CAD Software

CAD software can be divided based upon the technology used:

1. 2-D drawing. Its applications include,
   - mechanical part drawing
   - printed-circuit board design and layout
   - facilities layout
   - cartography
2. Basic 3-D drawing (such as wire-frame modelling)
3. Sculptured surfaces (such as surface modelling)
4. 3-D solid modelling
5. Engineering analysis

Some of the commonly available functions provided by CAD software are:

- Picture manipulation: add, delete, and modify geometry and text.
- Display transformation: scaling, rotation, pan, zoom, and partial erasing.
- Drafting symbols: standard drafting symbols.
- Printing control: output device selection, configuration and control.
- Operator aid: screen menus, tablet overly, function keys.
- File management: create, delete, and merge picture files.

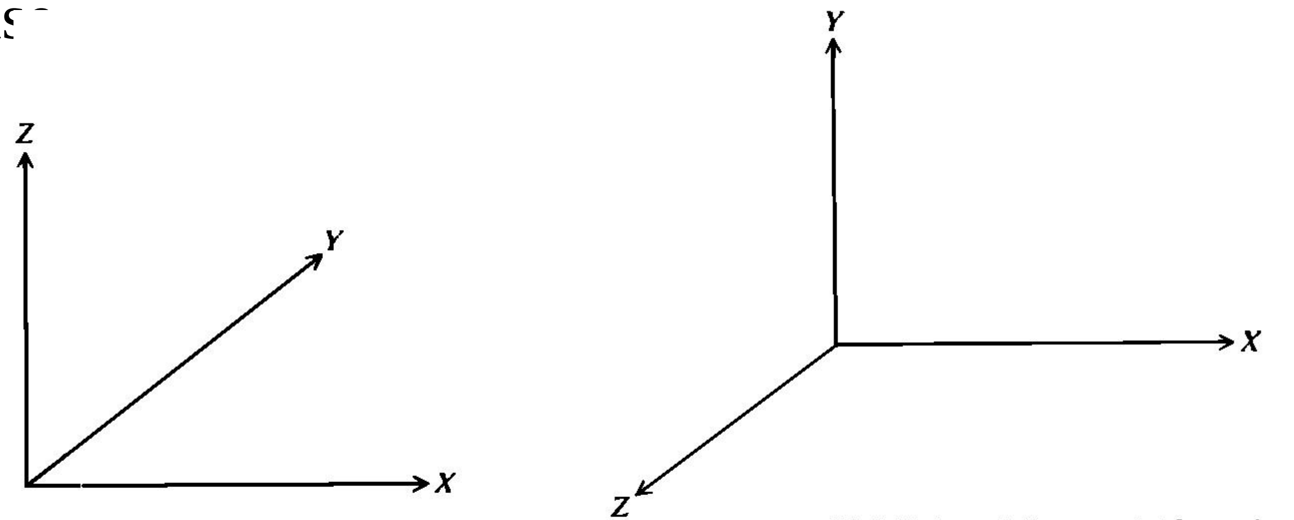
Coordinate Systems

1. The Model Coordinate System or (world coordinate systems) (MCS).
2. The Working Coordinate System (WCS).
3. The Screed Coordinate System (or device coordinate system) (SCS).

MCS : is the reference space of the model with respect to all the model geometrical data is stored.
WCS: is a convenient user-defined system that facilitates geometric construction.
SCS: is a two-dimensional device-dependent coordinate system whose origin is usually located ate the lower left corner of the graphic display.

The Model Coordinate System or (world coordinate systems) (MCS)

MCS is the only coordinate system that software recognizes when storing or retrieving geometrical information in or from a model databas



(*a*) *XY* plane defines model top view

(*b*) *XY* plane defines model front view

FIGURE 3-9
Possible orientations of MCS in space.

Example:

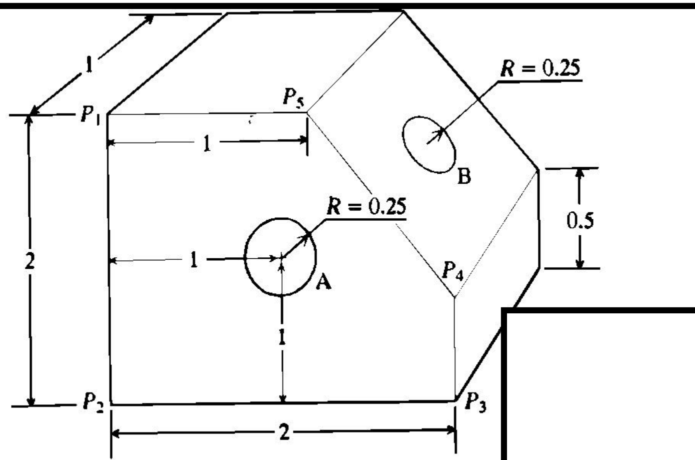
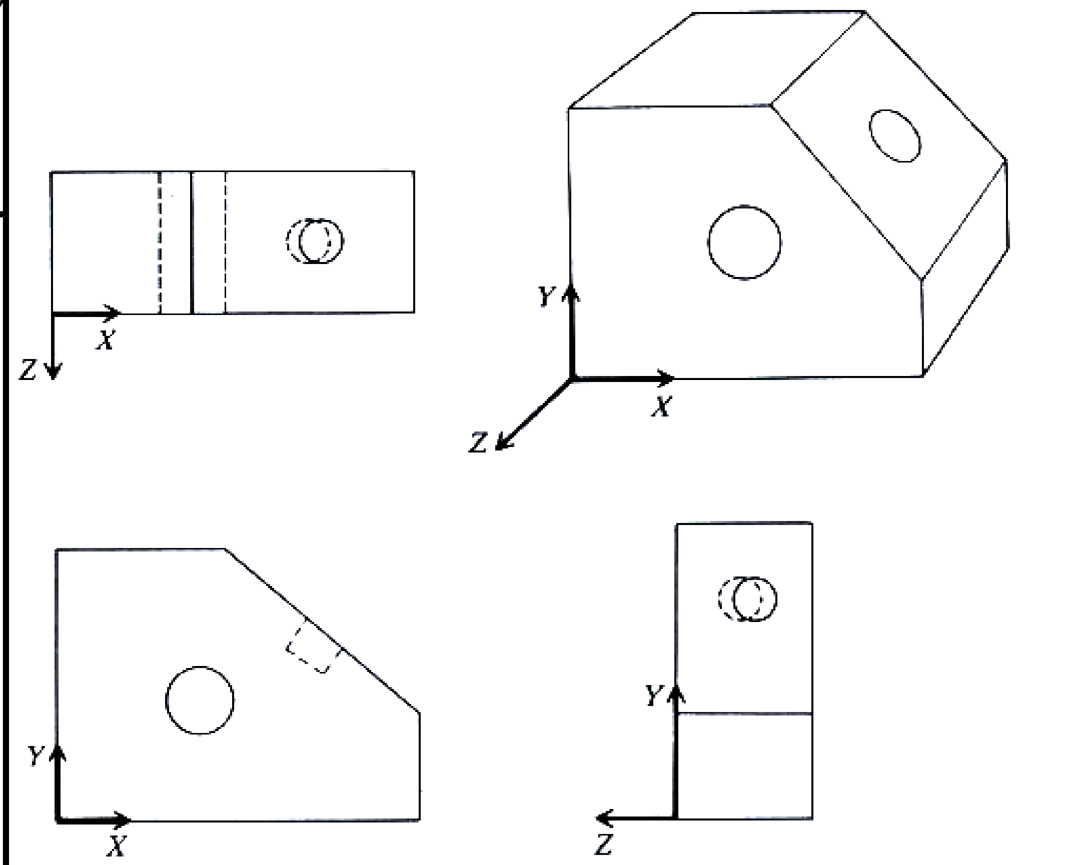


FIGURE 3-10
Geometric model of an object.



(*b*) Utilizing MCS of Fig. 3-9*b*

FIGURE 3-12
Views of object shown in Fig. 3-10.

The Working Coordinate System (user coordinate system) (WCS).

The software calculates the corresponding homogeneous transformation matrix between WCS and MCS to convert the inputs into coordinates relative to the MCS before sorting them in the database.

$$P = [T]P_W \tag{3.1}$$

where P is the position vector of a point relative to the MCS and P_W is the vector of a point relative to the active WCS. Each vector is given by

$$P = [x \quad y \quad z \quad 1]^T \tag{3.2}$$

The matrix $[T]$ is the homogeneous transformation matrix. It is a 4×4 matrix and is given by

$$[T] = \left[\begin{array}{ccc|c} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} {}^M_W[R] & {}^M P_{W,\text{org}} \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right] \tag{3.3}$$

where ${}^{M}_{W}[R]$ is the rotation matrix that defines the orientation of the WCS relative to the MCS and ${}^{M}P_{W,\,\mathrm{org}}$ is the position vector that describes the origin of the WCS relative to the MCS. The columns of ${}^{M}_{W}[R]$ give the direction cosines of the unit vectors in the X_W, Y_W, and Z_W directions relative to the MCS, as shown in Fig. 3-13.

The WCS serves another function during geometric construction. Its $X_W Y_W$ plane is used by the software as the default plane of circles. A circle plane is usually not defined using its center and radius. In addition, the Z_W axis of a WCS can be useful in defining a projection direction which may be helpful in geometric construction.

Example 3.2. Write a procedure to construct the holes shown in the model used in Example 3.1. Use the MCS shown in Fig. 3-9*a*.

