The point (7, 3) will be the initial point for region 2. The initial decision parameter for region 2 is P_{2k}

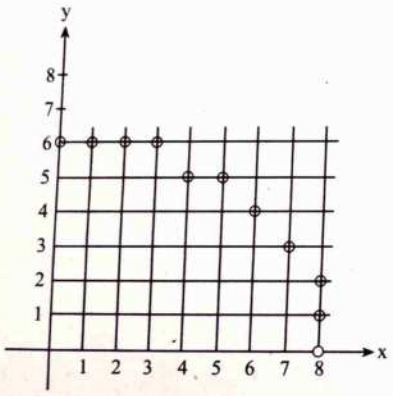
$$= 6^2\left(7+\frac{1}{2}\right)^2 + 8^2(3-1)^2 - 8^2 * 6^2$$

$= -23$

(x_k, y_k)	P_{2k}	(x_{k+1}, y_{k+1})	P_{2k+1}
(7,3)	–23	(8,2)	361
(8,2)	361	(8,1)	297
(8,1)	297	(8,0)	

So, the pixel positions or points to plot of the given curve in first quadrant using mid-point algorithm are

(0,6), (1,6), (2,6), (3,6), (4,5), (5,5), (6,4), (7,3), (8,2), (8,1) and (8,0)



Chapter-3

Two-Dimensional Transformations

3.1 Introduction

Complex picture can be treated as a combination of straight line, circles, ellipse, etc. and if we are able to generate these basic figures, we can also generate combinations of them. Transformation means changing the graphics by changing the position, orientation or size of the original graphics by applying rules. When transformation occurs in 2D plane then it is called 2D transformations. Geometrical transformation is a mathematical procedure which changes/alters orientation, size, and shape of objects i.e., the co-ordinate description of objects.

3.2 Basic Transformations

The basic transformations are:

- Translation
- Scaling
- Rotation

3.2.1 Translation / Shifting

Translation repositions an object along a straight line path from one co-ordinate location to another. A two dimensional point can be translated by adding translation distances t_x and t_y to the original co-ordinate position to move the point to a new position (x', y')

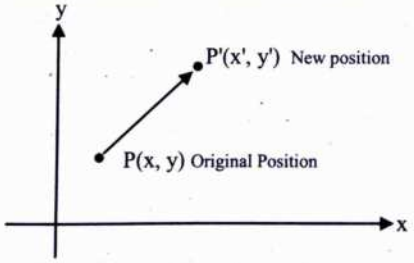


Figure 3.1: *Translation of a point*

$x' = x + t_x$

$y' = y + t_y$

Translation distance pair (t_x, t_y) is called translation vector or shift vector.

In matrix form, we can represent translation as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$P' = P + T$

In translation, just position is changed but shape and size are not changed. Straight line is translated by applying transformation equation to both end points and redrawing the line between those translated end points. Polygon is translated by adding translation vector to the co-ordinates position of each vertex and new polygon is regenerated using that new vertices.

Circle and ellipse are translated by translating the center co-ordinates and then, circle and ellipse are redrawn in new location.

3.2.2 Scaling

A scaling is a basic transformation that alters the size of object.

Points can be scaled by s_x along x axis and s_y along y axis in new points.

Transformation equations are:

$x' = x \cdot s_x$

$y' = y \cdot s_y$

In matrix form,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$P' = s.P$

If $s_x = s_y$, it is uniform scaling. If $s_x \neq s_y$, it is differential scaling.

If $(s_x, s_y) < 1$, then size is reduced.

$= 1$, size is unchanged (remains same).

>1, size is enlarged.

If scaling factor is less than 1, then it moves the object closer to origin. If scaling factor is greater than 1, then it moves the objects away from origin.

Fixed point scaling

The location of the scaled object can be controlled by choosing a position called fixed point that is to remain unchanged after the scaling transformation. Fixed point (x_f, y_f) can be chosen as one of the vertices, centroid of the object, or any other position.

Steps:

1. Translate object so that the fixed point coincides with the co-ordinate origin.
2. Scale the object with respect to the co-ordinate origin.
3. Use the inverse translation of steps 1 to return the object to its original position.

