APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

STUDY MATERIALS

CS 401 COMPUTER GRAPHICS

Module 3

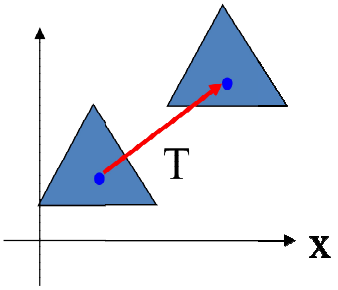
Two dimensional transformations. Homogeneous coordinate systems – matrix formulation and concatenation of transformations. Windowing concepts – two dimensional clipping.

Two dimensional transformations

Transformation means changing some graphics into something else by applying rules. Changes in orientation, size, and shape are accomplished with geometric transformations that alter the coordinate descriptions of objects. Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation. The basic geometric transformations are translation, rotation, and scaling. Other transformations that are often applied to objects include reflection and shear.

1. Translation

A translation moves an object to a different position on the screen. A translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another We can translate a point in 2D by adding translation coordinate (tx, ty) to the original coordinate (X, Y) to get the new coordinate (X', Y').



From the above figure, you can write that –

X' = X + tx

Y' = Y + ty

The pair (tx, ty) is called the translation vector or shift vector.

The above equations can also be represented using the column vectors.

P' = P + T

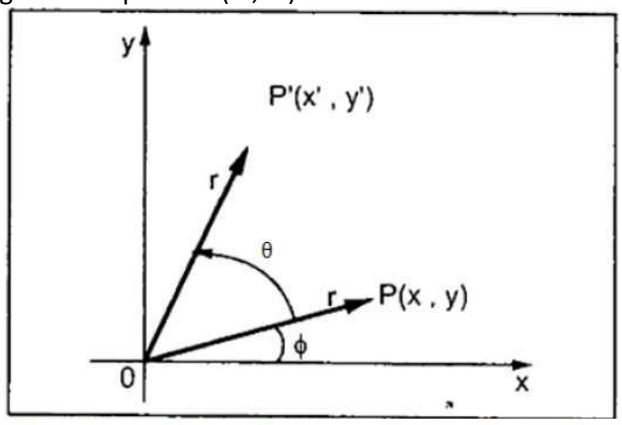
$$P = \begin{bmatrix} X \\ Y \end{bmatrix} \quad P' = \begin{bmatrix} X' \\ Y' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

NB: Refer class note for worked out example.

2.Rotation

In rotation, we rotate the object at particular angle θ (theta) from its origin. From the following figure, we can see that the point P(X, Y) is located at angle ϕ from the horizontal X coordinate with distance r from the origin.

Let us suppose you want to rotate it at the angle θ. After rotating it to a new location, you will get a new point P' (X', Y').



Using standard trigonometric the original coordinate of point P(X, Y) can be represented as −

$$X = r\cos\phi \ldots\ldots (1)$$

$$Y = r\sin\phi \ldots\ldots (2)$$

Same way we can represent the point P' (X', Y') as −

$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \ldots\ldots (3)$$

$$y' = r\sin(\phi + \theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta \ldots\ldots (4)$$

Substituting equation (1) & (2) in (3) & (4) respectively, we will get

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

Representing the above equation in matrix form,

$$[X'Y'] = [XY]\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} OR$$

$$P' = P \cdot R$$

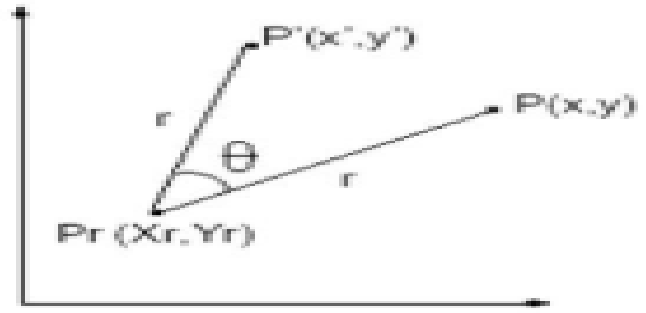
Where R is the rotation matrix

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

The rotation angle can be positive and negative.

For clock wise rotation angle is taken as negative and for anticlockwise direction angle is taken as positive.

Rotation about a fixed reference point (xr,yr)



$$x' = x_r + (x - x_r)\cos(\theta) - (y - y_r)\sin(\theta)$$
$$y' = y_r + (x - x_r)\sin(\theta) + (y - y_r)\cos(\theta)$$

NB: Refer class note for worked out example.

3.Scaling

To change the size of an object, scaling transformation is used. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.

Let us assume that the original coordinates are (X, Y), the scaling factors are (SX, SY), and the produced coordinates are (X', Y'). This can be mathematically represented as shown below −

X' = X . SX and Y' = Y . SY

The scaling factor SX, SY scales the object in X and Y direction respectively.

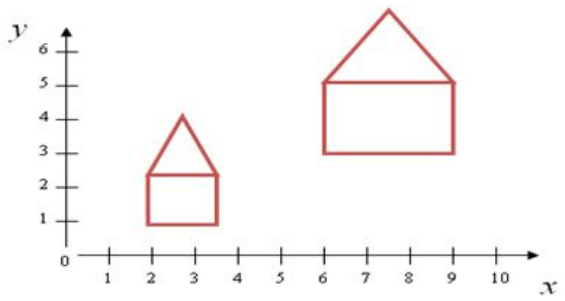
The above equations can also be represented in matrix form as below −

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

OR

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

Where S is the scaling matrix. The scaling process is shown in the following figure.

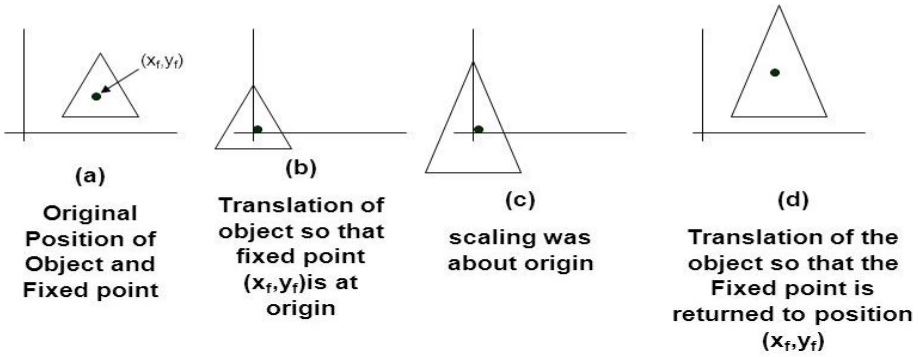


- If S_X & S_Y>1, the size of the object will increase. Also objects move away from origin
- If S_X & S_Y <1, the size of the object will decrease. Also objects move towards origin
- If S_X & S_Y =1, the size of the object will remain same. Also there will be no change in location.

