Win3D
User's Guide

Version 1: 1994
Version 2: 2022

Leen Ammeraal

1

Introduction

Axes and coordinates

We will use a right-handed coordinate system, with *x*-, *y*- and *z*-axes, as shown in Figure 1.

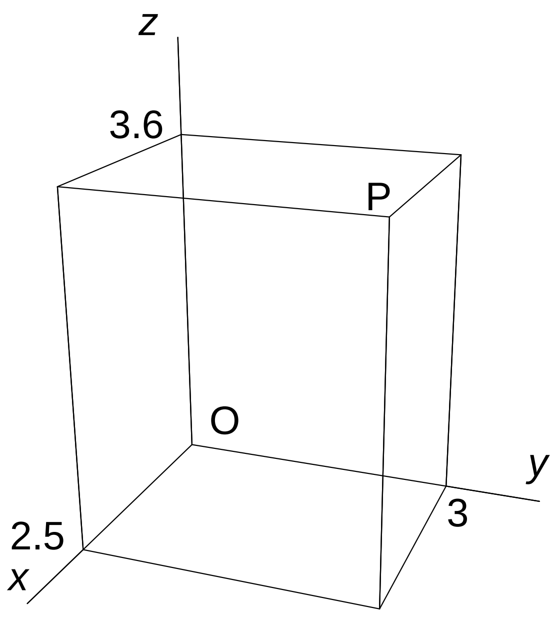


Figure 1. *Axes and point P(2.5, 3, 3.6)*

Our *xy*-plane (that is, the plane through the *x*- and *y*-axes) will be horizontal, with the positive *x*-axis pointing toward us, the positive *y*-axis pointing to the right and the *z*-axis upward. We can denote every point in three-dimensional space by its three coordinates. For example, we write

P(2.5, 3, 3.6)

to denote the point P, shown in Figure 1, with coordinates $x_P = 2.5$, $y_P = 3$ and $z_P = 3.6$. This point lies at a distance 2.5 in front of the *yz*-plane, at a distance 3 to the right of the *xz*-plane and at a distance 3.6 above the *xy*-plane. The point through which all three axes pass is called the *origin* O, for which we can also write O(0, 0, 0).

Special points, such as those at the eight corners of a cube, are called *vertices* (which is the plural form of *vertex*). Each vertex that we will be using is also assigned a unique number, which is a positive integer. We will frequently use sequences of vertices to define boundary faces of solid objects.

Win3D: What can you use it for?

Win3D is especially useful if you want to view three-dimensional objects in perspective, both on the screen and as high-quality line drawings, which you can import into text documents. This user's guide gives many examples of the kinds of object suitable for Win3D and the way they can be represented. Since this guide is printed in black and white, it shows most results in line drawings, but on the screen you can also use the `hidden faces' mode, with faces in various shades of yellow against a blue background. The cart shown in Figure 2 is an example of a line drawing, with hidden lines automatically removed.

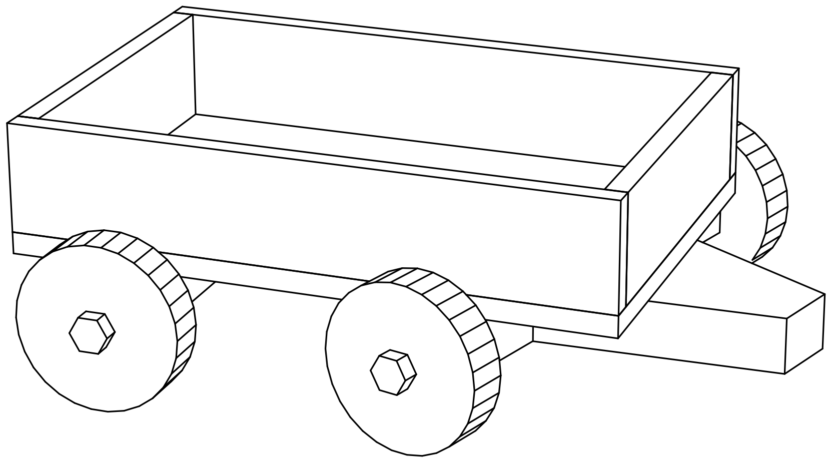


Figure 2. *Cart*

Building this object yourself on your computer might take you several hours, but you can also obtain it very quickly by opening the file CART.DAT. In either way, you can easily produce several views of it by changing your viewpoint. Figures 2 and 3 are examples of such views.

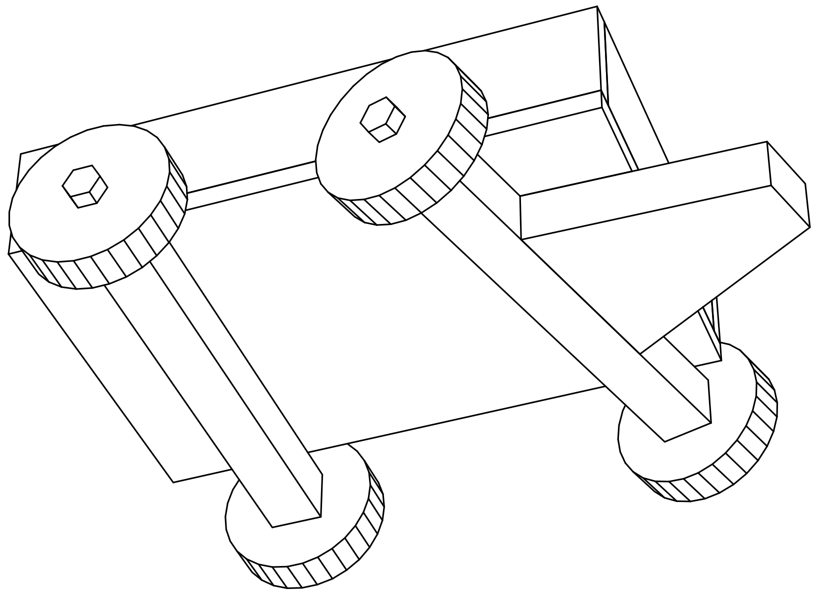
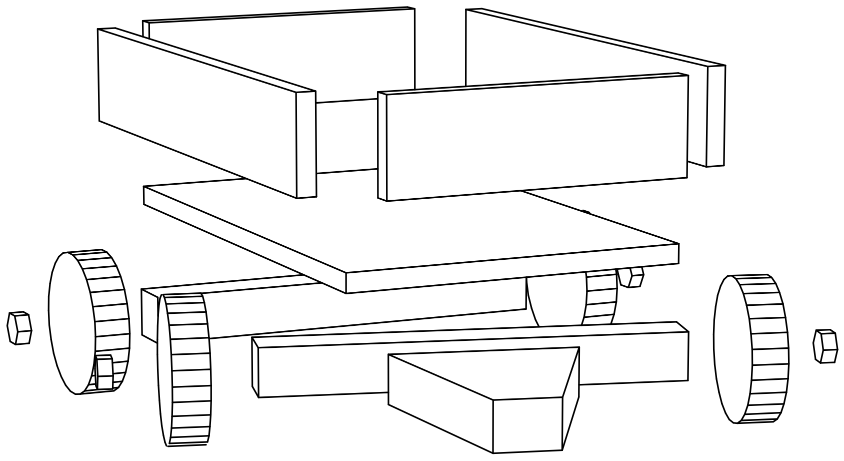


Figure 3. *Cart viewed from below*

You can also modify the object very easily because complicated objects can be divided into *groups*. The cart of Figures 2 and 3 consists of 16 parts, each forming a group. Groups can easily be moved, copied, rotated, and so on. Figure 4 was obtained by separating these groups.



Figuur 4. *Parts separated*

Another object is shown in Figure 5. It is a solid of revolution, or rather, three quarters of one.

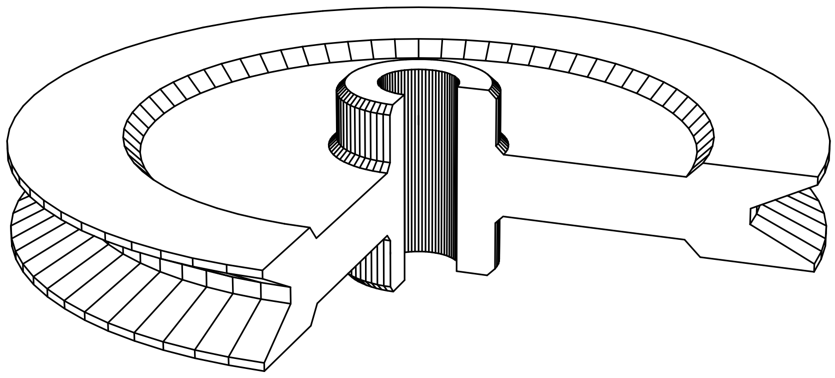


Figure 5. *Wheel for V belt*

Actually, this wheel is simpler than the cart we have just seen. It is generated in a single rotate/sweep transformation, with a rotation angle of 270°. This example can be viewed by applying the *Open* command to the file VWHEEL.DAT.

A rotation of 270° is as simple as one of 360°, applied to construct the lower part of the chess piece shown in Figure 6.

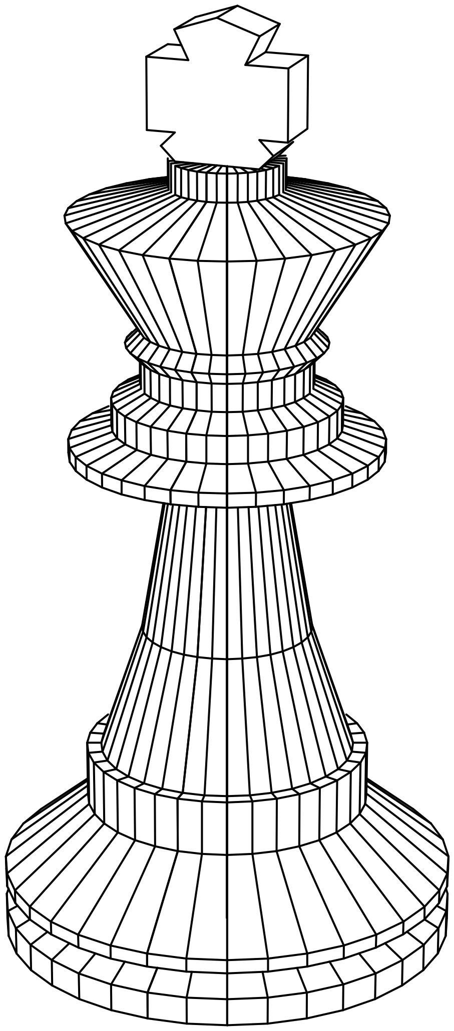


Figure 6. *King*

This object consists of two parts, placed one on top of the other. These two parts can separately be constructed and saved in data files.

Since the upper part, a cross, is symmetrical, we can use the *mirror* transformation to build it, as we will see in Chapter 6. You can view this example by opening the file CHESKING.DAT.

All capital and lower-case letters as well as the ten decimal digits are also supplied as 3D object files. These files are in the FONT subdirectory; their names are CAPA.DAT, ..., CAPZ.DAT for the capital letters, LCA.DAT, ..., LCZ.DAT for the lower-case letters, and DIG0.DAT, ..., DIG9.DAT for the digits. After having opened one such a file for the first character, we can import some other files one by one, each time moving the new character to its correct position. Figure 7 shows an example of a name constructed in this way.

Viewing modes

When working with Win3D on your computer, you can use these three viewing modes:

- wire-frame mode, with all lines drawn (see Figure 8);
- hidden-line mode, with hidden lines removed (see Figure 6);
- hidden-face mode, with a blue background and shades of yellow used for the visible faces of the object (see Figure 7 for a black and white surrogate).

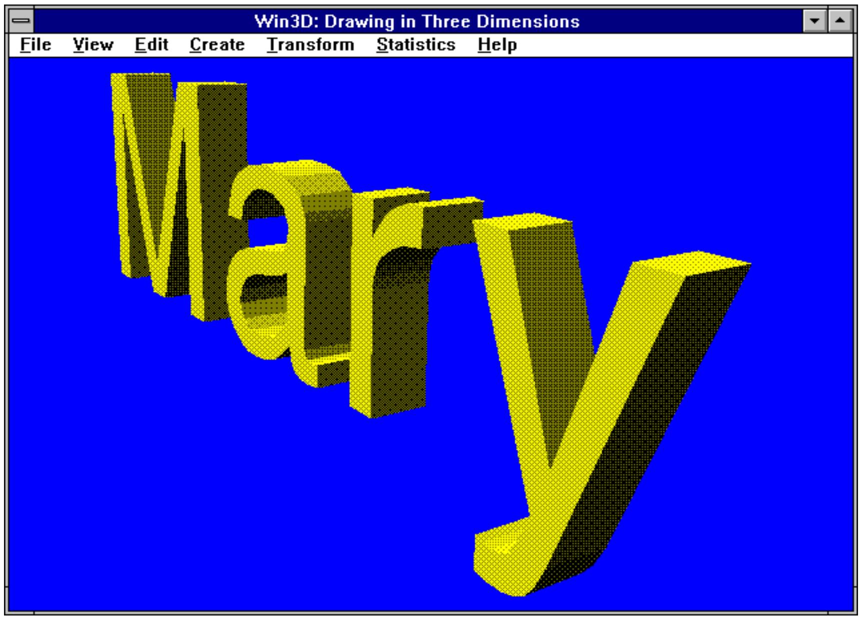

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 7. *Name displayed in hidden-face mode*

HPGL files

Win3D can export HPGL files, which you can import into text documents with most DTP packages. HPGL files generated by Win3D have successfully been imported by CorelDRAW, DrawPerfect, WordPerfect Presentations, WordPerfect, Word for Windows, Ventura Publisher and PageMaker. Vector-oriented drawing packages are useful to enhance the figures before we import them into our text documents. For example, Figure 1 was obtained in this way: in the first draft, all lines were drawn by Win3D; then, with CorelDRAW, some of these drawn lines were replaced with dashed lines and some text was added.

DXF files (for AutoCAD)

If you are an AutoCAD user, you can produce DXF files, which will be accepted by AutoCAD. The DXF files generated by Win3D describe the complete 3D objects, so in AutoCAD you can use different viewpoints for the object, in the same way as this can be done in Win3D. These DXF files are therefore not accepted by CorelDRAW or DTP packages, which expect only 2D input data. As discussed above, we use HPGL files for this purpose.

2

Menus: A Quick Tour

After starting Win3D, we obtain a start window, which displays some general information and disappears as soon as we press a key. Then, below the title bar, we find the *main menu*, showing the names of the *submenus* (or *menus*, for short): *File*, *View*, *Create*, *Transform*, *Statistics* and *Help*. Each of these menus contains a number of *menu items*, or *commands*.

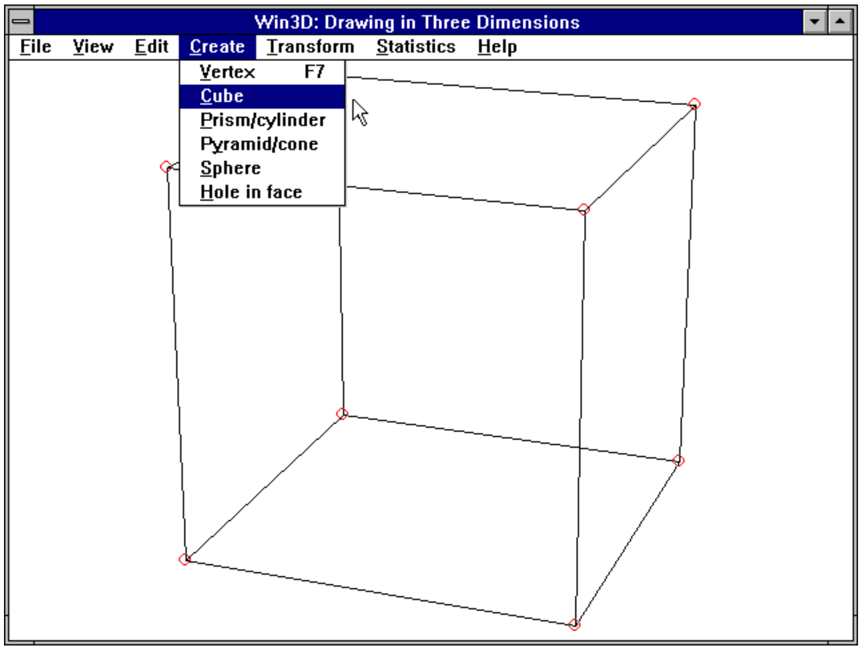


Figure 8. *Create menu*

In this chapter, we will examine each of the seven menus and demonstrate some of their commands. Let us begin with the *Create* menu, which, together with the title bar and the main menu, is shown in Figure 8. We select this menu either by clicking its name, *Create*, with the mouse, or by typing the first (underlined) letter, *C*, of this name. This menu enables us to create some standard three-dimensional objects (also called *stock objects*): a single point, a cube, a regular prism, a pyramid and a sphere. After selecting *Cube*, a cube appears, also shown in Figure 8. (The menu then disappears, so actually the *Create* menu was selected twice to produce this illustration.)

Immediately after creating a cube in this way, its corners (or *vertices*) are surrounded by red circles, which means that these eight points are currently *selected*. We can use selected points in several ways. Here we will have a look at their vertex numbers, and their coordinate values. We do this by using the *View* menu, in which we find the commands *Show nr*, *Show x*, *Show y* and *Show z*.

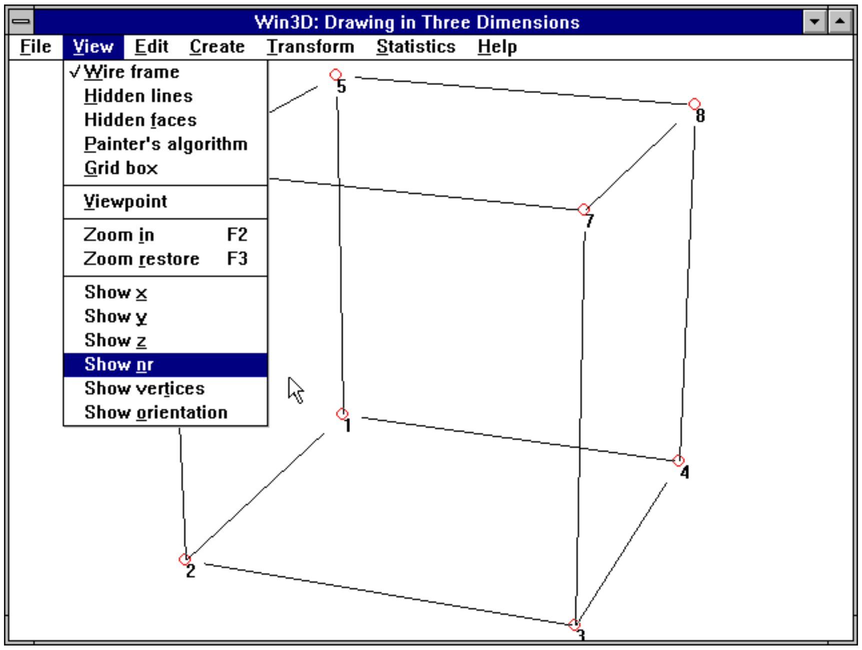


Figure 9. *View menu*

Since all eight points are currently selected, all eight vertex numbers are displayed as a result of the *Show nr* command. Figure 9 shows only seven because *View* was selected once again to display the *View* menu as well. Coordinates are displayed analogously by using the other three commands just mentioned. With more complex objects, point numbers or coordinate values would frequently overlap if we tried to display them all at the same time in this way, so that they would not all be readable.

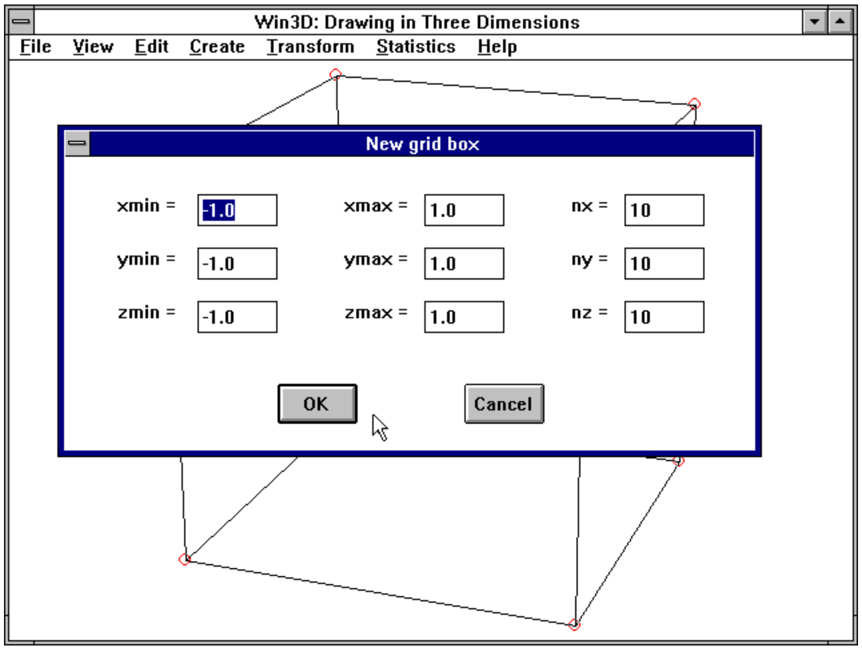
Selecting (or deselecting) a single point

Fortunately, we can select any subset of the set of vertices of the object we are dealing with, and we can also *deselect* any selected points. For a very limited number of vertices, the quickest way to select or deselect them is by clicking them, using the *right* mouse button.

Selecting (or deselecting) a set of points

Alternatively, we can press the *left* mouse button to define a corner of a rectangle; we then move the mouse while holding the mouse button down.

The position where we release the mouse button defines the opposite corner of the rectangle just mentioned. Any vertex lying inside this rectangle will now be selected if was not previously selected; conversely, it will be deselected if it was selected. (When we are in the *grid-box* mode, to be discussed next, we must choose *Box select* from the *Edit* menu to select points in this way.)


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
New grid box
OK
Cancel

xmin =	-1.0	xmax =	1.0	nx =	10
ymin =	-1.0	ymax =	1.0	ny =	10
zmin =	-1.0	zmax =	1.0	nz =	10

Figure 10. *Dialog box generated by* ***View | Grid box***

The grid-box mode

Figure 9 does not show the coordinate system we are using. We can make this visible by using the *Grid box* command, which can also be found in the *View* menu. A *grid box* is a box that surrounds the object that we are dealing with. It shows grid lines in the (horizontal) *xy*-plane. There is also a grid (although invisible) in the *z*-direction. We can define how far the grid points should lie apart, along with the minimum and maximum *x*-, *y*- and *z*-coordinates of the grid box. This is accomplished by means of a dialog box, which appears before the grid box is displayed, as shown in Figure 10.

By default, the minimum and maximum values of the grid box are equal to those of the current object. In this case the default grid box will have −1 and +1 as its minimum and maximum coordinate values, for each of the three coordinates *x*, *y* and *z*. As you can see in the above dialog box, the resulting *x*-range is by default divided into 10 intervals or *steps* of equal length; the same applies to the *y*- and *z*-ranges. If we do not change these nine default values (shown in Figure 10) the grid box of Figure 11 appears when we click the OK button.

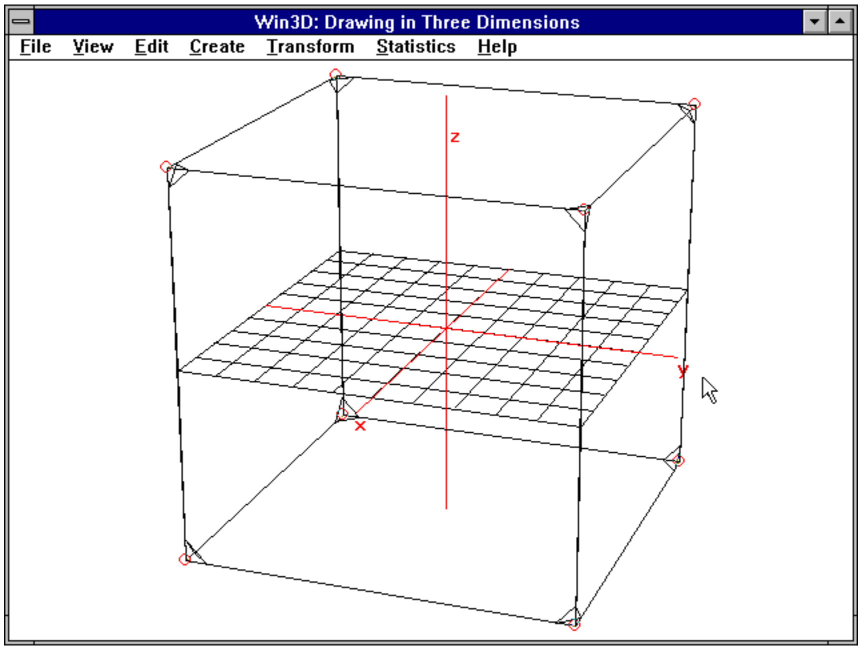

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Figure 11. *Cube in grid box*

A grid box enables us to introduce new points very easily. All we have to do is to click a grid point (of intersecting grid lines) in the *xy*-plane and *drag* this point upward or downward. The latter means that we hold the left mouse button down as we move the mouse button. At the point where we release the mouse button, a new point appears. Chapter 4 will show why this can be useful. We will now show some more views of our cube. As we have seen in Figure 9, there are also the commands *Wire frame*, *Hidden lines*, *Hidden faces* and *Painter's algorithm* in the *View* menu. These commands (and the *Grid box* command just discussed) are used to change the *view mode*.

Line drawings

Our initial cube, shown in Figure 9, was in the *wire-frame* view mode: it shows all twelve edges of the cube. In reality, these edges are visible only if the cube is a wire-frame model. If it is a solid (and opaque) object, three of its edges are invisible: they are obscured or hidden by the object itself. The *Hidden line* command of the *View* menu prevents such hidden lines from being drawn, as shown in Figure 12. (This figure also shows the *Statistics* menu, to which we will refer shortly.)

