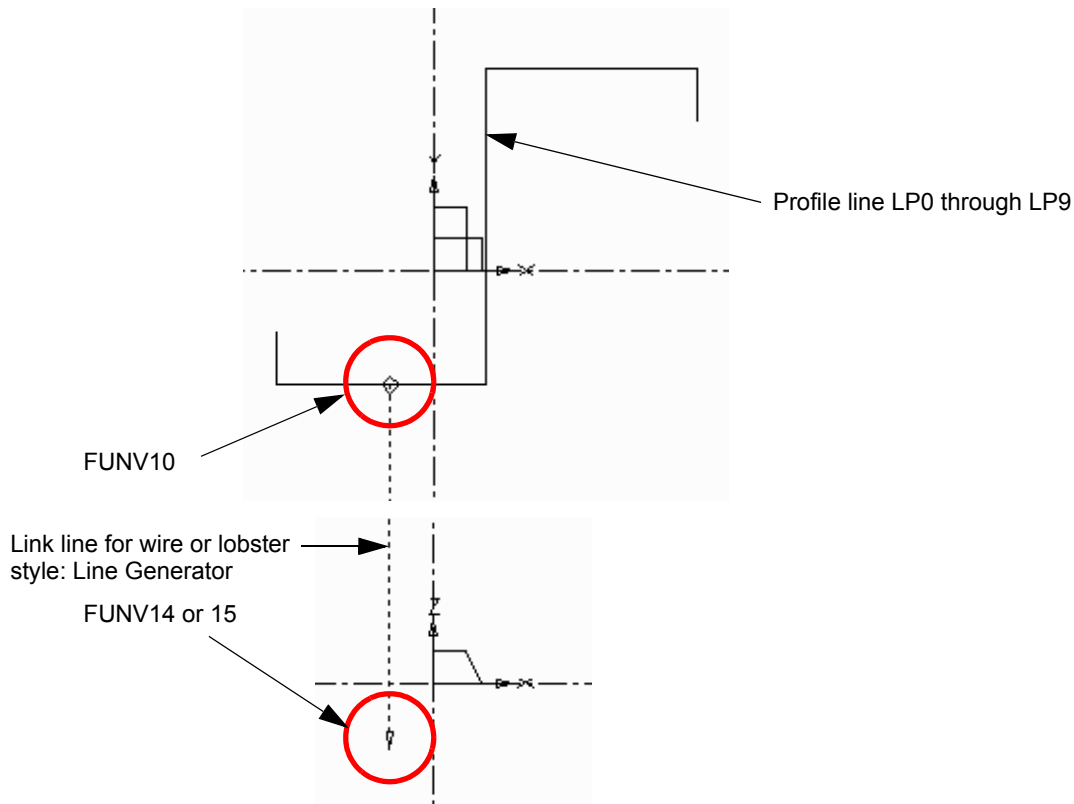
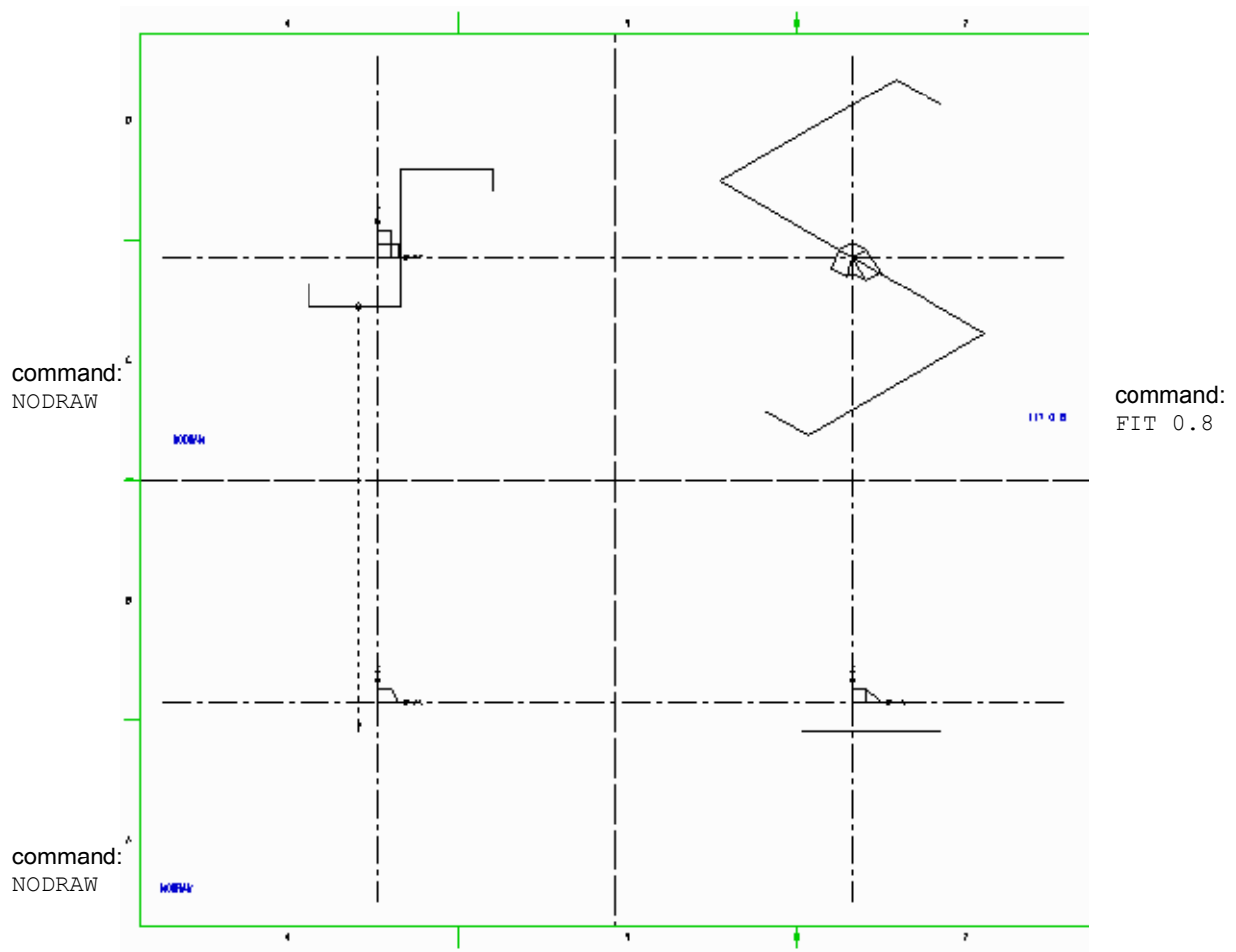


Figure 110 The Completed Model



Wire Patterns in Groups



Multiple wire patterns can be generated on one face of an object, using only one link line, if you create them as members of a 3D group. To create a new 3D group use the Empty Group tool . To create a 3D group containing the selected geometry use Group tool .


Figure 112 below shows star shaped elements created as members of a 3D group. The link line created with the Wire Lobster tool  is also included in this group. To generate a model from the wire patterns that are members of the group, you must attach the link line to one of the pattern profile lines using a circle point function (FUNV 26).

Figure 111 Definition of the Model

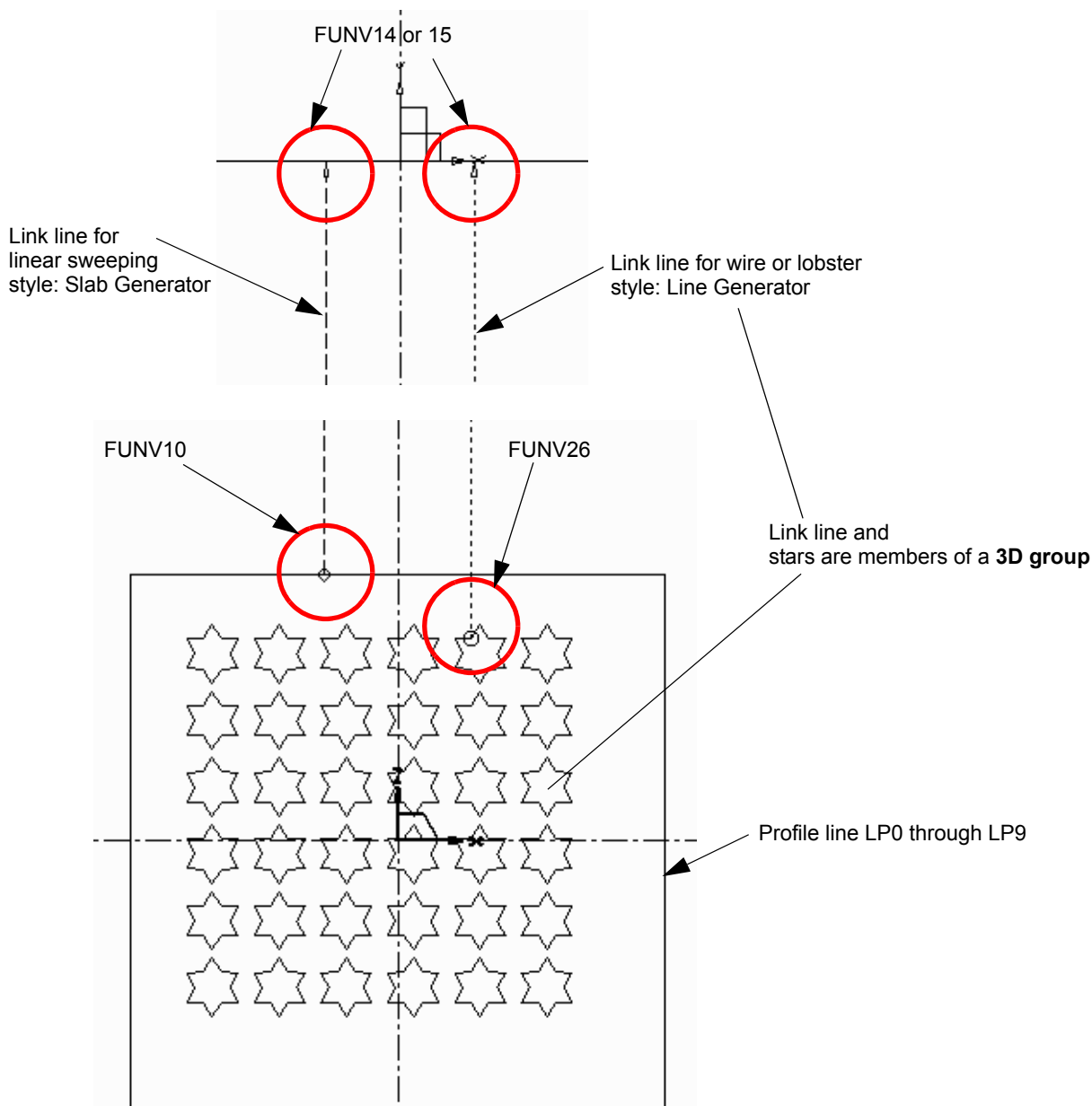
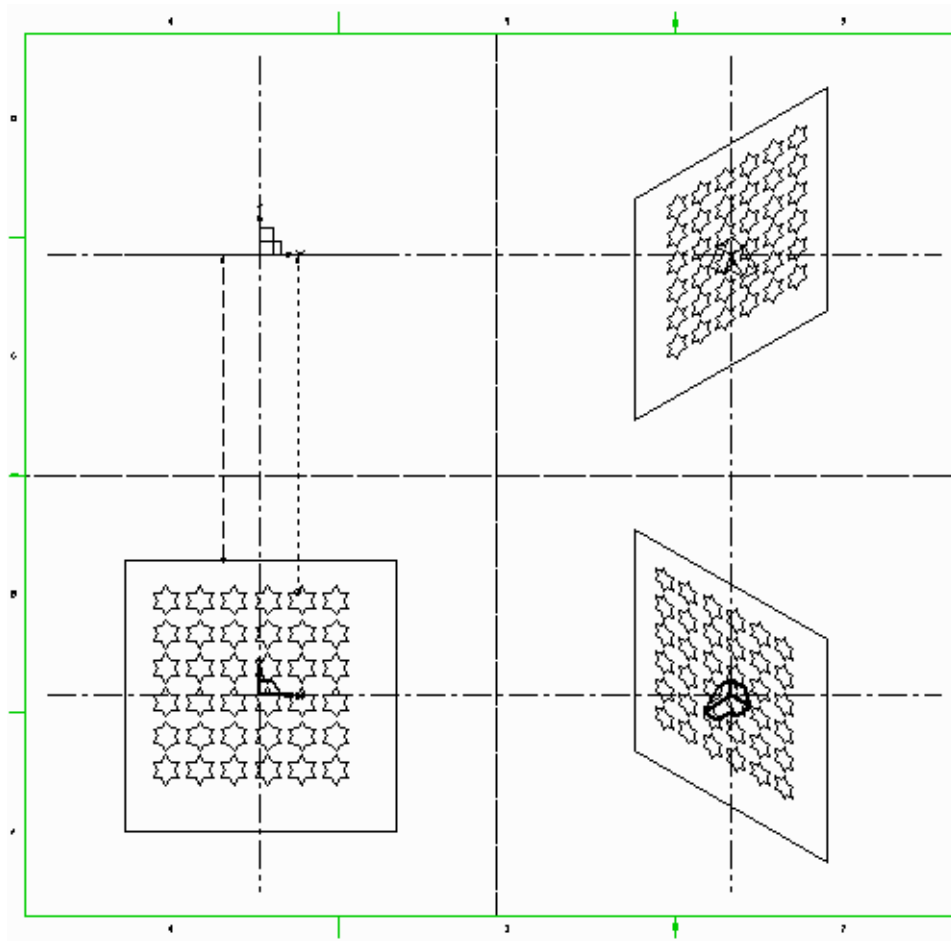


Figure 112 Wire Model Profile Lines in a 3D Group



Non-planar Wires

Non-planar (3D) wire models with complex geometry can be created. This section shows how to create:

- Wire models from primary and secondary profiles
- Wire models from projected profiles

Wires from Primary and Secondary Profiles

If a wire does not lie entirely in one plane, two profiles (a primary and a secondary) must be drawn in separate viewboxes to define the wire. The link line attaches to the primary profile with a diamond point function (FUNV 10) and to the secondary profile with a plus point function (FUNV 12).

When creating a non-planar wire, the Modeler matches a point on the primary profile to a corresponding point on the secondary profile. Therefore both profiles must contain the same number of points.

Note that when drawing complex wire profiles, it often happens that several points are required at the same location on a profile line to make the profiles match correctly. You can achieve this in one of two ways:

- By placing one point on the line and then giving a near probe for each additional point needed. This will superimpose points on the first and so create a multiple point. This way is used for creating the primary profile ([Figure 115](#)).
- By adding a diamond point function to a line point (FUNV 10). A diamond point function on a profile line is interpreted by the Modeling System as a multiple point. This way is used for creating the secondary profile ([Figure 115](#)).

Figure 115 shows the definition for a non-planar wire model.

Figure 113 Definition of Non-planar Wire

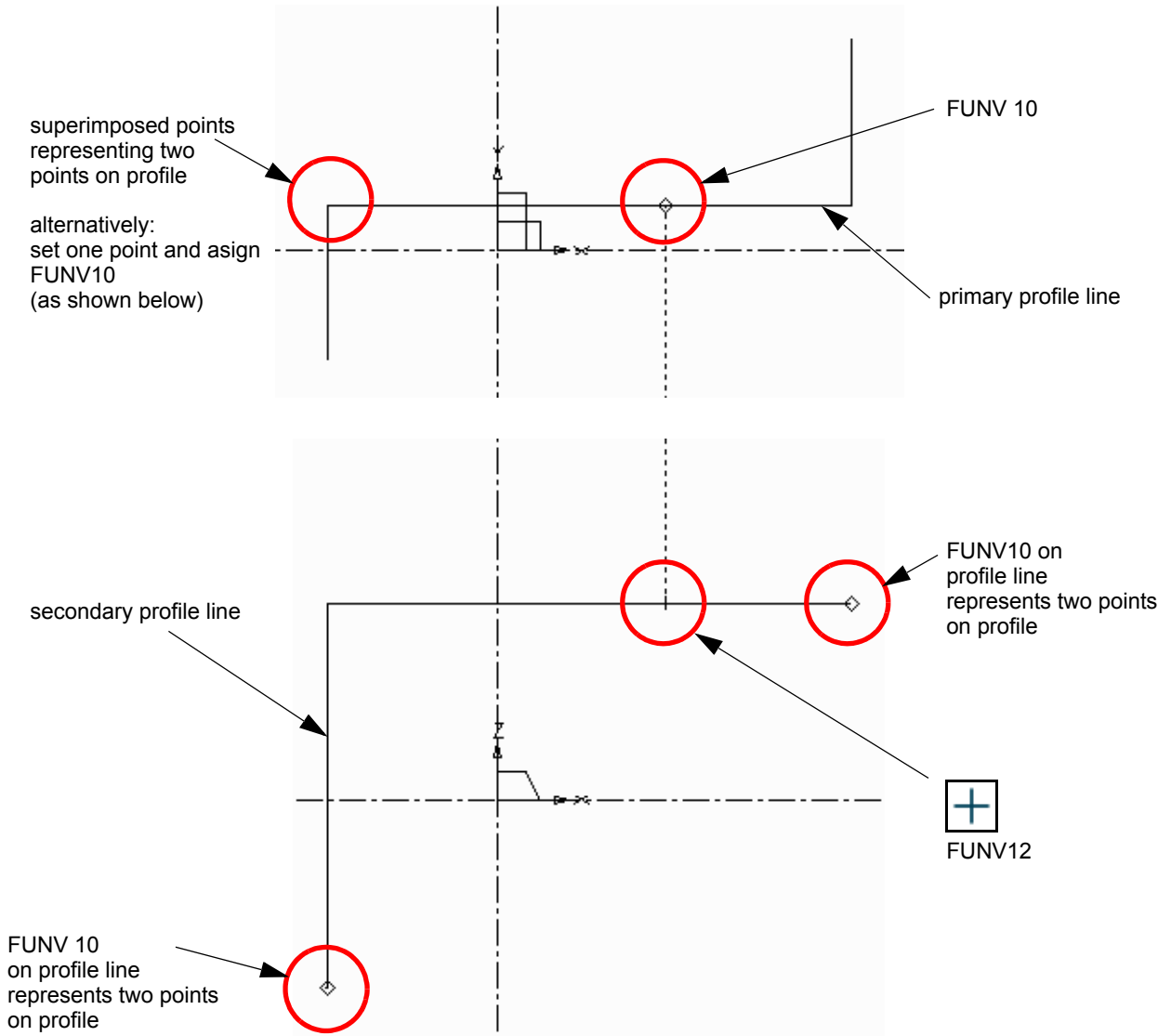
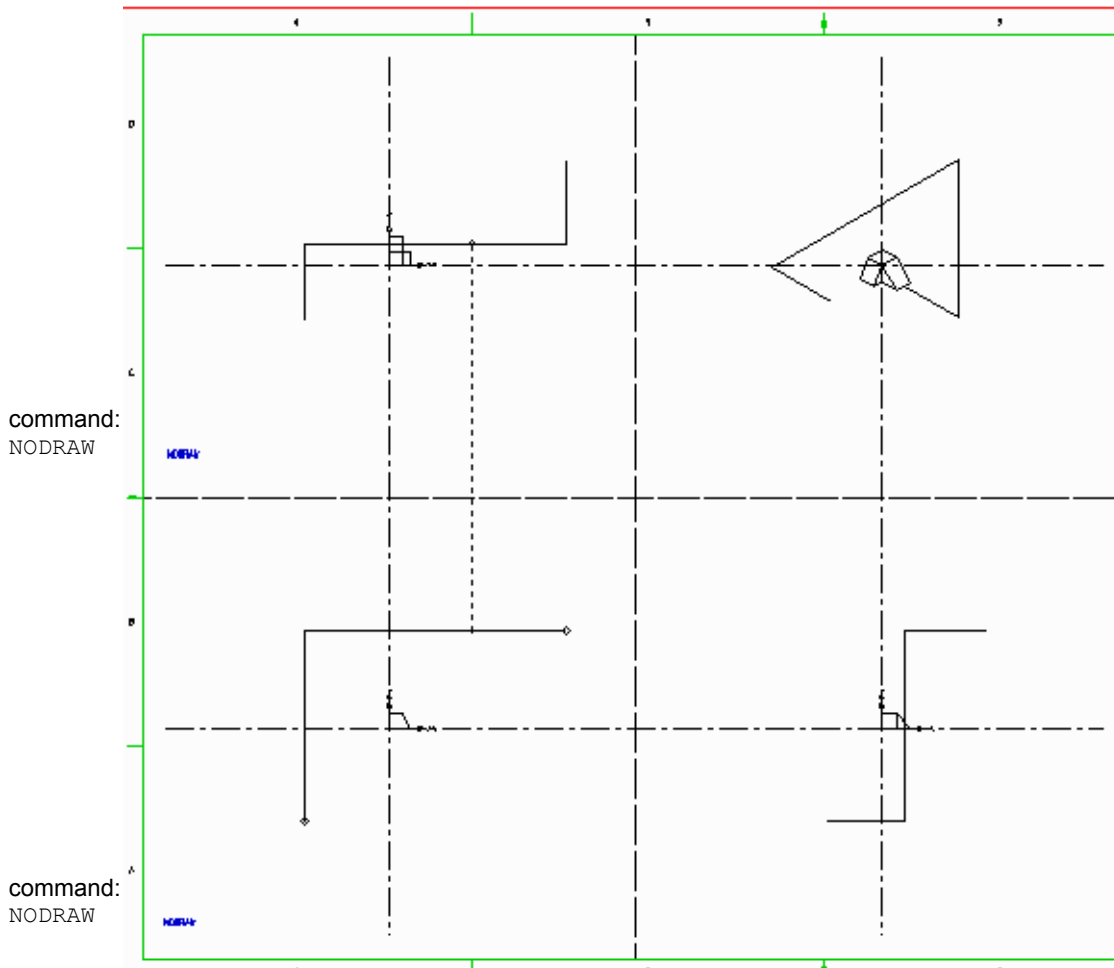


Figure 116 shows the completed model. In this example the positions of the points on the primary and secondary profiles must correspond exactly.

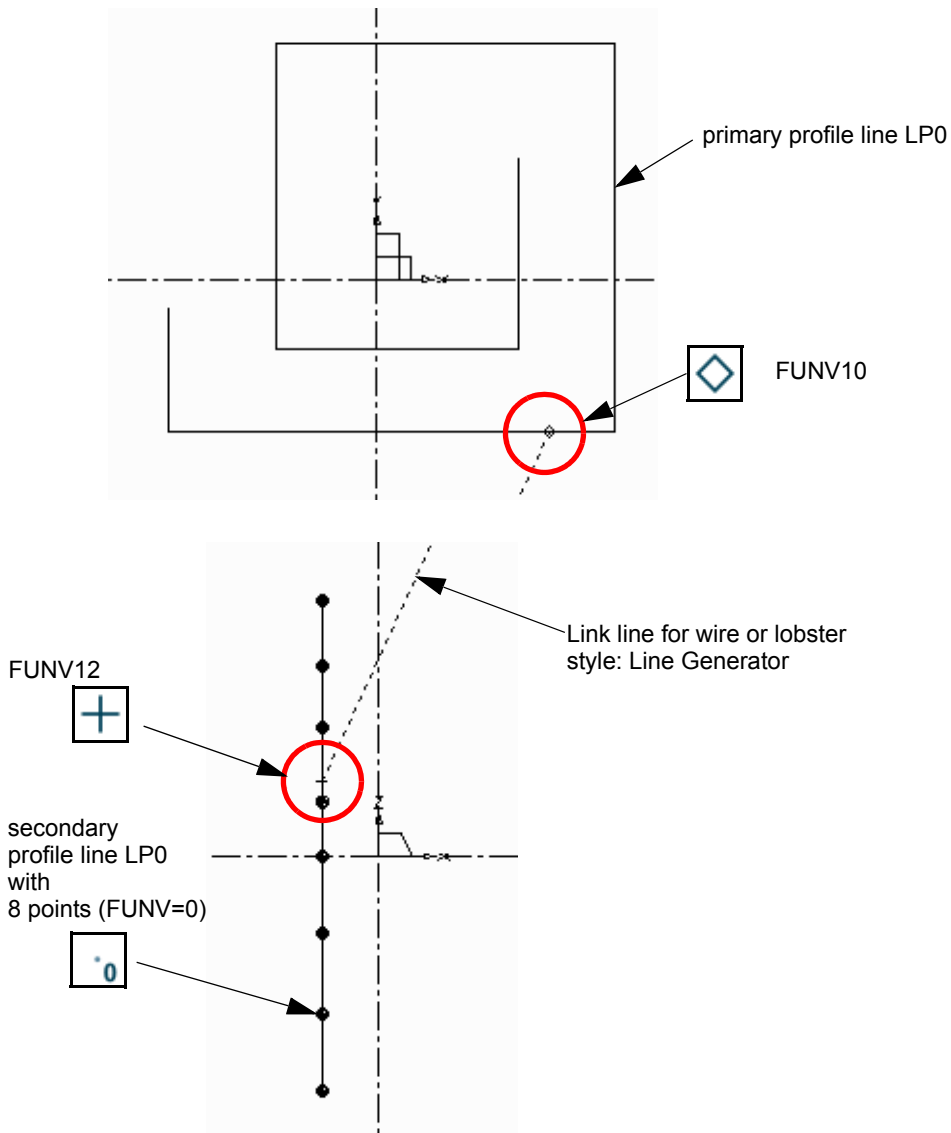
Figure 114 The Completed Model



Non-planar wires can also be modeled by constructing a simple secondary line in the definition. This line contains the same number of points as the profile line but is not a true view of the wire. The position of each secondary line point defines the position of the corresponding profile line point in the third dimension.

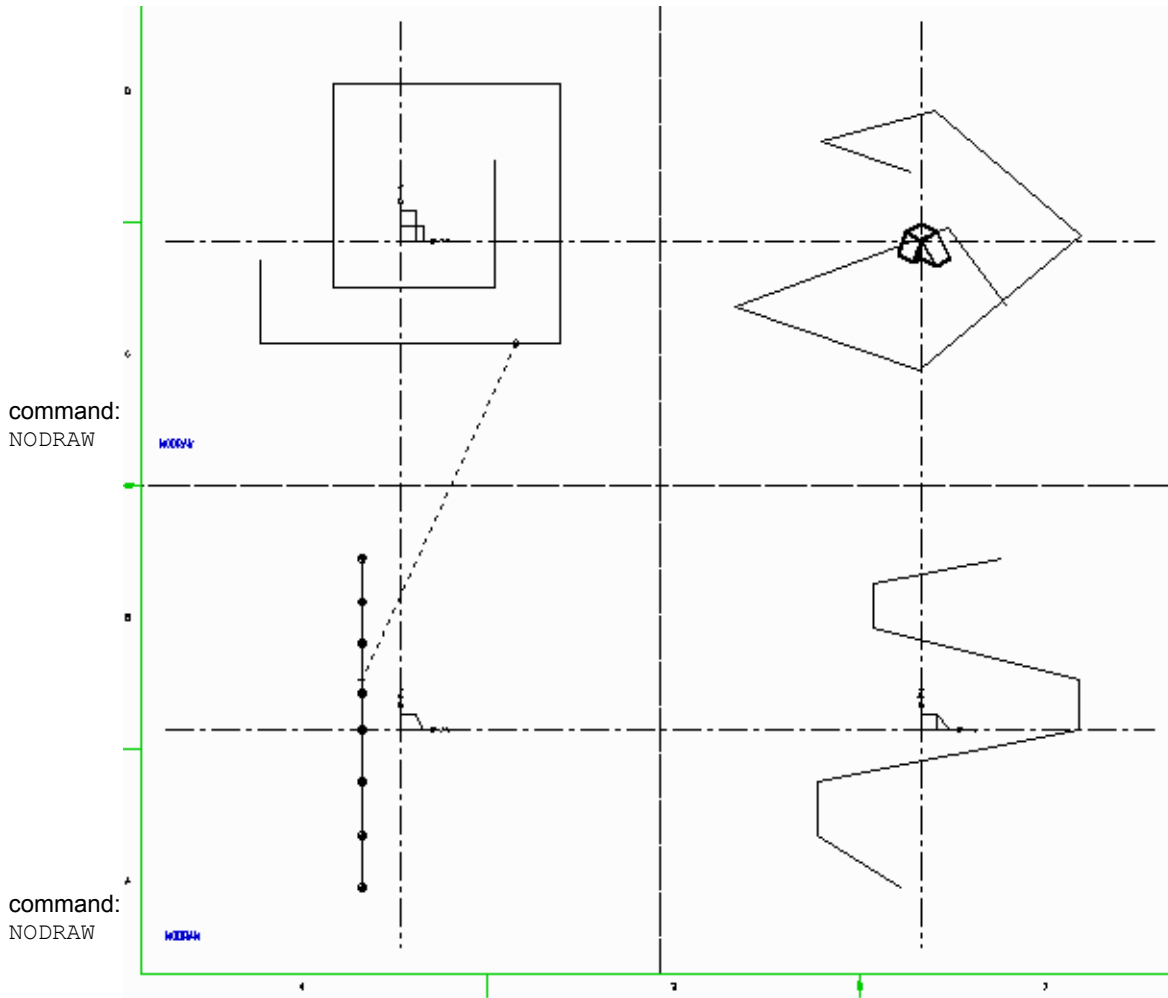
This technique is used in [Figure 115](#) which shows a non-planar wire definition.

Figure 115 A Non-planar Wire Definition Using a Secondary Line



[Figure 116](#) shows the completed model.

Figure 116 The Completed Model



Wires from Projected Profiles

A primary profile can be projected onto another profile to generate a non-planar wire model. The number of points in each profile is unimportant as no point matching is performed.

The link line is attached to the primary profile with a diamond point function (FUNV 10) and to the secondary profile with a tilde point function (FUNV 21).

Figure 117 shows the definition for a projected wire profile. Figure 118 shows the completed wire model.

Figure 117 A Projected Profile Wire Definition

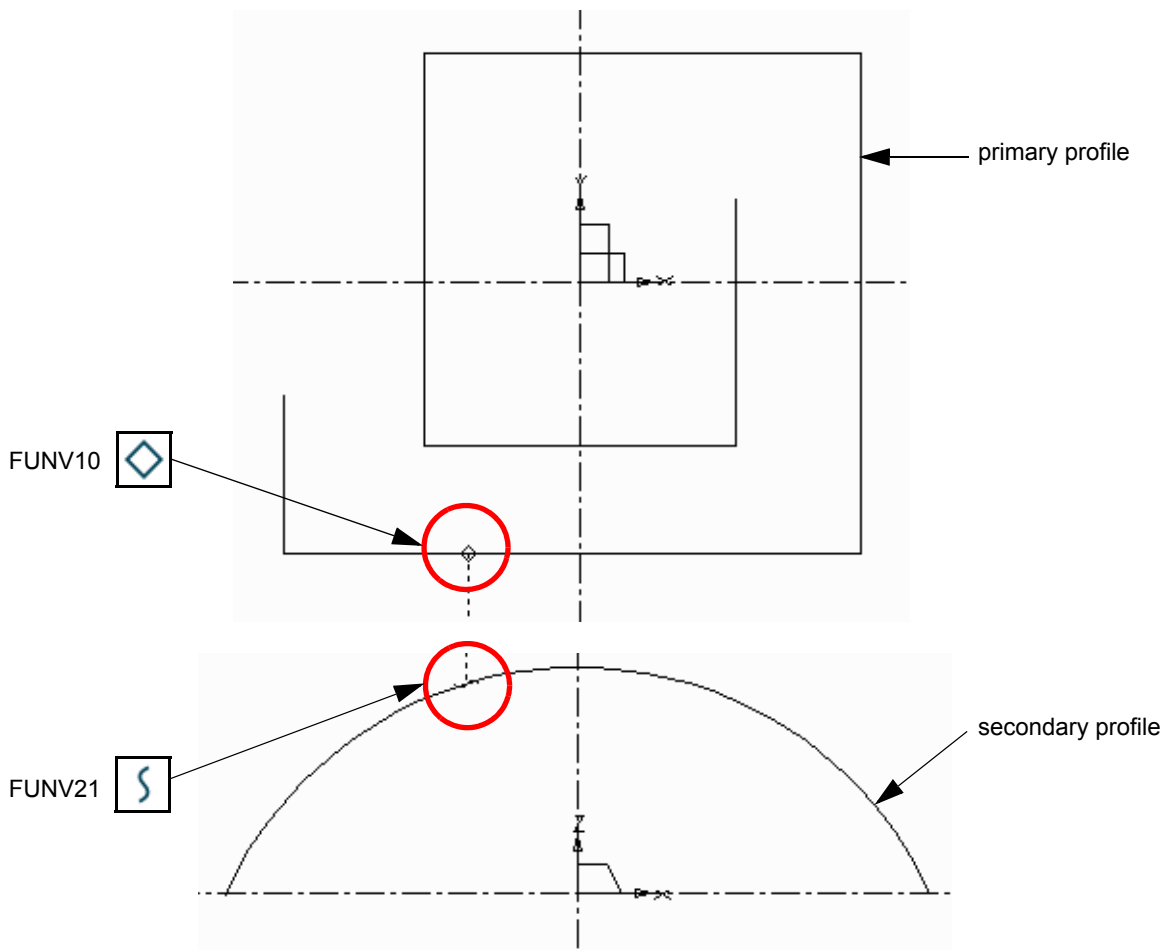
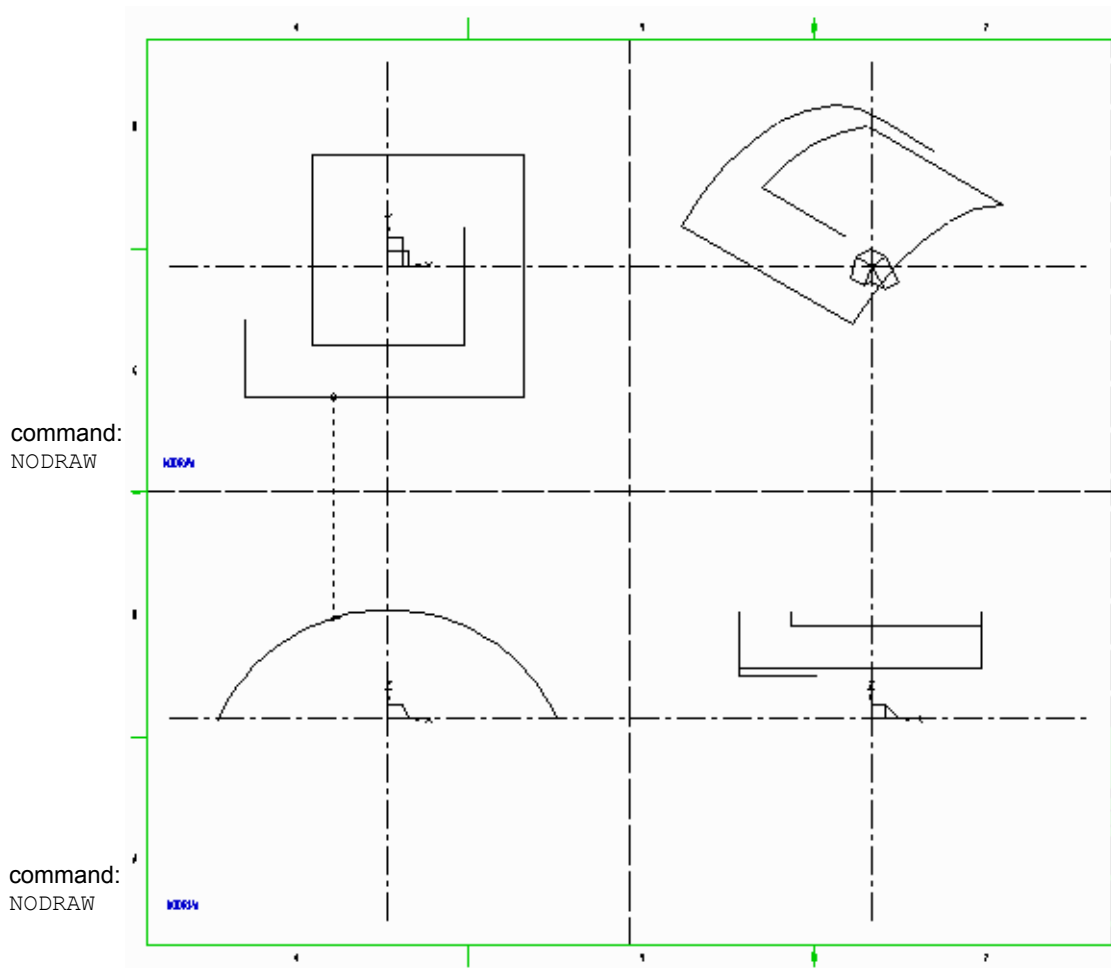


Figure 118 The Completed Model



Wire and Shell Models From Standard Definitions

You can also create wire models by attaching the TMG-text of style `Modeller Text ass. by Geometry:`

```
WIRE
```

to the link line of a standard model definition. This command causes a wire model to be formed from the patch boundaries of the model which would otherwise be created by the definition.

Shell models are created in a similar way by attaching the TMG-text of style `Modeller Text ass. by Geometry:`

```
SHELL
```


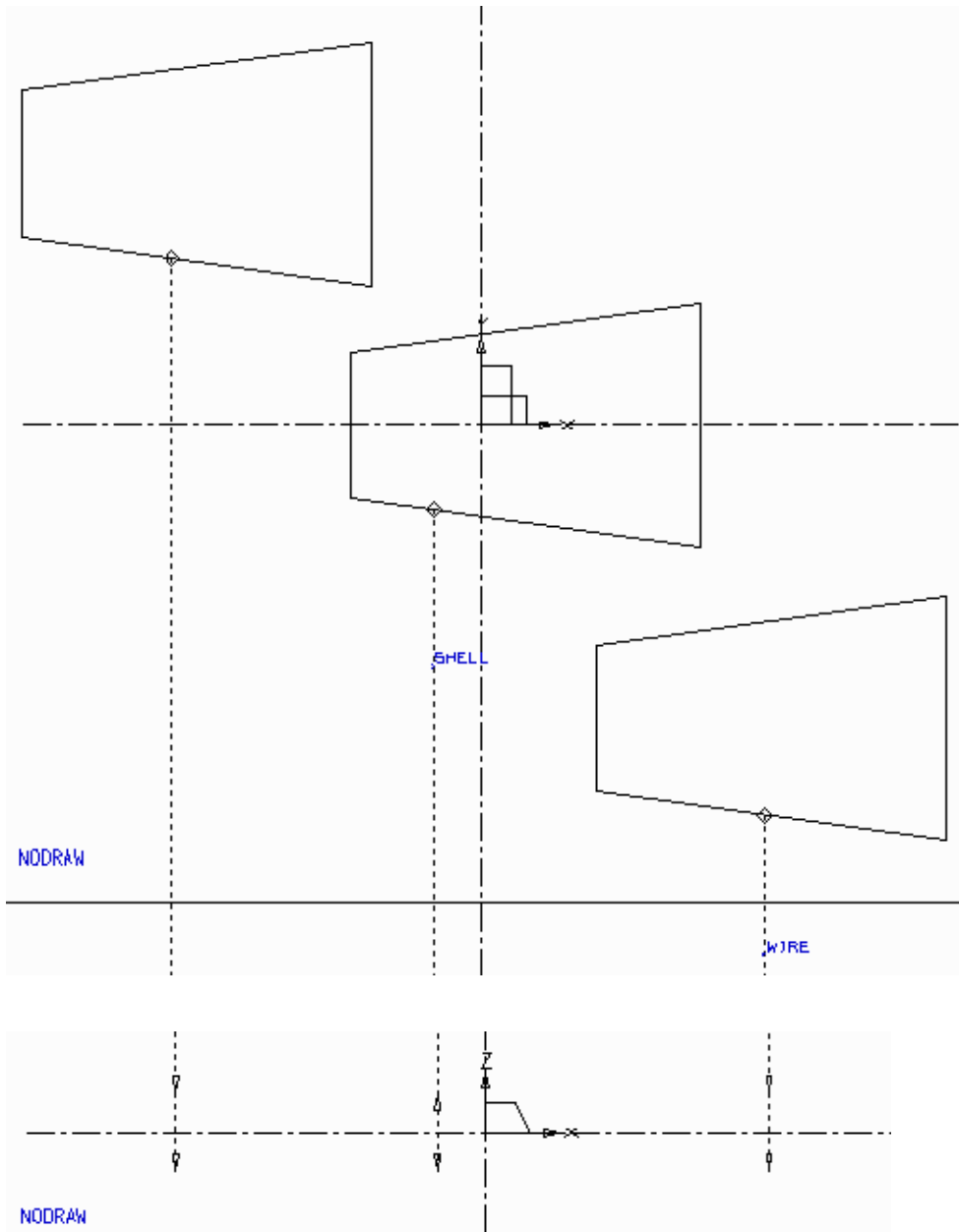
to the link line of a standard solid model definition created with the `Linear Sweep tool` . This command causes a model to be formed from the entire surface (or shell) of the solid model that would otherwise be created by the definition.

Figure 119 and shows how wire and shell objects are created by using the `WIRE` and `SHELL` commands in a solid sweep definition. The difference between the model types is revealed by using the `SECTION` command at the bottom right viewbox. This is a Viewer command which removes the front part of the model (sectioned in a plane perpendicular to the line of view) so that the internal structure of each object can be seen. In this example the model is sectioned at the origin.

Please note: There is **no separate link line** which defines a shell model.

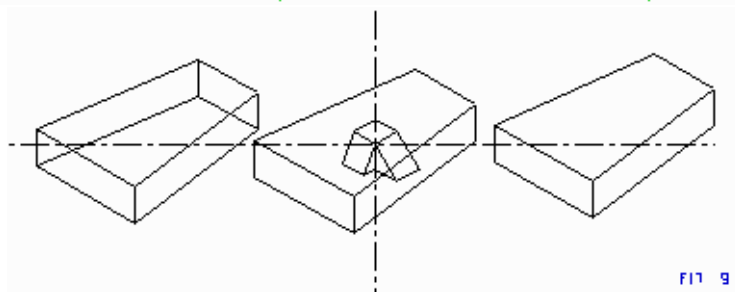
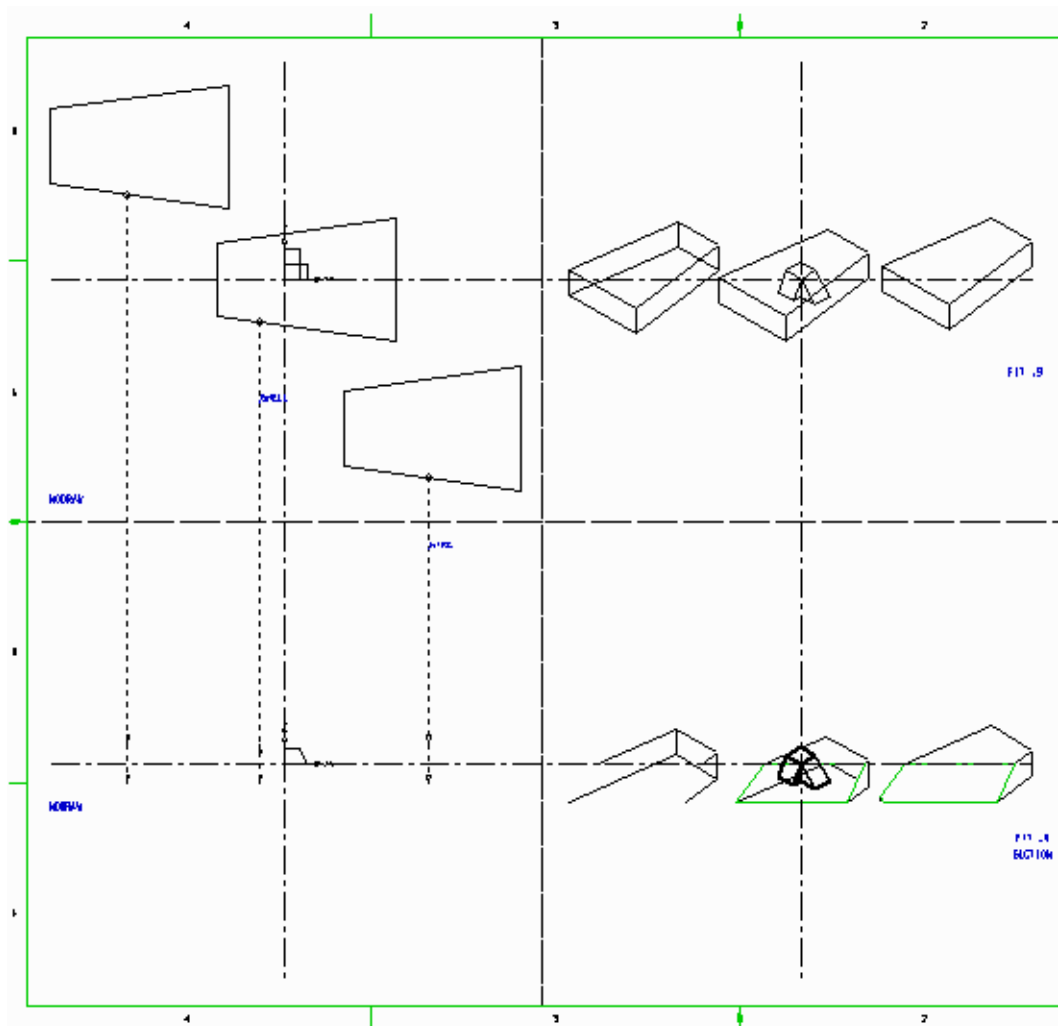
Figure 119 Definition of Solid, Wire, and Shell Models by using Text



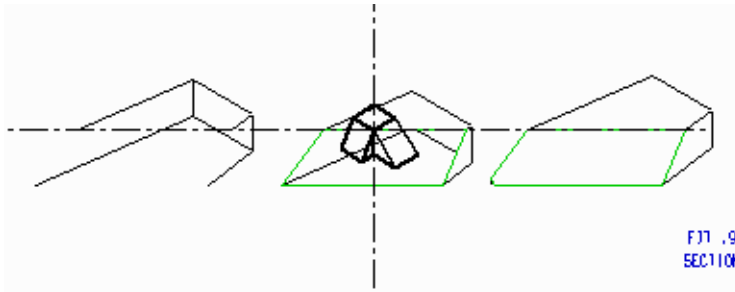
defines:	solid model	shell model	wire model
		command: SHELL	command: WIRE

all link lines
style: Slab Generator

Figure 120 Solid, Wire, and Shell Models



command:
FIT .9



command:
FIT .9
SECTION

Summary of Element Types

The following table gives a summary of the line types and point functions used to create Wire and Shell models.

Element Description		Element Style and Point Functions
Line Types		
	Profile lines	Profile LP0 - LP9
	Link lines	Line Generator
Point Functions		
	Pick up profile line	FUNV 10
	Indicate third dimension on the link line	FUNV 14 / FUNV 15
	Create a double point on profile line	FUNV 10
	Attach to secondary profile	FUNV 12
	Attach to projected profile	FUNV 21
	Attach to pattern profile	FUNV 26
Text Types		
	WIRE command	Modeller Text ass by Geometry (TMG)
	SHELL command	Modeller Text ass by Geometry (TMG)
	Depth texts	Modeller Text ass by Geometry (TMG)

DUCTS

The Duct generator is designed to create solid models of duct shaped objects. It can also be used to create irregularly shaped pipework type models which cannot be produced by the Pipe generator.

Duct models are generated by linking several profiles with a network of flat tiles so as to enclose a volume. The profiles define cross sections of the model at various positions along what is called the spine of the object. The spine is a 2D or 3D line which runs the length of the object, and is defined in either one or two orthogonal viewboxes.

The profiles are drawn in one viewbox connected together by a link line of style `Duct Generator` (type `LDT`). This line is then extended into another viewbox (orthogonal to the profile viewbox) to define the position of the model in the third dimension.

- [A Simple Model Definition..... 168](#)
- [Smoothing the Model..... 171](#)
- [Matching Points on Profiles 173](#)
- [Controlling Model Shape with Profile Orientation 175](#)
- [Creating a Model Using Non-Planar Centerlines 176](#)
- [Summary of Element Types..... 179](#)

A Simple Model Definition

Note that the following points apply to the construction of duct definitions:

- Each duct profile line must contain the same number of matching points. Plus point functions (FUNV 12) are required on the matching points to define where the spines of the duct pass through the profile.
- The centerline that defines the spine of the duct must contain one point for each profile. Plus point functions (FUNV 12) are required on these points to tell the Modeler where to attach the profiles to the spine.
- The order in which the link line picks up the profiles determines the order in which the Modeler attaches the profiles to the spine.

Use the procedure described below to create a simple model of a piece of rectangular ducting. [Figure 121](#) shows the model definition.




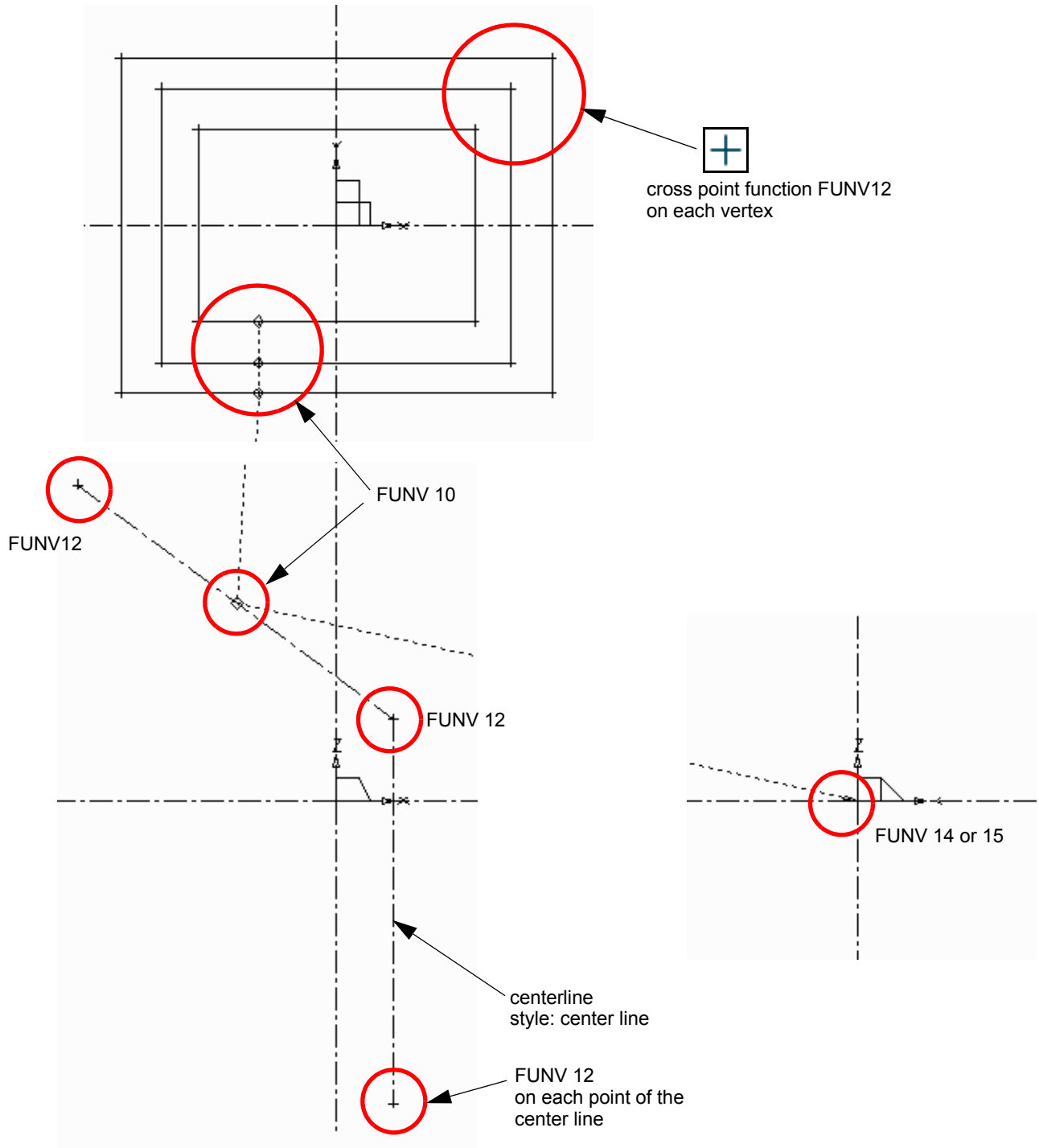
1. Select one of the profile line tools (e.g. Solid Thin , style LP0 through LP9).
2. Draw three rectangular profiles in an orthogonal viewbox as shown in [Figure 121](#).
3. Place plus point functions (FUNV 12) on each corner point. Edit a rectangle and choose Line Point Popup > Funv12 to all points from the popup menu.
4. Choose the Create Center Lines tool  and draw the spine of the duct in another orthogonal viewbox. Draw the spine using three points and place a plus point function (FUNV 12) on each point.
5. Select the Duct Generator tool  from the link line toolset.

Figure 121 Definition of a Simple Duct



6. Click the LMB on each rectangular profile to draw the link line. Each point gets a diamond point function (FUNV 10).
7. Click the LMB on the center line (spine) to attach the link line. Also this point gets a diamond point function (FUNV 10).


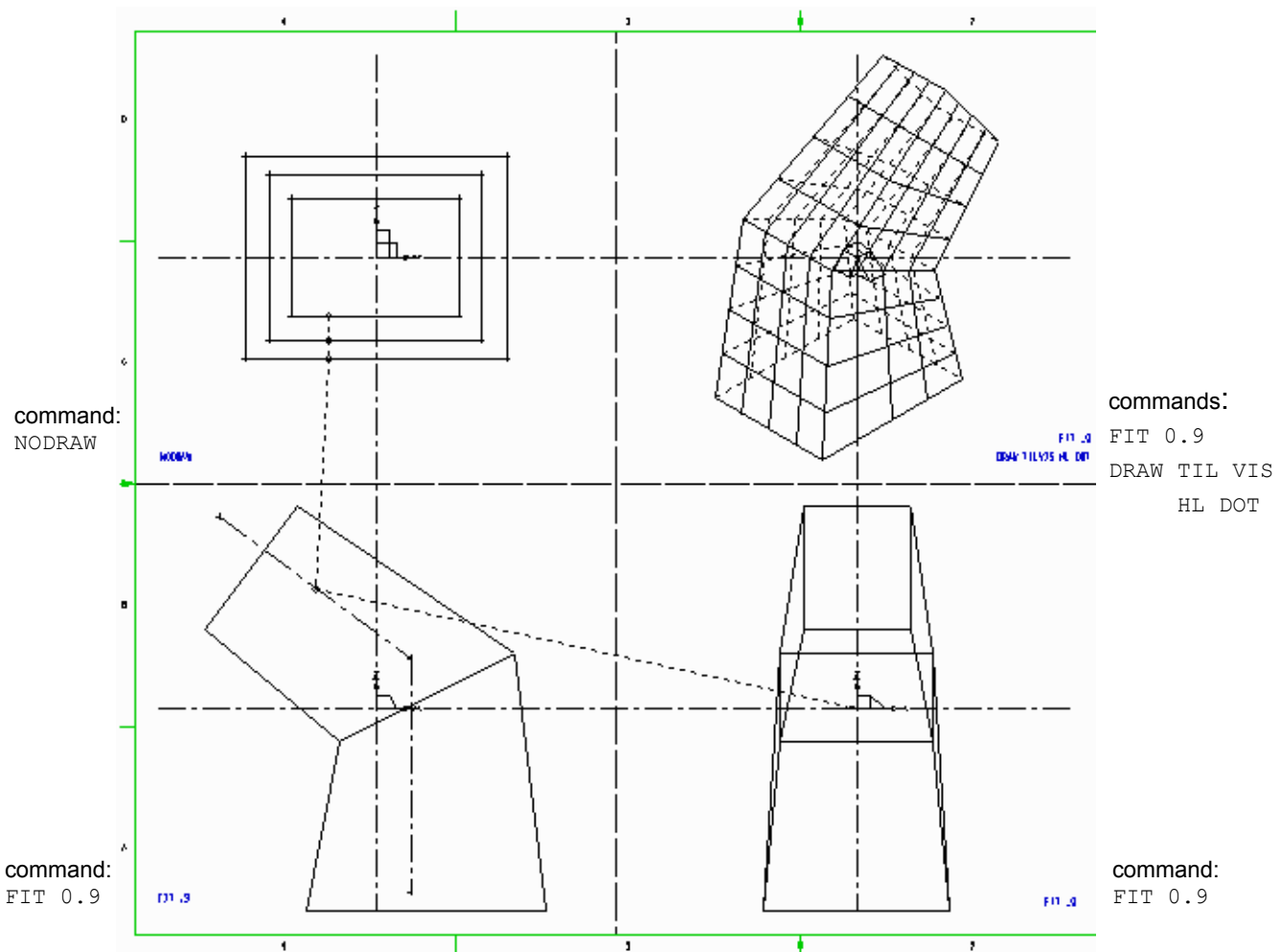
8. Extend the link line into another orthogonal viewbox to define the position of the ducting in the third dimension.
The last point of the link line is completed with a single arrowhead point function (FUNV 14 or FUNV 15).
9. Select the Model + Reconstruct tool  to generate and view the model.
Figure 122 shows the generated model.

Figure 122 The Completed Model



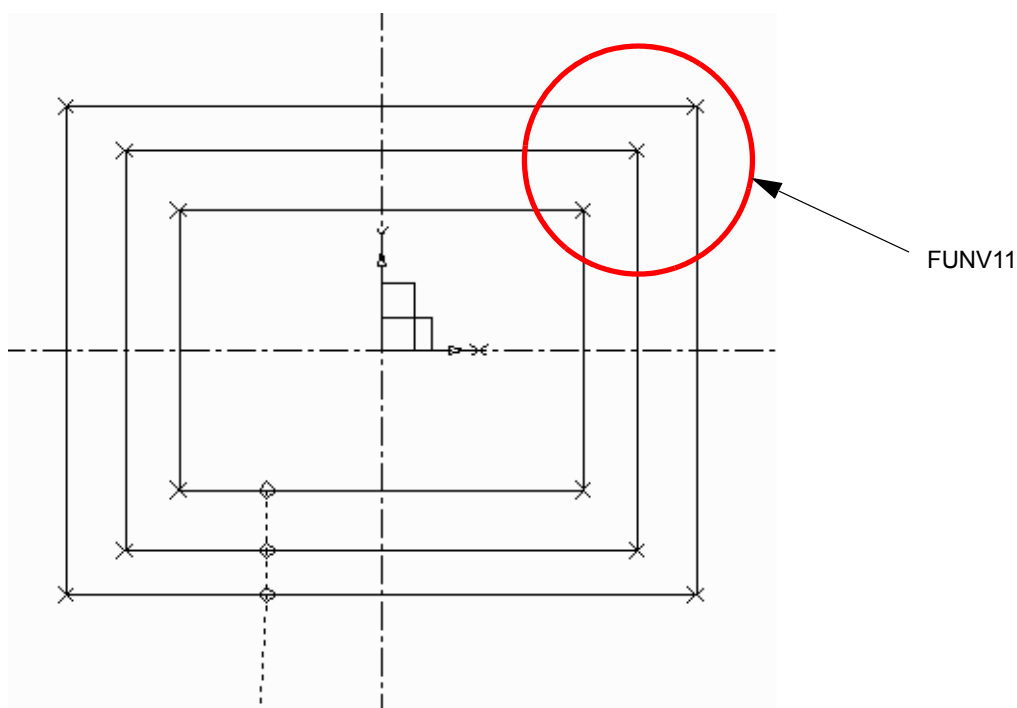
MEDUSA4 provides different default settings regarding the point functions of Ducting Profile lines. See the Set Point Function to Ducting Profile option in the 3D tab of the Default Settings dialog which is called up via File > Default Settings > 3D Products (see “The 3D Default Settings” on page 15).

Smoothing the Model

The type of point function used to match the points on the duct profiles determines the smoothness of the model form. In the simple definition given earlier in this chapter, plus point functions (FUNV 12) are used to match the profiles. This produces a straight-line (unsmoothed) model.

However, if cross point functions (FUNV 11) are used to match the points on the profiles, the Duct generator produces a smoothed model. This is shown in [Figure 123](#). The only difference between the definition of the model shown in [Figure 122](#) and [Figure 124](#) is the point function of the profile line corners.

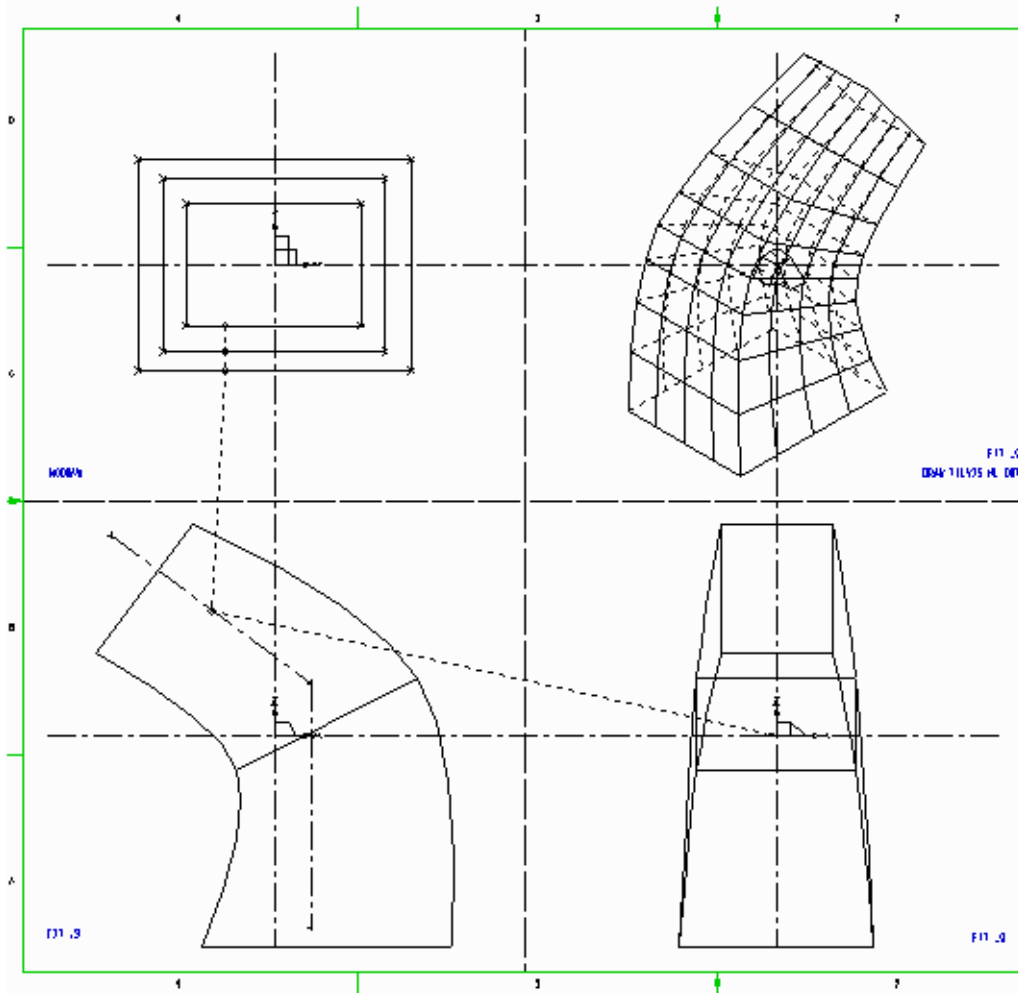
Figure 123 Smoothing the Model with FUNV 11 Point Functions



For default settings to smooth models, please refer to [“The 3D Default Settings”](#) on page 15.

Figure 124 shows the completed model. Compare this with the unsmoothed model shown in Figure 122.

Figure 124 The Completed Model



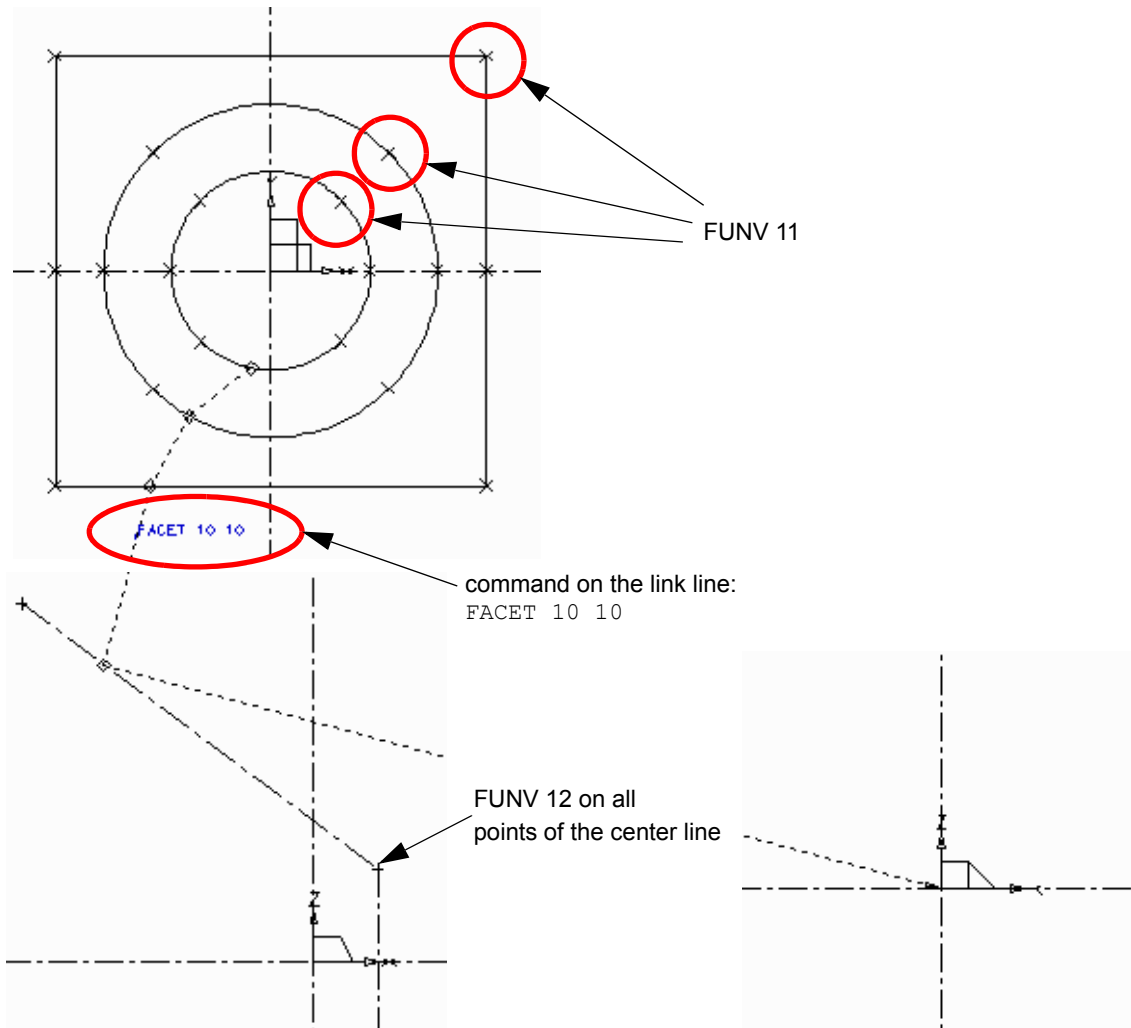
Matching Points on Profiles

The points of each profile must match corresponding points on the other profiles in the definition. This is so the Modeler can attach the spine of the model to the profiles. Any number of points can be used (up to the MEDUSA4 line limit of a 1000 points) to create a duct profile, but one point function of the following types must exist at each matching point:

FUNV 12	Generates an unsmoothed model
FUNV 11	Generates a smoothed model

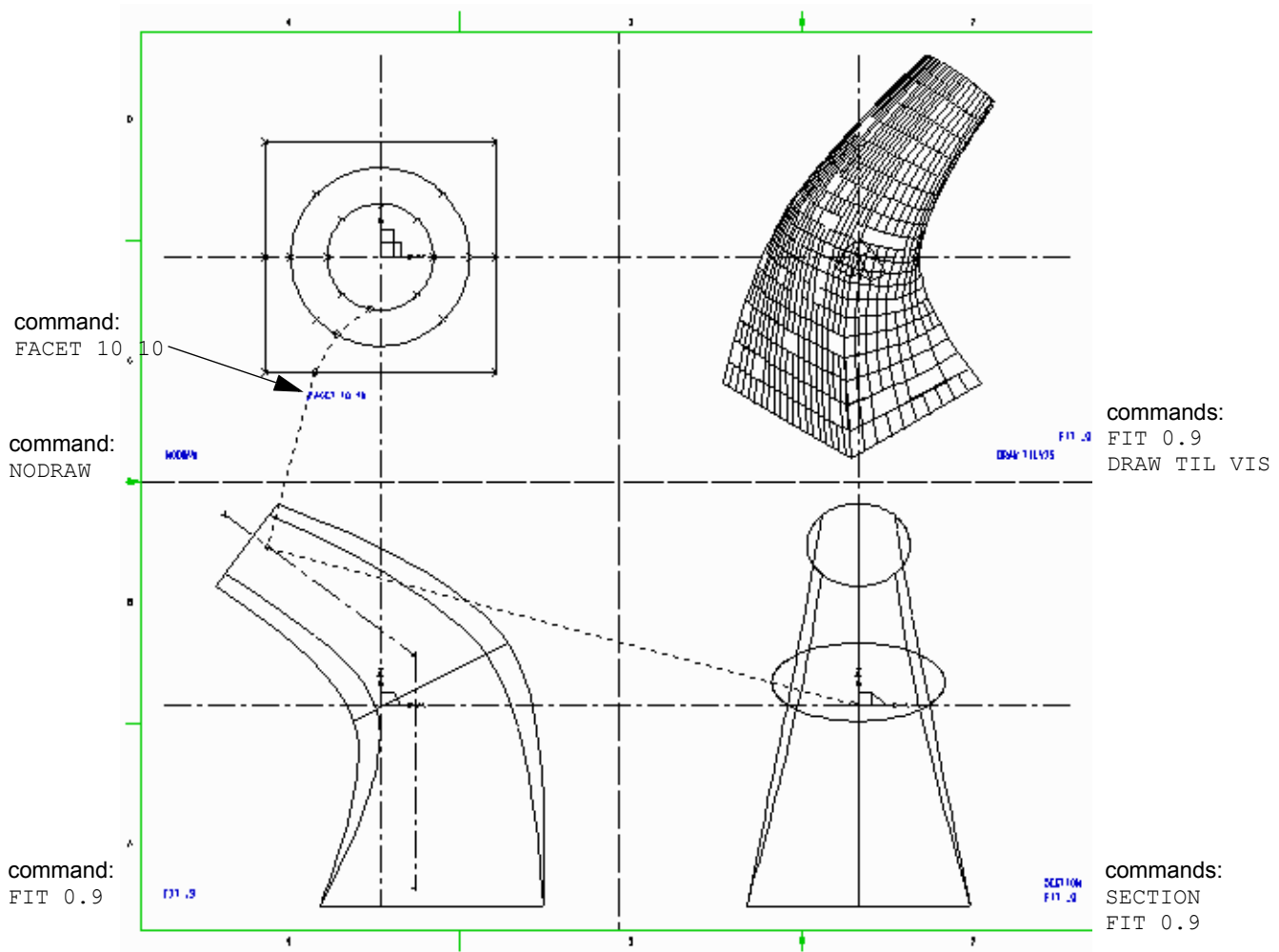
Point matching allows complex objects to be modeled. For example, [Figure 125](#) shows the definition of an irregularly shaped duct using octagonal, circular, and rectangular profiles. Six cross point functions (FUNV 11) are placed on each profile so that the Modeler can align and match these profiles. [Figure 126](#) shows the completed definition and model. For more detail on the techniques of point matching for profiles read [“Ruled Surfaces” on page 111](#).

Figure 125 Matching the Points On Profiles



Please note: The `FACET` command used in the model definition controls the surface definition of the model by specifying the number of tiles in a patch. This command is discussed in detail in “[Meshed Surfaces](#)” on page 191.
The default value is `FACET 4 4`.

Figure 126 The Completed Model



Controlling Model Shape with Profile Orientation

The point functions used on the spine determine both the position at which the profiles are attached to the spine of the model and the angle of the profiles relative to the spine. This affects the shape of the resulting model.

The following types of point function can be used on the spine:

FUNV 12	Attaches the profile to the spine so that it is aligned midway between the following and preceding spine segments.
FUNV 14	Attaches the profile to the spine in-line with the following spine segment.
FUNV 15	Attaches the profile to the spine in-line with the preceding spine segment.

Please note: The previous examples shown in this chapter use FUNV 12 points on the spine.

Figure 127 shows an example of a basic duct definition.

Figure 127 A Basic Definition

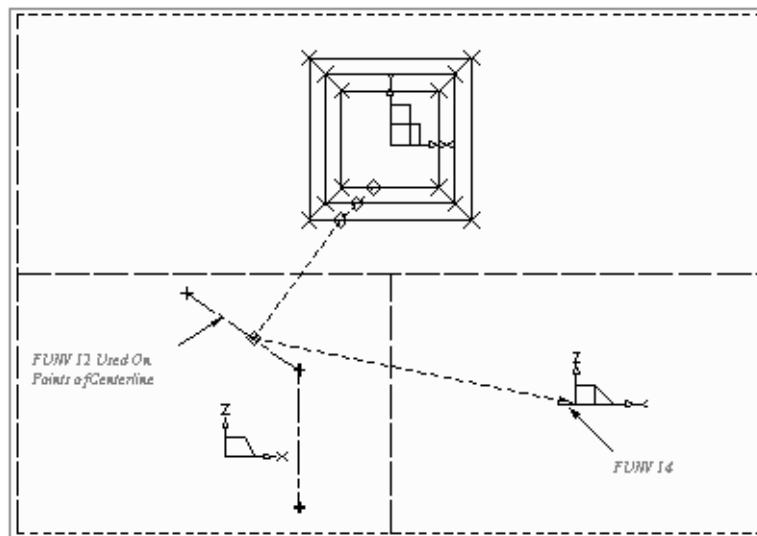
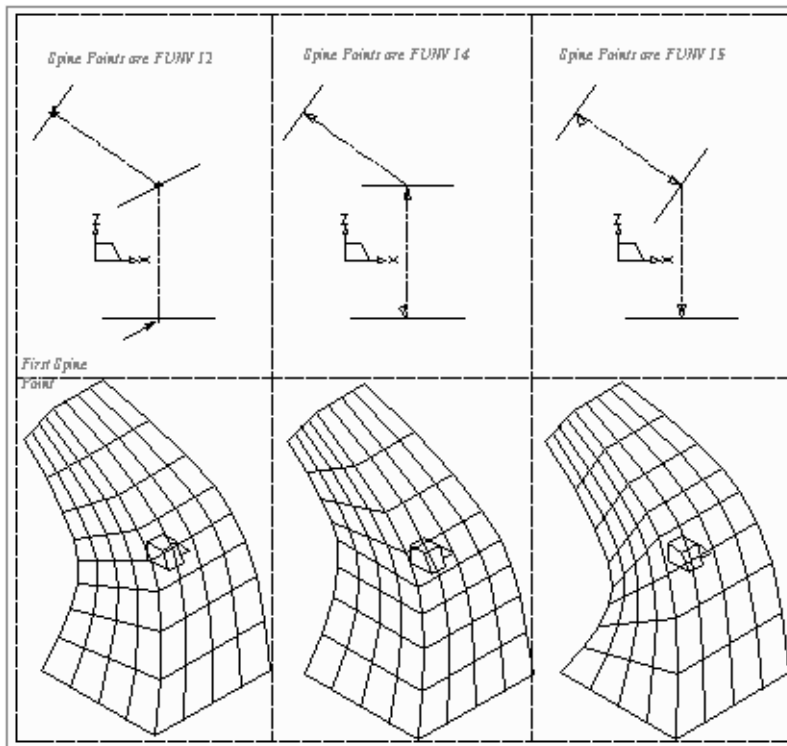


Figure 128 shows the effect on model shape of changing the point functions on the spine in the duct definition example in Figure 127. The lines drawn on each spine show how each profile is attached when using the different point functions. These lines are not part of the definition.

Figure 128 The Effect of Using Different Point Functions On the Spine



More than one type of point function can be used on a spine simultaneously. For example, [Figure 129](#) shows the definition of a spine that uses all three types of point function.

Creating a Model Using Non-Planar Centerlines

Non-planar (3D) centerlines can be used to create a more complex model, such as a helix. This is done by defining a 3D spine in two orthogonal viewboxes. [Figure 129](#) shows an example of such a definition.

Each centerline in the spine definition contains the same number of matched points as for a 3D wire model. The link line is attached to the primary centerline using a diamond point function (FUNV 10) and to the secondary centerline using a plus point function (FUNV 12). Note that the position of the secondary centerline in the viewbox defines the position of the model in the third dimension. Depth texts or arrowhead point functions are therefore not required.

Figure 129 Definition of a More Complex Model

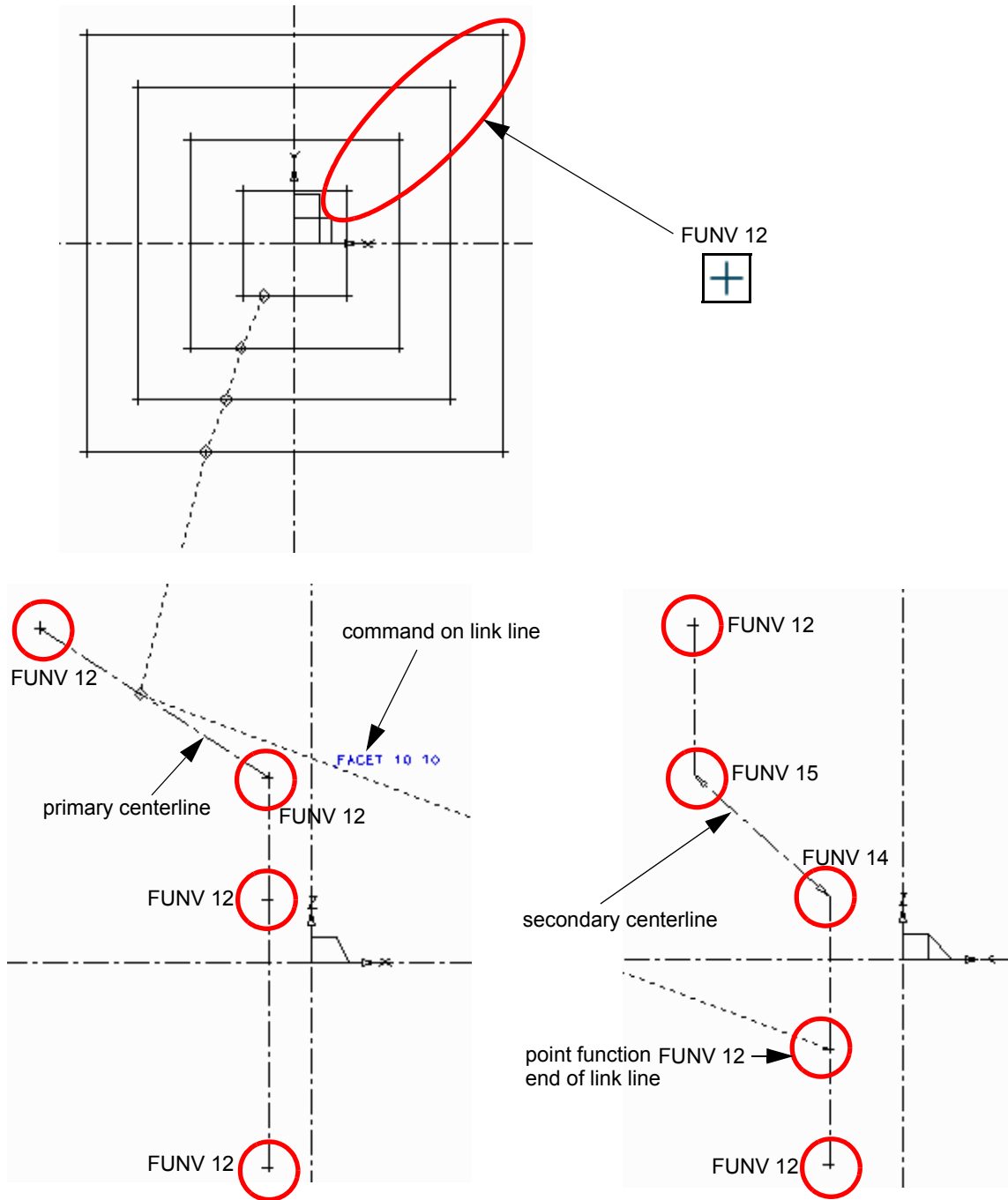
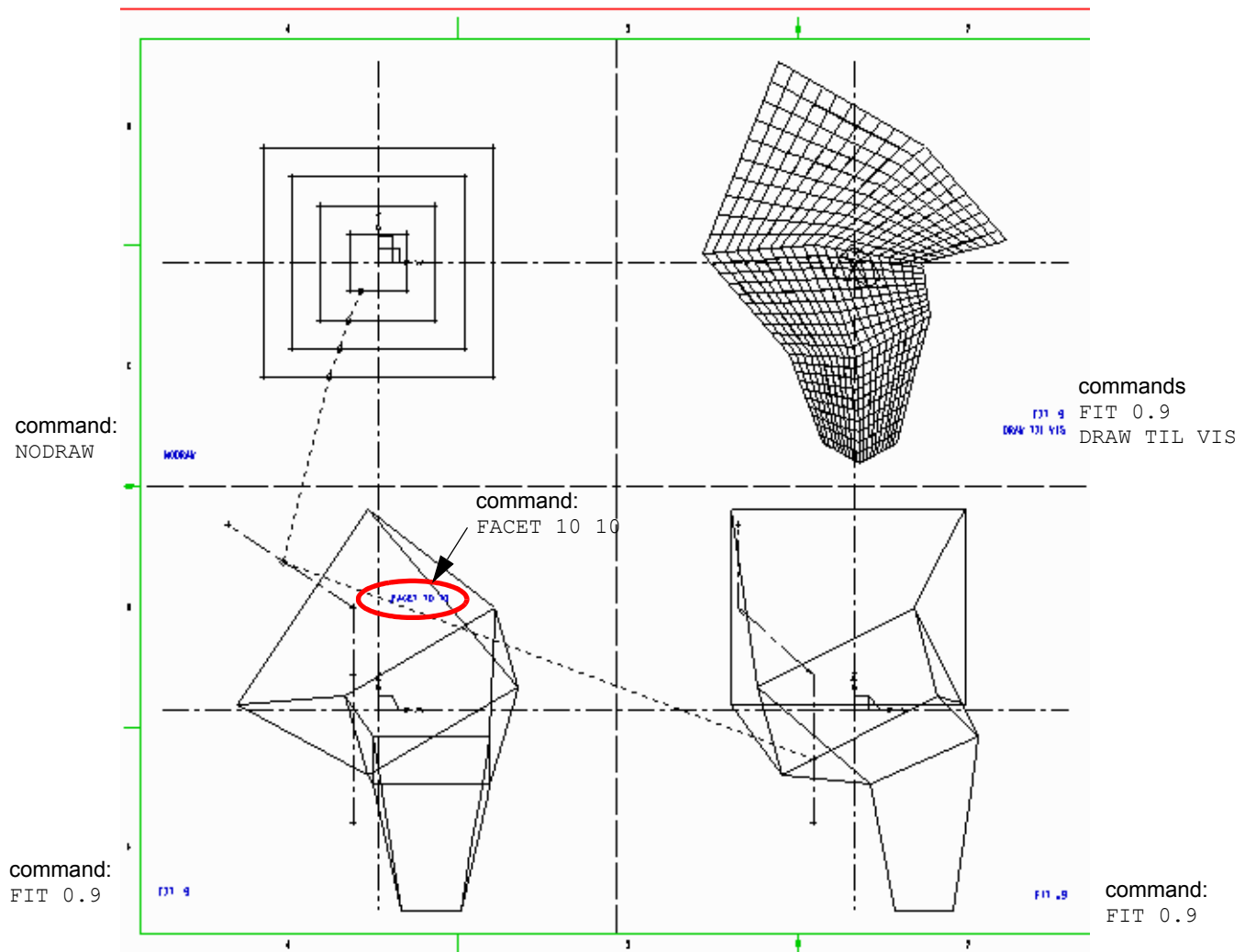


Figure 130 shows the completed model.

Figure 130 The Completed Model



In addition to the method described above, the projection method can be used to create complex duct models from a centerline formed by projection onto a curved plane. See “Wires” on page 151, for details on how to create a projected centerline.

Summary of Element Types

The following table gives a summary of the element types and point functions used to create models with the Duct generator.

Element Description	Element Style and Point Functions
Line Types	
Profile Lines	Profile LPO - LP9
Centerline	center line
Link Line	Duct Generator
Point Functions	
Pick up profile lines	FUNV 10
Pick up secondary centerline	FUNV 12
Pick up projected centerline	FUNV 21
Point matching (smoothed)	FUNV 11
Point matching (straight line)	FUNV 12
Indicate third dimension on the link line	FUNV 14 / FUNV 15
Align profile to the local spine segment	FUNV 12
Align profile to the following spine segment	FUNV 14
Align profile to the preceding spine segment	FUNV 15
Text Types	
Depth texts	Modeller Text ass. by Geometry (TMG)



POLYGONS

A shell model may be generated by defining it as a set of polygonal faces. This feature may be used in the MEDUSA4 Sheet Metal Design system.


The method consists of drawing each polygonal face as a profile in two complementary views in the model definition. Each profile must contain the same number of corresponding points, which means that the first point in one profile must correspond with the first point in the other profile, and so on. The two profiles are then joined by a link line of style `Face Poly Generator`. One profile is picked up by a diamond point function (FUNV 10) and the other by a plus point function (FUNV 12).

- [A Simple Model Definition..... 182](#)
- [Locking a Link Line to a Profile 186](#)
- [Summary of Element Types..... 189](#)

A Simple Model Definition

An example of this modeling technique is the creation of a tetrahedron as shown in the following figures. For clarity the face definitions are shown in separate view box pairs, whereas in practice they would all be drawn in the same pair as shown in [Figure 136](#).

The first figure ([Figure 131](#)) describes in detail, how to create one face of the model definition. Any other face has to be created in the according way (see [Figure 132](#) and [Figure 133](#)).

1. Select one of the profile line tools, e.g. Solid Thin  (LP0 through LP9).
2. Create a triangle in the first view box, in our example it is the view box containing the DXY prim.
3. Create an according triangle in the second view box, in our example, the view box containing the DXZ prim.

Please note: The second profile has to be drawn in the same sequence as you draw the first profile.


4. Choose the Polygon tool  from the link line toolset.
5. Probe on the first profile line in the first view box and then the second profile line in the second view box.
Point function FUNV10 is automatically assigned to the first point, FUNV12 to the second one.

Figure 131 Definition of the First Face of a Tetrahedron

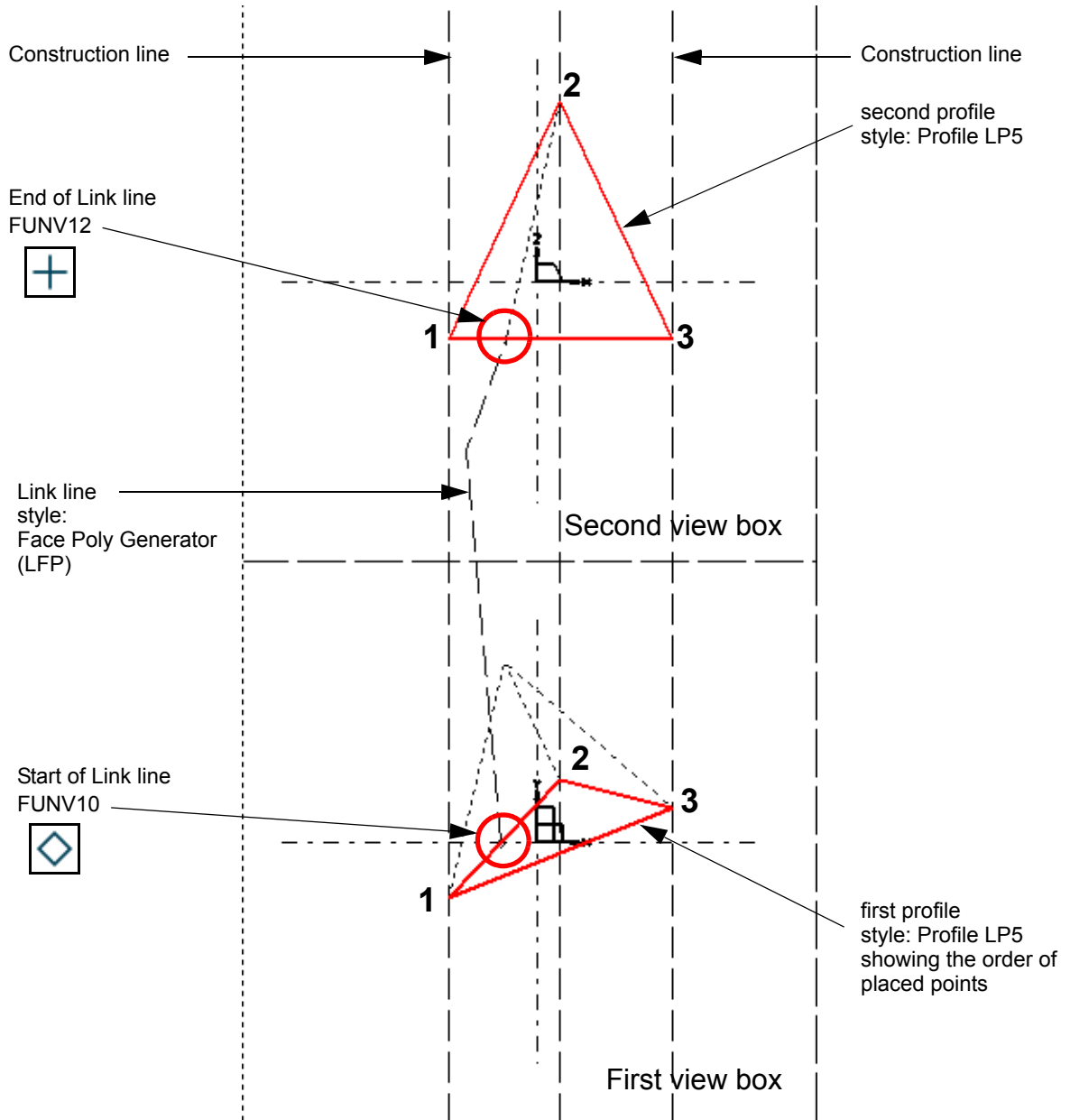


Figure 132 Definition of Second and Third Face of a Tetrahedron

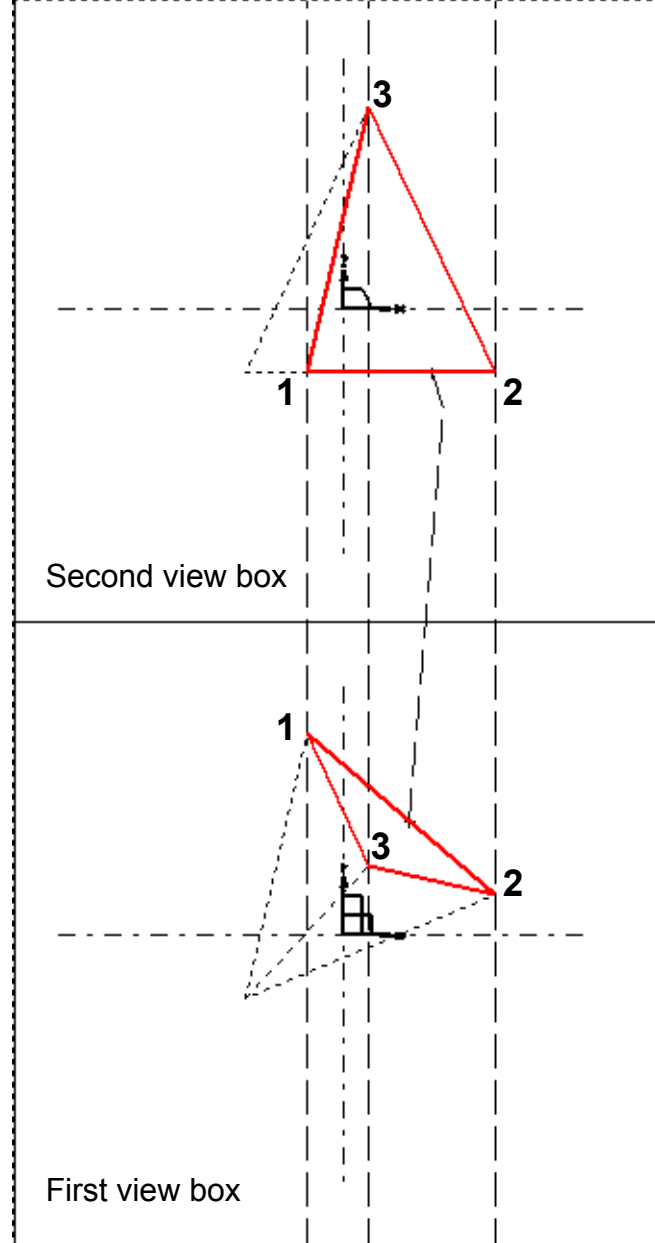
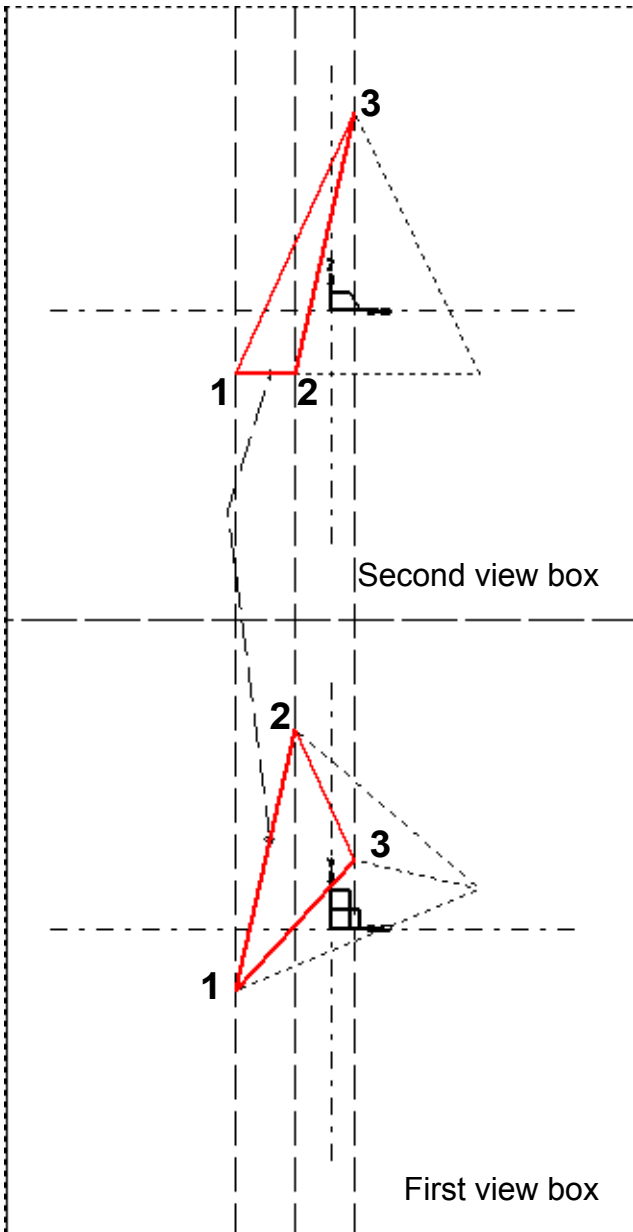
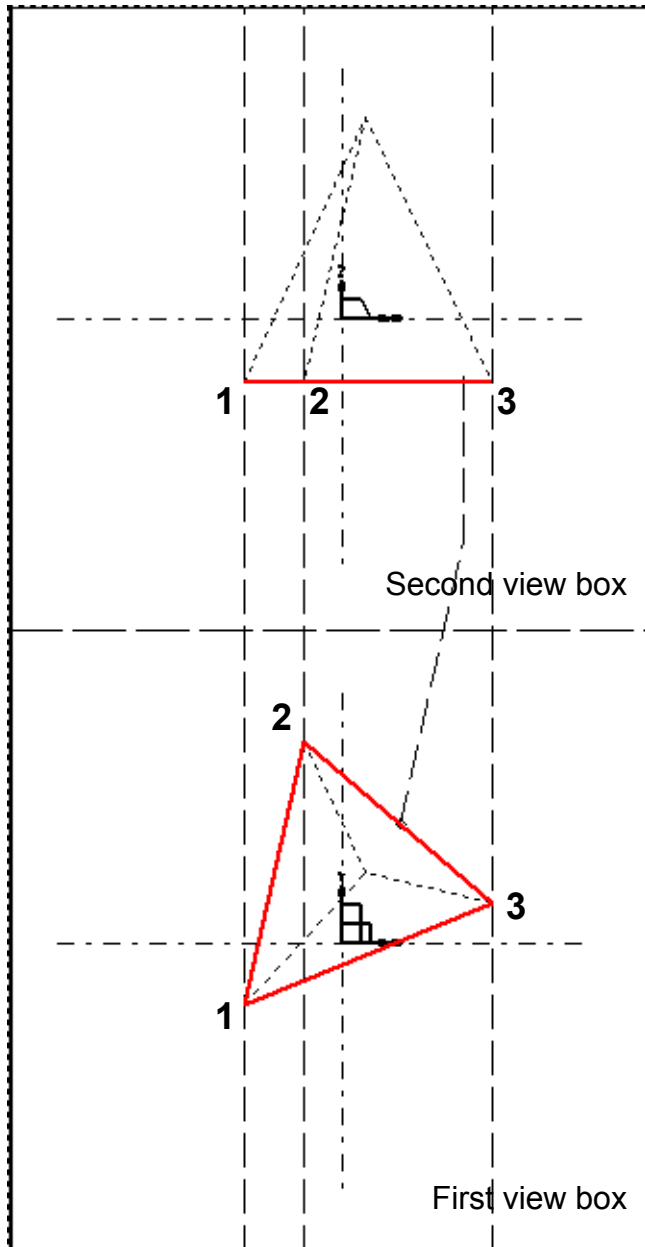


Figure 133 Definition of the Fourth Face of a Tetrahedron




Locking a Link Line to a Profile

A means of locking each link line to a specific profile is provided by the 3D group. A 3D group only accepts profile lines and link lines as members.

