- To produce 2D images of a mathematically described 3D environment
- Issues:
  - Describing the environment: *Modeling* (mostly later)
  - Computing the image: *Rendering*

Graphics Toolkits

- Graphics toolkits typically take care of the details of producing images from geometry
- Input:
  - Where the objects are located and what they look like
  - Where the camera is and how it behaves
  - Parameters for controlling the rendering
- Output: Pixel data in a *framebuffer*
  - Data can be put on the screen
  - Data can be read back for processing (part of toolkit)

OpenGL

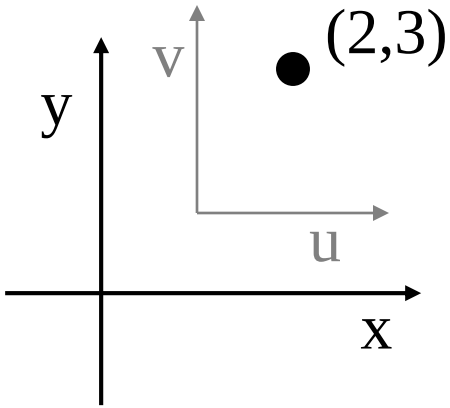
- OpenGL is an open standard graphics toolkit
  - Derived from GL toolkit
- Provides a range of functions for modelling, rendering and manipulating the framebuffer
- Why use it? Portable, hardware supported, simple and easy to program
- Alternatives: Direct3D, Java3D - more complex and less well supported respectively

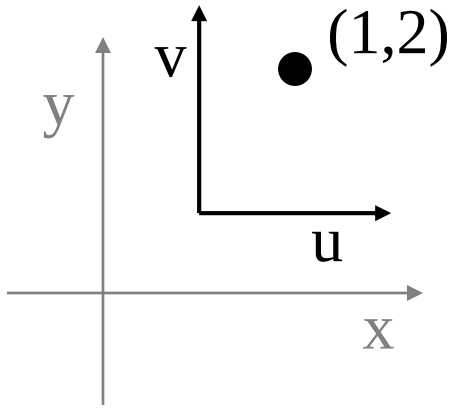
Coordinate Systems

- The use of *coordinate systems* is fundamental to computer graphics
- Coordinate systems are used to describe the locations of points in space
- Multiple coordinate systems make graphics algorithms easier to understand and implement

Coordinate Systems (2)

- **Different coordinate systems represent the same point in different ways**

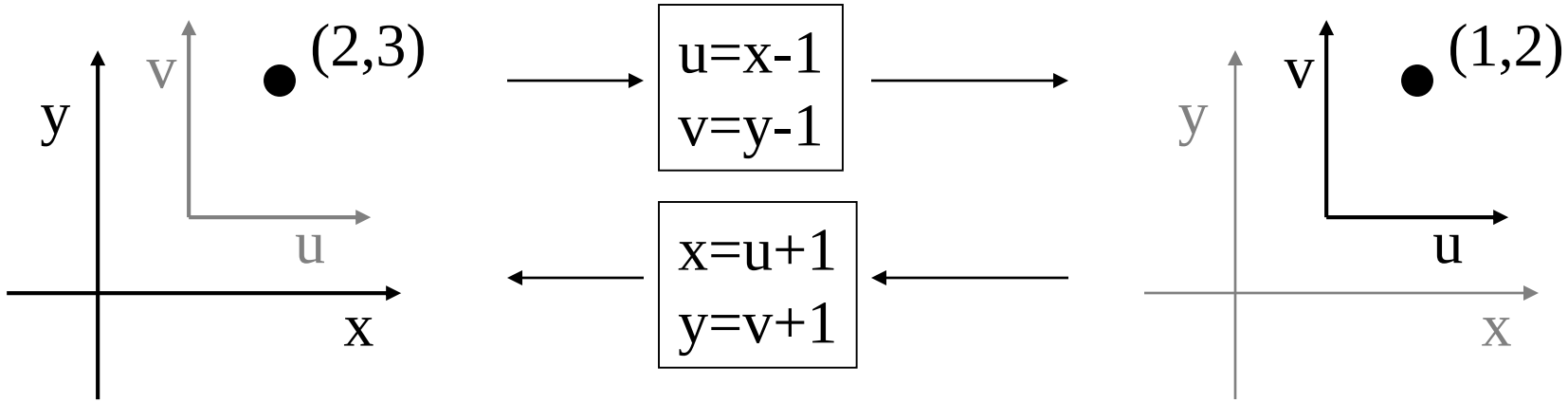




- Some operations are easier in one coordinate system than in another

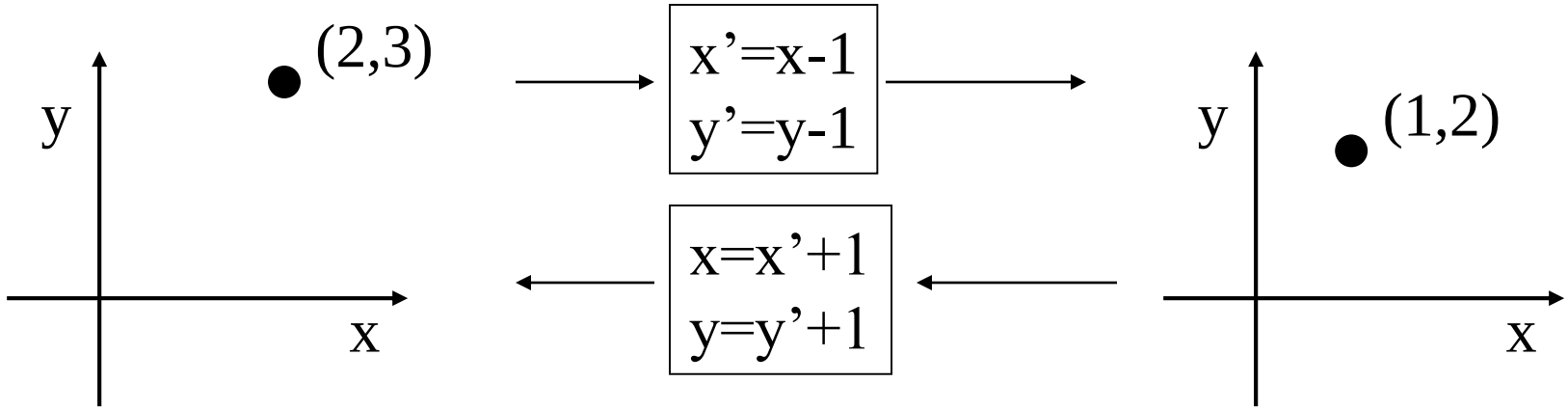
Transformations

- Transformations convert points between coordinate systems



Transformations (Alternate Interpretation)

- Transformations modify an object's shape and location in one coordinate system



2D Affine Transformations

- An *affine transformation* is one that can be written in the form:

$$x' = a_{xx}x + a_{xy}y + b_x$$

$$y' = a_{yx}x + a_{yy}y + b_y$$

or

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} \\ a_{yx} & a_{yy} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

Why Affine Transformations?

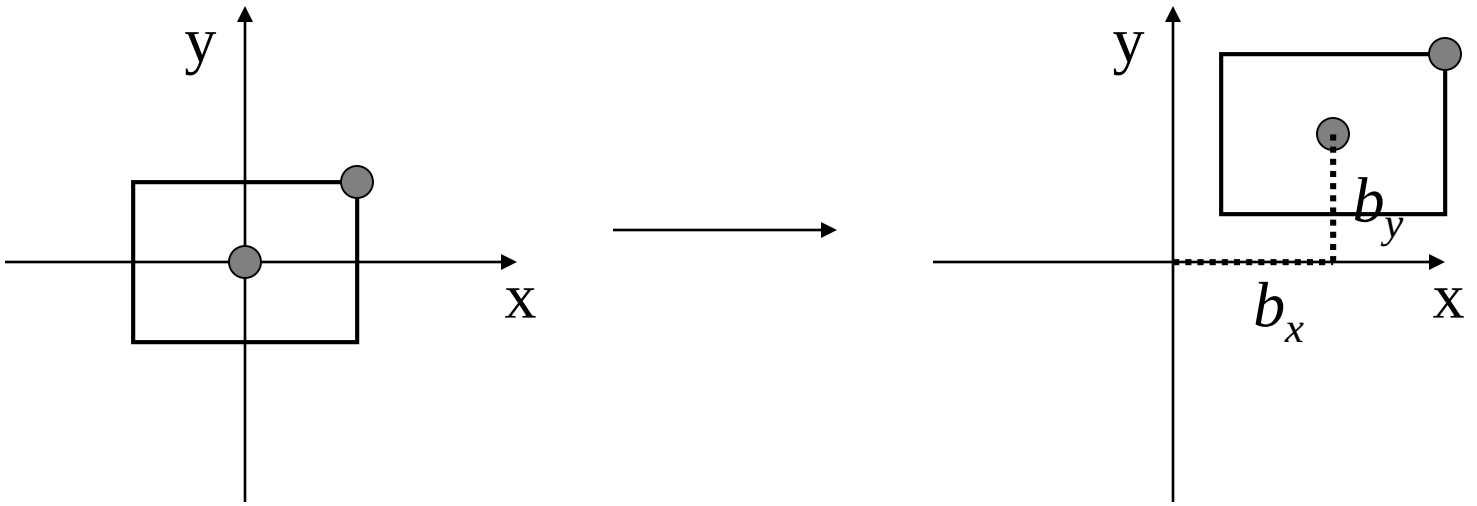
- Affine transformations are *linear*
  - Transforming all the individual points on a line gives the same set of points as transforming the endpoints and joining them
  - Interpolation is the same in either space: Find the halfway point in one space, and transform it. Will get the same result if the endpoints are transformed and then find the halfway point

Composition of Affine Transforms

- Any affine transformation can be composed as a sequence of simple transformations:
  - Translation
  - Scaling
  - Rotation
  - Shear
  - Reflection

2D Translation

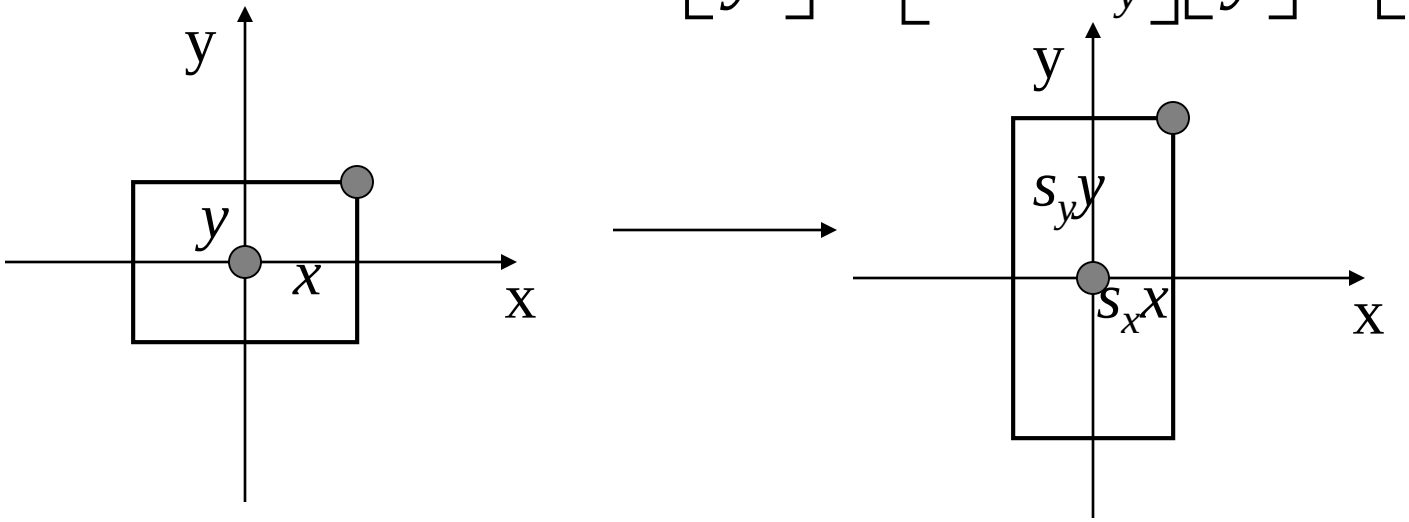
- Moves an object $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix}$



2D Scaling

- Resizes an object in each dimension

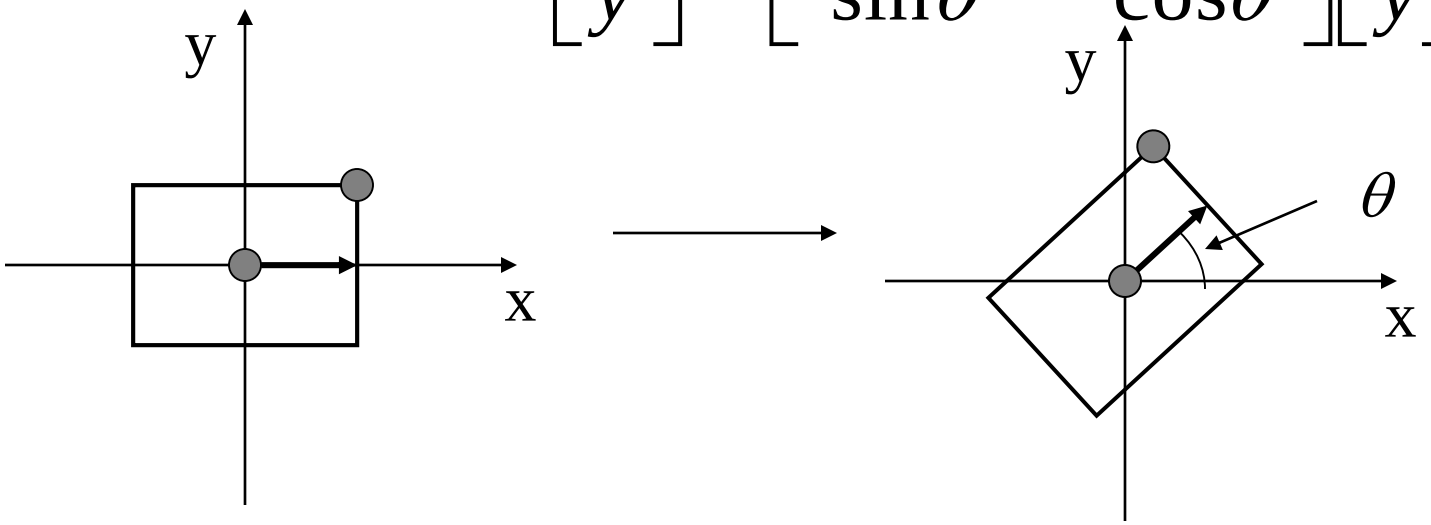
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



