Computer Graphics

Introduction to computer graphics

- **Computer graphics is an art of drawing pictures, lines, charts, etc. using computers with the help of** programming. Computer graphics image is made up of number of pixels. **Pixel is the smallest addressable** graphical unit represented on the computer screen.

Advantages of computer graphics

- Computer graphics is one of the most effective and commonly used ways of communication with
- computer.
- It provides tools for producing picture of “real-world” as well as synthetic objects such as mathematical
- surfaces in 4D and of data that have no inherent geometry such as survey result.
- It has ability to show moving pictures thus possible to produce animations with computer graphics.
- With the use of computer graphics we can control the animation by adjusting the speed, portion of
- picture in view the amount of detail shown and so on.

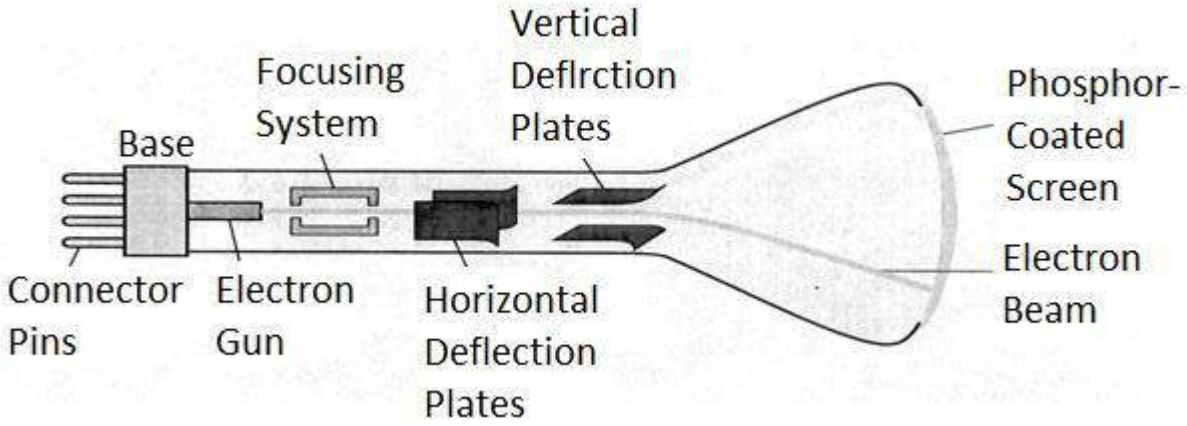
Application of computer graphics

- User interface: - Visual object which we observe on screen which communicates with user is one of the most useful applications of the computer graphics.
- Plotting of graphics and chart in industry, business, government and educational organizations drawing like bars, pie-charts, histogram's are very useful for quick and good decision making.
- Office automation and desktop publishing: - It is used for creation and dissemination of information. It is used in in-house creation and printing of documents which contains text, tables, graphs and other forms of drawn or scanned images or picture.
- Computer aided drafting and design: - It uses graphics to design components and system such as automobile bodies structures of building etc.

video-display devices

- Display devices are also known as output devices.
- Most commonly used output device in a graphics system is a video monitor

Cathode-ray-tubes



- It is an evacuated glass tube.
- An electron gun at the rear of the tube produce a beam of electrons which is directed towards the screen of the tube by a high voltage typically 15000 to 20000 volts
- Inner side screen is coated with phosphor substance which gives light when it is stroked bye electrons.
- Control grid controls velocity of electrons before they hit the phosphor.
- The control grid voltage determines how many electrons are actually in the electron beam. The negative the control voltage is the fewer the electrons that pass through the grid.
- Thus control grid controls Intensity of the spot where beam strikes the screen.
- The focusing system concentrates the electron beam so it converges to small point when hits the
- phosphor coating.
- Deflection system directs beam which decides the point where beam strikes the screen.
- Deflection system of the CRT consists of two pairs of parallel plates which are vertical and horizontal deflection plates.
- Voltage applied to vertical and horizontal deflection plates is control vertical and horizontal deflection respectively.

Random scan display

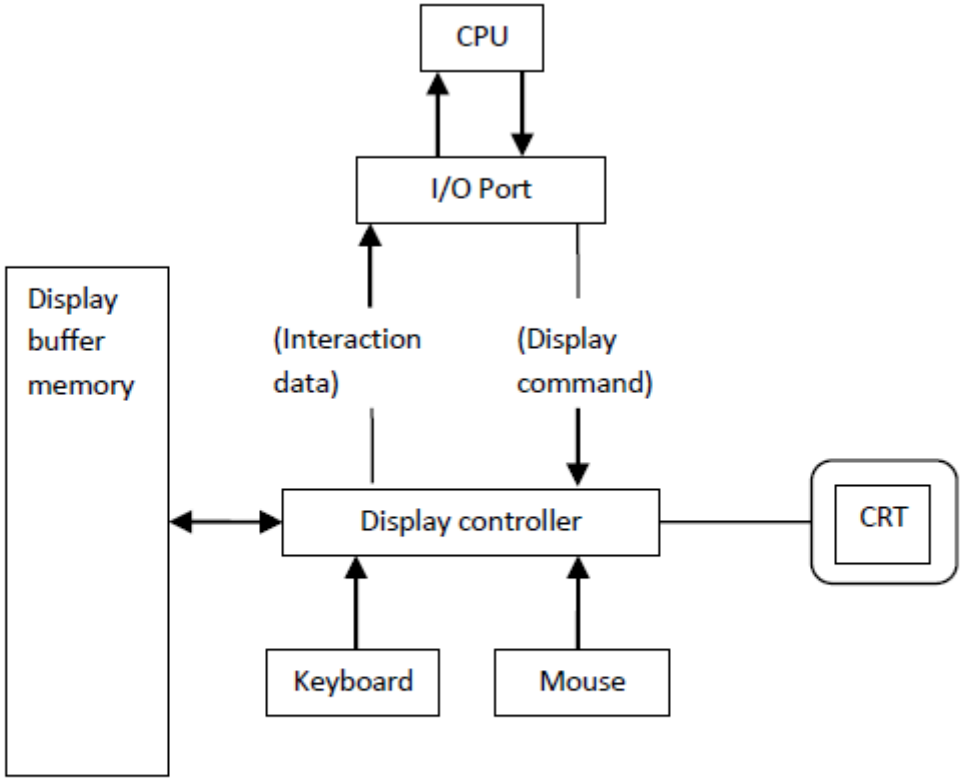
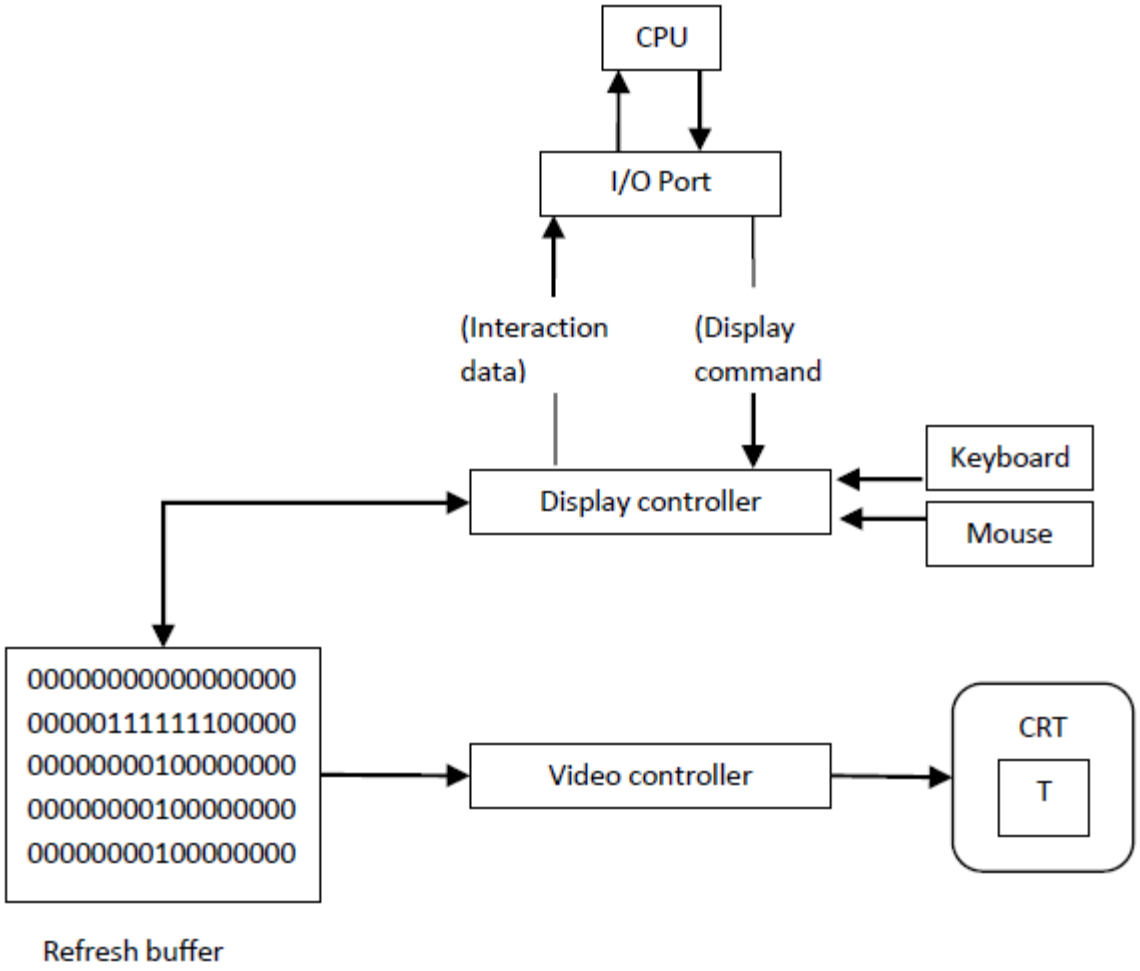


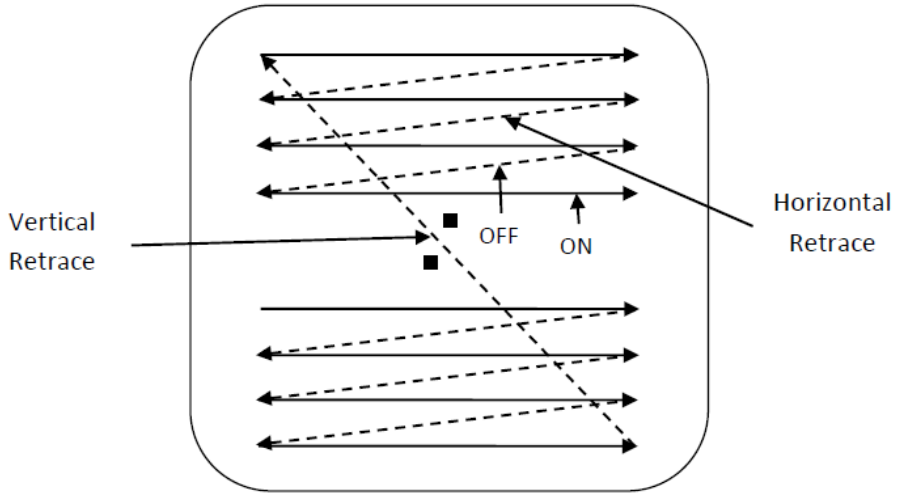
Fig. 1.2: - Architecture of a vector display.

- Vector scan display directly traces out only the desired lines on CRT.
- If we want line between point p1 & p2 then we directly drive the beam deflection circuitry which focus
- beam directly from point p1 to p2.
- If we do not want to display line from p1 to p2 and just move then we can blank the beam as we move it.
- To move the beam across the CRT, the information about both magnitude and direction is required. This
- information is generated with the help of vector graphics generator.
- Fig. 1.2 shows architecture of vector display. It consists of display controller, CPU, display buffer memory
- and CRT.
- Display controller is connected as an I/O peripheral to the CPU.
- Display buffer stores computer produced display list or display program.
- The Program contains point & line plotting commands with end point co-ordinates as well as character
- plotting commands.
- Display controller interprets command and sends digital and point co-ordinates to a vector generator.
- Vector generator then converts the digital co-ordinate value to analog voltages for beam deflection
- circuits that displace an electron beam which points on the CRT's screen.
- In this technique beam is deflected from end point to end point hence this techniques is also called
- random scan.
- We know as beam strikes phosphors coated screen it emits light but that light decays after few
- milliseconds and therefore it is necessary to repeat through the display list to refresh the screen at least
- 30 times per second to avoid flicker.
- As display buffer is used to store display list and used to refreshing, it is also called refresh buffer

raster-scan systems

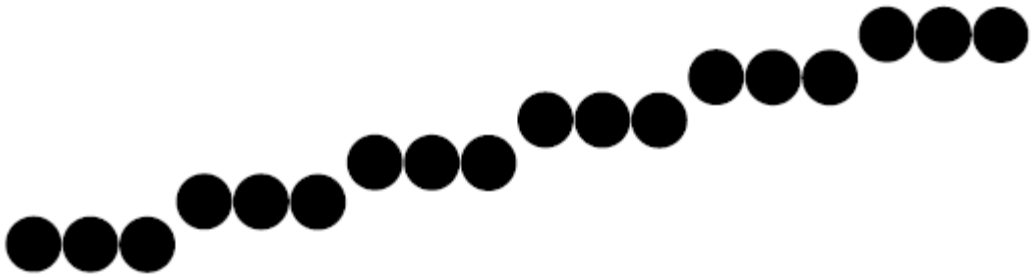


- Fig. 1.3 shows the architecture of Raster display. It consists of display controller, CPU, video controller,
- refresh buffer, keyboard, mouse and CRT.
- The display image is stored in the form of 1's and 0's in the refresh buffer.
- The video controller reads this refresh buffer and produces the actual image on screen.
- It will scan one line at a time from top to bottom & then back to the top
- In this method the horizontal and vertical deflection signals are generated to move the beam all over the
- screen in a pattern
- Here beam is swept back & forth from left to the right.
- When beam is moved from left to right it is ON.



Points and Lines

- Point plotting is done by converting a single coordinate position furnished by an application program into appropriate operations for the output device in use.
- Line drawing is done by calculating intermediate positions along the line path between two specified endpoint positions.
- The output device is then directed to fill in those positions between the end points with some color.
- For some device such as a pen plotter or random scan display, a straight line can be drawn smoothly from one end point to other.
- Digital devices display a straight line segment by plotting discrete points between the two endpoints.
- Discrete coordinate positions along the line path are calculated from the equation of the line.
- For a raster video display, the line intensity is loaded in frame buffer at the corresponding pixel positions.
- Reading from the frame buffer, the video controller then plots the screen pixels.
- Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints.
- For example line position of (12.36, 23.87) would be converted to pixel position (12, 24).