This locking capability is needed because sometimes it is necessary (or convenient) to draw profile lines superimposed on each other in a model definition. This can occur when defining model faces as polygons, as shown in [Figure 136](#) for example. In such a case, there is often no part of a profile which is not superimposed on another, and therefore that profile cannot be unambiguously picked up. Thus it is likely that the Modeler will associate a profile with the wrong link line.

This is the procedure to follow when you need to lock a link line to a profile:

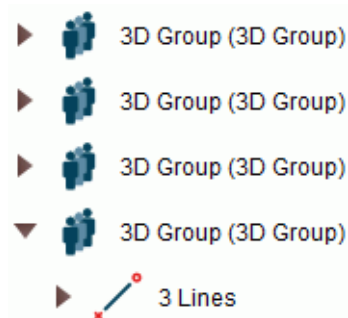
1. Choose the Empty Group tool  to open a 3D group.
2. Draw the profile line.
3. Select and attach the link line to the profile line, and then complete the link line.
4. Exit the tool and with it you close the group.

The definition of the tetrahedron consists of four different 3D groups. Each group consist of two profile lines and one link line.

To ensure that two required profile lines and the assigned link line are members of the same group:

- choose one of the group selection tools and select the link line, or
- select the 3D Group (3D Group) within the structure tree.

Figure 134 Selecting a 3D Group in the Structure Tree



The link line and the assigned profiles are highlighted as shown in [Figure 135](#).

Figure 135 Checking a 3D Group

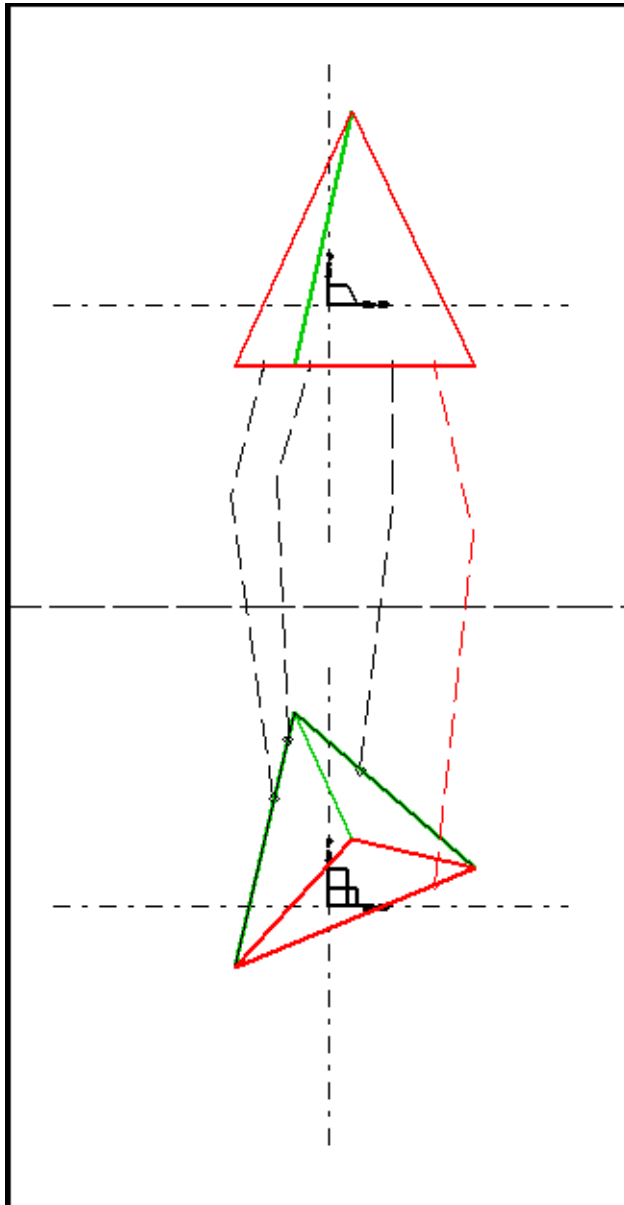
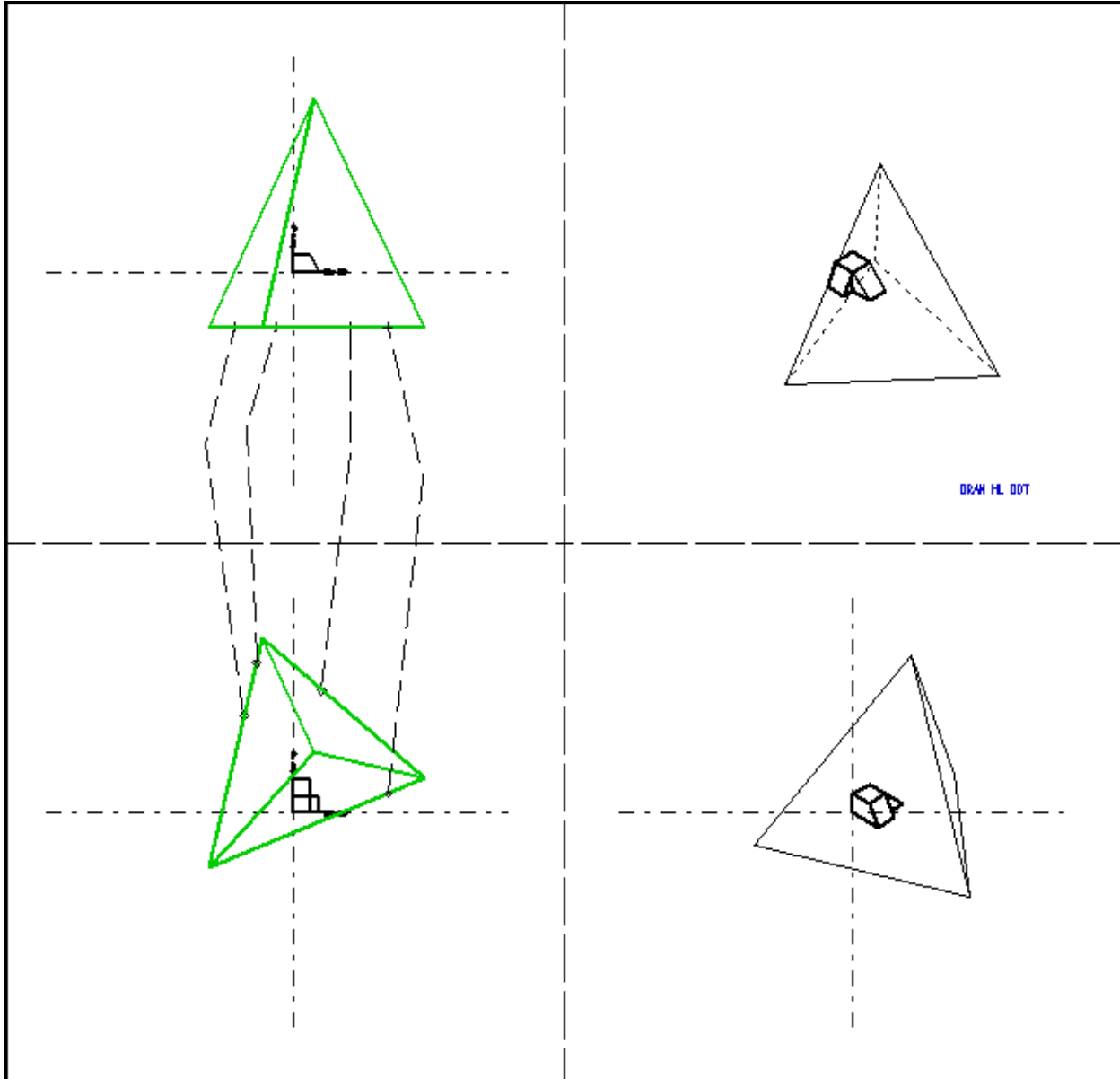


Figure 136 shows the shell model definition of the figures above reworked using the 3D group method and the generated model.

Figure 136 The 3D Group Method Used On the Tetrahedron Definition



Summary of Element Types

The following table gives a summary of the element types and point functions used to create models with the Polygon generator.

Element Description		Element Style and Point Functions
Profile Lines		
	All Polygon profiles	Profile LP0 - Profile LP9
Link Lines		Face Poly Generator
Point Functions		
	Pick up first profile lines	FUNV 10
	Pick up secondary profile lines	FUNV 12
Group Types		
	Lock link line to profile line	3D group (Profiles+Linklines)

MESHED SURFACES

The Meshed Surface generator produces a model from a mesh of wires that defines the model surface. It is used to model objects which have complex, doubly-curved surfaces.

The wires in the defining mesh are called **longitudes** and **latitudes**. Longitudes are open wires which run the length of the model, and latitudes are closed wires which encircle the model. The surface tiling is built on this mesh when the model is generated. Multiple link lines of style `Line Generator` (Wire Lobster tool) are used in the definition of meshed surface models.

You will need a good understanding of wire modeling to be able to use the Meshed Surface generator effectively, and a review of chapter “[Wires](#)” on [page 151](#), is recommended.

Please note: Any commands described within this chapter can be added as attribute or placed as text.

- [Defining a Meshed Surface Model](#)..... 192
- [A Meshed Surface Definition](#) 197
- [The WIRE Command](#)..... 201
- [The MESHTOL Command](#)..... 203
- [Surface Definition](#)..... 204
- [Summary of Meshed Surface Models](#)..... 207
- [Summary of Element Types](#)..... 208

Defining a Meshed Surface Model

A Meshed Surface model definition consists of the following elements:

- Latitude wire line definitions
- Longitude wire line definitions
- Linkline attribute or TMG-text of style `Modeller Text ass by Geometry`
The following examples use text for the commands, so that you can see the commands, which are applied.

This section describes each of these elements, as well as the idea of solder points and the use of solder prims.

Latitudes and longitudes define profiles along the length and around the circumference of a model. The number used depends on how much control over the geometry of the model is required. Exercising a high degree of control means using many latitudes and longitudes in the definition.

All latitudes and longitudes must carry a TMG-text of style `Modeller Text ass. by Geometry`. It is this text which instructs the Modeler to use the Meshed Surface generator instead of the Wire generator.

The Barrel Analogy

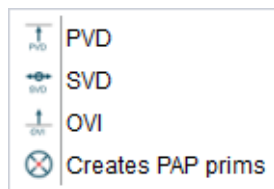
A useful analogy to keep in mind when defining a Meshed Surface model is that the model is like a barrel. The longitudes on the model can be likened to the staves of the barrel, with the latitudes representing the hoops. The surface shape of the model is topologically identical to the barrel because, however complex the shape of the model is made, it remains a continuous object with two flat ends.

Solder Points and Prims

The points at which latitudes and longitudes intersect on the surface of the model are called solder points. Solder points can be thought of as points where the wire mesh is soldered together.

There should be a probed point on the profile lines defining the latitudes and longitudes at the positions corresponding to each solder point. If this is not the case then a solder prim (style P_{AP}) must be placed at these positions instead. This prim is available in the toolset for oblique view prims tools.

Figure 137 Toolset of Oblique View Prims with the Solder Prim Tool



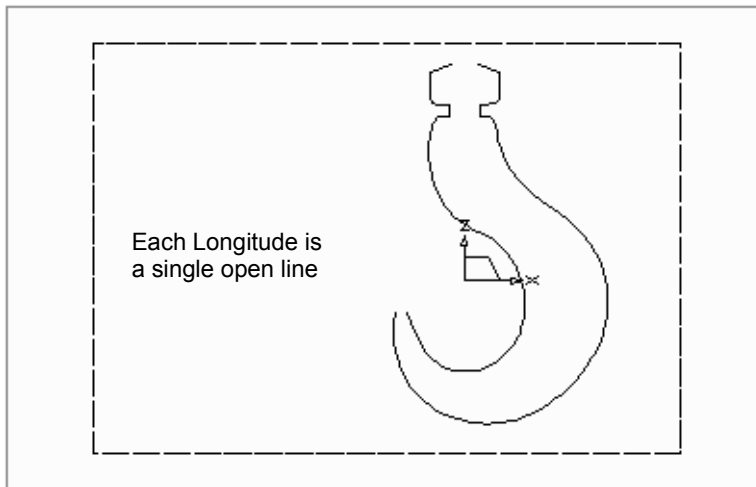
A mixture of probed points and solder prims can be used.

Longitudes

Longitudes are open wire definitions which define the lengthwise shape of a model. They can be planar or non-planar (see [“Wires” on page 151](#)). At least two longitudes are required in a model definition but you may use as many as are needed for acceptable control of form. An important point is that longitudes must not cross each other, just as the staves of a barrel do not cross (see [“The Barrel Analogy” on page 192](#)).

The number of points used in each longitude should at least equal the number of latitudes drawn, and normally these points will be where the lines are intended to mesh. This is because latitudes are intended to mesh with longitudes only at points in 2D space at which both lines possess a point (a solder point). [Figure 138](#) shows longitudes used to model a crane hook.

Figure 138 Longitudes Used To Model a Crane Hook

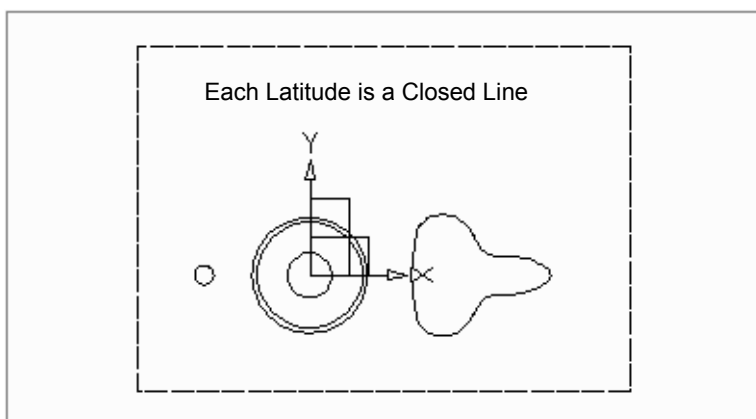


Latitudes

Latitudes are closed lines which define the sectional form of a model. They are used at every section of the model where exact control of form is required and can be planar or non-planar (“Wires” on page 151). An important point is that latitudes must not cross each other, just as the hoops of a barrel do not cross (see “The Barrel Analogy” on page 192).

At least two latitudes must be used but, as for longitudes, you may use as many as are needed for acceptable control of form. Figure 139 shows the latitudes used to model the crane hook.

Figure 139 Latitudes Used To Model the Crane Hook

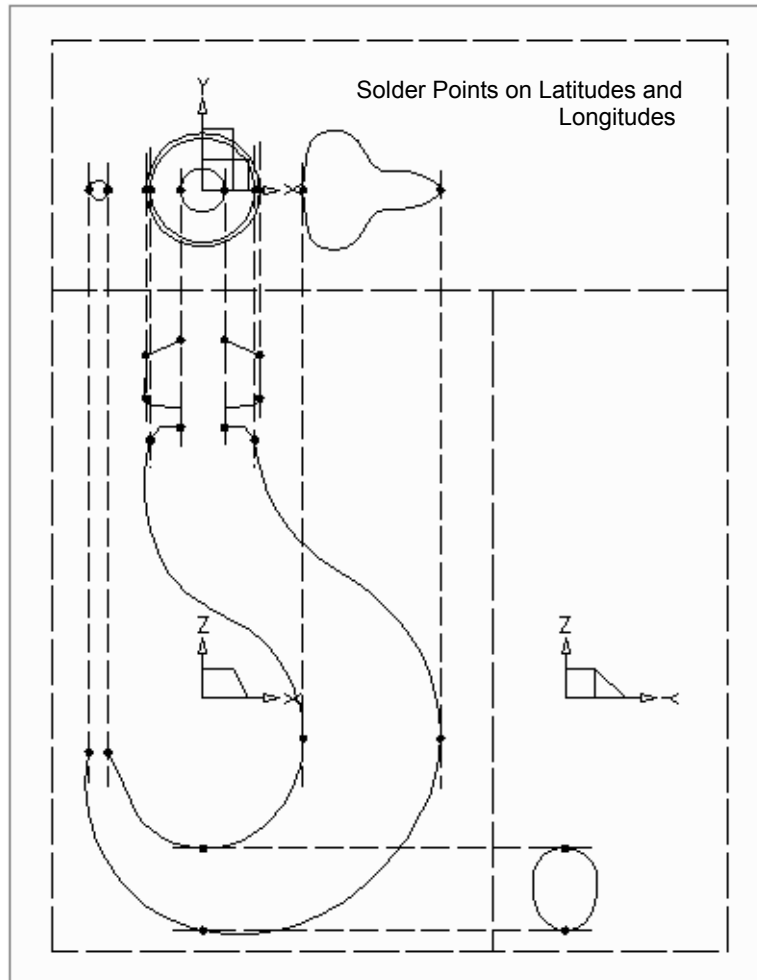


The following constraints apply to latitudes:

- If a model definition contains only two longitudes, the line direction of the latitudes must be such that they encircle the model in the same direction. If this is not the case a distorted model will be produced.

- Each latitude should contain a point (a solder point) at the exact position where it is to be meshed with a longitude point, as shown in [Figure 140](#).

Figure 140 Matching Latitudes and Longitudes



Circular Latitudes

Care must be taken in the construction of circular latitudes. For example, if you draw a circular latitude as a center-point circle, the Modeler may have difficulty in matching the latitudes to the longitudes.


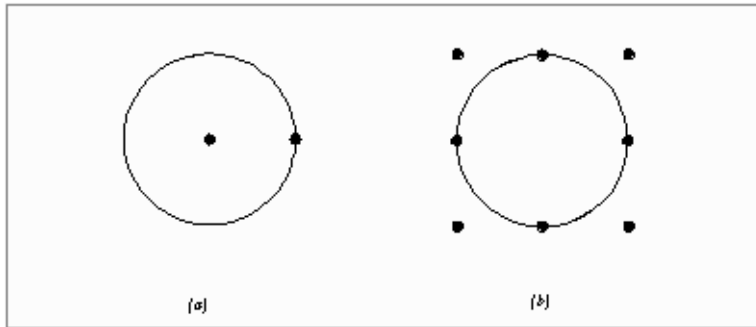


This is because, as [Figure 141](#) (a) shows, there are only two points in a center-point circle created with the `Circle Center/Point` tool ; one at the center and one at the circumference. The Modeler may have difficulty with this as the direction is neither clockwise nor counterclockwise. Therefore, to ensure a definite direction, draw each circular latitude as four circular arcs as shown in [Figure 141](#) (b).

Figure 141 Two Ways Of Constructing Circular Latitudes



For creating the circular arcs first use the Circular arc through 3 points tool  and then the Convert the selected Circular arc to tangent arc tool , both you can find in the Dashboard.

Please note: The direction of each of these arcs must be the same.

Naming Latitudes and Longitudes

All latitudes and longitudes defining a model must be named by a TMG-text of style `Modeller Text ass. by Geometry` of the form:

LAT *name*
and
LON *name*

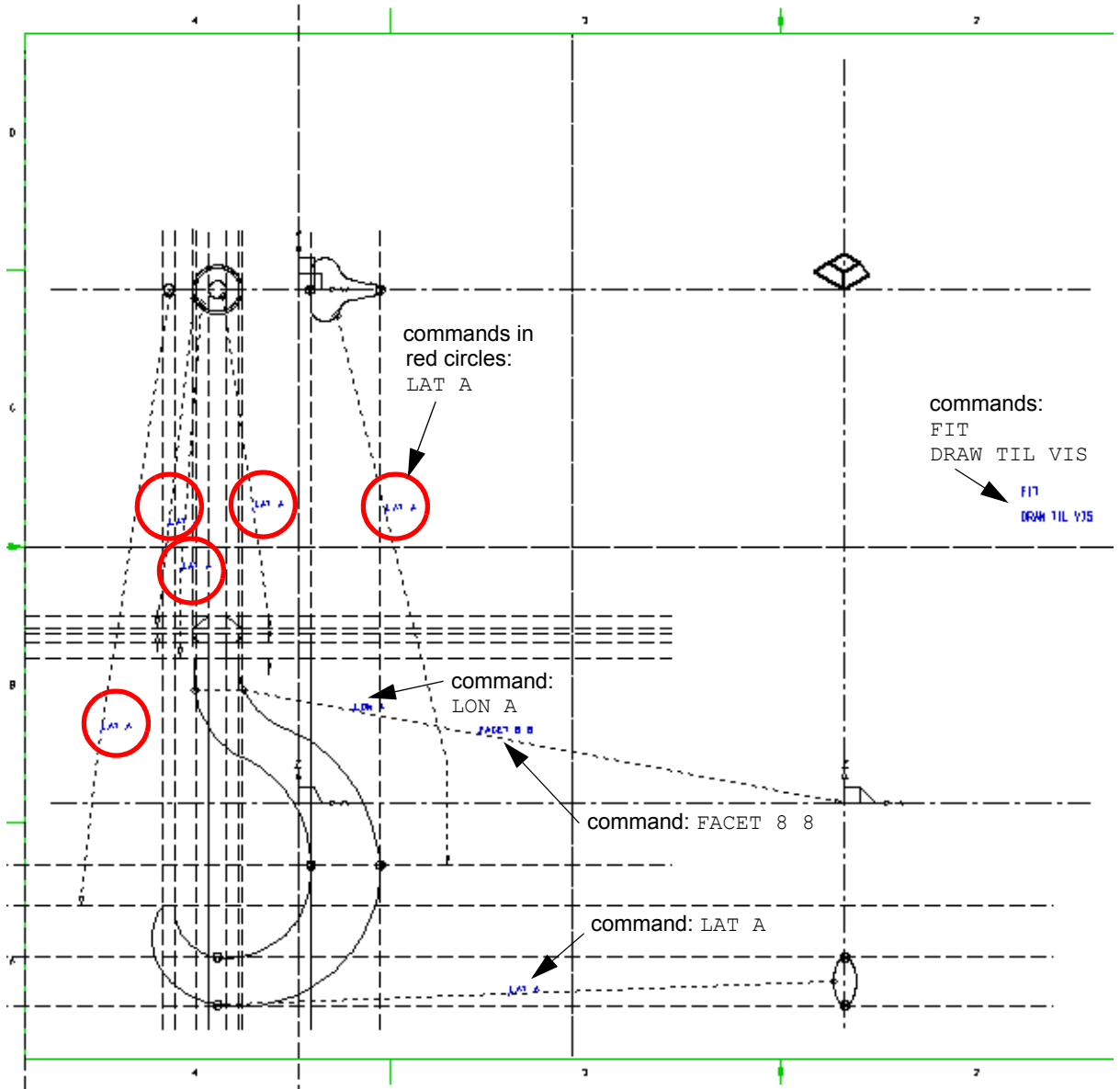
The *name* text can be any convenient alphanumeric string but must be the same for any one object.(see [Figure 142](#)).

The naming text is attached to each link line in the model definition by using a segment probe.

A Meshed Surface Definition

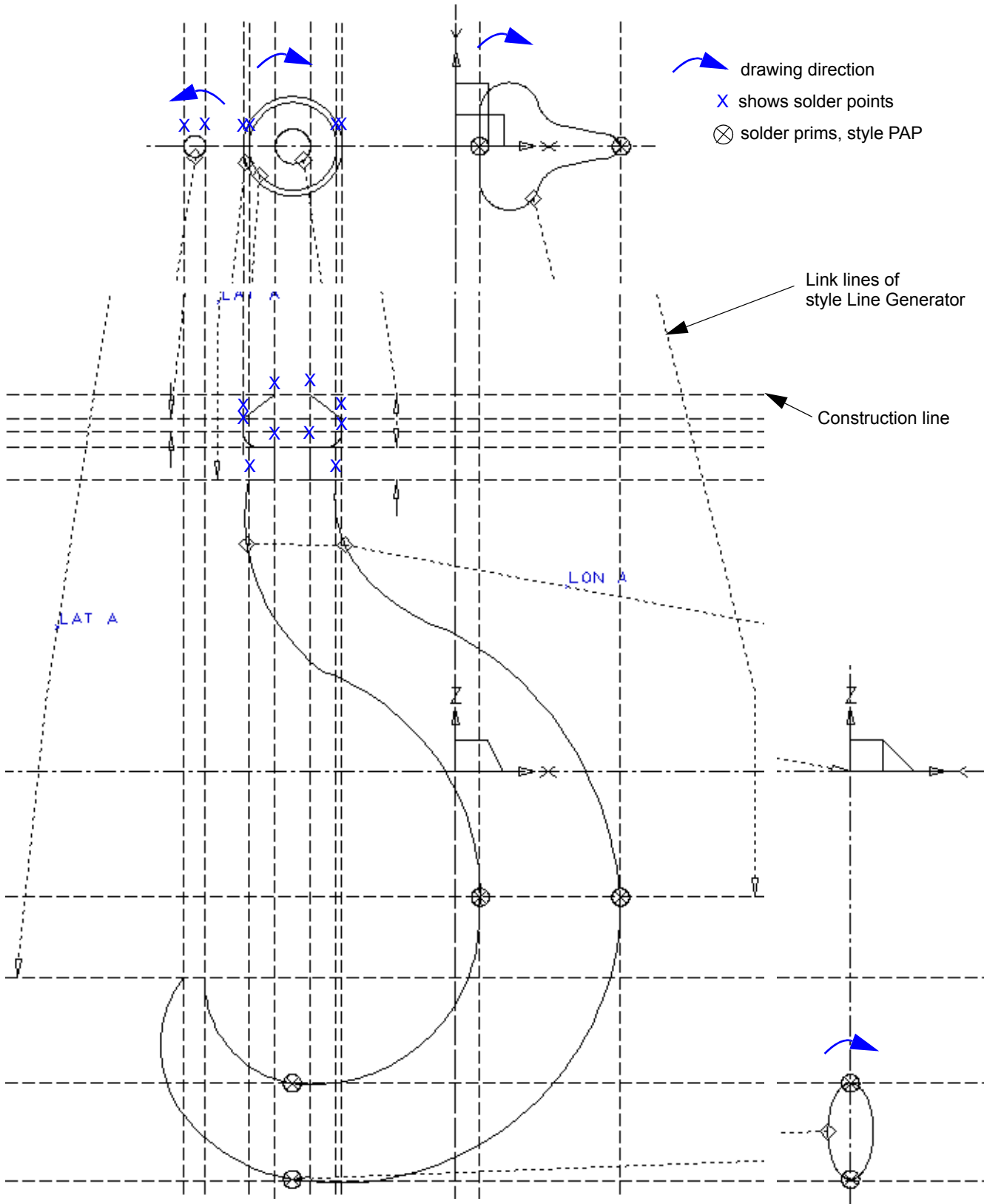
To create the model of the crane hook, follow the procedure described below. [Figure 142](#) shows an overview of the complete model definition. The required command texts are specially indicated. In order to show how you can construct the longitudes and latitudes exactly and to show the relation between the solder points the construction lines are visible in the sheet.

Figure 142 The Model Definition of the Hook



The following [Figure 143](#) shows details of the definition, for example, you can see where and which point functions are used, and where the link lines end.

Figure 143 Details of the Model Definition






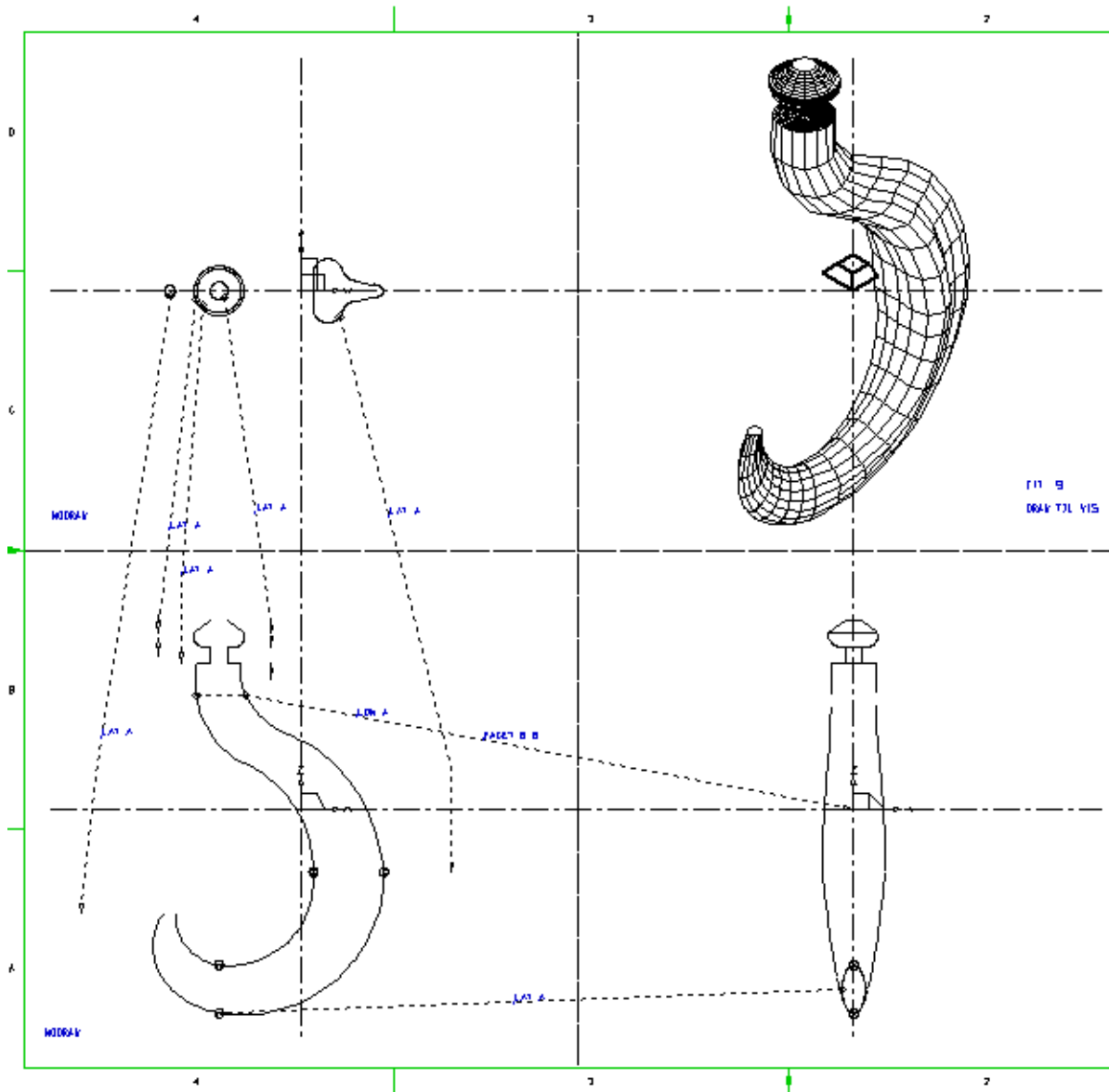
1. Draw the latitudes in the XY viewbox using a profile line of style `Profile LP0`. Construct each circular latitude from four arcs and ensure that the leftmost latitude is drawn in the opposite direction to the others. Ensure that there are solder points on each latitude as shown in [Figure 143](#). The arrows  show the drawing direction of the latitudes. It is essential to create the latitudes in the same direction!
2. Draw the two longitudes at the XZ viewbox using a profile line of style `Profile LP0`. Ensure that each longitude contains solder points as shown in [Figure 143](#).
3. Choose the Wire Lobster tool  and draw link lines of style `Line Generator as` shown in [Figure 142](#) and [Figure 143](#).
Please consider that the link lines give the right heights as shown in the details ([Figure 143](#)). Construction lines help you to draw them exactly.
4. Attach the TMG-text `LAN A` (style `Modeller text ass by Geometry` to all latitude link lines as shown in [Figure 142](#).
How to add command texts is described in chapter "[Basics of MEDUSA4 3D](#)" and section "[Depth Text](#)" on page 28.
5. Attach the TMG-text `LON A` (style `Modeller text ass by Geometry` to the longitude link lines as shown in [Figure 142](#).
6. Place the Viewer command `TIL VIS` (TVS text tool) in the isometric viewbox.
7. Select the Model + Reconstruct tool  to generate and view the model.

Figure 144 shows the completed model.



Figure 144 The Completed Model



The WIRE Command

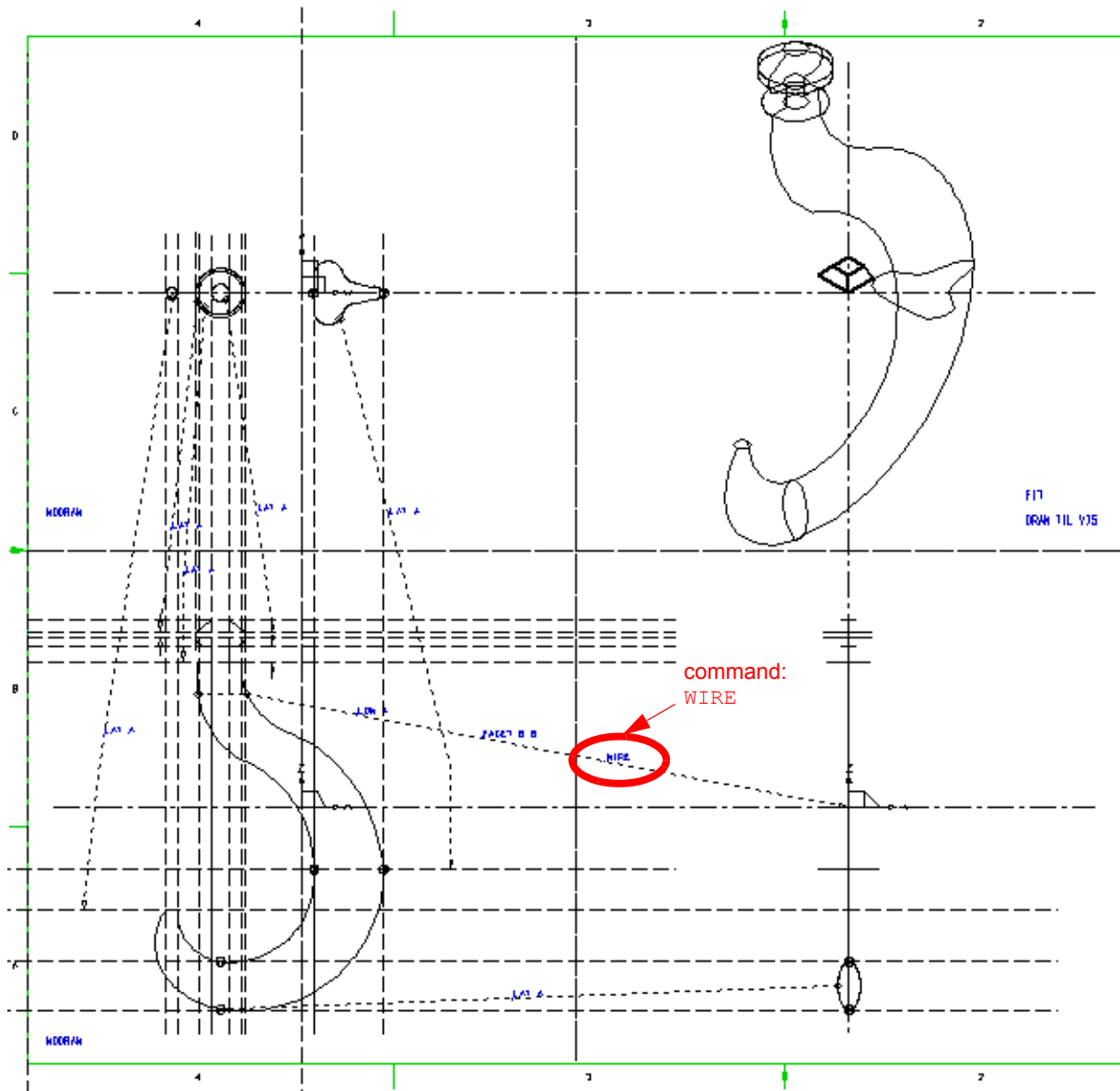
The `WIRE` command can be used for rapid verification of a meshed surface definition. This command cancels the effect of the text of style `Modeller text ass. by Geometry` on the link lines, and therefore a wire model is generated from the latitudes and longitudes. This is always created faster than a solid Meshed Surface model.

To use this facility:

1. Click on the `Creates a TMG text tool` .
2. Enter the `WIRE` command in the text input field.
The text is attached to the cursor.
3. Click on an arbitrary link line, either a longitude or latitude link line, to place the command.
4. Select the `Model + Reconstruct tool`  to generate and view the model.
5. If this verifies the definition (as in [Figure 145](#)), remove the `WIRE` text from the link line and run the Modeler again to produce the solid Meshed Surface model.

If the definition produces error messages instead of a model, modify the definition to try and remove the error and remodel. Repeat this process until the definition produces a wire model. When this has been done, the final model can be generated as described above.

Figure 145 Verifying a Meshed Surface Definition By Generating a Wire Model



The MESHTOL Command

The `MESHTOL` command specifies a meshing tolerance value. The purpose of this is to allow a slight mismatch of points in the model definition. This feature may be useful in some cases, although it is always preferable to use definitions which match exactly.

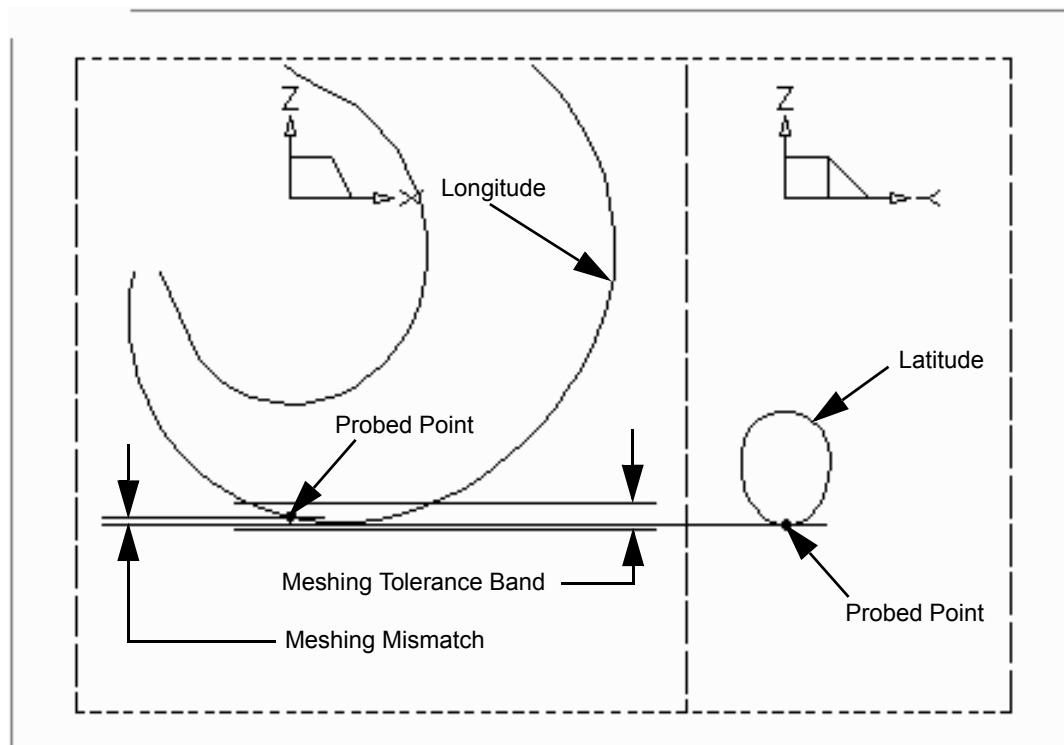
The tolerance value specifies the maximum axial separation (mismatch) that may be allowed to exist between points on the latitude and longitude lines which together define a solder point. If this value is exceeded then the points are not considered to be associated. The `MESHTOL` command is specified as TMS-text of style Model Specification Text as follows:

```
MESHTOL value
```

value is specified in sheet units. The default is 0.1. The Modeler will move the latitudes to meet the nearest longitudes if the mismatch between the points is not greater than the `MESHTOL` value.

Figure 146 shows the meshing tolerance principle using part of the hook definition as an example. In this example the mismatch is within tolerance and therefore would be acceptable to the modeler.

Figure 146 Mismatched Points Within the Meshing Tolerance



Surface Definition

The surface mesh of a Meshed Surface model can be controlled in two ways:

- With the `FACET` command
- By adding alignment points to the model definition

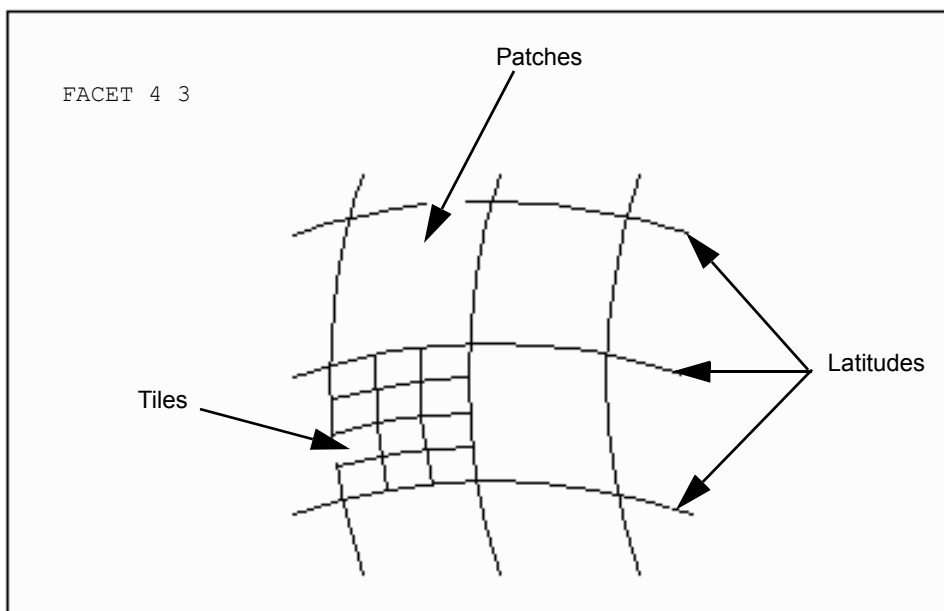
The FACET Command

The `FACET` command controls faceting (the arrangement of tiles) in a patch. It operates by dividing the sides of the patches into a number of longitudinal divisions (n); and latitudinal divisions (m); each division forming the side of a tile. The command is:

```
FACET n m
```

For example, the command `FACET 4 3` divides each longitude segment of a patch into 4 equal divisions, and each latitude segment into 3 equal divisions, giving a total of 12 tiles per patch as shown in [Figure 147](#). The default facet level is `FACET 4 4`.

Figure 147 Effect of the `FACET` Command



The `FACET` command allows curves and subtle changes in shape to be shown in greater detail. Increasing the number of tiles in a patch emphasizes the surface detail (and increases accuracy) but slows down the process of model generation and display.

Please note: The command `TIL VIS` must be used with the `FACET` command to display the surface mesh.

The `FACET` command can only be used with the Meshed Surface and Duct generators. To control faceting for other model generators, use the `CHOTOL` command. For more information on the `CHOTOL` command see the chapter [“General Modeler Commands” on page 281](#).

The `FACET` command is specified as TMG-text of style `Modeller Text ass. by Geometry` and can be attached to any one of the link lines in the definition sheet; it applies to the whole model.

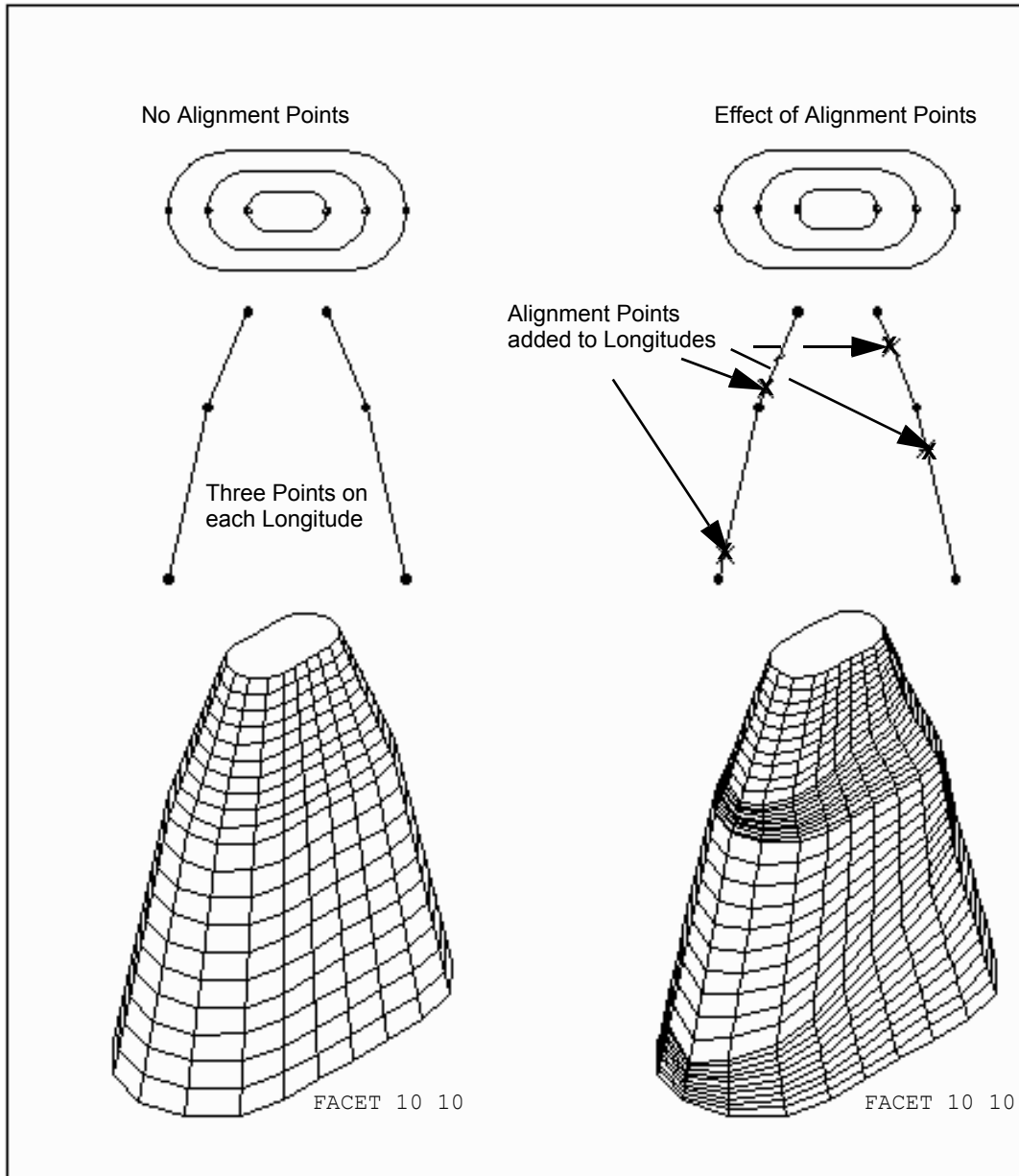
Alignment Points

Alignment points can be added to the longitudes of a Meshed Surface model definition to modify the surface mesh on the model. This feature can be useful for increasing tile density, and therefore enhancing detail, in places of high curvature on the model.

These points are added to a model definition by attaching pairs of cross point functions (`FUNV 11`) to the longitudes. This has the effect of dividing up the default patches to create more patches. The new patch boundaries are defined by the alignment points as shown in [Figure 148, “Effect of Alignment Points On Longitudes” on page 206](#).

[Figure 148](#) shows the same Meshed Surface model generated with and without alignment points on the longitude definitions. The command `FACET 10 10` was used on both models.

Figure 148 Effect of Alignment Points On Longitudes



Summary of Meshed Surface Models

A checklist for Meshed Surface model definition is described below:

1. All longitude and latitude definitions must be made in accordance with the rules for generating wire models.
2. All latitudes must encircle the model in the same direction if there are only two longitudes in the model definition.
3. A probed point or a `PAP` prim must exist on each latitude so as to match a corresponding point or a `PAP` prim on a longitude.
4. Longitude link lines must be named by the `LON` command.
5. Latitude link lines must be named by the `LAT` command.
6. The same name must be given to all latitudes and longitudes that define a single model.
7. The `WIRE` command can be used to verify a Meshed Surface definition.
8. The `MESHTOL` command can be used to specify the meshing tolerance.
9. The `FACET` command can be used to control facetting (the arrangement of tiles in a patch). Tiles are only visible when using the Viewer command `TIL VIS`.
10. Alignment points (FUNV 11) can be used to selectively enhance surface definition on a model.

Summary of Element Types

The following table gives a summary of the element types used to define a Meshed Surface model.

Element Description		Element Styles and Point Functions
Line Types		
	Profile lines	Profile LP0 - LP9
	Link lines	Line Generator
Point Functions		
	Pick up profile line	FUNV 10
	indicate depth on the link line	FUNV14 / FUNV 15
	Alignment points on profiles	FUNV 11
	Pick up linked latitudes	FUNV 12
Text Types		
	Depth text	Modeller Text ass. by Geometry
	Naming longitudes and latitudes (LAT and LON)	Modeller Text ass. by Geometry
Prim Types		
	Solder prims (points)	PAP

BOOLEAN OPERATIONS

Boolean operations are used to produce new objects by the addition, subtraction, or intersection of existing solid or shell objects. These operations can be used to create a model which cannot be produced by a standard model generator.

Boolean operations can also be used to simplify drafting procedures and analysis. This is because when working with complex definitions it can sometimes be difficult to isolate the source of an error. This can be overcome by building a complex model from simple objects using Boolean operator commands, which are more easily analyzed.

- Boolean Operators 210
- Naming an Object 214
- The MAKE Command 216
- Examples of Simple Boolean Operations 220
- Changing the Order of Operation..... 226
- Multiple Boolean Operations..... 229
- Complex Boolean Operations 231
- The BOOTOL Command 235
- Boolean Operations on Shells 236
- Summary of Element Types..... 240

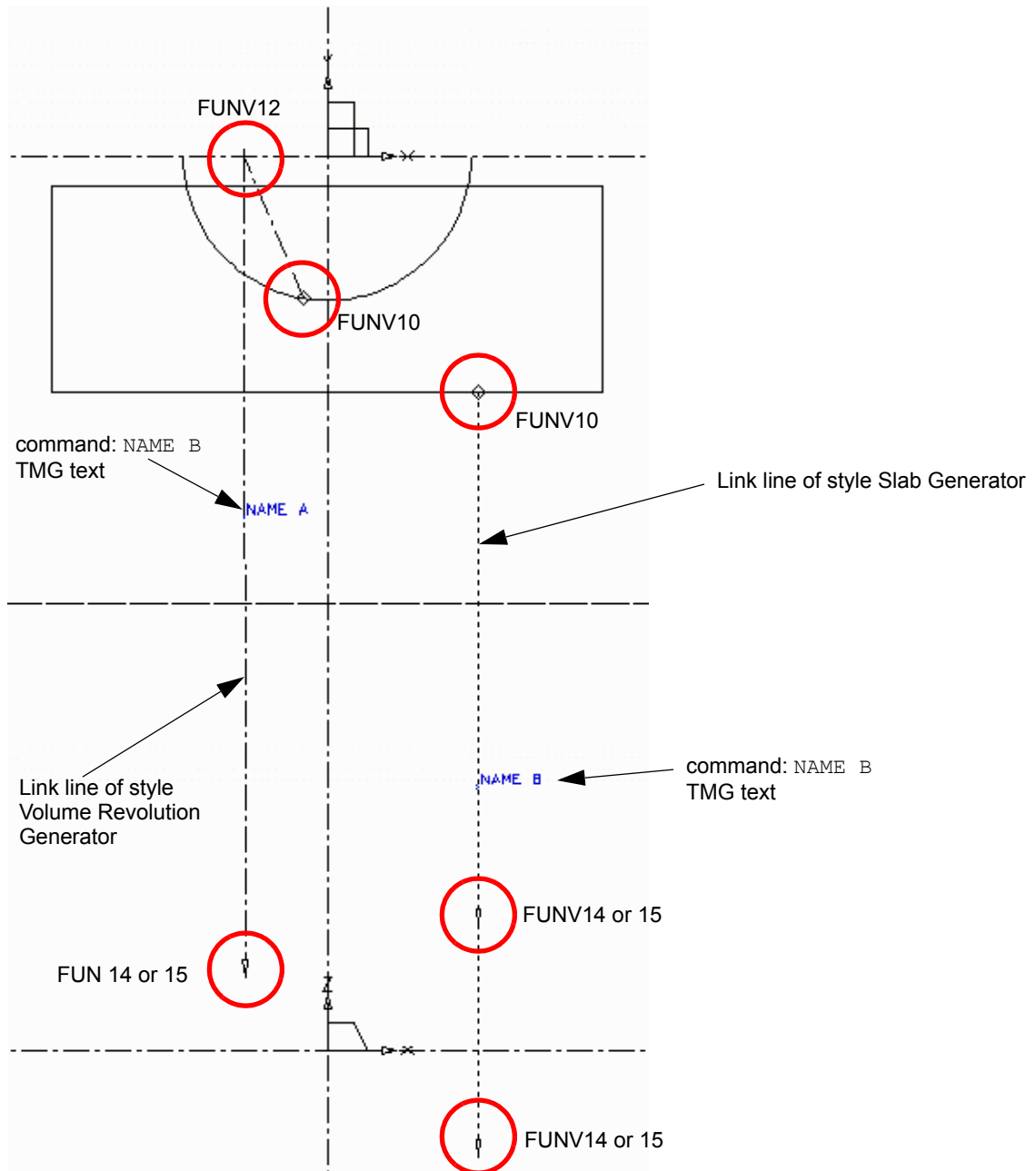
Boolean Operators

Boolean operations are invoked by Boolean operators in a `MAKE` command (see “[The MAKE Command](#)” on [page 216](#)) on the 3D sheet. The following three operators can be used:

Operator	Meaning
+	joins one object to another.
-	subtracts one object from another.
*	intersects one object with another to produce a new model from the common volume(s) of the objects.

[Figure 149](#) and [Figure 150](#) illustrate the effect of these operators. [Figure 149](#) shows two objects (a sphere and a rectangular block) defined as partially occupying the same space. [Figure 150](#) shows the result of Boolean operations on these objects.

Figure 149 Definition of Two Intersecting Objects



Please note: Objects are identified by the `NAME` command on the link lines (see “Naming a New Object” on page 217).

Figure 150 The Result of Boolean Operations on the Block and Sphere - Add

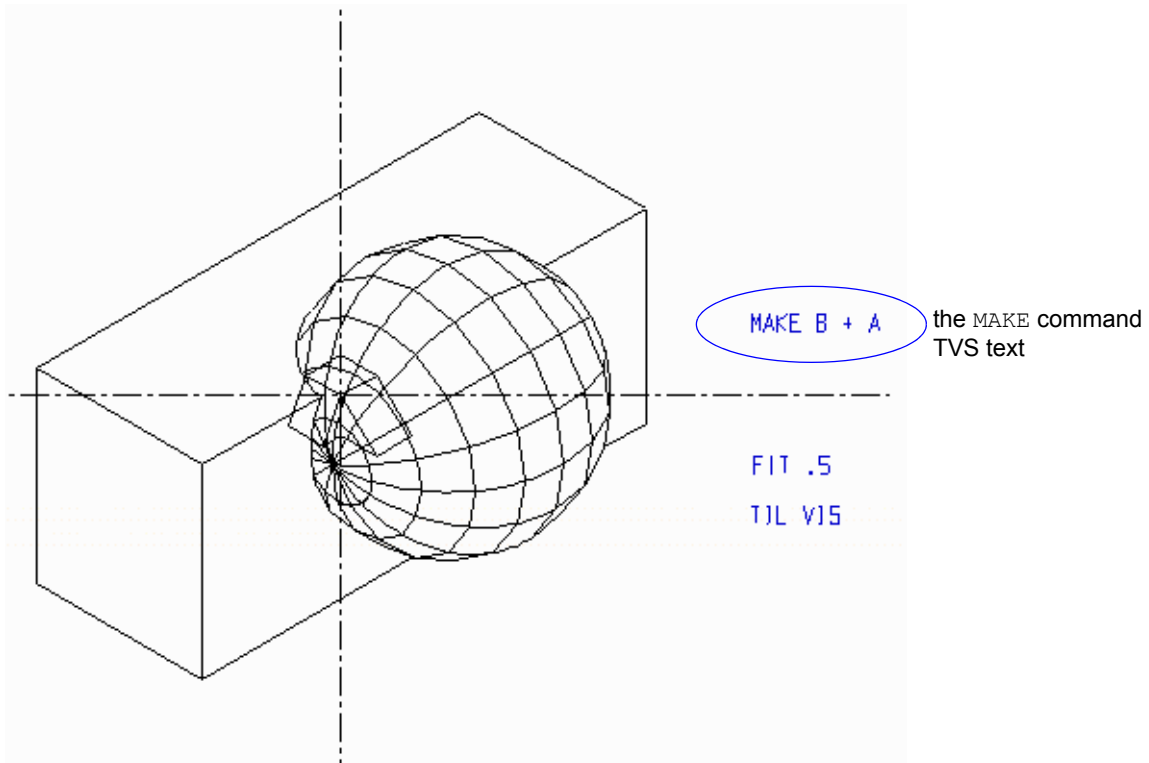


Figure 151 The Result of Boolean Operations on the Block and Sphere - Subtract

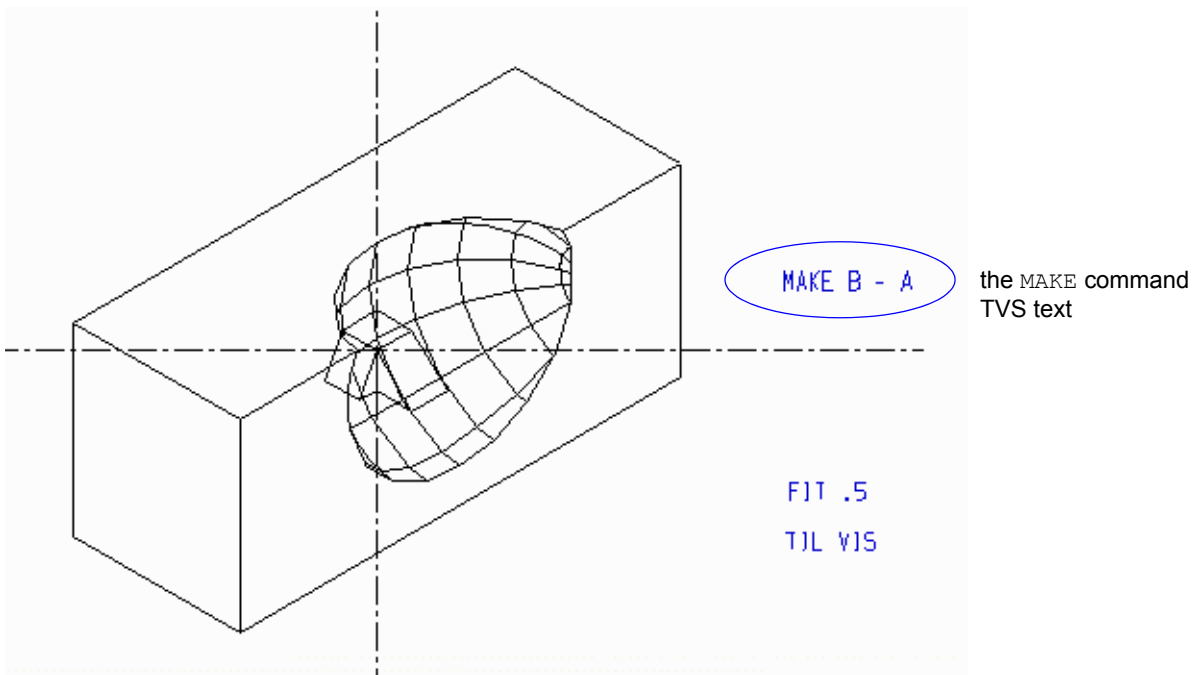
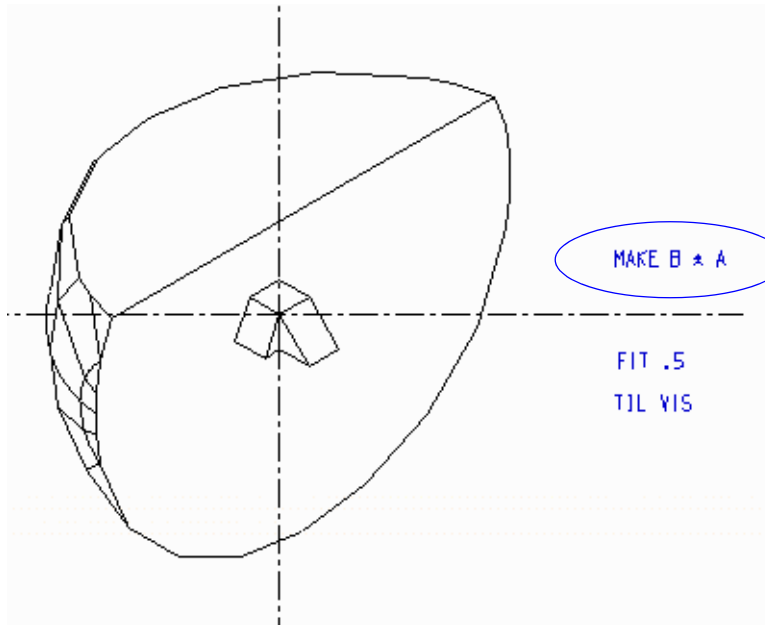


Figure 152 The Result of Boolean Operations on the Block and Sphere - Intersect



the MAKE command
TVS Text

Naming an Object

Each object that you want to include in a Boolean operation must be given a name. A name is specified by attaching a TMG-text of style `Modeller Text ass. by Geometry` to the link line of an object definition. The name can consist of either the command text `NAME` (alternatively `&`) followed by the object name. For example, an object can be given the name `BLOCK` either as `NAME BLOCK` or as `&BLOCK`.

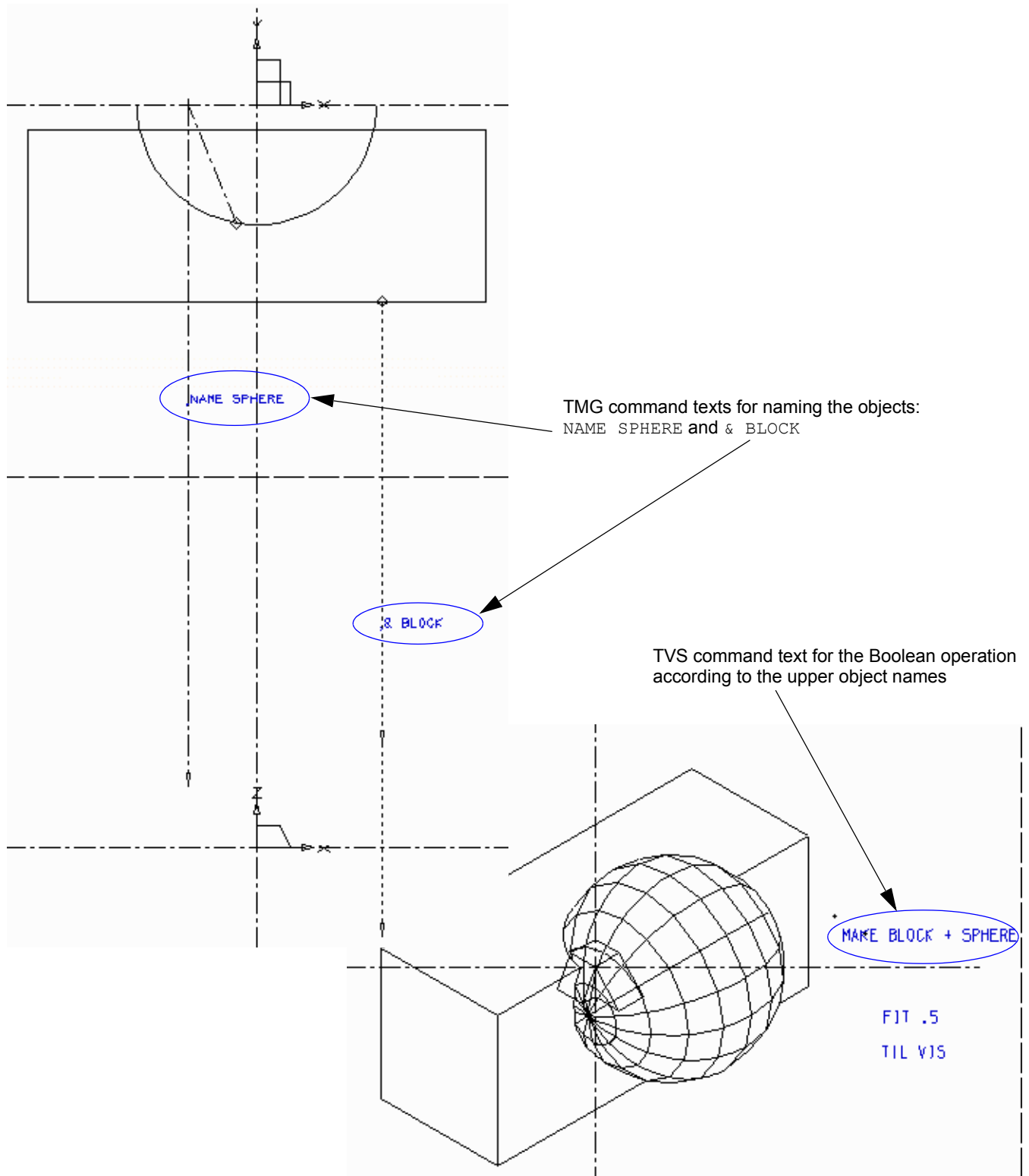
Enter the desired name of the object as value behind the `NAME` command.

The following points apply to the naming of objects:

- The name can have a maximum of 20 characters
- Both letters and numbers can be used in the name in any combination
- The `NAME` and `&` commands are not case sensitive

Figure 153 shows the two methods of naming an object.

Figure 153 Name Text On Link Lines



The MAKE Command

A Boolean operation is invoked by a `MAKE` command. This command defines the relationship between the named objects on the sheet. For example, to remove a volume named `SLOT` from an object called `BLOCK`, the following `MAKE` command could be used:

```
MAKE BLOCK - SLOT
```

The `MAKE` command is specified as TMS-text of style `Model Specification Text` and it can be placed anywhere on the 3D sheet within the sheet boundaries.

The following constraints apply to the `MAKE` command:

- There must be a space between words in the command and also between a word and a Boolean operator.
- Each command must not exceed 120 characters in length.
- The command is not case sensitive.
- Each command is executed from left to right unless parentheses are included to change the order of operation (see [“Changing the Order of Operation” on page 226](#)).

Naming a New Object

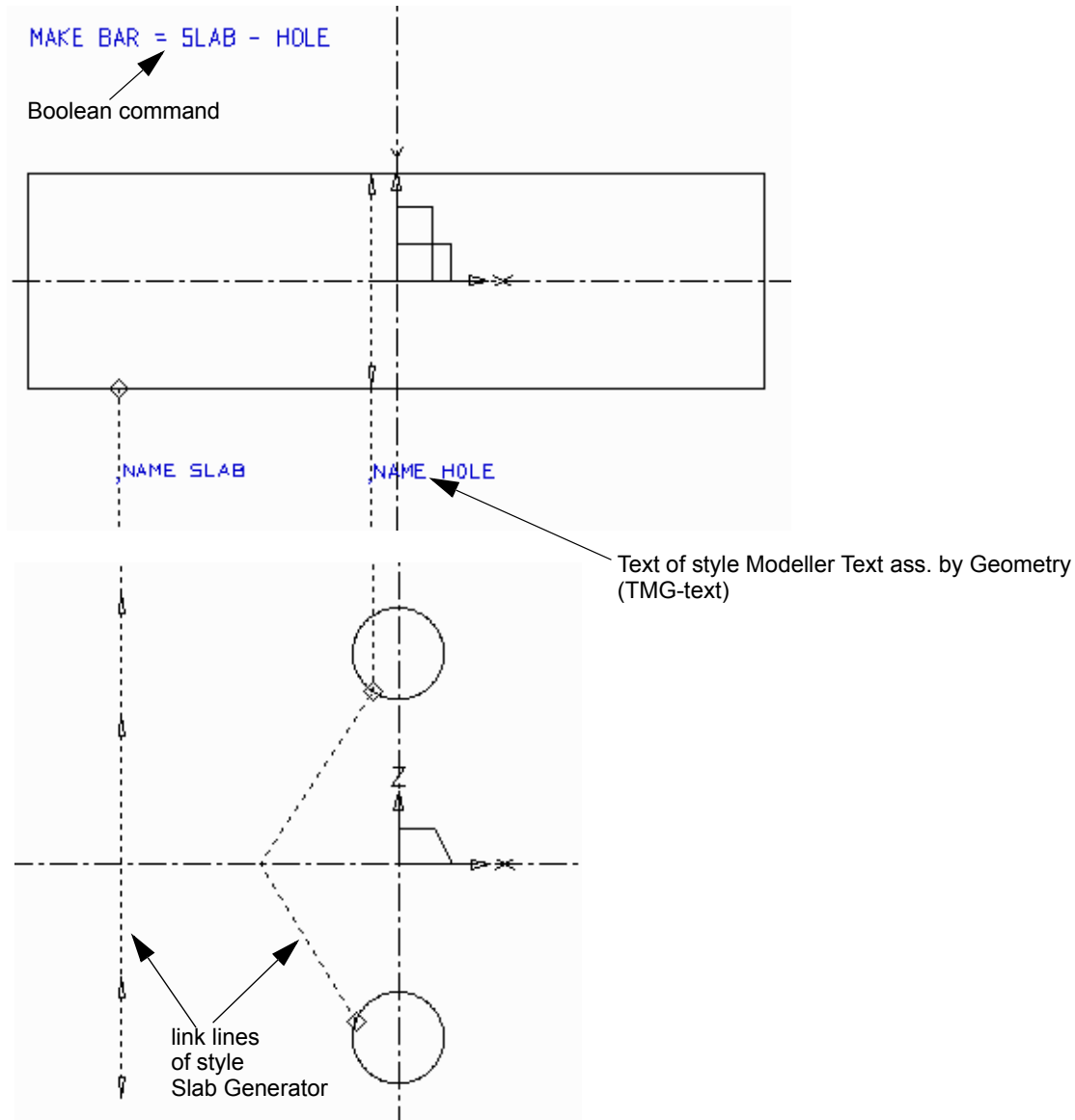
The object that is generated by the Boolean operation can also be given a name. This name can be useful in future operations. The name of the new object is defined by the Boolean command. For example, the object created by removing the volume `SLOT` from `BLOCK` can be given the name `BEARING` using the command:

```
MAKE BEARING = BLOCK - SLOT
```

If a name is included in the `MAKE` command it must be followed by an equals sign (=). Naming a new object can be used to avoid Boolean commands that would otherwise exceed the permitted 120 characters in length. A long command can be divided into a number of shorter commands using this method (see [“Multiple Boolean Operations” on page 229](#)).

If an object consisting of more than one part is generated, the name specified applies to all the parts. For example, [Figure 154](#) shows a definition for a two-part object generated as a multiple sweep.

Figure 154 A Two-Part Model Definition With Boolean Subtraction



The Boolean subtraction operator is used to cut a hole through each part of the object. The Boolean command used to perform this operation is:

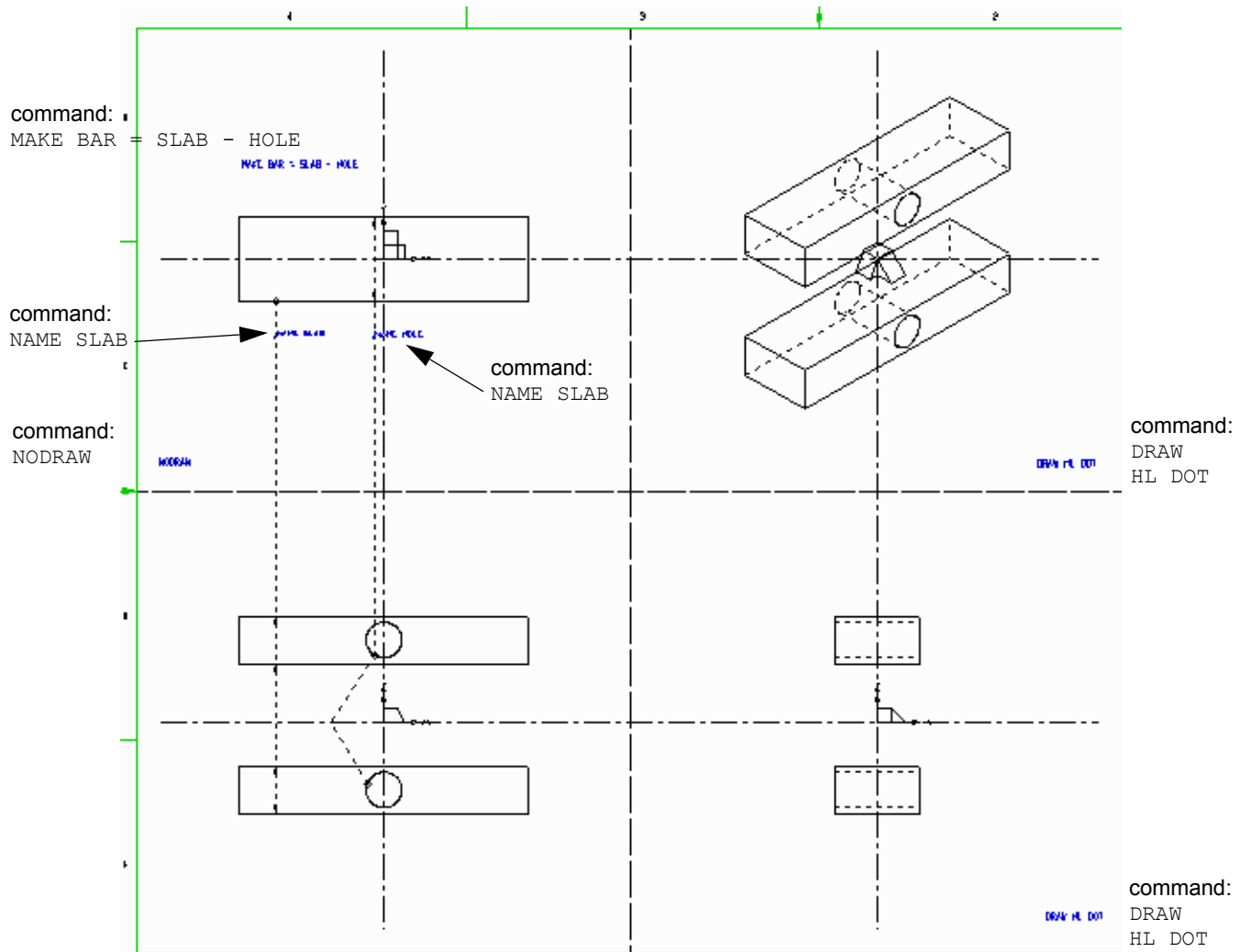
```
MAKE BAR = SLAB - HOLE
```

where `BAR` is the name of the new (two-part) object.

Please note: The Modeler treats the two parts as a single object called `BAR`, not as separate entities.

Figure 155 shows the completed model.

Figure 155 The Completed Model



Examples of Simple Boolean Operations

This section gives examples of each of the three types of Boolean operation:

- Addition
- Subtraction
- Intersection

Boolean Addition

Figure 156 shows a model of a collar created by the Boolean addition of three objects (named A, B, and C).

Figure 156 A Collar Modeled Using Boolean Addition

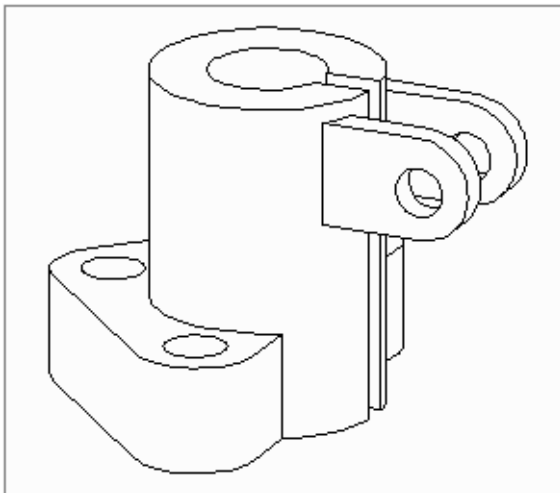
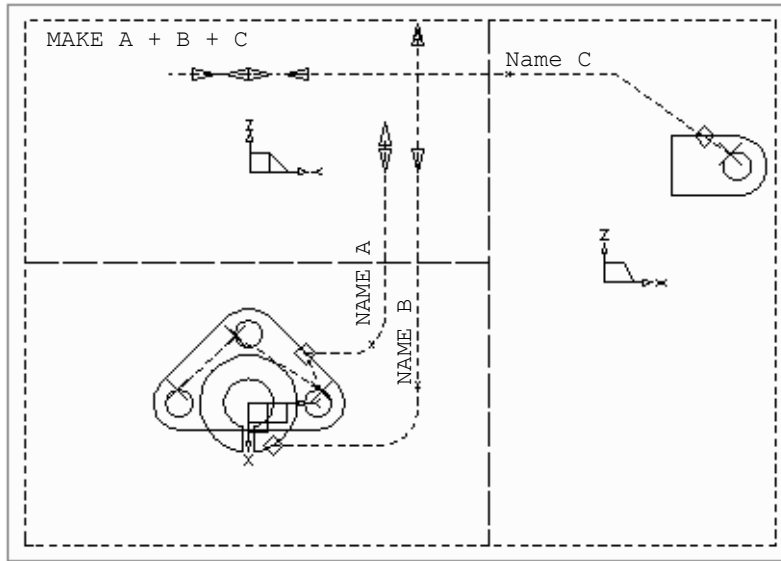


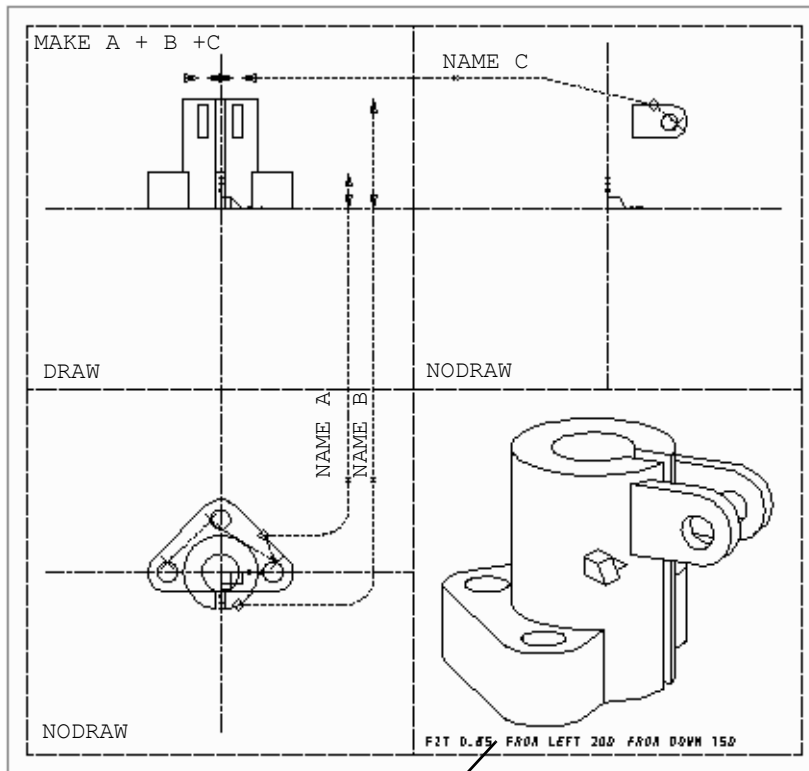
Figure 157 shows the model definition. Each object used to create the final model is generated as a solid sweep and named by attaching a TMG-text of style `Modeller Text ass. by Geometry` to the link line.

Figure 157 The Model Definition



The Boolean command: `MAKE A + B + C` which creates the final model is specified as TMS-text of style Model Specification Text. Figure 158 shows the complete definition and model.

Figure 158 The Completed Model

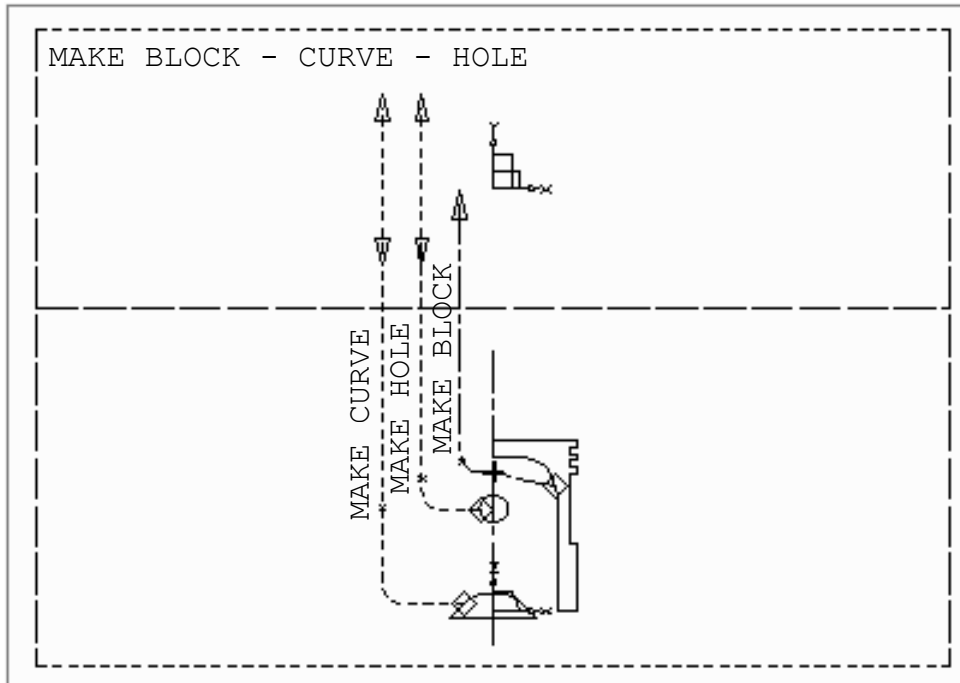


FIT 0.85 FROM LEFT 200 FROM DOWN 150

Boolean Subtraction

Figure 159 shows the definition of a model of a piston using Boolean subtraction. The model is formed by subtracting the two solid sweeps named `CURVE` and `HOLE` from the volume of revolution called `BLOCK`.

Figure 159 A Piston Modeled Using Boolean Subtraction



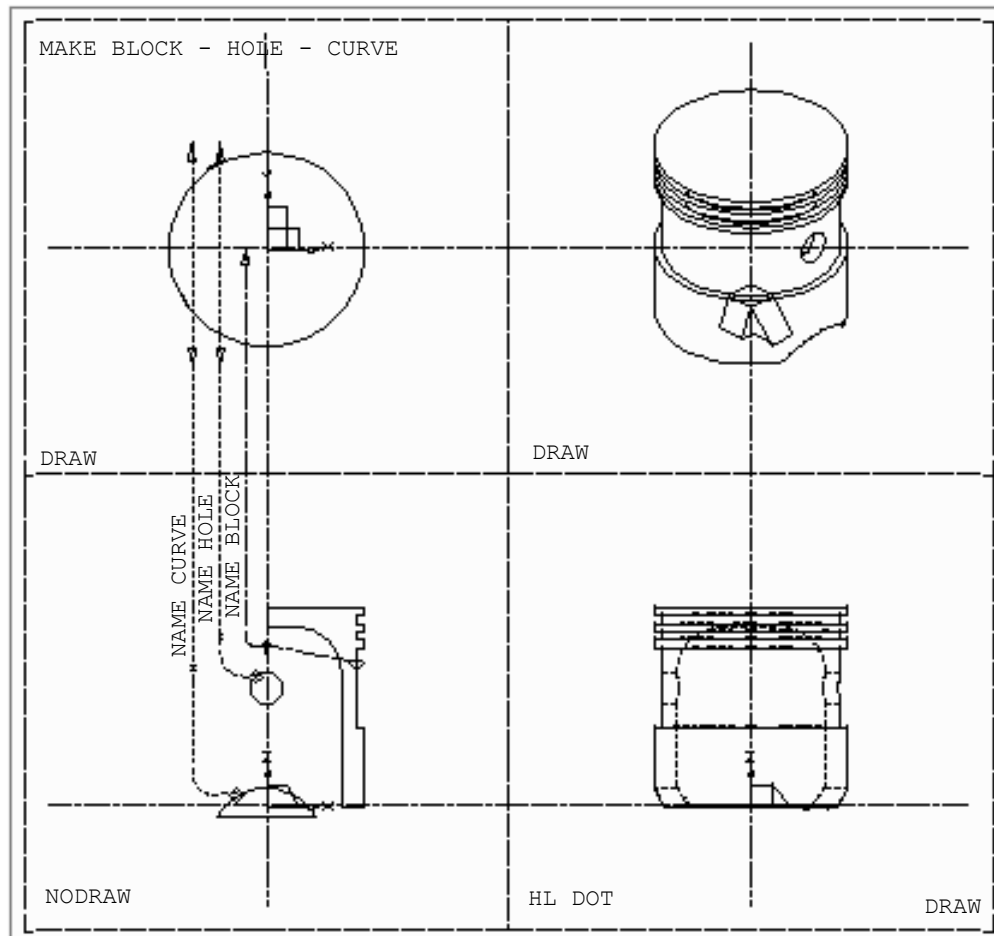
The Boolean command:

```
MAKE BLOCK - CURVE - HOLE
```

creates the final model.

Figure 160 shows the completed definition and model.

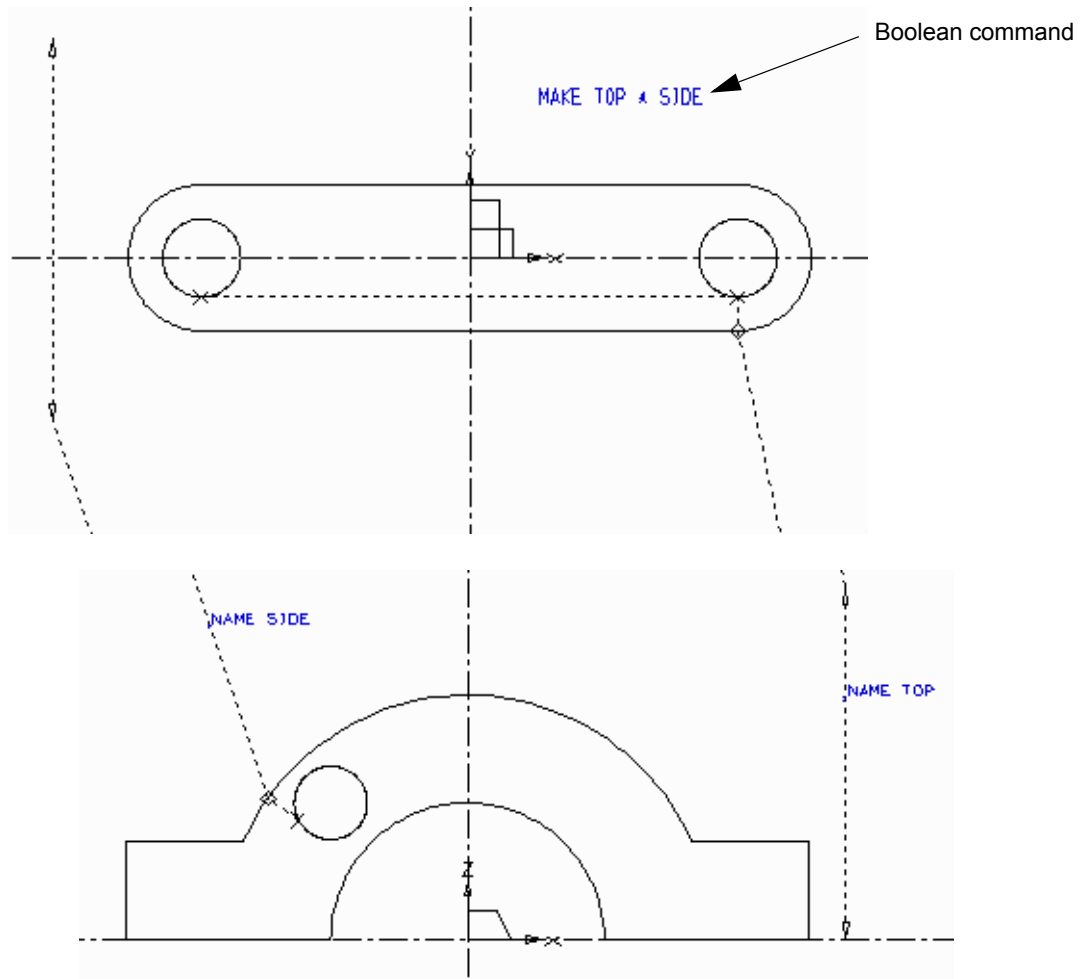
Figure 160 The Completed Model



Boolean Intersection

Figure 161 shows the definition of a model of a bearing endcap using Boolean intersection. The model is formed from the intersection of two solid sweeps named `SIDE` and `TOP`.

Figure 161 A Bearing Endcap Modeled Using Boolean Intersection



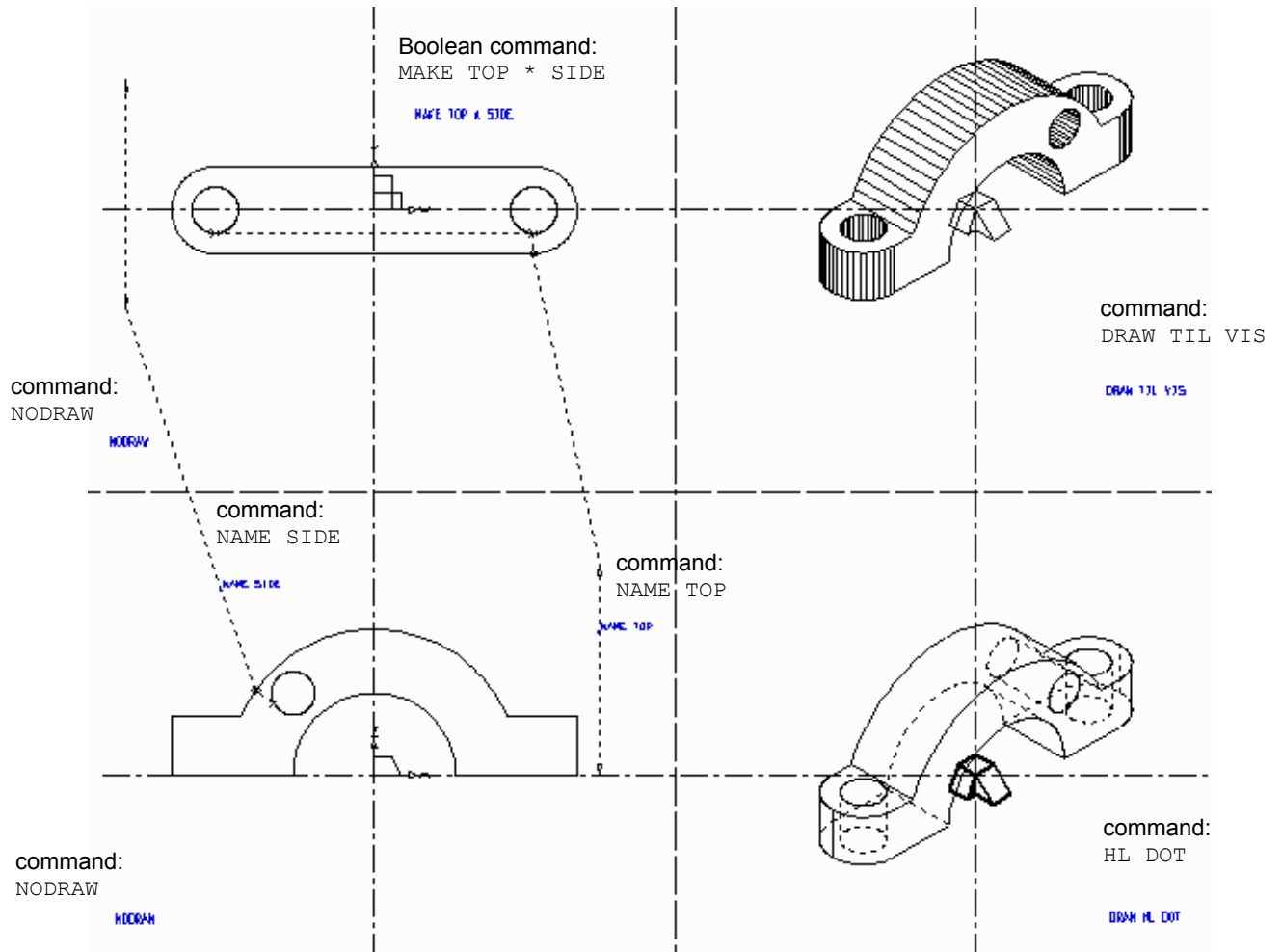
The Boolean command:

```
MAKE TOP * SIDE
```

creates the final model.

Figure 162 shows the completed definition and model.

Figure 162 The Completed Model



Changing the Order of Operation

The order in which the Modeler performs Boolean operations is from left to right unless brackets are used. As in the case of the operators, brackets must be preceded and followed by a space character.

For example, the command:

```
MAKE A = B + C - D * E
```

creates the model A by first adding B and C, then subtracting D, and finally intersecting the result with E.

If brackets are included in the above command, the order of operation is changed. For example, the command:

```
MAKE A = B + ( C - D ) * E
```

creates a model A by first subtracting D from C, then adding B, and finally intersecting the result with E.

[Figure 163](#) shows the definition of the model. For both models it is the same definition, except for the boolean command.

[Figure 164](#) and [Figure 165](#) show the effect of using brackets in Boolean operations.

Figure 163 The Definition of the Model

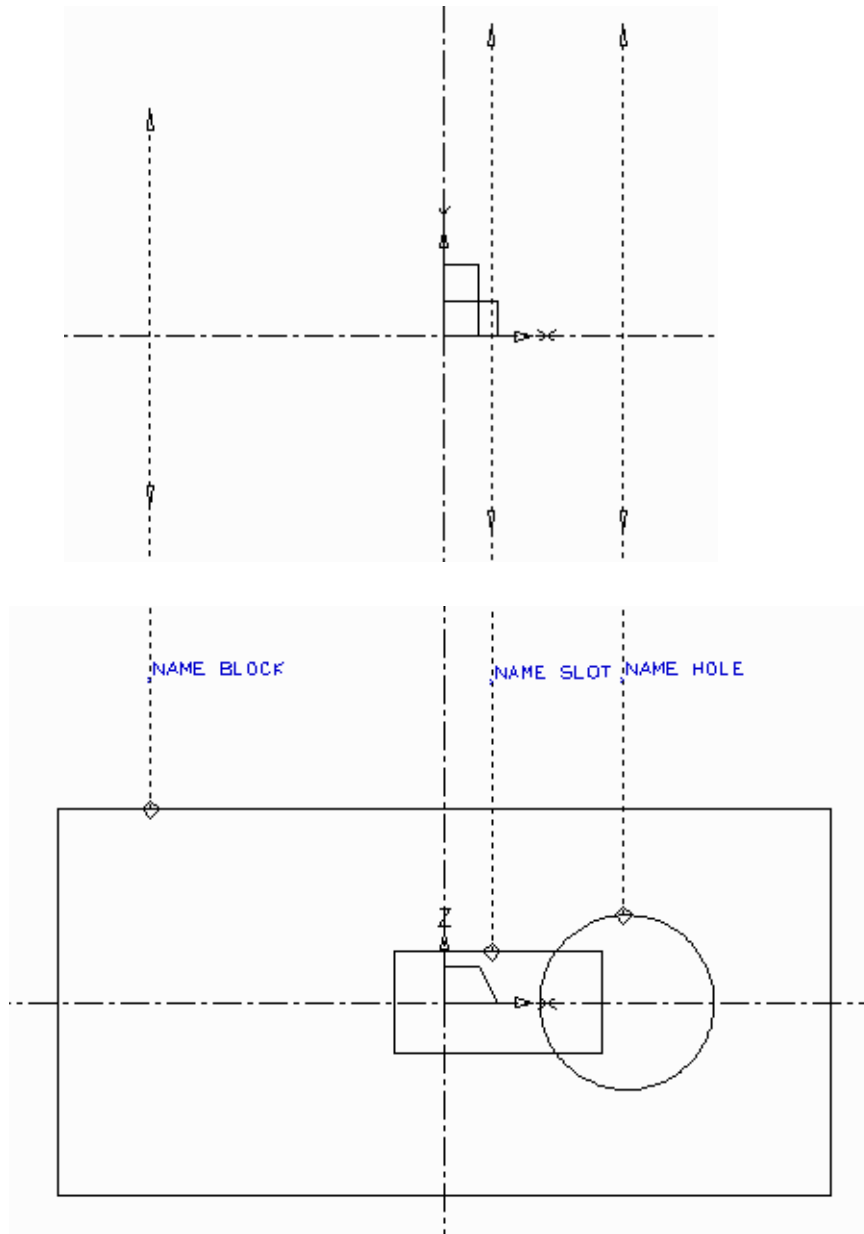


Figure 163 shows the definition of the model created by the command:

```
MAKE BLOCK - HOLE + SLOT
```

Figure 164 The Model Created by the Command Without Brackets

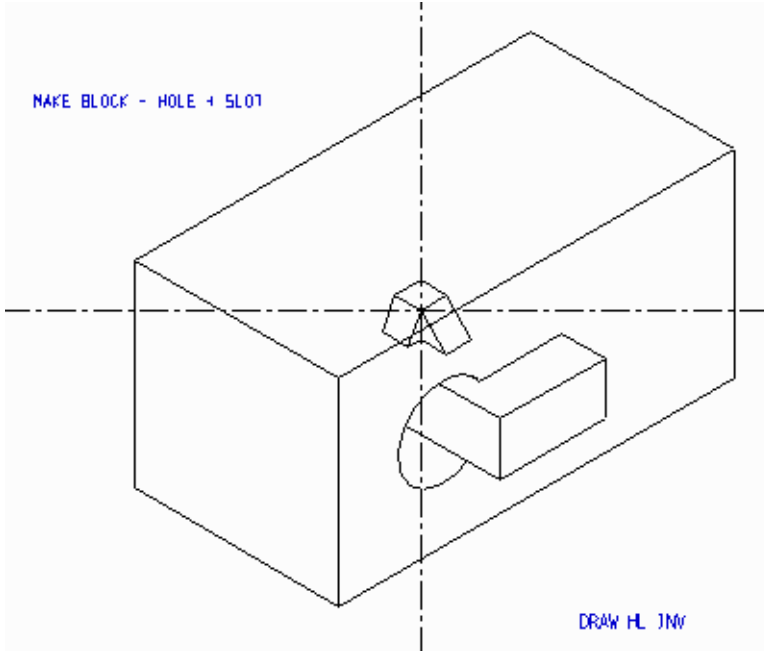
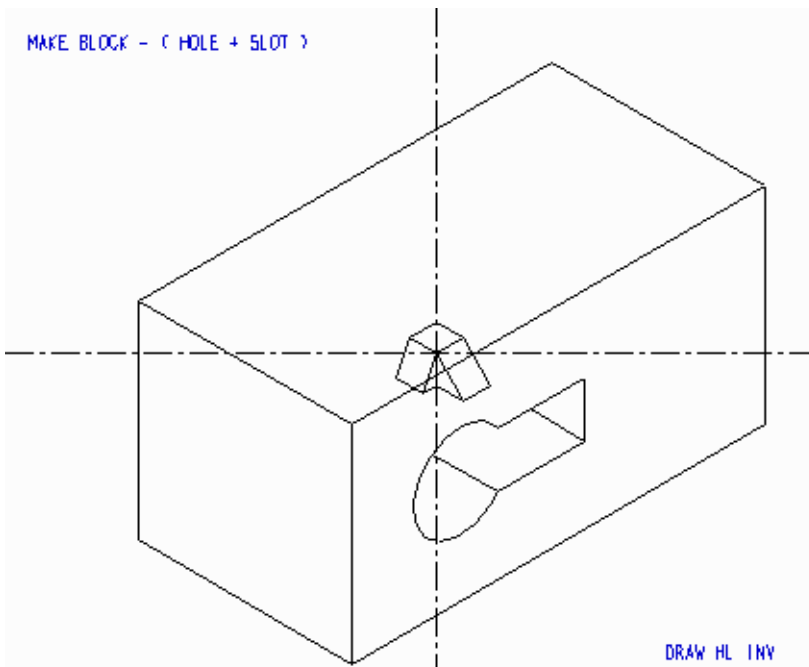


Figure 165 shows the model created by the command:

```
MAKE BLOCK - ( HOLE + SLOT )
```

Figure 165 The Effect of the Same Boolean Command With Brackets Added

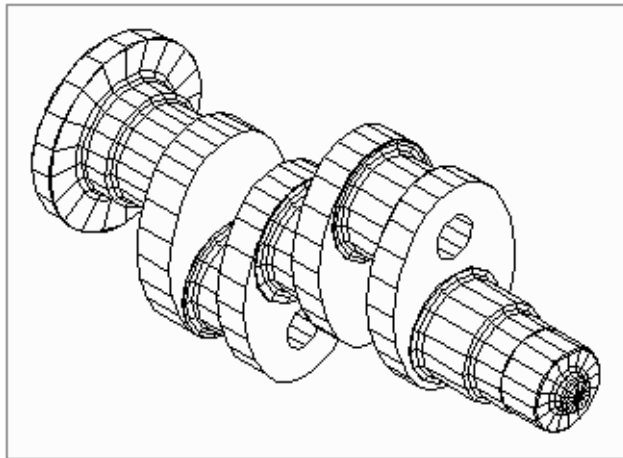


Multiple Boolean Operations

This section shows how multiple Boolean operations are used to create new objects which are then combined to create the final model.