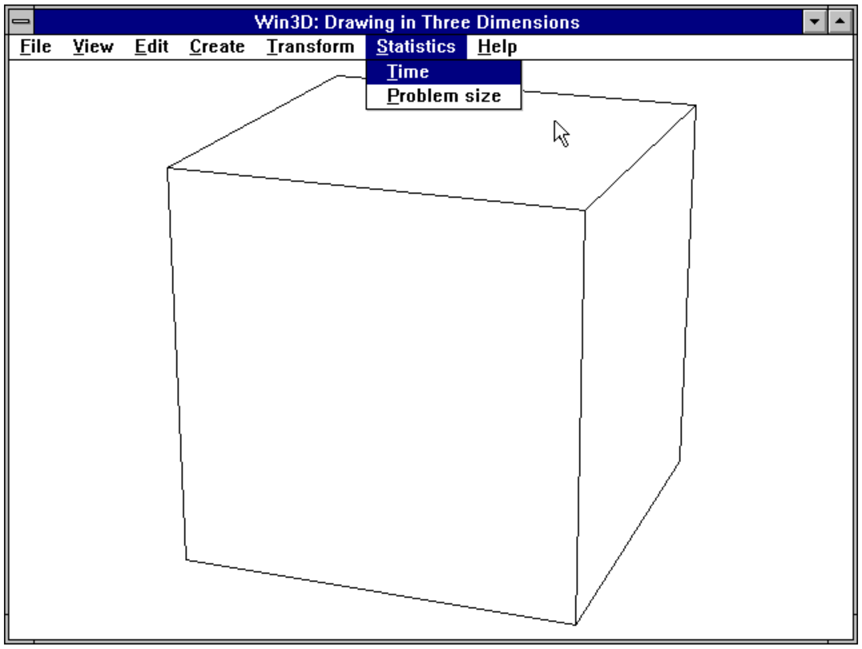

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Time
Problem size

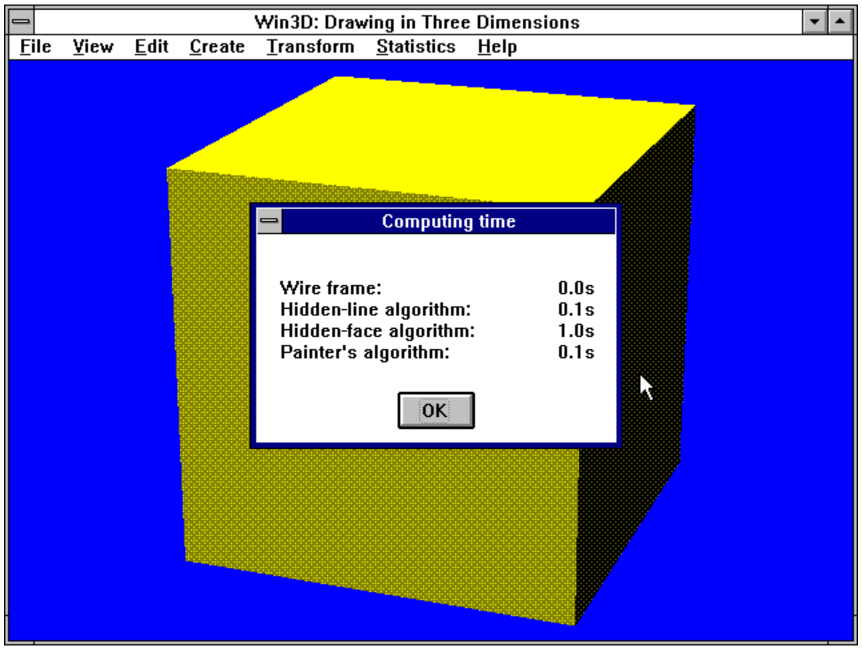
Figure 12. *a. Hidden lines removed; b.* ***Statistics*** *menu*

Filled faces

Instead of line drawings, we can represent the faces of our object as filled areas. The following two commands to achieve this can be found in the *View* menu (see Figure 9):

Hidden faces: the safer way
Painter's algorithm: the faster way on relatively slow machines (but it may give incorrect results)

These two commands are intended to give identical results; in most cases the Painter's algorithm gives correct results. If it does not, use *Hidden faces* instead. Incidentally, if you have a 486 DX machine, the latter command will probably work fast enough, so in that case you there is no need for you to use the *Painter's algorithm*. With both commands, the faces of the object are painted in shades of yellow and the background is blue. Since this User's Guide is only in black and white, the actual result on the screen looks much better than Figure 13. It was produced by both the *Painter's algorithm* and the *Hidden faces* commands, with the same result. Figure 13 also displays a message box, generated by using the *Computing time* item in the *Statistics* menu (shown in Figure 12) and displaying the amount of computation time required for the four viewing modes we have been discussing.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Computing time
OK

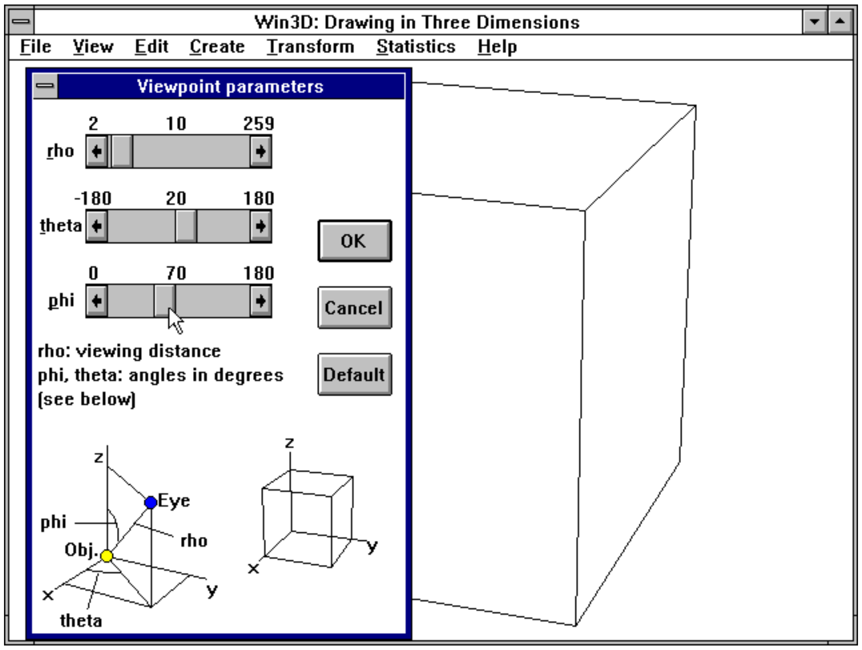
```
Wire frame:                0.0s
Hidden-line algorithm:     0.1s
Hidden-face algorithm:     1.0s
Painter's algorithm:       0.1s
```

Figure 13. *Hidden-face representation and computing times*

These figures were obtained on a 486 DX computer (66 MHz), and the object shown here is very simple. Obviously, computing-time figures will be considerably higher if your computer is slower or if you are dealing with more complicated objects.

Changing the viewpoint

One of the most interesting commands in the *View* menu is *Viewpoint*. We use it to define the position from where we view the object; in other words, it is the position of an eye that looks at the object, or of a camera that takes a picture of it. The image produced in this way is similar to that produced by Win3D. This eye or camera is directed to the center of the object. Let us write E for the viewpoint (or Eye) and O for the center of the object. (In our example, O is also the origin of the coordinate system, but this need not be the case; if this origin does not coincide with the center of the object, we still denote this center by O.) Then EO is also known as the *line of sight*. This line of sight is the axis of a pyramid with apex E, as shown in Figure 14.



rho: 2 10 259
theta: -180 20 180
phi: 0 70 180

rho: viewing distance
phi, theta: angles in degrees
(see below)

Figure 14. *Viewing pyramid*

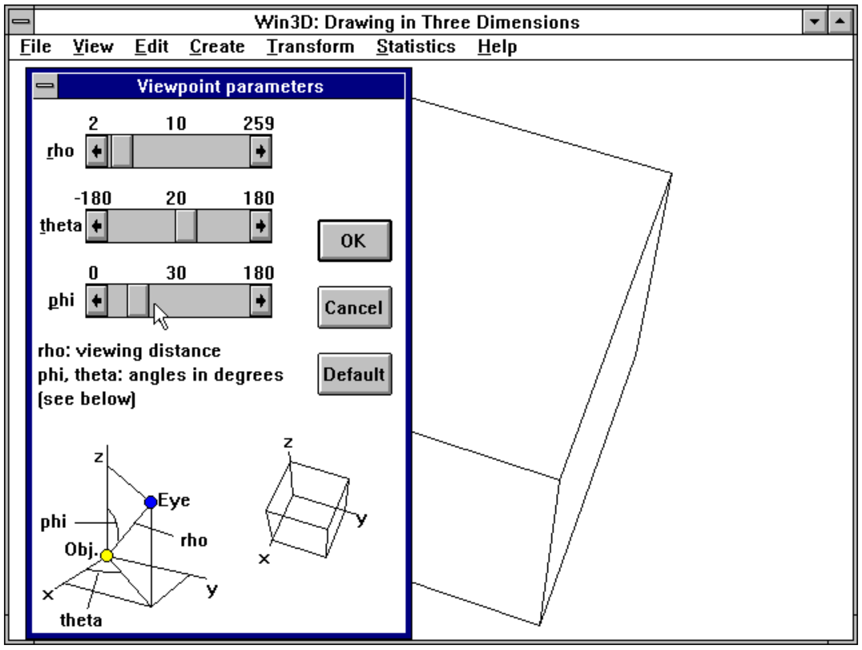
We consider a plane, the *projection screen*, perpendicular to EO and intersecting line EO somewhere between E and O. For every point P of the object, the line that connects P with point E intersects the projection screen in point Ṕ, the image of P. Since all projection lines, such as PE, pass through the same point E, this type of projection is known as *central projection*. It is required to obtain truly perspective images. By contrast, parallel projection, with parallel projection lines, is more common in traditional technical drawing practice because it is easier for manual drawing. If central projection is used with a very large distance between E and O (compared with the size of the object), the resulting images are almost the same as those obtained with parallel projection.

So far we have used the default position of viewpoint E. Win3D enables us to change this position very easily, so we can produce many different views of the same object. After selecting *Viewpoint* in the *View* menu, a dialog box appears as Figure 15 shows.

Instead of using Cartesian coordinates *x*, *y* and *z*, we denote the position of viewpoint E by its spherical coordinates ρ (= *rho*), θ (= *theta*) and φ (= *phi*). As the above dialog box shows, ρ is the length of EO, θ is an angle measured in the *xy*-plane and φ is an angle measured in a vertical plane. More specifically, φ is the angle between the positive *z*-axis and OE. As for θ, this is the angle between the positive *x*-axis and line OE′, where E′ is the projection of E onto the *xy*-plane. The angles θ and φ are expressed in degrees. The above dialog box shows a scroll bar for each of the spherical coordinates ρ, θ and φ. We can change each of these three values in many ways:

1. by dragging the slider of the associated scroll bar;
2. by clicking the buttons at the end of this scroll bar;
3. by clicking the areas of the scroll bar between the slider and the buttons just mentioned;
4. by pressing the PgUp and PgDn keys;
5. by pressing the arrow keys;
6. by pressing the Home and End keys.

You are strongly advised to experiment a little with each of these six methods. The picture at the bottom right of the dialog box then changes correspondingly, so we can see the effect. For example, suppose we want to place the viewpoint higher, to view the object more from above than is the case now. This means that we must reduce the angle φ. Using the above method 3, we can click the third scroll bar between its slider and its left button several times. Each time, φ is reduced by 10°, and the effect is shown by the axes and the small cube in the dialog box. After doing this four times we have reduced φ from 70° to 30°, as shown below:


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Viewpoint parameters
OK
Cancel
Default
z
phi
Obj.
Eye
rho
x
y
theta
z
x
y

rho: 2 10 259

theta: -180 20 180

phi: 0 30 180

rho: viewing distance
phi, theta: angles in degrees
(see below)

Figure 16. *Angle φ reduced*

The image outside the dialog box is not immediately updated. It is, however, as soon as we click the OK button, that is, if at least one of the spherical coordinates has changed. Their default values are θ = 20°, φ = 70°, while the default value of ρ depends on the dimensions of the object.

Deformation by moving some vertices

We can transform a cube into other objects by moving some of its vertices. The cube dealt with so far has the points (1, 1, 1) and (−1, −1, −1) as two opposite vertices. We can see this by selecting all vertices of the cube by using *Edit* | *Select all* (that is, by using *Select all* of the *Edit* menu) or by pressing F8, followed by *View* | *Show x* to find the *x*-coordinates; the *y*- and *z*-coordinates can be found analogously. Let us now move the two vertices right at the top a distance 1 downward. To do this, we begin by deselecting all points, using *Edit* | *Deselect all* or pressing F9. The *Edit* menu is shown in Figure 17.

Some frequently used commands have shortcuts, as indicated in the menus. As we see in the above *Edit* menu, we can use the *Select all* and *Deselect all* commands very quickly by pressing the functions keys F8 and F9, respectively. After deselecting all vertices, we select the two at the top by clicking them using the *right* mouse button. Alternatively, we can draw a dotted rectangle surrounding these two rectangles by moving the mouse while we hold the left button down. Then we use the *Move/Copy/Sweep* command of the *Transform* menu shown in Figure 18. The dialog box shown in Figure 19 now appears.

Since the height of the cube is 2, we can move the selected vertices down to the level $z = 0$ by adding −1 to their z-coordinates, as shown in Figure 19. Note also that the *Move* radio button is selected. (We will discuss *Copy* and *Sweep* later.) After clicking the OK button, the object has turned into a prism that is no longer a cube, as shown in Figure 20.

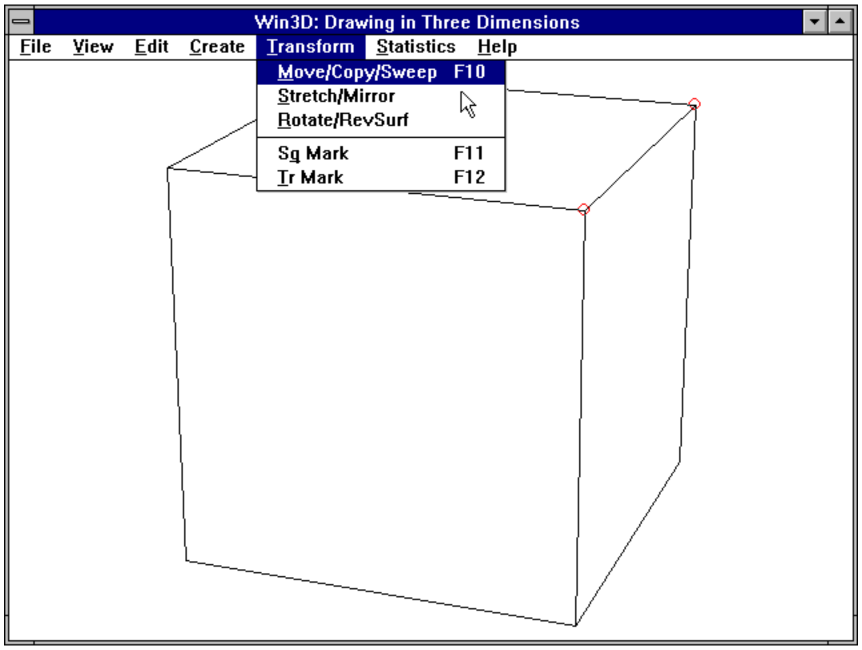

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Move/Copy/Sweep F10
Stretch/Mirror
Rotate/RevSurf
Sq Mark F11
Tr Mark F12

Figure 18. *Transform menu*

The figures we have seen so far in this User's Guide are in fact screen dumps, with lines of rather poor quality. We can do much better by using one of the commands *Print* and *Export HPGL*. Both commands can be found in the *File* menu, shown in Figure 21.

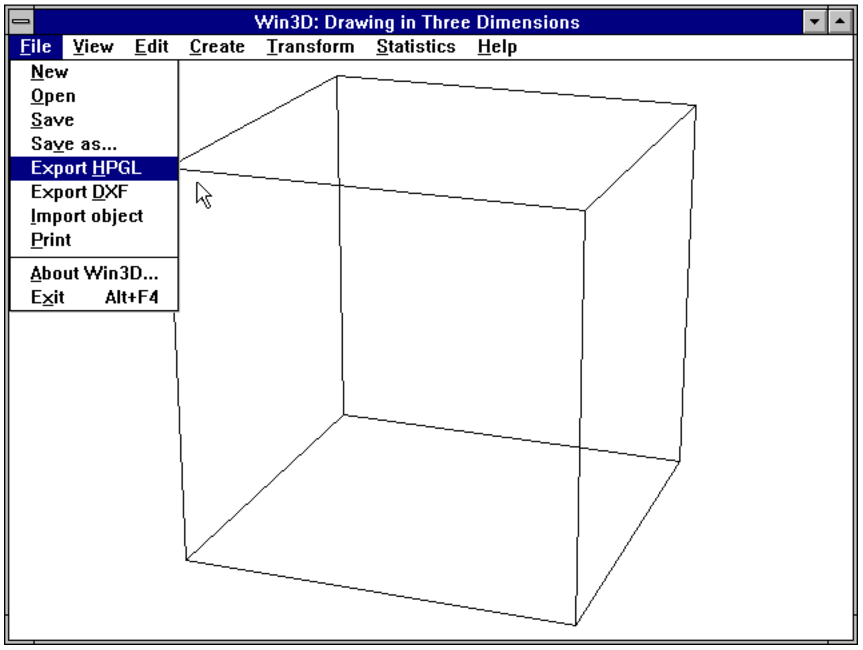


Figure 21. *File menu*

If we use these commands when in wire-frame mode, a wire-frame presentation is produced; in any other view mode we obtain a hidden-line image. The *Print* command prints directly, while the *Export HPGL* command produces a file in HPGL format, which can be imported into most text processors and DTP programs. Figure 22 was produced in the latter way:

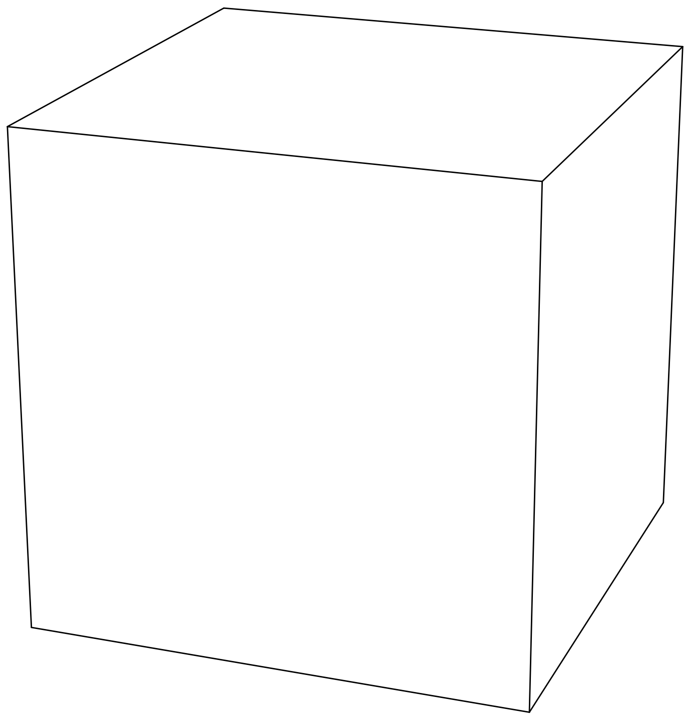


Figure 22. *Hidden lines removed*

On-line help

There remains only one item of the main menu to be discussed, namely the Help menu. This produces this very User's Guide that you are now reading!

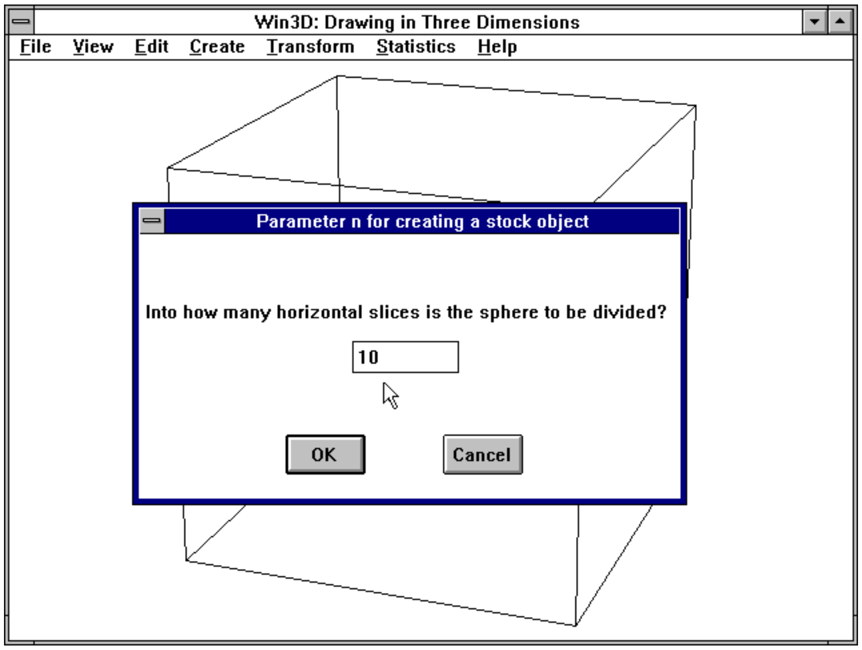
3

Point Selection and Groups

As we saw in Figure 17 when discussing the Edit menu, points can be *selected* and *deselected*. We also saw in Figure 20 that we can move selected points to deform an object. Selecting points is also useful to define *groups*, which are in fact parts of an object. Let us again create a cube by using *Create* | *Cube*. Immediately after using this command, we see that all eight vertices of the cube are surrounded by small red circles, which means that they are selected. We can now use the *Edit* | *New group* command (or press F5), to make a new *group* of the cube. The red circles then disappear, but the cube remains a group, which you can verify by using the *Edit* | *Select group* command (abbreviated F6). This command causes the red circles to reappear. You can let them disappear again by using *Edit* | *Deselect all* (or F9). Note that then all vertices of the cube still belong to a group, although they show no special marks. Thus, groups enable us to select set of points in a quick and convenient way.

Let us now create another standard object, say, a sphere, by using *Edit* | *Sphere*. Before constructing it, Win3D first inquires about the way we want the sphere to be approximated. This is done by means of the dialog box shown in Figure 24.

Instead of a real sphere, a set of flat faces is generated. We imagine a sphere divided into n horizontal slices of equal thickness, resulting in $n - 1$ horizontal circles on the sphere surface. There are also $2n$ vertical circles, all passing through the highest and the lowest point of the sphere, that is, through the two *poles*. After replacing the default value ($n = 6$) by $n = 10$, we obtain the sphere that, together with the cube created previously, is shown in Figure 25.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Parameter n for creating a stock object
OK
Cancel

Into how many horizontal slices is the sphere to be divided?

10

Figure 24. *Dialog box for creating a sphere*

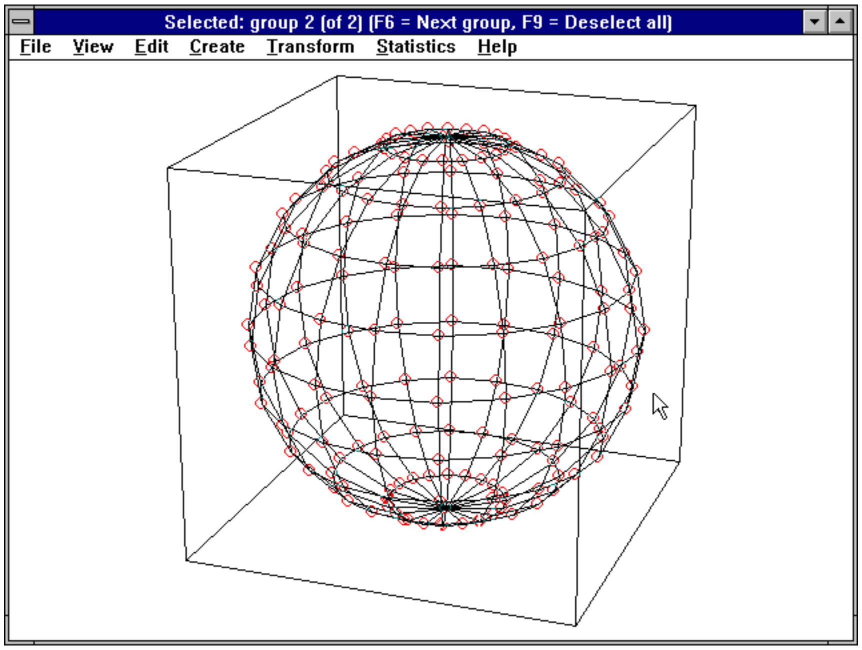

Selected: group 2 (of 2) (F6 = Next group, F9 = Deselect all)
File
View
Edit
Create
Transform
Statistics
Help

Figure 25. *Two groups*

Immediately after the (approximated) sphere is created, its vertices are selected, as you can see in Figure 25. By using the *Edit* | *New group* command (F5) again, these vertices form another group. The title bar temporarily indicates how many groups there are and which of them is currently selected. At this moment, it reads

Selected: group 2 (of 2) (F6 = Next group, F9 = Deselect all)

Apparently, the vertices of the approximated sphere belong to group 2. If we now press F6, the vertices of the cube become selected instead of those of the sphere; in other words, the small red circles on the sphere of Figure 25 disappear and eight of them reappear for the vertices of the cube. At the same time, the title bar changes into

Selected: group 1 (of 2) (F6 = Next group, F9 = Deselect all)

By pressing F6 once again, the situation of Figure 25 returns. If we use *Edit* | *Deselect all* (or F9), all red circles disappear, but there are still two groups. This is very convenient because it enables us to select either object (the cube or the sphere) very quickly. Depending on how often we have pressed F6 or F9, we can return to the situation of Figure 25 by pressing

F6 either once or twice. We do this to lift the sphere a little, placing it on top of the cube rather than inside it.