- The stair step shape is noticeable in low resolution system, and we can improve their appearance
- somewhat by displaying them on high resolution system.
- More effective techniques for smoothing raster lines are based on adjusting pixel intensities along the
- line paths.
- For raster graphics device-level algorithms discuss here, object positions are specified directly in integer
- device coordinates.
- Pixel position will referenced according to scan-line number and column number which is illustrated by
- follow

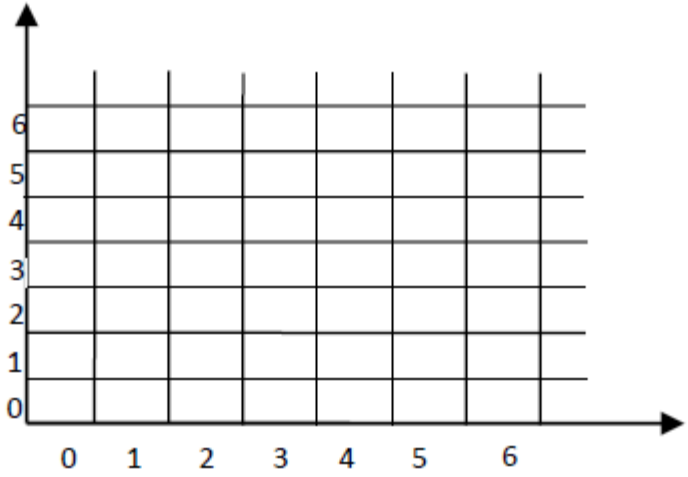


Fig. 2.2: - Pixel positions referenced by scan-line number and column number.

Line Drawing Algorithms

1. A line in Computer graphics is a portion of straight line that extends indefinitely in opposite direction.
2. It is defined by its two end points.
3. Its density should be independent of line length.

the slope intercept equation for a line:

$$y = mx + b \qquad (1)$$

where, **m** = Slope of the line

b = the y intercept of a line

- The Cartesian slop-intercept equation for a straight line is "$y = mx + b$" with 'm' representing slop and 'b' as the intercept.
- The two endpoints of the line are given which are say $(x1, y1)$ and $(x2, y2)$. We can determine values for the slope m by equation:
- $m = (y2 - y1)/(x2 - x1)$
- We can determine values for the intercept b by equation:
- $b = y1 - m * x1$
- For the given interval Δx along a line, we can compute the corresponding y interval Δy as:
- $\Delta y = m * \Delta x$
- Similarly for Δx:
- $\Delta x = \Delta y/m$

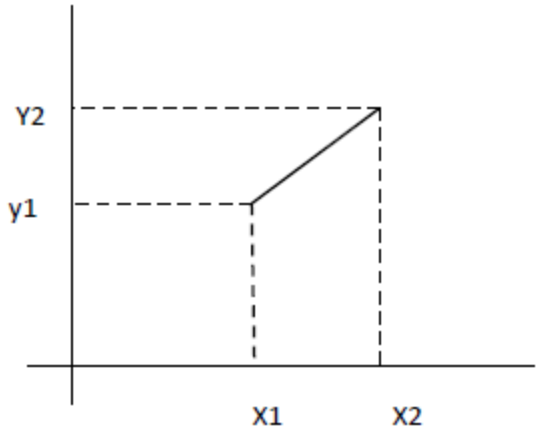


Fig. 2.3: - Line path between endpoint positions.

Midpoint Circle Algorithm

- We will first calculate pixel positions for a circle centered around the origin (0,0). Then, each calculated position (x,y) is moved to its proper screen position by adding xc to x and yc to y

- Note that along the circle section from x=0 to x=y in the first octant, the slope of the curve varies from 0 to -1

- Circle function around the origin is given by

$$fcircle(x,y) = x^2 + y^2 - r^2$$

- Any point (x,y) on the boundary of the circle satisfies the equation and circle function is zero

Scan-line polygon-fill algorithms

- For each scan-line that crosses the polygon, the edge intersections are sorted from left to right, and then pixel positions between and including each intersection pair are set to the specified fill color

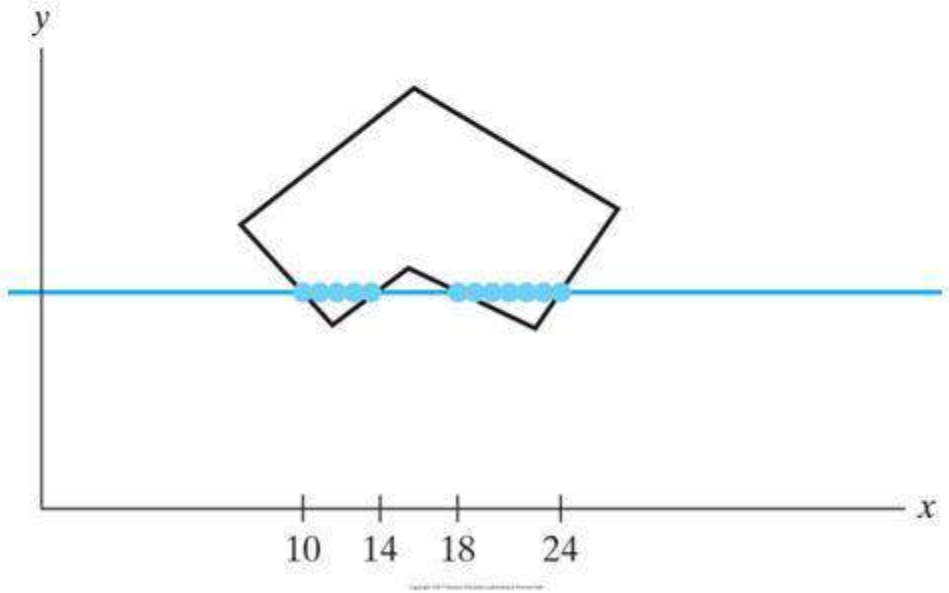
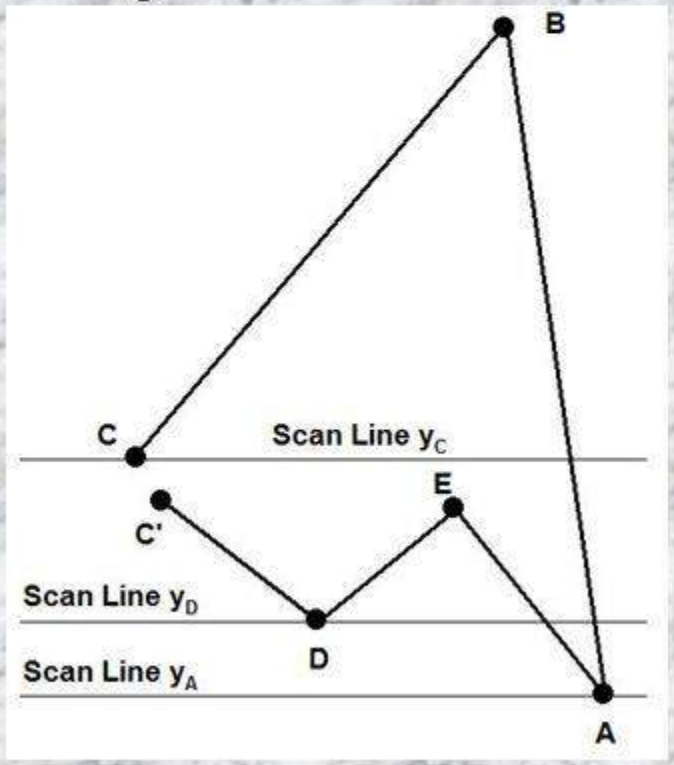
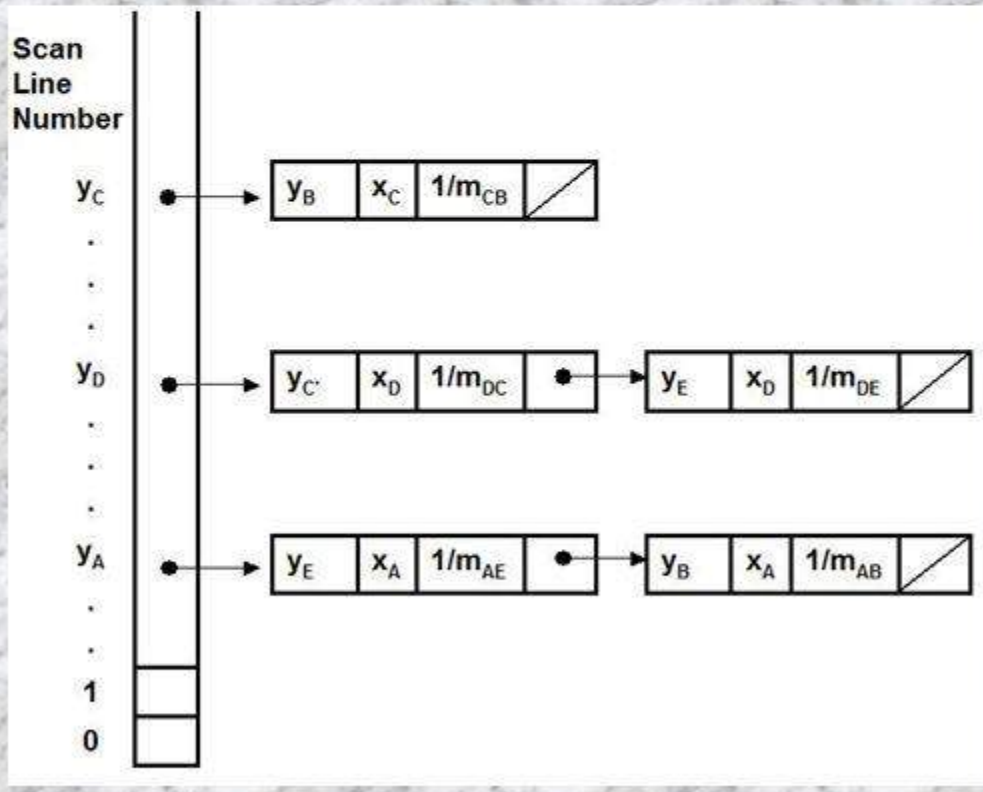


Figure 6-46 Interior pixels along a scan line passing through a polygon fill area.

The Scan-Line Polygon Fill Algorithm

- Each **entry** in the table for a particular scan line contains the **maximum y** value for that edge, the **x-intercept** value (**at the lower vertex**) for the edge, and the **inverse slope** of the edge.





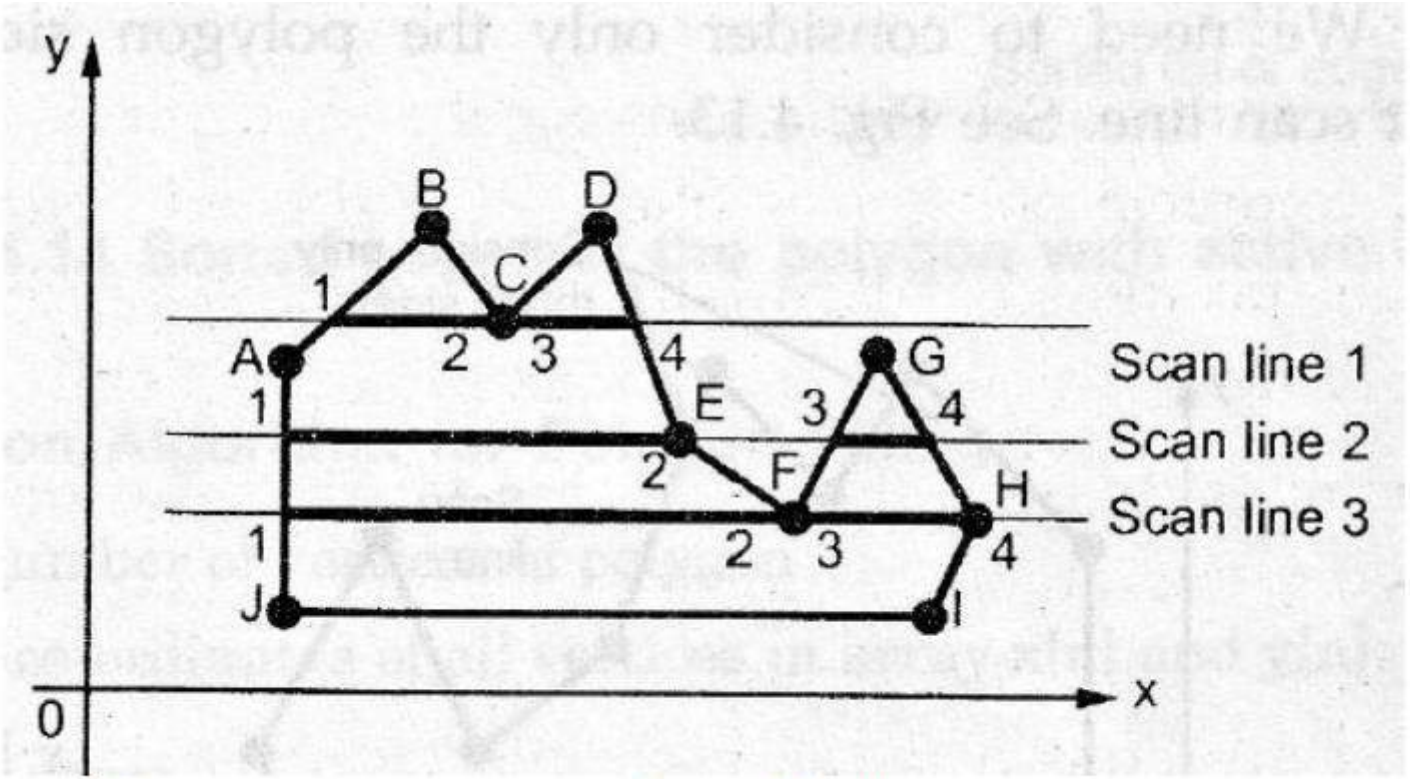
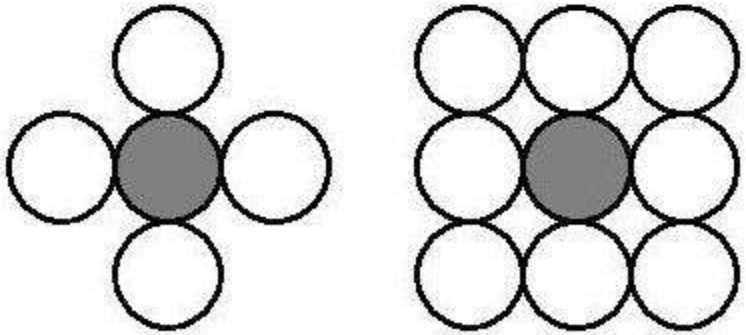


Fig. 2.18: - Intersection points along the scan line that intersect polygon vertices.