Solution. Let us assume that the user has defined the $(WCS)_1$ as shown in Fig. 3-14 to construct the model without the holes. The procedure to construct the holes becomes:

1. With the $(WCS)_1$ active, construct circle A with center (1, 1, 0) and radius 0.25.
2. Construct hole A by projecting circle A at a distance of -1.0 (in the opposite direction to Z_{W1}).
3. Define $(WCS)_2$ as shown by using points E_1, E_2, and E_3.
4. Construct circle B with center (x_c, y_c) and radius 0.25. The center can easily be found implicitly as the midpoint of line $E_2 E_3$.
5. Repeat step 2 but with a distance -0.5 (in the opposite direction to Z_{W2}).

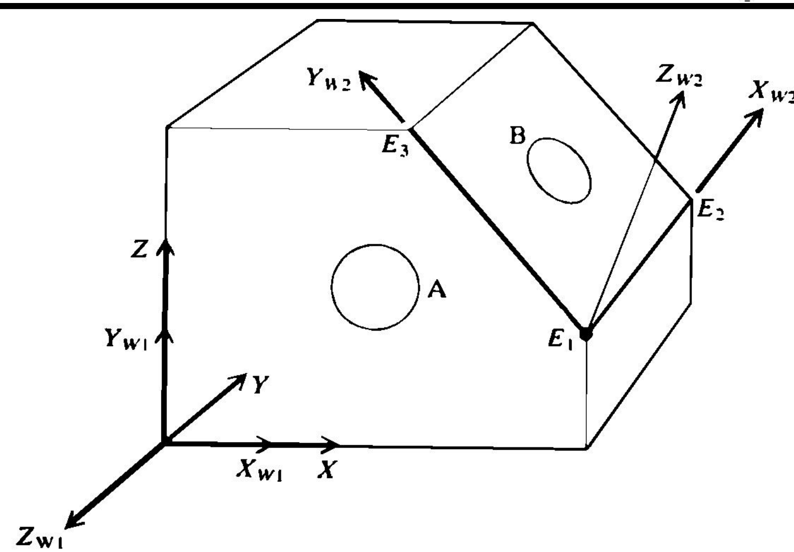


FIGURE 3-14
WCSs required to construct holes A and B.

The coordinates in step 1 are given relative to $(WCS)_1$ which is active at the time of construction. With reference to Fig. 3-14, these coordinates are (1, 0, 1) relative to the MCS and these are the values that are stored in the model database. To verify this, using Eq. (3.3) we can write:

$$C = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{3.4}$$

A similar approach can be followed to find the center of the hole B and is left as an exercise for the reader at the end of the chapter (see Prob. 3.4).

The Screed Coordinate System (or device coordinate system) (SCS).

The range and measurement unit of an SCS can be determined in three different methods:

1. pixel grid: a 1024x1024 display has an SCS with a range of (0,0) to (1024, 1024).
2. Normalized coordinate system. The range of the SCS be chosen from (0,0) to (1,1).
3. Drawing size that user chooses.

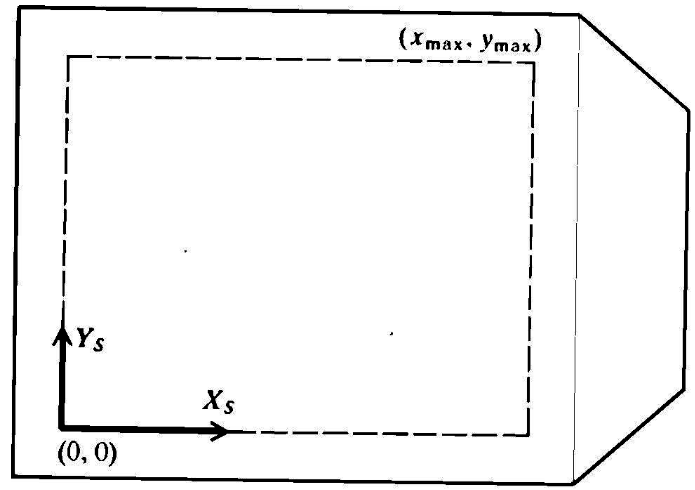


FIGURE 3-15
Typical SCS.

Example 3.3. A view is typically defined by a view origin and a view window. Use the possible three methods (pixel grid, normalized values, drawing size) to define the four views shown in Fig. 3-16. The origins of the top, front, and right views (O_T, O_F, O_R) must line up as shown and the origin of the isometric view O_I is assumed in the middle of its window. Assume a 1000×1000 pixel grid, a maximum normalized value of 1, and size A drawing for the three methods respectively.

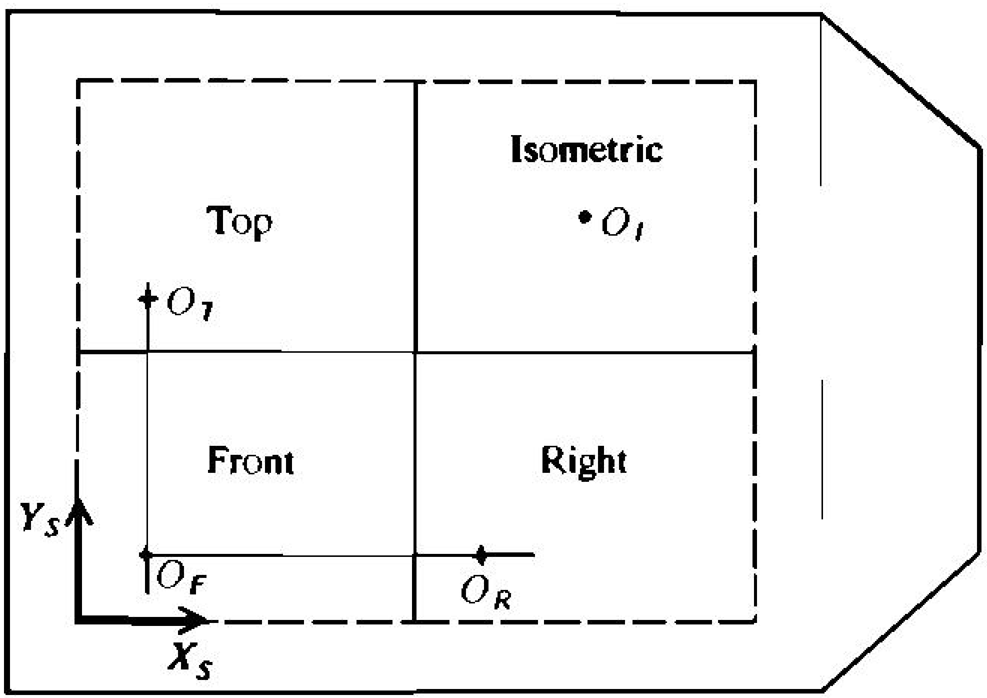


FIGURE 3-16
A typical screen layout.

Solution. A view window (viewport) is usually defined by one of its diagonals. Assume that the lower left ($x_{V,\min}$, $y_{V,\min}$) and the top right ($x_{V,\max}$, $y_{V,\max}$) corners of a view window are used to define such a window. The coordinates required to define the above views become

	Method		
	Pixel grid		
View	(x_0, y_0)	$(x_{V,\min}, y_{V,\min})$	$(x_{V,\max}, y_{V,\max})$
Front	10, 10	0, 0	500, 500
Top	10, 510	0, 500	500, 1000
Right	510, 10	500, 0	1000, 500
Isometric	750, 750	500, 500	1000, 1000

	Normalized values		
View	(x_0, y_0)	$(x_{v,\min}, y_{v,\min})$	$(x_{v,\max}, y_{v,\max})$
Front	0.01, 0.01	0, 0	0.5, 0.5
Top	0.01, 0.51	0, 0.5	0.5, 1.0
Right	0.51, 0.01	0.5, 0	1.0, 0.5
Isometric	0.75, 0.75	0.5, 0.5	1.0, 1.0

	Drawing size		
View	(x_0, y_0)	$(x_{v,\min}, y_{v,\min})$	$(x_{v,\max}, y_{v,\max})$
Front	0.5, 0.5	0, 0	5.5, 4.25
Top	0.5, 4.75	0, 4.25	5.5, 8.5
Right	6.0, 0.5	5.5, 0	11, 4.25
Isometric	8.25, 6.375	5.5, 4.25	11, 8.5

