# DeskArtes

## Industrial Design System

Reference Manual

Version 4.4

***Contact Address***

DESKARTES OY
Pihlajatie 28
FIN-00270 Helsinki
Finland

Tel. : +358–9–644335
Fax : +358–9–644330

Email: DA@deskartes.fi

http://www.deskartes.com/

# Contents

# FOREWORD

## About this Manual

This Manual describes in detail the complete functionality of the ***DESKARTES Industrial Design System***.

The Manual is divided into several sections. Depending on which modules of the system you have purchased, you may be more interested in some parts than others.

The Reference Manual sections are:

- ***Part 1: System Overview***

  This gives a basic introduction to the system at an easily understood level. It introduces various concepts that are referred to later in the rest of the manual.

- ***Part 2: A Typical Modeling Session***

  This gives an overview of the various stages that you will go through when developing a new design. It introduces many concepts but does not dwell on the actual commands used.

- ***Part 3: Menu Commands***

  This explains the commands that can be found in the menu bar.

- ***Part 4: Window Commands***

  This explains the commands associated with the Database, Object and Settings window.

- ***Part 5: Editing Functions***

  This explains the commands in the special editing modes for the modeling and visualization commands.

## How to use the Manuals

Users of the DESKARTES system will be provided with at least two manuals, a Tutorial, and this Reference Manual.

The easiest way to ***learn*** DESKARTES is to sit down at a workstation, and go through the DESKARTES Tutorial. It gives you systematic descriptions of how to actually use the system using an example model, proceeding from some relatively simple tasks to the most demanding functions you'll ever need.

The System Overview and Typical Modeling Session sections of this Reference Manual are also recommended reading to any new DESKARTES user who wishes to have a ***quick look*** at the functionality. The other parts of this manual are intended mainly for ***reference*** use, when you need to understand the details of a particular command. It would also be a good idea to go through the complete Reference Manual after you have used the system for a while, to really get to know the full extent of the different uses and possibilities for the software.

Users of rapid prototyping machines will be provided with a separate ***DeskArtes Rapid Tools Manual***, explaining DESKARTES more specialized functionality for their application.

For installation and other system related issues, consult the ***DeskArtes Technical Manual***.

## Notation

Key words and other interesting words in this manual are written in ***bold italic.*** Command names and some technical terms are written in `this typeface`. Keyboard function shortcuts are denoted in bold, like **a** and **A**.

The menu commands are written as `MenuName`⇒ `CommandName.` If the menu is associated with a particular window then the command is written as `WindowName`⇒ `CommandName`. However, when the corresponding menu or the window name is obvious from the context, it is not always given.

# 1 SYSTEM OVERVIEW

## *Launching DeskArtes*

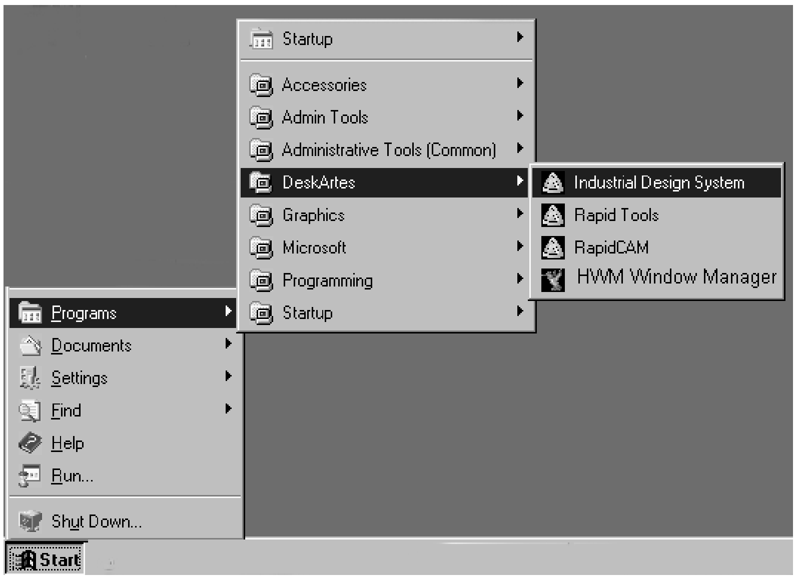
This section assumes you have DeskArtes installed and running on your workstation. The installation procedure is discussed in the Technical Manual. Most computers require you to login, telling the computer who you are. To do this you need a unique name and, usually, a secret password. Your system provider or system manager should be able to give you these and explain how to use them.

## *Starting Up – Unix/MacOS*

To start up DeskArtes, enter the command `DA` in a command window. After a while, the system displays the DeskArtes user interface.

## *Starting Up – Microsoft Windows*

Using Windows, starting up is a two-stage process. It is first necessary to start the window manager before starting the DeskArtes software itself. The commands for each of these procedures are found on the Start menu at the very bottom of the screen. To start the window manager choose the command `HWM Window Manager` from Programs under the Start button. After a few seconds, you should notice an Icon Manager window appear usually in the top left-hand corner of the screen. Now to start up DeskArtes, choose the command `Industrial Design System` again under the Start button. These menu choices can be seen in the following diagram. It is possible that your system will not appear exactly as the diagram but you should be able to locate the commands.



After a while, the system displays the DeskArtes user interface.

## The User Interface

### Input Devices

The ***keyboard*** works like a typewriter. It is used to enter text and numeric information into the system.

The ***mouse*** is used to move a ***cursor***, visible on the screen. The shape of the cursor varies with the state of the system.

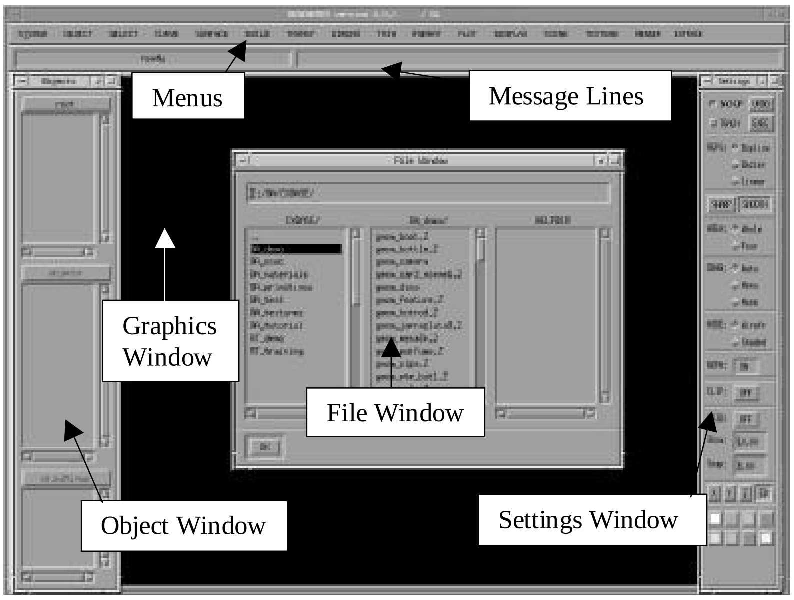
You may have access to a ***tablet***, ***scanner*** or digital camera to capture graphics. Such external devices are not accessed directly by DESKARTES. To use the information from such devices you must use one of the data exchange files, such as DXF, TIFF, etc. Scanned graphics can be used as images for texture mapping, or as templates for computer modeling.

### Mouse Functions

DESKARTES makes full use of the ***mouse***. It is best to operate the software using a three-button mouse although a two-button mouse is also acceptable. If you only have a two-button mouse, the third (middle) mouse button is emulated by pressing *both mouse buttons* together. This can be a little tricky at first, but don't worry because the most common commands have been assigned to the left and right buttons.

### Windows—What You See

The main windows of the DESKARTES ***user interface*** are shown in the following figure.



The windows and their uses are:

- The ***Graphics Window***, the area that takes up most of the screen, is used for graphical operations such as entering geometric data and displaying objects.
- The ***Menu Bar***, at the top, contains the DESKARTES commands. Each of the headings in the menu bar holds a menu of commands, chosen by pressing the left mouse button.
- The ***Object Window***, to the left, is used to determine the structure of your model and also shows which object is currently selected for the next command. The object in question is darkened, and is called the ***target object***. A pop-up menu appears with object-related commands, if the middle mouse button is pressed within the Object Window.

- The ***Settings Window,*** to the right, contains variables that control the display environment. Here you may control things like the viewing direction and select the color of your drawing, amongst other things.
- The ***File Window*** shows the model directories and the files they include. It also includes a number of commands used to control files and directories accessed through a middle mouse button pop-up menu.
- The ***Message Lines***, below the menu bar, display all the text messages, such as system status, information on the target object, answers to geometric queries, prompts, and error messages.

Other DESKARTES ***windows*** are:

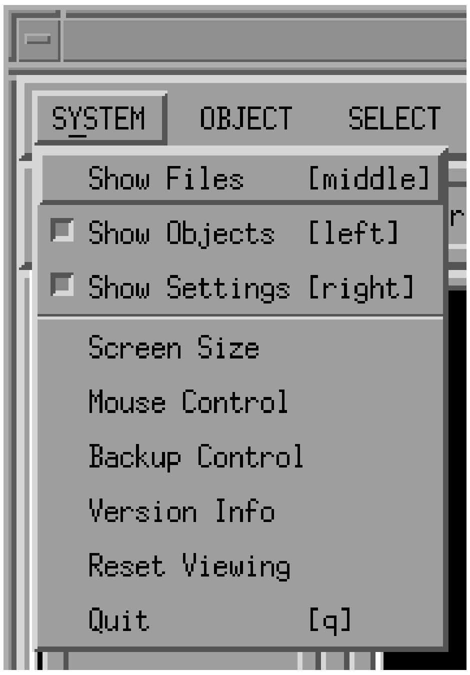
- ***Dialogue Boxes*** (also known as ***Parameter Windows***) are used to enter all numeric and text input to the system. Dialogue boxes pop up automatically when the system needs such input.
- ***Other Windows*** will appear at certain times during the modeling process.

## *Controlling DESKARTES*

### *Command Mode*

DESKARTES is controlled in two modes. The default mode is called ***command mode***. The principles of operation in command mode are explained in the following paragraphs. Some commands place the system in one of the ***edit modes***, which will be discussed later.

The DESKARTES commands are grouped into ***menus***. There are two kinds of menus: ***pull-down menus*** (located in the menu bar) and ***pop-up menus*** accessed by the right hand mouse button as already described.



To ***select a command*** from a pull-down menu, point the cursor at the desired menu title and hold down the left mouse button. Still holding the button down, drag the cursor to the desired command. ***Execute*** the command by releasing the mouse button.

Some commands may be entered using a single letter ***shortcut,*** without using the menus. This is a useful way of accessing often used or repeated commands. For example, the commands in the

`DISPLAY` menu are used so often that their shortcuts should be learned as quickly as possible. The available shortcuts are shown within brackets `[ ]` next to each menu item (such as `Quit [q]`).

Some things are controlled with ***buttons***. In particular, the Settings Window contains buttons that control the way DESKARTES works: whether to use a grid, how to display the objects, when to undo commands, and so on.

## *Executing a Command*

When a command has been given in one of the ways described above, the system displays a "busy" message and the cursor turns into a ***watch***. This indicates that execution of the command has been started.

Until the current command is completed or canceled, attempting to enter a new command is denied by the system. To show that it is ready to accept a new command, DESKARTES displays "ready" in the message lines and the cursor turns into an ***arrow***.

## *Interrupting a Command*

Some display commands, such as drawing a wire-frame or shaded pictures, can be ***interrupted*** by pressing the `ESC` key, or alternatively `Ctrl-C`, with the cursor in the graphics window. This can be useful if the execution seems to take too long and you wish to continue with other commands instead.

## *Edit Modes*

Commands that require you to graphically interact with your model switch the system to one of several ***edit modes.***

None of the normal commands are available in the edit modes. The Object Window and the menus disappear, and small pictures called ***icons*** appear instead.

The icons graphically describe what each command does. To execute an editing function, click on an icon with the left mouse button. Alternatively, press the shortcut key that is displayed in the upper left-hand corner of each icon.

When using function keys in edit mode, note that lowercase and UPPERCASE letters have different meanings. For instance, in the command `CURVE`⇒ `Design`: `Edit`, function **`c`** (lowercase) produces a rounded corner near the active point, while **`C`** (uppercase) makes the curve closed.

A brief ***help text*** of the icon on which the cursor is currently resting appears in the message lines if you click on the icon with the right mouse button.

To get back to command mode, click on the "Smile" or "Sorry" icons. Using the "Smile" icon, any changes you made in the edit mode session will be accepted. Using the "Sorry" icon any changes you made will be discarded.

## *Mouse interaction*

In many cases, ***e.g.***, when moving an object or a point to a new location, you are requested to define the actions interactively with the mouse.

The action may depend on which ***mouse button*** you press when starting the action. For instance, when moving an object, the left mouse button moves freely in all directions, while the middle button would move in one direction only.

There is also a difference between "clicking" and "pressing" a mouse button. ***Clicking*** means pushing on the mouse button, and immediately releasing it again. ***Pressing*** means pushing the mouse button down, and ***holding*** it down until the interaction is finished.

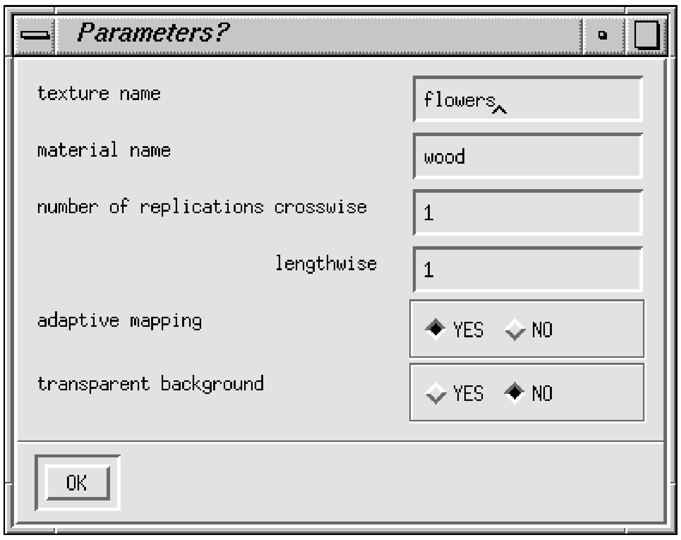
In most cases, DESKARTES doesn't mind if you click or press. For instance, to move an object, you may either

1) Press, move, and release when you done, or
2) First click, then move and click again when ready.

## *Parameters*

Many commands require ***parameters***. A parameter is a piece of specific information a command needs to know. Parameters are entered into ***dialogue boxes***, which appear whenever needed. Most dialog boxes require several pieces of information each having its own entry field. You can move between these fields by clicking in the required field with the left-hand mouse button or alternatively you can press the TAB key on the keyboard to move to the next field. This is especially useful when entering several pieces of data into a single dialog box. The TAB key is usually located above the caps lock key on the extreme left of the keyboard. It sometimes has the word TAB printed on it or alternatively two opposing arrows.

Note that when typing into a dialog box the mouse cursor must be positioned somewhere within the dialog box.



| | |
|---|---|
| texture name | flowers |
| material name | wood |
| number of replications crosswise | 1 |
| lengthwise | 1 |
| adaptive mapping | [✓ YES] NO |
| transparent background | YES [✓ NO] |

There are three kinds of parameters:

1. ***Numeric parameters*** (*e.g.*, 1.0, 5, .334)
2. ***Text parameters*** (*e.g.*, a name)
3. ***Alternative parameters*** (*e.g.*, yes/no)

Each of the parameters has a ***default value***, which is already selected when the dialogue box appears. If this default value is correct, you don't need to do anything more.

Clicking on the old value with the left mouse button and entering the new value may change numeric or text parameters. They can also be changed just partially by ***double-clicking*** at the area you want to change and typing in the new data.

You may enter ***real number parameters*** with or without the integer or fractional part. For example, a zero can be entered as 0, 0.0, 0., or .0 (but not just as “.”).

For ***text parameters***, use only digits 0–9, letters a–z and A–Z. The space character is automatically converted to an underscore. Avoid using special characters such as ; > % etc., as they could cause confusion with the operating system.

You may ***delete a character*** by using the `Del` or `BackSpace` (←) key.

When all parameters have been entered, close the dialogue box by clicking on the `OK` button at the bottom of the box, or by pressing the Return (↵) key.

If while entering the parameters you decide you don't want to perform the command after all, close the dialogue box by clicking on the `CANCEL` button instead.

## *Units*

DESKARTES represents all geometry in ***units***. Their interpretation is left to the user. If you are modeling a reasonably small object, you might interpret the units as millimeters, but in case of a larger model centimeters or even meters might apply better.

If you ***transfer data*** to or from other systems, many data transfer file formats, e.g. VDA-FS and STL assume the units to be ***millimeters***. However, an IGES file may also define the model in ***inches***. DESKARTES will automatically recognize the units when IGES files are read in.

***Measurements*** are usually displayed to the thousandth of a unit, ***e.g.***, 12.340 means 12 whole units and 34 hundredths. ***Angles*** for rotations and other such operations are given in degrees (not radians).

## *Command Series*

A ***command series*** may be used to “teach” the system a series of commands (a “lesson”) that can then be automatically repeated any number of times. You should use a lesson whenever you notice the modeling task would require you to perform the same operations repeatedly.

The command `Settings⇒ TEACH` tells DESKARTES to remember the following commands and editing operations. To end the lesson, click at `Settings⇒ TEACH` again. Afterwards, the lesson may be automatically executed any given number of times, with the `Settings⇒ EXEC` command.

## *Problems?*

DESKARTES has been designed to be as tolerant as possible against any kinds of errors. The system recognizes illegal user actions, tells what is wrong with them, and instructs how to proceed.

Entering several commands at the same time could confuse the model, so it is automatically prevented by the system. However, it is good to keep in mind the following ground rules:

- *Most commands cannot be interrupted*. Once you have started a command or an editing function, *wait for the command to be completed*. After the command is done, you may cancel it as explained below.
- *Don’t try to execute several commands at once*. The system won’t accept new commands while executing another one, but you might get it confused when trying to do so.

If anything goes wrong, you will always have a way to recover:

- Changes made in edit modes may always be canceled by exiting the edit mode by clicking the ‘Sorry’ icon (the **`q`** function). In most edit modes, it is possible to cancel individual actions, as well, with the **`u`** function.
- Menu commands may be canceled by choosing the command `Settings⇒ UNDO` immediately after execution. This does not apply to visualization and file commands, however. Also note that you can only undo the last menu command in this way.
- An annoying source of errors when using shortcut keys is the `Caps Lock` key. It locks the keyboard into UPPERCASE mode. The program responds to this by executing keyboard shortcuts as capital letters and so appears to behave irregularly, or not execute the functions at all. To solve this problem, just press `Caps Lock` again. Most keyboards have a small light which is lit when the `Caps Lock` key is pressed

- In more serious error situations the system may ***crash***, or get into a state where it does not seem to be doing anything (a so called "***infinite loop***"). The cause may be memory problems, user errors the system does not expect, or program errors.

  If the system becomes hung in a loop, you must ***kill it***. How this is done depends on the workstation you are using:

  - *On a UNIX/MACOS machine*, locate the window you launched DESKARTES from, move the mouse to that window, and press `Ctrl-C` (hold down the Ctrl key and press C).

  - *On a Windows machine*, press `Ctrl-Alt-Del` to provoke the Windows Task Manager. Select the processes tab. Terminate the process EXCEED.exe. This should remove the DeskArtes windows from the screen. If this does not happen, terminate also the processes hwm.exe, CubiX.exe, EXTrace.exe and shellemul.exe. Finally launch the HWM Window Manager and the DESKARTES software again.

- Even in the case of serious errors, your model won't be completely lost. To ***get back*** to where you left off when the system crashed or was killed, launch DESKARTES again, immediately select the model directory you used, and give the command `Settings`⇒ `UNDO`. You will be returned to the state immediately before the last command. You may now try to repeat the previous command. If it succeeds all is well. If it fails again and you can find no explanation for the failure, you may like to report this to your DESKARTES dealer.

- Sometimes your ***hard disk*** may be so full that file operations fail. To recover, you must ***archive*** or ***delete*** unnecessary files. The File Window provides you with all the commands you need for examining and cleaning up the DESKARTES database.

- ***RAM memory*** problems are usually indicated by system error messages that will be written to the window from which you launched DESKARTES. They may sometimes be resolved by simply restarting the program and going on where you left off. If this does not help, the model or image may just be too large for the ***swap space*** allocated for your workstation. In this case, please ask for assistance from your workstation dealer. On Windows systems the default is to automatically reserve hard disk space setting for swap space (***virtual memory, paging file***). You may wish to change this setting to pre-allocated files to avoid unexpectedly running out of swap space; use the Windows Control Panel → System → Advanced to change the settings, or consult your system administrator.

# 2 A Typical Modeling Session

Whenever you use DESKARTES for developing a new product, you will follow the same basic procedure. The different stages of this process are covered here. The concepts are explained and some commands may be mentioned briefly. This section is not intended to explain all the commands you will use. Command explanation is covered in later sections of this manual.

## Files

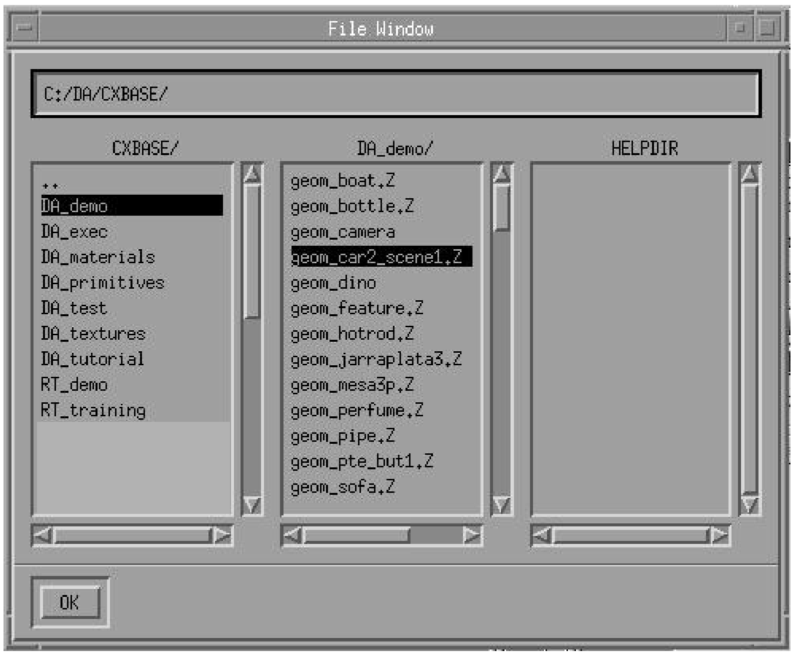
### File Window

All of the information stored on a computer is saved on a hard disk inside the box of the computer. To help you to locate the information when you need it you can use a series of ***directories*** (sometimes called folders) to segregate your work. Each piece of information actually saved, whether it is a geometric model or a picture is stored in a ***file***.

This structure can be very confusing to a computer newcomer but a simple analogy to a paper based office may make it easier to understand.

In an office, most information is stored in a filing cabinet for security and ease of location. Each filing cabinet may have several drawers. To further separate the information, to make it easier to find, each draw of the cabinet will have dividers. Inside each divider will be kept the pieces of paper on which the information (letters, memos, drawings, sketches etc.) is written.

Can you see the analogy? The filing cabinet is the computer. The draws are the disk drives in the computer (there may be one or several disk drives). The dividers are the directories and the pieces of paper are called files.



In general, DESKARTES simplifies this structure for you. You only need to worry about files and directories. The directory names can be seen in the left-hand pane (titled DATABASE) of the ***File Window***. The first thing you always do after launching DESKARTES is select or create the model directory where you wish to work. The file names stored in that directory then appear in the middle pane.

## *File Window Commands*

The File Window contains many ways to manipulate directories and files, such as creating and deleting directories, moving files from one place to another, etc. These commands are contained in ***pop-up menus*** accessed with the right mouse button. Each pane of the File Window field has its own pop-up menu so it is important to position your mouse over the appropriate pane before pressing the mouse button.

## *Selecting a Directory*

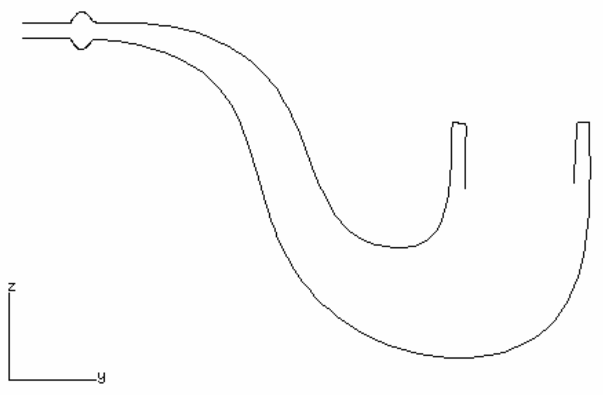
You can choose a directory from the File Window by positioning the cursor over the name in the left-hand pane and then clicking (pressing and releasing) the left-hand mouse button. You may wish to create a new directory instead. This is done by using the `File⇒ Directory`: `Create` command, from the directory field pop-up menu (remember the right hand mouse button access this command).

After selecting a directory, the File Window may be hidden. To do this, click on the **OK** button in the lower left corner of the File Window. Now DESKARTES is ready to accept modeling commands.

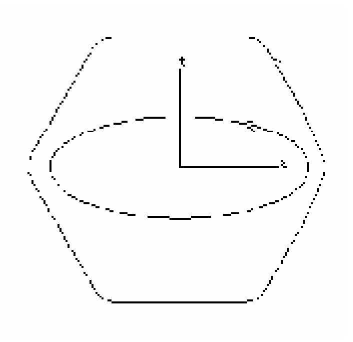
## *Modeling Concepts*

Since the computer screen on which you are working is only two-dimensional, it can prove difficult to interact with three-dimensional shapes. To avoid this problem, DESKARTES has been designed to, wherever possible, use two-dimensional curves to define the shape of the three-dimensional surfaces. This makes the surface modeling process a lot easier than direct interaction and surface manipulation in 3D, although these functions are also available within DESKARTES.

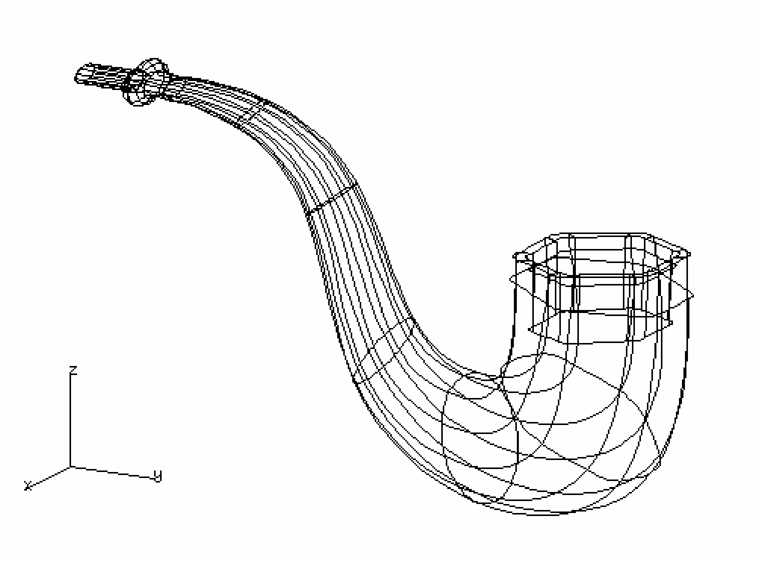
When creating surfaces, the shape of each surface is determined by one or more ***projection curves***. The projection curves define the surface's side view from the chosen direction of the three-dimensional space.



The projection curves are combined with ***cross-section curves*** to fully define the shape of the surface. You can have a single cross-section curve for a surface, or assign varying cross-sections to different positions of the surface.



The combination of projection curve and cross-section curves creates a three-dimensional framework over which DESKARTES stretches a 'skin' or surface.



## *Object Types*

DESKARTES recognizes three kinds of ***two-dimensional (2D)*** objects:

- ***Bézier curves,*** which smoothly pass through given control points.
- ***B-spline curves,*** smooth curves that approximate to given control points.
- ***Polygons,*** linear curves that consist of straight-line segments.

The type of curve used is generally the choice of the designer. However, there are certain instances when a particular type of curve is required by DESKARTES. These occasions will be highlighted in this manual. Most designers prefer to use B-spline curves whenever possible because with a little experience of their use they are the most natural type of curve.

Curves and polygons are used to create ***three-dimensional (3D)*** objects, ***i.e.*** surfaces. The surface types are:

- ***B-spline surfaces*** created from B-spline curves;
- ***Bézier surfaces*** created from Bézier curves;
- ***Faceted models*** created from polygons, or by converting from other surface types.

***Attribute objects*** are objects that have no function alone, but are associated to another object usually a surface. There are three kinds of attribute objects:

- ***Trim curves*** which are used to cut away parts of surfaces,

- ***Texture areas*** which position textures on a surfaces,
- ***Materials*** to define the appearance of the surface when rendered.

The correct use and organization of these objects is obviously very important when modeling. DESKARTES requires that you conform to the correct structure in all models. This structure is described in the next section.

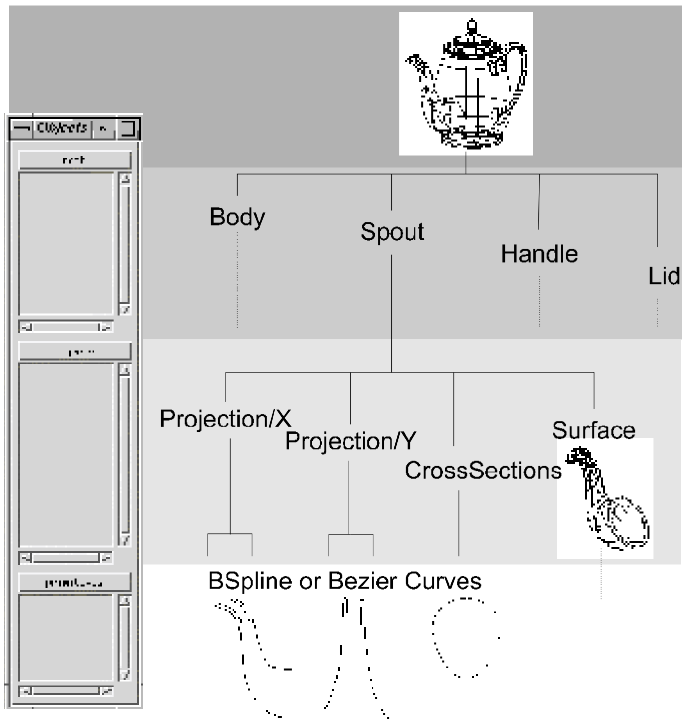
## *Dealing with Objects*

### *Object Hierarchy and Object Window*

In any modeling situation, the system contains several types of geometric information. This geometric information is organized as a ***four-level hierarchy***, where each level may only contain certain types of objects. This four level hierarchy is shown in the following diagram along with the Object window, which is used to interact with this structure. The diagram shows a model of a coffeepot.

The top level is called the ***root.*** It contains all the geometry contained in the system at the current time, in this case the complete coffeepot.

Below the root level, the model is divided into ***elements.*** Each element is created with the menu command `OBJECT⇒ New` and has a user-given name. An element can be considered as all of the information required to define a single part of the model. The first task of the designer when defining any new model is to decide how to break down the model. In this example, the designer has chosen to use four elements the Body, Spout, Handle and Lid.



Typically, an element contains all the ***geometry to define a surface***. This information is divided into different curve sets that include projection curves and section curves. Each curve set is created with the menu command `OBJECT⇒ New.` The Spout, for example has three curve sets – X and Y projections and cross-sections. This level also includes the surface itself, and its attribute objects. An element may also contain several surfaces. For instance, the command `OBJECT⇒ Copy` may be used to make a copy of a surface that can then be moved to a new position. If the example model was not a coffeepot but an Urn, with two identical handles, the first handle could be copied and the moved to the opposite side of the body. The Handle element would then contain two surfaces.

The lowest level of the object hierarchy contains the ***primitive objects.*** Individual curves or polygons are never stored at the same level as the curve sets (i.e. directly under “elements”). Instead, they are always stored at the lowest level and associated to one of the curve sets, either a projection or cross-section curve set. The curves are usually created with a command such `CURVE⇒ Design: Input`. The Spout for example, has five curves – two in each of the X and Y projections and a single cross-section curve. Surfaces also have their basic primitive control curves. However, they are never manipulated directly from the object window. They are only ever manipulated with higher-level commands without directly dealing with the primitives themselves.

## Target Object

At every stage of modeling, one object in the hierarchy is selected as the ***target object.*** The next command will apply to this object. It can be the root, an element, or some lower-level object. The target object is highlighted with a black background in the Object Window.

For example, if the *surface* in the Spout has been chosen to be the target object and the command `OBJECT⇒ Delete` is executed, the selected surface (and any items below it in the hierarchy) and will be deleted. The command will not affect the projections or section curves defining the deleted surface since they are at the same level.

If however the Spout ***element*** had been chosen to be the target object and the command `OBJECT⇒ Delete` executed, the entire Spout element will be deleted.

## Selecting a Target Object

You may ***select a target object*** by clicking on any object visible in the object window. To choose “root” for target object, click at the ***title bar*** of the elements field on the word root.

Instead of operating with the Object Window, you may also select an object by ***graphical picking***. Use the command `SELECT⇒ Object: Pick` and point at any object you see on the screen.

Any command that creates a new object also makes it the target object. For example, as the command `SURFACE⇒ Design: Rotate` creates a surface it becomes the new target object.

It is important to choose the correct target object for a command. For example, if a set of section curves is the target object, any command will affect all of the curves in the set. If you wish to alter just one of the section curves, make it the target object, in the lowest window, first.

When an item is selected, information relating to it is displayed in the message lines below the menu.

## Working With Curves

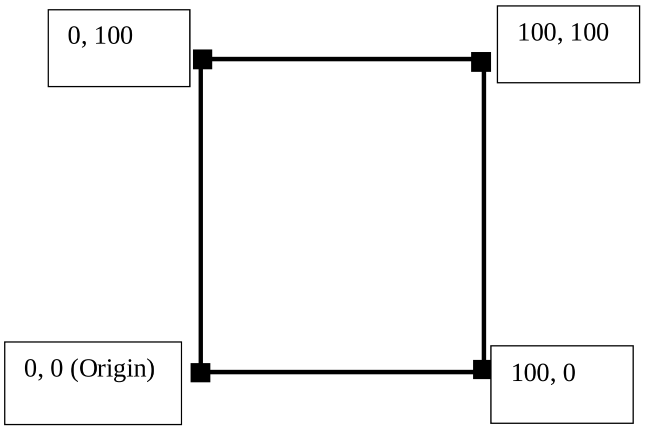
## Curve Coordinates

Curves are represented in a ***planar coordinate system*** that you can think of as being like a very large piece of graph paper. Just like plotting points onto graph paper, the points that control the curve are defined by a horizontal and a vertical distance from an ***origin.*** The origin is located at the center of the ‘page’.

Normally you will use the mouse to position your points visually, but on occasions you may need to type in the coordinates. When typing in coordinates the distance is normally entered relative to the origin. The horizontal distance is entered first followed by the vertical distance.

If the point is to the right of the origin, then the horizontal coordinate is a positive value. If the point is to the left, then the value should be entered as a negative number with a minus sign (-) immediately in front of the number. Similarly for vertical coordinates where points above the origin are positive and below are negative.

As an example of this, if you were to draw four points forming the corner of a 100x100 square with the bottom left corner at the origin then the coordinates you type in would be as follows….



If you are using the mouse to enter points then there is a special way to ***display the 2D coordinates*** of any point on the screen. Press the `Control` key, and the current cursor position will be shown in the message lines. The cursor value changes continuously as you move the cursor.

## *Grid*

To assist with locating coordinate points, a ***grid*** is available with the `Settings`⇒ `Grid` buttons. It displays guidelines at a chosen spacing, which help you to locate the cursor and to ***snap*** the input points onto the grid. With three-dimensional objects, the grid is shown as a ***floor*** on the xy-plane.

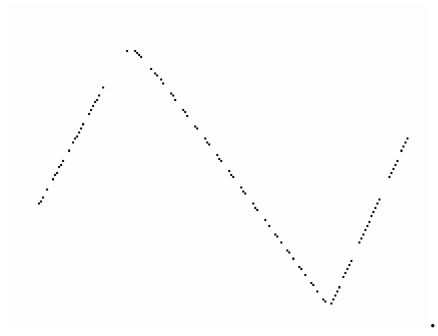
## *Curve Representations*

The curve objects are typically created and manipulated with the commands in the `CURVE` menu. The representation is chosen by choosing the appropriate ***input mode*** with the `Settings`⇒ `REPS` buttons. The possible input types are `B-spline`, `Bézier`, and `Linear`.

## *Linear*

Linear curves consisting of straight-line segments are called ***Polylines or Polygons*** (which form a closed loop). From this description, you may think that you will want to use this type of input a lot, whenever you need flat areas in your model. This is not the case! There are better ways of making part of a curve straight. In fact you will hardly ever use Linear input – it is really only available for internal use within the software.

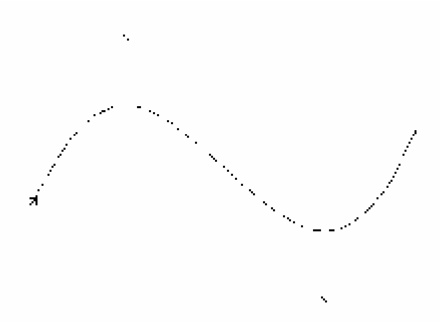
Polygons are created with the `CURVE` menu commands when the input type has been set to `Linear`.



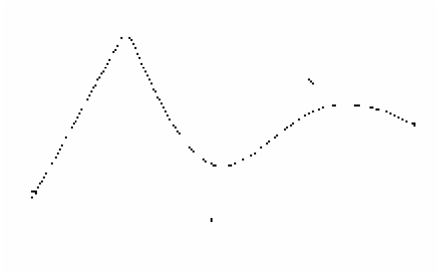
## *B-Spline Curves*

***B-spline curves*** follow the general shape of a ***control polygon***, the vertices of which are known as ***control points.*** These control points act like magnets which pull the curve towards them. With a little practice, you will find this a very natural way of controlling the shape of a curve.

To interactively create a B-spline curve, set the input mode to `B-spline`, and use the command `CURVE`⇒ `Design`: `Input` to enter the control polygon.



To move, add, or delete control points, enter the command `CURVE`⇒ `Design`: `Edit`. You can also produce sharp angles, straight-line segments and circular arcs etc. In other words, practically any curve shape may be defined by a B-Spline using the available curve editing functions.



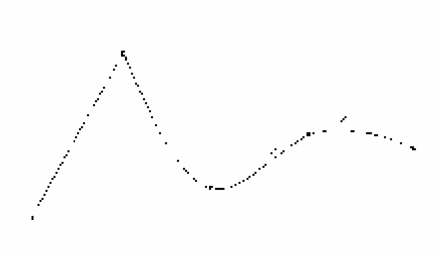
It is good to know the following facts about the B-spline curve properties:

- The curve follows the form of the control polygon smoothly, but it does not usually pass through the control points, it ***approximates*** them.
- Generally, a curve will be smooth if it does not have too many control points, and if the points are distributed in an even manner along the curve. Adding too many points will not only make DESKARTES work harder but is likely to produce a curve with small fluctuations as you have to be more precise in the positioning of the points.
- Moving a single control point only affects the part of the curve that is close to the point. The rest of the curve remains unchanged.
- You can create a ***double control point*** (two control points at the same location). This has the effect of pulling the curve much closer to the point. You have effectively doubled the strength of the 'magnet'. This produces a tight, smooth corner at the control point.
- You can create a ***triple control point*** (three control points at the same location). This has the effect of pulling the curve through the point producing a sharp corner at the control point.
- The ***endpoints*** of an open curve are automatically made triple. The last three control points of a closed curve (one which forms a closed loop) are the same as the first three. These facts are usually hidden from the user, but useful to know.
- The curve always leaves a triple point (such as an endpoint) in the direction of the next control point.

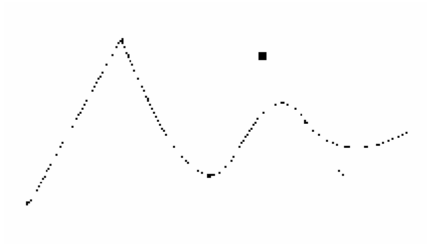
## *Bézier Curves*

The ***Bézier curve*** is another way of representing smooth curves. As with B-spline curves, Bézier curves are input with the command `CURVE⇒ Design: Input`, but first the `INPUT` type has to be set to `Bézier`.

When inputting a Bézier curve the curve will pass through all of the control points. Each control point has two ***handle points***. These control the curvature at the point.



With the command `CURVE⇒ Design: Edit` you may move the Bézier control points and handles, add new control points without changing the shape of the curve, define arcs and fillets, and further manipulate the curve shape.



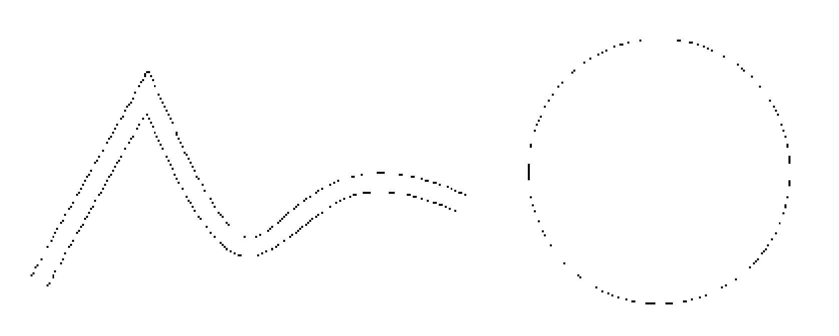
Whether to use B-spline or Bézier curves is largely a matter of application and taste although most designers find B-spline curve more natural to use. B-spline curves have fewer points and allow for easier sketching, while Bézier curves give more control for detailed work.

However, when designing a surface using the most important Build command, the projection curves must be B-spline curves. Thus, it is possible to do all your designs with B-spline curves, while Bézier modeling is not generally as applicable. When starting to learn DESKARTES, it may be best not to change the input type at all, and work just with B-spline curves. Bézier curves may be added to your repertoire later.

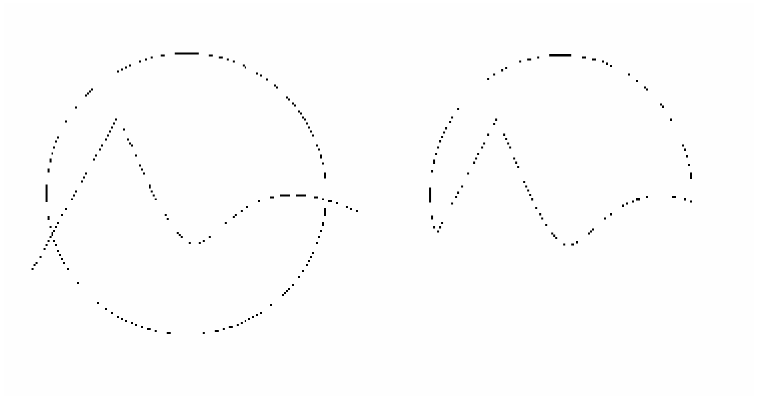
## *Curve Modeling*

The curve objects are created and manipulated with the commands in the `CURVE` menu. To create a curve, the target object must be another curve, or a curve set.

Curves are entered interactively with the commands `CURVE⇒ Design: Input` and edited with `CURVE⇒ Design: Edit`. Other commands that create curves are `CURVE⇒ Design: Polygon`, `Circle`, `Arc`, `Offset`, and `Spiral`. These commands need further user interaction such as the desired number of points, circle radius, offset distance, etc.



Curves may be ***intersected*** with each other, as well as ***combined*** into one. These are done with the commands `CURVE`⇒ `Combine`: `Cut and Join`.



A curve may also be designed using ***help curves*** as templates. Help curves define geometry that is not directly used for modeling, such as digitized curves, circles and straight lines. These can be used as a template over which the actual curves are designed.

## *2D Transformations*

Once you have created an object, you may change its size, location and shape using different kinds of ***transformations***.

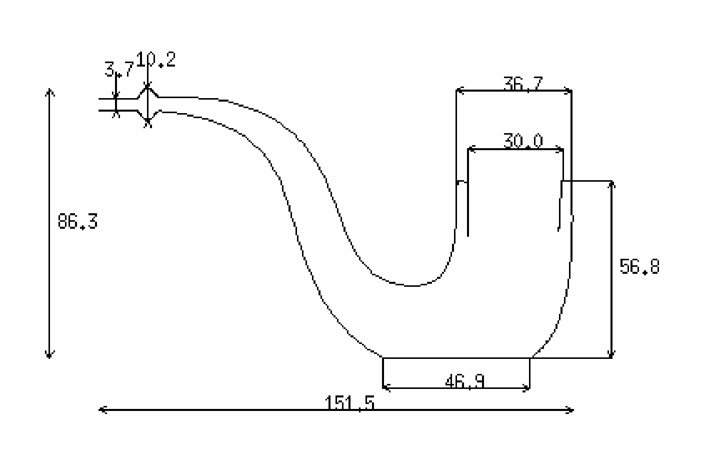
The standard transformation operations are ***moving***, ***scaling***, ***rotating***, and ***mirroring*** an object. The point around which the transformations are performed is called the ***fix point***, which can be changed as necessary.

Transformations may be controlled by numerical values, or with graphical interaction to move, scale and rotate the objects with the mouse.

## *Dimensions*

The `DIMENS` menu contains commands for finding out information about the objects ***size***.

The command `DIMENS`⇒ `Curve`: `Dimensions` leads to an edit mode where you may specify the display of various local dimensions of the curves, such as distances between points on the curves, circle arc radii, angles, etc. Remember that these dimensions report the size of the object you have drawn – they cannot be used to change the size of the curve.

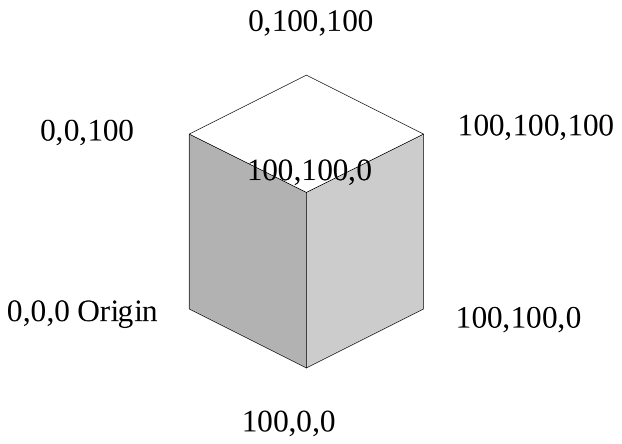


## Working With Surfaces

### Surface Coordinates

Three-dimensional objects are defined within a system of ***spatial coordinates.*** Spatial coordinates are like using three-dimensional graph paper! Since using a mouse for such coordinates is very difficult, special consideration must be given for mouse input. This will be covered elsewhere in this manual. For typed input, again the distance is measured from the origin, but now a third distance needs to be entered. The distances are always called X, Y and Z-axes. The best way to visualize these axes directions is to consider your model sitting on a floor. The X and Y directions lie on the floor and the Z direction is the height above or below the floor. The origin is shown in DESKARTES by a special marker that also indicates the positive direction for each of the axes.

Using an example of a box, if you were to draw eight points forming the corner of a 100x100x100 box with one corner at the origin then the coordinates you type in would be as follows….



The coordinate of the hidden corner is not shown in this diagram. Can you work out what it should be and why?

## Surface Representations

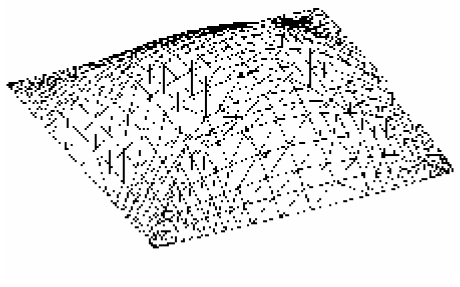
Much like curve objects, there are three different representations for surfaces: faceted models, B-spline surfaces, and Bézier surfaces. The surfaces are created with the SURFACE menu commands. The type of curve used to define the surface determines the resulting surface representation.

## *Faceted models*

***Faceted models*** are surfaces made of polygonal ***facets***. They may be created by extruding or rotating a polygon. Most often, they are made by converting from a B-spline or Bézier surface.

One use for faceted models is a temporary representation for surfaces when computing areas or volumes. The required conversion is done with the command `SURFACE`⇒ `Change`: `Faceted Model`.

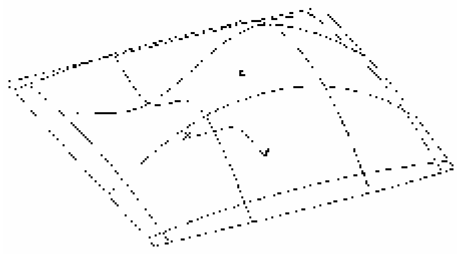
A second use is to transfer your surface model to another design system or manufacturing system, in particular rapid prototyping systems, which use the STL file format.



## *B-Spline Surfaces*

A B-spline surface is defined by a ***control mesh***. The surface follows it just like B-spline curves follow their control polygon. You may think of a B-spline surface as being made by stretching and shrinking an infinitely flexible rectangular ***rubber sheet*** over the control mesh.

The surface has two directions, called the ***cross-wise*** and ***lengthwise*** directions. The control mesh always has a number of control points in its two directions determined by the defining curves. You may shape the surface by moving control points, you can add an entire row of control points, but you can neither add nor remove a single point at a time. In other words, the B-spline surface is composed of a rectangular array of ***surface patches***.

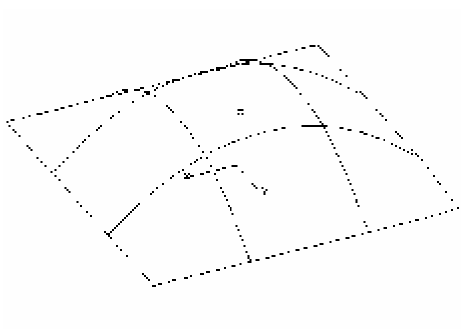


The B-spline surface is able to assume a variety of shapes that do not appear to resemble the rectangular patch structure at all. For instance, a sphere may be made and the ‘rectangular’ patches become like lines of longitude and latitude.

## *Bézier Surfaces*

The properties of Bézier surfaces, such as the rectangular patch structure, are quite similar to those of B-spline surfaces. The Bézier surfaces are created in the same way as B-spline surfaces, using one of the `SURFACE` menu commands to produce the surface according to the type of projection and section curves.

The main difference of B-spline and Bézier surfaces is with the local editing functions. Unless you wish to edit Bézier surfaces locally, you won’t have to worry about the difference between the two representations.



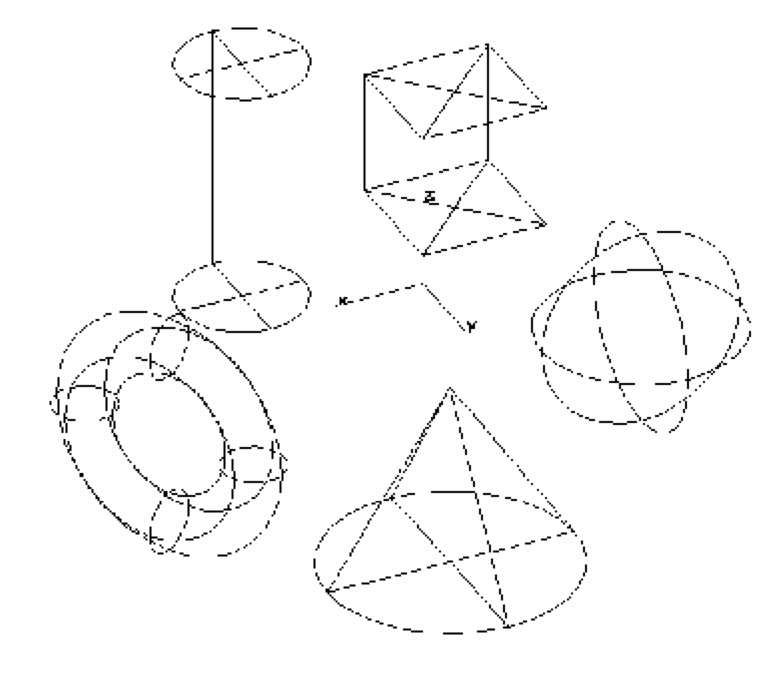
Some demanding computations, such as surface intersections and blends, require the surfaces to be in Bézier form. When necessary, DESKARTES automatically converts the surfaces from B-spline to Bézier representation.

## Surface Modeling

There are many ways of creating and manipulating surfaces in DESKARTES. Only the most common are described briefly here.

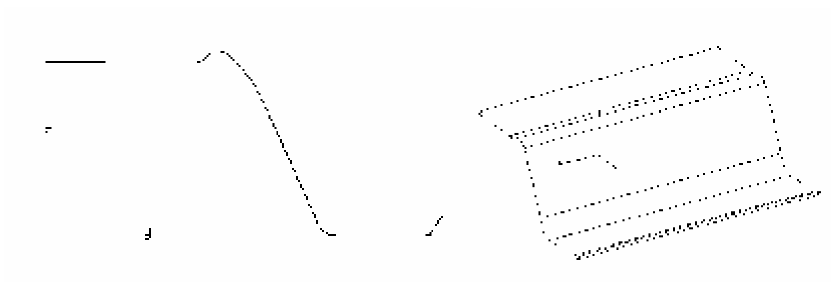
### Primitive Surfaces

The simplest means of creating a surface is by selecting one of the predefined ***primitive surfaces***. The different surface primitive types available are plane, box, sphere, cylinder, cone and torus. They are all created with the command `SURFACE⇒ Design: Primitive`, and choosing the primitive type, size, etc., as parameters.

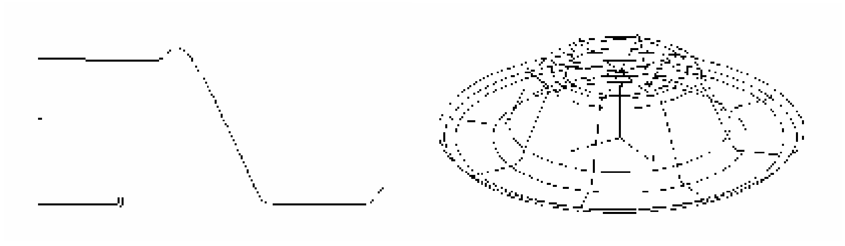


### Extruded and Rotational Surfaces

The commands `SURFACE⇒ Design: Extrude` and `Rotate` produce different kinds of ***sweep*** surfaces. The `Extrude` command sweeps a projection curve a given distance in one of the coordinate directions.

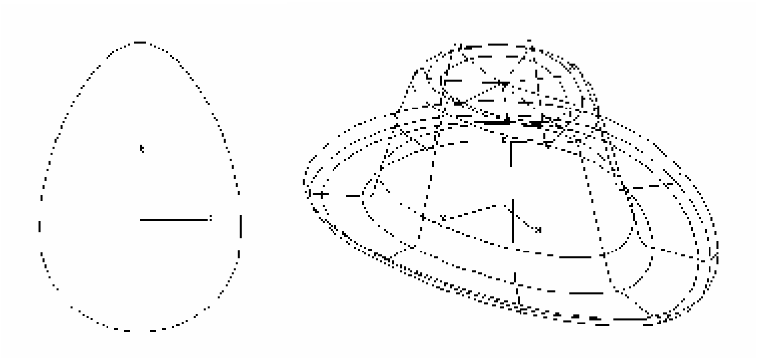


The `Rotate` Command ***revolves*** the projection curve around an axis.



Both commands require that at least one curve is defined in a projection curve set in the target element. If the `Extrude` command is given several curves in the same projection element then several surfaces will be created.

Command `SURFACE`⇒ `Design`: `Rotate` may also be given a single cross section curve, to produce a surface by ***free-form rotation.*** The projection curve will then be rotated around this cross section curve to produce the surface.



## *Building a Surface*

The command `BUILD`⇒ `Create`: `Surface` is the most flexible way to create surfaces. You may use it in several different ways, to obtain different results. It is also the most commonly used command so it deserves a more thorough explanation than the other surface creation commands.

Building requires projection curves and cross section curves. The combination of curves determines exactly what the outcome of the command will be.