Food Fill Algorithm	Boundary Fill Algorithm
1.In flood fill algorithm the area is defined with multiple colour	1 In Boundary fill algorithm the area is defined with single colour.
2. The interior is coloured with any colour.	2. The interior point is replaced with new colour.
3. The old colour is replaced with new colour.	3. The interior points are replaced with new colour.
4. A flood fill may use an unpredictable amount of memory to finish because it isn't known how many sub-fills will be spawned.	4. Boundary fill is usually more complicated but it is a linear algorithm and doesn't require recursion.
5. It is Time Consuming.	5. It is less time Consuming.

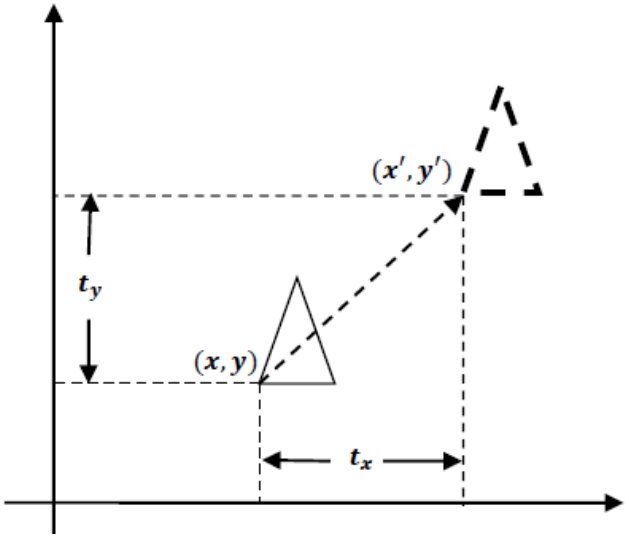
Boundary Fill Algorithm

- The order of pixels that should be added to stack using
- **4-connected**
  - above, below, left, and right.
- **8-connected** is above, below, left, right, above-left, above-right, below-left, and below-right.



Unit-2

Translation



- It is a transformation that used to reposition the object along the straight line path from one coordinate location to another.
- It is rigid body transformation so we need to translate whole object.
- We translate two dimensional point by adding translation distance $\boldsymbol{t_x}$ and $\boldsymbol{t_y}$ to the original coordinate position $(\boldsymbol{x}, \boldsymbol{y})$ to move at new position $(\boldsymbol{x'}, \boldsymbol{y'})$ as:

  $\boldsymbol{x' = x + t_x}$ **&** $\boldsymbol{y' = y + t_y}$
- Translation distance pair $(\boldsymbol{t_x}, \boldsymbol{t_y})$ is called a **Translation Vector** or **Shift Vector**.
- We can represent it into single matrix equation in column vector as;

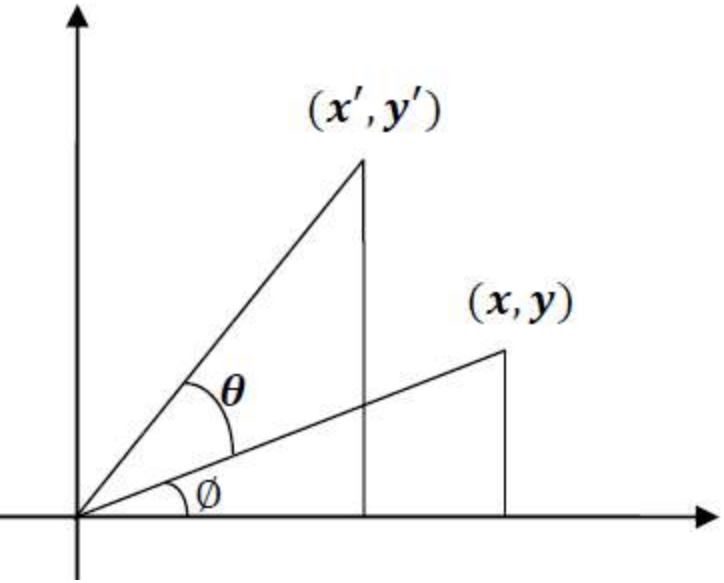
  $\boldsymbol{P' = P + T}$

  $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$
- We can also represent it in row vector form as:

  $\boldsymbol{P' = P + T}$

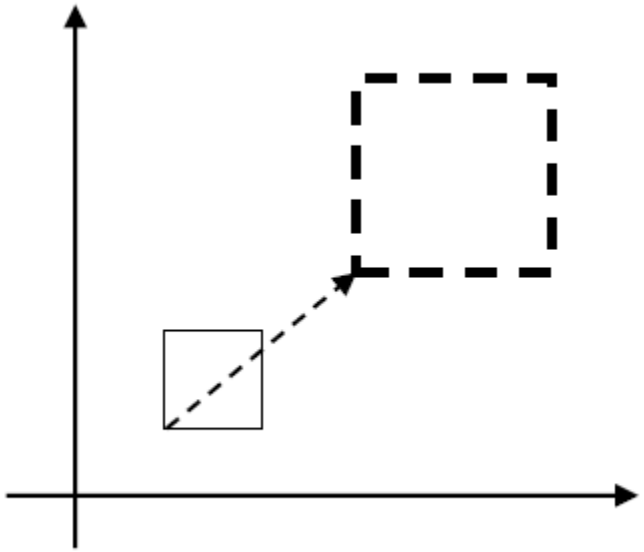
  $[x' \quad y'] = [x \quad y] + [t_x \quad t_y]$

Rotation



- It is a transformation that used to reposition the object along the circular path in the XY - plane.
- To generate a rotation we specify a rotation angle $\boldsymbol{\theta}$ and the position of the **Rotation Point** (**Pivot Point**) $(\boldsymbol{x_r}, \boldsymbol{y_r})$ about which the object is to be rotated.
- Positive value of rotation angle defines counter clockwise rotation and negative value of rotation angle defines clockwise rotation.
- We first find the equation of rotation when pivot point is at coordinate origin$(\mathbf{0}, \mathbf{0})$.

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



SCALING

- It is a transformation that used to alter the size of an object.
- This operation is carried out by multiplying coordinate value $(\boldsymbol{x}, \boldsymbol{y})$ with scaling factor $(\boldsymbol{s_x}, \boldsymbol{s_y})$ respectively.
- So equation for scaling is given by:

  $\boldsymbol{x' = x \cdot s_x}$

  $\boldsymbol{y' = y \cdot s_y}$
- These equation can be represented in column vector matrix equation as:

  $\boldsymbol{P' = S \cdot P}$

  $$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_x} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{s_y} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$
- Any positive value can be assigned to $(\boldsymbol{s_x}, \boldsymbol{s_y})$.

REFLECTION

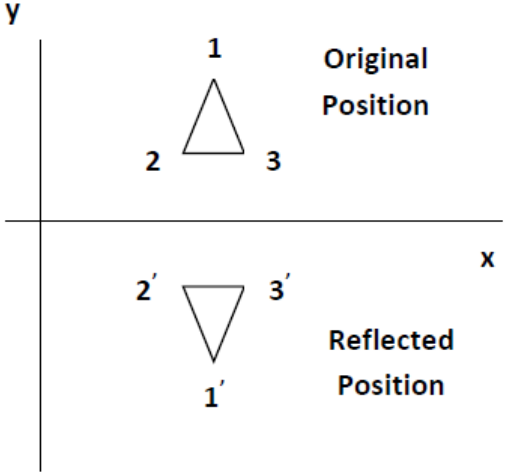


Fig. 3.9: - Reflection about x - axis.

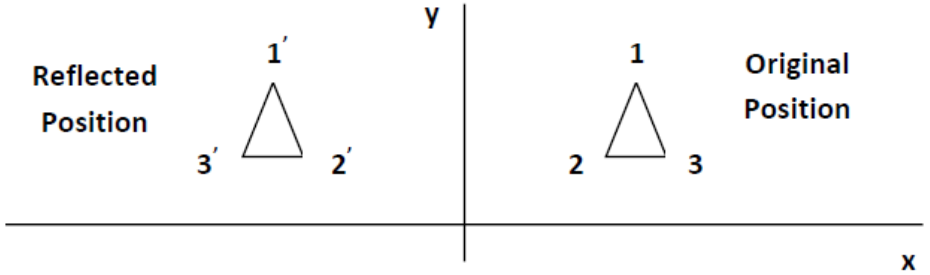


Fig. 3.10: - Reflection about y - axis.

- A reflection is a transformation that produces a mirror image of an object.
- The mirror image for a two –dimensional reflection is generated relative to an **axis of reflection** by rotating the object 180° about the reflection axis.
- Reflection gives image based on position of axis of reflection. Transformation matrix for few positions are discussed here.

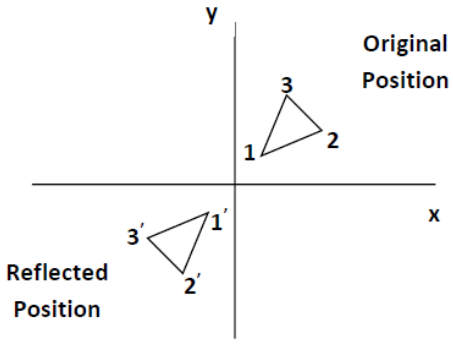


Fig. 3.11: - Reflection about origin.

Shear

- A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called **shear**.
- Two common shearing transformations are those that shift coordinate x values and those that shift y values.

Shear in $\boldsymbol{x - direction}$.

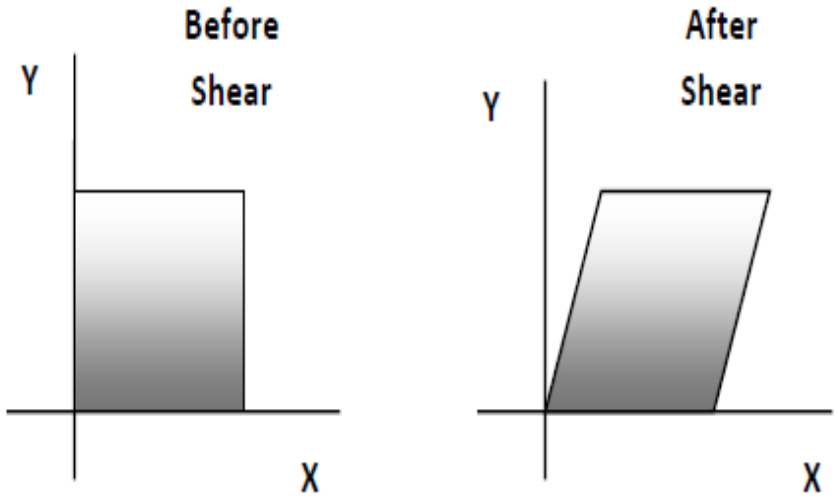


Fig. 3.13: - Shear in x-direction.

Shear in ***y – direction*** .

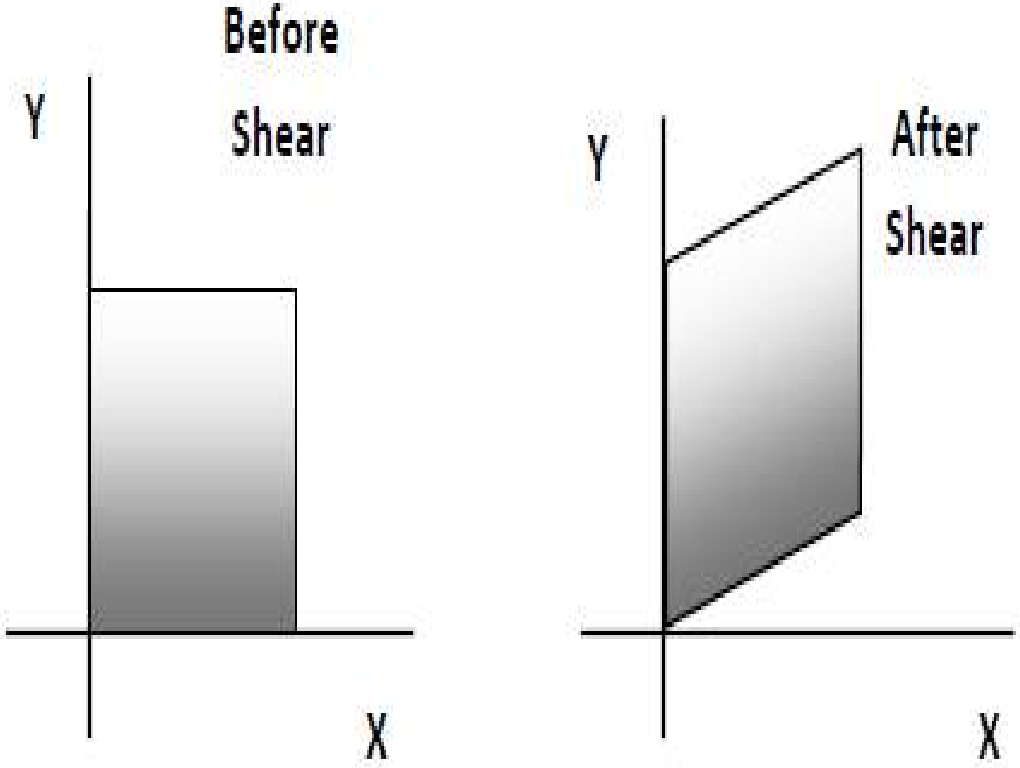


Fig. 3.14: - Shear in y-direction.

Matrix Representation and homogeneous coordinates

- Many graphics application involves sequence of geometric transformations.
- For example in design and picture construction application we perform Translation, Rotation, and scaling to fit the picture components into their proper positions.
- For efficient processing we will reformulate transformation sequences.
- We have matrix representation of basic transformation and we can express it in the general matrix form as:

$$P' = M_1 \cdot P + M_2$$

Where P and P' are initial and final point position, M_1 contains rotation and scaling terms and M_2 contains translation al terms associated with pivot point, fixed point and reposition.

Translation

$$P' = T_{(t_x,t_y)} \cdot P$$

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

NOTE: - Inverse of translation matrix is obtain by putting $-t_x$ & $-t_y$ instead of t_x & t_y.