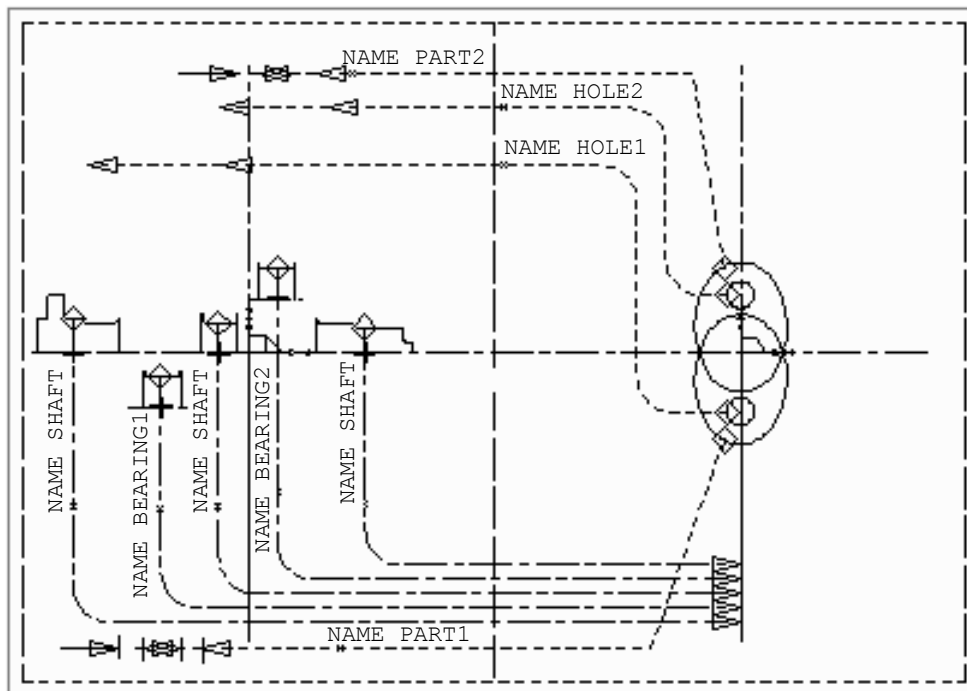
Figure 166 shows a crankshaft modeled in this way. Figure 167 shows the definition sheet.

Figure 166 Model of a Crankshaft



The objects labeled PART1, PART2, HOLE1, and HOLE2 are generated as solid sweeps. The others are generated as volumes of revolution.

Figure 167 The Crankshaft Model Definition



Naming a new object is essential if you are performing multiple Boolean operations on multiple objects. If several uniquely named objects are created by Boolean operations they can be combined again to create the final model.

In this example, new objects are created using the following Boolean commands:

```
MAKE CRANK1 = BEARING1 + PART1 - HOLE1
```

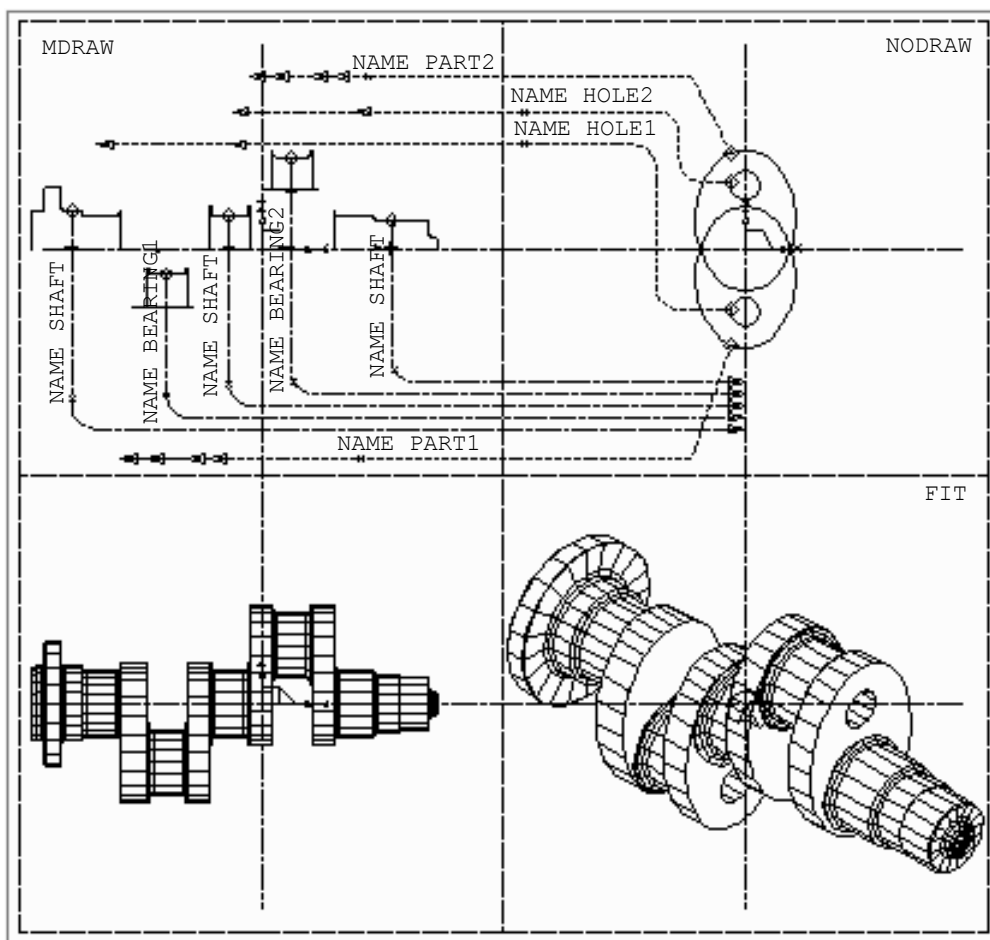
```
MAKE CRANK2 = BEARING2 + PART2 - HOLE2
```

The final model is created from the command:

```
MAKE CRANK1 + CRANK2 + SHAFT
```

Figure 168 shows the definition and completed model.

Figure 168 The Complete Definition and Model



Complex Boolean Operations

The following section looks in detail at the MEDUSA4 Boolean process:

- It examines numerical aspects of the process.
- It identifies the few cases where numerical accuracy may cause problems and gives some advice on how to avoid these problems.
- Finally, it describes a facility which may be considered as a last resort to improve matters.

The Boolean Process

The two operands of a Boolean operation are examined, to determine which parts of the original objects should form the resultant object.

Objects whose boundaries intersect are examined more closely at facet level and facets which intersect are compared carefully with each other:

- Points of intersection are found and are used to form the start and end points of intersection edges.
- The intersection edges, along with appropriate parts of the original facets are used to generate facets of the resultant object. This object has to be sewn together to form a complete boundary representation. In a solid, each facet edge is shared by another facet.

Tolerance

In all but the simplest cases of intersection between two facets, a decision about whether the geometry is coincident must be made. For example:

- Do these points share the same space?
- Are these edges co-linear?
- Do these points lie on a plane?
- Are these faces co-planar?

To represent coincidence, a tolerance is used. The tolerance is based upon the precision with which the machine can store numbers, and the size of the objects being intersected. Its value, in engineering terms, is insignificant.

Numerical Accuracy

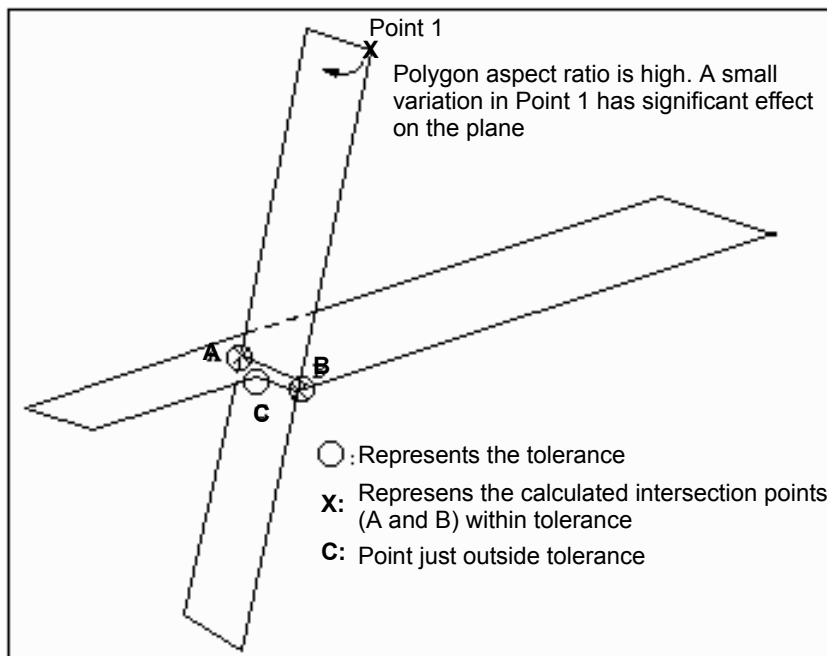
Original MEDUSA4 models represent the 2D profiles used for generation, to the precision of the machine. The mathematics involved in transforming those profiles into 3D space causes a very slight deviation. Any transformation, for example, like those of instancing, has a mild effect on the relationship between facets. Boolean operations are made upon the faceted representation, whereas unbooled facets may be considered to represent the true surface (that is coordinates of the facet lie on the original surface). By implication, intersected facets no longer represent the original surface.

New points, resulting from Boolean operations, are calculated to a value within the tolerance. These new points, therefore, do not lie exactly on the plane. An increasing number of new points generates a more and more ambiguous facet and repeated Boolean operations causes the geometry to deteriorate.

The issue is complicated further by aspect ratio. Two intersecting facets may give ambiguous results as their aspect ratios become very large. Because tolerance is very small in engineering terms, the effect on geometry is insignificant and usually unnoticeable. However, in numerical terms, enquiries upon such geometry may give ambiguous results.

In [Figure 169](#) illustrated below, points A and B are within the tolerance, when point C is just outside the tolerance.

Figure 169 Example of Points Within or Outside Tolerance



What Is Valid?

Any problem encountered by the modeler during a Boolean operation is reported in the form of a Modeler error message (see [“Error Messages”](#) on page 521). Where the resultant object is a

solid, you can use the Model Validator to determine whether or not the modeler has generated a complete manifold boundary representation, that is, each edge is shared by two facets. For details of how to use the Model Validator, see chapter [“The Model Validator” on page 361](#).

There are circumstances when a difficulty reported in either of these two ways does not pose a problem. For example, any problem area local to a few facets may well be discarded in a subsequent Boolean operation. An invalid model which is rendered, for example, by the reconstructor, may still be drawn satisfactorily.

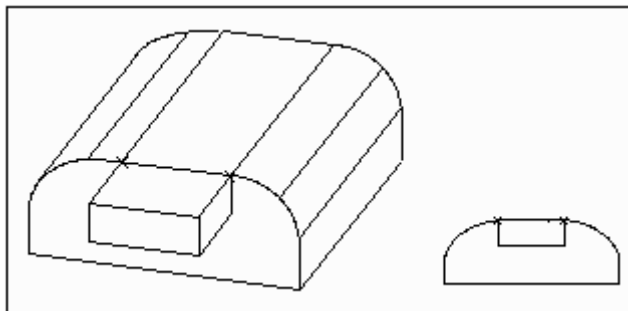
Object Definition

There are a number of steps that can be taken to minimize the numerical problems associated with geometric modeling. Adherence to such guidelines will generally promote the generation of accurate, predictable models:

1. Use `Near` probes to pick up existing profiles when creating subsequent profiles.
2. Superimpose the link line depth points of the separate link lines on one another by using `Near` probes.
3. Where two curved surfaces are to be coincident, use the same construction method for both curves and use tangent point arcs if possible. This gives predictable results as the Modeler always converts curves to tangent point arcs.

The number of facets on an object's surface may be closely managed by the use of the `CHOTOL` command as a text of style `Modeller Text ass. by Geometry` (type `TMG`) on the link line. The alignment of facets of abutting objects can be managed by splitting each of the intersecting curves at the same point, and setting the second `CHOTOL` argument to the number of facets required. See [Figure 170](#) shown below.

Figure 170 Alignment of Facets of Abutting Objects



You can generate doubly curved surfaces using, for example, the solid of revolution, meshed surface, or slide generators. The location of the facets of a doubly curved surface is an arbitrary representation of the true surface. It is their vertices which lie on the surface. You should therefore avoid aligning the plane of such facets with the geometry of other objects.

4. Avoid co-planar alignment of facets with high aspect ratios since they are numerically unstable.

Order of Operations

You should follow an order of operations which deals with:

1. Coincident faces
2. Coincident geometry
3. General operations

Since geometric stability is affected by preceding boolean operations, the earlier coincident faces are dealt with, the less likely numeric ambiguity may occur.

Isolating the Faulty Operation

Use the Model Validator, described in chapter [“The Model Validator” on page 361](#), to check the model resulting from a Boolean operation. Remodel the objects to make sure they were valid prior to the Boolean operation. If the Booleans are asked to intersect an invalid facet, problems may arise.

The BOOTOL Command

The `BOOTOL` command is designed for use with Boolean operations when two faces of the input objects are almost coincident, that is, they are so close that the Modeler cannot resolve them into a single face on the resulting model. This is usually the result of a twisted tile. You may then use the `BOOTOL` command to specify a larger coincidence tolerance which allows the Modeler to accept such faces as truly coincident.

Only use `BOOTOL` if you have checked the model, as discussed in section “[Isolating the Faulty Operation](#)” on page 234, but have still been unsuccessful in creating a valid model.

The tolerance value for Boolean operations represents the range within which two points (or two lines, faces, or edges) are considered coincident. There is a default value provided by the system which is related to model size.

This default tolerance can be changed using the TMS text:

```
BOOTOL value
```

where value is in object units.

Selecting a different value of `BOOTOL` may help resolve ambiguity by moving the values wholly inside or wholly outside the tolerance.

Always try and use the smallest possible value, as large values will adversely affect model accuracy. If you use `BOOTOL` in multiple Boolean operations, a tolerance buildup can occur which may seriously affect model accuracy, and actually produce the twisted tiles which can cause problems in subsequent operations.

Note that if value is negative it is treated as a multiplier of the default tolerance value. This is the recommended way to use this command. For example, as a first try, the command can be given as:

```
BOOTOL -2
```

If this is unsuccessful, you should next consider the reduction of `BOOTOL`, for example:

```
BOOTOL -.5
```

You could then try a larger tolerance.

It is reasonable to increase the multiplying value by a factor of two each time you specify a larger tolerance. For example, when you have tried `BOOTOL -2`, you can then try:

```
BOOTOL -4
```

`BOOTOL` should be restricted to the isolated faulty operation and not to the whole sheet.

This is demonstrated in the following example:

A problem in the sequence $A + B + C + D$ is isolated to the operation $A + B$. In this case, $A + B$ should be modeled independently with an appropriate `BOOTOL`. The resultant model should be instanced on to a second sheet without `BOOTOL`, to generate $INSTANCE + C + D$.

Boolean Operations on Shells

Boolean operations can be performed on shell type objects. However, not every Boolean operation on a shell produces a meaningful model. With some combinations the Modeler is unable to determine how the finished model should be represented, and unexpected results may occur.

The table below shows the possible operations that can be performed on solids and shells. Only the operations shown in uppercase can produce a valid model.

SOLID + SOLID	shell + shell	solid + shell	shell + solid
SOLID * SOLID	shell * shell	SOLID * SHELL	SHELL * SOLID
SOLID - SOLID	shell - shell	solid - shell	SHELL - SOLID

For example, “[Definition of a Shell Model Using Boolean Operations](#)” on page 237 shows the definition of a block created as a shell with a hole passing through it. The hole has been defined as a solid sweep and the Boolean command:

```
MAKE BLOCK - HOLE
```

creates the final Boolean shell type object. [Figure 171](#) shows the definition of the model.

Figure 171 Definition of a Shell Model Using Boolean Operations

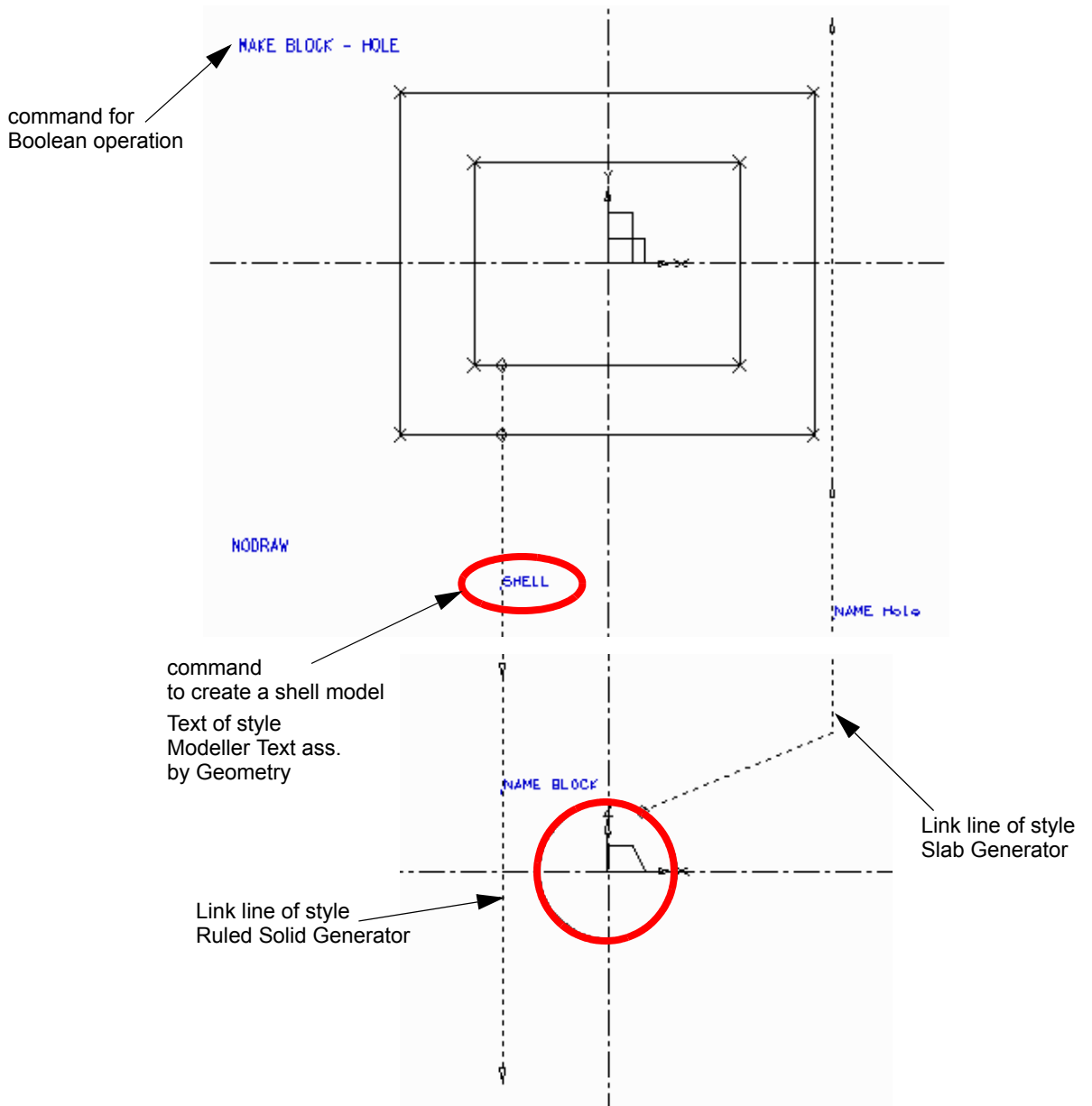
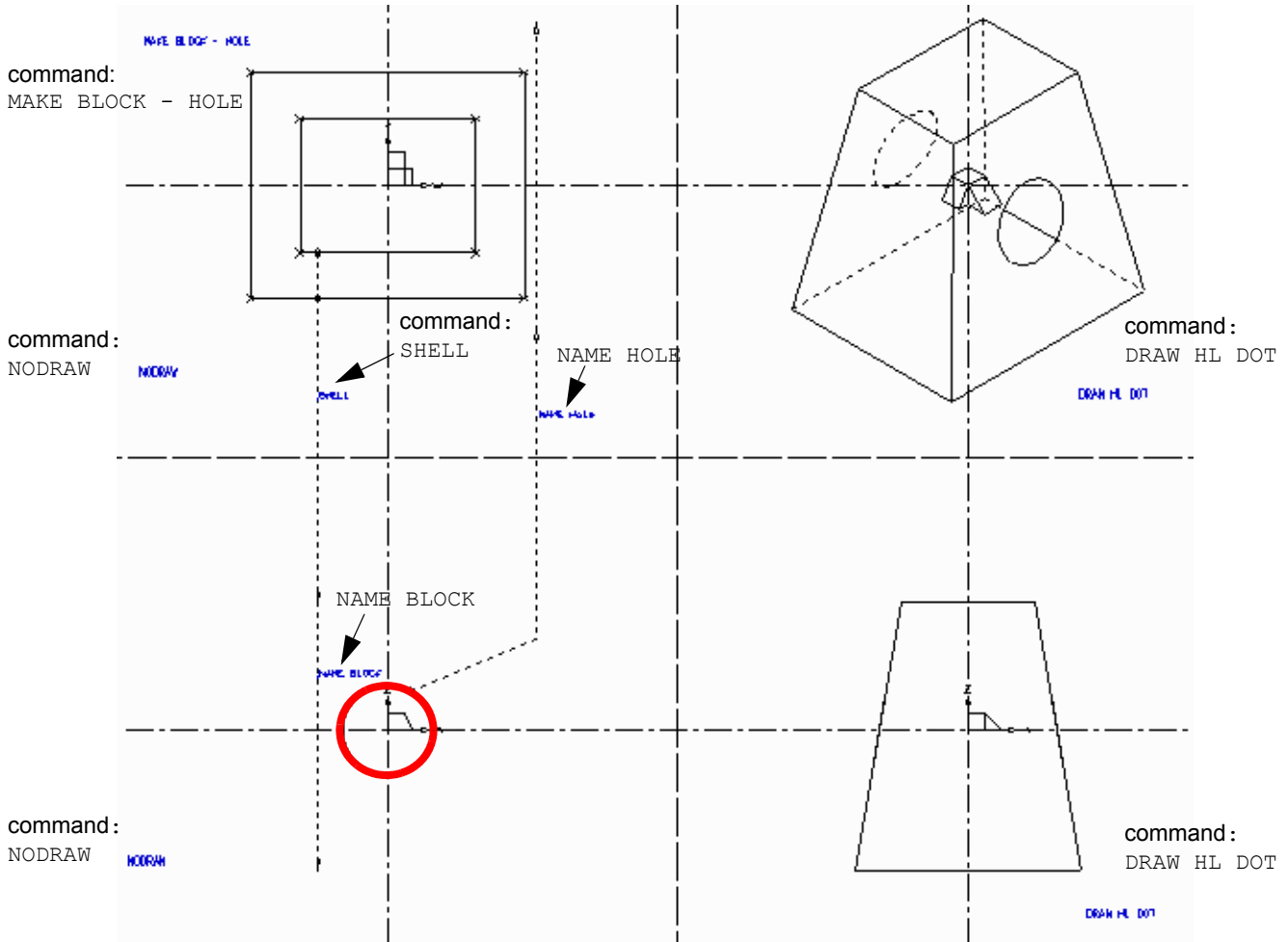


Figure 172 shows the model created from the definition in Figure 171.

Figure 172 The Completed Shell Model



The MSHELL Command

Shell models can also be generated using the `MSHELL` command. This command is used in the same way as the `MAKE` command. The `MSHELL` command is also specified using a TMS-text of style `Model Specification Text`.

For example, to create a shell model from the addition of two solid objects called `BAR` and `PLATE`, the following Boolean command is used:

```
MSHELL BAR + PLATE
```

The new model can also be named using the `MSHELL` command. As an example, the new model specified by the previous command could be called `ASSEMBLY` using the command:

```
MSHELL ASSEMBLY = BAR + PLATE
```

The `MSHELL` command can be added to the sheet in the area provided (see [“The MAKE Command” on page 216](#)).

Summary of Element Types

The following table gives a summary of the text types used in the definition of Boolean operations.

Element Description	Element Style
Text Type	
MAKE command	Model Specification Text (TMS)
MSHELL command	Model Specification Text (TMS)
BOOTOL command	Model Specification Text (TMS)
NAME and & command	Modeller Text ass. by Geometry (TMG)

ASSEMBLIES AND EXPLODED VIEWS

This chapter explains how you can create assemblies and exploded views.

- Overview 242
- Instance Groups..... 244
- Positioning and Scaling the Instanced Model 247
- An Example of an Instanced Model 251
- Naming Objects in Instanced Models 256
- Creating an Assembly View With the Help of AVIEW 263
- Exploded Views 271
- Summary of Element Types..... 272

Overview

Assembly models and exploded views can be created using instance groups placed on a 3D sheet. The Modeler recognizes an instance group as representing a model contained in a model file, which means that a component modeled only once can be used repeatedly when creating other models or assemblies.

An assembly is created by instancing separately modeled components on the same sheet, with the correct orientation, so as to form a complete assembly.

A very useful time saving feature in assembly modeling is provided by the `AVIEW` command. This command allows you to create views of an assembly built from instanced objects before the assembly sheet is sent to the Modeler.

An exploded view is created by instancing separately modeled components on the same sheet but with the models displaced axially relative to each other.

For example, [Figure 173](#) shows a model of a turbine blade, and [Figure 174](#) shows the assembled model of a turbine created from instances of the turbine blade model.

Figure 173 Model of a Turbine Blade

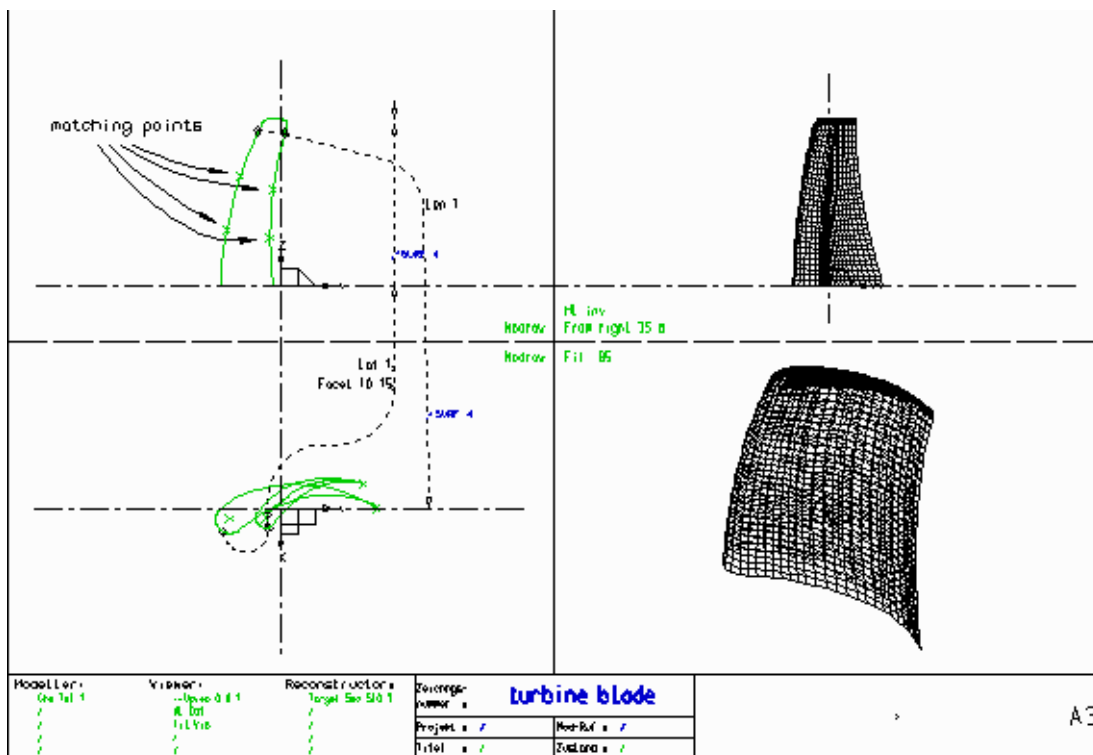
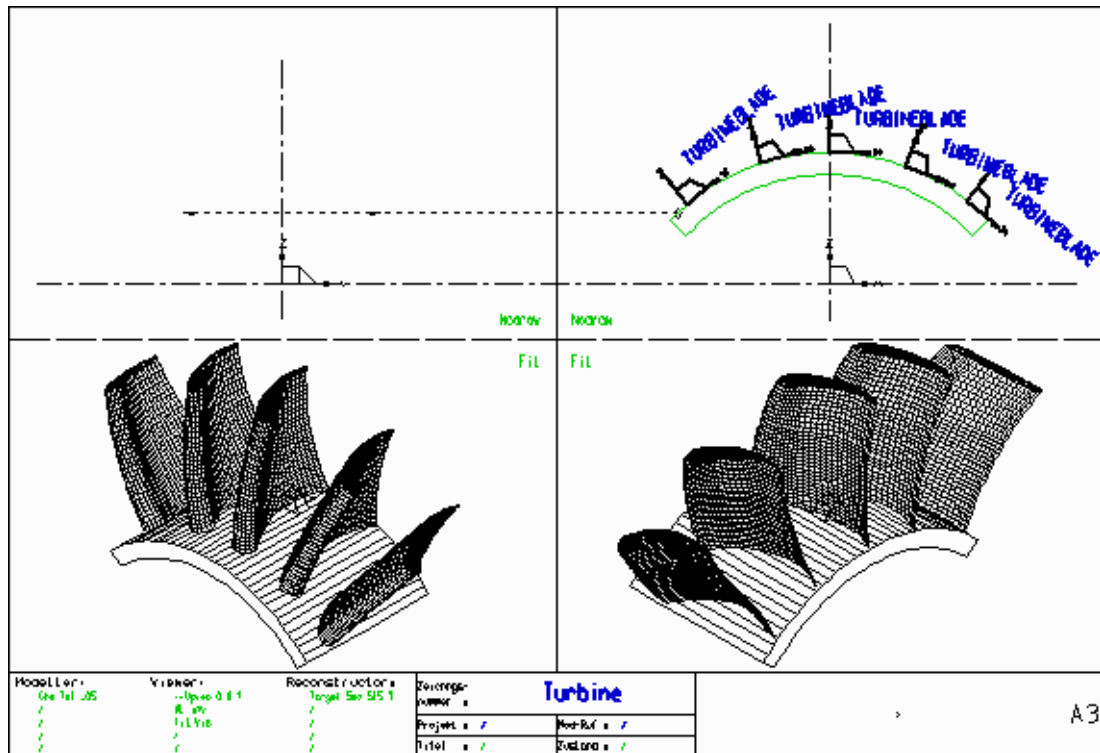


Figure 174 The Assembled Turbine



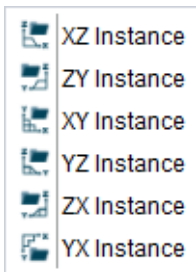
Please note: A sheet can contain any combination of ordinary model definitions and instance groups.

Instance Groups

An instance group is actually a group of type 3D Instance Datum which contains one view-prim (of style 3D Instance XZ Elevation, 3D Instance YZ Elevation and so on) and two text strings (style 3D Instance Project Name and 3D Instance Model Name, type SMO and SPR). It is loaded into a viewbox on a 3D sheet which is to be used as an assembly drawing.

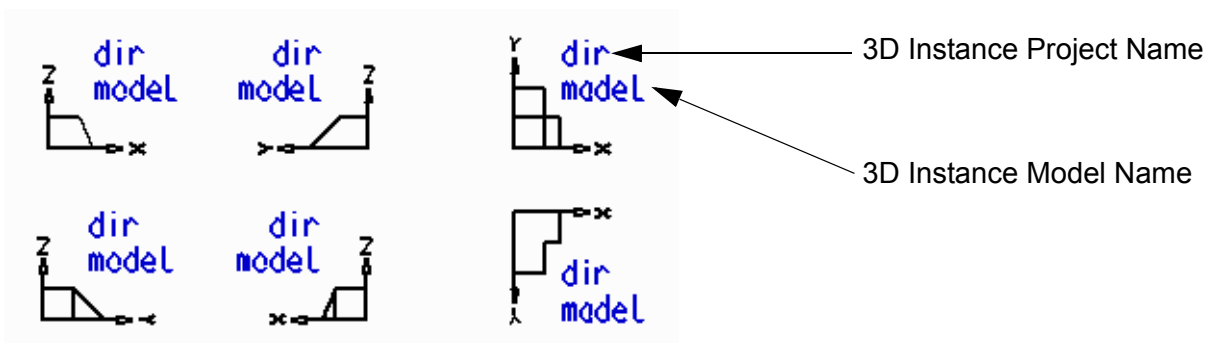
You can select instance groups from the toolset shown in [Figure 175](#).

Figure 175 Instance Groups Toolset



[Figure 176](#) shows the standard set of instance groups.

Figure 176 Standard Instance Groups



1. To load an instance group into a viewbox click on one of the tools shown in [Figure 175](#).
The Instance group dialog appears.
2. Select the required model file by choosing the directory where the file is stored and load the required file by:
 - clicking twice on the file name or
 - clicking once on the file name and then pressing the Open button.The instance group is attached to the cursor.
3. Probe in the sheet to place the prim at the desired position.
The prim remains on the cursor and you can place it on the sheet as often as required.
4. Finish the procedure by choosing Exit Tool from the popup menu.

The Instance Group Text

An instance group contains two text strings of following types:

- 3D Instance Project Name
specifies the directory name (including appropriate punctuation) under which the model file is stored. This text is only required if the model file is not held in the current directory. If the model file is in the current directory, this text should be deleted from the group.
- 3D Instance Model Name
specifies the name of the model file that is to be included in the assembly at the position defined by the instance viewprim.

The following figures show instance groups referencing the model file *turbineblade.mod*. In [Figure 177](#) the model file is not held in the current directory but in a subdirectory called *sheets* in the main directory of hard disk *E*. In [Figure 178](#) the model file is in the current directory.

Figure 177 Example of an Instance Group With Specification of the Path

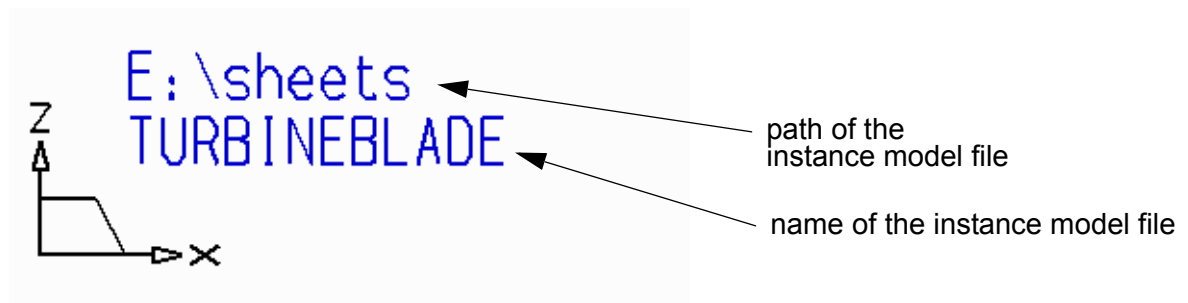
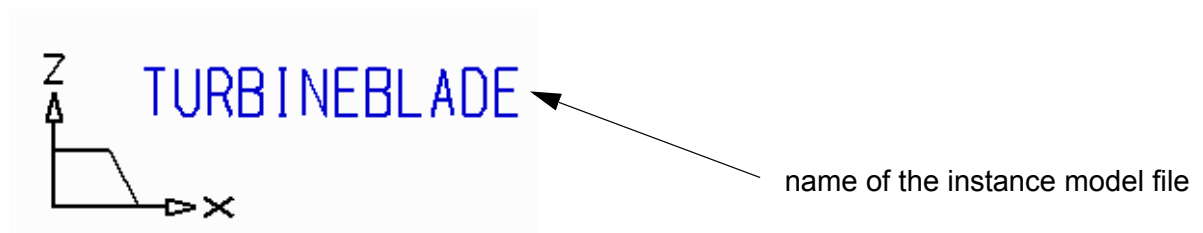


Figure 178 Example of an Instance Saved in the Current Directory



The Instance Viewprim

The purpose of the viewprim in the instance group is to:

- position the instanced model in the viewbox and
- define the orientation of the instanced model in the viewbox.

Therefore, when choosing which instance group to use, the first thing is to decide which view of the model is correct for a particular viewbox in the assembly drawing sheet. The instance group which corresponds to this view can then be selected in the appropriate toolset and placed at the required position in the viewbox.

Please note: The instance viewprim does not need to match the viewbox viewprim, and it can be rotated in the plane of the sheet to produce the required view.

Positioning and Scaling the Instanced Model

The position of an instance viewprim datum point on an assembly sheet marks the origin of that instanced model coordinate system in the two viewbox axes.

As an example, the complete definition and model of an object (sheet name `BLOCK`) is shown in [Figure 179](#) and [Figure 180](#), and shows the instance group representing this model added to the assembly sheet and the resultant position of the model.

Figure 179 The Model `BLOCK`

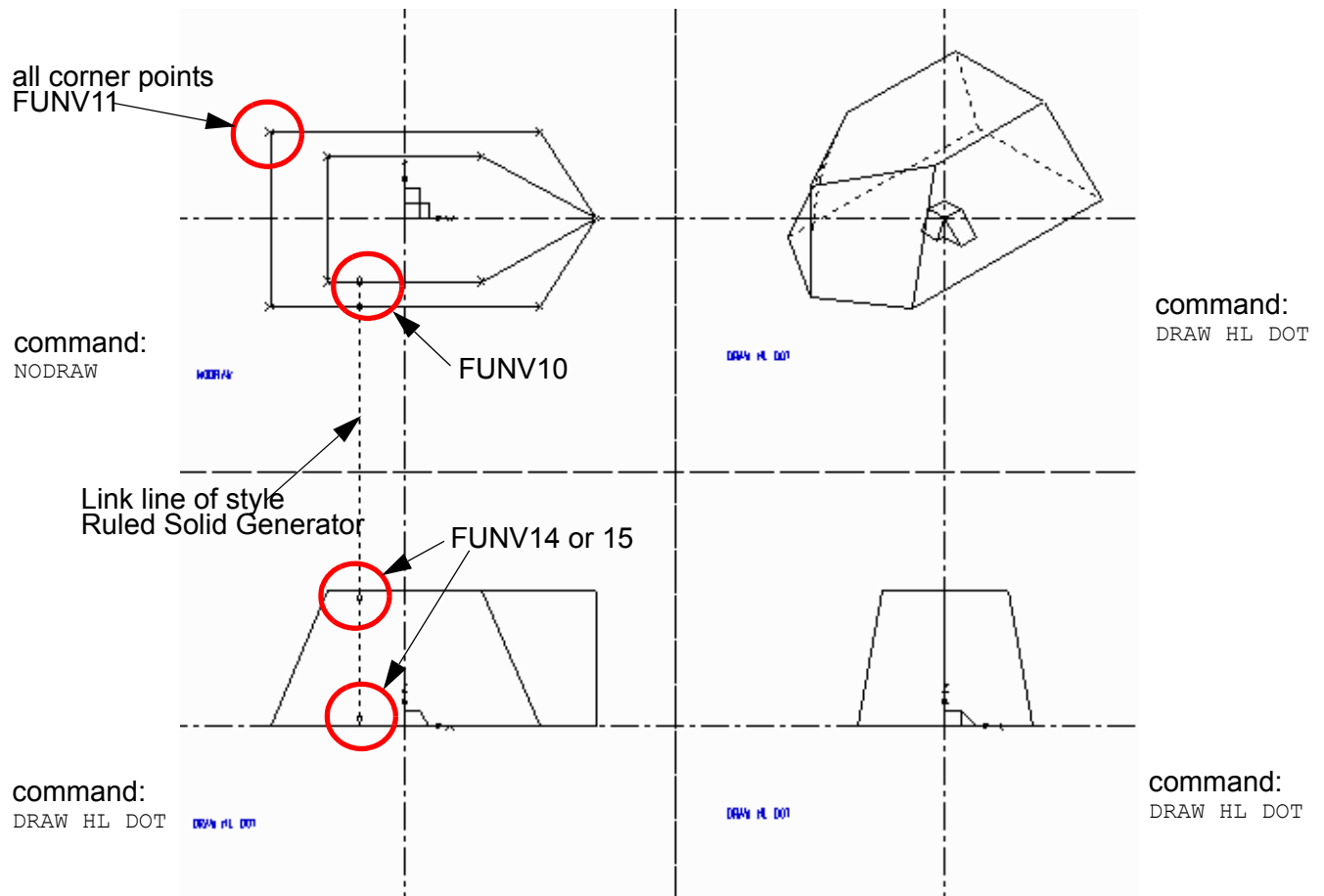
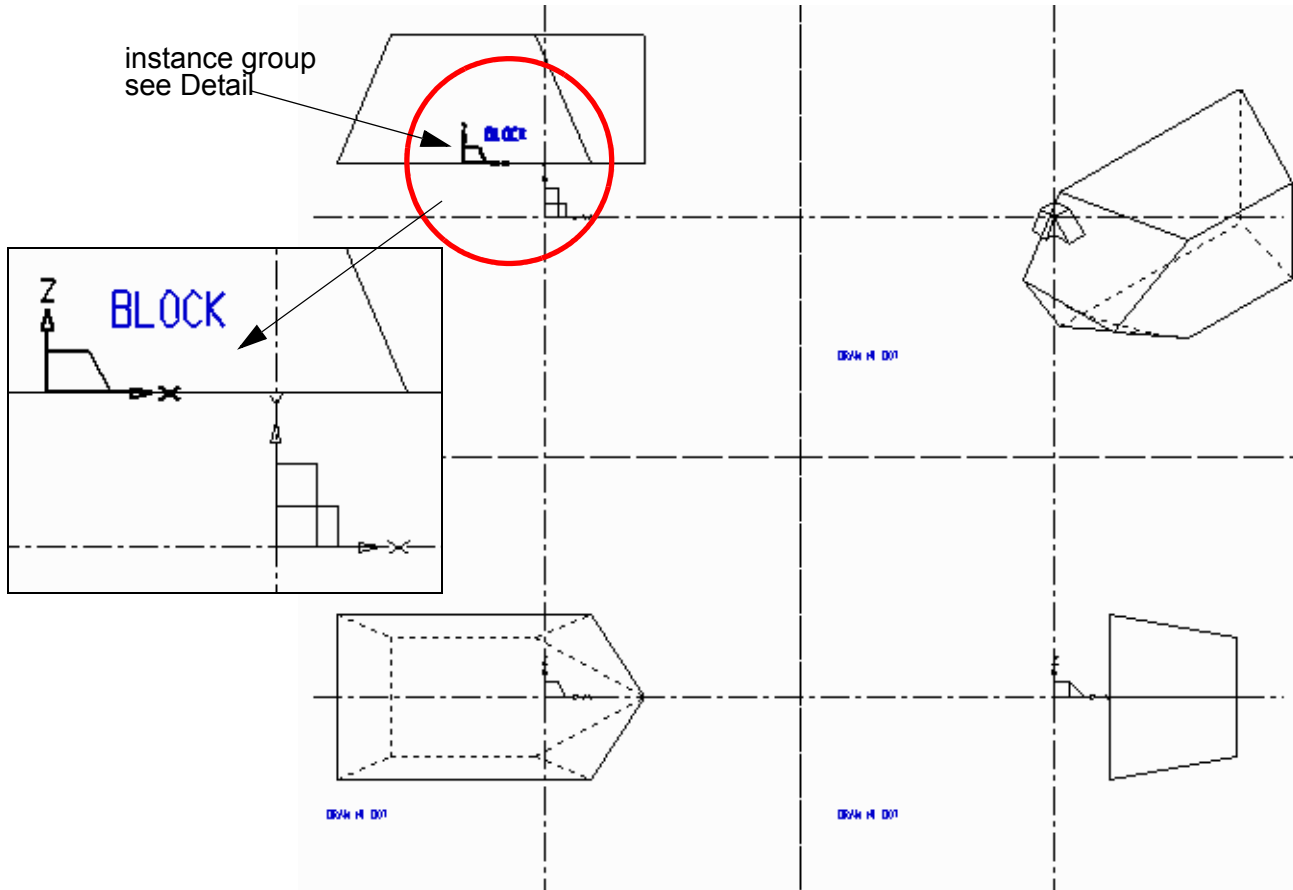



Figure 180 The Model BLOCK Added to the Assembly Sheet



Please note: The instance group shown in [Figure 180](#) references a model file in the current directory.

Positioning the Model in the Third Axis

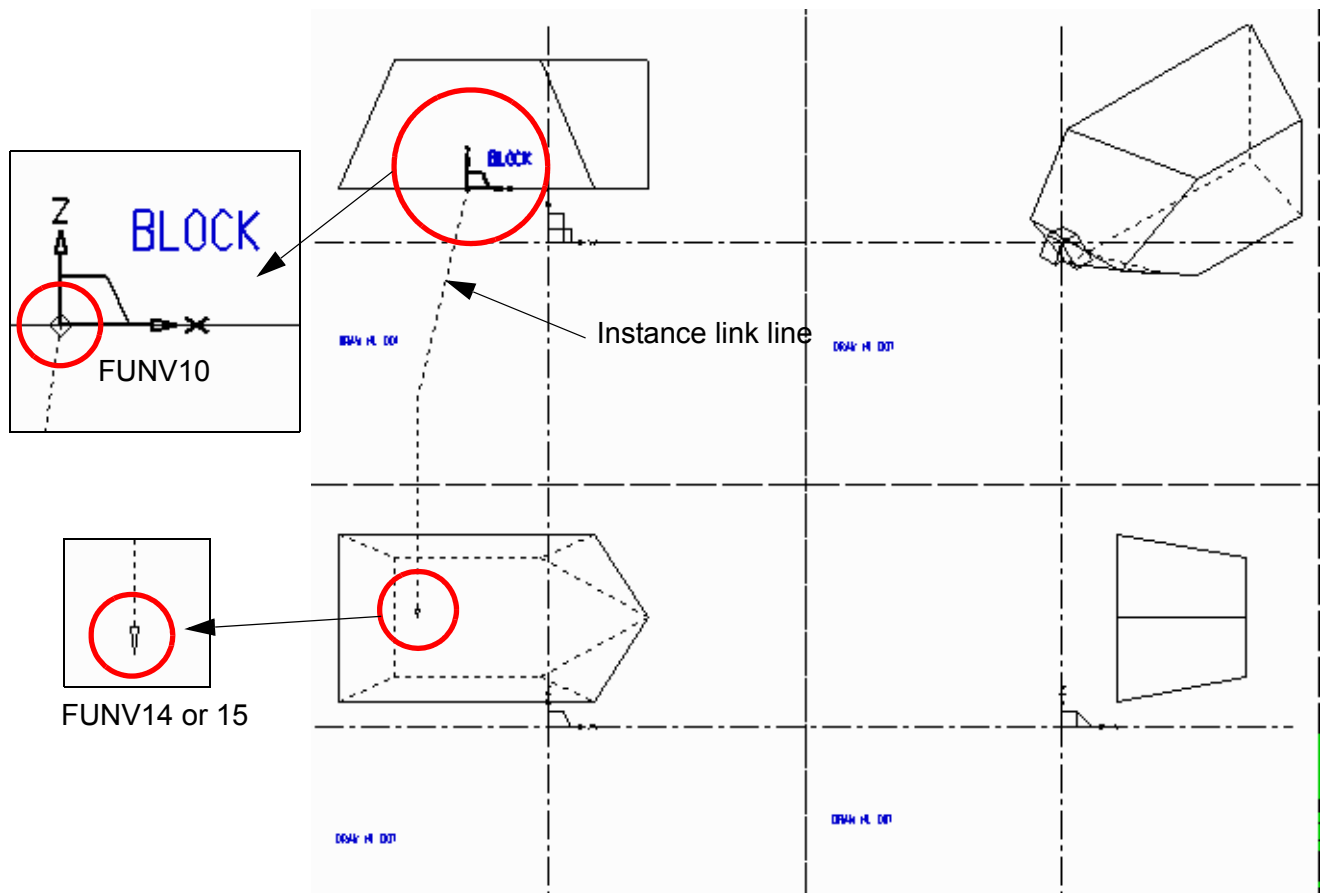
Although the position in the third axis is zero by default, this can be overridden by adding a link line of style `Instance link line` to the instance group. The following procedure shows how to do this:

1. The link line has to be part of the instance group, therefore select either:
 - the according 3D group in the Structure Tree OR
 - the prim of the instance group in the viewbox or
 - the text of the instance group in the viewbox
2. Choose the `Creates instance link line` tool  from the Lines tool group.

3. Attach the link line to the datum of the instance group using a diamond point function (FUNV 10).
4. Extend the link line into another viewbox and define the position of the instance model in the third dimension by placing a single arrowhead point function (FUNV 14 or 15) at the end of the link line.
5. Select the Model + Reconstruct tool to generate and view the completed assembly (only one instance model in this case).

Figure 181 shows the same assembly drawing sheet as Figure 180 but with an instance link line defining a non-zero third-axis position for the instanced model.

Figure 181 Positioning the Model BLOCK With a Link Line



Multiple instanced models can be created on an assembly sheet by placing additional arrowhead point functions (FUNV 14 or FUNV 15) or depth texts on an instance link line. Depth texts are specified using a SMI-text of style `Modeller Text ass. by Set`. See the following subsection to find out how to select SMI-text of style `Modeller Text ass. by Set`.


Scaling an Instanced Model

It is possible to specify that an instanced model is to be modeled at a scale other than full size in the assembly drawing, although this facility is seldom required. The command to use is:

```
SCALE scale_factor
```

This command is specified as SMI-text of style `Modeller Text ass. by Set` and added to the instance group. Scaling is performed relative to the datum of the instanced model during modeling of the assembly sheet. A scale factor of greater than 1 increases the size of the model, and vice versa.

To include the `SCALE` command in an instance group follow the procedure described below:

1. Select the prim of the instance in the viewbox.
2. Choose the `Creates a SMI text tool` .
3. Insert the `SCALE` command followed by the desired scale factor in the text input field.
The command is attached to the cursor.
4. Place the command close to the instance group.
5. Select the `Model + Reconstruct` tool to generate and view the completed assembly.
The instanced model is created according to the added scale factor.

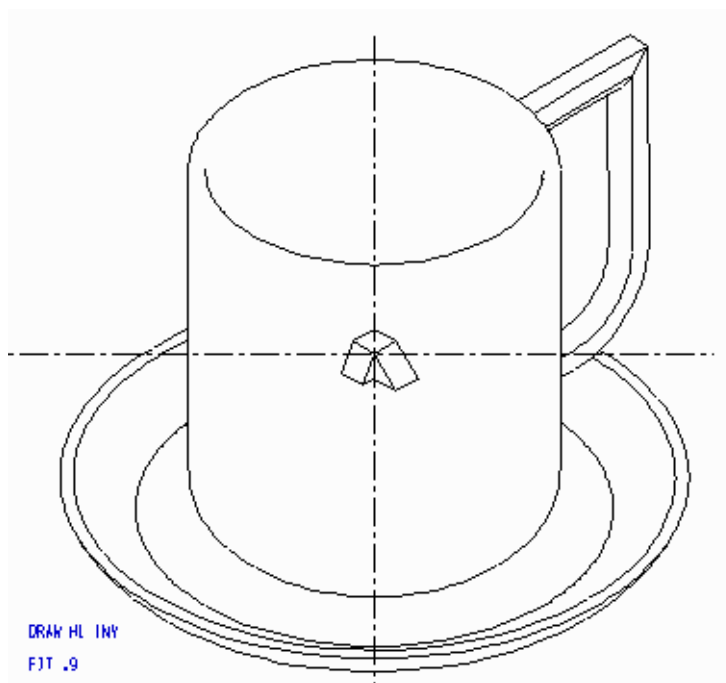
Modeler commands can also be added to an instance group using text of style `Modeller Text ass. by Set`. For details see [“General Modeler Commands” on page 281](#).

An Example of an Instanced Model

A sheet can contain both ordinary model definitions and instance groups. This means that instance groups can be accurately positioned on the profile of a model in the construction of assembly drawings.



For example, [Figure 182](#) shows a model of a cup and saucer. The body of the cup is defined as a volume of revolution on a 3D sheet. The cup handle and saucer are drawn as separate models and instanced onto the sheet that contains the cup profile. The cup handle is defined as a slide and the saucer as a volume of revolution.

Figure 182 Model of a Cup and Saucer



Definition of the Cup Corpus

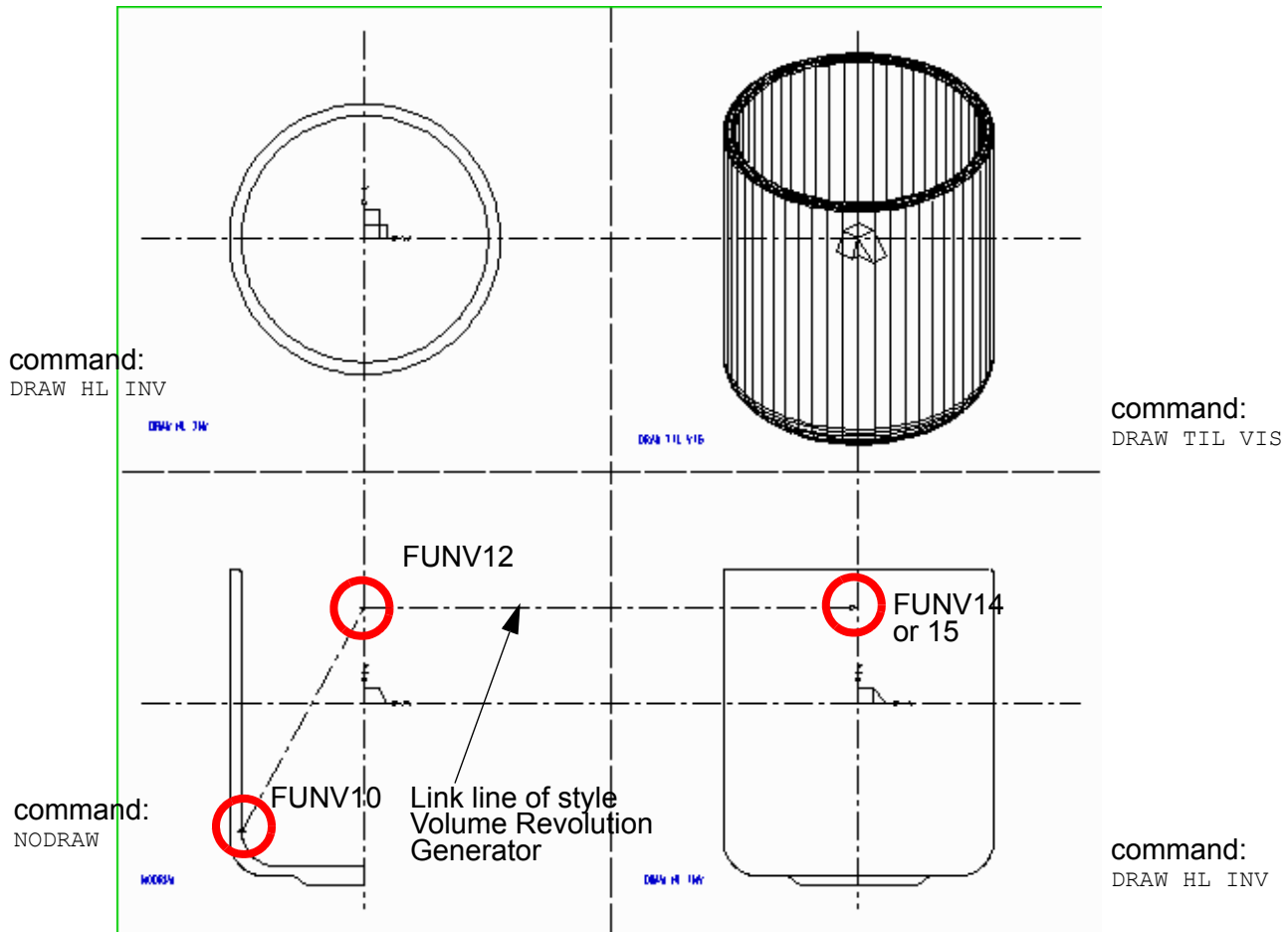
Open a sheet and draw the definition of the cup corpus using the volumes of revolution modeling technique (see [“Volumes of Revolution” on page 95](#)). This sheet is also the one on which you should later create the complete cup, that is, it is your assembly sheet.

1. Draw the profile of the cup corpus in a viewbox containing a ZX viewprim. Use a profile line of style `Profile LP0` through `LP9`.
2. Choose the Volume Revolution tool  and draw a link line (style `Volume Revolution Generator`) starting at the profile line (`FUNV10`), attached to the centerline (`FUNV12`) and ending with point function `FUNV14` or `15` as shown in [Figure 183](#).
3. Select the Model + Reconstruct tool  to generate the model file and show the model.

Please note: Make sure that you used a link line of style `Volume Revolution Generator`



and the link line is attached to the centerline.
 Consider the use of the right point functions according to [Figure 183](#).

Figure 183 Definition and Model of the Cup Corpus



Definition of the Cup Handle

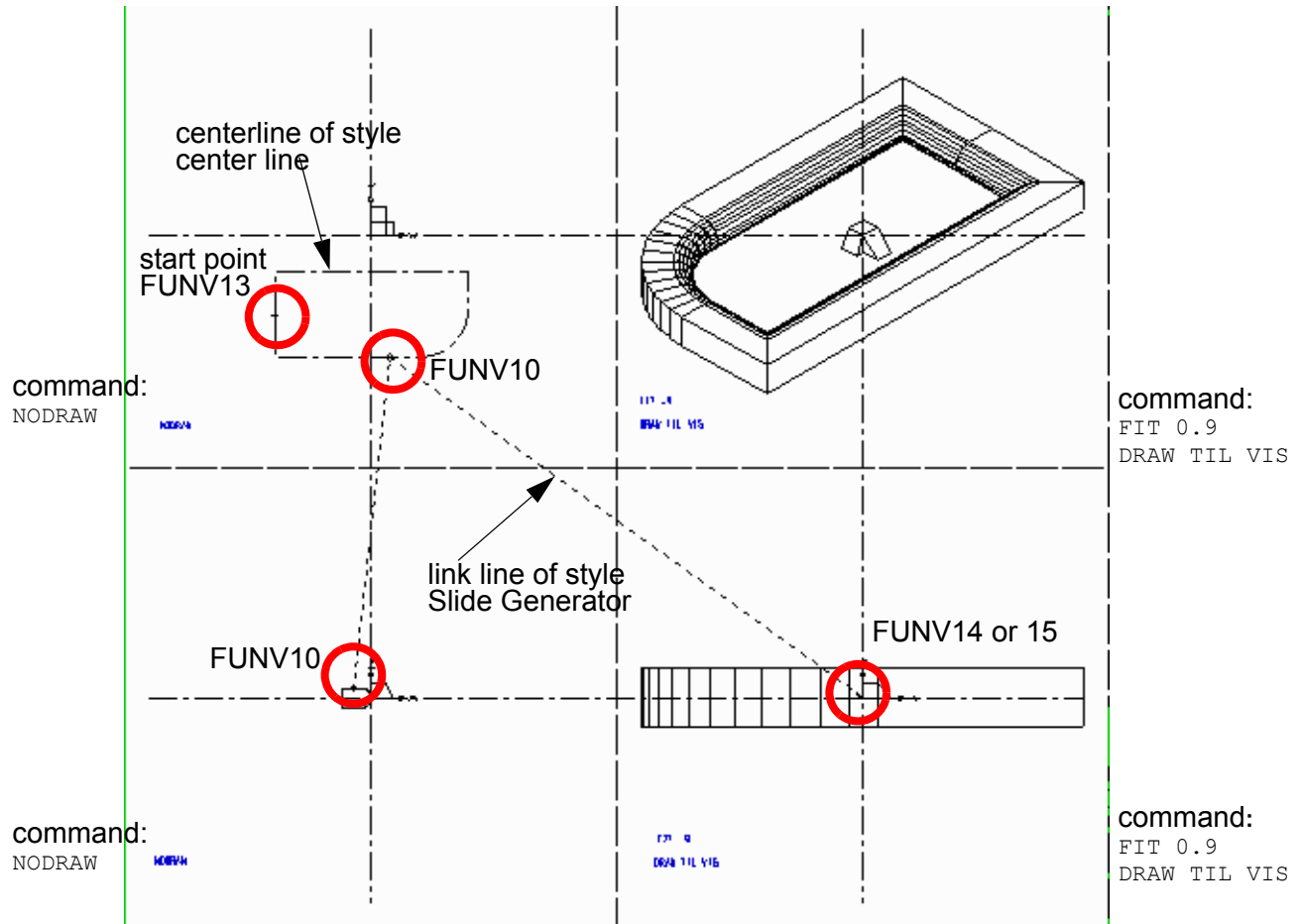
Open a new sheet and draw the definition of the cup handle using the slide modeling technique (see [“Slides” on page 125](#)).

1. Choose the **Create Center Lines** tool  to draw the centerline of the handle.
2. Set a start point (FUNV13) on the centerline as shown in [Figure 184](#).
3. Draw the profile of the handle using a profile line of style `Profile LP0` through `LP9`.
4. Choose the **Slide** tool  and draw a link line of style `Slide Generator` starting from the profile line with point function FUNV10, attached to the centerline (FUNV10) and ending with FUNV14 or 15 as shown in [Figure 184](#).

Please note: Consider the position of the YX viewprim in relation to the centerline. It is on the side of the handle that will be joined to the cup and is positioned halfway up the

handle.

Figure 184 Definition and Model of the Cup Handle



5. Select the Model + Reconstruct tool  to generate and view the model of the handle.

Definition of the Saucer

Draw the definition of the saucer using the volumes of revolution modeling technique (see “Volumes of Revolution” on page 95).



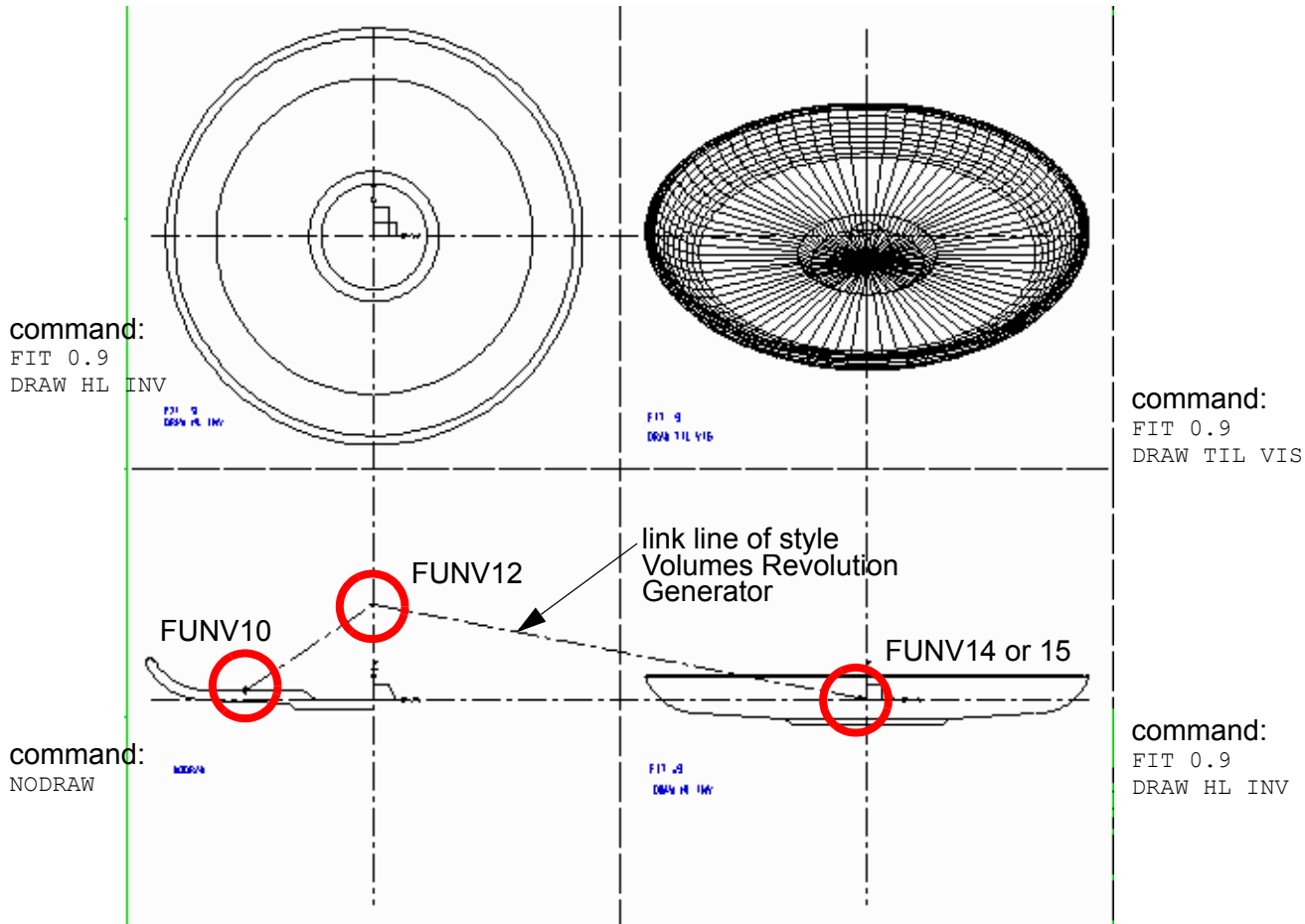



1. Draw the profile of the saucer in a viewbox containing a ZX viewprim. Use a profile line of style Profile LP0 through LP9.
2. Choose the Volume Revolution tool  and draw a link line (style Volume Revolution Generator) starting at the profile line (FUNV10), attached to the centerline (FUNV12) and ending with point function FUNV14 or 15 as shown in Figure 185.
3. Select the Model + Reconstruct tool  to generate the model file and show the model.

Figure 185 Definition and Model of the Saucer



Now any required parts of the cup are completed and you can start the assembly.

Adding the Handle and Saucer to the Cup Corpus

1. Make the sheet with the cup corpus, which you created at first, to the current sheet.
2. To add the handle of the cup to the assembly choose the XY Instance tool  and select the *cup_handle.mod* file from the Instance group dialog as described in the section “Instance Groups” on page 244.
3. Position the datum of the prim on the profile line of the cup as shown in Figure 186. The text which specifies the path name can be deleted as the model file is held in the current directory. This contributes to the clarity of the drawing.
4. Select the Model + Reconstruct tool  to generate and view the model of the cup. Verify that the handle is in the correct position. If not, adjust the position of the instanced group in the assembly sheet and generate and view the model again.
5. To add the saucer to the assembly choose the XZ Instance tool  and select the *saucer.mod* file from the Instance group dialog as described in the section “Instance Groups” on page 244.


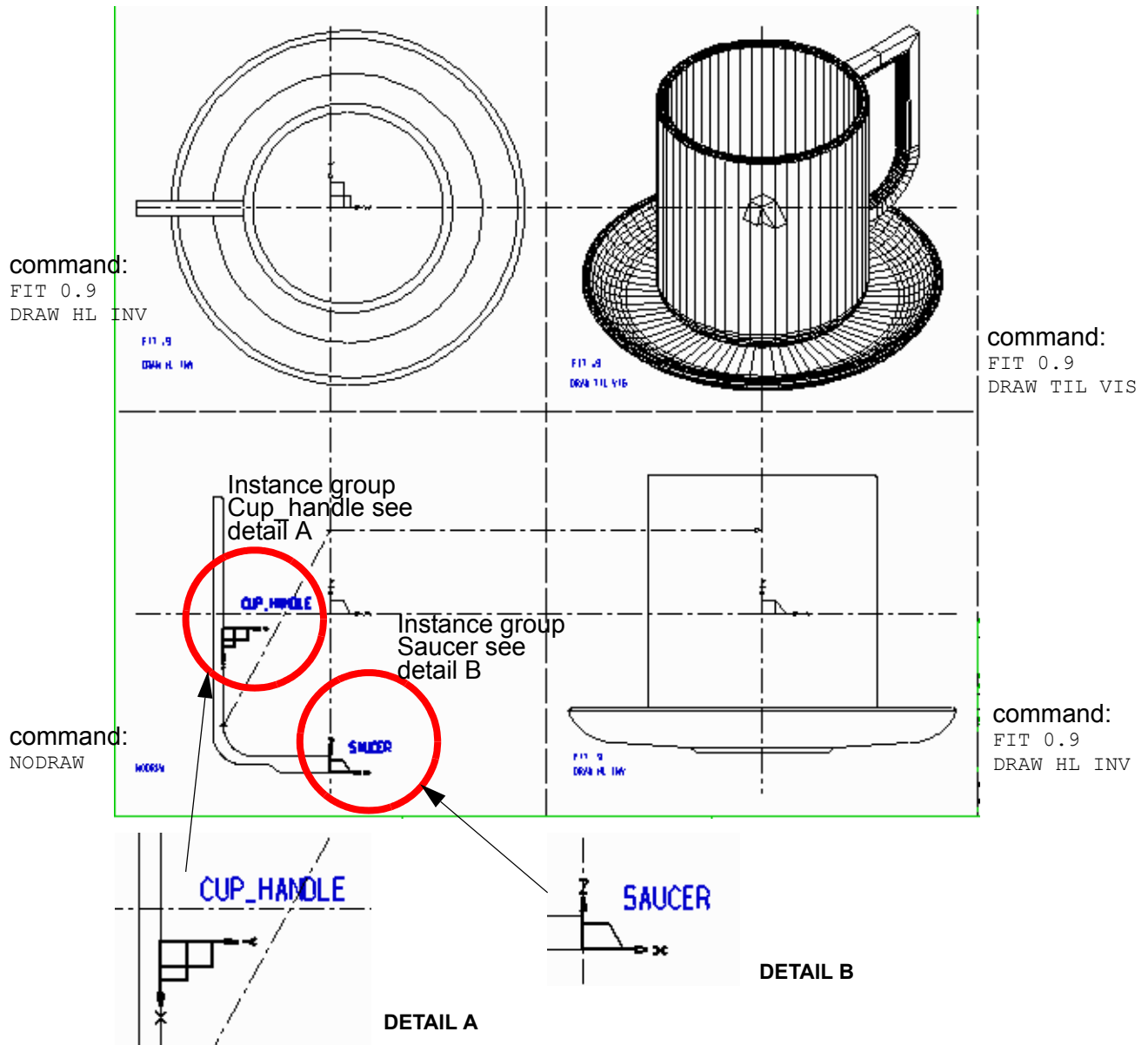
6. Position the datum of the prim on the profile line of the cup as shown in [Figure 186](#). The text which specifies the path name can be deleted as the model file is held in the current directory. This contributes to the clarity of the drawing.
7. Select the Model + Reconstruct tool  to generate and view the model of the cup. Verify that the saucer is in the correct position. If not, adjust the position of the instanced group in the assembly sheet and generate and view the model again.

Figure 186 Completed Assembly Model of the Cup with Handle and Saucer



Naming Objects in Instanced Models

When instance groups are incorporated on a 3D sheet and modeled, the objects in the new model file created as a result of this process are named according to a certain convention. This convention imposes a naming structure that allows you to apply Boolean operations to individual objects created within an assembly.

Naming command text is included in an instance group as SMI-text of style `3D Instance Model Text`. The name of the object can be specified in one of two ways:

```
NAME object_name  
& object_name
```

This SMI-text of style `3D Instance Model Text` is added to the instance group by the method shown [“Scaling an Instanced Model” on page 250](#).

The Naming Convention

[Figure 187](#) shows how the naming convention operates on object names in the resultant model file when the instance symbols do not carry naming text.

Figure 187 The Naming Convention Without SMI Text in the Instance Groups

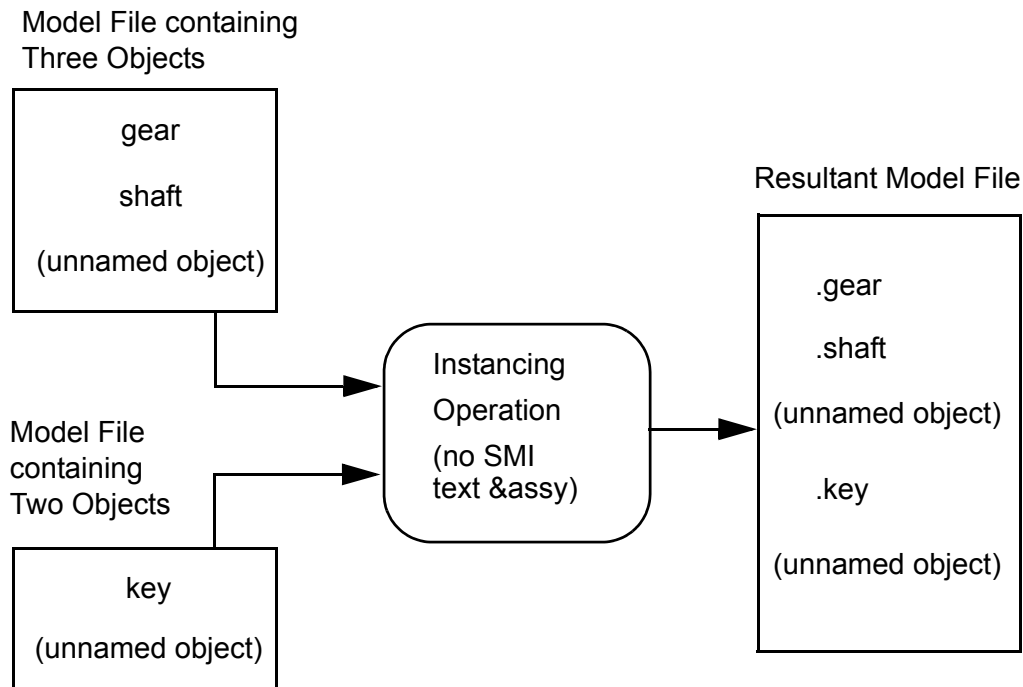
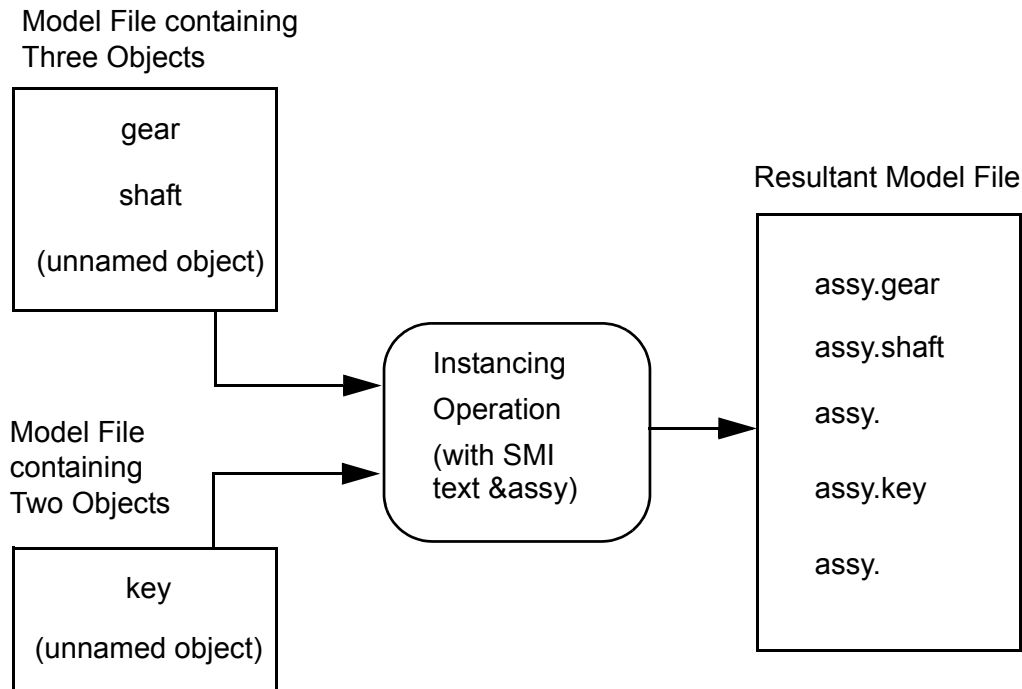


Figure 188 shows how the naming convention operates on object names in the resultant model file when the instance symbols carry naming text. Compare with Figure 187.

Figure 188 The Naming Convention With SMI Text in the Instance Groups

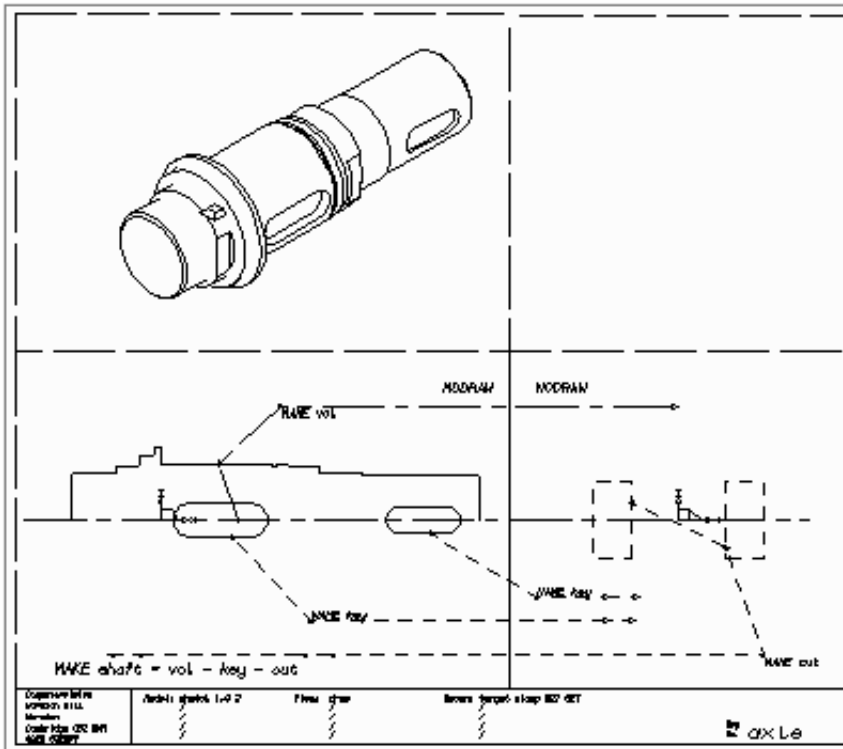


Examples

An example follows, that illustrates the naming convention process described in the previous section.

First, as shown in Figure 189, the model of an axle is generated. An object named `shaft` is created by the `MAKE` command.

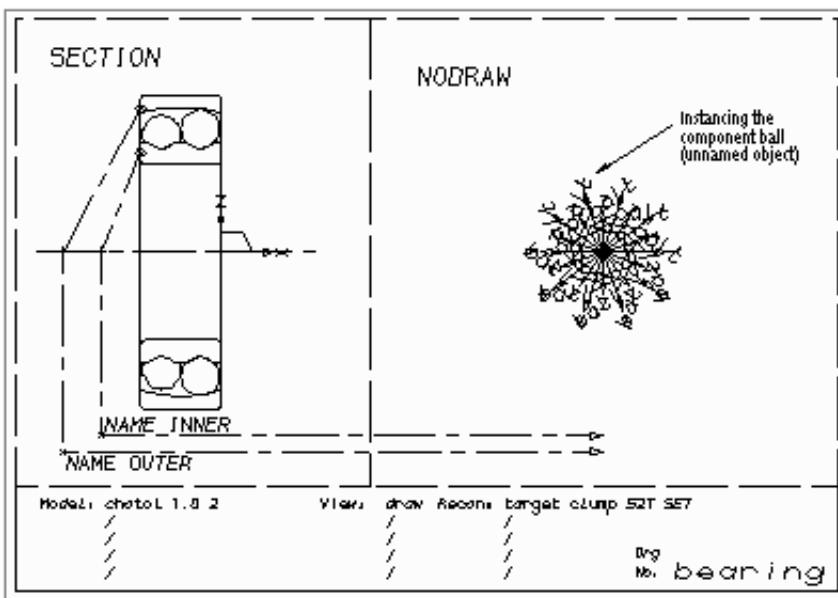
Figure 189 Model of an Axle



Then, as shown in Figure 190, the model of a bearing is generated, which contains several objects:

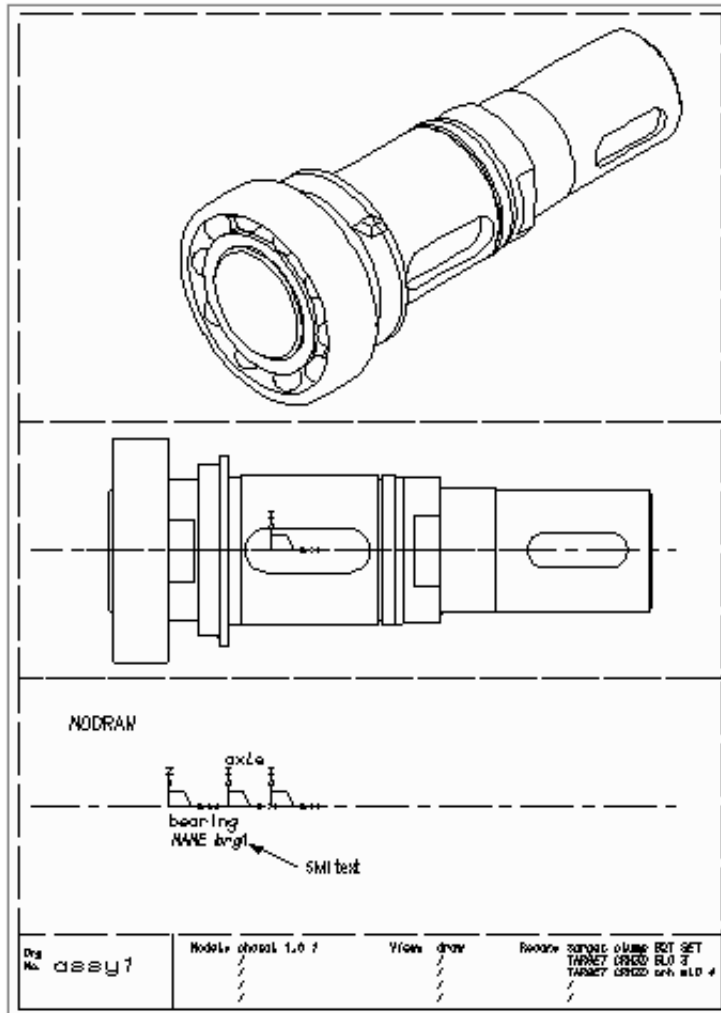
- Two objects that are named (INNER and OUTER)
- Several unnamed objects (the component ball which is instanced ten times)

Figure 190 Model of a Bearing



Finally, as shown in [Figure 191](#), the axle and bearing are instanced to generate the assembled model `assy1`. Only the instance group representing the bearing, has instance naming text of type SMI (NAME `brg1`) included in the instance group.

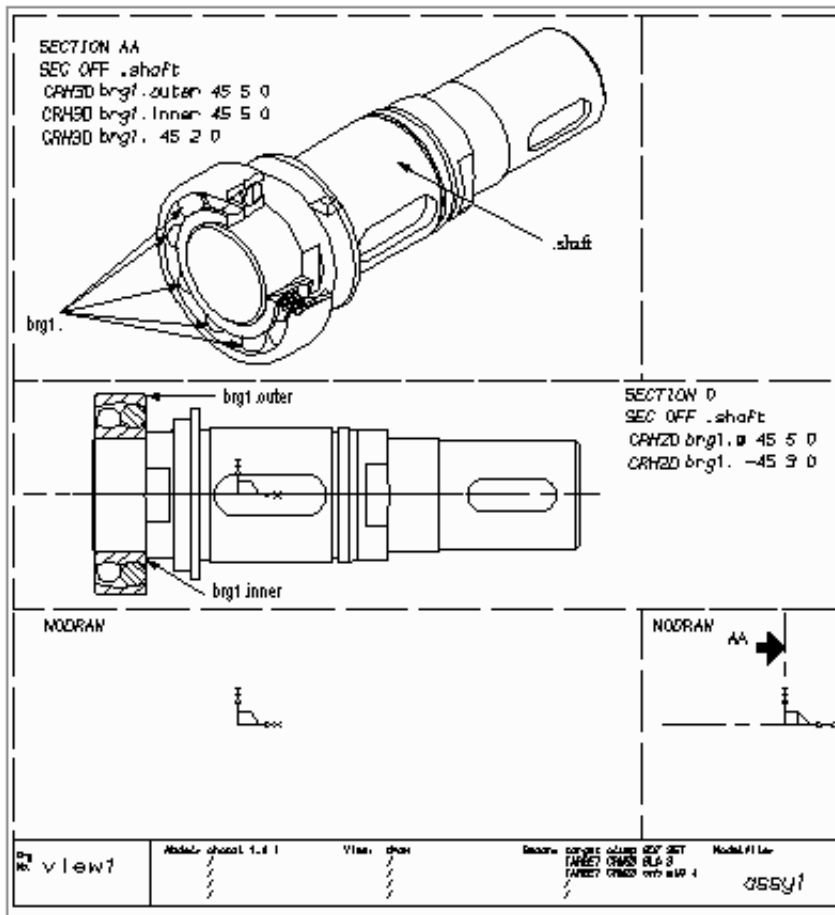
Figure 191 Completed Assembly Model of Axle and Bearing



View1, in [Figure 192](#), outlines how the naming convention operates on objects names, in an assembly. The model bearing, which was instanced and named `brg1` with SMI text, is identified as object `brg1.` or `brg1.inner` and `brg1.outer` for the inner and outer bearings. The object shaft is identified as `.shaft` as the instanced model axle was not named with text of style 3D Instance Model Text.

To reveal the name of the objects contained in an assembly you may find useful to run the Model Validator which generates a report on the model's validity, listing the objects on the screen, with their name. For further information on the Model Validator, see [“The Model Validator” on page 361](#)

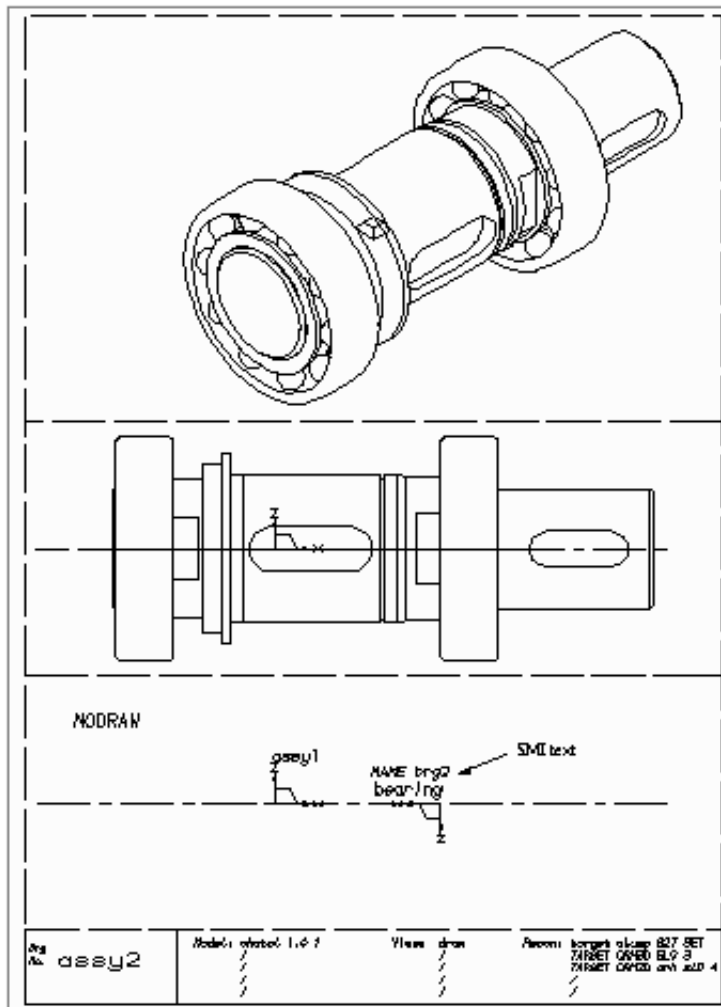
Figure 192 View1 Illustrating Naming Convention



Please note: In [Figure 192](#) and [Figure 194](#), the commands (SEC OFF, SECTION, CRH2D and CRH3D), associated with the objects shown, are described in “[Sectioning the Model](#)” on page 318, where these views are shown again.

As shown in [Figure 193](#), `assy1` and a second bearing are instanced to generate the assembled model `assy2`. Only the instance group representing the second bearing, has instance naming text of type SMI (NAME `brg2`) included in the instance group.

Figure 193 Completed Assembly Model of Axle with Two Bearings



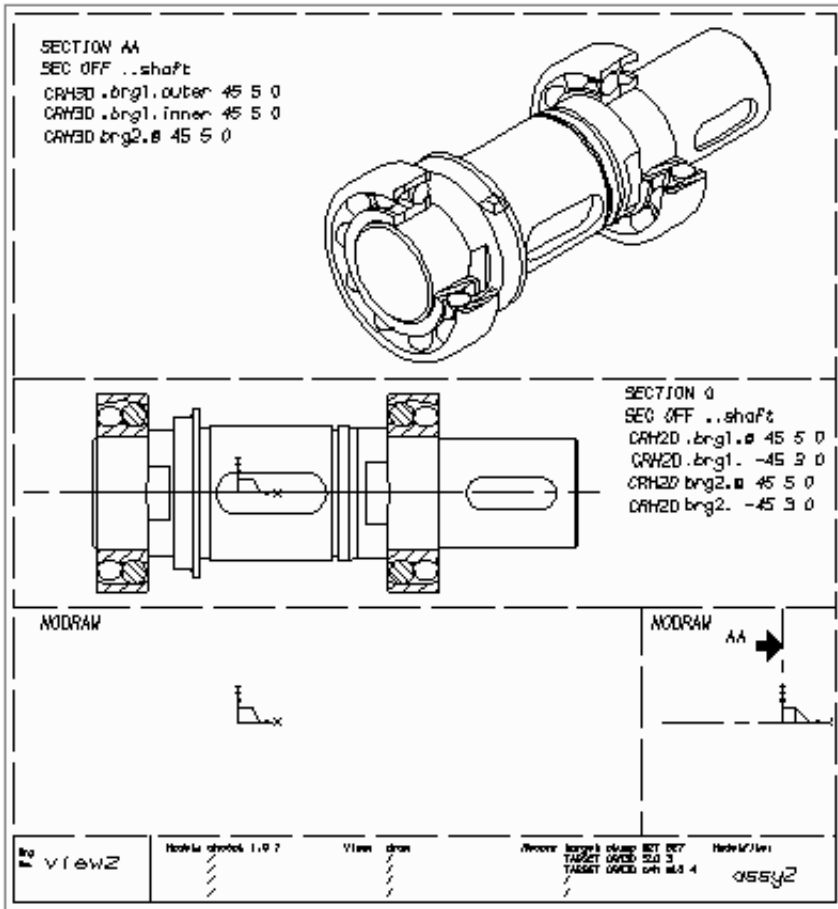
In View2 (Figure 194), the second model bearing, which was instanced and named `brg2` with text of style Modeller Text `ass. by Set`, comprises the objects:

- `brg2.`
- `brg2.inner` and `brg2.outer`

Both `brg2.inner` and `brg2.outer` are selected with `brg2.@`. For information on the wild card `@` see page 338.

The objects `.shaft`, `brg1.`, `brg1.outer` and `brg1.inner`, contained in the model file `assy1`, are all identified by the dot prefix (`.`) as the model file `assy1` was not named with text of style Modeller Text `ass. by Set` when instanced.

Figure 194 View2 Illustrating Naming Convention



Creating an Assembly View With the Help of AVIEW

The `AVIEW` command allows you to create views of an assembly built from instanced objects before the assembly sheet is sent to the Modeler. This feature means that you can align the instances quickly and easily, by trial and error, without having to know the precise spatial relationship between them.

Please note: The `AVIEW` command creates views in `SKETCH` mode only.

When viewing with the `AVIEW` command you can:

- Generate a view with two or more instances in the same viewbox.
- Generate any type of view by using any standard viewprim and the following TVS-text of style `View Specification Text` commands:
 - FROM
 - TO
 - PER
 - PAR
 - UPVEC
 - ONTO
 - ANGLE
- Use pan and zoom.

Please note: The `AVIEW` command does not alter the model files in any way.

Building the Assembly

The procedure below shows how to build and view the initial assembly drawing. You will place the parts of the assembly step by step using instance groups.


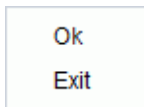
1. Place an instance group on the sheet and follow the subsequent steps.
2. Make the instance group current by either:
 - opening the Structure Tree and select the appropriate 3D group or
 - choosing the *Selects sheet-level named groups* tool  from the In Graphics Tool Bar and select the group in the sheet.
3. Move the cursor over the toolset with the instance group tools and press the RMB. (See [Figure 175, "Instance Groups Toolset" on page 244](#)).
The *AVIEW* command appears.
4. Click on *AVIEW* using the LMB.
This puts the system in *AVIEW* mode and verifies the instance symbol.
5. Click the LMB in the viewbox where you want the view to appear.
6. Click the RMB to open the following popup menu.

Figure 195 The *AVIEW* Popup Menu



OK

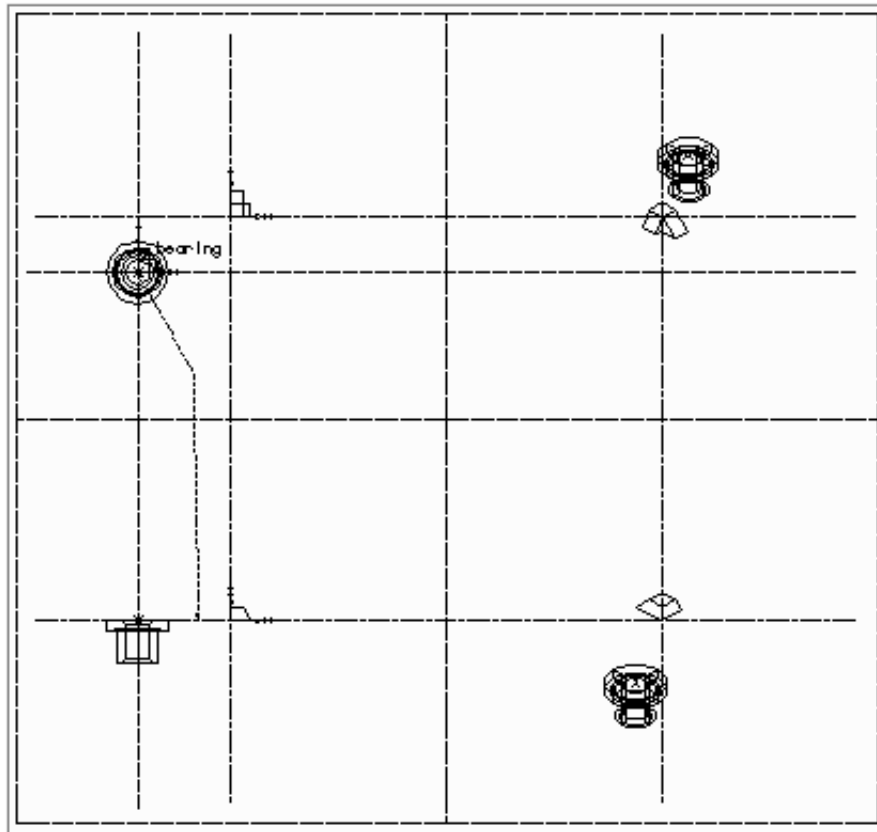
draws a view of the instanced object in the viewbox.