## *Single Projection Building*

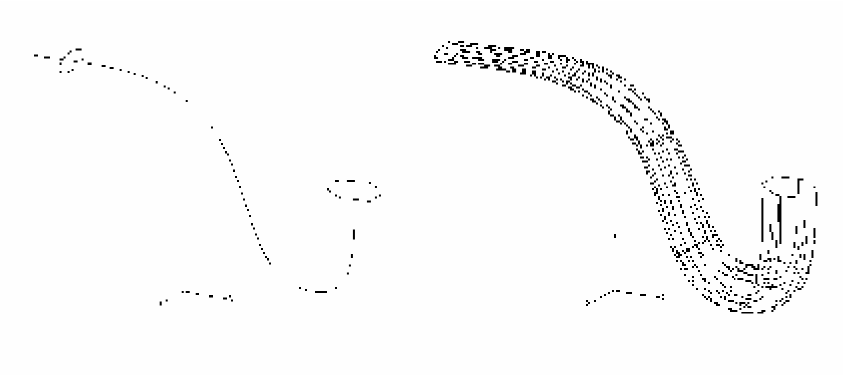
Building may be used with just one projection curve. In this case, the projection curve defines the central ***spine*** of the surface.

The ***cross-sectional*** shape of the surface may be defined by just one section curve, or several section curves assigned to different positions along the surface. If you have just one section curve defined, it will be used for all the cross-sections of the surface. ***Varying section curves*** are positioned along the surface by assigning them appropriate ***section numbers.***

To build the surface, DESKARTES combines the projection curve and section curves. The section curve *center points* (or *origins*) are located on the projection curve and the section curves are rotated so that they are perpendicular to the projection curve.

If you have chosen not to define all the cross section curves, DESKARTES internally calculates the intermediate sections. The cross sections between two defined sections will be a combination of these sections so that the surface will gradually change shape or morph.

The surface is then created by stretching a 'flexible sheet' over this framework of curves.

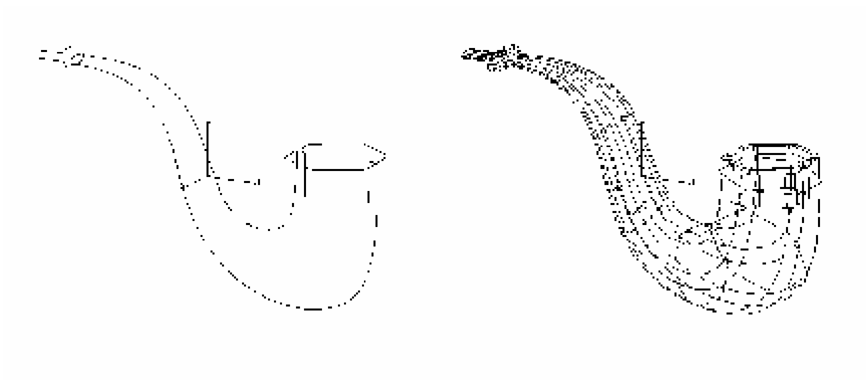


## *Double Projection Building*

Building may also be used with two projection curves. In this case, the projection curves define the extents of the surface when viewed from that axis direction.

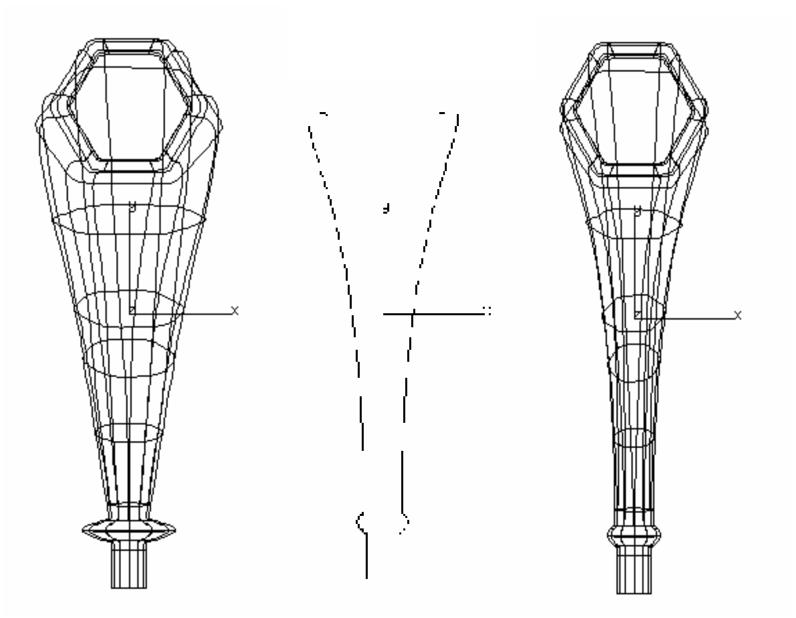
Just like building with single projections, double projection building can use one or several ***cross-sectional*** curves. This time however, DESKARTES combines the projection curve and section curves by locating the section curve **between** the projection curves, scaling them as necessary to fit.

The surface is then created by stretching a 'flexible sheet' over this framework of curves.



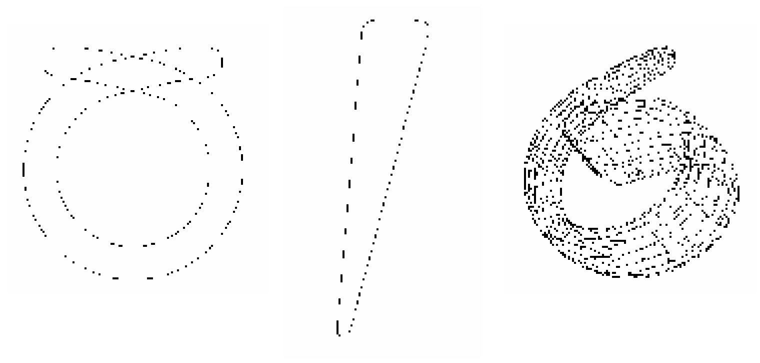
## *Secondary Projection Curves*

The `Build` command may be given ***two projection sets*** as its design input. The first projection set is called the ***primary projection*** and the second one is the ***secondary projection.*** These two projection sets are always defined from different directions. The secondary projections are created by the command `BUILD`⇒ `Create`: `Secondary Projection`. DESKARTES first generates a ***default form*** for the secondary projection, which will exactly describe the current shape of the surface when viewed from the chosen direction. The secondary projection can then be edited (within certain restrictions) adjusting the shape of the surface.



## *Length projections*

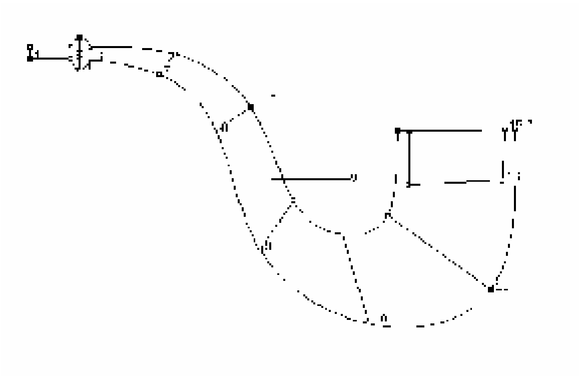
The ***length projections*** control the build surface shape similarly to other (X/Y/Z) secondary projections. However, they do not relate to any particular axis direction, but reflect the surface's shape as it would be spread out along its "length direction". Using length projections is often especially useful in case of "worm-like" objects, such as rings, handles, etc.



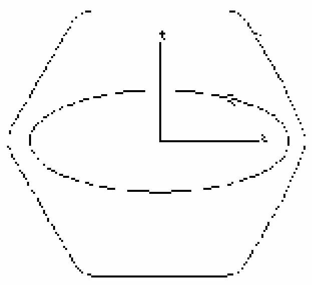
## *Rules about Building*

There are certain rules that must be obeyed when building surfaces:

- For certain mathematical reasons, *the projection curves must always be B-splines*. (Section curves may be either B-splines or Béziers.)
- All projection curves (including the secondary curves) must have an equal number of control points. Adding a control point to one projection curve means that a similar point must be added to all other projection curves.



- There does not need to be an equal number of points in each of the section curves. However, if sections have unequal number of points DESKARTES will convert them before building so that they do have the same number of points. This automatic conversion may not be ideal so you may prefer to create all sections with equal points. It is usually best to start modeling the section curves with the most complex then the simpler ones.
- In order to avoid twisted surfaces, all section curves must have their starting points aligned. Likewise, they must be oriented in the same direction (clockwise or counter clockwise).



- When editing a secondary projection, DESKARTES will only allow you to move the control points in one direction (the direction of the primary projection). Without this restriction it would be possible to have conflicting information between the primary and secondary projection curves.

## *Building with 3D Projections*

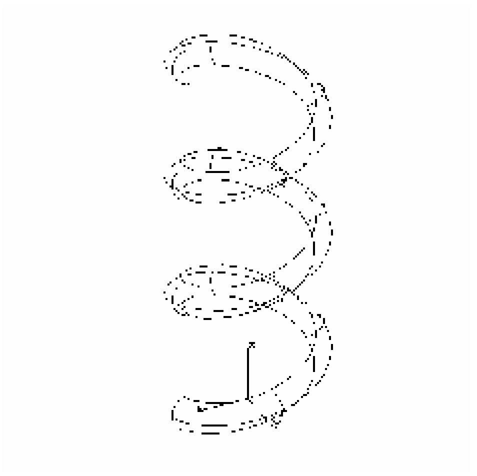
Building surfaces is also possible using ***3D projections*** and it works in a very similarly manner to the 2D case.

With 3D projections, the command `SURFACE`⇒ `Design`: `Build` creates the surface by transforming one or more 2D sections to match the 3D projection curve locations to make a wire framework and then ‘stretching’ a surface over this framework. The projection and section curves may be either B-splines or Bézier, but the resulting surface will always be created in the Bézier format.

The other BUILD menu commands, such as `Create`: `Cross-sections`, `Set`: `Build Parameters` and `Set`: `Section Parameters` work for 3D projections, too. However secondary projections do not have any meaning in the 3D case.

### *Single Projection 3D Building*

If there is just ***one 3D-projection curve*** defined, the cross-sections are positioned along it by attaching the ***cross-section origins,*** or alternatively their ***fix points***, at the projection curve, with the section curves perpendicular to the projection curve at that point.



The example above was produced by calling a 3D projection with CURVE⇒ `Design Spiral`, using a semi circle as a section curve, and executing BUILD⇒ `Create Surface`.

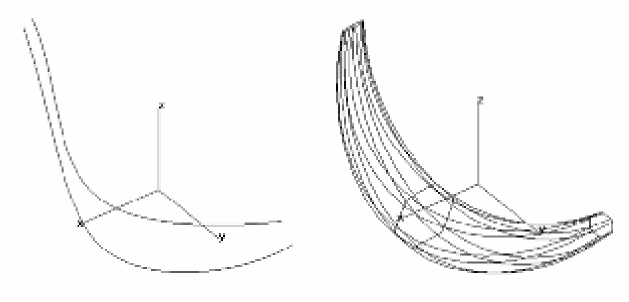
How the sections are rotated around the projection curve is determined with a special parameter, the ***orientation method.*** It has two alternatives: projective and principal normal. The projective method is generally the preferred method. The principal normal method should be used only in the case of very regularly shaped projection curves. The best example of this is perhaps a spiral curve, where the projective method may produce a slight twist to the surface.

There's another parameter with 3D building with one projection curve: ***section spin.*** It defines an angle of rotating the sections around their origin from one knot point to the other. The spin option is provoked only if the *section placement* in `Build Parameters` is set to *fix points*.

**Note**: sometimes the resulting surface may be "twisted", no matter which orientation method you use. The way to improve the result is to ***refine*** the 3D projection curve before building, by applying the curve edit function **`D`**.

## *Double Projection 3D Building*

With ***two 3D-projection curves***, *open section* curves are scaled between the projection curves by matching the section ***curve endpoints*** to the projection curve placement points. The command works also with *closed section* curves, by scaling the ***extreme points*** of the sections between the projection curves. In other words, it is a very straightforward generalization of how 2D building works.



## *Creating Curves for 3D Building*

DESKARTES provides various means to create and manipulate ***3D curves*** for building.

If you have selected a ***3D projection set***, the curves you create with CURVE⇒ `Design Input` are automatically taken as three-dimensional. You will then be able to ***input*** the curve's co-ordinates in ***three different views***. To further ***change the curve shape***, CURVE⇒ `Design Edit` works for 3D curves, too.

Alternatively, you may start by designing a curve first in 2D, and use `CURVE⇒ Change Dimension` to ***change it into 3D***.

Instead of local editing, the shape of 3D curves may also be changed by ***projecting*** them onto surfaces.

Perhaps the most flexible way to create 3D curves is to first obtain them by ***trimming***, as the intersection line between two surfaces, which can then be approximated in the required Bezier curve representation for 3D building (**Note**: use command `CURVE⇒ Change Approximate` for this, *not* `Change Representation` or `Interpolate` which would usually produce too many points).

## *Skinning*

The command `SURFACE⇒ Design Skin` interpolates a surface directly over a set of ***3D section curves***. It is one of the most general methods for creating surfaces with DESKARTES, although not always so easy to control.

The curves must be placed in a section set for the command to work, and they all must have an equal number of control points. When required, the number of control points can set equal using command `CURVE⇒ Change Approximate`.

## *Which Method to Use?*

Some of the most important decisions when making a model are made at the outset of the design process – how can the model be broken down into surfaces that are simple enough to manage?

Remember it is often easier to work with several simple surfaces than a few complex ones. If during the modeling process you find that this initial decision is wrong, don't be afraid to break the model down even further.

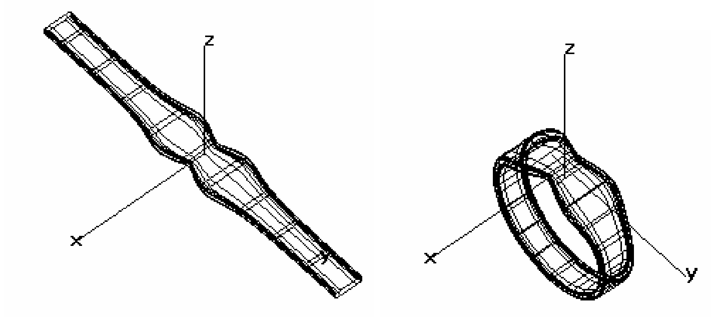
Having decided what surfaces are needed then each surface may be constructed in several different ways. Some ways may be difficult to achieve, while some can be very easy. In particular, the choice of projection and section directions is important when building a surface.

Before starting the design, carefully consider which coordinate direction works best for the primary projection, which one for the secondary projection, and which for section curves. As a rule, the most complex shape should be chosen for the primary projection whereas symmetry is good for secondary projections.

When building a surface with section curves that are substantially different from each other, it may be difficult to control what happens between the sections. The resulting surface may be wrinkled. This may imply that the surface is too complex and needs to be broken down further.

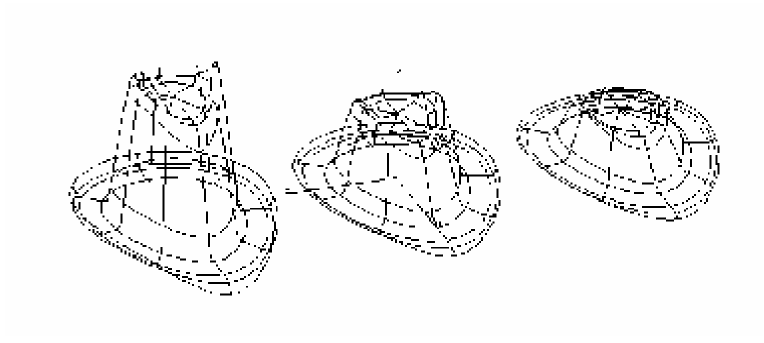
## *Deformations*

Further surface design methods are provided with the commands `SURFACE⇒ Deform: ...`, which provide for ***changing the shape*** of existing surfaces in various ways. The following picture shows the ***bending*** operation as an example.



## *3D Transformations*

Once you have created an object, you may change its size, location and shape using different kinds of ***transformations***. The standard transformations (moving, scaling, rotating, and mirroring) apply to surfaces as well as curves. If a model has repeated features, it is often useful to copy the surface using `OBJECT`⇒ `Copy` before moving or rotating.



## *Dimensions*

The `DIMENS` menu contains commands for finding out information about the objects ***size***. Dimension commands related to surfaces allow you to find the location of any point on a surface, as well as their area and volume.

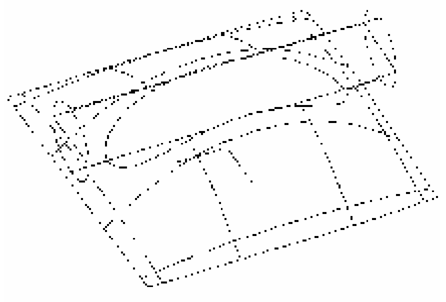
Remember, to calculate the area or volume of a surface(s) you must first convert to faceted representation using the command `SURFACE`⇒ `Change`: `Faceted Model`.

## *Combining Surfaces*

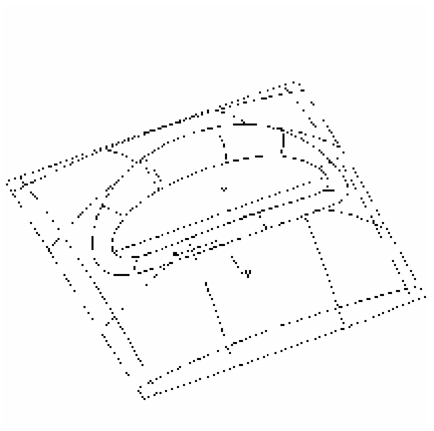
Usually, the entire model cannot be built of just one surface. Instead, it is better to build the model from several surfaces and match them together with the commands in the `TRIM` menu.

### *Intersecting and Blending*

The command `TRIM`⇒ `Intersect`: `Surface` computes an ***intersection*** – where two surfaces cross over each other - and ***cuts*** the superfluous parts away from the surfaces.



It is also possible to ***blend*** surfaces by creating an additional surface that meets smoothly with both the connected surfaces. The three blending commands are `TRIM`⇒ `Blend`: `Rolling Ball`, `TRIM`⇒ `Blend`: `Variable Radius` and `TRIM`⇒ `Blend`: `Free Form`. These commands give you unlimited control of the shape of the blend.



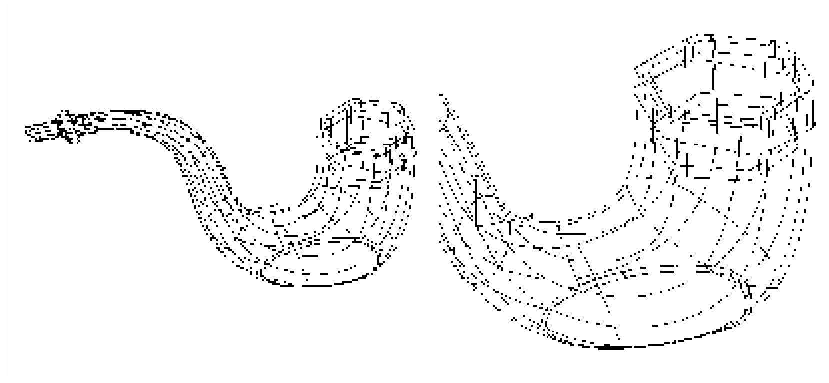
Both trimming and blending cut away part of the surface. This is recorded as an attribute of the surface that is indicated by the line "`->TRIM`" in the Object Window below the surface. Clicking on this will cause the trim curves to be displayed. If a surface is trimmed a second time DESKARTES will automatically combine the trim curves.

Building surfaces is the driving force of the DESKARTES system but intersections and blending are an equally important part of the modeling process. You should spend time understanding how to use these commands and how to use them as modeling tools.

## *Viewing*

### *Viewing Window*

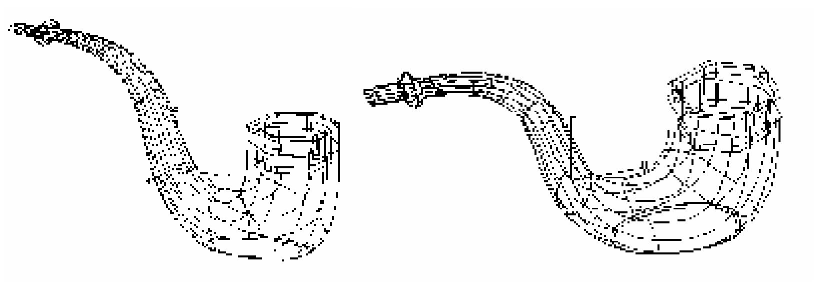
The ***viewing window*** determines which part of the model is shown on the screen. To control the viewing window, use the commands `DISPLAY`⇒ `View`: `Zoom` and `Pan`. By zooming, the objects are shown larger or smaller. By panning, you see different parts of the model without changing the scale.



Note that the viewing window commands have no effect whatsoever on the physical size of the geometry of the model. If, for example, you zoom in to look at the model more closely it appears bigger on the screen but the defined size of the model does not alter. The size of the model on the screen bears no relation to physical size of your model.

### *Eye Point*

To look at 3D objects from different angles, you may change the ***eye point.*** This is done with the command `DISPLAY`⇒ `View`: `Eye Point`. You'll enter a command state where the viewing direction can be chosen interactively with the mouse.



A quick way to place the eye point directly on one of the axis directions is provided with `Settings⇒ X/Y/Z`. To get back to the 3D view, click at `Settings⇒ 3D`.

## *Shaded / Wireframe Modes*

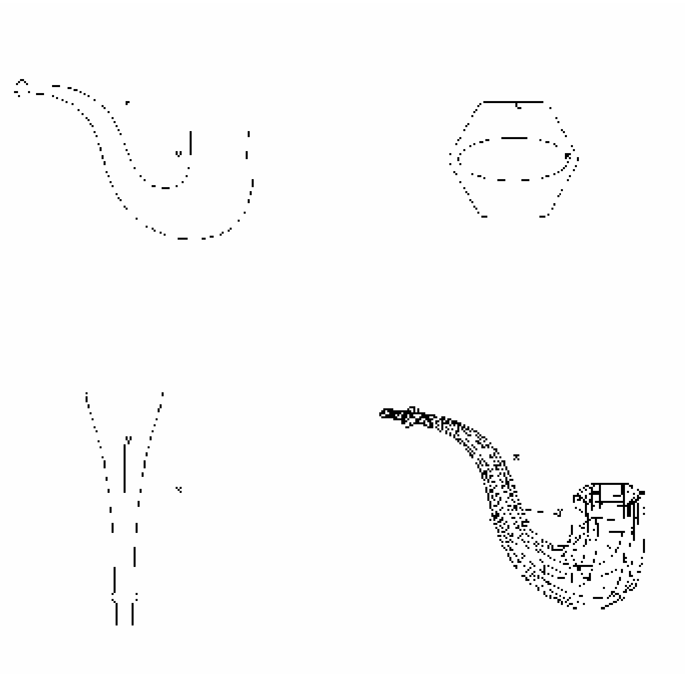
DESKARTES normally draws all surfaces on the screen as a wire frame display. If you choose to, you can work with a shaded display using the `MODE: Wirefr/Shaded/Both` buttons in the Settings Window. The speed of this display will depend directly on the hardware of your computer. If the model is complex or your machine too slow, it may be necessary to use the wire frame mode for most of the time.

When the object is drawn shaded for the first time it will take a little longer as the data for shading is generated. Subsequent displays will be much faster.

## *Display Areas*

In its default state, DESKARTES draws all objects in the ***display area*** that covers the entire graphics window.

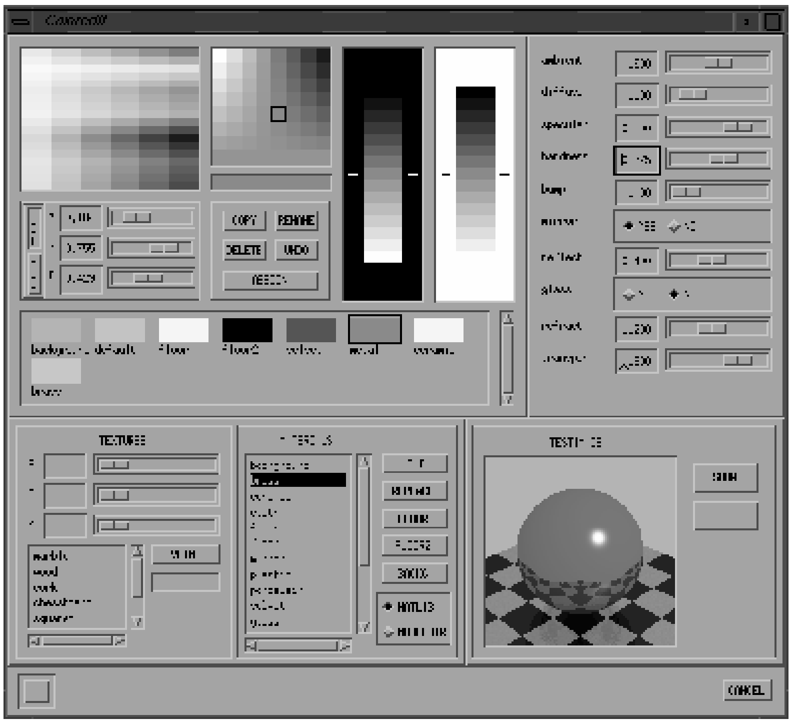
The `Settings⇒ AREA: Four` buttons allow you to display different kinds of objects in ***four viewports***, in the different ***quarters*** of the screen. Surfaces will be displayed in one viewport, and projections and section curves in the others. It also allows you to display and work with surfaces in all four ***orthogonal views*** simultaneously.



## *Materials*

### *Edit Materials*

The command `SCENE⇒ Edit: Materials` pops up the ***material palette window***. Using it, you may ***define*** the color and other material properties of the objects. You may define several materials and ***assign*** them to different objects.



When using the material palette window the material ***colors*** are chosen from the color palette, or by mixing any colors in the color-mixing table.

Other material properties affect the appearance of the surface of the material. These include things like *ambient, specular,* and *diffuse* reflectivity, *roughness* and *bump coefficients, mirroring,* and *transparency.* These may all be set with sliders. Different material appearances can be simulated by using various combinations of these parameters.

The palette also contains options for viewing and selecting predefined materials from the ***material library*** that will often already contain suitable materials for your models. The palette has a ***test image*** available for checking how the selected material properties will appear.

All of these material definitions are stored in a special “MAT” element that appears at the top level of the Object Manager window.

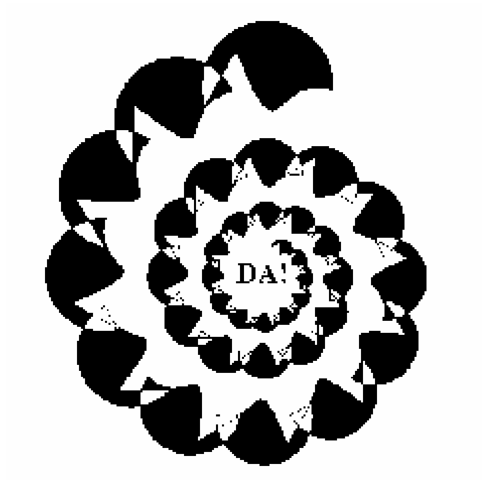
## *Textures*

In addition to material properties, surfaces may be decorated with ***graphic images***, such as logos, artwork, scanned images, bump maps and other effects such as marble or wood. All of these images are called ***textures.***

You may design the textures within DESKARTES by using the ***TextPaint*** program. Alternatively, scanned images that have been created with other, more sophisticated, programs may be used by DESKARTES if they are in either of the standard TIFF or BMP file formats.

Applying the pictures onto curved surfaces is called ***texture mapping***. Textures are defined, and mapped on surfaces, with the commands in the `TEXTURES` menu.

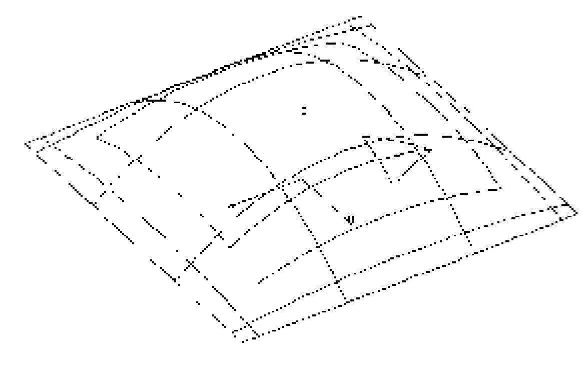
Textures are normally only shown in DESKARTES by an outline box. The command `RENDER⇒ GLWindow: Camera View` can be used to visualize textures but even this has some limitations (i.e. only one texture per surface). Only by using the ray-tracing program, ExTrace, can all textures be fully visualized.

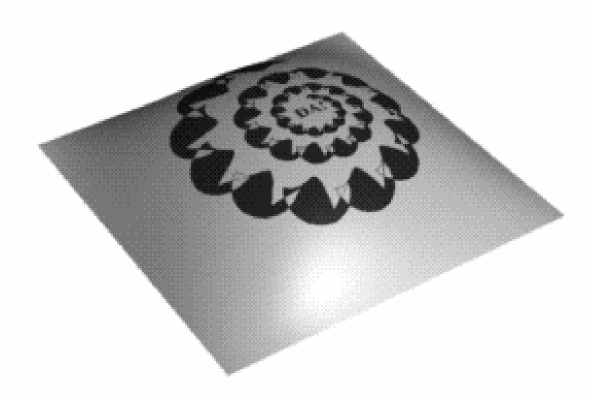


## *Texture Areas*

Textures are mapped onto a surface by rectangular ***texture areas***. A texture area describes which part of the surface will be texture mapped, as well as which texture will be applied to it and the application method. There can be several texture areas on the same surface, each with its own texture definitions.

The command `TEXTURE⇒ Area: Create New` creates a new texture area. You will be asked for the name of the texture, its placement on the surface, etc. The various parameters may be later changed with commands `TEXTURE⇒ Area: Parameters`, `Placement`, and others.

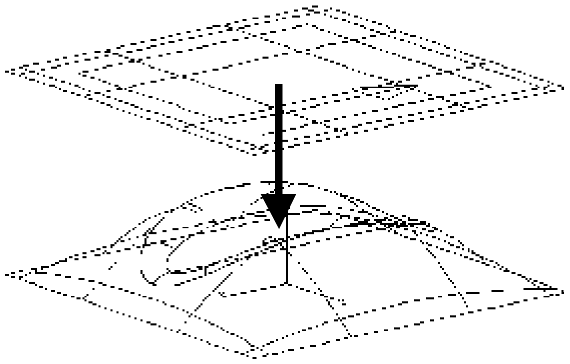


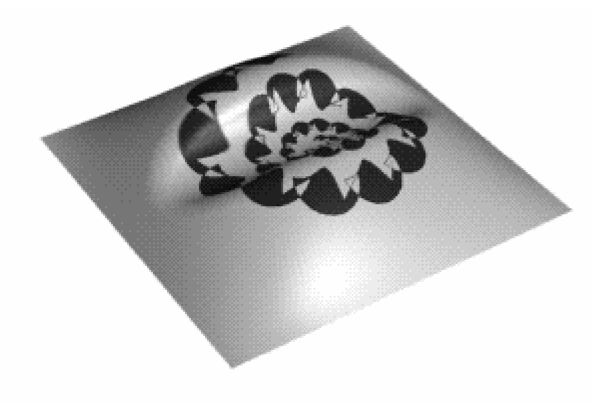


## *Bounding Surfaces*

Sometimes the desired texture mapping is difficult to define just in terms of the rectangular texture areas. The standard texture areas are always rectangular and are restricted by their orientation on the surface. Furthermore, spreading a single texture simultaneously across multiple surfaces is not possible using standard texture areas.

In such cases, an alternative method is provided. This requires you to first define the texture on a separate ***bounding surface***. The bounding surface is not part of the model and so will never be visualized. The texture will be visualized by being ***projected*** onto the desired surfaces.





## *Camera and Light Points*

As the equivalent to the eye point, which is used during modeling work, the ***camera point*** determines the view from which the ***visualization*** (or ***rendering***) programs compute shaded images.

The camera point location may be defined in two alternative ways. Command `SCENE`⇒ `Match`: `Camera to Eye` places the camera to match the current ***eye point viewing direction***.

Another way is using DESKARTES comprehensive ***scene editor***, with command `SCENE`⇒ `Edit`: `Lights`. There you can change the camera location and parameters (viewing angle, target, etc.) arbitrarily in 3D space using graphic interaction.

Moreover, the scene editor allows you to define various other scene components, such as the position, colors and intensities of ***lights*** and ***spotlights***. All of these scene parameters, light points etc., are stored in a special “SCENE” element that appears at the top level of the Object Manager window.

## Fast Shading

### Render Commands

DESKARTES uses a system called OpenGL to generate shaded views of your model. The speed of the shading will be greatly affected by the quality of the graphics card in your computer. A good quality graphics card will allow you to work in shaded mode all the time if you wish! The shading you normally see in the graphics window is the lowest quality and therefore fastest shading available in DESKARTES. To achieve this speed it does not use any of the material properties assigned to the objects. Even the color of the models is derived from element color and not the material.

Higher quality shading is available using the command `RENDER⇒ GLWindow`: `Shaded View.` This automatically renders all the objects that are currently shown on the screen from the eye view, using a single light, the defined colors, but not accounting for the material properties or textures. The shaded object is displayed in separate graphics window not in the main window. This window has a pop-up menu accessed by the right hand mouse button. This menu contains many useful commands for controlling the display of the model.

The command `RENDER⇒ GLWindow`: `Camera View` renders the model using all the scene definitions that can be shown with OpenGL and is therefore the highest quality rendering available.

The ***accuracy*** of the shading with both modes is controlled with the `RENDER⇒ Preferences` command. Rendering achieves high speed shading by first internally converting the your surface models into ***faceted*** models made out of many small triangles. The accuracy affects how fine the model is faceted. The higher the accuracy the smoother the model will appear but at the expense of triangulation and rendering speed.

## Ray Tracing

The highest quality pictures available in DESKARTES are calculated by a technique called ***ray tracing.*** DESKARTES provides one of the most efficient ray tracers on the market. It works directly with surfaces and can simulate shadows, reflections, transparency, complex textured surfaces, and all the other photo realistic components for the final presentation images.

Presentation images are usually generated at the final stage of the design cycle. The efficiency of the ray tracing module in DESKARTES means that the highest quality images can be used as a design tool as part of the design process.

### Interactive Refining

The ray trace module, called ExTrace, is started in its own window with the command `EXTRACE⇒ Start/Update`. The surfaces currently on the screen will be used to generate the picture.

The drawing is started at a ***low resolution*** and, unless interrupted, ***refined*** until the final resolution is reached. This technique is particularly effective because you can very quickly check that the picture is as you expected. You can interrupt the computation at any time, and make changes to the image if required.

## *Batch Processing*

It may take a little time to compute full resolution ray traced images. Therefore, when you are confident that the picture is as required, the final picture will be calculated as a ***background process***. A background process is a job that needs no user interaction and so the computer can work on it whenever it gets a spare moment. You can continue working on the computer though you may find a slight reduction in the responsiveness of the computer, particularly on Windows machines.

ExTrace will ask when batch processing should be begun, as well as the name and the desired size of the image. After being processed, the final ExTrace image is viewed by double clicking on the file in the file manager window.

## *Manufacturing*

All over the world, models designed with DESKARTES are routinely manufactured with CNC machining, without even making changes to the models in the manufacturing system.

However, the ability to manufacture a surface designed with DESKARTES, or the suitability of the DESKARTES system for a particular design purpose, may not be automatically guaranteed. As explained here, there are some limitations of the surfaces, which should be considered before starting a costly design process. If possible, use ***rapid prototyping*** or ***test machine*** your model first into a soft material before creating final tools in hard materials.

## *Numerical accuracy*

In order to economize on memory space, curves and surfaces within DESKARTES are stored with single precision floating point accuracy. Loss of accuracy may occur as projection and section curves are interpolated into surfaces. Transformation commands may cause some additional loss of accuracy, as well. The final accuracy is generally correct up to the hundredths of units, but it may require verification for some manufacturing purposes.

## Circular arcs

B-spline circular arcs are quite rough approximations unless a sufficient number of control points is used. Bézier arcs give better accuracy.

## Surface smoothness

The B-spline surfaces created with DESKARTES are always mathematically smooth. However, having a smooth surface does not automatically guarantee that it "looks nice". In particular, using too many control points on projections or widely differing section curves to build surfaces may produce fluctuations in the surfaces.

## Blends

The most demanding modeling operation in DESKARTES is the blending of two surfaces. In particular, computing a blend inside a surface corner which has a smaller radius than the rolling ball is mathematically not a well-defined case, and its computation requires some assumptions by the system. Consequently, the blend surface may be difficult to manufacture.

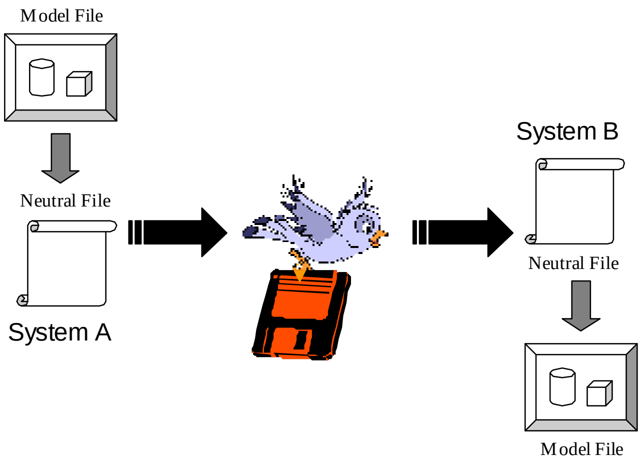
## Offsets and Fillets

Surface offsetting and filleting are mathematically difficult problems, and you should always be cautious about their results.

## Data Transfer

Like all CAD systems stores its information in a proprietary file format that other systems cannot read. DESKARTES does provide a number of ways for ***transferring*** geometric models and graphic images to and from other systems. DESKARTES can read and write DXF, IGES, VDA-FS, and versions of PostScript as well as the STL data exchange format used by most ***rapid prototyping*** systems.

The data transfer process involves writing all or part of your model into a neutral file, which can be read by most other CAD and CAM systems. The file created should then be physically transferred (using CD, e-mail, etc.) to the receiving system where it can be read and converted back into a model. This process is notoriously prone to losing data or introducing errors so you should always check the received model before continuing work.



## *Outputting Geometry*

Geometric models are transported from to other CAD/CAM-systems using the command `Files`⇒ `FILE`: `WRITE`. This stores everything under the *current target object* (surfaces, curves, etc.) into the neutral data exchange file.

DESKARTES represents its curves and surfaces mathematically as ***cubic polynomials***. This is the most generally accepted representation for surface models among the various CAD and CAM systems. There should be no problems for other systems to read in curves and surfaces generated by DESKARTES.

Trimmed surfaces may require some special attention. To further improve the efficiency and success of data transfer of DESKARTES surfaces, it is recommended you use the command `FORMAT`⇒ `Surface`: `Optimize` before converting the file.

## *Inputting Geometry*

The command `FILE`: `READ` allows you to read geometry from other systems into DESKARTES. The supported file formats are DXF, IGES, VDA-FS, and STL. These have been tested with all major modeling systems, and DESKARTES has been proved to have one of the best data input links of the surface modeling systems on the market.

## *Converting Graphic Images*

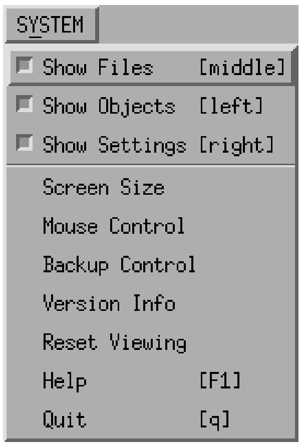
DESKARTES stores all of its graphic images in the Sun Raster File format. The images may be converted into the TIFF or BMP format with the `FILE`: `CONVERT` command. These file formats are the most widely used by the various 2D graphics systems.

# 3 MENU COMMAND EXPLANATION

## SYSTEM MENU

The `SYSTEM` menu contains commands that deal with the workings of the DESKARTES system itself.

Menus are available to control the display of the more important sub-windows and to set up some major operating parameters of DESKARTES.



### SYSTEMÞ Show Files

This command shows the ***File Window***, and hides it again. If the File Window is visible, a black square appears next to the menu item. The File Window appears in the center of the graphics window and can be used to manage the files and directories created by DESKARTES.

An alternative method of controlling the display of the File Window is by clicking a mouse button in the graphics window. The choice of mouse button for this function is defined by the command `SYSTEM⇒ Mouse Control`.

### SYSTEMÞ Show Objects

This command shows the ***Object Window***, and hides it again. If the Object Window is visible, a black square appears next to the menu item. The Object Window appears on the left of the graphics window. It is used to manage the parts of your model created in DESKARTES.

An alternative method of controlling the display of the Object Window is by clicking a mouse button in the graphics window. The choice of mouse button for this function is defined by the command `SYSTEM⇒ Mouse Control.`

### SYSTEMÞ Show Settings

This command shows the ***Settings Window***, and hides it again. If the Settings Window is visible, a black square appears next to the menu item. The Settings Window appears on the right of the graphics window. It is used to control specific functions in DESKARTES.

An alternative method of controlling the display of the Settings Window is by clicking a mouse button in the graphics window. The choice of mouse button for this function is defined by the command `SYSTEM⇒ Mouse Control.`

## SYSTEM Þ Save Layout

This command is only available on Unix/MacOS based versions of DESKARTES.

You may ***customize the layout*** of DESKARTES by moving and resizing any of the DESKARTES windows. The command `Save Layout` saves the current sizes and positions of the windows for later use. The saved layout will be used when you launch DESKARTES the next time.

To ***move*** a window, point the mouse cursor on the title bar of the window, press and hold the left mouse button, and drag the window to its new location. To ***resize*** of a window, point the mouse cursor on the lower right corner of the window, and drag with the mouse as with moving windows. Edit mode icons may be moved in a similar way, however they should not be resized.

## SYSTEM Þ Screen Size

This command is only available on Windows based versions of DESKARTES.

You may choose between ***four standard resolutions for the screen size*** in DESKARTES. You should match the screen resolution you prefer (set in the control panel of Windows under Display) to the DESKARTES resolution. If you do not match these resolutions either the DESKARTES window will not fill the screen or it will be too big to fit on the screen.

Most people prefer to use the highest resolution possible as it gives the largest working area. The graphics card you are using restricts the resolution. Some graphics cards are unable to display the highest resolutions or the display may be unstable or flickering. It is unwise to use flickering displays for long periods of time as they may damage your health.

## SYSTEM Þ Set Colors

This command is only available on Unix/MacOS based versions of DESKARTES.

You may use this command to ***change all of the user interface colors*** used by DESKARTES including the drawing vector colors.

When the command is executed, the User Interface Color Window appears. In the interface window the colors are divided into three groups:

- ***vector colors***, defining the colors in which the objects are displayed;
- ***user interface***, defining the colors of buttons, windows and their background;
- ***grid colors***, defining the three colors used for drawing the grid.

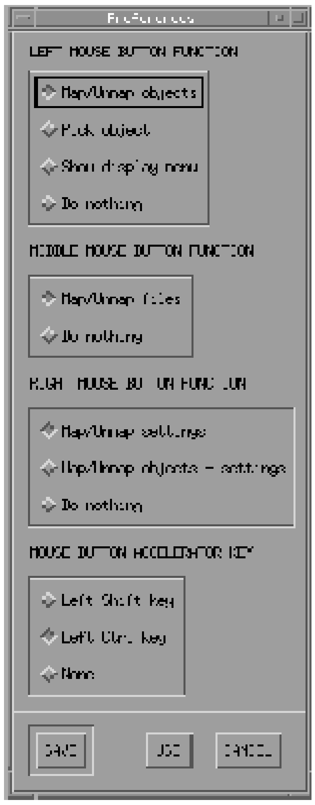
All the colors are changed in the same way. Select the color(s) you want to change by clicking on the small button to its left. Slide the red, green and blue rulers to change the color. When you are satisfied with the color, click on the small button again.

You may change each color one at a time. It is also possible to change several colors at the same time by clicking on several small buttons.

Once all of the color changes have been made you may use the colors for the current modeling session by clicking the `APPLY` . If you wish to use the new colors for all subsequent sessions click on the `SAVE` button. Finally, to discard all the changes and revert to the original colors click on `CANCEL`.

You may cancel all changes without exiting from the color selection window by pressing the button `DEFAULT` (for the currently selected colors) or `ALL DEFAULT` (for all colors).

## SYSTEM Þ Mouse Control



This command allows you to assign ***different function to the mouse buttons*** when clicked with the cursor in the graphics window.

Normally the left mouse button pops up the Object Window. With this command, it can alternatively be made to pick objects (like command `SELECT`⇒`Object`: `Pick`), show a menu of display commands or do nothing.

The middle and right mouse buttons can be assigned to pop up the File and Settings Windows individually, or you can make the right button pop up both the Object and Settings Windows simultaneously. The latter option is particularly useful if you have assigned picking to the left button.

By default, you have to press down the `SHIFT` key before clicking the mouse buttons to achieve the effect explained above. You can choose to use the `CTRL (Control)` key or no key press at all. Most users choose whichever key is in the bottom left of the keyboard.

## SYSTEM Þ Backup Control

Normally, before DESKARTES makes a modification to a model it first stores a copy of the current model in a ***backup file*** called `safe_system.` This allows an incorrect modification to be undone. This command allows you to decide if the backup files are stored in the current model directory, or into the machine's temporary directory.

This selection has a significant impact on the operation speed if DESKARTES is being used ***across a network*** and the model directory is not located on the current workstations. Storing the backup files on the machine's local disk in the temporary directory is much faster than writing files over the network.

If you are not using DESKARTES over a network this command should be set to current directory.

The choice of the backup directory will be recorded between sessions if you store the choice with the SAVE button.

## SYSTEM Þ Version Info

This command displays information about the current ***version of the software*** that you are running. If you have to report a problem with DESKARTES this information should be supplied.

This box also contains the ***System ID number***. This number identifies your computer uniquely. When you first receive your software you should report this number to your suppliers and they will provide a software 'key' which will allow you to use your computer for running DESKARTES. If you do not do this you will only be able to run the software in demo mode, and many of the functions, in particular saving data will not be allowed.

## SYSTEM Þ Reset Viewing

This command ***resets*** certain parameters to default values. Currently this includes the viewport, display window, eye point and grid size. In addition, the memory assigned to rendering graphics will be cleared. This can be useful if the graphics on the screen have become confused and do not appear to reflect the current model correctly.

## SYSTEM Þ Help

This Reference Manual is also provided as an HTML document for On-Line Help.

The Help system can be navigated by typical to www practices, by clicking at various items and fields seen on the Help screen, and it is equipped with features like Cross-Reference and Search.

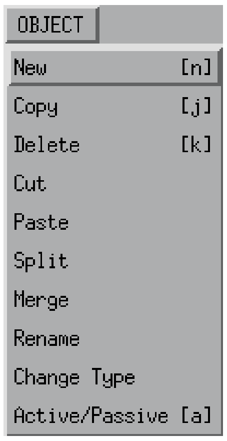
In order for it to work, the `Help` command requires a recent version of the Netscape Communicator or Microsoft Explorer to be installed on the workstation.

## SYSTEM Þ Quit

This command ***exits*** DESKARTES. The system asks if you really want to quit. Choose YES to exit or `NO` to change your mind.

## *OBJECT MENU*

The `OBJECT` menu contains commands that allow you create and manage objects. This is a most important concept for the efficient use of DESKARTES.



The commands in this menu are related to the Object Window, which is displayed to the left of the main window. Many of the commands are repeated in the pop-up menu that appears when you press the right-hand mouse button over the Object Window.

In general the current target object affects these commands. The current target object will be highlighted in black in the Object Window.

### OBJECTÞ New

The `New` command creates an ***empty container object***, in preparation for storing curves and/or surfaces. The command is context sensitive, which means that the options offered by this command vary with the current situation.

If the created object is to be an element stored in the top window of the object window, DESKARTES asks for the name of the element. If it is a projection set stored in the middle window, below the current object, DESKARTES asks for the projection direction.

For example, when ***beginning a new model***, you would start by choosing the command `OBJECT`⇒ `New` to create an empty element. Then you would choose `OBJECT`⇒ `New` again, to create an empty set of projection curves. To create a curve in the projection set, you might use the command `CURVE`⇒ `Design:Input` and so on. So most times, when you start a new part of your model, you will use the `OBJECT`⇒ `New` command **twice**.

### OBJECTÞ Copy

The command `Copy` creates an exact ***duplicate*** of the target object.

If the target object is an element, stored in the top window of the object window, then you will be asked for a new name for the copy.

If the target object is a curve set you will be asked to confirm the operation as this is an unusual action. One use for this option is if you later wish to cut and paste the copy of the curve set into a new element.

If the action is a curve or surface, no further action is required.

In each case the copy will appear immediately below the original in the Object Window and will become the target object.

## OBJECTÞ Delete

This command ***deletes*** the target object and all objects below it (the so-called children) from the object hierarchy.

As a special feature, when deleting a set of ***trim curves***, the system asks if the corresponding trim curves of other surfaces should be deleted too. Unless you have a good understanding of trim curves it is recommended that choose YES to this option. The system then not only deletes the other trim curves, but it also automatically merges the remaining trim sets. This ensures that the surfaces reflect the current state of the trim curves.

## OBJECTÞ Cut

The commands `Cut` and `Paste` can be used to change the order of objects within the object hierarchy and move things between elements.

The command `Cut` deletes the target object but remembers it by placing it into an internal object memory ready to be pasted into another place in the hierarchy. The object will remain in the object buffer until the next `Cut` command is executed.

You may want to copy the object, to make a duplicate, before cutting and pasting.

## OBJECTÞ Paste

The commands `Cut` and `Paste` can be used to ***change the order*** of objects within the object hierarchy and move things between elements.

The command `Cut` is first used to place an object into an internal object buffer. Next, you must choose a new target object where the object is to be placed. Finally, execute `Paste` to insert the buffer object after the current target object.

It is possible to use the `Paste` command several times in succession, creating several copies of the object in the buffer.

## OBJECTÞ Split

This command ***splits an element or an object*** into two.

If the target object is an object directly under an element, the element is split into two. All objects before the selected object remain in one element. The target object and those after it are placed into a new second element. The name of the second element is requested as a parameter.

If the target object is a primitive object, for example a curve in a curve set, the result is two curve sets: one containing the objects before the target object, and a new set containing the target curve and the curves after it.

## OBJECTÞ Merge

The command `Merge` ***joins*** the target object to the next object, combining the contents of the next element to the target element.

The objects to be joined must be of the ***same type***. For example, you may not join a curve-set to surface.

When you join ***curves or polygons***, the command works in the same manner as `CURVE⇒ Combine: Join Two`. You will be asked at which end the curves are to be joined. In addition, it asks whether you wish to join them at both ends to create a closed curve.

In case of ***surfaces***, no such queries are made. The second surface is added directly as the continuation of the first. It may be necessary to turn the surfaces to consistent directions with the command SURFACE⇒ Change: Top to Bottom. Furthermore, when joining surfaces they must both have the same number of control points in the cross-sectional direction.

## OBJECT Þ Rename

The command Rename gives a new user entered ***name*** to the target ***element***. Only elements can be renamed. Curve-sets, Curves and Surfaces cannot be renamed.

## OBJECT Þ Change Type

The command Change Type only applies to curve-sets. It changes the ***object type*** of a set of curves, for example from x-projections into section curves.

Using this command you can change the direction of a projection set for example changing x-projections into y-projections.

This command is especially useful when one wants to change the ***interpretation*** of existing curves for the command BUILD⇒ Create: Surface. The same surface may be built in many different ways, and sometimes it can become apparent that the chosen section curves would work better as projections, and vice versa.

You might also want to change 2D projections into 3D projections ready for 3D building. However, the command Change Type won't automatically change the dimensionality of the curves. You will need to use the command CURVE⇒ Change: Dimension as well.

## OBJECT Þ Active/Passive

The command Active/Passive changes the state of the target object between ***active*** and ***passive***. Passive items are indicated by the appearance of a hyphen (-) in front of their name.

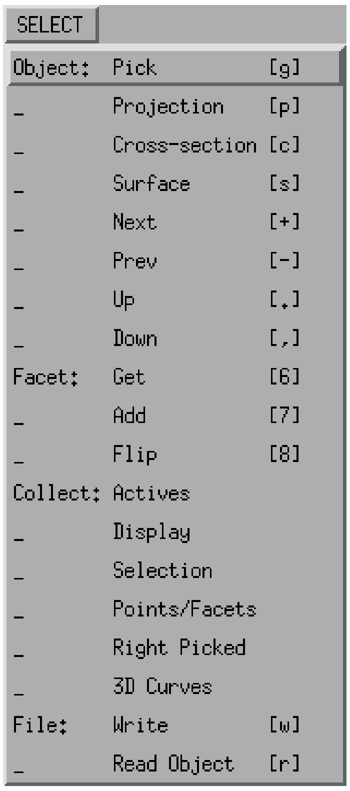
Certain commands such as DISPLAY⇒ All and SELECT⇒ Collect: Actives only apply to active objects. Passive objects would be ignored by these commands. Objects that are not of immediate interest should be made passive. They will then not normally be drawn on the screen.

For example, there might be a part of your model that has two design alternatives, both of which you have modeled. You could make the elements of one of the alternatives passive so that the second alternative can be seen on its own, and vice versa.

## SELECT MENU

The SELECT menu contains alternative ways of executing ***object selections***. It also gives easy access to a couple of ***file*** commands.

In particular, you must use these menu commands when creating ***command series*** with the TEACH function, as it does not allow you to choose objects from the Object Window directly.The SELECT menu also allows objects to be ***collected*** together, which can be useful when transferring data to other systems or for creating libraries of standard shapes for use within DESKARTES.



## SELECT Þ Object: Pick

This command allows you to choose the ***target object*** by ***pointing*** to an object in the graphics window and clicking the left mouse button. You can pick anything that is shown on the screen. This is a handy way of moving among different elements, instead of selecting from the Object Window.

If you wish to pick a ***trim curve*** or a ***texture area*** then the trim-set or texture-set must be the target object. It is then possible to pick either from the 2D window or directly from the 3D view.

Sometimes it may be difficult to point exactly on the desired object, and a wrong object is picked. To ***reselect***, you can repeat the command, and just point at the same place as previously. The system will then retrieve the ***next closest candidate*** as target. You can repeat this process until the correct item is picked.

Another way to reselect is to choose the command again, then press ***any key on the keyboard***. This way the previous, unintentionally selected target object is erased from the display, and the next nearest object is chosen as the target.

## SELECT Þ Object: Projection

Chooses a ***projection set*** in the current object as the target object. If the target object is already a projection set, the command selects the next projection set as the target object.

If there is no projection set, an empty projection set is created much like the command `Object⇒ New`.

## SELECT Þ Object: Cross-section

Chooses a ***section set*** in the current object as the target object.

If there is no projection set, an empty section set is created much like the command `Object⇒ New`.

## SELECT Þ Object: Surface

Chooses a ***surface*** in the current object as the target object.

## SELECT Þ Object: Next

This command selects the ***next object*** among objects at the same level of the object hierarchy. This is a very handy way of moving between several curves in the same curve set.

## SELECT Þ Object: Prev

This command selects the ***previous object*** among objects at the same level of the object hierarchy. This is a very handy way of moving between several curves in the same curve set.

## SELECT Þ Object: Up

This command moves ***one level up*** within the object hierarchy, for example from a single curve to the curve set or from a curve set to the element.

## SELECT Þ Object: Down

This command moves ***one level down*** within the object hierarchy, for example from the curve set to the first curve in the set or from a curve set to the element.

## SELECT Þ Facet: Get

This command enables picking and deleting of individual facets within a faceted model.

The command has three modes of operation, depending on which mouse button is pressed:

***Left button***: picks the facet pointed by the cursor;

***Middle button***: the user draws a box enclosing a set of facets, near the first mouse click. The system then asks if the facets inside the box should be either deleted, or separated into a new object;

***Right button***: deletes the facet pointed by the cursor. The system remains in the facet delete mode until some other mouse button is clicked.

Deletion of facets can sometimes be required for manual repair of faceted models. A typical use of the command is to delete all facets near the problem areas (detected by command `FORMAT⇒ Faceted Verify`), and then let command `FORMAT⇒ Faceted Repair/Orient` fill in the missing areas again with better facets. Also, the next command `Facet:Add` is available for more detailed filling of the holes in the model.

## SELECT Þ Facet: Add

This command enables adding individual facets into a faceted model.

The facet is normally added by pointing at three existing vertices in the model, using ***left*** mouse button.

If several facets are to be added at the same area, there is a fast way to add facets following the first one. For this, the user just shows the new vertex of the next facet with the ***right*** mouse button. The two other vertices are taken as the last points show for the previous facet. (Care must be taken not to input the vertices in wrong order!)