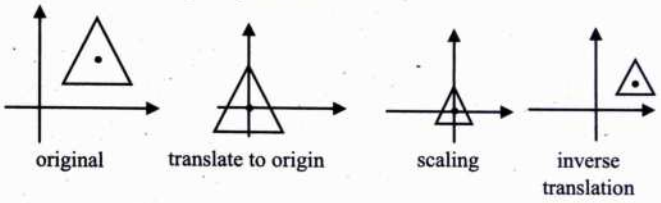


Figure 3.2: *Fixed point scaling of a triangle*

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

$T(x_f, y_f).\ S(s_x, s_y).\ T(-x_f, -y_f) = S(x_f, y_f, s_x, s_y)$

$x' = x_f + (x - x_f)\ s_x$

$y' = y_f + (y - y_f)\ s_y$

OR

$x' = x.s_x + x_f(1-s_x)$

$y' = y.s_y + y_f(1-s_y)$

where the terms $x_f(1-s_x)$ and $y_f(1-s_y)$ are constant for all points in object.

3.2.3 Rotation

Rotation repositions an object along a circular path in the xy plane. To generate rotation, we specify a rotation angle θand the position (x_r, y_r) of the rotation point about which the object is to be rotated. If θ is positive, object is rotated in counterclockwise direction and if θ is negative, object is rotated in clockwise direction.

1. Transformation equation for rotation when pivot point is origin

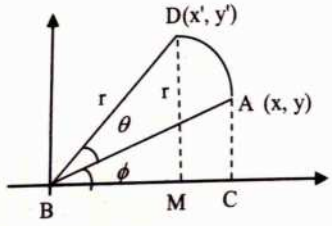


Figure 3.3(a): *Rotation of a point*

From figure, using standard trigonometric identities

In ΔABC,

$x = r\cos\phi$, $y = r\sin\phi$

In ΔDBM,

$x' = r\cos(\theta+\phi)$

$= r\cos\theta\cos\phi - r\sin\theta\sin\phi$

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

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

$= r\sin\theta\cos\phi + r\cos\theta\sin\phi$

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

Representing in matrix form,

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

$P' = R.P$

2. Rotation of a point about an arbitrary pivot position

1. Translation object so that pivot point is moved to co-ordinate origin.
2. Rotate object about origin.
3. Translate object so that pivot point is returned to its original position.

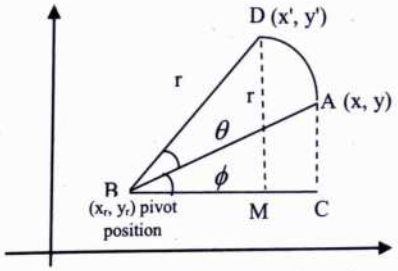


Figure 3.3(b): *Rotation of a point*

Composite matrix, C.M. = T'R(θ)T

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

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

3.3 Homogeneous Co-ordinates and Matrix Representations

The matrix representation for translation, scaling, and rotation are respectively:

$P' = T + P$

$P' = S.P$

$P' = R.P$

Translation involves addition of matrices, whereas scaling and rotation involve multiplication. We may have to perform more than one transformation in same object like scaling the object, then rotate the same object, and finally translation. For this, first co-ordinate positions are scaled, then this scaled co-ordinates are rotated and finally translated. A more efficient approach would be to combine the transformation so that final positions are obtained directly from initial co-ordinates thereby eliminating the calculation of intermediate co-ordinates. This allows us to represent all geometric transformation as matrix multiplication.

We represent each Cartesian co-ordinate position (x, y) with homogeneous triple co-ordinate (x_h, y_h, h)

where $x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$

Thus, general homogenous co-ordinate representation can also be written as

$(x_h, y_h, h) = (h.x, h.y, h)$

where h may be any non-zero value. But for convenience, h = 1 is used.

So, 2D position is represented with homogeneous co-ordinates (x, y, 1).

Expressing position in homogeneous co-ordinates allows us to represent all geometric transformation equation as matrix multiplication.