Window-To-Viewport Mapping

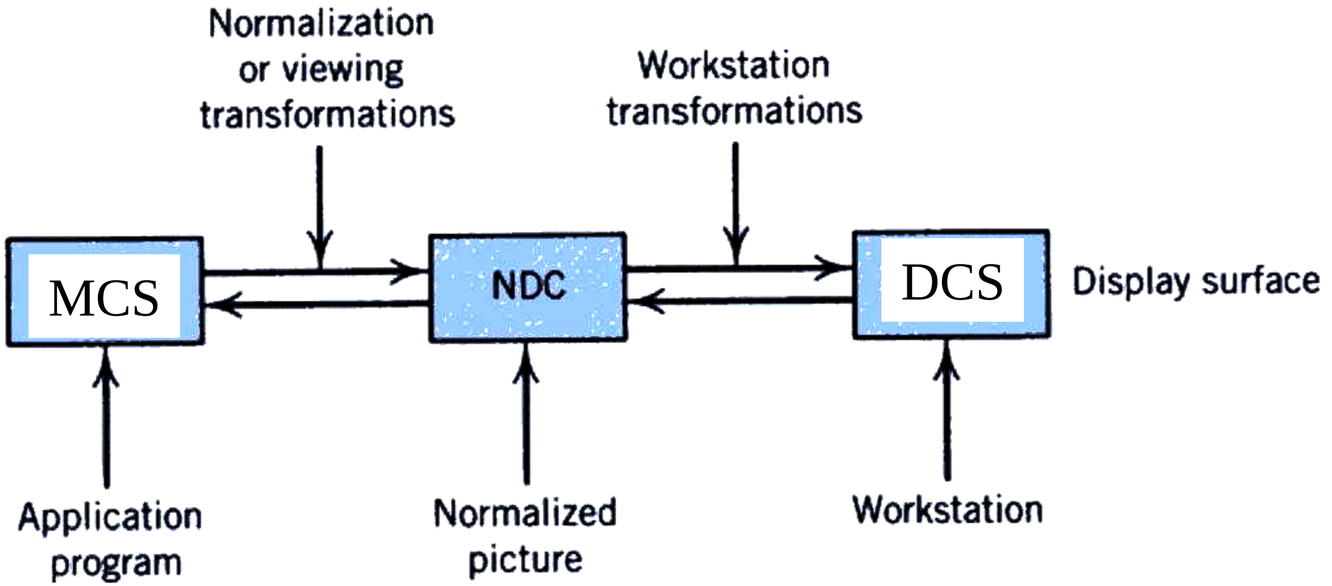
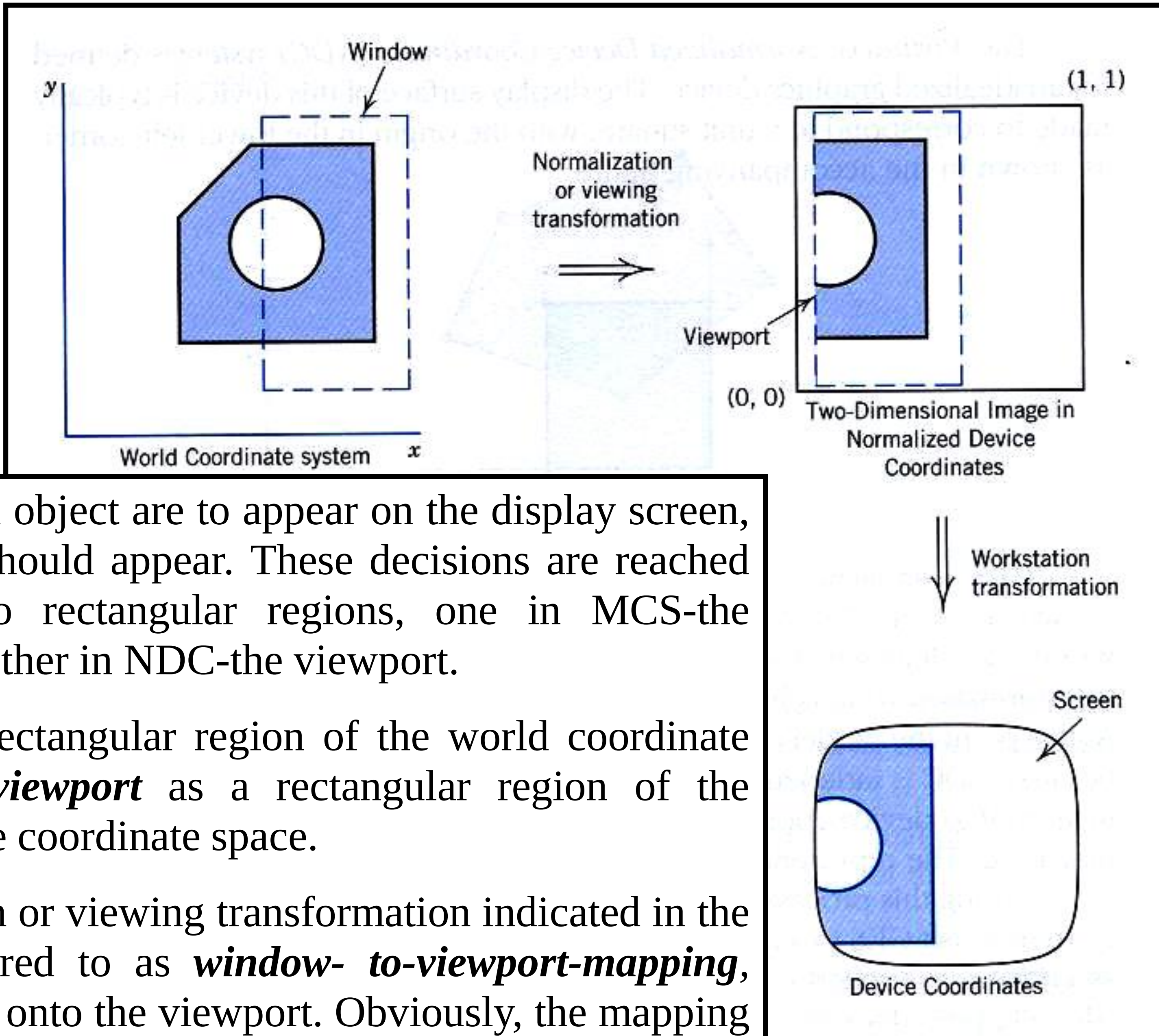


FIGURE 5.3 GKS Coordinate systems and transformations.

NDC = Normalized Device Coordinate System

Window and viewport definitions

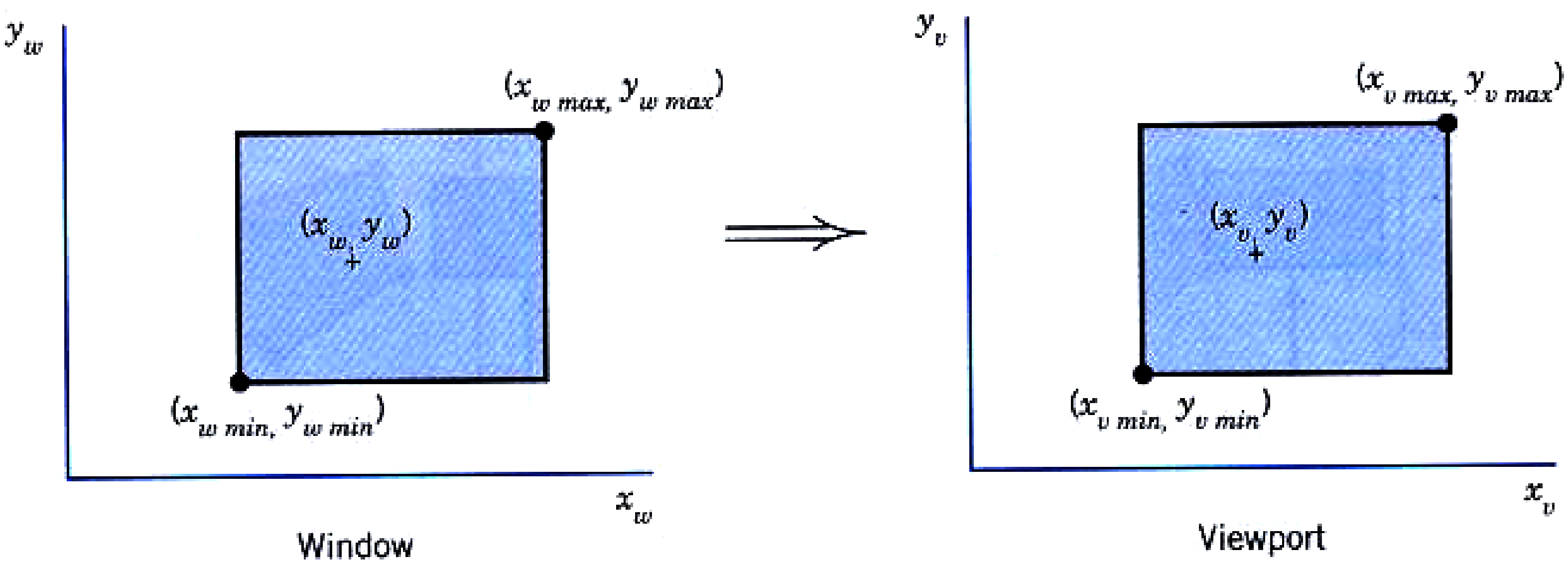


Which parts of an object are to appear on the display screen, and where they should appear. These decisions are reached by choosing two rectangular regions, one in MCS-the window-and the other in NDC-the viewport.

A ***window*** as a rectangular region of the world coordinate space, and the ***viewport*** as a rectangular region of the normalized device coordinate space.

The normalization or viewing transformation indicated in the figure, also referred to as ***window- to-viewport-mapping***, maps the window onto the viewport. Obviously, the mapping is carried over to the device through a workstation transformation.

Window-to-viewport mapping



A window-to-viewport mapping can be expressed by the following relationships, based on elements shown in Figure 5.5:

$$\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}} \tag{5.1}$$

and

$$\frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}} = \frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}} \tag{5.2}$$

So that

$$x_v = (x_w - x_{wmin}) \left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}} \right) + x_{vmin} \tag{5.3}$$

$$y_v = (y_w - y_{wmin}) \left(\frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}} \right) + y_{vmin} \tag{5.4}$$

The terms

$$\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}} \right) \quad \text{and} \quad \left(\frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}} \right) \tag{5.5}$$