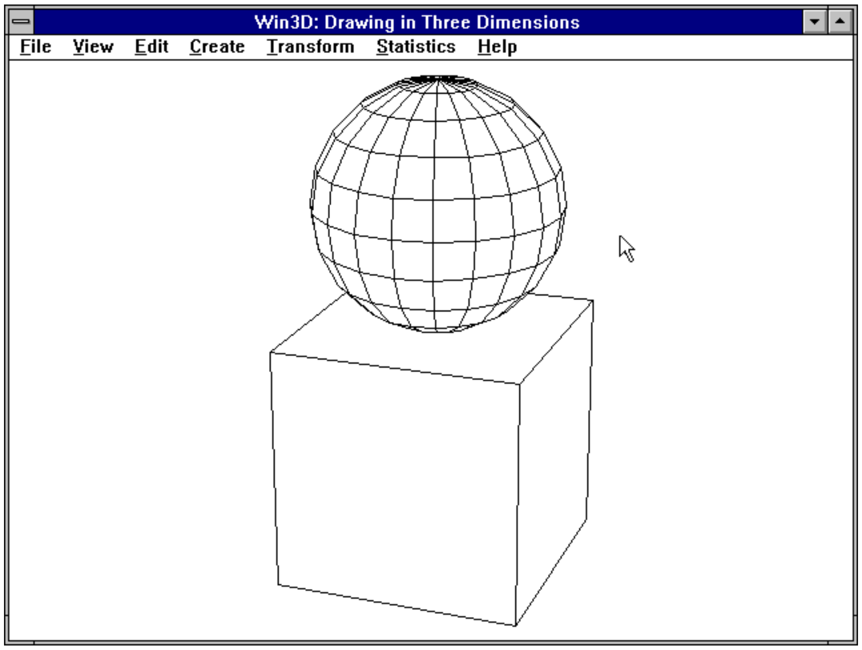

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 26. *Situation after increasing z-coordinates of sphere by 2*

Since the height of both the cube and the sphere is 2, the sphere is to be moved upward two units. We therefore use the command *Transform | Move/Copy/Sweep*, or F10 (see Figure 18). The dialog box shown in Figure 19 then appears, and we type 2 in the *z*-box, indicating that the *z*-coordinates of all selected points are to be increased by 2. If we then press F9 to remove the red circles and use the command *View* | *Hidden lines* (see Figure 9) we obtain the situation of Figure 26. Note that we cannot tell from Figure 26 alone that there are two groups. It will be useful to remember the following rules about selected points and groups:

1. A point can belong to one group or to no group at all; it cannot belong to more than one group at the same time.
2. If there are *n* groups ($n \geq 1$) they are numbered 1, 2, ..., *n*. Each time we press F6 the next group is selected (where 1 is considered the successor of *n*).
3. If all currently selected points (surrounded by red circles) form a group, we can dispose of these red circles by using *Edit* | *Deselect all* (or F9), without losing any information about groups.

4. If all currently selected points (showing red circles) do *not* form a group, it may be wise to make a group of them by using *Edit | New group* (or F5) before we dispose of these circles (see 3.).

The command *Edit | Ungroup* disposes of groups. If we use it in our example, the object seems to remain the same, but if we then press F6 to select the next group, the message *No groups selected* appears. In other words, we can no longer distinguish between the cube and the sphere in the convenient way used before. The *Ungroup* command should therefore be used with caution.

The command *Edit | Add group* (or Ctrl+F6) selects the next group, as does F6; however, with Ctrl+F6, the points of the currently selected group remain selected, while F6 causes that group to become deselected. We will now see how we can use Ctrl+F6 to combine several groups.

Combining groups

Suppose we have quite a few groups, say, 1, 2, 3, 4, 5, 6. We may now want to combine two or more of these groups. This can easily be done if these groups have consecutive numbers. Let us assume that groups 2, 3 and 4 are to be combined to one group. The new, combined group will have number 2 and the old groups 5 and 6 are renumbered 3 and 4:

```
1    2    3    4    5    6
--   --------------   --   --
↓         ↓         ↓    ↓

1         2         3    4
```

We begin by using *Edit* | *Select group* (or F6) as many times as is necessary to select group 2. We then use *Edit* | *Add group* (= Ctrl + F6) twice, so groups 3 and 4 are selected as well. The *Edit* | *New group* command (F5) makes a group of all points that are currently selected; in the present case this means that groups 2, 3 and 4 are combined into one group. Win3D takes care that group numbers remain consecutive: the new, combined group is assigned number 2 and the old groups 5 and 6 are automatically given the numbers 3 and 4, respectively. Things are slightly more complicated if the groups we want to be combined do not have consecutive numbers. But, fortunately, we can change group numbers so we can make them consecutive, as we will see now.

Changing group numbers

To change the number of a group, we press F6 until the vertices of that group, say group *n*, are selected. We then use the command *Edit* | *Change group numbers*. We are now requested to enter the new number, say, *k*, to be assigned to this group. Then group *n* becomes group *k* and any group numbers between *n* and *k* are shifted one position. To be more precise, we have:

> Group *n* becomes group *k*.
> If $k < n$, the numbers of the old groups k, $k + 1$, ..., $n - 1$ are increased by 1.
> If $n < k$, the numbers of the old groups $n + 1$, $n + 2$, ..., k are decreased by 1.

This may sound very complicated, but some examples will make clear what will happen. Suppose there are six groups A, B, C, D, E, F, currently numbered as follows:

> 1 = A2 = B3 = C4 = D5 = E 6 = F

We want to assign consecutive numbers to the groups B and E. We can do this by selecting group (*n* =) 5 and assigning the new group number (*k* =) 3 to it. Then the groups are numbered

> 1 = A2 = B3 = E 4 = C5 = D6 = F

The groups C, D and E (with old numbers 3, 4 and 5) now have the numbers 4, 5 and 3, respectively.

Things are simpler if *k* and *n* only differ by 1. In such cases the group numbers simply swap. For example, if we use *n* = 5 and *k* = 4 in the above case, we have:

Old situation:

1 = A 2 = B 3 = C 4 = D 5 = E 6 = F

New situation:

1 = A 2 = B 3 = C 4 = E 5 = D 6 = F

Copying objects

To illustrate what we have just been discussing, we will now generate an approximation of a cone, and we want this to be inserted between the cube and the sphere of Figure 26. We therefore use the *Pyramid/cone* command of the *Create* menu, shown in Figure 7. Instead of a circle, the base of our approximated cone will be a regular polygon. In a dialog box that then appears we have to specify how many vertices this polygon will have. Instead of the default value of 4 (which would generate a traditional pyramid, with a square base) we use 12; in other words, we approximate the circular base of a cone by a regular polygon with 12 vertices. Like the standard cube and sphere we have dealt with, the cone generated in this way has height 2, and the radius of its base is 1, but, unlike the cube, its base has zero *z*-coordinate. Consequently, the result is as shown in Figure 27.

Immediately after we create the cone, it is selected, as you can see in Figure 27. We make a group of it by pressing F5. Note that it is rather important to do this immediately: selecting points that do not form a group requires more work than selecting the points of a group. If we now repeatedly press F6, we see that the cube, the cone and the sphere have group numbers 1, 3 and 2, in that order. Since these three objects are eventually to appear in this order, we prefer 1, 2 and 3 as their respective group numbers. As we have seen, we can easily swap the group numbers 2 and 3 by selecting either group 2 or 3, using command *Edit* | *Change group numbers*, and supplying the desired group number (3 or 2, respectively) when requested.

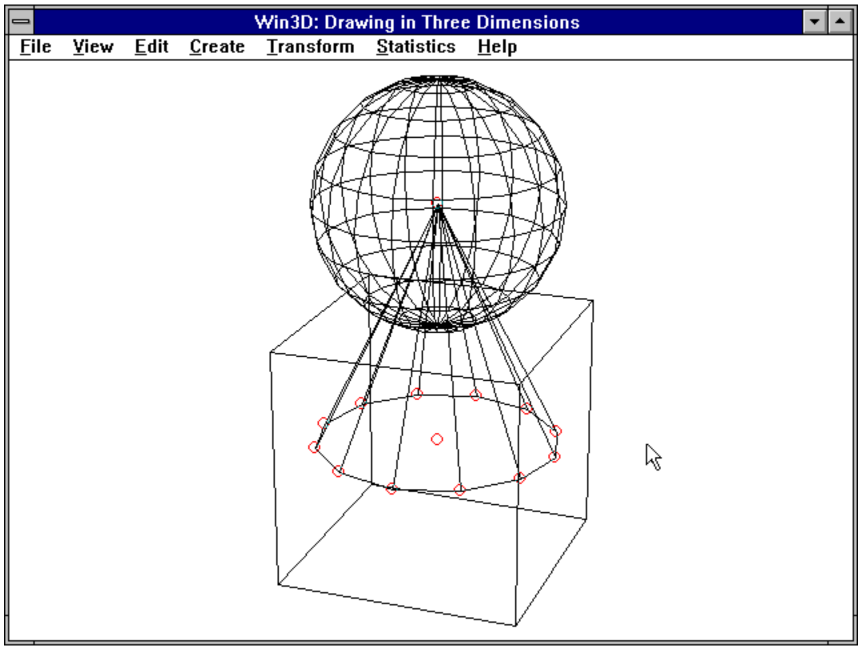


Figure 27. *Cone in initial position*

After this, we use F6, if necessary, to make the cone selected again, and, although it eventually has to move upward, we first move it downward by using *Transform* | *Move* (or F10) and entering −1 in the box for *z* (as we did in Figure 19). In this way, we make its top (also called its *apex*) touch the sphere.

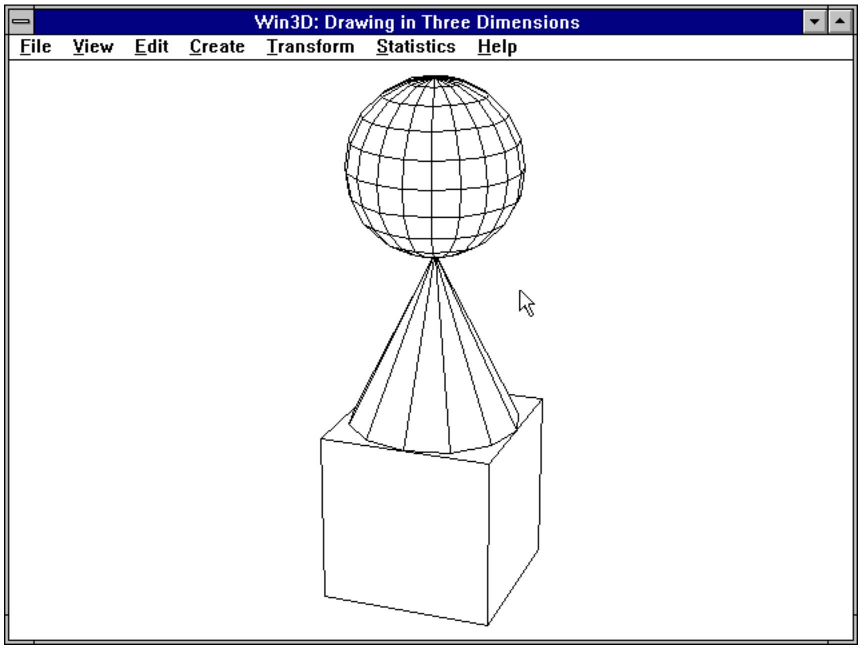

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

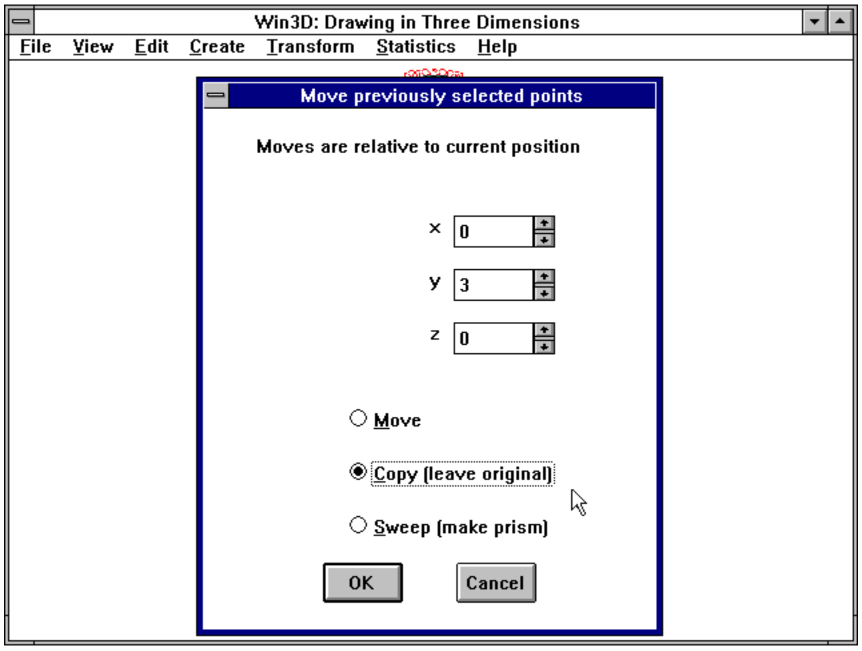
Figure 28. *Tower of cube, cone and sphere*

All we now have to do is move both the cone and the sphere, that is, group 2 and group 3, two units upward. We first press F6 as often as is required to select group 2; then we use Ctrl+F6 to add group 3 to the set of selected points. Then we press F10 and enter 2 in the box for *z*; after clicking OK, pressing F9 (to deselect all points) and using *View* | *Hidden lines*, we obtain the tower of Figure 28.

Although hidden-line and hidden-face representations look better, we often prefer a wire-frame model of an object while we are transforming it for reasons of efficiency. We use therefore *View* | *Wire frame* before we make a duplicate of it, which will be our next task.

The *Copy* radio button

Suppose that we want to copy the tower of Figure 28. This copying operation is much similar to moving. After pressing F8 to select all points, we use *Transform* | *Move/Copy/Sweep* (or F10) once again. So far, we have not changed the default setting *Move* of the radio buttons, but this time we click the *Copy* button, shown in Figure 29.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Move previously selected points

Moves are relative to current position

x 0

y 3

z 0

Move [✓ Copy (leave original)] Sweep (make prism)

Figure 29. Copy radio button selected

After entering 3 in the *y*-box and pressing the OK button of this dialog box, we obtain a copy of the original tower 3 units to the right. In other words, the only difference between *moving* and *copying* an object lies in what happens with the original. We lose it if we use *move*, but we keep it if we use *copy*. Note that this is similar with text processors. After removing the hidden lines, we obtain Figure 30.

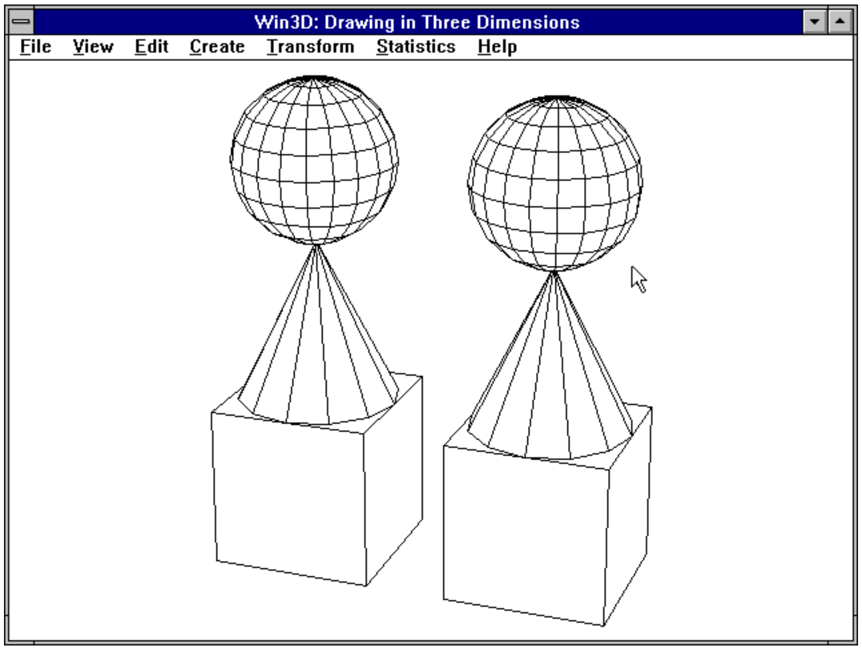

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 30. *Result of copying*

Other ways of selecting and deselecting

Now that we have so often selected the vertices of a group by pressing F6, we must not forget that we can also select and deselect points in two other ways. First, we can click them one by one, using the *right* mouse button, and, second, we can *drag* the *left* mouse button; when in grid-box mode we first have to select *Box select* from the *Edit* menu. As we move the cursor while we hold the left button down, a dotted rectangle is visible; one corner of this rectangle is the point where we pressed the mouse button down, and the other moves with the mouse cursor until we release this button. The dotted rectangle then disappears, but all vertices inside it change their select/deselect status: they become selected if they were not and vice versa. If we want the vertices selected in this way to become a new group, we can simply press F5.

Zoom in and zoom out

Sometimes two vertices lie so closely together that we have difficulty in telling them apart, which may be necessary to select them in one of the ways just discussed. The *View* | *Zoom in* command (with F2 as shortcut) will then be useful. After pressing F2, the cursor takes the form of a magnifying glass, and we define two opposite corners of a dotted rectangle, similar to what we did a moment ago to select points. This is shown in Figure 31.

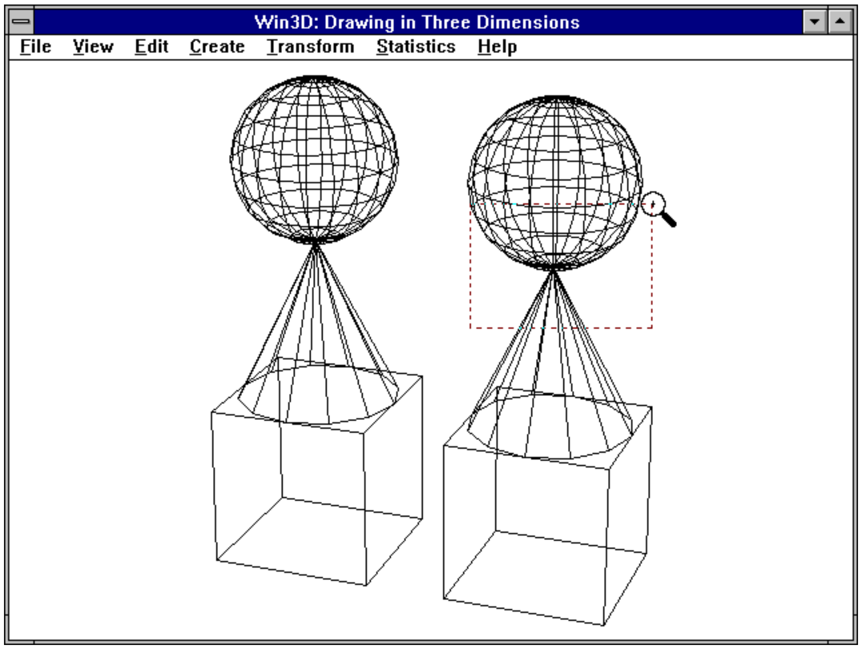

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 31. *Defining the area to be zoomed in*

This time, the rectangle defines the area that we want to magnify. Unless it is extremely small, it will approximately be magnified to the entire screen. Figure 32 was obtained in this way. It shows the fragment that in Figure 31 lies inside the dotted rectangle.

To revert to the original screen we use *View* | *Zoom out* (or its shortcut, F3), as indicated in the title bar. Zooming may change the mode. If we are using the wire-frame mode, the grid-box mode or the Painter's algorithm, the mode remains the same; otherwise (that is, while in hidden-line or hidden-face mode), there is an automatic change to wire-frame mode.

Immediately after copying an object, the copy is selected, so we could have made a group of it by pressing F5. In our example we would then have had four groups altogether: three for the left, original tower and only one for the right, copied one.

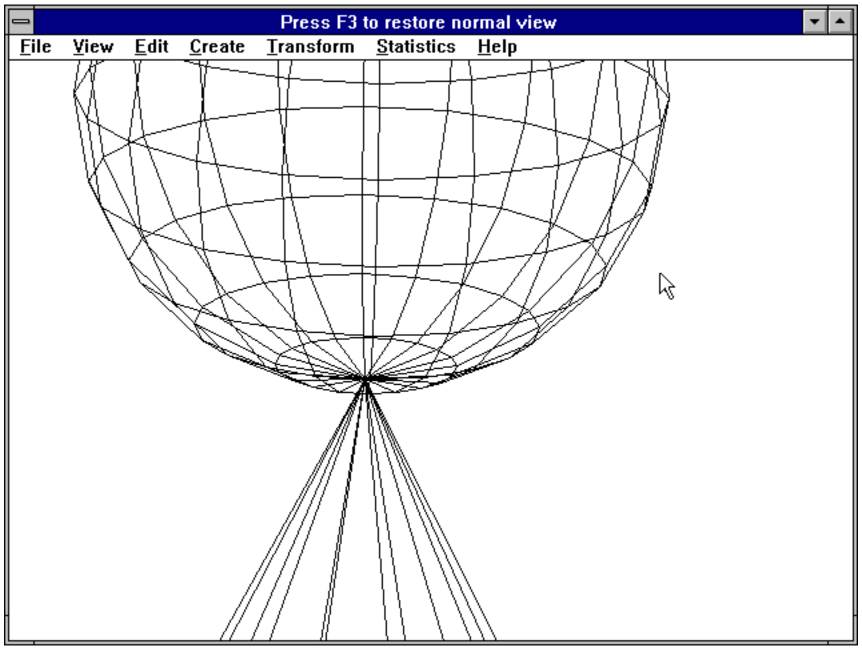

Press F3 to restore normal view
File View Edit Create Transform Statistics Help

Figure 32. *Result of zooming in*

Now that we have not done this, we can select the tower on the right in another way, discussed above: we press the left mouse button when the cursor is at the bottom of the screen between the two towers; then we move the cursor to the upper right corner of the window while we hold the mouse button down. All vertices lying in the dotted rectangle, displayed while we are doing this, are then selected.

The *Delete* and the *Undo* commands

When the tower on the right (or any other set of points) is selected, we can use the command *Edit* | *Delete* (or press the Del key) to let it disappear. If, immediately after doing such a rigorous action, we regret it, we can use the *Edit* | *Undo* command (or press the Esc key) to restore the old situation.

The *stretch* transformation

We have frequently been dealing with cubes, and you may think this rather unpractical because very few objects in real life are cubes. However, we can transform a cube into many other shapes, as we saw already in Figure 20. The *Transform* | *Stretch/Mirror* command is another way of transforming objects. We will use it to transform the cube at the bottom of our tower into a different rectangular prism: we will reduce its height to 25% of its original value, and at the same time we will stretch its length and its width by 300%. This information is not sufficient. We must also specify the *fixed point* of the stretch transformation; this point will remain at its current position. The command to be used for this purpose is *Transform* | *Sq Mark* (see Figure 18). We first select the cube (by pressing F6), and then use the command just mentioned. This changes the cursor into a rather large arrow, pointing upward, as shown in Figure 33.

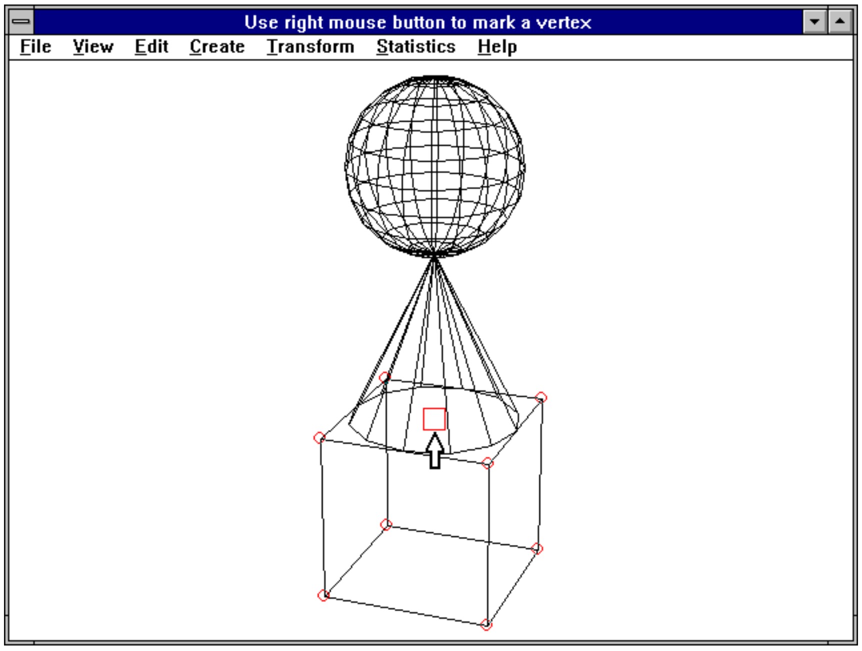
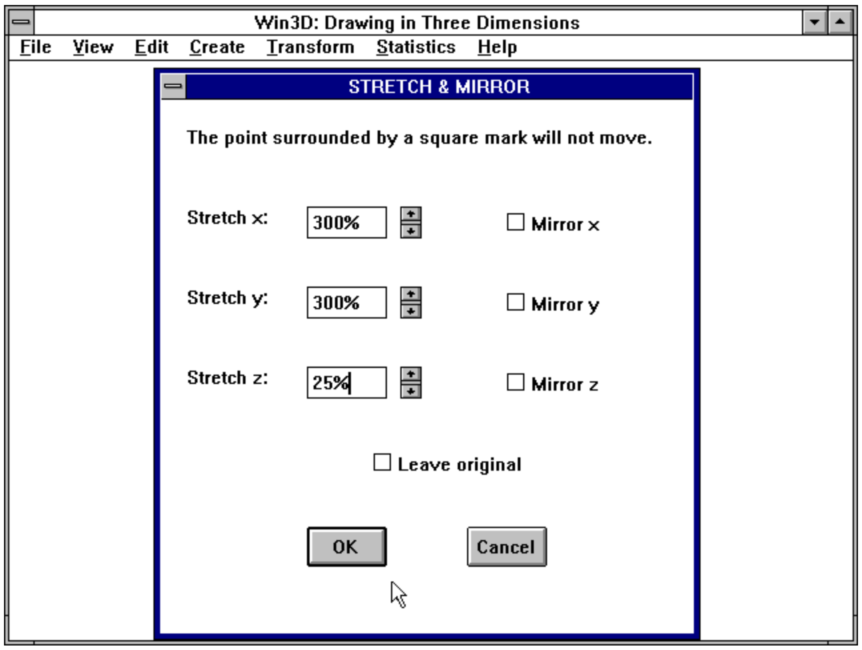


Figure 33. *Position of fixed point, indicated by a square mark*

As you can see, the vertical arrow points to the center of the top face of the cube. This is really an existing vertex, belonging to the cone and representing the center of its base. If we press the *right* mouse button at this moment, a square mark appears, which surrounds this fixed point. Note that the title bar indicates that we should use the right button for this purpose. For each selected point, its horizontal distance to the fixed point will be multiplied by a factor 3, and its vertical distance by a factor 0.25. We specify this in the dialog box which now appears on the screen and is shown in Figure 34.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
STRETCH & MIRROR
OK
Cancel

The point surrounded by a square mark will not move.