1. Translation

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

$P' = T(t_x, t_y).P$

Inverse of translation matrix can be obtained by replacing the translation parameter t_x and t_y with their negatives. i.e., $-t_x$ and $-t_y$.

2. Rotation

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

$P' = R(\theta).P$

Inverse rotation matrix can be obtained by replacing θ with $-\theta$.

3. Scaling

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

$P' = S(s_x, s_y).P$

Inverse translation matrix can be obtained by replacing s_x and s_y with $\frac{1}{s_x}$ and $\frac{1}{s_y}$ respectively.

Code in C for 2D Transformation

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void draw(int a[2][5]);
void translate(int a[2][5]);
void scale(int a[2][5]);
void rotate(int a[2][5]);
void main()
{
int a[2][5], i, gd = DETECT, gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
printf("Enter the coordinate positions of the lines");
for(i=0;i<5;i++)
{
```

```
scanf("%d %d",&a[0][i],&a[1][i]);
}

draw(a);
translate(a);
draw(a);
scale(a);
draw(a);
rotate(a);
draw(a);
getch();
closegraph();
}

void draw(int a[2][5])
{
int i=0;
for(i=0;i<4;i++)
{
line(a[0][i],a[1][i],a[0][i+1],a[1][i+1]);
}
}

void translate(int a[2][5])
{int i;
int tx,ty;
scanf("%d %d",&tx,&ty);
for(i=0;i<5;i++)
{
a[0][i]=a[0][i]+tx;
a[1][i]=a[1][i]+ty;
}
}

void scale(int a[2][5])
{int i;
int sx,sy;
scanf("%d%d",&sx,&sy);
for(i=0;i<5;i++)
{
a[0][i]=a[0][i]*sx;
a[1][i]=a[1][i]*sy;
}
}