Rotation

$$P' = R_{(\theta)} \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling

$$P' = S_{(s_x,s_y)} \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

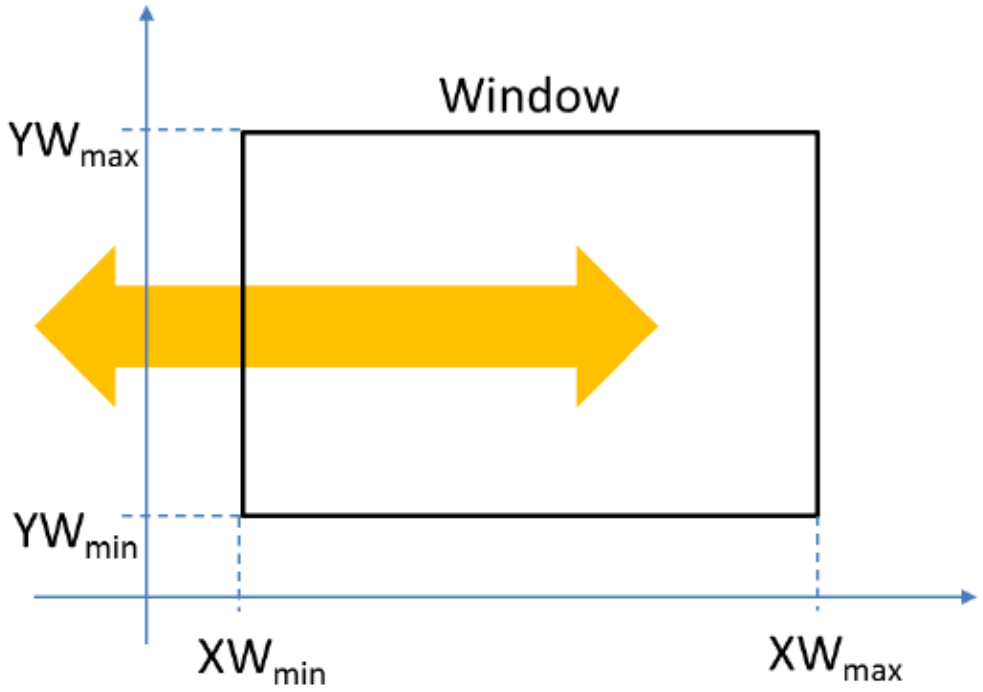
s_x s_y

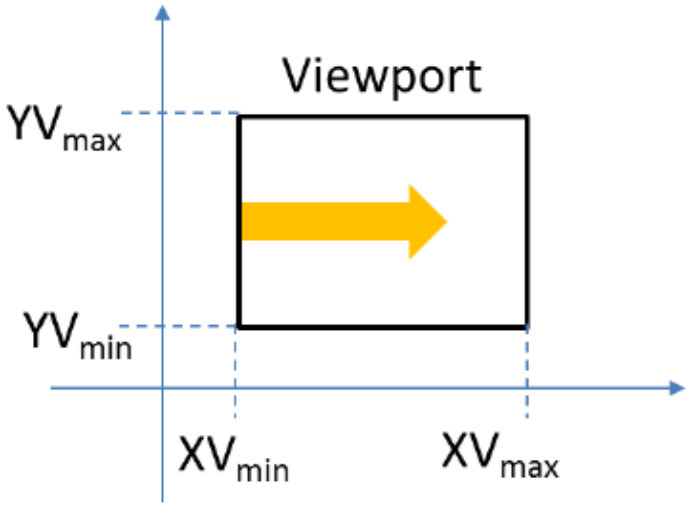
Composite Transformation

- We can set up a matrix for any sequence of transformations as a **composite transformation matrix** by calculating the matrix product of individual transformation.
- For column matrix representation of coordinate positions, we form composite transformations by multiplying matrices in order from right to left.

The Viewing Pipeline

- **Window:** Area selected in world-coordinate for display is called window. It defines what is to be viewed.
- **Viewport:** Area on a display device in which window image is display (mapped) is called viewport. It defines where to display.
- In many case window and viewport are rectangle, also other shape may be used as window and viewport.
- In general finding device coordinates of viewport from word coordinates of window is called as **viewing transformation.**
- Sometimes we consider this viewing transformation as window-to-viewport transformation but in general it involves more steps.





Viewing Coordinate Reference Frame

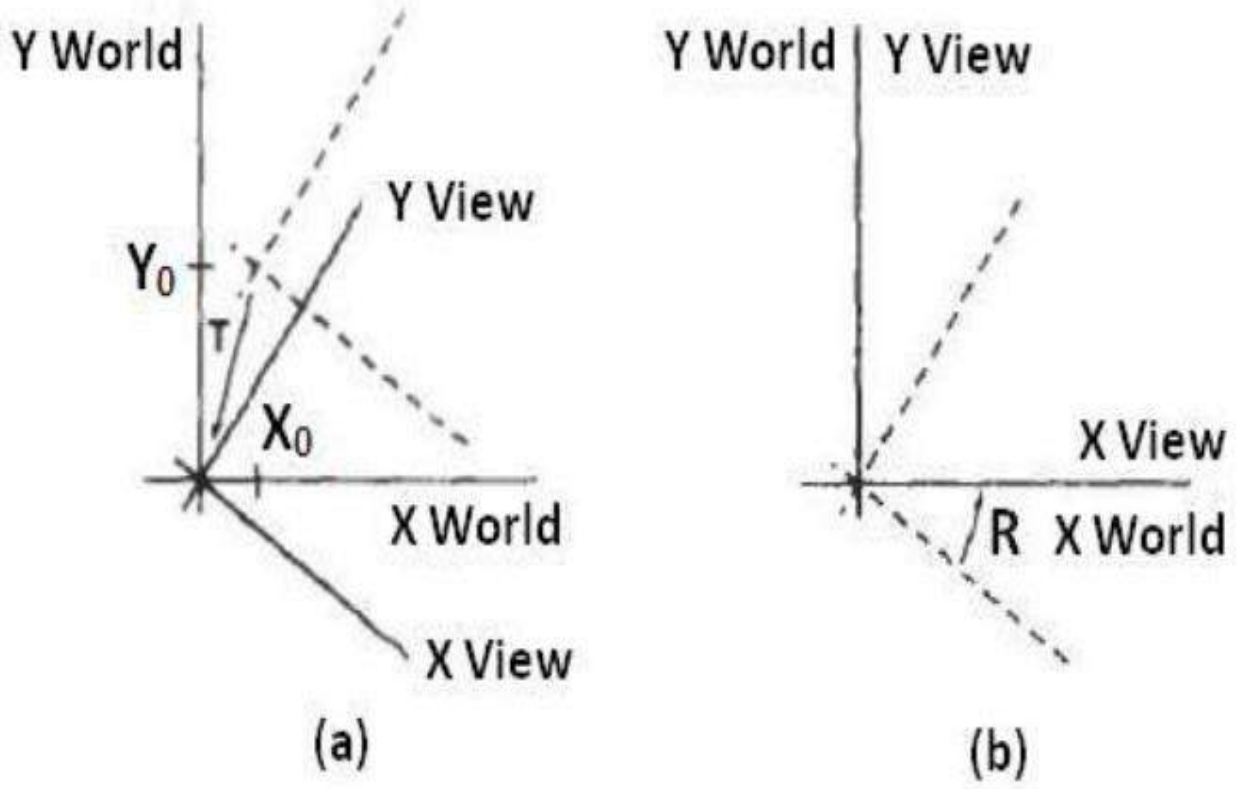


Fig. 3.3: - A viewing-coordinate frame is moved into coincidence with the world frame in two steps: (a) translate the viewing origin to the world origin, and then (b) rotate to align the axes of the two systems.

- We can obtain reference frame in any direction and at any position.
- For handling such condition first of all we translate reference frame origin to standard reference frame origin and then we rotate it to align it to standard axis.
- In this way we can adjust window in any reference frame.

$$\mathbf{M_{wc,vc} = RT}$$

- Where T is translation matrix and R is rotation matrix.

Window-To-Viewport Coordinate Transformation

- Mapping of window coordinate to viewport is called window to viewport transformation.
- We do this using transformation that maintains relative position of window coordinate into viewport.
- That means center coordinates in window must be remains at center position in viewport.
- We find relative position by equation as follow:

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

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

- Solving by making viewport position as subject we obtain:

$$x_v = x_{vmin} + (x_w - x_{wmin})s_x$$

$$y_v = y_{vmin} + (y_w - y_{wmin})s_y$$

- Where scaling factor are :

$$s_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$s_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

Number of display device can be used in application and for each we can use different window-to-viewport transformation. This mapping is called the **workstation transformation.**

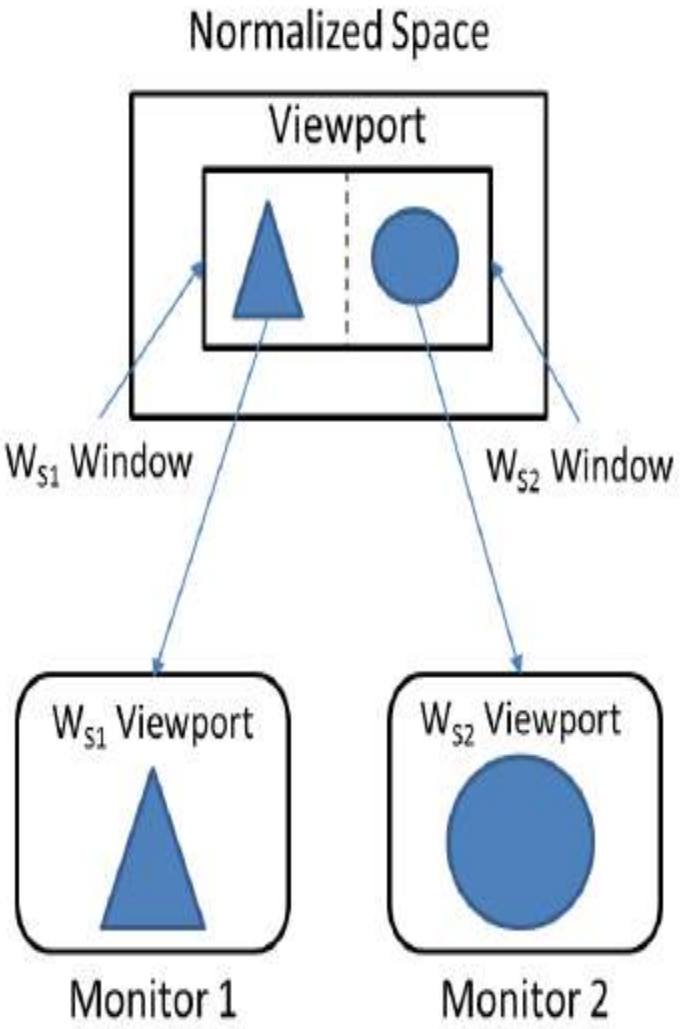


Fig. 3.4: - workstation transformation.

Line Clipping

- Line clipping involves several possible cases.
  1. Completely inside the clipping window.
  2. Completely outside the clipping window.
  3. Partially inside and partially outside the clipping window.

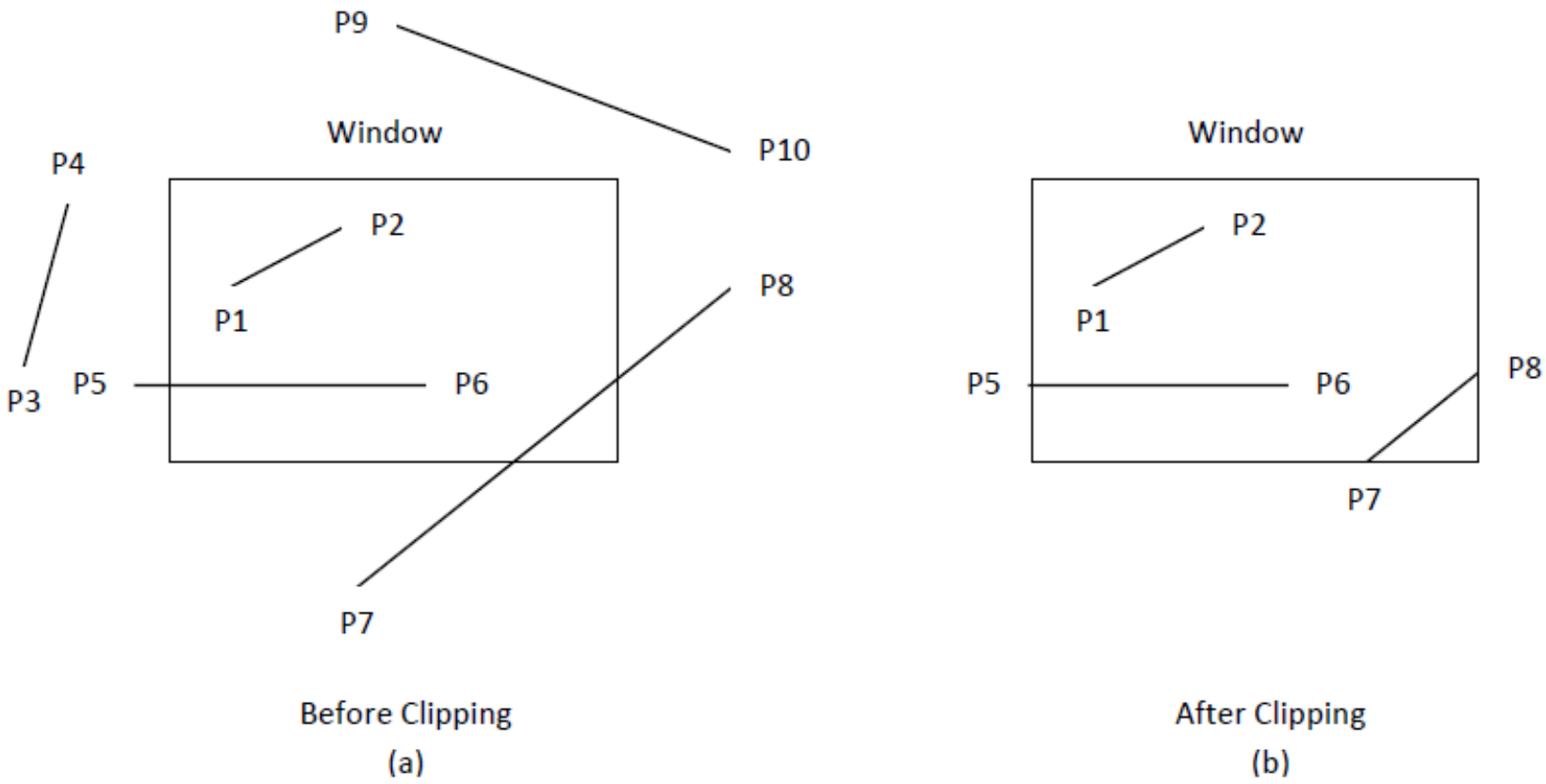


Fig. 3.5: - Line clipping against a rectangular window.

Cohen-Sutherland Line Clipping

- This is one of the oldest and most popular line-clipping procedures.

Region and Region Code

- In this we divide whole space into nine region and assign 4 bit code to each endpoint of line depending on the position where the line endpoint is located.