## SELECT Þ Facet: Flip

This command inverts the selected facet, in case its normal side is opposite to the neighboring triangles.

The command may be required for special editing of faceted models for Volume calculation and Rapid Prototyping.

## SELECT Þ Collect: Actives

This command ***copies*** all active objects of the same type as the target into a new element called ALL. This means that you can collect together either curves or surfaces dependant on the current target object type. Any existing element named ALL will be deleted. If you wish to avoid this, rename the old ALL element before executing this command.

The ALL element will automatically be made passive. This is to avoid drawing the same surfaces twice with commands like DISPLAY⇒ All.

A useful application for this command is to gather all surfaces that make up a model together before converting them to a faceted model and writing it out to an STL file for rapid prototyping, for instance

## SELECT Þ Collect: Display

This command gathers all the ***objects currently visible*** on the display, and places them into a new element. You are asked for the name of the new element the default for which is ALL. Any existing element of the same name will be deleted. You are given the option to copy the objects before they are gathered.

## SELECT Þ Collect: Selection

This command allows you to interactively gather ***chosen objects*** into a new element of user defined name.

A parameter box appears which allows you to specify if you wish to select objects that are totally inside a rectangle which you'll draw on the screen or all objects that are close to it. You will also be asked whether you wish to keep a copy of the items in the original elements.

After giving the above parameters, draw a rectangle on the screen to select the objects. They will be stored in the new element.

## SELECT Þ Collect: Points/Facets

This command ***separates a set of points or facets*** away from the target object into a separate object. This can be used for example to take out erroneous or floating parts out of a faceted (STL) model, or to split point cloud data into smaller parts.

The user draws a polygon on the screen using the left mouse button (middle to cancel, right to close the polygon). After the polygon is drawn, the system asks whether the points/facets inside the polygon should be deleted, or separated into another object.

## SELECTÞ Collect: Right Picked

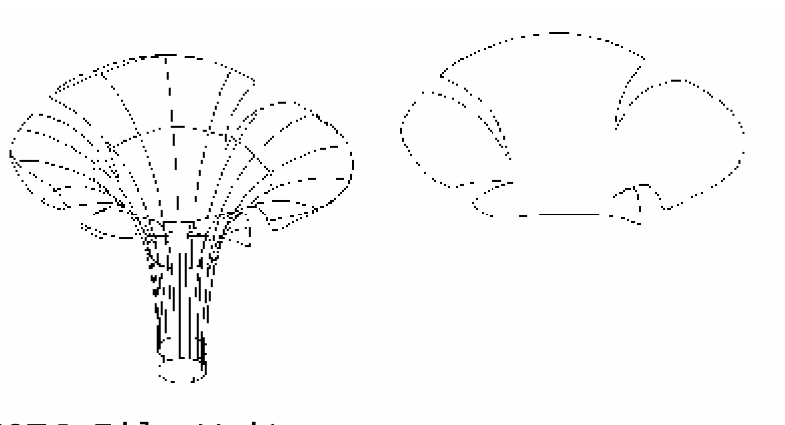
This command is intended for ***separating only some curves*** away from a larger set of curves. It deletes all curves in the target curve set which are not highlighted (colored yellow).

The command works in combination with object picking. If you use the command `SELECT⇒ Object: Pick` and *pick a curve with the* ***right*** mouse the curve will be highlighted. Thus, you can color those curves you want to keep in a set. When all the desired curves are highlighted, command `SELECT⇒ Collect:Right Picked` deletes the rest of the curves.

## SELECTÞ Collect: 3D Curves

This command enables picking of any 3D curves that are shown on the display. The curves can be either already existing curves or isoparametric curves of a surface.

The user picks the curves using the ***left*** mouse button. Picking ends when the ***right*** mouse button is clicked. The selected curves are placed into a new 3D curve set in the target element.



## SELECTÞ File: Write

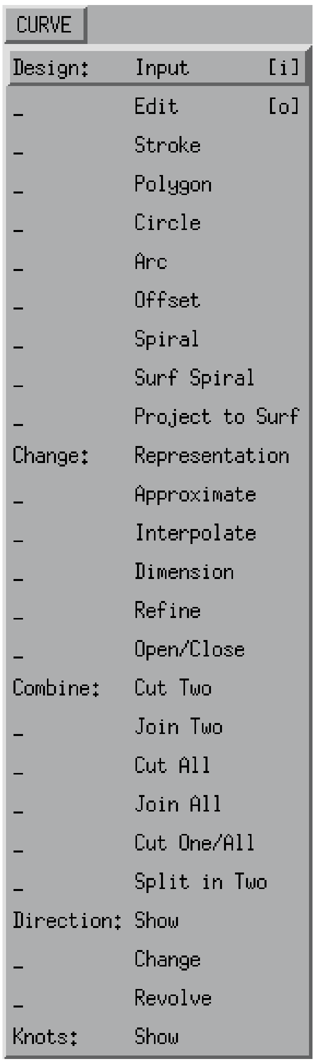
This command allows you to ***save*** the current model or part of it ***to the hard disk***. The command is identical to the File Window command of the same name. Please refer to the explanation of this command for more details.

## SELECTÞ File: Read Object

This command allows you to ***load*** a model from disk. The command is identical to the File Window command `FILE READ` but reads only object files, the identifier of which is `cobj`. This is useful for reading in your own standard design.

## CURVE MENU

The `CURVE` menu commands produce or manipulate polygons, B-Spline curves, and Bézier curves. To use the commands, the target object must be a curve or a curve set.



If a ***new curve*** is created, its geometric representation is determined by the current `Settings`⇒ `REPS` parameter. Likewise, the ***input shape*** is affected by the `Settings`⇒ `SHARP` and `SMOOTH` buttons.

If a `CURVE` command is directed at an ***existing curve***, the nature of the operation is determined automatically according to the representation of the curve. For example, the ***representation*** of the curve may later be changed with the command `Change Representation`.

## CURVE Þ Design: Input

This command creates a curve and changes to a separate input mode where the user can ***interactively place points*** to define the shape of the curve. The command requires a curve-set to be selected as the target object. If a 3D-projection set is selected, the command will create a 3D curve, otherwise a 2D curve will be created. The functions of the input mode are discussed in detail in the Editing Functions part of this Manual.

The type of curve created is determined by variables in the Settings Window. The representation of the input curve is determined with the `Settings`⇒ `REPS` mode. A `linear`, `B-spline` or `Bézier` curve can be created.

If the object created is linear (a polygon), the points entered are to be the vertices (corners) of the polygon.

If the object created is a B-spline, the points are taken as its control points. If `Settings`⇒ $\boxed{\texttt{SMOOTH}}$ is selected, the result is a smooth curve that approximates the control points. However if `Settings`⇒ $\boxed{\texttt{SHARP}}$ is selected, all the B-spline control points are automatically made triple so that the curve will have sharp corners just like a polygon.

If the created object is a Bézier curve, the curve will pass through the input points. In other words, the Bézier curve always interpolates the given points. If `Settings`⇒ $\boxed{\texttt{SMOOTH}}$ is selected the curve will be smooth. With `Settings`⇒ $\boxed{\texttt{SHARP}}$, the curve will have sharp corners at the input points just like a polygon.

During curve input (and editing) it is possible to **snap the input points to any object** (curve, surface or faceted model) shown on the screen. The snap mode is activated by **pressing the SHIFT key while the middle mouse button is pressed down** in “move point” functions. Note that the speed of snapping depends on how large models are displayed on the screen.

To input a 3D curve, the user is requested to

1. Select a 3D curve set, and
2. Select the viewing direction from one of the X/Y/Z axes.

For example, if the X direction is chosen, the new curve is created in the YZ plane. The resulting 3D curve can then be further edited with command `CURVE`⇒ `Design Edit`.

## CURVE Þ Design: Edit

This command changes to a separate curve-editing mode where you may ***edit the shape*** of the target curve. The available editing functions depend on the representation of the curve. Each type of curve (B-spline or Bézier) has its own editing mode. The functions of the edit mode are discussed in detail in the Editing Functions part of this Manual.

## CURVE Þ Design: Stroke

Command `Design Stroke` allows you to create and input a polyline in the target curve set by ***drawing with the mouse***. The stroke is started as you press the left mouse button down. The movements of the mouse are recorded and used to define the shape of the polyline. The command ends as you release the left mouse button.

If the last stroke point is close to the first one, the polyline is automatically closed to form a loop.

As it stands this would not be very useful for modeling as it is a polyline and has many points. To make a smooth curve suitable for modeling from the stroke polyline, use the command `CURVE`⇒ `Change`: `Approximate`.

## CURVE Þ Design: Polygon

This command creates a ***straight-sided curve*** in the target curve set. A parameter box appears requesting the polygon’s edge length, the coordinates of its center point, and number of edges. The curve type is defined with `Settings`⇒ `REPS`.

This command is often used to create cross-section curves.

## `CURVE Þ Design: Circle`

This command creates a ***circular curve*** in the target curve set. A parameter box appears requesting the radius of the circle, the coordinates of the location of the center point, and the number of control points.

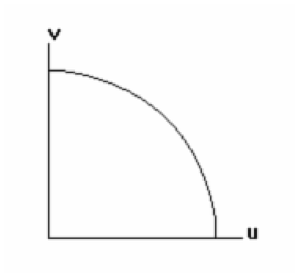
The ***number of control points*** chosen is important. Circles with low numbers of points will have poor accuracy compared to a true circle. Given the same number of knot points, Bézier circles are more accurate than B-splines.

The curve type depends on `Settings⇒ REPS`.

## `CURVE Þ Design: Arc`

This command creates a ***circular arc*** in the target curve set. A parameter box appears requesting the start and end angles for the arc, the coordinates of the location of the center point, and the number of control points to be used for the arc.

Angular values in DESKARTES are measured anti-clockwise in degrees, with zero being located at 3 O'clock. The example below shows an arc starting at 0 degrees and ending at 90 degrees.

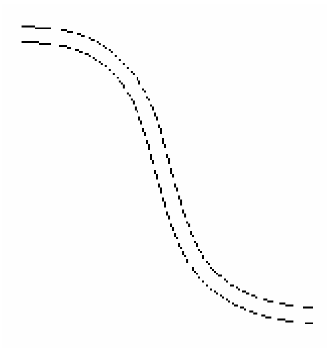


The ***number of control points*** chosen is important. Arcs with low numbers of points will have poor accuracy compared to a true circle. Given the same number of knot points, Bézier circles are more accurate than B-splines.

The curve type depends on `Settings⇒ REPS`.

## `CURVE Þ Design: Offset`

This command produces an ***offset*** of the target curve. The offset curve follows the shape of the target curve, but at a specified distance away on one side of the original curve as shown in the diagram below.

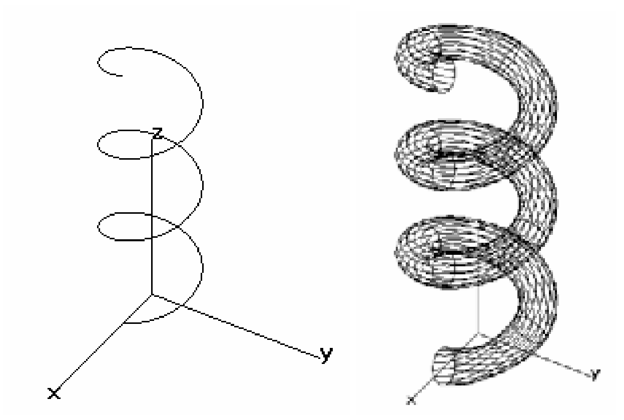


The ***side*** on which the offset curve is produced depends on the sign (positive or negative) of the distance entered. If the offset comes on the wrong side of the original, undo the command and enter a negative value for the distance.

The ***accuracy*** of the offset curve depends on the number of control points used for the original curve.

## CURVE Þ Design: Spiral

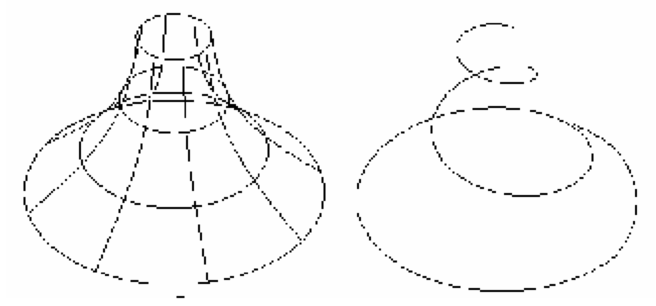
This command creates a ***3D spiral curve***. A parameter box appears requesting how many loops the spiral should contain, as well as the pitch and diameter of each loop.



One use for such a curve is for creating a spiral surface by 3D building. This would be achieved by creating a spiral in a 3D-projection curve set and a circle in the cross-section curve set.

## CURVE Þ Design: Surf Spiral

This command creates a ***3D spiral curve*** which ***wraps around a surface*** which should be the target object. A parameter box appears requesting how many loops the spiral should contain and the tolerance. The smaller the tolerance the closer the curve will match the surface but it will be defined using more points.

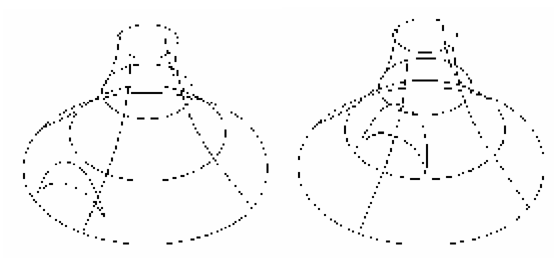


The result is a 3D Bézier curve contained within a 3D projection set in the current element. This curve set could then be cut and pasted to a new object and used as the basis for further design work such as ***3D building***.

## CURVE Þ Design: Project to Surf

This command ***projects*** the target 3D B-spline or Bézier curve or curve set onto a surface. You may use the resulting 3D curves as ***projections for 3D building***.

Before executing the command, the curve should be positioned on one side of the surface and the surface for the projection should be visible on the screen. After executing the command, you will be asked to select the surface by pointing and clicking with the left-hand mouse button and then to give the direction for the projection. The direction may be either x, y or z, polar or the current viewing direction. The command then projects the curve onto the nearest side of the surface. This is shown in the before and after pictures below.



With the ***polar*** projection the curve is projected towards the vertical axis that goes through the projection surface's fix point.

The ***accuracy*** of the result depends on the shape of the curve(s) and the surface, and on the number and distribution of the control points on the curves. If the number of points in the curve are low, ***refining*** the curve with the `CURVE⇒ Change: Approximate` command before projecting will improve the accuracy of the result. The points of the curve that do not project onto the surface (i.e., they "fall outside") are left untouched, while the others are projected.

## CURVE Þ Change: Representation

This command changes the ***representation*** (polygon, Bézier or B-spline) of the target curve or curve set. The ***shape*** of the curve will be preserved in the change as much as possible.

When changing a B-spline/Bézier curve to a polyline, you may specify how accurately the change is done. The command asks how many points the resultant polyline curve is to have.

A B-spline curve may always be changed into a Bézier curve without changing shape. However, the change from Bézier to B-spline may alter the shape of the curve slightly. If the Bézier curve has originally been converted from a B-spline curve, and it has not been subsequently edited, the reverse conversion will not change the shape.

## CURVE Þ Change: Approximate

This command creates a new curve that ***approximates*** and replaces the target curve. A parameter box appears requesting the representation and the new number of points of the resulting curve.

The shape of a curve generally changes when approximated. The amount of deviation is dependent on the ***number of points*** chosen. It is generally best to set a relatively large number of points for the approximation.

The number of point's option is available for all kinds of approximations. In the case of converting from polygons to another format, a second option is available for controlling the approximation. Instead of setting the number of points, you may define the greatest permissible deviation, or ***tolerance***, from the original polygon. DESKARTES will then determine how many points are required to achieve this deviation.

## CURVE Þ Change: Interpolate

This command creates a smooth B-spline or Bézier curve that goes through (***interpolates***) all the points of the target polygon. The program asks what type the resulting curve should be.

The interpolated curve will be stored after the target polyline. You may thus compare the resulting curve to the original polygon before deleting the polygon.

With interpolation to a B-spline curve, if there are significant differences in the distances between the points to be interpolated, the resulting curve may not be as smooth as desired. With Bézier curves, the spacing of the points is not so important.

It is not advisable to interpolate a very large number of points, use the command `CURVE⇒ Change: Approximate` instead.

## CURVE Þ Change: Dimension

This command changes the ***dimensionality*** of the target curve or curve set, from 2D to 3D or vice versa. You will be asked for the projection direction. For curves already drawn in a projection set the default is the projection set direction. In general the projection direction should be chosen so that the resultant 3D curves will be correctly oriented.

An example of the application of this command can be found when creating a skinned surface. The command `SURFACE⇒ Design: Skin` requires 3D section curves. You may design the section curves first in 2D, then use this command to change them to 3D, and finally position them for skinning.

The command `CURVE⇒ Change Dimension` can also be used to ***combine the primary and secondary projection curves*** in 2D into 3D projections. This way the user can first start his design using 2D projections, and switch to 3D projections automatically without loosing previous modeling information. The primary projection must be selected as target to apply this new feature.

## CURVE Þ Change: Refine

This command refines the curve with more control points, exactly the same way as the curve edit function **D** does.

Refining the curves often improves the accuracy of some modeling, such as 3D Build and Project to Surf.

## CURVE Þ Change: Open/Close

This command opens or closes a curve in the same manner as the curve edit functions ‘B’ and ‘C’.

## CURVE Þ Combine: Cut Two

This command ***cuts*** two overlapping curves with each other. The curves to be cut must belong to the same curve set.

Before executing the command, the curves should be visible on the screen. The two curves are then picked from the graphics window by pointing and clicking with the left-hand mouse button on each curve in turn.

Since each of the two curves is cut, the result will be four curves.

**Note:** The command is only able to cut Bézier curves and polylines. B-spline curves are automatically converted to Bézier form before cutting.

## CURVE Þ Combine: Join Two

This command ***joins*** two curves into one. The ends of the curves to be joined do not have to be touching but the two curves do need to reside in the same curve set.

Before executing the command, the curves should be visible on the screen. The two curves are then picked from the graphics window by pointing and clicking with the left-hand mouse button on each curve in turn.

The ends of the curves nearest to the picks are joined together by adding a straight segment between them.

## CURVE Þ Combine: Cut All

The command ***cuts all the curves*** in the target curve set with each other. As the result, the curves are split into several curve pieces, which end at the intersection points of the curves.

This command is particularly useful for advanced users who need to manually combine trim curves. If a trim curve set contains curves which need editing they may first be cut with CURVE⇒ Combine: Cut All then extra portions of curve can be deleted before re combining with CURVE⇒ Combine: Join All. Of course, this operation will not normally be required as DESKARTES handles trim curves automatically.

**Note:** The command is only able to cut Bézier curves and polylines. B-spline curves are automatically converted to Bézier form before cutting.

## CURVEÞ Combine: Join All

The command attempts to ***join all the curves*** in the target curve set with each other, to form closed curves out of the separate open ones.

If any open curves remain after the operation, you will be given the option to keep them or delete them.

This command is particularly useful for advanced users who need to manually combine trim curves. If a trim curve set contains curves which need editing they may first be cut with CURVE⇒ Combine: Cut All then extra portions of curve can be deleted before re combining with CURVE⇒ Combine: Join All. Of course, this operation will not normally be required as DESKARTES handles trim curves automatically.

## CURVEÞ Combine: Cut One/All

This command ***cuts all the curves*** in the current curve set ***with the target curve***.

**Note:** The command is only able to cut Bézier curves and polylines. B-spline curves are automatically converted to Bézier form before cutting.

## CURVEÞ Combine: Split in Two

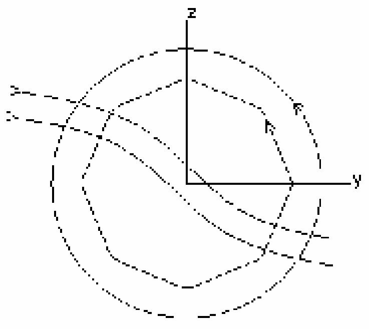
This command splits a curve into two pieces at the point shown by the user. The curve can have any representation (B-spline, Bezier or linear) and dimension.

The point where splitting takes place is the nearest existing (control) point on the curve. If required, the user may add new points to the curve by editing it.

In case of a closed curve, the start point of the curve determines the other end of the split curves. It is therefore a good idea to set the start point of the curve with command CURVE⇒ Direction Revolve before splitting.

## CURVEÞ Direction: Show

This command turns on the ***display of curve directions*** on all curves. Executing the command again turns off the display.



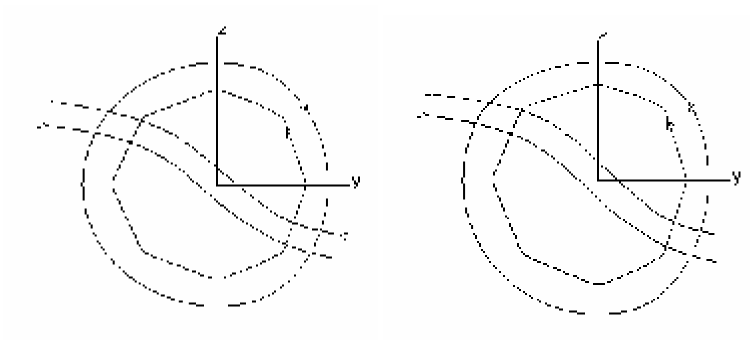
Arrows drawn at the start of the curve show the curve direction. This includes closed curves, which also have a direction and a start point.

This is particularly important with section curves, as to avoid twists when building surfaces, the start points need to be aligned as in the circle and octagon below. Similarly for projection curves the curves should run in the same direction like the open curves.

## CURVE Þ Direction: Change

This command ***reverses the direction*** of the target curve. The command can also be used for a set of curves at once. It then goes through the curves one by one and the user may change the direction of each with the *left mouse button*, or use *right button* to leave the direction as it was.

The direction is particularly important with section curves, as to avoid twists when building surfaces, the directions need to be the same as in the circle and octagon below right. Similarly for projection curves the curves should run in the same direction like the open curves below right.

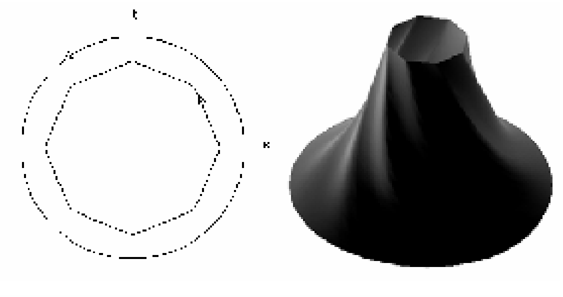


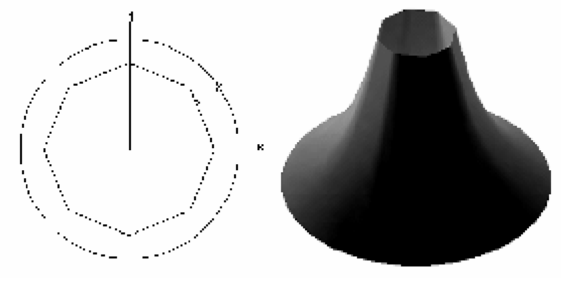
## CURVE Þ Direction: Revolve

This command sets the ***start point of a closed curve*** or curve set to a new location.

The start points can be displayed with the command CURVE⇒ Direction: Show. For each curve, point and click with the left mouse button to indicate the location of the new start point. The control points of the curve determine the possible start points. The new start point will be located as near as possible to the point indicated with the mouse.

This is particularly important with section curves, as to avoid twists when building surfaces, the start points need to be aligned. To align the start points select the cross section curve-set and issue the command. Depending on the geometry of the sections, it may not be possible to align the start points exactly. Since the actual position of the start-point of closed cross-sections is rarely important (it is the alignment that is important), you may try repeating this command and choosing a different location for the start point.



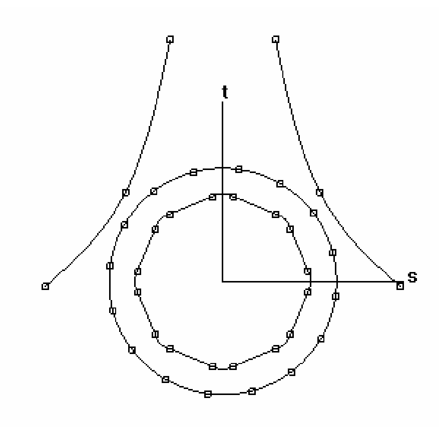


## CURVE Þ Knots: Show

This command turns on the display of curve ***knot points*** on all curves. Executing the command again turns off the display.

Markers along the curve show the curve knot points. The knot points are directly related to the positions of the control points of the curve.

This is particularly important with projection curves for ***building surfaces*** as the knot points determine the possible locations for cross-section.

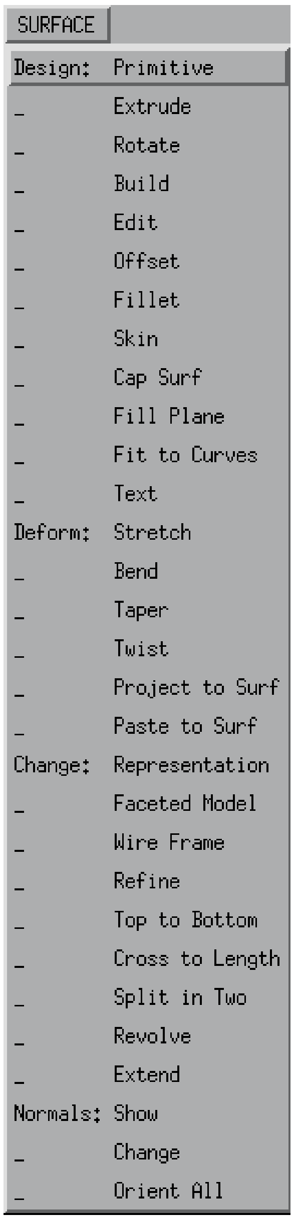


## *SURFACE MENU*

The `SURFACE` menu holds commands for creating and manipulating surfaces.

When creating new surfaces, the type of curves with which the surface was created automatically determines the representation of the surface. The curves defining the surface must be defined in different curve sets (projection and section sets) in the same element.

The surface manipulation commands work automatically according to the representation of the surface. For example, the command `SURFACE`⇒ `Design`: `Edit` always leads to the edit mode appropriate for the selected surface.

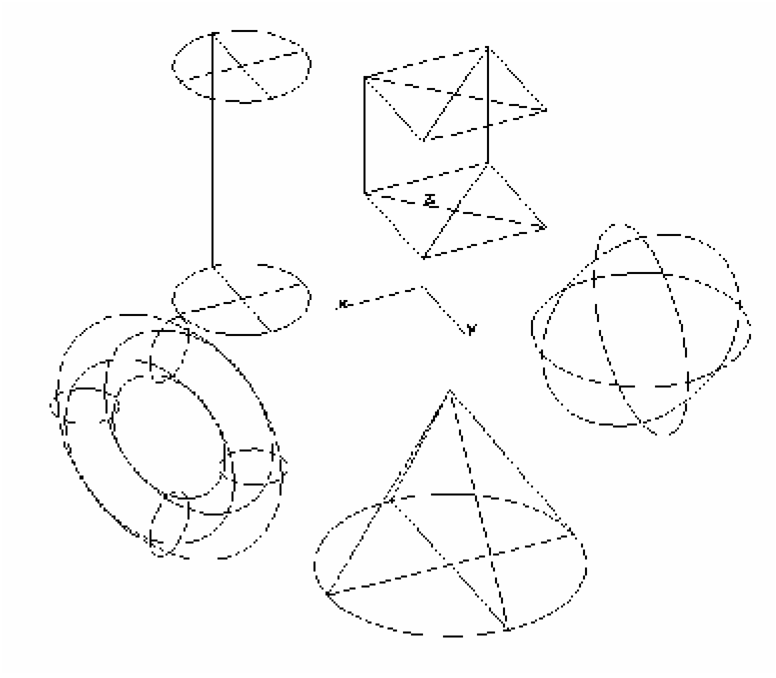


## `SURFACE`Þ `Design`: `Primitive`

This command creates a ***primitive surface*** in the current element, represented in Bézier form.

The available primitive types are:

- ***Plane***
- ***Box***
- ***Sphere***
- ***Cylinder***
- ***Cone***
- ***Torus***
- ***Other***



For each primitive a parameter box appears requesting appropriate dimensions (e.g. a cylinder's center, radius, and height), and the axis direction (x, y, or z) along which the primitive is to be created.

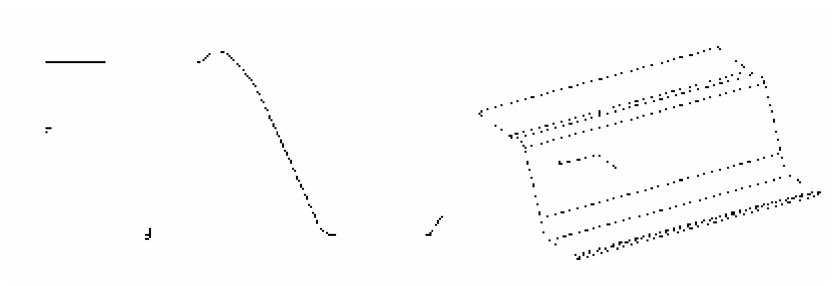
The "Other" option allows for the creation of a primitive by loading any surface model that the users have saved into the `user_primitives` library. The names of all available models are shown as a scrollable list. Please refer to the corresponding option `Files`⇒ `FILE WRITE` for details of how to store such user-defined primitives.

**Note**: It is strongly recommended that surface primitives are created in there own element and not in the same element as other surfaces. Whenever a surface is remade from the defining curves, all surfaces in that element, including primitives, are deleted.

## SURFACE Þ Design: Extrude

This command creates a surface by ***extruding*** all of the projection curves or 3D curves in the target element the given distance along the projection direction.

In case of 2D projection curves the projection direction is implied from the type of projection curve set, e.g. X projections project along the direction of the X-axis.



A parameter box appears requesting the ***start and end distances*** of the extrusion. The default values are set by DESKARTES so that the created surface will be slightly (1.2 times) longer than all the other surfaces in the model.

In case of a 3D curve the following parameters are requested from the user:

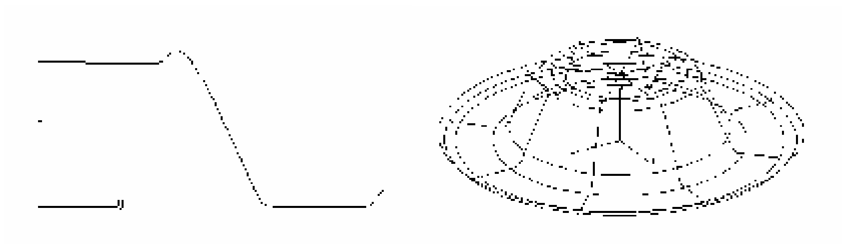
- *extrude direction*. This can be one of the coordinate directions, or the curve's normal direction if it is not one of the axis directions.
- *extrude method*. This defines whether the curve is extruded in just one direction from the curve, or in both directions to either side.
- *extrude distance*. Defines how length of the extruded surface.

Another option is whether the extruded surface(s) should be ***capped at the ends*** by adding a flat surface, in effect, closing the surface. If you were to extrude a circle without end caps the result would be a tube. If you requested end caps the result would look like a solid bar. This option only applies to closed projection section curves. ***Note***: this capping method assumes that the curve shape is relatively simple, so that the caps can be defined within the single surface. More complex caps can be created using the command `SURFACE⇒ Design: Fill Plane`.

As a side effect of the extrusion, all previous surfaces and faceted models within the element will be assumed to be redundant and they will be deleted.

## SURFACE⇒ Design: Rotate

This command creates a surface by ***rotating*** the first projection curve in the target element around an axis.



A parameter box appears requesting the ***axis of rotation***, which can be either the horizontal or the vertical axis of the projection curve. The axis of rotation can also be chosen to pass through the projection curve's origin or through the projection curve's fix point.

If the element does not contain a section curve set, the command will create a surface with a circular cross-section. If the element does contain a section curve set, a so-called ***free-form rotation*** is produced in which all of the surface's cross-sectional curves will be the same shape as the 2D-section curve. Lengthwise shapes will obey the given projection curve shape. Only one cross-section curve can be used at a time.

The projection curve and the section curve must both be smooth curves, or both linear polylines. In other words, a B-spline projection can be rotated around a Bézier section curve and vice versa, but not around a polyline.

Some further instructions of creating rotational surfaces are:

1) In order to produce a ***symmetrical surface***, the cross-section curve must be located symmetrically relative to its origin.

2) The projection curve must be located on one side of the rotation axis. Otherwise, the surface created will cross over itself or ***self-intersect***.

   For example, rotating a complete circular projection curve around a full circle cross section curve would result in the sphere having a ***double surface***.

3) If an open projection curve ***begins and ends*** on the rotation axis and is swept along a closed section curve shape, a closed surface will be produced. Otherwise, there will be a hole at either end of the surface.

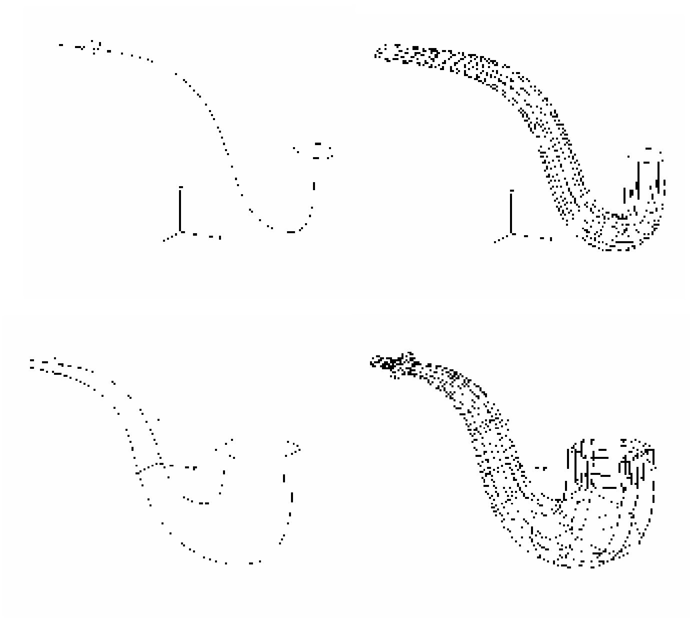
   For example, a **sphere** can be produced by rotating a semi-circular projection curve, with end points on the axis, around a full circle with center at the origin.

4) The projection curve does not have to be open, and the section curve does not have to be closed. However, these are the most common cases.

## SURFACE Þ Design: Build

This command is an exact equivalent the command BUILD⇒ Create Surface. The command is included in the SURFACE menu for consistency, so that all the surface creation commands are located in one menu.

The Build command creates a surface based on the projection and section curves contained within the target element. It is the most flexible surface creation tool available within DESKARTES. There are several different forms of the command, depending on the types of information the projection and section sets contain. This is explained in the 'Typical Modeling Session' section of this manual.



Building requires that there be at least one projection set containing one or two B-spline projection curves, and that the section set contains at least one section curve which may be either B-spline or Bézier. Further control of the surface creation is given by the other commands in the BUILD menu.

As a consequence of creating the surface, all previous surfaces and faceted models within the element will be assumed to be redundant and they will be deleted.

## SURFACE Þ Design: Edit

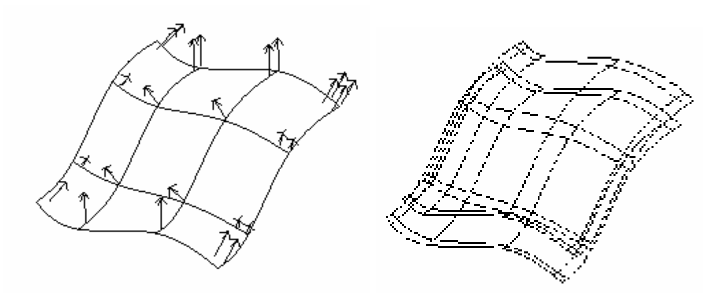
This command changes to a separate ***surface-editing mode*** where you may edit the shape of the target surface. The available editing functions depend on the representation of the surface. Each type of surface has its own editing mode. The functions of the edit mode are discussed in detail in the Editing Functions section of this manual.

Any edits that are performed on a surface are not preserved if you recreate the surface using one of the other SURFACE⇒ Design commands. It is therefore wise to only use this command when you are confident that the overall shape of the surface is satisfactory.

## SURFACE Þ Design: Offset

This command produces an ***offset surface*** of the target surface. The offset surface follows the shape of the target surface, but at a specified distance away on one side of the original surface.

Two parameters are asked from the user to determine the offset distance and direction. A series of arrows are drawn on the surface (so-called surface normals) indicating the positive side of the surface:



With curved surfaces, the offset surface is always created in Bézier form, even if the original surface is B-spline. The accuracy of the offset surface depends on the number of control points used for the original surface but is also better for the smaller offset distances.

For non-trimmed surfaces, you are given the option to create joining surfaces**,** which combine the original surface and its offset, forming a closed surface set, like a sheet of material. This is not done with trimmed surfaces, but the trim curves are copied to the offset surface instead.

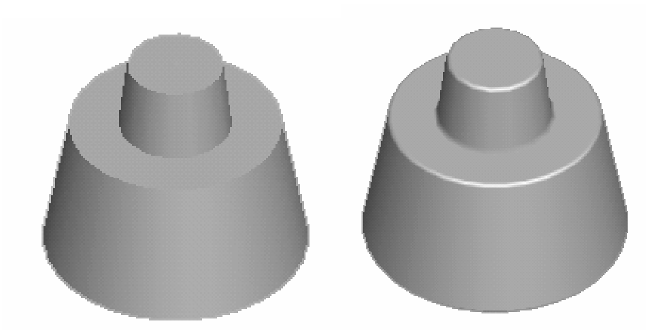
Offsetting works for a complete element of surfaces, too. You may thus offset a complete model at once. If the surfaces meet each other smoothly, the result will generally be smooth too. Before offsetting several surfaces, you should check all of the surface directions (with command SURFACE⇒ Normals: Change) so that the offsets will be computed in the same direction for all the surfaces.

The command also works with faceted (STL) models, in which case the resulting offset surface is also a faceted model.

## SURFACE Þ Design: Fillet

This command automatically ***rounds off*** some or all the ***sharp corners*** of the surface.

A parameter box appears requesting whether you wish to fillet all edges simultaneously, or only the ones in the cross-wise or the lengthwise surface direction.



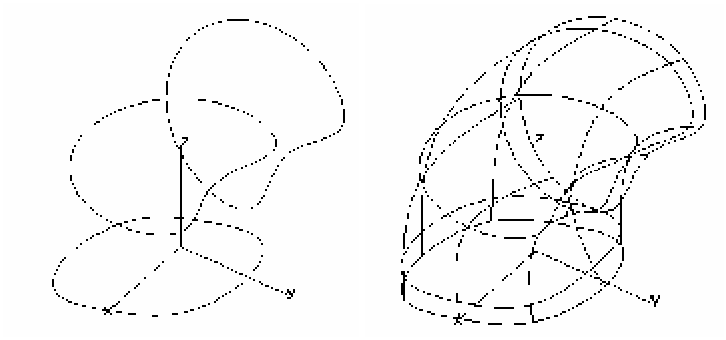
Another parameter specifies whether a single radius should be used for all rounds, or whether you want to specify each rounding interactively. In the latter case, the program displays a profile of the surface in the chosen direction, and asks for the radius of each sharp corner.

A third parameter specifies the radius, which only applies to the global rounding operation.

**Note:** Only relatively small radii may be used for filleting. If there is not enough room for the fillet between the surfaces cross-wise or lengthwise curves, the command may produce unexpected results.

## SURFACE Þ Design: Skin

This command creates a ***skin surface*** directly over a set of ***3D section curves***. The defining curves all have the same number of control points and be in the correct order.



A parameter box appears requesting the interpolation method. The choices are Bézier and B-spline. B-spline interpolation generally gives smoother results than Bézier. However, if the section curves are very unevenly spaced, then Bézier interpolation may produce better results.

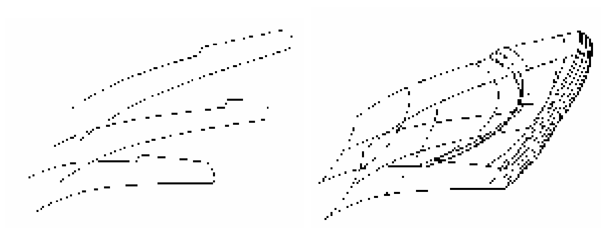
There are several ways to produce the defining curves. One way is to input them as 2D curves first, then change them to 3D curves with `CURVE⇒ Change: Dimension`. Finally, use 3D editing and transformations for modifying and locating them in 3D.

You may of course input 3D curves directly or read them in from a data transfer file. Remember that all curves need to have the same number of control points.

A special method of controlling the lengthwise shape of a skin surface is achieved as follows. Select the 3D sections, and issue the command `CURVE⇒ Change: Interpolate`. This produces the surface boundary curves in a 3D-projection set. You may edit these by changing their control points (not knots, as they are tied to the section locations!). When you give command `Skin` again, the resulting surface will follow the edited lengthwise shape. This works with open section curves only.

This is often the most versatile way of creating surfaces. For example consider the modeling of a plate with an ornate rim shown below. The three curves on the left below were created by drawing one of the end curves in a 2D cross section curve set. This curve was then copied twice and the first copy was edited in the area of the rim. The cross section curve set was then changed to 3D curves using `CURVE⇒ Change: Dimension` choosing the X projection direction. The second curve in the set was then rotated around Z by 15 degrees and the third curve was also rotated by 30 degrees.

This gives three curves around a 30-degree arc where the first and last curves are identical. These curves were then skinned to produce the surface shown.



This portion of the plate rim was then copied and rotated by 30 degrees around Z 11 times to form the complete rim.

## SURFACEÞ Design: Cap Surf

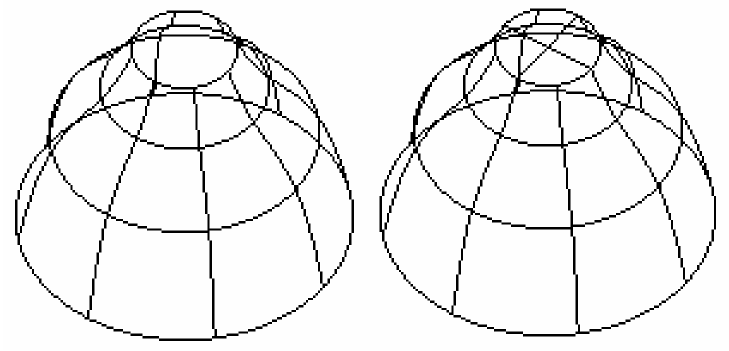
This command closes an open surface with a separate surface. This command allows arbitrarily complex gap curves and closing surfaces. For example, surfaces generated with the SURFACE⇒ Design: Extrude command can benefit from this command.

## SURFACEÞ Design: Fill Plane

This command ***fills a planar (flat) gap*** in the target 3D curves.

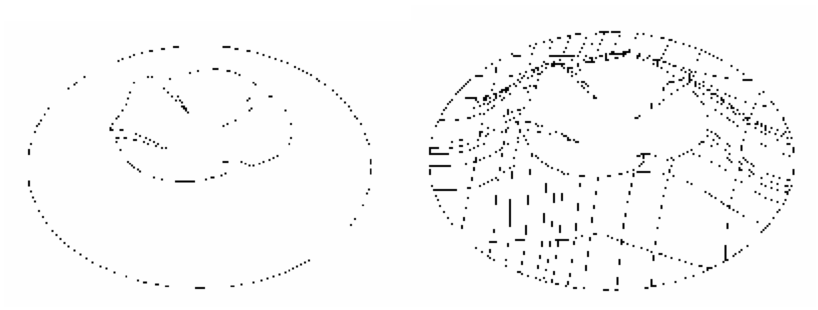
The gap is first selected with the command `SURFACE⇒ Collect:3D Curves`. Curves are created from these edges and are stored in a 3D curve-set. The curve set produced by this command should be the target. This command is then used to fill these areas with planar surfaces. If all the curves in the set are not located on an equal level the command issues a warning, but still creates a planar surface.

In this example the top edge of the revolved surface was selected with `SURFACE⇒ Plane: Select` before filling the gap with `SURFACE⇒ Plane: Fill` to get the result on the right. It would not be possible to fill both the gaps at the top and bottom of this surface in one go with this technique. You would have to repeat the procedure twice. There are other (better) ways of achieving this result in this simple case. This command is usually used for more complex 'gaps'.



## SURFACEÞ Design: Fit to Curves

This command computes a surface through given 3D curves, and trims the surface with the curves.



The target object for the command should be the curves or curve set through which the surface is to be computed. The command then asks for the following parameters:

***tolerance***: value which controls the overall accuracy of the computation.

***sharp edges***: answer YES if the result surface should contain internal sharp edges. However, it should be noted that sharp edges are not treated perfectly with the program, and they take very long to compute.

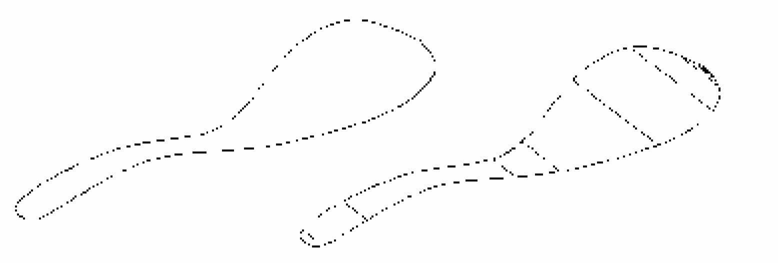
***surface side***: decides whether the surface is computed inside to the curves, or if an exterior part is produced. (This choice can later be changed using the command `TRIM`⇒ `Change: Invert Cut Part`.)

This parameter is not asked if none of the curves to be fitted is a closed curve. In that case, the program just computes an untrimmed surface that follows the shape of the given open curves.

***surface shape***: decides if the resulting surface should be a *plane*, or *ruled* in one direction, or *curved* in both directions.

With the ruled option, the program displays two lines on the screen, and asks which one better represents the ruler direction..

The computation of the surface may take quite a long time. But it is usually worth waiting! Overall,

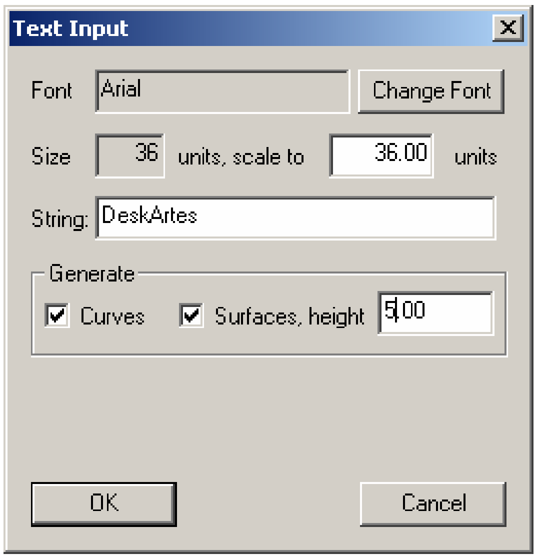


this command is extremely powerful, and it has practically unlimited amount of applications. It can be used generally to fill in any surfaces that are missing in the model, such as end caps, parting surfaces and design mistakes.

In case of filling missing surfaces in the model, the curve boundaries for missing surfaces can be found with the automatic command `FORMAT`⇒ `Surface Verify`, or command `SELECT`⇒ `Collect 3D Curves` for interactive selection.

## SURFACE Þ Design: Text

This command (only available in the Windows version) allows you to generate 3D geometries from the common TrueType fonts available in your Windows. Geometries can be combined either into surface models or triangulated and combined into STL models.


Text Input

Font: Arial

Size: 36 units, scale to 36.00 units

String: DeskArtes

Generate: [✓ Curves] [✓ Surfaces], height 5.00

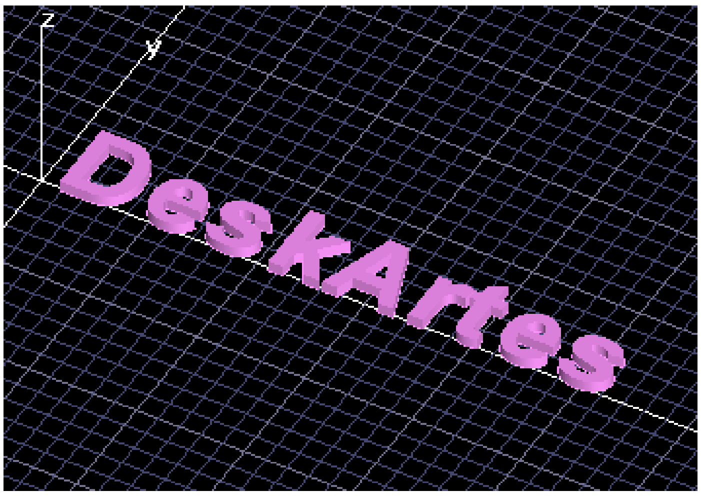
The text font is selected with the standard Windows font dialog by pressing the **Change Font** button. The **Size** will give the size in units selected in the Font dialog, and is **scaled to** the user given size when the surfaces are generated. Generally, the bigger the size chosen in the Font dialog, the more accurate the resulting geometries will be.

The wanted text is typed into the **String** edit field.

**Generate** settings **Curves** and **Surfaces** toggle on/off the respective result type. For surfaces selection the height of the letters must also be given.

Pressing **OK** generates the curves and or surfaces from the text.

An example 3D text with Arial Narrow font:



## SURFACE Þ Deform: Stretch

This command ***stretches*** a surface in an axis direction. It is different from scaling in the sense that it in effect ***adds material*** at a fixed position, while the shape of the surface at the different sides that location is preserved as closely as possible.

As parameters, the command asks:

1) The ***axis directio***n in which the surface is stretched,

2) The ***position*** around which the stretching happens,

3) The ***distance*** how much the object is stretched.

**Note**: when stretching ***trimmed surfaces***, there should be enough control points on the surface around the stretching plane – otherwise the trim curves may not be stretched accurately. To prevent this from happening, you may ***refine*** the surface(s) before stretching.

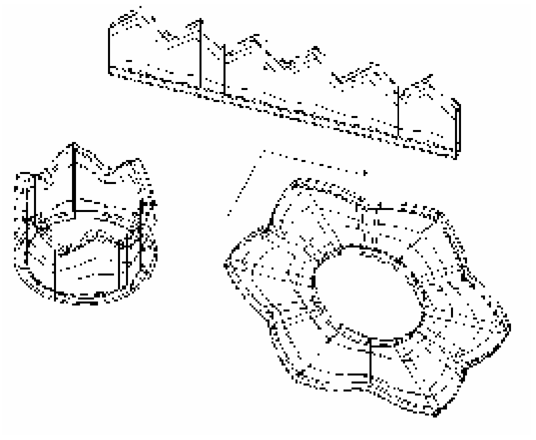
## SURFACE Þ Deform: Bend

This command ***bends*** the target surface ***around an imaginary cylinder***.

After executing the command a parameter window appears.

- ***bend around***: the surface will be bent around the selected axis (X, Y or Z).

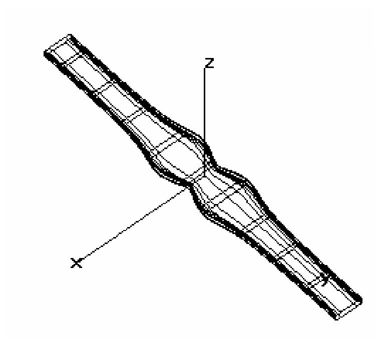
- ***bend along***: this parameter tells the direction where the surface(s) start and end. In the following example, the top surface is bent along the Y axis in two ways: the bottom left surface is bent around Y, and the bottom right one around X.

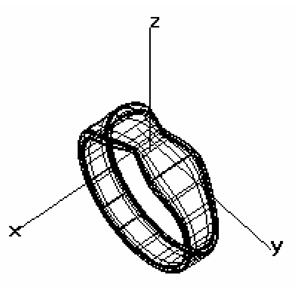


The **bending angle** is controlled interactively by the user with the mouse movement, and the result is immediately shown on the screen. As an alternative, the user can press any key on the keyboard to enter the angle numerically.

It is possible to bend B-spline and Bézier surfaces, however, to ensure that smooth surfaces are generated it is better to bend B-spline surfaces. The ***accuracy*** of the bending depends on the definition of the surface. It is sometimes required that the command `SURFACE⇒ Change: Refine` is applied to the surface before bending.

An example of the use of this command is in the design of jewelry, in particular rings. It is often easier to define the shape of a more complex ring by drawing it flat as can be seen on the left. This command can then be used to bend the ring to its true shape. The bend options chosen in this example are Y direction and 360 degrees.





## SURFACE Þ Deform: Taper

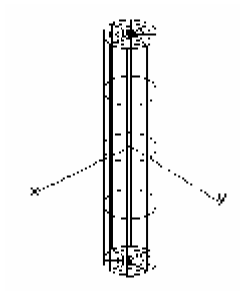
This command ***shrinks*** the target surface in ***proportion to its distance form a neutral line***. The part of the surface on the centerline will not be affected whereas more remote parts of the surface will be scaled by an increasing amount. This has the effect of giving a taper or relief to the sides of a model as would be required for casting or injection molding.

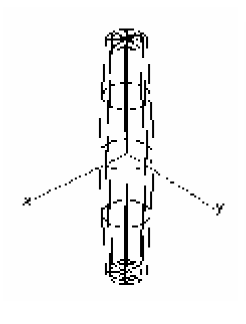
After executing the command the user is asked to give the ***axis direction*** for the taper operation. The surface will not change along the chosen axis but it will be reduced in size in the other two axes.

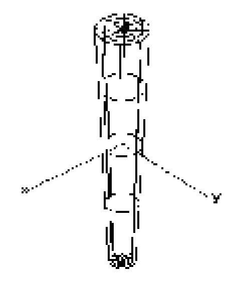
The operation is then interactive: the user controls the scaling with mouse movements. Alternatively, a parameter dialog can be opened by pressing any key on the keyboard. The following parameters are available in the dialog:

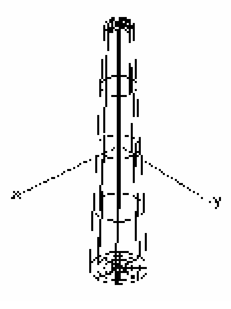
- The maximum ***scaling values*** in the other two directions;
- The position where the neutral centerline is to be positioned. There are three options for this: at the fix point or at either extreme of the surface in the twist axis;
- Whether the surface should be refined before the transformation.

The illustrations below show an initial cylinder and the three possible results. In each case the taper axis is Z and the scaling values where 0.5. The first result is for the fix point option (the fix point was at the origin). The other two are for the +Z and –Z options.









## SURFACE Þ Deform: Twist

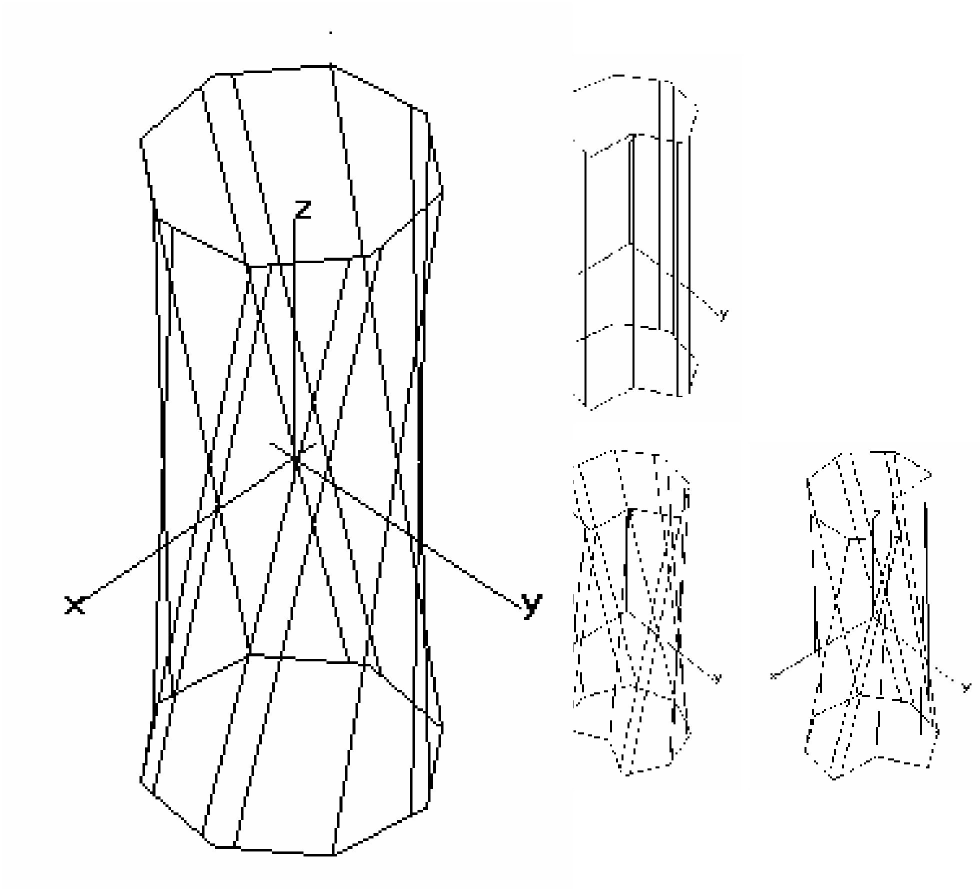
This command ***twists*** a target surface around an axis ***in proportion to its distance form a neutral line***. The part of the surface on the centerline will not be affected whereas more remote parts of the surface will be twisted by an increasing amount.