EXIT

quits the *AVIEW* mode.

7. Choose OK to draw the view of the bearing.
8. Continue drawing other views by clicking the LMB in any other viewbox and then choosing OK from the popup menu.
[Figure 196](#) shows an instance of a bearing which is to be assembled with its mating carrier plate. The bearing is displayed in all four viewboxes.

Figure 196 An Instanced Bearing Drawn In All Viewboxes On the Assembly Sheet Using AVIEW

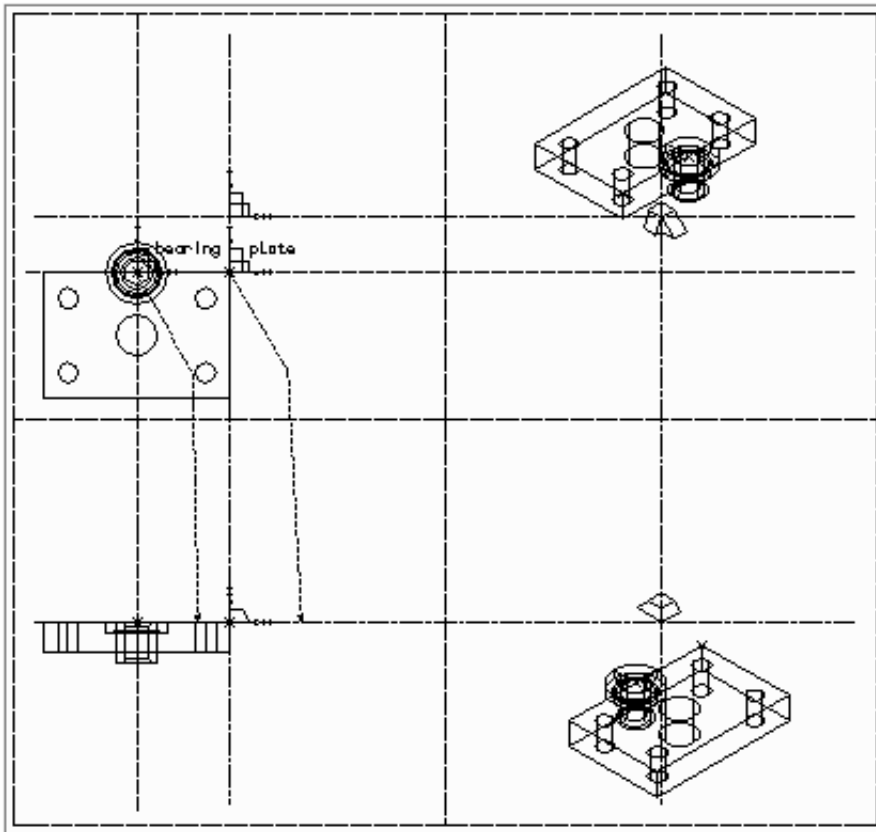


9. Leave the AVIEW mode by choosing the EXIT option from the popup menu.
10. Delete the views generated in this mode using the Delete tool  (for details on this tool see [page 32](#)).

Editing and Modeling the Assembly

Figure 197 shows the carrier plate added to the assembly sheet. The carrier plate is shown in all four viewboxes to see the position of the plate relating to the bearing. This is the first attempt at assembly, and it is obvious that the bearing and plate are misaligned. The position of the bearing relative to the plate is correct in the X-direction, high in the Y-direction, and low in the Z-direction.

Figure 197 An Instanced Carrier Plate Added to the Assembly Sheet

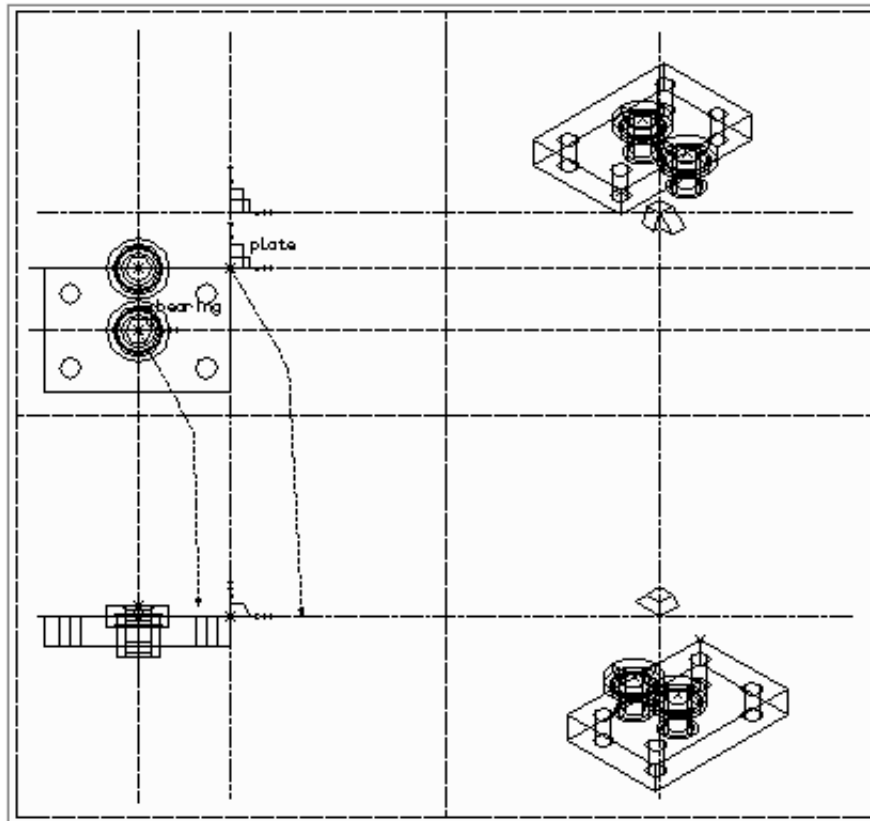


The procedure below shows how to edit and view an assembly drawing.

1. Identify an object that you want to move; in our example it is the bearing.
2. Move the instance group of this object and/or edit the link line or depth text as required, using the views of the other objects for positioning.
3. Use AVIEW as mentioned before to draw a view of the instance at its new position.

Figure 198 shows the bearing instance moved into alignment with the carrier plate. This was done by moving the instance group in the Y-direction, and the last point of the link line in the Z-direction.

Figure 198 The Bearing Moved Into Alignment With the Carrier Plate



4. If the new position is satisfactory, go on to the next object and repeat the above steps for this object. Otherwise repeat the above steps for the previous object.


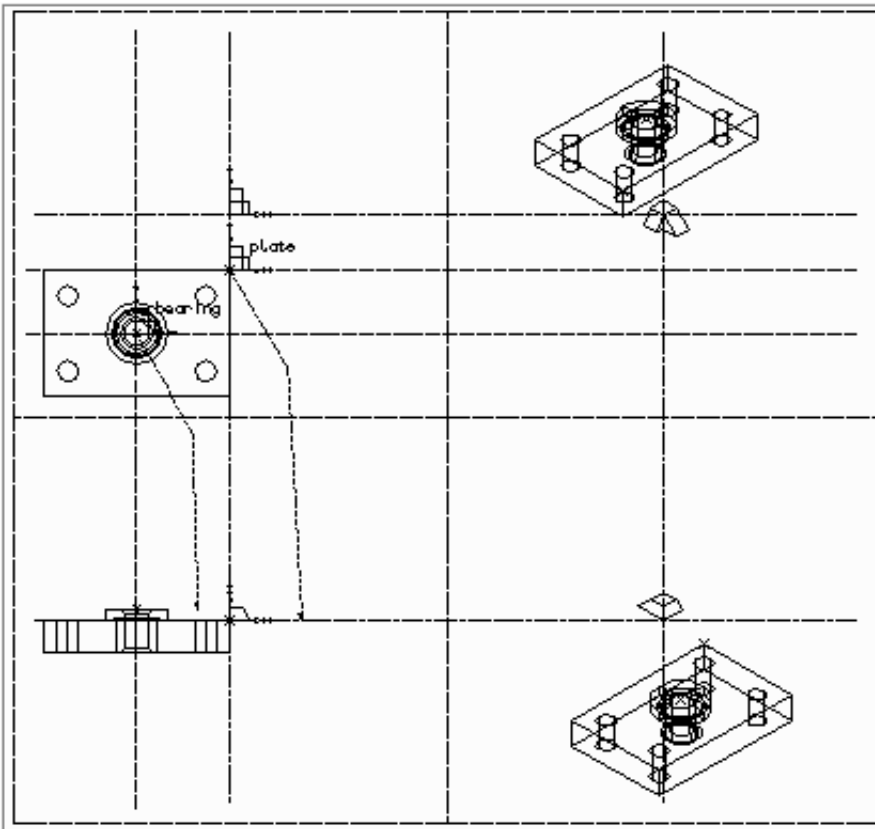
Please note: Note that the sheet retains the previous assembly views. If these views make the sheet confusing, delete all views by using the delete button . To restore the most recent view, make the appropriate instance symbol current and use AVIEW. If you prefer, individual views can be removed by finding the relevant view group and deleting it.

Figure 199 shows the bearing/plate assembly example with the previous views of the bearing deleted. This shows clearly that the bearing and carrier plate are correctly assembled on the sheet.

Figure 199 The Previous Views of the Bearing Deleted to Show the Assembly Clearly





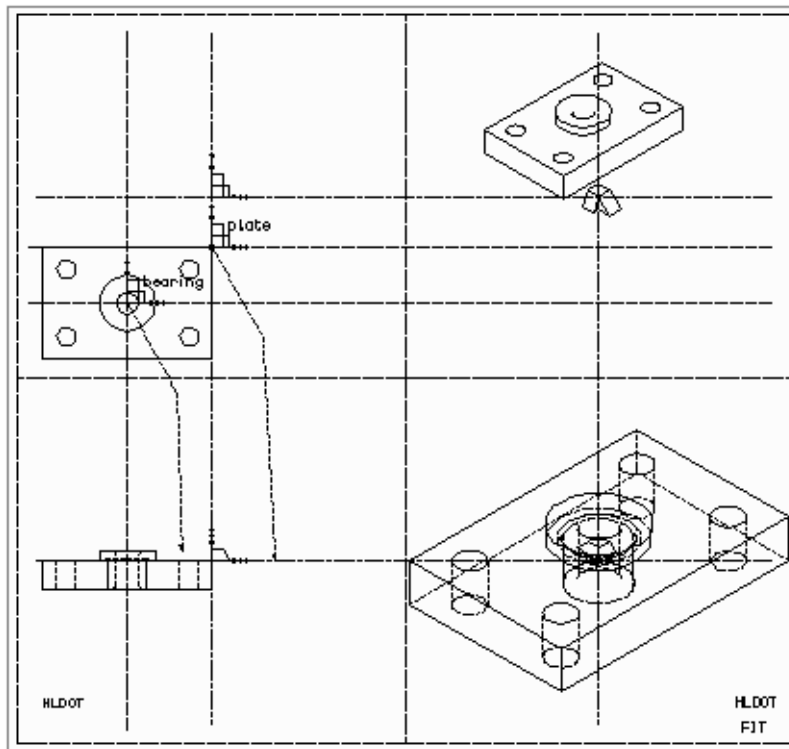
5. Repeat the above steps until all objects are correctly positioned.
6. Delete all views by using the Delete button .
7. Select the Model + Reconstruct tool  to model the completed assembly.

Figure 200, “The Completed Assembly of the Bearing and Carrier Plate” on page 269 shows the completed assembly of the bearing and carrier plate modeled and viewed as in the last step above. All previous views have been deleted, and normal viewing commands added to several viewboxes to produce the desired result.

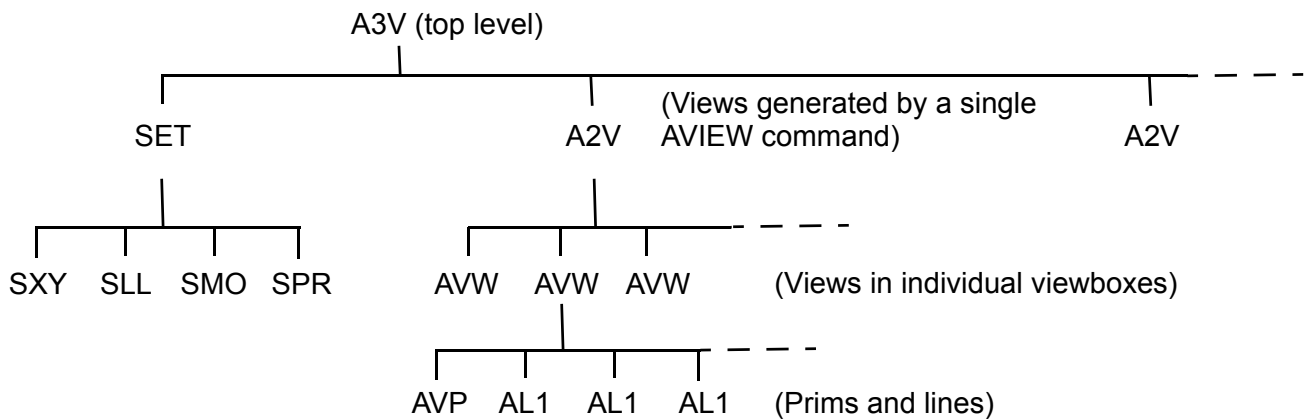
Figure 200 The Completed Assembly of the Bearing and Carrier Plate



The AVIEW Group Hierarchy

There is a special element group structure which is used for views generated by AVIEW. This is shown diagrammatically in [Figure 201](#).

Figure 201 The AVIEW Group Hierarchy



The advantage of this group structure is that it allows you to:

- Use `AVIEW` by making any element current that forms part of either the instance group (`SET`) you wish to view, or the `A3V` group generated by a previous selection of `AVIEW`.
- Delete or otherwise examine all views of the instance associated with a particular selection of `AVIEW`, on an instance group (`SET`), by making the relevant `A2V` group current.

Characteristics of the Group Structure

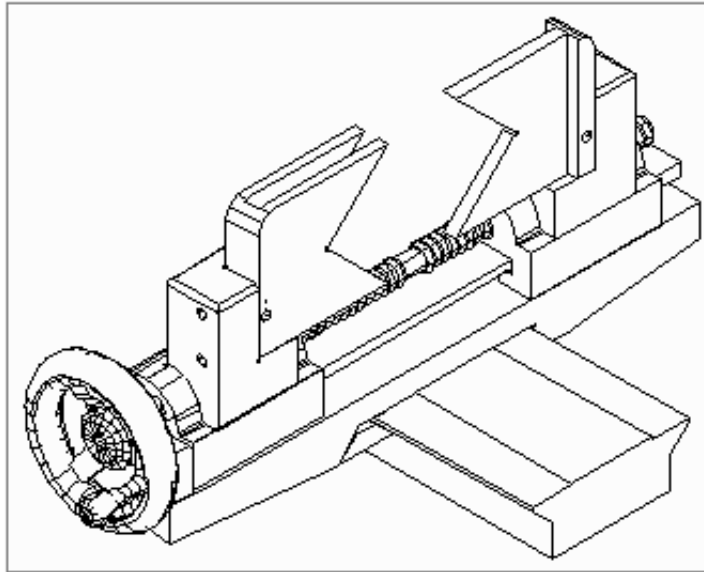
The `AVIEW` group structure has the following characteristics:

- The first time you probe `AVIEW` an `A3V` group is created and the `SET` group (the instance symbol) is moved down one level. Thus the `SET` group becomes part of the `A3V` group. Note that when you probe `Model + Reconstruct`, the Modeler processes all instance symbols in the usual way.
- All instances you generate by probing `AVIEW` are part of the same `A2V` group. Each time you probe `AVIEW`, a new `A2V` group is generated. These `A2V` groups parallel the instance symbol `SET` group. They are all members of the top-level `A3V` group.
- The individual views of a particular instance generated in each viewbox are `AVW` groups. These `AVW` groups are members of the `A2V` group.
- Another `A2V` group is generated each time you probe `AVIEW` on the same instance symbol.
- A new `A3V` group is generated for each instance symbol you probe `AVIEW` on. Each `A3V` group contains only one `SET` group. If you specifically load other `SET` groups into the `A3V` group when you probe `AVIEW`, the first `SET` group found will be used.

Exploded Views

An exploded view shows each object in an assembly separated from the others but with the original orientation maintained. For example, [Figure 202](#) shows a normal assembly and [Figure 203](#) shows an exploded view of the same assembly.

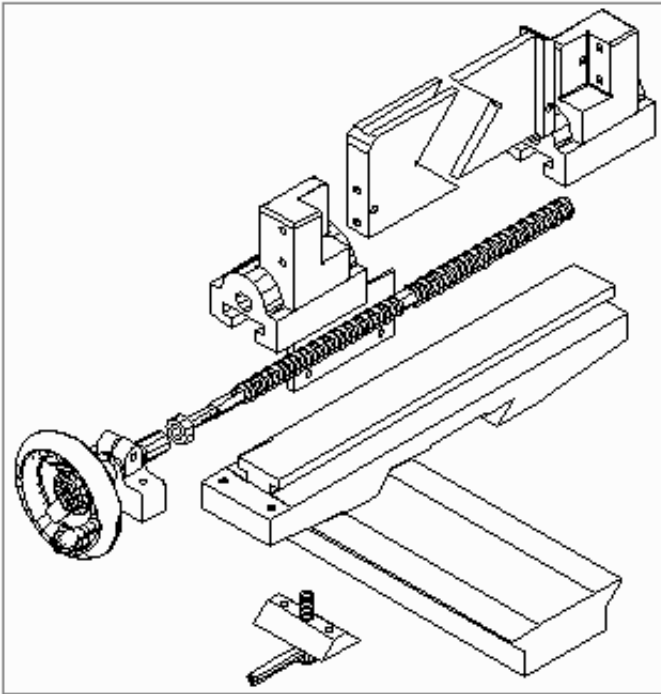
Figure 202 An Assembly



An exploded view can be produced by the straightforward method of instancing objects on a 3D sheet at the positions required for each object in the view. This method gives total control over spacing and alignment in the view.

As an alternative, exploded views can be produced using the MEDUSA4 Shrinker program. This program operates by shrinking each object about its center of gravity to produce the view. It is usually quicker than the above method but there is no control of alignment, which will depend on the distribution of mass in the objects comprising the assembly.

Figure 203 An Exploded View of the Assembly



Summary of Element Types

The following table gives a summary of the element types used in the creation of instanced models.

Element Description	Element Style and Point Function
Line Types	
Instance link line	Instance link line
Point Functions	
Attach link line to datum of instance group	FUNV 10
Indicate third dimension on instance link line	FUNV 14 / FUNV 15
Text Types	
Referencing model file	3D Instance Model Name
Referencing location of model file (directory specification)	3D Instance Project Name
Modeler commands in an instance group	3D Instance Model Text (SMI)
Depth texts	Modeller Text ass. by Geometry (TMG)

EXAMPLE CAMERA

Figure 204 shows a model of a single lens reflex camera created using the modeling techniques covered in the earlier sections of this manual.

Figure 204 Model of a Camera

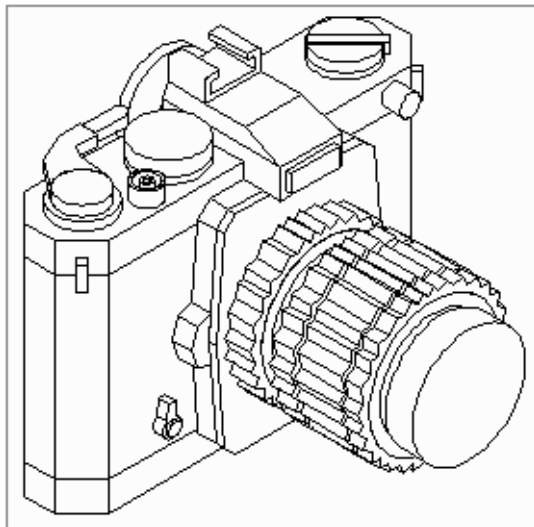


Figure 205 shows plan, front, and side views of the camera. Figure 206 lists the model generators which were used to produce each part.

Figure 205 Plan, Front, and Side Views Of the Camera

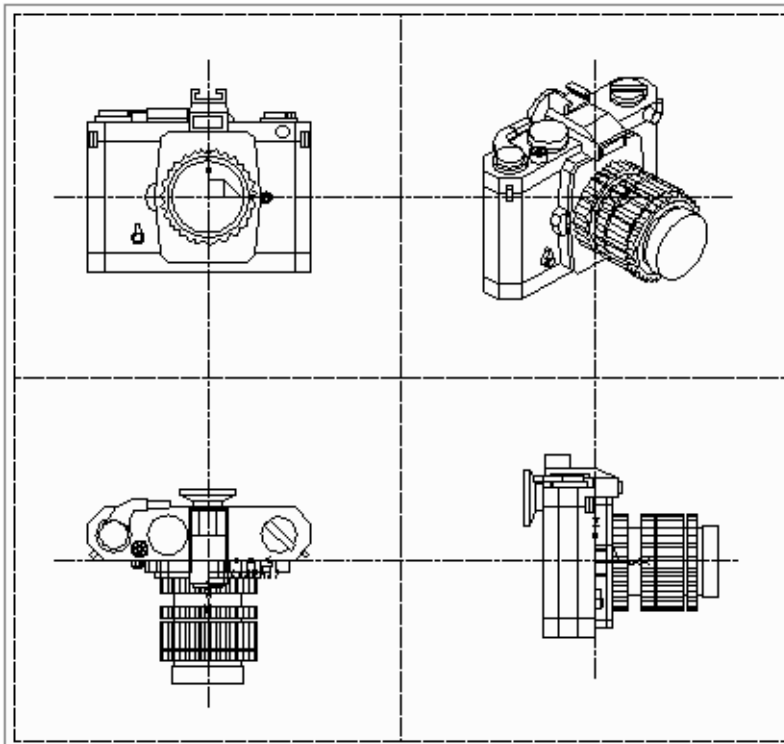
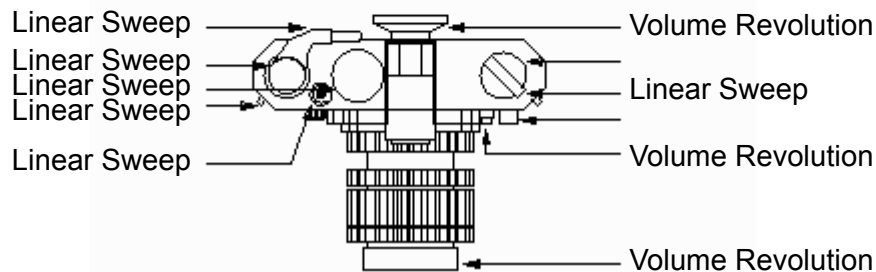
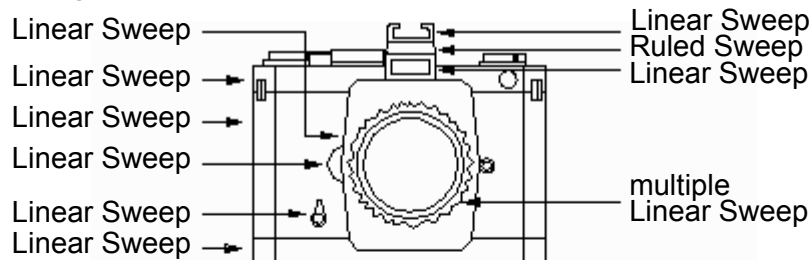


Figure 206 The Model Generators Used For Each Part Of the Camera



The camera definition is constructed as an assembly using instance groups on a 3D sheet. This is the normal technique to use on a multi-part model such as this. The advantage is that the

model definition is spread over many sheets rather than concentrated onto one. This keeps the part definitions clear and therefore easy to understand.

The following components of the camera are defined and modeled on separate sheets and then represented by instance groups in the final assembly definition:

- Lens
- Pentaprism
- Winder
- Rewinder
- Exposure delay lever
- Shutter release button
- Exposure time dial
- Viewfinder

The camera body is defined on the assembly sheet. This helps to position the instance groups in relation to the camera profile. Other simple profiles, for example the camera strap holders, are also defined on this sheet. [Figure 207](#) shows the completed assembly sheet. [Figure 208](#) shows an enlarged view of the XY viewbox (with link lines removed for clarity). Note the positioning of the datum points of the instance groups. The instanced models are loaded at these points.

Figure 207 The Camera Assembly Sheet

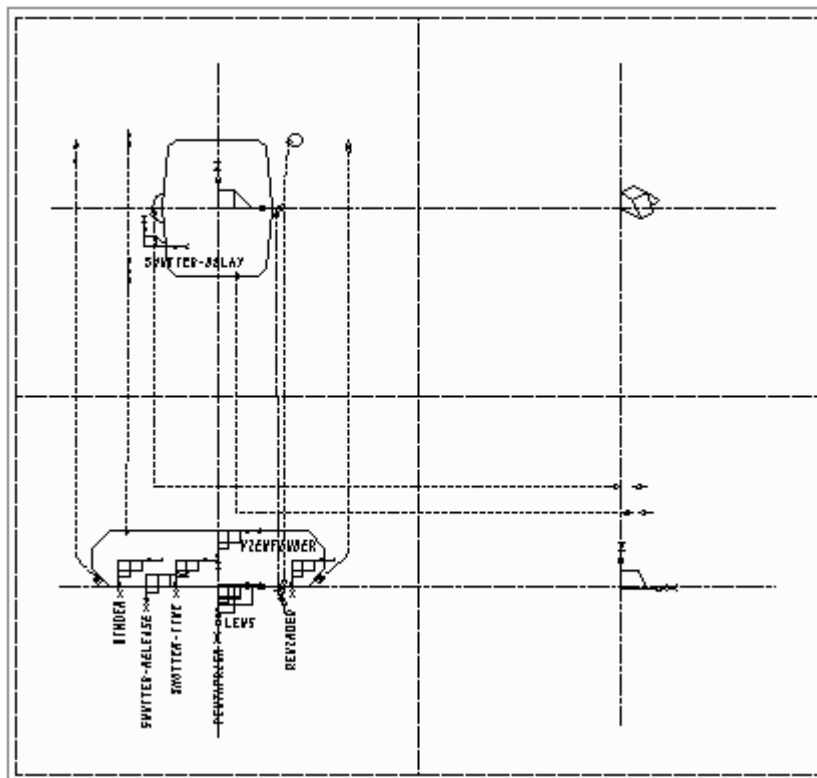


Figure 208 Enlarged View Of the XY Viewbox On the Assembly Sheet

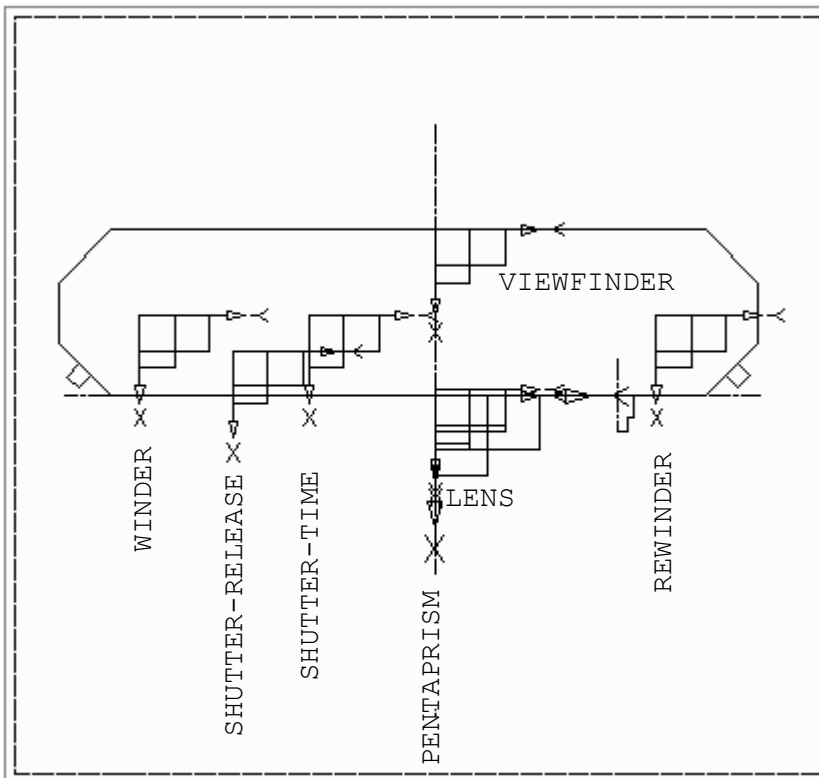


Figure 209 through Figure 216 show the definitions of the instanced camera components. The table below lists the model generators used to create those definitions.

Definition	Model generator	Figure number
Lens	Volume of Revolution	Figure 209
Ridges on lens	Multiple Solid Sweep	Figure 209
Pentaprism	Ruled Surface, Solid Sweep	Figure 210
Hot shoe	Solid Sweep	Figure 210
Winder	Solid Sweep, Volume of Revolution	Figure 211
Rewinder	Solid Sweep, Boolean Addition	Figure 212
Exposure delay lever	Solid Sweep	Figure 213
Shutter release button	Solid Sweep	Figure 214
Exposure time dial	Solid Sweep	Figure 215
View finder	Volume of Revolution	Figure 216

Figure 209 The Camera Lens Definition

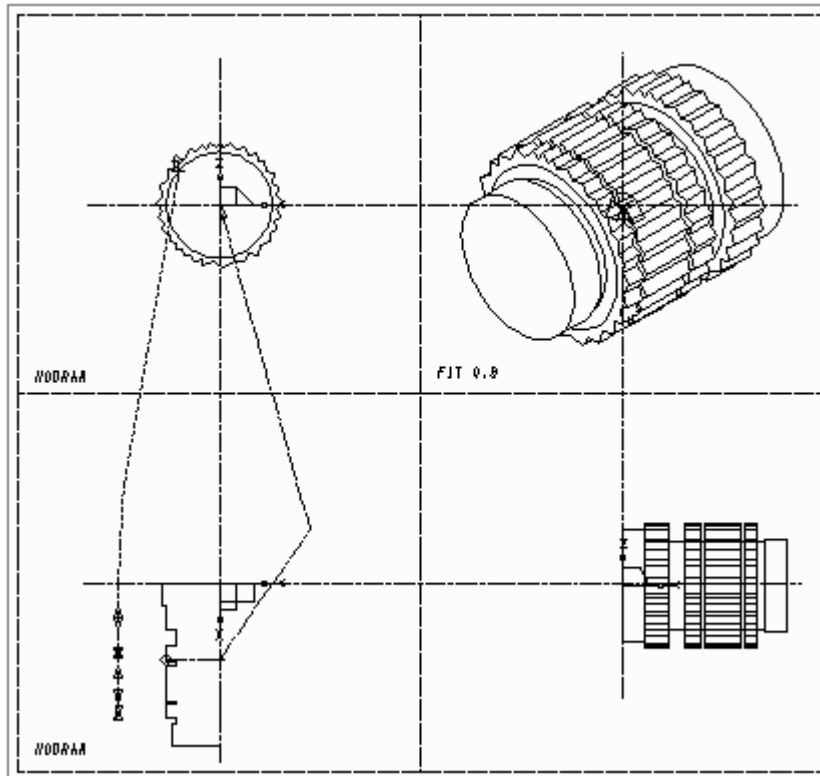


Figure 210 The Pentaprism and Hot Shoe Definition

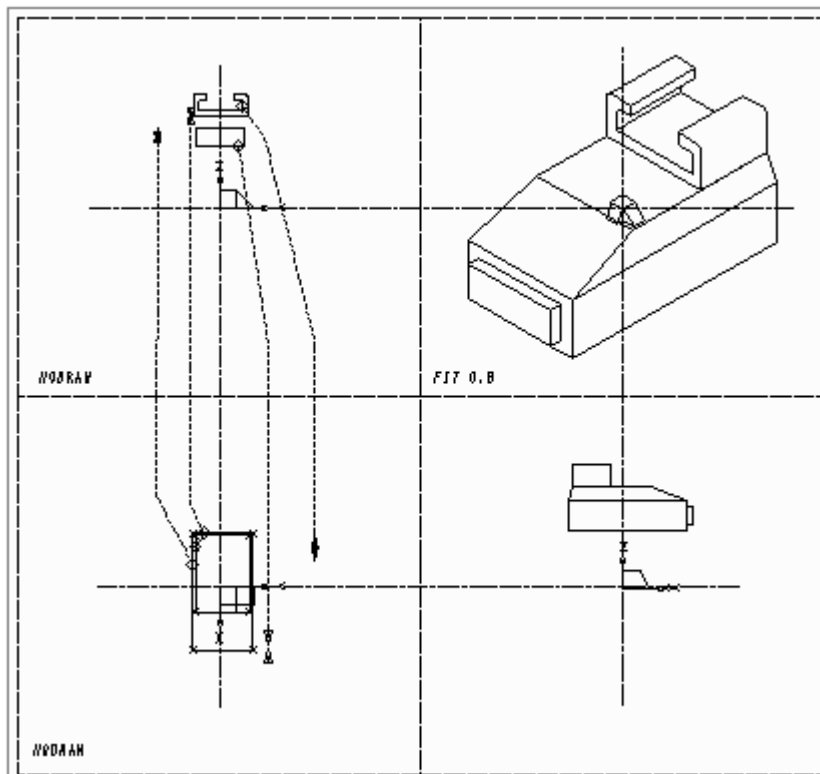


Figure 211 The Winder Definition

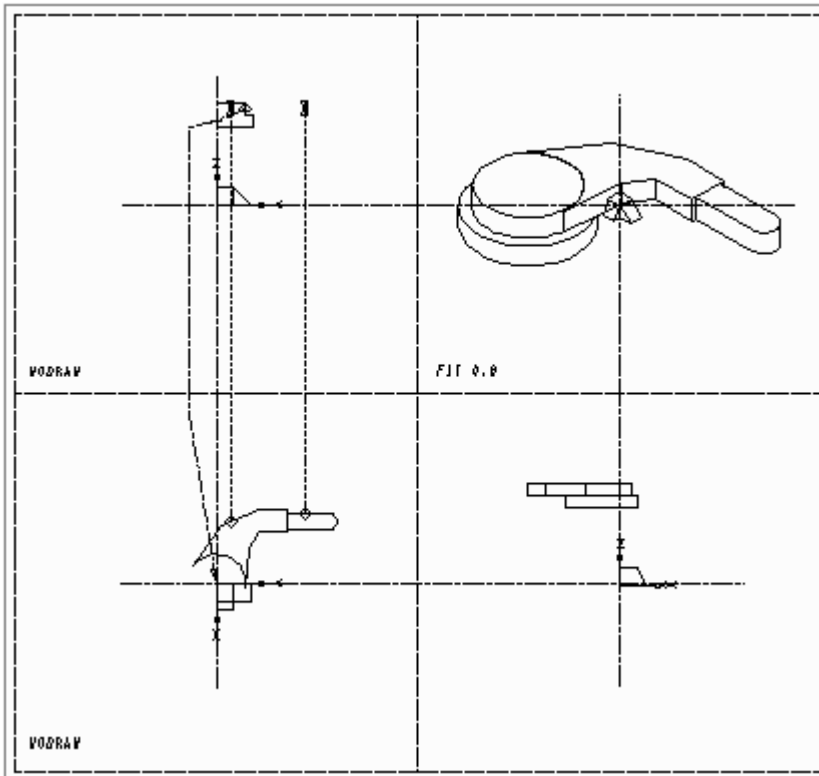


Figure 212 The Rewinder Definition

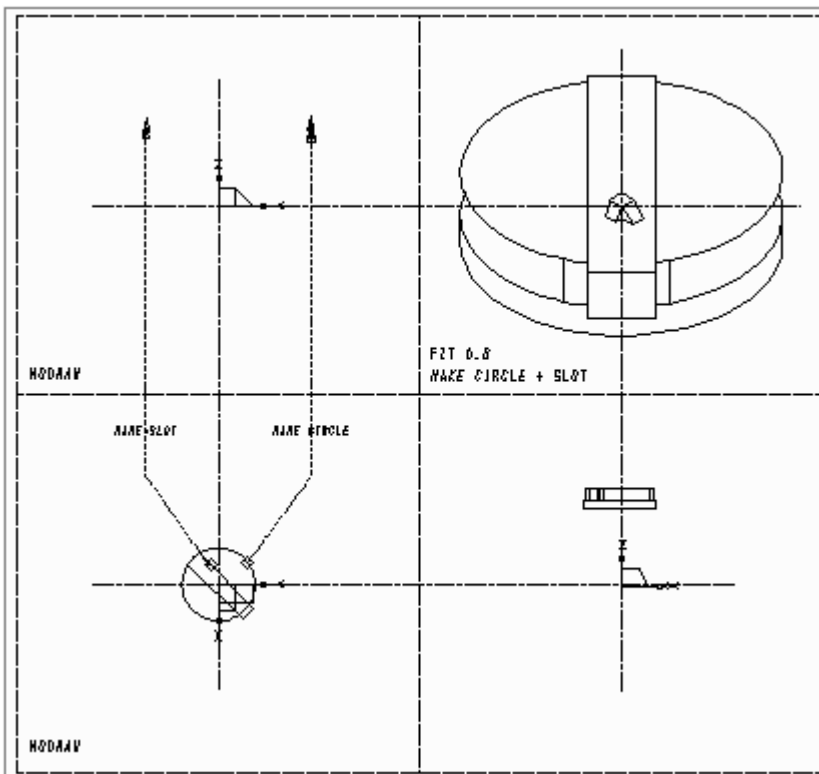


Figure 213 The Exposure Delay Lever Definition

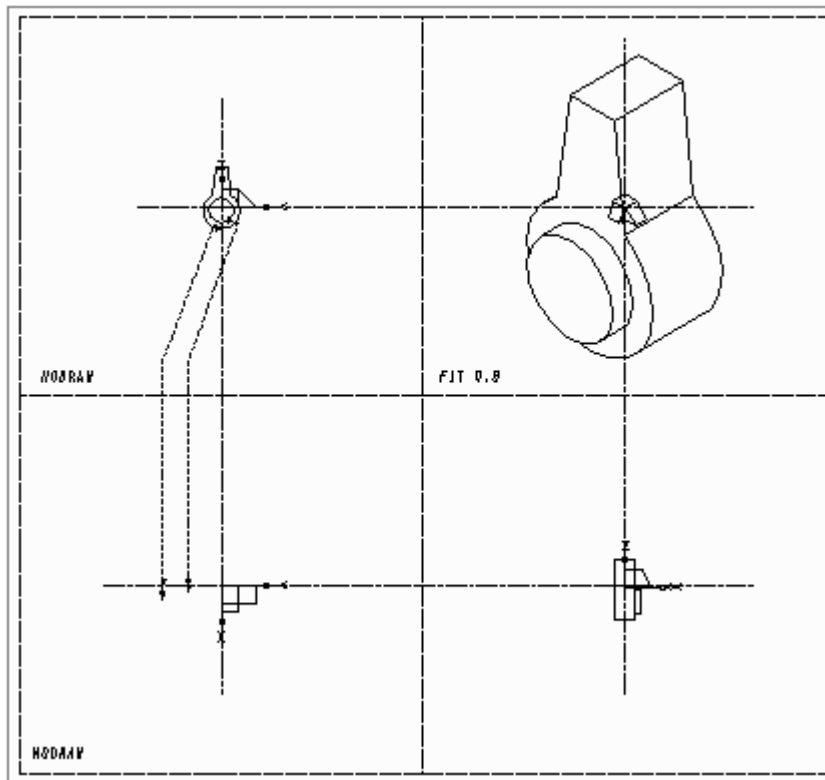


Figure 214 The Shutter Release Button Definition

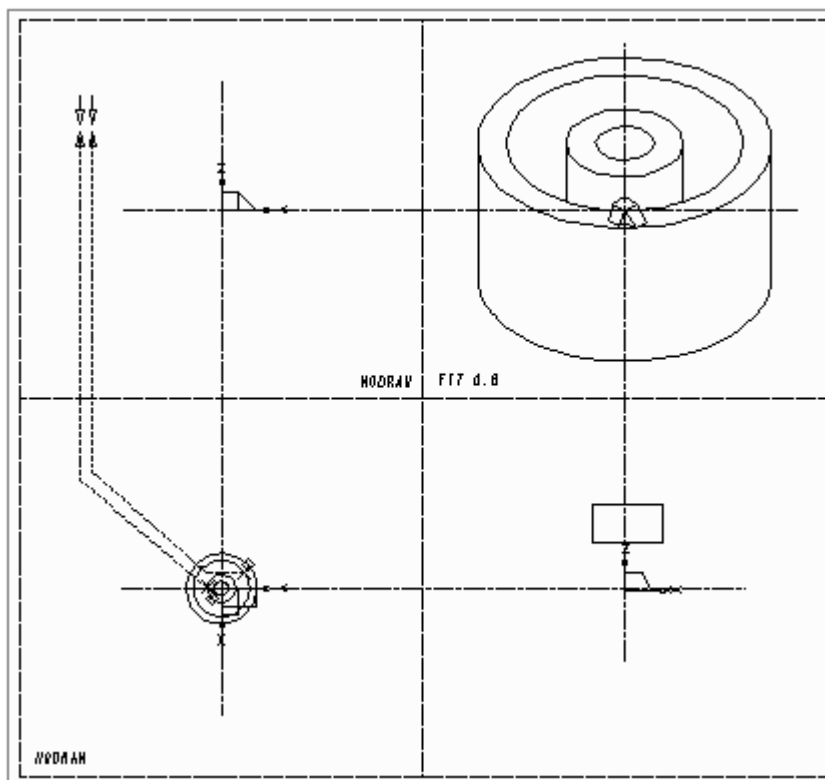


Figure 215 The Exposure Time Dial Definition

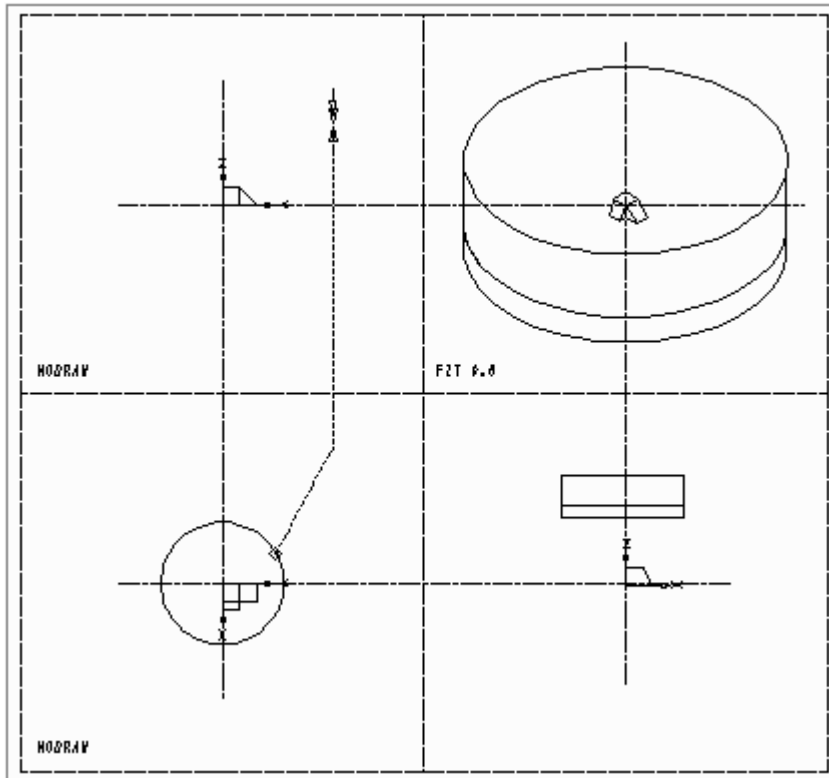
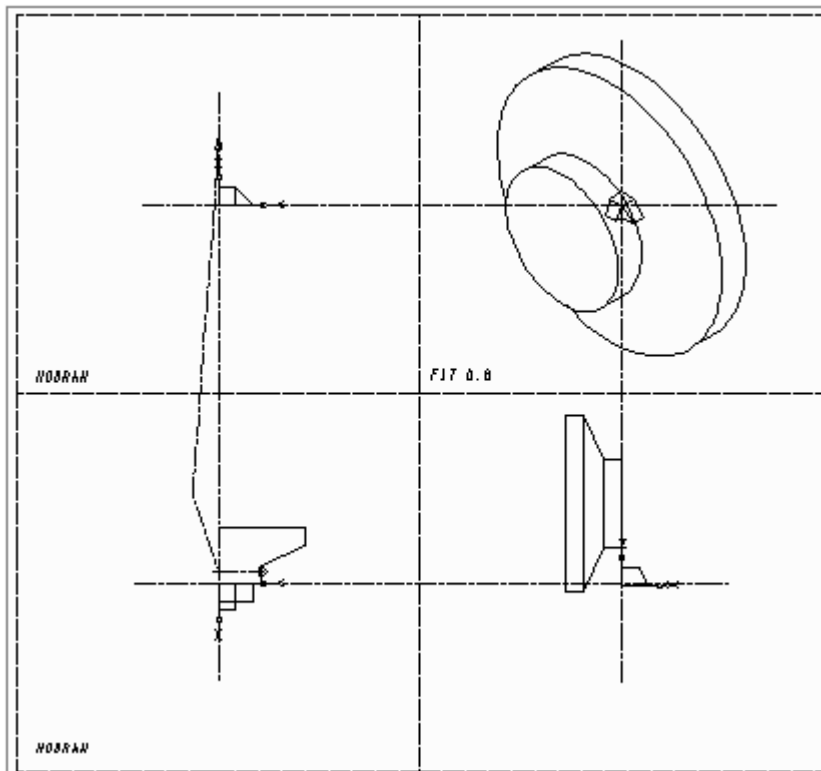


Figure 216 The Viewfinder Definition



GENERAL MODELER COMMANDS

This chapter describes Modeler commands that you can use with any of the model generators described in the previous chapters. The commands described allow you to, for example, specify the chord tolerance on curves, name objects, specify model properties (uncommitted properties) for use by other programs, and switch off model file compression.

- General 282
- Specifying the Chord Tolerance 283
- Showing Patch Boundaries and Tile Edges 286
- Naming Objects 287
- Specifying the Detail Level 288
- Sending Patch Data to a Model File 289
- Entering Properties for Other Programs 290
- Compressing a Model File 292

General

A Modeler command can be added to a sheet in one of the following ways:

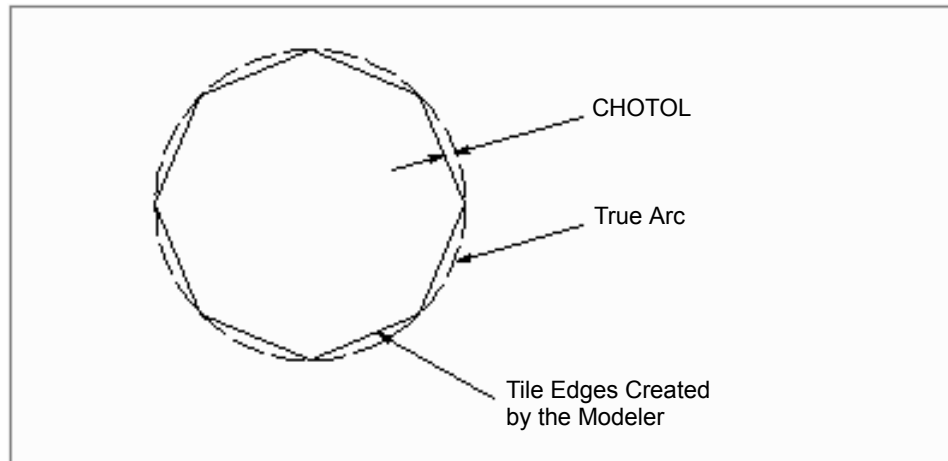
- As 3D sheet level attribute, alternatively as TMS-text of style `Model Specification Text` anywhere on the 3D sheet.
- As 3D linkline attribute, alternatively as TMG-text of style `Modeller Text ass. by Geometry` with datum positioned on a link line.
- As SMI-text of style `3D Instance Model Text` within an instance group. Uncommitted properties can also be included within an instance group.

Please note: TMG text must be uniquely assigned to a single link line; therefore, a TMG text datum must not be placed at the intersection of two link lines.

Specifying the Chord Tolerance

A curved face on a model is represented as a series of tile edges approximating the shape of the curve. The closeness of this approximation is controlled by the chord tolerance. The chord tolerance is defined as the maximum deviation of the tile edges from the true curve, as shown in Figure 217.

Figure 217 Definition of CHOTOL



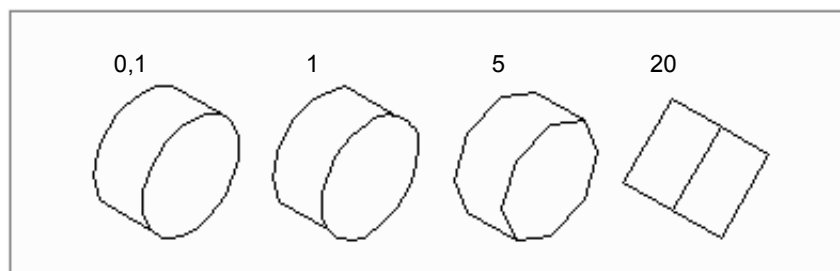
To specify the chord tolerance, use the command:

`CHOTOL tolerance_value`

where *tolerance_value* specifies the maximum deviation allowed in the current sheet units. For example, if metric units of measurement are used, the command `CHOTOL 1` sets the chord tolerance to 1 mm.

The lower the CHOTOL value is set, the closer will be the approximation to the true curve. Figure 218 shows the effect that changing the chord tolerance has on a model.

Figure 218 The Effect of the CHOTOL Value On Model Accuracy




The time taken to generate a model and produce a view is reduced by specifying a greater value for CHOTOL. This reduces the number of tile edges that have to be created by the Modeler. Therefore if the model contains a large number of curves, it may be advantageous to set a large chord tolerance during the initial stages of definition. Then a smaller chord tolerance can be used to produce the finished model. The default chord tolerance command is `CHOTOL 1.0 1`. The second argument is optional and is explained on page 285.


CHOTOL on the Sheet

There are different methods to add the `CHOTOL` command to the sheet:

Adding an Attribute

1. Open the 3D Attributes dialog using the 3D Sheet Attributes tool .
2. Enter the desired value in the `CHOTOL` field.
3. Apply the input using the `Apply` button, respectively apply and quit the dialog via `OK`.

Placing a TMS-Text

1. Choose the `Creates a TMS text` tool .
2. Enter the `CHOTOL` command followed by the desired value into the text input field.
The command is created as TMS-text of style `Model Specification Text` and is attached to the cursor.
3. Place it at any arbitrary position on the sheet outside of a viewbox.
It affects the whole sheet.

CHOTOL on the Link Line

The sheet chord tolerance can be overridden for a specified object, or group of objects, by adding the `CHOTOL` command as attribute to the link line or creating the command as a TMG-text of style `Modeller Text ass. by Geometry` and placing it on the link line that is used to create the object. (Please consider the note on [page 282](#).)

Improving Volume Calculation Accuracy Using CHOTOL

An additional argument can be given to the `CHOTOL` command to allow more accurate volume calculations to be performed:

```
CHOTOL tolerance_value refining_factor
```

The additional argument *refining_factor* specifies that an exact multiple of facets are to be created along a line.

For example, the command:

```
CHOTOL 1.1 2
```

would generate exactly twice as many facets as the command:

CHOTOL 1.0 1

or:

CHOTOL 1.1

This command can be added to the 3D sheet using a TMS-text of style `Model Specification Text` or attached to the link line of an object using a TMG-text of style `Modeller Text ass. by Geometry`. (Please consider the note on [page 282](#).)

Control of Curve Approximation Using CHOTOL

Use the command:

```
CHOTOL tolerance_value refining_factor
```

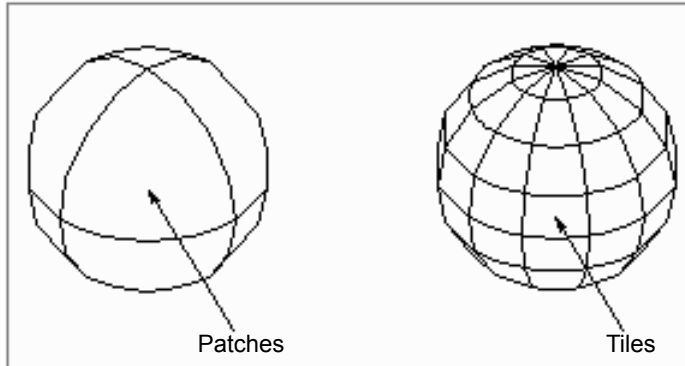
to increase the number of tile edges that are used to approximate arcs. Increasing the number of tiles reduces the likelihood of error if the curved faces of objects touch, or if there is a small gap between them which is less than the chord tolerance. Set the value of *refining_factor* to give the required increase.

Please note: The `FACET` command overrides the `CHOTOL` command. For details of `FACET` “[Meshed Surfaces](#)” on [page 191](#).

Showing Patch Boundaries and Tile Edges

A curved surface on a model is divided into a number of curved patches, and the shape of each patch is approximated by a number of flat tiles. This is shown in [Figure 219](#).

Figure 219 Patches and Tiles On the Model Surface



Whether or not the patch boundaries or tile edges are visible on the surface of a model is usually specified at the viewing stage but, if required, this can be done at the Modeler stage instead.

Patch boundaries are made visible by loading the Model Specification Text text (type TMS) `BOU VIS` onto the sheet.

Tile edges are made visible by loading the TMS-text `TIL VIS` (style Model Specification Text) onto the sheet. This command overrides the Viewer command `TIL INV` and can only be changed by remodeling.

By default, both boundaries and tiles are invisible (`BOU INV` and `TIL INV`).

If using instances, these commands can be issued as text of style `Modeller Text ass.` by `Set` (type SMI) and added to an instance group.

Naming Objects

By default, when a model is created from several objects, those objects lose their individual identities and are transformed into a new single object. But, by naming an object at the model generation stage, the identity of an object can be retained and therefore that object can be displayed separately at the Viewer stage. This is particularly useful for viewing assemblies piece by piece in the Model Viewer and for extracting the properties of an object with the Properties program.

You can name an object by adding the `NAME` command followed by the relevant value, e.g. `BLOCK`, as attribute on the link line or placing the TMG-text `NAME` of style `Modeller Text` `ass. by Geometry` on the link line of the object.

Please note: The commands used for the naming of objects ignore upper and lower case.

Alternatively to the command `NAME` you can add the attribute, respectively the TMG-text, `&object_name` to the link line to name the object, for example `&BLOCK`.
(Please consider the note on [page 282](#).)

For naming objects see also "[Boolean Operations](#)", "[Naming an Object](#)" on [page 214](#).

Specifying the Detail Level

Specific objects within a model can be assigned a detail level in the model file. This allows objects to be viewed separately by switching the detail levels on and off at the Viewer stage. An object can be created at any one of 256 detail levels (numbered 0 through 255) at the model generation stage. If you do not specify a detail level then all objects will be created at level 0.

To specify a detail level add the `DETAIL` attribute followed by the desired value, a number between 1 through 255, to the link line or place a TMG-text, e.g. `DETAIL 3`, of style `Modeller Text ass. by Geometry` on the link line of the object. (Please consider the note on [page 282](#).)

Sending Patch Data to a Model File

The surface of a model is represented by patches as described in the section “[Showing Patch Boundaries and Tile Edges](#)” on page 286. The Modeling System uses two types of patch equation to determine the surface of a model:

- Coons patch equations are used to model the surface of objects generated by the Meshed Surface and Duct generators
- Rational quadratic patch equations are used to model the surface of all other models generated by the Modeling system

These equations can be output to a model file using the command:

```
EQUATIONS ON
```

The patch equations are recognized and used by the MEDUSA4 data transfer programs MODASC, MODBIN, and MEDMIF. For further information see chapters “[Interfaces 1](#)” on page 401 and “[The Model File Translator](#)” on page 437.

The `EQUATIONS ON` or `EQUATIONS OFF` command can be specified on the 3D definition sheet as TMS-text of style `Model Specification Text`, or as a TMG-text of style `Modeller Text ass. by Geometry` attached to the link line of a model:

- As a text style `Model Specification Text`, the `EQUATIONS ON` command sends patch equation data to the model file for the entire model.
- As a text of style `Modeller Text ass. by Geometry`, the `EQUATIONS ON` command sends patch equation data to the model file for an individual object. (Please consider the note on [page 282](#).)

The `EQUATIONS OFF` command stops the output of patch equation data to the model file for an object or model. This is the default.

Please note: The MEDUSA4 Interpolator program can also output Rational and Coons patch data to a model file. For further information see chapter “[Text Driven Modeling](#)” on page 493.

Entering Properties for Other Programs

Commands that are used by other programs can be added to the model file. The command text added is ignored by the 3D system but can be used by other MEDUSA4 modules that access the model file, such as the Mass Properties program. These texts are known as **uncommitted properties**.

You can add uncommitted properties to a model file by attaching a TMG-text of style `Modeller Text ass. by Geometry` to the link line of an object on the 3D sheet with a segment probe. The first character of the text must be a plus sign. For example: `+SURF 3`
(Please consider the note on [page 282](#).)

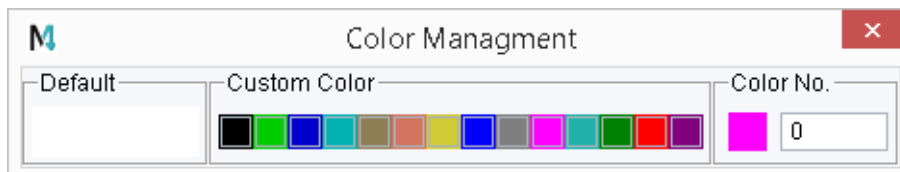
This command tells the Viewer to set up color number 3 and to assign it to a surface. The plus sign tells the Modeler to ignore the following text.

The `SURF` command is a commonly used uncommitted property which allows each object in a model to be assigned a different color, or different attribute.

There are several methods to add the `+SURF` command shown in the following sub-sections. In all cases the `Color Management` dialog is displayed. Two versions exist:

- For a standard project, e.g. the *master_project* in your installation path, the following dialog is shown:

Figure 220 Color Management Dialog - Standard Master Project





Please note: By default the `Color Management` dialog provides a certain number of colors which can be changed and extended arbitrarily by your Administrator. The colors are defined in the file `<medusa4>\med2d\m2d\src\colours.map`. For details see the “MEDUSA4 Administration Guide, chapter Administration, section Setting up of Standard Colors”.

Please note: Moving the cursor over the color buttons the `Color Management` dialog displays the number assigned to the single colors.



Using the Dashboard

1. Select the link line of the object, you want to give a new color, within the model definition.




In the Dashboard the button of the `+SURF` command is provided (see [Figure 31, “Link Line Dashboard” on page 50](#)).

2. Click on the button.
The Color Management dialog opens. It displays the palette of available colors.
3. Choose the required color by clicking on the relevant color button.
The dialog closes and the Dashboard displays the chosen color. If you want back the default color, open the Color Management dialog again and press Default.
4. To see the new color in the model, choose Model + Reconstruct  and then view the object with the Model Viewer using the button .

Using the Line Properties Dialog

1. Select the link line of the object, you want to give a new color, within the model definition.
2. Open the Line Properties dialog via the button in the Dashboard or choose Properties from the popup menu.
The Line Properties dialog is displayed showing the +SURF command button.
3. Click on the +SURF button to call up the Color Management dialog.
4. Choose the required color by clicking on the relevant color button.
The dialog closes and the Line Properties dialog displays the chosen color. If you want back the default color, open the Color Management dialog again and press Default.
5. To see the new color in the model, choose Model + Reconstruct  and then view the object with the Model Viewer using the button .

Creating Text by using the TMG-Text Tool

1. Choose the Creates a TMG text tool  from the Tools tool group.
A text input field opens in the Dashboard.
2. Enter the +SURF command followed by the number of the desired color. That assumes that you know the numbers assigned to the colors.
The text is attached to the cursor.
3. Place the TMG-text on the link line. (Please consider the note on [page 282.](#))
4. Exit the tool.
5. To see the new color in the model, choose Model + Reconstruct  and then view the object with the Model Viewer using the button .

Compressing a Model File

The Modeler can be instructed to perform a compression operation on model files as they are generated so that they use less disk space than they otherwise would have done. This feature gives a useful saving of disk space, especially for complex models. The compressed model files are automatically expanded when read.

This compression and expansion is completely transparent and requires no attention, but note that there is a time penalty involved in these operations which you may want to avoid. This situation may arise when you are remodeling frequently, such as during a design phase, where economy of disk usage is less important than modeling speed. If compression is required, then switch on compression with the command:

```
COMPRESSION ON
```

This command is add as sheet level attribute or placed as TMS-text of style `Model Specification Text` on the definition drawing.

The next time this sheet is modeled the model file will be compressed. When you have finished modeling a sheet for a time, use the `COMPRESSION ON` command just before the last modeling operation so that the model file will be compressed.

There is also a separate MEDUTIL utility called MODCOMP which you can use to compress or expand model files independently of the Modeler. This utility is useful if you prefer to handle model file compression and expansion as a separate task from modeling. You may also want to use a naming convention for files (or directories) in order to distinguish between compressed and uncompressed model files. See chapter [“Model File Compression” on page 517](#), for details of MODCOMP.

VIEWING THE MODEL

This chapter describes the commands which are used to produce a 3D view of a model.

- Introduction..... 294
- Viewing Concepts..... 294
- Changing the Type of Projection of a View..... 299
- Sizing the Image and Setting the Degree of Perspective 300
- Moving the Viewpoint 305
- Moving the Aiming Point..... 309
- Defining Oblique Views Using Viewprims..... 310
- Defining Oblique Views Using the Creates an Oblique View Tools 313
- Sectioning the Model..... 318
- The VIEWTOL Command..... 334
- Viewing Commands..... 335

Introduction

You can define views to be perspective or trimetric as required. For this viewing commands are used which can be applied as 3D attributes (see "3D-Attributes", "Viewer" on page 42) or as texts of style `View Specification Text` (type TVS) on the model definition sheet.

Specifying 3D views provides a permanent record of a particular view (or set of views) that is stored with the sheet, and which can be copied to another sheet or plotted out with the model definition sheet, as required. You can specify as many views as you want on the sheet as long as each view is contained in its own viewbox with one viewprim.

In sheet viewing, the system starts searching for commands at the top of the sheet hierarchy and progresses to the bottom. Commands are executed in the order they are found. This means that the order in which commands are added to the sheet can affect the resulting image.

This chapter shows you how to define 3D views with the help of texts on a MEDUSA4 2D sheet.

Viewing Concepts

The Viewer allows 3D views of a model to be created in a way which is very similar to taking photographs. As with photography, the factors which affect the size and orientation of the final 3D view can all be set. The main difference is that because the Viewer parameters are set with coordinates, lengths and angles in MEDUSA4 3D space, the viewing possibilities are much wider than is available in photography. For example; a model of a building can be viewed from underneath and an engineering component can be sectioned at any plane to view the internal structure.

As with photography, the size and orientation of the view image can be varied as well as the degree of perspective (in photographic terms for example from fisheye to telephoto). In addition, the Viewer produces trimetric views as easily as perspective views; a feature not available in photography.

In the following sub-sections the concepts and terminology used in the Viewer are discussed.

Viewing Considerations

When specifying a view of a model, there are normally three major factors of interest:

- The type of projection to be used (trimetric or perspective)
- The size of the image of the model
- The orientation of the image relative to the sheet viewbox

Type of Projection

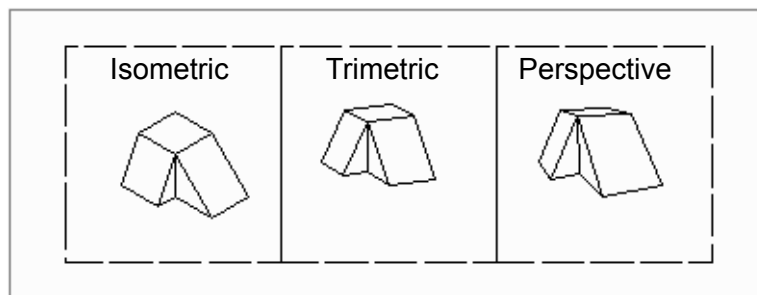
The Viewer either produces a three point perspective view of the model, as would a camera taking photographs of the real object, or an axonometric view (in general as a trimetric view, although the default is isometric).

The axonometric view is the traditional projection used for 3D representations of engineering components (normally limited to the isometric form for simplicity).

Perspective projection has traditionally been used in architectural work. However, because the Viewer produces one type of projection as easily as the other, it is worth considering using perspective for engineering applications to take advantage of the more realistic result.

Figure 221 shows examples of the projection possibilities.

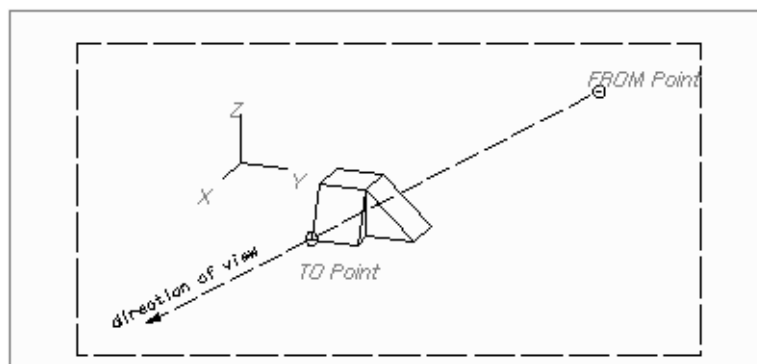
Figure 221 Types of Projection



The Viewpoint and Aiming Point

In the Viewer, the viewpoint (equivalent to the position of the camera in photography) is known as the FROM point. Similarly, the point at which the camera lens is aiming (normally in the vicinity of the object) is known as the TO point. These points are defined as 3D coordinates. The direction of view is along a line drawn through these points, as shown in Figure 222 below.

Figure 222 The FROM and TO Points



The image that you see according to the positions of the `FROM` and/or `TO` point to each other, may be changed in any order by using one of the many commands which move the `FROM` and/or `TO` point. For details see [“Moving the Viewpoint” on page 305](#) and [“Moving the Aiming Point” on page 309](#).

Size of the Image

An image is drawn in a viewbox at actual size by default, allowing for the effect of sheet scaling. This may be acceptable for your requirements but in some cases it could be better to change the size. There are many commands which allow this to be done, and the type of projection used for that view is an important factor in the choice of which command to use. For example, with perspective projection, the image size will be changed by any command which changes the viewpoint to aiming point distance. With a trimetric view, the same commands will have no effect on image size.

There are scaling commands provided for both projection types. A point to note is that these commands show the correct spatial relationship between the viewing point (the 3D prim datum) and the model.

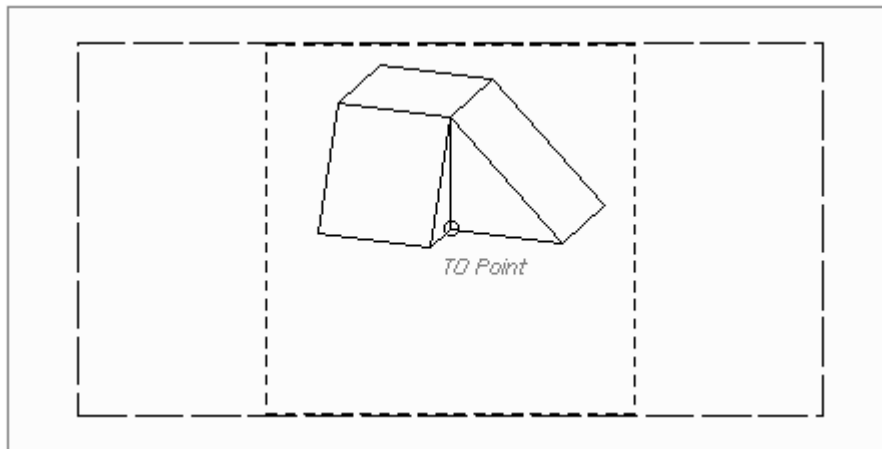
A very useful command for trimetric views is `FIT` which allows you to work without having to consider scaling factors at all, although the relationship between the viewing point and the model is not shown correctly. Note that this command will not operate on perspective views.

The image can always be sized satisfactorily in a viewbox, but it may take some experimentation to get just the size required. This is especially true for perspective projection. See [“Sizing the Image and Setting the Degree of Perspective” on page 300](#) for a description of those commands which affect the size.

Windows and Viewboxes

In sheet viewing, the image is drawn in a viewbox at full size, after taking into account sheet scaling (and foreshortening, if the view is trimetric or perspective). This means that the image can be badly sized for the viewbox, being either too large or too small. To correct this, you can define an imaginary square in the plane of the `TO` point, perpendicular to the line of view, which will then be automatically fitted to the viewbox with the `TO` point in the center (see [Figure 223](#)). This square is called a **window** and is analogous to the field of view of the camera lens at the object. This is a very useful feature which makes it easy to draw the model at a reasonable size in a viewbox at the first attempt.

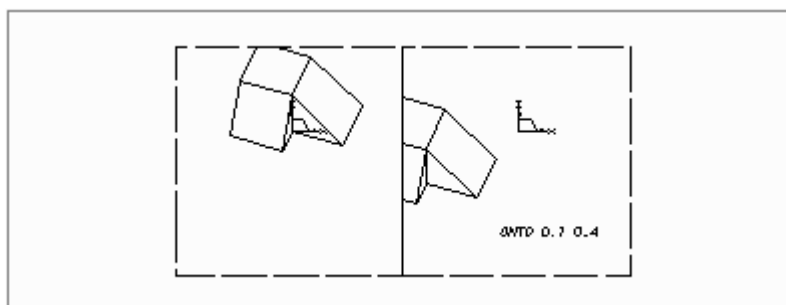
Figure 223 A Window in a Viewbox



The ONTO Point

The point at which the `TO` point is located in the viewbox is called the `ONTO` point. By default the `ONTO` point is the datum of the viewprim. The `ONTO` point can be positioned anywhere in the viewbox with the `ONTO` command (see [Figure 224](#)).

Figure 224 The Effect of the `ONTO` Command



The Upvector

The axis which is vertical in a viewbox is the projection of an upvector in that viewbox. By default this is set to be the Z-axis, but you can easily redefine it to be the X or Y-axis if required with the `UPVEC` command. For example:

```
UPVEC 0 0 1
```

defines the Z-axis to be the upvector (this is the default)

```
UPVEC 1 0 0
```

defines the X-axis to be the upvector

```
UPVEC 0 1 0
```

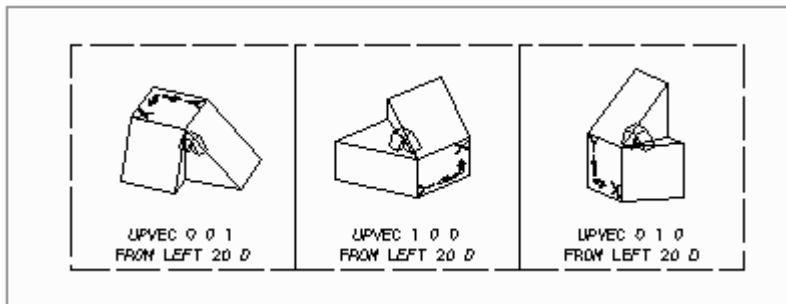
defines the Y-axis to be the upvector

By definition, the upvector is not allowed to be parallel to the FROM direction, that is, the angle between the two vectors must not be less than 0.5 degrees. If it is, the following error message is displayed:

```
Upvector not valid and the upvector original values are retained.
```

Figure 225 shows the effect of several upvector commands followed by a FROM command.

Figure 225 The Effect of Redefining the Upvector on a FROM Command



Rotating the Image

The image of the model can be rotated around the ONTO point by the command ANGLE. The rotation is in the plane of the view in the counterclockwise direction. The angle is measured in absolute mode between the current upvector and the vertical axis (Y).

For example:

```
ANGLE 60
```

rotates the image 60 degrees counterclockwise.

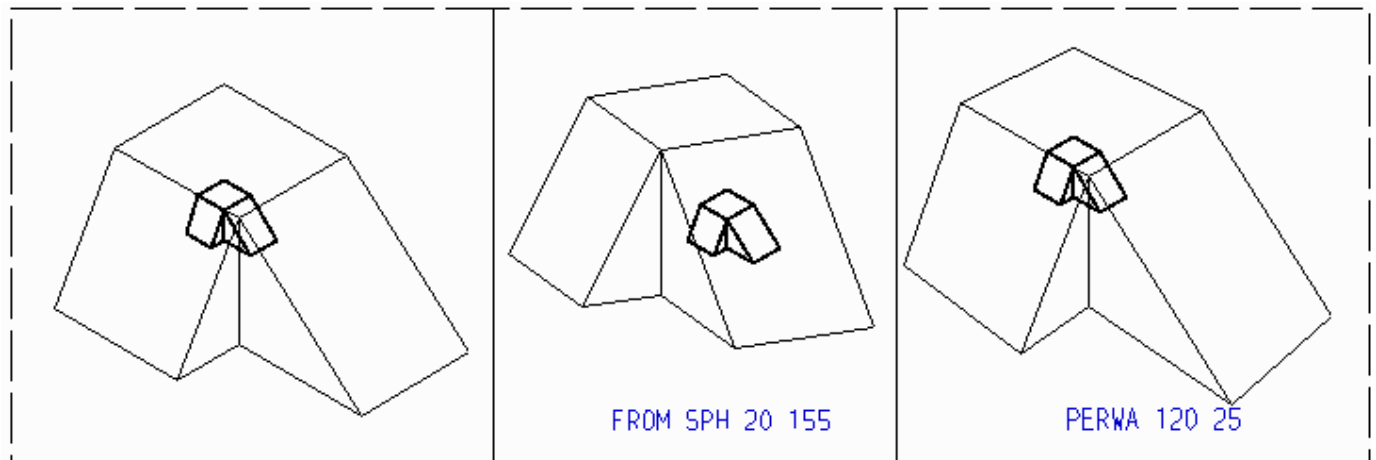
Changing the Type of Projection of a View

The default projection used in the top right viewbox on standard 3D sheets is defined as isometric by the viewprim `Isometric view datum prim 1 (IV1)` in that viewbox. This view can be changed to another type of projection by:

- Moving the `TO` and/or `FROM` points to produce a trimetric view
- Using a `PER` command to produce a perspective view

Figure 226 shows examples of changing the projection.

Figure 226 Changing an Isometric View to a Trimetric and a Perspective View



The `FROM` and `TO` commands are described in “Moving the Viewpoint” on page 305 through page 309. The `PER` commands are described in “Perspective Viewing” on page 301 et sqq.

Sizing the Image and Setting the Degree of Perspective

Changing the size of the image is straightforward with a trimetric type view. There are just three commands which cover all requirements, see ["Trimetric Viewing"](#) below.

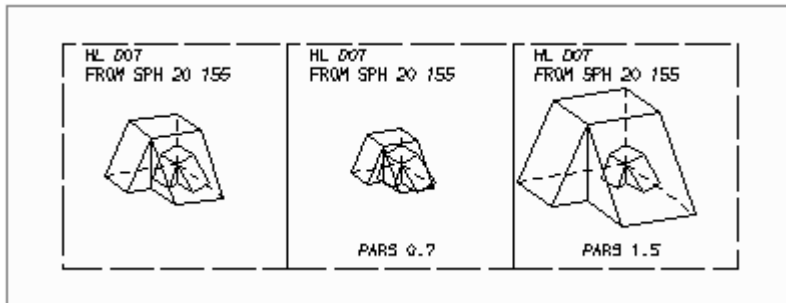
On the other hand, changing the size of a perspective image is related to setting the degree of perspective, and a command designed primarily for one of these aspects will almost certainly affect the other. This dependency is considered either explicitly in the command syntax, as in `PERWA length angle` or implicitly, as in `PERS factor`. For details see ["Perspective Viewing"](#) on page 301.

Trimetric Viewing

`PARS factor`

sizes the default image by the given *factor*. The effect of the `PARS` command is shown in [Figure 227](#).

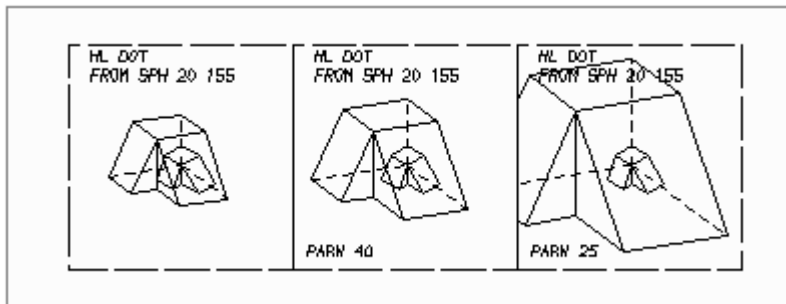
Figure 227 The Effect of the PARS Command



`PARW length`

sizes the image to the given window length. This is a good command to use when starting a trimetric view because it is based on well known data (that is, the size of the model).

Figure 228 The Effect of the PARW Command



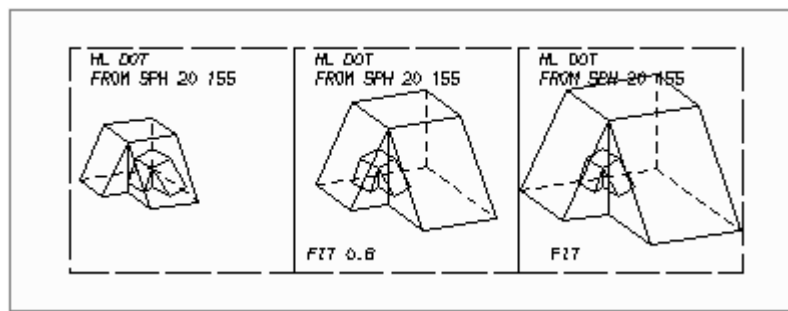
FIT scale_factor

fits the image to the viewbox. If the optional argument *scale_factor* is not used, the image will be sized to 0.98 of the limiting dimension.

The limiting dimension is that viewbox dimension (height or width) which would first cause the image to be clipped by the viewbox boundary as the image expands from the center of the viewbox. For this reason it limits the size of an unclipped image. It is not necessarily the smallest viewbox dimension.

If *scale_factor* is used then the image will be sized to the defined fraction of the limiting viewbox dimension. This is certainly the easiest trimetric sizing command to use but note that it does not show the correct spatial relationship between model and viewprim (the default TO point). This effect can be seen in [Figure 229](#) below. It is not usually a disadvantage but it is a fact which should be remembered.

Figure 229 The Effect of the FIT Command



Please note: The value of *scale_factor* can be greater than one, in which case the image will be clipped by the viewbox boundary.

Perspective Viewing

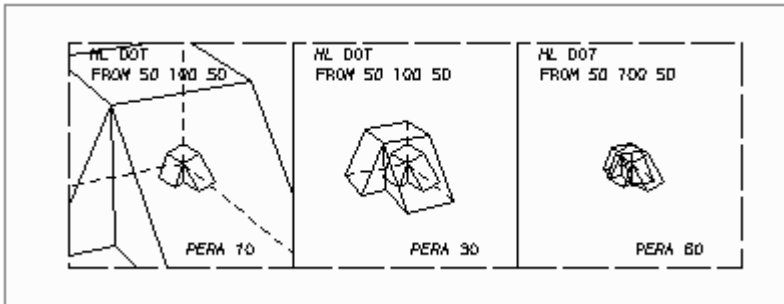
PERA angle

Explicitly defines the perspective angle. This is analogous to zooming with a camera lens. The most realistic results are obtained by using angles between 25 and 45 degrees. Angles outside this range produce noticeable distortion on normal sized images.

For example, increasing the angle makes objects in the field of view appear to recede, ultimately producing a fisheye effect. Conversely, decreasing the angle makes objects in the field of view appear to advance, leading to a telephoto effect, and which becomes a trimetric view at an angle of zero.

The effect of perspective angle variation is shown in [Figure 230](#).

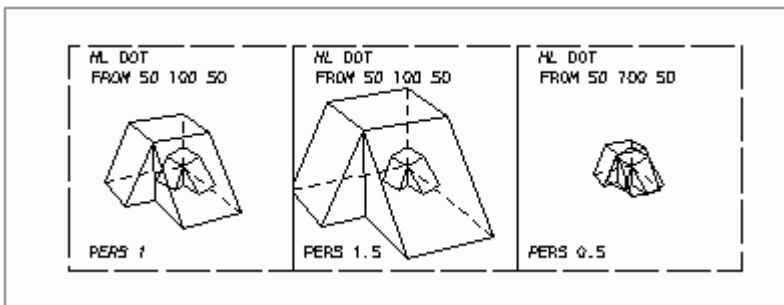
Figure 230 The Effect of the PERA Command



PERA factor

sizes the image by apparently changing the dimensions of the object by the scale factor given. **PERA** always works from the current **FROM** point, and so this should be set as required before you use this command, otherwise you will get a meaningless image. The effect of scale factor variation is shown in [Figure 231](#).

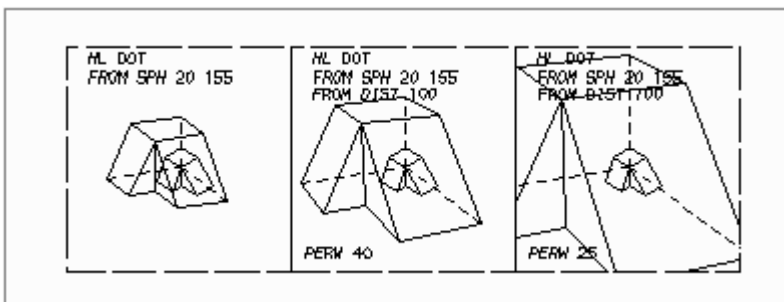
Figure 231 The Effect of the PERS Command



PERW length

sizes the image to the given window length. Perspective angle is changed with this command. **PERW** always works from the current **FROM** point, and so this should be set as required before you use this command, otherwise you will get a meaningless image. The effect of changing the perspective angle is shown in [Figure 232](#).

Figure 232 The Effect of the PERW Command

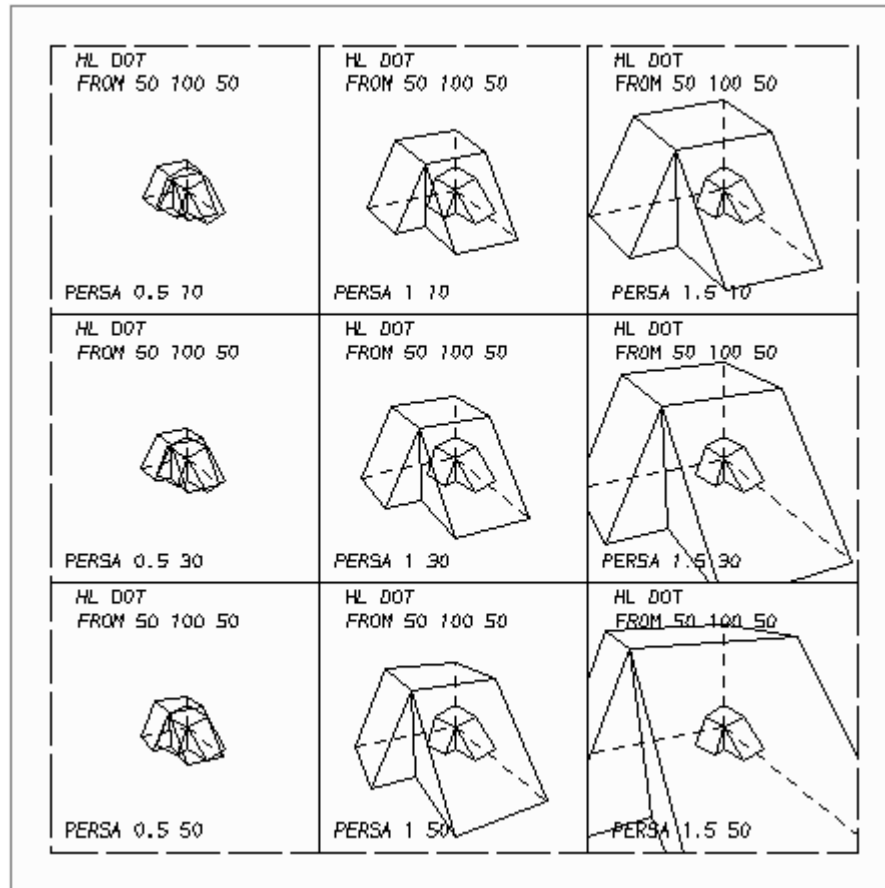


PERSA factor angle

sizes the image by apparently changing the dimensions of the object by the factor, while explicitly defining the perspective angle.

The effect of this command is shown in [Figure 233](#) below, which shows factor varying left to right, and angle varying top to bottom.

Figure 233 The Effect of the **PERSA** Command

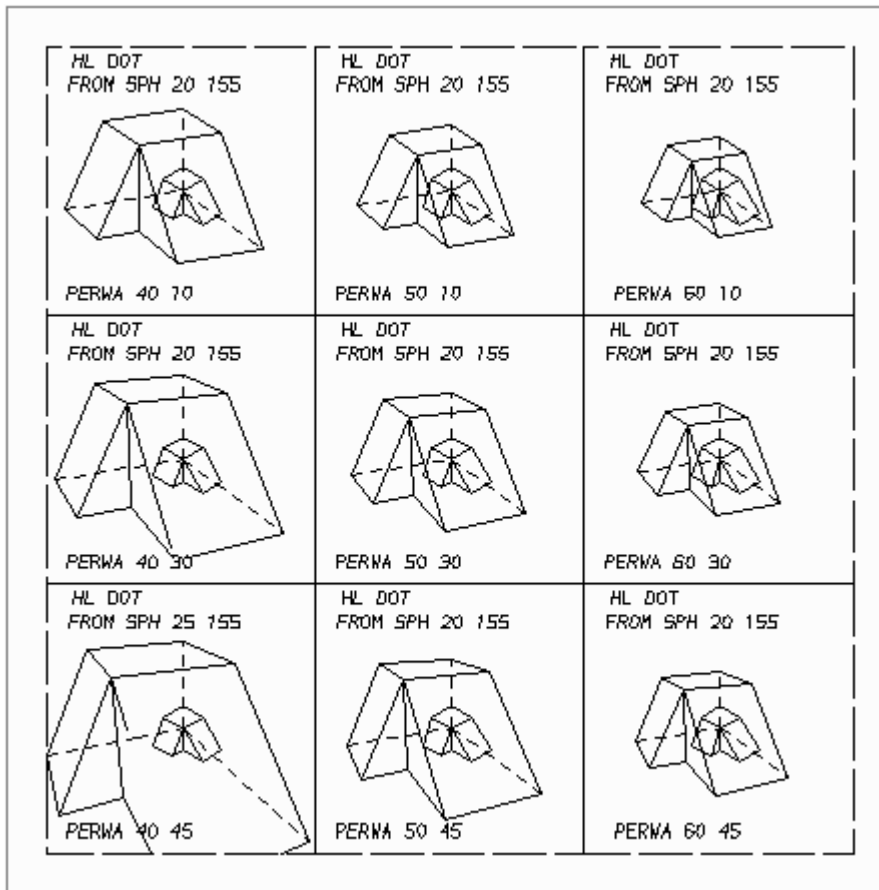


PERSA length angle

Sizes the image to the given window length at the **TO** point plane while explicitly defining the perspective angle. This is probably the best command to use when starting a perspective view because both parameters are based on known data (the size of the model and the degree of perspective required).

The effect of this command is shown in [Figure 234](#) below, which shows length varying left to right, and angle varying top to bottom.

Figure 234 The Effects of the PERWA Command



Moving the Viewpoint

There are many commands designed to move the viewpoint to the required position. They are known as the `FROM` commands. All except one define a new position relative to the current viewpoint or aiming point position.

Please note: The result of a `FROM`-command might be different in trimetric and perspective views.

Moving the FROM Point Absolutely

`FROM x y z`

defines the new `FROM` point position in absolute coordinates in the 3D system.

Example:

```
FROM 200 200 150
```

Please note: In isometric views the values are not absolute but relative to each other. So for the example above the values `2 2 1.5` show the same result as `200 200 150` in an isometric view.

Moving the FROM Point Relative to the Current TO Point

The commands listed in this subsection move the `FROM` point to a defined position relative to the current `TO` point.

`FROM TO x y z`

The arguments `x`, `y` and `z` are distances from the `TO` point (in the X, Y, and Z-directions). For example:

```
FROM TO 1000 -200 500
```

`FROM DIST distance`

Moves the `FROM` point along the viewline so that it is the defined distance from the `TO` point. For example:

```
FROM DIST 700
```

Please note: This command has a visible effect only in perspective views.

`FROM SPH latitude longitude`

Positions the `FROM` point according to the spherical coordinate system shown in

Figure 236. The range for *latitude* is -90 through 90 degrees, and for *longitude* it is -180 through 180 degrees. For example:

```
FROM SPH 35 -110
```

Figure 235 Examples of the FROM SPH Command

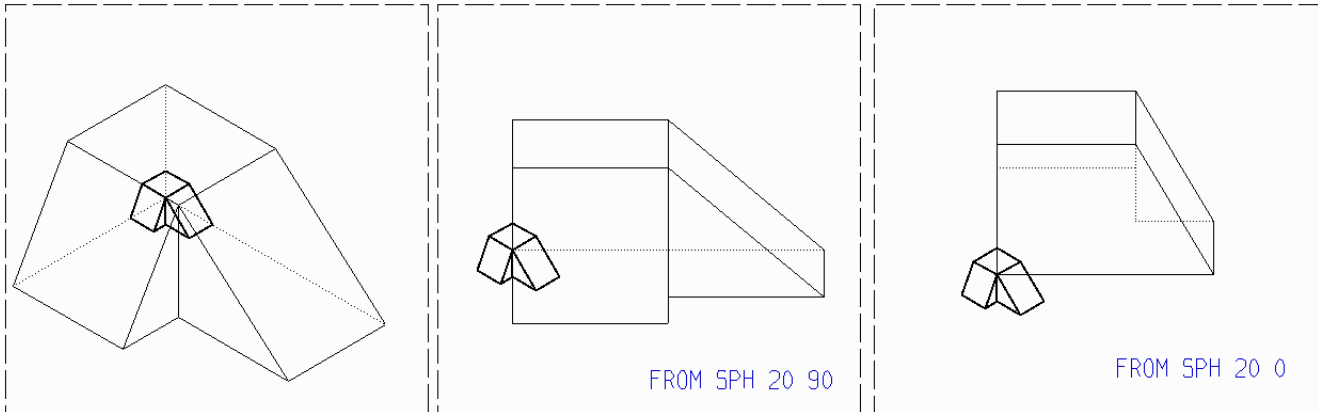
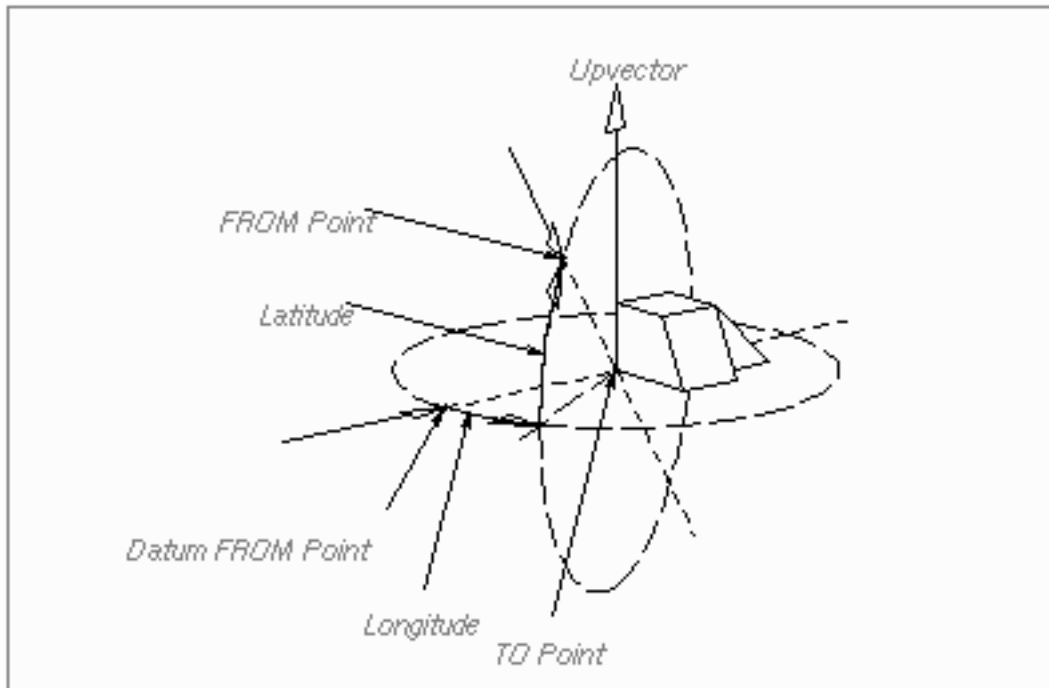


Figure 236 The Spherical Coordinate System



```
FROM DIR x y z
```

Specifies the viewline direction. The arguments x , y and z are distances from the TO point (in the X, Y, and Z-directions). For example:

```
FROM DIR 10 20 5
```

Please note: This command does not move the FROM point.

Moving the FROM Point Relative to the Current FROM Point

The commands listed in this subsection move the FROM point to a defined position relative to the current FROM point.

FROM FROM *x y z*

Three arguments *x*, *y* and *z* are distances from the FROM point (in the X, Y, and Z-directions). For example:

FROM FROM 50 90 -20

FROM DIST DIST *distance*

Moves the FROM point along the viewline by the defined *distance* from the current FROM point. For example:

FROM DIST DIST 75

Please note: This command has a visible effect only in perspective views.

FROM LEFT (RIGHT, UP, DOWN) *angle* D

Repositions the FROM point relative to the TO point according to the convention shown in [Figure 238](#). The argument *angle* is the angle in degrees by which the FROM point is rotated. For example:

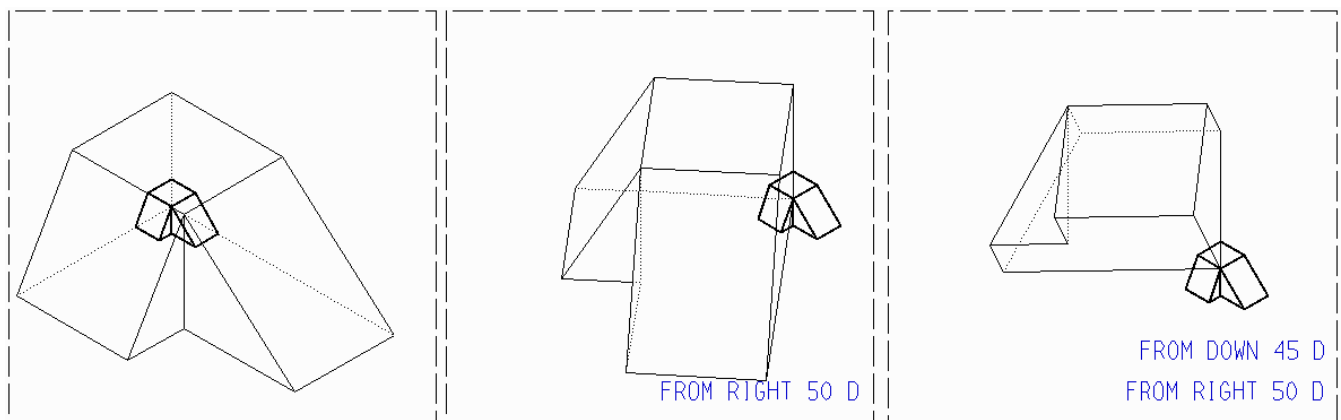
FROM LEFT 45 D

FROM UP 15 D

FROM RIGHT 30 D

FROM DOWN 10 D

Figure 237 Examples of the FROM RIGHT/DOWN Angle Commands



FROM LEFT (RIGHT, UP, DOWN) *distance*

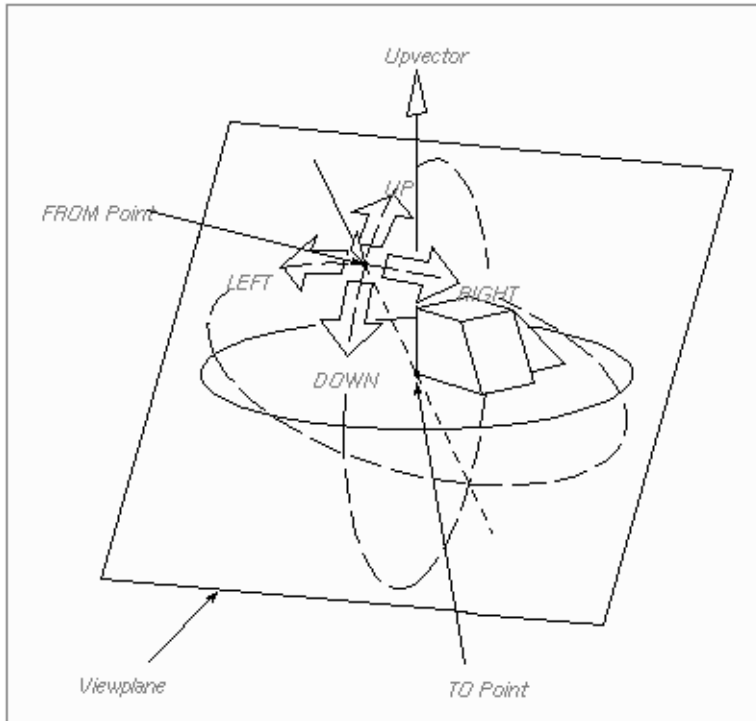
Repositions the FROM point relative to the TO point according to the convention shown in [Figure 238](#). The argument *distance* is the chordal distance that the FROM point is moved. For example:

FROM LEFT 100

FROM UP 5

FROM RIGHT 20
FROM DOWN 50

Figure 238 The Convention for Moving the FROM Point



Moving the Aiming Point

There are commands which move the aiming point in exactly the same way as the commands which move the viewpoint. These are known as the `TO` commands. With these commands the `FROM` point is the fixed point relative to which the `TO` point is moved; an exact reversal of the situation with the `FROM` commands. Because of this similarity, the `TO` commands are only listed in this section. For details of usage, see the `FROM` commands in section [“Moving the Viewpoint” on page 305](#).