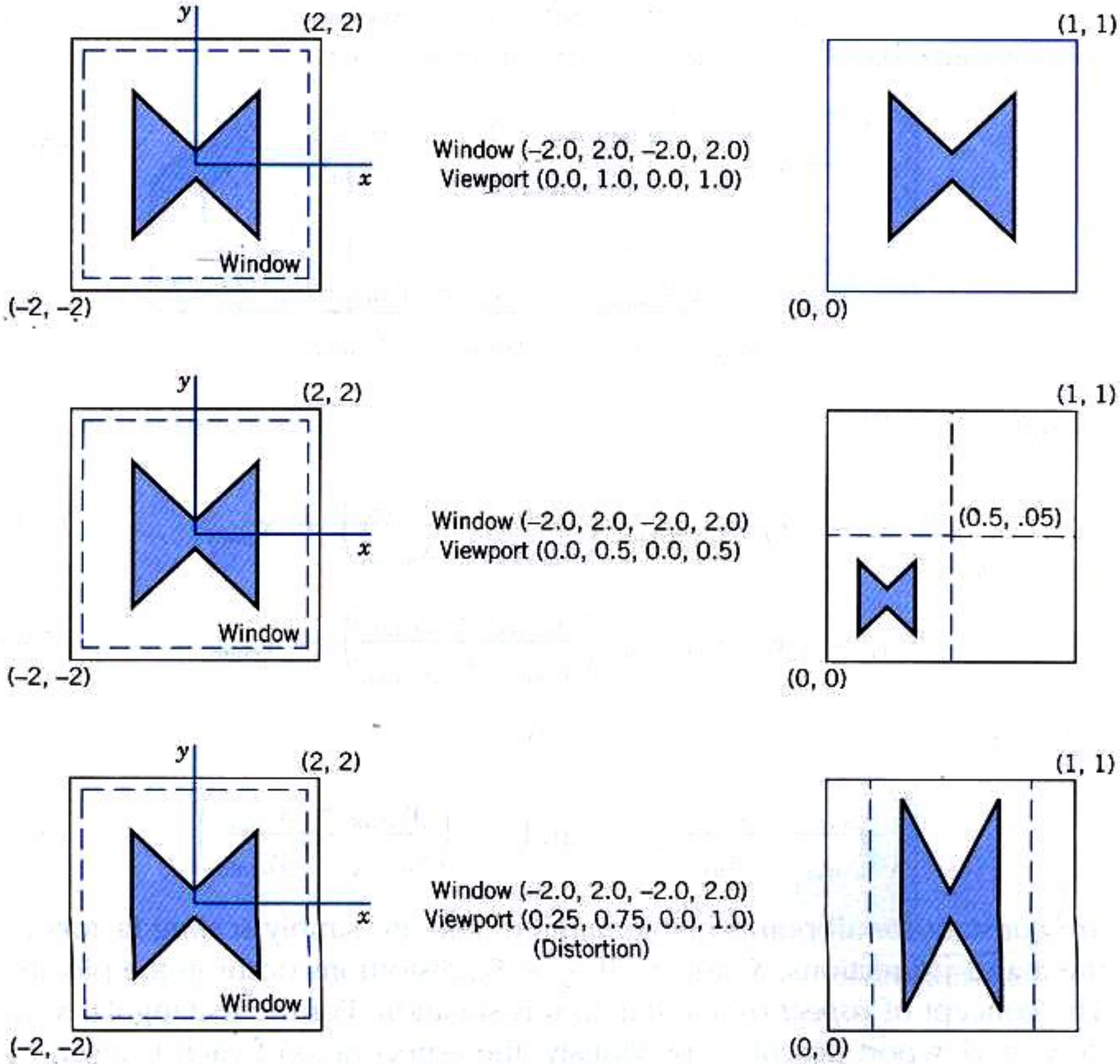
are constant for all points being mapped, and are simply scaling factors in the x and y directions, S_x and S_y. If $S_x \neq S_y$, distortions occur in the picture. The concept of *aspect ratio* refers to this situation. For the rectangular window or viewport described previously, the aspect ratio of each is given by the ratio of width to height:

$$AR = \frac{x_{max} - x_{min}}{y_{max} - y_{min}} \tag{5.6}$$

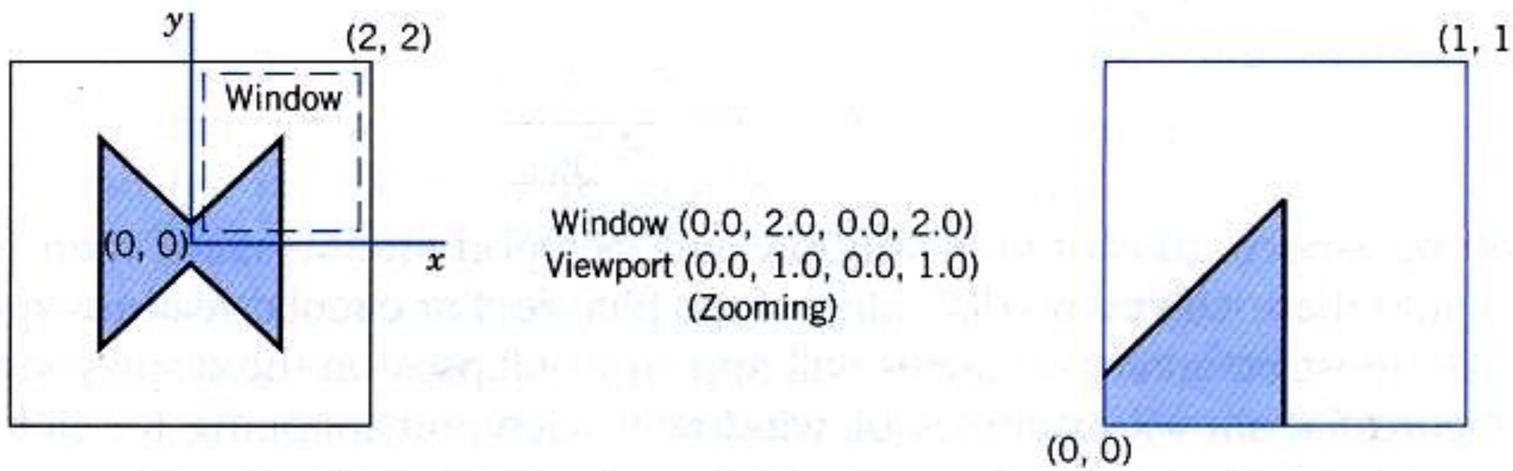
If the aspect ratios of both window and viewport are the same, then $S_x = S_y$ and there will be no distortion of the picture. For circular features special care must be taken, or circles will appear as ellipses on the display screen. Figure 5.6 shows examples of window-to-viewport mapping for different conditions. Notice that the parameters inside the parentheses are (x_{min}, x_{max}, y_{min}, y_{max}). The "zooming" effect shown in Figure 5.6 is obtained by mapping a smaller window to the whole viewport. It gives the impression that the user is located closer to the object.

Example 5.1

Find the transformation matrix that will map points contained in a window whose lower left corner is at (2,2) and upper right corner is at (6,5) onto a normalized viewport that has a lower left corner at $(\frac{1}{2},\frac{1}{2})$ and upper right corner at (1,1).



y
(2, 2)
x
Window
(-2, -2)
Window (-2.0, 2.0, -2.0, 2.0)
Viewport (0.0, 1.0, 0.0, 1.0)
(1, 1)
(0, 0)
y
(2, 2)
x
Window
(-2, -2)
Window (-2.0, 2.0, -2.0, 2.0)
Viewport (0.0, 0.5, 0.0, 0.5)
(1, 1)
(0.5, .05)
(0, 0)
y
(2, 2)
x
Window
(-2, -2)
Window (-2.0, 2.0, -2.0, 2.0)
Viewport (0.25, 0.75, 0.0, 1.0)
(Distortion)
(1, 1)
(0, 0)



Solution

The window/viewport parameters are

$$
\begin{aligned}
x_{w\min} &= 2 & x_{v\min} &= ½ \\
x_{w\max} &= 6 & x_{v\max} &= 1 \\
y_{w\min} &= 2 & y_{v\min} &= ½ \\
y_{w\max} &= 5 & y_{v\max} &= 1
\end{aligned}
$$

Therefore, based on Eqs. 5.3 and 5.4,

$$S_x = \frac{1 - 1/2}{6 - 2} = \frac{1}{8}$$

$$S_y = \frac{1 - 1/2}{5 - 2} = \frac{1}{6}$$

Equations 5.3 and 5.4 can be rewritten as:

$$x_v = (x_w - x_{w\min})\, s_x + x_{v\min}$$

$$y_v = (y_w - y_{w\min})\, s_y + y_{v\min}$$

Or, in matrix form,

$$[x_v \quad y_v \quad 1] = [x_w \quad y_w \quad 1] \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ (-s_x \cdot x_{w\min} + x_{v\min}) & (-s_y \cdot y_{w\min} + y_{v\min}) & 1 \end{bmatrix}$$

And the transformation matrix becomes:

$$M_{\text{map}} = \begin{bmatrix} \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{6} & 0 \\ \frac{1}{4} & \frac{1}{6} & 1 \end{bmatrix}$$

Geometric Modeling

Geometric modelling refers to a set of techniques concerned mainly with developing efficient representations of geometric aspects of a design. Therefore, geometric modelling is a fundamental part of all CAD tools.

Geometric modeling is the basic of many applications such as:

- Mass property calculations.
- Mechanism analysis.
- Finite-element modelling.
- NC programming.

Requirements of geometric modelling include:

- Completeness of the part representation.
- The modelling method should be easy to use by designers.
- Rendering capabilities (which means how fast the entities can be accessed and displayed by the computer).

Geometric Modeling Approaches

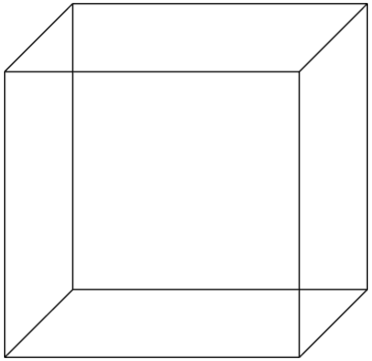
The basic geometric modelling approaches available to designers on CAD/CAM systems are:

1. Wire-frame modeling.
2. Surface modeling.
3. Solid modeling.

Wire-frame Modeling

Wire-frame modelling uses points and curves (i.e. lines, circles, arcs), and so forth to define objects.