2D Rotation

- Rotate counter-clockwise about the origin by an angle θ

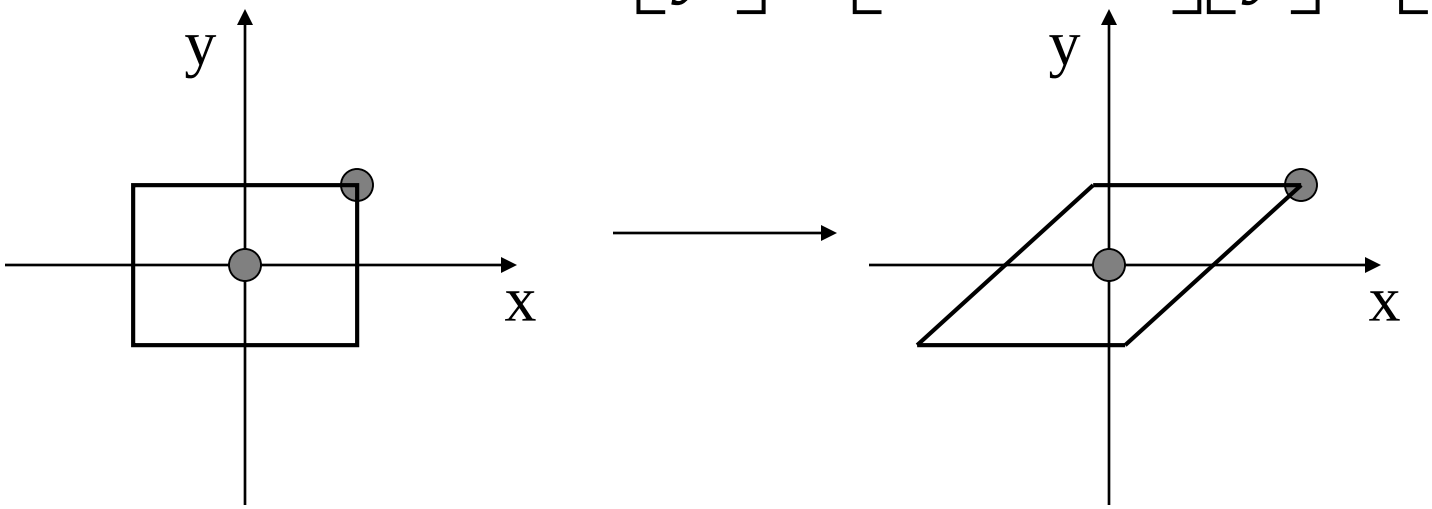
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



X-Axis Shear

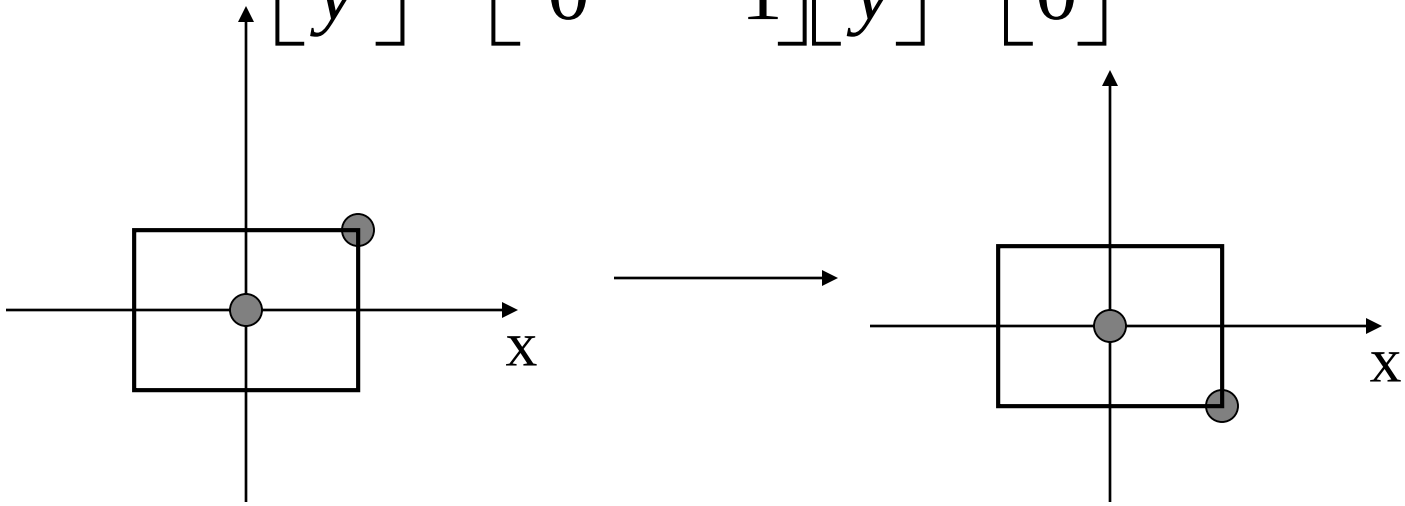
- Shear along x axis (What is the matrix for y axis shear?)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



Reflect About X Axis

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

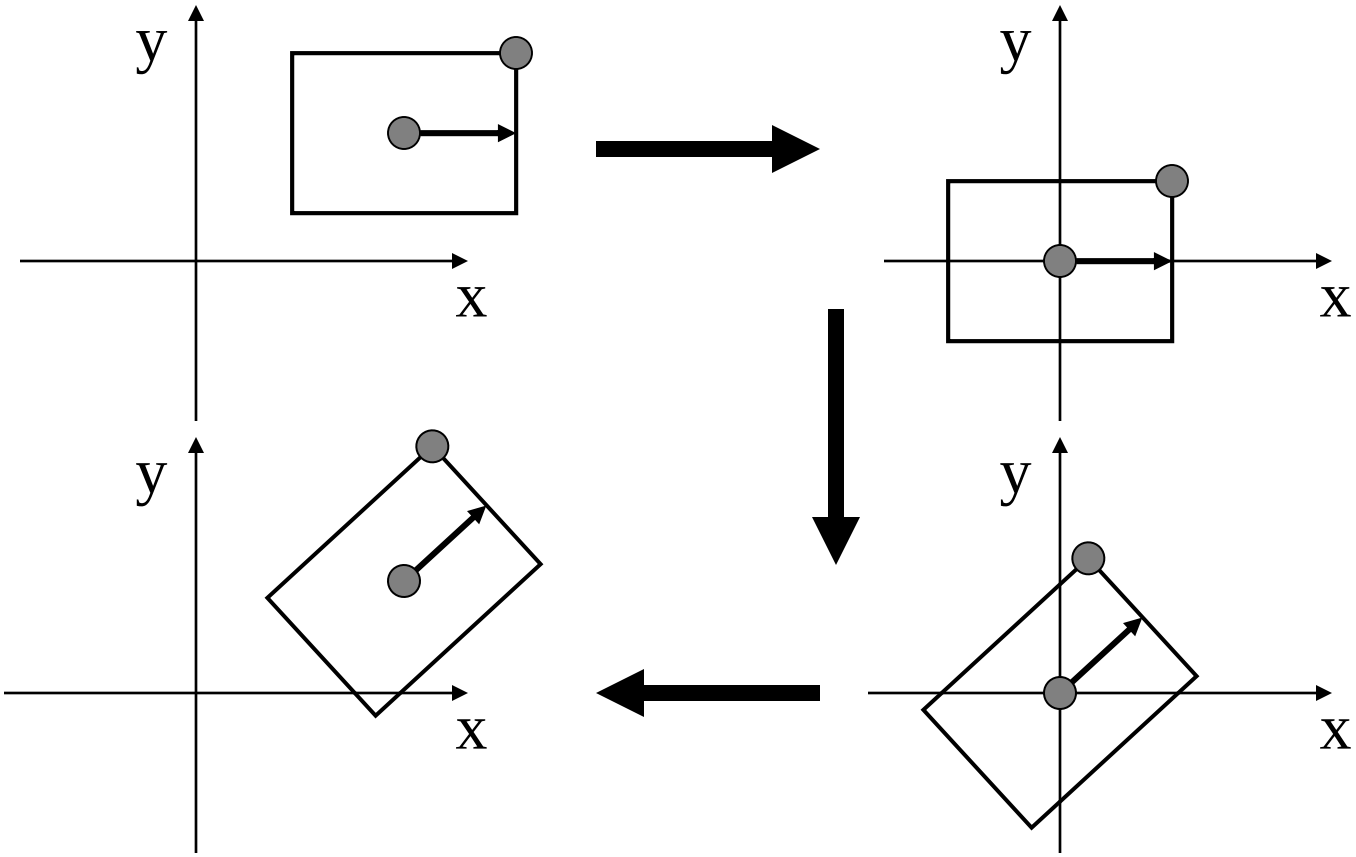


- What is the matrix for reflect about Y axis?

Rotating About An Arbitrary Point

- What happens when you apply a rotation transformation to an object that is not at the origin?
- Solution:
  - Translate the center of rotation to the origin
  - Rotate the object
  - Translate back to the original location

Rotating About An Arbitrary Point



Scaling an Object not at the Origin

- What also happens if you apply the scaling transformation to an object not at the origin?
- Based on the rotating about a point composition, what should you do to resize an object about its own center?

Back to Rotation About a Pt

- Say $\boldsymbol{R}$ is the rotation matrix to apply, and $\boldsymbol{p}$ is the point about which to rotate
- Translation to Origin: $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{p}$
- Rotation: $\boldsymbol{x}'' = \boldsymbol{R}\boldsymbol{x}' = \boldsymbol{R}(\boldsymbol{x} - \boldsymbol{p}) = \boldsymbol{R}\boldsymbol{x} - \boldsymbol{R}\boldsymbol{p}$
- Translate back: $\mathbf{x}''' = \mathbf{x}'' + \mathbf{p} = \mathbf{R}\mathbf{x} - \mathbf{R}\mathbf{p} + \mathbf{p}$
- The translation component of the composite transformation involves the rotation matrix. What a mess!

Homogeneous Coordinates

- Use three numbers to represent a point
- (x,y)=(wx,wy,w) for any constant w≠0
- Typically, (x,y) becomes (x,y,1)
- Translation can now be done with matrix multiplication!

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} & b_x \\ a_{yx} & a_{yy} & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Basic Transformations

- Translation:

$$\begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Scaling:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Transform Advantages

- Unified view of transformation as matrix multiplication
  - Easier in hardware and software
- To compose transformations, simply multiply matrices
  - Order matters: ***AB*** is generally not the same as ***BA***
- Allows for non-affine transformations:
  - Perspective projections!
  - Bends, tapers, many others

3D Transformations Watt Section 1.1

- Homogeneous coordinates: (x,y,z)=(wx,wy,wz,w)
- Transformations are now represented as 4x4 matrices
- Typical graphics packages allow for specification of translation, rotation, scaling and arbitrary matrices
  - OpenGL: glTranslate[fd], glRotate[fd], glScale[fd], glMultMatrix[fd]

3D Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D Rotation

- Rotation in 3D is about an *axis* in 3D space passing through the origin
- Using a matrix representation, any matrix with an *orthonormal* top-left 3x3 sub-matrix is a rotation
  - Rows are mutually orthogonal (0 dot product)
  - Determinant is 1
  - Implies columns are also orthogonal, and that the transpose is equal to the inverse

3D Rotation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & 0 \\ r_{yx} & r_{yy} & r_{yz} & 0 \\ r_{zx} & r_{zy} & r_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

and for example :

$$r_{xx} r_{yx} + r_{xy} r_{yy} + r_{xz} r_{yz} = 0$$

Problems with Rotation Matrices

- Specifying a rotation really only requires 3 numbers
  - Axis is a unit vector, so requires 2 numbers
  - Angle to rotate is third number
- Rotation matrix has a large amount of redundancy
  - Orthonormal constraints reduce degrees of freedom back down to 3
  - Keeping a matrix orthonormal is difficult when transformations are combined

Alternative Representations

- Specify the axis and the angle (OpenGL method)
  - Hard to compose multiple rotations
- Specify the axis, scaled by the angle
  - Only 3 numbers, but hard to compose
- Euler angles: Specify how much to rotate about X, then how much about Y, then how much about Z
  - Hard to think about, and hard to compose

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.

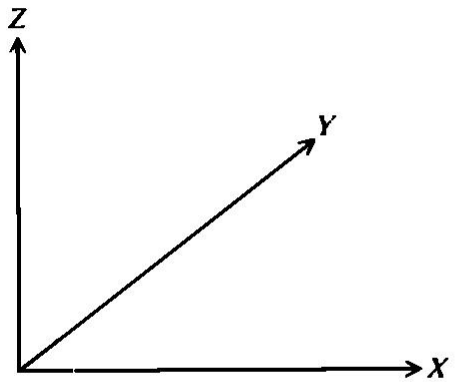
Coordinate Systems

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).