There are two types of scaling with respect to values of S_X & S_Y - Differential and uniform scaling. If SX & SY have same values, it is called uniform scaling. If S_X & S_Y have different values, it is called differential scaling.

Scaling with respect to a fixed point (xf,yf)

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



$$x' = x_f + (x - x_f) \cdot s_x$$
$$y' = y_f + (y - y_f) \cdot s_y$$
$$x' = x \cdot s_x + x_f(1 - s_x)$$
$$y' = y \cdot s_y + y_f(1 - s_y)$$

NB: Refer class note for worked out example.

Homogeneous co-ordinates:

We can represent the point by 3 numbers instead of 2 numbers, which is called Homogenous Coordinate system. In this system, we can represent all the transformation equations in matrix multiplication. Any Cartesian point P(X, Y) can be converted to homogenous coordinates by P' (Xh, Yh, h).

A point (x, y) can be re-written in homogeneous coordinates as (xh, yh, h).The homogeneous parameter h is a non- zero values such that:

$$x = \frac{x_h}{h} \qquad y = \frac{y_h}{h}$$

We can then write any point (x, y) as (hx, hy, h).We can conveniently choose h = 1 so that (x, y) becomes (x, y, 1)

Advantages of using homogenous coordinate: we can perform all transformations using matrix/vector multiplications. This allows us to pre-multiply all the matricesν together.

Translation

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

Rotation

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

Scaling

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

Other transformations:

Reflection

Reflection is the mirror image of original object. In other words, we can say that it is a rotation operation with 180°. In reflection transformation, the size of the object does not change.

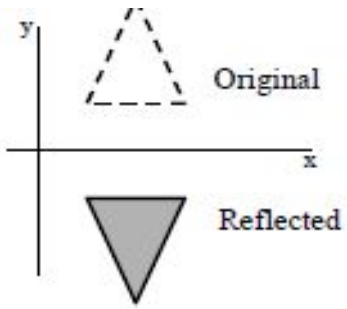
X axis reflection

x'=x

y'=-y

x-axis (y = 0)

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



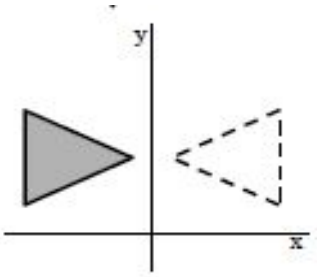
Y-axis reflection:

x'=-x

y'=y

y-axis (x = 0)

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

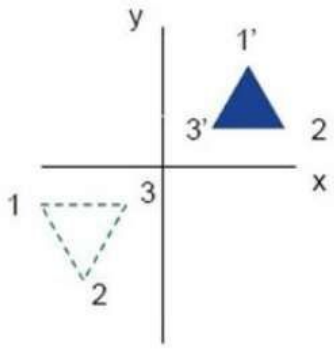


Reflection perpendicular to XY plane/ reflection with respect to origin

x'=-x

y'=-y

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

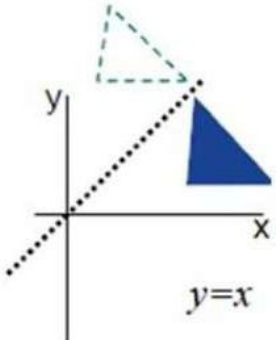


Reflection with respect to diagonal line , line y=x

x'=y

y'=x

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



Reflections about any line in xy plane can be accomplished with a combination of transformations:

1. Clock wise rotation through an angle 45 degree, which rotates line to coincide with x axis
2. Perform a reflection with respect to x axis
3. Rotate the line in anticlock wise rotation through an angle 45 degree, which moves it to original position.

NB: Refer class note for worked out example.

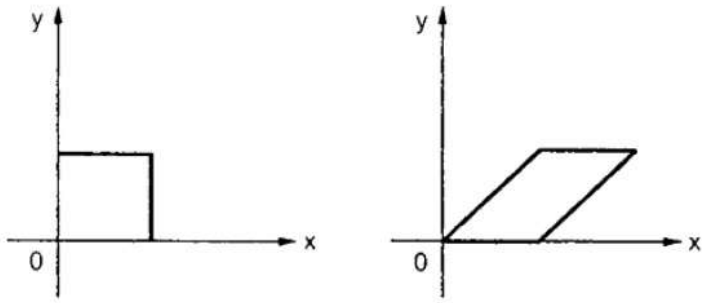
Shear

A transformation that slants the shape of an object is called the shear transformation. It 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 .

There are two shear transformations **X-Shear** and **Y-Shear**. One shifts X coordinates values and other shifts Y coordinate values. However; in both the cases only one coordinate changes its coordinates and other preserves its values. Shearing is also termed as **Skewing**.

X-Shear

The X-Shear preserves the Y coordinate and changes are made to X coordinates, which causes the vertical lines to tilt right or left as shown in below figure.



(a) Original object (b) Object after x shear

The transformation matrix for X-Shear can be represented as −

$$X' = X + Sh_x \cdot Y$$

$$Y' = Y$$

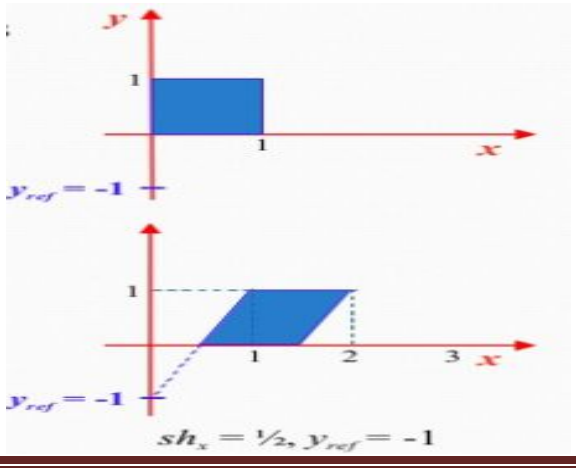
Matrix:

$$\begin{pmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

X-Shear relative to other reference line y_{ref}=-1

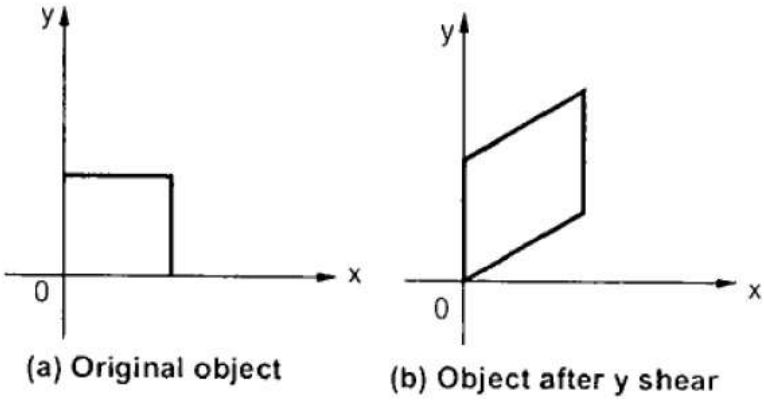
$$x' = x + sh_x(y - y_{ref})$$

$$y' = y$$



Y-Shear

The Y-Shear preserves the X coordinates and changes the Y coordinates which causes the horizontal lines to transform into lines which slopes up or down as shown in the following figure.



(a) Original object (b) Object after y shear

The Y-Shear can be represented in matrix from as −

$$\mathbf{Y' = Y + Sh_y \cdot X}$$

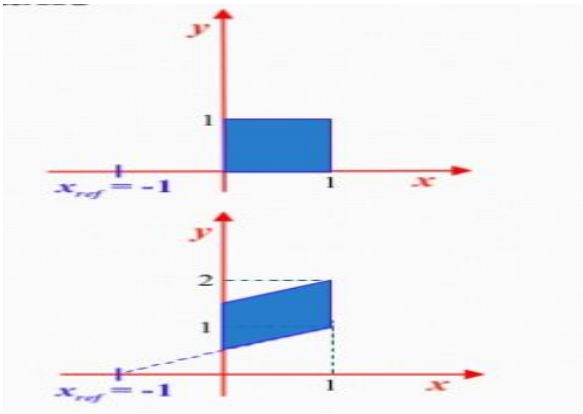
$$\mathbf{X' = X}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Y-Shear relative to other reference line x_{ref}=-1

$$x' = x$$

$$y' = y + sh_y(x - x_{ref})$$



NB: Refer class note for worked out example.

Composite transformations

We can set up a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of the individual transformations. Forming products of transformation matrices is often referred to as a concatenation, or composition, of matrices. For column-matrix representation of coordinate positions, we form composite transformations by

multiplying matrices in order from right to left. That is, each successive transformation matrix premultiplies the product of the preceding transformation matrices**.**

If a transformation of the plane T1 is followed by a second plane transformation T2, then the result itself may be represented by a single transformation T which is the composition of T1 and T2 . Transformations can be any combination of

- Translation
- Scaling
- Shearing
- Rotation
- Reflection

The change in the order of transformation would lead to different results, as in general matrix multiplication is not cumulative, that is [A] . [B] ≠ [B] . [A] and the order of multiplication. The basic purpose of composing transformations is to gain efficiency by applying a single composed transformation to a point, rather than applying a series of transformation, one after another.

Examples for composite transformation

1. **Translations:**

When two successive translations are applied to a point with translation vectors (t_{x1},t_{y1}) and (t_{x2},t_{y2}) , then

$$\begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x_1} + t_{x_2} \\ 0 & 1 & t_{y_1} + t_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{T}(t_{x_2}, t_{y_2}) \cdot \mathbf{T}(t_{x_1}, t_{y_1}) \cdot \mathbf{P} = \mathbf{T}(t_{x_1} + t_{x_2}, t_{y_1} + t_{y_2}) \cdot \mathbf{P}$$

This demonstrates that successive translations are additive in nature.

2. **Rotations:**

When two successive rotations are applied to a point ,

we can write rotation matrix $R(\theta_1)$ as

$$R(\theta_1) = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \text{ and } R(\theta_2) = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix}$$

$$R(\theta_1) \cdot R(\theta_2) = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \times \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1 \cdot \cos\theta_2 + \sin\theta_1 \cdot (-\sin\theta_2) & \cos\theta_1 \cdot \sin\theta_2 + \sin\theta_1 \cdot \cos\theta_2 \\ -\sin\theta_1 \cdot \cos\theta_2 + \cos\theta_1 \cdot (-\sin\theta_2) & -\sin\theta_1 \cdot \sin\theta_2 + \cos\theta_1 \cdot \cos\theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1 + \theta_2) & \sin(\theta_1 + \theta_2) \\ -\sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{bmatrix}$$

since,

$$\cos(\theta_1 + \theta_2) = \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2$$
$$\sin(\theta_1 + \theta_2) = \cos\theta_1 \sin\theta_2 + \sin\theta_1 \cos\theta_2$$

This demonstrates that successive rotations are additive in nature.

3. Scalings

When two successive rotations are applied to a point , with vectors (S_{x1},S_{y1}) and (S_{x2},S_{y2})

$$\begin{vmatrix} S_{x2} & 0 & 0 \\ 0 & S_{y2} & 0 \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} S_{x1} & 0 & 0 \\ 0 & S_{y1} & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} S_{x1} \cdot S_{x2} & 0 & 0 \\ 0 & S_{y1} \cdot S_{y2} & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$\mathbf{S(S_{x2}, S_{y2}) \cdot S(S_{x1}, S_{y1}) = S(S_{x1} \cdot S_{x2}, S_{y1} \cdot S_{y2})}$$

This demonstrates that successive scalings are multiplicative in nature.

4. **Rotation with respect to a fixed reference point**

5. Scaling with respect to a fixed reference point

6. Reflection with respect to a line y=x.

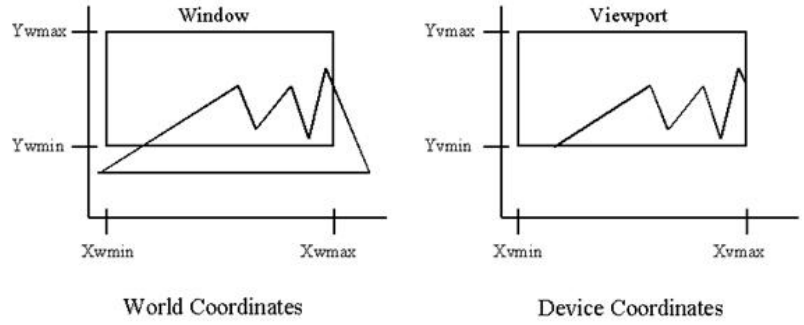
Hint: For points 4,5 and 6 refer notes given under basic transformations. Write the corresponding matrix for individual transformations. Show the composite transformation matrix as product of individual matrices in reverse order.