Stretch x: 300% ☐ Mirror x

Stretch y: 300% ☐ Mirror y

Stretch z: 25% ☐ Mirror z

☐ Leave original

Figure 34. *Dialog box to specify stretch factors*

After clicking the OK button and using *View* | *Hidden lines*, we obtain the result of Figure 35.

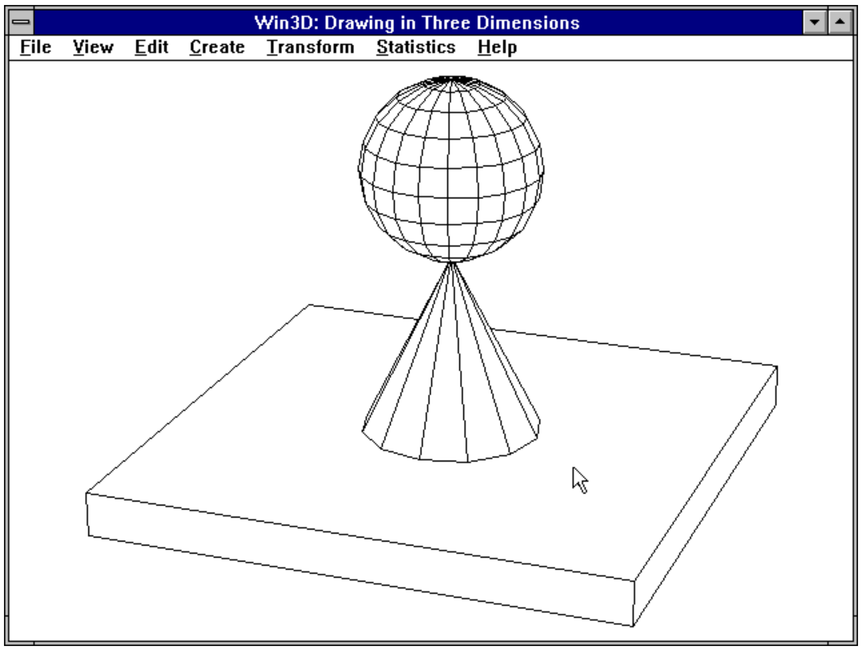
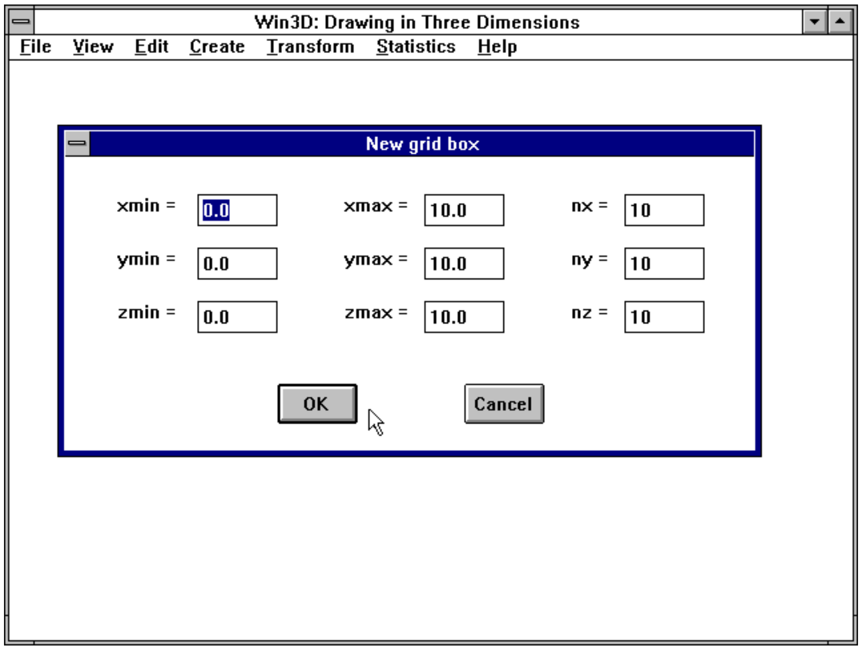


Figure 35. *Result of stretch operations applied to cube*

4

Building Simple Objects

Win3D enables us to define new vertices and faces of an object interactively. Instead of using a predefined object (by means of either *Create* or *File* | *Open*) we can start from scratch by using the *File* | *New* command. The dialog box shown in Figure 36 then appears.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
New grid box
OK
Cancel

xmin = 0.0
xmax = 10.0
nx = 10
ymin = 0.0
ymax = 10.0
ny = 10
zmin = 0.0
zmax = 10.0
nz = 10

Figure 36. *Grid-box dialog*

It shows default minimum and maximum values for *x*, *y* and *z* and values *nx*, *ny* and *nz*, which are numbers of intervals (or steps) to define a grid. If we do not alter these default values (which are 0 and 10 for the minimum and maximum values and 10 for the numbers of segments), we obtain the *grid box* shown in Figure 37.

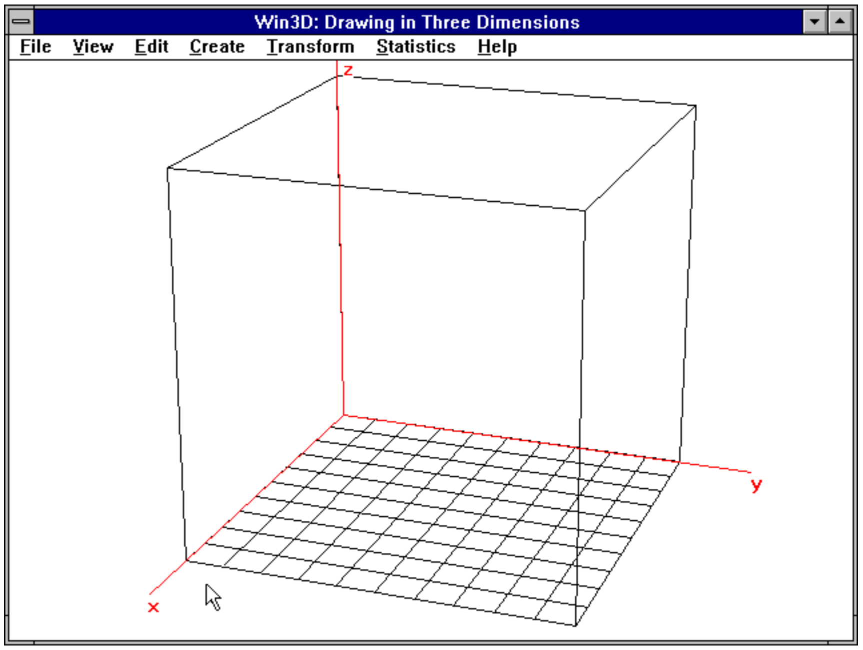

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Figure 37. *New grid box*

Because we have not changed the default values *nx* = *ny* = 10, there are ten intervals both in the *x*- and in the *y*-directions. Consequently, there are 11 × 11 = 121 points of intersection in the *xy*-plane; we call these points *base grid points*. Since we also have *nz* = 10, we divide each vertical line through these grid points and lying between the grid box into ten equal segments. This gives ten other, so-called *derived* grid points for each of the 121 base grid points. Each derived grid point has the same *x*- and *y*-coordinates as its base grid point, but a different *z*-coordinate. Base grid points have zero *z*-coordinate.

Thus, although not all visible in Figure 37, there are altogether 121 + 121 × 10 = 1331 grid points in this example. We can define each of these points, and make them visible as small blue dots. For the 121 base grid points, we do this simply by clicking them, using the left mouse button. To define a derived grid point, we begin by pressing the left mouse button while the cursor is located at its base grid point; we then move the mouse upward while holding the left button down. As we do this the current *z*-coordinate of the point we are moving is shown in the title bar. Actually, this point moves in discrete steps, determined by the values *zmin*, *zmax* and *nz* of the dialog box we have used (see Figure 36): in our example only the *z*-values 0, 1, 2, ..., 10 are possible. When we release this button, this *z*-value is used. In other words, we define the nearest grid point on the vertical line through the base grid point we selected when we pressed the left mouse button down.

Defined points are useful to define the bounding faces of three-dimensional objects. We will illustrate this by constructing the object shown in Figure 38, a prism with a large letter U as its base. Its dimensions are 10 × 10 × 10, so it just fits in our grid box. Its thickness is 3; in other words, the width of the hollow space in the middle is equal to 10 − (3 + 3) = 4 and its height is 10 − 3 = 7. (If we had wanted to use other dimensions, we could use either a different grid box or the *Stretch* command, as discussed at the end of the last chapter.)

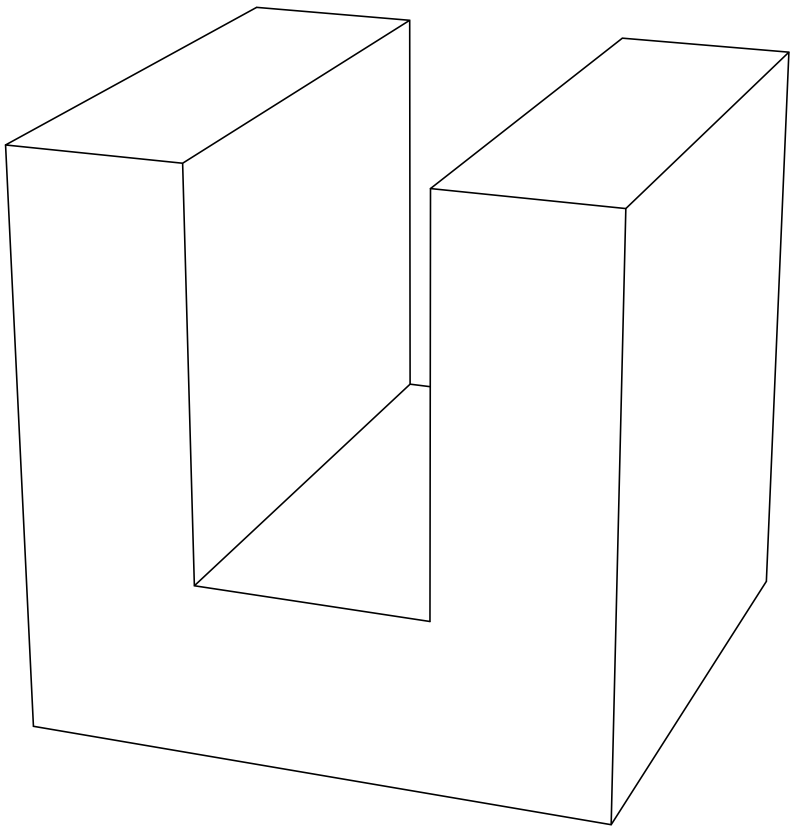


Figure 38. *Large letter U*

Starting with the grid box of Figure 37, we now define eight vertices of the front face by using the left mouse button, as we have discussed. These points are shown in Figure 39; in view of our discussion, their numbers are displayed here by selecting the points and using *Edit* | *Show nr.*

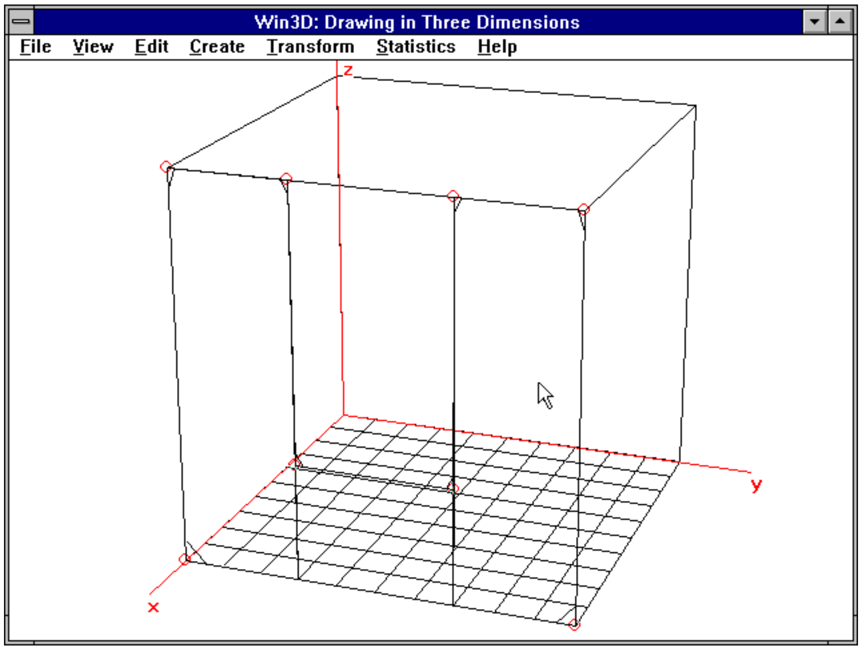

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Figure 39. *Vertices of front face*

Inserting a face

To specify that these eight points should be the vertices of a face, we must follow the edges one by one, and click each vertex as we encounter it. In other words, the order

1 7 8 6 4 3 5 2

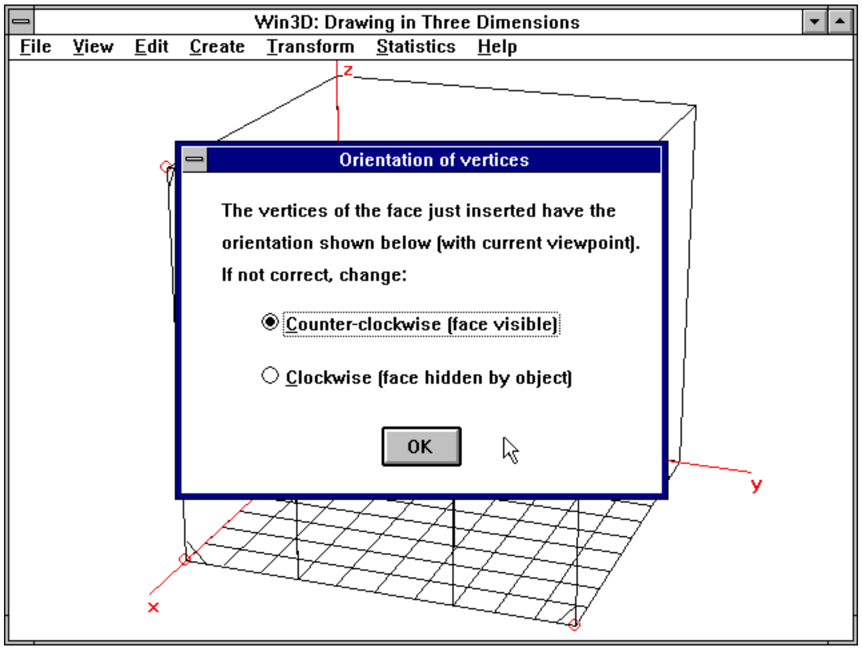
is correct, but it would be wrong to select these vertices in this order:

1 2 3 4 5 6 7 8

Also, the angle at the second selected vertex must be *convex*, that is, it must be less than 180°. As you can see in Figure 39, the angles at the vertices 3 and 4 are not convex. All others are.

Now that we have selected the vertices in the above correct order (so that they are surrounded by red circles), we will use the command *Edit* | *Insert face*, or simply press the Ins key. This causes the dialog box of Figure 40 to appear.

Internally, the vertex numbers of a face are stored in counter-clockwise order. Since the order 1 7 8 6 4 3 5 2 in which the vertices were selected is counter-clockwise, we can simply click the OK button.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x
Orientation of vertices

The vertices of the face just inserted have the orientation shown below (with current viewpoint). If not correct, change:

[✓ Counter-clockwise (face visible)]

Clockwise (face hidden by object)

Figure 40. *Dialog box to check orientation*

If we had instead selected them in the reverse, clockwise order, 2 5 3 4 6 8 7 1, we would now have been able to correct this by clicking the counter-clockwise radio button.

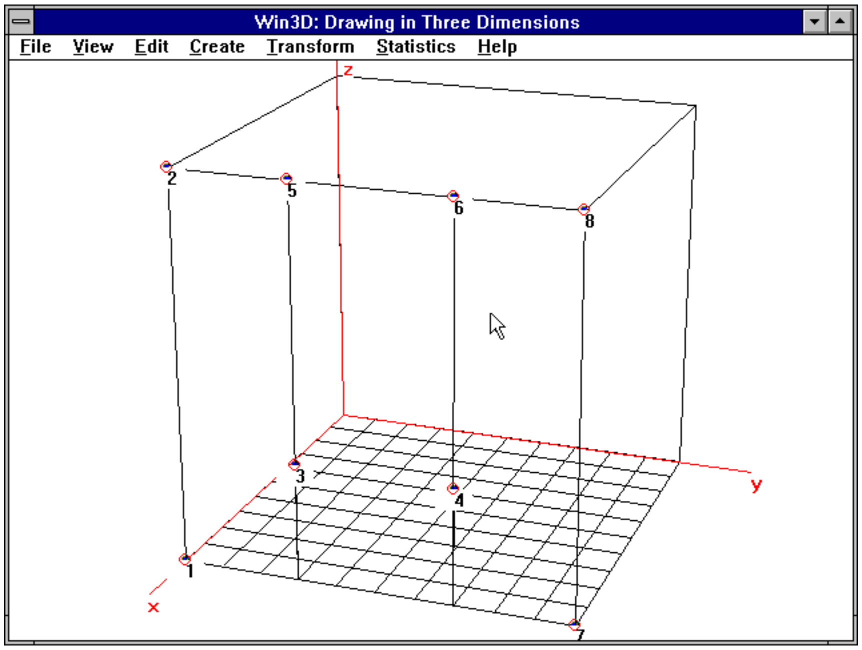
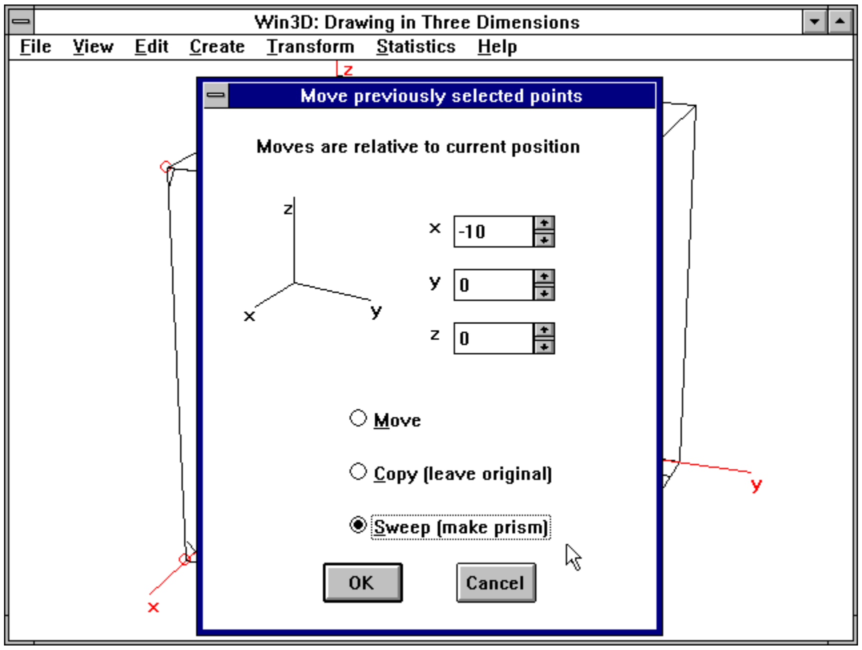

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x
1
2
3
4
5
6
7
8

Figure 41. *Front face inserted*

Remember, the face in question is a bounding face of a three-dimensional object. The order of the vertices should be counter-clockwise when this face is viewed *from outside the object*. (If our current viewpoint is such that the face is a back face, in reality hidden by the object, the order must be clockwise!) After we click the OK button, we can see that a face has been generated, as Figure 41 shows.

There are little triangles drawn at the corners, just to indicate that there is really a face (not only a set of vertices and line segments). This is only one face of the object of Figure 38, and there are altogether ten bounding faces, so you might expect that most of the work still has to be done. Fortunately, we can generate all nine other faces automatically, using the very powerful *sweep* command. When we do this, the vertices of the front face must be selected, but they still are. The command just mentioned is given by using *Transform* | *Move/Copy/Sweep*, and clicking the *Sweep* radio button in the corresponding dialog box, shown in Figure 42.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Move previously selected points
OK
Cancel
z
x
y

Moves are relative to current position

x -10
y 0
z 0

○ Move
○ Copy (leave original)
● Sweep (make prism)

Figure 42. *Clicking the Sweep radio button*

We must also enter the value −10 in the box for *x* and leave the *y*- and *z*-boxes zero. This means that for each vertex of the front face the *x*-coordinate is decreased by 10 and the *y*- and *z*-coordinates are left unchanged to obtain the corresponding back face. This would also be the case if, instead of *Sweep*, we had selected the *Copy* radio button. The difference is that the *Sweep* operation (also known as *extrusion*) automatically inserts all the rectangular faces that connect the front and the back faces. Also, the back face is given an orientation opposite to that of the front face.

Clicking the OK button of the dialog box in Figure 42 and using the command *View* | *Hidden lines* is all we have to do to obtain Figure 43. Note the difference with Figure 38, which was made by using command *File* | *Export HPGL* (see Figure 21).

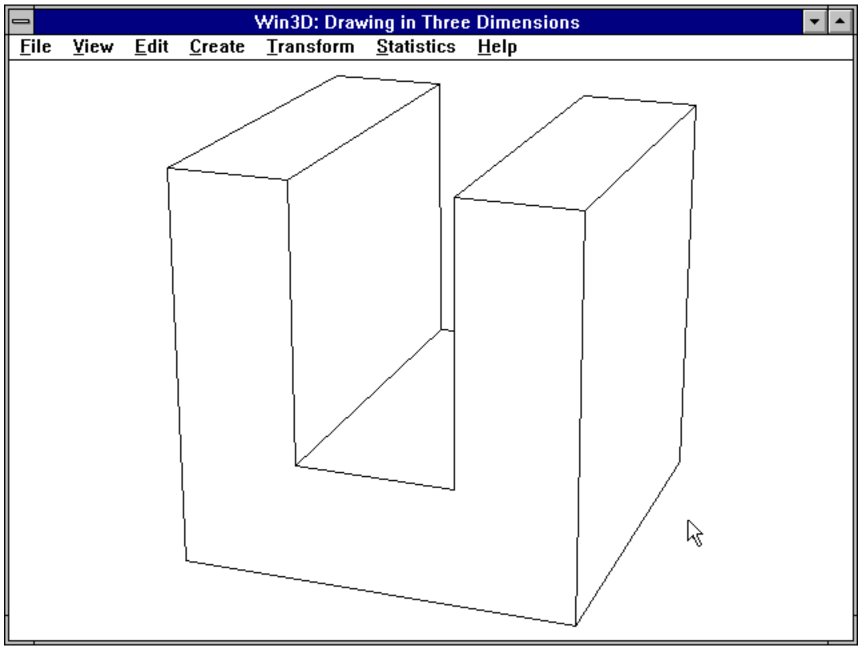

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 43. *Result of sweep command*

Rotations

We have seen that besides simple *move* operations, the related but more interesting *copy* and *sweep* operations are available. The same principle applies to rotations. Let us first see how to use simple rotations. Suppose, for example, that we want to turn the object of Figure 43 through, say, 30°, about the edge at the bottom left. We prepare for this rotation by two actions:

a. We select all vertices (by pressing F8), to indicate the points that are to be subjected to the rotation.
b. We specify the axis of rotation, which can be any line (and need not necessarily be horizontal or vertical).

Instead of using an undirected axis, we prefer using a *vector*, or *arrow*, lying on this line. Its length is irrelevant, but its direction provides an elegant way of specifying what we mean by saying that we want to perform a rotation through 30° about this vector. Associating this vector with a normal, right-handed screw, the rotation has the same orientation as turning the screw while it moves a little forward in the direction of the vector. Using the *Transform* menu (shown in Figure 18), we put a square mark around the tail of the arrow and a triangle mark around its head. We do this by using the commands *Sq Mark* and *Tr Mark*, respectively, and by clicking the end points of the edge in question, using the right mouse button. Figure 44 shows these two marks. Since the triangle lies in the back, the arrow is directed to the back, which means that a rotation through 30° is clockwise; in other words, most of the object will move down.