void rotate(int a[2][5])
{int i, temp1, temp2;
float angle;
scanf("%f",&angle);
for(i=0;i<5;i++)
{
temp1=a[0][i];
temp2=a[1][i];
a[0][i]=temp1*cos(angle)-temp2*sin(angle);
a[1][i]=temp1*sin(angle)+temp2*cos(angle);
}
}
```

3.4 Composite Transformation

3.4.1 Translation

If two successive transformation vectors (t_{x1}, t_{y1})and (t_{x2}, t_{y2}) are applied to a point P, the final position or transformed location P' is calculated as:

$$P' = T(t_{x2}, t_{y2})\{T(t_{x1}, t_{y1}).P\}$$
$$= \{T(t_{x2}, t_{y2})\, T(t_{x1}, t_{y1})\}.P$$
$$= T(t_{x1} + t_{x2}, t_{y1} + t_{y2})\}.P$$

The composite transformation matrix for this sequence of transformation is

$$\begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix}$$

This shows that two successive translations are additive.

3.4.2 Rotation

$$P' = R(\theta_2).\{R(\theta_1).p\} = \{R(\theta_2).R(\theta_1)\}.P = R(\theta_1 + \theta_2) * P$$

$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Successive rotations are additive.

3.4.3 Scaling

$$P' = S(Sx_2, Sy_2)\{S(Sx_1, Sy_1).P\} = \{S(Sx_2, Sy_2).S(Sx_1, Sy_1)\}.P = S(Sx_1, Sx_2, Sy_1.Sy_2).P$$

$$\begin{bmatrix} sx_2 & 0 & 0 \\ 0 & sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} sx_1 & 0 & 0 \\ 0 & sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} sx_1.sx_2 & 0 & 0 \\ 0 & sy_1.sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Two successive scaling are multiplicative

Composite transformation indicates combination of different basic transformation in sequence to obtain desire result or combination could be a sequence of two or more successive transformation (e.g., two successive translation), two successive rotation or two or more successive scaling, etc.

3.5 Other Transformations

Other transformations are:

- Shearing
- Reflection

3.5.1 Shearing

A transformation that distorts the shape of the object such that the transformed shape appears as if the object was composed of internal layers that had been caused to slide over each other is called shearing.

i) Shearing toward x-direction relative to x-axis is given by

$x' = x + sh_x * y$

$y' = y$

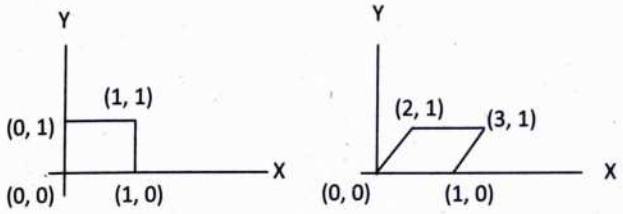


Figure 3.4: *Shearing towards x-direction*

In matrix form,

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

- sh_x is any real number
- co-ordinate position is shifted horizontally by an amount proportional to its distance y value from the axis.
- If sh_x is negative, object shearing is towards left.

ii) Shearing towards y-direction relative to y-axis is given by

$x' = x$

$y' = x \times sh_y + y$

In matrix form,

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

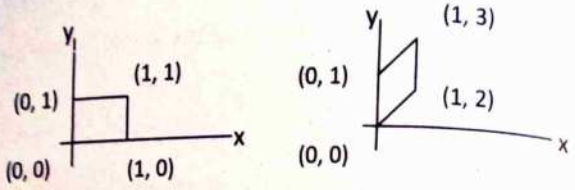


Figure 3.5: Shearing towards y-direction

iii) Shearing in both direction is given by

$x' = x + sh_x \times y$

$y' = x \times sh_y + y$

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

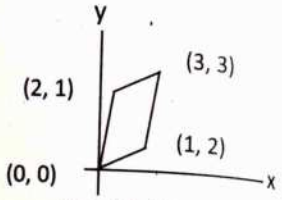


Fig. 3.6: Shearing towards both direction

iv) x-direction shearing relative to other reference line is

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

$y' = y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & -sh_x * yref \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

v) y-direction shearing relative to other reference line is

$x' = x$

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

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y * x_{ref} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This transformation shifts a co-ordinate position vertically by an amount proportional to its distance from the reference line $x = x_{ref}$.

3.5.2 Reflection

It is the transformation that produces mirror image of an object. The mirror image for a two-dimensional reflection is obtained by rotating the object 180^0 about the reflection axis.

i) Reflection about x-axis or about line y = 0

Keep 'x' value same but flip 'y' value

$x' = x$

$y' = -y$

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

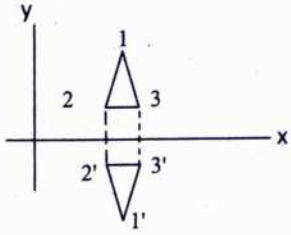


Fig. 3.7: Reflection about x-axis

ii) Reflection about y-axis or about line x = 0

This transformation keeps y value same but flip x value. That is,

$x' = -x$