After executing the command a parameter box appears requesting the ***axis direction*** for the twist. This direction is a centerline around which the surface will be twisted.

The twisting is then done interactively with the mouse or, by pressing any key of the keyboard, the parameters can be set numerically:

- Maximum angular twist;
- Where the neutral line is to be positioned. There are three options for this: at the fix point or at either extreme of the surface in the twist axis;
- Whether to refine the surface before the operation in order to ensure a better result.

The illustrations below show an initial hexagonal cylinder and the three possible results. In each case the twist axis is Z and the twist angle was 60 degrees. The first result is for the fix point option (the fix point was at the origin). The other two are for the +Z and –Z options. To understand the differences notice the position of the 'notch' in the cylinder. In the first case this has not moved at the center of the cylinder. In the second case it has not moved at the top of the cylinder. In the third case it has not moved at the bottom of the cylinder.



## SURFACE Þ Deform: Project To Surf

This command ***drops*** the target surface or surface set onto another surface, wrapping the surfaces around the receiving surface. The target surfaces can be Bézier, B-spline or faceted surfaces but the receiving surface must be B-spline or Bézier.

After executing the command the ***receiving surface*** should be indicated by graphically picking with the left mouse button. The ***projection direction*** is then requested which can be either X, Y, Z or ***polar***.

If the projection direction chosen is ***polar***, the target object should be initially located "around" the projection surface.

For the axial (x/y/z) modes of projection there are a couple of additional options:

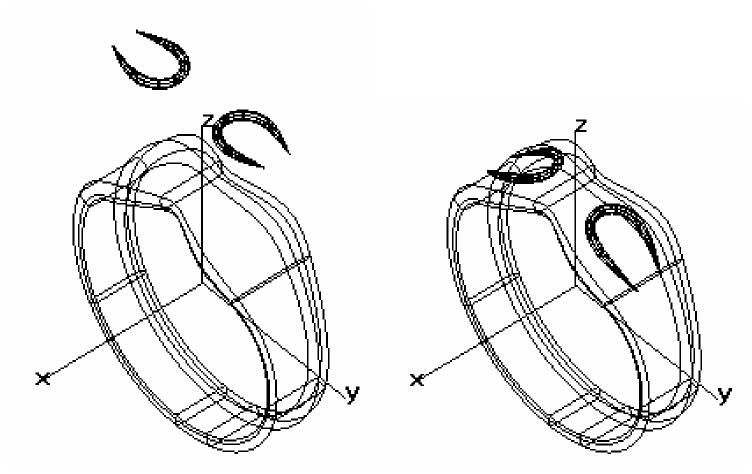
- ***orthogonal***: projection is done strictly in the axis direction.

- ***towards z-axis***: projection is basically in the axis direction, but bending the surface towards the vertical axis going through the projection surface's fix point.

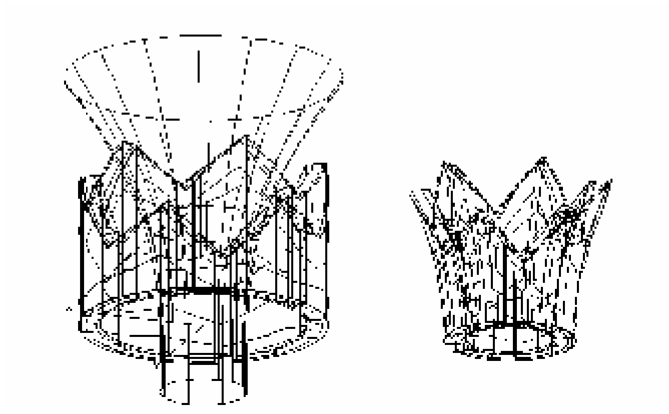
- ***towards fix point***: projection is basically in the axis direction, but bending the surface towards the projection surface's fix point.

As the result, the target surfaces are deformed so that they wrap around the receiving surface. The ***fix point*** of the target surfaces determines the level that will be matched to the receiving surface. Points that do not project onto the surface (i.e., they "fall outside") are left untouched.

An example of the use of this command is in designing jewelry where a feature surface can be design flat and then wrapped around a base ring surface. The surfaces to be projected are designed as normal. These are then positioned on the correct side of the ring and the fix point of these surfaces is set to the midpoint of the surface pair. The result of this command is then shown on the right with the Z projection option chosen.



Another example with the ***polar*** option:



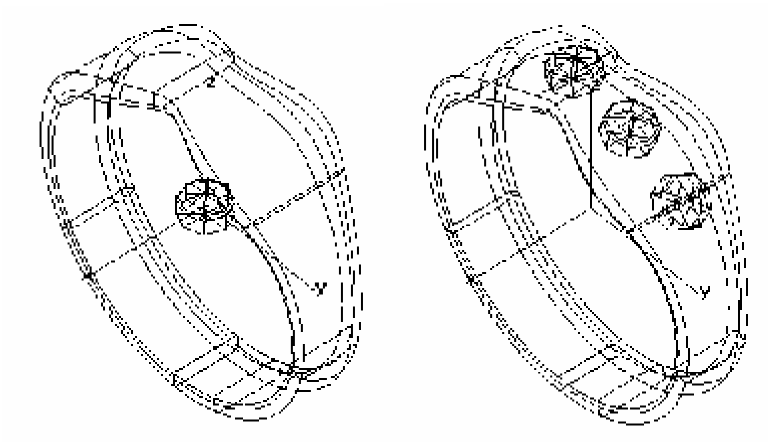
## SURFACE Þ Deform: Paste To Surf

This command can be used to ***position a surface onto the target surface***, reorienting the pasted surface so that the surface is at right angles to the target surface.

Before executing this command the surface to be pasted must be ***cut*** into the object memory by using the `OBJECT⇒ Cut` command. This command is then issued and clicking with the left mouse button indicates the location of the pasted surface. The pasted surface is then located so that its ***fix point*** matches that point on the target surface and rotated so that the original Z direction of the pasted surface points out from the target surface. In other words the Z-direction is aligned with the ***surface normal*** of the target surface at the paste point.

Since the pasted surface also remains in the object memory this command can be ***repeated several times*** if required.

An example of the command can be seen below. A ring has been designed along with a second surface, a jewel. The jewel is then cut into object memory and the ring made the target. The jewel was then pasted three times onto the surface.



## SURFACE Þ Change: Representation

This command changes the ***representation*** (Bézier or B-spline) of the target surface or curve set. The shape of the surface will be preserved in the change as much as possible.

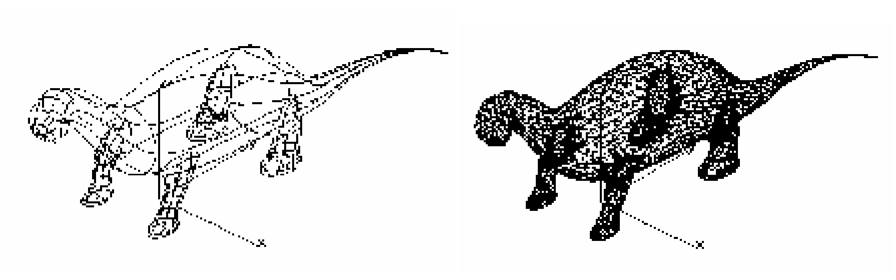
A B-spline surface may always be changed into a Bézier surface without changing ***shape***. A change from Bézier to B-spline may alter the shape of the surface slightly. However, if the Bézier surface has originally been converted from a B-spline surface, and it has not been subsequently edited, the reverse conversion will not change the shape.

This command cannot be used to convert a smooth surface to a faceted surface. A separate command `SURFACE⇒ Change: Faceted` is provided for this operation.

## SURFACE Þ Change: Faceted Model

The command converts the target surface or surface set into ***faceted model representation***.

Faceted models are made up of hundreds of small ***triangles***, which approximate the original shape of the smooth B-spline or Bézier surface. This representation is often used by other CAD or CAM systems, for use on the World Wide Web and by all of the rapid prototyping systems. One further use for this command is in preparation for the calculation of volume by DESKARTES, which can only be done on faceted models.



A parameter box appears requesting a ***tolerance*** for the faceting. This determines the ***maximum distance*** of the facets from the actual surface(s).

The parameter ***max triangle size*** can be used to force all triangles to be under a certain size. A typical use of this would be to prevent very long and thin triangles from appearing as these may cause problems for other systems.

An option called check directions asks whether the ***normal directions*** of the facets should be automatically oriented so that they all point in a consistent direction. This is required for volume computations and rapid prototyping.

The fourth option, ***check gaps***, asks whether any small gaps which might be created between the facets should automatically be filled in. The width of the expected gaps is given as a further parameter. Filling the gaps is vital before writing the model into the STL format, for rapid prototyping. This option must be licensed separately.

When using the “check gaps” option with command `SURFACE⇒ Change: Faceted Model`, the system also detects if the model contains “***floating components***” that are not connected to each other. This can typically happen if the user forgot to trim some parts of the model with each other.

In such a case, the system gives a note of the situation, allowing the user to check which floating components were found (and correct the error before triangulating again). The floating components are stored as separate faceted objects inside the resulting element.

Note that the floating components are not necessarily a design error, in particular, if they define walls inside a hollow solid model.

## SURFACEÞ Change: Wire Frame

This command converts a surface or an element of surfaces to a ***wire frame representation***. The conversion accuracy depends on the parameters in `DISPLAY⇒ Preferences` just as when displaying the surfaces.

This representation is of rather limited usefulness. The main use of the wire frame representation is for data exchange to other CAD/CAM systems, if just the surface outlines are required, or if the receiving system does not understand surfaces.

## SURFACEÞ Change: Refine

This command ***increases the number of control points*** in a B-spline or Bézier surface control mesh.

A parameter box appears requesting the ***directions*** in which the surface should be refined. Refining doubles the original number of control points in the given direction(s). Repeating the command would further double the number.

The command can also be used to refine several surfaces in an element at once. In this case, the command asks the ***maximum patch size*** allowed, and whether the refinement is to be performed to ***all*** surfaces inside an element or only to those surfaces which have patches larger than the given maximum patch size.

Increasing the number of points makes the surface more complicated and therefore slower to work with. However, this command may prove useful to ***increase accuracy*** for other surface commands such as `Bend` and `Twist`.

Also, if a ***trim*** operation fails, refining one or both surfaces may enable DESKARTES to perform the trim operation. In particular surfaces created by the command `SURFACE⇒ Design: Primitive` are in their simplest form possible which can lead to trimming difficulties.

## SURFACE Þ Change: Top To Bottom

This command changes the order of cross-wise curves of the target surface. The first section becomes the last, etc.

The command is only needed when joining two surfaces into one with `OBJECT⇒ Merge`, as the first surface must end where the second one begins.

## SURFACE Þ Change: Cross To Length

This command transposes the direction of the target surface. It changes the surface's lengthwise curves to cross-wise, and vice versa.

The command can be used in conjunction with the `SURFACE⇒ Change: Split In Two` command which only works in the surface’s cross-sectional direction.

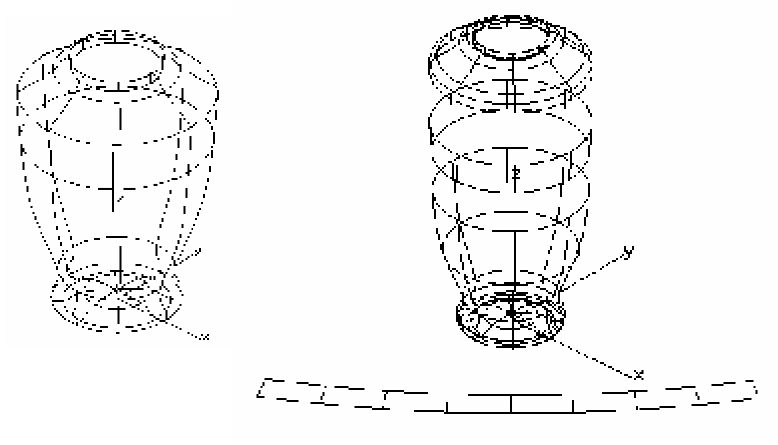
## SURFACE Þ Change: Split In Two

This command ***splits*** the target surface into ***two halves*** along one of its crosswise curves. The crosswise curve where the surface is to be split is located by pointing and clicking with the left-hand mouse button.

Since this command can only split the surface along a crosswise curve, the complimentary command `SURFACE⇒ Change: Cross To Length` can be used to switch the direction if necessary.

The crosswise curves are directly related to the position of the points in the defining curves. If a curve point is not located at the required position the appropriate defining curve can be edited and extra points added.

This command is particularly appropriate when applying textures using the surface flattening procedure. This requires that the strip to be flattened be cut out of the main surface first as shown below. On the left is the original surface. This surface has then been cut in two twice, giving three parts of the surface. The middle part has then been flattened into the shape of the required decal.



## SURFACEÞ Change: Revolve

All closed surfaces have a start point that creates an invisible ***seam*** along the surface. The location of this seam is directly related to the start point of the defining curves. This command allows the start point to be set anywhere in the cross-wise direction of the target surface.

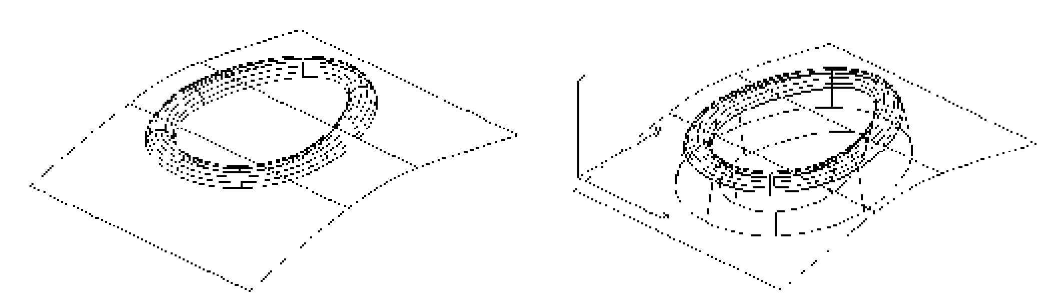
Since this command can only change surfaces along a crosswise direction, the complimentary command `SURFACE⇒ Change: Cross To Length` can be used to switch the direction if necessary.

## SURFACEÞ Change: Extend

This command ***extends*** the target surface making it longer by adding a linear surface strip to the surface's boundaries.

One application of this command is when ***blending*** surfaces built from 3D curves. If the 3D projections are obtained from curves projected onto a surface, the resultant surface may not intersect the original surface. If they do not intersect then these two surfaces cannot be blended together. Extending the 3D built surface makes the surfaces truly intersect and a blend can be created.

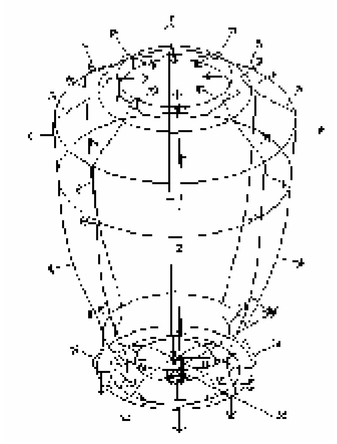
In the example below the two surfaces do not intersect so they cannot be blended together. After one of the surfaces is extended they now cross over each other and can be blended.



## SURFACEÞ Normals: Show

This command displays the target surface or elements ***normal direction***. The normal direction indicates which side of the surface DESKARTES considers being the positive side of the surface.

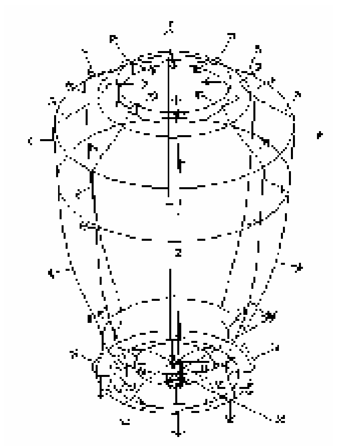
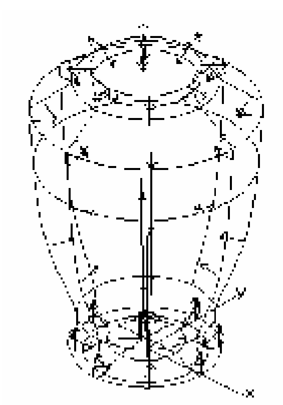
***Arrows*** pointing away from the surface show the normal direction. The normal direction is determined by a combination of a number of factors including the directions of the defining curves. The command also automatically toggles the single sided mode on and off if the model is shown in shaded (and double sided) mode.



The normal direction has an effect on several functions in DESKARTES including blending and offsetting but most of these commands are flexible enough to accommodate a surface no matter which direction the normals point. Generally you may just forget about the direction of the normals. The main exceptions to this are the commands `TRIM⇒ Intersect: Parting Lines` and `TRIM⇒ Blend: Multiple Rball`.

## SURFACE Þ Normals: Change

This command ***displays*** the target surface's or element's normal direction(s), and asks if the direction(s) should be ***changed***.

The normal direction has an effect on several functions in DESKARTES including blending and offsetting but most of these commands are flexible enough to accommodate a surface no matter which direction the normals point. Generally you may just forget about the direction of the normals. The main exceptions to this are the commands `TRIM⇒ Intersect: Parting Lines` and `TRIM⇒ Blend: Multiple Rball`.

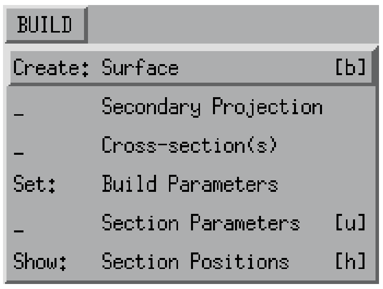
## SURFACE Þ Normals: Orient All

This command automatically orients all surfaces on the display, so that their normals point to a consistent direction (out from the model), as required by various trimming and tooling commands.

The command assumes that the surfaces to be oriented form a solid model, i.e., command `FORMAT⇒ Surface Verify` reports "no gaps" (or only very small gaps) in it.

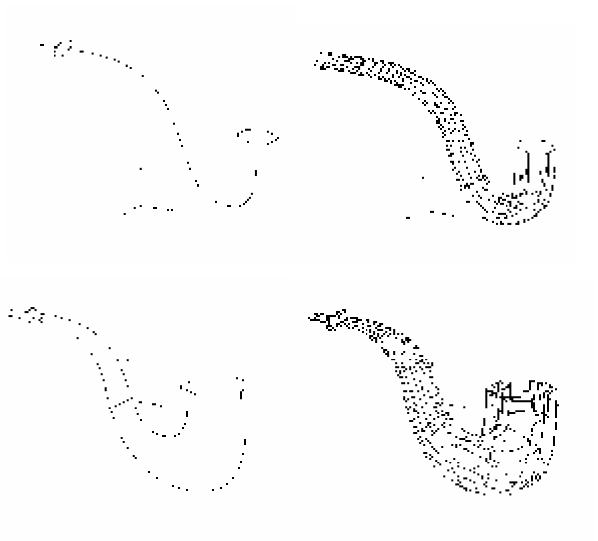
## BUILD MENU

All of the commands in the `BUILD` menu deal with creating surfaces by building which is the most flexible method of creating surfaces within DESKARTES. The command `BUILD`⇒ `Create`: `Surface` builds the surface, while the other commands control how the building is done.



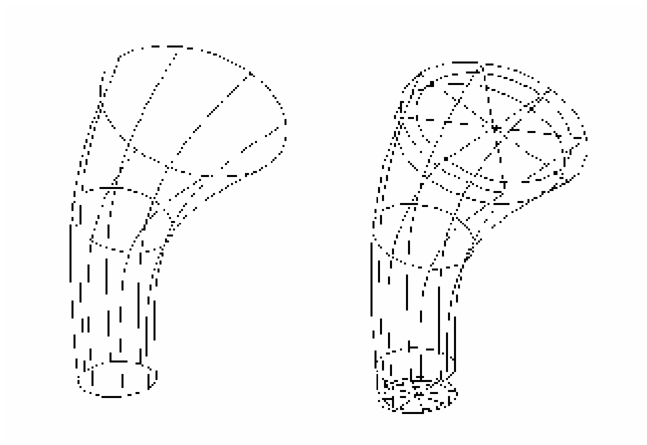
## BUILDÞ Create: Surface

This command creates a surface based on the projection and section curves contained within the target element. It is the most flexible surface creation tool available within DESKARTES. There are several different forms of the command, depending on the types of information the projection and section sets contain. This is explained in the 'Typical Modeling Session' section of this manual. Some examples are shown here.



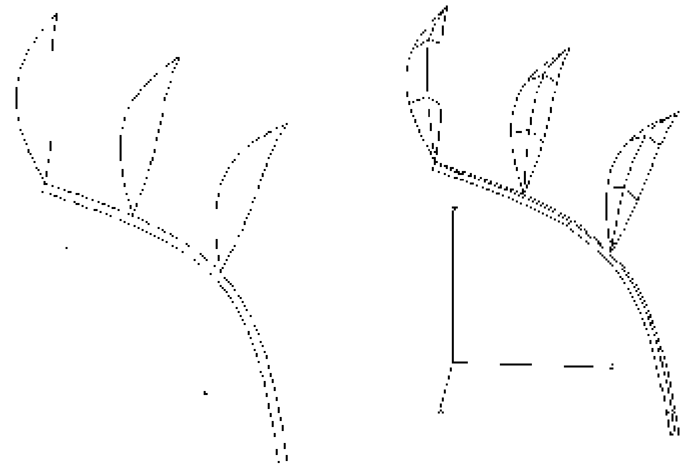
***Building with 2D curves*** requires that there be at least one projection set containing one or two B-spline projection curves, and that the section set contains at least one section curve which may be either B-spline or Bézier. Further control of the surface creation is given by the other commands in the `BUILD` menu. As a consequence of creating the surface, all previous surfaces and faceted models within the element will be assumed to be redundant and they will be deleted.

Parameter ***cap surface ends*** can be set to automatically create end caps to closed (cylindrical) surfaces whose ends would otherwise be open. The caps can be produced to relatively simple section shapes only. With complex section shapes, the caps may be self-intersecting. In such cases it is better to produce the caps separately using command `SELECT`⇒ `Collect 3D Curves` and `SURFACE`⇒ `Design Fit to Curves.`



The interface to ***building with 3D curves*** is otherwise similar, except that the projection curve(s) may be either B-spline or Bézier, which should be placed in a *3D Projection curve se*t. Additionally, with 3D curves the system asks for the ***interpolation method*** when starting the command. *Bézier interpolation* generally guarantees closer following of the projection curve shapes, while the *B-spline interpolation* may sometimes produce smoother results.

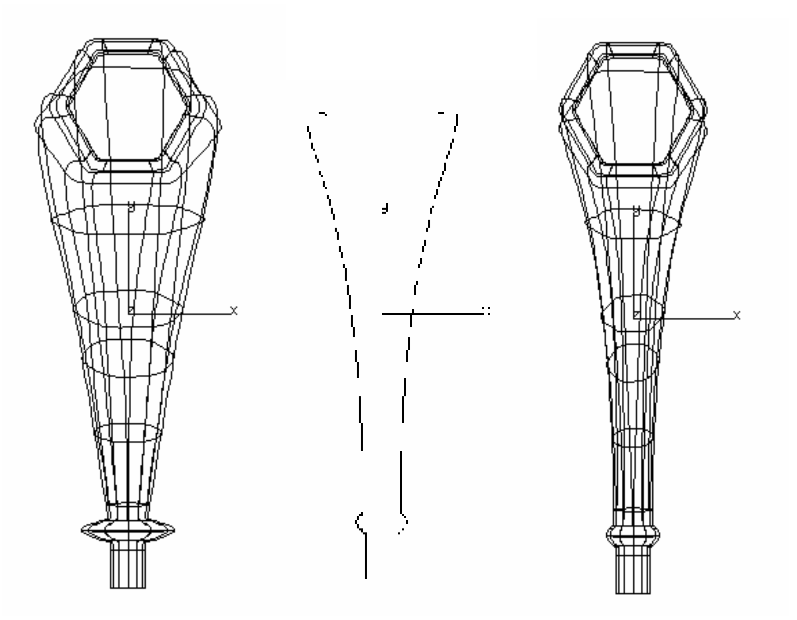
Several surfaces can be built at once if more than three curves are found in the projection set. The system then asks if the projection curves should be taken as multiple single projections, or pair wise as double projections. The system then computes a surface corresponding to each projection curve, or a pair of them, just like normal building.



The limitations to the multiple build method are that secondary projections cannot be used, and only one section curve is allowed. Otherwise, multiple building works with 3D projections just as well as with 2D projections.

## BUILD Þ Create: Secondary Projection

To use this command, the primary projection curves and at least one section curve must already be defined. When this command is first executed, DESKARTES ***creates*** a second projection curve set containing the specified number of curves. The points in these curves are calculated by DESKARTES based on the shape of the surface when viewed from the chosen direction.



The secondary projection curves describe the surface in the same way as the primary projection curves do. However to ensure that the surface created maintains planar section curves, editing of it is restricted. ***Moving the control points*** of the secondary projection curves is only allowed in the direction of the primary projection. For example, if the primary projection is a x-projection, the control points of the secondary projection curves may only be moved in the x-direction.

If the primary projection control points are moved, the secondary projection curves should be ***updated*** to ensure they are aligned with the primary projections. This is achieved simply by repeating the command `BUILD⇒ Create: Secondary Projection`. The same trick applies if the user has managed to move the secondary projection control points in some illegal way, for example by transforming the whole curve(s).

All projection curves must have an ***equal number of control points***. If the number of control points is altered in the primary projection, a similar change must be made to the secondary projection, and vice versa.

The build command only allows for one primary and one secondary projection curve-sets. If this command is executed in an element which already contains a secondary projection, but from a different direction, the existing projection curve set will be deleted. The surface may be manipulated from different directions through secondary projections, but not simultaneously.

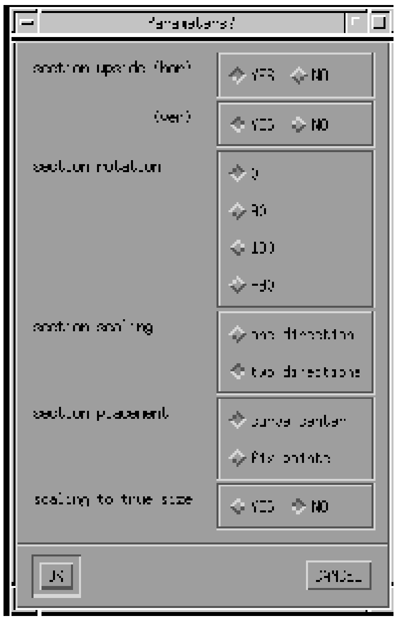
## BUILD Þ Create: Cross Sections

This command ***creates*** one or more of the missing ***cross-section*** curves for the target element. The new section curves may later be edited to adjust the shape of the surface.

A parameter box appears requesting whether all section curves should be created. There is also an option for creating only some of the sections. In this case you can type in the numbers of the required section in the subsequent parameter box. Repeat this procedure for each required curve choosing the finish option when no more are required.

## BUILD Þ Set: Build Parameters

The build parameters control some ***details of the building process***. The build parameters affect the entire surface. They may not be given to separate section curves.


Parameters

section upside (hor) YES NO

(ver) YES NO

section rotation 0 90 180 -90

section scaling one direction two directions

section placement curve center fix points

scaling to true size YES NO

OK CANCEL

When this command is executed, a parameter box appears containing all of the options as follows:

- ***Section upside / rotation***

  When drawing the initial cross-section curves it is often convenient to draw the curve in a certain ***orientation*** of your choosing. When the surface is created, this cross-section orientation may not be satisfactory. Rather than returning to the cross-section and rotating them to new positions, these options can be chosen.

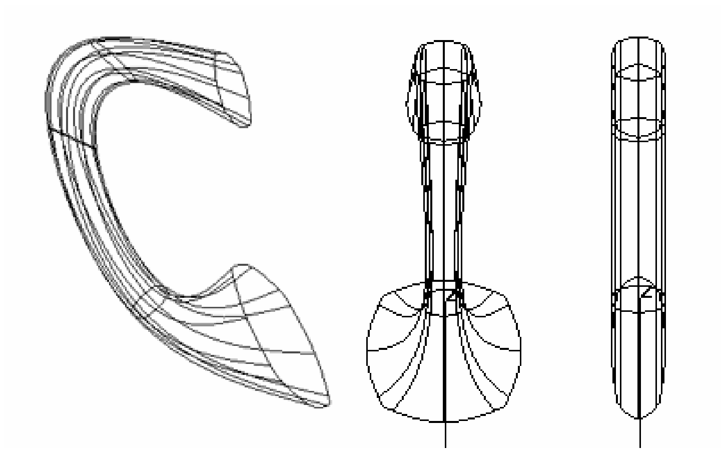
  These parameters do not change the original section curves, they just define how they are used for building.

- ***Section scaling***

  If the primary projection has been defined with two curves, and the secondary projection with one or none, the ***scaling*** of the section curves is only specifically controlled in one direction.

  If the option of scaling in ***both directions*** (the default) is chosen then DESKARTES will make the scaling in both the primary and secondary projections equal.

  If the option of scaling in ***one direction*** is chosen then the cross-section curves will not be scaled in the direction of the secondary projections. In this case, the size of the cross-sections is important. For example, if only one section curve has been defined, scaling in one direction produces a surface of constant width. This is demonstrated below where a handle has been designed with two primary projection curves. Because the distance between these curves varies so does the width of the handle. Setting this option to scaling in one direction produces a handle of constant width.

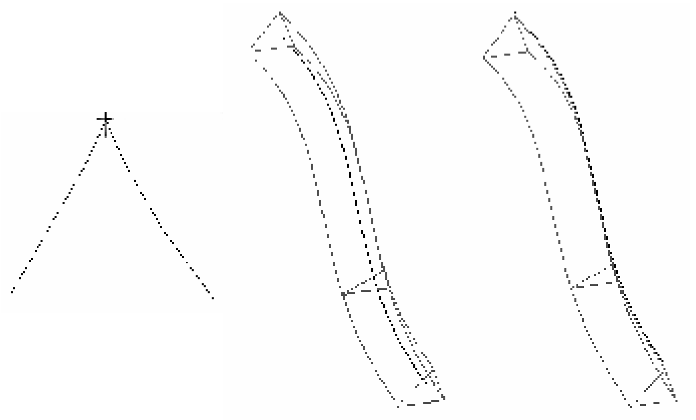


- ***Section placement***

  The section curves are normally positioned on the projection curves at the ***center point*** of the section curve. The center point is calculated by DESKARTES as the center of the section's bounding box. This means it does not matter where the sections are drawn. The resulting surface will always be the same.

  However, sometimes you may need to attach the sections to the projections at a specific point. In this case, choose the option for placement at the ***fix points***. For example, if you have defined just one projection curve, and wish a certain point of the section curve to follow the projection exactly, design the section curve so that the fix point is positioned at the location n question.

  In the example below the fix point of the section is set at one corner. With the curve center option chosen the section is attached to the projection curve at the triangle center. With the fix point option the section is attached at the corner of the triangle.



- ***Scaling to true size***

  With this option, when you next execute the build command DESKARTES will scale the section curves to the actual size they appear on the surface. This has no effect on single section building since no scaling takes place during the build process.

## BUILD Þ Set: Section Parameters

This command should be given to each of the section curves in a built surface separately.

When this command is executed, a parameter box appears containing the options as follows:

- ***Section number***

The number assigned to the section curve determines ***at what point on the projection curve*** the section curve applies. Use the command `BUILD⇒ Show: Section Positions` to show the possible positions and the appropriate number.

The section number is displayed in the Object Window and the message lines.

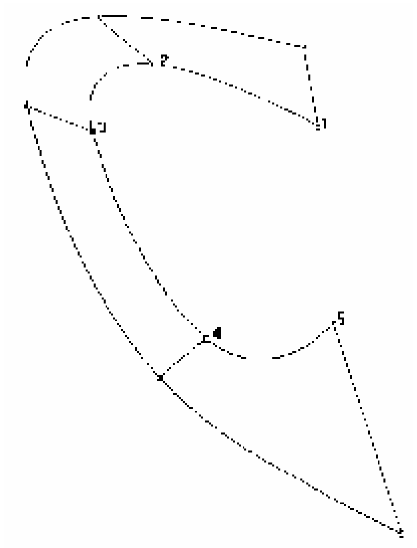
- ***Section weight***

  If some of the section curves are not defined by the user (as is usually the case), DESKARTES computes them as ***weighted averages*** of the existing ones. The way the averages are computed may be adjusted by increasing or decreasing the weight of a given section curve.

  The default value of the section weight is one. This means that all of the curves have an equal effect on the intermediate curves. If one of the forms needs to be emphasized, increase the weight of the section in question. The weights are relative, so changing the weights of all cross-sections from one to two will not affect the shape.

## BUILD Þ Show: Section Positions

This command displays a diagram showing the projection curves and ***numbers where sections can be placed***.



If a section curve has been defined, this position will be drawn in the color of the curve, and the rest in white.

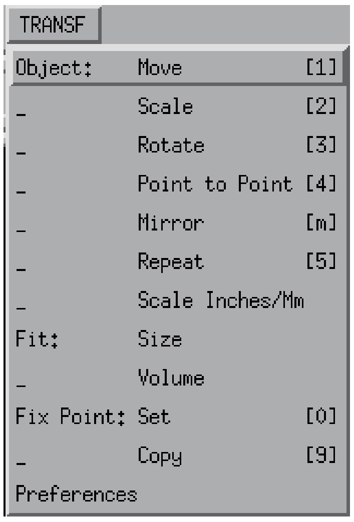
This command is used in combination with the command `BUILD⇒ Set: Section Positions`. First show the positions and note the numbers where you want to assign sections. Then set the numbers of each section you have drawn.

## *TRANSF MENU*

This menu contains commands that ***transform*** objects. Such commands involve moving, scaling, and rotating objects. Some transformations take place around the object's ***fix point***, which also can be set with these commands.

Each of the commands has several different ***operation modes***. Generally the ***mouse button*** pressed after executing the command controls the choice of operation mode.

It is important to realize that transformations applied to a surface will be overwritten if the surface is remade using commands such as rotate, extrude or build. These commands recreate the surface in the position defined by the associated curves. It may be more appropriate to move the defining curves, and then the surface will also move next time the surface is remade.



### `TRANSF` Þ `Object: Move`

This command ***moves*** the 2D or 3D target object. If an element or the root is the target, you are asked if you want to move the projection curves defining the objects as well.

The object's ***fix point*** does not affect the moving operation, instead, the fix point will be moved along with the object.

After executing the command, DESKARTES waits for you to click on a ***mouse button*** to initiate graphical movement. The effect of each mouse button is discussed below. For optimal speed with 3D transformations, only the object's bounding box is displayed during graphical movements. The transformation amount is displayed in the message lines while the transformation takes place.

#### *2D Graphical Object Transformations*

The following rules apply to all ***2D objects***, as well as to 3D objects when viewed from any ***axis direction***:

- ***Left*** mouse button moves freely. 3D objects are constrained to only move in the view plane, not 'in' or 'out' of the screen.
- ***Middle*** button moves in one axis direction only either horizontally or vertically on the screen. The first movement of the mouse determines the direction.
- ***Right*** button restricts movement to the grid snap values.

### *3D Graphical Object Transformations*

The following rules apply to all transformations with 3D objects in any ***3D view*** (not an axis direction):

- ***Left*** mouse button moves in the x direction
- ***Middle*** mouse button moves in the y direction.
- ***Right*** mouse button moves in the z direction.

### *Numeric Transformations*

A more exact method of movement is available. If you click on any ***key*** on the keyboard instead of clicking a mouse button, you will be able to enter the movement values ***numerically***. A dialogue box appears requesting horizontal and vertical movement for 2D objects or x, y and z movements for 3D objects.

## TRANSF Þ Object: Scale

This command ***scales*** the 2D or 3D target object. If an element or the root is the target, you are asked if you want to scale the projection curves defining the objects as well.

Objects are always scaled ***around*** the target objects ***fix point***.

After executing the command, DESKARTES waits for you to click on a ***mouse button*** to initiate graphical scaling. The effect of each mouse button is discussed below. For optimal speed with 3D transformations, only the object's bounding box is displayed during graphical movements. The transformation amount is displayed in the message lines while the transformation takes place.

### *2D Graphical Object Transformations*

The following rules apply to all ***2D objects***, as well as to 3D objects when viewed from any ***axis direction***:

- ***Left*** mouse button scales uniformly in all directions.
- ***Middle*** button scales in one axis direction only either horizontally or vertically on the screen. The first movement of the mouse determines the direction.
- ***Right*** button restricts scaling to the snap values of 0.8, 0.9, 1.0, 1.1, etc.

### *3D Graphical Object Transformations*

All mouse buttons scale uniformly in all directions when viewed from a ***3D view*** (anything other than an axis direction).

### *Numeric Transformations*

A more exact method of scaling is available. If you click on any ***key*** on the keyboard instead of clicking a mouse button, you will be able to enter the scaling values ***numerically***. A dialogue box appears requesting horizontal and vertical scale factors for 2D objects or x, y and z scale factors for 3D objects. Scale values are entered as factors. A factor of 0.5 means the result will be half as big, whereas a factor of 2 means the result will be twice as big.

## TRANSF Þ Object: Rotate

This command ***rotates*** the 2D or 3D target object.

Objects are always rotated ***around*** the target objects ***fix point***.

After executing the command, DESKARTES waits for you to click on a ***mouse button*** to initiate graphical rotation. The effect of each mouse button is discussed below. For optimal speed with 3D transformations, only the object's bounding box is displayed during graphical movements. The transformation amount is displayed in the message lines while the transformation takes place.

### *2D Graphical Object Transformations*

The following rules apply to all ***2D objects***, as well as to 3D objects when viewed from any ***axis directio***n:

- ***Left and middle*** mouse button rotates freely. 3D objects are constrained to only rotate in the view plane, not 'in' or 'out' of the screen.
- ***Right*** button restricts rotation to integer values.

### *3D Graphical Object Transformations*

The following rules apply to all transformations with 3D objects in any ***3D view*** (not an axis direction):

- ***Left*** mouse button rotates around the x-axis.
- ***Middle*** mouse button rotates around the y-axis.
- ***Right*** mouse button rotates around the z-axis.

### *Numeric Transformations*

A more exact method of rotation is available. If you click on any ***key*** on the keyboard instead of clicking a mouse button, you will be able to enter the rotation values ***numerically***. A dialogue box appears requesting a rotation for 2D objects or x, y and z rotation for 3D objects. For 2D objects, a positive angle rotates counterclockwise. A negative value for the angle rotates clockwise.

For 3D rotations, you can predict the ***direction*** by "gripping" the rotation axis with the right hand, so that the thumb points in the axis direction. The other fingers then point in the positive rotation direction.

## TRANSF Þ Object: Point To Point

This command moves the target object so that a ***point on the object matches a point on another object***. The target object may be either a curve, a curve set, a surface or an element. In the last case, the command will affect all surfaces in the element.

After executing the command, you are asked to select the ***other object*** by pointing. Then select the "***move point***" on the target object with the mouse. This is either done with the left button, in which case the nearest knot point on the object is selected, or the middle button, in which case the nearest corner or middle point on the object's bounding box is selected. Selecting the "***target point***" on the other object in a similar manner completes the command.

## TRANSF Þ Object: Mirror

This command ***mirrors*** the target object to the other side of a line or plane that passes through the fix point. If an element or the root is the target, only the 3D objects within it are transformed.

A parameter box appears requesting the axis ***direction*** for mirroring. For 2D objects, the choice is to mirror horizontally or vertical. For 3D objects, the choice is the x, y or z-axis

## TRANSF Þ Object: Repeat

This command ***repeats*** the previously executed (2D or 3D) transformation. This is a very useful function.

One use for this would be to move an object to the desired position in small steps. First, make a small movement either graphically or numerically. Then use this command to repeat the movement again and again until the desired position is achieved.

A second example of the command is to combine its use with the copy command. For example if you need a design element repeated around a circular surface say a vase. After designing the vase and the feature surface make a copy of the feature using `OBJECT⇒ Copy`. Now perform a numeric rotation around the z-axis by 30 degrees. Now simply by repeating the `OBJECT⇒ Copy` command followed by `TRANSF⇒ Repeat` a number of times the feature can be repeated around the vase. Since both of these commands have shortcuts this is achieved by simply pressing j (copy) and then 5 (repeat) a number of times.

## TRANSF Þ Object: Scale Inches/Mm

This command scales a model from ***inches to millimeters***, or the other way around.

You are asked whether you want to apply the scaling to the target object or to the whole model. The target or model will be scaled up or down as appropriate by a factor of 25.4.

## TRANSF Þ Fit: Size

This command transforms the target object by scaling and moving to a ***given size***.

A dialogue box appears requesting the size and position of the resultant model. These are entered as ***extent values*** from which DESKARTES will calculate the necessary transformations. The values that appear in this box by default are the current sizes of the object. If you do not change these values the command will have no effect on your model.

If you have made a design which you decide should now be an exact height, enter this value as the maximum Z value and 0 as the minimum Z value for the required size and DESKARTES will do the rest.

## TRANSF Þ Fit: Volume

This command scales the target faceted object so that it becomes the size of a ***specified volume***. You may scale the object in either X, Y or Z, or all directions at once.

The command should be applied to a faceted model created with the command `SURFACE⇒ Change: Faceted`, but the scaling factors are ***memorized*** by the system. So after you have scaled the faceted model to the given volume, you can repeat the operation and scale the actual surfaces.

## TRANSF Þ Fix Point: Set

The scaling, rotation and mirroring commands are executed relative to the object's ***fix point***. The fix point is displayed as a cross on the screen when a transformation command is given. The fix point can be repositioned but is by default located at the origin. If the object is moved, the fix point moves too.

After executing the command, DESKARTES waits for you to click on a ***mouse button*** to locate the fix point. The effect of each mouse button is discussed below.

### *2D Graphical Object Transformations*

The following rules apply to all ***2D objects***:

- ***Left*** mouse button locates freely.
- ***Middle*** places the fix point on the nearest handle point of the target object. The nine handle points are located at the corners, side mid points, and the center of the object's bounding box.
- ***Right*** button locates the fix point at the origin.

Any fix point assigned to a curve-set is copied to all the curves within the set, and vice versa - if a single curve is assigned a fix point, it is assigned to the curve set as well.

### *3D Graphical Object Transformations*

The following rules apply to all ***3D objects***:

- ***Left*** mouse button locates the fix point at the nearest point on the target surface.
- ***Middle*** mouse button places the fix point to the object's center. With 3D objects in an axial view, the middle button applies the grid snap value.
- ***Right*** button locates the fix point at the origin.

### *Numeric Transformations*

A more exact method positioning the fix point is available. If you click on any ***key*** on the keyboard instead of clicking a mouse button, you will be able to enter the position ***numerically***. A dialogue box appears requesting horizontal and vertical position for 2D objects or X, Y and Z position for 3D objects.

## TRANSF Þ Fix Point: Copy

This command can be used to ***copy the last shown fix point*** to the target object, for instance, the fix point of a surface to a whole element.

## TRANSF Þ Preferences

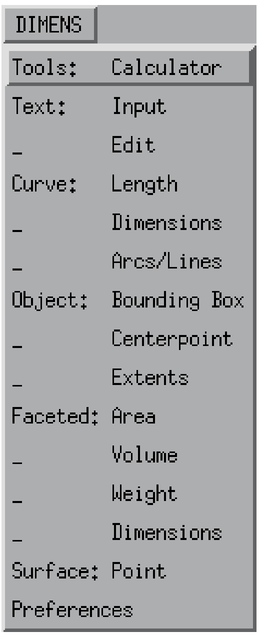
This command sets some ***preferences*** that control the fix point for 3D transformations take place, as well as how 3D transformations are reflected to 2D geometry.

After executing the command a parameter window appears with two options:

- ***Freeze 3D Fixpoint*** – setting this option to yes stabilizes the fix point so that the same fix point will be used for all subsequent transformations. This can save having to copy the fix point to several objects and is useful if the same fix point should be applied to transformations with several objects. The location of this global fix point is taken as the current target objects fix point.
- ***Transform projections*** – this determines whether the 2D projection curves should be transformed too when the element is moved or scaled.

## *DIMENS MENU*

This menu contains various commands for showing object ***dimensions***, and their representation as ***drawings***. The ***volume*** enclosed by faceted surfaces and their surface ***area*** can also be computed.



### `DIMENSÞ Tools: Calculator`

This command brings up a graphical ***pocket calculator*** on the screen. Use it for any mathematical calculations you may require.

### `DIMENSÞ Text: Input`

This command enables you to ***type text*** onto the target element. A drawing curve set is created to hold the text.

After executing the command locate with the mouse the position where you want the text to be placed, and then type in the text. Use `backspace` to delete characters, and `return` for line breaks. Click the mouse again to end typing. The font used is a fixed size screen font. This means that whenever the text is displayed it will always appear at the same size independent of the zoom level.

Multiple lines of text will be created as individual text objects. Text can be moved using the command `TRANSF⇒ Object: Move` but it cannot be scaled or rotated.

### `DIMENSÞ Text: Edit`

This command allows the existing text to be ***edited***. The text object should be the target object. After executing the command, a dialogue box appears containing the text that can be edited in the normal way.

### `DIMENSÞ Curve: Length`

This command reports the ***length of a curve***. The curve can be either free-form, or polygonal.

## DIMENS Þ Curve: Dimensions

This command allows ***dimension drawings*** to be created of the target curve or curve-set. The curves may be of any type, they don’t even need to be of the same type. The desired curves may be gathered into a single element before dimensioning with the command `SELECT⇒ Collect: Actives`.

After executing the command, the command changes to a separate ***dimensioning mode*** where you may create the dimensions. The functions of the dimensioning mode are discussed in detail in the Editing Functions part of this Manual.

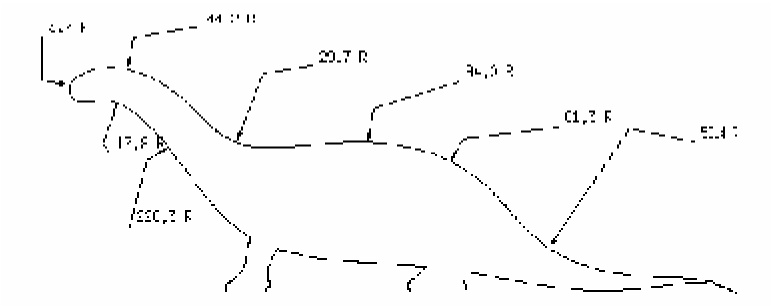
When exiting curve dimensioning, the dimensions may be ***stored*** in a separate curve set. This allows the curves and dimensions to be plotted using a different pen size and colors.

## DIMENS Þ Curve: Arcs/Lines

Normally curves are stored by DESKARTES as free-form curves, with constantly changing curvature. This command splits any curve, such as a free-form B-spline/ Bézier curve or plane slices of 3D models, into ***circlular arcs and straight line segments***.

The command should generally be used before entering `Curve Dimension` mode, as you can then exactly measure radii on the curves, and select the straight-line end points as dimension points. You may also use this command before writing curves into a DXF file, so that the receiving system better understands the curve measures.

As parameters, the command requires:



- ***radius tolerance***: determines how much the arc radii are allowed to differ from the actual curvature of the curve;

- ***join angle tolerance***: determines the greatest allowed angle between consecutive arcs in the result curve;

- ***sharp corner angle***: determines which corners are interpreted as sharp corners in the original curve.

Setting the parameter values optimally depends strongly of the application. Tightening the two first tolerances results in a better approximation, but it increases the number of curve segments (arcs) produced.

After execution of the command, the approximated curve is displayed on top of the original, to enable visual checking of the result. Those arcs which were recognised as exactly defined fillets are colored as red. Approximated arcs are colored with two shades of pink colors. Straight lines are colored with green.

The result is given as a separate Bezier curve for each arc. If desired, the separate curves can be also joined into a one with command `CURVE⇒ Combine: Join All`. Further checking of the result can be done by setting knot display on with command `CURVE⇒ Knots: Show`.

## DIMENSÞ Object: Bounding Box

The command calculates the dimensions of the ***bounding box*** of the target object. The bounding box is the smallest rectangular box that will just enclose the target object. The X, Y and Z values of this bounding box are displayed in the message lines.

## DIMENSÞ Object: Center Point

This command calculates the ***center point*** of the bounding box of the target object. The bounding box is the smallest rectangular box that will just enclose the target object. The X, Y and Z values of the center point are displayed in the message lines.

## DIMENSÞ Object: Extents

This command reports the ***extents*** of the bounding box of the target object. The bounding box is the smallest rectangular box that will just enclose the target object. The height, width and depth values of this bounding box are displayed in the message lines.

## DIMENSÞ Faceted: Area

This command calculates the surface ***area*** of the target ***faceted model*** or an element containing several faceted models. Surfaces may be converted to a faceted model using command `SURFACE⇒ Change: Faceted`.

The surface area is displayed in the message lines. The volume is expressed in ***square units***. You should interpret the units in the same way as your main model. For instance, if you intend the units to be millimeters, the volume “1000 square units” would be 1000 square millimeters.

## DIMENSÞ Faceted: Volume

This command calculates the ***volume*** of the target ***faceted model*** or an element containing several faceted models. Surfaces may be converted to a faceted model using command `SURFACE⇒ Change: Faceted`.

The surface volume is displayed in the message lines. The volume is expressed in ***cubic units***. You should interpret the units in the same way as your main model. For instance, if you intend the units to be millimeters, the volume “1000 cubic units” would be 1000 cubic millimeters.

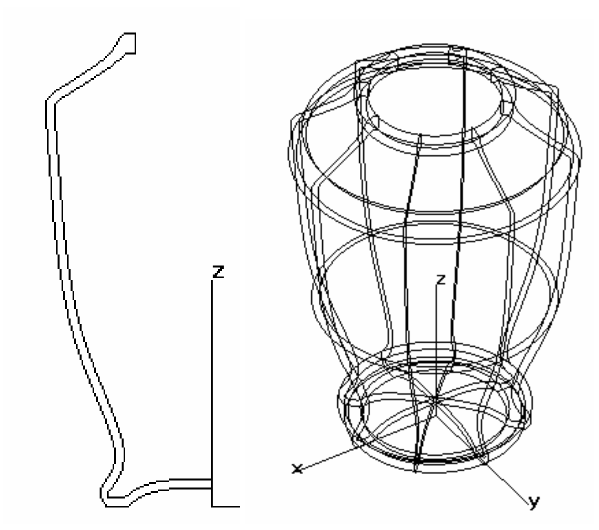
**Important:**

For the calculation of volume to be accurate the object must form a ***closed surface*** or ***solid model***. In other words, surface must have no holes in it, it must be “watertight” and it may have no overlapping surfaces.

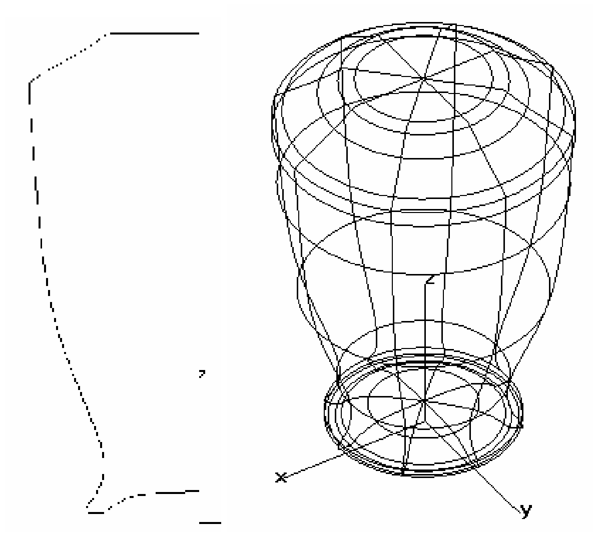
In addition, the faces of the faceted models must be consistently oriented having the ***same normal directions***. If you convert an element of surfaces into a faceted model with a single `SURFACE⇒ Change: Faceted` command, and use the check directions option, then the normals are guaranteed to be oriented correctly.

If you compute the faceted model without checking the directions, then shading the model with ***single shaded shading*** option (selected from `DISPLAY⇒ Preferences`) shows you the problem facets in dark blue color. Any such facets will mean errors in volume computation.

As an example say you have modeled a vase and you convert this to faceted format and asked for the volume. DESKARTES will report the volume enclosed by the surface. This will represent the volume of material needed to make the vase not the volume of liquid that can be held in the vase. To calculate the volume of liquid you will need to model the liquid. This can easily be achieved by copying the whole vase element and editing the projection curve so that it represents the liquid.



This is a model of a vase and the volume calculated would be the volume of material needed to make the vase.



This is a model of the liquid and the volume calculated would be the volume the vase could hold.

## DIMENS Þ Faceted: Weight

This command calculates the weight of a faceted model.

The command asks as a parameter the material's *density* as a value of grams per coordinate unit. This parameter value is memorized by the system from one modeling session to another, so when working with just a single material type its value has to be keyed in only once.

Technically the command follows exactly the same rules as volume calculation. Most importantly, the model must form a proper solid in order for the weight calculation to make sense.

## DIMENS Þ Faceted: Dimensions

This command allows displaying ***point wise dimensions*** of faceted models. Information about points, the distances between points, and the radius defined by three points can be obtained.

The system remains in a "dimension" ***mode*** for this command. During this time, certain keys indicate the type of dimension required as described below:

- **p** - Returns the XYZ-coordinates of the point located by pointing and clicking with the left mouse button.
- **d** - Returns information about the distance between two points. Pointing and clicking with the left mouse button indicates the two points.
- **r** - Returns information about the radius defined by three points. Pointing and clicking with the left mouse button indicates the three points.
- **a** - Returns information about the angle defined by three points. Pointing and clicking with the left mouse button indicates the three points.
- **b** - Returns information about the overall size of the model and other information such as the number of facets.
- **w** - exits from the dimension mode and stores a picture of the screen which can be viewed or printed.
- **q** - exits from the dimension mode without storing a picture.

With each function, the user must locate the text using the left mouse button before entering the next dimensioning function.

## DIMENS Þ Surface: Point

This command displays the ***point coordinates*** on a Bézier or B-spline surface. It also reports the point's coordinates in the parameter plane.

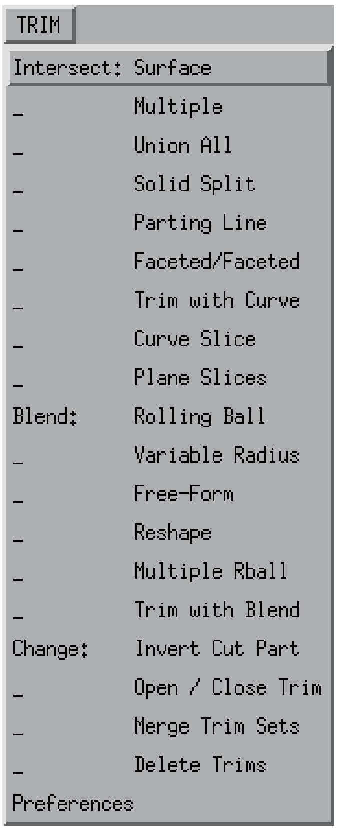
The required point is selected from the surface by pointing and clicking with the left mouse button. The values are displayed on the screen at a location chosen by pointing and clicking with the mouse again.

## DIMENS Þ Preferences

This command determines how many ***decimal places*** should be shown for curve dimensioning.

## *TRIM MENU*

The `Intersect` and `Blend` commands in this menu ***intersect*** surfaces with each other, and compute smooth ***blend surfaces*** between them. The commands' operation modes are controlled with `TRIM`⇒ `Preferences`.



The ***trim curves***, which define the area to be removed in the surface, are stored as 2D curves as explained here.