Moving the TO Point Absolutely

```
TO x y z
```

Moving the TO Point Relative to the Current FROM Point

```
TO FROM x y z  
TO DIST distance  
TO SPH latitude longitude  
TO DIR x y z
```

Moving the TO Point Relative to the Current TO Point

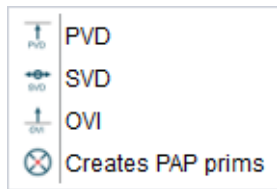
```
TO TO x y z  
TO DIST DIST distance  
TO LEFT angle D  
TO RIGHT angle D  
TO UP angle D  
TO DOWN angle D  
TO LEFT distance  
TO RIGHT distance  
TO UP distance  
TO DOWN distance
```

Defining Oblique Views Using Viewprims

Viewing a model in non-axial directions can be done by using the oblique viewprims shown in [Figure 239](#) instead of the `FROM` commands. These viewprims are used in complementary sets and work in conjunction with the orthogonal viewprims to define an oblique viewing direction and an oblique plane which is normal to that view direction.

Oblique viewprims are easier to use in some situations than the `FROM` commands as they give a direct, graphical impression of what is happening.

Figure 239 The Oblique Viewprims Tools



PVD - View direction 1
Style: Oblique View Primary Definition

SVD - View direction 2
Style: Oblique View Secondary Definition

OVI - View direction 3
Style: Oblique View (Real view)

Defining the View

A simple oblique view is defined by putting a PVD viewprim in a viewbox which contains an orthogonal viewprim. This oblique viewprim, in conjunction with the orthogonal viewprim in that viewbox, defines the primary view direction.

The `TO` point is the point of the arrow on the PVD symbol. This is relevant for perspective views and sectioned views, but not for trimetric views.

An SVD viewprim can be used in another orthogonal viewbox to define a secondary view direction if a compound view is required. Without an SVD prim, the position of the `TO` point in the third dimension is defined at zero (the orthogonal prim datum). With an SVD prim, the SVD datum (center of the circle) defines the `TO` point in the third dimension.

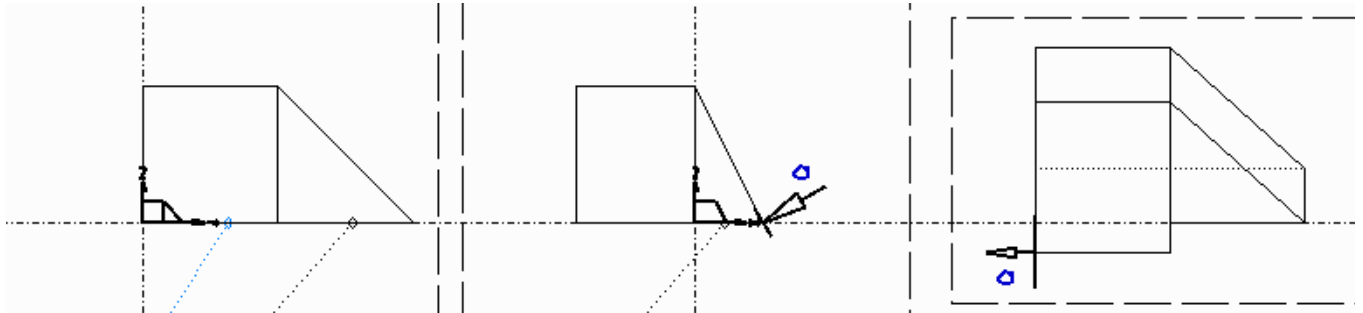
Drawing the View

The Viewer draws the defined oblique image in a viewbox which contains an OVI viewprim only. This is shown in the right hand viewbox in [Figure 240](#), which is an example of a simple oblique view defined by the PVD viewprim in the middle viewbox.

Note that the OVI prim, which defines the orientation of the oblique view in the viewbox, has been placed with the same orientation as the PVD prim. This ensures that the oblique view in

the right hand viewbox is drawn with conventional orientation relative to the main view in the middle hand viewbox.

Figure 240 A Simple Oblique View Definition



You can create any number of oblique views on a sheet. Corresponding sets of oblique prims are identified by a unique text label in each prim group. A dummy text with the name of the prim (which is `PVD`, `SVD` or `OVI`) and the style `Oblique View Label` is provided for this purpose, because this text can be edited easily for each prim.

[Figure 241](#) shows the effect of using the `SVD` viewprim to produce a compound view. The center viewbox shows the `SVD` prim added, but this does not change the oblique view as the orientation of the `PVD` plane is unchanged. The bottom right viewbox shows the effect of rotating the `SVD` viewprim. This tilts the `PVD` plane.

This method produces trimetric views by default (as shown above) but perspective views can be produced just as easily by including a perspective command in the `OVI` viewbox, as shown in [Figure 242](#).

Figure 241 Using the SVD Viewprim to Produce a Compound View

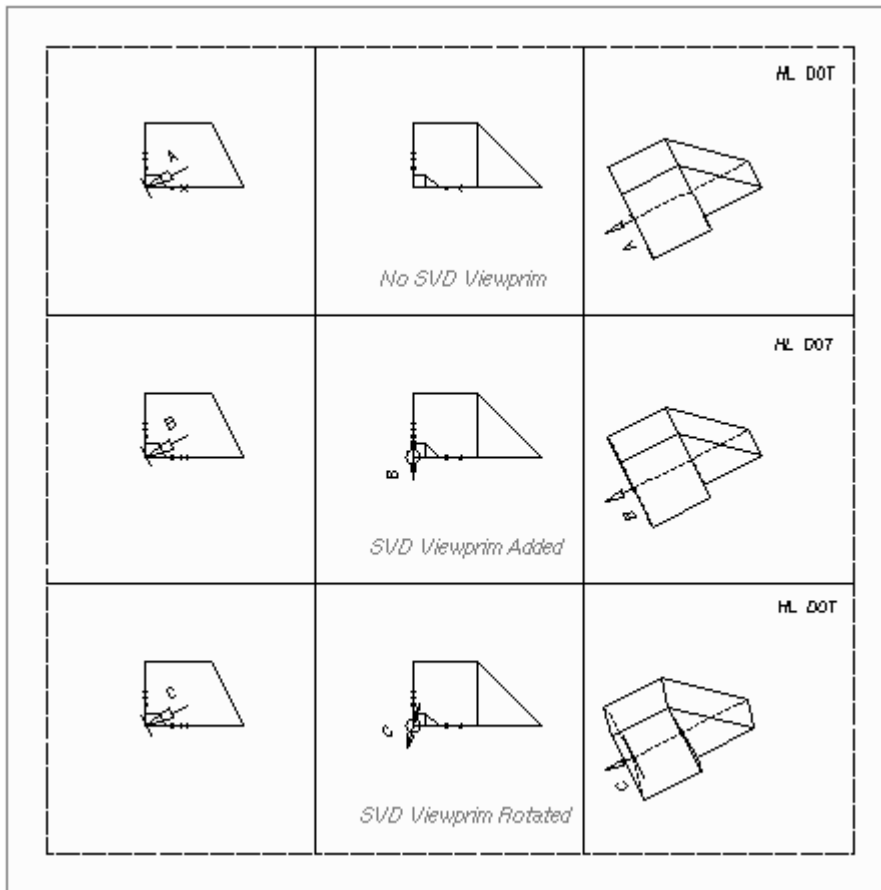
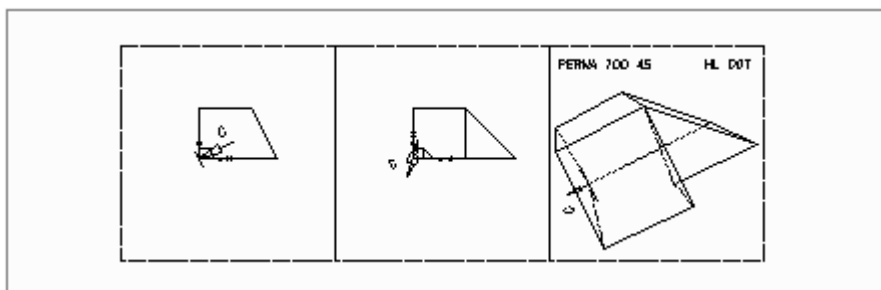


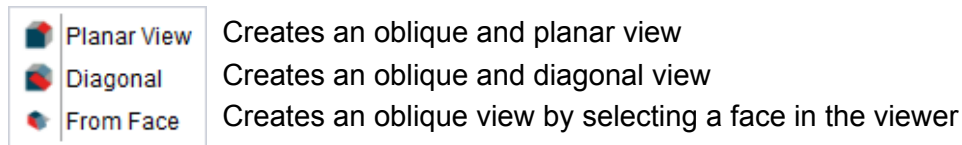
Figure 242 A Perspective Oblique View



Defining Oblique Views Using the Creates an Oblique View Tools

In addition to the procedure described in “[Defining Oblique Views Using Viewprims](#)” on [page 310](#) MEDUSA4 provides another possibility to define oblique views. Therefore a set of three special tools is available in the `Tools` tool group of the 3D tab.

Figure 243 The Creates an Oblique View Tools



The following describes the procedure how to work with the tools.

Creating an Oblique and Planar View




1. Open a 3D sheet and draw a simple rectangle with a profile line style into the viewbox which contains an orthogonal viewprim (DXY).
2. Define the depth of the model using a `Linear Sweep tool` .
3. Generate a model file and draw it by using the `Model + Reconstruct tool` .
4. Choose the `Planar View tool` .
The `Oblique View dialog` is displayed.

Figure 244 The Oblique View Dialog



5. Insert a name for the view in the input field and close the dialog with `OK`.
The name will be displayed later with the PVD prim.
6. Probe into the viewbox with the rectangle to define the first point of the view direction of the PVD prim.
You are prompted now to insert the second probe for the view direction (see the message line below the drawing area).
7. Probe again in the viewbox to define the second point of the view direction.
A PVD prim is placed on the drawing at the first probe point with the defined viewing direction.


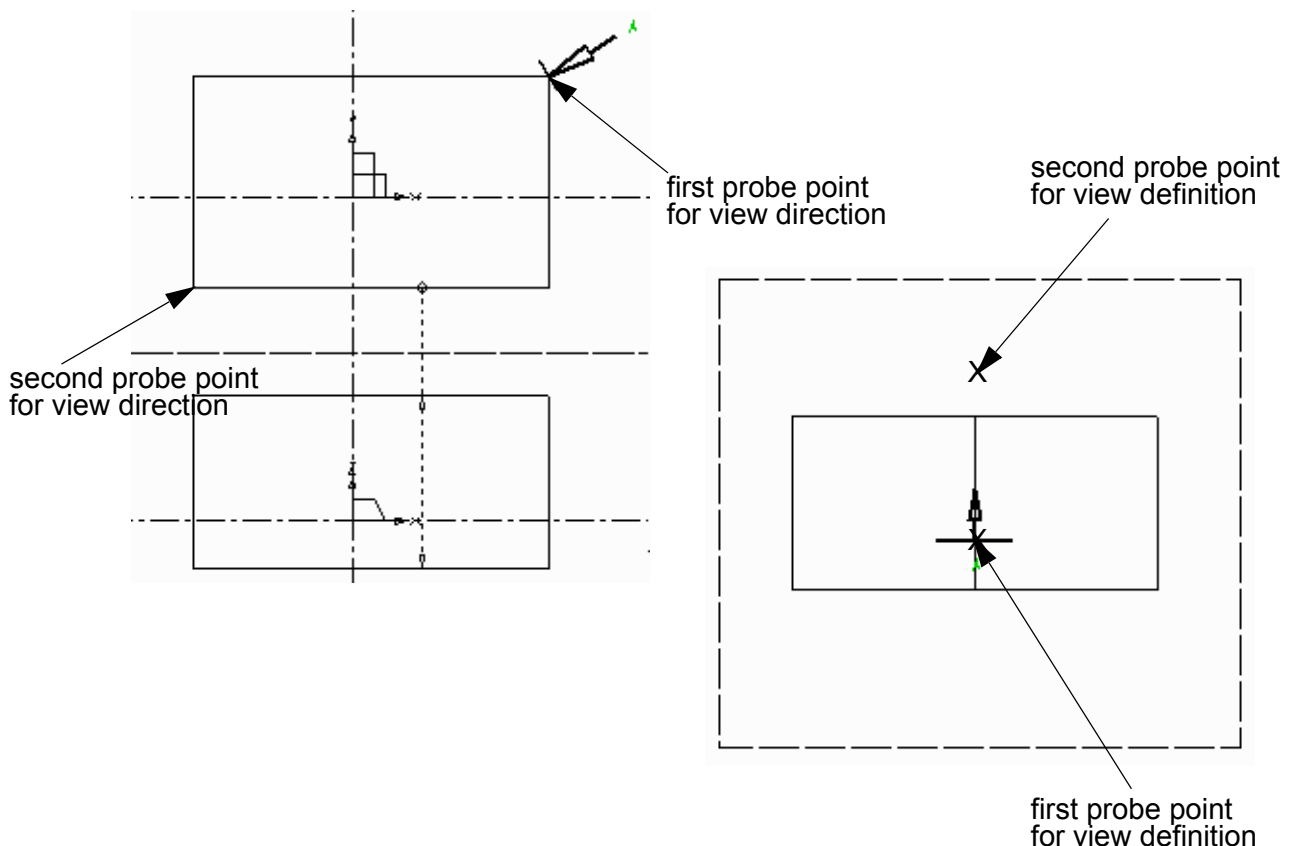

8. Now you are prompted to draw the first point of a new viewbox anywhere on the sheet, or to place the OVI prim in an existing, empty view box.
 - If you have an empty viewbox, go to the next step.
 - If you have no empty viewbox, draw it by clicking on the sheet to define two opposite corners.
9. Now you are prompted to place the OVI prim inside the empty viewbox. You also have two possibilities here:
 - If you click into the empty viewbox, the OVI prim is placed with the same view direction as the PVD prim.
 - If you select *Align oblique view prim* from the popup menu, you have to probe two points inside the empty viewbox to define the view direction of the OVI prim. In this case you can choose *Automatic alignment of oblique view prim* to switch back for placing the OVI prim with the same view direction as the PVD prim.
10. If the OVI prim is placed, choose the *View* tool  to generate the oblique and planar view. The following figure shows the definition and the views of the model.

Figure 245 Definition of an Oblique and Planar View of a Model



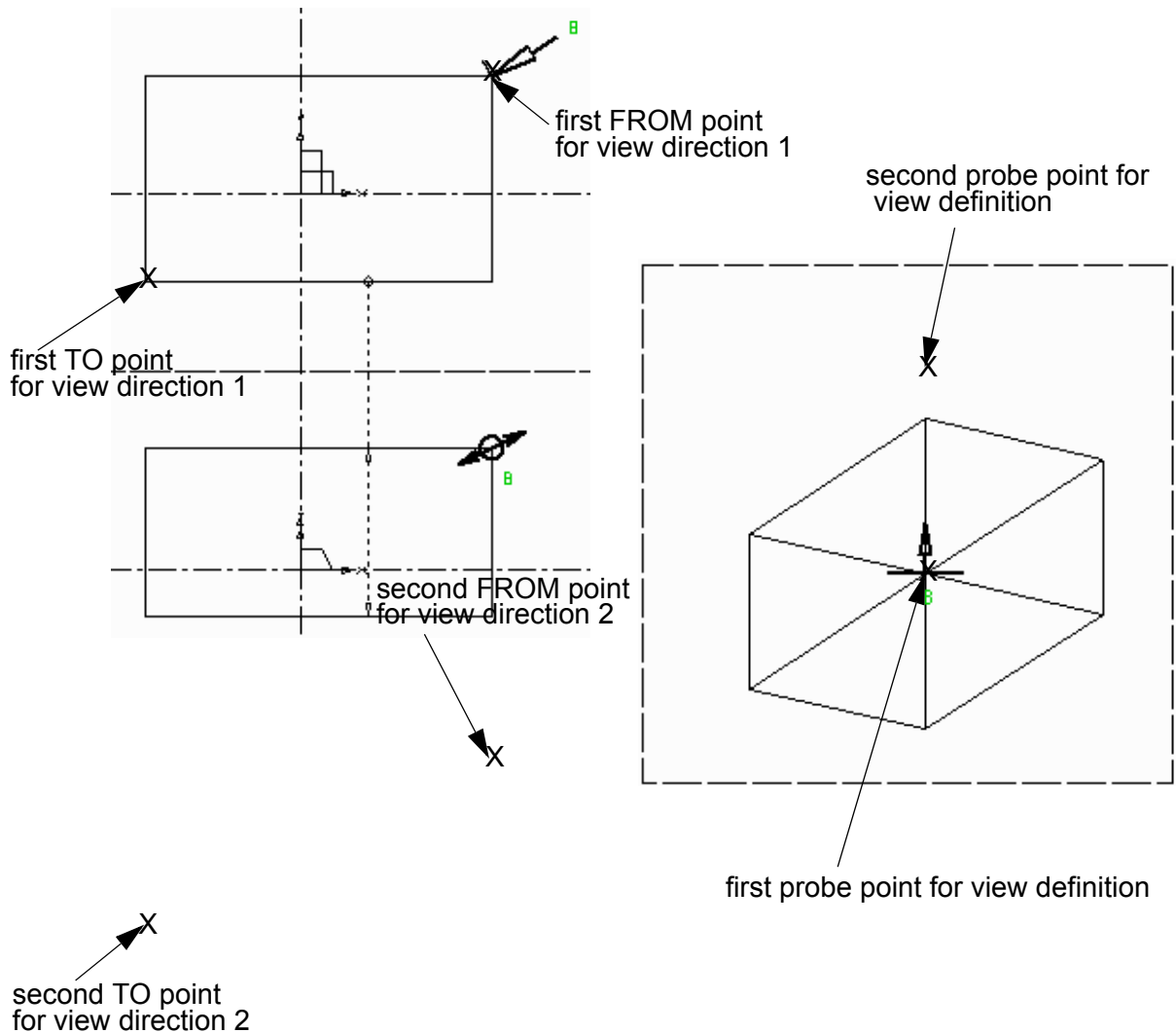
Creating an Oblique and Diagonal View

The procedure for creating an oblique and diagonal view is the same as described before. The differences are as follows:

- Use the Diagonal view tool .
- Having placed the PVD prim (first view direction) you are prompted to define the second view direction. Probe two points in the lower left viewbox. A SVD - prim is placed in the second viewbox. The added text complies with the name of the oblique and diagonal view you specified in the steps before.

The figure below shows definition and views of the model.

Figure 246 Definition and Oblique and Diagonal View of a Model



Creating an Oblique View By Selecting a Face in the Viewer

MEDUSA4 provides a comfortable method to create an oblique view by selecting a face directly in the viewer.


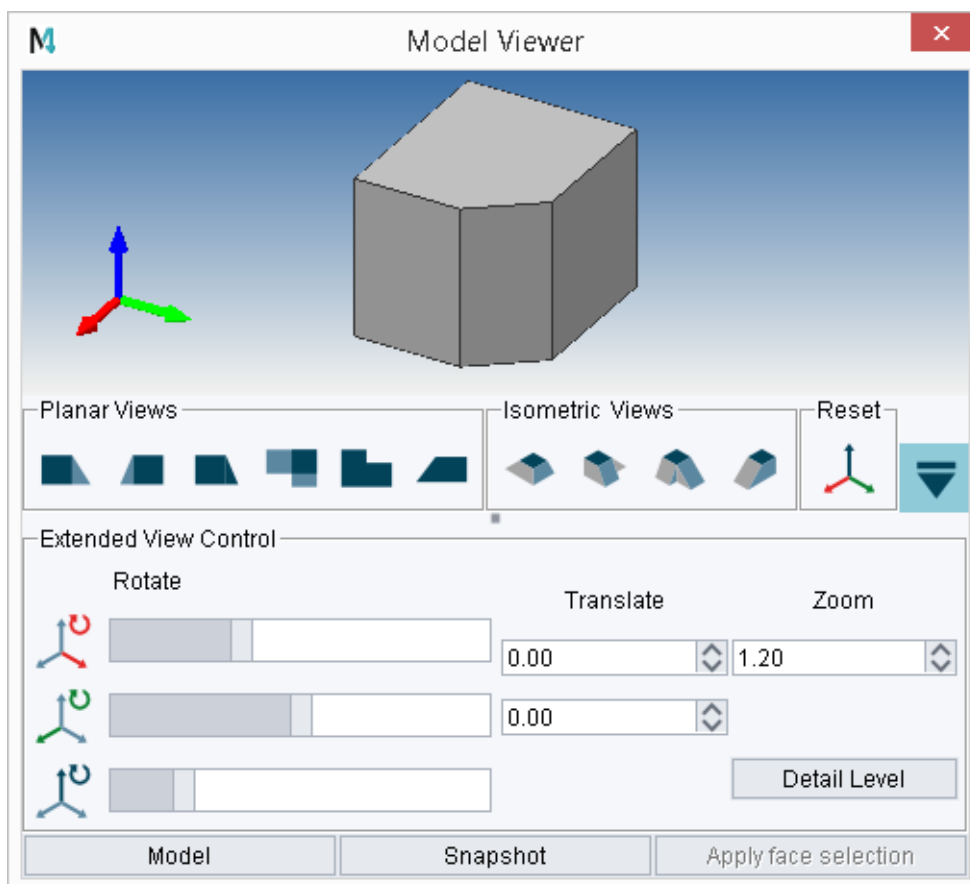
1. Define and generate a model file as you did before.
To get the same model as shown in the figures below chamfer one corner of the rectangle.
2. Choose the From Face tool .
The Model Viewer appears displaying the model.

Figure 247 The Model Viewer with the Model




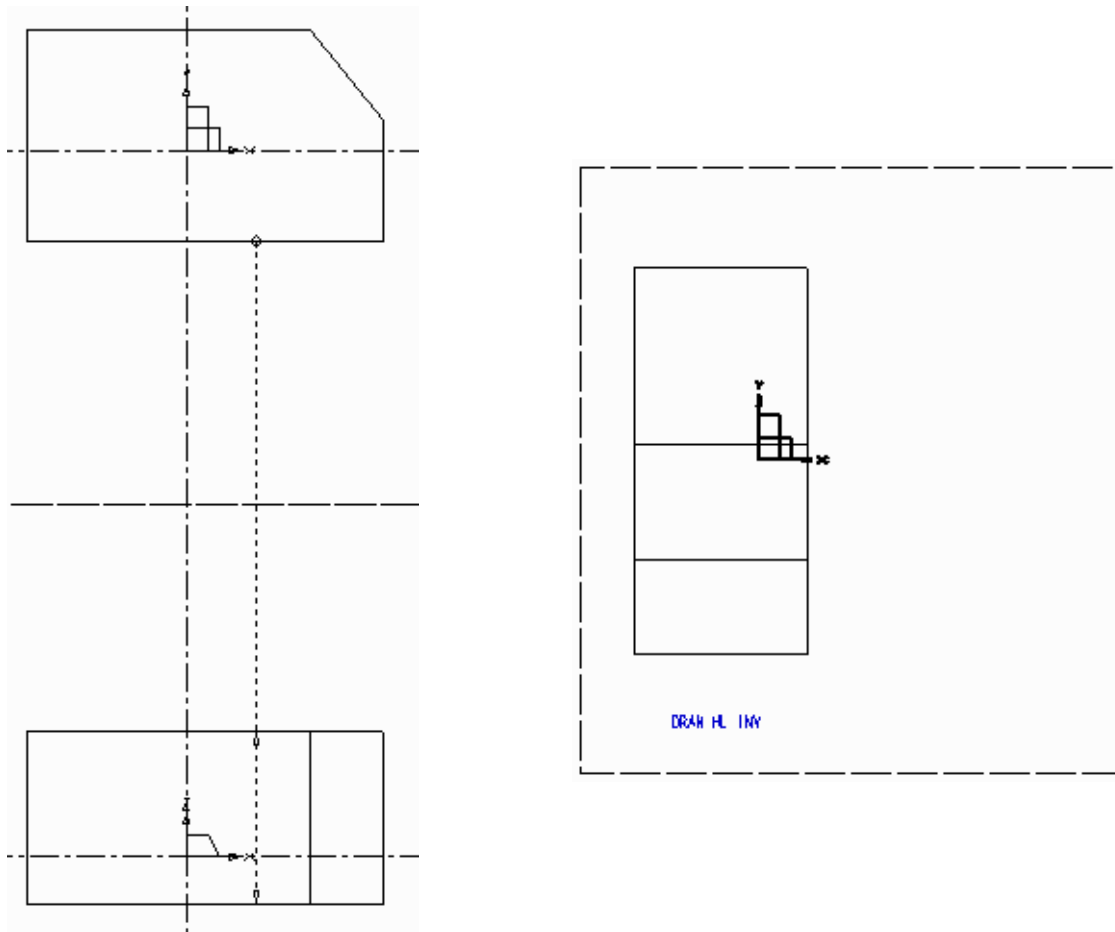
3. Select the face which you want to be displayed in the view as perpendicular view.
4. Click on the Apply face selection button.
The Model Viewer dialog is closed.
5. Create a new viewbox, if required, and probe in the viewbox.
An oblique prim is automatically placed at the probe point.
6. Generate the view by using the Model + Reconstruct tool .
The following figure shows the result of the described procedure.

Figure 248 Definition and Oblique View of the Model



Sectioning the Model

You can produce a view of a section through a model by using the `SECTION` command. This command will produce views of sections perpendicular to the viewing direction, partial or shaped sections and sections from cranked planes.

SECTION Command

The `SECTION` command can be added as `viewbox` attribute to a `viewbox` or can be created as `TVS-text` of style `View Specification Text` and placed on the sheet.

The `SECTION` command produces sections perpendicular to the viewing direction.

Please note: To use the `SECTION` command successfully you have to place the `Viewer` command `DRAW` in the `viewbox`, where the section is to be produced, or as a global viewing command outside of `viewboxes` anywhere on the sheet.

Sectioning cannot be performed in `SKETCH` mode.

Section Perpendicular to the Viewing Direction

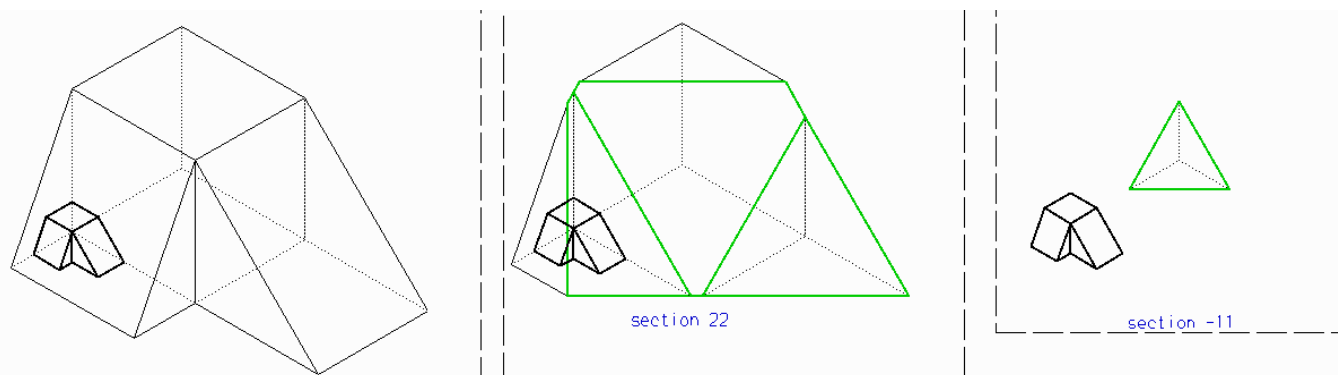
The sectioning plane runs through the origin by default, but it may be defined at a certain distance from the origin along the viewline by a command of the form:

```
SECTION distance
```

When the `SECTION` command is used with the `distance` option, the sectioning plane is always perpendicular to the viewing direction.

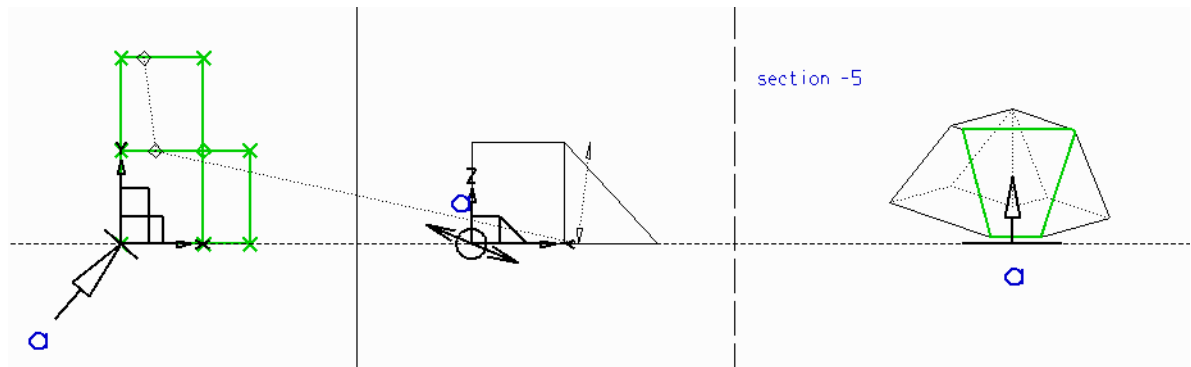
A positive value for `distance` sections the model closer to the viewpoint, and farther for a negative value. [Figure 249](#) gives examples of the `SECTION` command and shows how viewing commands in the `viewboxes` help to create this type of view.

Figure 249 Creating Some Sectional Views



Sectional views can be set up with oblique viewprims as well as commands. An example is shown in [Figure 250](#). Note that the PVD prim datum defines the position of the TO point in the XY plane, and that the SVD prim datum defines the position of the TO point in the Z-direction.

Figure 250 Setting Up a Sectional View Using Oblique Viewprims



Section From an Infinite End Plane

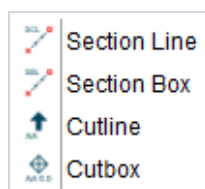
You create this type of section by defining a cutting surface using a profile line of style `Section Cut`. By adding any point function at one end of the profile line, you extend this end to infinity. If you wish to extend the whole line to infinity you must add a point function at both ends of the line. Once the cutting surface has been defined it must be named using text of style `3D Section text` and a section prim of style `Section line prim` must be added.



When the profile of the section is completed, you must place the `SECTION` command as text of style `View Specification Text` in the viewbox where you want the section to be produced. You should enter the command in this format.

```
SECTION cut_line_name
```

To produce a section through a model, from an infinite end plane, follow the described procedure. The tools for sectioning are located in the toolset shown in the figure below.

Figure 251 Toolset for Sectioning



1. Select the `Section Line` tool  to create a section line through the geometry as shown in the definition drawing, [Figure 252](#).
The profile should be an open shape made up of one or more straight segments and must not intersect itself.
2. Add a point function to any end of the section line.
3. Select the `Cutline` tool  to load a prim and attach it to the section line with a segment probe.

This section line prim contains a default text string of style `3D Section Text` which specifies the cut line name. Prim and text are within a group.

Please note: The section line prim should be attached to a segment and not to a vertex. If it is attached to a vertex, it will be considered as part of the segment ending at that vertex. If this segment is an invisible segment, the next segment will be used.

4. Edit the default text string (AA) and change it as required, in our example `CUTAA`.

Please note: The section line name may not contain spaces!


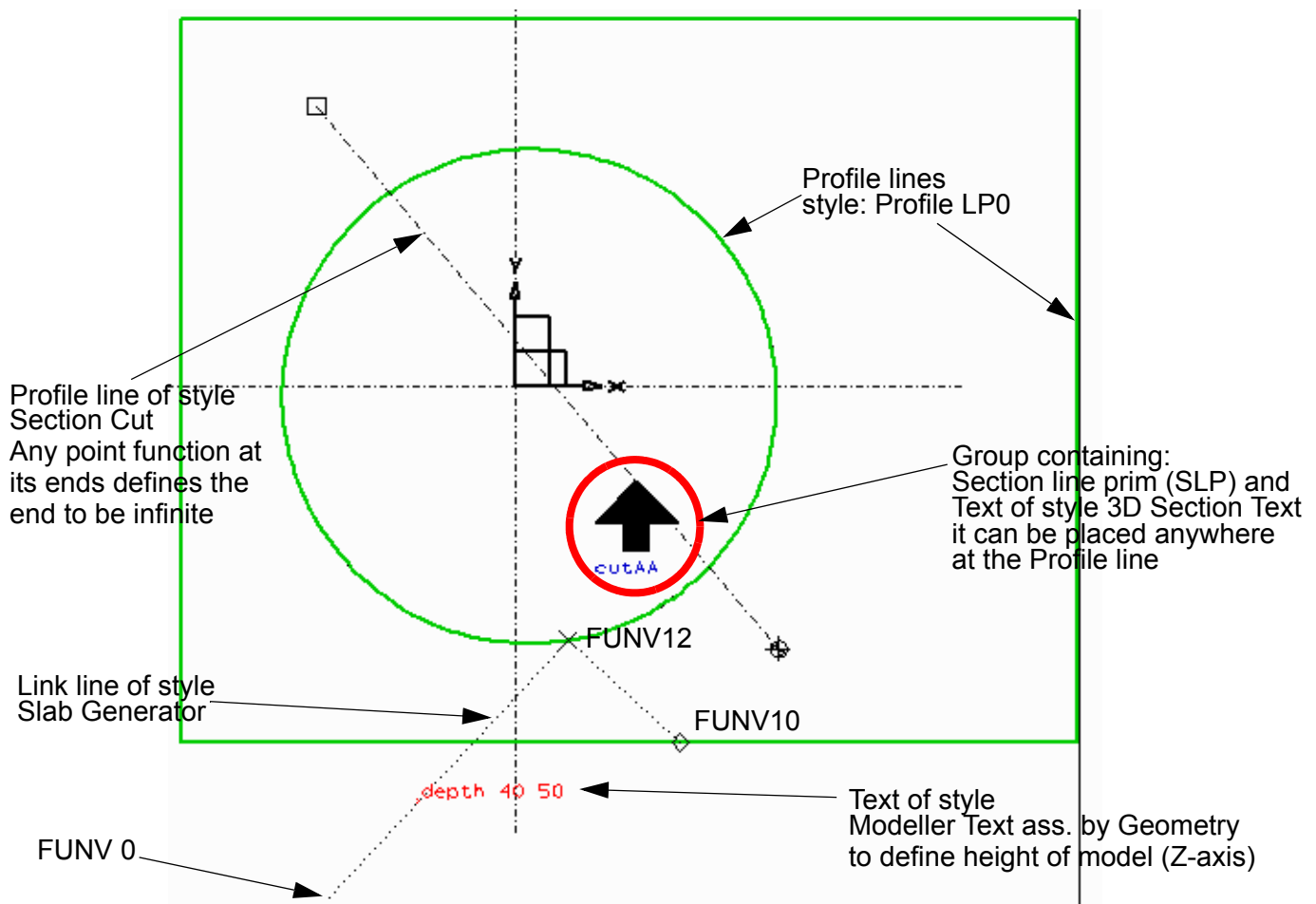
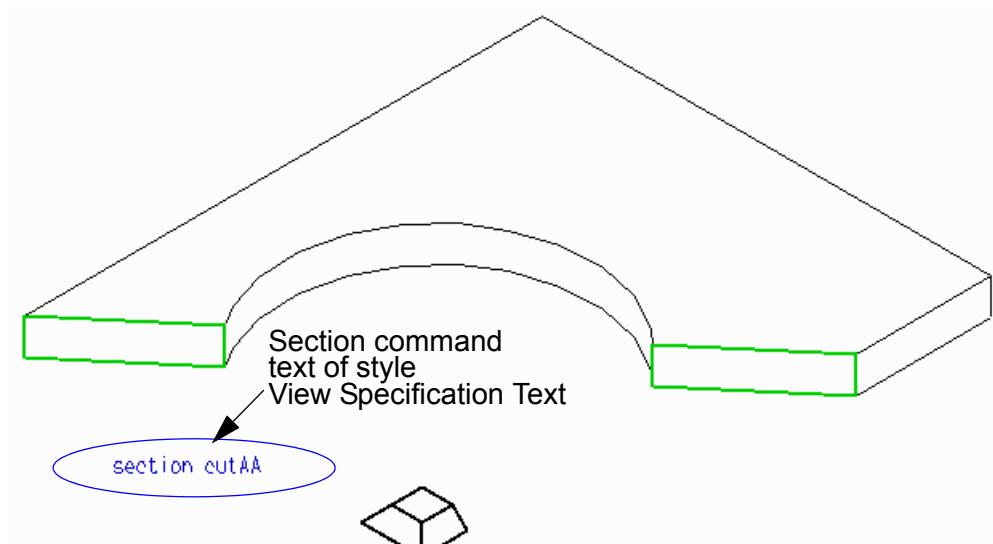
5. Choose the tool  for placing the `SECTION` command in a viewbox. In our example the command is: `SECTION CUTAA`.

Figure 252 Cross-Section From an Infinite End Plane, Definition of the Model



6. Select the Model + Reconstruct tool  to generate the sectioned model.

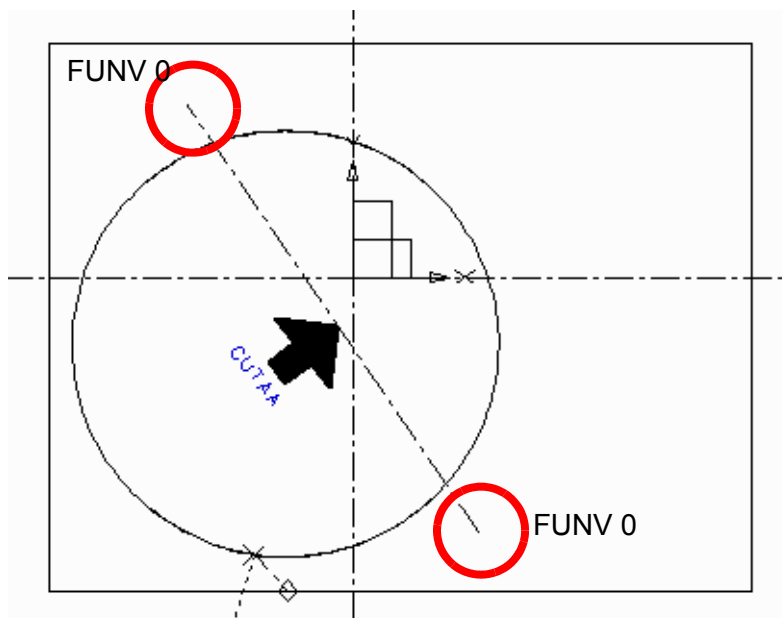
Figure 253 Cross-Section From an Infinite End Plane, Completed Model



Section From a Limited Plane

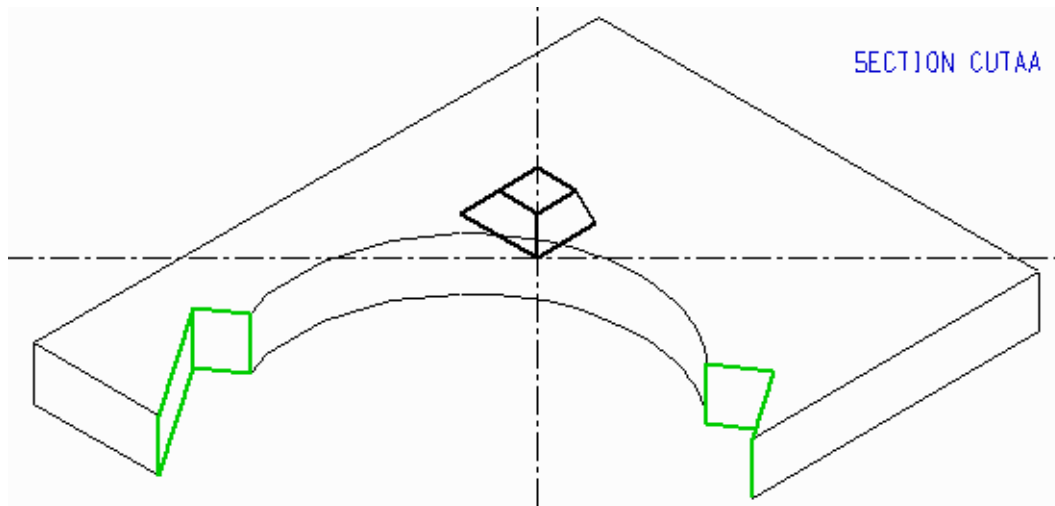
To create a section from a limited plane follow the same procedure as described above. The only difference is the section line which defines a limited plane. It **does not include any point function** at either end of the line. An example of partial section in a limited plane is illustrated in [Figure 254](#) below. It shows the same definition model as shown in [Figure 252](#), but without any point function at the section line.

Figure 254 Cross-Section From a Limited Plane, Definition



The result of sectioning is shown in the following figure. To see the difference of the sections (with and without point functions at the ends of the section line), compare [Figure 255](#) below with [Figure 253](#), “Cross-Section From an Infinite End Plane, Completed Model” on page 321.

Figure 255 Cross-Section From a Limited Plane, Completed Model



As you can see, at both ends of the section line additional line segments were added. These are vertical to the segment which has the section prim attached. The additional section line segments are displayed in the section view of the model.

Crosshatched Sections in 2D Space

You can crosshatch one named sectioned object or all sectioned objects in 2D space using the `CRH2D` command. The `CRH2D` command has the following format:

```
CRH2D object_name/ALL angle spacing offset
```

To place this command in the viewbox type the `CRH2D` command at the keyboard as TVS-text of style `View Specification Text`.

Please note: The parameter values you enter remain independent of the inclination of the surface and also no hidden lines are removed.

An example of section crosshatched in 2D space is illustrated in [Figure 256](#) and [Figure 257](#).

Figure 256 Crosshatched Section in 2D Space, Definition

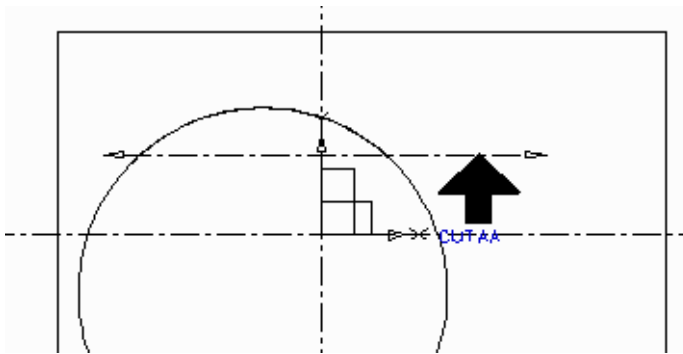
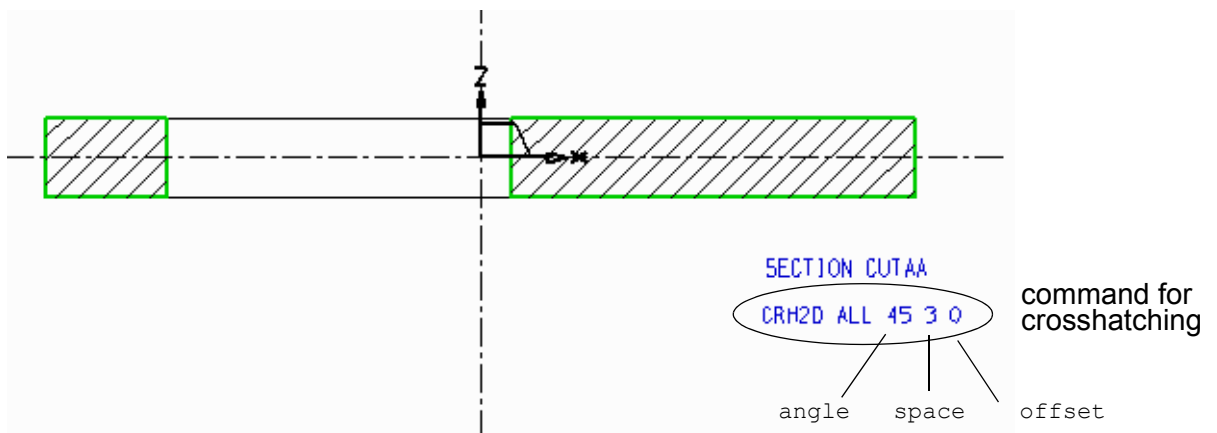


Figure 257 Crosshatched Section in 2D Space, Model



Crosshatched Sections in 3D Space

You can crosshatch the sectioned surfaces of a named object or all objects in 3D space, using the `CRH3D` command. Any crosshatching on a plane normal to the view direction will have the values you specified for the crosshatching *angle*, *spacing* and *offset* when entering the command as follows:

```
CRH3D object_name/ALL angle spacing offset
```

To place this command in the viewport type the `CRH3D` command at the keyboard as TVS-text of style `View Specification Text`.

The crosshatching lines are created within a group and all hidden lines are removed when there is an object in front.

The following example contains two named objects, a cut profile line without point functions, a crosshatching command and the command `SEC OFF B`. This command causes the cylinder, object `B`, to be viewed. If the command were missing, object `B` would not be displayed, because the hidden lines are removed by the crosshatching command.

Figure 258 shows the definition of the model.

Figure 258 Sectioned Object Crosshatched in 3D Space, Definition

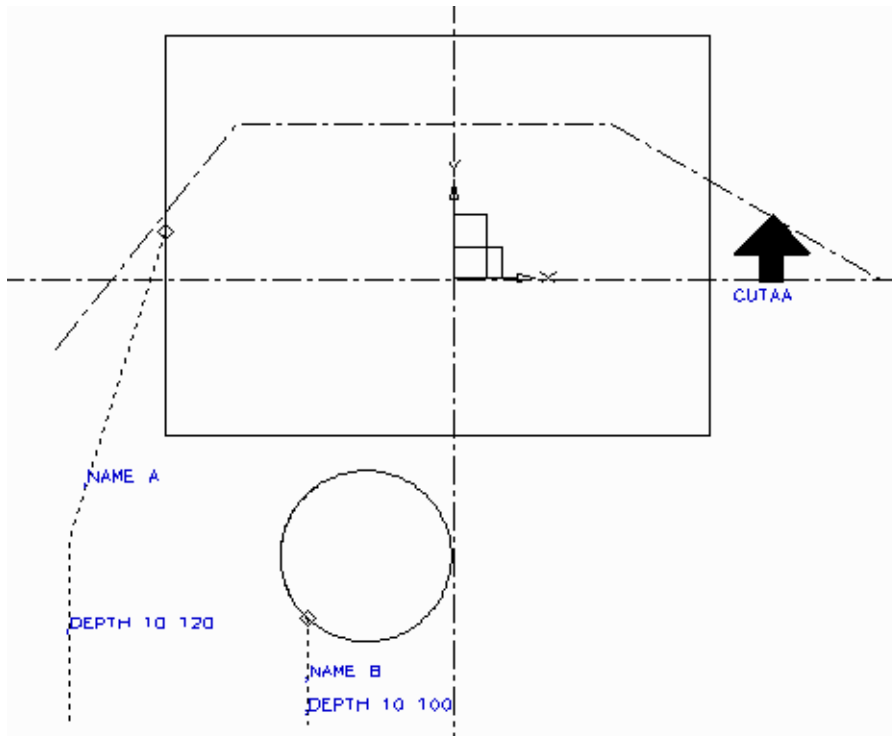


Figure 259 Sectioned Object Crosshatched in 3D Space, Completed Model

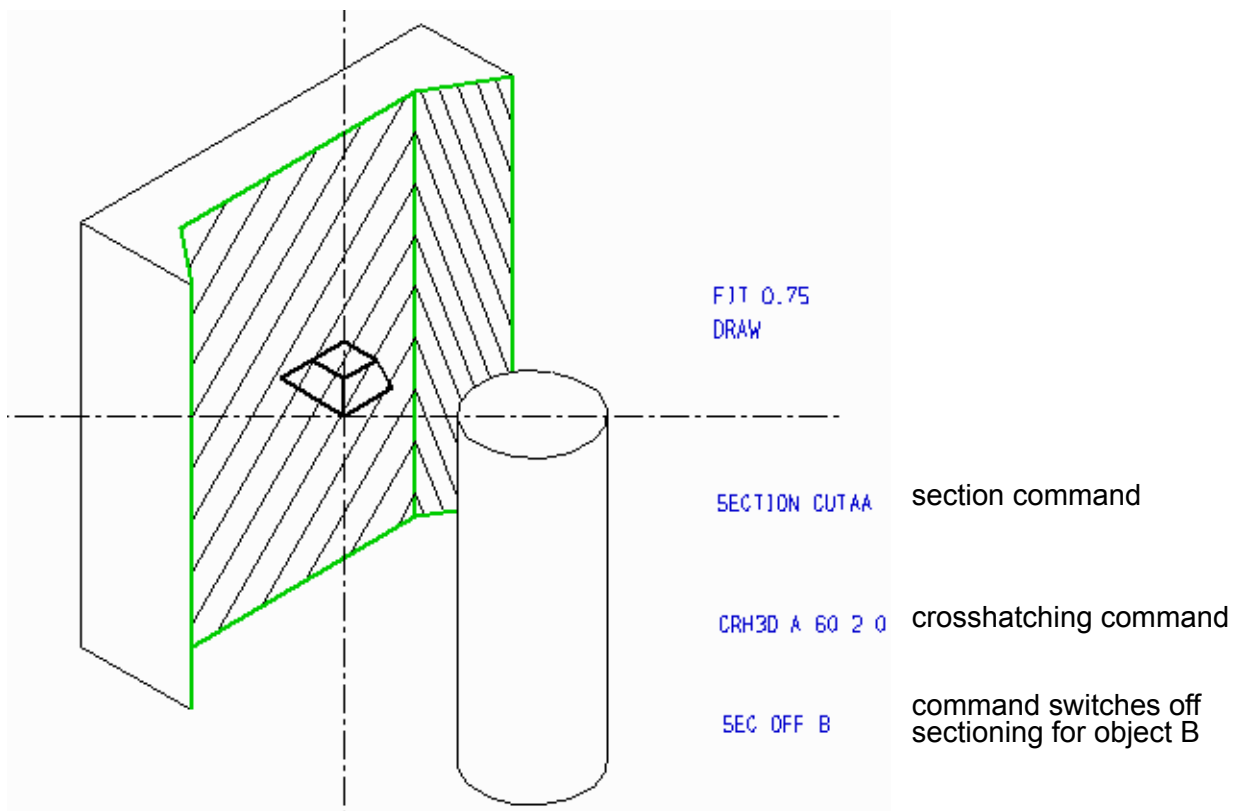
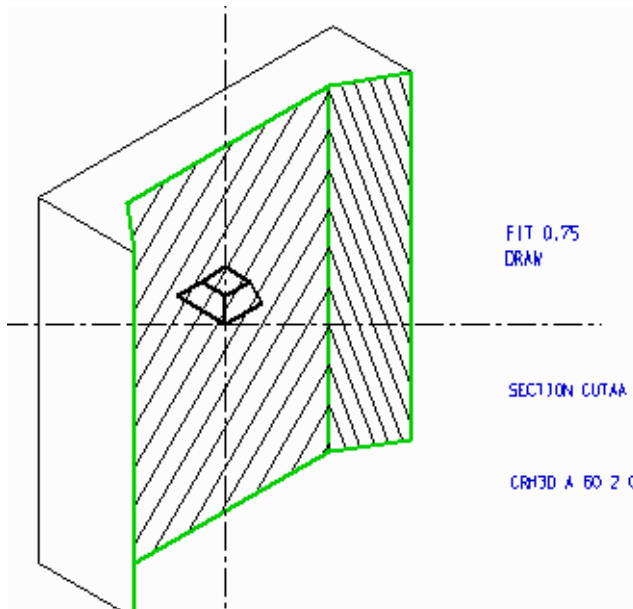


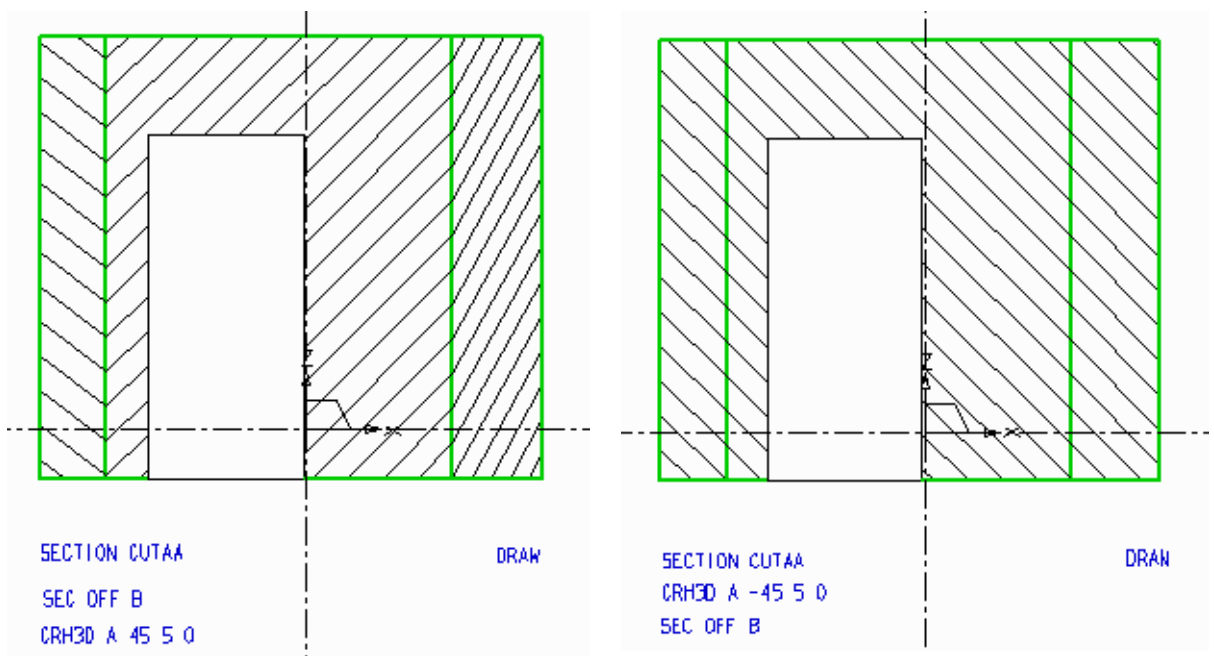
Figure 260 Completed Model without SEC OFF Command



If the angle value is positive, all the other cross-sectioned planes will have their crosshatching parameter values modified to achieve a 3D effect. See the side view of the object, illustrated in [Figure 261](#).

If you do not wish to obtain a full 3D effect, you must use a 3D cross-hatching command which specifies a negative angle value as shown in [Figure 261](#).

Figure 261 Side View of the Object Crosshatched in 3D Space with Positive and Negative Angle



Controlling the Sectioning

You can control sectioning either by switching it on or off for certain objects. To switch the sectioning of named objects ON or OFF, use the SEC command in this format:

```
SEC ON object_name  
SEC OFF object_name
```

To place this command in the viewbox, type the SEC command at the keyboard as TVS-text of style View Specification Text.

Please note: In one viewbox all the SEC commands must be either ON or OFF.

Figure 262 Switching On the Sectioning of a Named Object, Definition

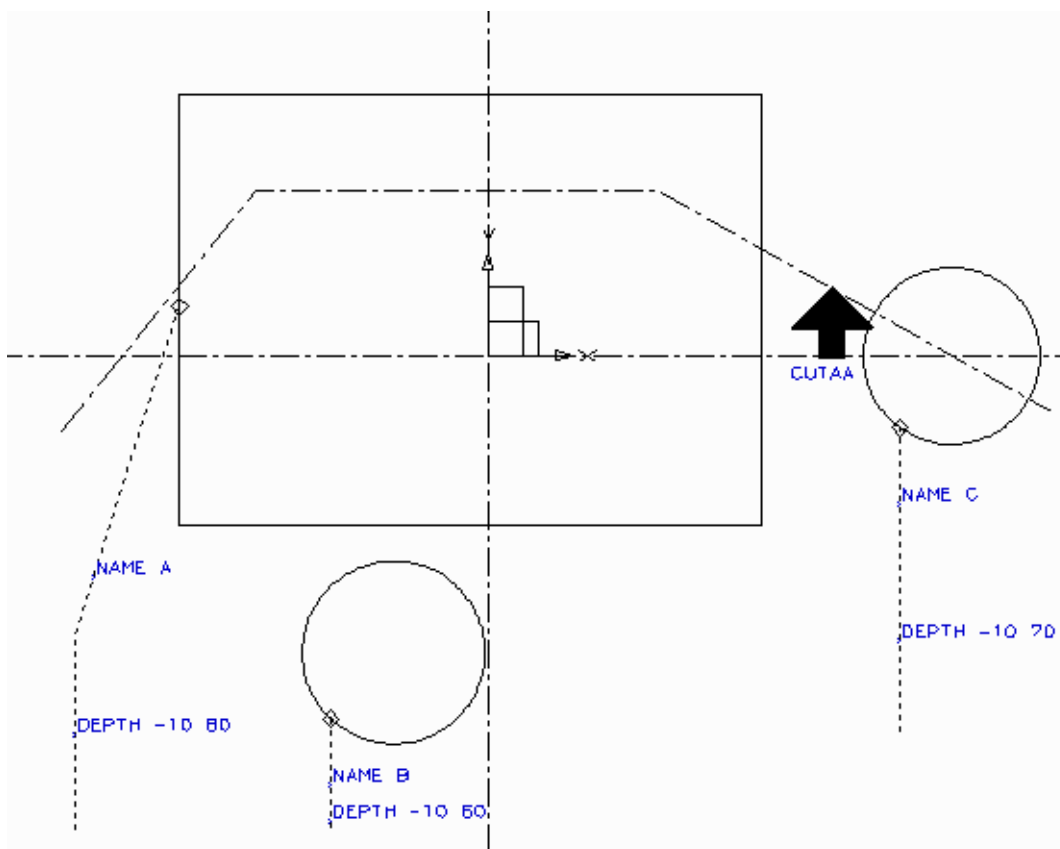
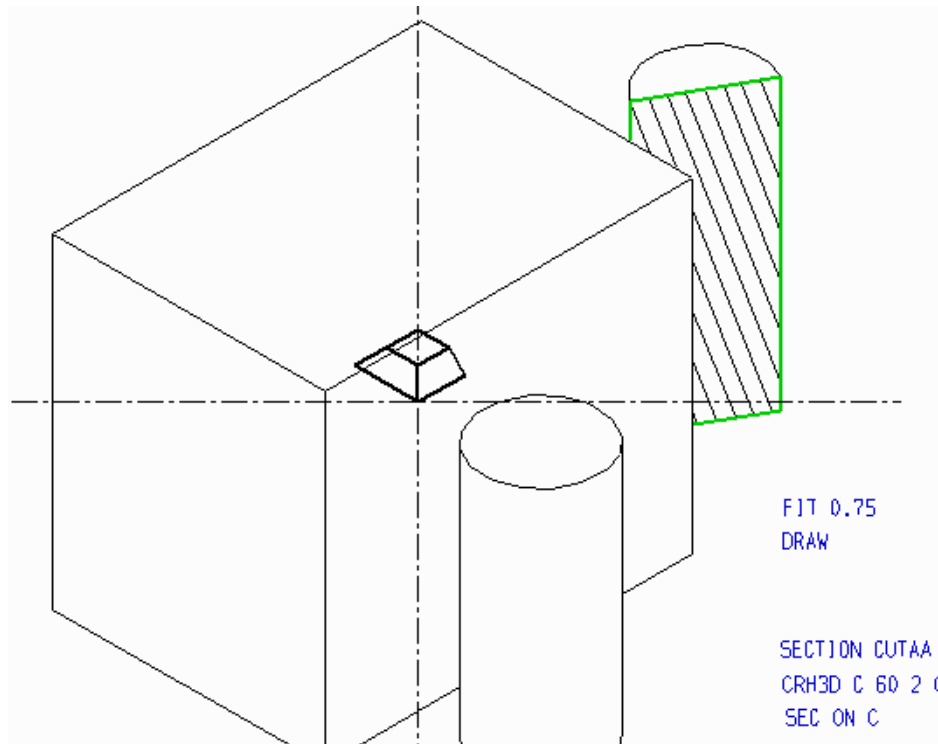


Figure 263 Switching On the Sectioning of a Named Object, Completed Model



View Sectioned Surface Only

To produce a view of the sectioned surface only, you can use the `SEC ONLY` command with the option `ON` or `OFF`, in this format:

```
SEC ONLY ON
SEC ONLY OFF
```

To place this command in the viewbox, type the `SEC ONLY` command at the keyboard as TVS-text of style `View Specification Text`.

Examples illustrating the `SEC ONLY` command are shown in [Figure 268, “Shaped Section with Negative Depth Value, Completed Model”](#) on page 331 and [Figure 269](#).

Sections From Cranked Planes

You can produce compound sections from cranked planes. To create the profile of the cutting surface you should follow the same procedure as described on [“Section From an Infinite End Plane”](#) on page 319. If one end of the profile has no point function (`FUNV 0`) then that end of the line is assumed to be constrained. Any point function means that the end extends to infinity. Once the section is created, you must place the `SECTION` command in a viewbox, in this format:

```
SECTION cut_line_name
```

To place this command in the viewbox, type the `SECTION` command at the keyboard as TVS-text of style `View Specification Text`.

An example of cutting surface created from a cranked plane with a constraint at one end is illustrated in [Figure 264](#) and [Figure 265](#). The sectioned object is displayed crosshatched and viewed at an angle.

Figure 264 Cross-Section From a Cranked Plane, Definition

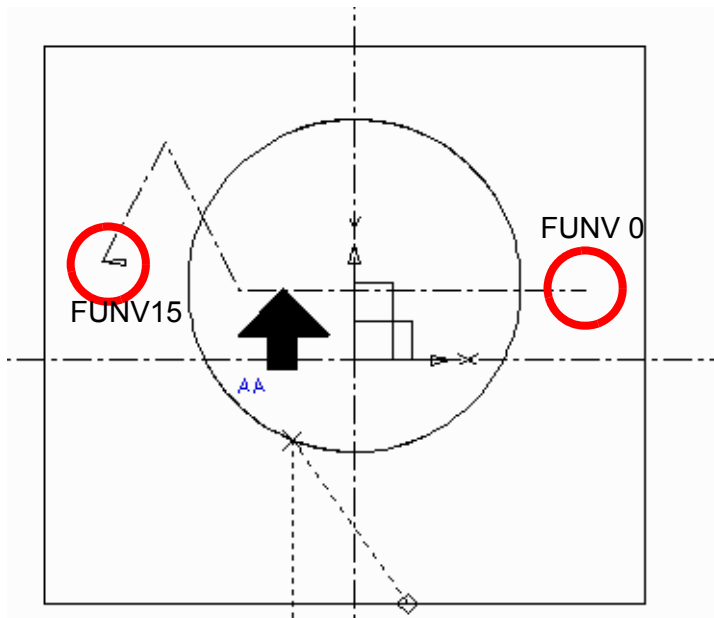
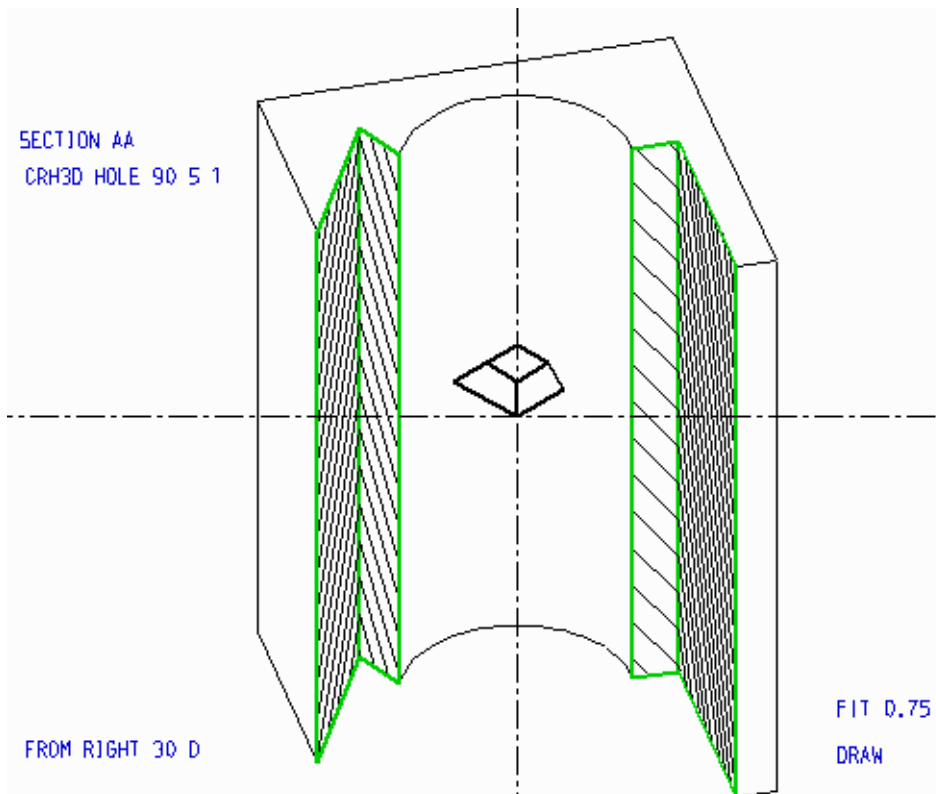


Figure 265 Cross-Section From a Cranked Plane, Completed Model

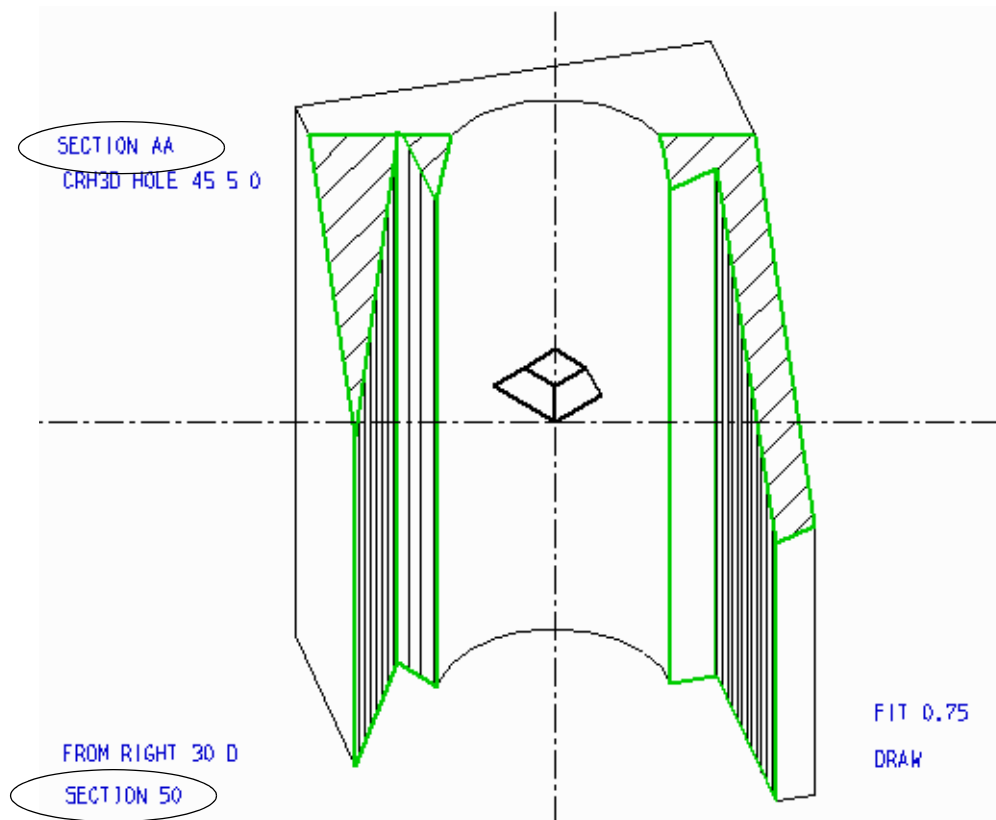


A similar example is shown in [Figure 266](#). The definition of the model is the same as shown in [Figure 264](#).but this time two different SECTION commands are placed in the viewbox:

```
SECTION distance
SECTION cut_line_name
```

As a result of the different viewing commands the object is sectioned several times.

Figure 266 Object Sectioned Several Times





Shaped Sections

A shaped section is created by defining a cutting surface which has a closed shape.

Once the section is created, you must place the SECTION command as TVS-text of style View Specification Text in a viewbox, in this format:


```
SECTION box_line_name
```

To produce a shaped section through a model, follow this procedure:

1. Select the Section Box tool  to create a line of style Section Box as closed profile in a view box containing a view prim.
The profile should contain three or more straight line segments and must not intersect itself.
2. Select the Cutbox tool  to create a prim of style Section Box Prim and attach it to the profile line using a segment probe.

This prim displays two default text strings, one specifies the box line name (by default AA), the other defines the depth of the cutting surface (default value is 0.0).

The section box prim and the 3D Section Text text are within a group. The depth of the cutting surface is measured from the viewprim datum line and can be positive or negative.

3. Change the box line name and the depth value as required.
4. Place the SECTION commands in the viewboxes as required.
Enter the SECTION commands as TVS-text of style View Specification Text using the keyboard.
5. Select the Model + Reconstruct tool  to create the shaped section.

Examples of shaped sections with a depth value of -30 and 30 are shown in [Figure 268](#), “Shaped Section with Negative Depth Value, Completed Model” on page 331 and [Figure 269](#).

Figure 267 Shaped Section, Definition of the Model

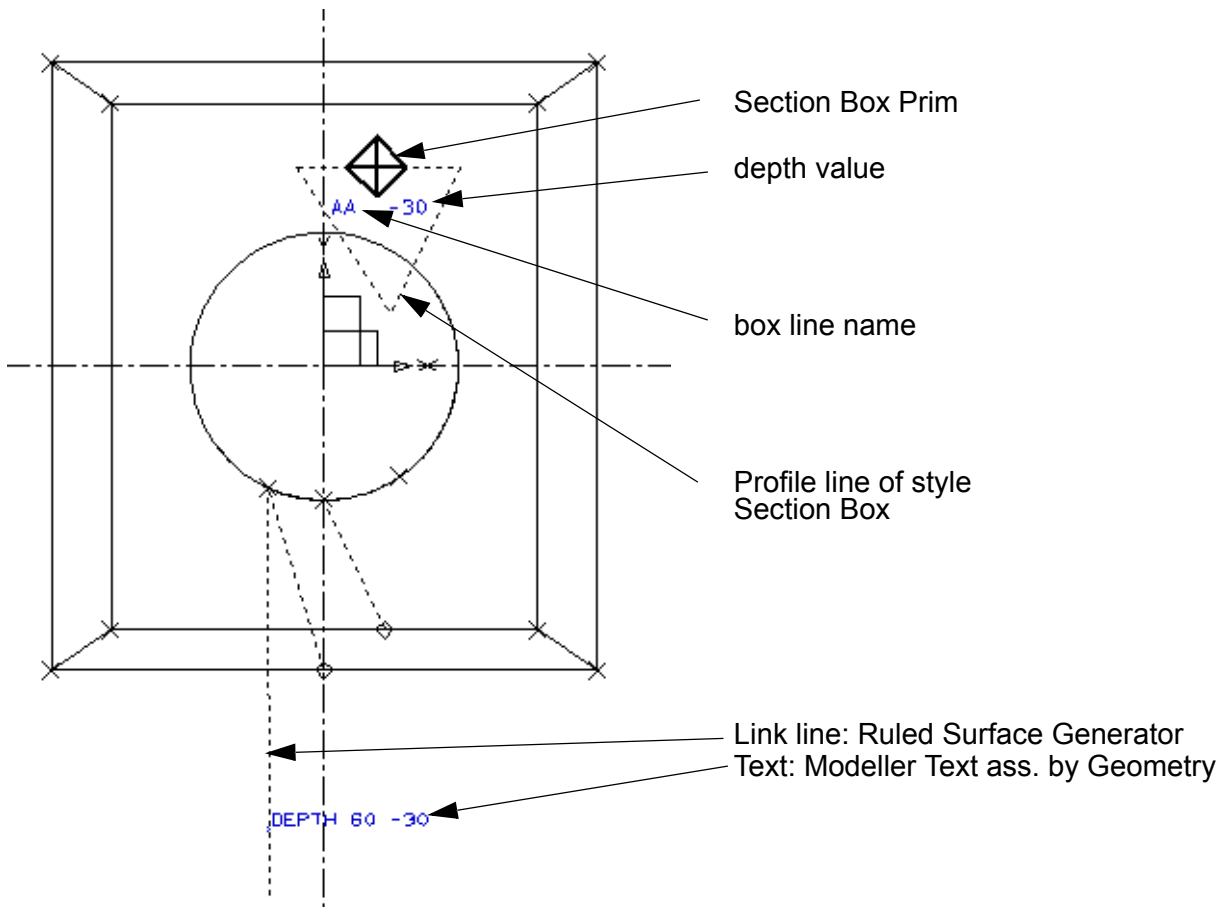


Figure 268 Shaped Section with Negative Depth Value, Completed Model

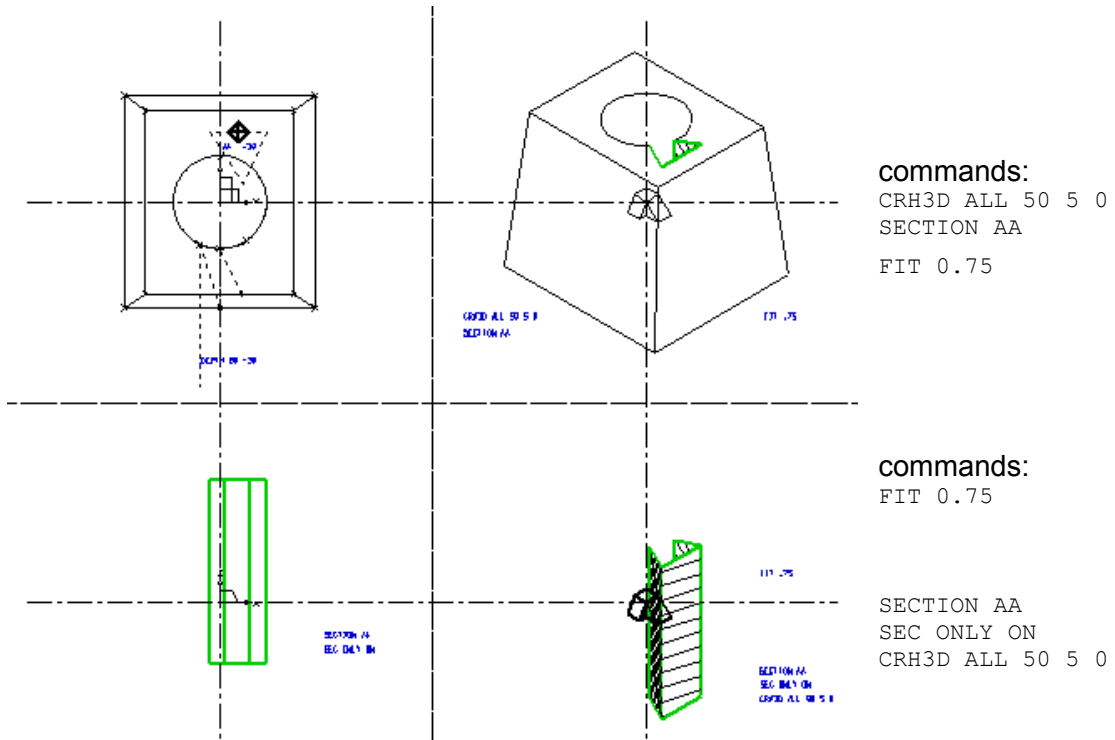
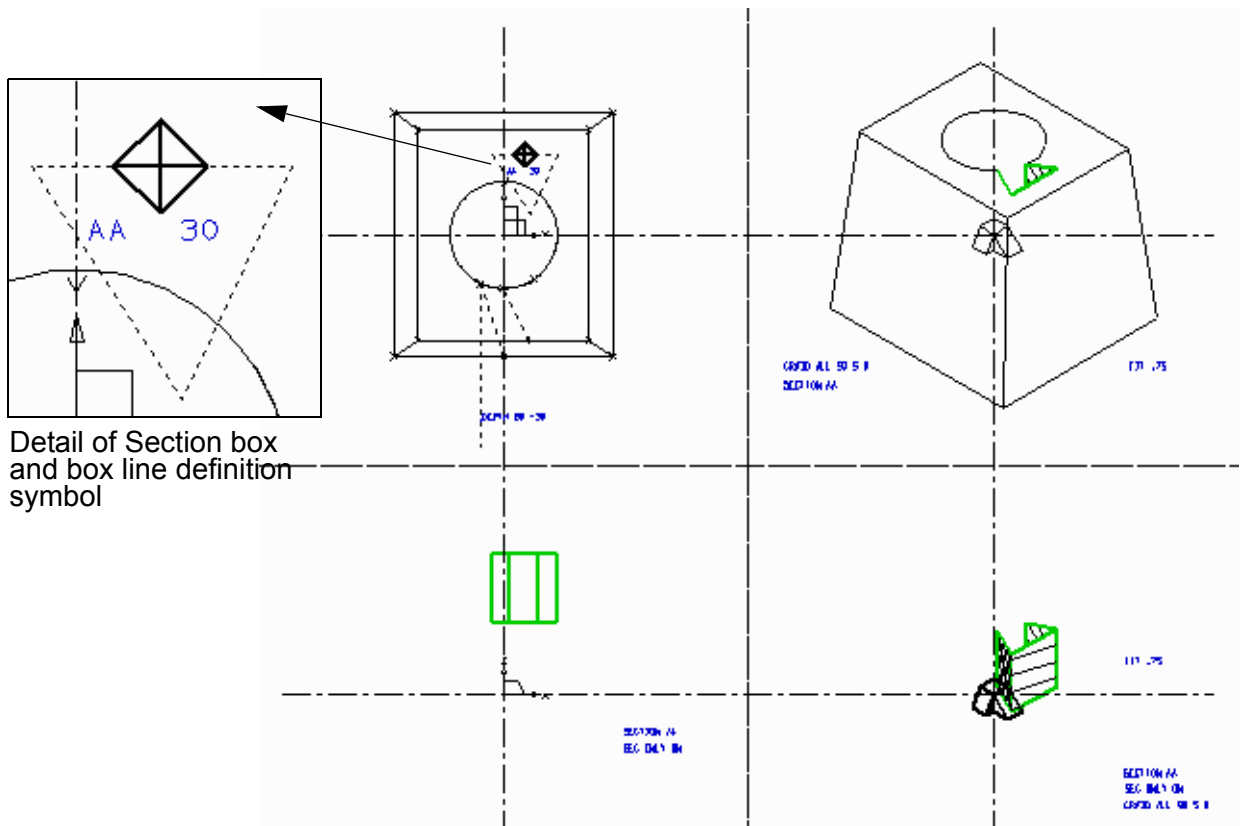


Figure 269 Shaped Section with Positive Depth Value



Examples of Sectioning and Crosshatching in Assemblies

The two following examples show the sectioning and crosshatching in 2D and 3D space, of objects contained in an assembly. The model definition sheets for the assemblies described in "Example 1" and "Example 2" are shown respectively in Figure 191, "Completed Assembly Model of Axle and Bearing" on page 259 and Figure 193, "Completed Assembly Model of Axle with Two Bearings" on page 261.

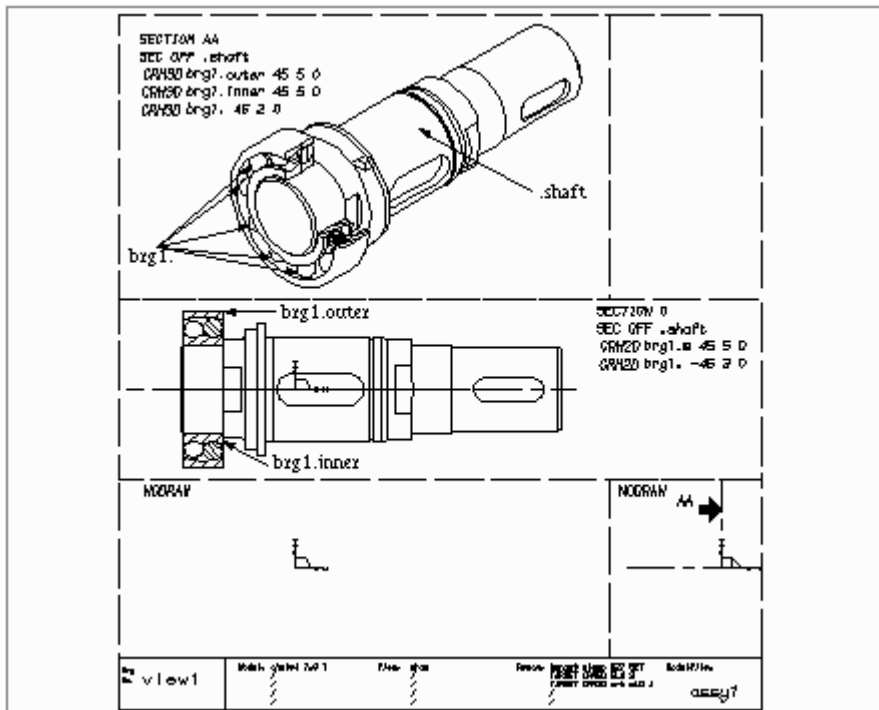
Example 1: Figure 270 shows an assembly (`assy1`) containing a shaft and a bearing. This assembly contains the following objects:

```
.shaft
brg1.outer
brg1.inner
brg1.
```

The section is created from a limited plane and among the four objects listed above. `.shaft` is the only one which is not sectioned. The sectioned objects are crosshatched in 2D and 3D space.

Please note: For information on the `@` wildcard, and the naming convention for objects in instanced models (dot character), see respectively page 338 and "The Naming Convention" on page 256.

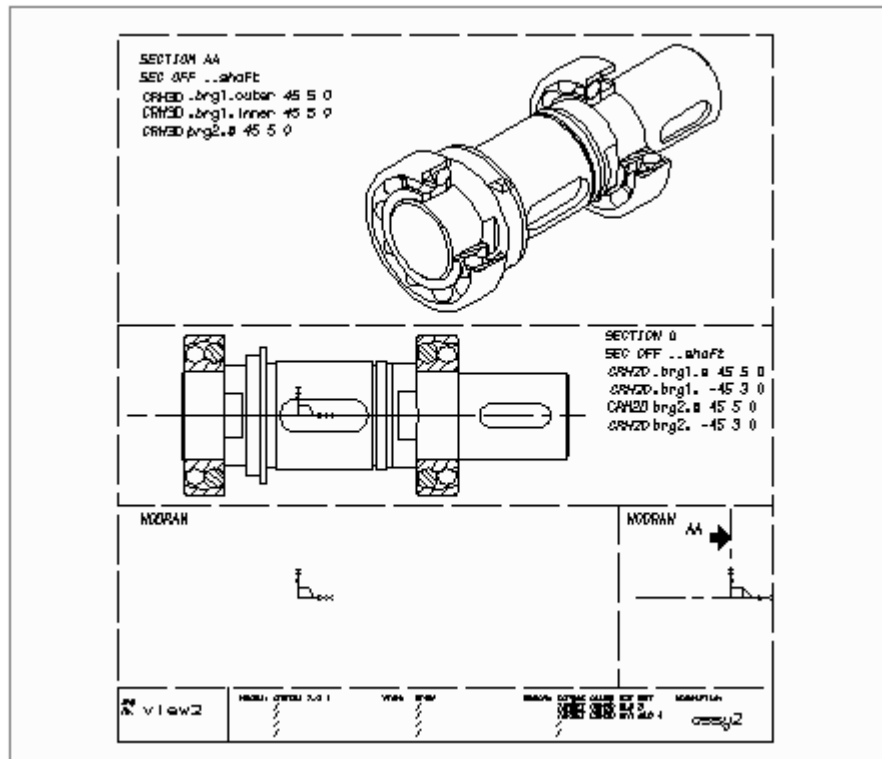
Figure 270 Assembled Model of Axle with One Bearing Sectioned and Crosshatched



Example 2: In [Figure 271](#), the assembly (`assy1`) is instanced into a new assembly (`assy2`) which includes a second bearing component. This second bearing component is sectioned and crosshatched in 2D and 3D space.

Please note: For information on the `@` wild card, and the naming convention for objects in instanced models (dot character), see [“OBJ” on page 338](#) and [“The Naming Convention” on page 256](#).

Figure 271 Assembled Model of Axle with Two Bearings Sectioned and Crosshatched



The VIEWTOL Command

The `VIEWTOL` command is related to the `BOOTOL` command (see “[The BOOTOL Command](#)” on [page 235](#)), and is designed to overcome problems which can arise in the Viewer when the sectioning plane passes through the plane of a tile on a model. This is partly because the model being sectioned may contain several instanced objects, with each of them modeled using a different value of `BOOTOL`.

As for `BOOTOL`, there is a default value for `VIEWTOL` provided by the system which is related to model size. This default tolerance can be changed using the TVS text of style `View Specification Text`:

```
VIEWTOL value
```

where *value* is in object units.

Please note: If *value* is negative it is treated as a multiplier of the default tolerance value. This is the recommended way to use this command.

For example, as a first try the command can be given as:

```
VIEWTOL -2
```

If this is unsuccessful, a larger tolerance can be specified. It is reasonable to increase the multiplying value by a factor of two each time, for example:

```
VIEWTOL -4
```

If this is still unsuccessful, use the command:

```
VIEWTOL -8
```

and so on. Always try and use the smallest possible value.

Viewing Commands

This section contains a complete list of the viewing commands. Commands may be typed from the keyboard in uppercase or lowercase form. They are shown in uppercase in the syntax graphs below.

ANGLE

→ ANGLE <angle> →

Sets the angle in degrees by which the image of a model is rotated counterclockwise. The angle is built by the current upvector and the Y-axis.

BOU

→ BOU → INV
→ VIS →

Sets the patch boundaries on curved surfaces to be visible (VIS) or invisible (INV). The default is BOU INV.

CRH3D

→ CRH3D → <object_name>
→ ALL → <angle> <spacing> <offset> →

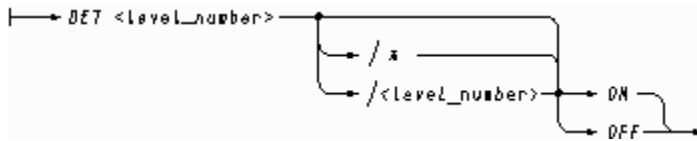
Creates crosshatching of the sectioned surfaces in 3D space so that any crosshatching on a plane perpendicular to the view direction will have the crosshatching you specified with the *angle*, *spacing* and *offset* options.

CRH2D

→ CRH2D → <object_name>
→ ALL → <angle> <spacing> <offset> →

Creates crosshatching of the sectioned surfaces in 2D space, using the values you specified with the *angle*, *spacing* and *offset* options.

DET

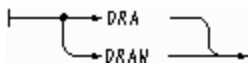


Switches the specified detail levels ON or OFF enabling you to view or validate objects selectively if the objects have been assigned to a detail level at the modeling stage. Only the detail levels switched ON can be seen, or processed.

The *level_number* specifies the level number to be switched ON or OFF. You can specify *level_number* as a number in the range 0 through 255. The default is 0.

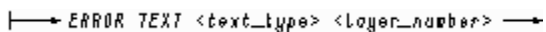
The option */** switches levels from the given level number to the end. To specify a range of level numbers to be switched ON or OFF, you must use the */level_number* option.

DRAW



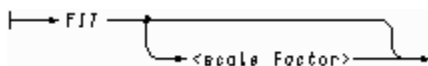
Draws the currently defined view on the screen.

ERROR TEXT



Sets the *text_type* and *layer_number* of the error text which is added to the sheet when an error occurs during normal operation. The `ERROR TEXT` command is defined on the sheet as text of style 3D Viewer Error Text for Viewer errors, and as 3D Model Error Text for Modeler errors. Instead of a text type you can use alternatively a CAN code or a style name in quotes (for example, style name "Viewer Error Text").

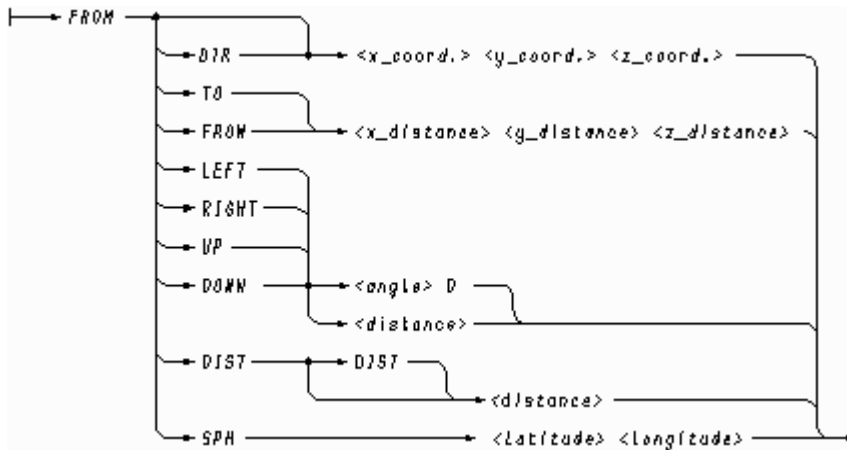
FIT



Fits the image of the model to the viewbox in axonometric views. If the optional argument *scale_factor* is not used, the image is sized to 0.98 of the limiting viewbox dimension. If *scale_factor* is used, the image is sized to the defined fraction of the limiting viewbox dimension.

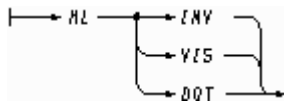
Please note: This command does not show the correct spatial relationship between model and viewprim (the TO point). The value of *scale_factor* can be greater than one, in which case the image of the model will be clipped by the viewbox boundary.

FROM



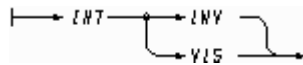
Moves the FROM point (the viewpoint) to the required position.

HL



Sets hidden lines invisible (INV), visible (VIS), or dotted (DOT) in views produced by the DRAW command.

INT



Sets object intersection lines to be visible or invisible.

NODRAW

$\rightarrow NODRAW \rightarrow$

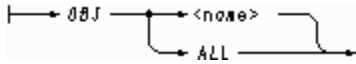
Suppresses graphical output.

NO FIT

$\rightarrow NO FIT \rightarrow$

Cancels the FIT command.

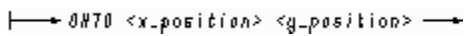
OBJ



Selects a named object or `ALL` objects for viewing. The argument *name* may contain a wild card character to mean any string. There are three wild card characters available:

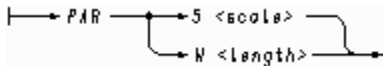
Characters	Description
+	Matches a single character
%	Matches any number of characters up to a full stop (full stop not included). It also matches any number of characters up to the character following the percent sign.
@	Matches any number of characters up to the end of the string, or up to the character following the @ sign.

ONTO



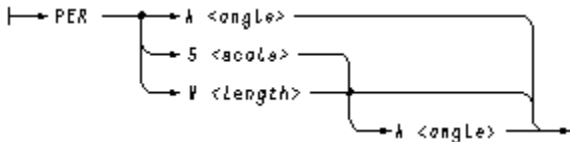
Defines the position in the viewbox at which the image of the TO point appears. The values of *x_position* and *y_position* must be in the range 0 through 1. The bottom left corner of the screen is taken as 0 0 and the top right corner is 1 1.

PAR



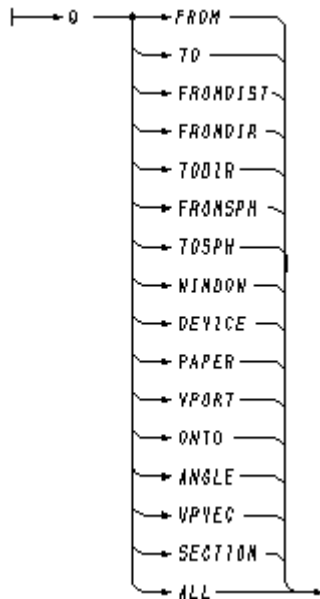
Sets an axonometric view by specifying the *scale* factor or window *length*.

PER



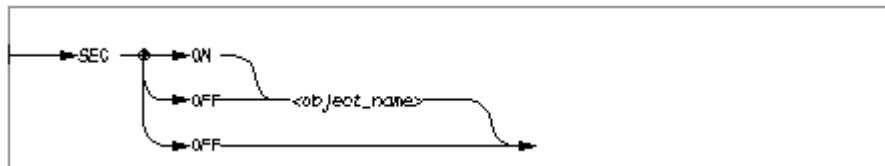
A perspective view can be set by specifying different parameters or a combinations of them. The values to define are perspective *angle*, *scale* factor and window *length*.

Q



Returns the value of the specified parameter. `ALL` shows all values.

SEC



Used with the `OFF` option, cancels the `SECTION` command.

Used with the `object_name` option, selects a certain object. In combination with the option `ON` this means that all other objects are not sectioned, with `OFF` all others are sectioned. So, if most of the objects in the model must be cross-sectioned, use the `OFF` option to define certain objects which should not be sectioned.

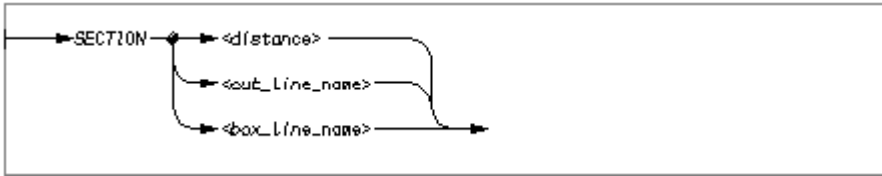
An error message is issued if both `SEC ON` and `SEC OFF` commands appear in one view box.

SEC ONLY



Produces a view of the sectioned surface only. The 3D crosshatching is also viewed.

SECTION

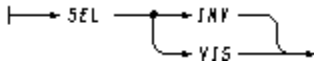


This command produces a cross-sectional view of the model according to the set option. The sectioning plane is normal to the viewline.

For the option *distance* the sectioning plane cuts the viewline at the specified *distance* from the TO point. *distance* is measured along the viewline towards the viewpoint. At most one SECTION command with this option may occur for any view box.

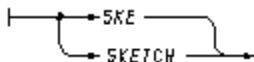
The *cut_line_name* and *box_line_name* options produce a cross-sectional view of the model based on the defined cutting line or box. Names of both the cut line and box are limited to twelve characters, beginning with an alphabetic letter. More than one SECTION command with these options may be used also in conjunction with one SECTION command with the *distance* option.

SEL



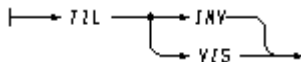
Sets self-intersection lines to be visible (VIS) or invisible (INV).

SKETCH



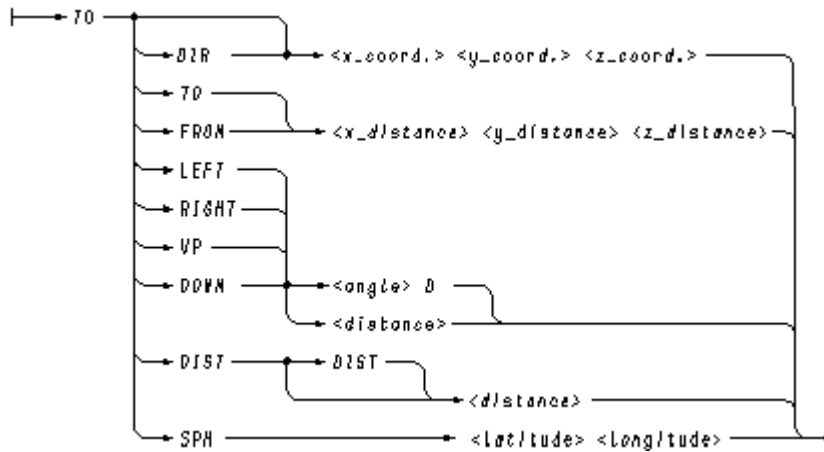
Produces a quick view of the model. For this things like silhouettes, hidden lines or sectioning are not generated.

TIL



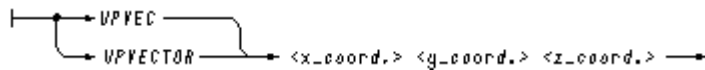
Sets the tile boundaries to be visible (VIS) or invisible (INV). The default is TIL INV.

TO



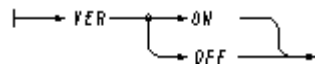
Moves the TO point (the aiming point) to the required position.

UPVEC



Specifies the direction of the upvector. The default is 0 0 1 (Z-axis up).

VER



Sets graphical verification ON or OFF. The default is VER ON.



THE MODEL VIEWER

This chapter describes how you can use the Model Viewer to visualize a model.

Please note: The model viewer is limited to display 800000 facets. If this limit is exceeded a message that the data has been truncated is displayed and the model objects loaded so far are displayed.