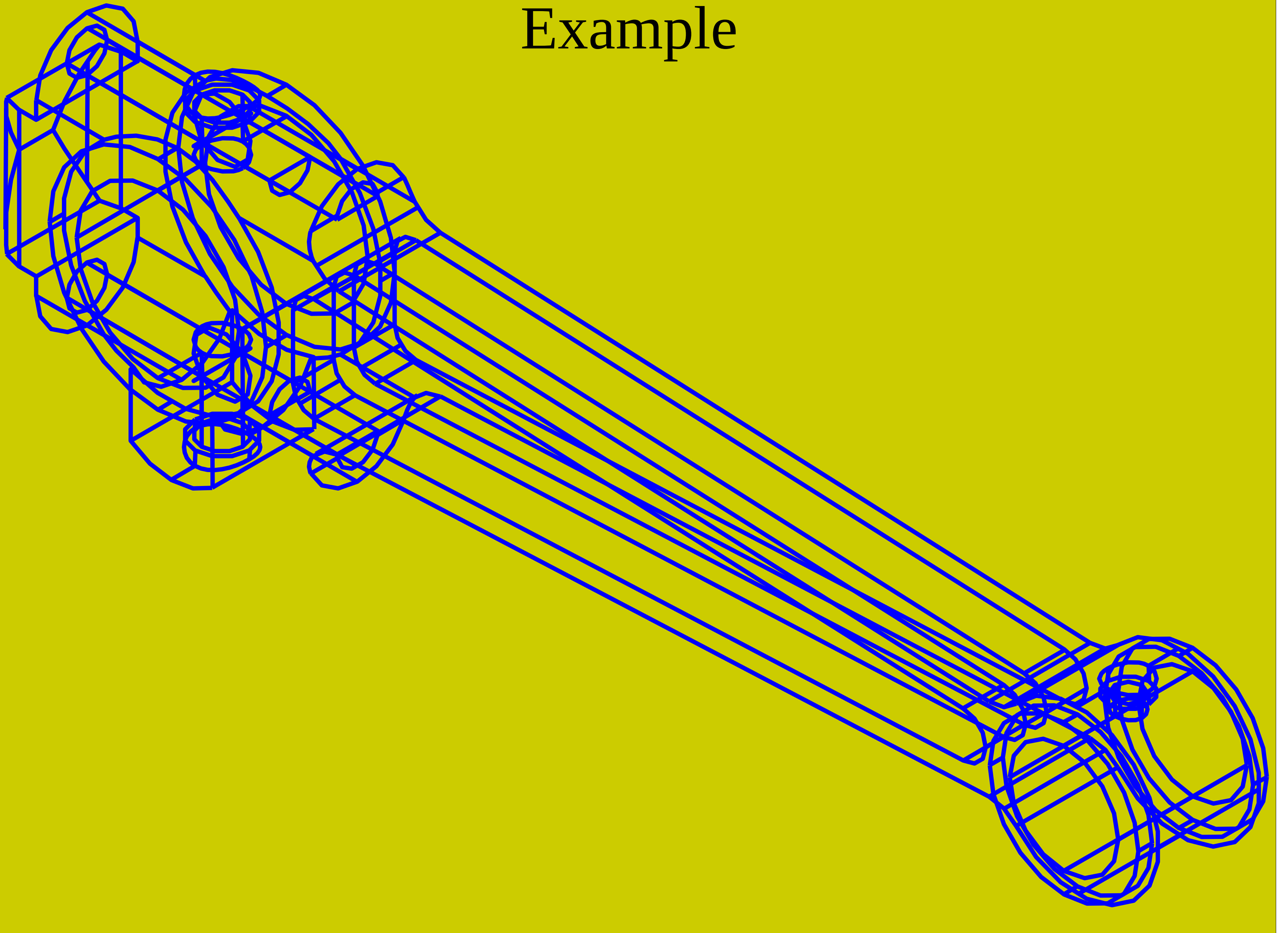
The user uses edges and vertices of the part to form a 3-D object



Wire-frame model

part

Example



Surface Modeling

Surface modeling is more sophisticated than wireframe modeling in that it defines not only the edges of a 3D object, but also its surfaces.

In surface modeling, objects are defined by their bounding faces.

Examples

SURFACE ENTITIES

Similar to wireframe entities, existing CAD/CAM systems provide designers with both analytic and synthetic surface entities.

Analytic entities include :

- Plane surface,
- Ruled surface,
- Surface of revolution, and
- Tabulated cylinder.

Synthetic entities include

- The bicubic Hermite spline surface,
- B-spline surface,
- Rectangular and triangular Bezier patches,
- Rectangular and triangular Coons patches, and
- Gordon surface.

Plane surface. This is the simplest surface. It requires three noncoincident points to define an infinite plane.

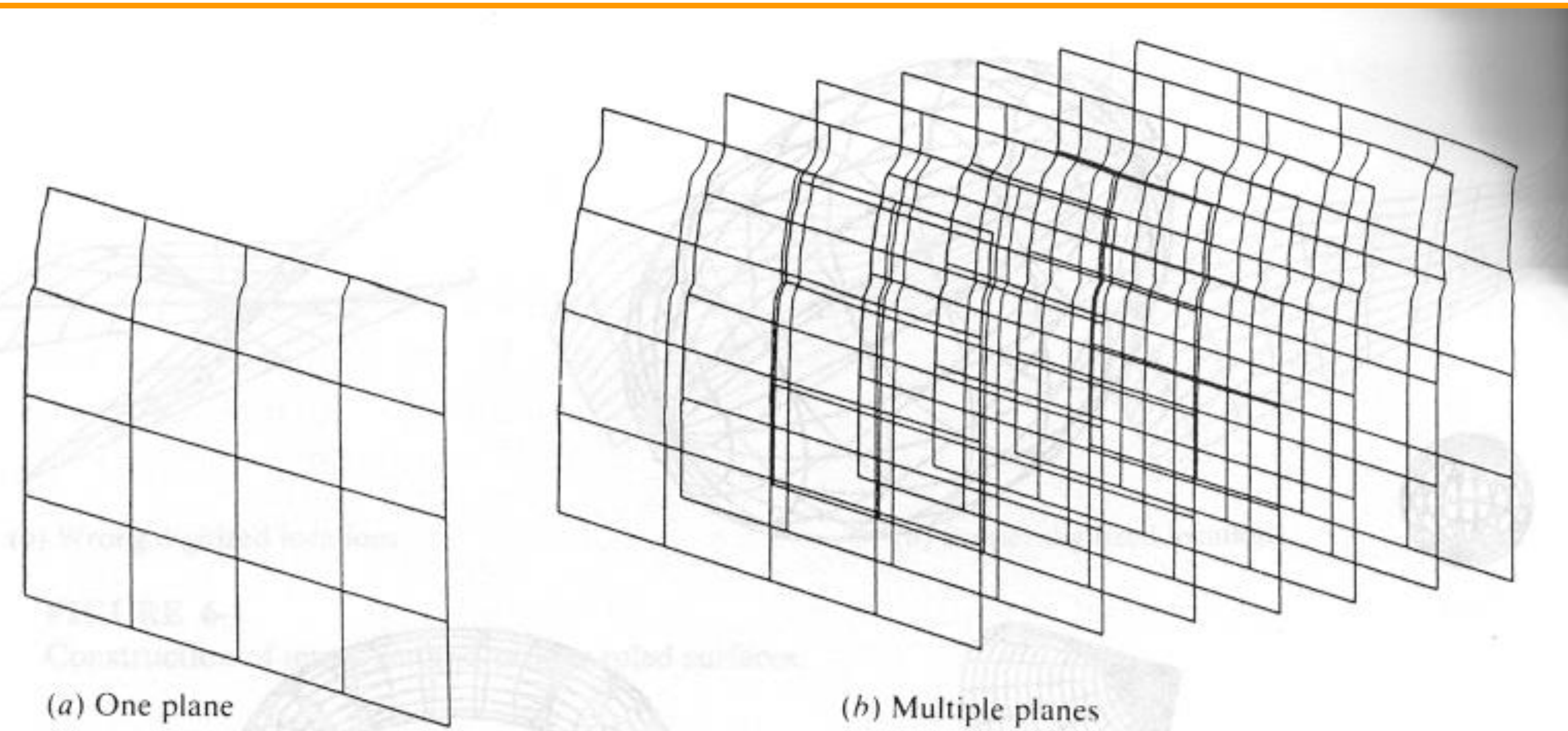


FIGURE 6-4
Plane surface.

Ruled (lofted) surface. This is a linear surface. It interpolates linearly between two boundary curves that define the surface (rails). Rails can be any wireframe entity. This entity is ideal to represent surfaces that do not have any twists or kinks.

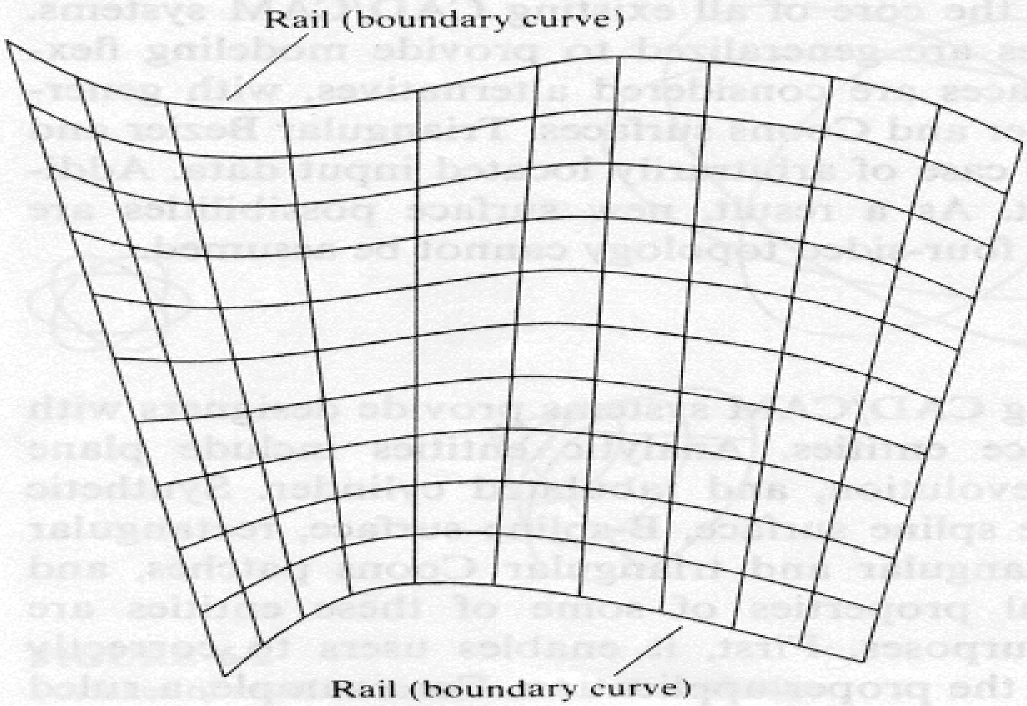


FIGURE 6-5
Ruled surface.