## *Trim Curves and the Parameter Plane*

Each surface in DESKARTES can be considered to be made by taking a rectangular rubber sheet and stretching this over a framework of curves to form the correct shape. If the rubber sheet is removed and allowed to shrink to its original shape, this represents what is called the ***parameter plane***. The number of points in the defining projection and cross-section curve determines the size of this parameter plane.

For instance, rolling the parameter plane around so that two of its boundaries meet each other would make a cylinder surface. Bending the other two boundaries together too would make a torus.

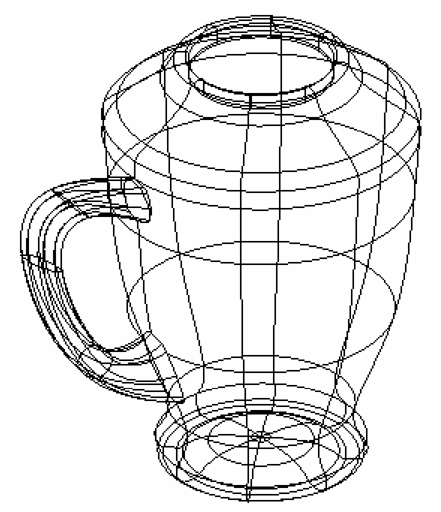
The concept of the parameter plane is extremely important when dealing with trimmed surfaces. A trimmed surface is one where portions of the surface are cut away, normally by calculating an intersection with another surface. The cut is stored by DESKARTES as ***two-dimensional trim curves*** which are ‘drawn’ on the parameter plane. The trim curves can be considered as the lines which mark areas needed to be cut out in the rubber sheet before it is stretched into the shape of the surface.

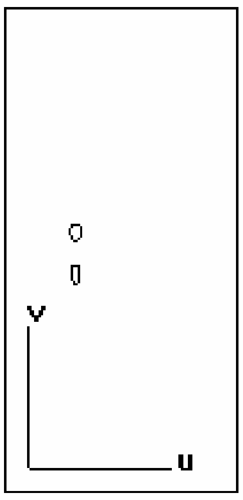
Since the rubber sheet is distorted during the stretching, the shape of the areas to be cut will also be distorted. The curves drawn on the parameter plane may not exactly be the shape of the final holes in the surface, though they will usually be somehow similar.

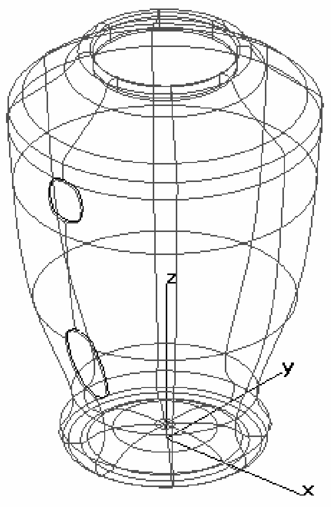
In order to cut out a part of the surface, the ***trim curve must be closed***. Closed trim curves are denoted in the Object Window as ***polygons***.

Open trim curves (polylines) have no effect on the surfaces, thus, they are not taken into account in modeling or visualization computations, nor are they transported to the different data exchange formats.

An example of this is shown below where a coffeepot has been modeled and trimmed with a handle. This produces two holes in the body surface which are stored as two polylines drawn on the parameter plane.







## TRIM Þ Intersect: Surface

This command computes the ***intersection*** between the target surface and another surface.

If the resulting intersection curves forms a closed area of the surface, that area is cut away from the surface. For the treatment of open intersections, see the command `TRIM⇒ Preferences`.

For this command, the target surface is called "***this surface***". The second surface is called "other surface".

Normally pointing and clicking with the left-hand mouse button after executing the command indicates the ***other surface***. Sometimes it may be difficult to point exactly at the surface that you want. The alternative way is to click any key on the keyboard, which brings up a dialogue box into which the name of the element where the surface is located and the number of the surface is entered.

After the other surface has been indicated, the program computes the intersection, and makes a guess at the parts of the surface which are to be cut away, displaying the result on the screen. If the guess that DESKARTES has made is incorrect, the user can change the result for either surface, by ***inverting*** this or the other surface.

If you intersect a previously trimmed surface, the system automatically combines, or ***merges*** the new trim curves with the old ones. The basic rule for the automatic combining of trimmings is that new trim commands always ***cut parts away*** of the old model, never adding anything. See the command TRIM⇒ `Change`: `Merge Trim Sets` for further discussion of this.

## TRIM Þ Intersect: Multiple

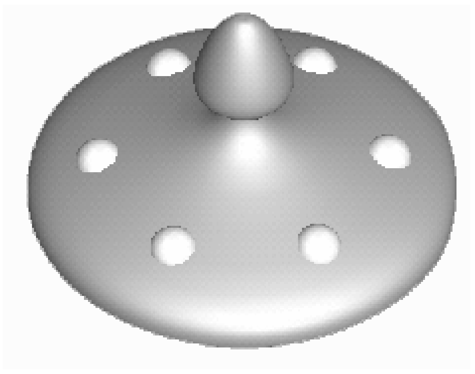
This command ***cuts*** the target surface with ***all other surfaces*** that are shown on the screen.

The command first displays a series of arrows on the target surface, and asks whether you want to keep the parts of the other surfaces on the side of the arrows, or the opposite side. Further, there is an option to handle the case where the multiple surfaces consist of individual, separate surfaces. Use this option when the multiple surfaces look more like "separate" than "solid".

After the parameters have been given the intersection of the target surface with each of the other surfaces on the screen is calculated in turn.

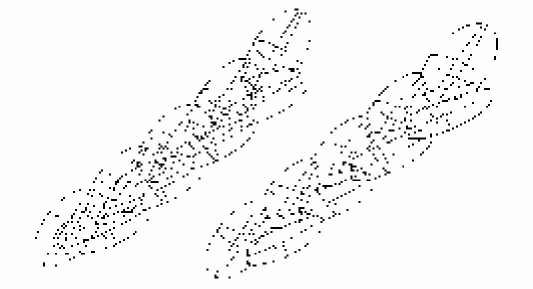
If you intersect a previously trimmed surface, the system automatically combines or merges the new trim curves with the old ones. The basic rule for the automatic combining of trimmings is that new trim commands always cut parts away of the old model, never adding anything. See the command TRIM⇒ `Change`: `Merge Trim Sets` for further discussion of this.

To trim all the spherical parts with the lid shown below the lid surface must be the target.



## TRIM Þ Intersect: Union All

This powerful command trims all the surfaces on the display with each other, and forms the union of them all.

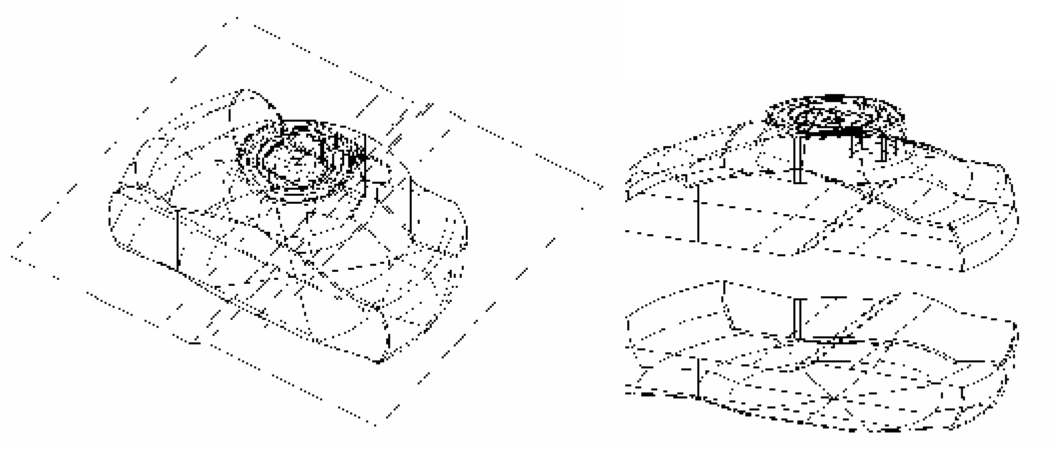


The surfaces should be consistently oriented (c.f., command `SURFACE⇒ Normals Orient All`) before issuing the command.

As a general rule, the command works best with non-trimmed closed surfaces**. In particular, the surfaces should not have been previously trimmed with each other**.

## TRIMÞ Intersect: Solid Split

This command splits all the surfaces on the display with the target surface. If the model to be split forms a solid, the result is two solid halves of the model, stored in two separate elements.



## TRIMÞ Intersect: Parting Line

This command calculates the ***split line***, or parting line of the target surface. The split line is calculated as horizon of the surface when viewed from the Z direction.

The program asks as parameters the computation ***tolerance*** ("normal tolerance"), as well as the ***joining accuracy*** for the resulting 3D parting line pieces. Small normal tolerance improves the accuracy, while a large joining accuracy may help the system to better construct the 3D parting line.

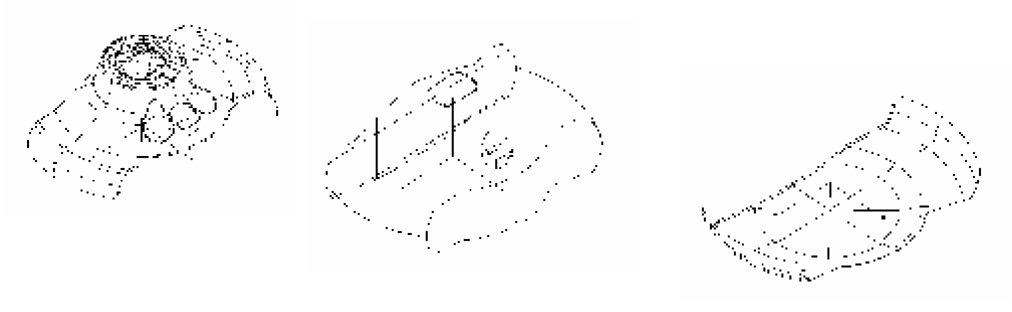
The surface model is split into three or four elements as follows:

“upper”, containing the up facing parts of the model

“lower”, containing the down facing parts of the model

“parting”. This element contains both the parting curves that pass through curved surfaces, as well as the vertical walls of the model.

“problem”, contains problem areas of the model, if any.

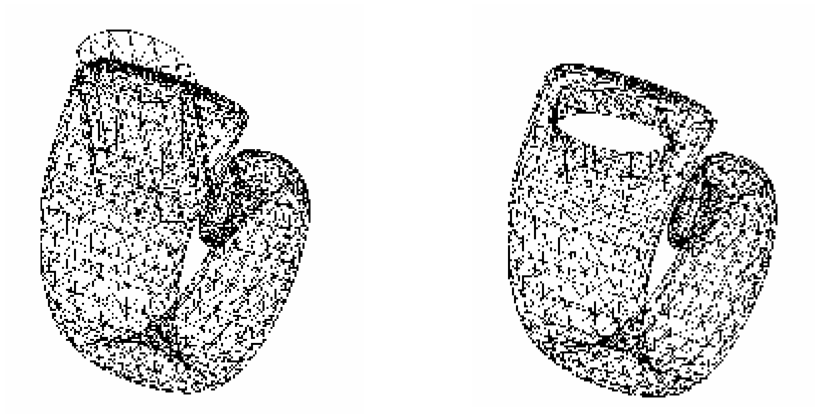


If there exist vertical walls within the model, the user must decide if they represent design errors or if they can be used for the mold. In the latter case, the user can interactively place them from the "parting" element into the "upper" or "lower" elements using `OBJECT⇒ Cut & Paste`.

Note that the parting curve in the "parting" element does not include curves at surface boundaries. The complete parting line can be obtained by passing either the "upper" or "lower" element through the command `FORMAT⇒ Surface Verify`.

## TRIMÞ Intersect: Faceted/Faceted

This new command ***cuts a faceted object with another one***, and produces a Boolean combination of the models. This way even complex surface models can first be faceted, and only after that "trimmed" with each other.



The alternatives for the Boolean operation are:

- ***union of this and other***
- ***subtract this from other***
- ***subtract other from this***
- ***intersection of this and other.***
- ***split into four parts***

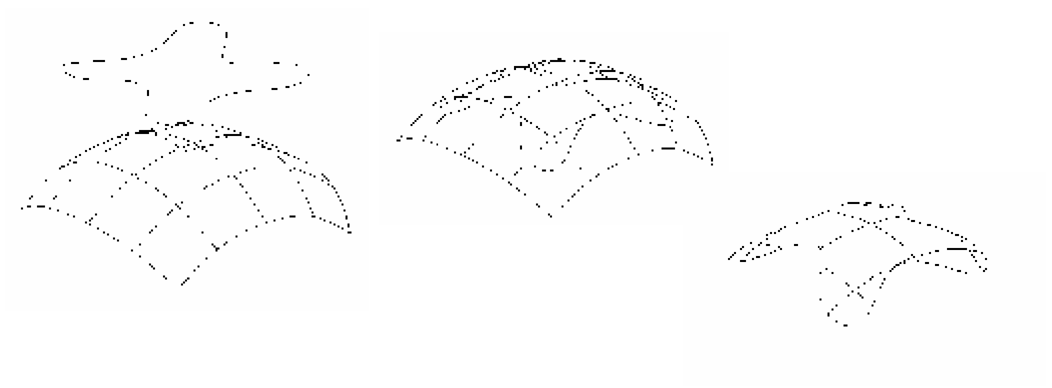
where "this" denotes the target object, and "other" is the second faceted model which was indicated to the command by graphic picking.

While the meaning of the first four options is obvious, the fifth option splits both the faceted models into two parts; the parts that are not required can afterwards be thrown out by the user.

In order for the command to work, both the faceted models should have their normal directions oriented consistently, and (with the first four parameter options) be proper solids. To ensure this is the case, the models can first be passed through command `FORMAT⇒ Faceted: Repair/Orient`.

## TRIMÞ Intersect: Trim with Curve

This command ***trims a surface with a 3D curve***. A typical way to use the command is first to project a separately designed curve onto the surface (c.f., command `CURVE⇒ Change: Project`), after which trimming with it can take place.



To execute the command, the 3D curve should be chosen as target object. The command then asks for the surface to be indicated by graphic picking. At the end of the command's execution, it asks which part of the surface to keep (just like command `TRIM⇒ Intersect: Surface`.)

The 3D curve can be of any representation. However, in order to cut something away from the surface, the curve should be a closed one. Also, the 3D curve should be located relatively close to the surface.

## TRIMÞ Intersect: Curve Slice

This command computes the intersection between the target surface and all the other displayed surfaces. However, it does not actually cut the surfaces, but stores the resulting intersections as ***3D curves***.

As a typical application, these curves are most often used as projections for 3D Building. For this, the curves need first to be *approximated* into the Bezier form.

## TRIMÞ Intersect: Plane Slices

This command intersects all the surfaces in the target element with a set of ***parallel planes***, and collects the resulting 3D curves in a separate element.

A parameter box appears requesting the ***plane direction*** (which can be set orthogonal to the X, Y or Z axis), the ***bottom and top*** plane positions, and the ***distance between the plane***s. The default top and bottom values are the minimum and maximum dimensions of the target object.

Another parameter, called ***tolerance,*** specifies how accurately the plane intersections are to be computed. The smaller the value entered the more accurate the curve calculation.

As the final parameter, the system asks if it should expect ***small details*** in the model. The computation will be faster if small details are not checked for.

One use of this command is to review the design as it progresses. You may view the slices from different directions and inspect them in detail to ensure the shape of the model is satisfactory.

It is also theoretically possible to use these slices directly for controlling some of the rapid prototyping machines by using one of the various data exchange links with DESKARTES. However this is not the recommended method of interfacing with such machines, it is better to use a faceted model output in the STL file format.

## `TRIMÞ Blend: Rolling Ball`

This command creates a r***olling ball blend*** between the target surface and another surface, which must intersect. In addition, it trims the surfaces where the blend touches the surfaces.

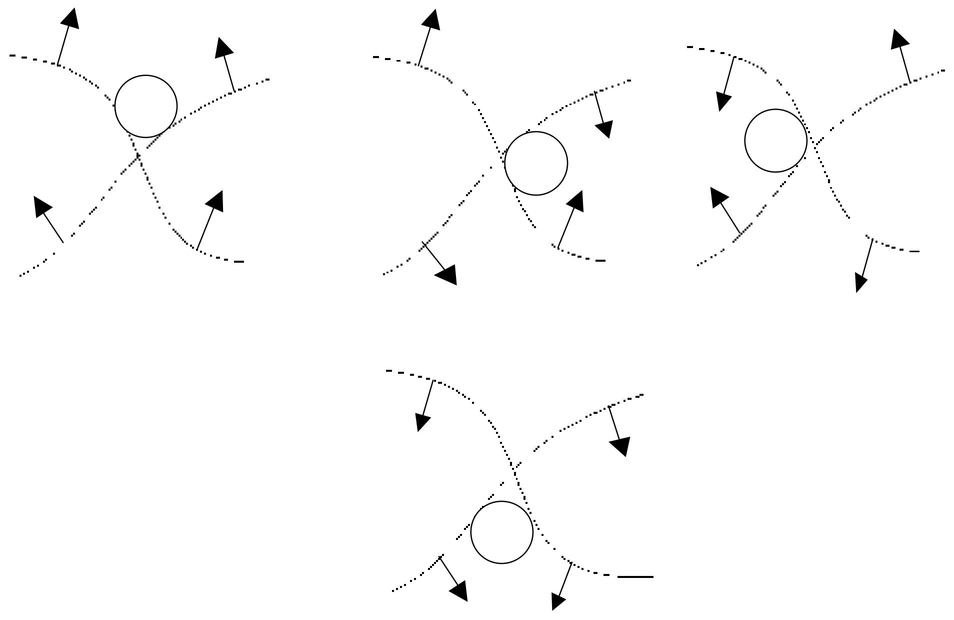
The blend surface created by this command resembles part of the surface of a ball rolling around the two surfaces. The blend surface has a constant radius, circular cross-section and it joins the two surfaces tangentially.

Selecting the surfaces to be blended is done exactly the same way as with command `TRIM⇒ Intersect: Surface`.

Before computing the actual blends, the program computes the ***intersection(s)*** between the two surfaces. If there are several intersections DeskArtes displays numbers on each intersection, and asks for which intersection number you wish to calculate the blend.

The command next displays a series of ***arrows*** on the two surfaces that will be used to indicate which blend surface is to be calculated.

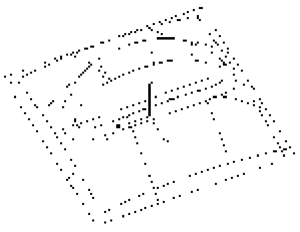
A parameter box is then displayed requesting the ***radius*** of the blend and the ***direction*** of the blend. Choosing whether the ball will be rolled on the side of the surface pointed to by the arrows (normal side) or the opposite side (opposite side) indicates the direction. This needs to be indicated for both surfaces giving four possible combinations (normal-normal, normal-opposite, opposite-normal and opposite-opposite) producing four possible blends as represented below.



If the intersection does not form a closed loop, it is called an ***open intersection*** and the system also asks where the blend should terminate. The options are at the trim curve or from the edge of either of the surfaces. Thus, it is possible to extend the blend outside the actual surfaces.

After entering these parameters DeskArtes calculates and displays the blend. If the blend is satisfactory it can be accepted, if not reject the blend. If you reject the blend, you will have the option of changing some of the parameters and recalculating.

If there were multiple intersections, the command cycles through the above operations allowing you create blends at each intersection if required. Notice that you can change the directions, and use different radii for different intersections.



**Note:** Unlike surface intersections, blends *cannot* be computed over ***sharp corners***. They should be redefined with small round corners before blending.

## `TRIMÞ Blend: Variable Radius`

This command creates a ***variable radius rolling ball blend*** between the target surface and another surface, which must intersect. In addition, it trims the surfaces where the blend touches the surfaces.

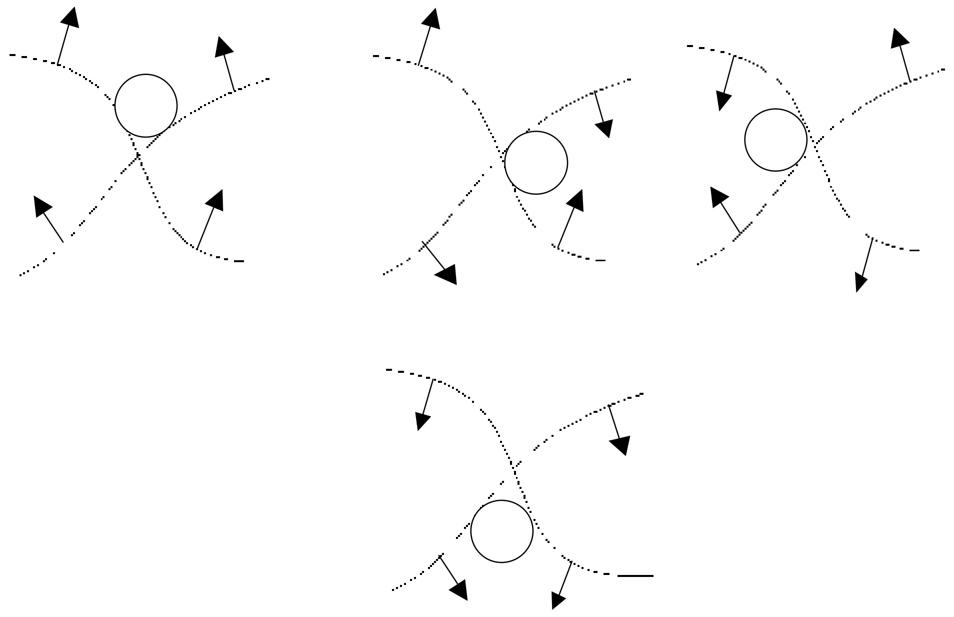
The blend surface created by this command resembles part of the surface of a ball rolling around the two surfaces. The blend surface has a varying radius circular cross-section and it joins the two surfaces tangentially.

Selecting the surfaces to be blended is done exactly the same way as with command `TRIM⇒ Intersect: Surface`.

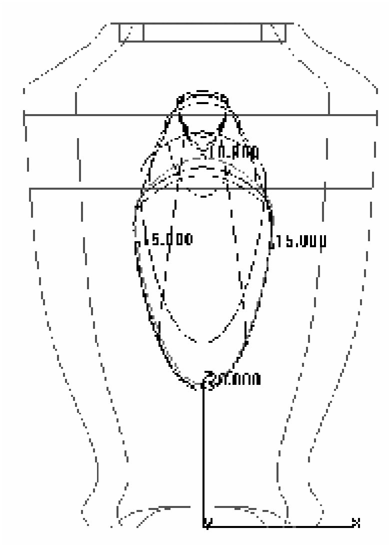
Before computing the actual blends, the program computes the intersection(s) between the two surfaces. If there are several intersections DESKARTES displays numbers on each intersection, and asks for which intersection number you wish to calculate the blend.

The command next displays a series of arrows on the two surfaces that will be used to indicate which blend surface is to be calculated.

A parameter box is then displayed requesting the radius of the blend and the direction of the blend. Choosing whether the ball will be rolled on the side of the surface pointed to by the arrows (normal side) or the opposite side (opposite side) indicates the direction. This needs to be indicated for both surfaces giving four possible combinations (normal-normal, normal-opposite, opposite-normal and opposite-opposite) producing four possible blends as represented below.



The ***radius values*** now need to be defined. This is done ***graphically*** by pointing and clicking with the *left-hand mouse button* at a position on the trim curve and then typing in the desired radius value with the dialogue box that appears. Repeat this process, defining new radius values at as many different places as required. To ***cancel*** a value you gave, click at the *right mouse button*. If you need to ***pan*** whilst defining the radii, you may use the *middle mouse* button to start the normal panning operation.



To end the radii definition process, select the ***ready*** to blend option in the dialogue box or click any key on the keyboard. The program then computes the blend surface.

If there were multiple intersections, the command cycles through the above operations allowing you create blends at each intersection if required.

**Note1:** Blends *cannot* be computed over ***sharp corners***. They should be redefined with small round (“tight”) corners before blending.

**Note2:** It is *not* allowed to define a large variable ball radius value *inside* a ***tight corner***, whose radius is less than the ball radius. Instead, you must define the large radius at *two locations* just *before and after* the tight corner.

## TRIM Þ Blend: Free Form

This command creates a ***free-form blend*** between two user-defined trim curves on two different surfaces.

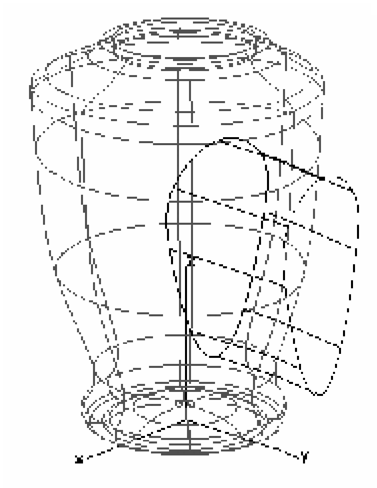
The blend surface created by this command resembles part of the surface of a ***rubber ball*** rolling around the two surfaces. The blend surface is squeezed between the two surfaces and it joins the two surfaces tangentially.

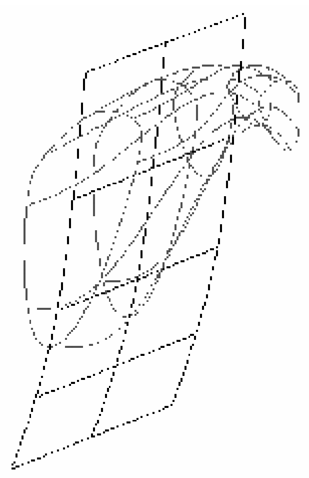
There are several operations required to set up for this command. In general, you need to define a ***boundary Bézier curve*** (in some way) on each surface**.** There should exist one, and only one, such curve within the trim curve-set of each of the surfaces for the command to work.

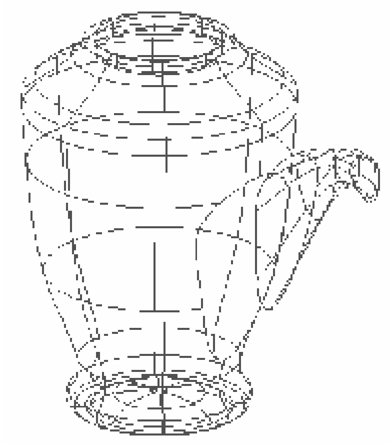
There are several ways to create the boundary curves. One way is to first compute a rolling ball blend between the surfaces. Then change the trim curves to Bézier representation using `CURVE⇒ Change: Approximate` and *edit* the curves to a give a new boundary curve shape.

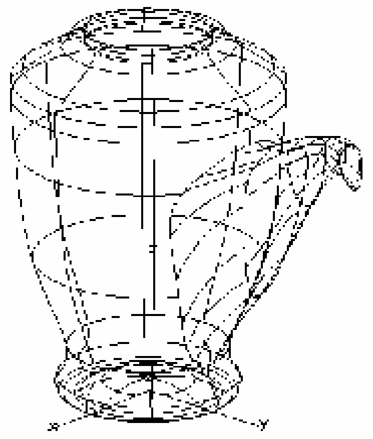
A second more general method is to use ***tool surfaces***. A tool surfaces is an ordinary surface, created by any appropriate technique (extrude, rotate, build etc), which intersects one of the surfaces to be blended. This tool surface should then be `TRIM⇒ Intersect: Surface` with one of the blend surfaces to create a trim curve. After this operation the tool surface can be made inactive (or even deleted!) as it is not really part of the model. This operation can then be repeated with a second tool surface for the second surface to be blended.

The first two of the four diagrams below show the two tool surfaces in position. The third diagram shows the results of the trimming in readiness for the blend operation. The next step is to change the two trim curves to Bézier representation using `CURVE⇒ Change: Approximate` to a give a boundary curve shape.









Once the boundary trim curves are defined, the command `Blend Free Form` can be executed which will create the actual blend surface as shown in the fourth diagram above. Selecting the surfaces to be blended is done just like with command `TRIM⇒ Intersect: Surface`.

There are two alternatives of how to compute the blend using ***rib control*** *between the surfaces*, or *along curve length*. Use the along curve length option only if the blend creation fails with the between the surfaces option.

After the blend is computed, the system asks if the ***trim polygons*** should be calculated or not. Answer NO, if you already have the trim curve(s) existing as polylines, if you copied them before converting to Bézier curves otherwise answer YES.

## TRIMÞ Blend: Reshape

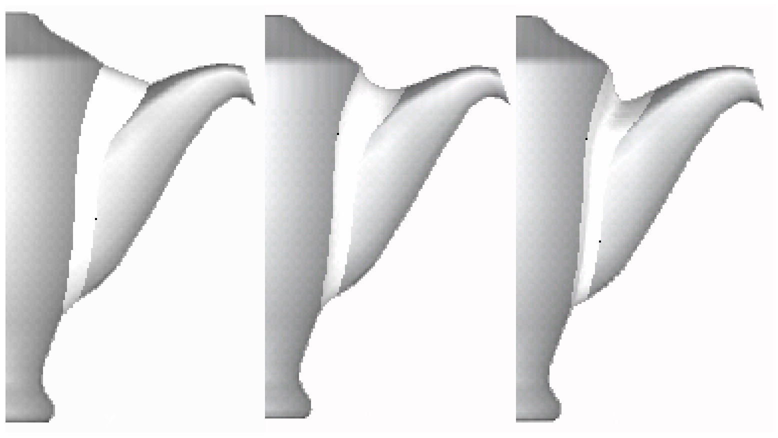
This command changes the ***shape of the blend surface***, while maintaining its tangency with the connected surfaces.

The blend should be selected as the target first. Varying two ***shape factors*** (one for each edge of the blends) alters the shape of the blend.

By default, the shape factors will be equal to 1.0 at both blend boundaries. The blend shape can be made tighter by assigning higher shape factors, or looser with lower ones.

With zero shape factors at both ends the blend will form a linear chamfer. The maximum value for the shape factor is 2.0 as above this value the blend may form a loop.

Examples of the extreme values are shown below. On the left the shape factor is 0 on both surfaces. In the middle the values are 1 – the original value. On the right the values are both 2.

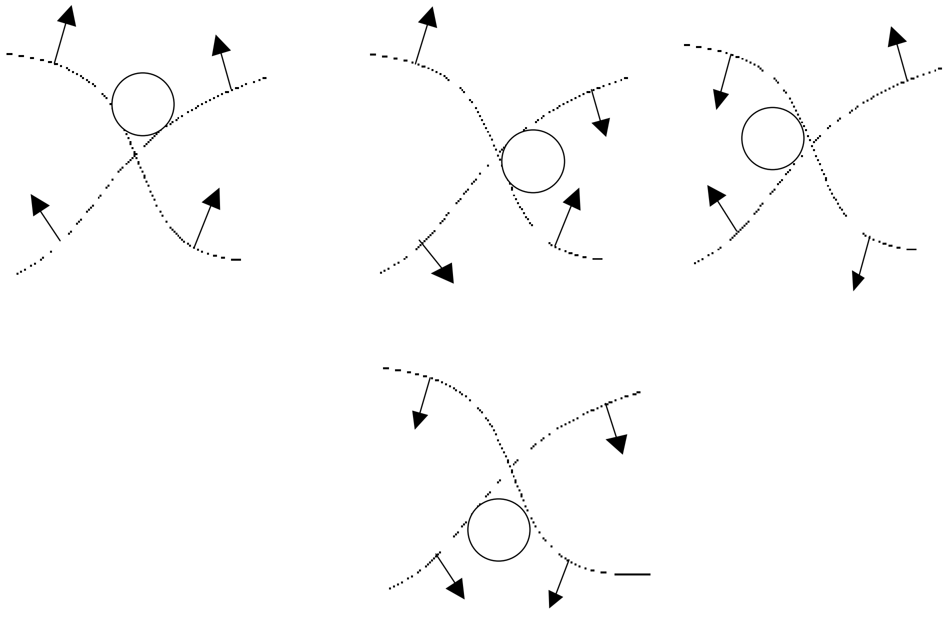


## TRIM⇒ Blend: Multiple Rball

This command generates ***rolling ball blends*** between the target surface and ***all other surfaces*** visible on the screen. It is the blend equivalent of the `Trim Multiple` command.

In preparation for this command you must first check the directions of all of the ***surface normals*** using the command `SURFACE⇒ Normals: Show`. They should all be oriented consistently. You can change them as required with `SURFACE⇒ Normals: Change` .

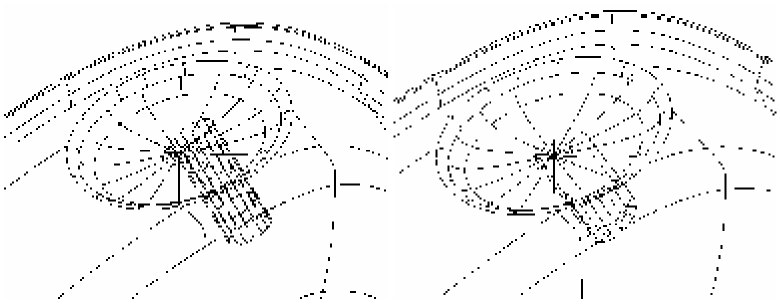
After executing the command you are first asked to confirm that the surfaces are correctly oriented. Next a parameter box appears requesting the ***radius*** of the blend and the ***direction*** of the blend. Choosing whether the ball will be rolled on the side of the surface pointed to by the arrows (normal side) or the opposite side (opposite side) indicates the direction. This needs to be indicated for both surfaces giving four possible combinations (normal-normal, normal-opposite, opposite-normal and opposite-opposite) producing four possible blends as represented below.



Blends will be calculated with each of the surfaces on the screen in turn.

**Note:** This command should generally not be used with very large blend radii values. In particular, if the surfaces already contain blend surfaces, the radius for the multiple blend should not be smaller than the smallest radius of the existing blends.

An example of the use of this command is when designing a washbasin. The washbasin below has been designed with a single surface but then a soap recess has been added with a second surface. The soap recess has been blended with a rolling ball blend to the basin. This makes a total of three surfaces. A fourth cylindrical surface has been created in order to create a run off for excess water from the soap tray. This needs to be blended with all of the other surfaces. The result of this command is shown on the right. All of the blends have been created but as you can see some of them extend beyond there correct endpoints. The related command `TRIM⇒ Blend: Blend With Trim` can be used to complete the design.



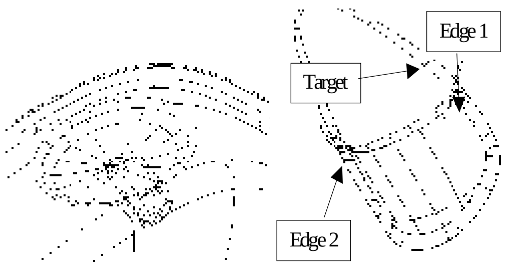
## `TRIMÞ Blend: Trim With Blend`

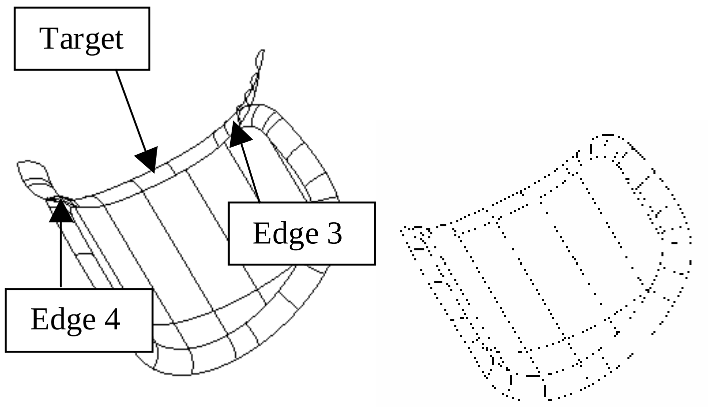
This command is used to ***trim*** the target surface ***with the edge*** of another surface – usually a ***blend***. It's main use is to work in conjunction with the command `TRIM⇒ Blend: Multiple RBlend`. This command ***cleans up blends*** that extend past their correct endpoint.

The blend to be cut should be made target and then the command executed. DESKARTES then waits for you to point (normally with *left mouse button*) to the edge of another blend on the screen. The target blend will then be cut with this edge.

If the target is a ***closed blend***, it should typically meet other blends at ***two different points***. In such a case, use the *right mouse button* to locate two edges of the other blends, and the part between will be trimmed away.

In the example of the `TRIM⇒ Blend: Blend With Trim` command we were left with extended trims as shown below. To better enable you to see what is going on the second picture isolates the relevant surfaces. To complete this model requires us to use this command three times. First make the largest blend the target, execute the command and point to edges 1 and 2 using the *right mouse button* (target being a closed blend!). The result of this is shown in the third picture. Execute the command again and point to edge 3, now with the *left mouse button* (target is an open blend). Repeat the same with edge 4. The completed model is shown in the fourth picture.





## TRIMÞ Change: Invert Cut Part

This command ***inverts*** the results of intersection and blending commands. The surface parts that have been cut away will be brought back and vice versa. The target for this command can be a surface, a trim curve-set or a trim curve.

To invert individual cuts of a ***multiply trimmed surface***, you may use the command on the trim history sets (not the "merged" one). After doing so you must use the command `TRIM⇒ Change: Merge` for the effects of the command to be see on the model.

## TRIMÞ Change: Open/Close Trim

The target trim curve is ***opened or closed***.

Normally trim curves are handled automatically by DESKARTES. Experienced users of DESKARTES may wish to manually interact with these trim curves. Trim curves are stored as closed polygons. This command opens a closed polygon converting it into a polyline (polylines have no effect on the surface), or closes a polyline into a polygon. Importantly, this command correctly handles trim curves which extend over the seam of a closed surface.

## TRIMÞ Change: Merge Trim Sets

This command ***merges*** the trim history of the target surface.

If you intersect or blend a previously trimmed surface, DESKARTES automatically combines or merges the new trim curves with the old ones.

The old and new trim curves are also stored separately, in so-called ***trim history sets*** which appear below the surface with a name beginning with '..Trim/'. Each of these history sets contains the results of just one intersection/blending command. The last part of the name of the history set in the Object Window show which "other" surface produced the trimming. These are combined together into the trim set below the surface called '-> Trim/merge' which actually trims the surface.

There may be occasions, such as when you have been ***manually modifying the trim curves***, when you want to ensure that the trim curves are correctly combined. The command `TRIM⇒ Change: Merge Trim Sets` merges the history sets into one again.

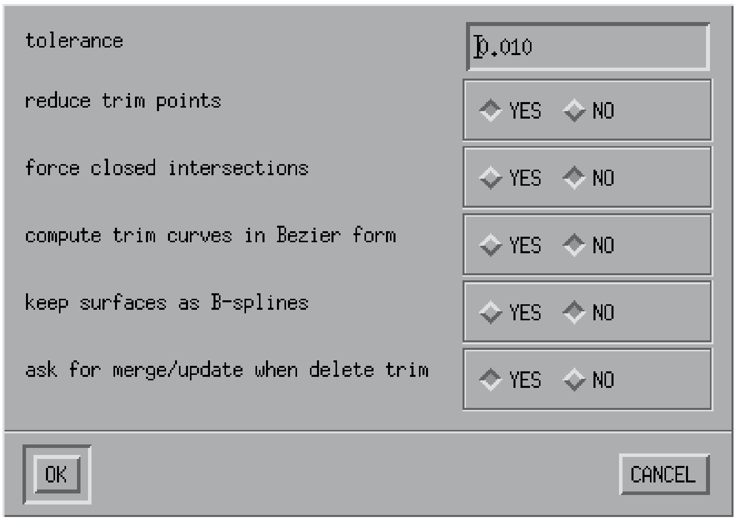
For instance, if you wish to ***undo an earlier intersection command***, select the corresponding history set, delete it, and combine the rest with `TRIM⇒ Change: Merge Trim Sets`.

## TRIMÞ Change: Delete Trims

This command simply ***deletes all of the trim curves and blends*** in the target element. As an option to the command it is possible to delete the ***history trims only***. This is mainly to make the objects may look more compact in the Object Window. As the practical consequence, all the objects will stay trimmed as they were, but the trims can not be edited and merged anymore.

## TRIM Þ Preferences

This command displays a parameter box containing the parameters that control the ***general behavior*** of all the commands in the TRIM menu.



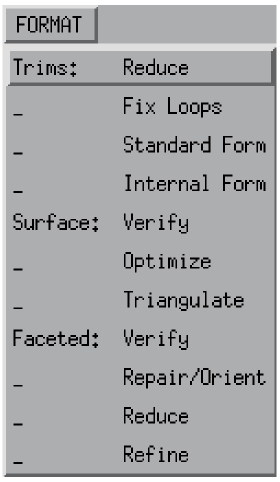
| | |
|---|---|
| tolerance | 0.010 |
| reduce trim points | [✓ YES] NO |
| force closed intersections | YES [✓ NO] |
| compute trim curves in Bezier form | YES [✓ NO] |
| keep surfaces as B-splines | YES [✓ NO] |
| ask for merge/update when delete trim | [✓ YES] NO |

These are:

- ***Tolerance*** **-** This defines the accuracy to which the trim curves will be computed. The smaller the tolerance value, the longer the computations will take, but the more accurate they will be. DESKARTES has built in routines to further modify this value based on the size of the surfaces being trimmed.
- ***Reduce trim points*** – This parameter controls if number of points on the trim curves is automatically reduced in each trimming/blending command, in order to make the model more compact and save in memory space. However, this may result in small gaps between the trimmed surfaces. In most applications such gaps do not matter and using automatic trim reduction is a good idea. However, when transporting the models to Solid Modeling systems, the highest possible trim curve accuracy may be required. In such case, set the “reduce trim points” parameter to NO.
- ***Force closed intersections*** **-** This parameter controls whether the intersection and blending commands should always produce closed trim curves or not. Remember that open trim curves don't actually cut anything away from the surfaces. Use this option to force cutting.
- ***Compute trim curves as Bézier*** **-** This parameter allows you to compute the trim curves as Bézier curves in the parameter plane. They may then be directly used to define the blend boundaries for free-form blends, instead of having to open/close, approximate, etc. Since they are not polygons, they will not cut anything from the surface
- ***Keep surfaces as B-splines*** **-** This option determines whether the surfaces should automatically be converted to Bézier representation during the trim operation, or kept as B-splines.
- ***Ask for merge when delete trim*** **-** When deleting a set of trim curves, this option causes the trim curves of the "other" surface to be automatically deleted, too. Both surfaces trim history sets will be merged too.

## *FORMAT MENU*

These commands are available for different consistency checks, in particular for ***data exchange*** of surface models.



## FORMAT Þ `Trims: Reduce`

This command ***reduces the number of trim curve points*** of the target surface or element, according to a user-given 3D tolerance.

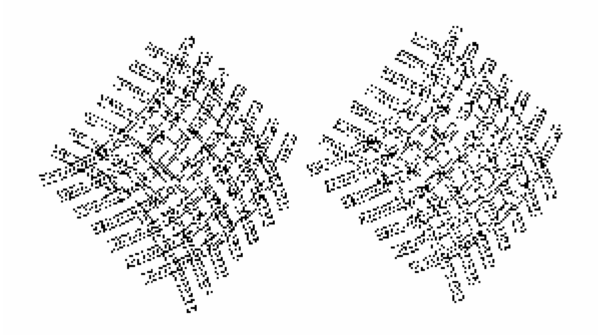
Often when reading surfaces from other systems the trim curves are presented much too accurately to be practical. Reducing the amount of data makes ***displaying and shading*** of the models much faster. It also reduces the number of facets when surfaces are ***triangulated*** into faceted models.

## FORMAT Þ `Trims: Fix Loops`

This command automatically detects and fixes loops in the trim curves. A typical symptom of trim loops is that the surfaces are not shaded correctly, even if their wire frames look right.

Trim loops typically occur when two cylinders of the same radius are intersected with each other. This results in trim curves that intersect themselves ("loop"), such as the number "8".

The following picture shows a more complex example. The surfaces were created with building of multiple projections, with a simgle cross-section shape. The surfaces were then automatically trimmed with each other using command `TRIM⇒ Intersect: Union All`. Finally command `FORMAT⇒ Surface: Fix Looping Trims` was used to get the surfaces shaded correctly.



## FORMATÞ `Trims: Standard Form`

DeskArtes uses its own internal format for representing the trim curves, which is slightly different from the VDAFS and IGES standards.

This command changes the trim curves of a surface, or element of surfaces, from the internal to the ***standard representation***. The required conversion is actually done automatically when writing a data transfer file, but you may sometimes want to check the conversion yourself.

## FORMATÞ `Trims: Internal Form`

DeskArtes uses its own internal format for representing the trim curves, which is slightly different from the VDAFS and IGES standards.

This command converts a trim curve set from standard format into the ***internal format*** used by DeskArtes.

DeskArtes can handle either format in most of its functionality. However, the internal format is required in merging of trim sets if you cut parts away form already intersected surfaces.

The benefit of the internal format against the standard one is that trim curves are kept as separate objects, which can be interacted with individually. The internal representation trim curves obey the following rules:

- With DeskArtes, trim curves are allowed to touch, or even intersect each other, which is illegal with the standard formats.
- If an area of a parameter plane is delimited by an odd number of trim curves, that area is taken to be part of the surface.
- If an area of a parameter plane is delimited by an even number of trim curves, that area of the surface has been cut away.
- When trim curves in DeskArtes "go around" the parameter plane, they are slightly offset from the parameter plane boundaries.

Otherwise, the DeskArtes trim curves behave just as defined in the standards.

Further explanations of trim curves can be found from the IGES or VDA-FS standard descriptions, or in parametric surface modeling textbooks.

## FORMATÞ `Surface: Verify`

This command ***verifies a surface model for faceting***. As result the command produces the curves showing the gaps in the model.

As parameter, the command asks for the "***size of gaps to be filled***". This parameter has the same meaning as in command `FORMAT⇒ Surface: Triangulate`.

Note that even if the model contains no gaps, it may still contain other kinds of errors. The most typical class of other errors are intersections within the model, i.e., parts which the user forgot to trim. Such errors are noticed when the actual faceting is done. However using this command is much faster than producing the actual facets.

## FORMATÞ `Surface: Optimize`

The `Optimize` command is good to perform each time before a surface model is transported to other CAD/CAM/CAE systems using IGES or VDA-FS. It asks for two parameters:

- *discard multiple/empty parts*. It essentially means different clean-up checks and optimizations to the model, such as removing duplicate and degenerated surfaces. Using this option never does any harm, it can only improve the model.

- *split cylindrical parts*. This automatically splits all cylindrical surfaces in two halves, which is how some Solid Modeling systems like to receive the models.

## FORMAT Þ Surface*:* Triangulate

This command is similar to the command `SURFACE⇒ Change`: `Faceted Model`/`check gaps`, except that it offers some additional parameters for advanced faceting control.

- ***Triangulation accuracy***: equal to "tolerance" in command `SURFACE⇒ Change`: `Faceted Model`.
- ***Trim curve accuracy***: setting a small value to this parameter may help the system to stitch the gap curves together. (In command `SURFACE⇒ Change`: `Faceted Model`, its value is set always to one fourth of the triangulation tolerance.)
- ***Size of gaps to fill***: controls how big gaps between the surfaces the system tries to repair, like in command `SURFACE⇒ Change`: `Faceted Model`.
- ***Max triangle edge length***: equal to "max triangle size" in command `SURFACE⇒ Change`: `Faceted Model`.
- ***Orient normals***: with this parameter the user can choose to separately process the faceted model with command `FORMAT⇒ Faceted`: `Repair`/`Orient`. The reason to do this would be to use the "move vertices" option for repairing the gaps.

## FORMAT Þ Faceted*:* Verify

This command ***displays all kinds of errors within a faceted model***.

As options, the user can choose to check only for ***gaps*** between the triangles, or to check also for ***intersections*** between the triangles.

The curves showing the problem areas are stored in new elements after the model. The element named "model.gaps" contains the gap boundaries. The element named "model.misc" contains all other kinds of errors, such as intersecting triangles. If one or the other of these elements is not produced, it means that such errors were not found.

Gaps within the model can be typically repaired with command `FORMAT⇒ Faceted`: `Repair`/`Orient`. Intersecting triangles, on the other hand, are typically caused if the user forgot to trim some parts in the model, and they are best dealt with by working further with the surface model.

## FORMAT Þ Faceted*:* Repair / Orient

This command ***orients the normals*** of a faceted model, and also tries to ***repair gaps and holes*** within the model.

The program asks how big gaps should be filled with the command. Also, there are two ***gap fill methods*** provided:

- ***move vertices***: this method does not create any new triangles, but only moves existing triangle vertices to match with each other. It should be used with relatively small gap sizes only.
- ***fill between***: this method can be used to fill in large holes or even missing parts in the model. A possible way to attempt repairing all problems at once is to use this option with a very large gap size value.

A recommended way to use the command is to first stitch gaps with a small gap size value, and in second pass fill between all remaining holes using a large gap size.

Besides producing the repaired faceted model, the command also displays all remaining gaps in the model in a “model.gaps” element.

## FORMAT Þ Faceted*:* Reduce

This command ***reduces the number of triangles*** within a faceted model. Having less triangles in the model can be useful for saving disk space and computation time.

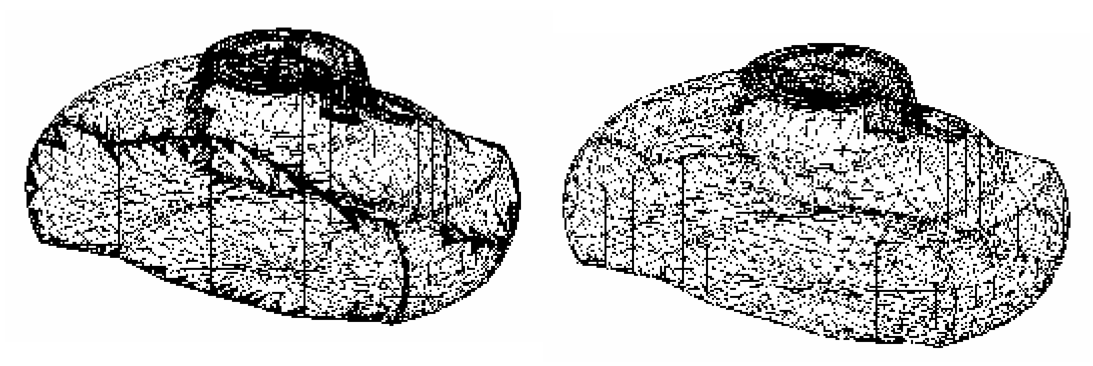
The command first asks if the ***reduction method*** should be for *all triangles*, or for *thin triangles* only. With the thin triangles option the program will not try to remove any other kind of triangles at all.

After the reduction method is selected, the program asks for the following parameters:

***tolerance***: the resulting triangulation is not allowed to deviate more than the given value from the original one.

***maximum angle***: no triangles are removed at those areas where the model has sharp edges, with angle below the given value. This parameter is asked with the *all triangles* reduction method only.

With the thin triangles option, triangles may be removed also at areas of sharp corners. It is therefore recommended to use the thin triangles option with relatively small tolerance values only.



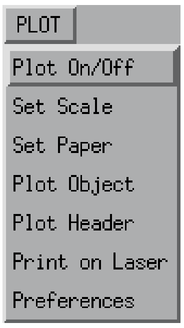
## FORMAT Þ Faceted*:* Refine

This command automatically ***refines the triangulation***, in order to have the model defined with more evenly sized and shaped triangles. Such behavior is often required by various FEM analysis packages.

## *PLOT MENU*

The plotter commands draw the objects in the desired scale for printing on a ***laser printer*** or a ***pen plotter*** that accepts HPGL format plotter files.

The objects are plotted to a file rather than to graphics screen. The file is then sent to the plotter by using the appropriate operating system commands.



### PLOTÞ Plot On/Off

This command ***initiates the plotting process***. The name of a file is requested after which an extra message, `PLOT,` will be shown in the message lines of DESKARTES.

After the plot file has been opened the command `PLOT⇒ Plot Object` can be used to output the drawing into this plotter file.

***Repeating*** the command `PLOT⇒ Plot On/Off` ***ends*** access to the plot file. The plot file can then be sent to the plotting device using the operating system commands, or the command `PLOT⇒ Print On Laser.`

### PLOTÞ Set Scale

This command sets a ***numerical scale*** for the plotter file. A scale of 1: 1, the default, draws the objects at their true size.

Use this command after `PLOT⇒ Plot On/Off` if you wish to define an exact scale for the drawing. After entering the scale numerically the system displays a rectangle corresponding to the paper size defined in `PLOT⇒ Plot Preferences`. You may then position this rectangle as you wish to enclose the objects to be drawn. When the position is set the left mouse button should be clicked to confirm it is in the correct position. If the outline of the paper is too large to fit on the screen, it is automatically placed around the center point of the screen.

### PLOTÞ Set Paper

This is an alternative to `PLOT⇒ Set Scale`, allowing you to ***set the drawing scale graphically***. Draw the paper as a rectangle by clicking the left mouse button with the cursor in the bottom left corner then click again with the cursor in the top right. When the position is set the left mouse button should be clicked to confirm it is in the correct position. The scale you have chosen will be reported in the message lines.

## PLOTÞ Plot Object

This command ***draws the target object into the plotter file***. Execute the command on all the objects that you want to be plotted.

This command should be executed after having initialized the plotter file with the command `PLOT⇒ Plot On/Off`, and setting the scale with `PLOT⇒ Set Scale` or `PLOT⇒ Set Paper.`

## PLOTÞ Plot Header

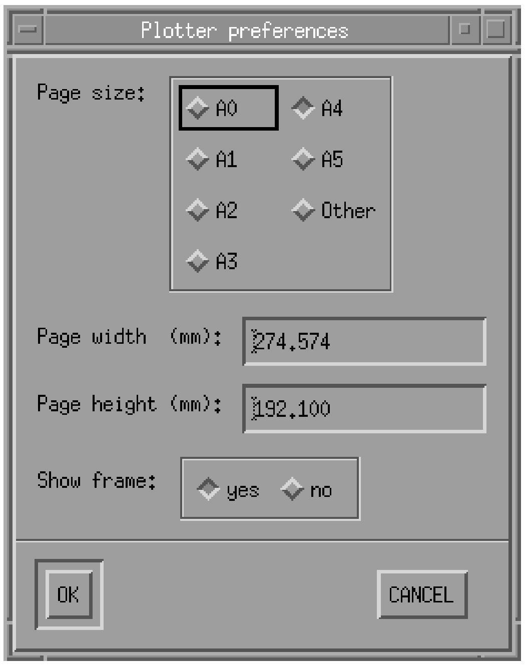
This command should be executed immediately after having initialized the plotter file with the command `PLOT⇒ Plot On/Off`. It writes a ***header line to the plot file***. The header will include the customer company name, the date, scale of drawing, and other information.

## PLOTÞ Print On Laser

This command prints a previously created plotter file on a PostScript ***laser printer***, if one is available. The name of the file to be printed is requested in a dialog box.

## PLOTÞ Preferences

This command allows for the definition of different ***paper sizes and plotter options***.


Plotter preferences

Page size: A0 A4 A1 A5 A2 Other A3

Page width (mm): 274.574

Page height (mm): 192.100

Show frame: yes no

OK CANCEL

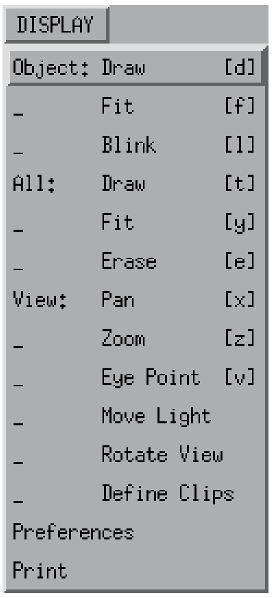
It asks for the following parameters:

- ***Page size***: options being from A1 to A5, or "Other". The last option allows for an arbitrary definition of the paper size.
- ***Page width and height***: the values display the chosen paper size in millimeters. If you have selected "Other" as the previous parameter, the values can be edited to define the paper size.
- ***Show frame***: enables or disables the drawing of the plotting area frame around the plotting area.

## *DISPLAY MENU*

This menu contains commands to ***display*** objects, and adjust the ***viewing window and direction***. The display commands are very frequently used, so their keyboard shortcuts should be learned as early as possible when starting using DESKARTES.

Any of the display commands may be interrupted by pressing `ESC` or `Ctrl-C` in the graphics window.



### `DISPLAY Þ Object: Draw`

This command ***draws the target object*** in the viewing window. The current contents of the display remain and the view scale and pan are not altered so it is possible that some, or the entire object, will not appear on the screen. Use `DISPLAY⇒ Object: Fit` or `DISPLAY⇒ All: Fit` to ensure this does not happen.

### `DISPLAY Þ Object: Fit`

This command ***fits the target object*** into the viewing window. Other objects that were shown on the screen are erased.