Affine transformation

Affine transformation is a linear mapping method that preserves points, straight lines, and planes. Sets of parallel lines remain parallel after an affine transformation. An affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line. Examples of affine transformations include scaling ,translation, reflection, shearing and rotation.

Affine transformations have the general properties that parallel lines are transformed into parallel lines and finite points map to finite points.An affine transformation involving only rotation, translation, and reflection preserves angles and lengths, as well as parallel lines. For these three transformations, the lengths and angle between two lines remains the same after the transformation.

2D viewing pipeline



Window: Window is world coordinate area selected for display. Window is area of a picture selected for viewing .Window defines what is to be viewed.

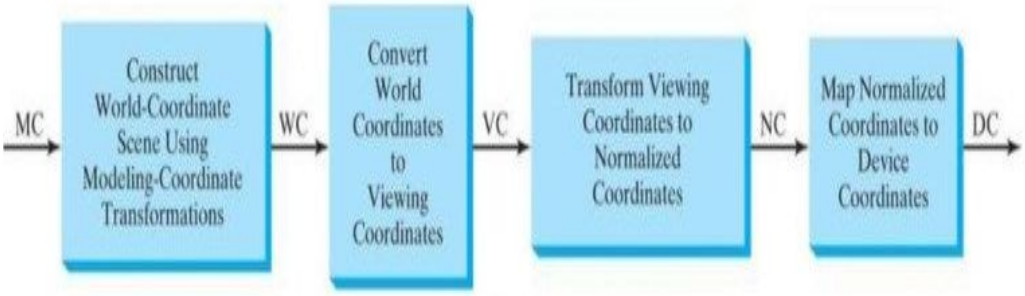
View port: An area on the display device to which a window is mapped is called view port. View port defines where is to be displayed.

Viewing transformation.

- Mapping of a part of world coordinate scene to device coordinate is called viewing transformation.
- It is to specify the view that is to be presented and the portion of the output display area that is to be used.
- Sometimes the two-dimensional viewing transformation is simply referred to as the *window-to-viewport transformation* or the *windowing transformation*

Steps in 2D viewing pipeline

The term Viewing Pipeline describes a series of transformations, which are passed by geometry data to end up as image data being displayed on a device.



The coordinates in which individual objects (models) are created are called model (or object) coordinates (MC). When several objects are assembled into a scene, they are described by world coordinates(WC). After transformation into the coordinate system of the camera (viewer) they become viewing coordinates(VC). Their projection onto a common plane (window) yields device-independent normalized coordinates (NC). Finally, after mapping those normalized coordinates to a specific device, we get device coordinates(DC).

Window to viewport transformation

- Mapping of a part of world coordinate scene to device coordinate is called viewing transformation.
- It is to specify the view that is to be presented and the portion of the output display area that is to be used.
- 2D viewing transformation is also called as window to view port transformation or windowing transformation
- If a coordinate position is at the centre of window, it will be displayed at the centre of view port.
- Relative positions in the two areas should be same to maintain the relative placement.

$$(U-U_{min})/(U_{max}-U_{min})=(X-X_{min})/(X_{max}-X_{min})$$

$$(V-V_{min})/(V_{max}-V_{min})=(Y-Y_{min})/(Y_{max}-Y_{min})$$

$$U= U_{min} + (X-X_{min})S_x$$

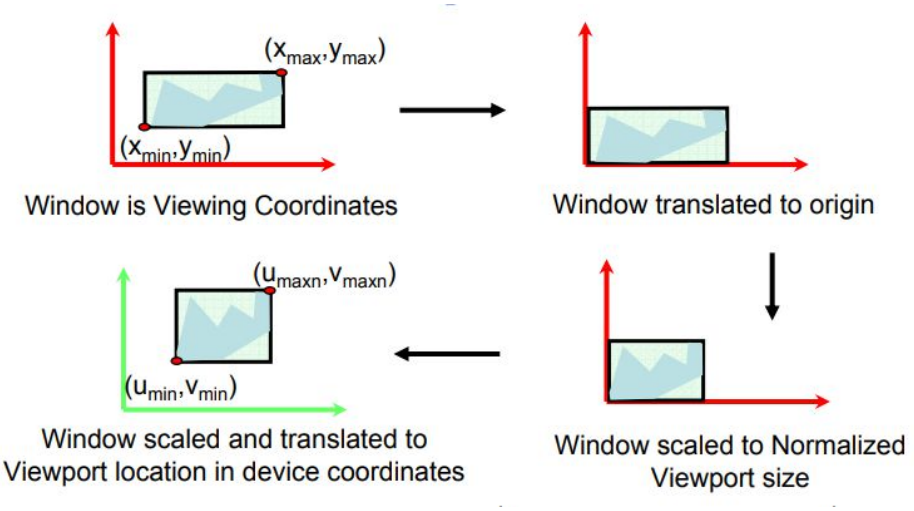
$$V= V_{min} + (Y-Y_{min})S_y$$

Where S_x and S_y are scaling factors.

$$S_x = (U_{max}-U_{min}) /(X_{max}-X_{min})$$

$$S_y =(V_{max}-V_{min}) /(Y_{max}-Y_{min})$$

Sequence of Transformations required: Scaling transformation that scales window area to size of view port. Translate the scaled window area to the position of view port.



$$T(-x_{min}, -y_{min}) \qquad S\left(\frac{u_{max}-u_{min}}{x_{max}-x_{min}}, \frac{v_{max}-v_{min}}{y_{max}-y_{min}}\right) \qquad T(u_{min}, v_{min})$$

$$M_{wv} = T(u_{min}, v_{min}).S\left(\frac{u_{max}-u_{min}}{x_{max}-x_{min}}, \frac{v_{max}-v_{min}}{y_{max}-y_{min}}\right).T(-x_{min}, -y_{min})$$

$$= \begin{bmatrix} 1 & 0 & u_{min} \\ 0 & 1 & v_{min} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{u_{max}-u_{min}}{x_{max}-x_{min}} & 0 & 0 \\ 0 & \frac{v_{max}-v_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_{min} \\ 0 & 1 & -y_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

Hint: Write the final matrix.

To get the point in viewport, multiply point (x,y) with calculated matrix.Thus new point (U,V) is calculated as

$U = U_{min} + (X - X_{min})S_x$

$V = V_{min} + (Y - Y_{min})S_y$

Where S_x and S_y are scaling factors.