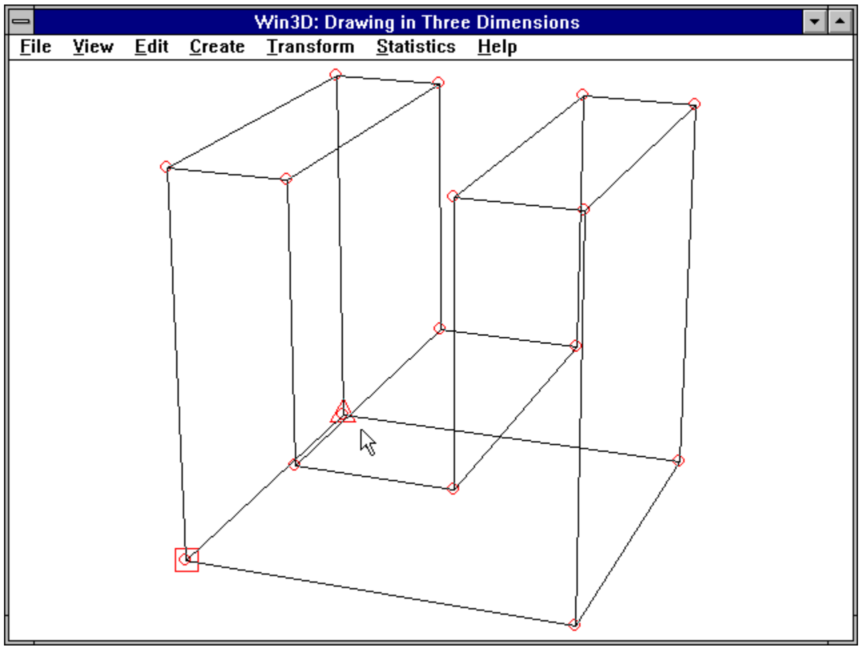
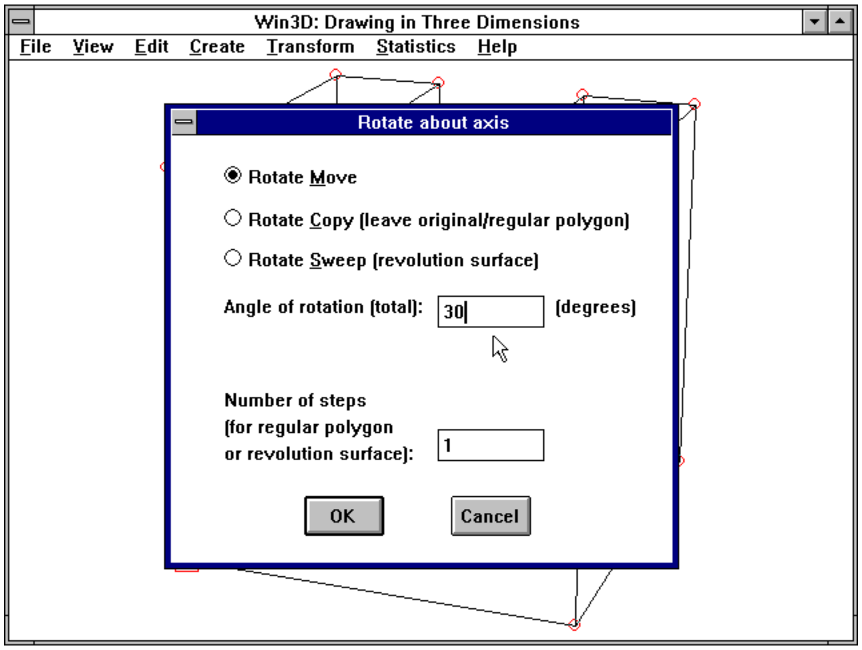


Figure 44. *Directed axis of rotation*

After the above preparations (a. and b.), we now use command *Transform* | *Rotate/RevSurf*, which causes the dialog box of Figure 45 to appear. We enter 30 in the *angle* box and click its OK button. The result, in hidden-line representation (and with all marks removed by pressing F9), is shown in Figure 46.



Win3D: Drawing in Three Dimensions

Rotate about axis

(● Rotate Move)

○ Rotate Copy (leave original/regular polygon)

○ Rotate Sweep (revolution surface)

Angle of rotation (total): 30 (degrees)

Number of steps (for regular polygon or revolution surface): 1

Figure 45. *Rotate dialog box*

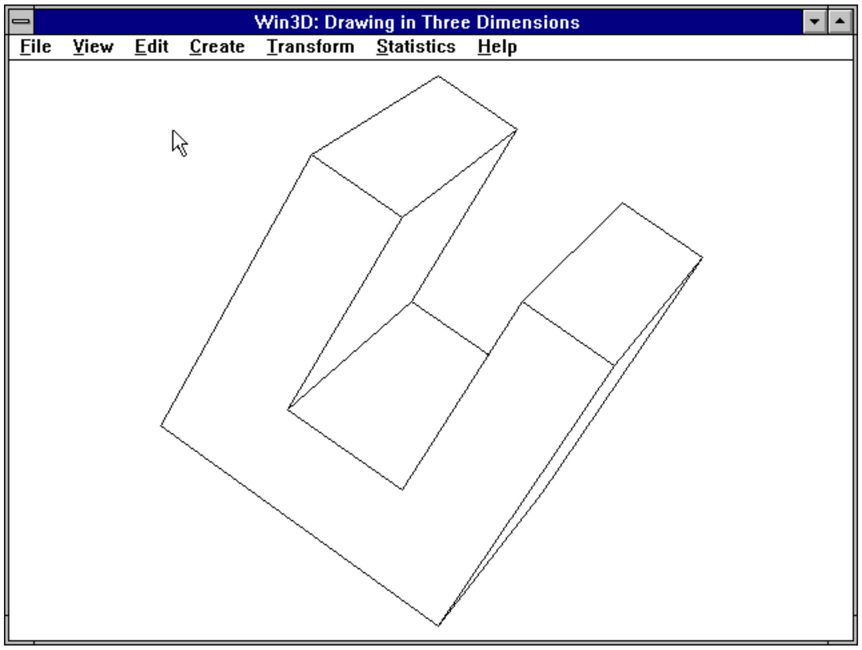

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 46. *Rotated object*

Rotate and copy

In the above example, we have rotated a complete three-dimensional object. It is also possible to rotate only part of it, in the same way as with the *move* command; we demonstrated this in Chapter 1 to obtain the deformed cube of Figure 20. We will apply this principle now, but we will combine this with another new aspect: instead of a *rotate/move*, we will now use a *rotate/copy*. This example is more interesting than those we have seen so far. It is shown in Figure 47.

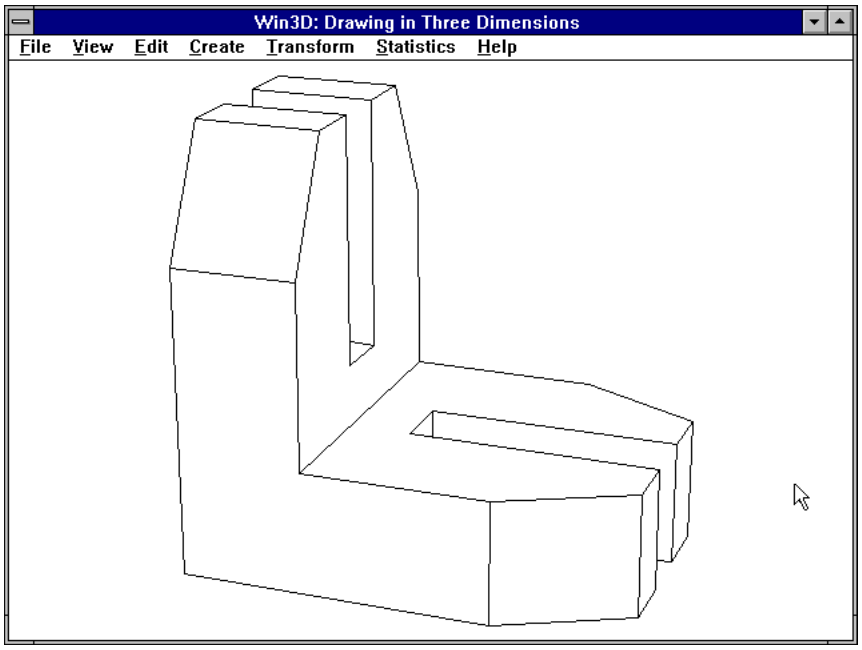
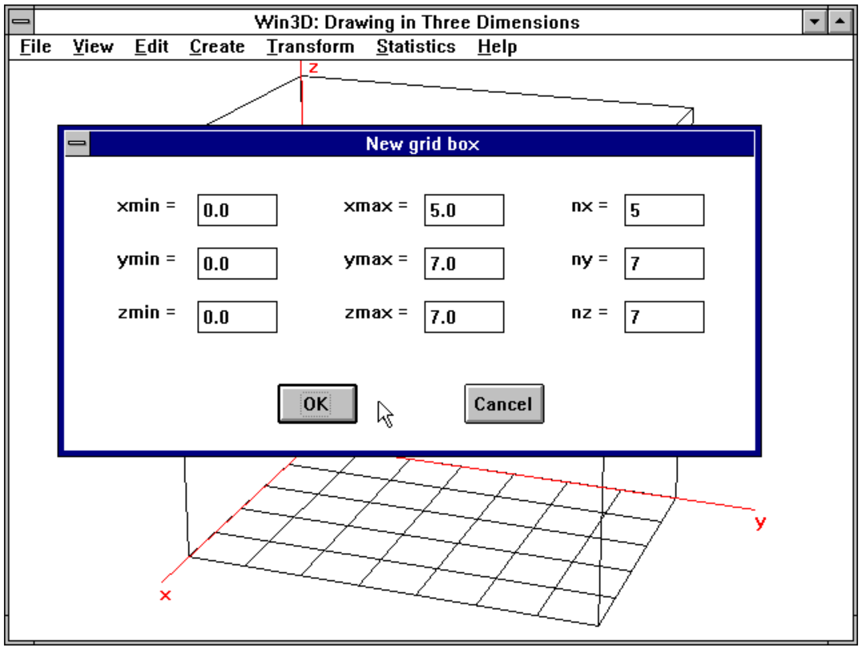

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 47. *L-shaped object*


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
New grid box
OK
Cancel
z
y
x

xmin = 0.0 xmax = 5.0 nx = 5
ymin = 0.0 ymax = 7.0 ny = 7
zmin = 0.0 zmax = 7.0 nz = 7

Figure 48. *Grid box dialog box*

This object was created by means of the grid box shown in Figure 48. In this box *x* ranges from 0 to 5 and both *y* and *z* range from 0 to 7, all with grid intervals of length 1 (which follows from *nx* = *xmax* − *xmin* = 5 − 0 = 5 etc.).

The bottom face (in the *xy*-plane) was constructed in the usual way. Obviously, the grid box was chosen in such a way that all vertices of the object are grid points. Note that for this bottom face the order in which we select the vertices, before using the *Insert face* command, must be clockwise because we view the face from the inside.

Show orientation and *Select face*

Fortunately, if we select the vertices in the wrong order, we can easily change the orientation by using *View* | *Show orientation.* This command not only *shows* the orientation of a face, but it also offers the dialog box with two radio buttons we have seen in Figure 40. We can therefore quickly change the orientation, and this applies not only to a face with few vertices. If there are many, say 50, we need not select all these vertices to specify which face is meant. It is sufficient to select as many vertices as are required to determine the face in a unique way. In our example, once the bottom face has been inserted, we need to select only three of its vertices to be able to apply command *Show orientation* to it. If we select only two, there are two faces that share these vertices; Win3D will indicate this by means of a message box. (Note: the command does not work either if not all the points selected lie in the same plane.)

The same principle applies to command *Edit* | *Select face*. This does not generate a dialog box, but simply selects all vertices of the face uniquely determined by some of these vertices, which we have previously selected. Both commands are very useful for this example because the bottom and the left face have as many as 10 vertices each (and there are two other, equally complex faces as well).

Rotating a face

After defining the bottom face, we can use this to generate the left face by rotating it through 90° about the left edge. We must then use the *Show orientation* command to alter the orientation of this left face.

The same principle can be applied to the two other, slightly smaller faces of ten vertices: after we have defined one of them, we can derive the other from it by a *rotate/copy* command. Actually, after we have defined the bottom face, we can derive the similar horizontal face, visible in Figure 47, from it. We can first copy the bottom face and place the copy two units above the bottom face. Then we move its two vertices on the left two units to the right. Then, as we have just discussed, we create the identical, vertical face by using the *rotate/copy* command, and we apply the *Show orientation* each time to make the orientation of the faces counter-clockwise, when viewed from outside the object.

There are two possible stumbling blocks in this example. First, the grid-box mode will become confusing when the number of faces increase. It is then wise to switch to the wire-frame mode. Second, rotating a face about one of its edges, as discussed above, will cause the end points of this edge to appear twice in the result. In other words, there will be two pairs of coinciding vertices, which may be confusing. The command *Unify coinciding vertices* solves this problem. This command will be discussed in Chapter 6. Alternatively, instead of rotating a *face*, we can rotate/copy its *vertices* except for those that we give square and triangular marks, and use the *Insert face* command after this rotation. This avoids the introduction of coinciding vertices. With either method, there will be 36 (not 40) vertices in the final result.

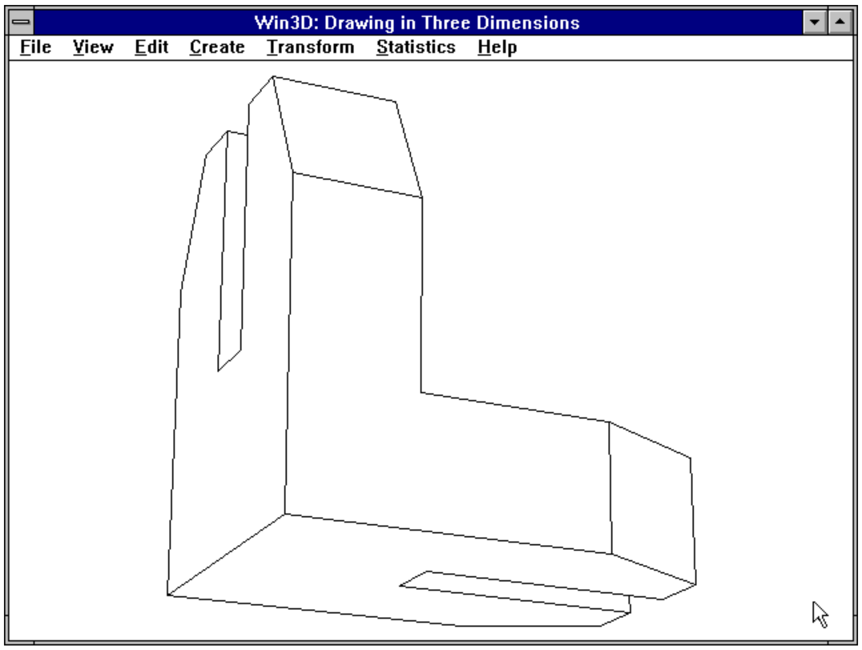

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 49. *View with $\theta = -25^\circ$ and $\varphi = 110^\circ$*

Having done all this, we are by no means ready. There are as many as 14 rectangular faces and two L-shaped faces to be inserted, so altogether there are 4 + 14 + 2 = 20 bounding faces. (You can check this in the final result by using *Statistics* | *Problem size*.) The grid box makes it reasonably convenient to insert all the remaining faces. We must be very careful with the orientation. If they are incorrect, the wire-frame model will look the same, but the hidden-lines and hidden-faces representations will be wrong. We can check the final result by switching to the hidden-line mode (using *View* | *Hidden lines*) and by using the commands *View* | *Viewpoint* several times to view the object also from different viewpoints. An example of this is Figure 49.

5

Solids of Revolution and Holes in Objects

Solids of revolution

Like the *Move* dialog box (see Figure 19), the *Rotate* dialog box, shown in Figure 45, has three radio buttons for *Rotate Move*, *Rotate Copy* and *Rotate Sweep*. Having dealt with the first and the second, we will now see how to use the third, *Rotate Sweep*. We use it to make solids of revolution, such as the one shown in Figure 50.

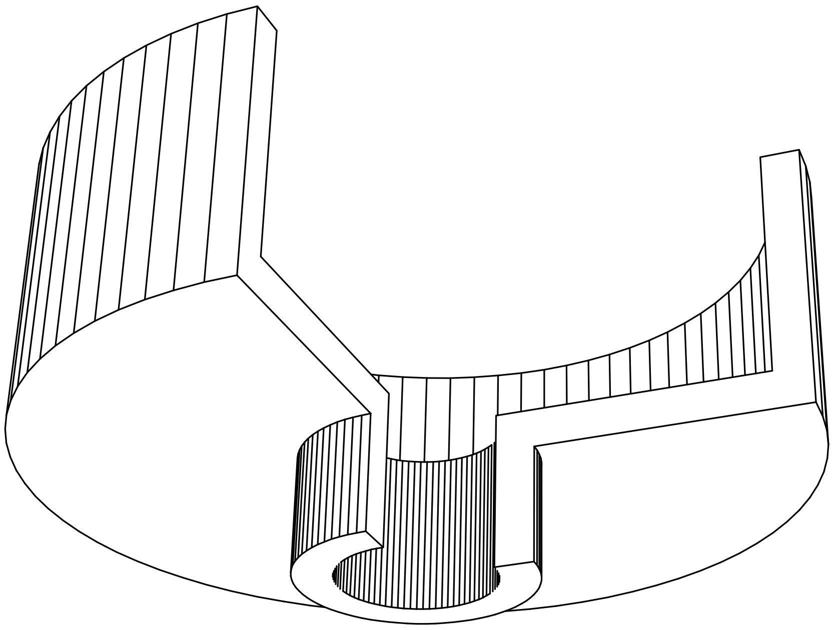


Figure 50. *Solid of revolution (270˚)*

Building this object with Win3D is surprisingly simple. We begin by constructing the Z-shaped polygon, shown in Figure 51. As in Figure 44, we place a square and a triangular mark around two points, which will be the tail and the head, respectively, of a vector on the axis of rotation. It is essential that this axis lies in the same plane as the Z-shaped polygon; apart from this restriction, we can use any polygon and any axis.

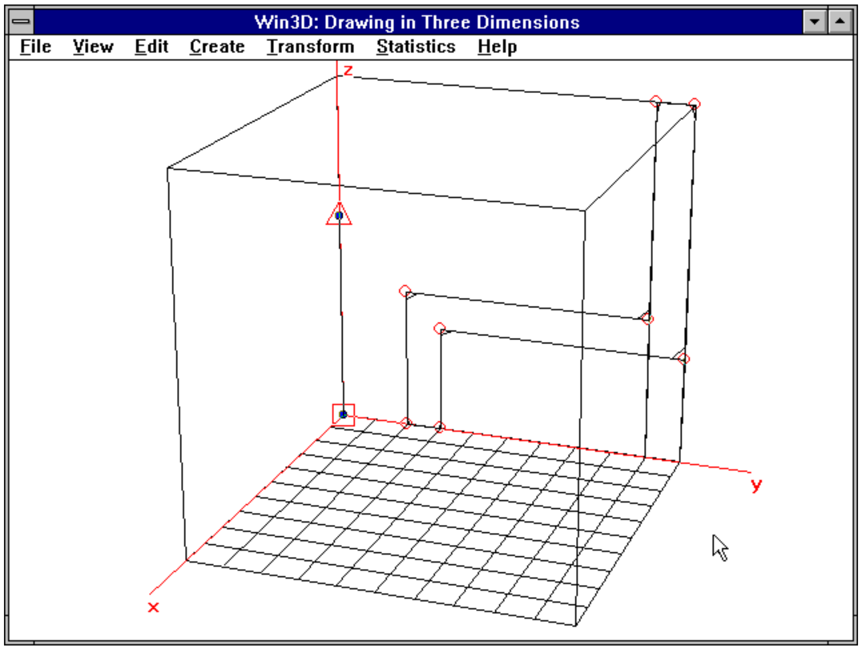
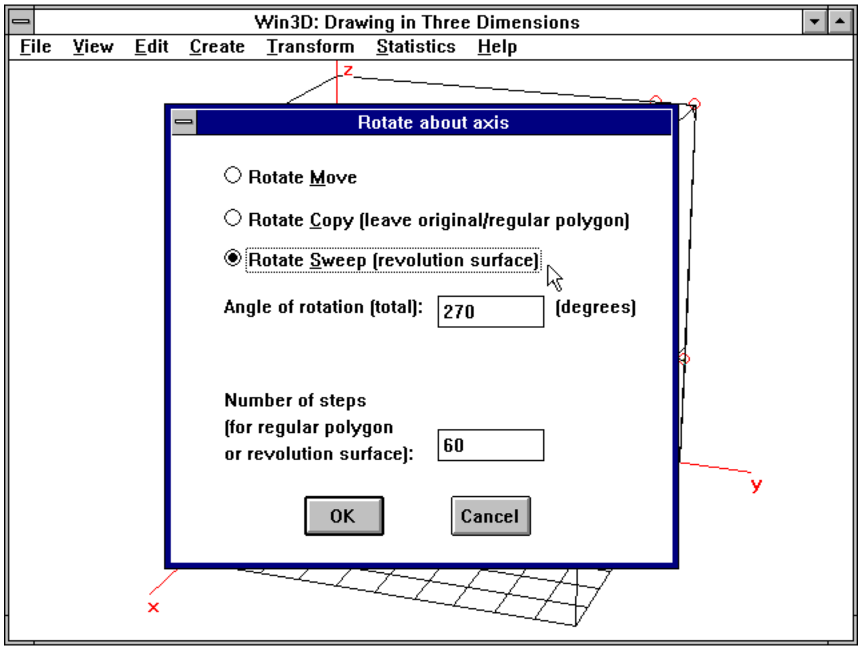


Figure 51. *Z-shaped polygon in grid box*

We now use the command *Transform* | *Rotate/RevSurf*, which causes the dialog box of Figure 52 to appear.

If we had wanted a full rotation, we should have entered an angle of 360°, so we enter 270 to obtain three-quarters of a rotation. Rather arbitrarily, the value 60 is used here for the number of steps. Remember, instead of curved surfaces, we actually use a great number of flat faces. This is related to the fact that in a circle the arc of a sector can be approximated by a straight line if the angle of that sector is very small. In our application this angle is equal to 270/60 = 4.5°. Clicking the OK button of Figure 52 is all we need to do to generate the object of revolution in question. Since we are in the grid-box mode, the result may at first look weird, as Figure 53 shows.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x
OK
Cancel

Rotate about axis

( ) Rotate Move
( ) Rotate Copy (leave original/regular polygon)
[✓ Rotate Sweep (revolution surface)]

Angle of rotation (total): 270 (degrees)

Number of steps (for regular polygon or revolution surface): 60

Figure 52. *Rotate dialog box, used for Rotate/Sweep*

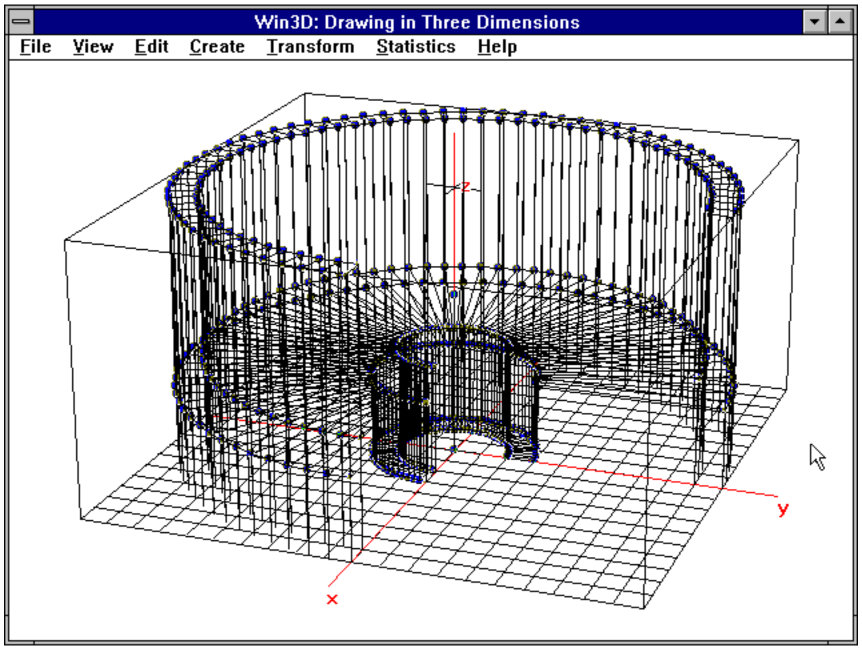

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Figure 53. *Solid of revolution in grid box*

Starting at the given polygon, each next one is obtained by rotating the previous one through the small angle just mentioned, and the orientation is related to the rotation vector (pointing upward) in the same way as the turning of a screw is related to the way this screw moves along its axis.

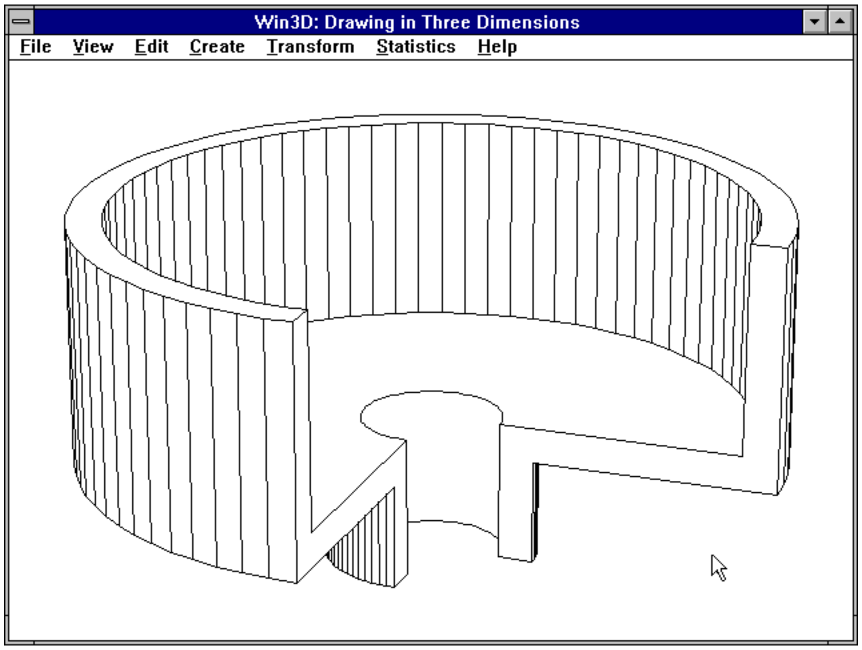

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 54. *Solid of revolutions, hidden lines removed*

The result is much better if we use *View* | *Hidden lines*, as Figure 54 shows. This view was obtained with the default values of the viewing angles θ = 20° and φ = 70°. If we want to see the bottom of this object, we can use, for example, φ = 120°, which gives the image of Figure 50.