### `DISPLAY Þ Object: Blink`

This command ***blinks the target object*** by quickly erasing it and then redrawing it. This is a handy way of checking which is the target object.

### `DISPLAY Þ All: Draw`

This command ***displays all active objects*** that are of the same type as the target object. For example, if the target object is a surface or element, all other surfaces in active elements are displayed. Alternatively, if the target object is an X projection curve-set, all the other X projections are displayed.

To display all surfaces together with projection curve (e.g.) perform the following steps:

1. Select the X-Projection curves;

2. Set the eye point to X direction in the Settings window;

3. Issue ***twice*** the command `DISPLAY⇒ All Draw [t]`.

## DISPLAY Þ All: Fit

This command ***fits all active object***s that are of the same type as the target object into the viewing window. For example, if the target object is a surface or element, all other surfaces in active elements are fitted in the screen. Alternatively, if the target object is an X projection curve-set, all the other X projections are fitted in the screen.

## DISPLAY Þ All: Erase

This command ***clears the whole display area*** leaving a blank ‘canvas’ of the background color.

## DISPLAY Þ View: Pan

This command ***moves the viewing window***. Pan the view by clicking with the left mouse button at any point on the screen, then move the cursor to the new location of the first point, and release the button. The size of the object on the screen will not be changed, but only moved to a new location.

If you press the middle mouse button, the view will change continuously as you move the mouse around. However, in four views and the edit modes, zooming always works in box mode as above.

## DISPLAY Þ View: Zoom

This command alters the ***scale of the viewing window***. The scale of the window can be set either graphically, or relative to the current scale.

- To zoom in freely by a view box, press and hold the ***left*** mouse button with the cursor at the center of where you want to zoom. Move the mouse up or down and a rectangle showing the new window is displayed. The window is fixed by releasing the mouse button.
- The view will change continuously if you press the ***middle*** mouse button whilst moving the mouse. However, in four views and the edit modes, zooming always works in box mode as above.
- By clicking the ***right*** button, the window becomes twice as big as it currently is — the objects on the screen are shown smaller.
- By clicking on any ***key*** on the keyboard, you will be able to pan instead of zooming. Pan the view by clicking with the left mouse button at any point on the screen, then move the cursor to the new location of the first point, and release the button. This feature is particularly useful when zooming in the various ***edit modes***, where only the zoom function is available.

Note that although the object may change size on the screen the actual size at which the model is defined does not change. The size of the object on the screen bears no relation to the physical size.

## DISPLAY Þ View: Eye Point

This command defines the ***viewing direction***, the so-called ***eye point,*** for the different viewing commands.

When the command `View Eye Point` is executed, horizontal movements of the mouse rotate the view sideways and vertical movements of the mouse rotate the up and down. The object rotates continuously as you move the mouse. Click again on the mouse button to use this as the current viewing direction.

The point about which rotation takes place depends on which mouse button you click.

- The ***left*** button selects the closest point on the model as the rotation center
- The ***middle*** button selects the rotation center as the middle of the target object.
- The ***right*** button continues the rotation around the last defined center point.

## DISPLAYÞ View: Move Light

When in shaded mode as set with `Settings`⇒ `MODE`: `Shaded` the object is shaded as though lit with a single light point. The command `DISPLAY`⇒ `View`: `Move Light` allows you to continuously ***move the light point*** around the object and the shading will update.

After clicking the left mouse button, horizontal movements of the mouse rotate the light sideways and vertical movements of the mouse rotate the light up and down around the center point of the object. The light point is displayed as a 3D crosshair during the operation.

## DISPLAYÞ View: Rotate View

The command `DISPLAY`⇒ `View`: `Rotate View` allows you to ***continuously rotate*** the eye point around a chosen axis.

The ***rotation axis*** is chosen by clicking one of the mouse buttons: left for X-axis, middle for Y-axis and right for Z-axis.

The rotation continues until a mouse button is clicked again. The r***otation speed*** can be made slower by moving the cursor UP on the screen, and faster by moving the cursor DOWN.

## DISPLAYÞ View: Define Clips

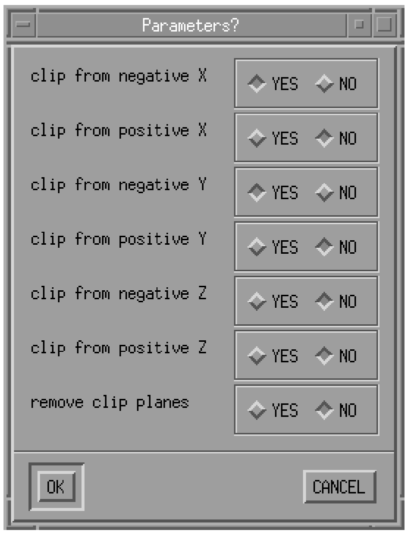
The command `DISPLAY`⇒ `View`: `Define Clips` allows you to define one or several ***clipping planes***. These restrict the display of 3D objects so that only parts on one side of the clipping plane are displayed. This can be used to see sectioned views through your model.

After executing the command, you choose the clipping plane direction by clicking the left mouse button for X-axis, middle for Y-axis and right for Z-axis direction. To move the clipping plane, move the cursor UP and DOWN.

You may define up to six clipping planes at a time, two for each axis direction. The clipping planes will stay in their place even if you change the eye point.

Pressing any KEY on the keyboard after launching the command displays the following dialog box which allows some or all clipping planes to be turned off or on.

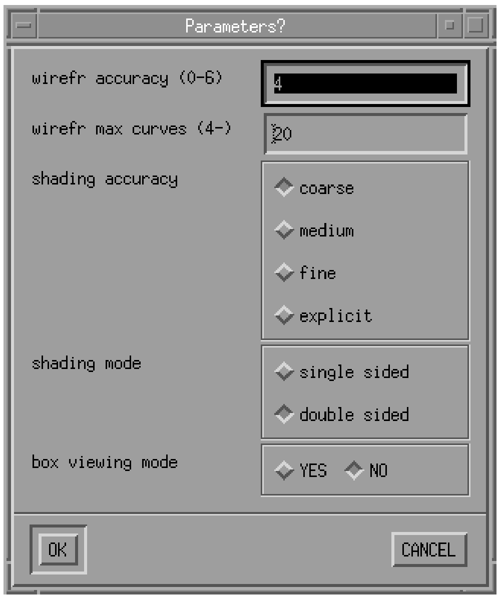
Also, the button `Settings`⇒ `CLIP` can be used to turn on and off the display of all clip planes at once.


Parameters?

clip from negative X [✓ YES] NO
clip from positive X YES [✓ NO]
clip from negative Y [✓ YES] NO
clip from positive Y YES [✓ NO]
clip from negative Z YES [✓ NO]
clip from positive Z YES [✓ NO]
remove clip planes YES [✓ NO]

## DISPLAYÞ Preferences

The command `DISPLAY⇒ Preferences` display a dialog box.


Parameters?

wirefr accuracy (0-6): 4
wirefr max curves (4-): 20
shading accuracy: [✓ coarse] medium fine explicit
shading mode: single sided [✓ double sided]
box viewing mode: YES [✓ NO]

This controls various modes of display and shading as follows:

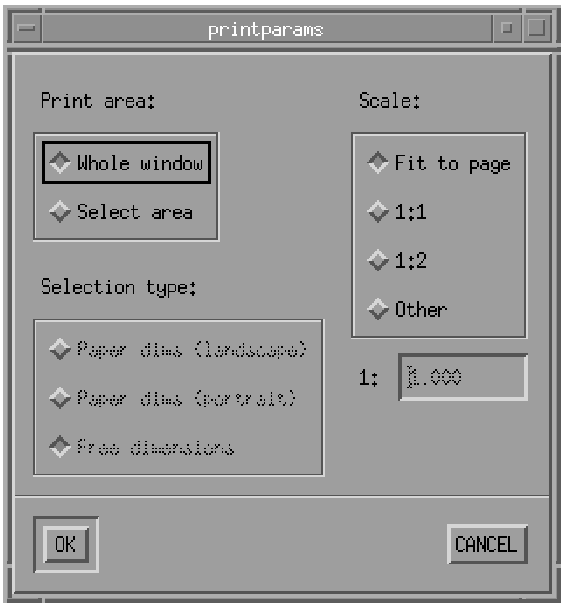
- ***Wirefr accuracy*** - determines the accuracy of drawing curves. It also affects the accuracy of display of wire frame surface models. Each increase in the parameter's value doubles the number of points used. A low number will make smooth curve appear to be made from straight lines.
- ***Wirefr Max curves*** – determines the density of curves to be drawn on a surface in both directions. Using a small value makes displaying faster, and saves system memory.
- ***Shading accuracy*** - specifies how accurately surface models are faceted for shading in the graphics window. The choices are coarse, medium and fine and explicit. Explicit requests a tolerance value that defines the accuracy.
- ***Shading mode*** - specifies whether to shade models as two-sided regardless of their normal directions, or single-sided in which parts of models not facing the viewer are displayed in blue. This feature can be very useful when using commands such as blending, which require the user to know which way the surfaces are oriented.

- ***Box viewing mode*** – this option is useful for working with very complex models or slow machines. When this option is set on, only the bounding box of the model is displayed when using the command `DISPLAY`⇒ `View`: `Eye Point`

## DISPLAY Þ Print

The command `DISPLAY`⇒ `Print` prints the current contents of the display on to the system printer. This command is only available with the Windows version.

The command first displays a parameter window as shown below:

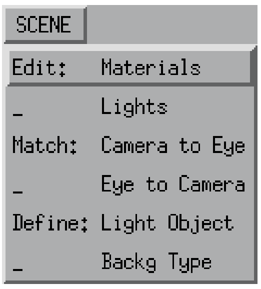


There are options to set the scale and to choose to plot the whole graphics area or just a small portion of the display.

After clicking $\boxed{\text{OK}}$ you will be asked to confirm you are ready to print.

## *SCENE MENU*

The commands in the SCENE menu control the way surfaces are ***visualized***. The scene definitions apply to both camera view rendering and ExTrace visualization.



The visualization components include the ***camera*** and ***light points***, which may be interactively defined and edited with a comprehensive scene editor (command `SCENE`⇒ `Edit`: `Lights`). Other scene variables are the object ***colors*** and ***materials***, whose interaction is achieved through the command `SCENE`⇒ `Edit`: `Materials`.

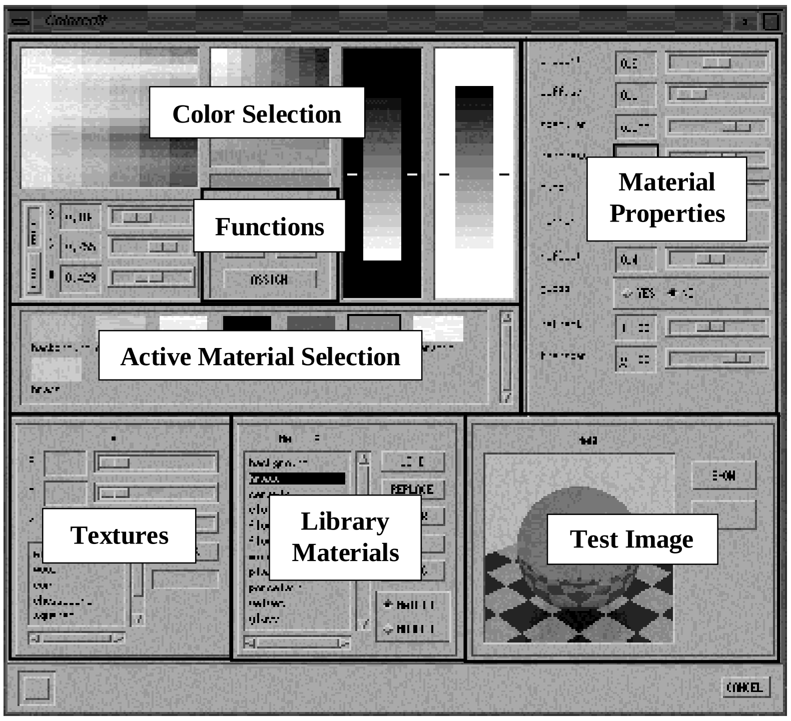
Most scene parameters, such as the camera, lights and background type, are stored in the "SCENE" element. Materials are stored separately in the "MAT" element. Thus, the scene definitions may be saved into the model files, for use in different modeling and visualization sessions.

Any changes to the scene components are immediately transported to camera view shading and ExTrace, and vice versa if any graphical operations are executed in the rendering and ExTrace windows.

Camera view rendering does not take into account all the scene definitions. Reflective objects will only reflect the background, but not other objects. Transparency is not displayed by shaded rendering, and only certain types of textures are shown. To display all these features, use ExTrace for visualization.

## `SCENE`Þ `Edit`*:* `Materials`

This command is used to define and assign different material properties to surface objects. After executing the command, an extensive ***material palette window*** appears which includes various ***fields*** to perform different operations. Any number of materials may be simulated from this material palette and predefined materials may be loaded from a ***material library***. The defined materials can then be assigned to different surfaces or elements.



## *Active material selection*

The colors and names of all the ***currently defined materials*** are shown in a separate field below the color fields. The currently selected ***active material*** is highlighted by a black box surrounding the color.

You may ***select a material***, making it active, by clicking on it with the ***left*** mouse button. The rest of the parameters displayed in the box will then change to reflect the properties of this material. Any changes you make to the colors and other properties only affect the currently chosen active material.

If you wish to ***select a color*** of a material, but not make the material active, click at the material box with the ***middle*** mouse button.

## *Material functions*

Above the active material selection, in the center of the window, are five general functions to manage with the materials. These are:

- ***COPY -*** Copies the active material to a new material allowing you to enter another name for the new material.
- ***DELETE -*** Deletes the active material. If the material name has been assigned to some objects in the model, the system asks you to enter the name of another material to be used instead.
- ***RENAME -*** Gives the active material another name. Any objects to which this material has been assigned will be updated to reflect this change.
- ***UNDO -*** Discards all the changes you have made to the active material since you last selected it.
- ***ASSIGN -*** Assigns the active material to the currently selected object (surface or an element of surfaces). You can use the Object Window for selecting the target object at any time whilst editing the materials.

## *Color selection*

The fields at the top left are available for ***color selection***. The color may be defined graphically, or numerically using either the HSV (Hue–Saturation–Value) or RGB (Red-Green-Blue) models.

The color fields are operated with the mouse as follows:

- The ***basic color field*** is located at the top left. Each row of colors represents one shade of color. Each column represents color intensity. The required color is selected with the left mouse button.

  If you need a more exact shade of a particular color, you may ***zoom*** into the selected color field by pressing the ***middle*** mouse button. This displays more variations of color, which vary slightly around the selected color. Clicking the middle mouse button again returns you to the basic colors.

- The ***color-mixing field*** is located next to the basic color field. This operates much like an artist palette where up to four standard colors can be chosen and mixed together. Each of its ***corners*** may be assigned a color. To do this first select a color from the basic color field. Then position the cursor over a corner square in the mixing field and click the right mouse button. The shades inside the mixing field change correspondingly.

  If each corner has a different color, four colors are mixed. If two neighboring corners have the same color, three colors are mixed, etc.

  The shades in the color-mixing field may also be ***selected*** with the ***left*** mouse button, and ***zoomed*** with the ***middle*** button.

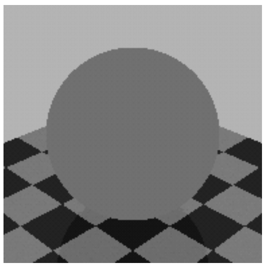
- To the right, ***shades*** of the selected color are shown against two backgrounds. By default, these are black and white, but you may change them by clicking the right mouse button in the background area which will assign the currently selected color to that background.
- Below the basic color and mixing fields, the selected color is displayed numerically as either ***RGB or HSV values***. You may change these values using the sliders next to the values, or by typing in new values. An explanation of these values can be found in most color books.

## *Material properties*

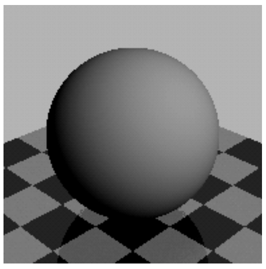
On the extreme right of the window, a series of parameters allow you to adjust the ***properties*** of the material. They can be varied by slider or by entering a numeric value. Each explanation below includes a picture of a test ball rendered with the lighting parameter inquestion.

- ***Ambient*** **-** This defines the general lightness of the surface. Its value ranges from 0.0–1.0. There is no source for the light which means it acts like ***"day light"*** coming from all directions.

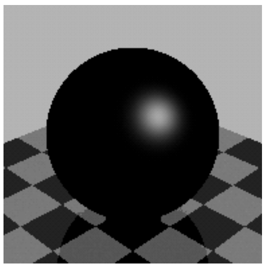
  Note that all the other material parameters, such as the diffuse specular and reflection components, are added to the ambient value. With mirroring objects, in particular, the ambient component should be made quite small to avoid objects appearing too bright.



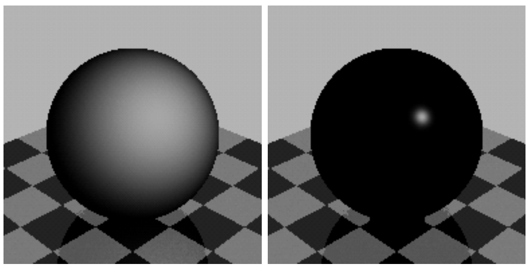
- ***Diffuse*** **-** This defines the general reflectivity of the surface. Its value ranges from 0.0–1.0. The greater the value, the greater the effect of the light points on the surface. The diffuse factor emphasizes the curvature of surfaces.



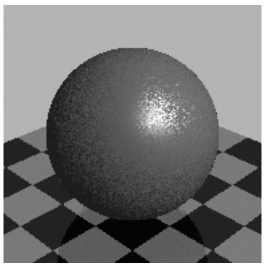
- ***Specular*** **-** This defines the ***polish*** of the surface. Its value ranges from 0.0–1.0. This parameter works in conjunction with the hardness parameter. It controls the appearance of ***highlights*** on the parameter component. The greater the specular value, the brighter the highlights.



- ***Hardness*** - This defines the hardness of the surface. Its value ranges from 0.0-1.0. It controls the appearance of ***highlights*** on the parameter. With small values, the material looks soft and the highlights on the surface appear on large areas. With large values, the highlights are small and sharp. This parameter therefor controls the size of the highlights. For glass and other hard materials, use a large hardness value.



- ***Bump*** - This gives the surface a "bumpy" appearance. The range of the bump values is 0.0 to 1.0. This parameter cannot be used to define large bumps on the surface. It can only produce the appearance of a slight texture on the surface, similar to that found on many plastic injection molded objects.

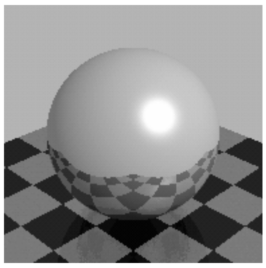


- ***Mirror*** - This is a switch, which makes the material mirroring, or not. This parameter works in conjunction with the reflect parameter.

- ***Reflect*** - The `reflect` parameter only has an effect if the `mirror` parameter is set to `YES`. It defines how much a mirroring material reflects the environment around it. For example, with the value 0.3, 30 per cent of the environment would be reflected, and 70 per cent would be absorbed.

  The amount of reflected light is added to other material properties. Consequently, you should not use too much `ambient` or `diffuse` light on mirroring materials.

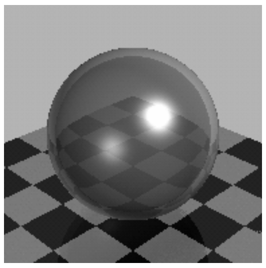
  With glass, in particular, the amount of reflected light should be proportional to the transparency. As a rule of thumb, a value of `reflect` equal to 2*(1-`transpar`) gives good results.



- ***Glass*** - This is a switch, which makes the material transparent, or not. This parameter works in conjunction with the `refract` and `transpar` parameters.

- ***Refract*** - The `refract` parameter only has an effect if the `glass` parameter is set to `YES`. The `refract` parameter determines, how strongly the material bends (refracts) light which passes through it. The practical range is 0.5 to 2.5. Glass has values from 1.4 to 1.6. Values of less than 1.0 bend light in the opposite direction to most materials. Other factors can be found in physics tables.

- ***Transpar*** - The `transpar` parameter only has an effect if the `glass` parameter is set to `YES`. It determines the amount of light that passes through the material. The transparency range is 0.0—1.0. For example, a value of 0.9 means that the surface will be 90% transparent and 10% of its own color will show.



## *Library materials*

The library material field, at the bottom of the window, allows you to load in materials from the ***material library*** or from your current model directory. The choice of where to read from is made with the two buttons `MATLIB` and `MODELDIR` at the bottom of the area.

The contents of the directories can be seen in a scrollable ***name list***. You may select any library material by pointing and clicking with the left mouse button. The following functions to operate on the selected library material:

- ***STORE*** - Stores the active material into the material library. It will be saved under the current name of the material.
- ***LOAD*** - Loads the selected library material as a new active material, with the library material name. This material can then be modified if required before assigning to an object.
- ***REPLACE*** - Replaces the active material with the selected library material, keeping the active material's name.
- ***FLOOR*** - Loads the selected library material, replacing the special material named `floor`. This material is used for the floor in ExTrace images.
- ***FLOOR2*** - Loads the selected library material, replacing the special material named `floor2`. This material is used as the second material for textured floors in ExTrace images.
- ***BACKG*** - Loads the selected library material, replacing the special material named `backg`. This material is used as the background color for camera view shading and ExTrace images.

## *Textures*

A list of various ***solid textures*** can be found in the `TEXTURES` field, at the bottom left of the window. You may use them to preview there appearance on the test object, however, they are not assigned to the materials. You may later assign such a texture to the objects with the `TEXTURE`⇒ `Area`: `New` command.

The available textures are shown in a scrollable ***name list***. Pointing and clicking with the left button on a name chooses the textures.

Some textures need ***two materials***. The name of the second material is shown below the `VEINS` button. You can choose any material by clicking on the VEINS button then selecting by pointing and clicking on the material table. With wood and marble textures the name may be left empty and a default black material is used for the veins.

The X, Y and Z scales adjust the ***density*** of the texture in the directions of the three axes. The bigger the value the denser the texture.

Texturing can be ***removed*** by selecting the current texture name again.

### *Test image*

A spherical ***test object*** can be ray traced with the chosen material properties by pressing the SHOW button in the TESTIMAGE field in the bottom right corner of the window. Pressing the STOP button in the same field can interrupt the rendering.

For transparent or mirroring materials, a chessboard floor automatically appears for these effects to be more clearly seen. The chessboard materials are `floor` and `floor2`, likewise, the background is rendered using the `background` material.

### *Transporting to ExTrace*

Clicking the button EXTRACE shows the changed materials in the EXTRACE window. This means that you need not exit the Material Palette to see the changes. You can change the materials and immediately check the effect in the ray traced image.

In order to use this button, the ExTrace program must have been started before entering the Material Palette, and the ExTrace picture must not be currently computing. You may always press the STOP button in the ExTrace window to stop its computation.

Thus, the interaction with the materials and ExTrace is typically:

1. Start ExTrace.
2. Change a material.
3. Click STOP in the ExTrace window.
4. Click EXTRACE in the Material Palette window.
5. Return to step 2 as often as required.

### *Exiting*

Material editing is exited with the OK or CANCEL buttons at the bottom of the material palette window. The OK button stores all the material changes in the model, and CANCEL rejects them. Materials are actually stored in a special element in the object window called MAT.

**Note**: the CANCEL button is disabled if you have assigned any material to the objects. To cancel such changes exit and use `Settings`⇒ `UNDO`.

## SCENE Þ Edit: Lights

This command controls the general ***viewing and lighting parameters*** for the shaded and ExTrace images. These include such items as the camera and light points, spotlights, light colors and intensities. The camera defines the view direction and angle for rendering shaded images and in ExTrace.

The command enters an ***edit mode*** where the positions of all the lights and the camera can be edited graphically. After executing the command you are asked whether the current eye point view should be adopted as the starting location for the camera or whether the camera should remain in it's previously defined location.

The changes to lights and camera are stored in a special element in the object window called `SCENE` after exiting the editor.

The available scene editing functions are explained separately in the Editing Functions part of this Manual.

## SCENEÞ Match: Camera To Eye

The ***camera*** defines the view direction and angle for rendering shaded images and ExTrace. This command positions the camera to ***match the current viewing direction***, the so-called eye point. The view angle is requested as an additional parameter. This will ensure any rendered views or ExTrace pictures will look the same as the current screen.

This is often *the most convenient way to define the camera position*.

## SCENEÞ Match: Eye To Camera

The eye defines the view direction and angle for displaying in the normal graphics window. This command positions the eye to match the current camera position. This will ensure the current screen will look the same as any rendered views or ExTrace pictures.

## SCENEÞ Define: Light Object

Normally lights are not visible with DESKARTES. This command makes a surface appear as a ***visible light source*** in ExTrace. Using this technique you can simulate lamps or bulbs which are actually seen in the picture.

When executed this command first asks if the object should be made a light object, or (if it is already defined as such) if this property should be removed.

The command then offers three different possibilities to associate the light object with a light point:

- move object to light
- move light to object
- create new light at object

If you wish to define several light points inside a light object, you must use scene editing to create and then move the light points inside the object. The light object has the total intensity and the combined color of all the light points contained within the object's bounding box. Note, however, that if a light object is later moved, the light points will not automatically follow.

## SCENEÞ Define: Backg Type

Determines the background type for shaded and ExTrace images. There are three possibilities:

- ***Single color*** - The scene will have a plain background color specified with the command `SCENE⇒ Edit: Materials`. This is the default.
- ***Sliding color*** - This option makes the background color more interesting by varying the shade from dark to light.

  Multiple colors can be placed in the background with the command `EXTRACE⇒ Locate: Backg Colors`. This feature is not available for shaded images.
- ***Texture*** - A picture is drawn as the background. The user is asked for the name of the file. Shadows will not fall on the background picture, nor will light points affect it.

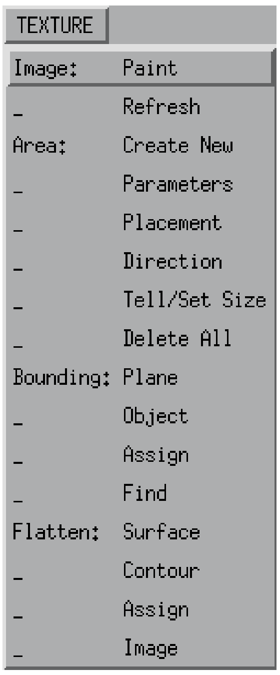
  If the scene and background pictures are of different proportions then a part of the background picture is automatically clipped away.

Another parameter available in this dialog box is

- ***Environment map.*** This is extremely useful for highly reflective images such as gold and silver. If environment is turned on you will be asked to choose and image file from a scrollable list. The texture image is wrapped all around the scene. The image is not shown in the background but mirroring materials reflect it.

## TEXTURE MEN U

The texture menu contains commands for manipulating ***textures***. Textures are graphical ***images*** that can be applied to a surface in much the same way as a decal can be stuck onto a surface. This menu also contains option for ***flattening*** surface areas so that the correct size and shape of decals can be calculated.



## *General Information For Texture Mapping*

### *Texture areas*

A ***texture*** is a color or relief pattern that is wrapped onto a surface and then visualized.

Textures are defined in rectangular parts of a surface, known as ***texture areas***. They are given ***parameters*** like the name of texture, material, etc. Texture areas may be defined directly by ***pointing*** at the surface. A texture area may cover the entire surface or a smaller part of it, and there may be several texture areas defined on one surface.

In addition to ***size and location***, texture areas include the ***direction*** or ***orientation*** of the texture. The direction is displayed by an arrow in the area box, pointing from the lower left corner to the lower right corner of the texture image.

### *Parameter plane*

In fact, texture areas are defined as rectangular areas in the ***parameter plane***, similarly to trim curves. They may be manipulated in the same manner, though in practice the user never needs this facility except for possibly deleting texture areas.

Among the texture areas is shown the ***texture frame***, representing the boundaries of the parameter plane. A texture can lie partly ***outside*** of the texture frame, extending either to the left or below it. This happens if the texture goes over the ***seam*** of a closed surface.

### *Bounding surfaces*

Instead of texturing surfaces directly, it is also possible to create a ***bounding surface*** from which the texture will be ***projected*** onto the actual ones. This way, it is possible to create a texture that goes seamlessly over several surfaces.

### *Displaying and visualizing*

The command `RENDER⇒ GLWindow: Shaded View` are intended for basic form checking, and they never show textures.

The commands `RENDER⇒ GLWindow: Camera View` are able to show material and some color image textures, but not bump maps or bounding surface texturing. Showing the gray scale image just as if it were an ordinary color image texture simulates bump maps.

Textures are shown with `RENDER⇒ GLWindow: Camera View` only if the corresponding parameter with `RENDER⇒ Preferences` is switched on.

All the possible texture definitions can be visualized with ExTrace.

If the texture parameters are changed while ExTrace is running on screen, you must insert the corresponding surfaces to ExTrace again. The system will automatically prompt you for doing this.

## *Flattening procedure*

Surface ***flattening*** can be used to obtain the shape of a decal required to fit onto a surface. It can also be used to render an image with the decal applied to the surface with minimal distortion. The normal texture areas commands, which are much simpler to use, can be used to produce a similar rendered image but minor distortions may be introduced. Only in the most stringent of applications (such as ceramic tableware decoration) will surface flattening be needed to create a render image.

In ***summary***, the basic steps of using surface flattening are as follows:

Cut out a strip of the surface you will texture using `SURFACE⇒ Change: Split in Two`. The general flattening mode requires that the start point of a symmetrical surface is located at a symmetry line of the surface strip. You may use `SURFACE⇒ Change: Revolve` to obtain this. Any trimming operation on the surface strip should be performed only after revolving.

When ready for texturing, flatten the surface strip using and to obtain its 2D contour use `SURFACE⇒ Flatten: Contour`

Think of the contour as the shape of a decal to be applied onto the surface. You may then use TextPaint, or any painting program to decorated this contour. When ready, apply the texture image onto the unflattened surface strip using the command `SURFACE⇒ Flatten: Assign`.

Render the surface with ExTrace. The texture automatically appears on the surface.

## `TEXTURE Þ Image: Paint`

This command launches the ***TextPaint*** program, which allows you to ***paint*** and ***edit*** own texture images.

See the Editing Functions part of this Manual for details.

## `TEXTURE Þ Image: Refresh`

Scanned and other images , e.g., sketches, can be read into `TextPaint` program, and thus be later used as ***design templates*** to draw B-spline curves on top of the image.

When exiting `TextPaint`, the user is asked if such a design template should be created. If the answer is affirmative, the image is stored into the system memory. The design template can then be ***hidden and shown again*** by using the command `TEXTURE⇒ Image: Refresh On/Off`.

## TEXTURE Þ Area: Create New

This command defines a ***new texture area*** on the target surface, or element of surfaces. Any number of texture areas may be defined for each surface.

There are two stages to defining a texture. First define the ***properties*** and then define the ***position***. For the parameter explanations, see the commands `TEXTURE⇒ Area: Parameters` and `Area: Placement`. The position is asked at this stage only if the command is given to a single surface, not an element.

To later change the texture area properties or placement, use the commands `TEXTURE⇒ Area: Parameters`, `Placement`, and `Direction`. To remove a texture area, you must select the texture area in the lowest pane of the Object Manager window and issue the command `OBJECT⇒ Delete`.

## TEXTURE Þ Area: Parameters

This command changes the ***parameters*** of a previously defined texture area. Before executing the command, you should select as the target object either the texture area, or the surface. If the selected surface has several textures, DESKARTES asks you which texture area you wish to alter.

The command first asks you what ***texturing method*** you wish to apply, then some ***texturing details*** depending on the style. The style alternatives are:

- ***Material*** - Another material is applied to the texture area, without any graphic pattern. For example, a gold band around the rim of a cup.
- ***Color image*** - A picture is mapped on the texture area. For example, a decal of a company logo on a product.
- ***Bump map*** - A gray-scaled picture can be used to produce a bump map effect on the surface. For example, engraving on jewelry. You can convert any image into a gray-scaled image with the command `FILES⇒ File: Convert`. Note that bump maps are visual tools. They do not really affect the surface so cannot actually be used for manufacture
- ***Solid texture*** - A procedural 3D solid texture is applied on the texture area. For example, wood grain passing through an object. It does not require a texture image to be specified, only the materials and scale coefficients for the texture. Procedural textures are really mathematical formulae that take the X, Y and Z coordinates of a point in space and calculates a color. Different formulae give the appearance of different materials.
- ***Bounding surface*** – Use this option if you wish to create a texture that spreads over several surfaces or if a surface is difficult to texture using texture areas. The texture is first defined on an invisible bounding surface, from which it will be projected onto the actual surfaces.

Depending on which of the above options you have chosen you will be asked different parameters. These are explained now.

### *Material, Image and Bump Mapping*

These texturing choices first show a ***list of texture files***. These textures are found either in the current model directory, or in the texture library `DA_textures`. You may select any one of these files for texturing. For bump mapping, though, you may first need to convert the image to a gray scale (see command `FILES⇒ File: Convert`).

If you do not find the desired texture file in the list, you may have the wrong model directory selected, the image may be named inconsistently, or it may be in a wrong format. The file must be saved with a name ending `...pic` or `...pic.Z or ...rgb` or `...rgb.Z`. The File Window command `FILES⇒ File: Convert` converts most other commonly used file formats to these formats.

Next, the system asks for one or more of the following ***details*** to be specified:

- ***Material name*** - A texture area may be given a different material than the rest of the surface. An example of this would be for placing a plastic sticker on a metal surface. The surface would be defined with a metal appearance but the material entered here would be plastic. If no material is specified, the surfaces own material is used.
- ***Number of replications*** - This allows the texture to be repeated a number of times in the texture area's horizontal and/or vertical directions. For instance, to define wallpaper with a regular flower pattern, draw one flower and replicate the image on the wall a number of times. This saves the trouble of making a large and memory intensive picture with many flowers on it.
- ***Blurring*** - Smoothes the borders between individual pixels in the texture image making it look more natural and less computer generated. This technique is sometimes called anti-aliasing on other computer systems.
- ***Adaptive mapping –*** It is common for textures that have been applied to surface to become distorted due to the underlying definition of the surface. Adaptive texturing minimizes this distortion of the texture. Adaptive mapping should nearly always be used, unless intentional distortion is required for a special effect.
- ***Transparent background*** - Controls the background color of the texture. If transparent background is chosen, the background will be rendered in the same color as the actual surface. Otherwise, the texture will be displayed on the surface with its original background color.
- ***Bump depth*** - This parameter determines how dramatic the bump effect will be when applying bump mapping. Higher values cause deeper bumps.

## *Solid textures*

The following solid textures are available:

- ***Marble*** - This produces a marble-like pattern on the textured surface. Any existing material can be defined for the marble veins. If you don't specify the vein material but just leave the name field empty, a darker shade of the texture area's material is used.

  You can determine the density of the veins with three scale coefficients. They adjust the density of the veins in the x, y and z directions. Larger scale values produce denser veins, and vice versa. You can use the command `SCENE⇒ Edit: Materials` to check for suitable values for these scales.
- ***Wood –*** This command produces an irregular pattern on the surface that resembles wood grain. The color depends partly on the texture material, changing the wood color to different brown variations. No vein material is required.

  You can determine the density of the veins with three scale coefficients. They adjust the density of the veins in the x, y and z directions. Larger scale values produce denser veins, and vice versa. You can use the command `SCENE⇒ Edit: Materials` to check for suitable values for these scales.
- ***Cork*** - This command produces a cork-like pattern on the surface.

  You can determine the density of the veins with three scale coefficients. They adjust the density of the veins in the x, y and z directions. Larger scale values produce denser veins, and vice versa. You can use the command `SCENE⇒ Edit: Materials` to check for suitable values for these scales.

- ***Stripes/squares -*** Generates a regular pattern of stripes or squares on the surface. You are asked for the number of squares.
- ***Bump -*** Produces a bumpy relief pattern on the surface. The scaling factor determines the density of the bumps.

### *Bounding surfaces*

This option should be chosen if you wish to create a texture that runs over several surfaces or if a surface is difficult to texture using texture areas. This way the texture is defined first on a ***bounding surface***, from which it is ***projected*** to the actual surface(s).

Bounding surface texturing is normally created using the special `TEXTURE⇒ Bounding: ..` commands. This command may be used if you later wish to check or change the parameters.

The normal texture area parameters (whether the texture is a color image or a bump map, etc.) are defined on the bounding surface, while the actual surface(s) only hold the information on the projection direction. See commands `TEXTURE⇒ Bounding: Object` and `Assign` for the parameter explanations.

## TEXTURE Þ Area: Placement

This command changes the ***position*** of a previously defined texture area. Before executing the command, you should select as the target object either the texture area, or the surface. If the selected surface has several textures, DESKARTES asks you which texture area you wish to alter.

If the texture area is defined on just one surface (not an element), the system asks for its placement on the surface using one of the following options:

- ***Corners -*** The texture area is positioned by pointing at the lower left and upper right corners of the desired area on the surface and clicking with the left mouse button.
- ***Center point -*** The texture area is positioned by pointing at the center of the desired location and clicking with the left mouse button. Next, show the placement of the texture’s upper right corner with the left mouse button.
- ***Around -*** The texture area is positioned over the entire cross-wise section of the surface. The texture borders in the lengthwise surface direction are shown by pointing at the bottom and top of the texture area on the surface with the left mouse button. This is useful for creating bands of patterns around an object.
- ***Numeric -*** The texture area is positioned numerically. The parameters requested are the minimum and maximum values of the texture area in each direction of the parameter plane.
- ***Whole surface -*** The texture area is positioned over the whole surface. The texture direction is not determined by graphical interaction, so you may need to use the command `TEXTURE⇒ Area Direction` to position it correctly.

## TEXTURE Þ Area: Direction

This command changes the ***orientation*** of a previously defined texture area. Before executing the command, you should select as the target object either the texture area, or the surface. If the selected surface has several textures, DESKARTES asks you which texture area you wish to alter.

The options are:

- ***Mirroring*** can be used to flip the texture up side down and/or left to right.
- ***Rotation*** turns the texture around by either 90, 180 or 270 degrees.
- ***Wrapping*** the texture places the texture the opposite way around a closed surface.

## TEXTURE Þ Area: Tell/Set Size

The texture positioning commands in DESKARTES do not automatically guarantee to maintain the aspect ration of any texture placed. To avoid distorting the texture, the ratio should be about the same as on the actual texture image.

This command reports the 3D ***height/width*** of a texture area on a surface. Furthermore, it allows the user to change the texture's size to desired values. Thus, the user simply keys in the desired new texture height/width values. The texture area on the surface changes accordingly, keeping the center of the texture area at its old position.

The ratio the actual texture image sides may be calculated by dividing the vertical and horizontal pixel resolutions of the picture. These are reported when the `FILES`⇒ `File`: `Info` is executed on the texture file.

## TEXTURE Þ Area: Delete All

This command deletes all textures in an element at once.

## TEXTURE Þ Bounding: Plane

The command `Bounding Plane` is a special, easy-to-use way of applying a ***planar bounding object*** (bounding plane) to the target surface(s).

Before executing the command set the viewing direction onto the model to one of the X, Y or Z ***axis directions***. Also, ensure that only the surfaces that are to receive the texture are visible on the screen. The texture will be projected on to these surfaces in the viewing direction.

If you have recently stored an ***image with TextPaint***, the system asks if that is the image that should be applied as a texture. If you accept the last stored image as a texture, no further interaction is required to complete the command. It is assumed that you have drawn the texture image in TextPaint using the actual model as a template for painting so the image will automatically be correctly positioned.

An alternative method is to ***draw a rectangle*** on the screen. The rectangle is used to position the image for projection onto the model.

The command asks for the texturing ***parameters*** in a similar way to the command `TEXTURE`⇒ `Area`: `Create New`. However, those parameters that are not appropriate with bounding objects are not requested.

Additionally, the command asks for a ***name*** for the bounding object. This is required to assign the correct bounding object to the right surfaces. If you give a name that is already in use, the program complains and asks for a different name.

## TEXTURE Þ Bounding: Object

This command defines a surface as a ***bounding object*** for texturing other objects. It works in conjunction with the command `TEXTURE`⇒ `Bounding`: `Assign` which actually applies the bounding object to the surfaces.

A bounding surface is an 'invisible' surface that will not show in the ExTrace picture. It is used only to project the texture onto the actual surfaces. Likewise, a bounding surface will automatically be made passive, so as not to be displayed with the actual objects.

This command asks for the texturing ***parameters*** in a similar way to the command `TEXTURE`⇒ `Area`: `Create New`. However, those parameters that do not have significance with bounding objects are not requested.

In particular, *adaptive mapping is not available* with bounding objects. Therefore, the bounding surface should be relatively simple, in particular, the points in the projection and cross-section curves should be as evenly spaced as possible. Simple primitive surfaces or those extruded or rotated from Bézier curves usually make the best bounding surfaces.

Additionally, the command asks for a ***name*** for the bounding object. This is required to assign the correct bounding object to the right surfaces. If you give a name that is already in use, the program complains and asks for a different name.

## TEXTURE Þ Bounding: Assign

This command ***assigns a bounding object*** to a surface or an element. It works in conjunction with the command TEXTURE⇒ Bounding: Object which actually applies the bounding object to the surfaces.

The command first asks for the ***name***, which should be the one that was given when defining the bounding object.

The ***projection method*** determines how the bounding surface texture is placed on the surface. It works by sending a projection ray from each point on the actual surface in the defined direction, and giving the point the color that the ray hits on the bounding object. The available projection methods are:

- ***Vertical (z) -*** Often, a plane above or below the surface(s) can be used as a bounding surface. In such cases, the projection rays will be sent vertically upwards.
- ***Horizontal (x) -*** Works the same way as the previous method, except that the rays are sent to the bounding surface in the direction of the x-axis.
- ***Horizontal (y) -*** Works the same way as the previous method, except that the rays are sent to the bounding surface in the direction of the y-axis.
- ***To object center -*** The projection rays are sent from the approximate center of the surface through the surface points.
- ***Normal direction -*** The rays are sent to the bounding object at right angles to the surface's from each point.
- ***Reflection direction -*** This is the so-called environment mapping effect. It sends the rays to the bounding surface in the reflection directions, as seen from the currently chosen camera view.

## TEXTURE Þ Bounding: Find

This command is available for locating and selecting the target object's bounding surface. Since each bounding texture area is told the name of the bounding surface this command looks for a bounding surface of that name.

## TEXTURE Þ Flatten: Surface

Command Flatten Surface computes a ***minimally distorted flattening*** of a 3D surface storing the result in the xy-plane. For the results to be accurate the command should only be applied to rotational-like surfaces only.

The command has two modes of operation:

- The ***rotational mode*** is intended surfaces with circular cross-sections. It computes the flattening directly without asking for any further parameters.
- The ***general mode*** computes a flattening for "almost" rotational surfaces, for example those with hexagonal cross-sections. This mode only applies to B-spline surfaces. It asks for two additional parameters.

The ***area/distance weight*** parameter determines whether areas or distances on the surfaces should be preserved as closely as possible. A value close to zero puts more importance on distances, and a value close to one appreciates areas. Small weight values are generally preferable, but they can lead to self-intersecting flattened surfaces — increasing the weight value prevents this.

The maximum number of ***iterations*** determines the accuracy and calculation time of the result. Using a large value will gives more accurate results, but will take longer to compute.

## TEXTURE Þ Flatten: Contour

This command converts the edges of the target surface, which has already been flattened, into 2D polylines. This is required prior to transporting the flattened shape to external painting systems. The 2D polylines can be used in other painting packages for designing the texture.

## TEXTURE Þ Flatten: Assign

This command assigns the texture parameters of the flattened surface to the actual surface, which should be selected as the target. It works like TEXTURE⇒ Area: Parameters, except that only those parameters are requested which apply for flattened textures.

## TEXTURE Þ Flatten: Image

This command *bends a rectangular texture image onto an arc*, in the same way that command TEXTURE⇒ Flatten: Surface does for a surface. This way, a ceramic designer can figure out which shape and pattern for a flattened image should be produced, if only the square image originally exists.

The flattened image is produced as a ray traced picture, according to the following steps:

1. Define the surface strip where the texture image is to be applied, using command SURFACE⇒ Change: Split in Two etc. (just like it would be done for the TEXTURE⇒ Flatten: Surface procedure.)
2. Apply the square texture image to the selected surface, using command TEXTURE⇒ Area: Create New. Use the “adaptive mapping” option when defining the texture area.

Now the question is – how should the bent texture strip be in order to be placed on the surface with minimal distortion of the image? To obtain the image of the strip:

3. Flatten the (textured) surface, using command TEXTURE⇒ Flatten: Surface.
4. Now give command TEXTURE⇒ Flatten: Image to the flattened surface.
5. From the Settings Window, set the viewing direction to Z axis.
6. Give command SCENE⇒ Match: Camera to Eye, defining a small view angle (e.g. 1.0 degree).
7. Immediately after this, go and delete the SCENE element !!
8. Now finally ray trace the surface with command EXTRACE⇒ Start/Update, and see the flattend texture being computed.

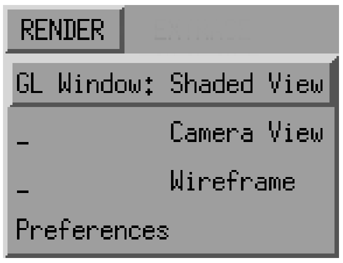
(Maybe one day we make this procedure more automatic ...) To obtain the final image, zoom in, change the backgound color to your liking, and compute the image as batch process if required.

## RENDER MENU

***Fast rendering*** is a method of creating ***shaded*** images of surfaces. It has various modes of operation. There are two choices on the level of image detail, various controls for shading accuracy and an option between software and hardware shading:

- ***Shaded view*** renders the image from the eye view, using just one light source, default material properties, no reflections or transparency, and no textures.
- ***Camera view*** renders the scene from the camera view, using all the light points, colored and spot lights, accounting for the materials, including some texture features, etc.
- The `RENDER⇒ Preferences` command control the accuracy of all shaded and hidden line images, as well as which scene components actually to include in camera view shading.

All the shading commands automatically render all the surfaces that are shown on the screen when the render command is executed.



## Render Window Commands

### Window position and size

After you have given the appropriate `RENDER⇒ ..` command, a ***new window*** appears containing the rendering. ***Position*** the window in the desired location with the mouse, and press the left mouse button.

You can ***resize*** the window by clicking on a corner of the with the left mouse button. To get the window to occupy the full screen, click on the large rectangle at the top right corner of the window.

### Pop-up menu

If you press and hold the ***right mouse button*** down with the cursor in the window, a ***pop-up menu*** appears which contains commands for interacting with the display. Most of these commands interact with the mouse, following the movement regardless of whether a mouse button is pressed down or not. Clicking the right mouse button again finishes the execution of a command.

If you need to ***change the cursor location*** during some command, press the `Ctrl`-button when moving the mouse and the mouse movement won't affect the image.

The various functions are explained here:

- **Move Camera -** The command moves the camera around the model, just like the command `DISPLAY⇒ View: Eye Point.` Horizontal mouse movement moves the viewer horizontally around the object, and vertical movement moves up and down.

  If you hold the middle mouse button down, up and down mouse movement zooms into and out of the object.

  End the function with the right mouse button.

**Note**: If your model is very large, or the machine is slow, you should set the `Move Mode` (below) to the `Box` option. This gives much better interaction than waiting for the shaded image to be continuously updated.

- **Move Light -** The command moves a light around the object. Horizontal mouse movement moves the light horizontally around the object, and vertical movement moves up and down.

  Þ **Light number**

  If there are more than one light point, you may select the light you want to move from the command's sub menu.

- **Move Mode -** This command controls how the shaded object is updated with camera and light movements. It has two options. The current option is displayed in gray, while the other option (which can be selected) is black.

  Þ **Model**

  `Model` is the default, updating the shaded image continuously as you move the mouse.

  Þ **Box**

  This option shows a box around the object as you move the mouse, and shades only after the moving is ended. The `Box` option is recommended with large models and/or slow machines.

- **Pan -** Move the center point of the display with the mouse. You must press and hold the mouse button down when drawing the panning line.

- **Zoom -** Zoom in or out. Press and hold the mouse button down when drawing the zooming window.

- **Twist -** Changes the tilt angle of the camera.

- **Rotation -**

  Þ **X**, **Y**, **Z**

  Provides continuos 360-degree rotation around an axis. It may be interrupted at any time with the right mouse button.

  Þ **Infinite**

  This command rotates the shaded model infinitely on the screen. Depending on the cursor location, the model is spun slower or faster, and in different directions. Clicking the right mouse button ends the command.

- **Clipping -** There are two clipping planes that can be used to cut of the front or rear parts(s) of the model, using the following options:

  Þ **Move near**

  Þ **Move far**

  Change the clipping planes farther or nearer the viewpoint with up and down mouse movements.

  Þ **Double**

  Þ **Half**

  To double or halve the mutual distance between the clipping planes.

- **Save view**

  Þ **Camera**

  Þ **Lights**

Þ **All**

To save the camera and/or light points into the SCENE-element.

- **Textures**

  Þ **Blurring: On / Off**

  Controls whether blurring is applied to textures or not.

  Þ **Colors: Mix with Material / Plain Colors**

  Decides whether texture colors are mixed with the surface colors, or if textures are shown non-shaded in their original colors.

- **Turn two-sided On / Off** - Normally all the parts of the model are shaded as two-sided, regardless of their orientation. However, it may be useful to check that all the parts of the model are consistently oriented. When two-sided shading is turned Off, the incorrectly oriented parts of the model will appear darker than the others.
- **Hide axes** - To disable the drawing of the coordinate axes.
- **Save image -** Stores the current image into a file.
- **Lost in space! –** This command redraws the objects fitted into the window. Very useful if you get lost!
- **Done -** This command closes the window.

## RENDER Þ GLWindow: Shaded View

This command renders a ***default shaded image*** in a separate window. Shaded view renders the image from the eye view with perspective, using just one light source, default material properties, no reflections or transparency, and no textures. Settings in the `RENDER Þ Prferences` command control the display.

The render window interaction rules are the same as in the Render Window Operation section above.

## RENDER Þ GLWindow: Camera View

This command renders the ***complete scene*** in a separate window. Camera view renders the scene from the camera view, using all the light points, colored and spotlights, accounting for the materials, including some texture features. Settings in the `RENDER Þ Preferences` command control the display.

The render window interaction rules are the same as in the Render Window Operation section above.

## RENDER Þ GLWindow: Wireframe View

This command draws a wire-frame display in a separate window.

The window interaction rules are the same as in the Render Window Operation section above

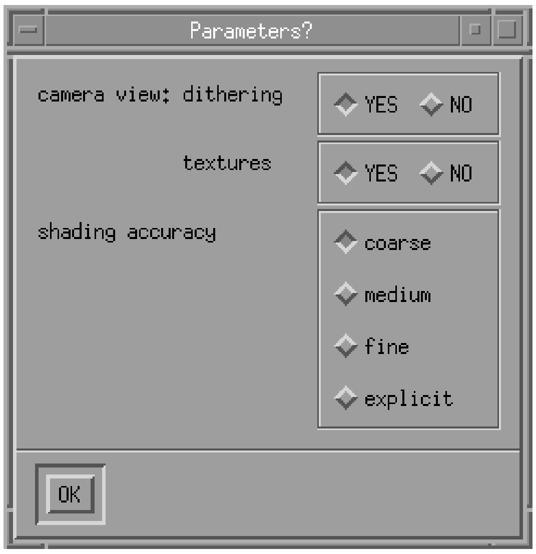
## RENDER Þ VR export

This command allows you to output VRML from your geometry data for web display. The command can output the model in three different levels as defined in the VRML standard. If necessary, the trim curves can be removed from the model before VRML export.

*Note:* Texture images are not automatically converted to .jpg or .png format. If your model contains textures they have to be converted manually with other image processing software products from .tiff to .jpg or .png..

## RENDER Þ Preferences

The `Preferences` parameters give you various controls for the render image quality.



These first three options control some camera view shading features.

- ***Camera View: dithering***

  This defines whether or not the image should be dithered to improve colors on limited graphics machines.

- ***Camera View: textures***

  This defines whether textures should be included in the image. Even if textures are rendered there are limitations as to what textures can be rendered.

The rest of the parameters define the accuracy of the rendering the surfaces. They apply equally to all of the options in the `RENDER` menu.

- ***Accuracy: coarse / medium / fine***

  The rendering is achieved by triangulating the model into small facets. The number of facets generated determines the apparent smoothness of the model in the picture and also the speed of rendering. These options control implicitly the number of facets generated.

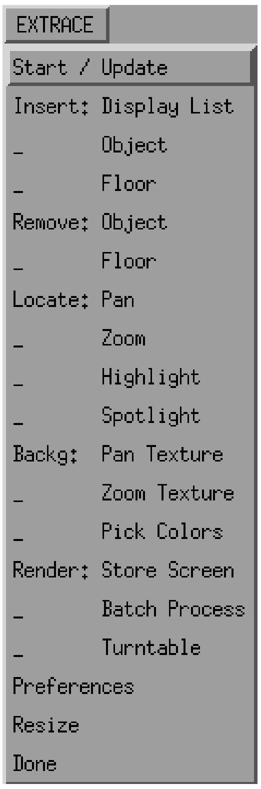
- ***Accuracy: explicit***

  This command allows you to define a numerical value for the tolerance. The value is typed into parameter window which appears after this dialog box is closed.

  Do not use very small tolerances, like 0.001. The conversion to polygon form could then take even longer than exact shading or ray tracing.

## EXTRACE MENU

This menu contains commands for generating the highest quality pictures in DESKARTES. They are generated by a technique called ***ray tracing***. The software module that creates the pictures is called ExTrace, hence the menu title. Most of the definition of the appearance of the image is achieved by commands in the `SCENE` menu.



### EXTRACE Þ Start/Update

This command ***starts*** the ExTrace program, or begins the ***update*** of the ExTrace display if you have made changes to the model or the image parameters.

As you start an ExTrace session, all the scene definitions (lights, camera and materials) are automatically loaded for the image. Any changes you make with the `SCENE` menu will immediately update the ExTrace session too. You'll see the changes when you next render the image.

A parameter window appears requesting the following information:

- ***Insert display list*** - This option automatically inserts all the objects currently shown on screen into the ExTrace image, and renders them.
- ***Match camera to eye*** - This sets the view of the rendered scene to match the current display in the main window. It has the same effect as the command `SCENE⇒ Match: Camera to Eye`. If you have already set up the camera to the correct position set this option to `NO`.
- ***Picture width*** - This number defines the horizontal size, in pixels, of an image rendered on screen. You may later redefine the size with the command `EXTRACE⇒ Resize`. This size is only used for the picture on the screen not for the final picture so don't make this value too large.

- ***Picture height*** **-** This number defines the vertical size, in pixels, of an image rendered on screen. You may later redefine the size with the command `EXTRACE⇒ Resize`. This size is only used for the picture on the screen not for the final picture so don't make this value too large.

After the parameters are given, the ***ExTrace window*** appears. It should be moved to a suitable location on the screen and fixed by clicking the left mouse button.

ExTrace will first render the picture with low pixel resolution, and progressively ***refine*** it to the final resolution. The resolutions may be controlled with the `EXTRACE⇒ Preferences` parameters.

You may always ***interrupt*** the on-screen computation with the `Stop` button in the ExTrace window. Alternatively, the keys `ESC` or `Ctrl-C` do the same thing when the mouse cursor is positioned in the ExTrace window. The scene specifications, objects, textures, etc., may then be further changed, and he rendering re-started by issuing command `Start/Update` again.

After the final resolution has been reached, if it greater than zero, the ExTrace function button `Divide` is enabled to further refine the image.

On machines that don't support 24-bit color, you are additionally given the options `Optimize` and `Dither`. These functions optimize the image for display in 8-bit colors. Which of the two methods gives the best result depends on the particular image. When the picture is computed as a background process, such optimizations are performed automatically for the best possible 8-bit image quality.

## EXTRACEÞ Insert: Display List

This command ***inserts*** all the surfaces shown on the screen into ExTrace. Next time you render the ExTrace scene, you will see the inserted objects.

You are asked whether you wish to remove those surfaces already in ExTrace. If you remove the surfaces, only the new ones inserted will be shown. If you do not remove the surfaces, these new surfaces will be added to those already in ExTrace. If, by mistake, you add in a surface that already exists in ExTrace you will get a duplicate surface. This will make the computation slower and may produce unusual effects in the picture. If you suspect this has happened simply reinsert all surfaces choosing to remove the originals.

When inserted for ExTrace visualization, the surfaces are written into special ExTrace object files. These are stored a sub directory called `ExSpl_files` within the model directory. A unique number identifies each object. DESKARTES uses this number to decide whether a surface has changed since it was last used in ExTrace. If it has not changed it simply reuses the previous definition.