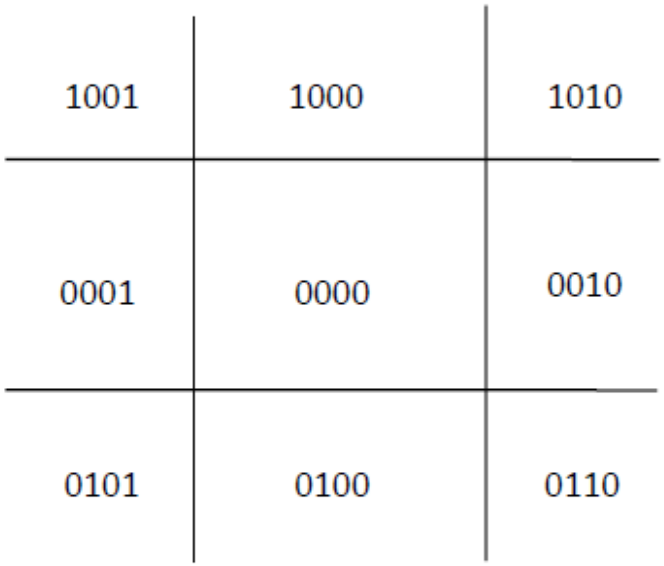


Fig. 3.6: - Workstation transformation.

- Figure 3.6 shows code for line end point which is fall within particular area.
- Code is deriving by setting particular bit according to position of area.

  Set bit 1: For left side of clipping window.

  Set bit 2: For right side of clipping window.

  Set bit 3: For below clipping window.

  Set bit 4: For above clipping window.
- All bits as mention above are set means 1 and other are 0

Algorithm

Step-1:

Assign region code to both endpoint of a line depending on the position where the line endpoint is located.

Step-2:

If both endpoint have code '0000'

Then line is completely inside.

Otherwise

Perform logical ending between this two codes.

If result of logical ending is non-zero

Line is completely outside the clipping window.

Otherwise

Calculate the intersection point with the boundary one by one.

Divide the line into two parts from intersection point.

Recursively call algorithm for both line segments.

Step-3:

Draw line segment which are completely inside and eliminate other line segment which found completely outside.

Cyrus Beck

- Cyrus Beck generally clips only once or twice unlike the Cohen Sutherland Algorithm where the lines are clipped about four times.
- Clipping the line twice has the idea that the line will be clipped for the 1st time when it enters the box and 2nd time when it exists.
- Cyrus-Beck is a general algorithm and can be used with a convex polygon clipping window unlike Sutherland-Cohen that can be used only on a rectangular clipping area.

Sutherland-Hodgeman Polygon Clipping

- For correctly clip a polygon we process the polygon boundary as a whole against each window edge.
- This is done by whole polygon vertices against each clip rectangle boundary one by one.
- Beginning with the initial set of polygon vertices we first clip against the left boundary and produce new sequence of vertices.
- Then that new set of vertices is clipped against the right boundary clipper, a bottom boundary clipper and a top boundary clipper, as shown in figure below.

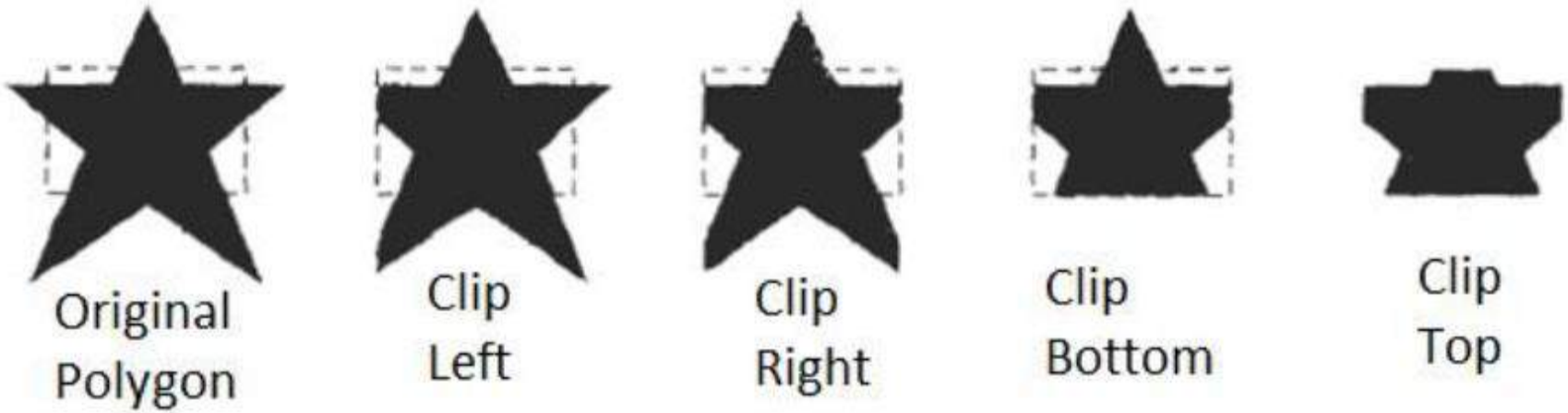


Fig. 3.11: - Clipping a polygon against successive window boundaries.

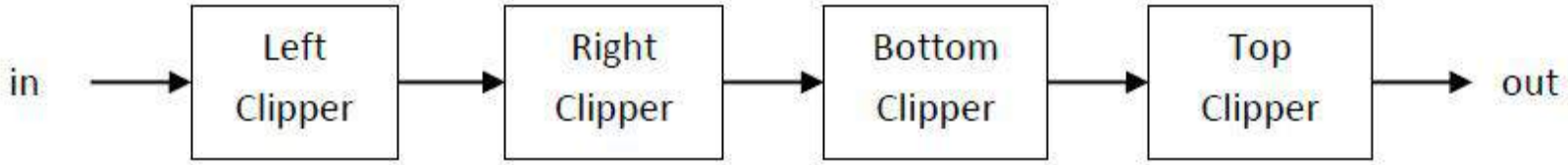
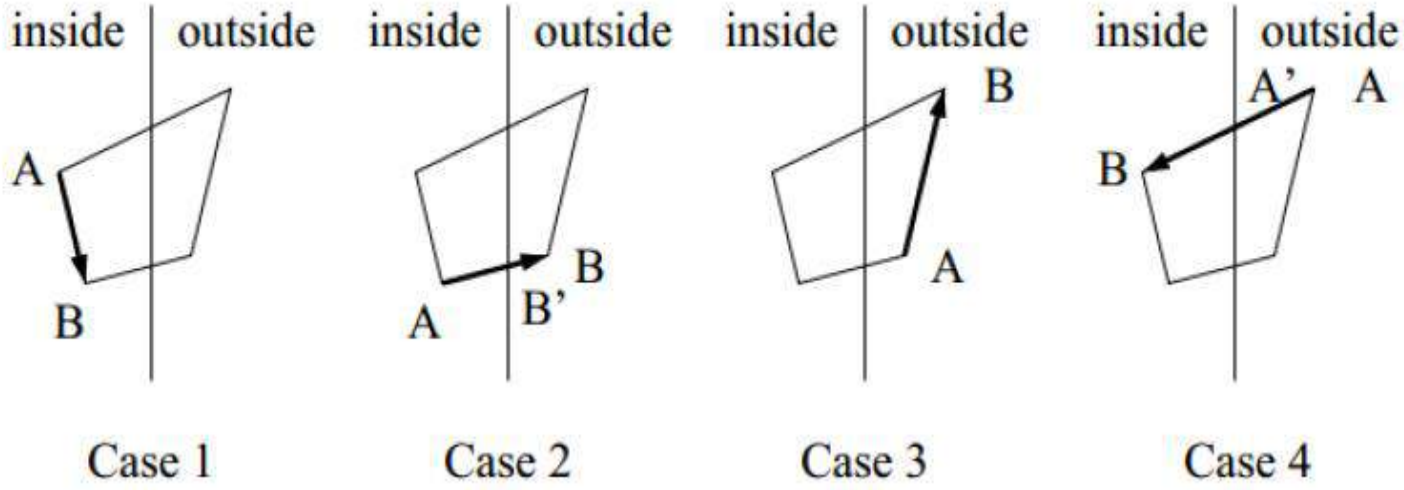
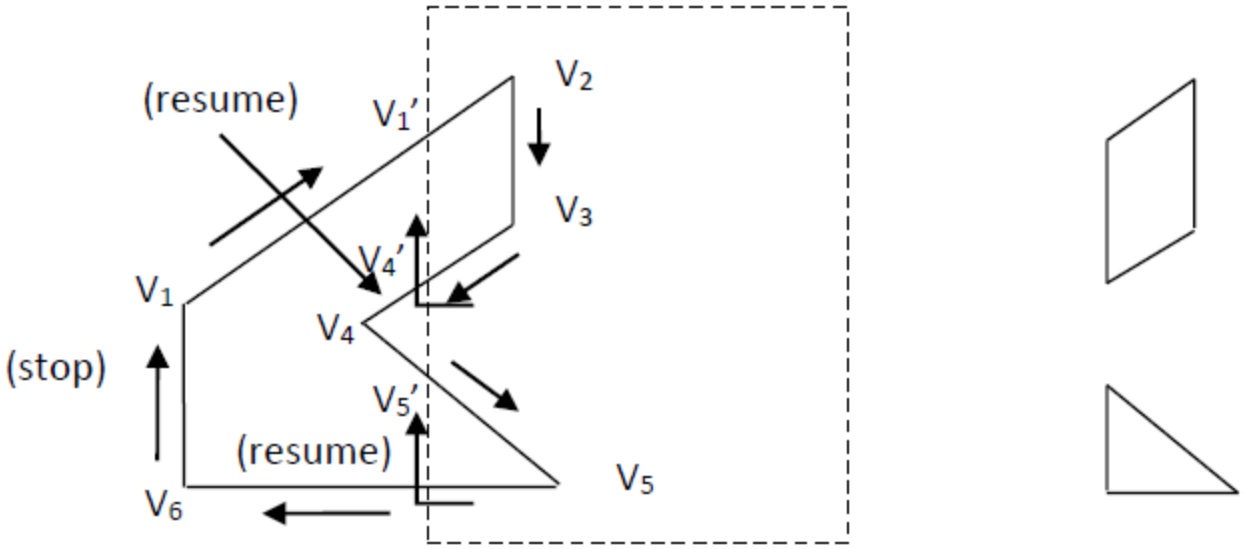


Fig. 3.12: - Processing the vertices of the polygon through boundary clipper.

- There are four possible cases when processing vertices in sequence around the perimeter of a polygon.





Unit-3

Polygon Surfaces

- Most commonly used boundary representation.

Polygon table

- Specify a polygon surfaces using vertex coordinates and attribute parameter.

Polygon data table organized into 2 group.

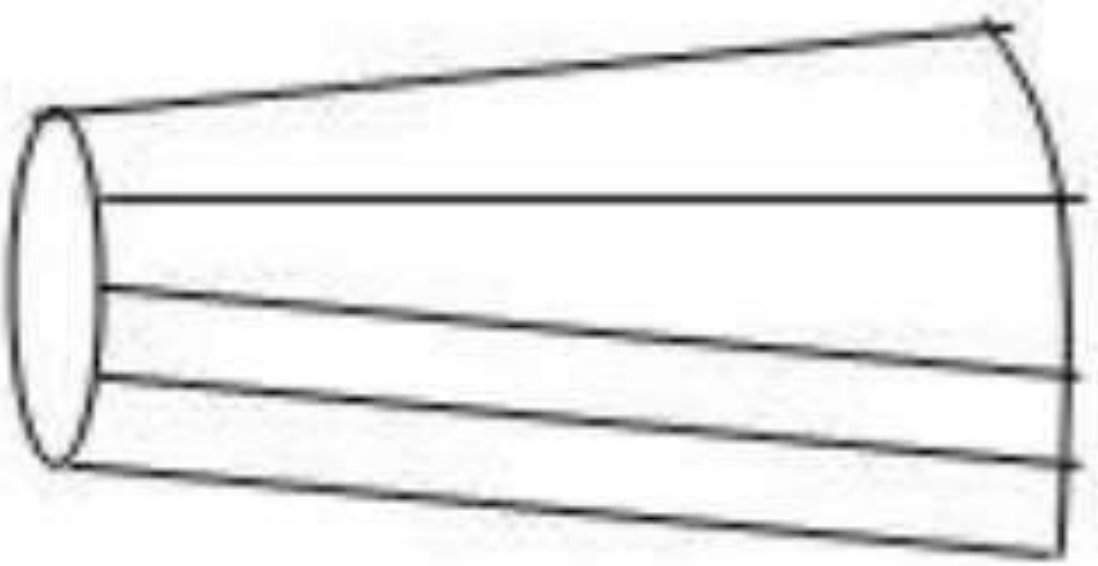
1. **Geometric data table**: vertex coordinate and parameter to identify the spatial orientation.

   3 lists

   Vertex table –coordinate values of each vertex.

   Edge table - pointer back to vertex table to identify the vertices for polygon edge.

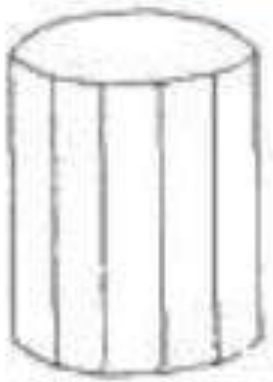
   Polygon table- pointer back to edge table to identify the edges for each polygon



A 3D object represented by polygons

Polygon Surfaces

- most commonly used boundary presentation
- set of surface polygons enclose the object interior
- Used in most graphics systems
- simplifies and speeds up the surface rendering and display of objects



ADRI JOVIN J.J., Asst. Professor, Department of Information Technology,
SRI RAMAKRISHNA INSTITUTE OF TECHNOLOGY

QUADRIC SURFACES

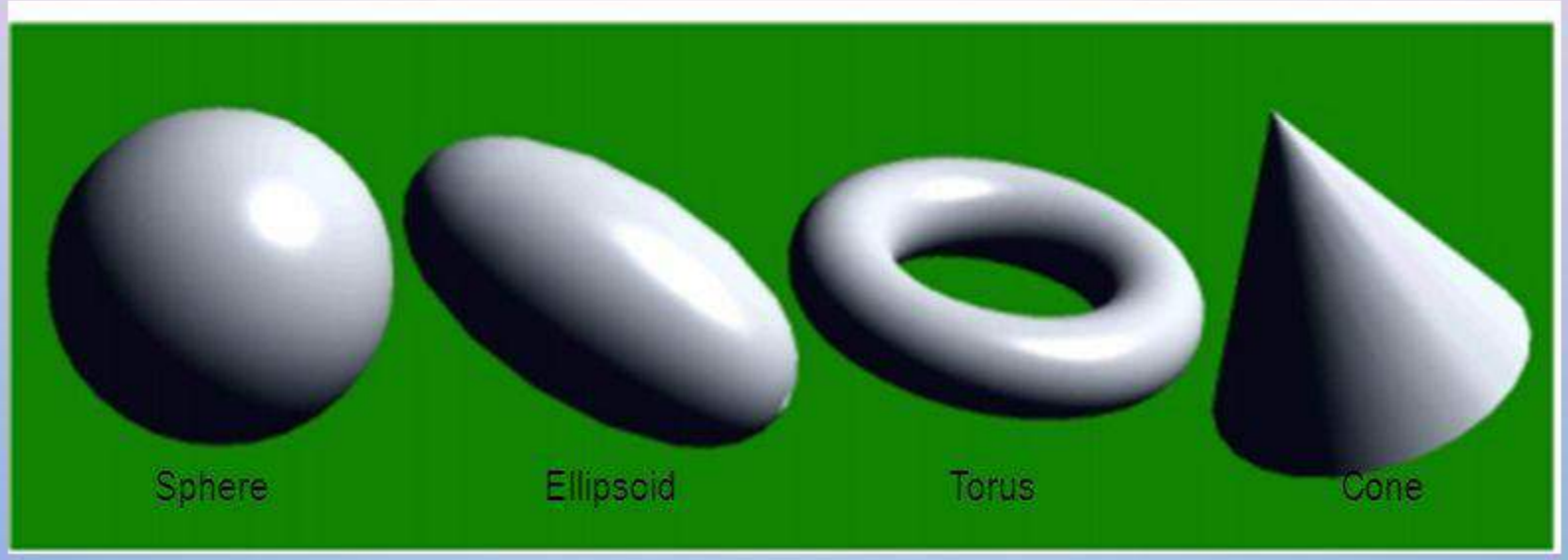
- A frequently used class of objects are the quadric surfaces, which are described with second-degree equations (quadratics). They include spheres, ellipsoids, tori, paraboloids, and hyperboloids.