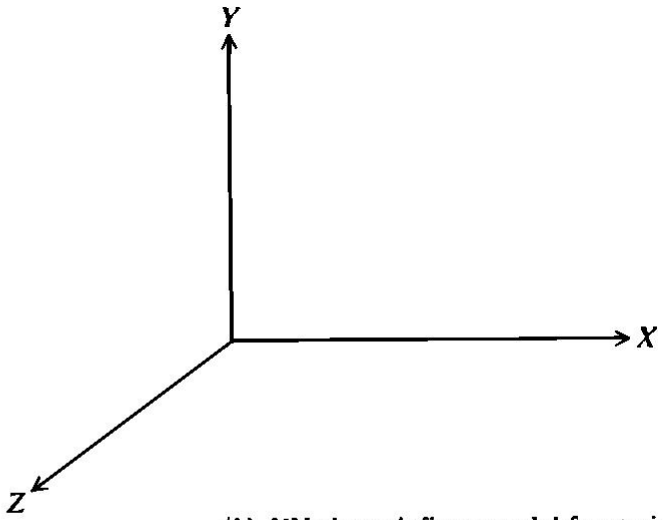
MCS : is the reference space of the model with respect to all the model geometrical data is stored.
WCS: is a convenient user-defined system that facilitates geometric construction.
SCS: is a two-dimensional device-dependent coordinate system whose origin is usually located ate the lower left corner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



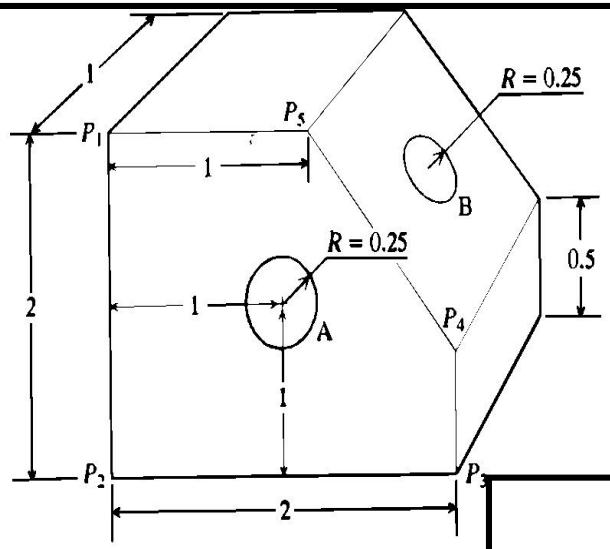
(*a*) *XY* plane defines model top view



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

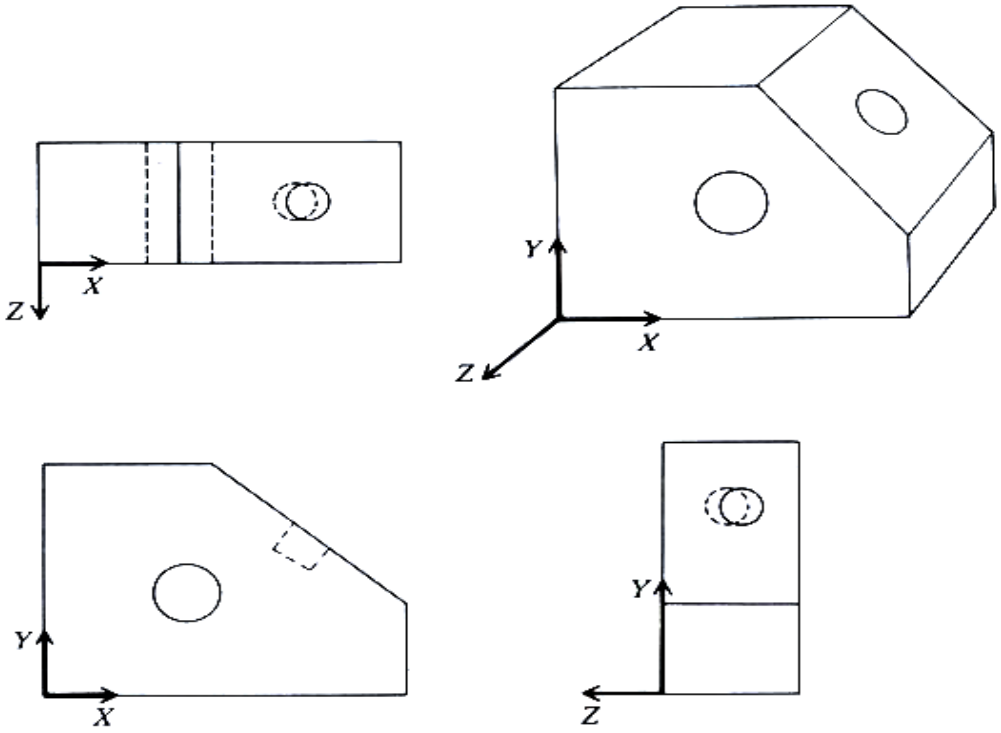
FIGURE 3-9
Possible orientations of MCS in space.

Example:



All dimensions in inches

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

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

where ${}^{M}_{W}[R]$ is the rotation matrix that defines the orientation of the WCS relative to the MCS and ${}^{M}P_{W,\,org}$ 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_2E_3.
5. Repeat step 2 but with a distance -0.5 (in the opposite direction to Z_{W2}).

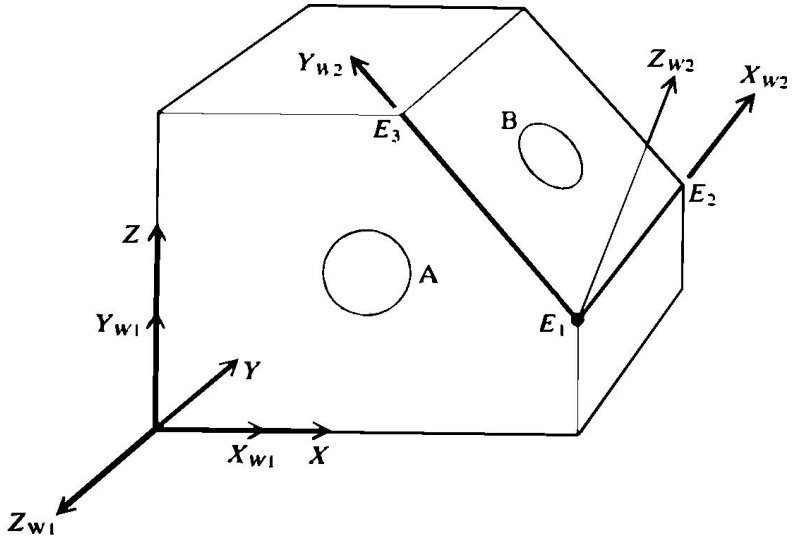


FIGURE 3-14
WCSs required to construct holes A and B.

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:

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

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.

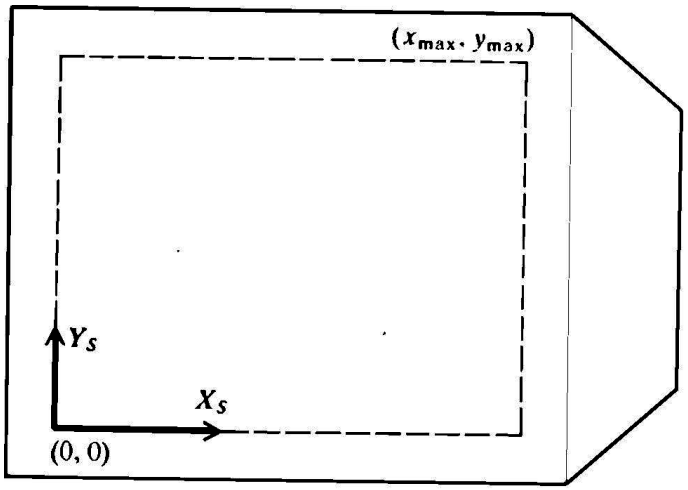


FIGURE 3-15
Typical SCS.

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.

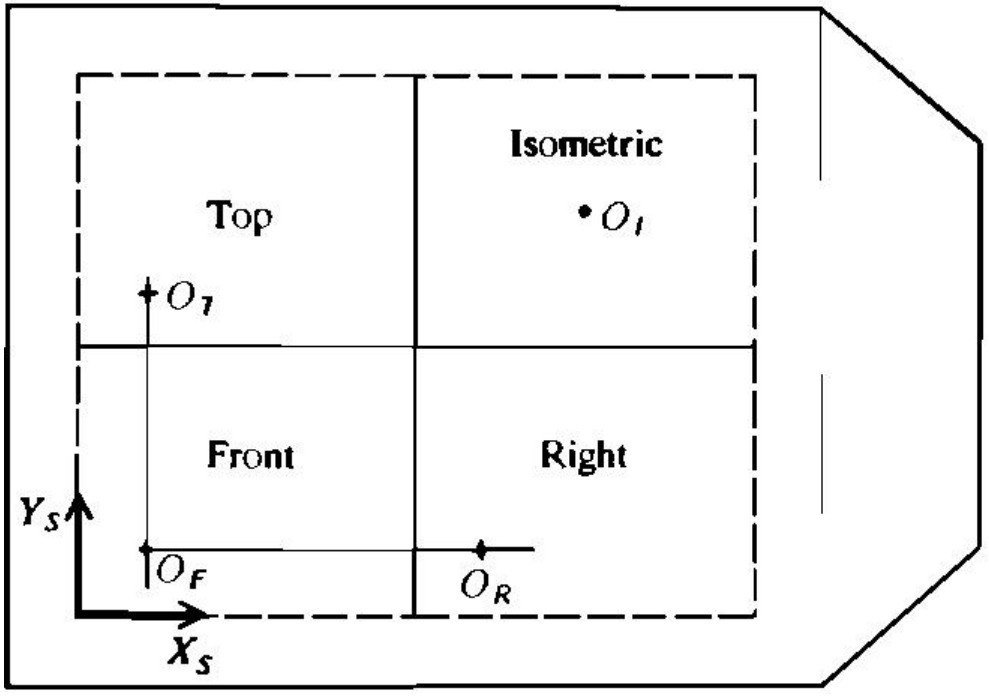


FIGURE 3-16
A typical screen layout.

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

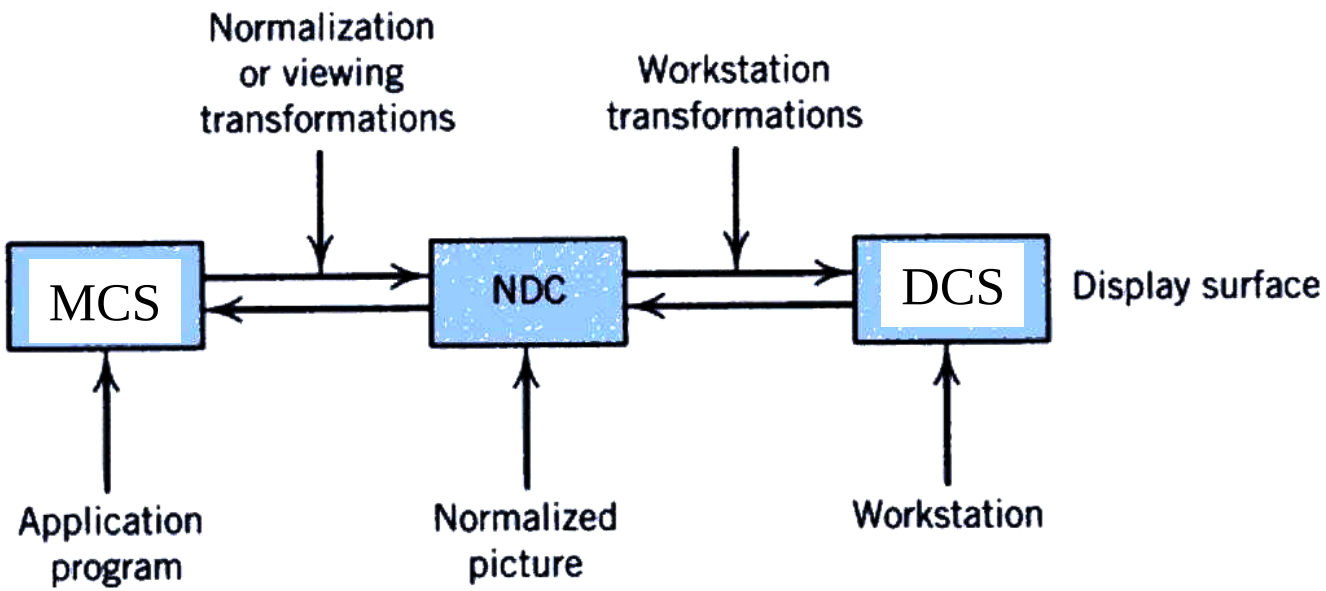
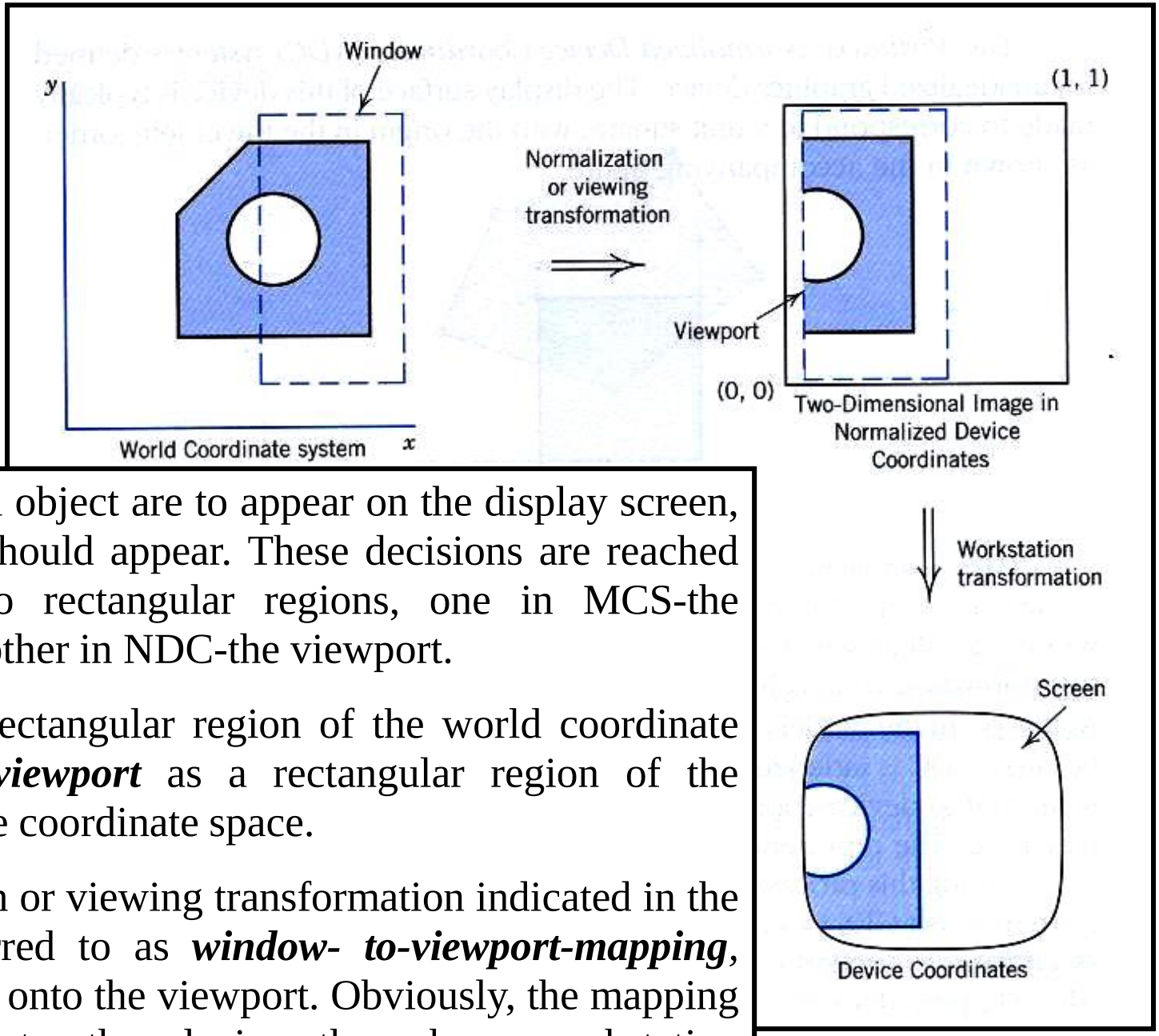


FIGURE 5.3 GKS Coordinate systems and transformations.