## EXTRACEÞ Insert: Object

This command ***inserts the target surface or element*** into ExTrace adding it to the surfaces already there. Next time you render the ExTrace scene, you will see the inserted objects.

When inserted for ExTrace visualization, the surfaces are written into special ExTrace object files. These are stored a sub directory called `ExSpl_files` within the model directory. A unique number identifies each object. DESKARTES uses this number to decide whether a surface has changed since it was last used in ExTrace. If it has not changed it simply reuses the previous definition.

## EXTRACEÞ Insert: Floor

This command defines a ***floor object*** below all the objects that have been inserted into ExTrace. The floor is a square block, with user-specified width.

The floor materials are taken as the default `floor` and `floor2` materials. To change them, use command `SCENE⇒ Edit: Materials`.

The floor appearance may be single colored, patterned, or contain a texture image. Appropriate parameters are requested depending on the choice made.

If you later wish to change the size, location or appearance of the floor, just give the command `Insert Floor` again.

## EXTRACE Þ Remove: Object

This command ***removes the individual target surface or element*** from the ExTrace picture.

If you have set the EXTRACE⇒ `Insert` parameter "object updating" to "automatic", this command will not be needed as changed objects are automatically updated by DESKARTES.

Otherwise, if you wish to ***change a surface and manually update it*** in ExTrace, first remove the old object with EXTRACE⇒ `Remove`: `Object`, then change the object (for example rebuild or transform), and finally reinsert the modified surface with EXTRACE⇒ `Insert`: `Object`. Next time you render the ExTrace scene, you will see the updated objects.

## EXTRACE Þ Remove: Floor

This command ***removes the floor*** from the ExTrace image. Next time you render the ExTrace scene, the floor will not be visible.

## EXTRACE Þ Locate: Pan

This command allows you to graphically determine the ***center point*** of the ExTrace image. ***Click*** and ***hold*** the left mouse button with the cursor at a point in the ExTrace window, ***move*** to where you want the point to be then ***release*** the mouse button.

The update of the ExTrace image will begin immediately. This command actually updates the view direction of the camera in the SCENE element in the Object Window.

## EXTRACE Þ Locate: Zoom

This command increases or decreases the ***scale*** of the ExTrace view.

To zoom in ***click*** and ***hold*** the left mouse button with the cursor at the center point of the new view in the ExTrace window, ***move*** the mouse vertically until the box encloses the view then ***release*** the mouse button.

To ***zoom out*** click the right mouse button.

The update of the ExTrace image will immediately begin. This command actually updates the view angle of the camera in the SCENE element in the Object Window.

## EXTRACE Þ Locate: Highlight

This command changes the location of a specified light point graphically, according to where you wish is to cast the ***highlight*** on a surface. The light number is requested as a parameter.

Point at the ExTrace image with the left mouse button to show where you would like the gleam from the light point to be. When pointing, there must be an actual object (other than the background) drawn at the selected point.

It is also possible to define ***two highlight locations*** for the same light, by pointing to both locations with the middle mouse button instead of the left. Be careful with this feature, as it is not possible to locate the light point if the high lights come from opposite directions.

The update of the ExTrace image will immediately begin. This command actually updates the position of the lights in the SCENE element in the Object Window.

## EXTRACE Þ Locate: Spotlight

This command changes the ***spot light field*** graphically. The light number is requested as a parameter.

Point at the ExTrace image with the left mouse button to show where the extents of the gleam from the light point should end. When pointing, there must be an actual object (other than the background) drawn at the selected point.

The update of the ExTrace image will immediately begin. This command actually updates the position of the lights in the SCENE element in the Object Window.

## EXTRACE⇒ Backg: Pan Texture

This command allows the user to ***pan the background texture***, which has been defined with `SCENE⇒ Define: Backg`. It works in conjunction with the command `EXTRACE⇒ Backg: Zoom Texture.`

***Click*** and ***hold*** the left mouse button with the cursor at a point in the ExTrace window, ***move*** to where you want the point to be then ***release*** the mouse button.

The update of the ExTrace image will immediately begin.

## EXTRACE⇒ Backg: Zoom Texture

This command enables the ***zooming of the background texture***, which has been defined with `SCENE⇒ Define: Backg`. It works in conjunction with the command `EXTRACE⇒ Backg: Pan Texture.`

To zoom in ***click*** and ***hold*** the left mouse button with the cursor at the center point of the new view in the ExTrace window, ***move*** the mouse vertically until the box encloses the view then ***release*** the mouse button.

To ***zoom out*** click the right mouse button.

The update of the ExTrace image will immediately begin.

## EXTRACE⇒ Backg: Pick Colors

This command allows you to specify ***multiple background colors*** at different levels of the image.

The left mouse button is clicked at the desired height in the ExTrace window, after which the color palette will pop up for color selection. Use the command for each background color that you wish to define.

The chosen background colors may also later be edited with command `SCENE⇒ Edit: Materials`.

To disable the variable color background, choose another background option with command `SCENE⇒ Define: Backg Type`.

## EXTRACE⇒ Render: Store Screen

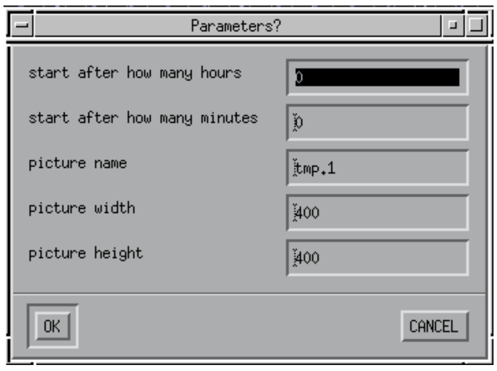
This command allows you ***save the image rendered on screen into a file***. You are asked for the name of the file. This should only be used for quick generation of images, as the picture in the visible ExTrace window will never be generated to the highest quality. The highest quality images are generated as a batch process after exiting from ExTrace.

## EXTRACE⇒ Render: Batch Process

This command computes the picture as a ***batch process***. A batch process is a job given to the computer that it completes in the background. You may continue to work on the computer but you may notice a slight reduction in response, especially with Windows machines.

After executing the command a parameters window appears requesting the ***image name and size***. The size is usually different from the size of the window on the screen but the proportions of the new sizes should be the same as the original sizes. For example, if you originally chose a size of 400x600 then new sizes of 600x900 or 800x1200 would be suitable.

The startup time of the batch processes is given as hours and. If both the hours and the minutes are set to zero, then the batch process is started directly without any delay. Otherwise the minimum delay is two minutes. This way, you may start computing the picture at night without bothering interactive work on your workstation.


Parameters?

| | |
|---|---|
| start after how many hours | 0 |
| start after how many minutes | 0 |
| picture name | tmp.1 |
| picture width | 400 |
| picture height | 400 |

The size you choose is determined by how you expect to view the picture. If you intend to view the picture on the screen there is no point making a picture bigger than the screen resolution. If you wish to output the picture to a printer higher resolutions may be needed. For example a typical printer resolution is 300 DPI or dots per inch. If you wanted to print the picture 10 inches by 8 inches then the maximum size of the picture needed would be 10x300 by 8x300 or 3000 by 2400 pixels (this image would require 3000x2400x3/1024/1024 over 20 megabytes of storage). Often you can get an acceptable result with less pixels as the printer software will smooth out the picture.

You may start ***several batch processe***s during the same session to create versions of the same scene. However, avoid running several ExTrace batch processes at the same time, otherwise all the computer time may be spent managing the processes, and no actual picture computation is done. Assign different starting hours to each batch process, so as not to overload the machine.

## *Batch Job Queues*

There is also a simple queueing system for ExTrace batch. The queueing system forces the ExTrace batch jobs to run sequentially, one at a time.

The queueing functionality is implemented in the ExTrace program, so no additional server process is needed. Each ExTrace batch process checks the queue at startup and adds itself into the queue. If there is another batch job already running, the process goes into a wait mode checking periodically if the other job has already finished. When this happens, it enters the image calculation. After finishing the calculation, the process removes itself from the queue, allowing the next process to start.

The queueing system is not enabled as default, but has to be switched on using the command line options for ExTrace.

There are three new options available to control the queueing system:

| | |
|---|---|
| -u | enables the queueing system for batch jobs (the -n option has to be given also) |
| -t secs | sets the time that EXTrace sleeps between the queue checks (default is 300 secs = 5 min) |
| -v cnt | sets the number of batch jobs allowed to run concurrently (default is 1) |

For the batch jobs the command line parameters are given in their startup script. In WinNT version this file is called "ExTBatch.ksh" and on Unix/MacOS platforms "ExTraceBatch".

Example commands, using the queueing options, are found in comments, in the script files.

The queueing system creates two files in a temp directory of the machine:

| | |
|---|---|
| DAEXTqueue | queuefile |
| DAEXTqlog | logfile containing information about queueing events |

On WinNT the path of the temp directory is taken from the system environment variable TEMP.

On Unix/MacOS platforms the /tmp is used as temp directory.

Both queueing system files are in simple ascii format, so they can be viewed in any text editor.

If the queueing system is in use and a EXTrace process crashes during the image calculation, before it removes itself from the queue, the queue gets stuck.

The queue can be released by manually removing the line for that crashed process from the queue file DAEXTqueue. However when manually editing the queue file, be careful to delete only whole lines (records), otherwise the queue gets corrupt.

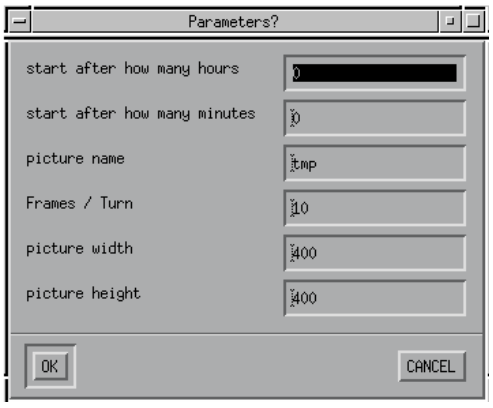
In case of problems, the queueing system can be reset by deleting the queuefile (DAEXTqueue) and killing the EXTrace processes. The queuefile will be automatically recreated, if it doesn’t exist when the first queueing EXTrace process starts.

## EXTRACE Þ Render: Turntable

This command computes ***multiple images*** from the scene as a batch process. The scene and model are the same in every image, but the ***eye point moves around the objec***t. In other words, the command creates an animation sequence of the object rotating around its center.

A batch process is a job given to the computer that it completes in the background. You may continue to work on the computer but you may notice a slight reduction in response, especially with Windows machines.

After executing the command a parameters window appears requesting, the image ***name and size***. The size is usually different from the size of the window on the screen but the proportions of the new sizes should be the same as the original sizes. For example, if you originally chose a size of 400x600 then new sizes of 600x900 or 800x1200 would be suitable. Turntable animations can take a lot of calculation time and storage space so don’t make the picture too large.


Parameters?

| | |
|---|---|
| start after how many hours | 0 |
| start after how many minutes | 0 |
| picture name | tmp |
| Frames / Turn | 10 |
| picture width | 400 |
| picture height | 400 |

In addition, you will be asked for the ***delay*** before starting the computation. The startup time of the turntable processes is now given as hours and minutes. This way, you may start computing the picture at night without bothering interactive work on your workstation.

In addition, the program asks for the ***number of Frames / Turn***. This indicates how many images are to be calculated for one complete turn around the model. The higher the number the smoother the final animation but the longer the calculation will take and the larger the animation file will be.

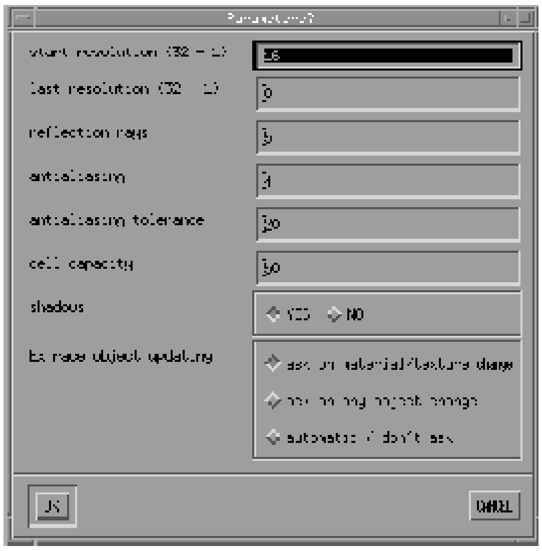
Once the images are ready, they will be stored in a separate directory in the File Window. Select the ***movie directory*** in the File Window, and use the `FILE`⇒ `Files`: `Convert` command to create the movie.

The parameters for advanced turntable control are collected into a ExTrace turntable preference dialog (`EXTRACE`⇒ `Prefs: Turntable`).

Note: The `Turntable` command is only available for Windows and SGI workstations with IRIX 5.3 or higher.

## EXTRACE Þ Prefs: General

You can control the accuracy of the image with various ***image computation parameters***. The most important ones of these are the on-screen image resolution parameters. An experienced user may also find the parameters useful, to optimize the computation times and image quality.



| | |
|---|---|
| start resolution (32 - 1) | 16 |
| last resolution (32 - 1) | 0 |
| reflection rays | 5 |
| antialiasing | 4 |
| antialiasing tolerance | 50 |
| cell capacity | 50 |
| shadows | YES NO |
| ExTrace object updating | ask on material/texture change<br>ask on any object change<br>automatic / don't ask |

The available options are:

- ***Start and last resolutions -*** These two parameters define the drawing resolution for computing images on screen. They determine how large the macro pixels will be used to start the rendering, and how small they will be refined too.

  A macro pixel is a ‘super’ pixel that is a multiple of the normal pixels. For example, a value of 8 would mean computing 8 × 8 pixel samples of the actual image then refining these to 4 then 2 then 1. Each refinement of the picture halves the value, thus, only powers of two (1, 2, 4, 8, 16,...) are allowed for the macro pixel sizes.

  `Last resolution` value determines when the refinement stops. A value of zero leads to antialiasing the image in the last refinement.

- ***Antialiasing -*** Antialiasing means that the picture is refined one step further than the single pixel size, to the so-called sub-pixel resolution. This results in a high-quality picture where the intensity variations between individual pixels have been smoothed out. The picture looks less ‘computer generated’.

  The default is to compute four samples per pixel for antialiasing. This is enough for most purposes, but for very high quality images, you may try larger values. Small antialiasing values, on the other hand, make the computation faster. A value of zero means no antialiasing.

- ***Antialiasing tolerance*** **-** This parameter affects where in the image antialiasing is applied. Antialiasing is not done for pixels whose intensity is already close enough to the neighboring pixels. If you only wish to have antialiasing where the intensity variations are noticeable, such as object silhouettes, choose a large tolerance value. This may save a lot of computation time.
- ***Cell capacity*** **-** This parameter affects some optimizations regarding how objects are stored in ExTrace. You may only need to alter it when dealing with very, very large models. Setting a large cell capacity slows down the computation, but it helps with memory problems.
- ***Reflection rays*** **-** The maximum number of individual ray bounces can be altered. The default value of five is more than enough for all practical purposes, but it may consume a lot of computation time in complex mirroring scenes.

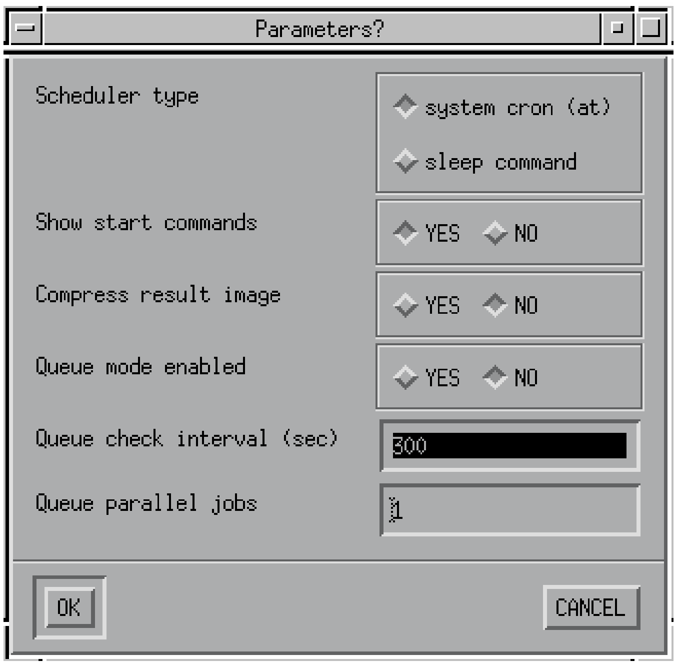
  The value of two is usually a sensible minimum choice. You will then see objects mirroring each other, and in the mirrored images one further level of mirroring.
- ***Shadows*** **-** Determines whether shadows are computed or not. By default, they will, but with rendering glass objects shadows might not be required, and it could save a lot of computation time to switch the shadows off.

The last parameter option controls ***how the ExTrace model is updated*** when the objects, materials and textures are changed. The options are

- ***ask on material change*** **–** Using this option, only material and texture changes are updated into ExTrace, and the permission for each change is asked from the user separately.
- ***ask on any object change*** **–** With this option, material and texture changes are updated into ExTrace without asking, but the system will also update any other object changes after asking for permission from the user.
- ***automatic update / don't ask*** **–** With this option the system updates all object changes into ExTrace automatically, without any user interaction.

## EXTRACE Þ Prefs: Batch Process

The parameters for controlling the ExTrace batch processes can be changed in this command. The following dialog appears:


Parameters?

Scheduler type: [✓ system cron (at)] sleep command
Show start commands: [✓ YES] NO
Compress result image: YES [✓ NO]
Queue mode enabled: YES [✓ NO]
Queue check interval (sec): 300
Queue parallel jobs: 1

**Scheduler type :** selects scheduler method to delay the startup of the batch processes. If the startup delay is defined as zero hours and minutes, this doesn't have any meaning.

**Show start commands :** print startup commands plus other information in the DeskArtes startup window. This is useful when trying to resolve problems.

**Compress result image :** the resulting image file is compressed using gzip compression.

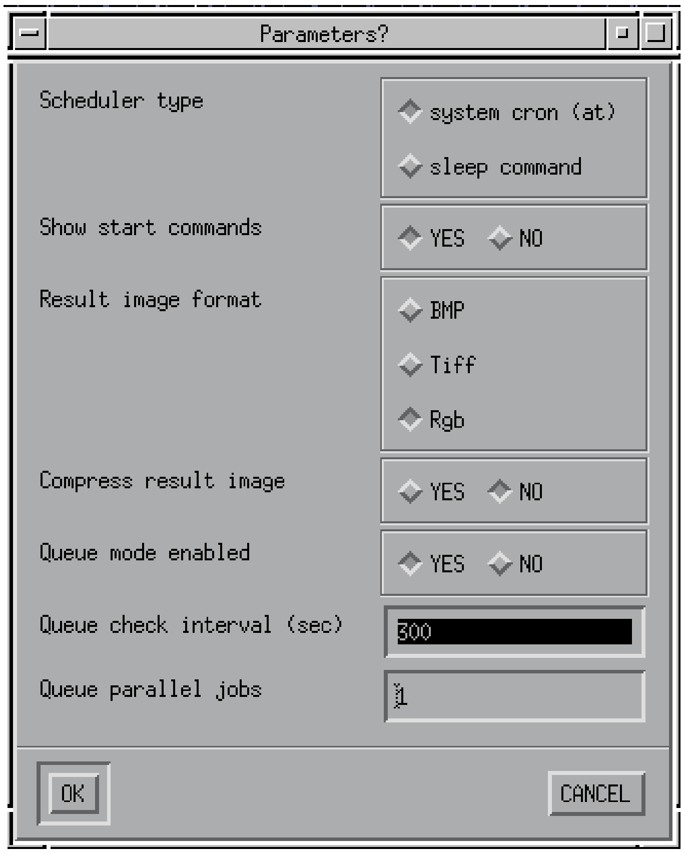
**Queue mode enabled :** enables the queueing system for batch processes. The queue system forces the batch processes to run serially.

**Queue check interval :** sets the time that ExTrace sleeps between the queue checks (default is 300 secs = 5 min).

**Queue parallel jobs :** sets the number of batch processes allowed to run concurrently (default is 1).

## EXTRACE Þ Prefs: Turntable

This command enables changing the default behavior of the turntable animation calculation.



Parameters?

| | |
|---|---|
| Scheduler type | [✓ system cron (at)] sleep command |
| Show start commands | [✓ YES] NO |
| Result image format | BMP Tiff [✓ Rgb] |
| Compress result image | YES [✓ NO] |
| Queue mode enabled | [✓ YES] NO |
| Queue check interval (sec) | 300 |
| Queue parallel jobs | 1 |

OK CANCEL

**Scheduler type :** selects scheduler method to delay the startup of the turntable processes. If the startup delay is defined as zero hours and minutes, this doesn't have any meaning.

**Show start commands :** print startup commands plus other information in the DeskArtes startup window. This is useful when trying to resolve problems.

**Result image format :** the resulting image files are automatically converted to the selected format.

**Compress result image :** the resulting image files are compressed using gzip compression.

**Queue mode enabled :** enables the queueing system for turntable processes. The queue system forces the turntable processes to run serially with other ExTrace batch processes. The turntable processes are always run one at a time.

**Queue check interval :** sets the time that ExTrace sleeps between the queue checks (default is 300 secs = 5 min).

**Queue parallel jobs :** sets the number of batch processes allowed to run concurrently (default is 1).

## EXTRACE Þ Resize

This command allows you to ***resize*** the ExTrace window while running it on screen. The new size is given as the width and height values in pixels, in the same way as the window was originally launched.

## EXTRACE Þ Done

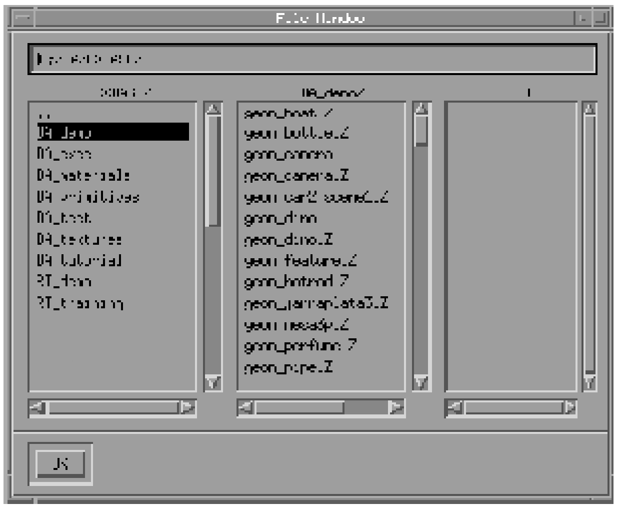
Ends the execution of ExTrace on-screen. If no batch process has yet been started, you are given the option of starting one at this point.

# 4 WINDOW COMMAND EXPLANATION

Three important sub windows when using DESKARTES are the File Window, the Object Window and the Settings Window. These each have commands associated with them that are explained here.

## File Window

Through the ***File Window***, you may read and write information into DESKARTES, display pictures on the screen, and perform various tasks related to file maintenance.



The File Window is shown and hidden with the command `SYSTEM⇒ Show: Files`. The File Window commands are structured in three ***pop-up menus*** each associated with one of the three panes of the File Window. By pressing and holding the middle mouse button in the leftmost pane you'll get the directory pop-up menu, which deals with commands for ***directories***. By pressing and holding the middle mouse button in the middle pane you get the file pop-up menu, which deals with commands for ***files***. Pressing the same button in the rightmost field pops up the ***help directory*** menu.

The File Window commands apply to the ***selected*** files or directories, which are highlighted in black. You may select a file or a directory by pointing and clicking on its name with the left-hand mouse button.

## File Identifiers

All file names have an ***identifier***, either a ***prefix*** at the beginning of the file name, and/or a ***suffix*** at the end. Every time a file is written from DESKARTES, the system automatically adds an identifier. The identifier enables you to determine the contents of the file.

Typically, ***prefixes*** are used for files that are ***internal*** to DESKARTES, and not recognized by other systems. ***Suffixes*** are used with ***general*** file formats, which are used as standards for communicating between different systems.

The following identifiers are in use:

| | |
|---|---|
| `geom` | model file containing an entire model |
| `da` | another model file suffix for compatibility with the DeskArtes Expert Series (since version 4.2.3). |
| `safe` | backup copy (the latest modeling situation) |

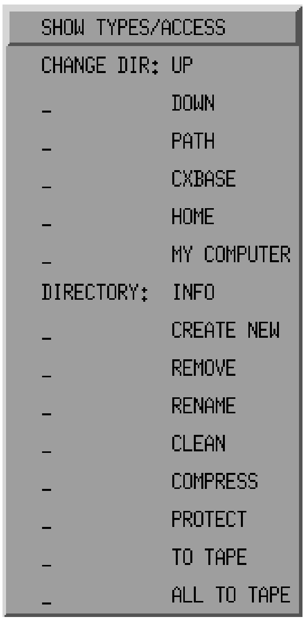
| | |
|---|---|
| `cobj` | object file containing part of a model |
| `bump` | object file containing a 3D bump object |
| `cmat` | object file containing a material definition |
| `log` | command series file |
| | |
| `tex` | texture picture file |
| `pic` | any picture file |
| `pvpic` | PreView picture file |
| `expic` | ExTrace picture file |
| `exscene` | ExTrace command file |
| `ExSpl` | ExTrace part directory |
| | |
| `dxf` | DXF data transfer file |
| `vda` | VDA-FS data transfer file |
| `igs` | IGES data transfer file |
| `stl` | Rapid Prototyping data transfer file |
| `asc` | ASCII coordinate data file |
| `hpg` | HPGL plotter file |
| `eps` | Encapsulated PostScript file |
| `aps` | Adobe Illustrator PostScript file |
| `tif` | TIFF format file |
| `movie` | Directory with PNM files for creating a turntable animation |

A suffix ".Z" or ".gz" after the file name means that the file is ***compressed***. Compressed files take much less disk space than uncompressed ones. Most DESKARTES commands are able to treat compressed files just as uncompressed ones.

**Note**: The identifiers of file names are ***not*** typed in when the system asks for file names as parameters. For example, if the texture you want to use for an ExTrace picture is contained in a file called tex_flower.pic.Z (or tex_flower.pic.gz), you would only type "flower" when asked for the texture file name. DESKARTES will automatically add the missing prefix and suffix.

## *Directory Pane Commands*

### Show Types/Access



This command displays and hides the ***authorizations*** of all the directories and files in the File Window. The authorizations determine who is allowed to access particular files or directories. The access control is based on the name you used to logon to the computer.

Authorizations are displayed as a three-letter combination for "everyone-group-owner", where "-" means no authorizations, "r" means read authorization, and "w" allows both writing and reading. For example, "-rw" would mean that the owner of the directory is allowed to read and write to the directory. Other people in the group may read the directory but others have no access at all.

See command `DATABSE`⇒ `Directory`: `Protect` for further explanations.

### Change Directory

These commands allow for ***selecting the model directory*** anywhere in the ***file system hierarchy***. The available commands are:

| | |
|---|---|
| `UP` | Moves one level up the directory tree. Alternatively, you may just double-click on the ".." item at the top of the directory name field |
| `DOWN` | Moves down into the currently selected directory. Alternatively, you may just double click on the directory's name in the file name field. |
| `PATH` | Asks for the full path of the desired directory. |
| `CXBASE` | Moves to the default DESKARTES Database (set with the environment variable $DA_DECABASE). |
| `HOME` | Moves to the user's home directory (set with $HOME) |
| `MY COMPUTER` | Moves to the 'My Computer' directory in Windows. |

## `Directory`: `Info`

This command displays the ***directory information***, i.e., size, read and write authorizations, owners, and the last modification and access dates. The information is shown in a separate information window.

## `Directory`: `Create New`

Asks for the name of a ***new directory***, then creates it. DESKARTES allows you to use directory names that have two parts. These parts are described as the user name and the model name, which are joined together as `username_modelname.` However, you can use the two parts of the name however you want depending on how you wish to organize your work. For example, you may choose to use the first part as a project, customer or range name instead of a user name.

## `Directory`: `Remove`

***Deletes a directory*** and all the files within it. This cannot be undone. DESKARTES asks for confirmation before deleting.

## `Directory`: `Rename`

Gives a new name to an existing directory.

## `Directory`: `Clean`

Deletes the "unnecessary" files within a directory. These are the object (`cobj`), ExTrace spline (`ExSpl`), ExTrace scene (`exscene`), teach (`log`) and size (`.sz`) files. This cannot be undone. DESKARTES asks for confirmation before deleting.

## `Directory`: `Compress`

This command ***compresses*** all of the files within the selected directory. This saves disk space. To decompress the files, use the command `DATABSE`⇒ `Directory`: `Compress` again.

## `Directory`: `Protect`

This command sets the ***authorizations*** of a directory. Only the user who created the directory may change the authorizations of a directory. You are asked, who is to be given read and write authorization. The possibilities are `all`, `group`, and `me`.

The `all` option gives everybody the right to use the directory. This is useful especially for general directories like `DA_materials` and `DA_textures`.

`Group` typically means all of the people working on a project. The `group` may be given rights to examine your model directories (read authorization) while keeping the right to change them to yourself (`me`).

Note that this function is based on the login name. If all users are using DESKARTES with the same login name, the protection functions have no meaning whatsoever.

## Directory: To Tape

Writes the content of the selected directory on magnetic tape, or diskette, or some other data storage device. This command is available only for the SGI platform.

## Directory: All To Tape

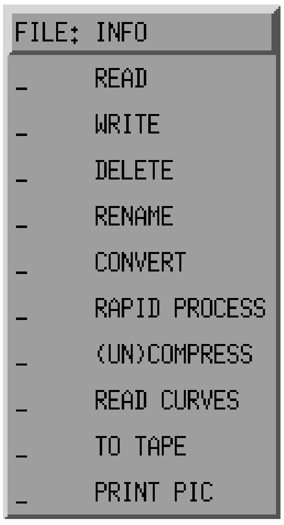
Makes a tape backup of all model directories, i.e., the complete DESKARTES database. This command is available only for the SGI platform.

Perform this function regularly, otherwise all your model and image data could be permanently lost if a hardware failure or other such disaster occurred.

## *File Pane Commands*

## File: Info

Displays the ***file information***, such as name, type, read and write authorizations, owner, size and time last changed in the file info window. If the file is an image (in one of the supported formats), its width and height are displayed, as well.



## File: Read

***Reads a geometry file*** into DESKARTES or ***displays an image file*** on the screen.

Alternatively to selecting this command from the menu, you may ***double click*** on the file name to read a file.

**Note:** If you try reading a file that has been created by another user, the read might fail due to a protection violation. See the `Files`⇒ `INFO` and `PROTECT` commands.

The following types of files are available for reading:

***Model file*** *(identifier `geom`)*

Reads a DESKARTES model for further work within DESKARTES. If needed, the system asks if the current model should be replaced, or if the file model will be combined with the current one.

***Object file*** *(identifier `cobj`)*

Reads a DESKARTES object after the current target object, or as its sub-object.

***DXF file*** *(identifier `dxf`)*

The command reads a DXF file created by another modeling system for use within DESKARTES. Version 11.0 of the DXF standard is supported.

***IGES file*** *(identifier `igs`)*

Reads an IGES file (created by another modeling system) for use within DESKARTES. Version 5.0 of the IGES standard is supported. A list of entities found and processed in the IGES file is printed to the command window where DESKARTES was launched.

The command asks for an ***approximation tolerance*** as a parameter. It is used for data conversions, if the file contains curves or surfaces of a higher degree than cubic polynomial.

As another parameter, the ***tolerance for optimizing trim curves*** is asked for. If set to a positive value, it automatically performs the command `FORMAT`⇒ `Surface`: `Reduce Trims` to the model. This is useful to prevent having "too" accurate, but slow trim curves.

The third option, ***recompute all trim curves***, forces the program to trim all surfaces with the 3D trim curves in the IGES file, instead of the 2D parameter space trim curves. The 2D trim curves are generally preferred by DESKARTES, and they are faster to read, too. However, if there are any problems with reading the file then you may try to improve the situation by recomputing the trim curves from 3D.

If there are no 2D trim curves defined in the IGES file, then the recomputation is done automatically.

The parameter ***force tangent continuity*** determines how tangent continuity between neighboring surface patches is treated when the surfaces are converted into bicubic representation.

By default tangent continuity between patches is not forced, and each surface patch in the file is converted within the given approximation tolerance without worrying about the neighbors. This leads to less data in the resulting surfaces than when forcing tangent continuity.

The next parameter option is to ***read incomplete data or not***. Incomplete data here refers to all kinds of data in the IGES file that does not seem to be part of the actual geometry, or is incorrectly defined, but may be nevertheless processed for viewing.

The final option with reading IGES is whether to convert the ***data units*** into *millimeters* or *inches*.

It might take a long time to read complex models from IGES files, especially if the 2D trim curves are not defined. A separate window shows how the reading of the file is proceeding, and it also allows for canceling of the reading if required.

***VDAFS file*** *(identifier `vda`)*

Reads a VDAFS file (created by another modeling system) for use within DESKARTES. Version 2.0 of the VDAFS standard is supported.

The three first parameters are the same as when reading IGES files. As a fourth parameter, DESKARTES asks, whether the surfaces should be stored in just one element, or ***separate elements*** for each surface. The latter is usually better if there are not many surfaces in the VDAFS file.

It might take quite a long time to read complex models from VDA files. The message lines show how many percentages have been computed as the processing goes on.

***Rapid Prototyping file*** *(identifier `stl`)*

Read the contents of an STL file (stereolithography standard) as a ***faceted model***. Both the binary and ASCII formats are supported.

***ASCII Point Data*** *(identifier `pnt`)*

This option allows the reading of ***arbitrary ASCII point data files*** into the system. In order for the system to recognize the file as ASCII points, it has to have an identifier "`.pnt`" at the end of the file name.

The file must start with the string "`START POINTS`", or "`START FACETS`" in its first line, and end with "`END POINTS`", or "`END FACETS`" in the last line. The user must typically add these lines to the original file created by some other system.

The difference between a "`POINTS`" and a "`FACETS`" file is that the former is interpreted as a disconnected point cloud, whereas in the latter case the points are interpreted as consecutive triangle vertices.

The ***numbers*** in the file can be in floating point form, with or without the decimal point, but ***without an exponent part***. For example, 12345.67 would be recognized correctly, but 123.4567E+3 not.

There can be any non-numeric characters between the numbers, such as alphabetic characters, blanks and commas. For example, both the lines "X 123.45, Y -67.8, Z +90" or "123.45 -67.8 90" would be recognized as the same three numbers.

***Sun Raster file*** *(suffix `pic` or `rgb`)*

Displays a Sun Raster file format picture on the screen. The picture may have been created with DESKARTES, or any other CAD system.

The picture may be erased from the screen by pointing at it with the mouse and hitting `q` on the keyboard.

***TIFF file*** *(suffix `tif`)*

Shows a TIFF picture file on the screen. The picture may be removed from screen with the `q` key.

***BMP file*** *(suffix `bmp`)*

Shows a BMP picture file on the screen. The picture may be removed from screen with the `q` key.

***Text file***

If the file to be read is not any of the formats mentioned above, it is assumed to be an (ASCII) text file, and it is loaded for text editing in a separate window.

You may choose one of the UNIX/MACOS text editors “more”, “vi” or “emacs” for viewing the file. The “more” editor is easiest for users who don’t know UNIX, while the two latter also allow for text editing, *i.e.*, changing the contents of the file.

## `File`: `Write`

This command writes geometry, pictures, or other data into a file, from which it may later be read.

**Note**: writing a file may fail if you don't have the ***authorizations*** to write in the chosen model directory. Another possible reason can be that the file system is full. The authorizations may be checked with command `DIRECTORY`: `INFO`. The available disk space is seen with the UNIX command “`df`”, issued in a command window.

As a parameter, the command asks for the type of file to be written. The possibilities are:

***Model*** *(identifier* `geom`*)*

Writes all of the model geometry (under `root`) into a model file. This is the best way to save your models permanently.

***Object*** *(identifier* `cobj`*)*

Writes only the target object, *e.g.*, a curve, into an object file.

***Picture*** *(identifier* `xwpic`*)*

Stores the contents of any window on the screen or selected parts of the graphics window into an image file. Which window or window part to store is selected as parameter, and/or shown with the mouse.

***Primitive***

Stores a surface into DESKARTES's extendable primitive library, `user_primitives`. It can later be conveniently restored from there with the command `SURFACE`⇒ `Design`: `Primitive`, or read normally as an object file.

***Data transfer***

Various data transfer formats are available to move geometry to other CAD/CAM and Desktop Publishing systems.

The available data transfer formats are:

-> ***IGES*** *(identifier* `igs`*)*

Writes the contents of the target object (curve, curve set, surface, element, or root) into an IGES file. This is used to transfer data into other modeling systems, *e.g.* for CNC machining. Version 5.0 of the IGES standard is supported.

The command writes everything under the target object to the file. To write only surfaces, for instance, you may first gather them into one element with command `SELECT`⇒ `Collect`: `Actives`.

As parameters, the command first asks for the ***file name***.

The second parameter gives a choice of ***optional flags***. The optional flags can be used to define which information is not required to be written to the IGES files. With the ***default*** choice such flags are left out, while the ***explicit*** option includes them. Which choice to use depends on the flavors of the receiving CAD/CAM system, but it usually does not make any difference at all.

The third parameter, ***surface representation***, gives a choice of outputting the surfaces as Parametric Splines, or Rational B-splines (NURBS): The latter alternative (Rational B-splines) is generally recommendable, as it makes the data transfer file size much smaller than the Parametric Splines.

The fourth parameter gives a choice of ***trim representation***. It has two alternatives: Polylines, and Splines. The Polylines option outputs all trim curves as linear polygonal curves, while the Splines option converts all trim curves into smooth parametric curves. Many receiving systems seem to prefer having the trim curves defined as polylines, but not always.

The last parameter, spline trim tolerance, specifies how accurately the conversion from polylines to splines should be done if the *Splines* alternative of the previous parameter is used.

-> ***VDA-FS** (identifier `vda`)*

Writes the contents of the target object (curve, curve set, surface, element, or root) into a VDAFS file. Version 2.0 of the VDAFS standard is supported.

Other details with using the command are similar to IGES output.

-> ***DXF (identifier dxf)***

Writes the content of the target object into a DXF file, as with IGES and VDA-FS output above. Version 11.0 of the standard is supported.

The DXF standard is mainly used to transfer data into 2D drafting systems. The standard can also define faceted surfaces and even parameteric surface descriptions, but not surface intersections.

The command asks for the following parameters:

- ***Include DXF header or not?*** Some systems, like AutoCAD, prefer the header to be included, other do no not require it.
- ***Separate arcs/lines?*** This option in effect performs the command `DIMENS`⇒ `Curve`: `Arcs/Lines` to all curves when writing them. This helps the receiving system interpret geometric features in the curves.
- ***How to store the faceted model?*** The alternatives are 3D faces, and polyface mesh. The latter takes less space to store, but the 3D faces seem to be more generally recognized by the receiving systems.
- ***How to separate the DXF layers?*** This determines what the receiving system stores in its different layers, to make the manipulation of the objects easier. The alternatives are to separate the objects (faceted models) as they are stored in DESKARTES inside different elements, or according to their material definitions.
- ***Store all to ENTITIES section?*** YES or NO, depending on what the receiving system likes better.

-> ***SGI Inventor (identifier iv)***

This outputs the selected faceted model in the SGI Inventor format. The shading colors assigned to the models will be transported, too. The option is available only with Silicon Graphics operating systems IRIX 5.x (or higher).

-> ***PostScript (identifier eps or aps)***

Writes a 2D object (curve, polyline, set of curves, text etc.) into a PostScript file.

Two slightly different PostScript flavors are available, and can be chosen as a parameter: Encapsulated PostScript (`eps`) and Adobe Illustrator PostScript (`aps`)

**Note**: The PostScript preview image is not included in the output file.

-> ***Point cloud***

Writes an ASCII point cloud file

***Rapid Prototyping***

The following data formats are specific to transporting data to Rapid Prototyping, as well as some Analysis and Manufacturing systems.

-> ***STL (identifier stl)***

Writes a faceted model or an element of those into the `.stl` file. DESKARTES asks, whether the data should be stored in ASCII or binary format.

Note that the model should have no holes, and its normals should point into the same direction, if it is to be manufactured. The same rules apply to the creating solid faceted models as with command `DIMENS`⇒ `Faceted`: `Volume`.

-> ***Slice (identifier slc, cli or sli)***

Writes a set (element) of planar 3D polylines into a slice file format, specific to different rapid prototyping systems.

## `File`: `Delete`

Deletes the selected file.

**Note**: The file is deleted permanently, the command cannot be undone.

## `File`: `Rename`

Changes the name of the file. The new name is asked as a parameter.

**Note**: This command requires you type in the file identifiers (prefix or suffix), too. It is good to keep the identifier of the file—if there is no special reason for not doing so—DESKARTES needs them for recognizing the file types.

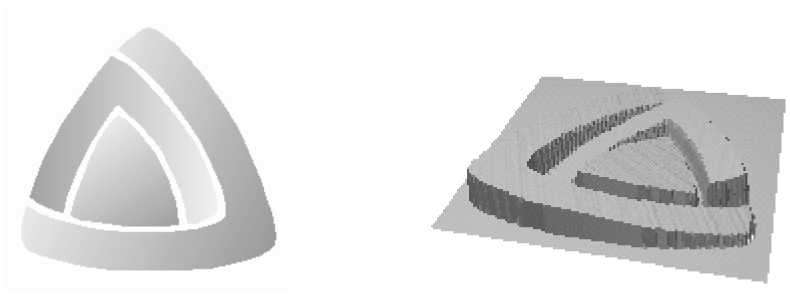
## `File`: `Convert`

This command has various different uses:

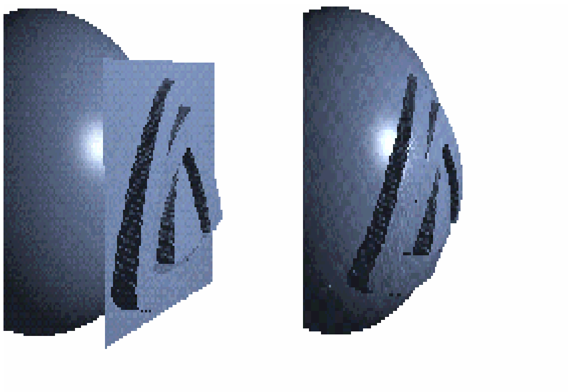
1. It ***converts an image file*** into the desired file format. The user is asked which format to convert to. For the time being, standard Sun Raster files, Rgb files, BMP, TIFF files and XWD files may be converted into each other's. Other conversion options may be requested from your DESKARTES dealer.
2. The "`greyscale`" option converts an colour `image` file into a similar file, but only containing ***grey scale colours***. This is required for applying ***bump mapped textures***; see `TEXTURE` menu.

3 The "`3Dbump`" option converts an `rgb` file into a "`bump_`" file, that contains a ***faceted representation for a 3D bump map of the image***.

The number of facets in the horizontal direction is asked as a parameter, default being according to the number of pixels in the image. The Z heights in the bump maps are taken from the pixel intensities in the image.



Once the `bump_` file is read into DeskArtes, it can be further scaled and transformed. One way to apply such 3D bumps is projecting them as decorations on other surfaces; see command `SURFACE⇒ Project to Surf`.



The “bump” object also contains a surface “box” around the bump object. The application of the surface box is as follows. When the user transforms and projects the bump object onto another surface, the surface box can be transformed together with the bump object. As result, the surface box will finally cut through the surface where the bump was placed, and it can be used to trim away the part beneath the bump.

4. If the selected item is a directory, which name ends with 'movie', the pnm images inside the directory will be combined into ***turntable animation***. This command only works for SGI IRIX 5.3 and higher.

   The command starts an SGI utility called movieconvert, which converts a series of images into a movie. Type in 1 in the From field and the amount of images into the To field. The names of the images are #.pnm. Press enter after having typed in the name.

   After the conversion is finished, you can run the movie by double clicking it from the operating system.

   Note that the default location for the movie file is the DESKARTES startup directory.

6. The command also works as the interface for the DESKARTES Rapid Tools commands. These are special routines for various Rapid Prototyping applications, described in a separate set of Manuals.

7.

## File: Rapid Process

This command contains functions specific to the DESKARTES Rapid Tools product, and it is explained accordingly in the Rapid Tools user manuals.

### File: (Un)Compress

***Compresses or decompresses a file***. Compressing makes the file much smaller, saving a lot of disk space. Compressed files have “.Z” or “.gz” appended to their names.

### File: Read Curves

This command reads a 3D polyline curve from an ***ASCII text file***, which may be produced with a ***3D scanner*** for example. The file should contain three coordinate values per line, and the file name should end with an “.asc” postfix.

### File: To Tape

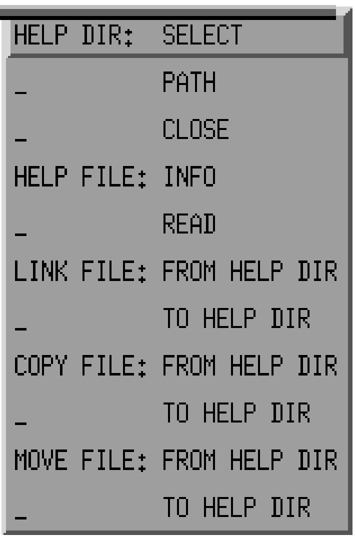
Writes the selected file onto magnetic tape, or some other data storage device. This command is availale only on the SGI platform.

### File: Print Pic

This command prints the selected image file onto the default (color or gray scale) printer. The printer is defined with the environment variable $DA_PRINTER.

## *Help Pane Commands*

### Help Dir: Select



***Selects a help directory*** to be shown in the rightmost column of the file manager. Help directories are required when you wish to move, copy or link files from one directory to another.

Unlike the other File Window commands, you first click the command button, and then the directory to be chosen.

### HelpDir: Path

This command allows you to ***select the help directory anywhere*** in the operating system directories.

You select the directory by typing in its complete operating system path name in a special help path window. The previously used help directories are listed there, too, and you may select them by pointing and clicking.

### Help Dir: Close

This command simply ***clears the help directory field***, if you don't want to have it shown.

## `Help File: Info`

Shows the ***information on the help directory***, similar to command `DATABASE`⇒ `Directory: Info`.

## `Help File: Read`

***Reads a file from the help directory***, just as command `DATABASE`⇒ `File: Read` for ordinary directories.

## `Link File: From Help Dir`

Creates a so-called ***symbolic link***. This makes a file chosen from the help directory assessable from your model directory, as well.

Linking files is usually better than copying files, as linked files only take up disk space for the original file. Note, however, that if you change the linked file, you will in fact affect the original one.

## `Link File: To Help Dir`

Creates a so-called ***symbolic link***. This makes a file chosen from your model directory assessable from the help directory, as well.

Linking files is usually better than copying files, as linked files only take up disk space for the original file. Note, however, that if you change the linked file, you will in fact affect the original one.

## `Copy File: From Help Dir`

Makes a ***copy of a file*** from the help directory into the model directory. The resulting copy has the same name as the original. If a file with this name exists, the user will be asked to confirm that the old file can be replaced.

## `Copy File: To Help Dir`

Makes a ***copy of a file*** from the model directory into the help directory. The resulting copy has the same name as the original. If a file with this name exists, the user will be asked to confirm that the old file can be replaced.

## `Move File: From Help Dir`

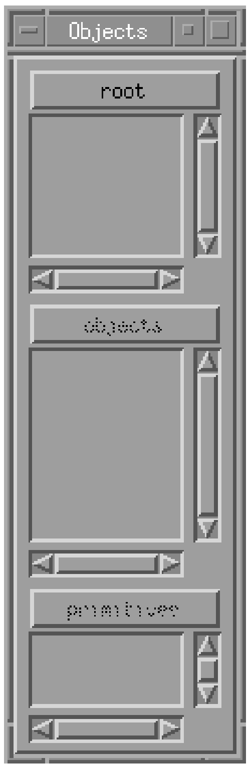
***Moves the selected file*** from the help directory into the model directory. If a file with this name exists, the user will be asked to confirm that the old file can be replaced.

## `Move File: To Help Dir`

***Moves the selected file*** from the model directory into the help directory. If a file with this name exists, the user will be asked to confirm that the old file can be replaced.

## Object Window

The ***Object Window,*** which is situated on the left side of the DeskArtes user interface, contains a pop-up menu with general commands that deal with objects.



### Choosing the Target Object

Pointing the cursor at the desired object name and clicking the left mouse button ***chooses the target object***.

The possible ***contents*** of the target object are visible in the lower window fields. For example, if an element has been chosen for target object, the lower fields display the objects that make up the element.

If there are more objects than fit in the window field, you may ***scan*** the list of objects with the mouse. Slide the right hand scroll bar of the window field with the middle mouse button.

### Displaying the Target Object

If the display mode is set to `SETTING`⇒ `Draw`: `Auto`, DeskArtes ***automatically displays*** the target object when you select it. If the new target object is of different type than the previous one, e.g. surface vs. projection curve, the old screen contents are erased.

Selecting the target object again ("double selection") will ***fit*** it nicely in the middle of the screen.

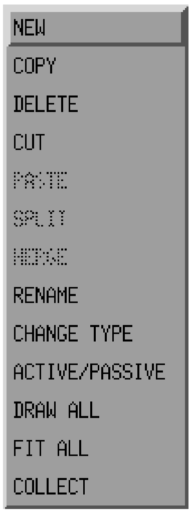
As an exception (to avoid undesired erasing), nothing changes on the screen if an ***element*** is selected. However, selecting the target element ***again*** will draw all the surfaces under it. Selecting the same element a ***third time*** will fit its surfaces in the middle of the screen.

Sometimes you may wish to avoid automatic displaying, for instance to view 2D curves simultaneously with 3D surfaces. The ***display mode*** `Settings`⇒ `Draw`: `Menu` disables automatic displaying. How the objects are displayed and erased is then controlled entirely with the `DISPLAY` menu commands.

A third alternative is to set displaying entirely off, with `Settings`⇒ `Draw`: `None`. This is useful when displaying has no significance and would only slow down the system, e.g. when executing command series.

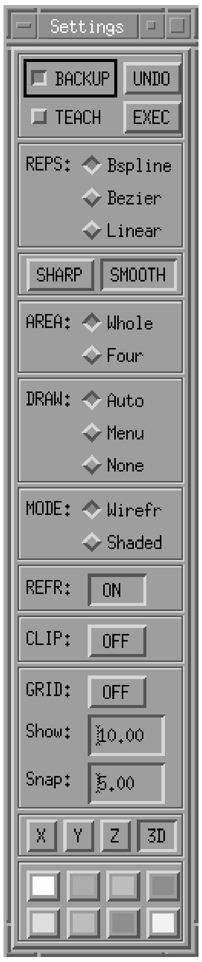
### Object Window Pop-Up Menu

The Object Window pop-up menu commands are identical to those in the `OBJECT` and `DISPLAY` menus. They are included in the Object Window for the greatest ease of use, to be found near the object lists.



## *Settings Window*

The Settings Window contains commands that are related to the way objects are input and displayed. The Settings Window can be displayed and hidden by clicking the right mouse button within the graphics window.



### BACKUP

After each command that affects the geometry of the model (not, for example, display or file commands), DESKARTES makes a so-called ***backup file*** (called `safe_system`) which contains the modeling situation preceding the command. This is used for ***undoing*** commands with the `UNDO` command.

By clicking at `BACKUP`, you may prevent DESKARTES from making backups. You might want to do this with very large models, as it might take too long to write the backup file after each command. To allow making backups, click at `BACKUP` again.

However, there's an alternative to make the backups faster. Normally the backup files are stored in the current model directory. If the model directory is located somewhere else in the network than the workstation from which DESKARTES is used, it will be much faster to store the files on the machine's local disk. This behavior can be controlled with command `SYSTEM`⇒ `Backup Control`.

### UNDO

This command ***cancels*** the last command, which has changed the geometry of the model.

If `BACKUP` has been disabled, `UNDO` reverts to the last backup file available.

If the commands is executed immediately after launching DESKARTES and selecting the last used directory, UNDO r***eads in the latest modeling situation*** (backup file) within the directory. This means that model geometry will not be lost in case of a program crash or power failure.

See command `SYSTEM`⇒ `Backup Control` on how to optimize the speed of `UNDO`ing.

**Note**: File Window or visualization commands may not be canceled with `UNDO`.

### TEACH

If the same modeling commands need to be performed repeatedly, it is useful to make a ***command series*** of the commands, i.e. teach a lesson to the system.

The command `TEACH` writes the subsequent commands and editing functions into the specified command file, until the `TEACH` button is clicked again. The lesson may then be automatically ***repeated*** with the `EXEC` button (see below).

**Note**: During the time the command series is being recorded, you may not use the Object Window or the File Window commands. Target object selection must therefore be performed with the `SELECT` menu commands.

## EXEC

This command ***executes*** a command series, created with `TEACH`, a given number of times.

Before execution starts, you are asked whether new command parameters will be requested at every step of the execution, or whether the ones in the lesson will be used.

You can also determine whether the commands should be executed automatically, or if you want to confirm each step before executing.

Finally, there's an option to have a brief delay between the commands, to make the execution look nicer. If you want to run the command series as fast as possible, just use zero delay.

**Note**: As you give `EXEC`, the commands are executed just as they were taught, with no intelligence added. In particular, be sure you have the right target object selected before you start, the program won't be able to decide it for you. Note also that graphic picking may be dangerous under `TEACH`: it could pick wrong objects if the viewing window or the objects change with `EXEC`.

## REPS

The buttons `B-spline`, `Bézier` and `Linear` next to the text `REPS` determine the ***input representation of curves*** produced with the `CURVE` menu commands.

For example, if the type is set to `B-spline`, the command `CURVE`⇒ `Design`: `Input` assumes that the points entered are control points for a B-spline curve.

The default value for the input type is `B-spline`, which generally applies well to any kind of modeling work.

## SHARP/SMOOTH

The `SHARP` and `SMOOTH` buttons control the ***input curve shape***. Depending on which button is pressed down, a curve adopts a sharp or a smooth shape with command `CURVE`⇒ `Design`: `Input`.

If `Linear` has been chosen for representation, the `SHARP/SMOOTH` buttons have no significance — polygons are always "sharp".

## AREA

The buttons next to the `AREA` text allow you to change the ***display area*** that DESKARTES uses to view the objects.

Selecting `AREA`: `Whole` makes the ***entire graphics window*** the display area.

The `AREA`: `Four` option divides the graphics window into ***four quarters***, called viewports. This works with 3D objects only, 2D objects are not displayesd at all when the four quarters are chosen. All 3D objects are displayed simultaneously in the X, Y, Z and 3D view quarters of the display. A title text is displayed in each viewport quarter, showing its viewing direction.

The four views mode is especially useful with ***3D transformations***. Each quarter can be separately zoomed and panned with the corresponding DISPLAY menu commands, or used for graphical picking of objects. Furthermore, the `Area`: `Four` option is fully integrated with scene and surface editing, as well as 3D curve input and editing. All editing changes are shown in the four viewports simultaneously.

## DRAW

Change the ***display mode*** with the three buttons. The possible settings are `Auto`, `Menu`, and `None`. The default setting is `Auto`.

The meanings of the display modes are:

- **Auto** - Objects are displayed automatically as they are selected.

  Normally, the current display contents are erased when a new type of object is selected. However, the screen is not erased when section curves are created/chosen, or if a projection set is created from the current viewing direction (eye set to `X`, `Y` or `Z`).

- **Menu** - Objects are not drawn or erased automatically when selected, but only when new objects are created or deleted, or when specifically using the DISPLAY menu commands.

- **None** - Nothing is ever displayed, except when specifically using the DISPLAY menu commands. Use this option when displaying takes too long and is not needed, e.g. when deleting several objects in a row, or when executing long command series.

## REFR

Generally all objects on the screen are redrawn when a menu or some other window disappears from the screen. With large models this feature may be too time consuming. Setting the ***refresh mode*** to OFF disables this type of display refreshing.

## CLIP

This toggle button allows you to temporarily set ***clipping*** on and off, when clip planes have first been defined with command `DISPLAY`⇒ `Define Clips`.

## MODE

With the GL version you may switch between ***wireframe*** drawing to ***shaded*** mode at any time. The mode is chosen with the $\boxed{\texttt{Wirefr}}$ and $\boxed{\texttt{Shaded}}$ buttons. Note: When the object is shaded for the first time it will take longer, but after that the objects are kept in the GL display lists and will be rendered faster.

## GRID

The ***grid*** makes it possible to snap points to exact coordinate values, and helps visualize the dimensions of an object.

The grid affects the points automatically only with the `CURVE`⇒ `Design`: `Input` command. In curve editing, the grid is used only when requested with the editing function `g`.