- Quadric surfaces, particularly spheres and ellipsoids, are common elements of graphics scenes

Quadric Surfaces

- Surfaces represented by second- degree polynomials are quadratics. Sphere, ellipsoid, torus and cone come under general quadrics.

- Quadratic surfaces, particularly spheres and ellipsoids are common elements of graphic scenes.



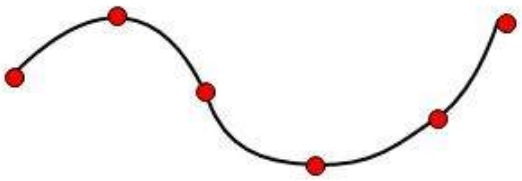
- In computer graphics, the term spline curve now refers to any composite curve formed with polynomial sections satisfying specified

continuity conditions at the boundary of the pieces.

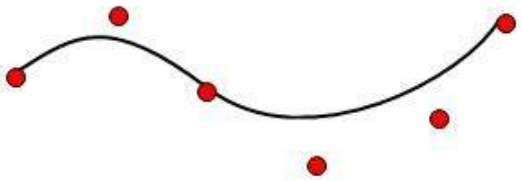
- **A spline surface can be** described with two sets of orthogonal spline curves.
- Splines are used in graphics applications to design curve and surface shapes, to digitize drawings for computer storage, and to specify animation paths for the objects or the camera in a scene

Spline Representations

- Spline curve: Curve consisting of continous curve segments approximated or interpolated on polygon control points.
- Spline surface: a set of two spline curves matched on a smooth surface.
- Interpolated: curve passes through control points
- Approximated: guided by control points but not necessarily passes through them.



Interpolated



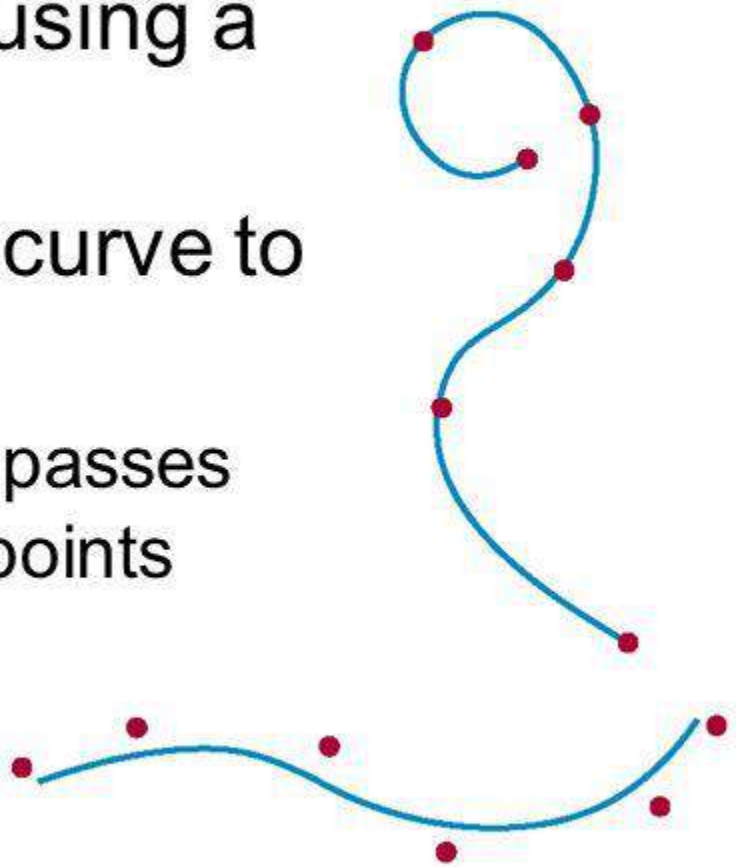
Approximated

Interpolation Vs Approximation

A spline curve is specified using a set of **control points**

There are two ways to fit a curve to these points:

- **Interpolation -** the curve passes through all of the control points
- **Approximation -** the curve does not pass through all of the control points



Splines are used in

1. Graphic applications to design **curve and surface** shapes.
2. To digitize drawings for computer storage.
3. To specify ***Animation paths*** for the objects for the camera in a scene.
4. Typical ***CAD*** applications for splines include the design automobile bodies.
5. Aircraft, Spacecraft surfaces & Ship hulls.

Hermite Curves

- A mathematical representation as a link between the algebraic & geometric form
- Defined by specifying the end points and tangent vectors at the end points
- Use of control points
  - Geometric points that control the shape
  - Algebraically: used for linear combination of basis functions

Hermite curve

- A **Hermite curve** is a **spline** where every piece is a third degree polynomial defined in **Hermite** form: that is, by its values and initial derivatives at the end points of the equivalent domain interval. Xk+1) individually. The resulting **spline** become continuous and will have first derivative

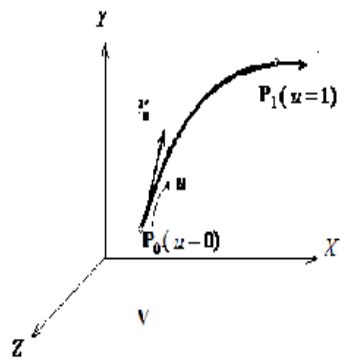
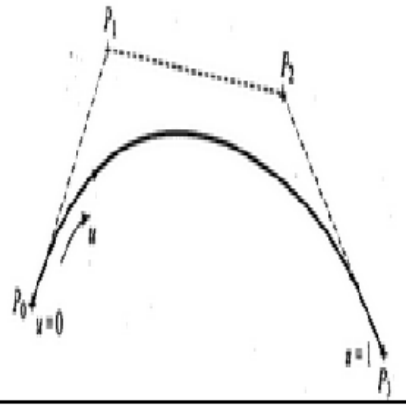
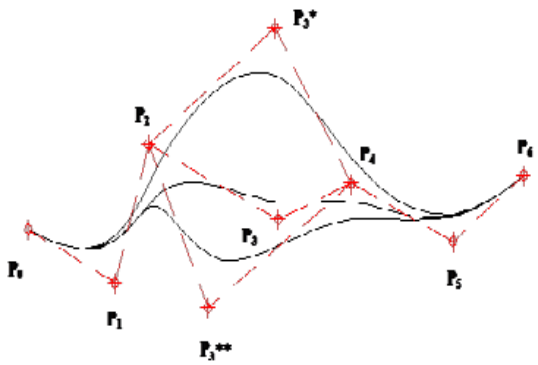
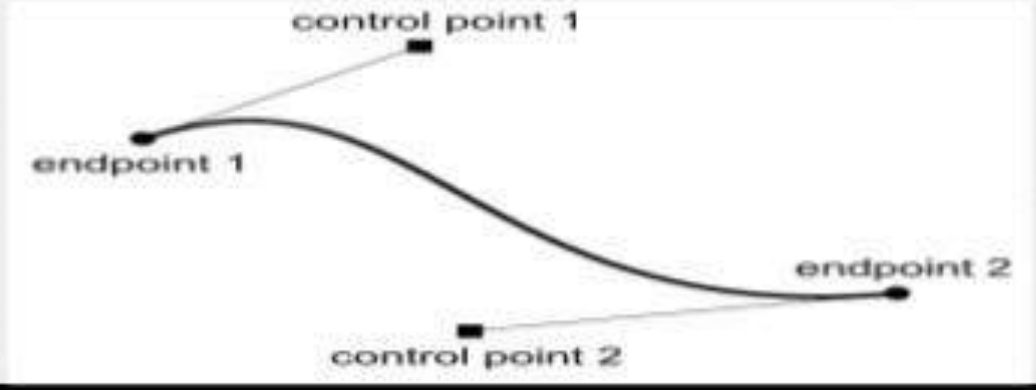
Parameter	Hermite cubic spline curve	Bezier curve	B-spline curve
Definition	It is 3rd degree polynomial with 4 data points and 4 coefficients.	It is curve of nth degree polynomial with n+1 no. of data points.	It is Bezier curve with varying degree.
Formula	$P(u)=\sum_{i=0}^{3} C_i u_i, 0 \le u \le 1$	$P(u)=\sum_{i=0}^{n} P_i B_i, n(u), 0 \le u \le 1$	$P(u)=\sum_{i=0}^{n} P_i N_i, k(u), 0 \le u \le u_{max}$
Advantages	Easy for computation	Curve can pass smoothly through number of data points. Smooth due to high degree continuity.	It has local control over shape of curve. Degree of curve is independent of data points.
Disadvantages	It cannot have control over more than 4 data points. Less smooth than Bezier and B-spline curves	Computation required is more due to higher degree. As it has global control over shape of curve, movement of points will give different shape to the curve.	Computational time increases with complexity of curve.
Figures			

Table 1: Comparison between synthetic curves

Definition

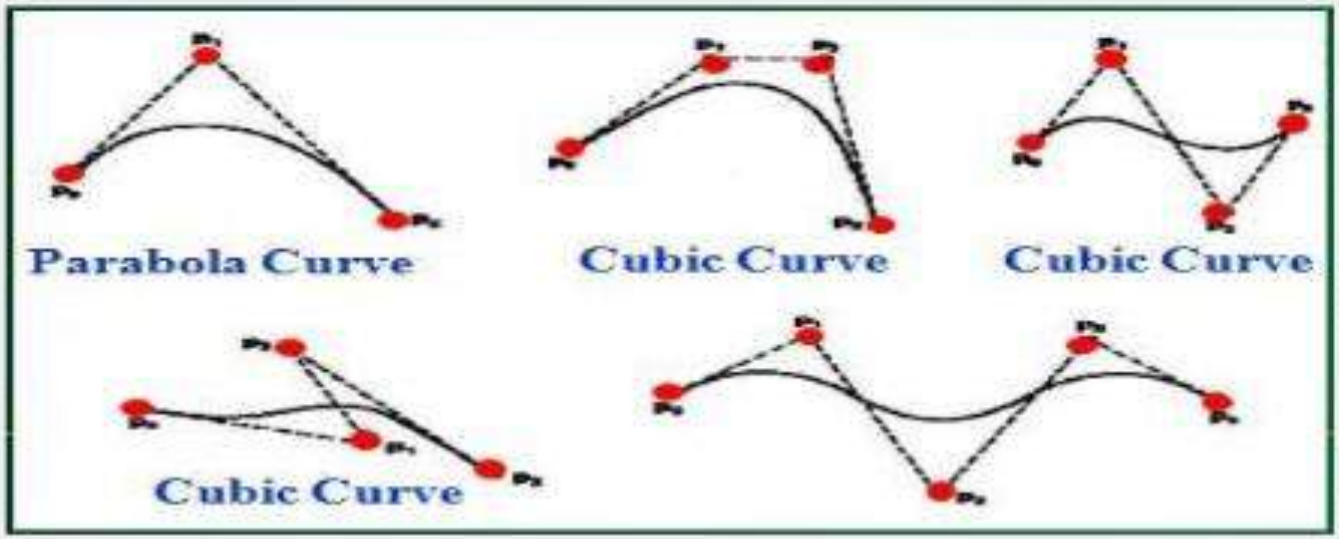
A Bezier curve is a mathematically defined curve used in two-dimensional graphic applications. The curve is defined by four points: the initial position and the terminating position (which are called "anchors") and two separate middle points (which are called "handles"). The shape of a Bezier curve can be altered by moving the handles. The mathematical method for drawing curves was created by Pierre Bezier in the late 1960's for the manufacturing of automobiles at Renault.