NDC = Normalized Device Coordinate System

Window and viewport definitions

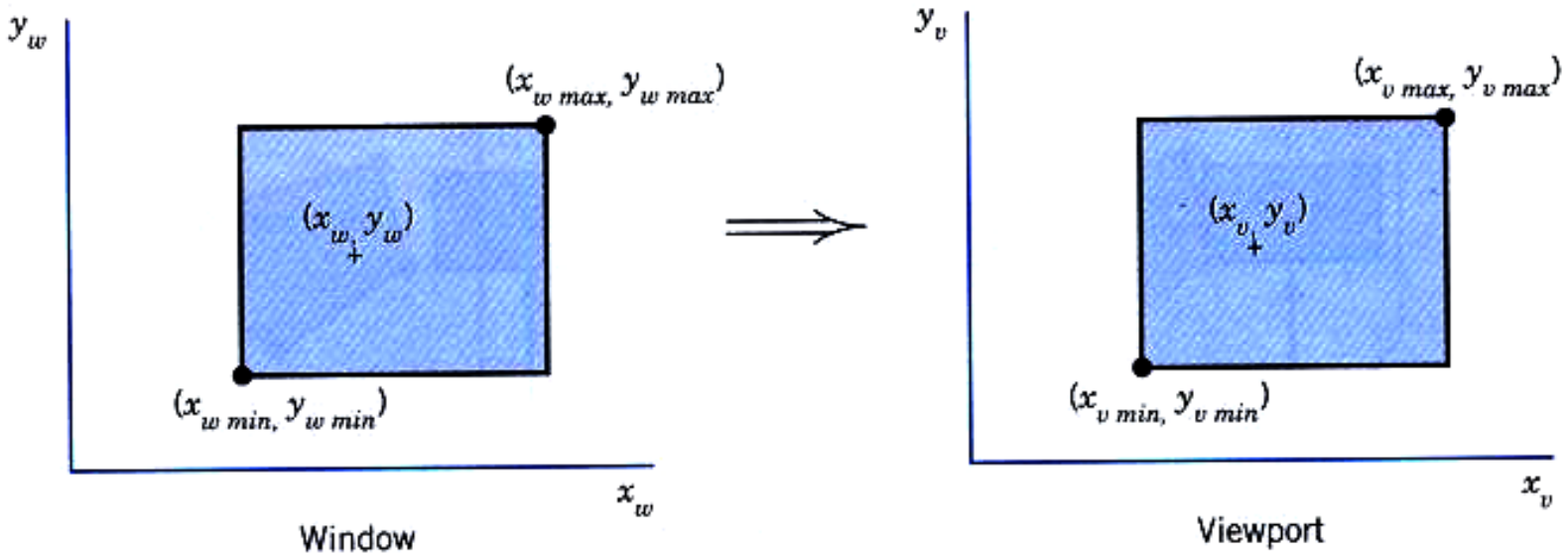


Which parts of an object are to appear on the display screen, and where they should appear. These decisions are reached by choosing two rectangular regions, one in MCS-the window-and the other in NDC-the viewport.

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:

$$\frac{x_v - x_{v\min}}{x_{v\max} - x_{v\min}} = \frac{x_w - x_{w\min}}{x_{w\max} - x_{w\min}} \tag{5.1}$$

and

$$\frac{y_v - y_{v\min}}{y_{v\max} - y_{v\min}} = \frac{y_w - y_{w\min}}{y_{w\max} - y_{w\min}} \tag{5.2}$$

So that

$$x_v = (x_w - x_{w\min}) \left(\frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}} \right) + x_{v\min} \tag{5.3}$$

$$y_v = (y_w - y_{w\min}) \left(\frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}} \right) + y_{v\min} \tag{5.4}$$

The terms

$$\left(\frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}} \right) \quad \text{and} \quad \left(\frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}} \right) \tag{5.5}$$

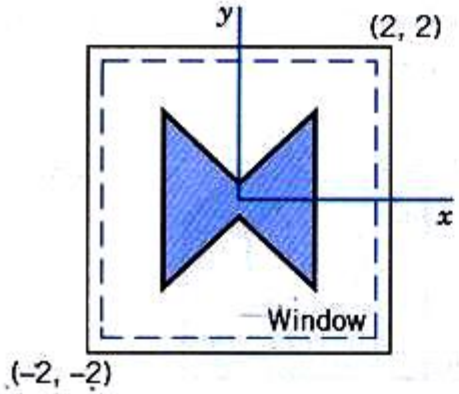
are constant for all points being mapped, and are simply scaling factors in the x and y directions, S_x and S_y. If $S_x \neq S_y$, distortions occur in the picture. The concept of *aspect ratio* refers to this situation. For the rectangular window or viewport described previously, the aspect ratio of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

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

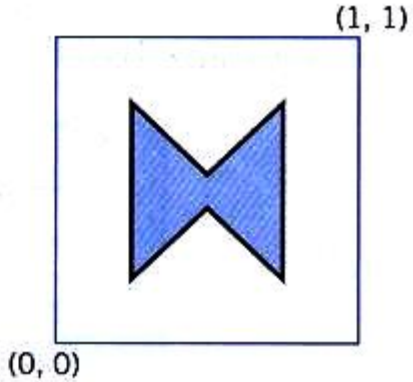
If the aspect ratios of both window and viewport are the same, then $S_x = S_y$ and there will be no distortion of the picture. For circular features special care must be taken, or circles will appear as ellipses on the display screen. Figure 5.6 shows examples of window-to-viewport mapping for different 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.

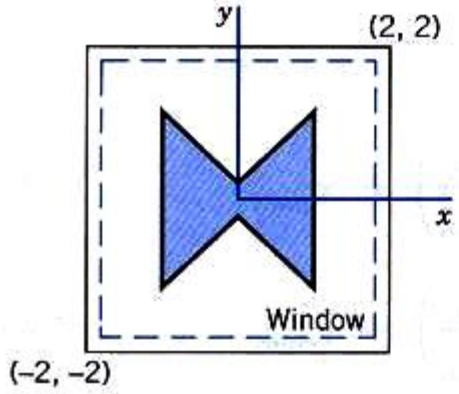
Example 5.1

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).

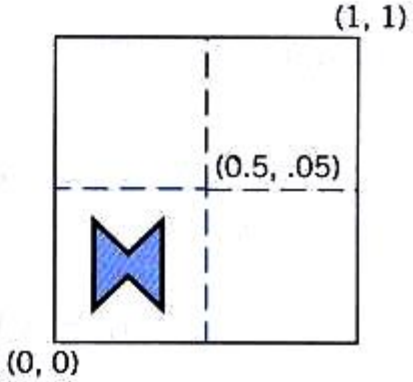


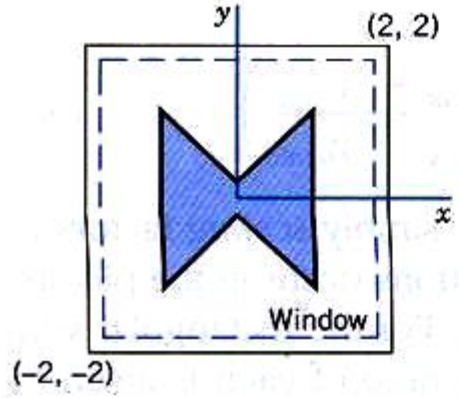
Window (–2.0, 2.0, –2.0, 2.0)
Viewport (0.0, 1.0, 0.0, 1.0)



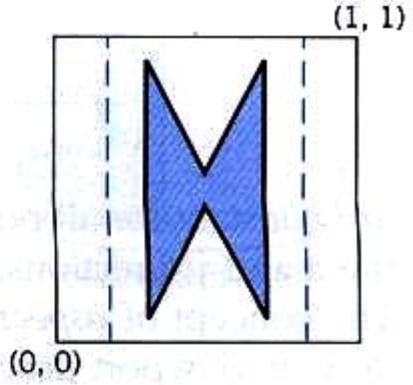


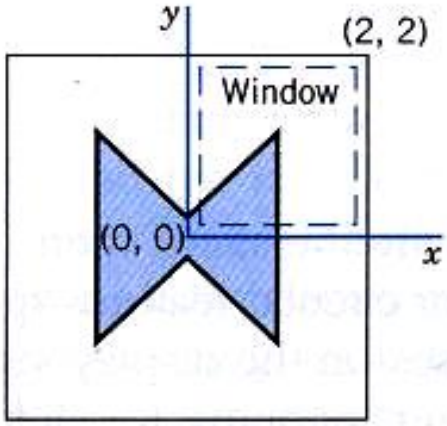
Window (–2.0, 2.0, –2.0, 2.0)
Viewport (0.0, 0.5, 0.0, 0.5)



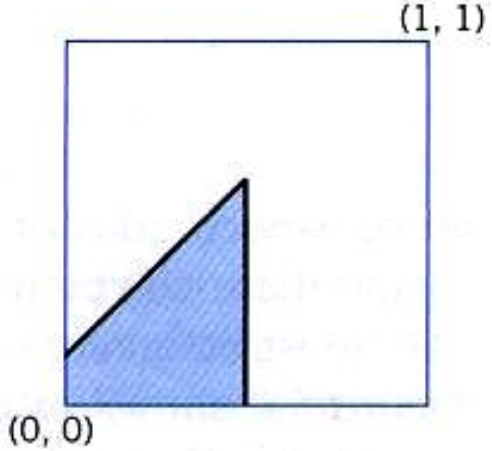


Window (–2.0, 2.0, –2.0, 2.0)
Viewport (0.25, 0.75, 0.0, 1.0)
(Distortion)





Window (0.0, 2.0, 0.0, 2.0)
Viewport (0.0, 1.0, 0.0, 1.0)
(Zooming)



Solution

The window/viewport parameters are

$$x_{w\min} = 2 \qquad x_{v\min} = \tfrac{1}{2}$$
$$x_{w\max} = 6 \qquad x_{v\max} = 1$$
$$y_{w\min} = 2 \qquad y_{v\min} = \tfrac{1}{2}$$
$$y_{w\max} = 5 \qquad y_{v\max} = 1$$

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

Computer representation of the geometry of a component using software is called a geometric model. Geometric modeling is done in three principal ways. They are:

i. Wire frame modeling
ii. Surface modeling
iii. Solid modeling

These modeling methods have distinct features and applications

WIRE FRAME MODELING

In wire frame modeling the object is represented by its edges. In the initial stages of CAD, wire frame models were in 2-D. Subsequently 3-D wire frame modeling software was introduced. The wire frame model of a box is shown in Fig. 6.2 (a). The object appears as if it is made out of thin wires. Fig. 6.2(b), 6.2(c) and 6.2(d) show three objects which can have the same wire frame model of the box. Thus in the case of complex parts wire frame models can be confusing. Some clarity can be obtained through hidden line elimination.

WIRE FRAME MODELING

Though this type of modeling may not provide unambiguous understanding of the object, this has been the method traditionally used in the 2-D representation of the object, where orthographic views like plan, elevation, end view etc are used to describe the object graphically.

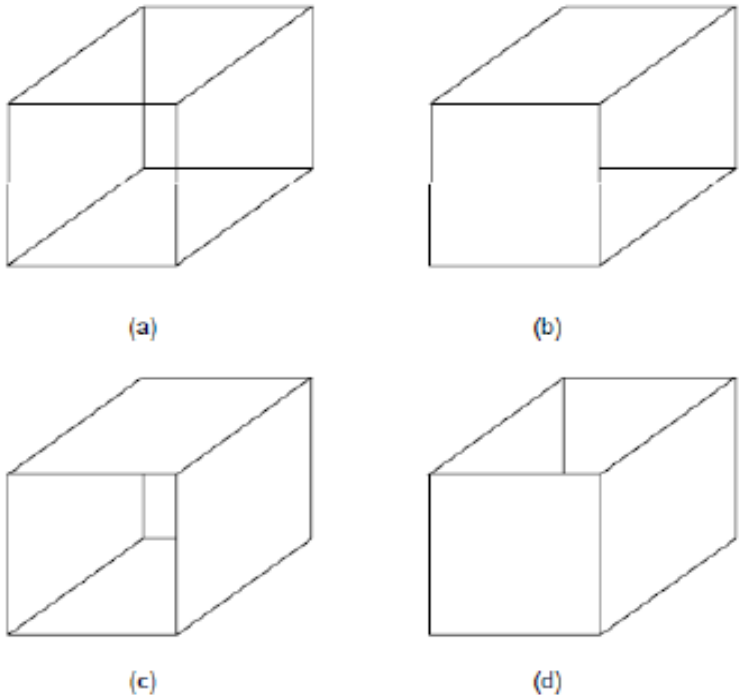


Fig. 6.2 Ambiguity in Wire Frame Modeling

WIRE FRAME MODELING

A comparison between 2-D and 3-D models is given below:

2 - D Models	*3-D Wire Frame Models*
Ends (vertices) of lines are represented by their X and Y coordinates	Ends of lines are represented by their X, Y and Z coordinates.
Curved edges are represented by circles, ellipses, splines etc. Additional views and sectional views are necessary to represent a complex object with clarity.	Curved surfaces are represented by suitably spaced generators. Hidden line or hidden surface elimination is a must to interpret complex components correctly.
3-D image reconstruction is tedious.	2-D views as well as various pictorial views can be generated easily.
Uses only one global coordinate system	May require the use of several user coordinate systems to create features on different faces of the component.