Surface of revolution. This is an axisymmetric surface that can model axisymmetric objects. It is generated by rotating a planar wireframe entity in space about the axis of symmetry a certain angle.

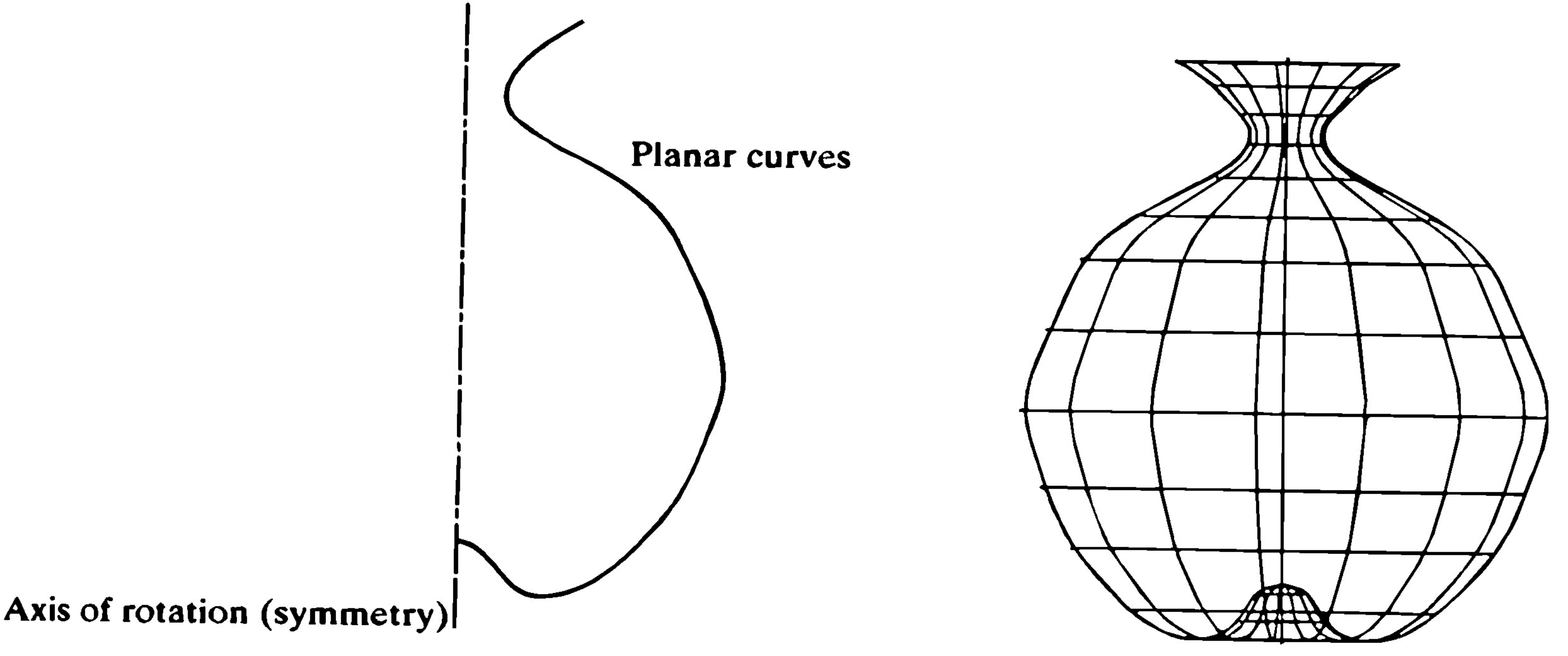


FIGURE 6-6
Surface of revolution.

Tabulated cylinder. This is a surface generated by translating a planar curve a certain distance along a specified direction (axis of the cylinder).

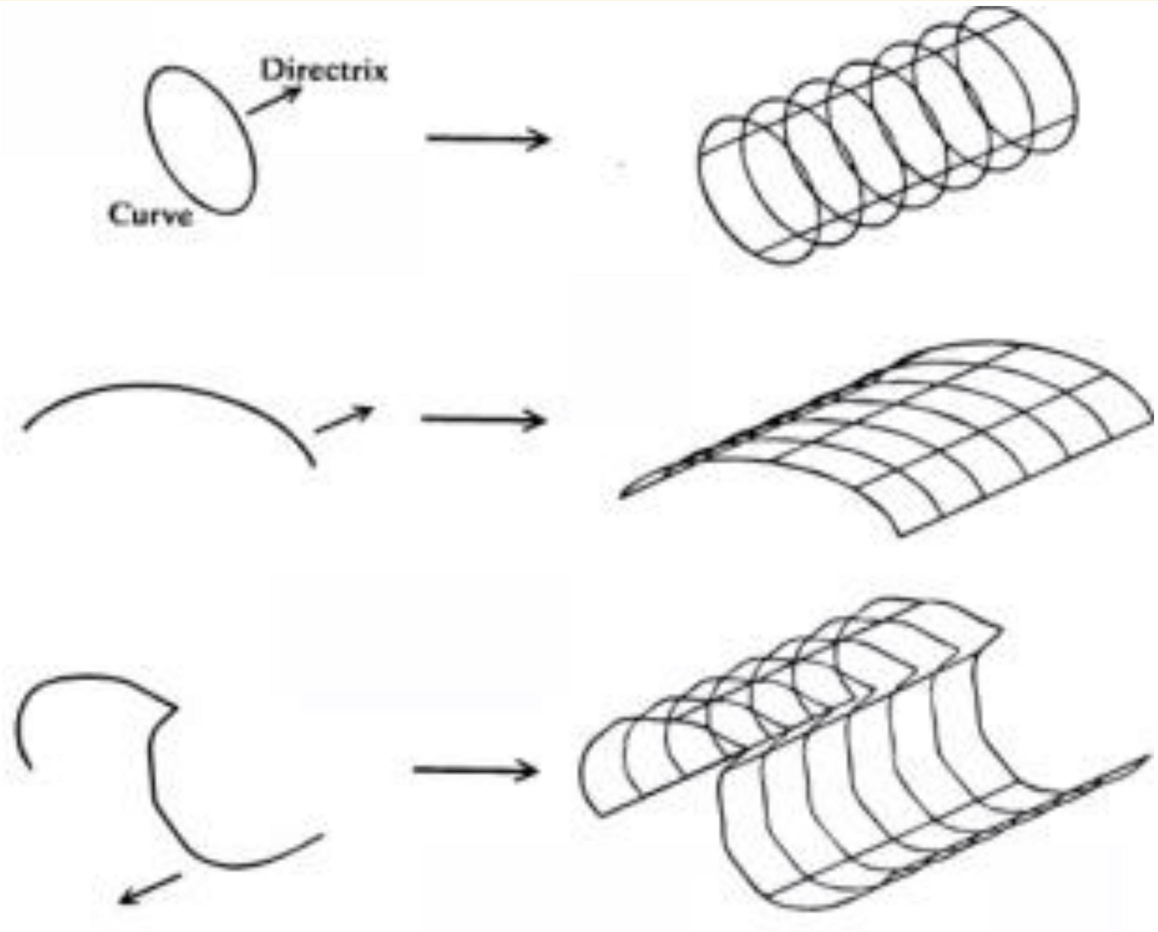


FIGURE 6-7
Tabulated cylinder.

Bezier surface. This is a surface that approximates given input data. It is different from the previous surfaces in that it is a synthetic surface. Similarly to the Bezier curve, it does not pass through all given data points. It is a general surface that permits, twists, and kinks . The Bezier surface allows only global control of the surface.