File format

The file format used by the *File* | *Save* command is very simple. A file produced in this way is in ASCII code and contains at most three sections:

1. A *vertex section*, which is a set of lines of text of the form

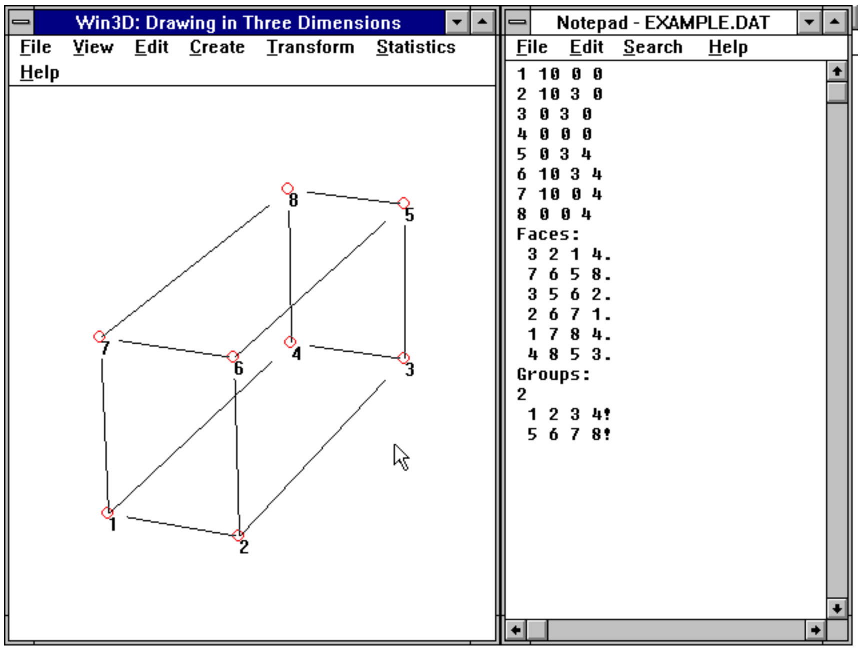
   nr *x* *y* *z*

   For each vertex there must be such a line, specifying the number and the coordinates of this vertex.
2. A *face section*; this begins with *Faces:* and specifies a vertex-number sequence for each face. Each such sequence is followed by a period. When we visit each vertex of a face in the order of the vertex-number sequence, the orientation must be counter-clockwise when we view the face from outside the object and the first three vertices must denote a convex angle.

3. An optional *group section*; this begins with *Groups:* and specifies first the number of groups and then the vertex number of each group, followed by an exclamation mark.

For example, Figure 55 shows a prism, with its vertex numbers, on the left, and the corresponding data file, obtained with the *File* | *Save* command, on the right. The four vertices of the bottom face form group 1 and those of the top face group 2. As is usual with Windows applications, the window has been reduced in size to display also another window, obtained by clicking the MS-Windows *Notepad* icon.

If you like, you can make alterations in this data file yourself to modify your three-dimensional object. Or, if you are familiar with programming, you can even have such a file generated by your own program (see also Appendix B).

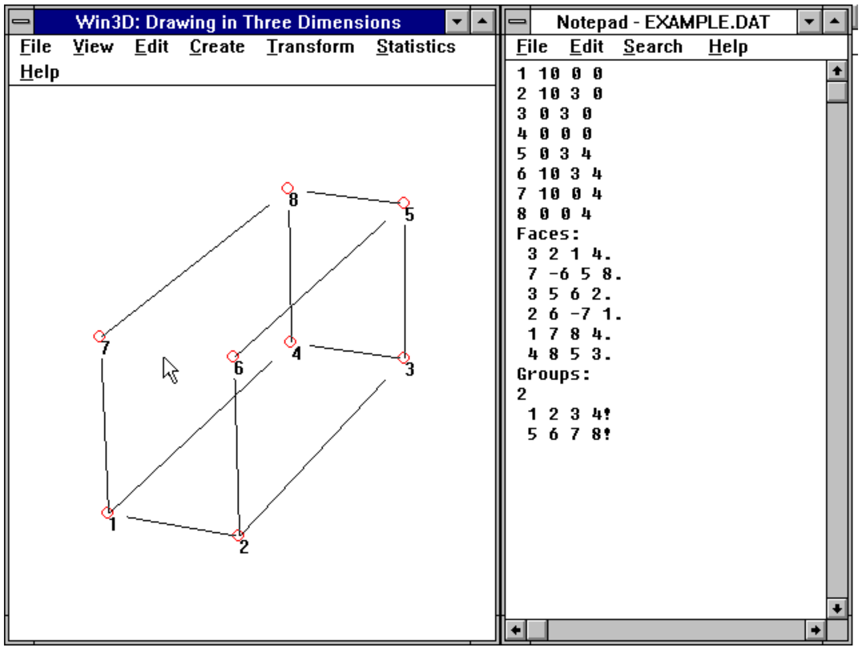

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Notepad - EXAMPLE.DAT
File Edit Search Help
8
5
7
6
4
3
1
2

```
1 10 0 0
2 10 3 0
3 0 3 0
4 0 0 0
5 0 3 4
6 10 3 4
7 10 0 4
8 0 0 4
Faces:
 3 2 1 4.
 7 6 5 8.
 3 5 6 2.
 2 6 7 1.
 1 7 8 4.
 4 8 5 3.
Groups:
2
 1 2 3 4!
 5 6 7 8!
```

Figure 55. *An object and its data file*

Making visible edges invisible

We can make a visible edge of a polygon invisible by first selecting both end points and then using command *Edit* | *Invisible*. If we use the *File* | *Save* command for an object of which some lines have been made invisible in this way, there will be minus signs in front of some vertex numbers. Figure 56 shows these minus signs on the right, resulting from making edge 6-7 invisible in this way. Note that there are two faces for which there is such a minus sign: the top face, specified as 7 −6 5 8, and the front face, 2 6 −7 1.


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
Notepad - EXAMPLE.DAT
File Edit Search Help
1
2
3
4
5
6
7
8

```
1 10 0 0
2 10 3 0
3 0 3 0
4 0 0 0
5 0 3 4
6 10 3 4
7 10 0 4
8 0 0 4
Faces:
 3 2 1 4.
 7 -6 5 8.
 3 5 6 2.
 2 6 -7 1.
 1 7 8 4.
 4 8 5 3.
Groups:
2
 1 2 3 4!
 5 6 7 8!
```

Figure 56. *Edge 6-7 made invisible*

You may wonder about the use of making visible edges invisible, and you are right if you think the above example unpractical. However, invisible edges enable us to create holes, as we will see next.

Holes in faces

So far we have dealt with objects bounded by polygons. Although we can construct many interesting objects in this way, we have not yet built an object that contains holes, as Figure 59 shows. The way to do this is by using two coinciding, invisible edges. Figure 57 shows two squares, a large and a small one, lying in the same horizontal plane. They were defined as two faces. Command *Create* | *Hole in face* (see Figure 57) enables us to combine these two polygons into one. Before using this command we must specify where the invisible edges should be. We do this by using a square mark (using F11) for a vertex of the large polygon and a triangular mark (using F12) for a vertex of the small one. As Figure 57 shows, these two vertices are preferably chosen close together. Although very little seems to happen, the internal data structure changes from two faces to one. Expressed in terms of data files, the two faces are initially specified by, for example,

```
1 2 3 4.
5 6 7 8.
```

After using the *Hole in face* command, however, these two faces are combined into one, which would appear in a data file as follows:

1 2 −6 5 8 7 6 −2 3 4.

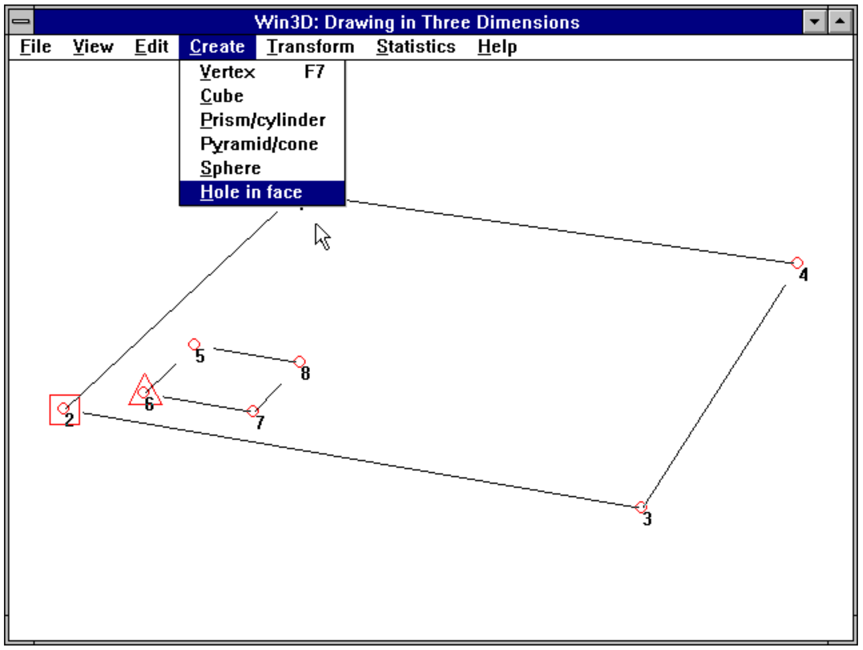


Figure 57. *Preparation for square hole*

Following the vertices of this face, we traverse the vertices in that order. The minus sign in 2 −6 indicates that the edge from vertex 2 to vertex 6 should be invisible, and so does the second minus sign, in the pair 6 −2.

Since the outer and the inner squares now form one polygon, we can use it to combine it with a second hole, as Figure 58 shows. Although not necessary, we have now switched to grid-box mode, which displays the two coinciding artificial edges from vertex 2 to vertex 6. Remember, we should always use a square mark on the larger face and a triangular one on the smaller face, which is to be the new hole.

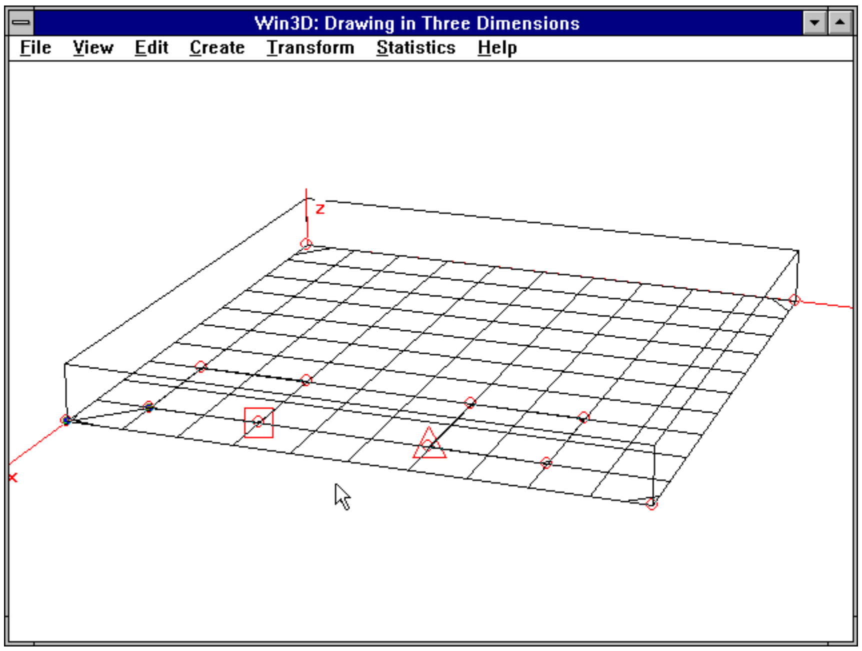

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
x

Figure 58. *Second hole in the same face*

After applying command *Hole in face* for the second time, we can use the *Move/Sweep* command of the *Transform* menu to build the object shown in Figure 59.

Note that we have not required a specific orientation of the vertices of the face and hole that are to be combined. The *Hole in face* command actually reverses the order of the face vertices, if necessary, so we need not worry about this. As you can see from the above number sequence, the final sequence of the combined face will always be similar to the way you can cut a sheet of paper with scissors: the material to be retained will always be on the same side, say, on the left, of your hand.

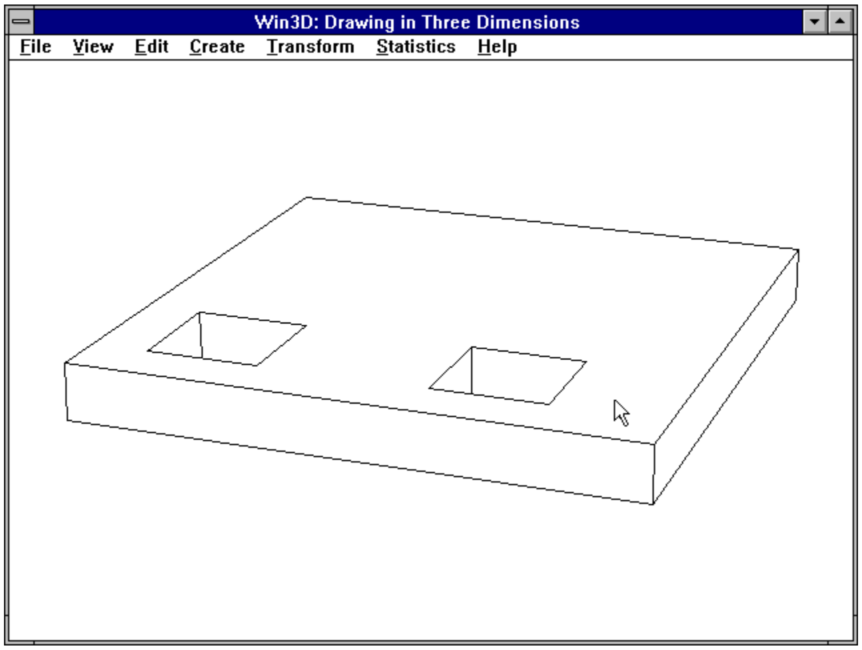

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 59. *Object obtained by sweeping face with hole*

Cylindrical holes

Since holes are usually made by drilling, they are more often circular (or, rather *cylindrical* in the final object) than square. Recall that we approximate a circle by a regular polygon, so all we have to do is replace the square hole of our previous example by one that is a regular polygon. We can generate the vertices of such a regular polygon by using the *Transform | Rotate/RevSurf* command (see Figure 18). As Figure 60 shows, we begin by creating an axis of rotation and a single point, which is to be rotated and copied many times.

Figure 61 shows how to use the *rotate* dialog box for this purpose. We click the *Rotate Copy* radio button and we enter 360° as the angle of rotation and (rather arbitrarily) 30 as the number of steps. The given point (which must be the only point that is selected) is then copied 29 times, each time by rotating the previous copy through an angle 360/30 = 12° about the axis specified by the square and triangular marks.

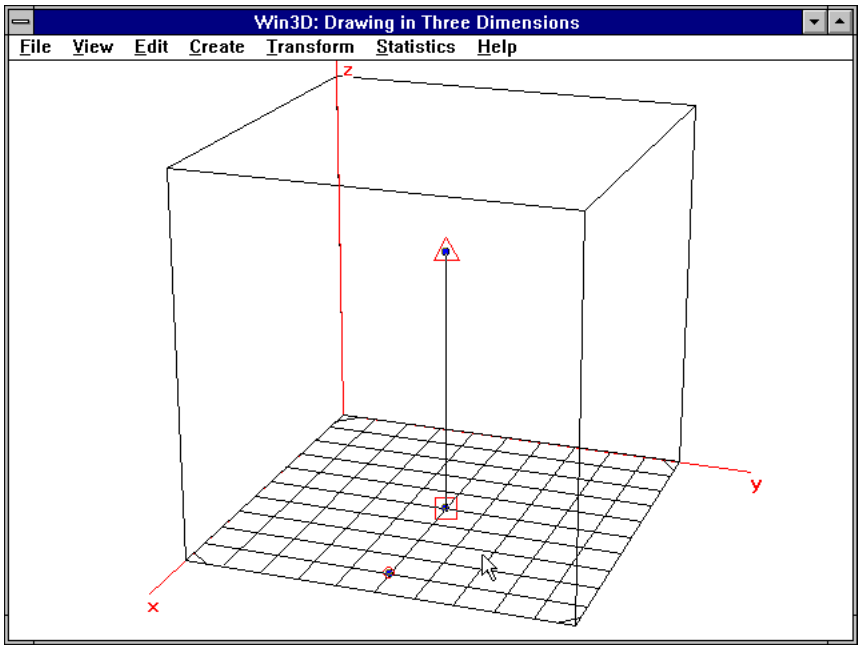
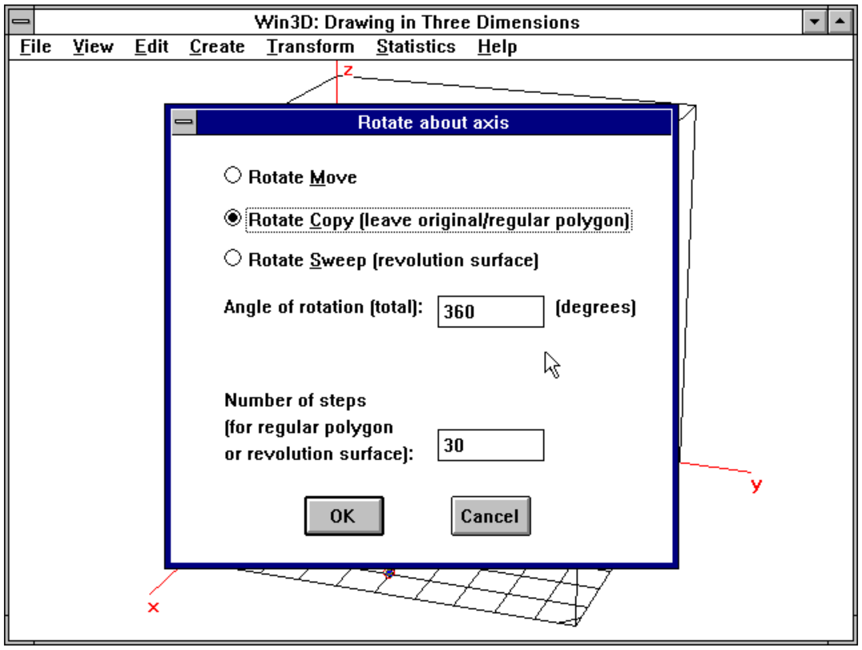

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Figure 60. *Preparation for circle in square*


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Rotate about axis

Rotate Move

[✓ Rotate Copy (leave original/regular polygon)]

Rotate Sweep (revolution surface)

Angle of rotation (total): 360 (degrees)

Number of steps (for regular polygon or revolution surface): 30

OK Cancel

Figure 61. *Generating points on a circle*

Since a set of points is not the same as a face, we must not forget to use *Edit | Insert face*, or press the Ins key. After doing this and defining where the invisible edge should be, we have the situation of Figure 62, which is essentially the same as that of Figure 57.

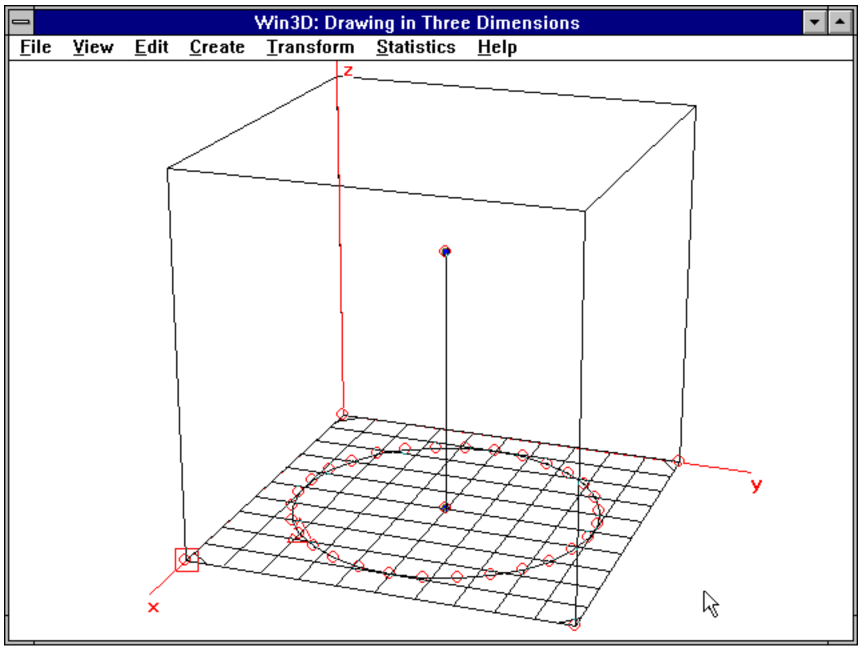

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Figure 62. *Circle generated*

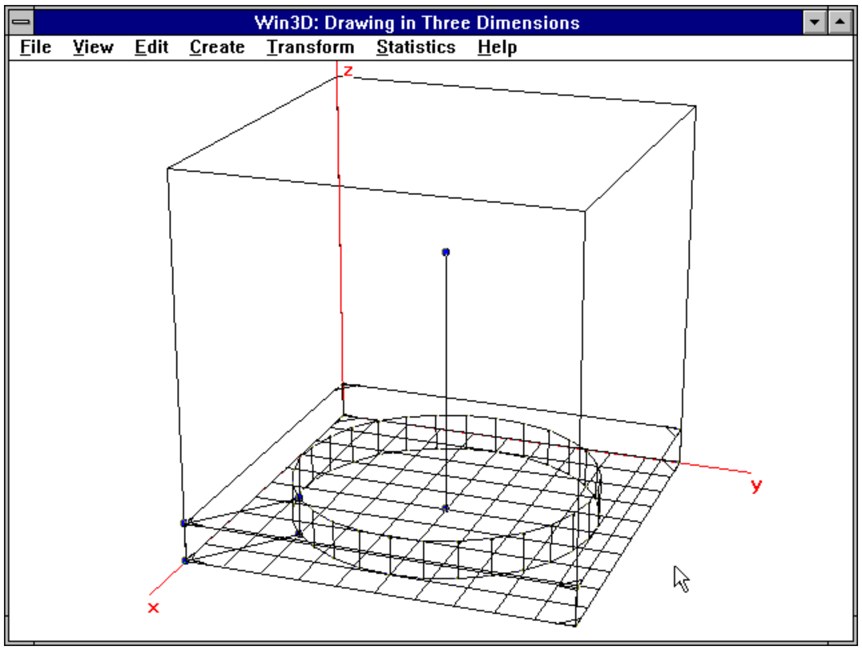
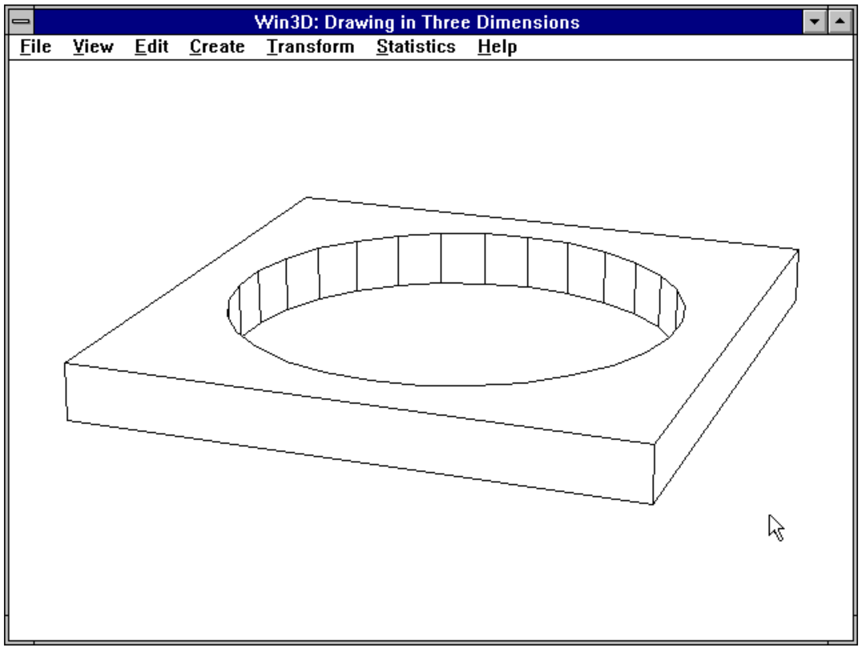

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
y
x

Figure 63. *Cylindrical hole displayed in grid box*

Note that the current viewing mode (wire frame or grid box) is not relevant. After using *Transform* | *Move/Sweep*, we obtain the object shown in Figures 63 and 64.

If we use *Statistics* | *Problem size* or inspect the file obtained by the *File* | *Save* command, we can verify that the number of faces for this object is equal to

2 (horizontal) + 4 (outer, vertical) + 30 (inner, vertical) = 36



Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 64. *Hidden-line representation of cylindrical hole*

6

Unification and Reflection

Coinciding vertices

Figure 65 shows two cubes: the one on the left was generated by using *Create* | *Cube*, the other one by using *Transform* | *Move/Copy/Sweep*, entering 2.5 in the *y*-box and clicking the *Copy* radio button.

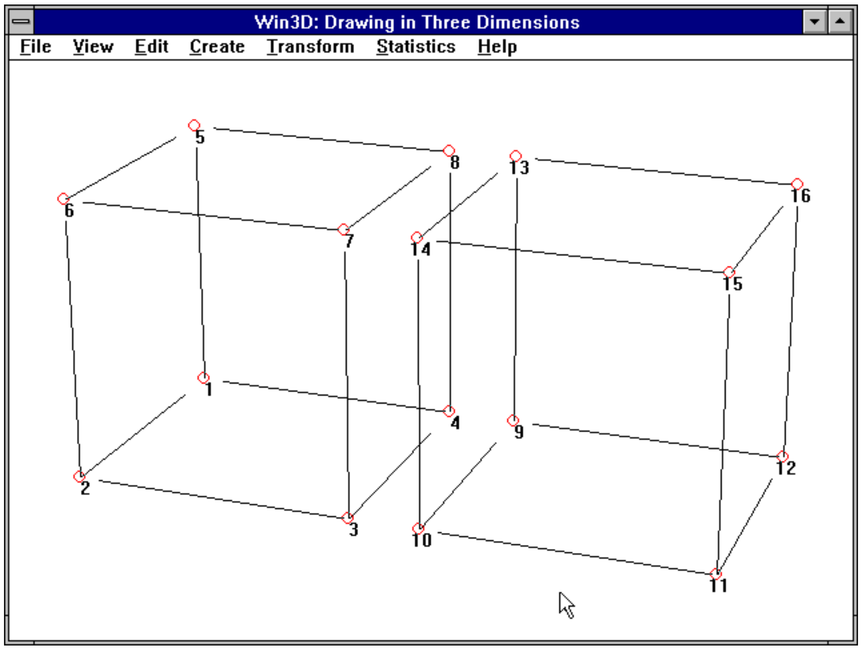

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
5
8
13
16
6
7
14
15
1
4
9
12
2
3
10
11

Figure 65. *Two cubes*

If we now move the right cube 0.5 unit to the left, the vertices 3, 4, 7 and 8 of the left cube will coincide with vertices 10, 9, 14 and 13 of the right one.