SURFACE MODELING

In this approach, a component is represented by its surfaces which in turn are represented by their vertices and edges. For example, eight surfaces are put together to create a box, as shown in Fig. 6.3. Surface modeling has been very popular in aerospace product design and automotive design dies and moulds.

Surface modeling has been particularly useful in the development of manufacturing codes for automobile panels and the complex doubly curved shapes of aerospace structures and

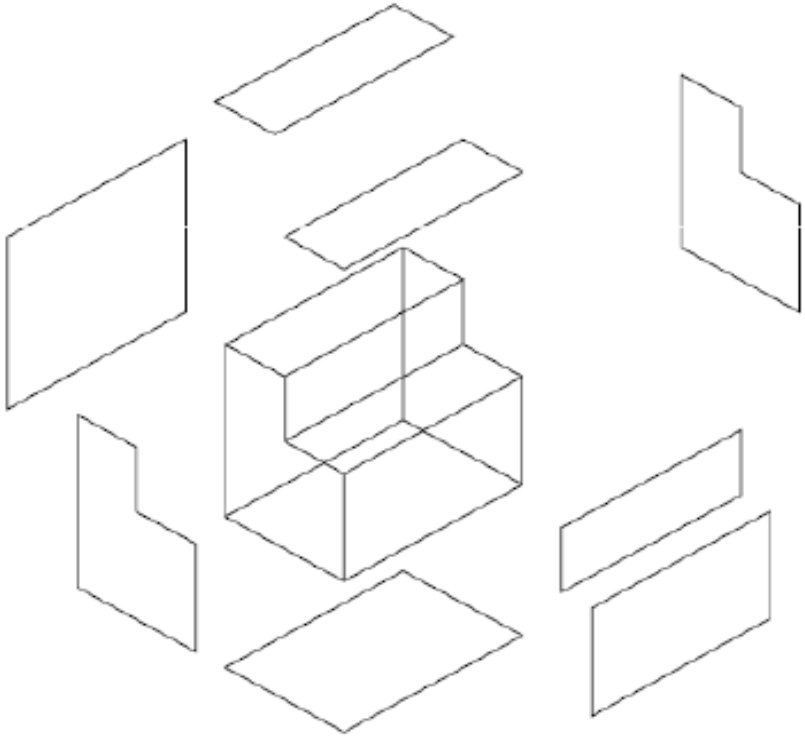


Fig. 6.3 Surface Representation

SURFACE MODELING

Apart from standard surface types available for surface modeling (box, pyramid, wedge, dome, sphere, cone, torus, dish and mesh) techniques are available for interactive modeling and editing of curved surface geometry. Surfaces can be created through an assembly of polygonal meshes or using advanced curve and surface modeling techniques like B-splines or NURBS (Non-Uniform Rational B-splines). Standard primitives used in a typical surface modeling software are shown in Fig. 6.4. Tabulated surfaces, ruled surfaces and edge surfaces and revolved are simple ways in which curved geometry could be created and edited.

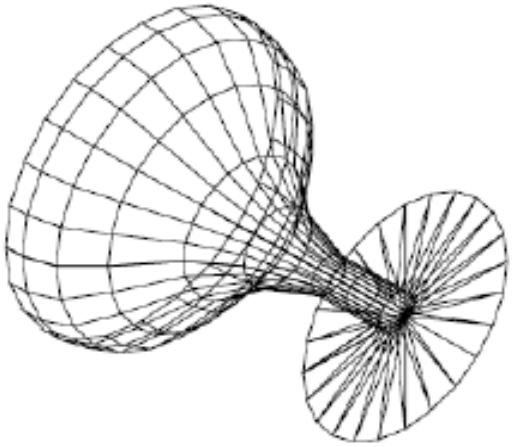


Fig. 6.25 A Typical Revolved Surface Model

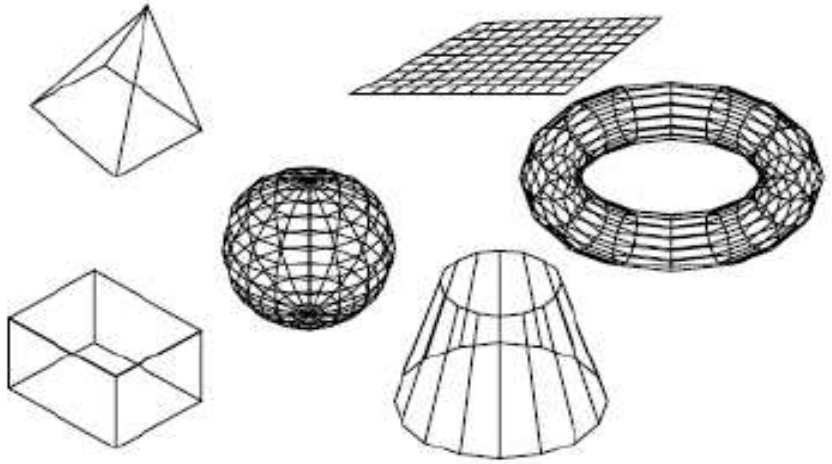


Fig. 6.4 Typical Approaches in Surface Modeling

SURFACE MODELING

Figure 6.25 is typical of several components, one comes across in engineering. The surface of this component can be produced by revolving a profile about an axis of rotation.

A surface model is defined in terms of points, lines and faces. This type of modeling is superior to wire frame modeling discussed earlier in this chapter. A major advantage of surface modeling is its ability to differentiate flat and curved surfaces. In graphics, this helps to create shaded image of the product. In manufacture, surface model helps to generate the NC tool path for complex shaped components that are encountered in aerospace structures, dies and moulds and automobile body panels.

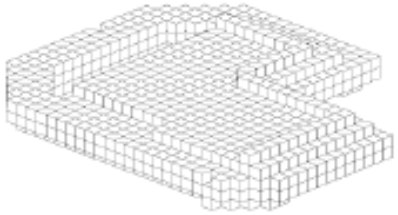
A surface can be created in several ways:

i. Creating a plane surface by the linear sweep of a line or series of lines.
ii. Revolving a straight line about an axis. Cylindrical, conical surfaces etc. can be generated by this technique.
iii. Revolving a curve about an axis.
iv. Combination of plane surfaces.
v. Analytic surfaces: Planes, cylinders, cones, ellipsoid, parabolic hyperboloid etc can be defined by mathematical equations in terms of X, Y and Z co-ordinates.
vi. Sculptured surfaces: These are also called free form surfaces. These are created by spline curves in one or both directions in a 3-D space. These surfaces are used in the manufacture of car body panels, aircraft structures, mixed flow impellers, telephone instruments, plastic containers and several consumer and engineering products.

SOLID MODELING

The representation of solid models uses the fundamental idea that a physical object divides the 3-D Euclidean space into two regions, one exterior and one interior, separated by the boundary of the solid. Solid models are:

Spatial Enumeration

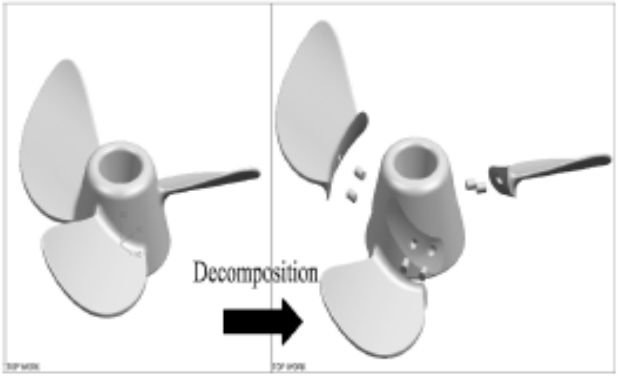


- bounded
- homogeneously three dimensional
- finite

There are six common representations in solid modeling.

i. Spatial Enumeration: In this simplest form of 3D volumetric raster model, a section of 3D space is described by a matrix of evenly spaced cubic volume elements called voxels.

ii. Cell Decomposition: This is a hierarchical adaptation of spatial enumeration. 3D space is sub-divided into cells. Cells could be of different sizes. These simple cells are glued together to describe a solid object.

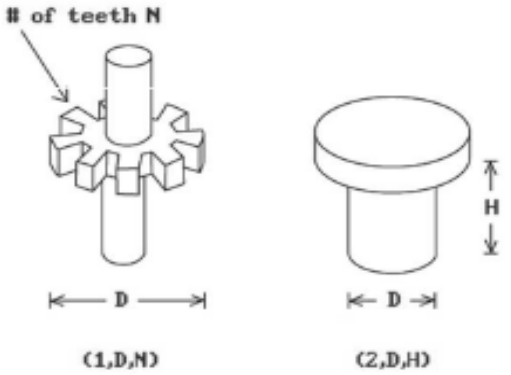


SOLID MODELING

iii. Boundary Representation: The solid is represented by its boundary which consists of a set of faces, a set of edges and a set of vertices as well as their topological relations.

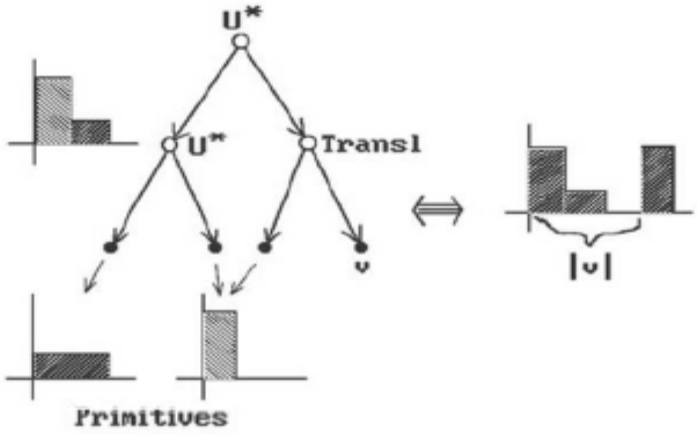
iv. Sweep Methods: In this technique a planar shape is moved along a curve. Translational sweep can be used to create prismatic objects and rotational sweep could be used for axisymmetric components.

v. Primitive Instancing: This modeling scheme provides a set of possible object shapes which are described by a set of parameters. Instances of object shape can be created by varying these parameters.



SOLID MODELING

vi. Constructive Solid Geometry (CSG): Primitive instances are combined using Boolean set operations to create complex objects.



In most of the modeling packages, the approach used for modeling uses any one of the following three techniques:

i. Constructive solid geometry (CSG or C-Rep)
ii. Boundary representation (B-Rep)
iii. Hybrid method which is a combination of B-Rep and CSG.

Constructive Solid Geometry (CSG)

In a CSG model, physical objects are created by combining basic elementary shapes known as primitives like blocks, cylinders, cones, pyramids and spheres. The Boolean operations like union (∪), difference (–) and intersection ∩ are used to carry out this task. For example, let us assume that we are using two primitives, a block and a cylinder which are located in space as shown in Fig. 6.5.

A "union" operation (A ∪ B) will combine the two to convert them into a new solid.(Fig. 6.5 (c)) The difference operation (A – B) will create a block with a hole (Fig. 6.5. (D)). An intersection operation (A ∩ B) will yield the portion common to the two primitives. (Fig. 6.5(E)).

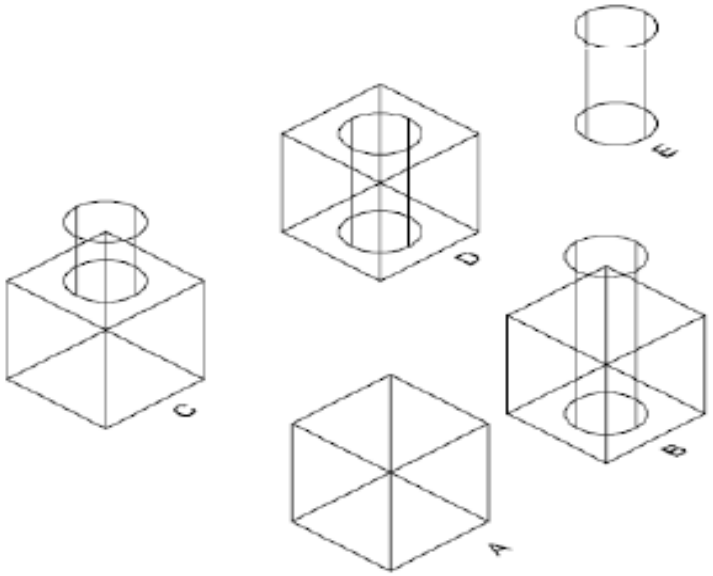


Fig.6.5 CSG Operation

Boundary Representation

Boundary representation is built on the concept that a physical object is enclosed by a set of faces which themselves are closed and orientable surfaces. Fig. 6.6 shows a B-rep model of an object. In this model, face is bounded by edges and each edge is bounded by vertices. The entities which constitute a B-rep model are:

Geometric entities	***Topological entities***
Point	Vertex
Curve, line	Edge
Surface	Face

A solid model is a 3-D representation of an object. It is an accurate geometric description which includes not only the external surfaces of part, but also the part's internal structure.
A solid model allows the designer to determine information like the object's mass properties, interferences, and internal cross sections.