- [The Model Viewer Dialog..... 344](#)
- [Default Settings..... 347](#)

The Model Viewer Dialog


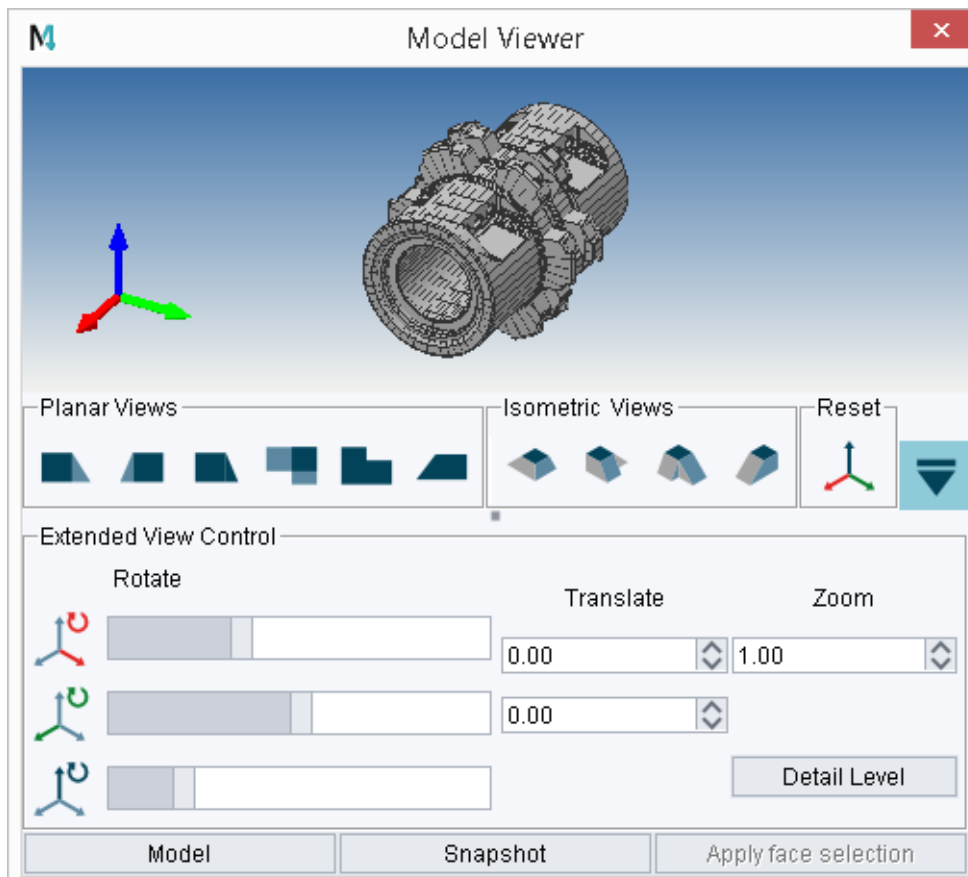
To open the Model Viewer dialog choose the Interactively view current model tool  from the Model + View tool group in the 3D tab. If only the upper part of the dialog is displayed, click on the arrow down right to open the lower part of the dialog.

Figure 272 The Model Viewer Dialog



The upper part of the dialog shows the isometric display of the model of the current sheet. The symbol on the left side gives the direction of the x, y, and z coordinates. Figure 272 shows the default view direction and settings, when you open the dialog.

You can change the view direction (by pressing one of the buttons in the Planar Views or Isometric Views areas), rotate and zoom the model. If you move the cursor over the model, hold the MMB pressed and move the mouse, the model rotates continuously variable so that you can view the model from any direction. Use the mouse wheel to zoom in and out.

The areas and options of the dialog are explained in the following:

Planar Views

You can choose between six view directions perpendicular to one side of the model: Front View, Right View, Left View, Top View, Bottom View and Back View.

Isometric Views


You can choose between four isometric view directions: Spatial Right View, Spatial Left View, Spatial Back View and Spatial Front View.

Reset

sets all settings back to the defaults.



Arrow button

opens the lower part of the dialog with the Extended View Control entry fields and the Model, Snapshot and Apply face selection buttons.  closes the lower part.

Extended View Control

Rotate

With the help of the sliders you can rotate the model smoothly in three different directions around the x, y and z axis.

Translate

moves the model on the x-axis (upper field) and y-axis (lower field) according to the inserted values

Zoom

scales down (value > 1) or magnifies (value < 1) the model.

Model

opens the Open Model File window, where you can select a model file from an arbitrary directory, irrespective from the current sheet, which you want to be displayed in the Model Viewer.

Snapshot

opens the Filename dialog in which you can choose a directory for saving the currently displayed model as Bitmap (.bmp) or Portable Network Graphics file (.png).

Snapshot

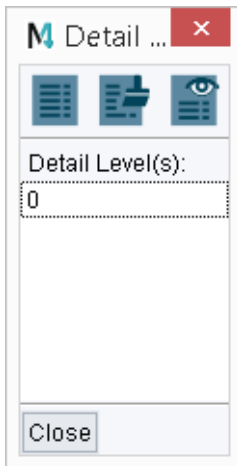
öffnet den Dialog Dateiname, in dem Sie ein Verzeichnis wählen können, um das derzeit angezeigte Modell als Bitmap- (.bmp) oder Portable-Network-Graphics-Datei (.png) zu speichern.

Please note: On machines where the snapshot function fails setting the environment variable MED_SNAP_PIXMAP can be tried to see if that gives a better result.




Detail levels

Opens the Detail Levels dialog.

Figure 273 Detail Level Dialog




If several detail levels were assigned to an object with the model definition, these are listed in the dialog (for more information on the detail levels see [“Specifying the Detail Level” on page 288](#)). When the Model Viewer dialog opens, by default the whole model is firstly displayed. Now you are able to display only one, several selected or all detail levels. For this the icons at the top of the dialog are used:

Icon	Tool Tip	Meaning
	All	Selects all detail levels in the list. If you click the Apply button, the complete model is displayed in the Model Viewer dialog.
	Clear	Deselects all detail levels in the list. If you click the Apply button, the Model Viewer dialog remains empty.
	Apply	Displays the detail levels selected in the list in the Model Viewer dialog.

Quit the dialog via the Close button.

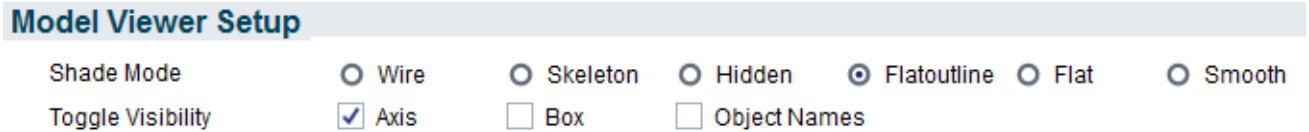
Apply face selection

The button is only activated, if you choose the From Face tool . Then you can select a face in the displayed model and press the button to create an oblique view of this face in the 3D sheet (for details to this functionality see [“Creating an Oblique View By Selecting a Face in the Viewer” on page 316](#)).

Default Settings

The default settings of the Model Viewer can be changed in the Default Settings available via File > Default Settings > 3D Products.

Figure 274 Default Settings for the Model Viewer



Shade Mode

controls the display of a model. Flatline is the default setting. Following figures show the display for Skeleton and Flat.

Figure 275 Model Viewer Dialog - Skeleton

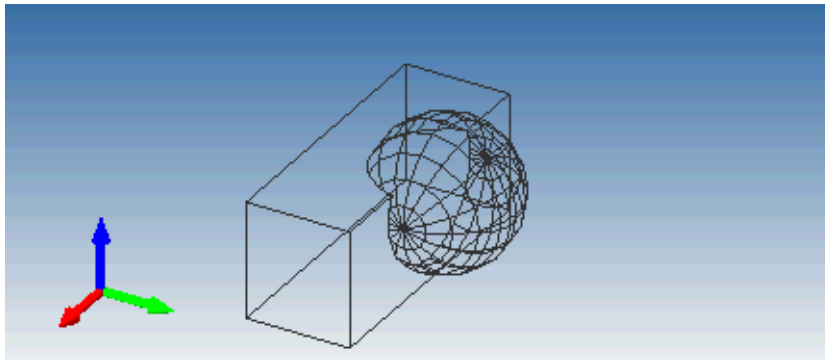
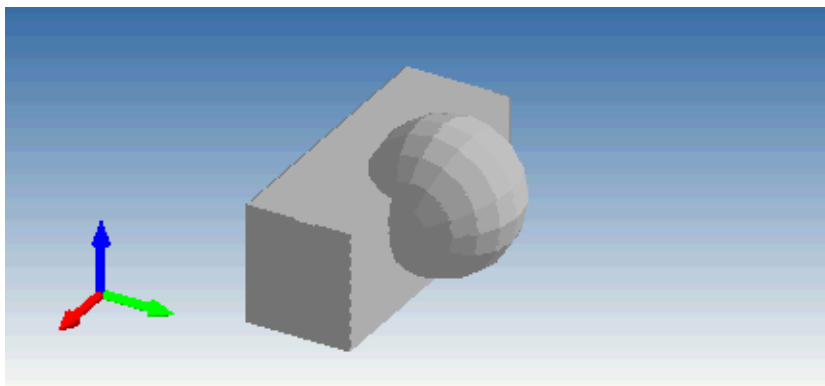


Figure 276 Model Viewer Dialog - Flat Option



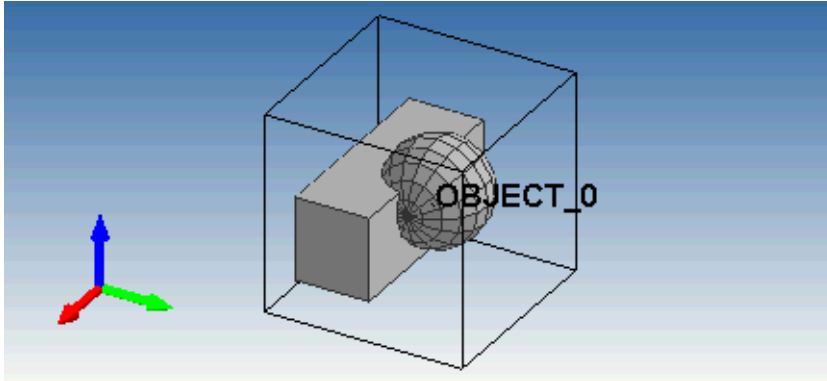
Toggle visibility

the options Axis (default setting), Box and Object Names are available. Several options are possible at the same time. In the example below all three options are activated.

- Axis
shows the coordinate axes
- Box
encloses the whole object by a box

- Object Names
displays the name of an object

Figure 277 Model Viewer Dialog - Toggle visibility



For more details on this dialog see ["Introduction"](#), ["The 3D Default Settings"](#) on page 15.

RECONSTRUCTION COMMANDS

The views of the model generated by the Viewer are translated into the data structure of the two-dimensional drawing sheet and then drawn, or reconstructed, in the sheet viewboxes. The Reconstruction commands control the reconstruction process.

- General Notes 350
- Specifying the Group Type 351
- Specifying the Line Type and Layer 352
- Specifying the Reconstructed Geometry Group Structure 355
- Specifying the Reconstructed Geometry Identifier Text 356
- Specifying Database Keys 357
- Curve Reconstruction Accuracy 359
- Specifying the Type of Reconstructed Arc 359
- Controlling Line Compression 360
- Highlighting the Sectioned Points on a Wire 360

General Notes

Commands are provided to control this reconstruction process, and they give you extensive control over how the views are reconstructed on the 3D sheet. All reconstruction commands are specified as TRS text of style `Reconstructor Text` on the sheet and they apply globally, that is, they affect all viewboxes on the sheet.

Using Attributes or Texts

Alternatively to the method described in the following, where commands are given by texts, you can give the same commands by adding 3D attributes (see [“3D-Attributes” on page 37](#)).

Using Style or Type

Wherever `<line_type>` or `<text_type>` is given in the syntax of a `TARGET` command you can **alternatively use style names** in quotes. For example:

```
TARGET VIS SL0 3
TARGET VIS "elucidation" 3
```

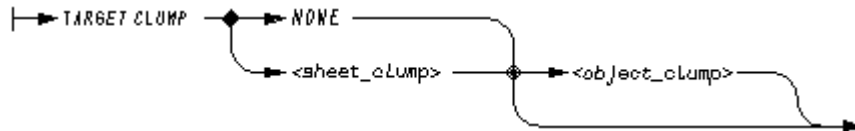
Wherever `<clump>` is given in the syntax of a `TARGET` command you can **alternatively use a group name** in quotes. For example:

```
TARGET CLUMP S2T SET
TARGET CLUMP "Scene" "3D Object"
```

Specifying the Group Type

When views of a model are drawn onto a 3D sheet, the lines that make up those views can be created as sheet level lines or as members of a group. It is normal to reconstruct lines at group level with the `TARGET CLUMP` command specifying the group type. The default group type used for reconstructed lines is `Group (3D Object)`.

The syntax of the `TARGET CLUMP` command is shown below:



This command can also be used to create separate groups for storing the lines that make up individual objects in a model.

For example, to produce a sheet level S2T group for each view of the complete model, and have each of these groups contain a `Group` type group for each object in the model, place the following command on the 3D definition sheet:

```
TARGET CLUMP S2T Group
```

If you want to create the reconstructed lines at sheet level, rather than as set lines in a group, use the command:

```
TARGET CLUMP NONE
```

Please note: As the reconstructed lines are by default stored in groups, SL line types are used. However, when reconstructing the view using lines at sheet level, SL line types cannot be used and must be changed to sheet level lines. The sheet level lines required must be specified using the `TARGET` command. The `TARGET` command is described [“Specifying the Line Type and Layer” on page 352](#).

Specifying the Line Type and Layer

The type and layer of reconstructed lines is specified by the `TARGET` command. There are five types of reconstructed lines:

- 3D Section Cut lines, which are created when an object is sectioned
- Object lines, which show the basic shape of the object
- Implied edge lines, which represent the intersection lines between intersecting, non-Boolean objects
- 2D and 3D crosshatching lines, which are created when the surface of a sectioned object is crosshatched
- Section edge lines, which represent the edge lines of the sectioning object

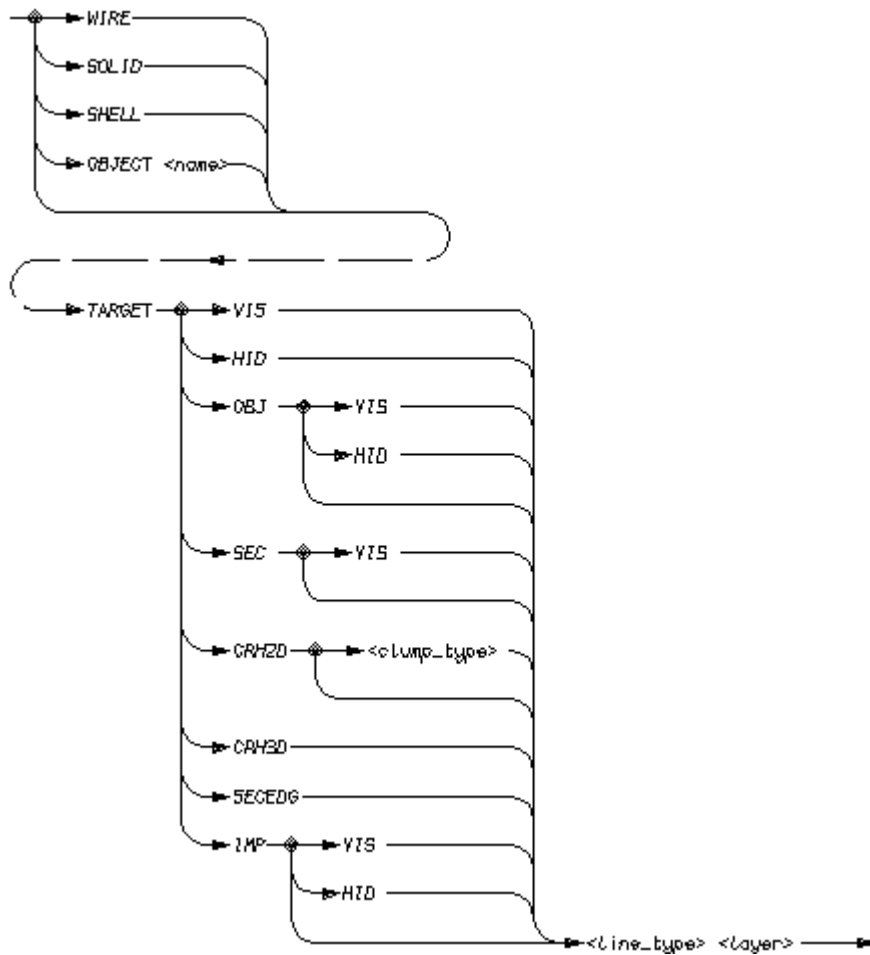
The `TARGET` command also allows you to distinguish between visible lines (`VIS`) and hidden lines (`HID`) for object and implied lines. Section lines are always visible.

The `TARGET` command can be used to reconstruct lines from the following objects:

- Wires
- Shells
- Solids
- Named objects
- Named sub-objects within an instance group

Objects are named at the model generation stage, using the `NAME name` or `& name` command. Named sub-objects are objects which have been given a name within an instanced model. For example, suppose the model `SHAFT` contains the named objects `COG` and `CLIP`. These would be referenced as `SHAFT.COG` and `SHAFT.CLIP` respectively.

The syntax of the `TARGET` command is shown below. (Notice that the order of commands is important in some cases and examples of this are shown on [page 353](#)):



Please note: `HID` sets the line style to be used by the Viewer command `HL DOT`. With `CRH2D` (2D crosshatching), an optional group may be specified in which the crosshatching line will be created.

Examples of the TARGET Command

The following examples show the use of the `TARGET` command to specify the style and layer of reconstructed lines:

```
TARGET SEC "solid_thick" 50
```

Specifies that all section lines are of style `solid_thick` on layer 50.

```
TARGET OBJ VIS "solid_thin" 103
```

Specifies that all visible object lines are to be drawn using line style `solid_thin` and placed on layer 103.

TARGET IMP "chain_thin" 3
Specifies that all the implied edges of a model be drawn using line of style `chain_thin` and placed on layer 3.

WIRE TARGET HID "dotted_thin" 108
Specifies that all hidden lines of wire objects be drawn using line style `dotted_thin` placed on layer 108.

SHELL TARGET OBJ HID "dashed_short_thin" 39
Reconstructs all hidden lines that define the shapes of shell objects as style `dashed_short_thin` lines on layer 39.

SOLID TARGET OBJ "solid_medium" 91
Specifies that all lines used to show the shape of solid objects be drawn using line style `solid_medium` placed on layer 91.

OBJECT PISTON TARGET VIS "solid_medium" 40
Reconstructs all the lines (including any section lines) of a named object called **PISTON** as style `solid_medium` lines on layer 40.

OBJECT SHAFT TARGET CRH2D SET "dashed_short_medium" 36
Reconstructs all the 2D crosshatching lines of the sectioned surfaces of an object named **SHAFT** in a **SET** group as style `dashed_short_medium` lines on layer 36.

OBJECT PISTON TARGET SECEDG "dashed_long_medium" 6
Reconstructs all the edge lines of a sectioning object named **PISTON** as style `dashed_long_medium` lines on layer 6.

OBJECT SHAFT TARGET CRH3D "dashed_short_medium" 41
Reconstructs all the 3D crosshatching lines of the sectioned surfaces of an object named **SHAFT** as style `dashed_short_medium` lines on layer 41.

OBJECT SHAFT.COG TARGET VIS "dotted_thin" 3
Reconstructs all the lines of a named instanced sub-object called **SHAFT.COG** as style `dotted_thin` lines on layer 3.

OBJECT SHAFT.* TARGET INV "hidden_lines" 4
Reconstructs all the invisible lines as style `hidden_lines` on layer 4 for the entire instanced model.

Specifying the Reconstructed Geometry Group Structure

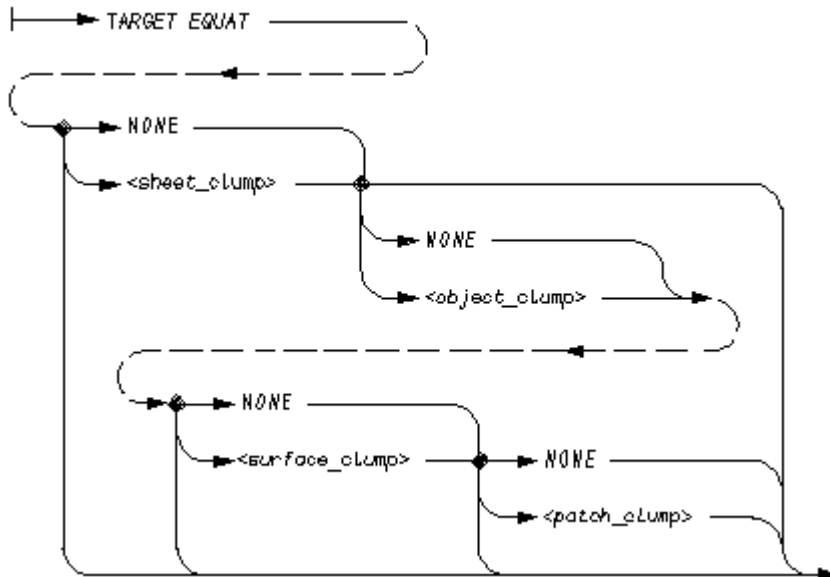
The command `EQUATIONS ON` placed on a model definition sheet as TMS text creates surface information in the model file, and the reconstructed views of the model are produced with a default data structure which separates surface information out into individual groups.

This default group structure for the reconstructed geometry can be overwritten by using the `TARGET EQUAT` command. The use of this command is in customizing the group structure so that surface information can be picked up from the sheet in a certain form, usually for passing downstream to custom applications.

Using the `TARGET EQUAT` command you can overwrite, or customize, the default group structure which contains the reconstructed geometry. This enables you to pick up surface information from the sheet in the form required, for example, by custom applications.

Please note: The `TARGET EQUAT` command has no effect unless the `EQUATIONS ON` command is used on the model definition sheet.

The syntax of the `TARGET EQUAT` command is shown below.



For example, to produce:

- No sheet level groups
 - An object level S3T group for each object in the model
 - An S2T group for each surface on each object
 - A SET type group for each patch on that surface for each of these groups
- you would place the following command on the 3D definition sheet:

```
TARGET EQUAT NONE S3T S2T SET
```

This is in fact the default for this command.

Specifying the Reconstructed Geometry Identifier Text

The reconstructed geometry is identified by invisible text placed in the groups. The default is TGI-text on layer 3, but this can be overwritten with the `TARGET GID` command.

The syntax of the `TARGET GID` command is shown below.

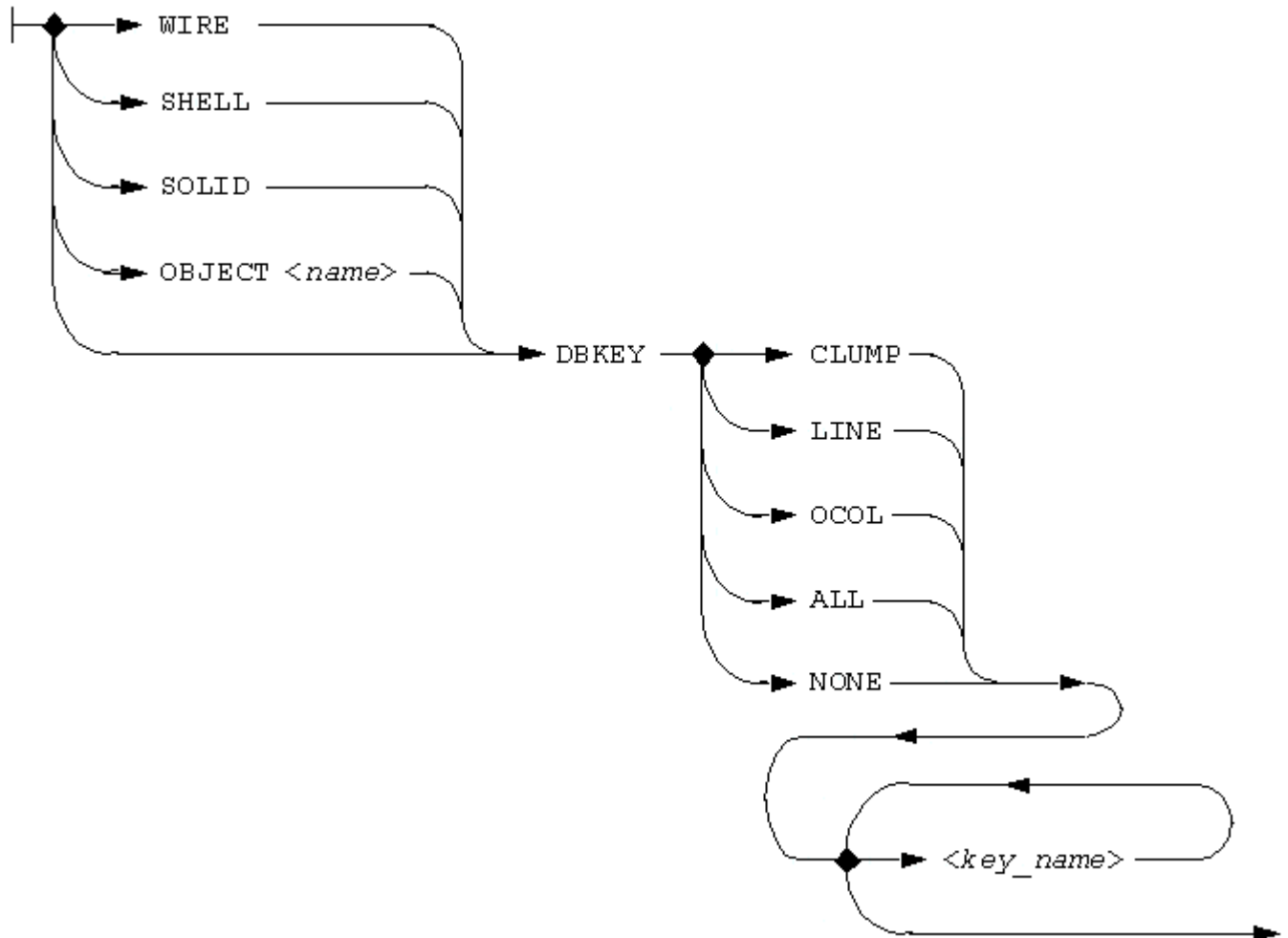
```
|——▶TARGET GID ——▶ <text_type> ——▶ <layer> ——▶
```

The identifier text created can be made visible on the sheet by switching to code 8 (enter `cod 8`) and running the sheet through the Viewer.

Specifying Database Keys

The `DBKEY` command is used to determine the reconstruction of database keys. Database keys are named in the model file by UPR texts (uncommitted properties) in the Assembly Manager.

The syntax of the `DBKEY` Command is shown below.



The `DBKEY` command can be applied to the following objects in the model file:

- Wires
- Solids
- Shells
- All objects

The command specifies the level at which database keys are reconstructed. They can be reconstructed at:

- Group level
- Line level

- Color level
- All levels
- You can also specify that no reconstruction takes place for a database key using the DBKEY NONE icon. This is the default.

Individual database keys can also be selected for reconstruction. For example, the command:

```
DBKEY LINE PIPE1 PIPE2
```

specifies that only the database keys PIPE1 and PIPE2 are reconstructed, and that they are reconstructed at line level.

If you do not specify any database key names, all keys are reconstructed at the selected level. For example, the command:

```
SHELL DBKEY CLUMP
```

specifies that all database keys for shell objects are reconstructed at group level.

For color level database key commands *key_name* represents the RGB values:

```
DBKEY OCOL <no1> <no2> <no3>
```

where the numbers <no1> <no2> <no3> are integer values.

The DBKEY command is specified using TRS-text of style `Reconstructor Text`.

Curve Reconstruction Accuracy

Curved lines are produced by the Modeler and Viewer as straight line segments that follow the outline of the facets of an object. The `CURVES` command specifies how the curves are drawn onto a sheet.

The system tries to fit the best conic between the points produced by the Viewer. It passes a conic through the first and last points of a surface boundary line, and the maximum amount by which the conic deviates from this line is known as the curve tolerance.

To set the curve tolerance, use the command:

```
CURVES tolerance_value
```

A smaller *tolerance_value* produces a more accurate representation of a curve than a larger value. The standard command block on a 3D sheet contains the `CURVES` command. Note that `CURVES tolerance_value` is specified in sheet units. The Modeler command `CHOTOL` is specified in model units.

Dimensioning Reconstructed Curves

You should be aware that a reconstructed curve which theoretically should be a circular arc can be radially dimensioned, but the resulting dimension value cannot be guaranteed to be the true value in all cases. This is because a conic may have been produced because a circular arc did not fit when `CIR ON` was in force, or the curve was produced without `CIR ON`. This situation may arise after using Boolean operations.

For dimensioning purposes you can improve curve accuracy by reducing the `CHOTOL` value. This increases the number of points in the curve and so improves accuracy, but note that it also degrades performance.

Specifying the Type of Reconstructed Arc

The `CIR` command specifies the type of arc used for reconstruction. There are two types:

- Tangent-point arcs
- Center-point arcs

By default, tangent-point arcs are produced. Center-point arcs can be produced using the command:

```
CIR ON
```

The `CIR` command operates in conjunction with the `CURVES` command. For example, in the case where `CIR ON` is used and there are six or more points in a line, the Viewer attempts to

fit a circular arc to the line minus the two end points. If it fails it attempts to fit a conic to the line. If it succeeds, the Viewer then attempts to include the end points in this arc by checking that they fall within the `CURVES` tolerance (see “[Curve Reconstruction Accuracy](#)” on page 359). If they do not, as may be the case after Boolean operations, each end segment of the curve is output as a straight line, and the rest of the curve remains as a circular or conical arc.

Please note: If the Viewer fails to fit a conic to the line, it fits a straight line instead.

As an alternative to the above, you can suppress arc production by ensuring that only straight line segments are output for all reconstructed curves. Do this by specifying a small `CURVES` tolerance, for example:

```
CURVES .01
```

Controlling Line Compression

By default, reconstructed lines are compressed into as few MEDUSA4 lines as possible. They are joined by invisible line segments. This compression minimizes the size of sheets and makes it easier to perform certain drafting operations, for example, crosshatching of section lines. However, you can reconstruct each line separately using the command:

```
PACK OFF
```

The command `PACK ON` is the default.

Highlighting the Sectioned Points on a Wire

The points at which a wire is cut through by a section (the sectioned points) can be highlighted by using the command:

```
SECFUN point_function
```

This specifies the type of point function that is placed at these points. For example, to specify FUNV 20 type points, use:

```
SECFUN 20
```

THE MODEL VALIDATOR

The Model Validator is used to determine the validity of a solid model or of a solid object within a model. A valid model is a model in which all tile edges and patch boundaries meet exactly, neighboring edges are in opposite directions and share a unique identifier (GID), and there are no spurious gaps or holes.

It is important to use the Model Validator on all solid models to verify their integrity before continuing beyond the design stage. This is because further Boolean operations, viewing in either of the Shaders, and mass property calculations will not work on invalid models. The only exception to this is the case where, in a Boolean operation, an invalid object does not form part of the resultant model.

The Model Validator takes a model file as input and generates a report on the model's validity which can be displayed on the screen or written to a specified file.

- [Running the Validator.....](#) 362
- [Basic Commands.....](#) 363
- [Validating Individual Objects.....](#) 364
- [Syntax of Commands.....](#) 365

Running the Validator

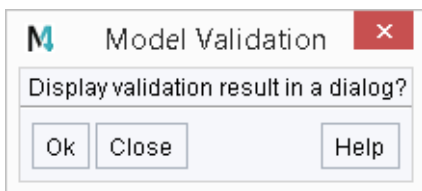
The Model Validator can be accessed either from inside MEDUSA4 using the relevant button or from outside MEDUSA4 using the MEDUSA4 Utility Access facility, MEDUTIL.

Running the Validator inside MEDUSA4

Choose the Check validity of current model  button to run the Model Validator program.

When the scan is finished the Model Validation window appears.

Figure 278 Model Validation Dialog



- OK opens a dialog which displays the validation result.
- Close closes the window.

Running the Validator outside MEDUSA4

The program can be entered outside MEDUSA4 using the `VALID` command in MEDUTIL, the MEDUSA4 Utility Access facility. The following example shows a brief introduction to this facility. Start MEDUTIL by using the `medutil` command:

```
medutil projectname

MEDUSA4 Utility Control
~~~~~
Type help for list of commands...
Enter command>:valid
MEDUSA4 Model Validator Program
~~~~~
*macro <medusa4 installation path>\med3d\m3d\macro\valid.mac
Valid>help
```

Model Validator commands can be entered at the `Valid>` prompt.

Please note: Do not try to run MEDUTIL within MEDUSA4. This command starts too many processes and generates an error message.

Basic Commands

The commands in this section can be entered individually from the keyboard or by executing a macro file.

1. Enter the Model Validator using the procedure described in [“Running the Validator” on page 362](#).

2. At the `Valid>` prompt, specify the name of the model file to be validated with the command:

```
IN filename
```

3. If you do not want to see the results on the screen, specify an output filename, using the command:

```
OUT filename
```

4. To start the validation process, type the command:

```
GO
```

You can repeat this sequence of commands (`IN`, `OUT`, `GO`) as many times as required to validate different model files or to direct output to different destinations.

5. To leave the Model Validator, type the command:

```
QUIT
```

The following is an example of Model Validator output to the screen. User input is shown in boldface type.

```
MEDUSA4 Model Validator Program
~~~~~
*macro <medusa4 installation path>\med3d\m3d\macro\valid.mac
Type HELP for a list of Model Validator commands
Valid>in block.mod
Valid>go
Validating: BLOCK.MOD
Object number 1
Object name
***VALID***
Valid>quit
```

In this example the model is valid. If the Model Validator finds an invalid model, an error message is displayed which explains why the model has failed the validation. A list of possible error messages is provided in [“Error Messages”](#), [“Model Validator Error Messages” on page 541](#).

Please note: A self-intersecting model passes through the Model Validator without generating an error message. Difficulties may arise if Boolean operations or geometric property calculations are performed on a self-intersecting model.

Validating Individual Objects

It is often useful to be able to validate one object in a model file, instead of the whole model. This is possible if the object has been either uniquely named or created on a different detail level to the other objects at the model generation stage.

Named objects can be specified by the command:

```
OBJ name
```

All objects can be specified by the command:

```
OBJ ALL
```

If objects in the model have been created at different detail levels, specific levels can be selected using the `DET` command. The detail level can be switched ON or OFF as required. Only detail levels switched ON are validated. 256 detail levels are available, numbered 0 through 255.

For example:

```
DET level OFF
```

Switches the specified detail level OFF.

```
DET level_1/level_2 ON
```

Specifies a range of detail levels.

```
DET 3/20 ON
```

Validates all objects on detail levels 3 through 20.

```
DET level /* OFF
```

Switches all levels from the specified level to the last level OFF.

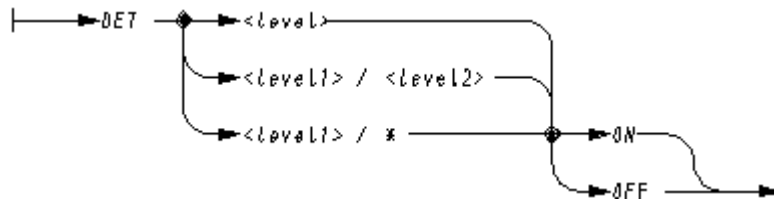
```
DET 200/* OFF
```

Turns OFF detail levels in the range 200 through 255.

Syntax of Commands

Commands may be typed in uppercase or lowercase. They are shown in uppercase for clarity. The commands in this section are presented in syntax graph form.

DET



Switches the detail level(s) ON or OFF. Specify *level* as an integer in the range 0 through 255.

GO

Starts the validation process.

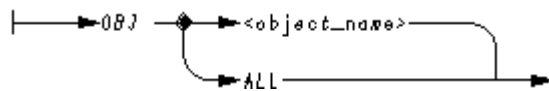
HELP

Lists the commands available.

IN filename

Specifies the name of the input model file. Specify *filename* as a text string.

OBJ



Selects a named object or all objects to be processed. The default is OBJ ALL which selects all objects. Specify *object_name* as a text string.

OUT filename

Specifies the name of the output file. Specify *filename* as a text string.

QUIT

Leaves the Validator and returns to MEDUTIL or MEDUSA4.



EXTRACTING MASS PROPERTIES

The MEDUSA4 Mass Properties program enables you to analyze a model held in a file previously created by the MEDUSA4 Modeler.

- Introduction 368
- Default Settings..... 369
- Running the Mass Properties Program..... 370
- Specifying Individual Objects 372
- Properties..... 374
- Black Boxes 375
- Units 376
- Sheet Output..... 377
- Sheet Output..... 377
- User Defined Axes 379
- Example of Mass Properties Program Output 380
- Commands..... 382

Introduction

The following information can be generated by the Mass Properties program:

- Density and mass
- Volume and surface area
- Radius of gyration about each axis and about each principal axis
- Moments of inertia about each axis
- Principal central moments of inertia
- Products of inertia
- Position of centroid and the principal axes through centroid
- Maximum extent of the object in the X, Y, and, Z-axes

If the model file contains more than one solid object, you can obtain details for each solid in the file, any combination of solids in the file, or totals for all objects specified with the `OBJECT` command.

The commands allow you to determine which particular properties of which objects are required, and you can set the units. Commands may be entered directly from the keyboard, or with a macro file.

Default Settings

The default settings for the mass properties program can be changed in the Default Settings dialog which is opened by File > Default Settings. It provides a 3D Products tab, where you can specify the default settings for Mass Properties.

Figure 279 The Mass Properties Entries in the Default Settings

Mass Properties	
Default Density	7850.000000
Length Units	Meters
Mass Units	Kilograms
Calculate for:	All Properties

Default Density

used for objects for which no other density was specified with the linkline definition.

Length Units

default unit for lengths.

Mass Units

default unit for masses.

Calculate for

properties for which calculation is performed.

Running the Mass Properties Program

There are two possibilities to access the Mass Properties program:

- from the operating system level "Using the MEDUTIL Command" or
- by "Using the Call the 3D Properties Program Tool" on page 371.

Please note: The Mass Properties program must be run using a modeled sheet only.

Using the MEDUTIL Command

To run the Mass Properties program with MEDUTIL:

1. Enter **PROP** at the `Enter command>` prompt.

The **PROP** command starts the Mass Properties program.

2. Once the program has started, specify the model file with the command:

IN pathname

3. To send the output to an output file enter the command:

OUT FILE pathname

The output goes to the terminal by default if the **OUT** command is omitted. To display output at the terminal, in addition to sending it to an output file, enter:

OUT TTY -ON

When output to the terminal is no longer desired, you can turn it off with:

OUT TTY -OFF

Output may also be sent to a MEDUSA4 sheet (see "Sheet Output" on page 377 for details).

4. To initiate the calculation process, enter:

GO

Please note: When calculating mass properties in MEDUTIL the default units for length, mass, and the default density are defined in a macro named *prop_defs.mac*. A template/default version of this file is located in the *med3d\m3d\macro* directory of the MEDUSA4 installation which sets these values to Meters, Kilograms and 7850 Kg per cubic meter. If want to use different default values, copy that macro to the equivalent location in your project area (for example *master_project\M3D\MACRO*) and modify it as required.

Using the Call the 3D Properties Program Tool


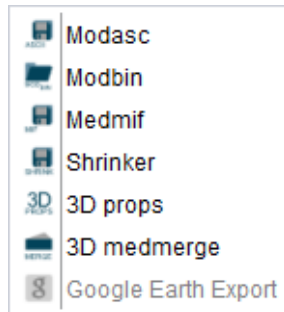
Choose the 3D props tool  which is located in the following toolset.

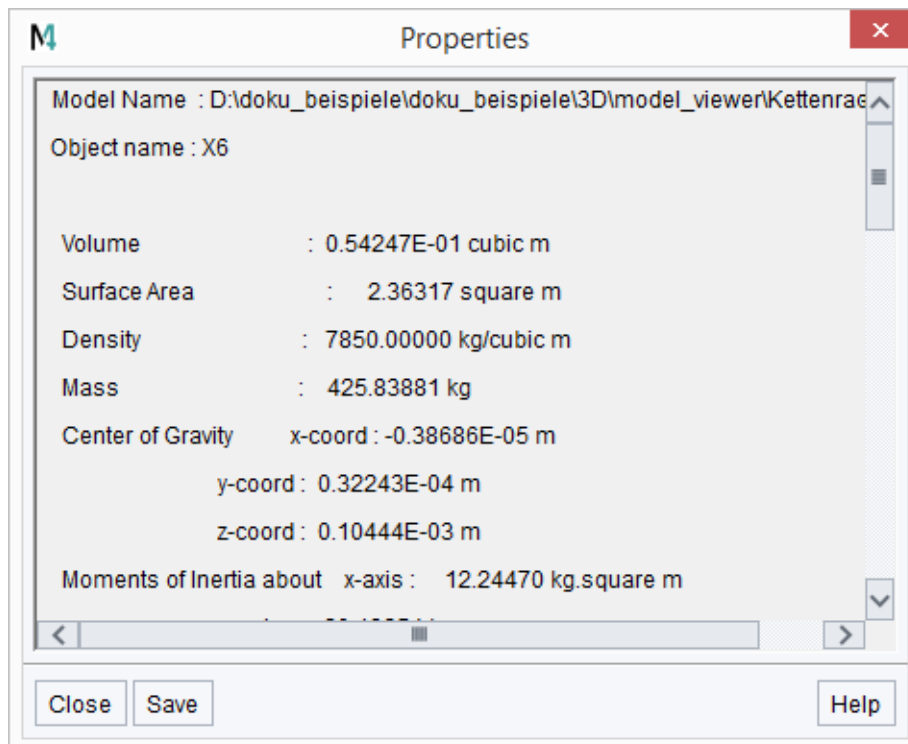
Figure 280 Toolset Containing the 3D Props Tool



The 3D properties program is started promptly.

The Properties dialog opens listing any properties of the current model.

Figure 281 The Properties Dialog



Specifying Individual Objects

By default, all solid objects and shells found in the model are included in the calculations. A separate list of details is produced for each object, and a total is given. To obtain details for an individual object only, the object can be named using TMG-text of style `Modeller Text ass.` by `Geometry`. Text data must be attached to the link line or the Modeler will not recognize the text. Unnamed objects are given the name `UOBJ-nnn` where `nnn` is the number of the object in the model file, for example `UOBJ-1` is the first object, `UOBJ-23` would be the 23rd.

You may request a specific object and specific properties using the `OBJECT` command:

```
OBJECT object_name property_list
```

The *property_list* for an object can be any number of the listed properties. The property list is also used to define certain properties for objects:

Property	Definition
DENSITY decimal_value	Specifies density as decimal_value
MASS decimal_value	Specifies mass as decimal_value
VOLUME decimal_value	Specifies volume as decimal_value
DENSITY	Finds the density
MASS	Finds the mass
VOLUME	Finds the volume
MI	Finds the moments of inertia about the coordinate axes
AREA	Finds the total surface area
RG	Finds the radii of gyration
PRG	Finds principal radii of gyration
PAXES	Finds the principal axes
PI	Finds the products of inertia
CENTROID	Finds the centroid
P MOMENTS	Finds the principal moments of inertia
MAX EXTENT	Finds the maximum extent of the object in the X, Y, and Z directions along the principal and object axes
NONE	Finds no properties for this object
ALL	Finds all properties for this object

For example, to find the properties of volume, mass, and radii of gyration for an object called RIVET, enter:

```
OBJECT RIVET VOLUME MASS RG
```

To set the density to 10.0 of the object called RIVET, enter:

```
OBJECT RIVET DENSITY 10.0
```

The density of 10.0 for RIVET overrides the current default density used for the model.

If desired, the MASS of RIVET can be defined by entering:

```
OBJECT RIVET MASS 100.0
```

This command, which is analogous to OBJECT RIVET VOLUME MASS RG, causes the current default density and the defined density to be overridden by the value determined from the calculated volume and the defined mass.

Density and mass may be defined for all objects in the model and all black boxes (see “[Black Boxes](#)” on page 375). In addition, the volume of black boxes may be defined. However, the volume of objects contained in the model may never be defined, because this value is always calculated.

Properties

To request a specific property for a list of objects, use the `PROP` command:

```
PROP property object_list
```

This command works in a slightly different manner from the `OBJECT` command. Here, `property` is any one of the properties from the property list “[Specifying Individual Objects](#)” on page 372.

The `object_list` is a list of valid object names (TMG-text of style `Modeller Text of ass. Geometry` on the sheet).

`PROP` calculates the specified property for each valid object name in the object list. You are informed of any invalid names that may be in the list.

For example, to request the calculation of the volume of two objects named `RIVET` and `BOLT`, enter:

```
PROP VOLUME RIVET BOLT
```

Requested properties for all objects can be reset to all or no properties. This is useful if you make a mistake in specifying properties for an object, and/or you wish to re-specify all the properties. For example:

```
ALL ALL
```

Resets all properties for all objects.

```
ALL NONE
```

Resets no properties for all objects.

Please note: `ALL` can be used instead of listing the names of every object. If no object name is given, `ALL` is assumed.

Black Boxes

Black boxes are objects not currently contained in the MEDUSA4 model. In other words, a black box can be any hypothetical object you imagine. For example, you may wish to add auxiliary fuel tanks to the wings of a modeled plane that does not currently have them included in the model. To get an idea of the effect of the boxes (tanks) on the properties of the model, specify each box's center of gravity along with minimal descriptive information. There are no predefined black boxes.

You may define the mass, density, and/or volume for each black box as shown in the examples on [page 372](#). However, along with the center of gravity, you must specify either the mass or the volume. If the density is not specified, the current default density is used. You cannot specify shape.

The properties calculated for each black box are added to the ensemble properties of the model in the same manner that the properties for objects contained in the model are added. However, black boxes have no effect upon the properties of surface area or maximum extent.

For example, to define a black box named `PLATE` with a center of gravity at 100.0, 200.0, 300.0, enter:

```
BLACK BOX PLATE 100.0 200.0 300.0
```

To define the properties of `PLATE`, either the `OBJECT` or `PROP` command is used.

```
OBJECT PLATE MASS 168.5 VOLUME 253.75
```

This command sets the mass of `PLATE` to 168.50 and the volume of `PLATE` to 253.75. The density of `PLATE` is calculated from the values for `MASS` and `VOLUME` as $\text{MASS}/\text{VOLUME}$, or $168.50/253.75$. This yields a density value of 0.66404.

Units

You may request one of a standard set of units, or you may define a unit in terms of one of the standard units by means of a conversion factor. Any of the standard units may be selected using the command syntax:

```
UNITS unit_name
```

The commands to select the standard units are:

Command	Description
UNITS MM	Sets length units to millimeters.
UNITS CM	Sets length units to centimeters.
UNITS DM	Sets length units to decimeters.
UNITS M	Sets length units to meters.
UNITS FT	Sets length units to feet.
UNITS YD	Sets length units to yards.
UNITS G	Sets mass units to grams.
UNITS KG	Sets mass units to kilograms.
UNITS OZ	Sets mass units to ounces.
UNITS LB	Sets mass units to pounds.

To define other units, you must give one of these commands:

```
LENGTH unit_name conversion_factor standard_unit
```

```
MASS unit_name conversion_factor standard_unit
```

The *conversion_factor* specifies the defined *unit_name* as a factor of a *standard_unit*.

For example:

```
LENGTH MILES 1760 YD
```

```
MASS TONNE 1000 KG
```

To set the default density, the command is:

```
DENSITY decimal_value
```

This is, the density value that is used whenever an object's density has not been set using the `PROP` or `OBJECT` commands.

The density value is assumed to be given in terms of the defined units. Thus, to set a default density of 10 lb/cu ft, the following command sequence is used:

```
UNITS LB
```

```
UNITS FT
```

```
DENSITY 10
```


Sheet Output

You may request sheet output in interactive mode simply by typing: `OUT SHE`

The properties are stored on the sheet in tabular form above the maximum window area. The `WINM` command can be used to bring the table into view. The properties appear as shown below.

CMX	X-coordinate of center of mass
CMY	Y-coordinate of center of mass
CMZ	Z-coordinate of center of mass
COSX1	X-direction cosine of principal axis 1
COSY1	Y-direction cosine of principal axis 1
COSZ1	Z-direction cosine of principal axis 1
COSX2	X-direction cosine of principal axis 2
COSY2	Y-direction cosine of principal axis 2
COSZ2	Z-direction cosine of principal axis 2
COSX3	X-direction cosine of principal axis 3
COSY3	Y-direction cosine of principal axis 3
COSZ3	Z-direction cosine of principal axis 3
DENS	Density
MASS	Mass
MAXXO	Maximum X-coordinate
MINXO	Minimum X-coordinate
MAXYO	Maximum Y-coordinate
MINYO	Minimum Y-coordinate
MAXZO	Maximum Z-coordinate
MINZO	Minimum Z-coordinate
MIX	X-moment of inertia
MIY	Y-moment of inertia
ZMIZ	Z-moment of inertia
PIXY	X-Y product of inertia
PIYZ	Y-Z product of inertia
PIZX	Z-X product of inertia
PM1	Principal moment of inertia 1

PM2	Principal moment of inertia 2
PM3	Principal moment of inertia 3
PRG1	Principal radius of gyration 1
PRG2	Principal radius of gyration 2
PRG3	Principal radius of gyration 3
RGX	X-radius of gyration
RGY	Y-radius of gyration
RGZ	Z-radius of gyration
SURFA	Surface area
VOL	Volume
WIRE	Wire line length

As the table is created as text, you can query the sheet text to find the value of any item of interest.

The columns of the sheet table are labelled with the object names. In the case of models with more than one object, however, the first column is labelled `ENSEMBLE` and holds the property values for the total model. If the model contains unnamed objects, the corresponding column headings are labelled `T$0001`, `T$0002`, `T$00003`, and so on. In the columns, properties of defined black boxes are listed following all objects found in the model.

By default, the table goes onto layer 51 above the sheet area. A previously created table and any other information that was on layer 51 is moved to layer 52. Anything on layer 52 is deleted. In other words, the first of each three iterations is overwritten and lost unless you override the default settings. You may override the default by using one of the dedicated layer setting commands:

Command	Description
<code>DLAYER layer_number</code>	Sets the layer for the latest table
<code>DLAYER2 layer_number</code>	Sets the secondary layer

User Defined Axes

In addition to having the inertial properties calculated about the X, Y, and Z principal axes, you may specify the origin of a set of axes that is parallel to the X, Y, and Z-axes. When this is done, the moments of inertia, products of inertia, and radii of gyration are calculated with respect to this set of axes, in addition to the regular X,Y, and Z-axes.

For example, to specify the origin of a set of axes at 10, 15, 20, parallel to the X,Y, and Z-axes, enter:

AXES ORIGIN 10 15 20

At any time you can turn off calculation of the inertial properties with respect to this set of axes by entering:

AXES OFF

Only one set of parallel axes is active at a time.

Example of Mass Properties Program Output

The example of Mass Properties program output shown below was calculated from the model generated by the 3D definition sheet shown in [Figure 282](#).

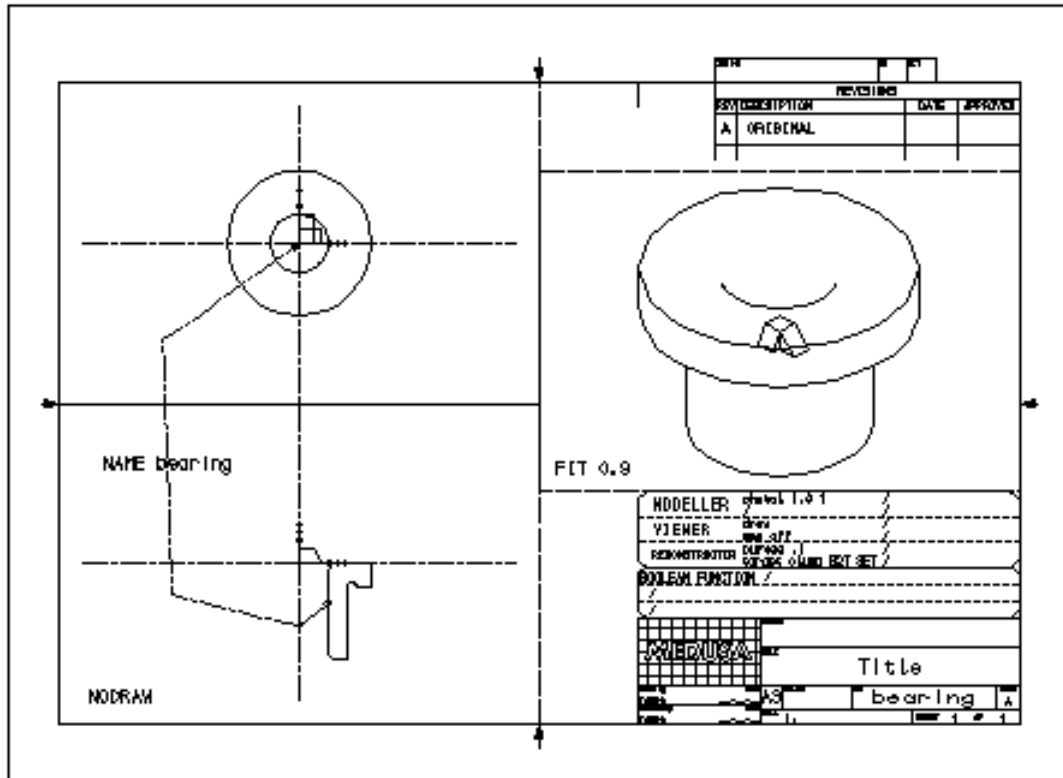
Output to file

```

Object name : BEARING
Volume      : 0.45E+05 cubic mm
Surface Area : 0.13E+0 square mm
Density     : 0.87E-02 g/cubic mm
Mass       : 392.72 g
Center of Gravity      x-coord : -0.85E-05 mm
                        y-coord : 0.85E-08 mm
                        z-coord : -15.10 mm
Moments of Inertia about x-axis : 0.22E+06 g.square mm
                        y-axis  : 0.22E+06 g.square mm
                        z-axis  : 0.15E+06 g.square mm
Products of Inertia in x-y plane : 0.61E-04 g.square mm
                        y-z plane : -0.45E-04 g.square mm
                        z-x plane : 0.38E-01 g.square mm
Radius of Gyration    x-axis  : 23.70 mm
                        y-axis  : 23.70 mm
                        z-axis  : 19.70 mm
Max and min X-coordinates : -30.00, 30.00 (mm)
  Maximum X length       : 60.00 mm
Max and min Y-coordinates : -30.00, 30.00 (mm)
  Maximum Y length       : 60.00 mm
Max and min Z-coordinates : -40.00, 0.92E-05 (mm)
  Maximum Z length       : 40.00 mm
Principal central moment 1 : 0.13E+06 g.square mm
  Direction is arbitrary.
Principal central moment 2 : 0.13E+06 g.square mm
  Direction is arbitrary.
Principal central moment 3 : 0.15E+06 g.square mm
  Direction cosines      (X) : 0.56E-06
                        (Y) : -0.24E-09
                        (Z) : 1.00
Radius of gyration  Prin. axis 1 : 18.27 mm
                    Prin. axis 2 : 18.27 mm
                    Prin. axis 3 : 19.70 mm

```

Figure 282 Definition Sheet for Object BEARING

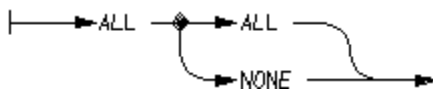


Commands

The following Mass Properties program commands are described in this section:

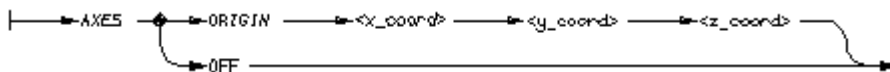
ALL	DLAYER	MODEL
AXES	DLAYER2	OBJECT
BEL	EXTRAPOLATE	OUT
BLACK BOX	GLAYER	PROP
CALC	GLAYER2	QUIT
CODE	GO	REFMOD
DDL	HELP	SHE
DENSITY	IN	SKETCH
DP	LENGTH	UNITS
DET	MASS	

ALL



Calculates all properties for all objects, or none of the properties for all objects. The default is ALL.

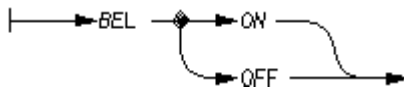
AXES



Option	Description
AXES_ORIGIN	Requests that the inertial properties (moments and products of inertia, and radii of gyration) be calculated about a parallel set of axes that have their origin at the specified X, Y, and Z-coordinates.
AXES_OFF	Turns OFF the calculation of inertial properties about the parallel axes, specified by AXES_ORIGIN.

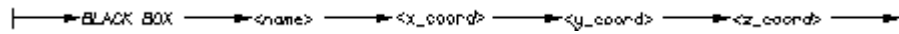
The default is AXES_OFF.

BEL



Switches the audible prompt signal ON or OFF. The default is BEL OFF.

BLACK BOX



Specifies a black box by name, with its center of gravity at the specified X, Y, and Z-coordinates.

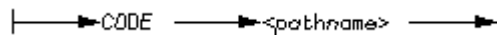
Other properties may be requested and defined for a black box using the OBJECT and PROP commands. Refer to section “Black Boxes” on page 375 for additional details.

CALC



Initiates the calculation of the requested properties.

CODE



Specifies the pathname of the CODE table to be used. If sheet output is desired, then both the CODE table and the DDL must be specified.

There is no default CODE table when the program is run using MEDUTIL.

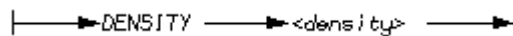
DDL



Specifies the pathname of the DDL file to be used. If sheet output is desired, then both the CODE table and the DDL must be specified.

There is no default DDL when the program is run using MEDUTIL.

DENSITY



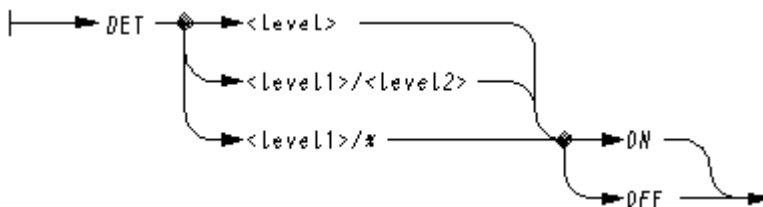
Sets the default density to a number you specify. The currently set units are assumed. The default density is one gram per cubic millimeter.

DP

| → DP → <decimal places> →

Sets the number of decimal places to be displayed in the output. The maximum is 6. The default is 5.

DET



Allows the specified detail levels to be switched ON or OFF. Only the detail levels switched ON are processed.

Specify the arguments *level*, *level1*, and *level2* as integers in the range 0 through 255.

DLAYER

| → DLAYER → <Layer> →

Sets the layer that will hold the MEDUSA4 table containing the properties output. The argument must be an integer from 40 through 89 or greater than 99 to avoid contention with other MEDUSA4 system layers.

If this command is not used, the default layer is 51. Anything found on this layer is moved to layer 52. This secondary layer may be changed by using the DLAYER2 command.

DLAYER2

| → DLAYER2 → <layer> →

Sets the layer that the old properties output will be moved to. Argument must be an integer from 40 through 89 or greater than 99 to avoid contention with other MEDUSA4 system layers. Anything that is on the specified layer is deleted.

If this command is not used the old properties output is moved to layer 52. Anything already on layer 52 is deleted.

GLAYER

| → GLAYER → <Layer> →

Sets the layer that will hold the sketches of the ellipsoid of inertia, principal axes, and center of gravity. Argument must be an integer from 40 through 89 or greater than 99 to avoid contention with MEDUSA4 system layers.

If this command is not used, the sketches of the ellipsoid of inertia, principal axes, and center of gravity are placed on layer 53. Anything already on layer 53 is moved to layer 54. This secondary layer may be changed with the `GLAYER2` command.

GLAYER2

| → `GLAYER2` → `<layer>` →

Sets the layer that holds the old sketches of the ellipsoid of inertia, principal axes, and center of gravity. Argument must be an integer from 40 through 89 or greater than 99 to avoid contention with MEDUSA4 system layers.

If this command is not used, the old sketches of the ellipsoid of inertia, principal axes, and center of gravity are moved to layer 54. Anything already on layer 54 is deleted.

GO

| → `GO` →

Initiates the calculation of the requested properties.

HELP

| → `HELP` →

Displays a list of available commands.

IN

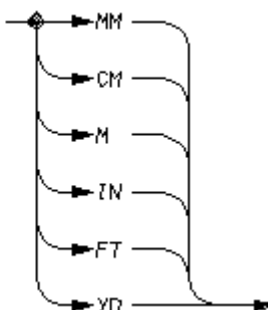
| → `IN` → `<filename>` →

Specifies input model *filename*.

There is no default model name when the program is run using MEDUTIL.

LENGTH

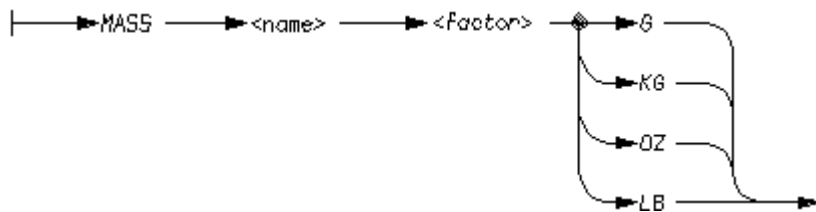
| → `LENGTH` → `<name>` → `<factor>` →



The diagram shows the command syntax: | → LENGTH → <name> → <factor> →. To the right of the arrow pointing to <factor>, there is a vertical list of unit options: MM, CM, M, IN, FT, and YD. Each option is connected to the <factor> arrow by a curved line that starts from the list and ends at the arrowhead.

Specifies units of length as a factor of standard units (see the `UNITS` command, [page 391](#)). The argument name is the nonstandard unit of measure, and factor is the conversion factor between the units.

MASS



Specifies units of mass as a factor of standard units (see the `UNITS` command, [page 391](#)). The argument name is the nonstandard unit of measure, and factor is the conversion factor between the units.

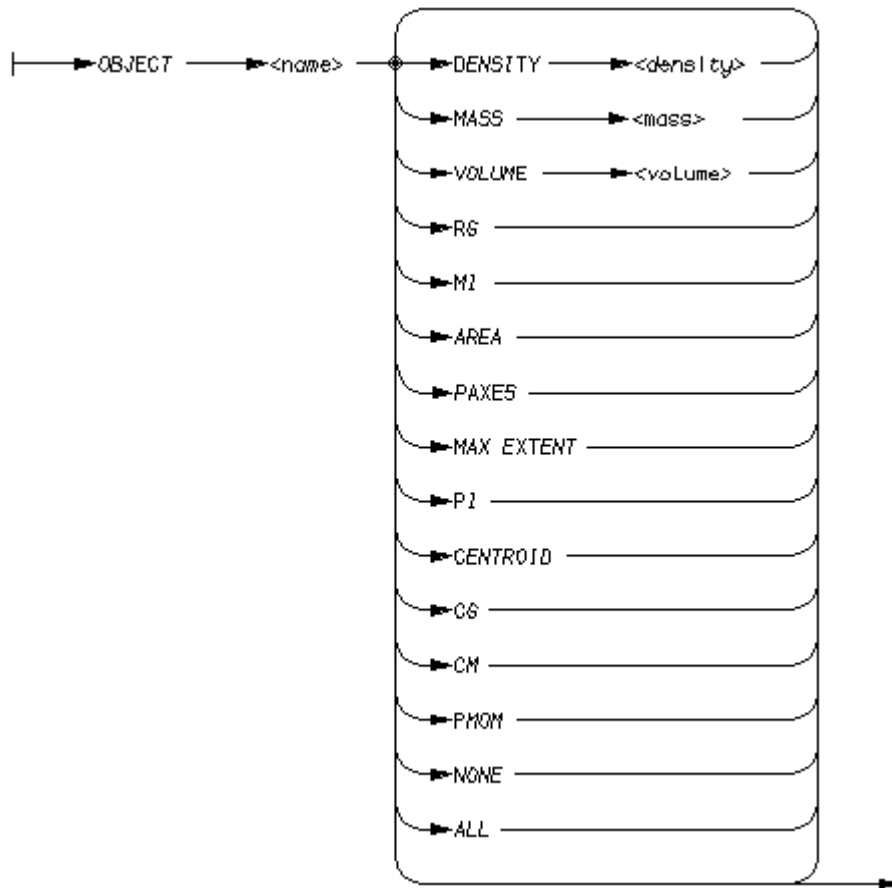
MODEL



Specifies the input model file.

There is no default model name when the program is run using MEDUTIL.

OBJECT



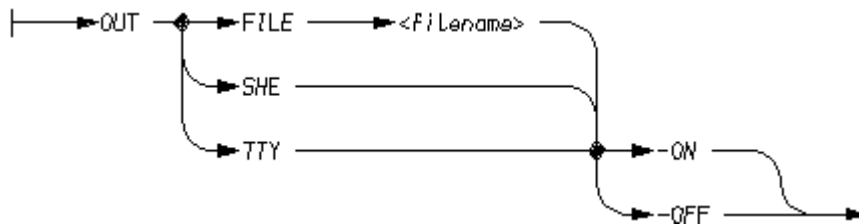
Specifies the name of the object, and the desired properties for that object.

Option	Description
ALL	All properties
AREA	Surface area
CENTROID	Centroid
CG	Centroid
CM	Centroid
DENSITY	Density
MASS	Mass
MAX EXTENT	Maximum extent
MI	Moments of inertia
NONE	No properties
PAXES	Principal axes

PI	Products of inertia
PMOM	Principal moments of inertia
RG	Radii of gyration
VOLUME	Volume

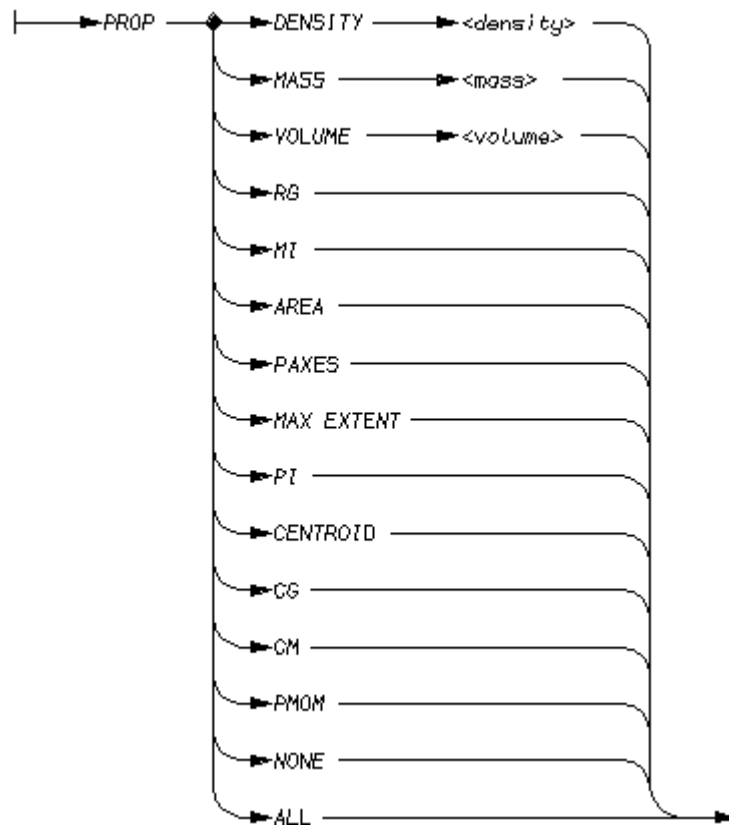
Please note: The argument volume for `VOLUME` may be specified for black boxes only. Refer to [“Black Boxes” on page 375](#) for more details.

OUT



Specifies form(s) of output: file (to *filename*), sheet, or terminal.
The default is `OUT TTY -ON`.

PROP



Specifies which objects are to have a specified property calculated for them (see the `OBJECT` command, [page 387](#)).

Please note: The argument `volume` for `VOLUME` may be specified for black boxes only. Refer to [“Black Boxes” on page 375](#) for more details.

QUIT

→ QUIT →

Quits from the program, closing all units, and returning to the starting point.

REFMOD

→ REFMOD → <filename> →

Specifies a refined model.

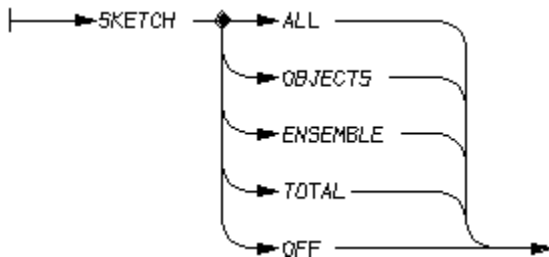
SHE

| → SHE → <filename> →

Specifies the filename of the sheet that will be used for sheet output.

There is no default when the program is run using MEDUTIL.

SKETCH

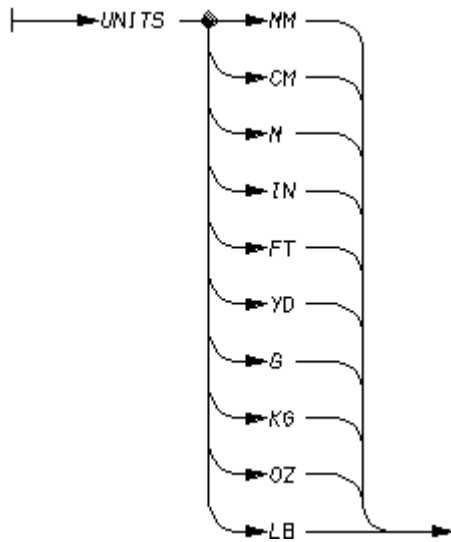


The available options are:

Option	Description
ALL	Displays output for all objects and the total model.
OBJECTS	Displays output for objects only and not for the total model.
ENSEMBLE	Displays output for the total model only.
TOTAL	Displays output for the total model only.
OFF	Does not sketch anything.

The default is SKETCH OFF.

UNITS



Sets one of the standard units of length and mass, which are:

Unit	Definition
MM	Millimeters
CM	Centimeters
M	Meters
IN	Inches
FT	Feet
YD	Yards
G	Grams
KG	Kilograms
OZ	Ounces
LB	Pounds

The default units are millimeters and grams.



THE SHRINKER

The Shrinker takes a model file consisting of one or more objects and shrinks each object by a specified factor about its center of gravity. It is mainly used on models consisting of more than one object to give an exploded view of the components. The output is a standard model file. One of the 3D Viewers can be used to view the completed model.

- [Running the Shrinker](#) 394
- [Using the Shrinker Program in MEDUTIL](#) 396
- [An Example of Shrinker Output](#) 397
- [Shrinker Commands](#) 399

Running the Shrinker

The Shrinker can be accessed using either of the following methods:

- Outside MEDUSA4 using the MEDUSA4 Utility Access facility, MEDUTIL
- From within MEDUSA4 by selecting the appropriate option on a menu

These methods are described below.

Running the Shrinker outside MEDUSA4

The program can be entered outside MEDUSA4 by means of the `SHRINK` command in MEDUTIL, the MEDUSA4 Utility Access facility. A brief introduction to MEDUTIL shows the following example. Note that in the following example, user input is in bold type.

```
medutil project_name

MEDUSA4 Utility Control
~~~~~

Type 'help' for list of commands...
Enter command> shrink

MEDUSA4 Model Shrinking Program
~~~~~

*macro <MEDUSA4 installation path>\med3dshrink\m3d\macro\shrink.mac
Type HELP for a list of Model Shrinking commands

Shrinker>
```

Shrinker commands can be entered at the `Shrinker>` prompt. A complete description of the available commands can be found in [“Shrinker Commands” on page 399](#).

WARNING: Do not try to access MEDUTIL from within MEDUSA4 by typing:

```
!MEDUTIL project_name
```

This command starts too many processes and generates an error message.

Running the Shrinker from within MEDUSA4


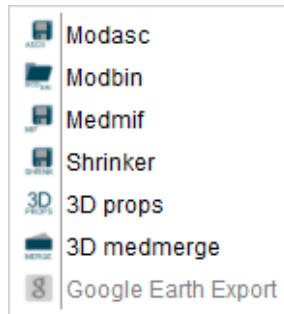
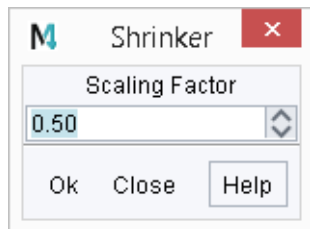
The Shrinker can be entered from within MEDUSA4 by using the Shrinker tool  from the toolset shown in the figure below.

Figure 283 Toolset Containing the Shrinker Tool



1. Choose the Shrinker tool  to open the Shrinker dialog.

Figure 284 The Shrinker Dialog



2. Enter the desired value for Scaling Factor by changing the default value with the arrows or by entering an arbitrary value.
3. Click OK.

The Shrinker starts and creates a new model file of the current sheet. This model file takes the sheet name followed by the suffix *_shrink*. For example, the Shrinker generates the file *example_shrink.mod* from the sheet *example.she*. To view the output of the Shrinker you can use the Model Viewer, see [“The Model Viewer” on page 343](#).

Using the Shrinker Program in MEDUTIL

Once you have started the Shrinker program in MEDUTIL you can enter commands at the `Shrinker>` prompt. Commands can be entered individually from the keyboard or by executing a macro file. Follow the procedure described below to use the Shrinker program:

1. Specify the name of the model file using the command:

```
IN filename
```

2. Specify the name of a model file to receive the output of the Shrinker, using the command:

```
OUT filename
```

3. Enter the shrink factor using the command:

```
FACTOR factor
```

The value *factor* has to be in the range of 0.0 to 1.0.

4. Start the processing of the model using the command:

```
GO
```

Step 1 to Step 4 can be repeated using different shrink factors or model files.

5. To leave the Shrinker, type the command:

```
QUIT
```

The Shrinker produces a scaled model and stores the model in the file named in the `OUT` command. To view the output of the Shrinker, you can use the Model Viewer, see [“The Model Viewer” on page 343](#).

Please note: By default, the Shrinker processes all the objects in a model file. You can also obtain data for objects that have been named in the 3D definition sheet by using the command: `OBJ name`. The command `OBJ ALL` obtains data for all objects. Specify the `OBJ` command before the `GO` command.

An Example of Shrinker Output

The Output of the Shrinker program can be made in two ways, on the 3D definition sheet or in the Model Viewer. [Figure 285](#) and [Figure 286](#) show an example of an exploded view of a model created with the Shrinker program. It is based on the assembly model of a cup, which was described in chapter “An Example of an Instanced Model” on page 251. The new model file *CUP_ASSEMBLY_shrink.mod* was generated from the *CUP_ASSEMBLY.mod* file and stored in the same directory.

Output on the 3D Definition Sheet



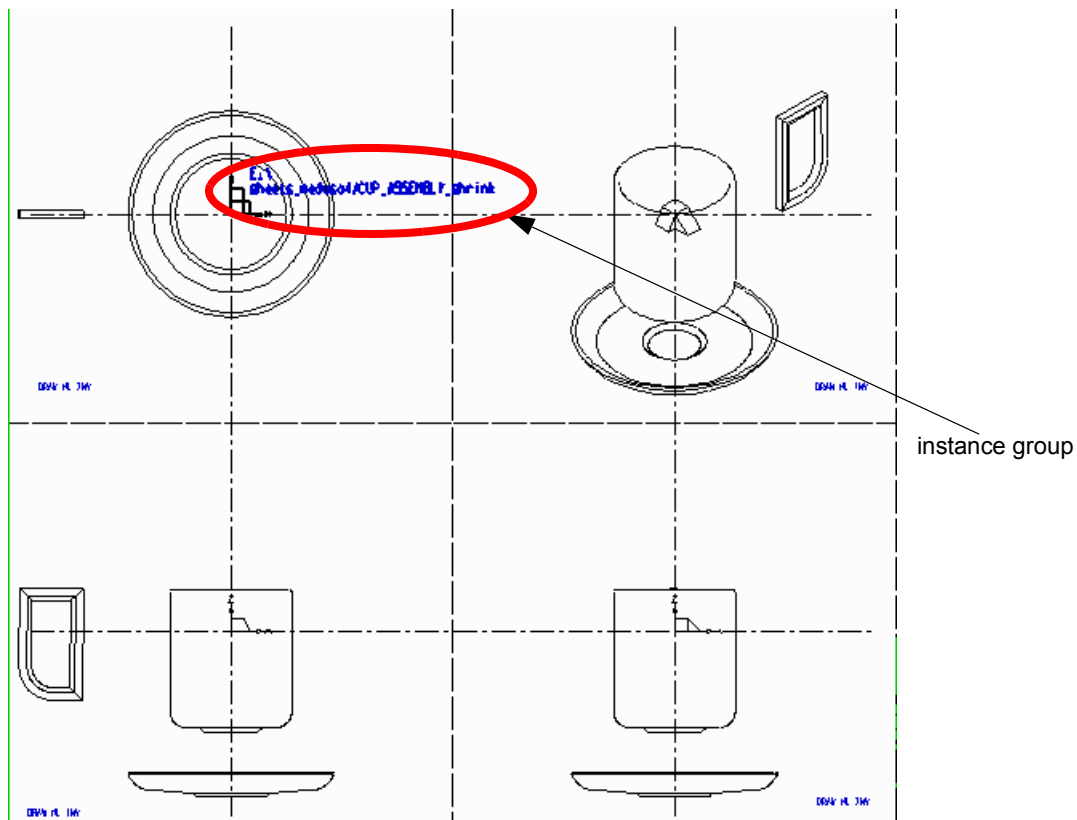
1. Open a new 3D sheet.
2. Choose one of the instance group tools, e.g. XY Instance .
The Instance group dialog will be opened.
3. Choose the model file which was created by the Shrinker (in our example it is *CUP_ASSEMBLY_shrink.mod*).
4. Place the instance group in a viewbox:
5. Choose the Model + Reconstruct tool .
The model of the current drawing is generated and the views are drawn into the viewboxes.

Figure 285 Example of Viewing a Shrunked Model



Output in the Model Viewer


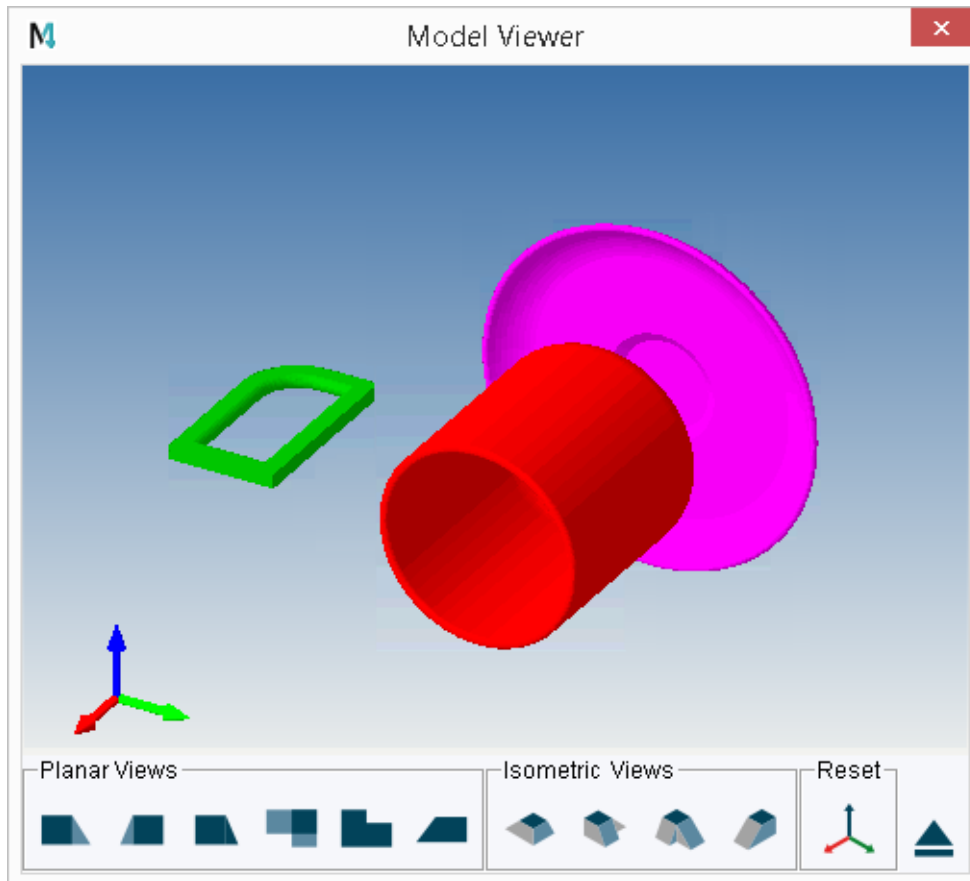
Having executed the steps from “Output on the 3D Definition Sheet” on page 397, choose the Open Model Viewer tool . The Model Viewer opens displaying the shrunk model.

Figure 286 Example of Viewing a Shrunk Model in the Model Viewer



Shrinker Commands

This section lists the commands that are available in the Shrinker program inside MEDUTIL. The syntax of each command is given as a syntax graph. The commands are listed in alphabetical order.

FACTOR

| → FACTOR <factor> →

Specifies the shrink factor, where *factor* is a real number in the range 0.0 through 1.0 inclusive.

GO

| → GO →

Starts processing the model file.

HELP

| → HELP →

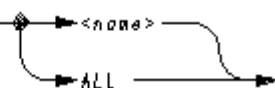
Displays a list of available commands.

IN

| → IN <filename> →

Specifies the name of the input model file, where *filename* is a character string.

OBJ

| → OBJ → 

Specifies that data is to be output for a named object, or for all objects, where object *name* is a character string.

OUT

| → OUT <filename> →

Specifies the name for the output model file, where *filename* is a character string.

QUIT

| → QUIT →

Leaves the Shrinker program.



INTERFACES 1

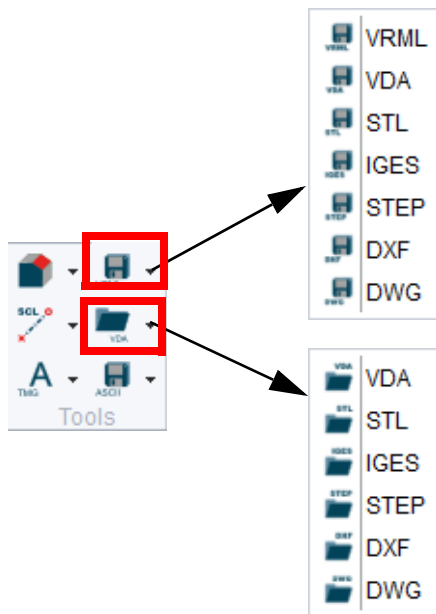
- Tools..... 402
- Running the Converter Tools 403
- Google Earth Export 405
- MEDUTIL 3D Converter Programs 407
- MEDVRML..... 408
- STL 411
- IGES 414
- STEP..... 417
- DXF and DWG 421

Tools

MEDUSA4 provides a number of tools to start conversion programs from inside MEDUSA4. The tools are located inside the 3D ribbon, **Tools** tool group.

This chapter describes the tools which are available on the pulldown menus shown below.

Figure 287 “Tools” Tool Group



Details on the tools are described with the help of the MEDUTIL commands:

- [“MEDVRML” on page 408](#)
- How to use the VDA tools is described in [“Running the Converter Tools” on page 403](#) (in MEDUTIL the programs are `modvda` and `vdamed` with a self-explanatory help)
- [“STL” on page 411](#)
- [“IGES” on page 414](#)
- [“STEP” on page 417](#)
- [“DXF and DWG” on page 421](#)

More converters are described in chapter [“Interfaces 2” on page 425](#).

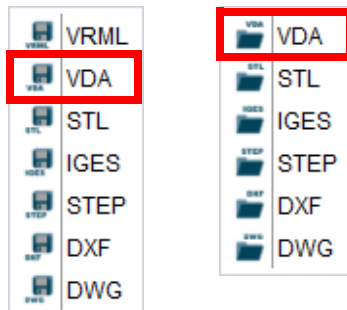
When running the converter programs, default settings are used. The converters can be controlled more detailed within MEDUTIL. So the most frequently used converter programs are explained in the sections which follow [“MEDUTIL 3D Converter Programs” on page 407](#).


Running the Converter Tools


Running the converter tools within MEDUSA4 works always the same way, for converting a model into another format as well as for translating another format into a MEDUSA4 model file.

The following sub-sections explain the procedures with the help of the MEDVDA und VDAMED tools.


Figure 288 "Tools" Tool Group - Example Tools



 Run 3D MEDVDA interface
converts a *.mod* file into a *.vda* file

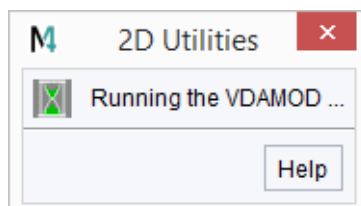
 Run 3D VDAMED interface
converts a *.vda* file into a *.mod* file

Converting MEDUSA4 Model into VDA File (MEDVDA)

If you have loaded a MEDUSA4 3D sheet from which already a *.mod* file exists, choose the Run 3D MEDVDA interface tool  to start the VDA converter.







The following message windows appear.

Figure 289 Message Window VDA




The window closes when the conversion is completed. The *.vda* file is automatically stored under the same name into the same directory as the *.mod* file.

For the other tools the procedures are the same, only the file extensions are different.

Tool	Name	Tooltip	Description	File Extension
	VRML	Run 3D MEDVRML interface	converts MOD into VRML	wrl
	STL	Run 3D MEDSTL interface	converts MOD into STL	stl
	IGES	Run 3D MEDIGES interface	converts MOD into IGES	igs
	STEP	Run 3D MEDSTEP interface	converts MOD into STEP	stp
	DXF	Run 3D MEDDXF interface	converts MOD into DXF	dxf
	DWG	Run 3D MEDDWG interface	converts MOD into DWG	dwg






Converting VDA File into MEDUSA4 Model (VDAMOD)

If you want to convert a *.vda* file into a *.mod* file, choose the Run 3D VDAMED interface tool  to call up a browser window where you can select the desired *.vda* file.

Double click on the file name or click on the Open button starts the converter.

A message window appears. The window closes when the conversion is completed. The *.mod* file is automatically stored under the same name and in the same directory as the *.vda* file.

For the other tools the procedures are appropriate:

Tool	Name	Tooltip	Description
	STL	Run 3D STLMED interface	converts STL into MOD
	IGES	Run 3D IGESMED interface	converts IGES into MOD
	STEP	Run 3D STEPMED interface	converts STEP into MOD
	DXF	Run 3D DXFMED interface	converts DXF into MOD
	DWG	Run 3D DWGMED interface	converts DWG into MOD

Google Earth Export

The Google Earth interface is a file converter, which converts model files (MOD) into Collada XML files (DAE, digital asset exchange). 3D MOD files are produced by MEDUSA4 3D or MPDS4. DAE files can be loaded in Google Earth as 3D models.

Usage


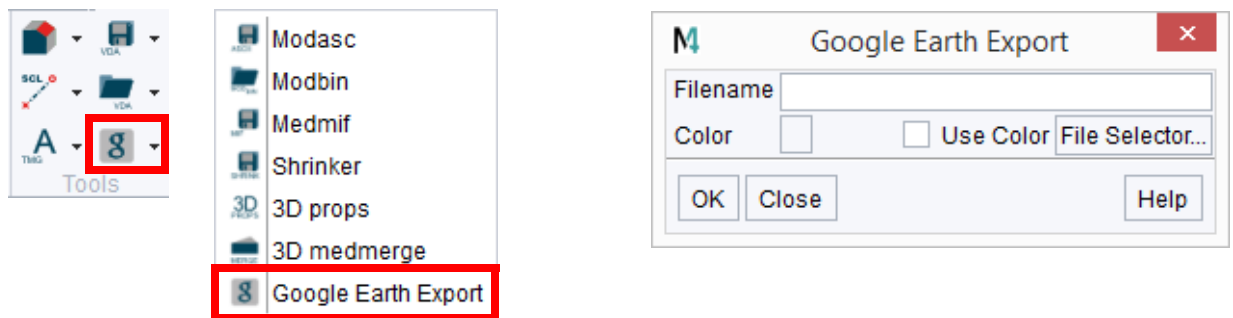
1. Choose the Google Earth Export tool  in the Tools tool group to open the Google Earth Export dialog:

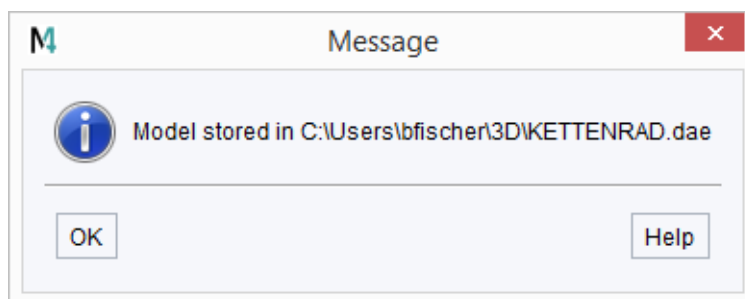
Figure 290 Google Earth Export Tool and Dialog



2. Select a model file using the File Selector button which opens a file selector dialog.
3. If you would like color to be fixed check the Use Colour options and enter a color number in the Colour text field.
4. Open the console so you are able to see the output messages.
5. Press OK.

A DAE file is produced in the input folder which can be read by Google Earth.

Figure 291 Message after running the Google Earth Export



Color is assigned in one of four ways whichever works first:

- color parameter entered
- MPDS4 object color
- surface pen color
- random color

How to Load the Model in Google Earth

1. Open Google Earth.
2. Go to the desired location.
3. Select Add -> Model.
4. Choose the button `Browse` to select the DAE file name of the model.
5. Choose `OK`.
The chosen DAE file is placed in the center of the current location.
If the file cannot be placed, an error comes up.

MEDUTIL 3D Converter Programs

In addition to the graphical user interface the 3D converter programs are also available in MEDUTIL. The following sections give explanations on the usage of some of them within MEDUTIL and the command syntax:

- “MEDVRML” on page 408
- “STL” on page 411
- “IGES” on page 414
- “STEP” on page 417
- “DXF and DWG” on page 421

MEDVRML

MEDUSA4 provides a converter for translating model files (*.mod) into VRML files. You can access the VRML converter from inside MEDUSA4 (see “[Running the Converter Tools](#)” on [page 403](#)) or by using MEDUTIL, see below.

Converting MEDUSA4 Model into VRML File

Start MEDUTIL and enter the `medvrml` command.

An example of using MEDVRML is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 Binary to VRML Model Conversion
~~~~~
Medvrml>in shaft.mod
Medvrml>out shaft.vrml
Medvrml>det 1 on
Medvrml>det 4 on
Medvrml>mode solid
Medvrml>go
Medvrml>quit
```

MEDVRML Commands

Commands may be typed in uppercase or lowercase.

IN <filename>

Opens the model file which is to be translated, where *filename* is the name of the file.

OUT <filename>

Specifies the VRML file to be produced where *filename* is the name of the file.

GO

Starts the translation process.

DET <level> ON | OFF

Allows the specified detail level to be switched ON or OFF. Only the detail levels switched ON are processed. Specify the argument *level* as integer in the range 0 through 255, e.g. DET 4 ON.

HELP

Displays a list of the available commands.

OBJ <object_name> | ALL

Specifies the objects to be processed where *object_name* is the name of the object. The default is ALL. This command is only usable if objects were named when the model was generated.

FMT ON | OFF

Specifies the format generated file on or off (default).

MODE SOLID | WIRE

Specifies to output object solid or wire framed.

QUIT

Exits from the program.

Color Map

MEDVRML uses following color map defined internally:

```
0.9 0.9 0.9 # lt gray
0.8 0.8 0.0 # yellow
0.0 0.4 0.0 # dk green
0.0 0.0 0.4 # dk blue
0.4 0.0 0.0 # dk red
0.0 0.9 0.0 # green
0.0 0.0 0.9 # blue
0.4 0.4 0.4 # mid gray
0.9 0.6 0.1 # orange
0.1 0.9 0.6 # lt green
0.1 0.6 0.9 # sky blue
1.0 0.5 0.5 # pink
0.6 0.9 0.1 # green yellow
0.6 0.1 0.9 # blue purple
```

To change the color map set the environment variable `MED_VRML_COLORS` to be the name of the file that contains the color map. The variable can be defined in the file *login.bat*, for example. The name of the file contains the full path.

The format used in the file is:

```
sp r g b # name
sp  : space
r   : red value in range 0 - 1
g   : green value in range 0 - 1
b   : blue value in range 0 -1
name: color name
```

Each line replaces the appropriate color index, for example, line 1 replaces color index 1, line 2 replaces color index 2 and so on.

Example:

Variable defined in the *login.bat*:

```
set MED_VRML_COLORS=C:\customproduct\vrml.col
```

The file *vrml.col* contains the following lines:

```
0.9 0.6 0.1 # orange
0.1 0.9 0.6 # lt green
0.1 0.6 0.9 # sky blue pink
1.0 0.5 0.5 # pink
```

These entries replace the first four colors.

STL

MEDUSA4 provides a converter for translating model files (*.mod) into STL files and vice versa. You can access the STL converter from inside MEDUSA4 (see [“Running the Converter Tools” on page 403](#)) or by using MEDUTIL, see below.

Converting MEDUSA4 Model into STL File - MEDSTL

Start MEDUTIL and enter the `medstl` command.

An example of using MEDSTL is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MEDSTL
~~~~~
MEDSTL>in shaft.mod
MEDSTL>out shaft_asc.stl
MEDSTL>ASCII
MEDSTL>go
MEDSTL>out shaft_bin.stl
MEDSTL>BINARY
MEDSTL>go
MEDSTL>quit
```

MEDSTL Commands

Commands may be typed in uppercase or lowercase.

IN <filename>

Opens the model file which is to be translated, where *filename* is the name of the file.

OUT <filename>

Specifies the STL file to be produced where *filename* is the name of the file.

GO

Starts the translation process.

HELP

Displays a list of the available commands.

ASCII | BINARY | GOOGLE

Specifies the format of the output file. Possible formats are:

- ASCII ASCII format
- BINARY Binary format
- GOOGLE Google format

QUIT

Exits from the program.

Converting STL File into MEDUSA4 Model - STLMED

Start MEDUTIL and enter the `stlmed` command.

An example of using STL MED is shown below. The input information is shown in boldface.

```
MEDUSA4 STL MED
~~~~~
STLMED>in shaft.stl
STLMED>out shaft_asc.mod
STLMED>ASCII
STLMED>go
STLMED>out shaft_bin.mod
STLMED>BINARY
STLMED>go
STLMED>quit
```

STLMED Commands

Commands may be typed in uppercase or lowercase.

IN *<filename>*

Opens the STL file which is to be translated, where *filename* is the name of the file.

OUT <*filename*>

Specifies the model file to be produced where *filename* is the name of the file.

GO

Starts the translation process.

HELP

Displays a list of the available commands.

ASCII | BINARY

Specifies to format of the output file. possible formats are:

- ASCII ASCII format
- BINARY Binary format

QUIT

Exits from the program.

IGES

MEDUSA4 provides a converter for translating model files (*.mod) into IGES files and vice versa. You can access the IGES converter from inside MEDUSA4 (see “[Running the Converter Tools](#)” on page 403) or by using MEDUTIL, see below.

Converting MEDUSA4 Model into IGES File - MODEXPOR

For converting a model into an IGES file the program MODEXPOR is used, which can create files of several formats. The file format which is created will be determined by the file extension defined for the output file, which is *.igs for an IGES file.

An example of using MODEXPOR is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MODEXPOR
~~~~~
MODEXPOR>in shaft.mod
MODEXPOR>out shaft_asc.igs
MODEXPOR>go
MODEXPOR>quit
```

MODEXPOR Commands

Commands may be typed in uppercase or lowercase.

IN <filename>

Opens the model file which is to be translated, where *filename* is the name of the file.

OUT <filename>

Specifies the IGES file to be produced where *filename* is the name of the file.

GO

Starts the translation process.

HELP

Displays a list of the available commands.

QUIT

Exits from the program.

SEWTOL

Tolerance used for sewing points (default 0.1).

LINTOL

Linear tolerance used in constructing shape (default 0.1).

ANGTOL

Angular tolerance used in constructing shape (default 20.0).

OBJECT LIMIT *<i>*

Number of objects to process before writing a file (default 500).

POINT LIMIT *<i>*

Number of points to process before writing a file (default 1000000).

DETAIL LEVEL *<level>*

Sets a detail to be produced. By default and if this command is not used, all detail levels are exported.

<level>

is the detail level.

MODE ATTR

Export object name and color attributes.

MODE NOATTR

Do not export object name and color attributes (default).

COLMAP <file>

Define the used color map file. If no file will be defined, the default file *med3d\m2d\src\3d_colors.map* will be used.

QCOLS

Query the colors. This displays the Red Green Blue values of the colors found in the loaded file and the MEDUSA4 color index that it will be mapped to. The MAPCOL command may be used to change this mapping. Example:

Index	Red	Green	Blue
1	173	173	173
2	219	94	56

MAPCOL <index> <red> <green> <blue>

Map color index <index> to RGB values (range 0-255) <red> <green> <blue>.

FINALTRANS <shiftx>, <shifty>, <shiftz>, <rotz>, <rotx>, <roty>

Applies a final transformation on the output file. The values for the rotation are specified in degrees.

MAXDEF <number>

Defines the distance that the fitted surface is allowed to be from the facets before the facets are used for output instead of the surface. If the *number* value is not zero a check is made how far the fitted surface is from the facets.

Converting IGES File into MEDUSA4 Model - MODIMPORT

For converting an IGES file into a model the program MODIMPORT is used, which can convert files of several formats. The file format which is converted will be determined by the file extension defined for the input file, which is **.igs* for an IGES file.

An example of using MODIMPORT is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MODIMPORT
~~~~~
MODIMPORT>in shaft.igs
MODIMPORT>out shaft.mod
MODIMPORT>go
MODIMPORT>quit
```


The MODIMPORT commands are described in "STEP", "MODIMPORT Commands" on page 418.

STEP

MEDUSA4 provides a converter for translating STEP files (*.stp) into a model file (*.mod) and vice versa. You can access the STEP converter from inside MEDUSA4 (see "Running the Converter Tools" on page 403) or by using MEDUTIL, see below.

Converting MEDUSA4 Model into STEP File - MODEXPOR

For converting a model into a STEP file the program MODEXPOR is used, which can create files of several formats. The file format which is created will be determined by the file extension defined for the output file, which is *.stp for a STEP file.

An example of using MODEXPOR is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MODEXPOR
~~~~~
MODEXPOR>in shaft.mod
MODEXPOR>out shaft.stp
MODEXPOR>go
MODEXPOR>quit
```

For the available MODEXPOR commands see "IGES", "MODEXPOR Commands" on page 414.

Converting STEP File into MEDUSA4 Model - MODIMPORT

For converting a STEP file into a model the program MODIMPORT is used, which can convert files of several formats. The file format which is converted will be determined by the file extension defined for the input file, which is *.stp for a STEP file.