The command `TRANSF`⇒ `Object`: `Move` applies the grid, if the right mouse button is pressed during the command. All other commands use the grid purely as a visual aid.

### *Grid*

The `Grid` button switches the grid ***on and off***, when clicked.

For three-dimensional objects, the grid is shown as a “floor” on the xy plane.

### *Show*

The ***density*** of the grid shown is displayed next to the `Show` text. Change the density by clicking at it and typing in the density.

### *Snap*

The "***magnet effect***" of the grid is controlled with the `Snap` value. Points will be snapped at the integer multiple values of the `Snap` value, when the grid is applied.

## *Eye Directions*

The ***eye direction buttons*** allow you to quickly set the eye point to one of the coordinate axes, and back to three-dimensional space. The buttons are labeled `X`, `Y`, `Z`, and `3D`. Each button sets the eye point to the corresponding view direction.

If you click at the same button twice, the viewing direction is changed to the opposite side, e.g., to negative X axis direction with the second clicking of the X button.

Whether the view direction is set to one of the axis, or to 3D, has a great impact on how 3D transformations work. See the `TRANSF` commands for further details.

## *Vector Color* `Buttons`

The eight ***vector color buttons*** in the bottom of the Settings Window change the wire frame display color of the target object.

What the colors actually are may be chosen interactively with the command `SYSTEM`⇒ `Set Colors`.

# 5 *EDIT MODE EXPLANATION*

The edit modes perform various ***functions*** that generally affect points, as opposed to objects in the command level. The ***menu commands are not available in editing mode***. However, some of the most important general commands have been made into editing functions. Such commands are, for example zooming (and panning), screen refreshing and undoing.

The functions in the edit modes are selected by clicking at their ***icons*** with the left mouse button. If you click with the right button instead, a brief explanation of the function is displayed in the message lines.

Most functions have a ***shortcut***, or a function key, assigned to them. They are typically the first letters of the function names. A function key executes the function just as if the corresponding icon had been selected, only that they are faster to use once learned.

Note the difference between lowercase and UPPERCASE function keys. They will have different meanings. If a shortcut key does appear to work, check that you haven't accidentally pressed the `CapsLock` key.

## *2D CURVE INPUT*

### *new point **(middle)***

This function adds a point to the polygon. The point is positioned by clicking with the left mouse button. The point may be moved and is shown attached to the previous point by a 'rubber band' until the left mouse button is clicked again.

### *delete point **(right)***

This function removes the last point drawn in the polygon.

### ***n**umerical coordinates*

This function adds a point by typing in its coordinates. The points are entered as horizontal and vertical coordinates relative to the origin. Points to the right and above the origin should be entered as positive values. Points to the left and below the origin should be entered as negative values.

### *rel**a**tive coordinates*

This function adds a point by typing in its coordinates. The points are entered as horizontal and vertical coordinates relative to the last point. Points to the right or above the last point should be entered as positive values. Points to the left or below the last point should be entered as negative values.

### ***w**rite curve*

This function finishes entering the points and saves the curve. If the last point in the polygon is near the first point, the curve will be closed (the last point will be exactly connected to the first one). Otherwise, the polygon will be open.

### quit curve

This function finishes entering the points but does not save the curve. All points input will be discarded. The systems state after the edit command will be unchanged. You are asked to confirm command.

## 3D CURVE INPUT

To create a 3D curve, you must have a 3D curve set (or a 3D curve) selected. To create a 3D set, you may select the `3D` option when creating a projection set with `OBJECT`⇒ `New`. The command `CURVE`⇒ `Design`: `Input` will then automatically enter the curves in 3D.

In 3D input, the screen is always split into four parts. The points may be entered and edited in the X, Y and Z quarters, while the 3D view is just for displaying the curve.

The first two points of a curve are located by clicking the left mouse button in one view quarter. This defines two of the point's coordinates. You may then click in another view quarter to show the third coordinate position. If the same quarter is clicked twice, the third coordinate component is set equal to the previous point, or zero for the first point.

It is often more convenient to enter the 3D coordinates numerically than interacting with the different quarters. To do this, use the `n` function. The points are entered as coordinates relative to the origin.

After the first two points have been input, the program enters the 3D curve-editing mode where further points may be added and edited.

## B-SPLINE CURVE AND POLYGON EDITING

This command allows you to interactively change the shape of a B-spline curve or a polyline, using functions directed at the control points.

The functions have been divided into two toolboxes: local and global functions. Each toolbox may be shown and hidden from the top two icons of the general icons. Local functions typically affect only the curve part near the chosen, active control point, whereas global functions change a larger part of the curve or affect display events.

When working with B-splines, it is often the case that the projection set contains two projection curves for building the surface. Thus, it is allowed to edit two curves at a time by selecting their curve set as the target object.

The same functions are used to edit polygons and B-splines, though some do not have significance for polygons. Bézier curves have slightly different functions, which are discussed later.

## *General Functions*

### `local icons`

This icon displays another window with icons for commands intended to control the shape of the edited curve by manipulating the control points of the curve. The remains visible until you click on this icon again.

### `global icons`

This icon displays another window with icons for commands intended to control the shape of the edited curve by manipulating the entire curve. It also includes display control commands. The new icon window remains visible until you click on this icon again.

### `**u**ndo`

Any action that changes the curve may be canceled by clicking this icon immediately after the erroneous action. Only the last one action can be canceled by this command. If you have made many incorrect edits they can be cancelled by exiting the edit mode by pressing **q**. This will cancels all changes made in this edit session.

### `curve dimension`

This function changes temporarily into curve dimensioning mode (see the command `DIMENS`⇒ `Curve`: `Dimension`).

### `**w**rite curve`

This function finishes editing the curve and saves the edits.

### `**q**uit curve`

This function finishes editing the curve but does not save the curve. All edits made will be discarded. The systems state after the edit command will be unchanged. You are asked to confirm the command.

## *Local Functions*

### `next point **+**`

This function makes the point following the current point active. This is useful when control points are located close together to ‘walk’ along the control points. B-spline curves can contain multiple points.

### `previous point **-**`

This function makes the point preceding the current point active. This is useful when control points are located close together to ‘walk’ along the control points. B-spline curves can contain multiple points.

### `select point **(left)**`

This function makes the point closest to the cursor the active point.

## multiple select **(right)**

With this function, it is possible to select multiple points for editing them simultaneously.

The function selects a point, called the “last point”. The last point is highlighted by a double box drawn around the point. All points between the active point and the right point are affected by certain editing functions including `(left)` (graphical move), `x` (axial move), `n` (numerical move), and `p` (projecting points).

Removing the last point selection is achieved by selecting any point with left mouse button again.

## move point **(middle)**

When a point has been made active, clicking the middle mouse button allows graphical moving. The point follows the cursor movement, and the curve interactively changes shape according to the position of the point. When the middle mouse button is clicked again, the function is ended and the curve remains in its new form.

When editing secondary projections, the moves are automatically restricted to the primary projection direction. See the explanations with `BUILD`⇒ `Create`: `Secondary Projection`.

The function may also be used for moving multiple points, by using the ‘last point’ selection technique with the right mouse button.

## a**x**ial move

When a point has been made active, it may be graphically moved in an axis direction by this command. The first movement made with the mouse after executing the command and clicking the left mouse button chooses the axis direction (horizontal or vertical). The point follows the cursor movement, and the curve interactively changes shape according to the position of the point. When the left mouse button is clicked again, the function is ended and the curve remains in its new form.

The function may also be used for moving multiple points, by using the ‘last point’ selection technique with the right mouse button.

## **m**irror point

The active point is moved opposite another reference point. The mirror axis passes through the object's fix point. The mirroring direction (horizontal or vertical) is chosen to be the nearest to the axis. If this is not the required direction, you may have to move the point so that it is nearly opposite first.

The reference point (against which the active point is going to be moved opposite) is pointed to with the mouse after choosing the function.

As a useful special case: if you select the active point itself as the reference point, it moves directly onto the nearest axis through the fix point. This is useful for curves that are to be made into a surface by rotating around the axis, as it avoids creating a hole or an overlap on the axis line.

## **l**evel point

This function aligns a point with another point, parallel to one of the axes. The aligning direction (horizontal or vertical) is chosen to be the nearest to the axis. If this is not the required direction, you may have to move the point so that it is nearly aligned first.

The reference point (against which the active point is going to be aligned) is pointed to with the mouse after choosing the function.

As a useful special case: if you align a point twice with another point, it is aligned in both directions, set to the same location as the other point. This is very useful when editing two curves and the end points of the curve need to be made coincident.

## **p**roject point

Moves the active point along a line between the points preceding and following it.

The function may also be used for moving multiple points, by using the ‘last point’ selection technique with the right mouse button. In this case all the points between the active and the last point are projected onto the line between the first and last selected points.

When editing secondary projections, the multiple point projection is automatically restricted to one of the axial directions, so that only one of the control points' coordinates change.

## move in/**o**ut

Moves the active point continuously towards or away from the object's fix point.

## **n**umerical value

This function changes the coordinates of the active point by entering numeric parameters for the coordinates of the point.

The function may also be used for moving multiple points, applying the “last” point selection. In this case, the amount of movement in the horizontal and vertical directions should, not the absolute coordinate values.

## **g**rid move

This function graphically moves the active point, keeping it at the grid nearest grid snap points set with `Settings`⇒ `Grid`: `Snap`.

## **i**nsert point

This function adds a new point next to the active point, at a location shown with the mouse. The point is moved continuously until the left mouse button is clicked again.

The point is added to the nearest side of the active point. If appropriate, the point is added to the end of a curve.

## **d**elete point

This function deletes the selected point from the control polygon.

## **c**opy point

The function copies (duplicates) a point. The copy is placed in exactly the same location as the active point.

If a single point is copied, a double point will result. The B-spline curve will form a rounded corner near the double point. Double points are displayed in yellow.

If a double point is copied again, a triple point will result. The B-spline curve will go through the point, where it will have a sharp corner. Triple points are displayed in red.

There is no advantage to copying a triple point. It has no effect on the geometry.

Many functions treat double and triple points like single ones; e.g., moving the points with the middle mouse button.

### ***b**eta points*

This function adds two points between the active point and the points immediately adjacent to it. This has the effect of creating a gently rounded corner. It may be edited as desired but if the function **p** is used the size of the rounding can be adjusted by moving the points near to the active point.

### ***r**ounding*

This function creates a circular rounding or fillet at the active point. A parameter window is displayed requesting the radius and the number of points to be used to create the rounding.

### ***U**ndo rounding*

The quickest way of removing a rounding which has just been created is with the **u** function, undo. If the rounding has been created some time earlier, this function can be used to cancel it. After executing the function, indicate the rounding to be removed by pointing with the mouse at the first and last point. These points and all the points in between are deleted and the original corner point is added.

### *two point arc**s***

This function creates an arc through two points using the chosen number of control points. The user is prompted to locate the two end points and then indicate the desired approximate center point of the arc with the mouse. A parameter window is then displayed requesting the arc radius numerically.

### ***t**hree point arc*

This function creates an arc through three user-selected points, using a given number of control points. The user is prompted to locate the three points with the mouse. Since three points exactly define a radius no further user intervention is required.

As a special case, if two of the selected points are the same, the arc will continue in the tangent direction from the doubly chosen point.

## *Global Functions*

### *open curve - **B***

This function opens a closed curve.

### ***C**lose curve*

This function closes an open curve.

### ***F**irst point*

This function makes the active point the first point of a closed curve.

It is often required to set the first point of closed section curves for the command `BUILD`⇒ `Create`: `Surface`. A more important use for this function is temporarily moving the first point for certain global function. In particular, the functions **s**, **t**, **T**, and move multiple point’s functions may affect the wrong half of the curve. Moving the control point to a different part of the curve can cure this problem.

## ***T**wist points*

This function rotates several control points simultaneously around a given point. Select the first and last points to be rotated with the left mouse button, then locate the position of the center point of rotation, which does not have to be on a control point. The rotation angle is entered numerically.

## ***S**harpen points*

This function duplicates or triples all the control points of a curve, like the function **c** for individual points, or makes all the control points single.

With double points, the B-spline curve will form a rounded corner near the double point. Double points are displayed in yellow.

With triple points, the B-spline curve will go through the point, where it will have a sharp corner. Triple points are displayed in red.

Many functions treat double and triple points like single ones; *e.g.*, moving the points with the middle mouse button.

## ***R**ound all corners*

This function replaces all double and triple control points with a circular arc rounding in the same way as function **r** does. A parameter window is displayed requesting the radius and the number of points to be used to create the rounding.

## **A**dd points

This function changes the number of control points used to define a curve. This function may be used to define projection curves, for the command BUILD⇒ Create: Surface, with an equal number of points although other methods of achieving this exist.

**Note:** changing the number of control points usually changes the shape of the curve.

## **D**ouble number of points

This function doubles the number of control points used to define a curve.

**Note:** Unlike the previous function **A**, this function does ***not*** change the shape of the curve.

## **z**oom and pan

This command alters the scale of the viewing window. The scale of the window can be set either graphically, or relative to the current scale.

- To zoom in freely by a view box, press and hold the left mouse button with the cursor at the center of where you want to zoom. Move the mouse up or down and a rectangle showing the new window is displayed. The window is fixed by releasing the mouse button.
- The view will change continuously if you press the middle mouse button whilst moving the mouse. However, in four views and the edit modes, zooming always works in box mode as above.
- By clicking the right button, the window becomes twice as big as it currently is — the objects on the screen are shown smaller.
- By clicking on any key on the keyboard, you will be able to pan instead of zooming. Pan the view by clicking with the left mouse button at any point on the screen, then move the cursor

to the new location of the first point, and release the button. This feature is particularly useful when zooming in the various edit modes, where only the zoom function is available.

### Zoom back

This function returns the viewing window to what it was when editing was begun.

### Refresh curve

Erases the screen and redraws the contents. This mat be needed to clean up the display after graphical moving

The function has a useful side effect. If you give the function twice in a row, the second execution displays the last TextPaint image contents. You can then use this as template for curve editing, matching the shape of the curve to the shape in the picture.

### Knots

This function displays and hides the knot points. The knot points determine the positions of the cross-sections for the command `BUILD`⇒ `Create`: `Surface`. They are associated with `control points and` so will be located on the curve quite close to a control point.

### Extents

This function changes the curve back to the size it had when editing was begun. This may be useful when editing cross-sections for building a surface, to keep them in the scales forced by the projection curves.

### Volume

This function computes the volume of a surface that would be created by rotating the edited curve around the vertical axis. If the curve does not touch the axis then the volume is calculated as the end of the curve was projected onto the axis.

## *BÉZIER CURVE EDITING*

This command allows you to interactively change the shape of a Bézier curve.

The functions have been divided into two toolboxes: local and global functions. Each toolbox may be shown and hidden from the top two icons of the general icons. Local functions typically affect only the curve part near the chosen, active control point, whereas global functions change a larger part of the curve or affect display events.

When the command starts, DESKARTES displays the Bézier curve with its knot points. The knot points are marked as small boxes on the curves. One of the knot points is selected as active.

At both sides of the selected knot point, two crosses are displayed. They are the so-called handles, which allow changing the curve shape without moving the knots. The handles are located in the curve’s tangent direction relative to the corresponding knot point.

The curve may be smooth at a knot point, so that the curve continues smoothly from one curve segment to another. In this case, both the knot point and its handles lie in a line. Otherwise, the curve is sharp, which means that there is a corner at the knot.

## *General Functions*

### `local icons`

This icon displays another second window containing icons intended to control the shape of the edited curve by manipulating the control points of the curve. The new window remains visible until you click on this icon again.

### `global icons`

This icon displays another window containing icons to control the shape of the edited curve by manipulating the entire curve. It also includes display control commands. The new window remains visible until you click on this icon again.

### `**U**ndo`

Any action that changes the curve may be canceled by clicking this icon immediately after the erroneous action. Only the last one action can be canceled by this command. If you have made many incorrect edits they can be cancelled exiting the edit mode by pressing **`q`**. This cancels all changes made in this edit session.

### `curve dimension`

This function changes temporarily into curve dimensioning mode (like the command `DIMENS`⇒ `Curve`: `Dimension`).

### `**w**rite curve`

This function finishes editing the curve and saves the edits.

### `**q**uit curve`

This function finishes editing the curve but does not save the curve. All edits made will be discarded. The systems state after the edit command will be unchanged. You are asked to confirm the command.

## *Local Functions*

### *`next point`* **`+`**

This function makes the point or handle following the current point active. This is useful when control points are located close together to ‘walk’ along the control points.

### *`previous point`* **`-`**

This function makes the point or handle preceding the current point active. This is useful when control points are located close together to ‘walk’ along the control points.

### `select point` **`(left)`**

This function selects the closest knot point or handle to the cursor.

## multiple select **(right)**

With this function, it is possible to select multiple points for editing them simultaneously.

The function selects a point, called the “last point”. The last point is highlighted by a double box drawn around the point. All points between the active point and the right point are affected by certain editing functions including **`(left)`** (graphical move), **`x`** (axial move), **`n`** (numerical move), and **`p`** (projecting points).

Removing the last point selection is achieved by selecting any point with left mouse button again.

## move point **(middle)**

When a point has been made active, clicking the middle mouse button allows graphical moving. The point follows the cursor movement, and the curve interactively changes shape according to the position of the point. When the middle mouse button is clicked again, the function is ended and the curve remains in its new form.

If the curve is smooth at the active point, the function moves both the handle points in such a way that the curve remains smooth. If you wish to move just one handle tangent continuously at a time, use function **`p`** instead.

If the curve is not tangent continuous at the knot point, only the active point moves.

The function may also be used for moving multiple points, by using the ‘last point’ selection technique with the right mouse button.

## a**x**ial move

When a point has been made active, it may be graphically moved in an axis direction by this command. The first movement made with the mouse after executing the command and clicking the left mouse button chooses the axis direction (horizontal or vertical). The point follows the cursor movement, and the curve interactively changes shape according to the position of the point. When the left mouse button is clicked again, the function is ended and the curve remains in its new form.

The function may also be used for moving multiple points, by using the ‘last point’ selection technique with the right mouse button.

## **m**irror point

The active point is moved opposite another reference point. The mirror axis passes through the object's fix point. The mirroring direction (horizontal or vertical) is chosen to be the nearest to the axis. If this is not the required direction, you may have to move the point so that it is nearly opposite first.

The reference point (against which the active point is going to be moved opposite) is pointed to with the mouse after choosing the function.

As a useful special case: if you select the active point itself as the reference point, it moves directly on the nearest axis through the fix point. This is useful for curves that are to be made into a surface by rotating around the axis, as it avoids creating a hole or an overlap on the axis line.

## **l**evel point

This function aligns a point with another point, parallel to one of the axes. The aligning direction (horizontal or vertical) is chosen to be the nearest to the axis. If this is not the required direction, you may have to move the point so that it is nearly aligned first.

The reference point (against which the active point is going to be aligned) is pointed to with the mouse after choosing the function.

As a useful special case: if you align a point twice with another point, it is aligned in both directions, set to the same location as the other point. This is very useful when editing two curves and the end points of the curve need to be made coincident.

## *move point (**middle**)*

When a point has been made active, it may be gripped by clicking the middle mouse button. The point follows the movements of the cursor. The function is ended when the middle mouse button is clicked again.

If the curve is smooth at the active knot, the function moves both the handle points in such a way that the curve remains smooth. If you wish to move just one handle and keep the curve smooth, use the function **p** instead.

If the curve is not smooth at the knot point, only the active point (knot or handle) moves.

## **p**roject point

Moves the active handle point along the tangent line of the nearest knot point.

The function may also be used for moving multiple points, by using the ‘last point’ selection technique with the right mouse button. In this case, all the points between the active and the last point are projected onto the line between the first and last selected points.

## **a**ngle

This function relocates the active control point so that the curve will become sharp at the nearest knot point. The angle of the sharp corner is requested in a parameter window. The angle should be entered in degrees.

## **n**umerical value

This function changes the coordinates of the active point by entering numeric parameters for the coordinates of the point.

If the curve is smooth at the active knot, the function moves both the handle points in such a way that the tangent continuity is preserved. If you wish to move just one handle and keep the curve smooth, use the function **p** instead.

If the curve is not tangent continuous at the knot point, only the active point (knot or handle) moves.

The function may also be used for moving multiple points, applying the “last” point selection. In this case, the amount of movement in the horizontal and vertical directions should, not the absolute coordinate values.

## **g**rid move

This function graphically moves the active point, keeping it at the grid nearest grid snap points set with `Settings`⇒ `Grid`: `Snap`.

## **i**nsert point

This function adds a new point (and its handles) at the nearest point to the location shown with the mouse.

The shape of the curve does not change and tangent continuity is preserved. If appropriate, the new point is placed at one end of the curve.

### **d**elete point

This function deletes the active selected point (and its handles) from the curve.

### **c**orner

This function is used to create sharp corners on a curve.

After pressing a mouse button, the active point follows the cursor and all other points remain where they are. Stop moving the point by clicking the mouse button again.

### smooth - **b**

Moves the active point so that the curve becomes smooth at the closest knot point.

### **r**ounding

This function creates a circular rounding or fillet at the active point. A parameter window is displayed requesting the radius and the number of points to be used to create the rounding.

### **U**ndo rounding

The quickest way of removing a rounding which has just been created is with the **u** function, undo. If the rounding has been created some time earlier, this function can be used to cancel it. After executing the function, indicate the rounding to be removed by pointing with the mouse at the first and last point. These points and all the points in between are deleted and the original corner point is added.

### two point arc**s**

This function creates an arc through two points using the chosen number of control points. The user is prompted to locate the two end points and then indicate the desired approximate center point of the arc with the mouse. A parameter window is then displayed requesting the arc radius numerically.

### **t**hree point arc

This function creates an arc through three user-selected points, using a given number of control points. The user is prompted to locate the three points with the mouse. Since three points exactly define a radius no further user intervention is required.

As a special case, if two of the selected points are the same, the arc will continue in the tangent direction from the doubly chosen point.

## *Global Functions*

### open curve - **B**

This function opens a closed curve.

### **C**lose curve

This function closes an open curve.

## **F**irst point

This function makes the active point the first point of a closed curve.

It is often required to set the first point of closed section curves for the command `BUILD`⇒ `Create`: `Surface`. A more important use for this function is temporarily moving the first point for certain global function. In particular, the functions **`s`**, **`t`**, **`T`**, and move multiple point's functions may affect the wrong half of the curve. Moving the control point to a different part of the curve can cure this problem.

## **T**wist points

This function rotates several control points simultaneously around a given point. Select the first and last points to be rotated with the left mouse button, then locate the position of the center point of rotation, which does not have to be on a control point. The rotation angle is entered numerically.

## ***R**ound all corners*

This function replaces all sharp corner knot points with a circular arc rounding.

## ***S**harpen all corners*

The curve will take the shape of a polygon that goes through the knot points.

## *smooth **A**ll corners*

All corner points become smooth.

## **D**ouble number of points

This function doubles the number of points used to define a curve. This is often required to improve the accuracy for 3D Building.

## **z**oom and pan

This command alters the scale of the viewing window. The scale of the window can be set either graphically, or relative to the current scale.

- To zoom in freely by a view box, press and hold the left mouse button with the cursor at the center of where you want to zoom. Move the mouse up or down and a rectangle showing the new window is displayed. The window is fixed by releasing the mouse button.
- The view will change continuously if you press the middle mouse button whilst moving the mouse. However, in four views and the edit modes, zooming always works in box mode as above.
- By clicking the right button, the window becomes twice as big as it currently is — the objects on the screen are shown smaller.
- By clicking on any key on the keyboard, you will be able to pan instead of zooming. Pan the view by clicking with the left mouse button at any point on the screen, then move the cursor to the new location of the first point, and release the button. This feature is particularly useful when zooming in the various edit modes, where only the zoom function is available.

## **Z**oom back

This function returns the viewing window to what it was when editing was begun.

### Re**f**resh curve

Erases the screen and redraws the contents. This mat be needed to clean up the display after graphical moving

The function has a useful side effect. If you give the function twice in a row, the second execution displays the last TextPaint image contents. You can then use this as template for curve editing, matching the shape of the curve to the shape in the picture.

### **K**nots

This function displays and hides all the handles and knot points.

This is especially useful to check that neighboring handles are not going past each other, which might result into fluctuating behavior on the curve.

### **E**xtents

This function changes the curve back to the size it had when editing was begun. This may be useful when editing cross-sections for building a surface, to keep them in the scales forced by the projection curves.

### **V**olume

This function computes the volume of a surface that would be created by rotating the edited curve around the vertical axis. If the curve does not touch the axis then the volume is calculated as the end of the curve was projected onto the axis.

## *3D CURVE EDITING*

If you have selected a 3D curve as the target object, the command `CURVE`⇒ `Design`: `Edit` will enter the 3D curve edit mode. You will also enter this mode when inputting a 3D curve after inputting the first point.

In 3D editing, the screen is always split into four quarters. The points may be edited in the x, y and z quarters, while the 3D view is just for displaying the curve.

3D editing applies to Bézier curves and polylines only, it can not be used for B-spline curves. The 3D curve editing functions are similar to the 2D Bézier curve case.

The general differences compared to the 2D Bézier editing functions are:

- The local functions apply to the chosen view quarter, e.g., vertical point move with `x` moves the point vertically in the chosen view.
- Global functions act independently of the view. Also, the corner fillet function `r` affects applies in 3D, independently of the chosen view quarter.
- Last point selection is not available, instead, functions `N` and `P` perform the multiple point functionality of `n` and `p`.
- The functions `x`, `m` and `l` ask for the direction as a separate parameter, as it is difficult to determine the directions automatically in 3D views.
- The function `L` is special to the 3D curve editing functions. It ends the curve at the chosen knot point, while the other half of the curve is added to the object list. Its particular use is

for creating two 3D projection curves out of one closed (trim) curve for `BUILD`⇒ `Create`: `Surface`.

## *B-SPLINE SURFACE EDITING*

This command edits B-spline surfaces with local changes.

The editing is accomplished either using the entire display area from the eye point, or in four quarters of the screen displaying the object from four different views.

DeskArtes inquires whether you wish to edit the surface using the control mesh or the surface points. If surface points are chosen, you may directly change the locations of the points on the surface. The control mesh, on the other hand, affects the surface indirectly the same way a control polygon affects a curve.

The surface points are situated at the crossings of the lengthwise and crosswise curves that define the surface. A surface point is displayed only if it is active, whereas the control mesh is always displayed in its entirety. In the following section, “point” refers to either a point on the control mesh or a surface point, depending on the chosen mode.

In practice, it is recommended that all changes be made editing surface curves (the **`c`** and **`l`** functions): interaction in plane is always simpler than in three-dimensional space.

### `next cross point` **`+`**

This function makes the point following in the cross-wise direction the current active point active.

### `previous cross point` **`+`**

This function makes the point preceding in the cross-wise direction the current active point active.

### `next length point` **`>`**

This function makes the point following in the length -wise direction the current active point active.

### `previous length point` **`<`**

This function makes the point preceding in the length-wise direction the current active point active.

### `select point` **`(left)`**

Clicking the left mouse button activates the point nearest to the cursor.

### `move horizontal` **`(middle)`**

The point is moved continuously, so that the x and y coordinates change while z remains constant.

If four display areas are used, the point is moved in the plane where the point was selected.

### `move vertical` **`(right)`**

The point is moved continuously, so that the z coordinate changes while x and y remain constant.

If four display areas are used, the point is moved in the plane where the point was selected.

## **n**umerical values

This function changes the coordinates of the active point by entering numeric parameters for the coordinates of the point.

## **a**dd to coordinate values

This function changes the coordinates of the active point by adding values entered as numeric parameters to the coordinates of the point. This has the effect of moving the point by the amount entered.

## **c**ross-wise editing

This function picks a cross-wise curve of the surface for curve editing on a plane. The chosen curve is the one that goes through the active point, or, if the control mesh is being edited, the nearest cross-wise section.

Usually, the cross-wise curves of a surface are three-dimensional curves and have to be projected into a plane for editing. DESKARTES asks for the projection direction. The default choice is an arbitrarily oriented plane that would deform the curve least.

The alternative is to project the curve in one of the coordinate axis directions. If the chosen direction is z, the curve will be edited within the xy-plane, etc. The values of the curve in the projection direction will be returned to the curve after editing is complete.

The two-dimensional projection of the curve is displayed in the lower left corner of the graphics window, scaled to a size to fit the screen. If four display areas are used, it is displayed in the middle.

The projected curves may be edited using all of the `CURVE`⇒ `Design`: `Edit` functions. However, the number of control points may not be changed.

Curve editing can be finished with the **`w`** function, which applies the curve changes to the surface. If the **`q`** function is used to exit, the changes are not applied to the surface.

## **l**engthwise editing

This function picks a length-wise curve of the surface for curve editing on a plane. The chosen curve is the one that goes through the active point, or, if the control mesh is being edited, the nearest cross-wise section.

Usually, the length -wise curves of a surface are three-dimensional curves and have to be projected into a plane for editing. DESKARTES asks for the projection direction. The default choice is an arbitrarily oriented plane that would deform the curve least.

The alternative is to project the curve in one of the coordinate axis directions. If the chosen direction is z, the curve will be edited within the xy-plane, etc. The values of the curve in the projection direction will be returned to the curve after editing is complete.

The two-dimensional projection of the curve is displayed in the lower left corner of the graphics window, scaled to a size to fit the screen. If four display areas are used, it is displayed in the middle.

The projected curves may be edited using all of the `CURVE`⇒ `Design`: `Edit` functions. However, the number of control points may not be changed.

Curve editing can be finished with the **`w`** function, which applies the curve changes to the surface. If the **`q`** function is used to exit, the changes are not applied to the surface.

## **u**ndo

Any action that changes the curve may be canceled by clicking this icon immediately after the erroneous action. Only the last one action can be canceled by this command. If you have made many incorrect edits they can be cancelled by exiting the edit mode by pressing **q**. This will cancels all changes made in this edit session.

### Re**f**resh display

Erases the screen and redraws the contents. This may be needed to clean up the display after graphical moving

### write surface

This function finishes editing the surface and saves the edits.

### **q**uit surface

This function finishes editing the surface but does not save the surface. All edits made will be discarded. The systems state after the edit command will be unchanged. You are asked to confirm the command.

## *BÉZIER SURFACE EDITING*

This command edits Bézier surfaces with local changes, either by moving points on the control mesh or by editing curves on the surface.

DESKARTES will ask you if the surface is to be edited by displaying the entire control mesh, or showing just a part of it around the active point.

The control mesh describes the surface the same way the control points of a curve describe a Bézier curve. At the corners of the surface patches, the control points lie on the surface itself.

In practice, it is recommended that changes are be made by editing surface curves, with interaction on a plane, instead of in three-dimensional space.

### next cross point **+**

This function makes the point following in the cross-wise direction the current active point active.

### previous cross point **+**

This function makes the point preceding in the cross-wise direction the current active point active.

### next length point **>**

This function makes the point following in the length -wise direction the current active point active.

### previous length point **<**

This function makes the point preceding in the length-wise direction the current active point active.

### select point **(left)**

Clicking the left mouse button activates the point nearest to the cursor.

## move horizontal **(middle)**

The point is moved continuously, so that the x and y coordinates change while z remains constant. Neighboring points may change in order to keep the surface smooth.

If four display areas are used, the point is moved in the plane where the point was selected.

## move vertical **(right)**

The point is moved continuously, so that the z coordinate changes while x and y remain constant. Neighboring points may change in order to keep the surface smooth.

If four display areas are used, the point is moved in the plane where the point was selected.

## **n**umerical values

This function changes the coordinates of the active point by entering numeric parameters for the coordinates of the point. Neighboring points may change in order to keep the surface smooth.

## **a**dd to coordinate values

This function changes the coordinates of the active point by adding values entered as numeric parameters to the coordinates of the point. This has the effect of moving the point by the amount entered. Neighboring points may change in order to keep the surface smooth.

## **N**umerical value

This function changes the coordinates of the active point by entering numeric parameters for the coordinates of the point. Neighboring points will not change so this function may produce a sharp corner.

## **A**dd to values

This function changes the coordinates of the active point by adding values entered as numeric parameters to the coordinates of the point. This has the effect of moving the point by the amount entered. Neighboring points will not change so this function may produce a sharp corner.

## **c**ross-wise editing

This function picks a cross-wise curve of the surface for curve editing on a plane. The chosen curve is the one that goes through the active point, or, if the control mesh is being edited, the nearest cross-wise section.

Usually, the cross-wise curves of a surface are three-dimensional curves and have to be projected into a plane for editing. DESKARTES asks for the projection direction. The default choice is an arbitrarily oriented plane that would deform the curve least.

The alternative is to project the curve in one of the coordinate axis directions. If the chosen direction is z, the curve will be edited within the xy-plane, etc. The values of the curve in the projection direction will be returned to the curve after editing is complete.

The two-dimensional projection of the curve is displayed in the lower left corner of the graphics window, scaled to a size to fit the screen. If four display areas are used, it is displayed in the middle.

The projected curves may be edited using all of the `CURVE⇒ Design`: `Edit` functions. However, the number of control points may not be changed.

Curve editing can be finished with the **w** function, which applies the curve changes to the surface. If the **q** function is used to exit, the changes are not applied to the surface.

### lengthwise editing

This function picks a length-wise curve of the surface for curve editing on a plane. The chosen curve is the one that goes through the active point, or, if the control mesh is being edited, the nearest cross-wise section.

Usually, the length -wise curves of a surface are three-dimensional curves and have to be projected into a plane for editing. DESKARTES asks for the projection direction. The default choice is an arbitrarily oriented plane that would deform the curve least.

The alternative is to project the curve in one of the coordinate axis directions. If the chosen direction is z, the curve will be edited within the xy-plane, etc. The values of the curve in the projection direction will be returned to the curve after editing is complete.

The two-dimensional projection of the curve is displayed in the lower left corner of the graphics window, scaled to a size to fit the screen. If four display areas are used, it is displayed in the middle.

The projected curves may be edited using all of the `CURVE`⇒ `Design`: `Edit` functions. However, the number of control points may not be changed.

Curve editing can be finished with the **w** function, which applies the curve changes to the surface. If the **q** function is used to exit, the changes are not applied to the surface.

### insert surface curve

This function inserts a new curve on the surface. Point and click with the left mouse button where you want to add the new curve. The system asks whether you want to add a cross-sectional curve, a lengthwise curve or both.

### undo

Any action that changes the curve may be canceled by clicking this icon immediately after the erroneous action. Only the last one action can be canceled by this command. If you have made many incorrect edits they can be cancelled by exiting the edit mode by pressing **q**. This will cancels all changes made in this edit session.

### Refresh display

Erases the screen and redraws the contents. This may be needed to clean up the display after graphical moving

### write surface

This function finishes editing the surface and saves the edits.

### quit surface

This function finishes editing the surface but does not save the surface. All edits made will be discarded. The systems state after the edit command will be unchanged. You are asked to confirm the command.

## *DIMENSIONING*

This mode produces dimension drawings of a curve or a curve set.

The dimensions are positioned by using the mouse to draw a line that crosses the curve. If the line does not intersect any curve, the nearest curve endpoint will be selected.

After the dimension is computed and its text is displayed you'll see the cursor change into a question mark. The dimension text can now be positioned at a chosen location by clicking on the left mouse button or any other button if you want to keep the current location.

## **b**ounding box

This function computes the horizontal and vertical dimensions of the curves. You are asked to position the two dimension lines where you want, then place the texts.

## **p**oint value

This function computes the location of a curve point. Draw a line intersecting the curve with the left-hand mouse button, and place the dimension text.

## **d**istance

This function computes the minimum distance between two points.

The points are selected one at a time by drawing a line with the left-hand mouse button. Position the dimension where required, then place the text.

## **t**hickness

This function computes the distance between two points.

The points are selected with just one intersection line drawn with the left-hand mouse button. This line should cross over the curve(s) twice. Position the dimension where required, then place the text.

## **h**orizontal distance

This function computes the horizontal distance between two points.

The points are selected one at a time by drawing a line with the left-hand mouse button. Position the dimension where required, then place the text.

## **v**ertical distance

This function computes the vertical distance between two points.

The points are selected one at a time by drawing a line with the left-hand mouse button. Position the dimension where required, then place the text.

## **a**ngle

This function computes the angle between two straight lines, shown with intersection lines. The dimension lines and text may be placed either inside or outside of the dimension lines.

## **f**illet angle

This function computes the angle of a B-spline or Bézier curve fillet. Only one intersection line is required, going through the fillet part of the curve.

The function only works if the fillet is originally produced with the curve edit function `r`, and has not been scaled or otherwise modified.

## **r**adius of fillet

This function computes the radius of a B-spline or Bézier curve fillet. There are two modes of operation, depending on the part of curve to be measured.

The system first tries to see if there exists a exactly defined radius, originally produced with the curve edit function **r**. In this case only one intersection line is required, and the radius is reported immediately.

If an exact fillet is not found, the system asks the user to locate two other points on the curve. The program then computes and displays a circular arc which passes through the three points shown, and reports the radius of that.

## **u**ndo

Any action that changes made may be canceled by clicking this icon immediately after the erroneous action. Only the last one action can be canceled by this command. If you have made many incorrect changes they can be cancelled by exiting the edit mode by pressing **q**. This will cancel all changes made in this edit session.

## **z**oom scene

This function allows zooming into the model during dimensioning, like in curve editing.

## **Z**oom back

This function returns back to the default viewing state.

## write dimensions

This function finishes dimensioning and saves the edits.

**Note**: this function is disabled when dimensioning is called from one of the curve edit modes.

## **q**uit dimensions

This function finishes dimensioning but does not save the dimensions. All edits made will be discarded. The systems state after the edit command will be unchanged. You are asked to confirm the command.

## *LIGHT EDITING*

The scene editor shows the model objects along with camera and light points.

The camera or a light may be chosen active by pointing with the left mouse button, and changed with various edit functions. All the changes may be instantly shown in the true environment and with camera view shading and ExTrace.

A very useful way to interact with the scene editor is selecting `Settings`⇒ `AREA`: `four` before starting the editor. This way you will see the camera and the light points in four views simultaneously, and change them in any one of the views at a time.

## select **(left)**

This function selects an object (light point or camera) as active by pointing and clicking with the left-hand mouse button.

## move horizontally **(middle)**

In a 3D view, the function moves the selected object horizontally.

If the view direction is from one of the axes, the function moves the object freely in the corresponding view plane.

## move vertically **(right)**

In a 3D view, the function moves the selected object vertically.

If the view direction is from one of the axes, the function moves the object freely in the corresponding view plane.

## **t**arget point

This function moves the camera or spotlight target point. After executing the function the middle mouse button moves the selected object horizontally and right mouse button moves the selected object vertically.

## **a**ngle

This function changes the camera or spot angle. Click on the left-hand mouse button and move up or down. The angle is displayed in the message line.

## **n**ew light

This function adds a new light to the scene. The light is automatically placed above the "right shoulder" of the camera. It may be outside of the field of view. You may need to use the **z** function to make it visible.

## **d**elete light

This function deletes the current active light.

## **p**osition values

This function displays a parameter box allowing the currently selected object's position parameters (location, target and angle) to be changed numerically.

## **l**ighting values

This function displays a parameter box allowing the currently selected object's lighting parameters to be changed numerically.

For a light point, the lighting parameters are the color and intensity of the light, whether it's a lighting spot light or not, and the dimming option which makes the light intensity vary according to distance in ExTrace. A light point can also be made "non-shadowing lights": by setting the "*cast* *shadow*" option to NO, the corresponding light point will only cast light, but no shadows. The overhead of ray tracing non-shadowing lights is marginal compared to shadowing lights. Thus, one can define several (up to hundreds) of light pointsto illuminate a scene without increasing computation times too much.

For the camera, the parameters are the global ambient light , and total intensity of the individual lights which can thus be scaled at once

.

## **v**iew

Shows a view of the model as seen from the camera. This will give an impression of what the final picture will contain but since the display does not take into account perspective there may be some inaccuracies.

If a spotlight is selected, the function shows the model as "seen" from the light source. This shows the area that the spotlight will light.

## **s**hade

Shows a shaded image of the model ware shading, with the currently chosen scene definitions and preferences.

## **g**l shade

Shows a shaded view of the model as seen from the camera in a separate shading window, where it may be interactively rotated etc.

Keep the window on screen while performing the scene editing functions, and all the changes made will be updated in the window after each operation is finished.

## **G**L continuous shade

Shows a shaded view of the model as seen from the camera in a separate shading window, where it may be interactively rotated etc.

Keep the window on screen while performing the scene editing functions, and all the changes made will be updated in the window continuously as the mouse is moved. However, this is only useful if you have a fast enough machine to really show the changes in real time.

To exit the continuous shade mode, give function **G** again.

## **e**xtrace image

This function begins update of the ExTrace window to show the current scene modifications. For it to work, ExTrace must have been started before entering scene editing. This is a most effective way of seeing the effect of the edits you have made.

## **z**oom and pan

This command alters the scale of the viewing window. The scale of the window can be set either graphically, or relative to the current scale.

- To zoom in freely by a view box, press and hold the left mouse button with the cursor at the center of where you want to zoom. Move the mouse up or down and a rectangle showing the new window is displayed. The window is fixed by releasing the mouse button.

- The view will change continuously if you press the middle mouse button whilst moving the mouse. However, in four views and the edit modes, zooming always works in box mode as above.
- By clicking the right button, the window becomes twice as big as it currently is — the objects on the screen are shown smaller.
- By clicking on any key on the keyboard, you will be able to pan instead of zooming. Pan the view by clicking with the left mouse button at any point on the screen, then move the cursor to the new location of the first point, and release the button. This feature is particularly useful when zooming in the various edit modes, where only the zoom function is available.

### `write scene`

This function finishes editing the scene and saves the edits.

### `quit scene`

This function finishes editing the scene but does not save the scene. All edits made will be discarded. The systems state after the edit command will be unchanged. You are asked to confirm the command.

## *TEXTPAINT EDITING*

TextPaint is a simple painting program tailored for use within DESKARTES. Specifically, it can be used to design and edit textures and backgrounds for ray tracing, or to edit pictures created with ExTrace, for example, to add text.

TextPaint includes painting tools, pattern tools, tools for cropping, cutting, and pasting graphics, polyline tools, and various controls for the above tools. It also enables reading and writing of image files, as well as converting curves from the DESKARTES design modules to be used as a basis for a texture image.

### *Starting TextPaint*

As the program is started, the program asks if the current display contents should be converted to TextPaint. This way, you can use the entire curve editing functionality of DESKARTES to make a layout of the graphic design, and fill the curve areas with chosen colors or patterns with TextPaint.

Another alternative is to keep the contents of the current display area visible, but not include it in the resulting image. This allows you to have a surface model shown as an underlying template while drawing the picture. The picture can thus later be conveniently texture mapped onto the surface using the bounding plane method, for instance.

The third alternative for starting TextPaint is to erase the background first, and start drawing on a blank screen.

### *The toolboxes*

The TextPaint tools are organized into a number of toolboxes.

The icons for the toolboxes are shown when TextPaint is started. Clicking on its icon with the left mouse button opens a toolbox. This displays the tools the toolbox contains. To close a toolbox, click on its icon again.

## *General tools*

The general toolbox contains the most commonly used tools for making graphics. These include

- Painting tools such as the pen, the airbrush, the eraser, and area fill.
- File tools that read graphic and save graphic.
- Undo tool to cancel the last operation.
- Typographic tools for managing text.
- Tools that deal with cutting and pasting.

The program also includes functions for reading and writing existing texture files. This means that for example ray traced images may be edited, to change the color of individual pixels, add text, etc.

Both reading and writing may be directed at only a part of the texture or the whole image. If the image is too large to fit into the graphic window, scrolling is available to work on selected parts of the image at a time. You may thus manipulate very large images, limited only by the computer memory available.

**Note**: only 8-bit color images may be read into TextPaint. If you wish to edit 24-bit images, you must first convert them to 8-bit. Use the command `FILES`⇒ `File`: `Convert` for this.

## primitive tools

The primitive tools are used to create primitive elements, such as rectangles, polygons, and ovals. They can be either hollow or filled with a solid color or a pattern. The left column of the primitive toolbox has the hollow tools and the right one contains the filled tools.

## control tools

The ways the different tools work depend on a number of adjustable controls. For example, the pen or airbrush width, pattern, and color may all be chosen. The appearance of the filled primitives depends on what kind of fill is selected.

## tile tools

The effect of the various painting tools depends partly on the “mode” set with the control tools: solid, stipple, or tile. “Solid” `means that whatever is` drawn is drawn in the active color. “Stipple” uses a “transparent” pattern, with the background showing through transparent parts of a pattern. “Tile” means that a non-transparent pattern is used. The tile tools make it possible to select, create, and edit different kinds of tiling patterns.

## Color Chart

### *Selecting from color chart*

You may select the drawing color from the color chart by pointing at the desired color and clicking the left mouse button. The selected color becomes the active drawing color and the drawing method is set to solid. At the same time, the content of the color icon in the control tool changes to show the new drawing color and drawing method.

### *Selecting from picture*

You can select a color by clicking anywhere in the picture with the middle mouse button: the active color changes to the color of the pixel you pointed. This works while using the magnifying tool, too.

### *Selecting from color palette*

You can change any color of the color chart by clicking at it with the middle mouse button. This pops up the color palette window and allows you to edit the color in the usual way, as with command `SCENE`⇒ `Edit`: `Materials`.

The selected drawing color may also be changed with the color palette by pointing at the color icon within the control tools, and clicking the middle mouse button.

However, note that changing a color in the color chart will change the picture: those parts of the picture that are colored with the changed color will change as well.

## *Selecting the tools*

Tools are the actual functions used to create and edit graphics. All of the activity in TextPaint takes place through them. To use — or select — a tool, click at the tool icon with the left mouse button.

Unlike other edit modes, shortcuts are not available to activate the tools. To de-select a tool, click the right mouse button, or click at its icon again with the left mouse button.

If you click at a tool with the middle mouse button, a brief explanation of the command is shown in the message line.

## *Command menus*

All the commands in the toolboxes can also be started from the command menus at the top of the screen. In addition, there are two commands that are found only in menus.

Selecting the command Main Palette from the PALETTES menu hides all the toolboxes.

If the current display has been chosen as template when starting TextPaint, it can be redrawn at any time by selecting `Refresh Object` from `PICTURE` menu.

## *General tools*

### `Read Picture From File`

This function first requests the name of the file to be read. The name is entered in the usual way, without the preffixes and suffixes. For example, the file `tex_mydesign.pic.Z` is entered as just `mydesign`.

If the size of the picture to be read is smaller than the display area, a box the size of the picture appears. The box may be placed within the display area by dragging the box to the desired location and clicking the left mouse button.

If the picture is larger than the selected display area, the display area is automatically enlarged to hold the picture.

If the picture does not fit into the entire graphics window, scrolling is enabled to work with selected picture areas at a time.

**Note:** Reading a picture always changes some TextPaint color map slots and therefore may affect the earlier contents of the display, even the user interface colors. The program, however, tries to minimize the effect of this by selecting color map slots for the new picture so that they do not conflict with slots of previously read pictures. If this is not possible, a warning message is displayed so you may cancel the reading.

## Write Picture To File

First, the program asks you for the name of the file. The prefixes and suffixes should not be included as the program adds them automatically.

After this, you will be asked which part of the picture is to be saved. There are four options.

- Free select - Locate the lower left and upper right corners of the saved area with the mouse.
- Relative select - Specify the ratio of the horizontal and vertical sides of the image, then draw the area that will stick to the given ratios.
- Whole drawing area - The entire display area will be stored in the file.
- Last read picture - This is intended for storing large (retouched) images that don't fit in the screen at once.

## Undo Previous Action

This function returns the picture to the state preceding the previous action. It cancels the changes. Only the last action may be canceled.

## Erase Picture

This function erases the entire picture. You may still get the picture back with the `Undo` function, before starting another function.

## Select Font

A font selection window appears. It has three sub windows, showing the fonts available in the system by the font name, style and size.

A new font may be selected by pointing on its name, style and size in the selection windows and clicking the left mouse button. When an item from one of the lists is selected, the two other lists are updated so that they contain only the viable combinations with this selected item. You may also release the selections by clicking the selected item again with the left mouse button.

When all three attributes are specified, a separate window appears up below the selection window, with sample text displayed in the selected font.

To make the specified font active, click the OK -button at the bottom of the font selection window. The font selection is ended and an example text in the same, selected font appears in the font window at the right end of the control toolbox.

If you only wish to examine the appearance of different fonts without actually selecting one as active, end the font selection by clicking the CANCEL-button at the bottom of the font selection window.

## Type Text In Picture

A pointer box appears. Select the starting position of the text by setting the box to the desired location and click the left mouse button. The box remains there to show the position of the first character.

When the starting point has been chosen, you may type text normally. The pointer box always shows where the next character will appear. If you want to select a new starting point while the function is being performed, click the left mouse button within the graphics window and select the starting point as described above.

The `Return` key allows you to start a new row, assuming that it fits within the drawing area. The `Del` and `Backspace` keys allow you to delete characters, but only to the beginning of the row.

The colors, display method, pattern, and font of the text is determined by the active control settings. They may be changed normally while typing. However, if you change font, you may not delete characters with the Del key beyond such a point, even if they are on the same row.

## Enlarge Part of Picture

A selection box appears. Place it on the area you wish to enlarge and click the left mouse button. The selection box remains where you put it, and an enlarging window appears in the upper part of the screen.

The pixels of the enlarged part show as squares. You may move the enlarging window to wherever you want on the screen, so that the enlarged and the true-size pictures are visible simultaneously. You may also resize the enlarging window to cover a larger area of the picture.

The squares visible in the enlarging window may be colored with the active drawing color by pointing at them and clicking the left-hand mouse button. The middle button allows you to choose a new color either from the enlarging window or from the graphics window. The drawing color is set to the color of the pointed pixel or square.

## Free-Hand Erase

When this function is started, you may erase parts of the picture by holding the left mouse button and moving the cursor on the screen. The way the eraser track behaves is determined by the pen size and picture background color.

## Free-Hand Draw

When this function is started, you may draw by holding the left mouse button and moving the cursor on the screen. The appearance of the pen is determined by the pen size, drawing color, pattern, and drawing method. All of these may be changed as usual during drawing, as explained in the color chart and control tools sections.

## Airbrush

Paint by holding the left mouse button and moving the cursor on the screen. A track that resembles an airbrush track remains on the screen. The appearance of the track is determined by the spray size, drawing color, pattern, and drawing method. All of these may be changed while painting.

## Area Fill

Areas of a picture may be filled with a solid color or pattern. The filled area is taken as all the pixels of the same color as the starting point that are connected to each other. Depending on the controls, the area is filled by repeating the pattern over the entire area or by coloring all of the pixels with a single color.

When the function is started, select the starting point by clicking at the desired location with the left mouse button. When the area in question is colored, the function is left active, allowing you to color more areas by clicking at new starting points.

You may interrupt filling an area by clicking the middle mouse button. After this, you may select another starting point from that or some other area.

## Copy Part of Picture

Select an area of the picture. The selected part of the picture is stored for the `paste` function.

### Cut Part of Picture

Select an area of the picture. The selected part of the picture is cut away from the picture. The part of the picture is stored for the `paste` function, just as with the `copy` function.

### Paste Part of Picture

A selection box appears, containing the part of the picture that was last selected with the `cut` or `paste` function. Place it where you wish and click the left mouse button.

## *Primitive tools*

These tools create primitive elements composed of lines or arcs. They can be either hollow or solid. The hollow functions are in the left column; the solid ones to the right. The pen size, current color and pattern are applied to these tools the same way as to the general tools.

The tools are dealt with in pairs: the filled tool works just like its unfilled twin, except that it is filled. The sole exception is the line/polyline tool pair.

### Line/polyline

To draw a single line, select the line tool. Then click at each end of the line to draw it.

To draw a series of lines connected at the ends, use the polyline tool. Click at the starting point, then the intermediate points. To end the polyline, click the right mouse button.

### Polygon

The polygon tool works just like the polyline tool, except that it closes the polygon when the right mouse button is clicked. The solid polygon is filled with the active color and pattern.

### Arc/segment

To draw an arc of a circle, select the arc tool. Click at the center of the circle, then the starting point of the arc and then the end of the arc.

The arc is drawn counterclockwise: if the end is clockwise of the starting point, a full circle with a slice cut out is drawn.

The corresponding filled tool connects the ends of the arc and fills normally.

### Sector

This tool works just like the arc/segment tool, except that the sides of the sector (connecting the center of the circle and the end points of the arc) are displayed. The solid sector fills the area delimited by them and the arc.

### Oval

To draw an oval, select the oval tool. Then click at a corner of an imaginary bounding box of the oval, then at the other. The oval is drawn within the rectangle defined by the two clicks.

### Circle

Works just like the oval, only this makes a perfect circle.

### Square

To draw a square, select this tool and click at the diagonally opposite corners.

### Rectangle

Works just like the square tool only it does not force a perfect square.

## *Tiling tools*

### *Select Stipple Pattern*

Pops up a palette window, which enables you to choose a stipple pattern with the left mouse button. If there are more patterns than fit on the screen at a time, arrows appear in the corners of the window, which enable you to scan the patterns. Exit the function by clicking the right button within the palette.

A stipple pattern has no solid color: when drawing with it, the parts that look black are painted with the selected drawing color. The rest of the pattern is transparent: the underlying picture (or background) is visible through it.

It is not possible to define new stipple patterns with the TextPaint program.

### Select Tiling Pattern

Displays a window that allows you to choose a tiling pattern with the left mouse button. If there are more patterns than fit on the screen at a time, arrows appear in the corners of the window, enabling you to scan the patterns. Exit the function by clicking the right button within the palette.

In a tiling pattern each pixel has its own color, and the background is covered entirely when painting with it.

There are three ways of defining tiling patterns: 1) draw it yourself, 2) edit an old one or 3) select from picture. The three ways are explained in more detail below.

### Draw New Tiling Pattern

When the function is started, the actual picture is hidden so that it does not bother drawing the pattern. It will return to view when you are done with defining the pattern.

The empty picture allows you to draw and edit (with the Enlarge function) a picture that will be used as a pattern.

When the pattern is ready, click the icon of the function with the left mouse button. Select an area of the picture. The selected part of the picture is made the active drawing pattern and the creation function is ended.

### Edit Old Tiling Pattern

When the function is started, the active picture is hidden so that it does not interfere with the drawing of the pattern. It will return to view when you are done with defining the pattern. The selected pattern appears in its own box in the middle of the picture.

You may now draw and edit (with the Enlarge function) the old pattern.

When you are ready, click the icon of the function with the left mouse button. The pattern produced by changing an old one is saved and made the active pattern, after which the function ends. The old pattern remains within memory also: the changed pattern is, in fact, a copy.

### Select Tiling Pattern

This function allows you to select a part of the picture as a new tiling pattern.

Select an area of the picture. The selected part of the picture is made the active drawing pattern and the creation function is ended.

### Delete Tiling Pattern

Deletes the active tiling pattern. You will be asked for confirmation, because this cannot be undone.

## *Control tools*

### Change pen size

The pen size may be changed with the leftmost slider control at the control toolbox or with the two buttons above it.

### Shrink Pen

Shrinks the pen width (the width of the trace in pixels) by two. The smallest possible pen size is 1. The number next to the icon of this function displays the active pen size.

### Enlarge Pen

Enlarges the pen width (the width of the trace in pixels) by two. The largest possible pen size is 99. The number next to the icon of this function displays the active pen size.

### Change spray size

The spray size may be changed with the middle slider control at the control toolbox or with the two buttons above it.

### Shrink Spray

Shrinks the spray size (the radius of the trace in pixels) by one. The smallest possible spray size is 10. The number next to the icon of this function displays the active spray size.

### Enlarge Spray

Enlarges the spray size (the radius of the trace in pixels) by one. The largest possible spray size is 100. The number next to the icon of this function displays the active spray size.

### Change enlarging factor

The enlarging factor may be changed with the rightmost slider control at the control toolbox or with the buttons above it.

## Smaller Zoom Factor

Makes the enlarging factor smaller by one. The smallest possible enlarging factor is 2. The number next to the icon of this function displays the active enlarging factor.

## Bigger Zoom Factor

Makes the enlarging factor bigger by one. The largest possible enlarging factor is 32. The number next to the icon of this function displays the active enlarging factor.

## Drawing Method Solid

Makes a solid color the drawing pattern. The active color is visible in the color window above the font window.

## Drawing Method Stipple

Makes a stipple pattern the drawing pattern. The active pattern is visible in the color window above the font window.

## Drawing Method Tile

Makes a tile pattern the drawing pattern. The active pattern is visible in the color window above the font window.

## Exit TextPaint

Quits TextPaint. If you have not stored your picture into a file, the program asks whether it should now be stored. The program also asks if the image should be stored as a design template: see command `TEXTURE`⇒ `Image`: `Refresh On/Off`.