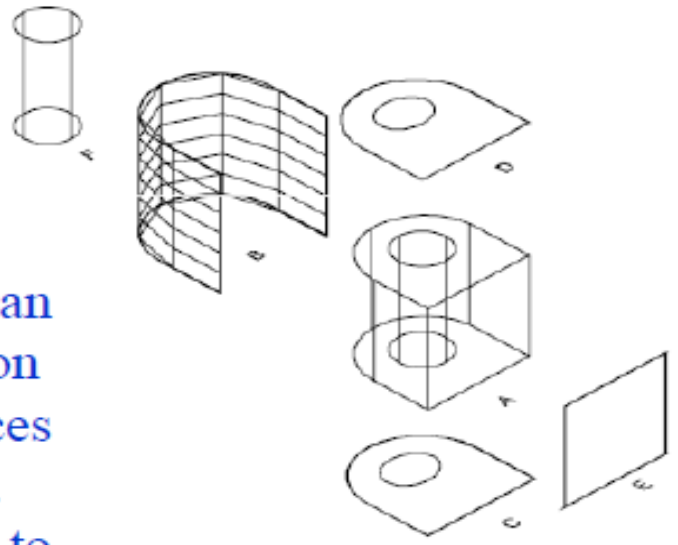


Fig. 6.6 B-Rep Model

SOLID MODELING

Solid models differ from wire frame and surface models in the kind of geometric information they provide. Wire frame models only show the edge geometry of an object. They say nothing about what is inside an object. Surface models provide surface information, but they too lack information about an object's internal structure. Solid models provide complete geometric descriptions of objects.

Engineers use solid models in different ways at different stages of the design process.They can modify a design as they develop it. Since computer-based solid models are a lot easier to change and manipulate than the physical mock-ups or prototypes, more design iterations and modifications can be easily carried out as a part of the design process. Using solid modeling techniques a design engineer can modify a design several times while optimizing geometry. This means that designers can produce more finished designs in less time than by using traditional design methods or 2-D CAD drafting tools.

Solid models can be used for quick and reliable design analysis. Solid models apart from geometric information provide important data such as volume, mass, mass properties and centre of gravity. The designer can also export models created to other applications for finite element analysis (FEA), rapid prototyping and other special engineering applications.

SALIENT FEATURES OF SOLID MODELING

FEATURE-BASED DESIGN

The most fundamental aspect in creating a solid model is the concept of feature-based design. In typical 2-D CAD applications, a designer draws a part by adding basic geometric elements such as lines, arcs, circles and splines. Then dimensions are added. In solid modeling a 3-D design is created by starting a base feature and then adding other features, one at a time, until the accurate and complete representation of the part's geometry is achieved.

Because features have the ability to intelligently reference other features, the changes made will navigate through design, updating the 3-D model in all affected areas. Figure 6.7 shows a ribbed structure. It consists of feature like ribs and holes.

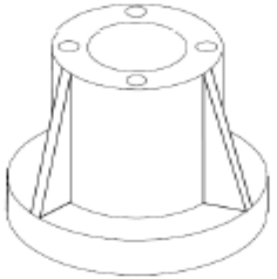


Fig. 6.7 A Ribbed Structure

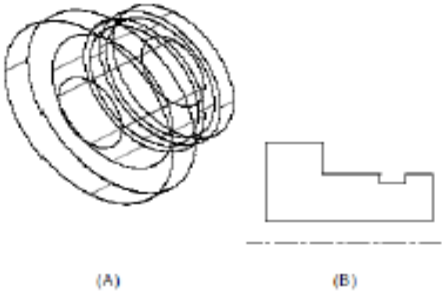


Fig. 6.8 Flanged Part

Similarly, if a flanged part shown in Fig. 6.8 (A) is to be created, the one approach is to sketch the cross section as shown in Fig. 6.8 (B) and then revolve through 360°.

SALIENT FEATURES OF SOLID MODELING

Features/available in typical solid modeling software are:

Extrude	Revolve	Thin
Blend	Slot	Cut
Protrusion	Shaft	Round
Hole	Flange	Rib
Chamfer	Push	Dome
Draft	Ear	Shell
Offset	Lip	
Pipe	Sweep	

Fig. 6.10 Exploded View of an Assembly

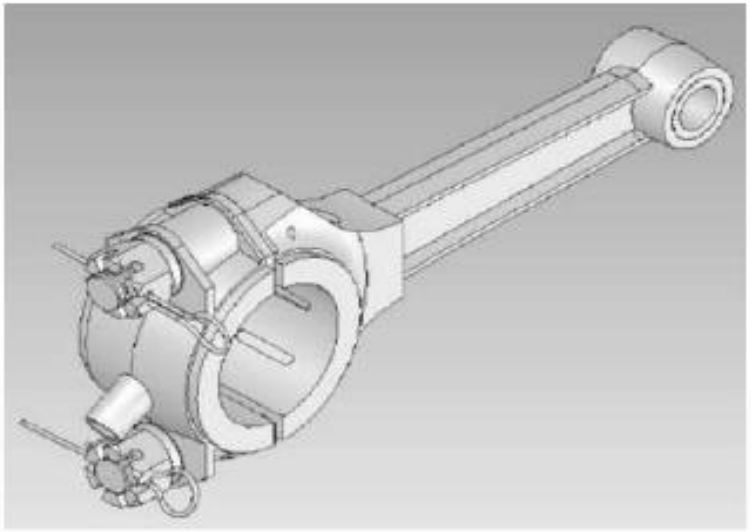


Fig. 6.9 Part with Complex Geometry

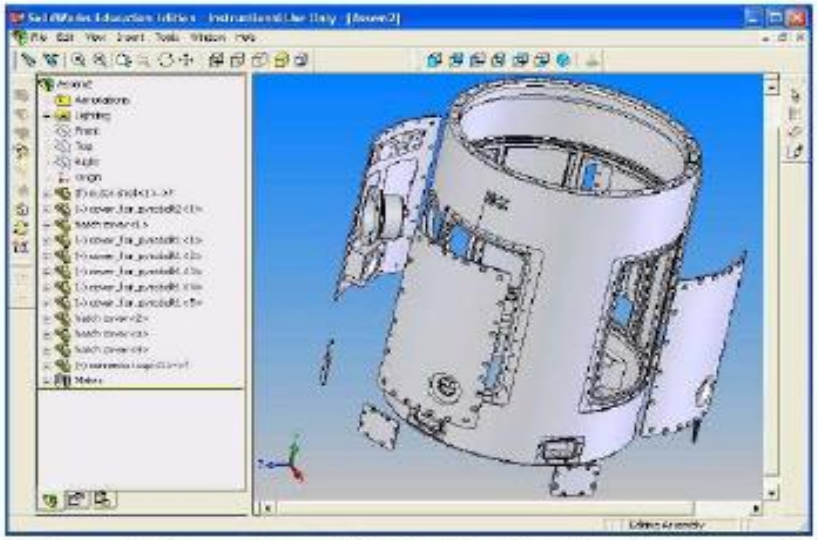


Fig. 6.13 Assembly Made up of Several Parts (Top): Exploded View of the Assembly (Bottom)

The Design Process

The process of designing something is characterized as an interactive procedure, which consists of six identifiable steps or phases:

1.Recognition of need.
2.Definition of problem.
3.Synthesis.
4.Analysis and optimization.
5.Evaluation.
6.Presentation.

Application of Computers in Design

The design related tasks performed by CAD system are:

1.Geometric modeling.
2.Engineering analysis.
3.Design review and evaluation.
4.Automated drafting.

Engineering Analysis

- **Mass properties analysis**, which involves the computation of such features of a solid object as its volume, surface area, weight, and center of gravity. It is especially applicable in mechanical design.
- **Interference checking**
- **Tolerance analysis**

After a particular design alternative has been developed, some form of engineering analysis often must be performed as part of the design process. The analysis may take the form of stress-strain calculations, heat transfer analysis, or dynamic simulation. Computations are often complex and time consuming, and before the advent of the digital computer, these analyses were usually greatly simplified or even omitted in the design procedure. The availability of software for engineering analysis on a CAD system greatly increases the designer's ability and willingness to perform a more thorough analysis of a proposed design. The term **computer aided engineering (CAE)**- is often used for engineering analyses performed by computer. Examples of engineering analysis software in common use on CAD systems include:

Engineering Analysis

•**Kinematic and dynamic analysis**. Kinematic analysis involves the study of the operation of mechanical linkages to analyze their motions. A typical kinematic analysis consists of specifying the motion of one or more driving members of the subject linkage, and the resulting motions of the other links are determined by the analysis package. Dynamic analysis extends kinematic analysis by including the effects of the mass of each linkage member and the resulting acceleration forces as well as any externally applied forces.

•**Finite element analysis**. Software for finite element analysis (FEA), also known as finite element modeling (FEM), is available for use on CAD systems to aid in stress strain, heat transfer, fluid flow, and other engineering computations. Finite element analysis is a numerical analysis technique for determining approximate solutions to physical problems described by differential equations that are very difficult or impossible to solve. In FEA, the physical object is modeled by an assemblage of discrete interconnected nodes (finite elements), and the variable of interest (e.g., stress, strain, temperature) in each node can be described

Engineering Analysis

by relatively simple mathematical equations. By solving the equations for each node, the distribution of values of the variable throughout the physical object is determined.

Automated Drafting

The fourth area where CAD is useful (step 6 in the design process) is presentation and documentation. CAD systems can be used as automated drafting machines to prepare highly accurate engineering drawings quickly.

Design Evaluation and Review

Design evaluation and review procedures can be augmented by CAD. Some of the CAD features that are helpful in evaluating and reviewing a proposed design include:

- **Automatic dimensioning** routines that determine precise distance measures between surfaces on the geometric model identified by the user.
- **Error checking.** This term refers to CAD algorithms that are used to review the accuracy and consistency of dimensions and tolerances and to assess whether the proper design documentation format has been followed.

Design Review

To set up the design review function the project administrator defines each design project, assign the members to the team, describe the actual process and define the workflow. For example, in the case of an electric motor design project the tasks to be included may be conceptual design, stress and field analysis, detailed drawings and a description of manufacturing and assembly process. One task, for example, the design analysis will be carried out by a group of three and approved by one. The design may have to be accepted by all the members of the group. The design review system gives considerable flexibility for the persons in charge of the project.

Design Evaluation

Assembly design software is a powerful tool for design evaluation. Since parts are designed in solid models the design can carry out checks for interference between mating parts and subassemblies. It is also possible to animate kinematic assemblies to evaluate how mechanisms work. Since variational geometry is used, the designer can vary the dimensions of the links to study the impact of design changes and articulate the mechanism in real time.

Assembly level mass and inertial properties can also be calculated. The rendering capabilities available in the modeling package enable true to life presentation of the assembly to management, marketing, manufacturing, purchase and other downstream users for review. Their feedback will be useful to avoid costly design changes at the end of product development cycle.

Automated Drafting.

The drafting process creates production drawings. When solid model based design is followed, the production drawings can be generated from the solid model itself. Some drafting systems allow either a single user or dual user. In the single user mode, the entire design/ drafting job is done by a single designer by creating a design with solid or assembly modeler and documenting that model using the drafting module. Bi-directional associativity helps to make changes to the solid model by incorporating a change in the detail drawing. The dual user approach is suitable for organizations where design and drafting tasks are carried out by separate persons. This mode of work enables the draftsman to begin his work even before the designer completes his design. Concurrent associativity allows the drawing to be updated while maintaining design change control with the design engineer.

Typical features that one expects in a drafting package are listed below:

Automated Drafting.

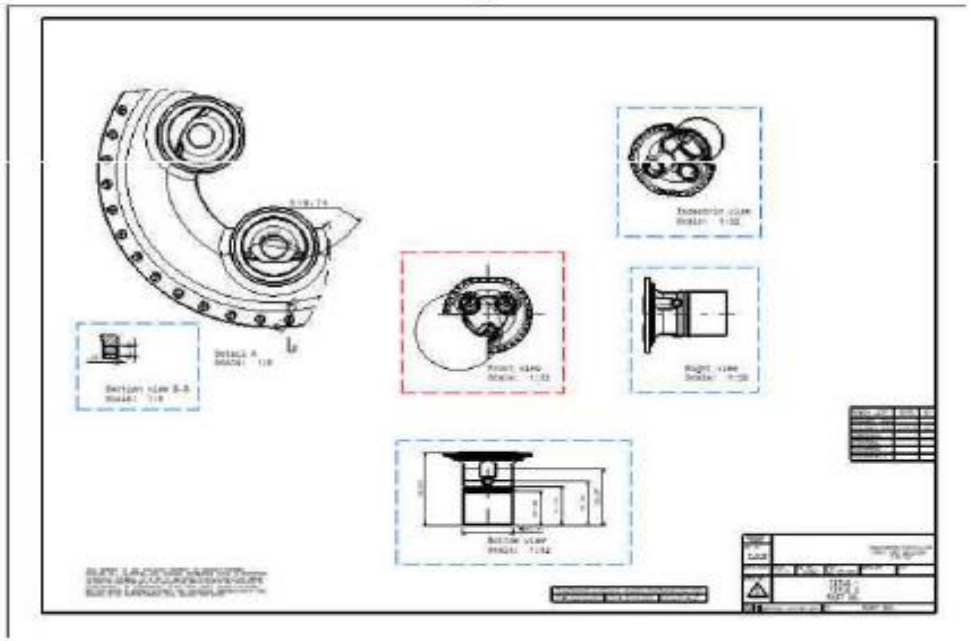
Typical features that one expects in a drafting package are listed below:

i. Drawing utilities: This includes selection of units, screen limits, scale, snap, grid, layers etc.
ii. Entity drawing: Several standard entities like line, circle, arc, polyline, polygon, ellipse etc are available to create the model required.
iii. Edit commands: A number of commands are available to modify or copy or replicate the entities or groups of entities in a model.
iv. Standard parts: Facilities are available to create symbols, shapes, and other standard parts. Frequently used parts can be stored as blocks which can be inserted into a drawing as and when required.
v. Display: The model can be enlarged, reduced in size, or moved across the screen, using display commands.
vi. Cross hatching: Sectional plans can be indicated through cross hatching.
vii. Dimensioning: Parts can be dimensioned using a number of standard dimensioning systems.
viii. Plotting: Facilities to get hard copies of drawings using a pen or electrostatic plotter and printer will be available.
ix. Configuration: A software has to be configured to a given hardware environment.
x. Customization: Drafting productivity can be enhanced through customizing software package. in

Automated Drafting.

xi. Drawing interchange: It may be necessary to import or export drawing files created in one software package to another. Facilities should be available to carry out this task. This is usually carried out by a post processor to convert the model into a neutral file like STEP and pre-processor to read the STEP file and convert the data into the model.