Start MEDUTIL and enter the `modimport` command.

An example of using MODIMPORT is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MODIMPORT
~~~~~
MODIMPORT>in shaft.stp
MODIMPORT>out shaft.mod
MODIMPORT>go
MODIMPORT>quit
```

MODIMPORT Commands

Commands may be typed in uppercase or lowercase.

IN *<filename>*

Opens the STEP file which is to be translated, where *filename* is the name of the file.

`modimport` can also convert 3D IGES (*.igs*) and 3D DXF files (*.dxf*) into model files. For IGES files an appropriate tool exists in the graphical user interface, see “IGES” on page 414. There is no tool in the graphical user interface for DXF files.

OUT *<filename>*

Specifies the model file to be produced where *filename* is the name of the file.

GO

Starts the translation process.

QUIT

Exits from the program.

HELP

Displays a list of the available commands.

DETAIL *<level>* *<mode>* *<size>*

Sets a detail to be produced.

<level>

is the detail level.

<mode>

0 : all detail

1 : box objects below *size*

- 2 : box all object
- 3 : Box model

<size>

size to used for mode 1. It is a real value and it is in the units of the file you are importing. If the length of the diagonal of a box that encloses the object being imported is less than this value it is replaced by a box. A file being imported may have multiple objects so only some will be made boxes.

RESETDETAIL

This clears all the current detail information back to producing level 0 and full detail.

NOATTLOAD

This command prevents handling attributes of the STEP input file. Additionally the commands that follow will not be performed.

The performance of STEP to model file conversion will be increased considerably with NOATTLOAD.

LOAD

Load the file opened with `IN` so that colors, names and layers information can be queried and mapping and blanking setup before the output is generated.

QCOLS

Query the colors. This displays the Red Green Blue values of the colors found in the loaded file and the MEDUSA4 color index that it will be mapped to. The `MAPCOL` command may be used to change this mapping. Example:

Red	Green	Blue	Index
29	118	163	11
172	59	40	2

Q NAMES

Query the names. This displays the names found in the loaded file. The `BNAME` command may be used to blank a name and stop it appearing in the output. Example:

```
Blank Names
0      KLOTZ_ ; _KLOTZ
0      KLOTZ_KLEIN_ ; _KLOTZ_KLEIN
```

QLAYERS

Query the layers. This displays the layers found in the loaded file. The `BLAYER` command may be used to blank a layer and stop it appearing in the output. Example:

```
Blank Layers
0      Layer1
0      Layer2
```

MAPCOL *<red>* *<green>* *<blue>* *<index>*

Maps one of the colors displayed by QCOLS to a new index (integer value).

```
<red> <green> <blue>
      integer value combination found by QCOLS
```

BNAME *<blank>* *<name>*

The BNAME command may be used to blank a name and stop it appearing in the output.

```
<blank>
      0 include in output
      1 exclude from output
<name>
      one of the names displayed by QNAMES. <name> can end in a single wildcard (*).
```

BLAYER *<blank>* *<layer>*

The BLAYER command may be used to blank a layer and stop it appearing in the output.

```
<blank>
      0 include in output
      1 exclude from output
<layer>
      one of the names displayed by QLAYERS.
```

FINE ON | OFF

Import each object faceted at the suggested parameters for that object. This produce larger models but it is useful if you want more exact results.

DXFSEW ON | OFF

Controls whether facet edges are sewed together (ON) or not (OFF, default).

ALLSHAPES ON | OFF

Controls the import of *xxx.stp.zip* files. If ON, zip files of the STEP models are allowed to be imported.

SEPMODMODE ON | OFF

Controls the generation of multiple .MOD files for the import of STEP, IGES or DXF/DWG files. If ON, a separate model of each object is generated.

MODINFOMODE ON | OFF

Controls the generation of a model info file with extension *.mod_info*. If ON, an info file is generated.

Reconstructing 2D Sheet

The database keys *ONAME* and *OLAY* are added to the model if the object has name or layer information. The first layer name will be in *OLAY1*, the second in *OLAY2* etc. The database keys can be used in the viewer to add this information to the sheet elements.

Database keys are described in ["Reconstruction Commands"](#), ["Specifying Database Keys"](#) on [page 357](#).

DXF and DWG

MEDUSA4 provides converters for translating DXF files (*.*dxf*) and DWG files (*.*dwg*) into MEDUSA4 model files (*.*mod*) and vice versa.

DXF

You can access the DXF converter from inside MEDUSA4 (see ["Tools"](#) on [page 402](#)) or by using MEDUTIL, see below.

Converting MEDUSA4 Model into DXF File - MODEXPOR

For converting a model into a DXF file the program MODEXPOR is used, which can create files of several formats. The file format which is created will be determined by the file extension defined for the output file, which is **.dxf* for a DXF file.

An example of using MODEXPOR is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MODEXPOR
T
~~~~~
MODEXPOR
T>in shaft.mod
MODEXPOR
T>out shaft.dxf
MODEXPOR
T>go
```

For the available MODEXPOR commands see ["IGES"](#), ["MODEXPOR Commands"](#) on page 414.

Converting DXF File into MEDUSA4 Model - MODIMPORT

For converting a DXF file into a model the program MODIMPORT is used, which can convert files of several formats. The file format which is converted will be determined by the file extension defined for the input file, which is *.*dxf* for a DXF file.

An example of using MODIMPORT is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MODIMPOR
T
~~~~~
MODIMPOR
T>in shaft.dxf
MODIMPOR
T>out shaft.mod
MODIMPOR
T>go
```

The MODIMPORT commands are described in ["STEP"](#), ["MODIMPORT Commands"](#) on page 418.

DWG

DWG file conversion works in the same way as for DXF, but to get a DWG output or use a DWG input file, environment variables have to be defined before entering MEDUTIL.

You can access the DWG converter from inside MEDUSA4 (see ["Tools"](#) on page 402) or by using MEDUTIL, see below.

The environment variable `MODIMPORT_DXF_OPTS` gives the option to use for the import conversion and the environment variable `MODEXPORT_DWG_OPTS` the options to use for the export conversion. The supported options are:

Option	Description
<code>-t <outtype></code>	sets the type of target file. <i>outtype</i> can be any of: DWG, DXF, DXB (default DWG)
<code>-v <outver></code>	sets the version of target file. <i>outver</i> can be any of: ACAD9, ACAD10, ACAD12, ACAD13, ACAD14, ACAD2000, ACAD2004, ACAD2007, ACAD2010 (default ACAD2000)
<code>-p <proxyopt></code>	explodes proxy block. <i>proxyopt</i> can be any of: BLOCK (entities are exploded into new block(s)) ENTITY (no new block(s)) OFF (default OFF)
<code>-x <xrefopt></code>	resolves external references. <i>xrefopt</i> can be any of: ON, OFF (default OFF)

When exporting a MEDUSA4 model to DWG, first `MODEXPORT` is run exporting a DXF file and then this DXF file is converted into DWG format. When importing a DWG file, it is converted into DXF and then translated into a MEDUSA4 model file with `MODIMPORT`.

So if you have a DWG to import you might define following:

```
set MODIMPORT_DXF_OPTS=-t DXF -v ACAD2000
```

If you want DWG output from export you might use:

```
set MODEXPORT_DWG_OPTS=-t DWG -v ACAD2010
```

Converting MEDUSA4 Model into DWG File - MODEXPORT

An example of using `MODEXPORT` is shown below. The input information is shown in boldface for clarity.

Please note: Consider that the output file has to be specified as *dx*. If you specify the file extension *dwg* as output nothing happens.

```
MEDUSA4 MODEXPORT
~~~~~
MODEXPORT>in shaft.mod
MODEXPORT>out shaft.dxf
MODEXPORT>go
```

The command `go` produces two results, *shaft.dxf* and *shaft.dwg*.

Converting DWG File into MEDUSA4 Model - MODIMPORT

An example of using MODIMPORT is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MODIMPORT
~~~~~
MODIMPORT>in shaft.dwg
  CsgDwgDxfConverter V3.0 Copyright...
  Source File:c:\directory\shaft.dwg
  Target File:c:\temp\username_1_modimport.dxf
  Converted c:\directory\shaft.dwg to c:\temp\username_1_modimport.dxf
  successfully
MODIMPORT>out shaft.mod
MODIMPORT>go
  Reading file
  Calculating....
MODIMPORT>
```

INTERFACES 2

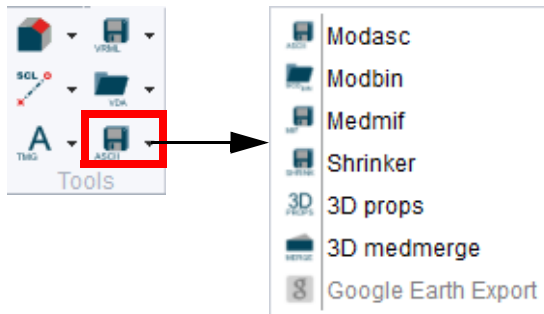
- Tools..... 426
- Running the Converter Tools 427
- MEDUTIL 3D Converter Programs 428
- MODASC, MODBIN, MODSMO - Overview 429
- MODASC 429
- MODBIN..... 431
- MODSMO 433
- MEDMERGE..... 435

Tools

MEDUSA4 provides a number of tools to start conversion programs from inside MEDUSA4. The tools are located inside the 3D ribbon, Tools tool group.

This chapter describes the tools which are available on the pulldown menu shown below.

Figure 292 Tools for Running 3D Converter Programs



The tools are described in the following sections:



- [“MODASC” on page 429](#)
- [“MODBIN” on page 431](#)
- The MEDMIF interface is explained in chapter [“The Model File Translator” on page 437](#)
- The Shrinker is explained in chapter [“The Shrinker” on page 393](#)
- 3D Props is explained in chapter [“Extracting Mass Properties” on page 367](#)
- [“MEDMERGE” on page 435](#)

In [“MODASC, MODBIN, MODSMO - Overview” on page 429](#) additional information is given on these converter programs. The MODSMO converter is not available as a tool in the graphical user interface but it is explained in [“MODSMO” on page 433](#) for completeness.


Please note: When running the converter programs in the graphical user interface, default settings are used. The converters can be controlled more detailed within MEDUTIL.

Running the Converter Tools


Running the converter tools within MEDUSA4 works always the same way, for converting a model into another format as well as for translating another format into a MEDUSA4 model file.

The following sub-sections explain the procedures with the help of the tools Modasc  and Modbin .


Converting MEDUSA4 Model into ASCII File (MODASC)

If you have loaded a MEDUSA4 3D sheet from which already a model file exists, choose the Modasc  tool to start the converter.

Having started the converter, a message dialog is opened. If the conversion is finished, the dialog is closed. The resulting file is automatically stored under the same name with file extension *.asc* into the same directory as the model file.

For the 3d medmerge  tool the procedure is the same. The file extension is *_merge.mod*.

Converting VDA File into MEDUSA4 Model (MODBIN)

If you want to convert an ASCII file into a model file, choose the Modbin  tool to call up a browser window where you can select the desired ASCII file. Double clicking on the file name or clicking on the *Open* button starts the converter.

Having started the converter, a message dialog is opened. If the conversion is finished, the dialog is closed. The resulting model file is automatically stored under the same name and in the same directory as the ASCII file.

MEDUTIL 3D Converter Programs

In addition to the graphical user interface the 3D converter programs are also available in MEDUTIL. The following sections give explanations on the usage of some of them within MEDUTIL and the command syntax:

- [“MODASC, MODBIN, MODSMO - Overview” on page 429](#)
- [“MODASC” on page 429](#)
- [“MODBIN” on page 431](#)
- [“MODSMO” on page 433](#)
- [“MEDMERGE” on page 435](#)

MODASC, MODBIN, MODSMO - Overview

There are three related utility programs, MODASC, MODBIN and MODSMO:

- MODASC takes a MEDUSA4 model file and creates an ASCII file.
- MODBIN produces a MEDUSA4 model file from a suitable ASCII file. These facilities provide access to MEDUSA4 3D data and allow existing 3D data to be used to create a MEDUSA4 model.
- MODSMO converts MEDUSA4 model files into smooth model files. This utility performs two functions:
 - Averages the planar polygon normals.
 - Bridges holes in a polygon to make a single profile polygon.

To use these programs to the full extent, you should be familiar with MEDUSA4 3D. If you are using MODASC and MODBIN simply to transfer ASCII files between machines, only a basic understanding of 3D is required.

MODASC

You can run MODASC from inside MEDUSA4 (see [“Running the Converter Tools” on page 427](#)) or by using MEDUTIL.

Running MODASC from within MEDUTIL

Start MEDUTIL and enter the `modasc` command.

An example of using MODASC is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 Binary to Ascii Model Conversion
~~~~~
Modasc>in shaft.mod
Modasc>out shaft1.asc
Modasc>dp 9
Modasc>go
Modasc>quit
```

MODASC Commands

Commands may be typed in uppercase or lowercase.

DET <level> | <level1>/<level2> | <level1>/* ON | OFF

Allows the specified detail levels to be switched ON or OFF. Only the detail levels switched ON are processed. Specify the arguments *level*, *level1*, and *level2* as integers in the range 0 through 255, e.g. DET 4/16 ON.

DP <decimal_points>

Allows you to set the number of decimal places (1 through 16) required for the output of data, where *decimal_points* is the number of decimal places required. The data is output in exponent format. The default is 16 decimal places.

GO

Starts the translation process.

HELP

Displays a list of the available commands.

IN <filename>

Opens the model file which is to be translated, where *filename* is the name of the model file to be translated.

OBJ <object_name> | ALL

Specifies the objects to be processed where *object_name* is the name of the object. The default is ALL. This command is only usable if objects were named when the model was generated.

OUT <filename>

Specifies the ASCII file to be produced where *filename* is the name of the ASCII file. If the OUT command is not given, then the ASCII output is sent to the terminal.

QUIT

Exits from the program.

MODBIN

You can run MODBIN from inside MEDUSA4 (see “Running the Converter Tools” on page 427) or by using MEDUTIL.

Running MODBIN from within MEDUTIL

Start MEDUTIL and enter the `modbin` command.

An example of using MODBIN is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 Ascii to Binary Model Conversion
~~~~~
Modbin><b>in shaft.asc</b>
Modbin><b>out shaft.bin</b>
Modbin><b>go</b>
Modbin><b>quit</b>
```

If no output file is given, the model file gets the name defined with the `IN` command using the extension `.mod`. For example, if the command `IN SHEET1` is given and no `OUT` command is defined, the generated model file gets the name `SHEET1.mod`.

MODBIN Commands

Commands may be typed in uppercase or lowercase.

ASC <filename>

Specifies the ASCII file to be used as input to MODBIN, where *filename* is the name of the ASCII file to be used.

BIN <filename>

Opens a file to which the binary output is written, where *filename* is the name of the model file to which the binary output is written. If this command is not given before a `GO` command then the model file will be given the name specified in the MODEL record.

GO

Checks that the input file has been specified and then starts the translation from ASCII to binary format.

HELP

Displays a list of the available MODBIN commands.

IN <filename>

Specifies the ASCII file to be used as input to MODBIN, where *filename* is the name of the ASCII file to be used.

OUT <filename>

Opens a file to which the binary output is written, where *filename* is the name of the file to which the binary output is written. If this command is not given before a GO command then the model file will be given the name specified in the MODEL record.

QUIT

Exits from the program.

TRANS

Checks that the input file has been specified and then starts the translation from ASCII to binary format.

MODSMO

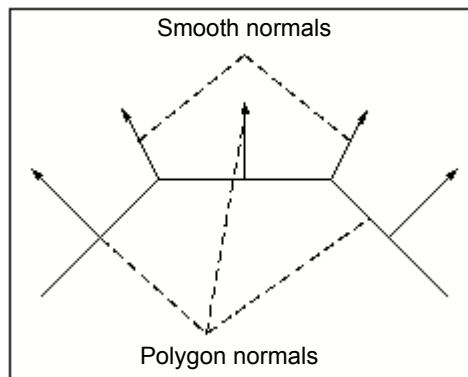
MODSMO converts MEDUSA4 model files into smooth model files. A smooth model file has additional element types that give a curved surface appearance to the model, also holes are bridged to create single profiles.

Please note: You can run MODSMO only from within MEDUTIL.

Averaging the Polygon Normals

When a MEDUSA4 model is created, curved surfaces are approximated using planar polygons. MODSMO allows the model to appear with a smooth curved surface by averaging the planar polygon normals across the surface of an object. [Figure 293](#) shows averaged normals.

Figure 293 Polygon Normals Averaged by the MODSMO Utility



Running MODSMO

Start MEDUTIL and enter the `modsmo` command.

An example of using MODSMO is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 Smooth Model Converter
~~~~~
Modsmo>in shaft.mod
Modsmo>bridge off
Modsmo>compression on
Modsmo>out shaft1.mod
Modsmo>go
```

```
Start of data generation
Object 1 B
Object 2 A
End of data generation
Modsmo>quit
```

MODSMO Commands

BRIDGE ON | OFF

Switches the hole bridging **ON** or **OFF** for the model file you specified. The default setting is **BRIDGE ON**.

COMPRESSION ON | OFF

Switches the compression operation **ON** or **OFF** for the model file you specified. For further information on model file compression, see [“Compressing a Model File” on page 292](#). The default setting is **COMPRESSION OFF**.

GO

Starts the process.

HELP

Shows all commands and a short description.

IN <filename>

Specifies the name of the input model file. Specify *filename* as a text string.

OUT <filename>

Specifies the name of the output smooth model file. Specify filename as a text string.

QUIT

Leaves MODSMO and returns to MEDUTIL.

MEDMERGE

The MEDMERGE utility program merges points belonging to a particular structure and having the same coordinates. You can run MEDMERGE from inside MEDUSA4 (see “[Running the Converter Tools](#)” on page 427) or by using MEDUTIL.

Running MEDMERGE

Start MEDUTIL and enter the `medmerge` command.

An example of using MEDMERGE is shown below. The input information is shown in boldface for clarity.

```
MEDUSA4 MEDMERGE
~~~~~
MEDMERGE>in shaft.mod
MEDMERGE>out shaft_merge.mod
MEDMERGE>go
MEDMERGE>quit
```

MEDMERGE Commands

GO

Starts the process.

HELP

Shows all commands and a short description.

IN <filename>

Specifies the name of the input model file. Specify *filename* as a text string.

OUT <filename>

Specifies the name of the output model file. Specify filename as a text string.

QUIT

Leaves MEDMERGE and returns to MEDUTIL.



THE MODEL FILE TRANSLATOR

The MEDUSA4 to Model Interface Format program (MEDMIF) translates the data in a binary model file into a new format and writes the translated data to an ASCII character file. The ASCII file can be read by programs using either standard formatted FORTRAN READ statements or a Basis2-type interface. Since the Model Interface Format (MIF) file is an ASCII file, it can be sent to a printer or modified using a standard editor.

- The Structure of a Model File 438
- The Structure of a MIF File 438
- Description of MEDMIF 439
- Running MEDMIF 441
- Specification of the MIF File Format 446
- MEDMIF Commands 452

The Structure of a Model File

In a model file, curved surfaces are represented by Rational Patches (ratches) or Coons Patches. For example, a sphere is represented by 8 ratches. Each ratch is divided into planar polygons called tiles. The boundaries of each ratch can be seen if the `BOU VIS` command is given in the Viewer, and the tiles can be seen if the `TIL VIS` command is given.

All the tiles which make up a given ratch have the same surface identifier (SID). Tile edges that meet in the model have the same geometric identifier (GID). If a series of tile edges makes up a continuous curve, then all the edges forming the curve have the same GID.

A model file consists of various header records, followed by a list of polygon (tile) records and point records. The implication of this format is that if points or edges are shared between polygons, then they are defined in each polygon. This further implies that the geometry must be analyzed extensively to establish topological relationships.

The Structure of a MIF File

The MIF file is designed to eliminate repetitions of data, such as coordinates of shared points, and to define clearly the topology of the model.

In the MIF file, each object is represented as a series of tables, each containing references by identifier to other geometric items as follows:

Table Type	Description
Point tables	Contain the coordinates of all the points in the object.
Edge tables	Contain the set of identifiers of the points referred to in each edge.
Surface tables	Contain the set of identifiers of the edges bounding each surface.

Other information is also stored, including the surface equations of curved surfaces and text relating to points, edges, surfaces, or the volume itself. The table format provides topological information without the need for searching polygon/point data. A full description of the structure of the MIF file is given later in this chapter, starting [“The Structure of a MIF File” on page 438](#).

Description of MEDMIF

MEDMIF performs the translation of data from model file format to interface format in two stages:

1. Surface edge polygons are formed around tiled surfaces, producing a temporary file.
2. The temporary file is used to build the point, edge, and surface tables, and also tables for the 3D texts and UPR texts.

Both stages involve a complete pass through the data, each object of the model file being processed separately. No attempt is made to relate the geometry of one object to that of any other. Any point, edge or surface shared by two or more objects will appear in the appropriate table for all the relevant objects, but will have a different identifier in each object.

In the model file, planar surfaces are represented by a flat polygon and curved surfaces by a set of tile polygons, possibly supplemented by a rational patch equation, stored as a set of control points, or as coons patches, stored as coordinates of vertices of boundary edges.

The treatment of curved surfaces in the MIF file is slightly different. These are represented by an edge polygon that traces the outer boundary of all the original tile polygons. The surface equations from the model file can be incorporated in the MIF file if the `EQUATIONS ON` command is placed in the model definition sheet.

Stage 1

The first stage of the translation to a MIF file involves scanning each object in turn, and generating a boundary edge polygon for each curved surface of the object. This is done by gathering into a sorted list each tile polygon of the surface, and stitching together the boundary or visible edges of the tiles into a single polygon. The result of this process is a new temporary model file containing, for each object, the original set of polygons, including tiles, plus any generated surface boundary edge polygons.

Stage 2

In the second stage of the translation, the temporary model file is scanned, with each object being processed in turn. All the entities of the object (polygons, 3D texts, etc.) apart from the tile polygons, are read into memory. For each object, each polygon is examined in turn. The neighboring polygon along each edge of each polygon is found by searching all the polygons of the object and comparing the end points of each edge of each polygon until a match is found.

Then the table of points is built up for the object by scanning all the polygons and adding to the table any points not already in the table. A given point is in the table if its coordinates match the coordinates of another point exactly. No tolerance is used in building the point table.

Next, the edge and surface tables are built by processing each surface of the object in turn. This involves changing a list of model file points, defined by coordinates and annotated by the identi-

fier of the edge leading up to it, into a set of MIF edges defined by references to two or more MIF points.

The last tables to be built are those containing the 3D texts to be associated with the points, edges, or surface of the object and the UPR (uncommitted properties) texts associated with the object itself. In the model file, the 3D texts associated with the geometric entities of the object are specified by a type (such as point, edge, or surface text), a set of coordinates, and a link identifier of the polygon containing the item. The `WINTOL` command assists in calculating which text is associated with which geometric item.

When all the tables for an object are complete, they are written in ASCII character format to the MIF file. Then the heaps and tables are cleared and the next object read in.

When all the objects of the model file have been read, an end-of-model record is written to the MIF file to indicate end-of-file.

Running MEDMIF

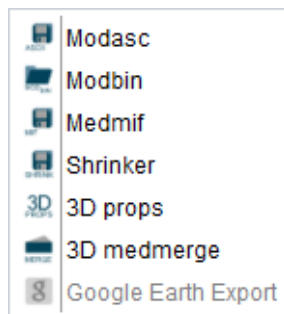
This section shows you how to run MEDMIF, and provides an example of a MEDMIF file. It also explains the format of a MEDMIF file. MEDMIF can be accessed using either of the following methods:

- "Running MEDMIF inside MEDUSA4".
- "Running MEDMIF outside MEDUSA4" on page 442

Running MEDMIF inside MEDUSA4

1. Load a 3D file from which a *.mod* file exists.
2. Choose the *Medmif* tool from the relevant toolset to start the Model Interface Format program (MEDMIF).

Figure 294 Toolset Containing the Medmif Tool



The 2D Utilities dialog appears displaying that the program is running and that the required ASCII file is created. If the process is completed, the 2D Utilities dialog is closed.

The resulting MIF file takes the current sheet name followed by the suffix *.mif* and is automatically stored in the directory of the input file.

3. You now can display the result in any text editor.

Running MEDMIF outside MEDUSA4

The program can be entered outside MEDUSA4 using the MEDMIF command in MEDUTIL, the MEDUSA4 Utility Access facility.

Please note: In the following example user input is shown in boldface type.

1. Start MEDUTIL by typing the `medutil` command in a console.

```
MEDUSA4 Utility Control
~~~~~
Type 'help' list of commands...
```

2. Enter `medmif` at the command prompt.

```
Enter command>: medmif

MEDUSA4 MIF File Generator
~~~~~
Type HELP for a list of MEDMIF commands

Medmif>
```

MEDMIF commands can be entered at the `Medmif>` prompt.

3. Specify an input model file using the command:

```
IN <file>
```

4. Specify the output (MEDMIF) file using the command:

```
OUT <file>
```

5. Start processing using the command:

```
GO
```

6. To exit from the MEDMIF program, use the `QUIT` command.

Full details of each of the commands that can be used within MEDMIF are described later in this chapter, starting in section [“MEDMIF Commands” on page 452](#).

Please note: Do not try to access MEDUTIL from within MEDUSA4 by entering:

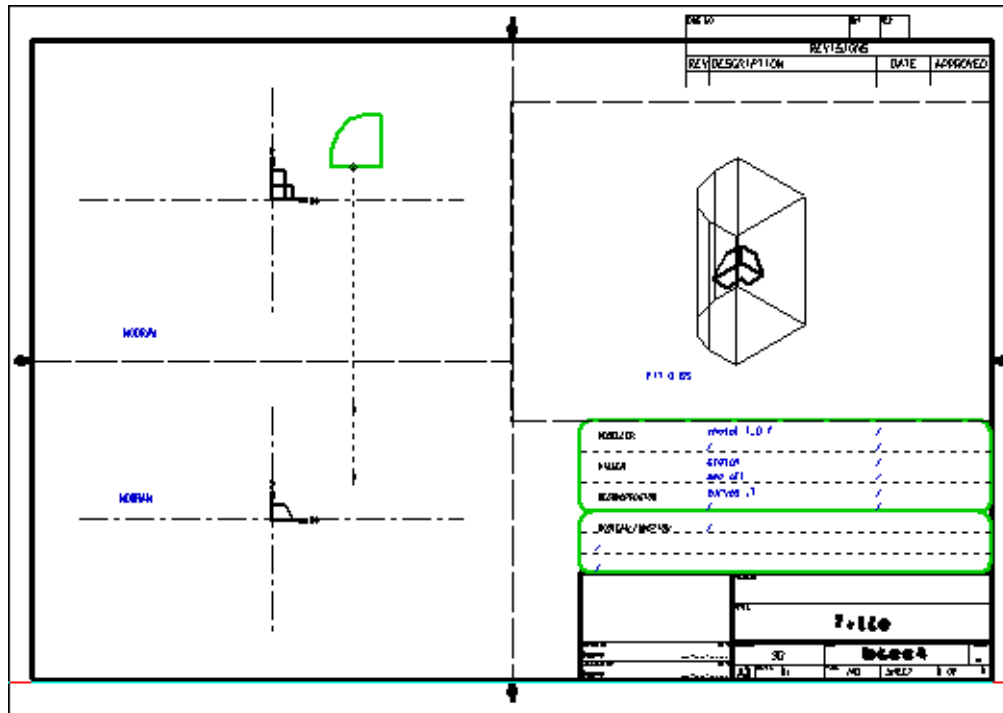
```
MEDUTIL project_name
```

This command starts too many processes and generates an error message. It is recommended that you only use the buttons in the graphical user interface to run a utility from within MEDUSA4.

Example

Figure 295 shows a model of a quadrant of a cylinder used as input to the MEDMIF program.

Figure 295 The Input to MEDMIF



The MIF file generated from the model shown in the figure above is listed below:

```

MEDMIF 6.0    w=0.01000 p=0.00100 e=0.01000 of model
                D:\work\doku_beispiele\3D\block__1.mod
0  1  39 D:\work\doku_beispiele\3D\block__1.mod
0  2
0  3  0
0  4  3  0
0  -3
0  6
0  7  1  1  0.2415762E+02  0.5388490E+02  0.4312336E+02  0.1000000E+01
0  7  1  2  0.4562499E+02  0.3581250E+02  0.1516177E+02  0.1000000E+01
0  7  1  3  0.3761325E+02  0.3597182E+02  0.1516177E+02  0.1000000E+01
0  7  1  4  0.3761325E+02  0.3597182E+02  0.4930380E+02  0.1000000E+01
0  7  1  5  0.3220474E+02  0.3409060E+02  0.1516177E+02  0.1000000E+01
0  7  1  6  0.3220474E+02  0.3409060E+02  0.4930380E+02  0.1000000E+01
0  7  1  7  0.2703138E+02  0.2856451E+02  0.1516177E+02  0.1000000E+01
0  7  1  8  0.2703138E+02  0.2856451E+02  0.4930380E+02  0.1000000E+01
0  7  1  9  0.2503258E+02  0.2162750E+02  0.1516177E+02  0.1000000E+01
0  7  1  10 0.2503258E+02  0.2162750E+02  0.4930380E+02  0.1000000E+01
0  7  1  11 0.2503258E+02  0.1410261E+02  0.1516177E+02  0.1000000E+01
0  7  1  12 0.2503258E+02  0.1410261E+02  0.4930380E+02  0.1000000E+01

```

```

0 7 1 13 0.4562499E+02 0.1406250E+02 0.1516177E+02 0.1000000E+01
0 7 1 14 0.4562499E+02 0.1406250E+02 0.4930380E+02 0.1000000E+01
0 -6
0 8
0 9 2 1 1 7 2 1 2
0 9 2 2 1 9 2 2 3
0 9 2 3 1 2 2 3 4
0 9 2 4 1 8 2 4 1
0 9 2 5 2 9 2 3 5
0 9 2 6 2 3 2 5 6
0 9 2 7 2 8 2 6 4
0 9 2 8 3 9 2 5 7
0 9 2 9 3 4 2 7 8
0 9 2 10 3 8 2 8 6
0 9 2 11 4 9 2 7 9
0 9 2 12 4 5 2 9 10
0 9 2 13 4 8 2 10 8
0 9 2 14 5 9 2 9 11
0 9 2 15 5 6 2 11 12
0 9 2 16 5 8 2 12 10
0 9 2 17 6 9 2 11 13
0 9 2 18 6 7 2 13 14
0 9 2 19 6 8 2 14 12
0 9 2 20 7 9 2 13 2
0 9 2 21 7 8 2 1 14
0 -8
0 10
0 11 1 1
0 12
0 13 4 1 2 3 4
0 -12
0 14 1
0 -14
0 -11
0 11 1 2
0 12
0 13 4 -3 5 6 7
0 -12
0 14 1
0 -14
0 -11
0 11 1 3
0 12
0 13 4 -6 8 9 10
0 -12
0 14 1
0 -14
0 -11

```

```

0 11 1 4
0 12
0 13 4 -9 11 12 13
0 -12
0 14 1
0 -14
0 -11
0 11 1 5
0 12
0 13 4 -12 14 15 16
0 -12
0 14 1
0 -14
0 -11
0 11 1 6
0 12
0 13 4 -15 17 18 19
0 -12
0 14 1
0 -14
0 -11
0 11 1 7
0 12
0 13 4 -18 20 -1 21
0 -12
0 14 1
0 -14
0 -11
0 11 1 8
0 12
0 13 7 -10 -13 -16 -19 -21 -4 -7
0 -12
0 14 1
0 -14
0 -11
0 11 1 9
0 12
0 13 7 -20 -17 -14 -11 -8 -5 -2
0 -12
0 14 1
0 -14
0 -11
0 -10
0 16
0 -16
0 -2
0 -1

```

Specification of the MIF File Format

Each record in the file begins with a pair of integers in the FORTRAN format (I1,I3). The first integer is the continuation flag, and the second is the record type.

Continuation Flag: The continuation flag is one if the next record is a continuation of the current record. The continuation flag is 0 or a space if the current record is complete in itself or is the last of a set of records. In this chapter all records which can be continued are denoted by continuation fields of [0|1].

Record Type: The second integer in the record denotes the record type. A negative code terminates a record block.

Please note: The convention `<...>*` denotes a record that can be repeated.

The overall structure of the MIF file is as follows:

```
Model file ::= MIF file header record
              Model name record
              <Object>*
              End model record
```

where following is valid:

```
Object ::= Object record
          Object header record
          Object category/detail record
          <UPR>*
          End object header record
          Point block
          Edge block
          Surface block
          Text block
          End object record
```

The detailed syntax of the records for each section of the MIF file is given below. The FORTRAN format for each record appears below the record definition.

Model

MIF file header record ::	= MEDMIF <i>version_no tolerance_values</i> of model <i>model_name</i>
Model name record ::	= 0 1 <i>nchars model_name</i> (<i>nchars</i> is the number of characters in the file name)
End model record ::	= 0 -1

Object

Object record ::	= 0 2
End object record ::	= 0 -2
Object header record ::	= 0 3 <i>nchars object_name</i> (<i>nchars</i> is the number of characters in the <i>object_name</i> ; it is 0 if there is no <i>object_name</i>)
End object header record ::	= 0 -3
Object category/detail record ::	= 0 4 <i>category detail</i> where <i>category</i> is 1 if WIRE 2 if SHELL 3 if SOLID and <i>detail</i> is in the range [0,255]

UPR

UPR records represent UPR (uncommitted properties) texts in the model file.

UPR ::	= [0 1] 5 <i>nchars</i> text (I1,1X,2(I3,1X),64A1) where <i>nchars</i> is the number of characters in the current record
--------	---

Point block

Point block ::	= Point block record <Point>* End point block record
Point block record ::	= 0 6 (I1,1X,I3)
<Point>* ::	= 0 7 type id X Y Z W (I1,1X,2(I3,1X),I6,4(1X,E14.7))
	where type is 1 if 3D 2 if 4D
	id is the identifier of the point

Edge block

Edge block ::	= Edge block record <Edge>* End edge block record
Edge block record ::	= 0 8 (I1,1X,I3)
End edge block record ::	= 0 -8 (I1,1X,I3)
<Edge>*::	= [0 1] 9 type id sl sr npnt prefs (I1,1X,2(I3,1X),3(I6,1X),I3,8(1X,I6))
	where type = 1 if chain/polygon 2 if line (2 points) 3 if arc (3 points) 4 if cubic (4 points)
	id is the identifier of the edge
	sl is the identifier of the surface to the left of the edge
	sr is the identifier of the surface to the right of the edge
	npnts is the number of points in the current record
	prefs are the identifiers of the referenced points
Continuation edge record ::	= [0 1] 9 'spaces' npnts prefs (I1,1X,I3,26X,I3,8(1X,I6))

Surface block

Surface block ::	= Surface block record <Surface>* End surface block record
Surface block record ::	= 0 10 (I1,1X,I3)
End surface block record ::	= 0 -10 (I1,1X,I3)

Surface

<Surface>* ::	= Surface header record <Profiles> Rational Patch Geometry or Coons Patch Geometry End surface header record
Surface header record ::	= 0 11 type id (I1,1X,2(I3,1X),I6) where type = 1 if flat 2 if Coons Patch 3 through 10 reserved for triangular regions > 11 for Rational patches id is the identifier of the surface Note: If type has a negative value, then the surface has been created as the result of a Boolean operation.
End surface header record ::	= 0 -11 (I1,1X,I3)

Profiles

<Profiles> ::	= Profile header record <Profile>* End profile header record
Profile header record ::	= 0 12 (I1,1X,I3)
End profile header record ::	= 0 -12 (I1,1X,I3)
<Profile>* ::	= [0 1] 13 nedges erefs (I1,2(1X,I3),10(1X,I6)) where nedges is the number of edges in current record erefs are the identifiers of the edges Note: If erefs is negative, the direction of the edge is the opposite to that defined in the edge table.

Geometry

Geometry ::	= Geometry header record <Geometry point>* End geometry header record
Geometry header record ::	= 0 14 type (I1,2(1X,I3)) where type = 1 if no extra geometry information (that is, the boundary is sufficient) 2 if full surface definition follows
End geometry header record ::	= 0 -14 (I1,1X,I3)
<Geometry point>* ::	= 0 15 type id X Y Z W (I1,1X,2(I3,1X),I6,4(1X,E14.7)) where type = 1 if 3D 2 if 4D

Coons Geometry

Coons Patch geometry::	= Coons header record <Coons edge>* End Coons header record Note: Coons patches are defined by 4 edges and appear in the order v=0, u=1, v=1, u=0.
Coons header record ::	= 0 18 (I1,1X,I3)
<Coons edge>*::	= Coons Edge header record <Coons point>* End Coons Edge header record
Coons Edge header record ::	= 0 19 (I1,1X,I3)
<Coons point>*::	= [0]1 20 type X Y Z W (I1,1X,I3,1X,I6,4(1X,E14.7)) where type = 1 for ordinary point 2 for alignment point 3 for control point
End Coons Edge header record::	= 0 -19 (I1,1X,I3)
End Coons header record ::	= 0 -18 (I1,1X,I3)

Text block

Text block ::	= Text block record <Text>* End text block record
Text block record ::	= 0 16 (I1,1X,I3)
End text block record ::	= 0 -16 (I1,1X,I3)
<Text>* ::	= [0 1] 17 type ref nchars text (I1,1X,2(I3,1X),I6,1X,I3,1X,64A1)
	where type = 1 if vertex 2 if edge 3 if face/surface
	ref is the id of the vertex, edge, or surface
	nchars is the number of characters in the current record
Continuation record ::	= [0 1] 17 'spaces' nchars text (I1,1X,I3,12X,I3,1X,64A1)

MEDMIF Commands

Commands may be typed in uppercase or lowercase. They are shown in uppercase for clarity.

EDGTOL

| → EDGTOL <tolerance> →

Sets the tolerance within which edges are considered to be coincident. The value of this tolerance is shown as e in the MIF file header.

Specify tolerance in millimeters.

GO

| → GO →

Starts the translation process.

HELP

| → HELP →

Lists the available commands.

IN

| → IN <filename> →

Specifies the input model file name, where *filename* is the name of the model file.

OUT

| → OUT <filename> →

Specifies the name of the output MIF file, where filename is the name of the MIF file.

PNTTOL

| → PNTTOL <tolerance> →

Sets the tolerance within which points are considered to be coincident. The value of this tolerance is shown as p in the MIF file header.

Specify *tolerance* in millimeters.

QTOL

| → QTOL →

Queries the tolerances set up using EDGTOL, PNTTOL, and WINTOL.

QUIT

| → QUIT →

Exits from the MIF program and returns to MEDUTIL.

WINTOL

| → WINTOL <tolerance> →

Sets the tolerance within which a 3D text is considered to be associated with an edge. The value of this tolerance is shown as *w* in the MIF file header.

Specify *tolerance* in millimeters.



THE TERRAIN MODELER

- Introduction 456
- Constraints 463
- The Model File 465
- Creating the Model File using MEDUTIL 465
- Creating the Model File within MEDUSA4 470
- Viewing the Model 473
- Smoothing the Model Surface 477
- Specifying the Base Height 479
- Creating Contour Lines and Cross-Sections 480
- Gridding the Model Surface and Emphasizing Spot Heights . 484
- Naming an Object and Assigning It to a Detail Level 486
- Manipulating the Model 487
- Terrain Modeler Commands 489

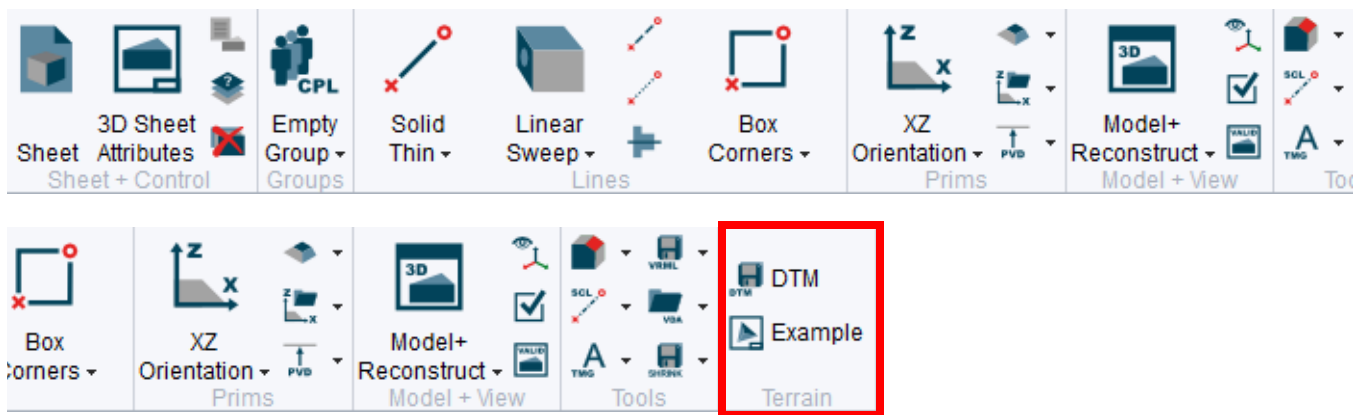
Introduction

The MEDDTM Product

The MEDUSA4 product **meddtm** offers the ability to create a 3D model of the ground surface. 3D viewers are used to display the model. **DTM** is the abbreviation for digital terrain modeler. The relevant buttons to run the Terrain Modeler from within the MEDUSA4 user interface are located in the Terrain tool group on the 3D tab.

The **meddtm** product must be integrated in the product list of your project to have the Terrain tool group available.

Figure 296 3D Tab - Terrain Tool Group



Functionality Overview

The Terrain Modeler creates a 3D model of the ground surface from X, Y, and Z-coordinates, i.e. a solid model that represents the surface of the ground or sub-surface strata. Typically, these models are used in the fields of cartography, civil engineering, geology, and energy exploration.

One advantage of the Terrain Modeler is the speed at which the Terrain Modeler produces a 3D representation of the ground surface. The terrain model then can be modified by using 3D features. In addition the geometric properties, such as center of gravity and volume, can be calculated. These features are useful in the fields of planning, design, and geographical information analysis. The model can also be viewed with varying light source, angle, perspective, and direction using the 3D Viewers. The Terrain Modeler has additional features such as contouring, cross-sectioning, and the highlighting of spot heights.

The Terrain Modeler uses the Delaunay triangulation to create the 3D model which works as follows:

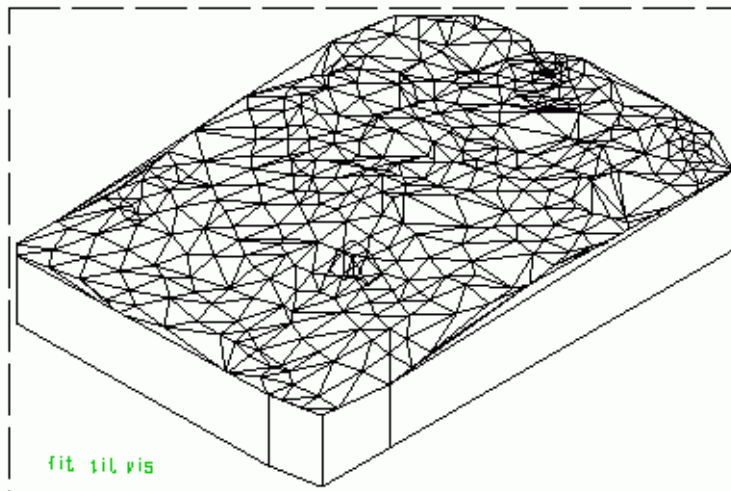
The input by which the surface is to be modeled can be either a text file containing X, Y, and Z-coordinates or a 3D sheet containing a viewbox with an XY prim and height values in a text standard format.

These points may be randomly distributed over the surface, and the Terrain Modeler fits an irregular network of triangles between them. To achieve the most natural looking representation of the terrain, the Terrain Modeler tries to fit the best shape of triangle between the points. It chooses which points to join to produce the most equilateral shaped triangles and avoids fitting points that would result in the creation of long, thin triangles and an unnatural surface appearance.

A model based on a maximum of 10,000 data points can be created, provided that the memory capacity of your machine is not exceeded. The triangles that fit between the coordinate points make up the surface of the model.

The triangles are made visible by using the viewer command `TIL VIS`. [Figure 297](#) shows an example.

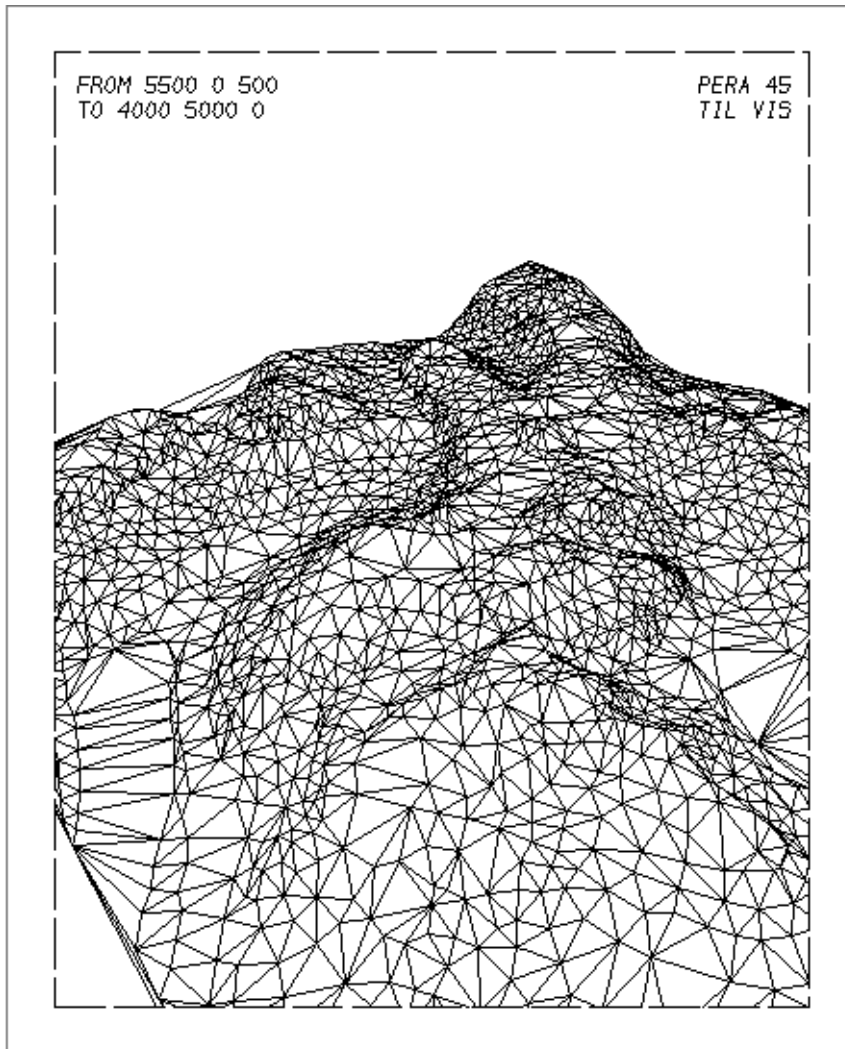
Figure 297 The Triangulation of the Model



Display Methods

Figure 298 shows a perspective view of the area. This view is created using the `FROM` and `TO` viewing commands so that the area is viewed from a height of 500 meters (meters are used as the unit of measurement in this example). The perspective angle is set at 45 degrees using the command `PERA 45`.

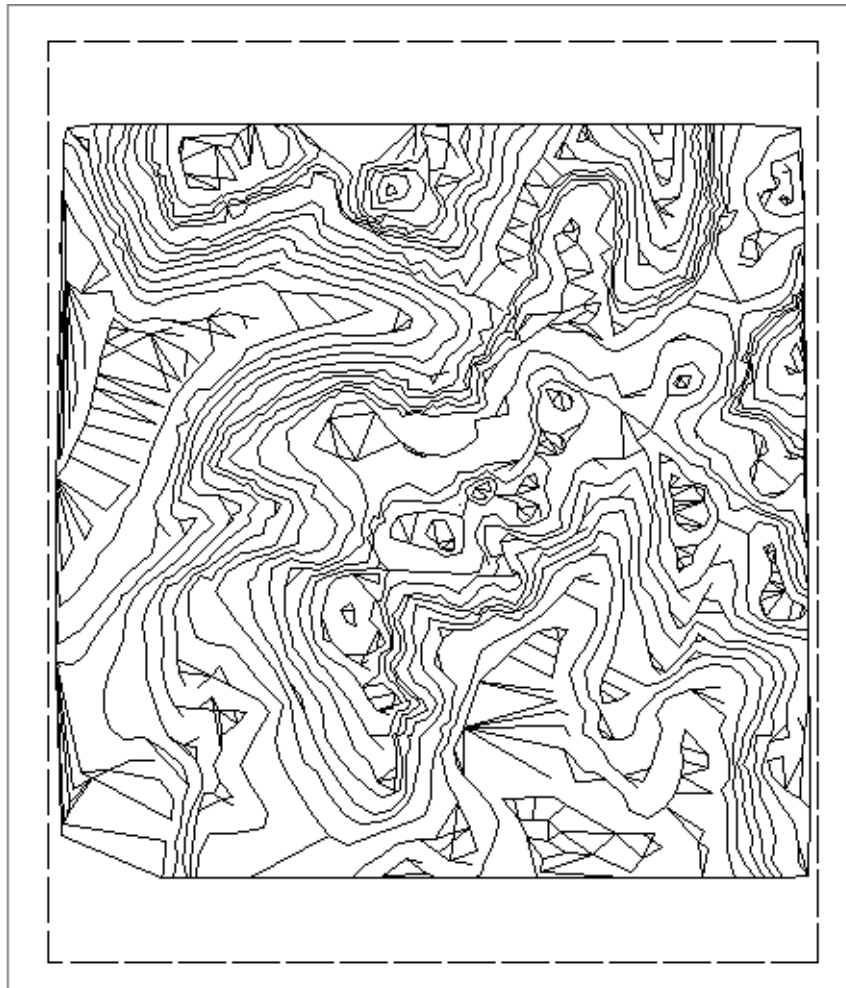
Figure 298 A Perspective View of the Model



The Terrain Modeler allows you to specify the height and the number of contour lines that are drawn on the model surface.

In [Figure 299](#), contour lines are drawn across the surface of the model at heights between 100 and 900 meters.

Figure 299 Contour Lines Drawn Between 100 and 900 meters



The Terrain Modeler allows you to section the model into separate objects if required. To do this you specify the number of sections and the direction in which they lie.

Figure 300 shows the model sectioned in the X-direction.

Figure 300 Sectioning the Model

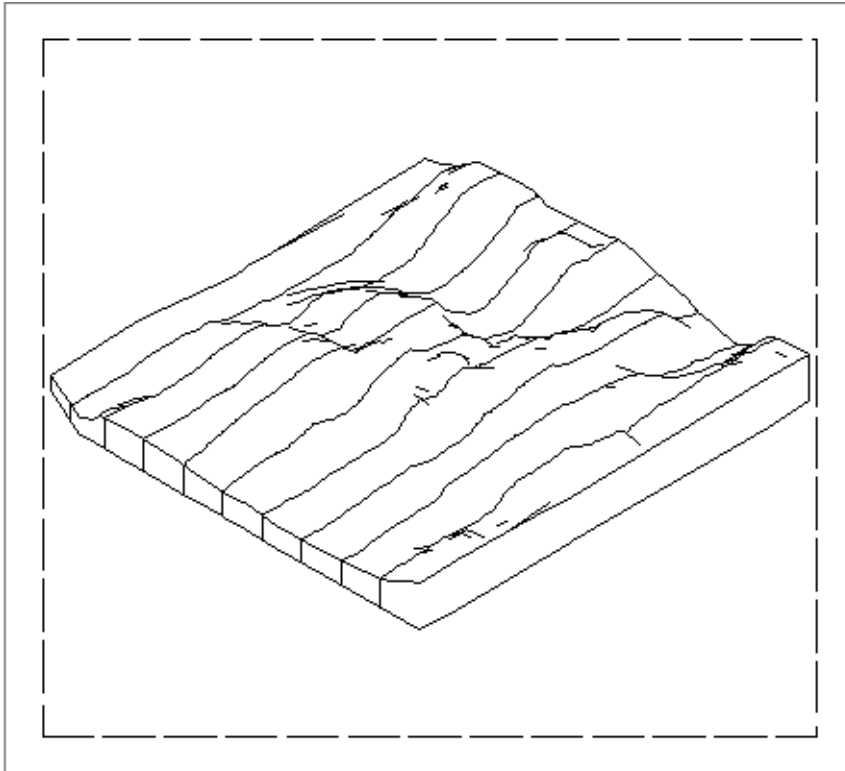


Figure 301 shows another feature of the Terrain Modeler where the spot heights of the model are represented by wire lines drawn from the base of the model to the surface. These spot heights are the X, Y, and Z-coordinates held in the data file.

Figure 301 Surface Spot Heights Represented by Wire Lines

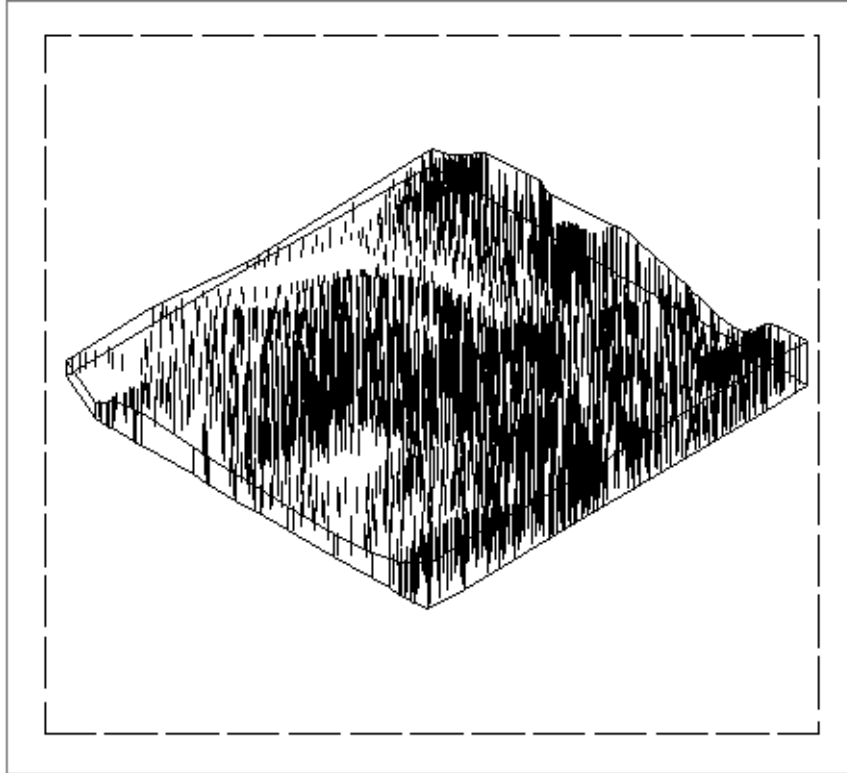
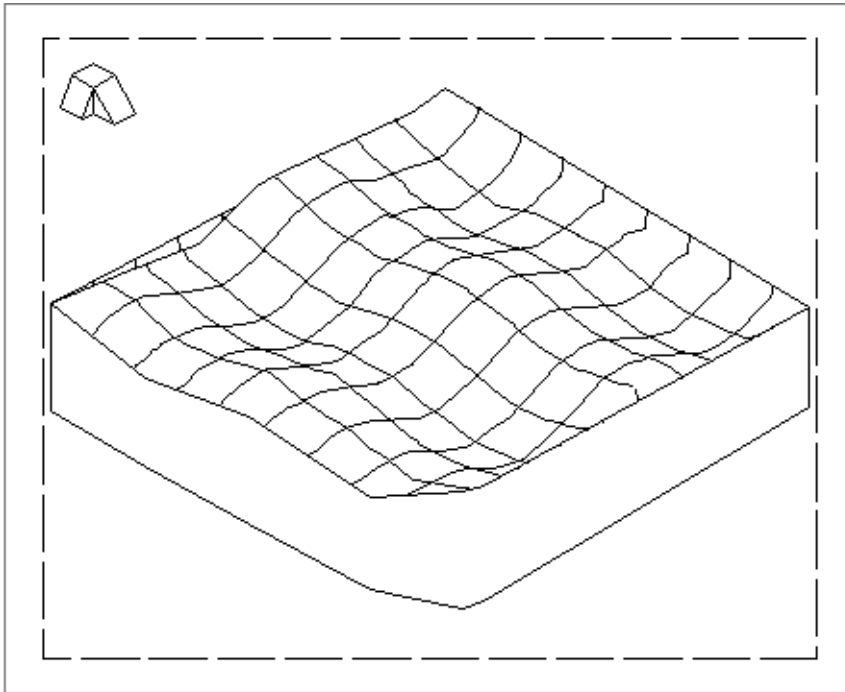


Figure 302 shows a square grid superimposed on the model surface.

Figure 302 A Grid Superimposed on the Model Surface



Constraints

The Delaunay triangulation, which is used by the Terrain Modeler to produce a model of the surface of the terrain, is most applicable when random coordinate points are used to represent the area to be modeled.

The choice of the Delaunay algorithm to generate a digital terrain model imposes certain constraints. The algorithm affects the representation of:

- Models produced from contour line data
- Concave models
- Boundary lines
- Breaklines
- Smooth contouring

The following subsections detail these constraints and suggest possible solutions.

Using Contour Line Data

The Delaunay triangulation algorithm may produce undesirable results when used with contour data. Data produced by digitizing contour lines results in a model with flat areas and a stepped cross-section. This is because the algorithm triangulates between points of equal height and therefore produces flat areas. To produce a realistic surface representation, use data points taken at random from the area.

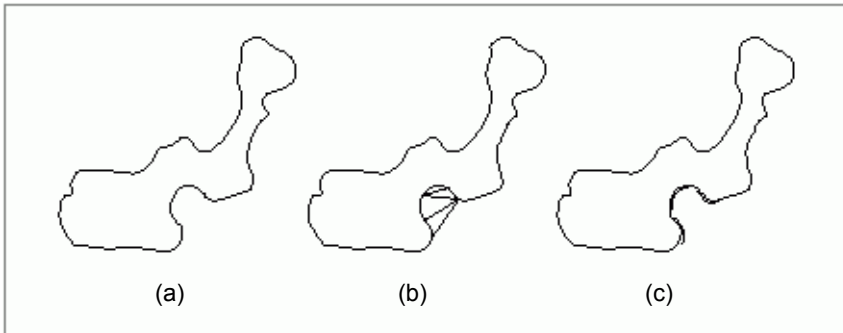
Concave and Convex Models

The Terrain Modeler is best suited to producing convex models. Concave models, which have hollows in the outer edge, are difficult to represent. [Figure 303 \(a\)](#) has many such hollow areas.

On concave areas, the triangulation extends beyond the boundaries of the model. As a result, the concave outline is broken up by the triangles. This effect is shown in [Figure 303\(b\)](#).

One way of avoiding this effect is to add data points to the coordinate data file so that there is a discontinuity around the concave area. For example, adding zero Z-coordinates creates a cliff outline so that the area of interest is delimited by the cliff edge, as shown in [Figure 303\(c\)](#).

Figure 303 Triangulation at Concave Model Boundaries



Boundary Lines

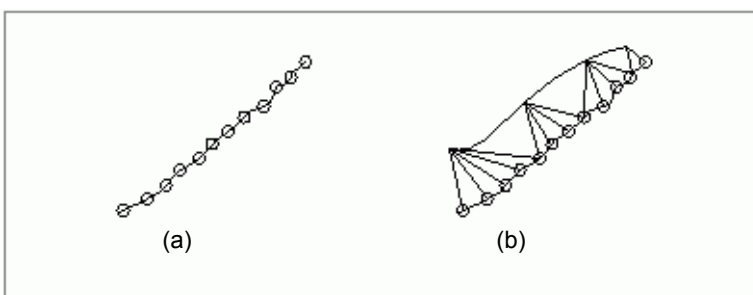
Boundary lines are not recognized by the Terrain Modeler. For example, an enclosed lake in the middle of an area may not be treated as a flat surface. The Terrain Modeler will triangulate across the surface of the lake so that it is undulating. The solution to this problem is to enter the MEDUSA4 Modeling System and use Boolean operations to superimpose a flat surface on the model.

Breaklines

Normally, the Terrain Modeler cannot define breaklines (such as river beds or streams) on the model surface. This is because the triangulation algorithm may triangulate across the discontinuity and produce a saw-toothed effect. One way of defining a breakline is to include additional closely spaced data points to represent the breakline. An example is shown in [Figure 304\(a\)](#).

The inclusion of additional points in the coordinate data file ensures that the triangulation will follow the breakline. This effect is shown in [Figure 304\(b\)](#).

Figure 304 Defining Breaklines for Correct Triangulation



Smooth Contours

The contouring algorithm that creates contour lines on the model surface currently only uses straight line interpolation. The contours are drawn as straight line segments between the points of the triangles and not as smoothed curved lines. It is possible to create smooth contours in the MEDUSA4 Drafting System by editing the contour line and using the `FITL` command. This command draws a smooth curve through all the points in the current line. Note that this technique may produce overlapping contours and that extra points are created on the contour line.

The Model File

The Terrain Modeler only creates a model file, but you will not see a terrain model on the screen. To display the terrain model, you must use one of the MEDUSA4 3D Viewers.

There are several ways to cause the Terrain Modeler to create a model file:

- from outside MEDUSA4 by using the MEDUTIL utility program
- from inside MEDUSA4 by using the DTM  tool on the 3D tab (for details see [“Creating the Model File within MEDUSA4” on page 470](#))

Creating the Model File using MEDUTIL

There are two ways to execute commands for the Terrain Modeler:

- By executing a macro file
- By typing individual commands

The usual procedure is to create a macro file containing Terrain Modeler commands.

The procedure by using a macro file is the following:

1. Create a data file containing X, Y, and Z-coordinates (text file)
2. Create a macro file containing Modeler commands (text file)
3. Access Terrain Modeler from MEDUTIL
4. Issue commands by running the macro file
5. Leave the Terrain Modeler

First of all you need the text file, which contains the X, Y, and Z-coordinates of a particular area. Section [“Creating a Data File”](#) describes what you need to consider when you create your own data file.

Please note: The meddtm product provides the example file *shutt.txt* which contains 200 coordinates. The file is located in:

<MEDUSA4 installation path>meddtm\m3d\example\shutt.txt

It has been provided to enable you to practise using the Terrain Modeler before using your own data.

To use the file, please, copy it into your own directory and name it, e.g. *surface.txt*.

The second step is to create the macro file with the modeler commands for the Terrain Modeler. The commands, which the Terrain Modeler at least requires to create a model file are introduced in section “[Basic Commands \(MEDUITIL\)](#)” on page 467. Section “[Creating a Macro File](#)” describes how to insert the commands into a macro file.

Creating a Data File

The Terrain Modeler uses a data file containing the X, Y, and Z-coordinates of an area. The data file must be created as a text file before the Terrain Modeler is run and must contain the coordinates as a list in the following 3-column format:

X-coordinates Y-coordinates Z-coordinates

Example:

```
4719 7000 413
4703 6985 441
4566 6985 441
4510 6846 440
4426 6599 439
4338 6448 439
```

The text file must meet the following requirements:

- The file may contain coordinate data only (no column heading!)
- Each column must be separated by at least one space
- No `Return` after the last coordinate
- A minimum of 3 data points
- A maximum of 10,000 data points
- The file name must contain the suffix *.txt*

Real numbers as well as integers are allowed in the file. You may need to scale the coordinate data before entering it in the data file because the MEDUSA4 3D Modeler draws the terrain model in the sheet in millimeters, if metric units of measurement are being used.

If the file contains any other information (comments, headers, footers, and so on) the Terrain Modeler generates the following error message when the data file is used:

```
Error in file at line <number>
```

The data file can be placed in any directory as the location of the file can be specified in the Terrain Modeler. The data file can be given any name, but must contain the suffix *.txt*.

An example coordinate data file is supplied with MEDUSA4. It has been provided to enable you to practise using the Terrain Modeler before using your own data.

Basic Commands (MEDUTIL)

The basic commands that are used in the Terrain Modeler are described below.

`IN filename`

Tells the Terrain Modeler to look in the specified file for the coordinate data. If the file is not located in the current directory, the full file specification must be given.

`SMALLEST ANGLE degrees`

Sets the smallest internal angle of each triangle that is used to produce the terrain surface. The Terrain Modeler generates an error message if the coordinate data produces a model containing angles smaller than the specified minimum. The default value is five degrees. The smallest angle must be in the range of 0 through 60 degrees.

`OUT filename`

Specifies the name of the model file in which the Terrain Modeler stores the completed model. The filename must contain the suffix *.mod*.

`GO`

Starts the triangulation process to produce the model of the terrain surface.

Creating a Macro File

The macro file contains the commands which the Terrain Modeler requires to create the model file. It can be created and edited by using the common text editor on your system. The commands at least must contain:

- the name of the text file containing the X, Y, Z coordinates
- the smallest internal angle of the triangles
- the name of the model file which is to be created

An example of a simple macro file to be run is shown below. Create this macro file in your own directory, which should also contain the data file. Call the macro file, e.g. *surface.mac*.

```
in surface.txt
smallest angle 0.01
```

```
out surface.mod
go
```

Running the Terrain Modeler by using MEDUTIL

The program can be entered outside MEDUSA4 by means of the `DTM` command in MEDUTIL, the MEDUSA4 Utility Access facility.

A brief introduction to MEDUTIL is shown in the following example.

Please note: In the example, user input is in bold type.

Start MEDUTIL by typing the `medutil` command:

```
      MEDUSA4 Utility Control
      ~~~~~
Type 'help' for list of commands...
Enter command> DTM
      MEDUSA4 Digital Terrain Modeler
      ~~~~~
Type HELP for a list of Digital Terrain Modeler commands
Dtm>
```

Digital Terrain Modeler commands can be entered at the `Dtm>` prompt. See [“Viewing the Model” on page 473](#) to see how to enter commands with the macro file.

At the prompt you can enter the command `HELP` to see a list of all the commands that are available in the Terrain Modeler.

WARNING: Do not try to access MEDUTIL from within MEDUSA4 by typing:

```
MEDUTIL project_name
This command starts too many processes and generates an error message.
```

Running the Macro File

You can now issue commands to the Terrain Modeler by running the macro file `surface.mac`.

1. To run the macro file, type the command:

```
MACRO SURFACE.MAC
```

The commands contained in the macro file are displayed separately on the screen as they are executed, in the example below:

```
SMALLEST ANGLE 0.01
OUT SURFACE.MOD
GO
```

The Terrain Modeler then reads the data file containing X, Y, and Z-coordinates. As the example data file contains 200 coordinates, the following message appears:

```
200 points read
```

Whilst the Terrain Modeler creates the model from the 200 data points held in the example file the following message appears:

```
200 points being triangulated
```

- . At the end of the model creation the system prompt (*) is redisplayed.
2. To leave the Terrain Modeler, type: `QUIT`
This command takes you back to the Utility Control Program (`medutil`).
3. To leave this program and return to the operating system, type `QUIT`

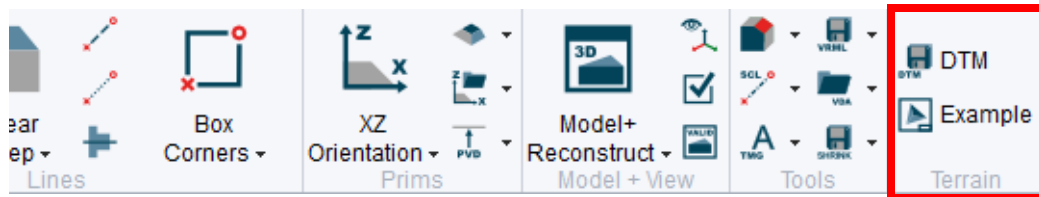
The created model file (example: *surface.mod*) now in the current directory.

This model can now be viewed by entering the MEDUSA4 Drafting System and running the 3D Viewer.


Creating the Model File within MEDUSA4

To create a model file you can run the Terrain Modeler also within MEDUSA4 using the DTM tool located in the Terrain tool group on the 3D tab.

Figure 305 3D Tab - Terrain Tool Group



The procedure is as follows:


- Open or Create an empty 3D sheet containing a viewbox with XY-prim and height values (Z-coordinate).
- Click on the DTM button  to run the Terrain Modeler

That means, you initially need a 3D sheet which contains the elements mentioned above.

Please note: An example is provided with the meddtm product which contains height data. The file is located in the directory:

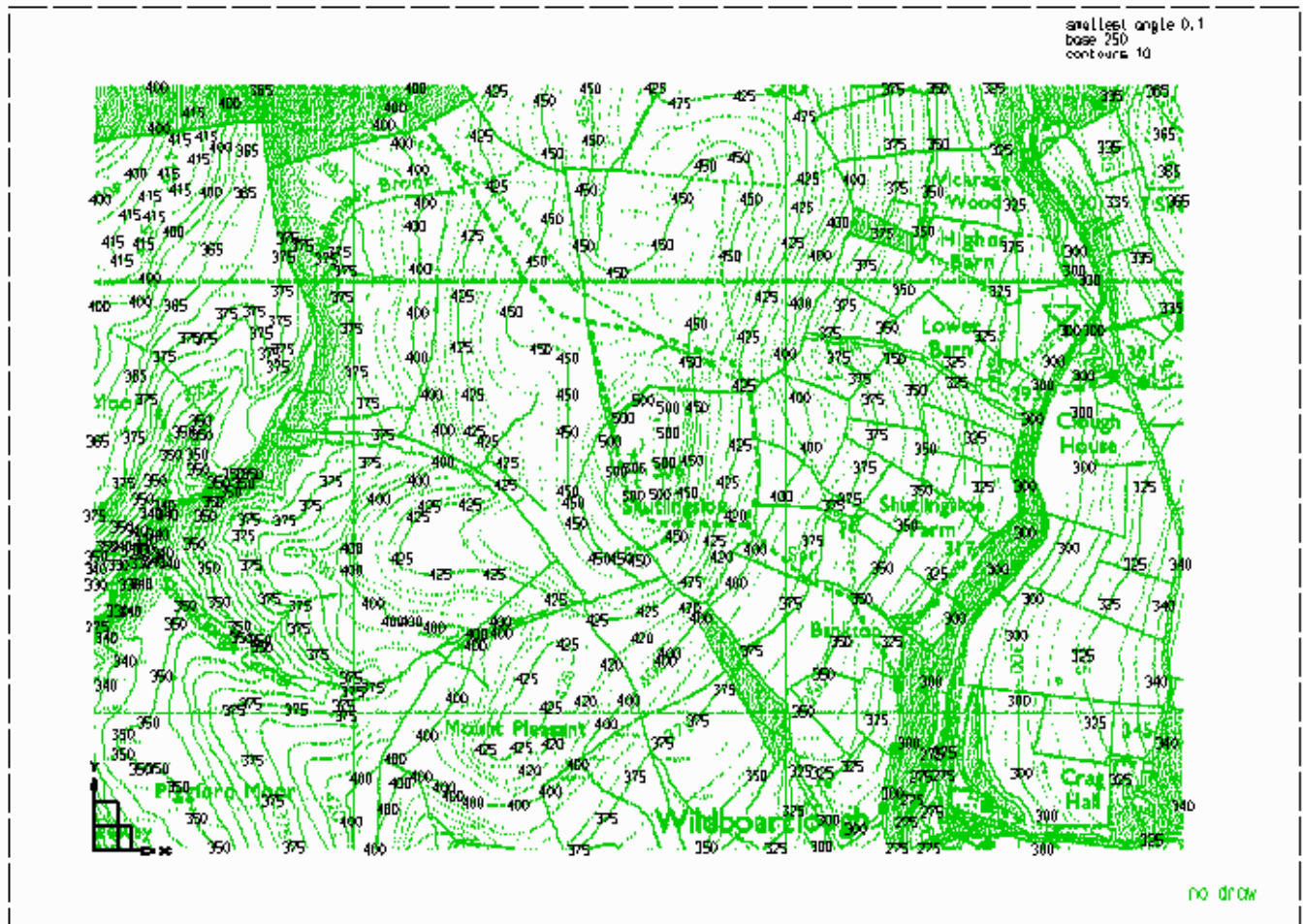
`<MEDUSA4 Installationspfad>\meddtm\m3d\example\shutt.she`


To use the file, please, copy it into your own directory and name it, e.g. *surface.she*.

Either load the example sheet by clicking on the Example button  in the Terrain tool group or create a new 3D sheet with viewbox, DXY prim and height data.

The following figure shows the viewbox of the example sheet which contains the DXY prim, the Z-coordinates and a contour map as raster backdrop. The height values are inserted as simple text in standard format. The datum point of the text specifies the X- and Y- coordinates.


Figure 306 Viewbox of the DTM Example Sheet



Run the Terrain Modeler by clicking on the DTM tool . The Terrain Modeler creates a *.mod* file of the current sheet. The model file is automatically saved in the same directory under the same name with suffix *.mod*.

Running the Digital Terrain Modeler from within MEDUSA4

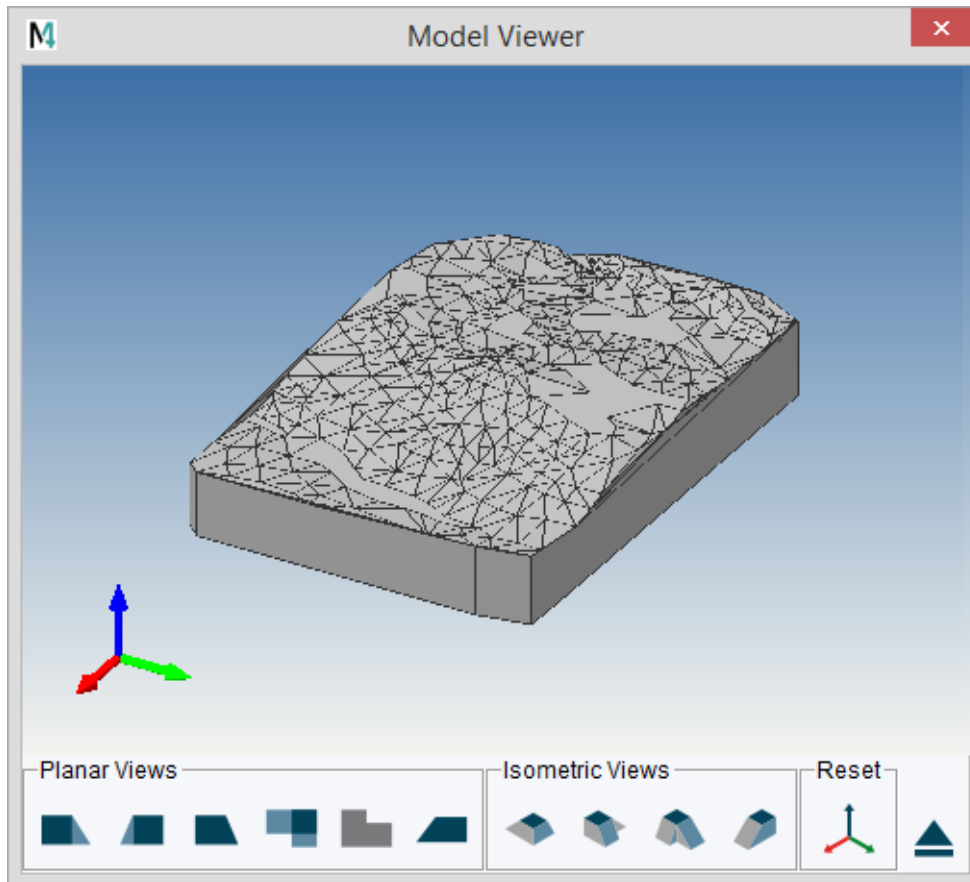
The tool to run the DTM program is located on the 3D tab in the Terrain tool group.

Choose the DTM tool . The Digital Terrain Modeler creates a *.mod* file from the currently loaded sheet. Precondition is, that the sheet contains height coordinates in a viewbox with a DXY prim. The model file is automatically stored under the same name in the same directory with the suffix *.mod*.

The sheet contains a map with contour lines as background raster image on which the values of the heights are inserted as text of simple standard format. The datum point of the text specifies the x- and y- coordinates.

Once the model file is created, the Model Viewer automatically opens and displays the 3D model.

Figure 307 Model Viewer - 3D Model of the Example

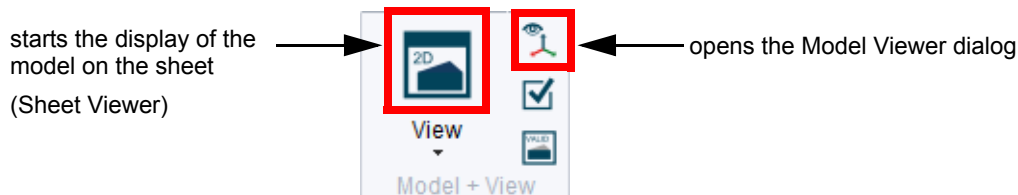


Viewing the Model

The model file *.mod* created by the Terrain Modeler can be viewed using the 3D viewers:


The tools to start the 3D viewers are located in the *Model + View* tool group.

Figure 308 Model + View Tool Group - View and Model Viewer Tools



The Sheet Viewer

This viewer can be used to produce a view of the terrain model on the 3D sheet, which can then be plotted. You can place viewer commands on the sheet in different ways:

- Use the 3D text creation tool  on the ribbon > 3D tab > Tools tool group to create TVS texts of style *View Specification Text*.
- Select the viewing commands on the *3D Attributes (Sheet Level)* dialog (commands affect entire sheet)
- Select the viewing commands on the *Viewbox Dashboard* (commands affect only selected viewbox)

For basic information on viewing commands see chapter [“Basic Viewer Commands” on page 53](#), for information on using the *3D Attributes* dialog see chapter [“3D-Attributes” on page 37](#).

The Model Viewer


The Model Viewer allows various views of the model. For details to the Model Viewer see chapter [“The Model Viewer” on page 343](#).


Displaying the Model after Creating by using MEDUTIL

Displaying on the Sheet

When you have created the model file by using the MEDUTIL utility program, only the text file containing coordinates and the *.mod* file (example *surface.mod*) exist but not yet a MEDUSA4

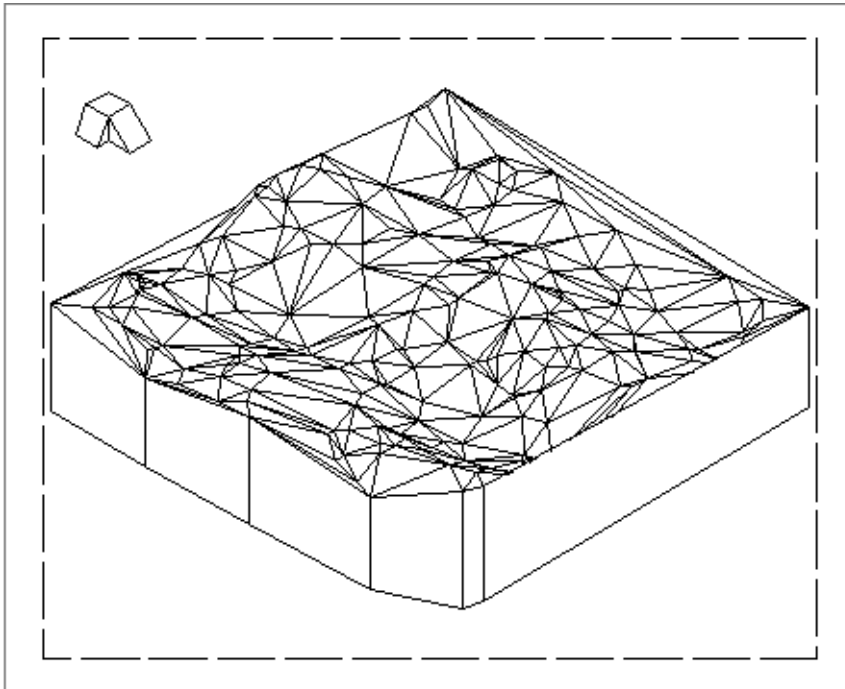
sheet. Thus, the first step is to open or create the sheet on which the model, created by the Terrain Modeler, is to be displayed.

1. Open a blank 3D sheet or create your own sheet with the required viewboxes and prims.
2. Give the sheet the same name as the model file created by the Terrain Modeler. In this example, enter `SURFACE` as Drawing No. and save the sheet (*surface.she*) in the same directory where the model file is saved.
3. Enter the viewing commands `TIL VIS` and `FIT` onto the 3D sheet. The different ways to create the commands are described in [“The Sheet Viewer” on page 473](#).
The `TIL VIS` command displays the triangles that make up the model surface.
The `FIT` command fits the model into the viewbox by altering the scale of the model.
4. To view the model, click on the View tool  which is located in the Model + View tool group (see [Figure 308](#)).

Please note: Do not use the Model + Reconstruct tool  because the 3D modeler will then use the blank sheet to create a new model and overwrite the model file created by the Terrain Modeler.

[Figure 309](#) shows the model created from the coordinates in the example file.

Figure 309 The Model Created from the Example Data



Displaying in the Model Viewer Dialog


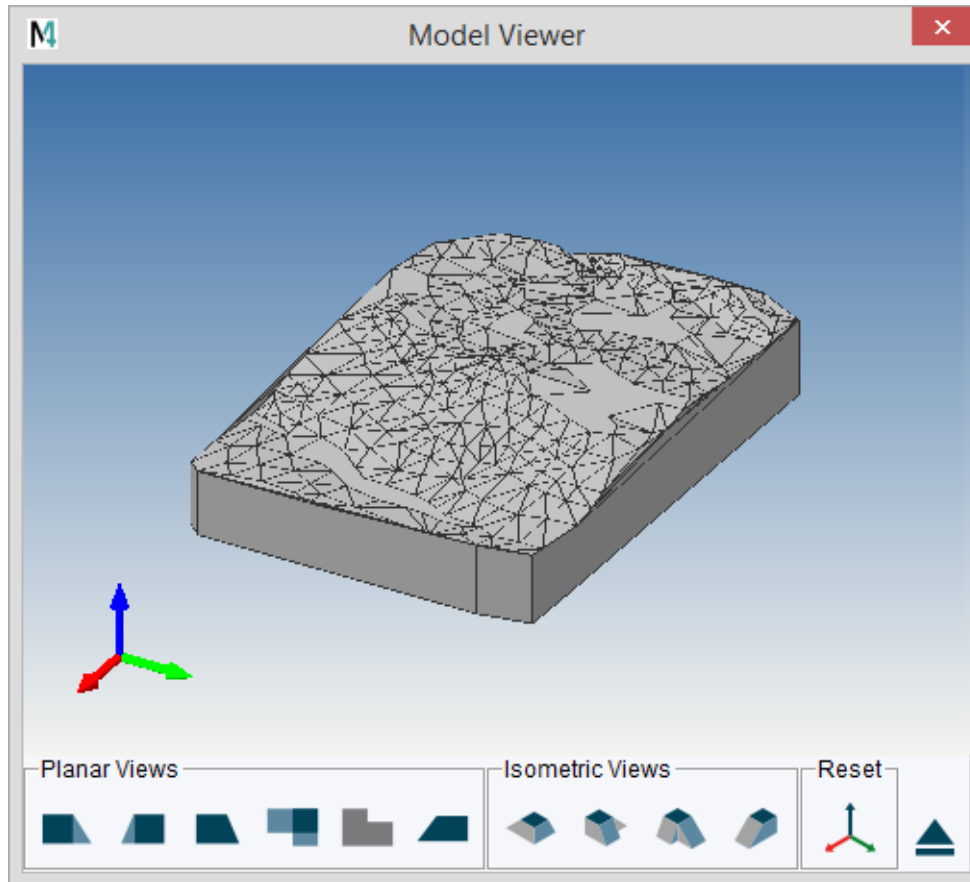

To display the model in the Model Viewer Dialog, choose the Interactively view current model tool .


Figure 310 Model Viewer - 3D Example Model



Displaying the Model after Creating it within MEDUSA4

Displaying on the Sheet


Creating a model file within MEDUSA4 requires an existing 3D sheet. At least it must contain a viewbox with DXY prim and height coordinates. The Terrain Modeler which runs by using the DTM tool  creates the model file in the same directory as the sheet.

1. Load the sheet, if it is not yet opened.
2. Add further viewboxes and prims.
3. Position the viewer commands on the sheet (see [“The Sheet Viewer” on page 473](#)).
4. Click on the View button  in the Model + View tool group.

The model views are displayed on the sheet.

Displaying in the Sheet Viewer Dialog

The Sheet Viewer dialog is automatically displayed once the model is created (see end of section [“Creating the Model File within MEDUSA4” on page 470](#)).

When the dialog is closed, you can click on the Interactively current view current model tool  to open it again. The possibilities to control the display in the Viewer dialog are described in chapter [“The Model Viewer” on page 343](#).

Smoothing the Model Surface

Smoothing the surface of the model means, that the net of triangles created by the Terrain Modeler will be refined to give the surface a more natural appearance. The Terrain Modeler smooths a surface by interpolating between the existing coordinates to create more data points. These additional data points are used by the algorithm to create more triangles on the model surface.

The size of the triangles created by the Terrain Modeler can be limited in two ways:

- An area tolerance can be specified for the triangles. All triangles larger than this limit are divided into three smaller triangles.
- The smallest internal angle of a triangle can be specified.

The commands used for smoothing are:

`SMOOTH ON`
Turns ON surface smoothing.

`SMOOTH OFF`
Turns OFF the surface smoothing. It is the default.

`SMOOTH AREA tolerance`

Specifies the largest area contained within a triangle. The default tolerance is 10 square units. If the area tolerance specified is too small or the default value is too small for your application, the Terrain Modeler generates one of the following error messages:

```
Ill-conditioned data, triangles created too thin  
Insufficient memory
```

These messages mean that the smoothing process has produced too many triangles and therefore a larger area tolerance is required.

`SMOOTH ANGLE degrees`

Specifies the smallest internal angle of each triangle. The default angle is 15 degrees. The value given to this angle cannot be less than the value given to the smallest angle in the triangulation process by the `SMALLEST ANGLE` command.

The example below shows a typical sequence of commands that includes smoothing:

```
IN SURFACE.TXT  
SMALLEST ANGLE 0.01  
SMOOTH ANGLE 5  
SMOOTH AREA 4  
SMOOTH ON  
OUT SURFACE.MOD  
GO
```

Figure 311 shows the effect of smoothing on the surface of the example model. Compare this figure with Figure 312, which shows the same model without surface smoothing. Note that more edges are created when you smooth the model surface.

Figure 311 The Effect of Smoothing the Model Surface

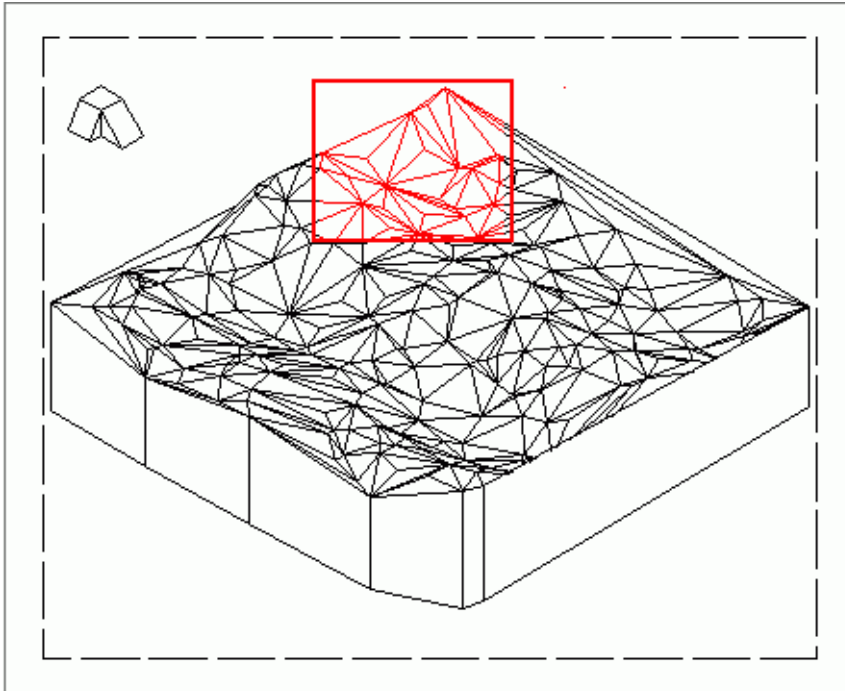
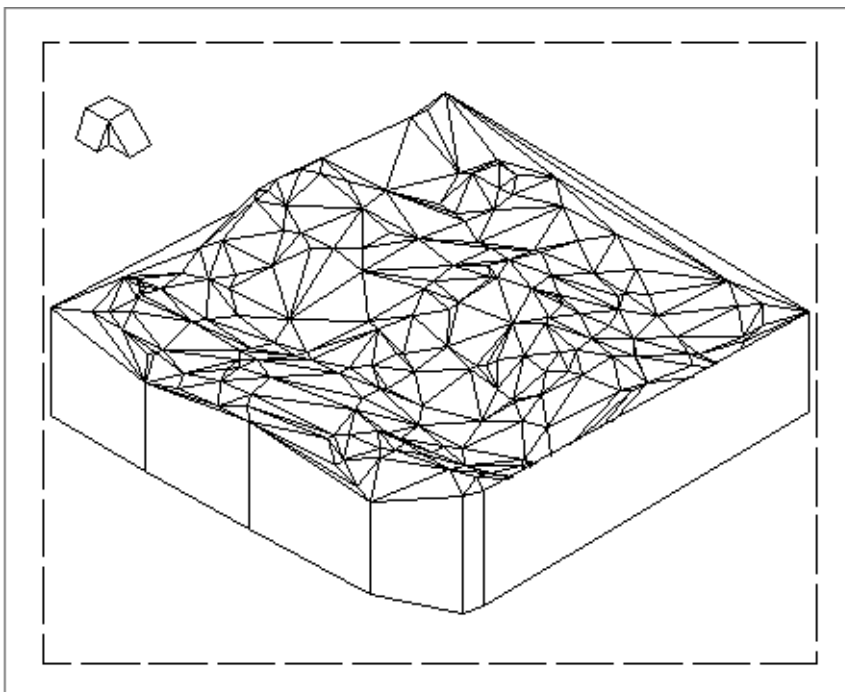


Figure 312 Model Surface without Smoothing



Specifying the Base Height

By default, the base height of a model is set to zero (at sea level). If your application requires a different base height, you can specify a new height for the model using the command:

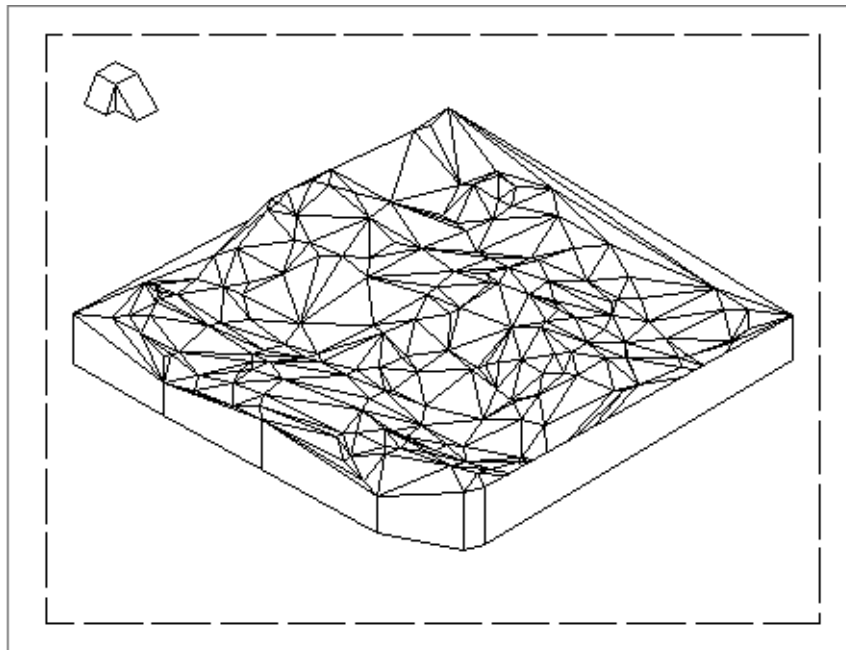
```
BASE height
```

As the elevation of a point (Z-coordinate value) is usually specified relative to sea level, the `BASE` command can be used to ensure that the base of the model is lower than the smallest Z-coordinate. This ensures that all points in the model created are above the base height.

Figure 313 shows the effect of specifying a base height of 3 model units for the model. (In this example meters are used as model units). Compare **Figure 313** with **Figure 311** to see the effect of changing the base height of the model. Use the following sequence of commands to reproduce this model from the data in the example data file:

```
IN SURFACE.DAT
SMALLEST ANGLE 0.01
BASE 3
OUT SURFACE.MOD
GO
```

Figure 313 The Effect of Specifying a New Base Height



Creating Contour Lines and Cross-Sections

The Terrain Modeler allows you to draw contour lines across the surface of the model and to create objects which represent cross-sections of the model.

Drawing Contour Lines

Contour lines can be drawn across the surface of the model in one of two ways:

- At a specific height
- At equally spaced intervals between the maximum height and the base height of the model

The Terrain Modeler uses straight line interpolation to create the contour lines. They are created as wirelines.

The commands used for drawing contour lines are:

`CONTOURLINE height`

Draws contour lines for a specific elevation.

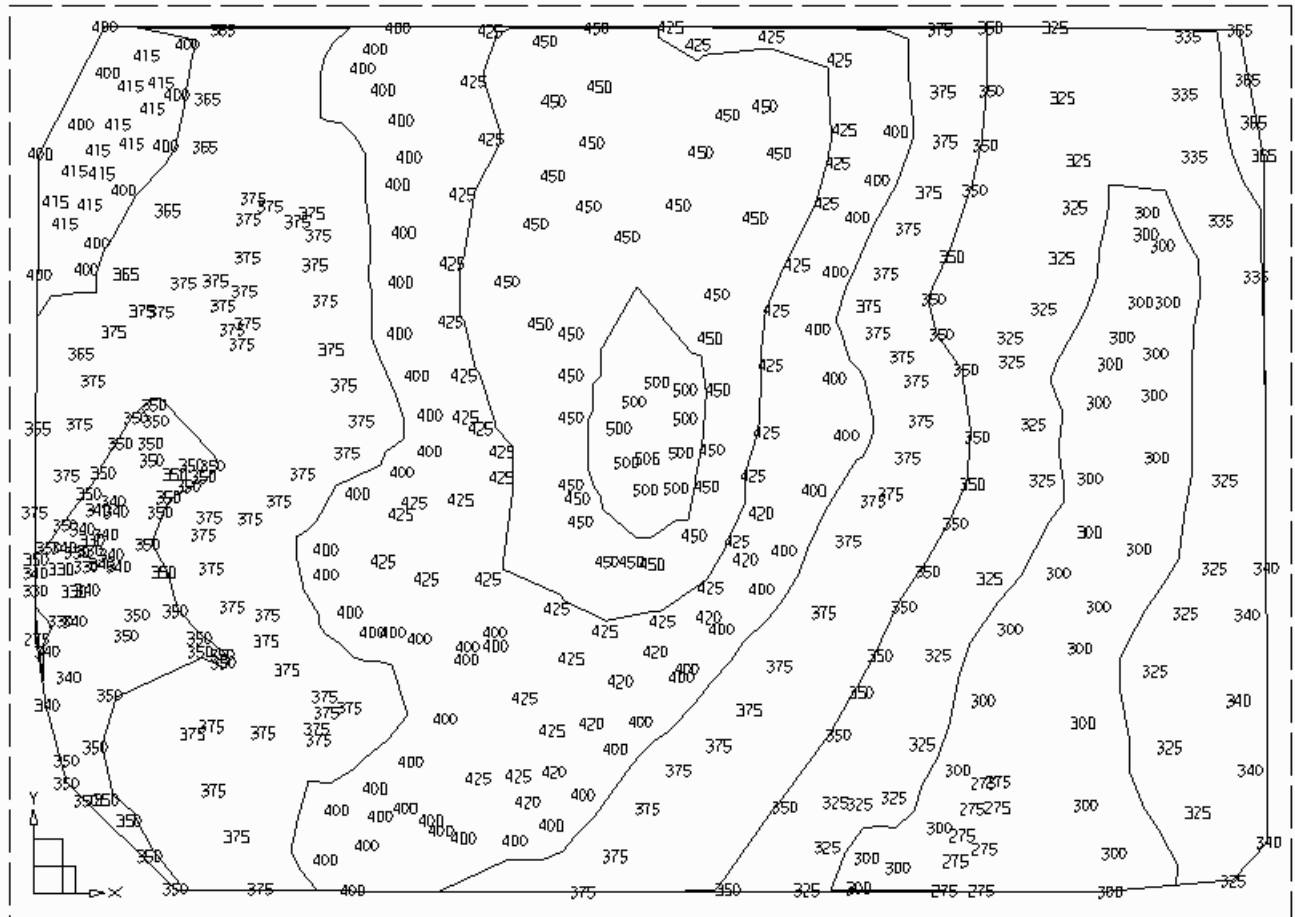
`CONTOURS no. of contours`

Draws a number of contour lines on the model surface.

The Terrain Modeler finds the minimum and maximum height of the model and calculates the heights at which to draw the specified number of contour lines so that the contours are equally spaced in height.

[Figure 314](#) shows the effect of drawing five contour lines on the surface of the example model using the command `CONTOURS 5` after the `GO` command in the macro file *surface.mac*.