$S_x = (U_{max} - U_{min}) / (X_{max} - X_{min})$

$S_y = (V_{max} - V_{min}) / (Y_{max} - Y_{min})$

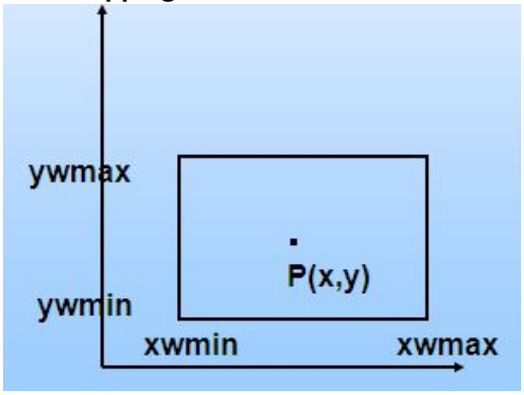
Clipping operations

Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a **clipping algo**rithm,or simply clipping. The region against which an object is to clipped is called a clip **window.**

Applications of clipplng include extracting part of a detailed scene for viewing; identifying visible surfaces in three-dimensiona1 views; antialiasing line ***seg***ments or object boundaries; creating objects using solid-modeling procedures;displaying a multiwindow environment; and drawing and painting operations that allow parts of a picture to be selected for copying, moving, erasing, or duplicating.

The primary use of clipping in computer graphics is to remove objects, lines, or line segments that are outside the viewing pane. The viewing transformation is insensitive to the position of points relative to the viewing volume − especially those points behind the viewer − and it is necessary to remove these points before generating the view.

Point Clipping



In case of point clipping, we only show points on our window which are in range of our viewing pane, others points which are outside the range are discarded.
So display Point (x,y) if it satisfies two conditions: xwmin <= x <= xwmax and ywmin <= y<= ywmax

Algorithm:

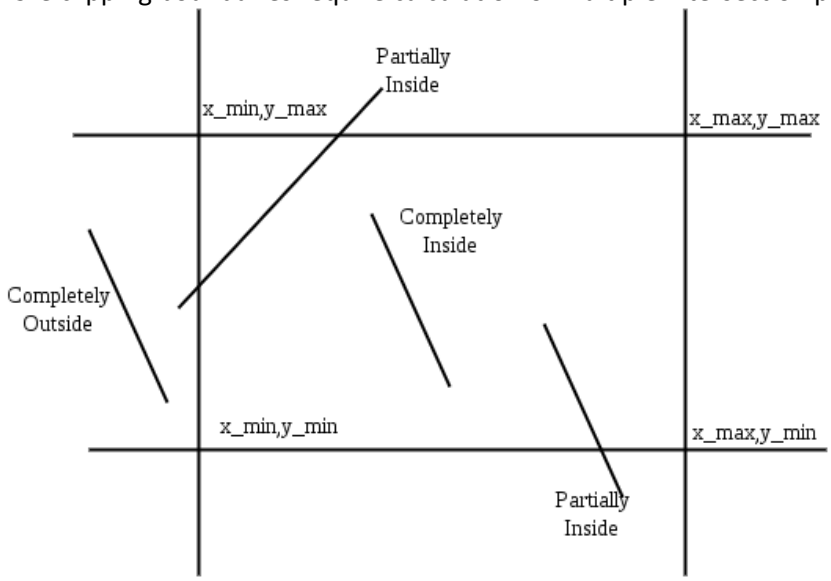
1. Get the minimum and maximum coordinates of both viewing window.
2. Get the coordinates for a point.
3. Check whether given input lies between minimum and maximum coordinate of viewing window
4. If yes display the point which lies inside the region otherwise discard it.

Line Clipping

The concept of line clipping is same as point clipping. In line clipping, we will cut the portion of line which is outside of window and keep only the portion that is inside the window.

Clipping using parametric representation of lines

A line clipping procedure involves several parts. First, we can test a given line segment to determine whether it lies completely inside the clipping window. If it does not, we try to determine whether it lies completely outside the window. Finally, if we cannot identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries. We process lines through the "inside-outside'' tests by checking the line endpoints. All lines which cross one or more clipping boundaries require calculation of multiple intersection points.



For a line segment with endpoints (x1,y1) & (x2,y2) and one or both endpoints outside the clipping rectangle, the parametric representation could be used to determine values at intersection point with clipping boundary coordinates.

x=x1+u (x2-x1)

y=y1+u (y2-y1) where 0<=u<=1

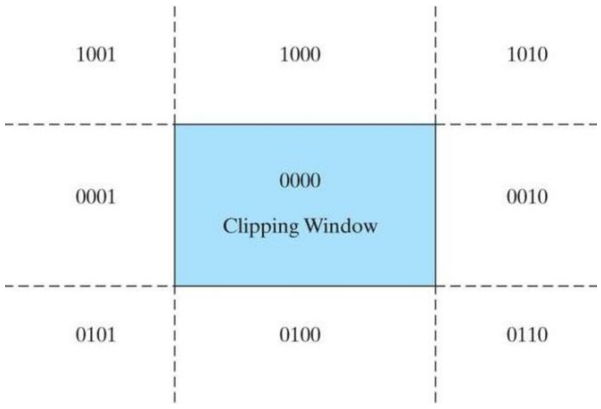
If the value of u for an intersection with a rectangle boundary edge is outside the range 0 to 1, the line does not enter the interior of the window at that boundary. If the value ol u is within the range from 0 to 1, the line segment does indeed cross into the clipping area. This method can be applied to each clipping boundary edge in turn to determine whether any part of the line segment is to be displayed. Line segments that are parallel to window edges can be handled as special cases.

Disadvantage: Required great deal of computations and not efficient when compared to other methods.

NB: Refer class note for worked out example.

Cohen-Sutherland Line Clipping

This is one of the oldest and most popular line-clipping procedures. Generally, the method speeds up the processing of line segments by performing initial tests that reduce the number of intersections that must be calculated. Every line end point in a picture is assigned a four-digit binary code, called a region code, that identifies the location of the point relative to the boundaries of the clipping rectangle. Regions are set up in reference to the boundaries as shown in Figure



Each bit position in the region code is used to indicate one of the four relative coordinate positions of the point with respect to the clip window: to the left, right,top, or bottom. By numbering the bit positions in the region code, the coordinate regions can be correlated with the bit positions as
bit 1: left
bit 2: right
bit 3: below
bit 4: above

A value of 1 in any bit position indicates that the point is in that relative position;otherwise, the bit position is set to 0. If a point is within the clipping rectangle,the region code is 0000. A point that is below and to the left of the rectangle has a region code of 0101.