Properties

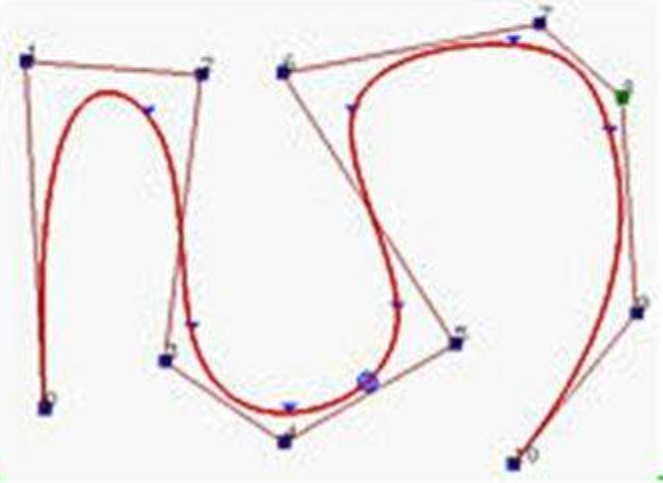
1.The degree of a Bézier curve defined b

$$C(u) = \sum_{k=0}^{n} \mathbf{p}_k B_{k,n}(u), \quad 0 \le u \le 1$$



B-Splines (for basis splines)

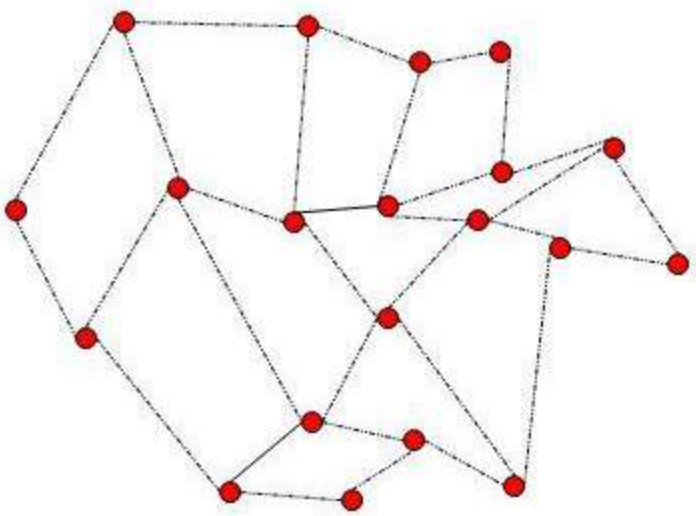
- B-Splines
  - Another polynomial curve for modelling curves and surfaces
  - Consists of curve segments whose polynomial coefficients only depend on just a few control points
    - Local control
  - Segments joined at *knots*

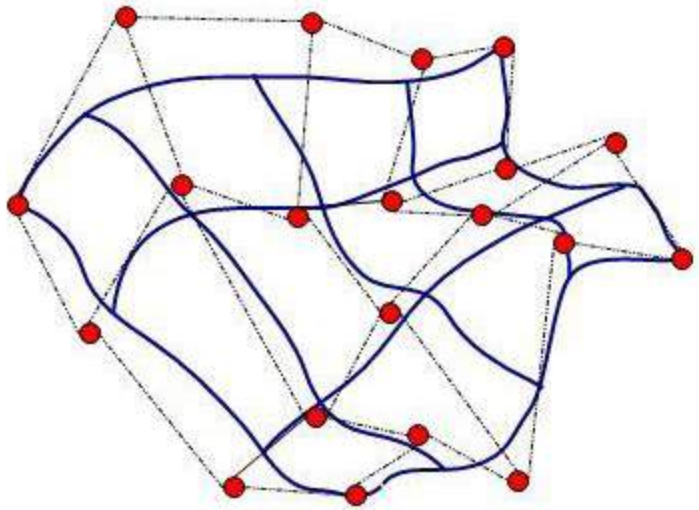


Bézier Surfaces

- Two sets of orthogonal Bézier curves are used.
- Cartesian product of Bézier blending functions:

$$\mathbf{P}(u,v) = \sum_{j=0}^{m}\sum_{k=0}^{n}\mathbf{p}_{j,k}\mathrm{BEZ}_{j,m}(v)\mathrm{BEZ}_{k,n}(u) \qquad 0 \le u,v \le 1$$





Bezier Curve and Surfaces

- This spline approximation method developed by the French engineer Pierre Bezier for use in the design of Renault automobile bodies.
- Easy to implement.
- Available in CAD system, graphic package, drawing and painting packages.

Bezier Curve

- A Bezier curve can be fitted to any number of control points.
- Given $n+1$ control points position

$$p_k=(x_k, y_k, z_k) \qquad 0\leq k\leq n$$

Illumination model

Figure 13 shows the specular reflection direction at a point on the illuminated surface.

In this figure,

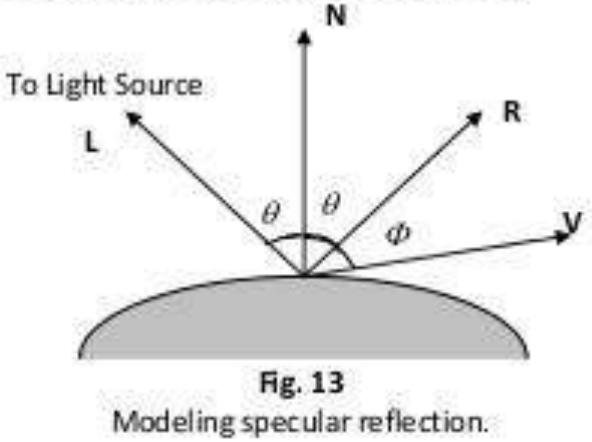


Fig. 13
Modeling specular reflection.

- **R** represents the unit vector in the direction of specular reflection;
- **L** – unit vector directed toward the point light source;
- **V** – unit vector pointing to the viewer from the surface position.
- Angle Φ is the viewing angle relative to the specular-reflection direction **R**.
- Viewing direction can be any where in 3D plane.

Basic Illumination Model

- Discussion about how object will illuminate?

An object is illuminated from the ambient light and from interrelated light source.

- The important components are:
  - Ambient light (light coming from the nearby objects after reflection).
  - Diffuse Illumination
  - Specular reflection

  Object Illuminates on the basis on following properties:
  1. Intensity of Ambient Light
  2. Type of Object surface
  3. Surface color

  Points to Remember:

  In illumination & shading we try to develop models which are close approximation of real world objects.

  Models developed with the inclusion of illumination and shading concept should not be true virtual but it should be very close approximation of realistic scene.

POLYGON RENDERING METHODS

- **RENDERING** means giving proper intensity at each point in a graphical object to make it look like real world object.
- Different Rendering Methods are
  - Constant Intensity Shading.
  - Gouraud Shading.
  - Phong Shading.
  - Fast Phong Shading.

Polygon Rendering Methods

- Surface representations:
  Usually polygon mesh approximations.
  - Simpler calculations
  - HW support
- Curved surfaces versus polyhedra:
  - If smooth surface is desired use interpolation

  - If you want to display polyhedra use constant-intensity surface rendering

Polygon Rendering Methods

- Constant intensity rendering:
  - No interpolation.
  - Also called "flat shading"

- Intensity interpolation rendering:
  - "Gouraud shading"
  - Based on intensity interpolation of vertices.

- Normal vector interpolation rendering:
  - "Phong shading"
  - Based on interpolation of normal vectors of vertices.

Unit-4

3D TRANSFORMATION

- When the transformation takes place on a 3D plane .it is called 3D transformation.
- Generalize from 2D by including **z** coordinate

Straight forward for translation and scale, rotation more difficult

Homogeneous coordinates: 4 components

Transformation matrices: 4×4 elements

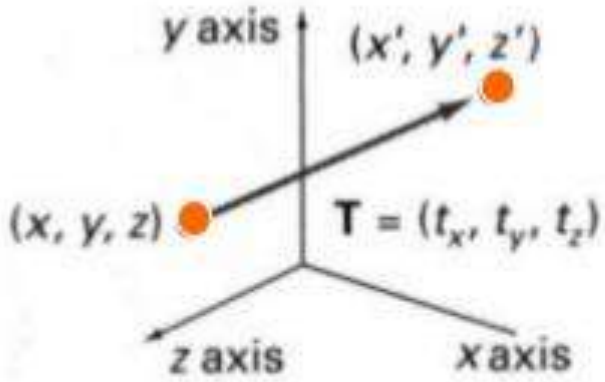
$$\begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D TRANSLATION

- Moving of object is called translation.
- In 3 dimensional homogeneous coordinate representation , a point is transformed from position P = (x, y , z) to P'=(x', y', z')
- This can be written as:-

Using **P' = T . P**

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

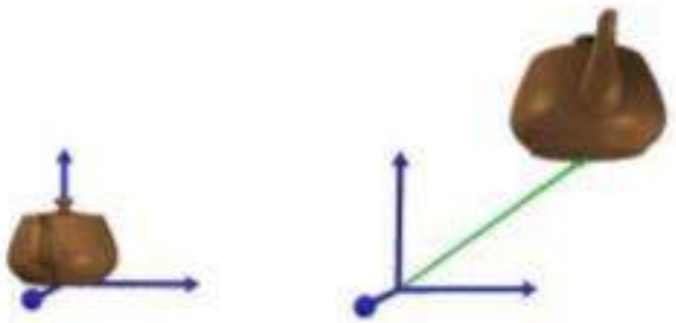


3D TRANSLATION

- The matrix representation is equivalent to the three equation.

$$x' = x + t_x, \quad y' = y + t_y, \quad z' = z + t_z$$

Where parameter t_x, t_y, t_z are specifying translation distance for the coordinate direction x, y, z are assigned any real value.

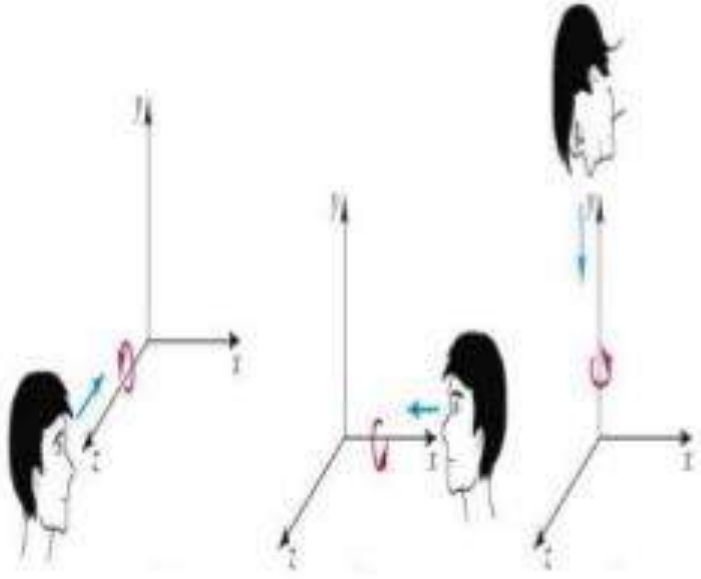


3D ROTATION

Where an object is to be rotated about an axis that is parallel to one of the coordinate axis, we can obtain the desired rotation with the following transformation sequence.

Coordinate axis rotation

Z- axis Rotation(Roll)
Y-axis Rotation(Yaw)
X-axis Rotation(Pitch)



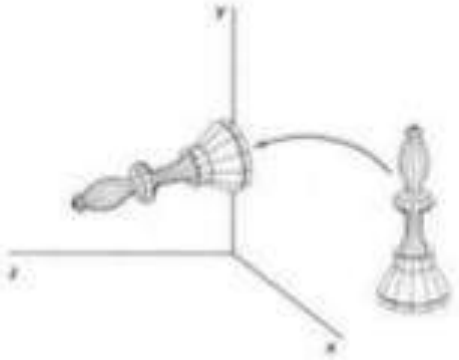
X-AXIS ROTATION

The equation for X-axis rotation

$\mathbf{x' = x}$

$\mathbf{y' = y \cos\theta - z \sin\theta}$

$\mathbf{z' = y \sin\theta + z \cos\theta}$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D Scaling

About any fixed point:

Scaling with respect to an arbitrary fixed point is not as simple as scaling with respect to the origin .

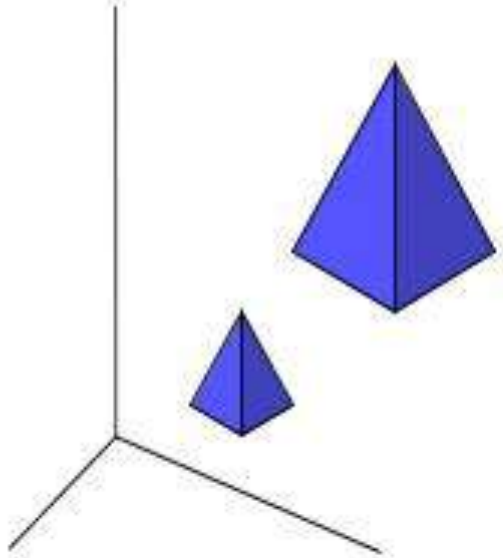
The procedure of scaling with respect to an arbitrary fixed point is:

- Translate the object so that the fixed point coincides with the origin.
- Scale the object with respect to the origin.
- Use the inverse translation of step 1 to return the objects to its original position.

3D SCALING

- Changes the size of the object and repositions the object relative to the coordinate origin.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



3D REFLECTION

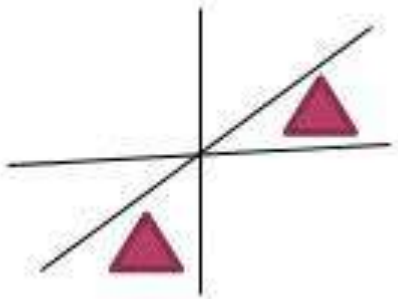
- The matrix for reflection about y-axis:-

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Reflection about z-axis:-

x'=-x y'=-y z'=z

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3D REFLECTION

- Reflection in computer graphics is used to emulate reflective objects like mirrors and shiny surfaces
- Reflection may be an x-axis

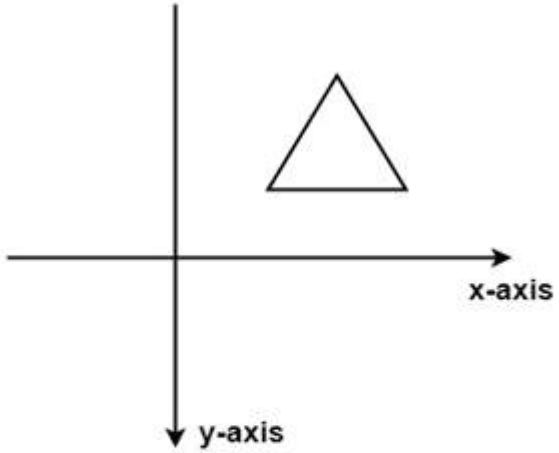
y-axis , z-axis. and also in
the planes xy-plane,yz-plane , and
zx-plane.
Reflection relative to a given
Axis are equivalent to 180
Degree rotations

Composite Transformation

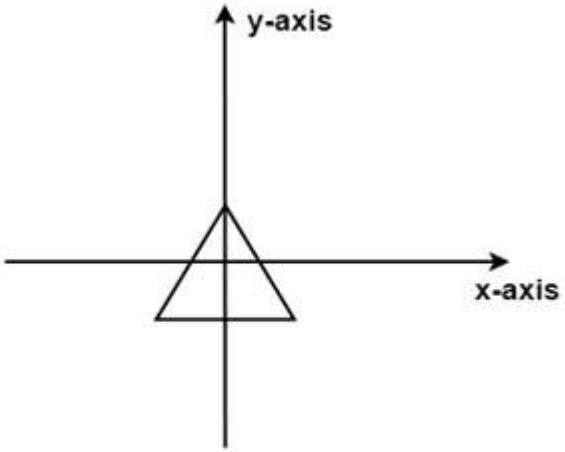
More complex geometric & coordinate transformations can be built from the basic transformation by using the process of composition of function.