Figure 314 Contour Lines on the Model Surface



Creating Cross-Sections

Objects can be created which represent cross-sections of the model. These objects are modeled as 2D shells. The cross-sectioning commands are:

`XSECT X no. of sections`

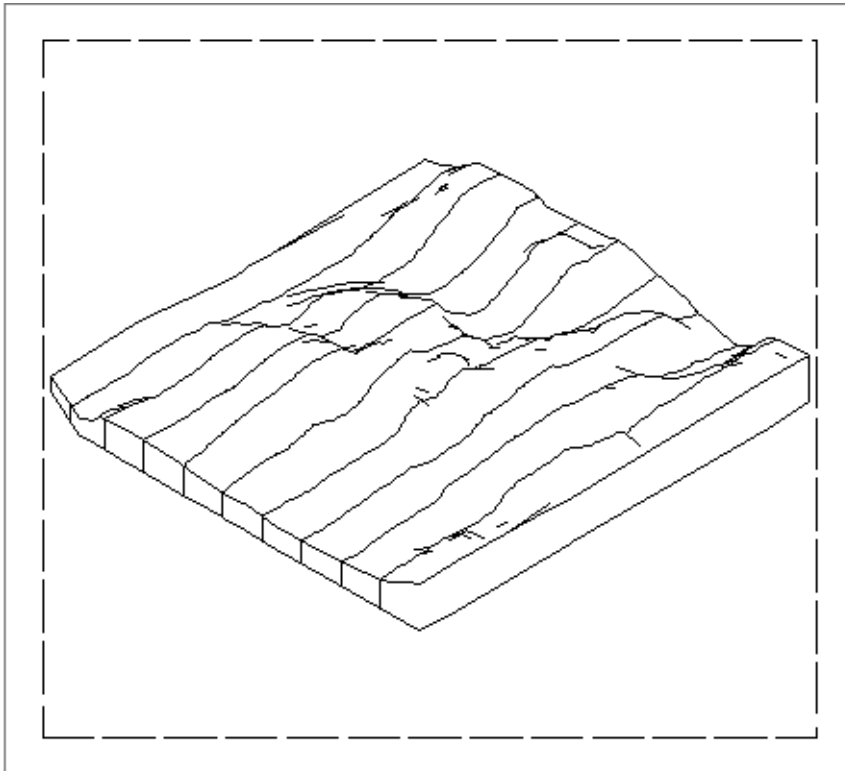
Creates the specified number of cross-sections in the direction of the X-axis of the model.

`XSECT Y no. of sections`

Creates cross-sections in the direction of the Y-axis.

Figure 315 shows eleven cross-sections in the direction of the X-axis of the model using the command `XSECT X 11`.

Figure 315 The Model Cross-Sectioned in the X-axis Direction



The Terrain Modeler also allows you to create any arbitrary cross-section through the model. The start and end points of the cross-section are specified using the command:

`XSECT x1 y1 x2 y2`

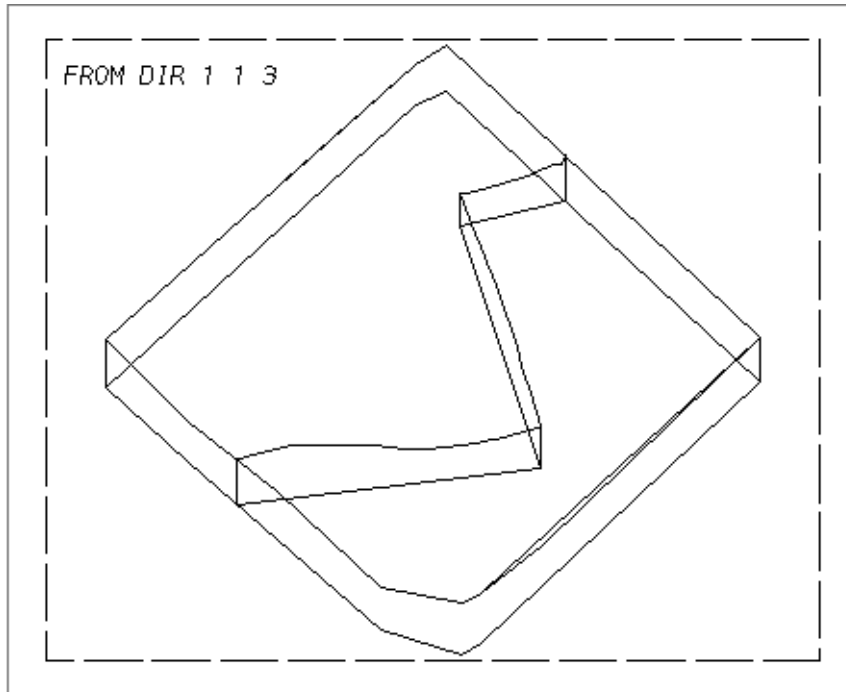
The Terrain Modeler creates a cross-section starting at the point specified by the coordinates `x1 y1` and ending at the point specified by `x2 y2`.

Figure 316 shows a zig-zag cross-section of the model. The following three commands are required to define this cross-section:

```
XSECT -10 0 -5 -3
XSECT -5 -3 2 10
XSECT 2 10 15 0
```

To reproduce this model, place these commands in the macro file *surface.mac* after the GO command.

Figure 316 Zig-Zag Cross-Sectioning



Gridding the Model Surface and Emphasizing Spot Heights

The Terrain Modeler can project a square grid onto the model surface or display spot heights by drawing vertical wirelines from the base of the model to its surface. The spot heights are the X, Y, and Z-coordinates contained in the original data file.

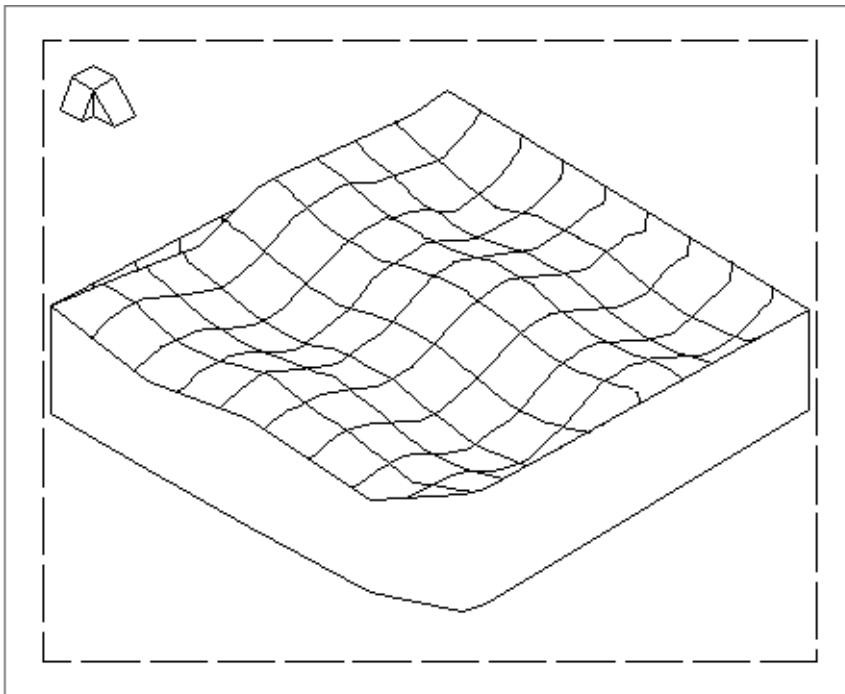
Projecting a Grid Onto the Surface

A grid can be projected onto the surface of the model using the command:

```
GRID square size
```

The square size is specified in model units. [Figure 317](#) shows a grid across the surface of the example model. You can reproduce this effect by placing the command `GRID 2.5` after the `GO` command in the macro file *surface.mac*.

Figure 317 A Grid Projected onto the Model Surface



Please note: The viewbox may not contain the `TIL VIS` command!

Displaying Spot Heights

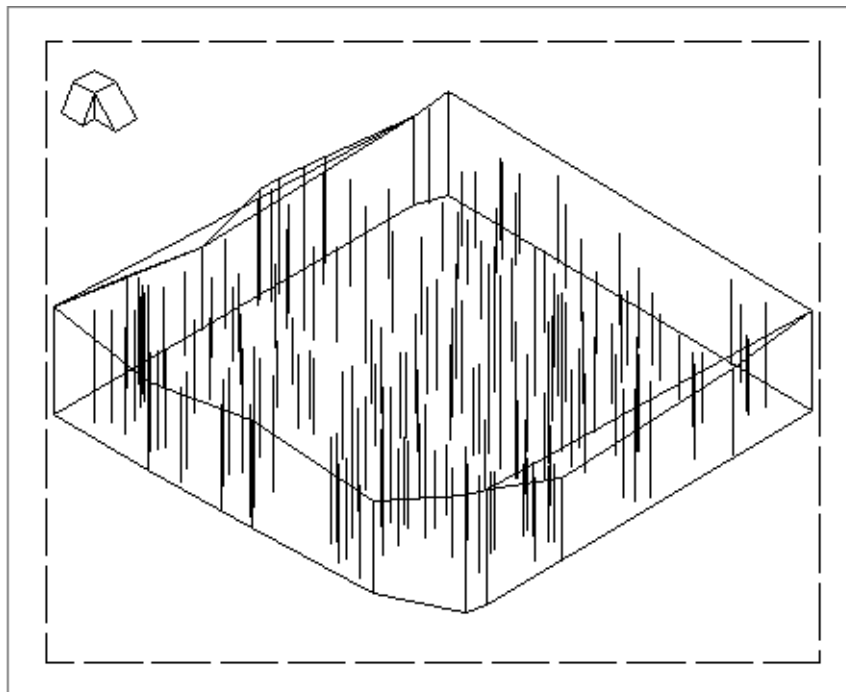
To display the spot heights of a model use the command:

```
SPOTHEIGHTS
```

Place this command after the `GO` command in the macro file *surface.mac*.

Figure 318 shows the spot heights of the example model.

Figure 318 Spot heights



Please note: To view the spot heights the view command `SKETCH` has to be placed either in the viewbox or the sheet.

Naming an Object and Assigning It to a Detail Level

Each object in the terrain model can be named and placed at a detail level in the model structure.

Naming an Object

You can give a name to each object in the terrain model by using the command:

```
OBJECT NAME name
```

All subsequent objects that you create after issuing this command are given the specified name until you select a new name with another `OBJECT NAME` command.

For example, to give the name `CROSSX` to the cross-sections parallel to the X-axis and the name `CROSSY` to the cross-sections parallel to the Y-axis, enter the following sequence of commands in the macro file `surface.mac` after the `GO` command:

```
OBJECT NAME CROSSX  
XSECT X 5  
OBJECT NAME CROSSY  
XSECT Y 5
```

When you view the finished model, use the viewer command `OBJ name`. This allows you to view the named objects separately. To view all the objects, use the command `OBJ ALL`.

Assigning a Detail Level to an Object

You can specify a detail level for each object by using the command:

```
DETAIL level
```

The detail levels lie in a range of 0 through 255. All objects created after the `DETAIL` command has been issued are given the specified detail level until you choose a new level. If you do not use the `DETAIL` command all the objects are created on the default level of 0.

As an example of the `DETAIL` command, the following sequence of commands show how contour lines of different heights can be given different detail levels:

```
DETAIL 4  
CONTOURLINE 4  
DETAIL 6  
CONTOURLINE 6  
DETAIL 8  
CONTOURLINE 8
```

This feature is very useful as it allows selected detail levels to be viewed using the Model Viewer. For example, the following Viewer commands can be used to switch selected detail levels ON or OFF:

```
DET detail_level ON  
DET detail_level OFF
```

Manipulating the Model

You can perform the following operations on the completed model:

- Model validation
- Analysis of mass properties
- 3D Boolean operations and instancing

Validating the Model

You can access the Model Validator from the Utility Control Program. The Model Validator verifies the model or part of the model to determine whether it is valid. On a valid model, all edges of tiles and boundaries meet exactly and there are no spurious gaps or holes. For more details on the Model Validator, see chapter [“The Model Validator” on page 361](#).

Analyzing the Mass Properties of the Model

The Mass Properties program is also entered from the Utility Control Program and can extract the following properties of a model:

- Volume
- Surface area
- Mass
- Center of gravity
- Moment of inertia about each major axis
- Radius of gyration about each major axis
- Length of wire line objects

An analysis of the properties of a model can be very useful. For example, in the construction industry the volume of excavation or fill material required can be calculated.

Full details of how to enter and run the Mass Properties program are described in [“Extracting Mass Properties” on page 367](#).

Applying 3D Operations to the Model

Boolean operations can be performed on the model. For example, Boolean subtraction can be used to remove a volume (representing a volume of earth, perhaps) and the remaining model can be analyzed and viewed. Using Boolean addition and intersection, features can be added to the model landscape for analysis and visualization.

Complex models can be generated using 3D techniques such as instancing. Instancing allows you to place models from different files onto one 3D sheet using an instancing symbol. In this way you can build up complex assemblies and exploded views.

Details of Boolean operations and instancing are described in [“Boolean Operations” on page 209](#).

Terrain Modeler Commands

This section lists each of the commands that are available in the Terrain Modeler. The commands are presented in alphabetical order.

BASE

| → **BASE** → *<height>* →

Sets the height of the base of the terrain model in model units. The default height is 0. This command is useful when the coordinate data file contains negative Z-values. The Terrain Modeler does not produce a model if the base height is above the lowest Z-coordinate.

Specify *height* as a real number.

CONTOURLINE

| → **CONTOURLINE** → *<height>* →

Creates a contour line on the surface of the model at the given height.

Specify *height* as a real number.

CONTOURS

| → **CONTOURS** → *<number_of_contours>* →

Creates the specified number of contour lines on the model surface. The contours are drawn as wirelines at equal distances between the minimum and maximum height of the model.

Specify *number_of_contours* as an integer.

DETAIL

| → **DETAIL** → *<level>* →

Sets the detail level for all subsequent objects. The default value is 0.

Specify *level* as an integer in the range 0 through 255.

GO

| → **GO** →

Processes the data using the current parameters and produces a model of the terrain in the specified file.

GRID

| → **GRID** → *<square_size>* →

Superimposes a wireline grid on the surface of the model. The square size is specified in model units.

Specify *square_size* as a real number.

HELP

| → **HELP** →

Displays a list of the available Terrain Modeler commands.

IN

| → **IN** → *<filename>* →

Specifies the name of the file that contains the coordinate data used by the Terrain Modeler to create a model. If the file is not in the current directory, the full file specification must be given.

OBJECT NAME

| → **OBJECT NAME** → *<name>* →

Gives a name to all subsequent objects, until you specify a new object name.

OUT

| → **OUT** → *<filename>* →

Names and opens a model file in which the terrain model can be stored. If a model file is already open, it is closed prior to the new model file being opened.

QUIT

| → **QUIT** →

Leaves the Terrain Modeler and returns you to the Utility Control program.

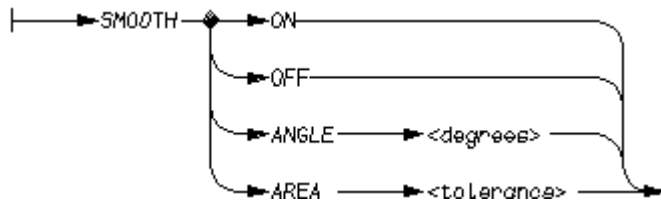
SMALLEST ANGLE

| → **SMALLEST ANGLE** → *<degrees>* →

Specifies the smallest angle to be created within a triangle during the triangulation process. If the data produces triangles containing angles less than the specified angle, a model is not generated. The default smallest angle is 5 degrees.

Specify *degrees* as a real number.

SMOOTH



Option	Description
ON	Turns ON the surface smoothing of the model.
OFF	Turns OFF the surface smoothing. It is the default.
ANGLE <i>degrees</i>	Sets the smallest angle to be created within a triangle during the smoothing process. During smoothing the triangles are subdivided as many times as possible without making any angle smaller than the specified angle. This angle cannot be set to a smaller value than the smallest angle used in triangulation (specified by the SMALLEST ANGLE command). The default smallest angle for smoothing is 15 degrees. Where <i>degrees</i> is a real number.
AREA <i>tolerance</i>	Sets the triangular area tolerance for smoothing. When the Terrain Modeler smooths the surface it interpolates between the existing points to create more data points. These additional data points are used by the algorithm to create more triangles. The smoothing stops when the area of the new triangles is smaller than that specified by the area tolerance. The default value is 10 square model units. Where <i>tolerance</i> is a positive real number.

SPOTHEIGHTS

|--> SPOTHEIGHTS -->

Draws vertical wirelines from the base to the surface of the model at points on the surface that correspond to the coordinates in the data file.

TILINV

|--> TILINV -->

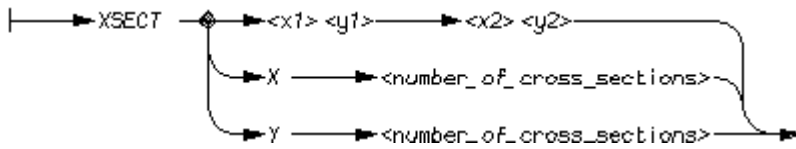
This command sets the internal tile edges that are created as Tile and with this invisible in MPDS4. This is the default.

TILVIS

|--> TILVIS -->

This command sets the internal tile edges that are created as Visible and with this visible in MPDS4.

XSECT



Creates a specified number of cross-sections.

Option	Description
x1 y1 x2 y2	Creates a cross-section starting at the point specified by the coordinates <i>x1 y1</i> and ending at the point specified by <i>x2 y2</i> . Where the coordinates are real numbers.
XSECT X	Creates the specified number of cross-sections, parallel to the X-axis, across the model surface. Where <i>number_of_cross-sections</i> is an integer.
XSECT Y	Creates the specified number of cross-sections parallel to the Y-axis across the model surface. Where <i>number_of_cross-sections</i> is an integer.

TEXT DRIVEN MODELING

The Interpolator program generates a model file from a text file (called a macro file) containing surface coordinate points. Two types of generator are available within this program: Sweeps and Meshed Surfaces. Sweeps are described in chapter “Sweeps” on page 69 and Meshed Surfaces in chapter “Meshed Surfaces” on page 191.

The output from the Interpolator program is a model file or files, each containing one or more objects that can be viewed interactively or reconstructed onto a MEDUSA4 sheet.

The Interpolator can be only accessed from MEDUTIL.

- [Running the Interpolator](#) 494
- [Using the Interpolator](#)..... 495
- [Sweep Models](#) 496
- [Sweep Examples](#) 500
- [Meshed Surface Models](#) 504
- [Meshed Surface Examples](#)..... 508
- [Interpolator Commands](#) 512

Running the Interpolator

The program can be accessed from outside MEDUSA4 using the MEDUSA4 Utility Access facility, MEDUTIL by means of the `INTERPOLATOR` command.

WARNING: Do not try to access MEDUTIL from within MEDUSA4 by entering:
`MEDUTIL`
This command starts too many processes and generates an error message.

Start MEDUTIL by typing the `medutil` command.


```
MEDUSA4 Utility Control
~~~~~
Type 'help' for list of commands...
Enter command> interpolator
MEDUSA4 Interpolator (Text Driven Modeler)
~~~~~
Type HELP for a list of Interpolator commands
Interpolator>
```


Interpolator commands can be entered at the `Interpolator>` prompt. The procedure to follow is described in [“Using the Interpolator” on page 495](#). Descriptions of the available commands are in section [“Interpolator Commands” on page 512](#).

Using the Interpolator

The following procedure gives a summary of how to use the Interpolator.

This procedure assumes that you are using a macro file for the data. You can also enter Interpolator commands interactively, but you are unlikely to want to do this except for extremely simple models.

1. Create a macro file containing Interpolator commands. A macro file is a text file that contains commands and can be created like any other text file using a text editor in the operating system. Use the Interpolator command `OUT filename` in the macro file to name the output model file.
2. Start the Interpolator from MEDUTIL.
3. Run your macro file using the command:
`MACRO filename`
Run other macro files as required.
4. Leave the Interpolator by entering:
`QUIT`
5. If you are in MEDUTIL, enter `QUIT` to leave, and then start MEDUSA4.
6. Select a blank 3D sheet and give it the same name as the created model file, for example, `EXAMPLE1`.
7. Run the 3D Sheet Viewer to view the model by using the View tool .

Please note: Do not use the Model + Reconstruct tool . This would create a model from the blank sheet and therefore overwrite the model file created by the Interpolator.

Sweep Models

This section describes how to create a sweep model from a macro file containing coordinates of points on a profile line defined in a plane. This is the sequence of operations:

1. Specify an output model file.
2. Enter object data.
3. Define the sweep profile.
4. Specify the sweep limits.
5. Smooth the profile.
6. Specify hole profiles.
7. Specify the chord tolerance.
8. Generate the model.

Specifying a Model File

Open a model file using the command:

```
OUT filename
```

where *filename* is the name of the MEDUSA4 model file to be created by the Interpolator. If a model file with the same name is already open, it is closed and a new file opened with the specified name. The Interpolator overwrites the existing file.

Entering Object Data

The first command to follow the `OUT` command is:

```
SWE object_name
```

where *object_name* is the name of the first object to be generated. Several objects may be generated, one after the other, as part of the same model.

The data for each object must be ended with the command:

```
ENDOBJ
```

The `ENDOBJ` command can be followed by another `SWE` command if another object is to be generated in the same model file.

Defining the Profile

The profile of the sweep is defined using the command:

```
PRO
```

This command is followed by the coordinates of points in the profile line. For the first point, all three coordinates should be given. If only X and Y-coordinates are specified, the Z-coordinate defaults to 0. For subsequent points, only the X and Y-coordinates need to be given if the Z-coordinate is the same. The macro file given in the example see [“Sweep Examples” on page 500](#) shows how to separate the coordinate points in the file.

Specifying the Sweep Limits

The distance through which the profile is to be swept is defined by the commands:

```
FRO x y z
```

```
TO x y z
```

The three coordinates following `FRO` define the start point for the sweep. The origin of the profile is moved to this point prior to the sweep. The three coordinates following `TO` define the end point of the sweep, that is, the point to which the origin of the profile is swept.

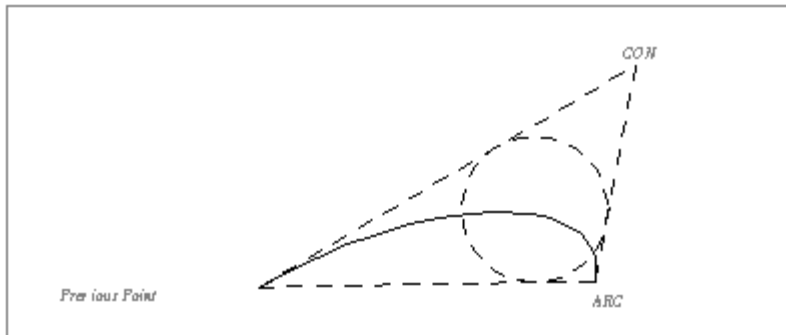
Smoothing the Profile

Following the coordinates of each point, a line function can be given to modify the profile as follows:

Command	Description
SMO	Smooths the profile line around the point, using conic segment approximation to a cubic spline.
LIN	Draws a straight line from the previous point.
DIS	Specifies a discontinuity in curvature at the point.
CON and ARC	These line functions must be given in pairs, <code>CON</code> followed by <code>ARC</code> . The <code>CON</code> function specifies that the profile should be modified so that a tangent point elliptical arc is drawn from the previous point (which can have any line function) to the next point (which must have an <code>ARC</code> function). The arc will pass through the incenter of the triangle formed by the previous point, the <code>CON</code> point and the <code>ARC</code> point. This is shown in Figure 319 below.

The default line function is DIS.

Figure 319 The CON and ARC Line Functions



The following example shows part of an input file that specifies different line functions to modify a profile. Note that point 1, point 2, .. is given in place of actual coordinates.

```
point 1
point 2    SMO
point 3    SMO
point 4
point 5    SMO
point 6    SMO
point 7    LIN
.         .
.         .
.         .
```

This sequence of commands produces a profile line smoothed from point 1 through point 4. At point 4 there is a discontinuity, and the profile is smoothed from point 4 through point 6. There is a straight line from point 6 through point 7. Note that if the first and last points in a profile are given SMO functions, the profile will be smoothed between them.

Specifying Hole Profiles

A hole can be introduced into a profile by the command:

```
HOL
```

This command is followed by the coordinates of points on the hole profile. Where required, the hole profile can be modified by line functions in the same way as an outer profile.

Several holes can be specified, each introduced by the `HOL` command.

Specifying the Chord Tolerance

The chord tolerance is specified using the command:

```
TOL tolerance
```

where *tolerance* is the maximum deviation between a true arc and the tile edges created by the Modeler to represent an arc.

Generating the Model

The model is generated by the command:

```
QUIT
```

This command is given after all other data has been entered.

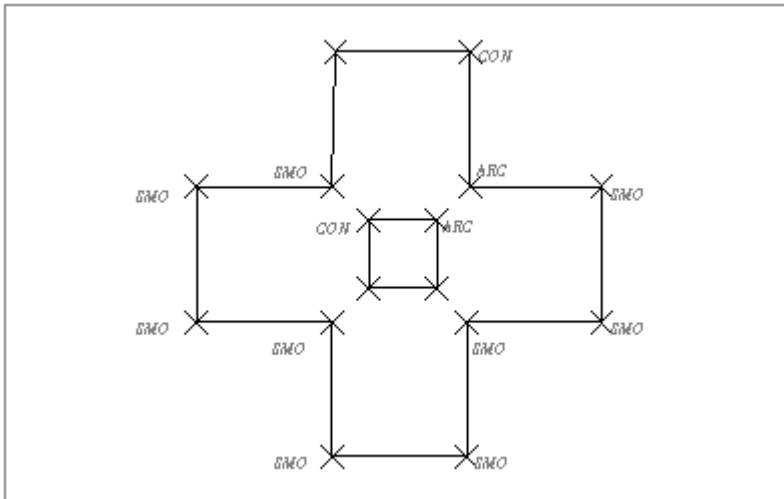
Sweep Examples

This section contains two examples of the macro files used to generate sweep models in the Interpolator.

Example 1

The following is a simple example of a model generated by an Interpolator `SWE` command. [Figure 320](#) shows the positions of the coordinate points and the line functions associated with them. [Figure 321](#) shows the effect of the line functions on the profile.

Figure 320 Unmodified Profile Labelled with Line Functions



The contents of the macro file are shown below:

```
OUT SWEEP1
SWE EXAMPLE1
PRO100 200 100 SMO
100 300 SMO200 300 SMO
200 400
300 400 CON
300 300 ARC
400 300 SMO
400 200 SMO
300 200 SMO
300 100 SMO
200 100 SMO
200 200 SMO
HOL
225 225
225 275 CON
```

```

275 275 ARC
275 225
FRO 0 0 0
TO 0 100 100
END
OBJ
QUIT

```

Figure 321 Profile Modified by Line Functions

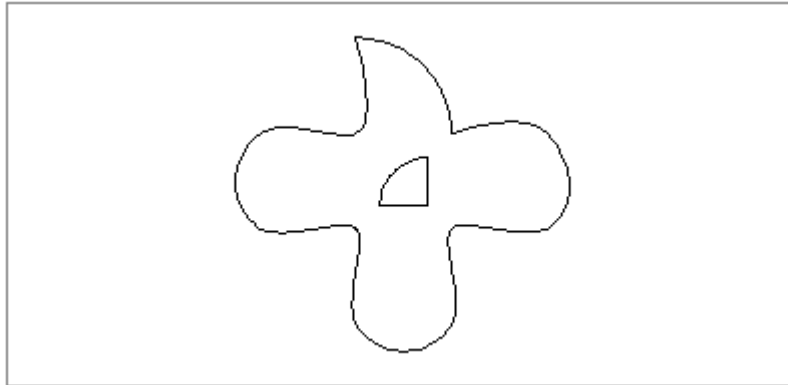
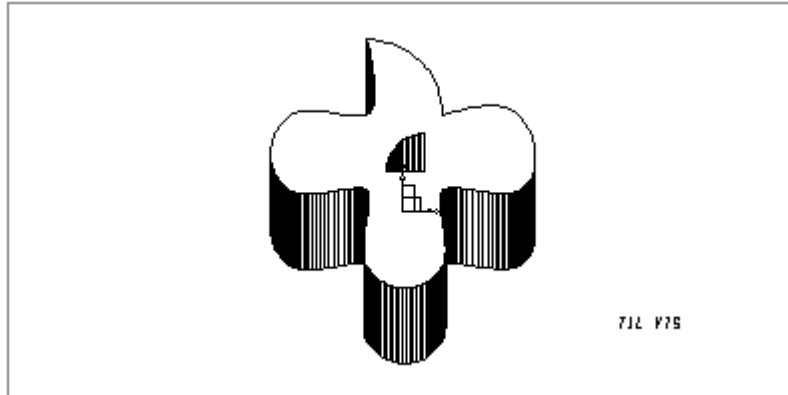


Figure 322 shows a view of the completed model.

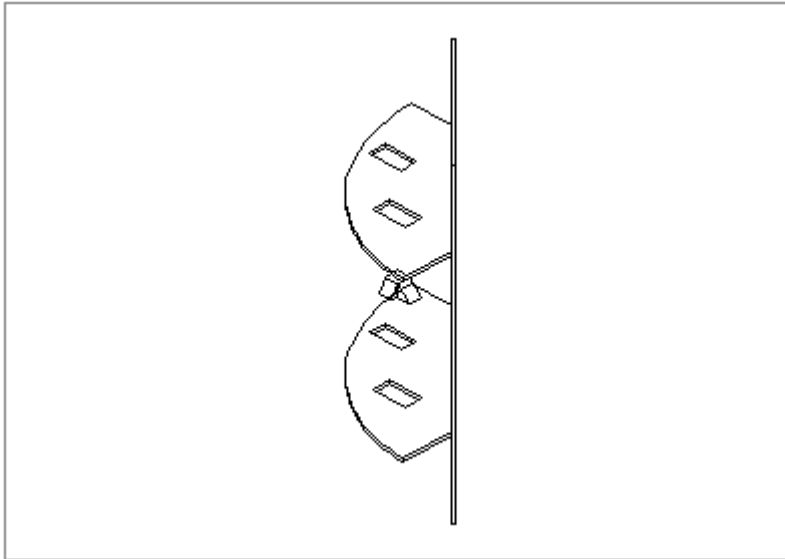
Figure 322 The Completed Model



Example 2

Several objects can be combined into one model as shown in [Figure 323](#).

Figure 323 A Model Comprising Several Objects



The macro file used to generate this model is shown below:

```
OUT SWEEP2
SWE PLATE01
PRO
0 0 LIN
70 70 LIN
100 70 LIN
100 -25 CON
0 -25 ARC
HOL
31 9 LIN
31 -11 LIN
41 -11 LIN
41 9 LIN
HOL
61 42 LIN
61 22 LIN
71 22 LIN
71 42 LIN
FRO 0 0 0
TO 0 0 2
ENDOBJ
SWE PLATE02
PRO
0 0 LIN
70 70 LIN
100 70 LIN
100 -25 CON
0 -25 ARC
HOL
```

```
31 9 LIN
31 -11 LIN
41 -11 LIN
41 9 LIN
HOL
61 42 LIN
61 22 LIN
71 22 LIN
71 42 LIN
FRO 0 0 100
TO 0 0 102
ENDOBJ
SWE PLATE03
PRO
0 0 LIN
70 70 LIN
68.586 71.414 LIN
-1.414 1.414 LIN
FRO 0 0 -50
TO 0 0 150
ENDOBJ
QUIT
```

Meshed Surface Models

This section describes how to generate Meshed Surface pot models from a macro file containing coordinates of points on the surface of the model. The surface is defined by lines of latitude and longitude. For an example of a Meshed Surface pot model, see [Figure 324](#).

This is the sequence of operations:

1. Specify an output model file.
2. Enter object data.
3. Specify latitudes.
4. Specify longitudes.
5. Specify the model type (optional).
6. Set the facet spacing (optional).
7. Specify the output of Coons patches (optional).
8. Specify the detail level (optional).
9. Smooth the profile (optional).
10. Generate the model.

Specifying a Model File

Open a model file using the command:

```
OUT filename
```

where *filename* is the name of the MEDUSA4 model file to be created by the Interpolator. If a model file is already open, it is closed and a new file opened with the specified name. The Interpolator overwrites the existing file.

Entering Object Data

The object data is introduced by the command:

```
MESH object name
```

where *object_name* is the name of the object. The end of the data is indicated by the command:

```
ENDOBJ
```

Any number of objects can be created in a file using the sequence of commands `MESH` and `ENDOBJ`.

Specifying Latitudes

The `MESH` command is followed by coordinates of points on latitudes. The start of each latitude is indicated by the command:

```
LAT
```

and the coordinates of each point are then entered in order. For the first point on each latitude, all three coordinates should be given. If only X and Y-coordinates are specified, the Z-coordinate defaults to 0. The Z-coordinates of subsequent points can be omitted if they are the same as that of the first point.

Specifying Longitudes

The longitudes can be specified in one of two ways. In both cases, the start of each longitude is indicated by the command:

```
LON
```

The first method gives the actual coordinates of each point in order. Three coordinates should be given for the first point on each longitude. If only X and Y-coordinates are specified, the Z-coordinate defaults to 0. The Z-coordinate of subsequent points may be omitted if they are all the same as that of the first point.

The second method uses the command:

```
REF latitude_number point_number
```

to reference a point on a latitude as the position of the longitude. Each latitude is considered to be numbered, that is, the first latitude entered is number 1, and so on. Each point on a latitude is similarly given a number. The points are then referred to by giving their line and point numbers. For example:

```
REF 3 2
```

means the longitude intersects with the second point on latitude number 3.

Please note: The intersections of latitudes and longitudes must be specified points on both lines, and the second method is a convenient way of ensuring that this rule is obeyed.

Rules for Specifying Latitudes and Longitudes

These are the rules for entering latitudes and longitudes:

- Latitudes and longitudes can go in any direction. The exception to this rule is if there are only two longitudes, in which case all the latitudes must run in the same direction, that is, all either clockwise or counterclockwise.
- Each latitude must intersect with each longitude at a point that is a specified point on both the latitude and the longitude.
- Latitudes must not cross other latitudes and longitudes must not cross other longitudes.
- Each longitude must start on the first latitude and end on the last latitude.
- The first or last latitude can be flat or curved. However, if it is both curved and has a concave boundary, poor results may be obtained for the end surface.
- The curve smoothing algorithms for latitudes and longitudes can oscillate if a sudden change of trend occurs. For example, this will occur if there are several points in a straight line with one point displaced.

Specifying the Model Type of an Object

The objects can be created as wire, shell, or solid models using the `CAT` command followed by `WIRE`, `SHELL`, or `SOLID` respectively:

Option	Description
<code>CAT WIRE</code>	The specified latitudes and longitudes are generated as wire lines.
<code>CAT SHELL</code>	A surface is generated passing through the specified latitudes and longitudes.
<code>CAT SOLID</code>	A volume contained by a surface is generated, passing through the latitudes and longitudes.

Setting the Facet Spacing

For shells and solids, the facet spacing can be set by the command:

```
FACET value1 value2
```

where *value1* specifies the number of facets between pairs of latitudes and *value2* specifies the number of facets between longitudes. The default is `FACET 4 4`.

Specifying Coons Patch Output

Coons patches can be sent to the model file in place of facets by specifying the command:

```
EQUATIONS ON
```

The default is `EQUATIONS OFF`.

Specifying the Detail Level

The detail level can be specified using the command:

```
DET level
```

The default detail level is 0.

Smoothing the Profile

This is done using the smoothing commands as for sweep models. See [“Smoothing the Profile” on page 497](#) for details.

Generating the Model

The model is generated by the command:

```
QUIT
```

This command is given after all other data has been entered.

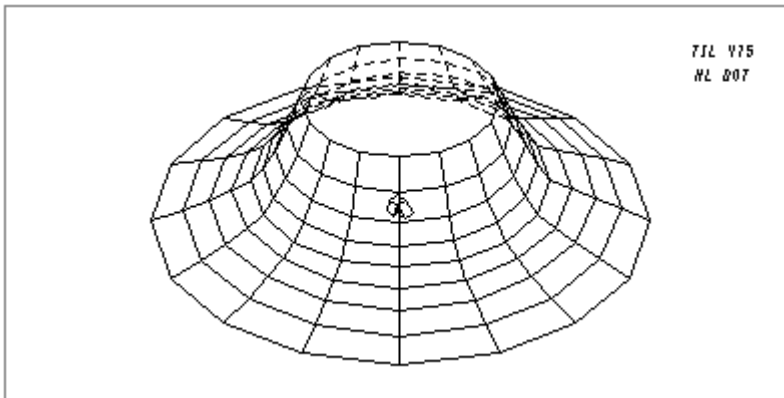
Meshed Surface Examples

This subsection contains two examples of macro files used to generate Meshed Surface pot models in the Interpolator.

Example 1

The following is an example of how to model a simple Meshed Surface-type object. [Figure 324](#) shows a view of the completed model.

Figure 324 The Completed Model



The contents of the macro file are shown below.

```
OUT Meshed_Surface1
MESH Meshed_Surface1
LAT
150 250 150 SMO
250 350 SMO
350 250 SMO
250 150 SMO
LAT
100 250 50 SMO
250 400 SMO
400 250 SMO
250 100 SMO
LAT
0 250 0 SMO
250 500 SMO
500 250 SMO
250 0 SMO
LON
REF 1 1
REF 2 1
```

```

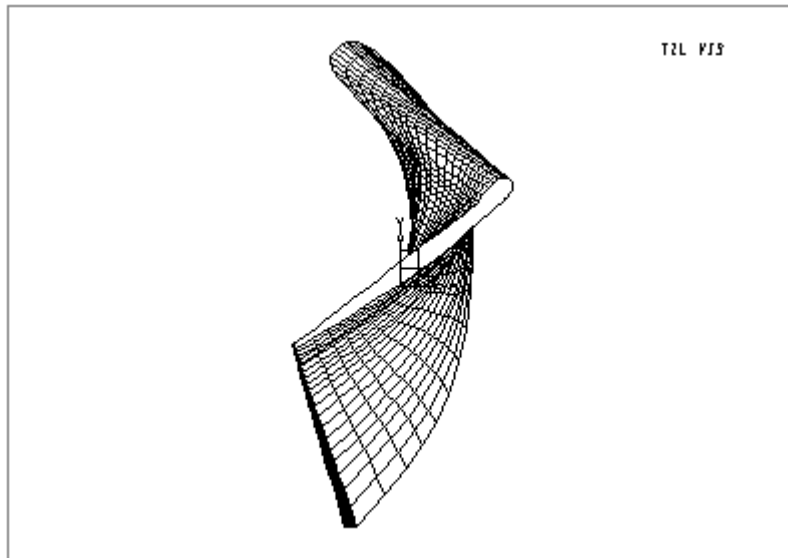
REF 3 1
  LON
  REF 1 2
  REF 2 2 SMO
  REF 3 2
  LON
  REF 1 3
  REF 2 3 SMO
  REF 3 3
  LON
  REF 1 4
  REF 2 4 SMO
  REF 3 4
ENDOBJ
QUIT

```

Example 2

The following example shows how to generate a model of a fan blade [Figure 325](#) shows a view of the completed model.

Figure 325 The Completed Model of a Fan Blade



The contents of the macro file are shown below:

```

OUT Meshed_Surface2
CAT SOLID FACET 5 15
MESH Meshed_Surface2
LAT
019.22 032.71 1000.00

```

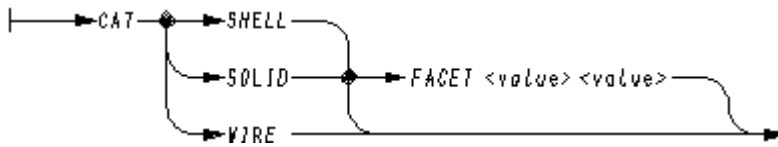
085.08	169.10	1000.00	SMO
109.63	318.40	1000.00	SMO
110.37	469.67	1000.00	SMO
048.24	611.09	1000.00	SMO
000.03	682.90	1000.00	SMO
013.90	699.48	1000.00	SMO
099.20	653.95	1000.00	SMO
185.46	506.41	1000.00	SMO
190.68	333.61	1000.00	SMO
140.79	168.85	1000.00	SMO
035.49	031.44	1000.00	
LAT			
-012.61	139.34	1900.00	
050.67	233.84	1900.00	SMO
095.47	338.18	1900.00	SMO
112.93	450.36	1900.00	SMO
103.85	563.21	1900.00	SMO
081.46	624.11	1900.00	SMO
093.56	634.87	1900.00	SMO
149.41	589.69	1900.00	SMO
187.39	465.58	1900.00	SMO
163.44	338.44	1900.00	SMO
090.91	229.98	1900.00	SMO
-001.45	137.44	1900.00	
LAT			
-034.43	214.05	2800.00	
027.99	281.99	2800.00	SMO
083.42	355.68	2800.00	SMO
125.69	437.57	2800.00	SMO
147.24	527.03	2800.00	SMO
151.35	579.19	2800.00	SMO
163.88	581.14	2800.00	SMO
182.91	528.06	2800.00	SMO
177.65	428.62	2800.00	SMO
122.94	344.66	2800.00	SMO
050.99	274.35	2800.00	SMO
-027.55	211.36	2800.00	
LAT			
-046.43	257.56	3700.00	
016.78	312.53	3700.00	SMO
078.80	368.83	3700.00	SMO
136.22	429.75	3700.00	SMO
183.73	498.68	3700.00	SMO
208.95	539.06	3700.00	SMO
219.94	535.38	3700.00	SMO
211.00	487.11	3700.00	SMO
165.53	413.13	3700.00	SMO
100.69	354.74	3700.00	SMO

```
030.63 302.66 3700.00 SMO
-041.70 253.74 3700.00
LAT
-052.97 283.99 4600.00
012.35 333.56 4600.00 SMO
077.23 383.70 4600.00 SMO
141.46 434.69 4600.00 SMO
205.22 486.23 4600.00 SMO
241.62 515.20 4600.00 SMO
250.07 508.26 4600.00 SMO
222.47 470.50 4600.00 SMO
160.07 415.42 4600.00 SMO
092.57 366.70 4600.00 SMO
022.28 322.05 4600.00 SMO
-049.43 279.72 4600.00
LON
REF 1 1
REF 2 1 SMO
REF 3 1 SMO
REF 4 1 SMO
REF 5 1
LON
REF 1 4REF 2 4 SMO
REF 3 4 SMO
REF 4 4 SMO
REF 5 4
LON
REF 1 6REF 2 6 SMO
REF 3 6 SMO
REF 4 6 SMO
REF 5 6
LON
REF 1 8REF 2 8 SMO
REF 3 8 SMO
REF 4 8 SMO
REF 5 8
LON
REF 1 12REF 2 12 SMO
REF 3 12 SMO
REF 4 12 SMO
REF 5 12
ENDOBJ
QUIT
```

Interpolator Commands

This section lists the commands that are available in the Interpolator. The syntax of each command is given as a syntax graph. The commands are listed in alphabetical order.

CAT



Specifies whether objects are to be generated as shell, solid, or wire models.

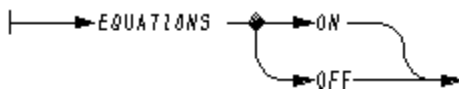
In the case of shells and solids, facet spacing can be specified. The first value is the number of facets between each pair of latitudes, the second value is the number of facets between each pair of longitudes. The default facetting is `FACET 4 4`.

DET



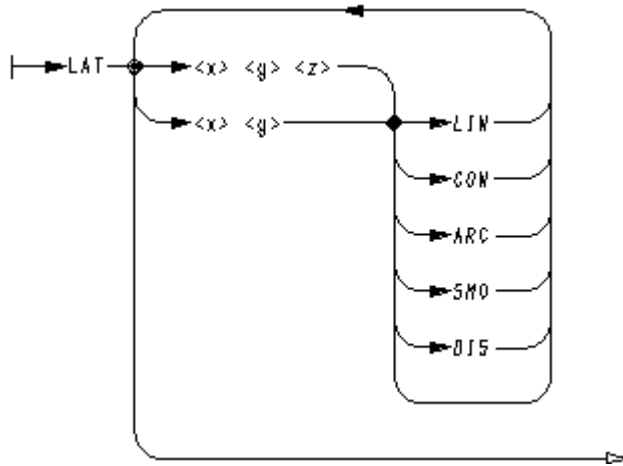
Specifies the detail level of the generated object.

EQUATIONS



Controls the output of Coons patches to the model file. If ON, Coons patches are output to the model file in place of facets.

LAT

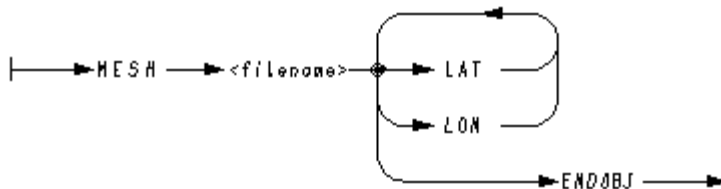


Creates a latitude. For the first point on each latitude, all three coordinates should be given. If only X and Y-coordinates are specified, the Z-coordinate defaults to 0. However, on subsequent points only two values are required.

For latitudes the line is closed, that is, a line section is inserted between the last point and the first point.

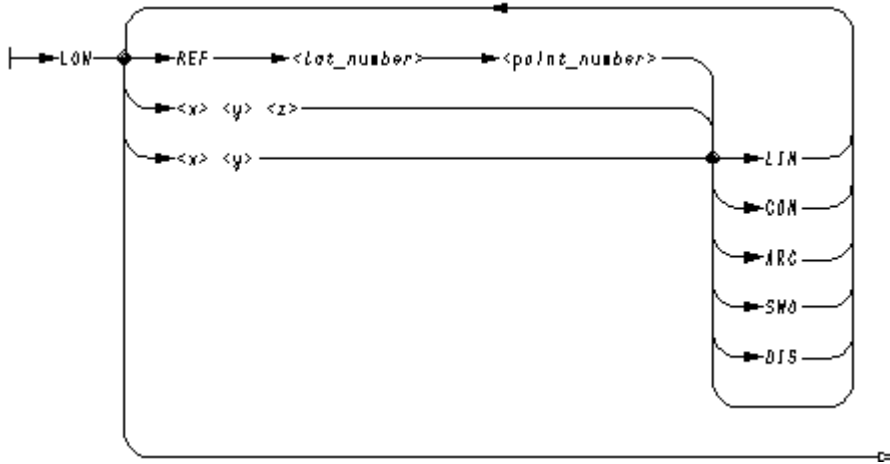
Following the coordinates of each point, a line function may be given to modify the profile. If none is specified, the default is DIS, giving a discontinuity at that point. The functions available are LIN, CON, ARC, SMO and DIS, as described in [“Smoothing the Profile” on page 497](#).

MESH



Defines the name of the object and specifies the coordinates of latitudes and longitudes. Details specific to latitudes and longitudes are given in the LAT and LON commands respectively. The end of the Meshed Surface data is indicated by the command ENDOBJ.

LON



Creates a longitude, that is an open line running from the first point specified to the last point. LON must be used only after all latitudes for an object have been defined.

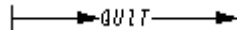
Option	Description
REF <i>lat_number point_number</i>	Specifies the coordinates of each point on a longitude by reference to a previously given point in a latitude where <i>lat_number</i> gives the number of the latitude (in order of input for each object) and <i>point_number</i> gives the number of the point (in order of input for each latitude).
x y z	Specifies the coordinates of each point on a longitude.
x y LIN	Modifies longitudes by means of line functions given after the coordinates. The functions available are LIN, CON, ARC, SMO and DIS, as described in “Smoothing the Profile” on page 497 . The default is DIS, a discontinuity at the point.

OUT

→OUT →<filename> →

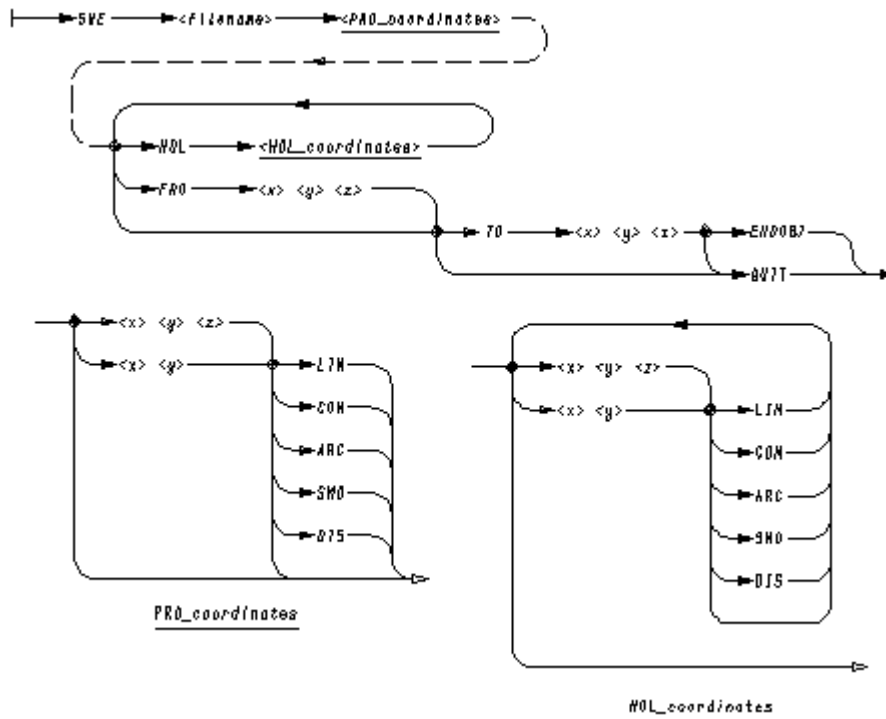
Opens a model file for the model generated by the Interpolator. If a model file with the same name is already open, it is closed and a new file opened with the specified name. The Interpolator overwrites the existing file.

QUIT



Starts processing and generates a model file from the coordinate data. If no model file is open, QUIT leaves the Interpolator program.

SWE



Generates a model by means of sweeping a defined profile line between two points. The file-name names the file that will become a MEDUSA4 model file provided that an OUT command has not been given. The profile is introduced by the PRO command followed by the coordinates of points on the profile.

The HOL command may be used to define a hole in the outer profile. It is followed by coordinates of points on the hole profile. Several holes may be defined, each introduced by a HOL command.

FRO and TO, each followed by three coordinates, are used to specify the points between which the profile is to be swept. ENDOBJ indicates the end of the data for each object.

TOL

| → *TOL* → *<tolerance>* →

Sets the chord tolerance for models generated by the `SWE` command. The value specified by *tolerance* represents the maximum deviation between a true arc and the tile edges created by the Modeler to represent an arc.

MODEL FILE COMPRESSION

This chapter describes the MEDUTIL utility MODCOMP which you can use to compress or expand model files independently of the Modeler. This utility is useful if you prefer to handle model file compression and expansion as a separate task from modeling.

Please note: Model file compression can be suppressed at the modeling stage by using the `COMPRESSION OFF` command on the 3D definition sheet (see [“Compressing a Model File”](#) on page 292 for details).

The MODCOMP utility can also be used to convert model files to an interchange format.

- [Running MODCOMP](#) 518
- [Using MODCOMP](#) 518
- [MODCOMP Commands](#) 520

Running MODCOMP

The program is entered from the MODCOMP command in MEDUTIL, the MEDUSA4 Utility Access facility. To start MEDUTIL type the `medutil` command.

```
MEDUSA4 Utility Control
~~~~~
Type 'help' for list of commands...
Enter command>modcomp
*macro D:\medusa4\med3d\m3d\macro\modcomp.mac

MEDUSA4 Model File Compressor/Expander
~~~~~
Type HELP for a list of MODCOMP commands
Modcomp>
```

MODCOMP commands can be entered at the `Modcomp>` prompt.

Using MODCOMP

You can use the MODCOMP facility to upgrade your model files to a MEDUSA4 Revision 12.2 portable format. Model files in this portable format can then be transferred to any other platform by the data transfer facility which is available. To convert a model file to a portable format, follow this procedure:

1. At the `Modcomp>` prompt, enter the name of the model file with the command:

```
IN <file>
```

A message is displayed that specifies which platform the model file is from. See the example on [page 519](#).

2. Specify an output filename, using the command:

```
OUT <file>
```

3. Start the conversion process with the command:

```
GO
```

You can repeat this sequence of commands (`IN`, `OUT`, `GO`) as many times as required for different model files, or to direct output to different destinations.

4. To leave MODCOMP, enter the command:

```
QUIT
```

The commands given in this section can be entered individually from the keyboard, or by executing a macro file.

Example

An example of using MODCOMP is shown below. The input information is shown in boldface type.

```
MEDUSA4 Model File Compressor/Expander
~~~~~
*macro D:\medusa4\med3d\m3d\macro\modcomp.mac
Type HELP for a list of MODCOMP commands

Modcomp>in block.mod
Modcomp>The file is from the current platform
Modcomp>out block.mod
Modcomp>go
Modcomp>quit
```

MODCOMP Commands

Commands may be typed in uppercase or lowercase. They are shown in uppercase for clarity. The commands in this section are presented in syntax graph form.

COMPRESS

|—> COMPRESS —>

Specifies that the model file is to be compressed.

EXPAND

|—> EXPAND —>

Specifies that the model file is to be expanded.

GO

|—> GO —>

Starts the process.

HELP

|—> HELP —>

Lists the MODCOMP commands available.

IN

|—> IN <filename> —>

Specifies the name of the input model file. Specify *filename* as a text string.

OUT

|—> OUT <filename> —>

Specifies the name of the output file. Specify *filename* as a text string.

QUIT

|—> QUIT —>

Leaves MODCOMP and returns to MEDUTIL.

ERROR MESSAGES

This appendix lists the error messages which you may encounter while using the system. All entries are listed in alphabetical order within each section.

- [Modeler Error Messages..... 522](#)
- [Viewer Error Messages..... 536](#)
- [Cross-section Views Error Messages 539](#)
- [Model Validator Error Messages 541](#)
- [Shrinker Error Messages 543](#)
- [MODASC Error Messages..... 544](#)
- [MODSMO Error Messages..... 545](#)
- [Model File Translator Error Messages 546](#)
- [Terrain Modeler Error Messages..... 547](#)
- [Text Driven Modeling Error Messages 549](#)
- [Model File Compression Error Messages..... 550](#)

Modeler Error Messages

0 or 2 profile points on centerline expected

Two profile points are expected on the centerline in the definition of a volume of revolution model. Use segment probes to attach the profile to the centerline.

3D edge text ambiguously positioned

A text is placed at a position that the Modeler cannot evaluate, for example, where two edges cross.

3D text not hitting edge

Attach the 3D text to the edge using a segment probe.

3D text not hitting face

Attach the 3D text to a face using a segment probe.

3D text not hitting vertex

Attach the 3D text to the vertex using a segment probe.

3D text outside viewbox

Some 3D texts, for example Viewer commands such as FIT, should be enclosed within a viewbox. Move the appropriate text inside a viewbox.

Ambiguous implied rotation for profile

A profile for a duct is defined in such a way that its final position in a completed model is ambiguous.

Ambiguous projection point, try smaller CHOTOL or cleaner profiles

The primary profile projects more than once onto the projection surface. Try using a smaller value for CHOTOL or cleaner profiles.

At least two outer profiles required

Two profiles are required for the generation of certain models, for example, ruled surfaces.

Attempt to use instance in meshed surface

The name in the instance viewprim is the same as that used to name a latitude or longitude in a meshed surface model in the sheet.

Bad angle for volume of revolution

The angle specified for a partial volume of revolution must lie within the range 0 through 360 degrees.

Bad instance model file name

Specify a valid model filename in the instance group.

Bad line: all segments invisible

A maximum of two invisible segments are allowed in wire profiles.

Bad line: more than two invisible segments

A maximum of two invisible segments are allowed in wire profiles.

Bad line: too many invisible segments

A maximum of two invisible segments are allowed in wireline profiles.

Bad line: too many points

There are too many points on the longitudes or latitudes of a duct or meshed surface model. A maximum of 500 points is allowed to surround any one Coons patch.

Bad number of depths

There are too few or too many depth point functions on the indicated link line.

Bad number of holes or profiles

Certain models require a corresponding number of holes and profiles, for example, ruled surfaces.

Bad number of implied longitudes

There must be between 2 and 100 matching points in a duct definition.

Bad number of profiles

There must be at least two profiles to define certain models, for example, ruled surfaces.

Bad number of profiles in face

There is a maximum of 100 holes allowed in a sweep model.

Bad number of profiles picked up

The link line must pick up an outer profile, then any holes present, then the second outer profile and the same number of holes.

Bad number of rulings

Ruled surface models must have at least two sets of profiles.

Bad sequence of point functions

The link line must pick up an outer profile, then any holes present, then the second outer profile and a matching number of hole profiles.

Bad start point given for centerline

Profile and centerline start vectors are in the same plane. Change the start point of the centerline.

Bad value for chord tolerance ignored

The Modeler requires a positive value for CHOTOL. It ignores the value specified and reverts to the default.

Bad value for detail ignored

The specified detail level is ignored, as it lies outside the range 0 through 255.

Bad value for meshed surface facetting ignored

The Modeler requires a value that lies in the range 0 through 70 for the FACET command. It ignores the value specified and reverts to the default value of FACET 4 4.

Bad value for meshed surface tolerance ignored

The Modeler requires a positive value for the MESHTOL command. It ignores the value specified and reverts to the default of MESHTOL 1.

Bad value for multiplier ignored

Specify a positive multiplier for the CHOTOL command.

Cannot create model - no sheet text

There must be a TSH-text of in the sheet so that the model can be given a name

Cannot evaluate

The 3D sheet contains a Boolean expression that cannot be evaluated, for example, a named object may not exist.

Cannot open file

The file does not exist or is protected.

Cannot open instance model file

The instance model file does not exist or is protected.

Cannot open model file

The model file is corrupt or does not exist.

Centerline must be closed and flat for open profiles

If an open profile is swept around a centerline, the centerline must be closed and flat in order to generate a solid model.

Centerline must be open

The centerline must not be closed.

Datum not in view boundary

The datum of a viewprim or text must lie within a view boundary.

Detail level out of range

The detail levels must be specified within the range 0 through 255.

Depth point in perspective view

A viewbox containing a perspective view command cannot be used to define a view for modeling.

Depths define a bad model

The depth point functions on a link line define a model that is infinitely thin, for example, they are positioned at the same point.

Different number of depths and profiles

The number of depth points on the link line must correspond to the number of matched profiles.

Duplicate name in set

There are two object names in the same set.

Even no. of depths required

The depth points define the plane of the two outer profiles for each object.

Even no. of holes required

An even number of hole profiles must be present, defining the top and bottom profile of each hole.

Even number of points required for edge and solid links

Edge and solid sweep link lines must contain an even number of depth points. Expect 0 or 2 profile points on centerline

Expect 0 or 2 profile points on centerline

Two profile points are expected on the centerline in the definition of a volume of revolution model. Use segment probes to attach the profile line to the link line.

Extra bit of latitude illegal

A latitude must start and end on a matching point.

Extra bit of longitude illegal

A longitude must start and end on a matching point.

File has no model header

Each model file contains a header that identifies it as a model file. The specified file may not be a model file.

First point in link must be a profile/hole point

For ruled surfaces the direction of the link line must be from the profile definition to the depth definition.

Heap full

The system limit is exceeded.

Heap overflow

The system limit is exceeded.

Hole lies outside profile

A hole profile must be enclosed within an object profile.

Illegal expression

Verify any Boolean expressions in the 3D sheet.

Illegal use of identifier

A name is used both as the left side of a Boolean statement and as an object name, for example, MAKE A = A + B.

Incompatible profiles

Wire profiles must have the same number of points.

Incompatible viewboxes

The viewbox containing the profile(s) and the viewbox containing the depth points are defining the same or nearly parallel views.

Instance model name is invalid

Specify a valid name for the instance model filename in the instance group.

instance group must have a model text

The name of the model to be used as an instance must be entered into the instance group.

Instance set must have a model text

Use a SMO text in the instance set to specify the name of the model to be placed in the assembly.

Instance set without a datum

Each instance set must contain only one viewprim.

Internal buffer overflow

The system limit is exceeded.

Internal overflow

The system limit is exceeded.

Internal text buffer overflow

The text buffer is limited to 120 characters. The system limit is exceeded.

Invalid mesh

The definition for a meshed surface model is incorrect. Verify that the points on the latitudes and longitudes are matching and correctly aligned.

Invalid range specified for detail level

The range of specified detail levels includes levels that have not been assigned to objects in the drawing.

Invalid sequence of point functions

Verify the point functions on the indicated link line. For example, the first point function on a link line cannot be a depth point function.

Invisible segment not allowed

Invisible segments are not allowed in projection surface lines.

LAT or LON does not fit into mesh

All the latitudes must intersect all the longitudes in the same order.

Latitude number does not mesh with longitude number

The specified latitude does not intersect with the specified longitude. Verify that the points on the latitudes and longitudes are matching and correctly aligned.

Link point in perspective view

A viewbox containing a perspective view command may not be used to define a view for modeling.

Link point in wrong viewbox

All profiles that are attached to a sweep link line must be defined in the same viewbox.

Link point not in viewbox

All primary profiles must lie within the same viewbox.

Link point without a profile

The link point must be attached to the profile line using a segment probe.

Longitude has no invisible segment

Each longitude must have one invisible segment.

Longitude number does not mesh with latitude number

The specified longitude does not intersect with the specified latitude. Verify that the points on the latitudes and longitudes are matching and correctly aligned.

Maximum number of vertices exceeded

The model file generated is invalid. Reduce the number of vertices per hole by increasing the CHOTOL value.

Mismatch of profiles and centerline points

The number of points on a centerline must match the number of profiles when defining a duct model.

Model file not found

The model file does not exist in the current/specified directory.

More than one prim in set

Each instance set must contain only one viewprim.

More than two points in centerline

The centerline must not contain points between the end points.

No centerline defined

Verify the line type if the centerline appears to be defined in the sheet.

No datum in viewbox

Each viewbox must contain a viewprim.

No datum prim

A datum viewprim is required in the viewbox into which an instance is loaded.

No depth points in instance link

Depth points must be indicated by arrowhead point functions or depth texts on an instance link line.

No depth points in link

The link line must have depth point functions or depth texts to define the third dimension of the model.

No instance prim in set

Each instance set must contain a viewprim.

No link line found

The datum of text associated with a link line must be placed exactly on a link line using a segment probe.

No model file

No file has been specified for the Viewer.

No model file open

There is no model file open.

Non-matching invisible segment

An invisible segment in the secondary wire profile does not correspond with an invisible segment in the primary profile.

Non-matching invisible segment on latitude

The invisible segments on a latitude must match.

Non-matching invisible segment on longitude

The invisible segments on a longitude must match.

Non-unique projection

The primary profile projects more than once onto the projection surface.

No profile points in link

The link line must have appropriate point functions where it intersects the profile lines.

No projection

The primary profile cannot be projected onto the projection surface.

Not a model file

The specified file is not a model file.

Null segment

There are two coincident points in a profile line.

Object name too long

An object name is limited to 20 characters.

Only layers 0-127 available

Specify a layer in the range 0 through 127 for Reconstructor commands.

Only one centerline allowed

More than one point function 12 is present on a centerline.

Only one link line allowed in set

Each instance must have only one link line.

Only one point in profile

Profile lines must have at least two points.

Only one profile/link

Only one profile line and one link line is used to define a volume of revolution model.

Only one profile/sweep if holes around

Each link line defining a sweep can either attach to several outer profiles (and so each profile generates a separate object) or attach to one outer profile and a number of hole profiles (and so generate one object).

Only two points allowed in CIR/CEN

An extra point has been inserted during the generation of a circular profile.

Overlapping section ignored

The Modeler ignores overlapping partial volumes of revolution. For example, the following sequence of commands creates overlapping sections:

```
ANGLE 30 50
```

```
ANGLE 40 60
```

Pipe has negative radius

A negative value has been specified for the radius of an LPS line used to generate a pipe model. Specify a positive radius.

Problem opening model file

The model file may be corrupted, or access protected.

Problem reading instance data

The instance model does not exist (in the specified directory) or it is corrupted.

Problem reading model file

The model file may be corrupted, or access protected.

Problem writing model header

The file may be read-only, or there is a problem in opening the file.

Profile has zero area

A valid profile must have a positive area.

Profile mismatch

Line segments of profiles must match in a ruled surface definition.

Profile not whole

Only one invisible segment is allowed in a profile.

Profiles incompatible

The two outer profiles must have the same number of segments (a diamond point function FUNV 10 counts as a segment).

Profiles intersect centerline

The profile line can touch but must not cross the centerline. Use a segment probe to attach the profile to the centerline.

Profiles must be closed

Open profiles are not allowed.

Profiles must have equal numbers of matching points

The profiles of ruled surface or duct models must contain equal numbers of matching points.

Sweep depths must come in pairs

Two depth point functions are required to define a sweep model.

Syntax error

An error made in typing.

Tiles visible with boundary invisible not allowed

Tiles on the surface of a model cannot be shown without visible boundary lines. Change the BOU INV command to BOU VIS.

TMG text not hitting a link line

A TMG text must be attached to a link line using a segment probe.

TMS text too long

A maximum of 120 characters is allowed.

Too few longitudes or too few latitudes

There must be at least two longitudes and two latitudes.

Too many 3D link lines/profiles

Only one link line is allowed to define the depth of a profile.

Too many CAN codes in text

A maximum of 10 CAN codes is allowed in a DBKEY command.

Too many depth points

Only two depth points are required to define the planes of the two outer profiles.

Too many depth texts

A maximum of 50 depth texts is allowed for each object.

Too many holes

The maximum number of holes allowed is 20.

Too many inferred points - reduce number of facets

The system limit for the number of points in a Coons patch is exceeded.

Too many partial revolutions

A maximum of 50 ANGLE commands is allowed for each object.

Too many profiles

The limit for the number of profile lines picked up by a link line is exceeded.

Too many section lines

There must be a matching number of line segments in each profile in the definition of a ruled surface model.

Too many sections in ruled surface

There must a matching number of line segments in each profile in the definition of a ruled surface model.

Too many user properties

A maximum of 20 user properties text strings is allowed for each object.

Too much 3D information on sheet

The system limit is exceeded.

Top and bottom faces coincident

If an open profile is swept around a closed flat centerline, the plane of the end points of the profile line must not be parallel to the plane of the centerline.

Two outer profiles required

Two outer profile lines must be attached to the link line using a diamond point function (FUNV 10).

Unable to open temporary file

The Modeler cannot open the required temporary file during the modeling process. There may be insufficient disk space.

Unevaluable centerline

The centerline cannot be used because the two points in it are coincident.

Unevaluable profile - all CON/NULS

Profile lines must contain at least two points to create a line.

Unevaluable view

There is no viewprim (isometric or orthogonal) in a viewbox, or an oblique viewprim is missing, or the text associated with an oblique viewprim is missing or incorrect.

Unknown text type

An unexpected text type is attached to a link line or vertex, edge, or face.

Viewbox has multiple prims

Only one viewprim is allowed in a viewbox.

WARNING: non perpendicular views

The defined views are not orthogonal to each other.

WARNING: object intersects itself

A volume of revolution model intersects itself when rotated around its centerline. Use segment probes to attach the profile to the centerline.

WARNING: problem resolving face of model

This is a warning of possible face to face problems when the Modeler performs Boolean operations.

Workspace overflow

The system limit is exceeded.

Viewer Error Messages

Cannot define new FROM/TO point

Check the arguments of the FROM TO command. The values you have entered may not be valid.

Cannot open file for reporting error messages

The file is corrupt or does not exist.

Depth confusion

Check the method you have used to define the depth and position of your model.

Detail level parameter out of range

The detail level parameter must be in the range 0 through 255.

Excessive shift/scale required to FIT

The scale factor you have specified with the FIT command is not acceptable. The image cannot be sized to fit to the viewbox.

FROM-TO distance too small

The order in which you have entered the commands has caused the FROM and TO location to be temporarily coincident. The given FROM or TO location will not be set.

Invalid range specified for detail level

The detail level parameter must be in the range 0 through 255.

Multiple view definition

You have used oblique viewprims as well as FROM commands to define oblique views.

No datum prim in view box

Each view must be contained in its own viewbox with one viewprim.

No Viewbox found on Sheet

A 3D sheet must contain a viewbox. The viewbox is constructed from a single closed line of type LVB.

Problem opening vector file

The vector file may be corrupted.

Problem opening workstation file

The workstation file may be corrupted, or access protected.

Problem reading workstation file

The workstation file may be corrupted, or access protected.

Sheet and model texts not defined

Make sure you have specified values for the arguments contained in the commands you have used.

Too many depths

Too many link lines are attached to the profile line.

Unable to retrieve text element

Make sure you have used a valid text type.

Undefined view

Make sure you have entered a valid viewing command.

Unevaluable clump

The group type specified is not a valid group type.

Unevaluable prim

The prim type specified is not a valid prim type.

Unevaluable view identifier

Make sure you have entered the correct name for the viewprim identifier.

Upvector not valid

The upvector is not allowed to be parallel to the FROM direction, that is the angle between the two vectors must not be less than 0.5 degree. You must redefine the upvector.

View boundary is not a line

The viewbox is constructed from a single closed line of type LVB.

View direction not valid

Make sure you have entered valid parameters with the following viewing commands:

Cross-section Views Error Messages

Cross-section prim not in any viewbox

You need to position an SLP prim in one of the viewbox.

Error generating cross sectioning object

Make sure your profile is correctly drawn and cross-section the object again.

Error processing cross section text

Make sure you have used the STX text type to name the cutting surface.

Matching cross-section symbol not found

The SLP prim is not available. Consult your System Administrator.

Must be a closed shape with straight line segments

Your profile must be a closed shape with three or more straight line segments and the shape must not intersect itself.

Must have all SEC ONs or all SEC OFFs

All the SEC commands for a view box must either be ON or OFF.

Only line segments allowed in a cut-line

You must create your profile with three or more line segments of line type SCL.

This cross-section prim has no sectioning line

Your SLP or SBP prim must be attached to one of the line segments.

Too many cross-section prims

A maximum of twenty cross-section prims is allowed on the profile.

Too many cut lines and sec boxes

A maximum of twenty section cut lines (SCL) and section box lines (SBL) is allowed on the profile.

Total length of cross-section object names too long

The maximum number of characters allowed for all the cross-section object names grouped together is 1000.

Unable to find cross-sectioning object
Make sure you have specified the correct object name.

Model Validator Error Messages

The following error messages are given in the Model Validator. The error messages are listed in alphabetical order.

Detail level parameter out of range (0-255)

The Model Validator only accepts detail levels that lie in the range 0 through 255.

Edge mismatch

There is a tile edge or face edge in the model that does not have an exactly equal and opposite edge in a neighboring tile. The model is invalid.

Error opening file

The file is already open, or it is protected. If you are running the Validator on a current sheet in MEDUSA4, this message may mean that there is insufficient directory space to create a file to hold the results of the validation.

File does not exist

The system is unable to find the model file specified in the IN command. Verify that you are attached to the correct directory and that the model file exists.

File does not have a valid model file header

Before processing starts, the Model Validator looks for the model file header. This message may mean that the specified file is not a model file.

GID mismatch

Each edge in a matching pair of edges should have the same geometric identifier (GID) associated with it. This is important in Boolean operations and reconstructions, but does not affect the Geometric Properties program.

Invalid range specified for detail level

The detail levels allowed are 0 through 255.

No input file specified

Specify the name of a model file before issuing the GO command.

Object detail level out of range, (0-255)

The detail levels allowed are 0 through 255.

Object has negative volume

The object has a negative volume, therefore the model is invalid (in most cases).

Object not solid

The Model Validator only accepts solid objects for validation. Non-solid objects are ignored.

Segment direction mismatch

Two tile edges line up correctly, but they are going in the same direction. Tile edges should go in opposite directions. The model is invalid.

WARNING - non-manifold object with n identical edges

The Model Validator has found more than two identical edges in an object. This means that the object is not completely solid. This message arises when objects are only connected by edge to edge contact. Problems may occur if further operations are performed on this model, although the model is valid.

Shrinker Error Messages

The following error messages may be generated while running the Shrinker program. The error messages are listed in alphabetical order.

Error opening file

The file is already open or is protected.

Error opening output file

The file is already open or is protected.

File does not exist

The system cannot find the model file specified in the IN command. Ensure that you are attached to the correct directory.

File does not have a valid model file header

Before processing starts, the program verifies the header in the model file. The message may mean that the file is not a model file.

Error writing model file header

Before processing starts, the program verifies the header in the model file. The message may mean that the file is not a model file.

Model file not specified

Specify a model file using the IN command before issuing the GO command. Output file not specified

Output file not specified

Specify an output file using the OUT command before issuing a GO command.

MODASC Error Messages

The following error messages may be generated while running MODASC. The error messages are listed in alphabetical order.

Defaulting to exponential format. Invalid number of decimal places specified.

The number of decimal places specified in the `DP` command must be in the range 1 through 8.

Error opening file

The message means that the file is already open or is protected.

File does not exist

The system cannot find the model file specified in the `IN` or `BIN` command. You should ensure that you are working in the right directory.

File does not have a valid model file header

Before translation starts the model file header is checked. This message indicates that the file is not a model file.

No input file specified

A model file must be specified, using the `IN` or `BIN` command, each time MODASC is run.

MODSMO Error Messages

The following error messages may be generated while running MODSMO. The error messages are listed in alphabetical order.

Cannot open input model file

The file you have specified cannot be found. Ensure that you are working in the correct directory.

Cannot open output model file

The filename given for the output smooth model file is not a valid filename.

File not found

The input model file you have specified has not been found. Model filenames are by default suffixed by *.MOD*. Ensure that you are entering the model filename correctly.

Model File Translator Error Messages

The following error messages may occur while running the MIF program. They are listed in alphabetical order.

Cannot open input model file

The system cannot find the specified file. You should ensure that you are working in the correct directory.

Cannot open output MIF file

The filename given for the output file is not a valid file name.

Invalid file header, file not a model file

The specified input file is not a model file or has been corrupted.

Maximum no. of MIF edge heap locations

Maximum no. of MIF surface heap locations

Maximum no. of MIF point table entries

Maximum no. of MIF edge table entries

Maximum no. of MIF surface table entries

A MIF program limit is exceeded.

No open input model file

A model file must be specified by an `IN` command before a `GO` command is given.

No open output MIF file

An output file must be specified using the `OUT` command before a `GO` command is given.

Previous output file closed - please open new file

A new output file must be specified for each input file.

Terrain Modeler Error Messages

This section lists the error messages that are given in the Terrain Modeler and explains what to do when you get these errors. The messages are listed in alphabetical order.

Area tolerance must be positive

You must enter a positive value for the `SMOOTH AREA` command. Negative tolerances are not allowed in the Terrain Modeler.

Base is too high

The Terrain Modeler generates this error if the base height is higher than one of the Z-coordinates in the data file. The default base height is 0.

Reduce the base height using the `BASE` command.

Base is too high - no object produced

The Terrain Modeler gives this error if you try to generate a model with a base height above some of the coordinate data points. Change the base height using the `BASE` command before trying to create another model.

Detail level parameter is out of range (0 - 255)

You have specified an invalid detail level. The Terrain Modeler only accepts detail levels that lie in the range of 0 through 255.

Error in file at line number

The data file should contain only X, Y, and Z-coordinate data. Edit the file to delete or change the specified line.

Ill-conditioned data, triangles created too thin

The coordinate data is ill-conditioned because it results in the creation of badly shaped triangles for the model surface. To overcome this problem specify a smaller value for `SMALLEST ANGLE`. If this error occurs when you are smoothing the model surface, specify a smaller value for `SMOOTH ANGLE` or a larger value for the `SMOOTH AREA` tolerance.

Insufficient data points

The Terrain Modeler requires at least three points in the coordinate file to generate a model.

Insufficient memory

There is not enough memory in your machine to create a model from the data you have specified. The Terrain Modeler automatically closes on this error and returns you to the operating system. If this problem occurs when smoothing, try again using a larger area tolerance.

No model file specified

You need to specify the name of a model file in which the Terrain Modeler can store the model. Use the `OUT` command to specify a model file.

Number of cross-sections must be greater than 0

Specify a positive number of cross-sections when using the `XSECT X` or `XSECT Y` commands.

Problem opening input file *filename*

The Terrain Modeler cannot open the specified data file, probably because it does not exist in the specified directory. Look for typing mistakes. If the file is not located in the current directory, ensure that the full file specification is given.

Problem opening output file *filename*

The Terrain Modeler cannot open the model file specified using the `OUT` command. Ensure that the model file has a valid name.

Removing repeated point (x, y, z)

The Terrain Modeler does not allow the same X and Y-coordinates to be given to two different heights (Z value). It automatically removes the repeated point when reading the coordinate data file.

Smallest angle is greater than the smoothing thin angle tolerance. Thin angle tolerance increased to 'angle'

The value given to `SMOOTH ANGLE` is smaller than the value given to the `SMALLEST ANGLE`. The Terrain Modeler does not allow the smallest angle for smoothing to be less than the smallest angle used in the triangulation process. The `SMOOTH ANGLE` value is automatically increased to equal the value of `SMALLEST ANGLE`.

Smallest angle must be in range 0 to 60 degrees

You have specified an angle outside the range 0 through 60 degrees for the smallest angle. Use the `SMALLEST ANGLE` command to specify a new angle.

Warning - program limited to 10000 points

The number of points in the data file exceeds 10,000. The Terrain Modeler can only handle a maximum of 10,000 points. To produce a model you must reduce the number of points in the file.

Text Driven Modeling Error Messages

The following error messages are generated by the Interpolator. The messages are listed in alphabetical order.

Bad sequence of CONS and ARCS

Each point that is given an ARC function must be preceded by a point with a CON function.

Duplicate points, cannot smooth

Smoothing is not possible between points that are coincident or nearly coincident.

Invalid Mesh

A mesh has been defined such that either one or more latitudes cross, or one or more longitudes cross.

Latitude does not exist

A mistake has been made in specifying the line number in a REF command.

Latitude x does not mesh with longitude y

The specified latitude and longitude do not intersect.

Longitude y does not mesh with latitude x

The specified latitude and longitude do not intersect.

Point does not exist

A mistake has been made in specifying the point number in a REF command.

Too few longitudes or too few latitudes

At least two longitudes and two latitudes must be defined.

Too many latitudes or longitudes

The number of latitudes or longitudes has exceeded the maximum limit of 200.

Model File Compression Error Messages

Cannot read file

The input model file may be access protected or corrupted.

Cannot save file

There is insufficient disk space to allow the output model file to be saved.

The file is NOT a Model

Model filenames are by default suffixed by .MOD. Ensure that you are entering the model filename correctly.

Unable to open file

The input model file does not exist or is access protected.

GLOSSARY

This glossary covers the terms commonly associated with the MEDUSA4 3D Modeling system. All entries are listed in alphabetical order.

Assembly

MEDUSA4 model created from several separate models using the instancing technique.

Bend radius

Centerline radius applied to bends on a pipe model.

Boolean operation

Logical operation involving the addition, subtraction, or intersection of models.

Chord tolerance

Maximum deviation between a true arc and the tile edges produced by the Modeler to represent a curved surface.

Coincident faces

Two models having a face (or part of a face) in common.

Command block

Area provided on a standard MEDUSA4 3D sheet in which Modeler, Viewer, and Reconstruction commands can be conveniently grouped.

Component

Object that makes up part of an assembly model.

Coons patch

Type of patch used on Meshed Surface models.

Coordinate zero

Position defined by the datum of a viewprim, at which the coordinates are $X=0$, $Y=0$, $Z=0$.

Cutting plane

The plane which bisects the model into two.

Database key

Name given to an object in the Assembly Modeler.

Datum

Corner point of a viewprim. This point is the coordinate zero of a viewbox.

Depth text

TMG text attached to the link line on a definition sheet to indicate the depth or height of the model in the third dimension.

Detail level

Level in the range of 0 through 255 that can be selected for the creation of objects in a model. The 3D Viewers can be used to display different levels selectively.

Duct

Model created by the Duct generator.

Edge

Type of model produced by the Sweep generator or the boundary of a tile.

Exploded view

View of an assembly that shows the components exploded away from each other.

Face

Type of model produced by the Sweep generator or another name for a surface tile.

Facet

Same as tile. Small planar polygon or face of which the surface of a model is composed.

GID

Geometric identifier used to identify patches and tiles in a model file.

instance group

Group on an assembly sheet containing a viewprim and text which defines the name and location of a model file, from which the Modeller will take a copy of the model (or object) and add it to the current assembly.

Instancing

Technique to represent models defined in other sheets on the current sheet. For details see “Assemblies and Exploded Views” on page 241.

Intersection lines

Lines that are created by the Reconstructor between intersecting, non-Boolean objects.

Isometric

A kind of spatial representation. All three axes are drawn at the same scale.

Latitude

Closed wire-like profile which defines a line around the surface of a Meshed Surface model.

Link line

Line attached to a model profile and extended into an orthogonal viewbox to define the type of model and the depth and/or position of a model in the third dimension.

Longitude

Open wire-like profile which defines a line along the surface of a Meshed Surface model.

Meshed Surface

Free-form model produced using the Meshed Surface generator.

Model

MEDUSA4 representation of a real object. A model consists of MEDUSA4 objects which are stored in a model file.

Model generator

Program which generates a particular type of model, for example, a sweep or a slide.

Model validator

Program used to determine the validity of solid model or a solid object within a model.

Modeller

Program used to produce a model from a two-dimensional definition. The model is stored in a model file.

Object lines

Lines created by the Reconstructor to show the basic shape of an object.

Oblique prim

Graphical element placed in a viewbox to allow oblique viewing or model generation.

Oblique view

View whose view direction is inclined to the MEDUSA4 3D axes.

Origin

Position defined by the datum of a viewprim (coordinate zero).

PAP prim

Prim used to define the solder points of the latitudes and longitudes on a Meshed Surface model.

Partial volume of revolution

Model created by rotating a profile around a centerline through less than 360 degrees.

Patch

Four-sided surface that can be completely described by an equation or a set of mathematical functions. The surfaces of MEDUSA4 models are composed of patches.

Patch boundary

Edge of a patch.

Pipe

Model created by the Pipe generator.

Polygon

Planar figure used to represent part of the surface of a model in a model file.

Profile line

Line in a model definition which represents the outline or cross-section of a model.

Rational patch

Patch whose equation is a rational polynomial function of its parameters.

Reconstruction

Process of translating the 3D views generated by the Viewer into the data structure of a 2D sheet.

Ruled surface

Model created by the Ruled Surface generator.

Section lines

Lines created by the Reconstructor when an object is sectioned.

Sectioned model

Parts of the model left after sectioning.

Sectioned surface

New surface of an object created by cross-sectioning.

Sectioning

Removal of the front section from a model by cutting through the model in a plane perpendicular to the line of view.

Shell

Model which represents only the surface of a real object.

Slide

Model created by the Slide generator.

Smoothing

Process used by the Duct generator to produce a continuous lengthwise surface on a duct model.

Solder point

Points at which the latitudes and longitudes intersect to form a mesh on the surface of a Meshed Surface model.

Solid

Model which completely represents a real object.

Spine segment

Line segment on the line which defines the basic shape of a duct.

Start point

Position at which the Modeller attaches a profile to a slide path.

Sweep

Model generated by the Sweep generator. There are three types of sweep: solid, face, and edge.

Tile

Same as facet. Small planar polygon or face of which the surface of a model is composed.

Uncommitted properties

Text that is used to label objects in a model file. This text is used by other MEDUSA4 programs, for example, the Standard or Full Feature Shaders. It is ignored by the Modeller.

Viewbox

Closed line, normally rectangular, of type LVB on a 3D sheet. It is used for containing part of a model definition and (if required) a reconstructed view of that model.

Viewprim

Graphical element placed in a viewbox and which determines the orientation of the coordinate system for that viewbox.

Viewer

Program used to display a model on the graphics screen (Interactive Model Viewer) or in a viewbox on a 3D sheet (Sheet Viewer).

Volume of revolution

Model created by rotating a profile around a centerline through 360 degrees.

Wire

Model created by the Wire generator.



INDEX

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Symbols

&BLOCK 214

Numeric

3D Attributes 37
 General 38
 link line
 TRANSPARANCY 51
 Linkline 50
 Modeler 41
 Reconstructor 44
 Sheet Level Attributes 39
 Viewbox 46
 Viewer 42
 3D commands
 Modeler commands 21
 3D cross-hatching command 325
 3D default settings
 Create Smooth Model 15
 3D line types 18
 3D sheet
 designing 20
 3D Sheet Attributes,tool 39
 3D specific components
 3D commands 18
 3D Viewers 465

A

addition 220
 Aiming Point 295
 moving 309
 Alignment points 205
 ALLSHAPES ON | OFF 420
 Assembly 551
 Assembly View
 create an ... 263
 Attributes
 3D - 37
 Averaging Polygon Normals 433
 AVIEW command 263
 Axes 379

B

Base Height, Model 479
 basic commands 467
 BEARING 217
 Bend radius 551
 Black boxes 375
 BLOCK 214
 Boolean Operation 210
 Addition,Substraction,Intersection 220
 complex 231
 invoke 216
 multiple 229
 Boolean operation 551
 Boolean Operations 13
 Boolean process 231
 BOOTOL command 235, 334
 Boundary lines, Terrain Modeler 464
 Breaklines, Terrain Modeler 464

C

Center-point arcs 359
 changing
 size of view 55
 type of projection of a view 299
 Chord Tolerance
 specify 283
 Chord tolerance 551
 chord tolerance 499
 CHOTOL command 284
 control of curve approximation 285
 Improving volume calculation accuracy 284
 CHOTOL tolerance_value command 283
 CHOTOL value 283
 CIR command 359
 closed 96
 closed profile
 defining a model using a - 100
 Coincident faces 551
 Command
 COMPRESSION ON 292
 DRAW 54
 NODRAW 54
 SCALE 250
 SKETCH 54

SURF 290
TARGET 352
Command block 551
Command text 18
Commands
 AVIEW 263
 Setting 54
complex models 86
Component 551
Components of 3D Sheet 18
compressing a model file 292
Compression 517
COMPRESSION OFF command 517
COMPRESSION ON command 292
Concave models 463
Continuation Flag 446
Contour line data 463
Contour Lines 480
controlling
 line compression 360
controlling the sectioning 326
conversion programs
 tools 402, 426
Converter Tools, how to run 403, 427
Convex models 463
Coons patch 551
Coons patch equations 289
coons patch output 507
Coons Patches 438
Coordinate zero 552
Create
 assembly view 263
 model of a pipe 144
 Model Using Non-planar Centerlines 176
 wire model 152
create
 wire models by attaching the TMG text 163
create by attaching the TMG text 163
Create Smooth Model, 3D default settings 15
Creates an Oblique View Tools 313
CreateShell models 163
Creating
 models containing holes 119
 multiple ruled surface models 121
 solid model 126
CRH2D command 322
CRH3D command 323
Crosshatched sections 322, 323
Cross-sections 480
Curve Reconstruction Accuracy 359
curved face 283
CURVES
 command 359
 tolerance_value 359
Cutting plane 552

D
Dashboard
 Linkline 50
 Viewbox 46
Data File, coordinates 466
Database key 552

database keys 357
Datum 552
DBKEY command 357
defaults.dat 31
defining
 start point 129
 view 310
Degree of Perspective
 setting 300
Delaunay triangulation 457
Depth Text 28
Depth text 552
Detail level 552
detail level 288, 507
Dialog
 3D Attributes, Modeler 41
 3D Attributes, Reconstructor 44
 3D Attributes, Viewer 42
 Line Properties for Linkline 51
 Line Properties for Viewbox 47
 Model Description 66
 Model Viewer 475
dimensioning
 reconstructed curves 359
Displaying model 475
DRAW command 54
drawing a view 54
drawing view 310
DTM, abbreviation 456
DTM, command in MEDUTIL 468
DTM, Digital Terrain Modeler 456
dtm, MEDUTIL command 468
DTM, tool 465
Duct 552
duct definitions 168
Duct generator 167, 289
Ducts 12
DWG 421
DXF 421
DXFSEW 420

E
Edge 552
Edge Sweeps 76
Element Types 94
EQUATIONS OFF command 289
EQUATIONS ON command 289, 355
Error Messages 522
Example
 Camera 273
 mass properties program output 380
 meshed surfaces 508
 Modeling and Viewing Process 30
 producing a solid model 71
 sectioning and crosshatching in assemblies 332
 Shrinker Output 397
 sweep 500
 TARGET command 353
example
 Data file for Terrain Modeler 466
example file, shutt.txt 465
example sheet, shutt.she 470

Exploded view 552
Extrapolation 377

F

Face 552
face sweep 74
Facet 552
FACET command 174, 204
FIT command 55
FIT scale_factor 301
FROM commands 305
FROM DIR x y z 306
FROM DIST DIST distance 307
FROM DIST distance 305
FROM FROM x y z 307
FROM LEFT (RIGHT, UP, DOWN) angle 307
FROM LEFT (RIGHT, UP, DOWN) distance 307
FROM Point
 moving absolutely 305
FROM SPH latitude longitude 305
FROM TO x y z 305
FROM x y z 305

G

GID 553
Google Earth Export 405
Google Earth interface 405
Grid
 projecting onto the surface, Terrain Modeler 484
group type
 specifying 351

H

hidden lines
 visible,invisible,dotted 56
hole in a volume of revolution
 defining a - 103
hole profiles 498

I

IGES 414
image
 rotating 298
 sizing 300
individual objects 372
Instance group 553
Instancing 553
Interpolator 494, 495
 program 493
 running outside MEDUSA 494
Interpolator Commands 512
Intersection
 lines 59, 60
intersection 223
Intersection Lines 60
Intersection lines 553
Isometric 553
isometric viewprims 23

L

Latitudes 194
Latitude 553
Latitude wire line definitions 192
latitudes 191, 505
Layer 352
Line Compression
 controlling 360
Line Patterns 90
line typ 352
Link Line
 locking a link line to a profile 89, 186
Link line 553
Link lines 26
Linkline Attributes 50
Linkline Dashboard 50
Loading a standard 3D sheet 19
Locking a link line to a profile 89, 186
Longitude 553
Longitude wire line definitions 192
Longitudes 191, 193
longitudes 505

M

macro file, running 468
macro file, Terrain Modeler 467
MAKE command 210, 216
Manipulating the Model 487
mass properties
 program 367
 program commands 382
Mass Properties Program 370
Matching Points on Profiles 173
Matching Profiles 116
MEDDTM Product 456
MEDMERGE 435
MEDMERGE Commands 435
MEDMIF 439
 running 441
 running outside MEDUSA 442
MEDMIF Commands 452
MEDSTL Commands 411
MEDUTIL
 Running the Terrain Modeler 468
MEDVDA 403
MEDVRML 408
MEDVRML Commands 408
Meshed Surface 12, 197, 553
 generator 289
 model definition 207
Meshed Surface generator 191
Meshed Surface Model 192
Meshed Surface model definition 192
Meshed Surface Models 504
meshing tolerance value 203
MESHTOL command 203
MESHTOL value 203
MIF File
 structure 438
MIF File Format 446
MODASC 429

MODASC Commands 430
MODBIN 429, 431
MODBIN Commands 431
MODCOMP 518
MODCOMP Commands 520
Model 553
 Displaying 475
 sectioning 318
model definition sheet 13
Model Description Text 63
 create on sheet 65
 language 67
 use wildcards 67
Model Description Text File
 create automatically 64
 create manually 66
Model File 465
 structure 438
model file 496
 compressing 292
Model File Translator 437
Model File, creating using MEDUTIL 465
Model File, creating within MEDUSA4 470
Model generator 554
Model Generators 12
model of a pipe
 create - 144
Model Orientation
 changing - 106
Model Saving Expression 31
Model Shape with Profile Orientation 175
Model Specification Text 21
Model surface, smooth 477
model type of an object 506
Model Using Non-planar Centerlines
 Create 176
Model Validator 361
Model validator 554
Model Viewer 343
Model Viewer Dialog 475
Model viewing 473
Modeler 30, 31
 3D Attributes 41
Modeler commands 21, 281
Modeling
 objects containing a single hole 78
 objects containing several holes 80
 oblique features 91
 process 29
 techniques 78
Modeling Techniques 134
Modeller 554
Modeller Text ass. by Geometry 21
Models concave 463
models containing holes
 creating 119
Models convex 463
MODEXPORT Commands 414
Modification of Viewer commands 54
MODIMPORT Commands 418
MODINFOMODE ON | OFF 421
MODSMO 429, 433
MODSMO Commands 434

moving
 aiming point 309
 FROM point absolutely 305
 TO point absolutely 309
 TO point relative to the current TO point 309
 viewpoint 305
MSHELL command 239
multi-object
 from separate profiles 84
multi-object models
 from a single profile 82
Multiple Boolean Operations 229

N

NAME BLOCK 214
NAME command 211
Naming
 a new object 217
 latitudes and longitudes 196
 object 214, 287
NOATTLOAD 419
NODRAW command 54
Non-planar (3D) centerlines 176
Non-planar (3D) wire models 156
Non-planar paths 137
numerical accurate 232

O

OBJECT command 372
Object lines 554
Oblique Features
 modeling 91
Oblique prim 554
Oblique view 554
oblique viewprims 310
Oblique Views 310, 313
ONTO Point 297
Open centerlines 136
open profile
 defining a model using an - 97
Open profiles 134
Origin 554
orthogonal viewprims 23, 310

P

PAP prim 554
PARS command 300
PARS factor 300
Partial volume of revolution 554
Partial Volumes of Revolution 107
PARW length 300
Patch 554
Patch Boundaries 59
Patch boundary 554
Patch Data 289
PER command 299
PERA angle 301
PERS factor 302
PERSA factor angle 302
perspective angle 301

- perspective viewing 301
 - PERW length 302
 - PERWA length angle 303
 - Pipe 555
 - Pipe Bend Radius 147
 - Pipes 12
 - Point Functions 27
 - Point functions 18
 - Polygon 555
 - Polygon Normals
 - averaging 433
 - Polygons 12
 - primary centerline 139
 - Prims 18
 - Product
 - MEDDTM 456
 - Profile line 555
 - profile lines 24
 - Profiles
 - open 96
 - projection
 - type of ... 295
 - projection surfaces 139
 - properties
 - uncommitted 290
- R**
- Rational patch 555
 - Rational Patches 438
 - Rational quadratic patch equations 289
 - reconstructed curves
 - dimensioning 359
 - reconstructed geometry
 - groupstructure
 - specifying 355
 - identifier text
 - specifying 356
 - Reconstruction 34, 555
 - Reconstruction commands 21
 - Reconstruction Process 29
 - Reconstructor
 - 3D Attributes 44
 - Reconstructor Text 21
 - Record Type 446
 - RESETDETAIL 419
 - rotating the Image 298
 - Ruled surface 555
 - Ruled Surfaces 12, 121
 - run Digital Terrain Modeler 465
 - running MEDMIF 441
 - running shrinker 394
 - Running the Digital Terrain Modeler 468
- S**
- SCALE command 250
 - scale_factor 301
 - SEC command 326
 - SEC ONLY command 327
 - SECTION
 - box_line_name 329
 - command 318
 - cut_line_name 319
 - Section
 - from a limited plane 321
 - from an infinite end plane 319
 - from cranked planes 327
 - perpendicular to the viewing direction 318
 - Section Box Prim 329
 - SECTION command 319
 - Section lines 555
 - Sectioned model 555
 - Sectioned Points on a wire
 - highlighting 360
 - Sectioned surface 555
 - Sectioning 555
 - sectioning
 - controlling 326
 - sectioning model 318
 - SEPMODMODE 421
 - setting
 - degree of perspective 300
 - Shaped sections 329
 - Shared points 117
 - Sheet Level Attributes 39
 - Sheet Mode, Viewer command 54
 - Sheet Viewer 557
 - Sheet Viewer, 3D 473
 - Shell 555
 - SHELL DBKEY CLUMP 358
 - shell model 88
 - Shell models 163
 - SHRINK command 394
 - Shrinker 393
 - running 394
 - running outside MEDUSA 394
 - Shrinker Commands 399
 - shutt.she, example sheet 470
 - shutt.txt, example file 465
 - Simple Ruled Surface Model 113
 - Size of a View
 - changing 55
 - sizing image 300
 - SKETCH command 54
 - Sketching a View 54
 - slab model 71
 - Slide 555
 - Slide generator 126, 129
 - Slides 12
 - Smoothing 556
 - Smoothing the Model 171
 - Smoothing the Model Surface 477
 - Smoothing the profile 497
 - Solder point 556
 - Solid 556
 - Solid Sweeps 71
 - specifying
 - database keys 357
 - group type 351
 - individual objects 372
 - layer 352
 - line type 352
 - reconstructed geometry group structure 355
 - reconstructed geometry identifier text 356
 - Type of Reconstructed Arc 359

Spine segment 556
Spot Heights, Terrain Modeler 485
Standard 3D Sheet 18
 loading 19
Start point 556
start point 129
STEP 417
STL 411
STLMED Commands 412
structure of a model file 438
structure of MIF file 438
subtraction 222
SURF command 290
surface
 detail 59
 model 74
 tiling 59, 60
Sweep 556
sweep limits 497
Sweep Models 496
sweeps 12

T

Tangent-point arcs 359
TARGET CLUMP command 351
TARGET command 352
 examples 353
TARGET EQUAT command 355
TARGET GID command 356
Terrain Modeler 455, 467
 Constraints 463
Terrain Modeler Commands 489
Terrain Modeler, basic commands 467
Terrain Modeler, creating macro file 467
Terrain Modeler, running 468
text NAME command 214
TIL VIS command 438
Tile 556
TO Point 309
 moving absolutely 309
 moving relative to the current TO point 309
TO point 295
tolerance_value 283
Tool
 3D Sheet Attributes 39
 DTM 465
TRANSPARANCY 51
trimetric viewing 300
type of projection 295
 of a view
 changing 299
Type of Reconstructed Arc
 specifying 359

U

Uncommitted properties 556
uncommitted properties 290
Units 376
Upvector 297

V

VALID command 362
Validating the model 487
Validator
 Basic Commands 363
 running outside MEDUSA 362
VDAMOD 404
view
 defining 310
 drawing 310
View Specification Text 21
Viewbox 556
Viewbox Attributes 46
Viewbox Dashboard 46
Viewbox Mode, Viewer command 54
Viewboxes 18, 22, 296
Viewer 30, 32, 557
 3D Attributes 42
 parameters 294
Viewer command 21
Viewer Commands
 Modification 54
 Sheet Mode 54
 Viewbox-Mode 54
Viewer commands 21
Viewing
 commands 335
 concepts 294
 model 13
 perspective 301
 process 29
viewing
 trimetric 300
viewing commands 474
viewing model 473
Viewpoint 295
 moving 305
Viewprim 556
Viewprims 18, 22
 oblique views by ... 310
VIEWTOL command 334
Volume of revolution 557
Volume Revolution Generator 97
Volumes of Revolution 12

W

Windows 296
WINM command 377
Wire 557
WIRE Command 201
wire model
 create 152
Wire models 156
 from projected profiles 156
wire patterns 154
Wires 12
Wires from primary and secondary profiles 156
Wires from projected profiles 161