There are 3 possibilities for the line –

- Line can be completely inside the window (This line should be accepted).
- Line can be completely outside of the window (This line will be completely removed from the region).
- Line can be partially inside the window (We will find intersection point and draw only that portion of line that is inside region).

Algorithm

Step 1 – Assign a region code for each endpoints.
Step 2 – If both endpoints have a region code **0000** then accept this line.
Step 3 – Else, perform the logical **AND** operation for both region codes.
Step 3.1 – If the result is not **0000,** then reject the line.
Step 3.2 – Else you need clipping.
Step 3.2.1 – Choose an endpoint of the line that is outside the window.
Step 3.2.2 – Find the intersection point at the window boundary using set of equations given below.(based on region code and window boundary).

Intersections with a vertical boundary:
y=y1+m(x-x1) (x is either xmin or xmax)
Intersections with a horizontal boundary:
x=x1+(y-y1)/m (y is either ymin or ymax)

Step 3.2.3 – Replace endpoint with the intersection point and update the region code.
Step 4 – Repeat step 1 for other lines.

NB: Refer class note for worked out example.

Liang Barsky Line clipping

The Liang–Barsky algorithm uses the parametric equation of a line and inequalities describing the range of the clipping window to determine the intersections between the line and the clip window. With these intersections it knows which portion of the line should be drawn. This algorithm is significantly more efficient than Cohen–Sutherland. The idea of the Liang–Barsky clipping algorithm is to do as much testing as possible before computing line intersections.

Consider first the usual parametric form of a straight line:

$x = x1 + u\,\Delta x$

$y = y1 + u\,\Delta y, \qquad 0 <= u <= 1$

$\Delta x = x2 - x1$ & $\Delta y = y2 - y1$

Interior of window defined as:

$$xw_{min} <= x <= xw_{max}$$
$$Yw_{min} <= y <= yw_{max}$$

Substituting parametric lines x & y; we get

$$xw_{min} <= x1 + u\Delta x <= xw_{max}$$
$$yw_{min} <= y1 + u\Delta y <= yw_{max}$$

In general equations can be written as:

$$u\,p_k \le q_k, \qquad k = 1, 2, 3, 4$$

NB:Taken point (x1,y1) as (x0,y0) in equations given below

$u_k = \dfrac{q_k}{p_k}$ where

$p_1 = -\Delta x$	$q_1 = x_0 - xw_{min}$	Left boundary
$p_2 = \Delta x$	$q_2 = xw_{max} - x_0$	Right boundary
$p_3 = -\Delta y$	$q_3 = y_0 - yw_{min}$	Bottom boundary
$p_4 = \Delta y$	$q_4 = yw_{max} - y_0$	Top boundary

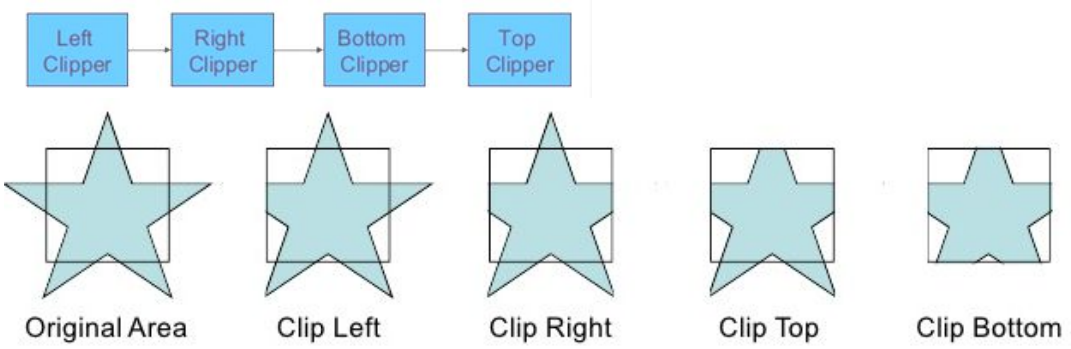
- If p_i=0, line is parallel to i^{th} edge of window.i.e if p_1=0,line is parallel to left edge.
- If q_i<0, it lies outside i^{th} boundary side. i.e If q_2<0, it lies outside right edge.
- If both conditions prevail, i.e p_i=0 and q_i<0, the line is completely outside clipping window.
  i.e p_1=0 and q_1<0, the line is parallel to left edge and it is completely outside left edge, so line is completely outside clipping window. So we can reject the line.
- If q_i=0,it is on i^{th} edge of window boundary.
- If q_i>=0, it lies inside visible area.
- The intersection point can be calculated by
  $u = q_i / p_i$
  Using this u value in parametric line equation, we can find out intersection point

NB: Refer class note for worked out example.

Sutherland Hodgeman polygon clipping algorithm

A polygon can also be clipped by specifying the clipping window. Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping. In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.

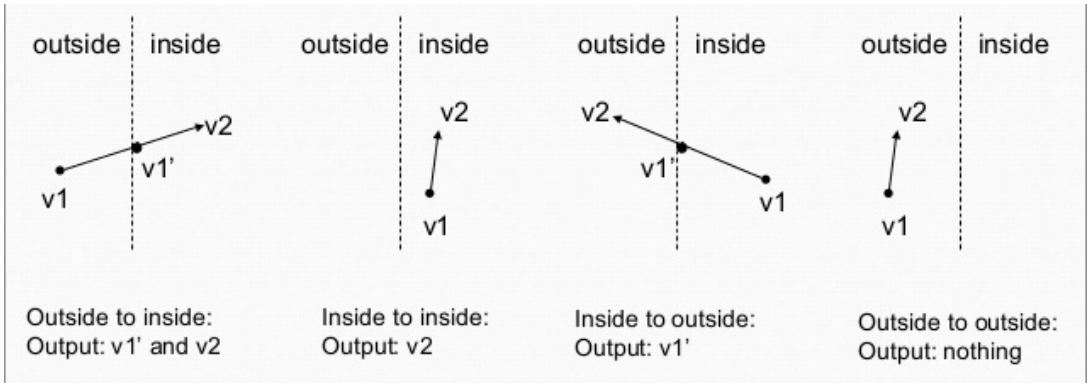
First the polygon is clipped against the left edge of the polygon window to get new vertices of the polygon. These new vertices are used to clip the polygon against right edge, top edge, bottom edge, of the clipping window as shown in the following figure. While processing an edge of a polygon with clipping window, an intersection point is found if edge is not completely inside clipping window and the a partial edge from the intersection point to the outside edge is clipped. The following figures show left, right, top and bottom edge clippings.