This is what you can create

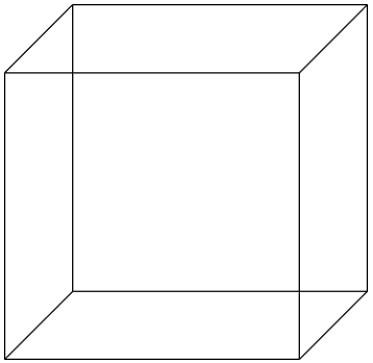


Cad-Cam. P.Radhakrishnan; Pg: 170

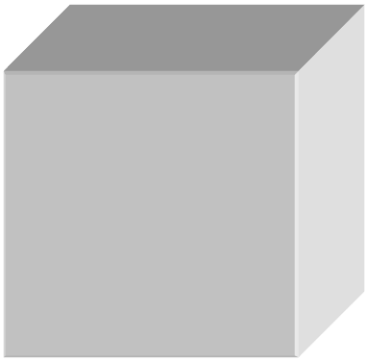
Wire-frame Modeling

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

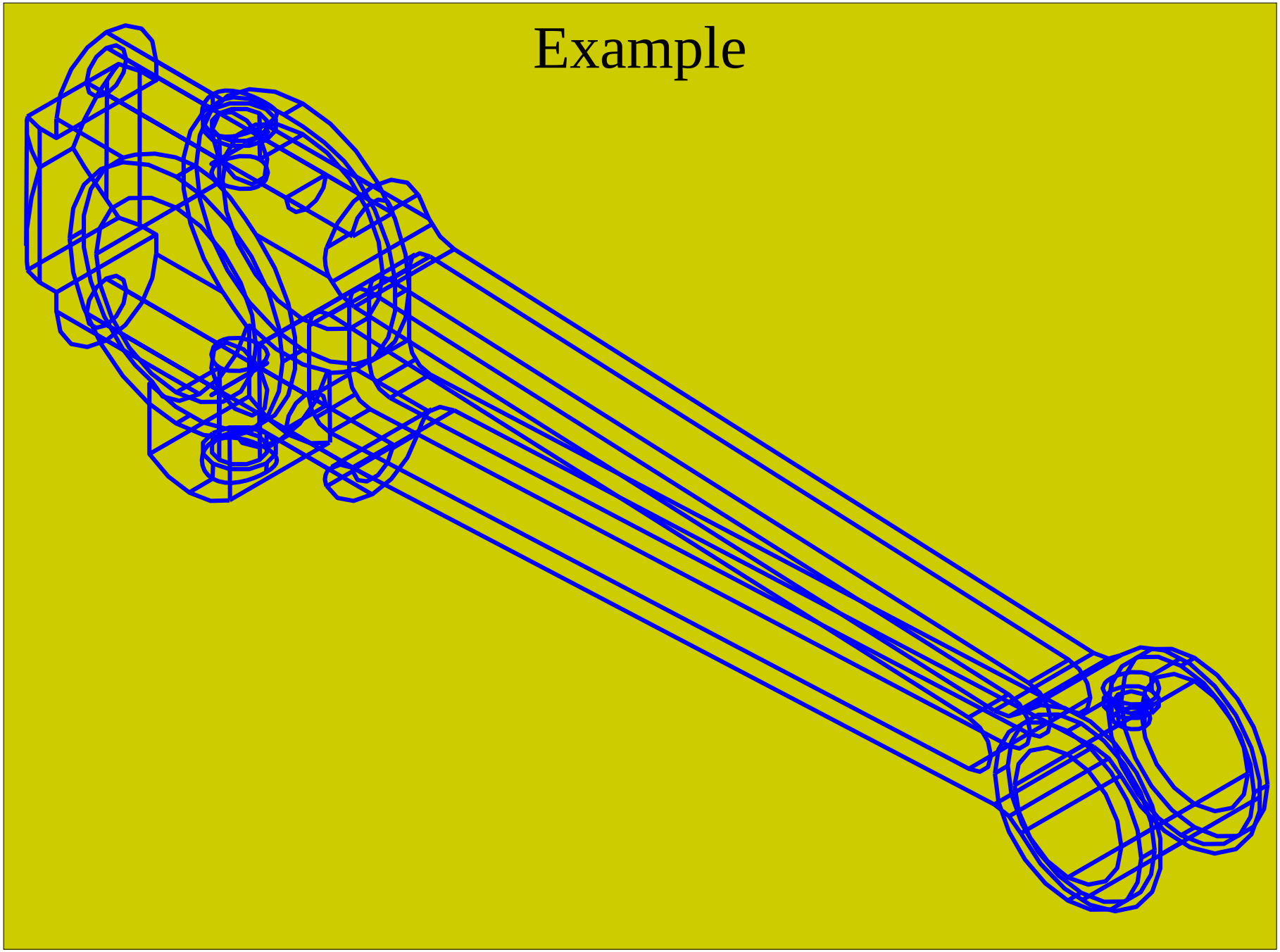


Wire-frame model



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

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

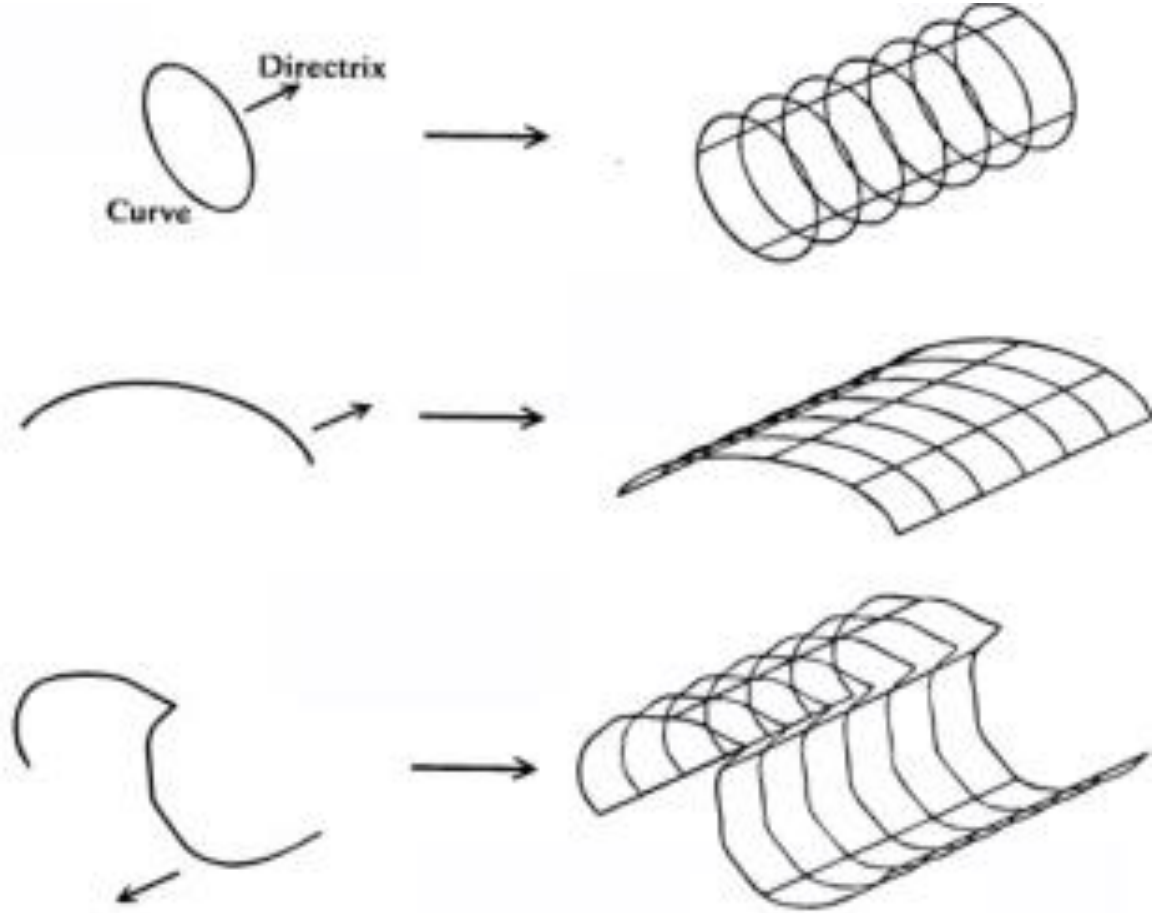


FIGURE 6-7
Tabulated cylinder.

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.

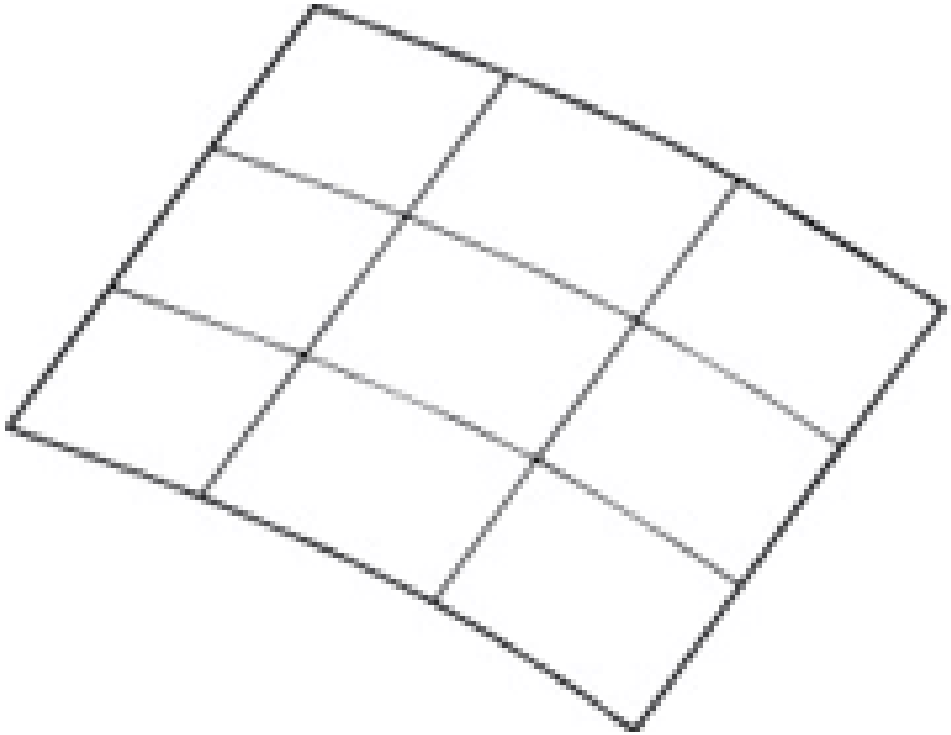
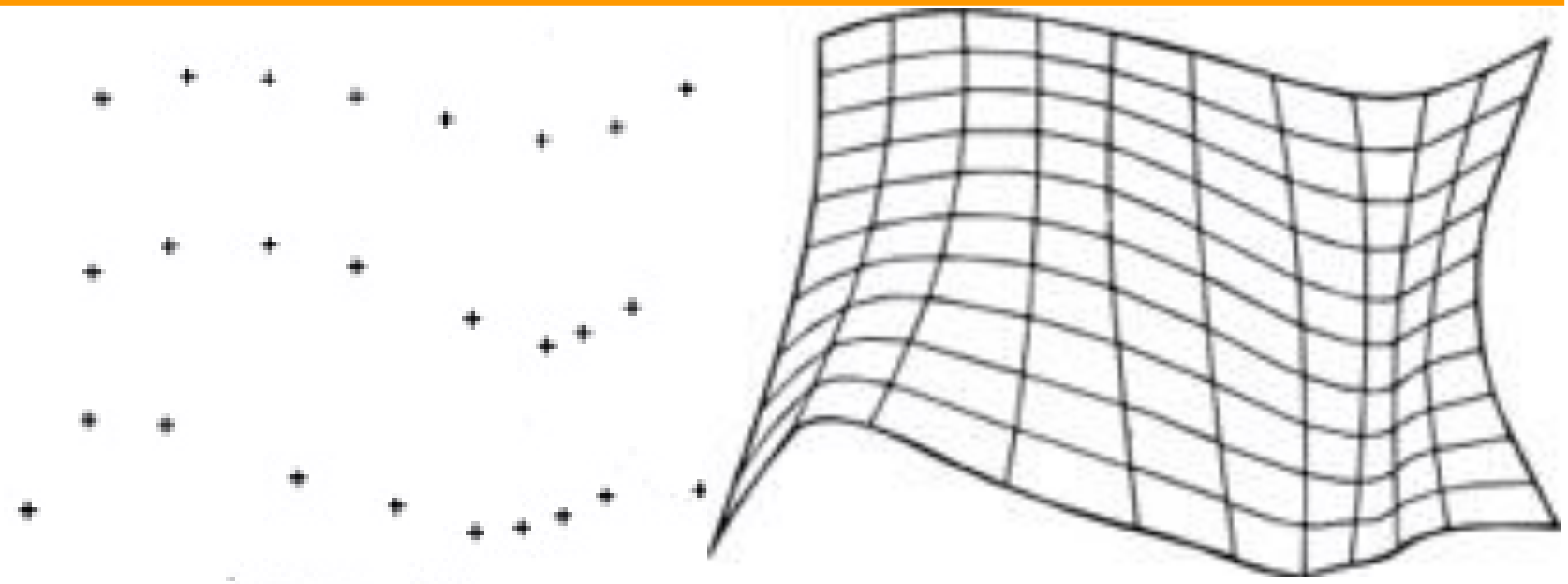


FIGURE 6-8
Bezier surface.