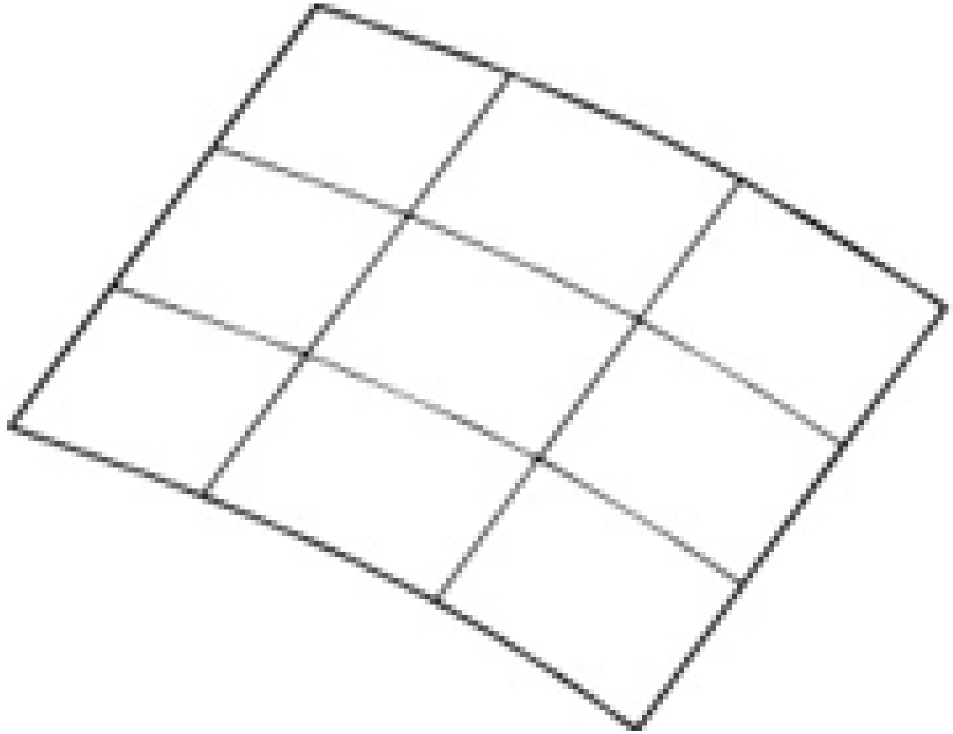
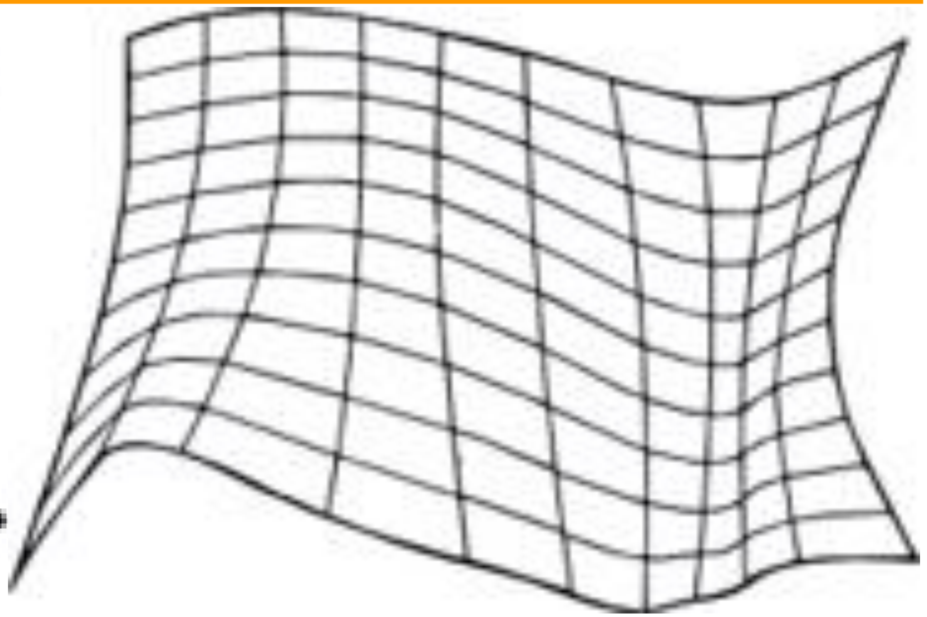


FIGURE 6-8
Bezier surface.

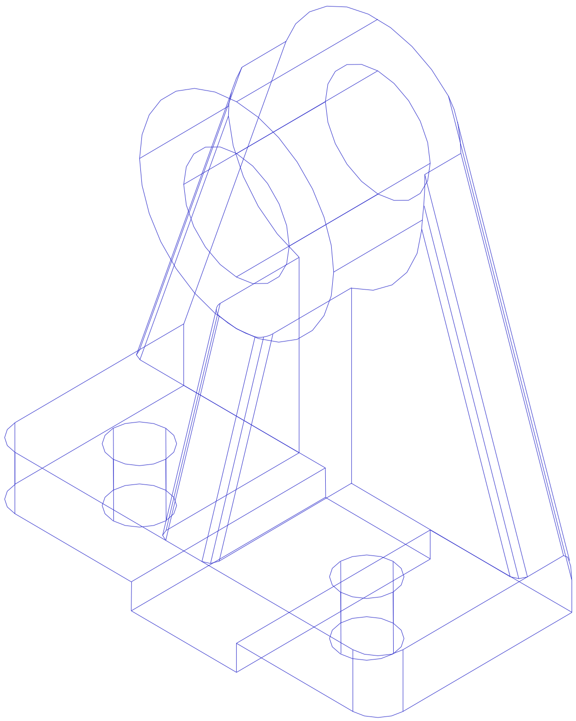
B-spline surface. This is a surface that can approximate or interpolate given input data (Fig. 6-9). It is a synthetic surface. It is a general surface like the Bezier surface but with the advantage of permitting local control of the surface.

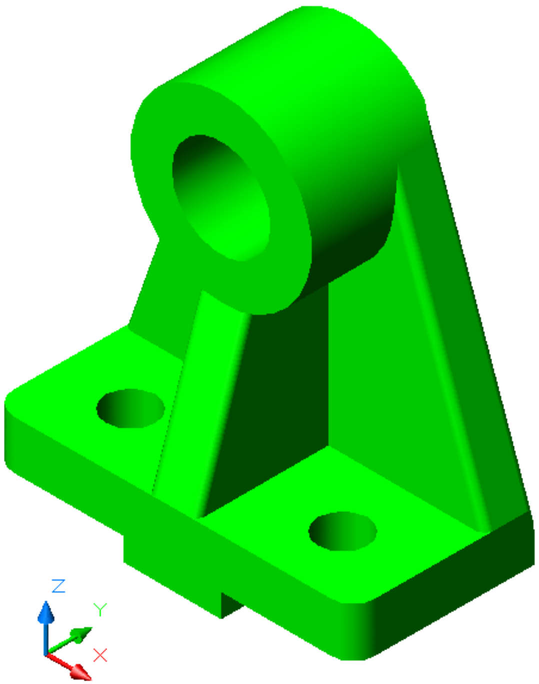
(a) Data points



Solid Modeling

Solid models give designers a complete descriptions of constructs, shape, surface, volume, and density.

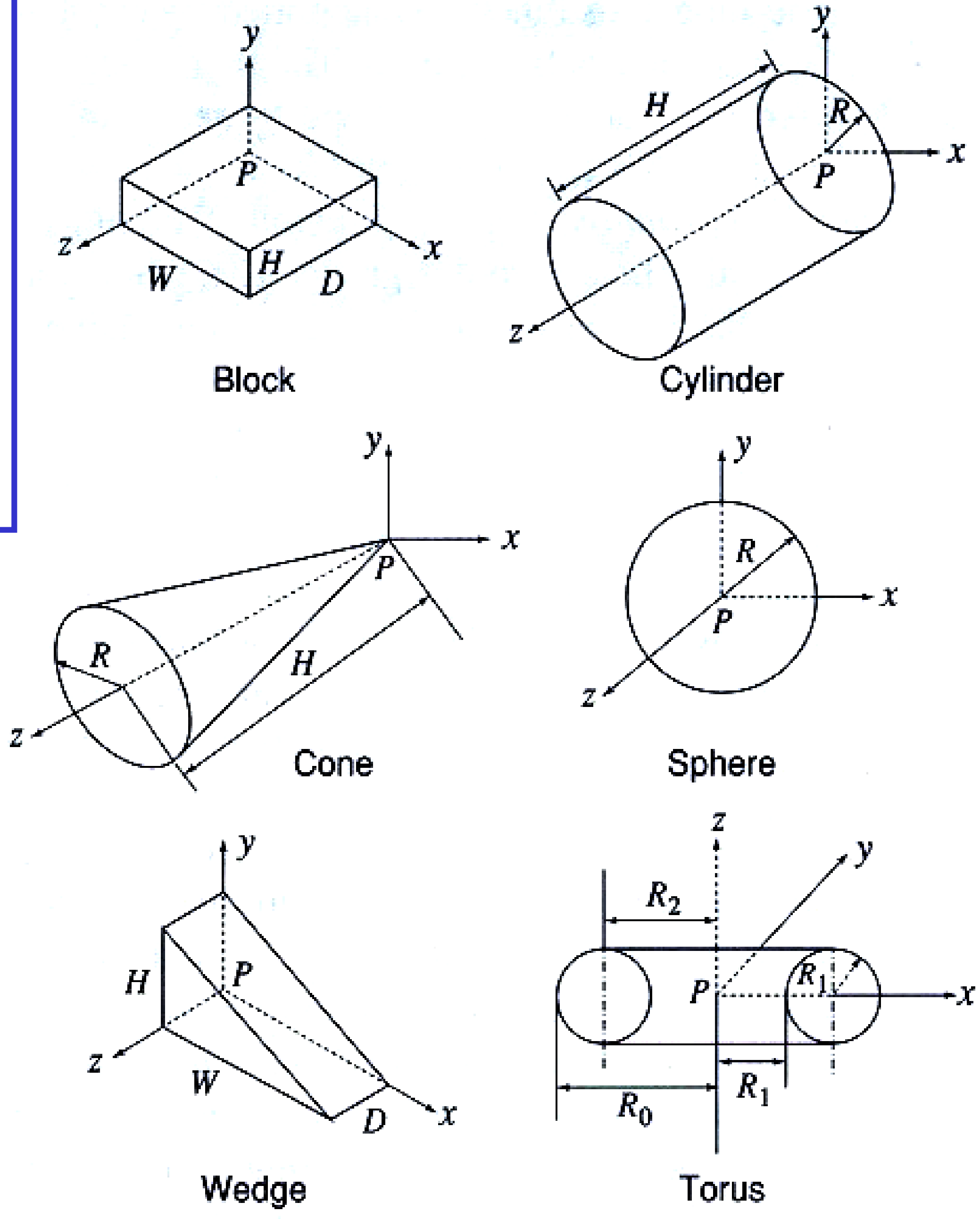




In CAD systems there are a number of representation schemes for solid modeling include:

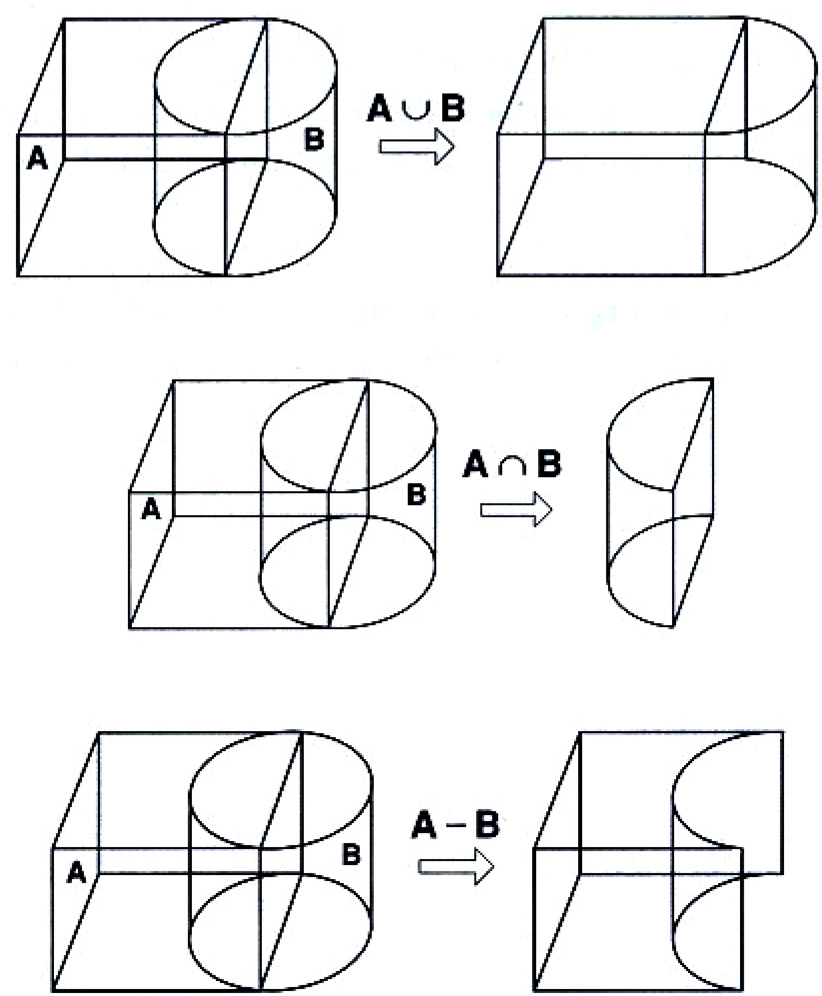
- Primitive creation functions.
- Constructive Solid Geometry (CSG)
- Sweeping
- Boundary Representation (BREP)

Primitive creation functions: These functions retrieve a solid of a simple shape from among the primitive solids stored in the program in advance and create a solid of the same shape but of the size specified by the user.



Constructive Solid Geometry (CSG)

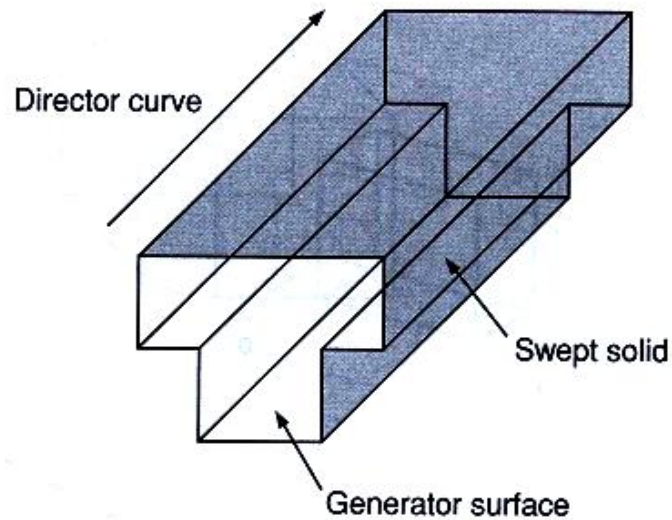
CSG uses primitive shapes as building blocks and Boolean set operators (U union, difference, and ∩ intersection) to construct an object.

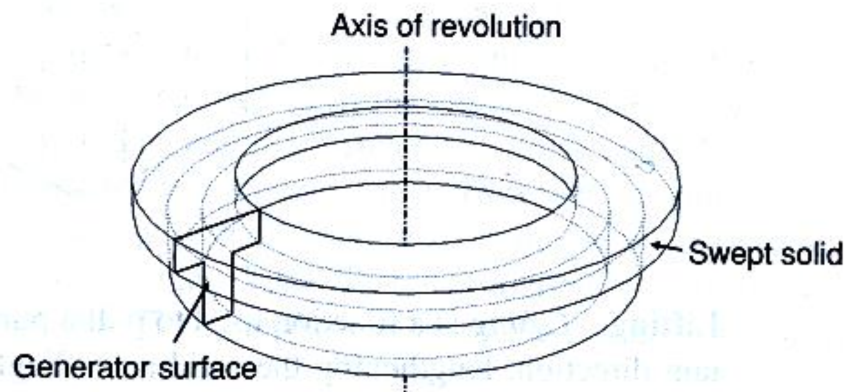


Example

Sweeping

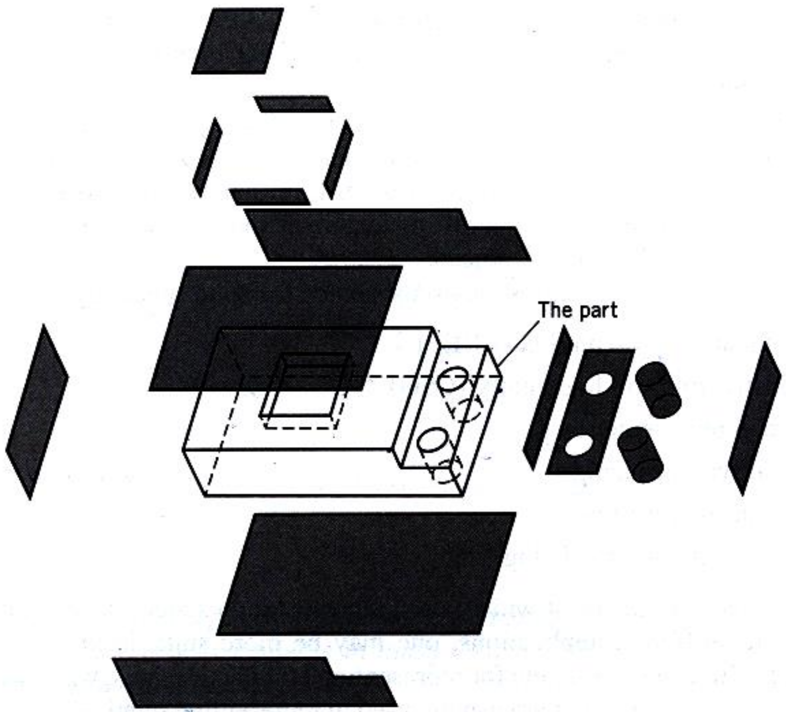
Sweeping *Sweeping* is a modeling function in which a planar closed domain is translated or revolved to form a solid. When the planar domain is translated, the modeling activity is called *translational sweeping;* when the planar region is revolved, it is called *swinging, or rotational sweeping*.

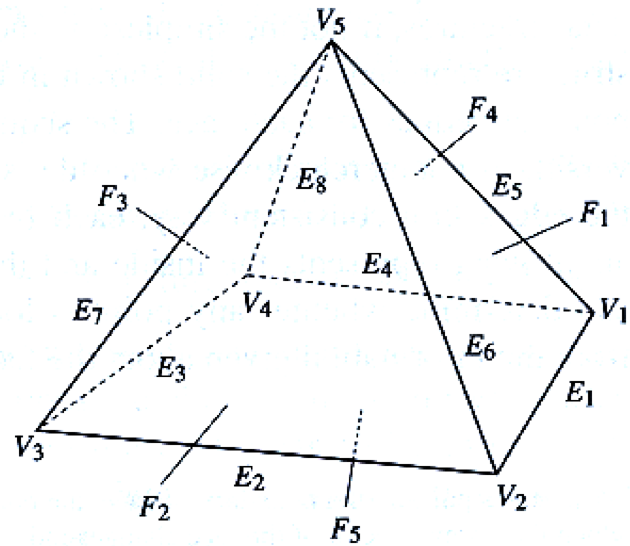




Boundary Representation

Objects are represented by their bounded faces.





B-Rep Data Structure

Three tables for storing B-Rep

Face Table

Face	*Edges*
F_1	E_1, E_5, E_6
F_2	E_2, E_6, E_7
F_3	E_3, E_7, E_8
F_4	E_4, E_8, E_5
F_5	E_1, E_2, E_3, E_4

Edge Table

Edge	*Vertices*
E_1	V_1, V_2
E_2	V_2, V_3
E_3	V_3, V_4
E_4	V_4, V_1
E_5	V_1, V_5
E_6	V_2, V_5
E_7	V_3, V_5
E_8	V_4, V_5

Vertex Table

Vertex	*Coordinates*
V_1	x_1, y_1, z_1
V_2	x_2, y_2, z_2
V_3	x_3, y_3, z_3
V_4	x_4, y_4, z_4
V_5	x_5, y_5, z_5
V_6	x_6, y_6, z_6