Example: **Scaling about a fixed point.**

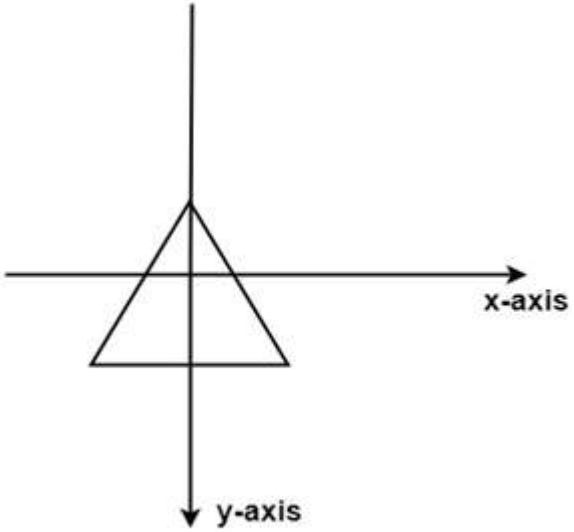
Transformation sequence to produce scaling w.r.t a selected fixed position (h, k) using a scaling function that can only scale relative to the coordinate origin are:-



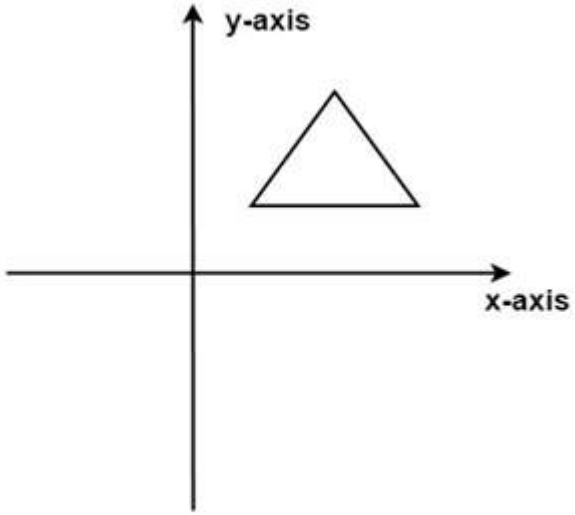
(original position of object)
Fig:(a)



(object is translated to origin)
Fig:(b)



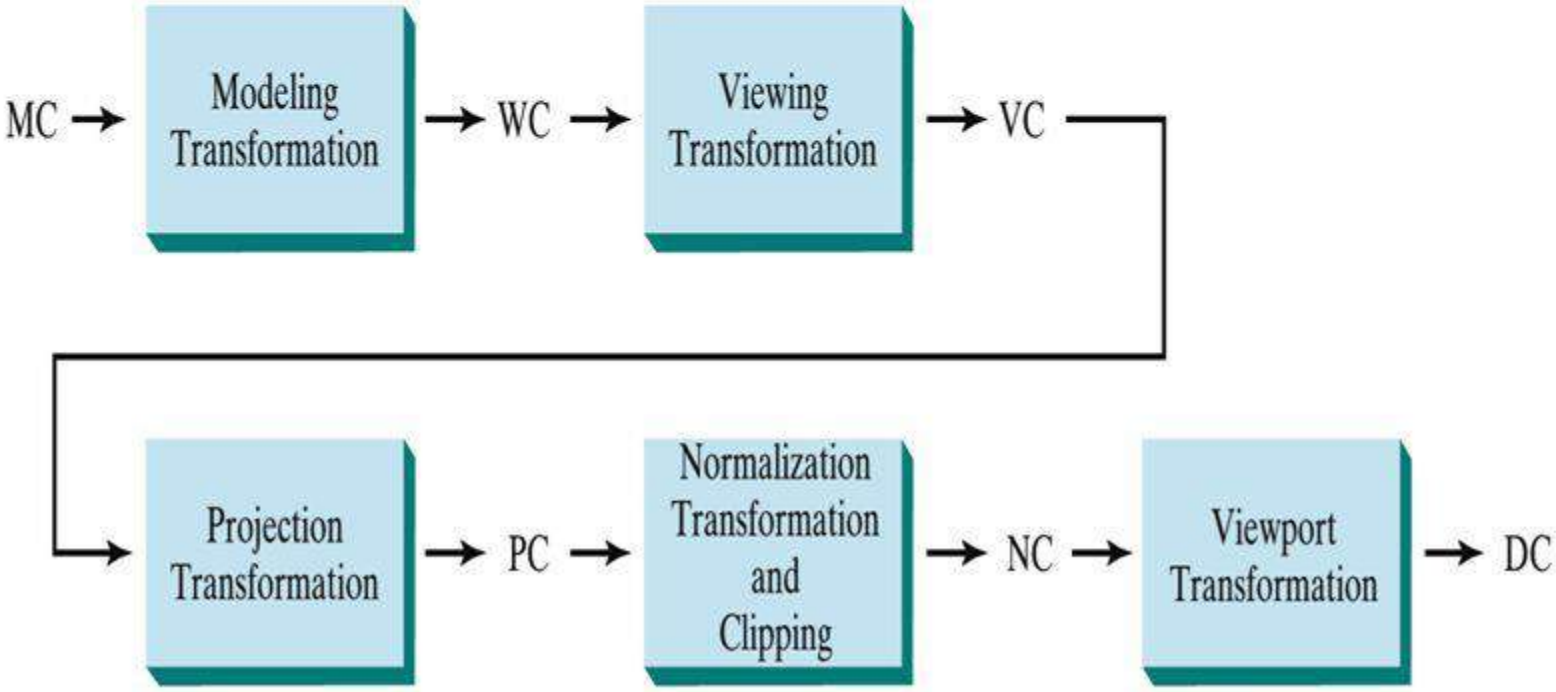
(object is enlarged)
Fig:(c)



(object is retranslated to original position)
Fig:(d)

Viewing Pipeline

- The general processing steps for modeling and converting a world coordinate description of a scene to device coordinates:



3D Viewing-Coordinate Parameters

- Select
  - View point (or viewing position, eye position, camera position) $P_0=(x_0,y_0,z_0)$
  - View-up vector V to define y_{view}
  - Direction to define z_{view}

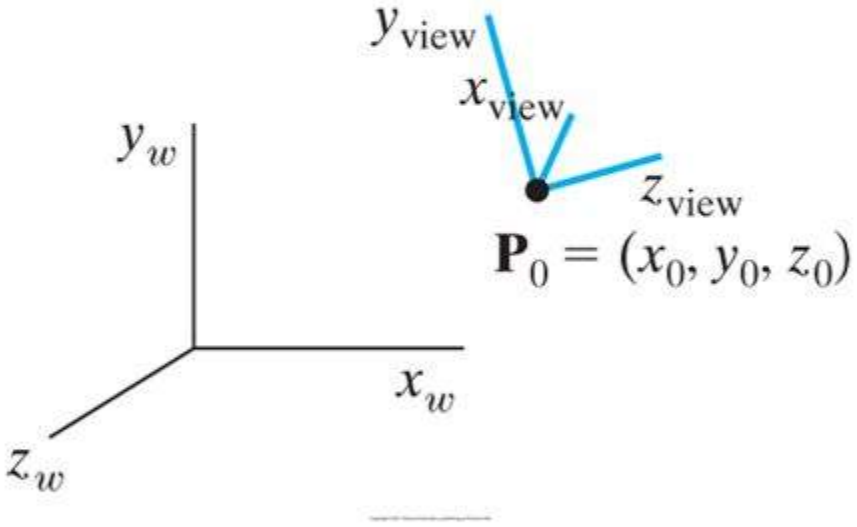
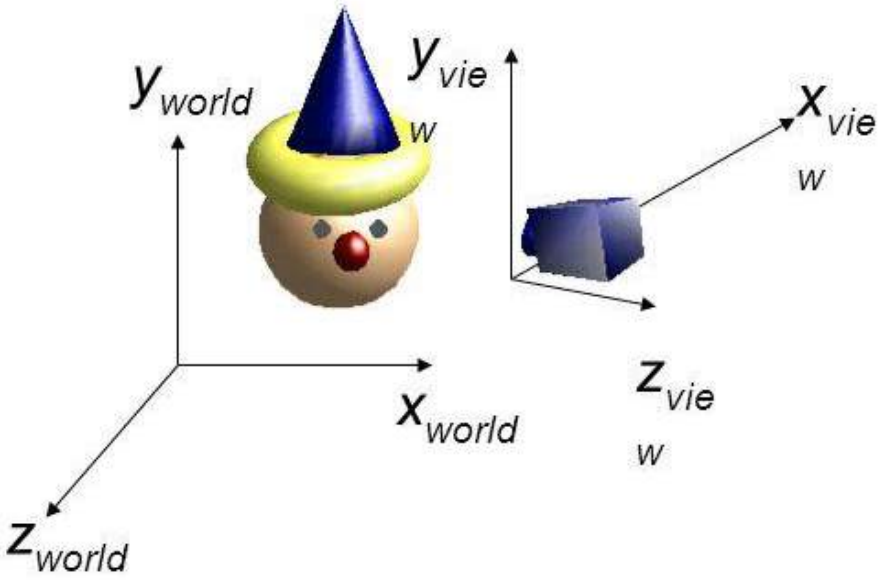


Figure 10-7 A right-handed viewing-coordinate system, with axes x_{view}, y_{view}, and z_{view}, relative to a right-handed world-coordinate frame.

3D Viewing-Coordinate Parameters

- World coordinates to Viewing coordinates: Viewing transformations

World coordinates



Viewing coordinates:
Viewers (Camera) position and viewing plane.

View volume

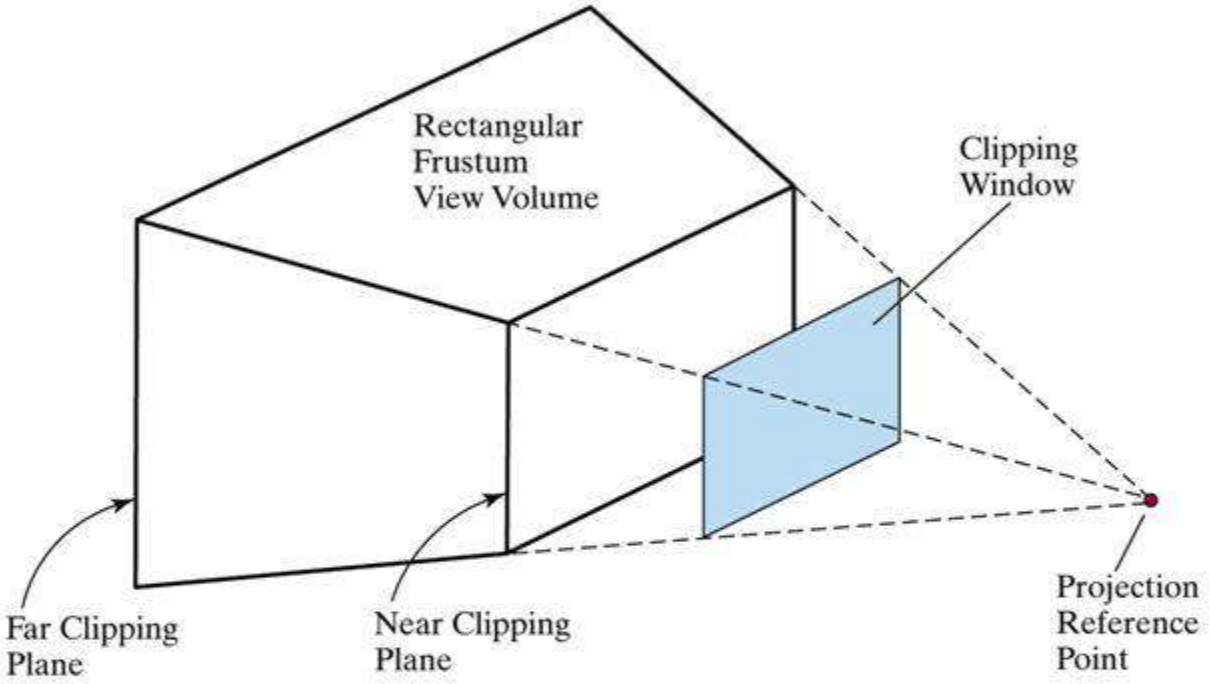
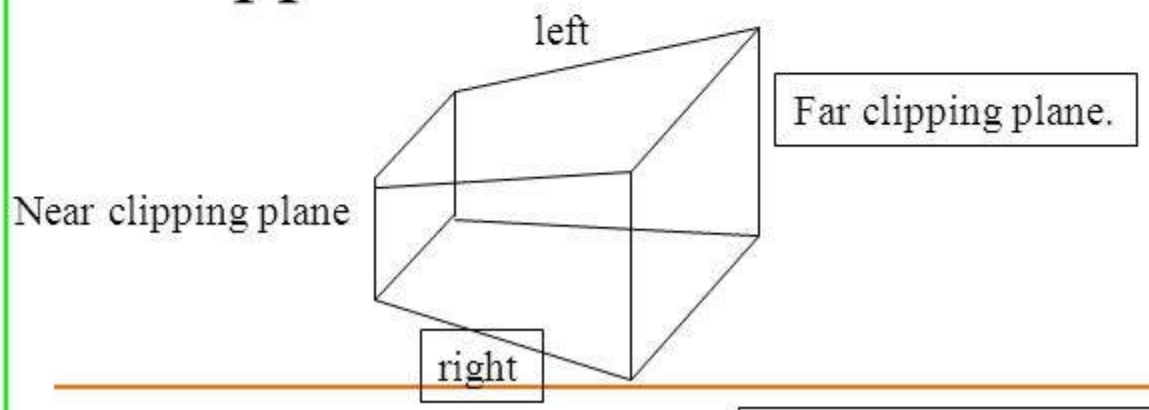


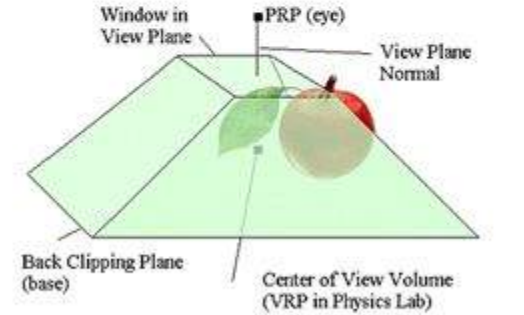
Figure 7-46

A perspective-projection frustum view volume with the view plane "in front" of the near clipping plane.

3D View Volume

- The volume in which the visible objects exist
- For orthographic projection, view volume is a box.
- For perspective projection, view volume is a *frustum*.
- The surfaces outside the view volume must be clipped





Example of view volume with partially clipped object (apple). The top surface acts as a projection window and is mapped to a view port then to the display screen.

Need to calculate intersection With 6 planes.

Transform 3D objects on to a 2D plane using ***projections***

2 types of projections

Perspective

Parallel

In **parallel projection**, coordinate positions are transformed to the view plane along parallel lines.

In **perspective projection**, object position are transformed to the view plane along lines that converge to a point called **projection reference point (center of projection)**