$y' = y$

$P' = R_{fy}.P$

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

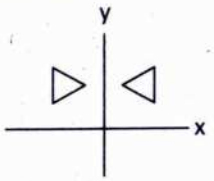


Fig. 3.8:Reflection about y-axis

iii) Reflection about origin

It flips both x and y value i.e.,

$x' = -x$

$y' = -y$

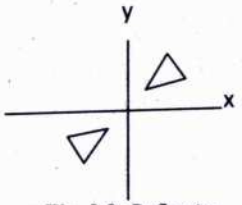


Fig. 3.9: Reflection about origin

iv) Reflection about the line y = x

Steps:

1. Rotate about origin in clockwise direction by 45°, rotates the line y = x to x-axis.
2. Take reflection against x-axis

3. Rotate in anti-clockwise direction by same angle.

$Rf_{(y=x)} = R(\theta)^{-1} \times Rf_x \times R(\theta)$

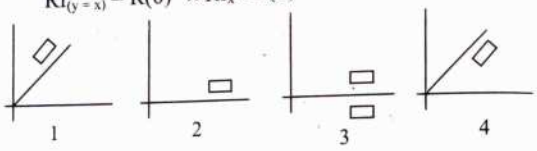


Figure 3.10: *Reflection about the line y = x*

$$R(\theta)^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Rf_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Rf_{(y=x)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

v) Reflection about the line y = −x

Steps:

1. Rotate about origin in clockwise direction by 45°, in, rotates line y = x to y-axis.
2. Take reflection against y-axis.
3. Rotate in clockwise direction by same angle.

$Rf_{(y=-x)} = R(\theta)^{-1} \times Rf_y \times R(\theta)$

where

$$R(\theta)^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Rf_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Rf_{y=-x} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

vi. Reflection about y = mx+c

Steps:

1. Translate line and object so that line passes through origin.

2. Rotate the line and object about origin until the line coincides with one of the co-ordinate axis.
3. Reflect the object through about that axis.
4. Apply inverse rotation about that axis.
5. Translate back to original location.

$C.M. = T^{-1}R(\theta)^{-1}R_f.R(\theta).T$

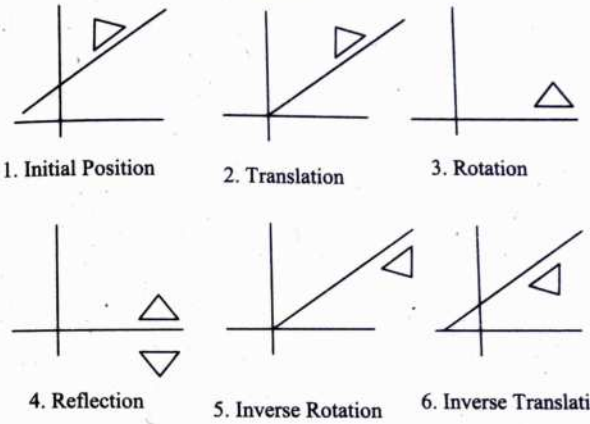


Figure 3.11: *Reflection about the line y = mx + c*

Here,

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

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

$$\text{OR, } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1-m^2}{1+m^2} & \frac{2m}{1+m^2} & \frac{-2cm}{1+m^2} \\ \frac{2m}{1+m^2} & \frac{m^2-1}{1+m^2} & \frac{2cm}{1+m^2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3.6 Two-Dimensional Viewing

Two- Dimensional Viewing is the method about how to display 2D object in display device. It is the formal mechanism for displaying views of picture on an output device. Graphics package allows a user to specify which part of a defined picture is to be displayed and where is to be displayed in device. Any convenient Cartesian coordinate system, referred to as the world co-ordinate reference frame, can be used to define the pictures.

For a 2D picture, a view is selected by specifying a sub area of the total picture area. A user can select a single area for display, or several areas could be selected for simultaneous display or for an animated panning sequence across a scene. The picture parts within the selected areas are then mapped onto specified areas of the device co-ordinates. When multiple view areas are selected, these areas can be placed in separated display locations, or some areas could be inserted into other, larger display areas.

Transformation from world to device co-ordinate involve translation, rotation, and scaling operations as well as procedures for deleting those parts of the picture that are outside the limits of a selected display area, also known as clipping. Windowing is the process of selecting and enlarging the portions of a drawing.

3.7 Coordinate Representation

With few exceptions general packages are designed to be used with Cartesian co-ordinate system. If in other coordinate system, it must be converted

Modeling coordinate/local coordinate/master coordinate

Every individual object or model has its own coordinate system. This coordinate system is called modeling coordinate or local co-ordinate or master co-ordinate. The individual object has its own dimension and its image can be constructed in separate coordinate system. It is known as modeling coordinate.

World coordinate