Beginning with the initial set of polygon vertices, we first clip the polygon against left boundary to produce new sequence of vertices.the new set of vertices is then successively passed to a right boundary , a bottom boundary, and a top boundary clipper.At each step, a new set of vertices is generated & passed to the next window clipper.

There are 4 possible cases when processing vertices in sequence around the polygon edges.For each pair of vertices , we make the following tests

1. **Both vertices are inside :** Only the second vertex is added to the output list
2. **First vertex is outside while second one is inside :** Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list
3. **First vertex is inside while second one is outside :** Only the point of intersection of the edge with the clip boundary is added to the output list
4. **Both vertices are outside :** No vertices are added to the output list



Example:

We illustrate this algorithm by processing the area in figure against the left window boundary.

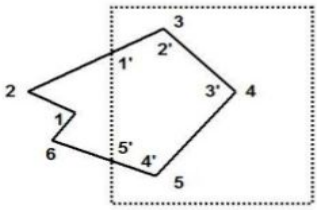
Vertices 1 and 2 are outside of the boundary.

Vertex 3, which is inside, 1' and vertex 3 are saved.

Vertex 4 and 5 are inside, and they also saved.

Vertex 6 is outside, 5' is saved.

Using the five saved points, we would repeat the process for the next window boundary.



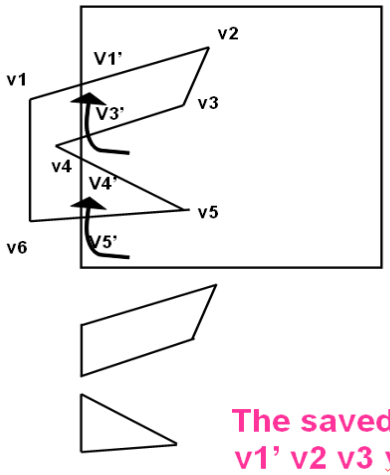
Disadvantage: The algorithm correctly clips convex polygons, but may display extraneous lines for concave polygons .So this algorithm is not suited for concave polygons.

Weiler Atherton polygon clipping algorithm

The vertex-processing procedures for window boundaries are modified so that concave polygons are displayed correctly. This clipping procedure was developed as a method for identifying visible surfaces, and so it can be applied with arbitrary polygon-clipping regions.

The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed, we sometimes want to follow the window boundaries. Which path we follow depends on the polygon-processing direction (clockwise or counterclockwise) and whether pair of polygon vertices currently being processed represents an outside-to-inside pair or an inside- to-outside pair. For clockwise processing of polygon vertices, we use the following rules:

- For an outside-to-inside pair of vertices, follow the polygon boundary.
- For an inside-to-outside pair of vertices, follow the window boundary in a clockwise direction.



- outside to inside pair follow the polygon boundary.
- inside to outside pair follow the window boundary in wise direction

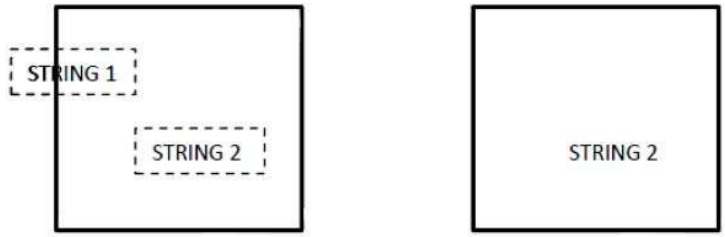
The saved vertices are
v1' v2 v3 v3' v4' v5 v5'

Text clipping

Various techniques are used to provide text clipping in a computer graphics. It depends on the methods used to generate characters and the requirements of a particular application. There are three methods for text clipping which are listed below −

- All or none string clipping
- All or none character clipping
- Text clipping

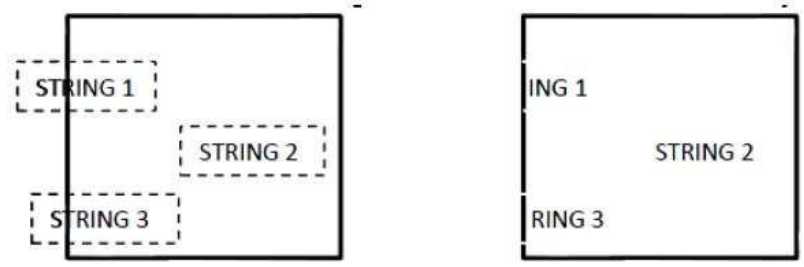
The following figure shows all or none string clipping −



Before Clipping After Clipping

In all or none string clipping method, either we keep the entire string or we reject entire string based on the clipping window. As shown in the above figure, STRING2 is entirely inside the clipping window so we keep it and STRING1 being only partially inside the window, we reject.

The following figure shows all or none character clipping −

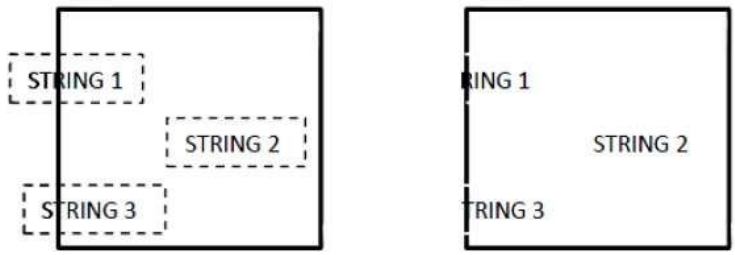


Before Clipping After Clipping

This clipping method is based on characters rather than entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then −

- You reject only the portion of the string being outside
- If the character is on the boundary of the clipping window, then we discard that entire character and keep the rest string.

The following figure shows text clipping −



Before Clipping After Clipping

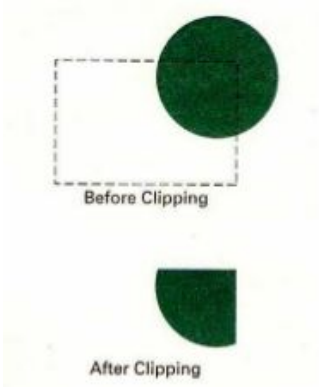
This clipping method is based on characters rather than the entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then

- You reject only the portion of string being outside.
- If the character is on the boundary of the clipping window, then we discard only that portion of character that is outside of the clipping window.