This may give some undesirable effects. For example, if we press F8 to select all vertices, there will be no red circles around the coinciding vertices, as Figure 66 shows. This is because these circles are deleted in the same way as they are drawn: drawing the second cancels out the first.

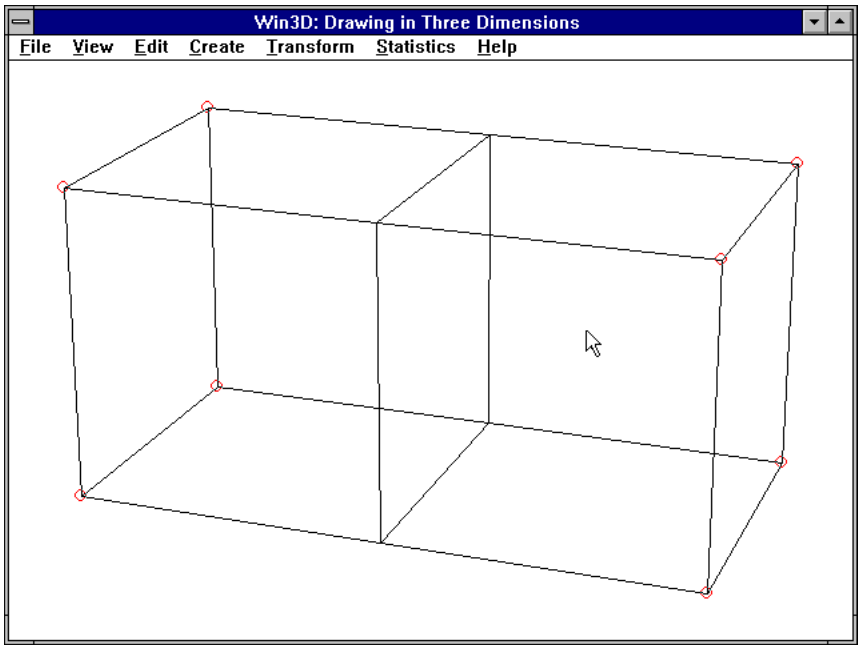

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 66. *Cubes with coinciding vertices*

We can verify that all vertices are really selected by using *Edit | Show nr*. However, this shows another undesirable effect. Since two vertex numbers cannot appear in the same position on the screen, the last one displayed in that position remains visible. Consequently, for these coinciding vertices, only the numbers 10, 9, 14 and 13 are displayed, not 3, 4, 7 and 8.

Finally, and more seriously, if we click these vertices red circles will alternately appear and disappear, but it is not immediately clear to the user whether, for example, vertex 3 or vertex 10 is selected. The situation is slightly better if each cube forms a *group*. In that case we can press F6 to switch from one cube to the other.

Unification

There is an easy way to dispose of superfluous numbers of coinciding vertices. All we have to do is use *Edit* | *Unify coinciding points*. This command has the effect that any two or more vertices in the same position (not belonging to distinct groups) are combined into one vertex. We will call this action *unification*. Figure 67 shows the situation after applying this command. The vertex numbers 3, 4, 7 and 8 are now unique for the points they represent: they belong now also to the right cube. The number of vertices has been reduced by 4. Since the vertex numbers 9, 10, 13 and 14 are no longer needed for their original purposes, these numbers can now be used for other vertices. In other words, the gaps created by unification are automatically closed by renumbering some other vertices. (Incidentally, this also happens when we delete some vertices.) You can see this in Figure 67: the vertical face on the extreme right now has vertex numbers 9, 10, 11 and 12, while the corresponding numbers were 11, 12, 15 and 16, respectively, in Figure 65.

Two coinciding vertices are not unified if they belong to different groups. If there are any such vertices, a message box will appear to notify that some more points can be unified after applying the *Edit* | *Ungroup* command.

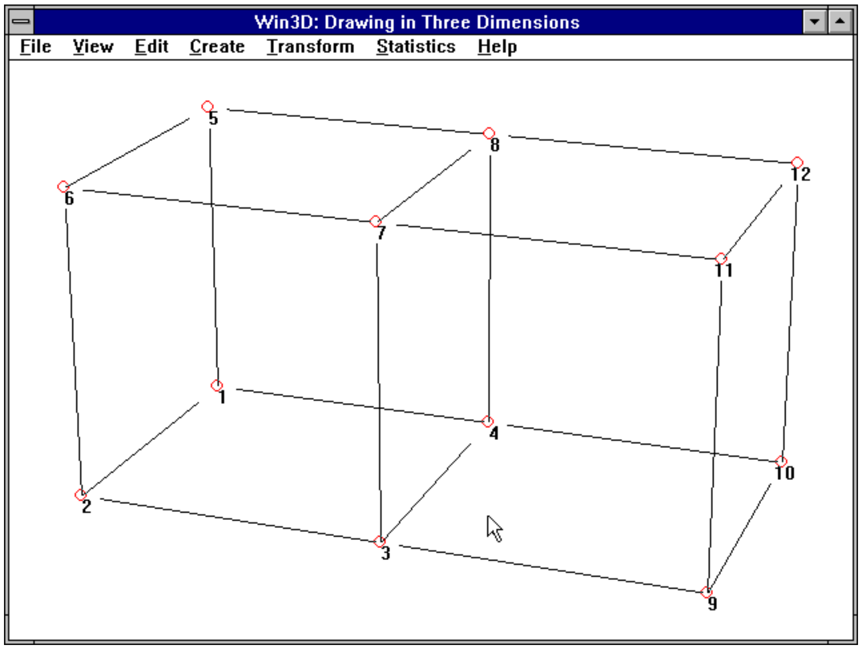

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
5
8
12
6
7
11
1
4
10
2
3
9

Figure 67. *Situation after unification*

Coinciding invisible edges

The *unify* command is particularly useful if we want to make coinciding edges invisible, as discussed in Chapter 5. After finishing our unification process, we can easily make the coinciding edges in Figure 67 invisible. (This would be considerably more difficult without unification.) We now simply select vertices 3, 4, 7 and 8 and use command *Edit* | *Invisible*, to obtain the result of Figure 68. The red circles show that the vertices just mentioned still exist, but the edges that connect them are now invisible.

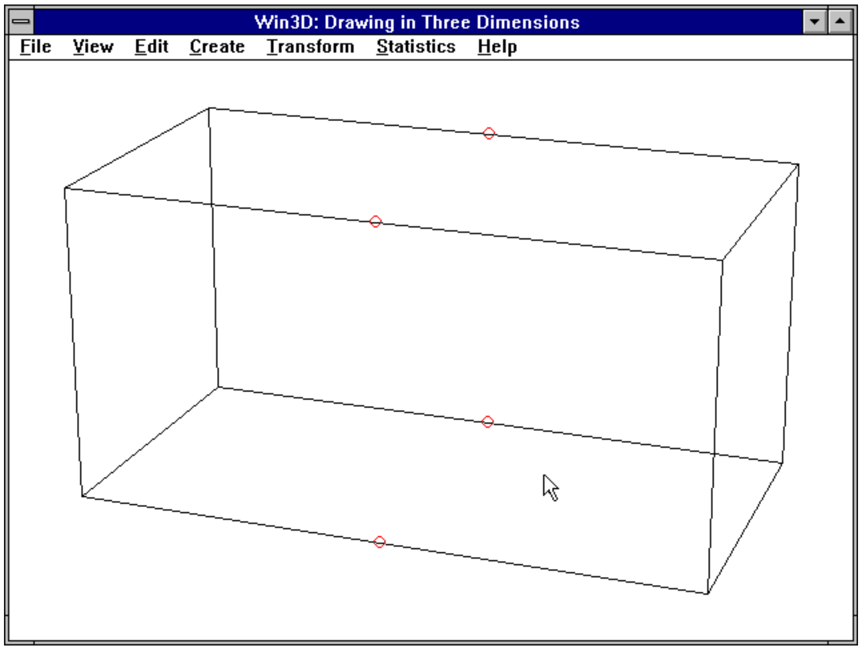

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help

Figure 68. *Coinciding edges made invisible*

Reflection

If an object is symmetrical, with a (horizontal or vertical) plane of symmetry perpendicular to the *x*-, *y*- or *z*-axis, we can benefit from its symmetry by constructing only half of the object and then generate its reflection by means of the *mirror* transformation. The command to be used for this purpose is *Transform* | *Stretch/Mirror*. We will see how this command works by constructing the upper part, a cross, of the king as used in the game of chess. We have seen such a king in Figure 6, at the end of Chapter 1. Its lower part is a solid of revolution, and constructing this is not essentially different from what we did in Chapter 5; we will therefore restrict ourselves here to the cross at the top of this king. The subject of reflection is not unrelated to unification and invisible edges, as we will see.

The *Import* command

We can construct this upper part separately, and later add it to the lower part (that is, to the solid of revolution). Adding two parts together is done by using command *File* | *Import*. Any clashes in vertex numbers are automatically resolved by proper renumbering. Importing objects is similar to combining several documents together with a text processor. Immediately after importing an object, its vertices are selected, so we can press F5 to make a group of it. This is very important, because it enables us to move it and to subject it to other transformations.

The Stretch/Mirror dialog box and reflection

We begin by constructing the front (or back) face of the right half of our cross object, as shown in Figure 69. The *x*-, *y*- and *z*-dimensions of the grid box used here are 5 × 10 × 10. All vertices displayed here have zero *x*-coordinate; their *y*-coordinates are immediately clear from Figure 69, and their *z*-coordinates are integers, temporarily displayed in the title bar as we define each vertex. The vertical auxiliary lines displayed here make it perhaps a bit difficult to see that we have a polygon, but you can simply follow the vertex numbers 1, 2, ..., 9 to see that this is indeed the case.

We select all vertices by pressing F8, and we press F11 and place a square mark at vertex 9. This is necessary to indicate that a (horizontal or vertical) plane through this point will be the plane of reflection. We then use command *Transform* | *Move/Copy/Sweep*. We then click the *Mirror y* and the *Leave original* check boxes, as shown in Figure 70. The *Mirror y* check box together with the square mark at vertex 9 indicate that the plane through that vertex and perpendicular to the *y*-axis will be the plane of reflection for this mirror operation.

After clicking the OK button, we obtain the face shown in Figure 71. We have switched to the wire-frame viewing mode (using *Edit* | *Wire frame*); also, vertex numbers were displayed here (by pressing F8 and using *Edit* | *Show nr*) to make the explanation clearer.

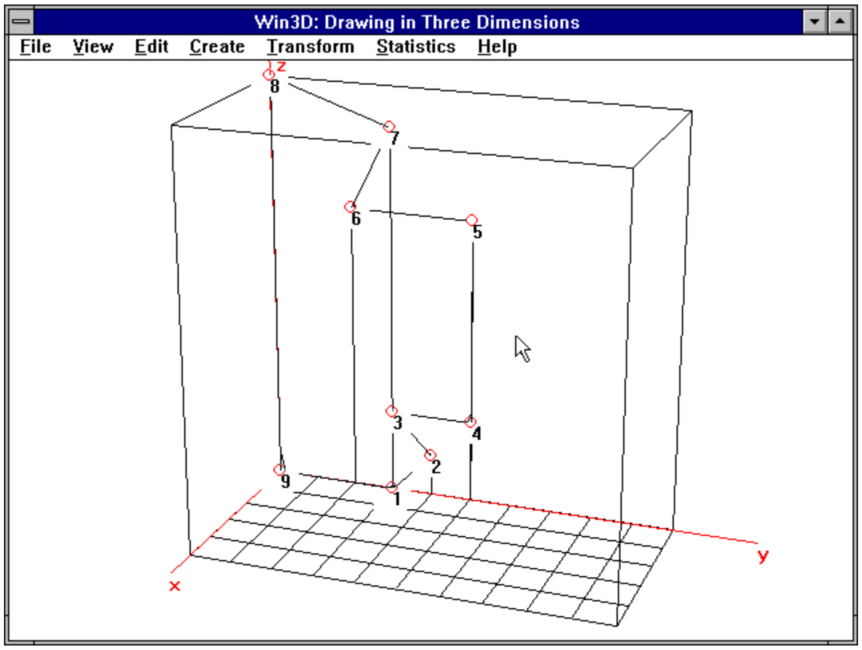
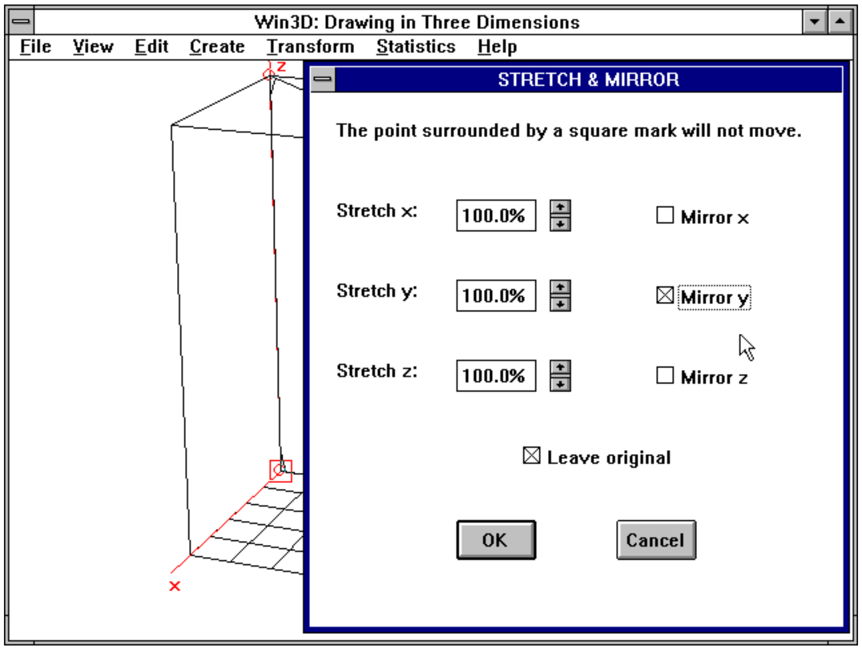

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
z
8
7
6
5
3
4
2
9
1
x
y

Figure 69. *Front face of right half*


Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
STRETCH & MIRROR
OK
Cancel

The point surrounded by a square mark will not move.

Stretch x: 100.0% ☐ Mirror x

Stretch y: 100.0% ☒ Mirror y

Stretch z: 100.0% ☐ Mirror z

☒ Leave original

Figure 70. *Dialog box for reflection*

Actually, the situation in Figure 71 is not as good as it looks: the vertices 17 and 18 of the left half coincide with vertices 8 and 9 of the right half, and the latter numbers are not displayed.

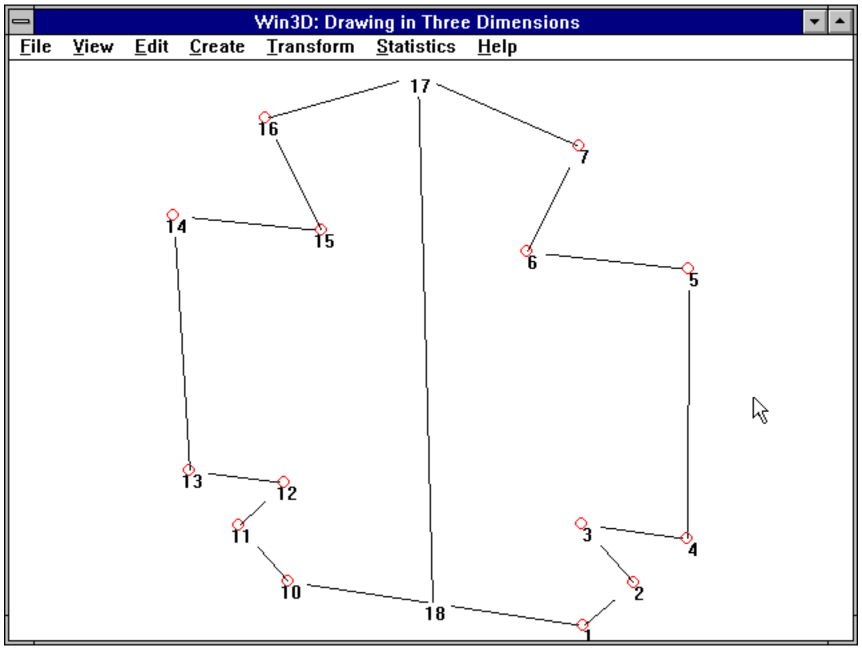
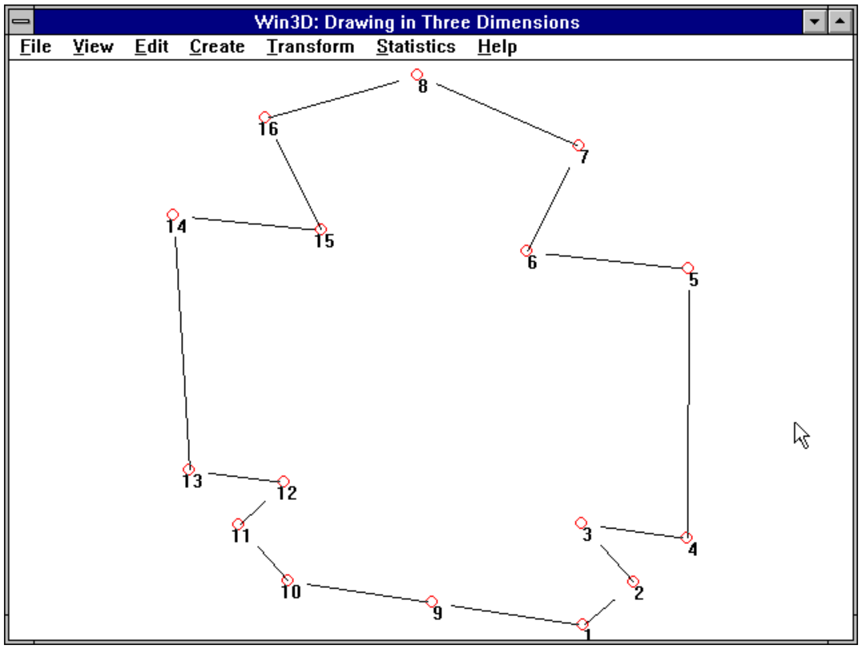


Figure 71. *Result of reflection*

This is relevant because we want to make the two coinciding edges 17-18 and 8-9 invisible. Thus, we first use *Edit* | *Unify coinciding points*. If, after doing this, we displayed the vertex numbers again, we would find that the numbers 17 and 18 in Figure 71 have been replaced with 8 and 9.



Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 72. *Edge made invisible (after unification)*

The highest vertex number of our object is now 16 instead of 18. We now press F9 to deselect all vertices and select vertices 8 and 9. Using *Edit | Invisible* then makes the vertical line in the middle of Figure 71 invisible, as shown in Figure 72.We now want to give this object its required thickness by using *Sweep*, but, unfortunately, our attempt to do this in one step fails and displays the message *Selected points do not identify a face*. This is because Figure 72 actually contains two polygons, 1-2-3-4-5-6-7-8-9 and 9-10-11-12-13-14-15-16-8.

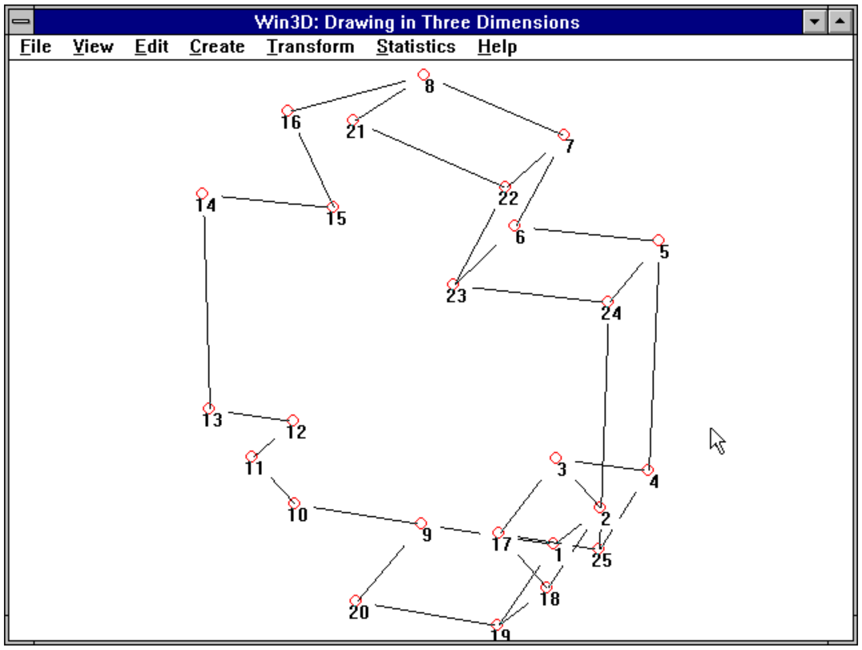


Figure 73. *Sweep applied to right half*

We therefore have to sweep these two faces one by one. We press F9 to deselect all vertices; then we select some vertices, say, 1, 2 and 3, of the right face, and we use *Edit* | *Select face*. All vertices of the right polygon are then selected. We then use *Transform* | *Move/Copy/Sweep*, click the *Sweep* radio button and enter 4 in the *x*-box (to give it a thickness of 4 mm). This produces the situation of Figure 73.

We then do the same for face 9-10-11-12-13-14-15-16-8 on the left, resulting in the complete object. We can dispose of the coinciding edges in the middle of the bottom face by using *Edit* | *Unify coinciding points*, selecting the vertices shown as 9 and 20 in Figure 73, and using *Edit* | *Invisible*. (This action is not really necessary because the bottom face of this object will coincide with the top face of the lower part of the king and will therefore be invisible anyway.) The result, in hidden-line representation with vertex numbers displayed, is shown in Figure 74.

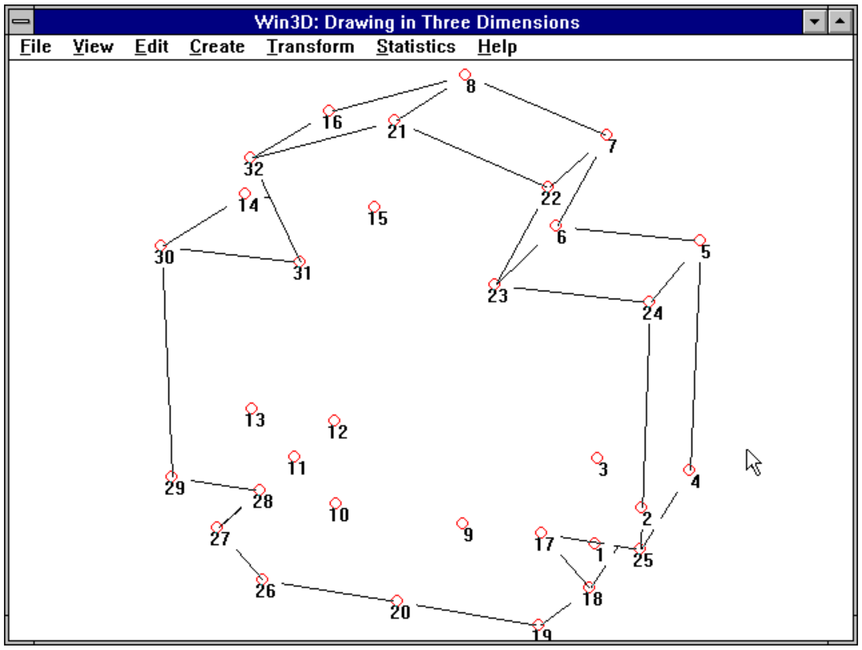

Win3D: Drawing in Three Dimensions
File View Edit Create Transform Statistics Help
8
16
21
7
32
22
14
15
6
5
30
31
23
24
13
12
11
3
4
29
28
2
10
27
9
17
1
25
26
18
20
19

Figure 74. *Left half also swept and bottom edge made invisible*

Although the object now has the right dimensions, it still has to be moved. We can do this either before or after we save it in a file. To produce the king of Figure 6 (at the end of Chapter 1), we should move it by pressing F8, using *Transform* | *Move/Copy/Sweep*, leaving the default *Move* radio button selected and entering −2 in the *x*-box and 57 in the *z*-box. (Recall that the thickness of this object is 4 and its back face lies in the *yz*-plane before this move.) The bottom part of the king should then have a height of 57, with its bottom face in the *xy*-plane and constructed by a rotation about the *z*-axis.

Before we combine the two parts, it is wise to save them in separate files. We can then *Open* one of these files and press F8 and F5 to make a group of the first part. We then *Import* the other file, and press F5 to make a group of the second part as well. Then if either part is not yet in its correct position, we can easily select it by pressing F6 and move it. Making a group of an imported object is good practice in general, even if we need not move it because we have already given it its final position, as suggested above.

Alternative approaches

When building the top part of a king, we have first used the *mirror* and then the *sweep* transformation. Instead we could very well have done it the other way round, first applying *sweep* only to the right half, giving this half its thickness and only then subjecting this three-dimensional object to the *mirror* transformation. Then, at the end, we would also have to use unification to make some common edges invisible. When applying the mirror transformation to a three-dimensional object, we need not worry about the orientation of the faces. These obviously need to be reversed, but Win3D does this automatically.

Now that we have done so much with sweep and mirror transformations, we must not forget that we can also define all bounding faces one by one, by first defining and selecting all vertices of a face and then using *Edit* | *Insert face* (or pressing the Ins key). If we define faces in this way, we must be careful with the order in which we select the vertices: we must do this by following all edges of a face in counter-clockwise order, when viewing the face from outside. In our present example, this method, too, would have worked reasonably well.

Yet another alternative, more similar to what we actually did in this chapter, avoids the problem of coinciding vertices and edges. We can define all vertices of the right half of the front face, except for those vertices that have zero *y*-coordinates. Then, *before inserting a face*, we can use the mirror transformation to generate the corresponding vertices of the left half of this face. We then define the two vertices of this face that lie on the *z*-axis. Each vertex of this face now occurs only once. We can then select them one by one, traversing all these vertices in a systematic order, say counter-clockwise, each time clicking the right mouse button. Then pressing the Ins key generates the face, and a single sweep operation completes the object.