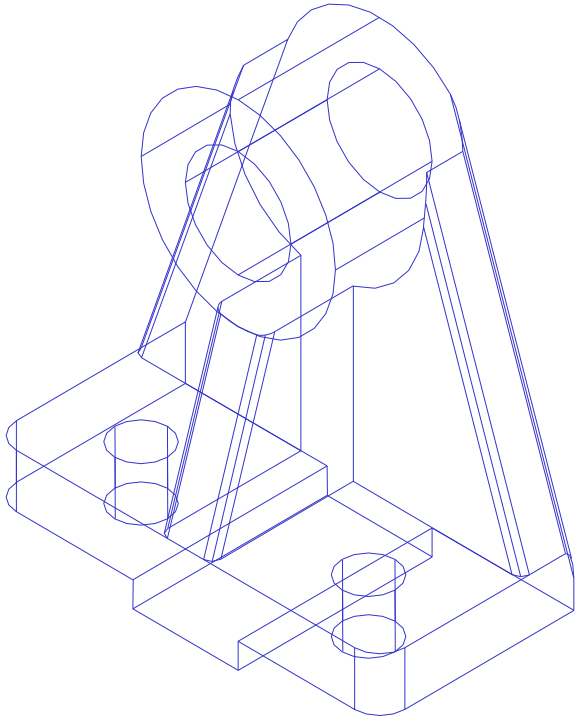
B-spline surface. This is a surface that can approximate or interpolate given input data (Fig. 6-9). It is a synthetic surface. It is a general surface like the Bezier surface but with the advantage of permitting local control of the surface.

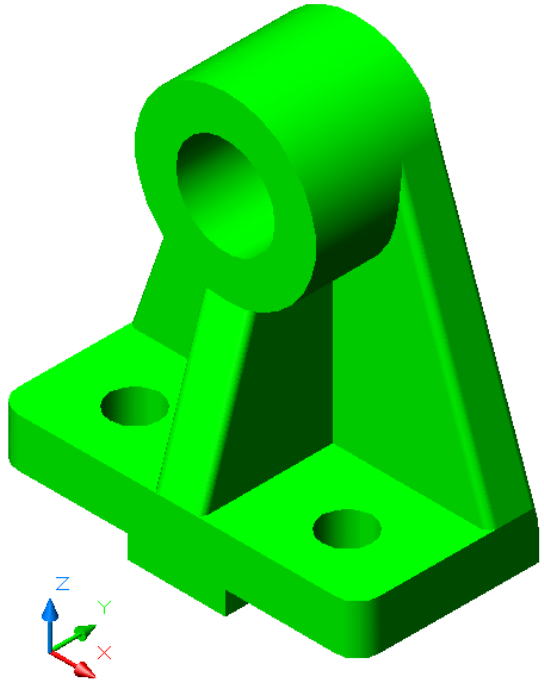


(a) Data points

Solid Modeling

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

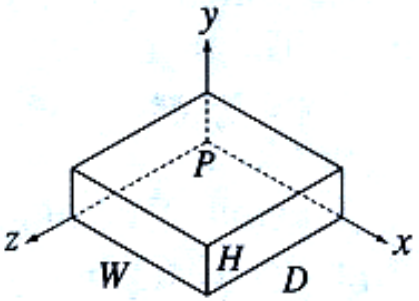




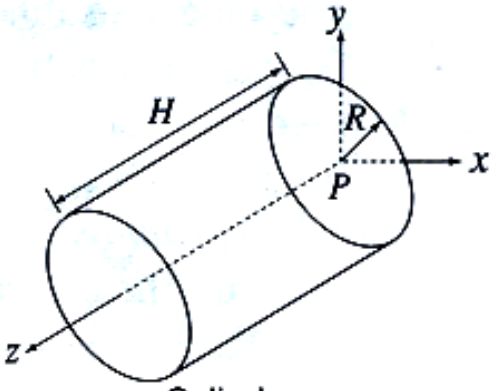
In CAD systems there are a number of representation schemes for solid modeling include:

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

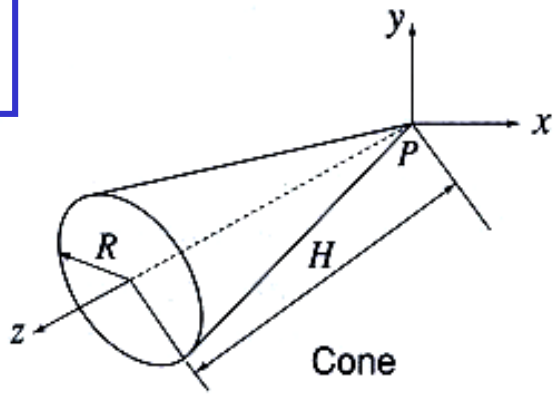
Primitive creation functions: These functions retrieve a solid of a simple shape from among the primitive solids stored in the program in advance and create a solid of the same shape but of the size specified by the user.



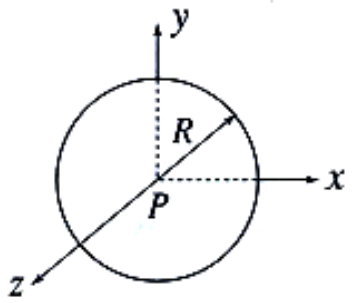
Block



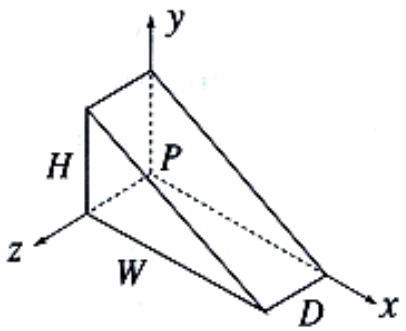
Cylinder



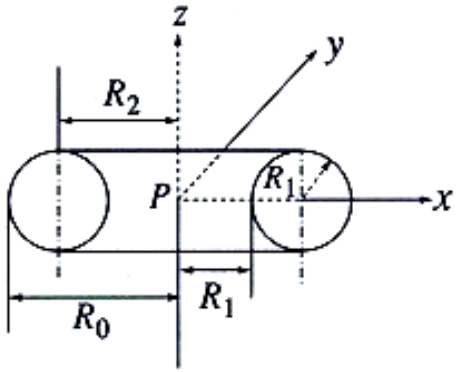
Cone



Sphere



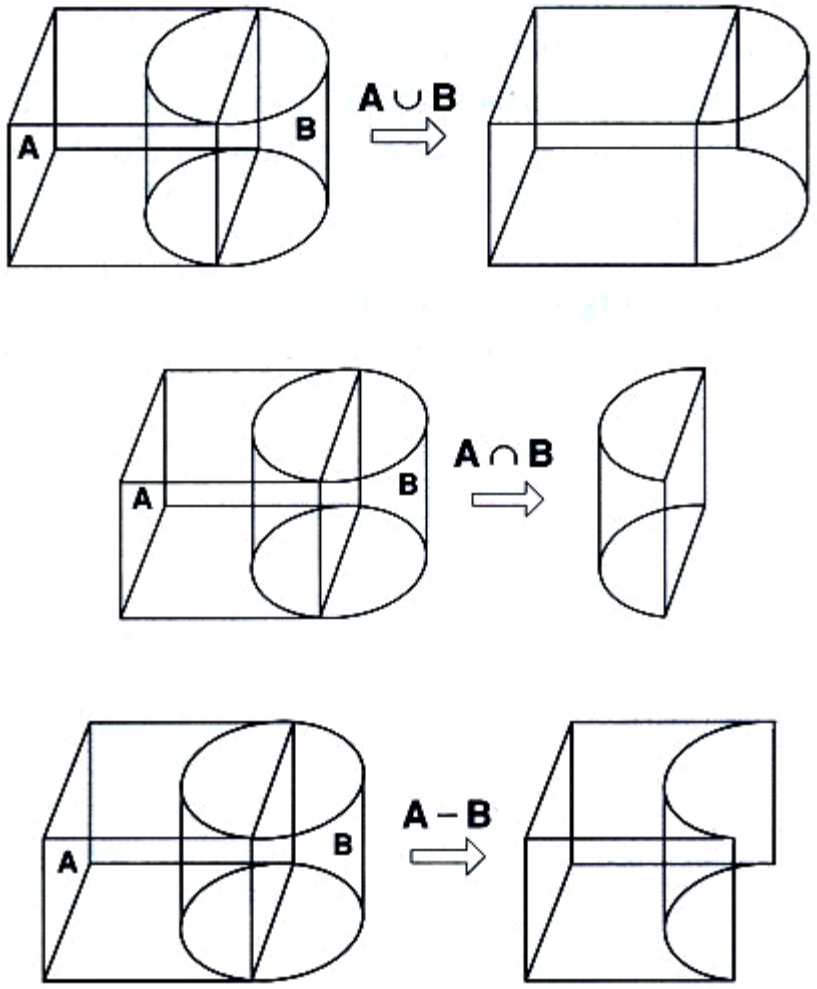
Wedge



Torus

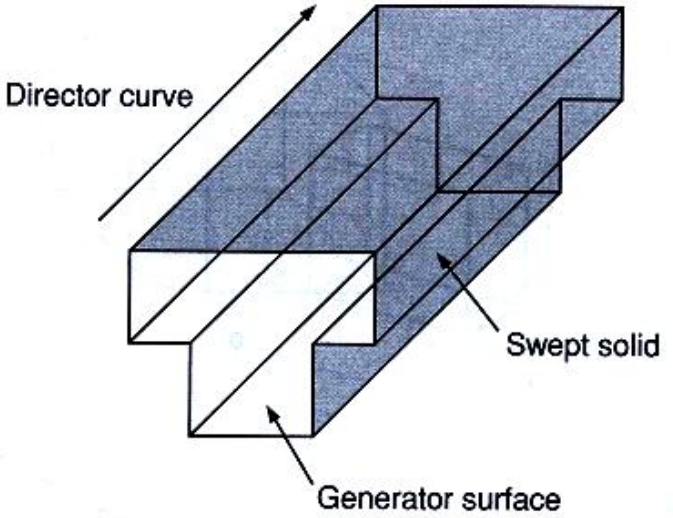
Constructive Solid Geometry (CSG)

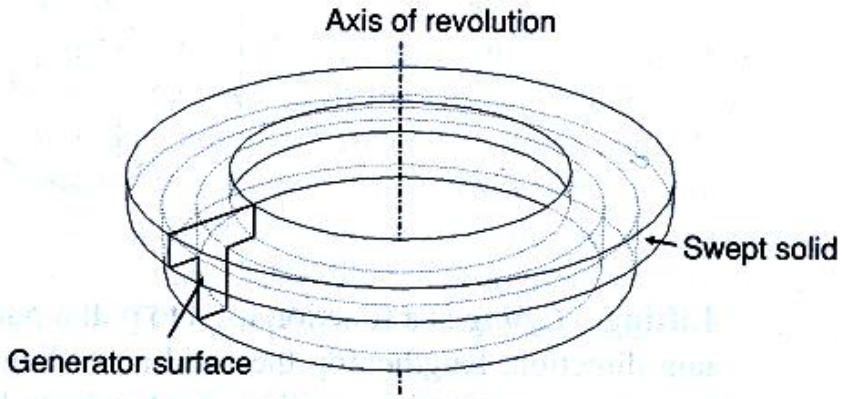
CSG uses primitive shapes as building blocks and Boolean set operators (U union, difference, and ∩ intersection) to construct an object.



Sweeping

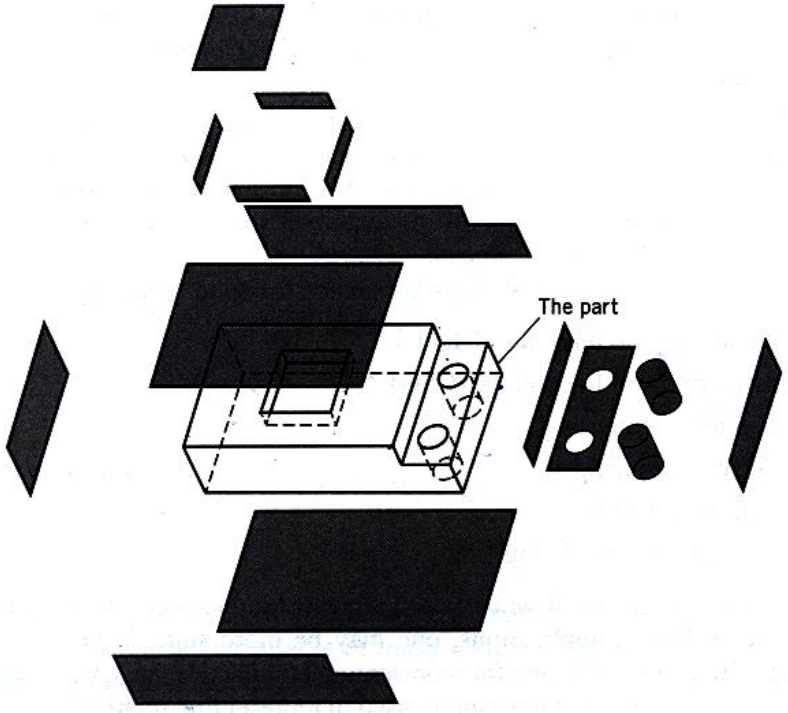
Sweeping *Sweeping* is a modeling function in which a planar closed domain is translated or revolved to form a solid. When the planar domain is translated, the modeling activity is called *translational sweeping;* when the planar region is revolved, it is called *swinging, or rotational sweeping.*

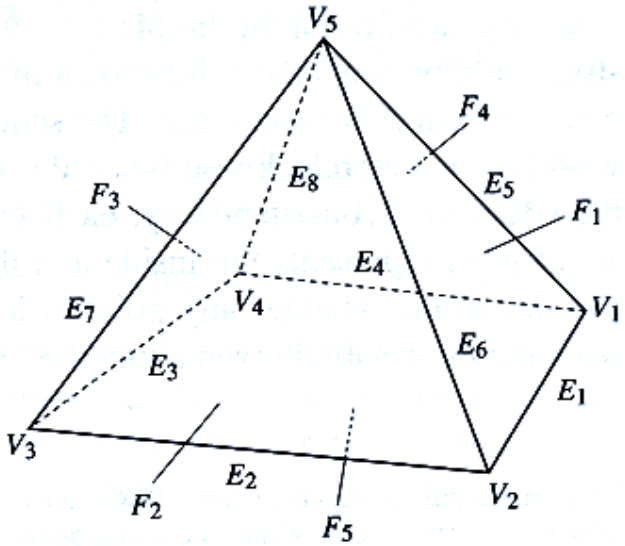




Boundary Representation

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