Curve Clipping

Curve-clipping procedures will involve nonlinear equations and this requires more processing than for objects with linear boundaries. The bounding rectangle for a circle or other curved object can be used first to test for overlap with a rectangular clip window.

If the bounding rectangle for the object is completely inside the window, we save the object.If the rectangle is determined to be completely outside window, we discard the object. In either case, there is no further computation necessary. But if the bounding rectangle test fails, we can look for other computation-saving approaches. For a circle, we can use the coordinate extents of individual quadrants and then octants for preliminary testing before calculating curve-window intersections.



Important questions

1. Derive the intersection point of a line with clipping window boundary.
2. Explain a clipping algorithm that works with concave polygons
3. Changing the order in which a sequence of transformations is applied may affect the transformed position of object. Illustrate with example
4. Summarize the steps in 2D viewing pipeline
5. Explain any 2 rigid body transformation
   Hint: Rigid transformation includes combination of rotation translation and reflection. Rigid body transformations are the ones which preserve the shape ans size of the object i.e. magnitude and the angle also.
6. What is a rigid body transformation.
7. Show the composition of two transformations is additive
8. Derive window to viewport coordinate transformation.
9. Explain Cohen Sutherland line clipping algorithm
10. How text clipping is done.
11. Describe short notes on1)Window 2)Viewport 3)World coordinates4)Device coordinates
12. Show that the composition of two rotation is additive by concatenating the matrix representations .
13. Explain 2D rotation
14. Explain 2D reflection
15. Explain Sutherland Hodgeman polygon clipping algorithm
16. Explain the steps involved in rotation of a 2D object about a selected pivot point
17. Explain a parametric line clipping algorithm
18. Discuss the advantages of homogenous coordinate system
19. What is cascading of 2D transformations. Explain
20. What is reflection. How reflection is done with respect to a line y=x.
21. Consider the square A(0,0) B(0,1) C(1,1) D(1,0).What is the new coordinate value when it is sheared in x direction(shear parameter value is ½) and relative to the line y_{ref}=-1.
22. Explain affine transformation.

23. Show that the order in which transformations are performed is important by the transformation of triangle A(1,0), B(0,1), C(1,1) by (a) rotating about 45 degree about the origin and then translating in the direction of vector I, and (b) translating and then rotating.
24. Demonstrate Rotation transformation by rotating the line AB with A(200,100) and B(400,400) by 90 degree angle .
    New points : A(200,100) and B(-100,300)
25. Magnify triangle A(0,0) ; B(1,1) ; C(5,2) with respect to fixed point C(5,2) by factor of 2 in both directions.
    A(0,0) ,B(1,1) and C(5,2) initial points
    After scaling A(-5,-2) ,B(-3,0) and C(5,2)
26. Derive the reflection transformation with respect to line x=0 and line y=0.
27. Translate the triangle with vertices A(30,40), B(20,20) and C(50,30) by shift vectors, tx=50 and ty=70.
28. Reflect the diamond shaped polygon whose vertices are A(-1,0), B(0,-2), C(1,0) and D(0,2) about horizontal line y=2.
29. Shear the polygon ABCD with A(1,1), B(3,1),C(3,3) and D(1,3) in y direction with shear factor sh_y=1.
30. Demonstrate Liang Barsky line clipping for a line AB with A(70,250) and B(250,450) with respect to a window with diagonal coordinate points as (100,200) and (300,400).
31. Demonstrate Parametric line clipping for a line AB with A(70,250) and B(200,300) with respect to a window with diagonal coordinate points as (100,200) and (300,400).
32. Demonstrate Cohen Sutherland line clipping for a line AB with A(70,250) and B(200,300) with respect to a window with diagonal coordinate points as (100,200) and (300,400).
33. Illustrate Liang Barsky Line clipping algorithm.
34. Explain Weiler Atherton Polygon clipping algorithm
35. Explain the significance of a region code in line clipping
36. What is curve clipping.
37. Demostrate any two composite transformations
38. Explain basic 2D transformations with necessary figures. Also write homogenous matrix for each transformations.
39. Explain shearing and reflection.

try it now

A KTU STUDENTS PLATFORM

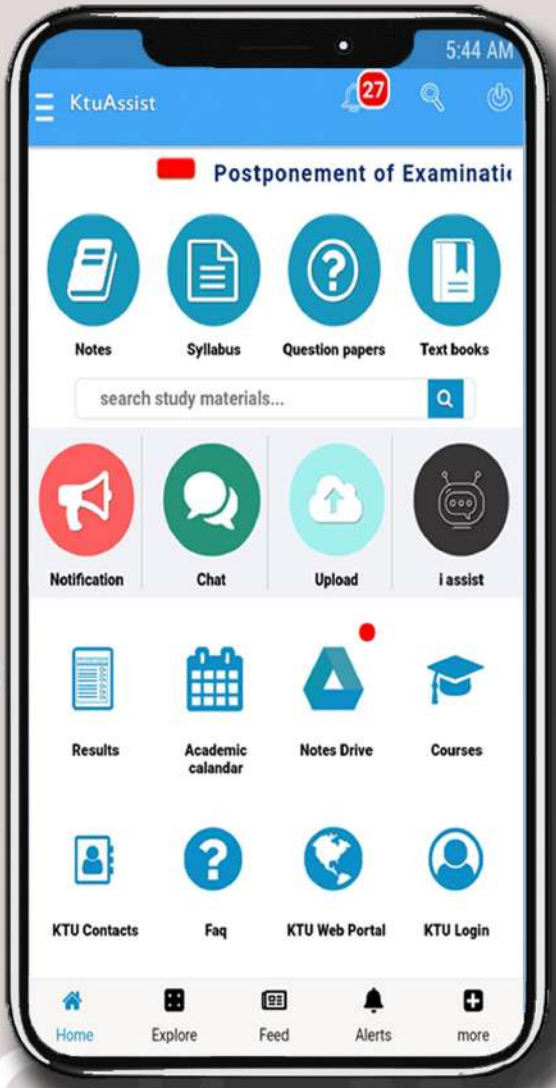
SYLLABUS

NOTES

TEXT BOOKS

QUESTION PAPERS

KTU NOTIFICATION



DOWNLOAD IT FROM GOOGLE PLAY

CHAT

FAQ

LOGIN

CALENDAR

MUCH MORE

DOWNLOAD APP

ktuassist.in

instagram.com/ktu_assist

facebook.com/ktuassist