A

Win3D Reference

A1. Modes of Operation

There are three ways to start using Win3D:

a. You can create *stock objects* using *Create*.
b. You can open files produced by Win3D, using *File* | *Open*.
c. You can build a new object using *File* | *New*.

A2. Defining New Points

After selecting *File* | *New*, and entering grid box information, a grid box appears. To define a new point (probably to be used as a vertex), move the cursor to a point of the horizontal grid shown in the *xy*-plane. Then press the left mouse button and keep it down as you move the cursor either upward or downward. When you release the mouse button, a new point appears in the form of a small blue circle. Then define the next vertex, and so on.

After defining points, you can select them to prepare for defining a new face.

A3. Selecting Points

You can select and deselect a single point by moving the mouse cursor to it and pressing the *right* mouse button. Selected points show red circles around them. When points are deselected, these red circles disappear. Selecting points should be distinguished from marking points, the latter being required for some transformations.

You can specify and display a rectangular bounding box to change the select/deselect status of all vertices that lie inside it. If you are working with a grid box (see A2), use *Edit* | *Box select* or press F4; otherwise this step is not required. Then press the left mouse button to define one corner of the bounding box and release it to define the opposite corner. For any vertices lying inside this box, the select/deselect state changes: if such a vertex was not selected, it becomes selected and a red circle appears around it. Analogously, any existing red circles inside the box disappear, the corresponding points being no longer selected.

Yet another way of selecting points is by creating groups, each of which contains a subset of all vertices. Once you have several groups, you can select each of them in turn by using the *Select group* command. Selecting points is useful for

a. Deleting points (using the Del key) and the faces to which they belong
b. Defining bounding faces (or line segments)
c. Defining a subset of points to be subjected to a *Transform* command
d. Selecting a face by means of *Edit* | *Select face* (see below)
e. Showing coordinates or vertex numbers
f. Creating groups

When you define a bounding face (see b.) the points in question will be the vertices of a polygon. These vertices should be selected in counter-clockwise order, when the polygon is viewed from outside the object.

To select a face that has a great many vertices, just select some of its vertices (not all belonging also to another face) and use *Edit* | *Select face*.

A4. Defining Faces

After selecting points, you can define bounding faces or (in case of only two points) a line segment, by pressing the Ins key. This causes a dialog box to appear showing the orientation of the specified vertices. You can change this orientation by selecting the appropriate radio button in this dialog box.

A5. Viewpoint

To choose a different viewpoint, click *View* | *Viewpoint* and change one or more of the spherical coordinates *rho* (= ρ), *theta* (= θ) and *phi* (= ϕ) by using scroll bars.

The smaller *rho* (that is, the viewing distance EO), the stronger the perspective effect will be.

The angle *phi* should be chosen small if you want to view the object from above. As shown in the Viewpoint dialog box, *phi* is the angle between the (vertical) *z*-axis and the line of sight EO, where E is the viewpoint and O the center of the object. A value of 70° for *phi* is very usual.

With *theta* between −90° and +90°, you view the object from the front. Increasing such a value by 180 means viewing the object from the back.

A6. Saving Objects

(This option is not available in the demo version of Win3D. Please refer to the address given by *File* | *About Win3D...* for the way you can obtain the complete version, which may also be improved in other respects.)

You can save your object by selecting *File* | *Save as*. Group information is also stored in the files produced in this way. The file name extension is .DAT. These files are in ASCII format and have a very simple structure. Except for the new facility of using groups, the file format is discussed in the first two books listed in the References.

A saved object can be retrieved later by means of *File* | *Open*. Also, you can use the *File* | *Import* command to combine several objects. Any groups in the imported file are added to those already in use.

A7. Moving, Copying and Sweeping

You can move, copy or sweep an object or part of it by using *Transform* | *Move/Copy/Sweep*. Checking the *Leave original* box (by clicking it) causes copying, while moving takes place if that box is unchecked. A checkbox is checked if it shows a cross (X) in it. The operation is applied to the currently selected points. Sweeping should be applied only to one polygon at a time (see below).

If one or more points are currently selected, only those points (as well as the faces and lines they belong to) are moved or copied.

Sweeping is one of the most powerful Win3D commands. You can apply it to a previously selected face (that is, to a polygon) to build a prism. For example, if you want to build a cube, you can simply define, say, its bottom face and sweep this in the same way as you would move or copy it to its top face, using *Transform* | *Move/Copy/Sweep* and selecting the *Sweep* radio button. Then, besides the top face of the cube, its four vertical faces appear as well. You need not bother about the orientation of the vertices of a face used in sweeping operations: if required, this orientation will automatically be reversed to make it counter-clockwise when, after building a prism in this way, the face is viewed from outside this prism.

A8. Stretching and Mirroring

You can stretch an object (changing its size) and you can produce its mirror image, using planes of reflection that are perpendicular to the *x*-, the *y*- or the *z*-axis. To keep the original, check the *Leave original* box. Only selected points can be stretched or reflected. If all points are to be stretched or reflected, use the *Select all* command of the Edit menu.

To prepare for stretching or mirroring, you must mark some point, which will be a *fixed point* of the transformation. Since that point will show a square around it, you should use *Transform | Sq Mark*. For example, if you stretch rectangle ABCD, using A as the fixed point and 200% as the three scale factors, then the new rectangle A′B′C′D′ will be such that A′ = A, A′B′ = 2AB, A′C′ = 2AC, A′D′ = 2AD. Selecting and marking are independent actions: both selected and unselected points can be marked. It is not possible for two points to have a square mark at the same time.

In the case of a reflection (that is, a mirror transformation), the marked point indicates the plane of reflection. For example, if you choose *Mirror z* when P shows a square mark, the horizontal plane through P will be the plane of reflection. Stretch and mirror operations can be combined, and you can use any combination of the *x*-, *y*- and *z*- directions. For each direction you can change the default scale factor of 100%.

A9. Rotations, Circles and Solids of Revolution

Win3D enables you to rotate objects or parts of them about any axis (which need not be horizontal or vertical), and through any angle. As with the normal *Move* and *Copy*, you can leave the original to perform a *Rotate Copy* instead of a *Move Copy*. If you want to approximate a *circle* by a regular polygon, select only one point, and use the same *Rotate Copy* radio button, but enter a not too small positive integer (for example, 30) in the *Number of steps* box. After generating the vertices of a regular polygon in this way, do not forget to insert a face by pressing the Ins key.

It is also possible to form a solid of revolution by selecting the *Rotate Sweep* radio button. In all three cases, there must be two marked points to determine the axis of rotation. These two points show a square and a triangle. Imagine an arrow, with the two points just mentioned as its tail and its head, respectively. This arrow denotes the direction in which a right-handed screw would advance when turned in the sense of the rotation.

To build a solid of revolution, the selected points must be the vertices of a face, and the two points denoting the axis must lie in the same plane as this face. You need not bother about the orientation of this face: if required, it will automatically be reversed, to make it counter-clockwise, when, after completion of the *Rotate Sweep* operation (with an angle less than 360°), the face is viewed from outside the solid of revolution.

A10. Importing Objects

Objects stored in distinct files can be combined by using *File* | *Open* for the first object and *File* | *Import object* for the second, the third etc.

Immediately after an object is imported, it is automatically selected. This enables you to transform it or to make a group of it so you can easily select it later. This is not required if there are already groups in the imported file; in that case these new groups are added to those that already exist. For example, if your object contains the groups 1, 2, 3, and you import another object containing the groups 1 and 2, the latter two groups will be renamed 4 and 5, so we obtain the groups 1, 2, 3, 4, 5 in the combined object.

Immediately after using the *Import* command, you can press the Del key to cancel the effect of this command. Do not use the *Undo* command in this case.

Frequently the scale of the imported object will be different from the one you were dealing with before using *Import*. You can then alter its size, using *Transform* | *Stretch*. Also, it may be desirable to move the imported object so you can place it beside the object you were originally dealing with.

A11. HPGL and DXF Files

The *File* | *Export HPGL* command enables you to obtain an HPGL file of a hidden-line or wire-frame representation. The recommended file-name extension for HPGL files is .PLT (or .HPG). These files can be imported in most draw programs (such as Corel Draw) and text processors (such as WordPerfect or Word for Windows).

The HPGL file will result in a wire-frame model if a wire-frame model is visible on the screen when you give the *Export HPGL* command. Otherwise, a hidden-line image will be written to the HPGL file. If, instead of line drawings, you want a hard copy of a hidden-face representation, you could use a screen-capture program for Windows applications. An example of such a program is Collage Plus (published by Inner Media, Inc.), which produces bitmapped files that text processors can import into documents.

The *File* | *Export DXF* command produces a DXF file, which can be imported into AutoCAD. In contrast to HPGL files, these DXF files will contain 3D information of the object. In other words, a DXF file generated by Win3D contains the *x*-, *y*- and *z*-coordinates of the object in question, while an HPGL file describes only its (perspective) image. Since our DXF files describe 3D objects, they are *not* accepted by CorelDRAW and other 2D draw packages, even though these include DXF in their lists of acceptable import formats. The DXF files these packages accept are 2D image files, for which we use the widely accepted HPGL format.

DXF files are not as compact as the normal Win3D data files (.DAT files) used by the *Save* and *Open* commands.

A12. Orientation

With Win3D, objects are defined by the polygons (called *faces*) that form their boundaries. When you view a face from outside the object, the orientation of its vertices must be counter-clockwise. This is relevant when we are selecting the vertices of the face just before pressing the Ins key (or before using the *Edit* | *Select face* command). For example, consider this square, ABCD:

A B

D C

To make the orientation counter-clockwise, you may select these four points only in any of the following ways: DCBA, CBAD, BADC, ADCB. After pressing the Ins key, a dialog box appears so you can check if you have used the correct orientation. If not, you can immediately change the orientation by clicking a radio button. The same dialog box appears if you first select some points of a face and then use the command *View* | *Show orientation*.

A13. Showing Coordinates and Vertex Numbers

It is sometimes desirable to know the coordinates or the numbers of one or more vertices. For example, when using the *Move* command you have to specify distances *x*, *y* and *z*. You may then want to know the coordinates of some of the vertices to be moved. You can obtain these by first selecting those vertices and then using the *Show x*, the *Show y* and the *Show z* commands of the View menu. Similarly, the *Show nr* command displays the numbers of the selected vertices. These numbers are referred to in error messages about points not lying in the same plane. You can also find them in the files obtained by using the *Save* and *Save as* commands.

To make the coordinates or numbers disappear, use *View* | *Hidden lines*, for example.

A14. Zooming

You can magnify part of your image by using *View* | *Zoom in*, or simply by pressing F2. This works by means of a rectangle, which, together with its contents, is scaled up to cover most of your window. You can define this rectangle by pressing the left mouse button at one of its corners and release it at the opposite corner. The normal view is then restored by using *View* | *Zoom restore*, or by pressing F3.

Note that zooming alters only the image, not the object itself, as the *Stretch* command does.

A15. Stock Objects

The *Create* menu enables you to create a cube, a prism, a pyramid and a sphere very easily. Both the prism and the pyramid have regular polygons as their bases. Except for the cube, these stock objects require an integer n. When you create a prism or a pyramid, n will be the number of vertices of its base. For large n, say, $n = 30$, a prism and a pyramid will approximate a *cylinder* and a *cone*, respectively. For a sphere, there will be n horizontal slices, with $n - 1$ approximated horizontal circles between a north and a south pole. Each of these approximated circles will be a regular polygon with $2n$ vertices. Remember, curved surfaces can always be approximated by a set of flat faces. The larger the value of n, the better this approximation will be but the more computing time will be required.

You can enter n in a dialog box, which appears when you choose one of the commands *Prism*, *Pyramid* and *Sphere.*

These stock objects all fit in a sphere with radius 1 and with the origin as its center, except for the pyramid, which has the origin as the center of its base.

There is also the command *Create* | *Vertex*, which enables you to define a new vertex by entering its x-, y- and z-coordinates. If you want to use this method for many points, use function key F7. ‘Loose’ points (not connected by lines) can be displayed as small blue dots by means of the command *View* | *Show vertices*, which actually shows all vertices this way.

A16. Printing

If your printer has graphics capabilities, you can use *File* | *Print* to print an image of the object you have created. If you do this when a wire-frame model is shown on the screen, the same wire-frame model is printed. In any other mode (hidden-line, hidden-face or grid box) a hidden-line model will be printed.

A17. Marking Special Points

A point marked by a surrounding square is required for stretching and mirroring transformations, while rotating transformations also require a second marked point, surrounded by a triangle. You can mark points by using the commands *Sq Mark* and *Tr Mark* of the *Transform* menu, or by pressing their shortcut keys, F11 and F12, respectively. Clicking a vertex of your object, using the *right* mouse button, then causes that vertex to be marked. At most one point can be marked with a square and at most one with a triangle. In other words, at most two points can be marked at the same time. In a stretching operation, the point marked with a square will be a fixed point. In rotations, the two marked points will be interpreted as the end points of an arrow (or a vector), the triangle corresponding to the arrow head.

A18. Groups

Complex objects are normally composed of several parts. When all vertices of such a part are selected (showing red circles) the *Edit | New group* command (or F5) makes a *group* of it. By repeating this process for other subsets of vertices you can create several groups. A vertex can belong to only one group. Later you can select the vertices of a group very quickly by using *Edit | Select group* (or F6). Groups are automatically numbered 1, 2, ..., *n*, where *n* is the number of groups currently defined. When using the *Select group* command repeatedly, you select these groups 1, 2, ..., *n* one by one. Each time the group number and the total number of groups is shown in the title bar.

Groups are particularly useful when a vertex, say, P, of one part of an object coincides with a vertex P′ of another. It would then be rather difficult to select P. You could do this by selecting some other vertices of a polygon of which P is a vertex and then using the *Select face* command. However, it is much easier to let these two parts be different groups. Then you can always quickly select one part (by means of *Select group*) and (temporarily) move it so P and P′ no longer coincide.

It may be desirable to select all vertices of two or more groups. This can be done by means of the *Edit | Add group* command (or Ctrl+F6). Like *Select group*, command *Add group* selects the next group, but any points already selected remain selected. In other words, this command adds the next group to the set of selected points.

This principle would be useless if the groups to be selected have numbers that are not consecutive. However, group numbers can be made consecutive by using *Edit | Change group numbers*. Suppose, for example, you have groups 1, 2, ..., 10, and you want to select the groups 3 and 8. Then you can select group 3 (by pressing F6 several times) and use *Change group numbers*. This generates a dialog box asking you for a new group number, for which you can enter 7. Group 3 now becomes group 7, and at the same time the old groups 4, 5, 6 and 7 become groups 3, 4, 5 and 6, respectively. Consequently, the groups you wanted to select are now numbered 7 and 8. All vertices of these two groups can now be selected by pressing F6 until group 7 is selected followed by pressing Ctrl+F6.

A19. The Undo Command

If you make a mistake, for example, by deleting your object or part of it, use *Edit | Undo* or press the *Esc* key immediately after your wrong command. The *Undo* command may also be useful immediately after transformations. For example, if you have performed a rotation about some angle, and you do not remember this angle, use *Undo* if you want to go back to the situation before the rotation. (For a simple *rotate move* for which you remember the angle you have used, you could instead use the inverse rotation, making the new angle the negative of the old one. Simply pressing the *Esc* key is easier, however.)

Since the *Undo* command and the preparations for it may take some time, it does not apply to some actions that can easily be undone in another way. For example, after the *File | Import* command, you can simply press the Del key to cancel the effect of this command. If you regret having changed your viewpoint you should use the *View* | *Viewpoint* command once again, possibly clicking the *Default* button. The *Undo* command does not work for *Zoom in* either. Instead, simply press the F3 key.

A20. Invisible Edges

If you use the *View* | *Hidden lines* command, all lines (or parts of them) obscured by the object are automatically removed, while all visible edges are drawn. However, it is sometimes desirable to make some of these visible edges invisible as well. This may be the case with curved surfaces, approximated by flat faces. A common edge of two such faces can explicitly be made invisible by selecting its two end points and using *View* | *Invisible*. Applying this command once again for the same edge turns it visible again.

If some edges of a face are made invisible, it may seem that the face is no longer there. However, it can still obscure other line segments. This shows that it is present although you do not see (all) its edges.

A21. Creating Holes

Before subjecting a polygon, say, face F, to a sweep move (using *Transform* | *Move/Copy/Sweep*, you can create a hole in it. You first create this hole as a face, say, H. Polygon H must lie in the same plane as polygon F and inside it. Then you select some vertices both of face F and one of face H (all vertices not belonging to F being deselected). You must now define an invisible edge, which will connect two vertices: one of face F and of hole H. Use a square mark for the vertex of F and a triangular mark for that of H. Preferably, these two marks should lie closely together; the invisible edge that connects them must not intersect another edge. Then all that remains is using *Transform* | *Create hole*. If you want several holes, you can do so. Once a hole has been created, each of its vertices can later be given a square mark to create an invisible edge to a new hole (showing a triangular mark). This is because creating a hole H in face F results in a new face F′, containing all vertices of F and H.

A22. Unification

If several vertices coincide, we can make one vertex of them by using command *Edit* | *Unify coinciding points*. There will be coinciding vertices, for example, if we generate two cubes of equal size and place them exactly on top of each other. Two vertices in the same position give problems when we want to select them. Unification solves these problems; it also renumbers any vertices that have numbers higher than those that have been disposed of. After applying this *Unify* command, all vertex numbers are consecutive. Two vertices in the same position will not be unified if they belong to different groups.

B

Files From Other Programs

History

Files similar to those used by Win3D first appeared in the first edition of my book *Programming Principles in Computer Graphics*. Published in 1986, it was one of the first books to use the C language for computer graphics. Today, you may find some newer related books more interesting:

Ammeraal, L., Zhang, K. : Computer Graphics for Java Programmers, 3rd Edition, Springer International Publishing (2017)

Ammeraal, L. Computer Graphics Programming in C++/Qt, Amazon, published both as paperback and as e-book (2019)

Since these books were written for programmers, they list all source code. By contrast, Win3D is intended for users, not for programmers. Examples of results generated by programs of my books are shown in this appendix. With the data files produced by these programs, you can produce the images shown below using Win3D.

Let us begin Figure 65. The first of these three objects is a *dodecahedron*, which has 12 faces, all regular pentagons of the same size. It is one of the five regular polyhedra. The same applies to the *icosahedron*, which has 20 equilateral triangles as its faces. The approximated sphere shown here has 80 triangular faces, but these do *no*t all have the same shape; consequently, the object on the right in Figure 65 is not a regular polyhedron. The other three regular polyhedra (not shown here) are a *cube*, a *tetrahedron* (with four triangular faces) and a *octahedron* (with eight triangular faces).

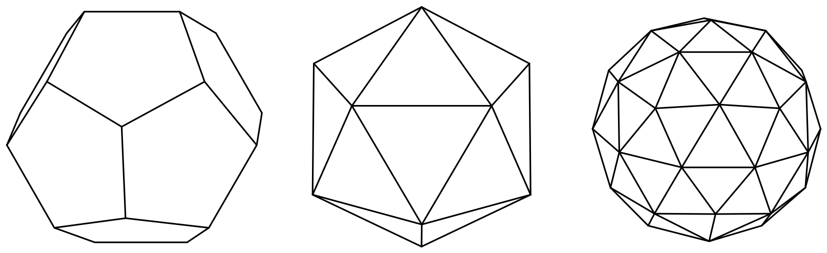


Figure 65. *Dodecahedron, icosahedron, approximated sphere*

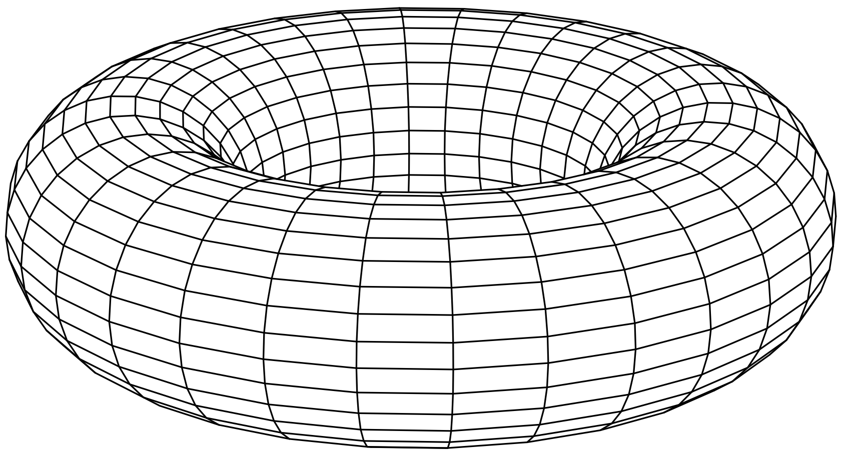


Figure 66. *Torus*

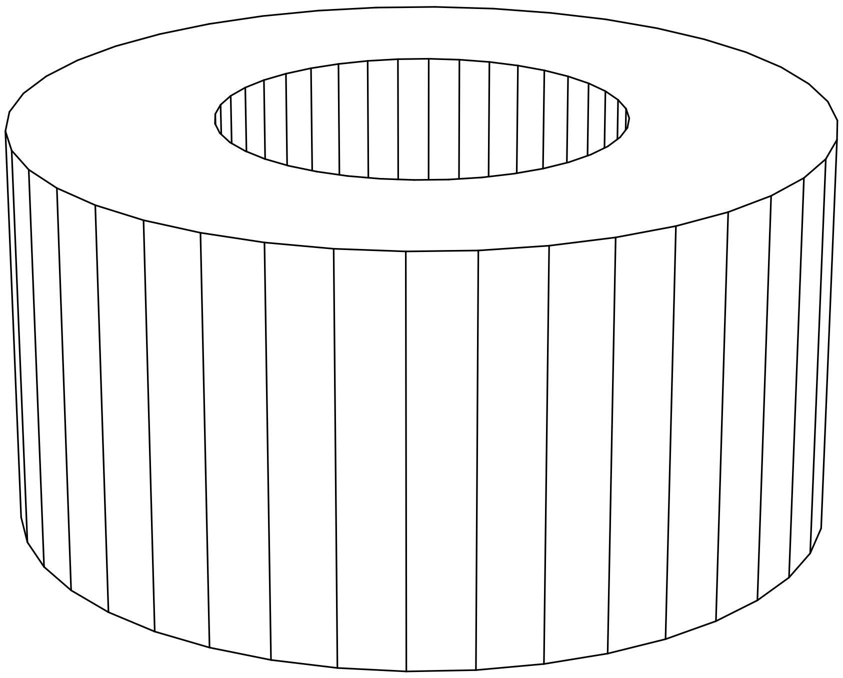


Figure 67. *Hollow cylinder*

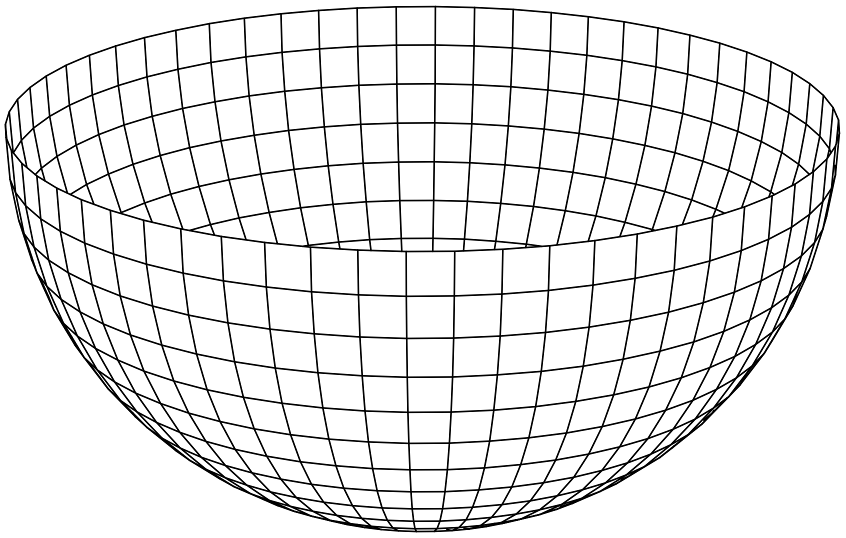


Figure 68. *Semi-sphere*

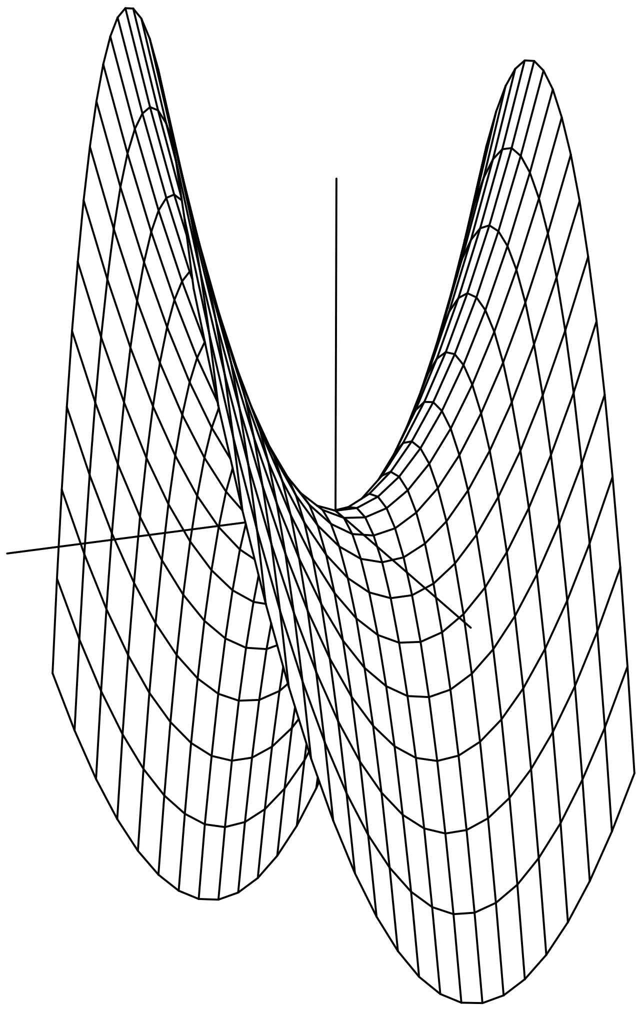


Figure 69. *Function surface*