Once individual objects have been identified or specified, those objects are placed into appropriate position within a scene using reference frame to interact each other. This reference frame is called world co-ordinate. It contains many objects with one unit.

Viewing coordinate

Viewing coordinate is used to define window in the world co-ordinate plane with any possible orientation, i.e. viewing some objects or items at a time.

Normalized device coordinate

In normalized coordinate the coordinates are in range 0 to 1. Generally, a graphical system, world coordinate positions are converted to normalized device coordinates before final conversion to specified device coordinate. This makes the system independent of the various devices that might be used at a particular workstation.

Device coordinate

When the world coordinate description of the scene is transformed to one or more output device reference frame for display; the display coordinate system is referred to a device coordinate or screen coordinates in the case of video monitor.

3.8 The Viewing Pipeline

Window

A world coordinate area selected for display is called window.

View port

An area on a display device to which a window is mapped is called view port.

Window defines what is to be viewed. View port defines where is to be displayed. Window and viewport are rectangular in standard position, with the rectangular edge parallel to co-ordinate

axis. Other shapes are also possible but take longest time to process.

The mapping of a part of world co-ordinate scene to device co-ordinates is referred to as a viewing transformation. Sometimes, 2D viewing transformation is simply referred to window to view port or windowing transformation.

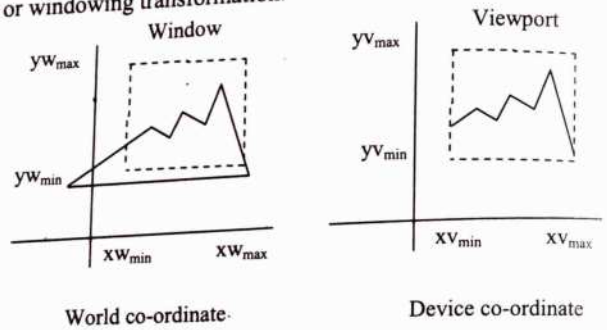


***Figure 3.12:** A viewing transformation using standard rectangles for the window and viewport.*

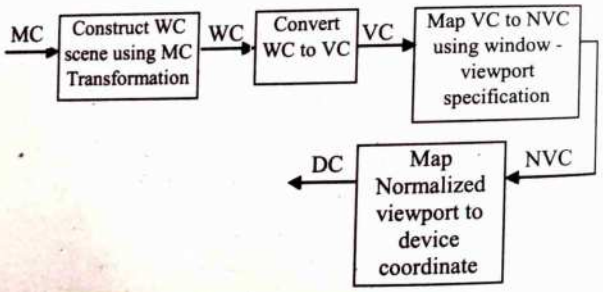


***Figure 3.13:** 2D viewing transformation pipeline*

As in the real life, we see through a small window or the view finder of a camera, a computer-generated image often depicts a partial view of a large scene. Objects are placed into the scene by modeling transformations to a master coordinate system, commonly referred to as the world coordinate systems (WCS).

A rectangular window with its edges parallel to the axis of the WCS is used to select the portion of the scene for which an image is to be generated (displayed).Sometimes an additional coordinate system called the viewing coordinate system (VCS) is introduced to simulate the effort of moving or/and tilting the camera. On the other hand, an image representing a view often becomes part of a larger image, like a photo on an album page, which models a computer monitor's display area.

Since monitor sizes differ from one system to another, we introduce a device-independent tool to describe the display area called normalized device coordinate system (NDCS) in which a unit (1X1) square whose lower left corner is at the origin of the coordinate system that defines the display area of the display device. A rectangular viewport with its edges parallel to the axes of the NDCS is used to specify a sub-region of the display area that embodies the image. The process that convert object coordinates in WCS to normalized device coordinate is called window to viewport mapping or normalization transformation.

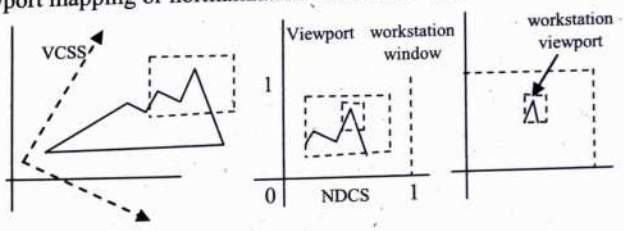


***Figure 3.14:** WCS, VCS, NDCS and workstation*

The process of mapping normalized device coordinates to discrete device coordinates is called workstation transformation, which is essentially a second window to viewport mapping, with a workstation window in the normalized device coordinate system and a workstation viewport in the device coordinate system. Collectively these two coordinate mapping operations are referred to as viewing transformation.

3.9 Window to Viewport Mapping (Coordinate Transformation)

We transfer object description to normalized device coordinates using a transformation that maintains the same relative placement of objects in normalized space they had in viewing coordinates.

If a coordinate position is at the center of the viewing window for instance it will be displayed at the center of viewport.

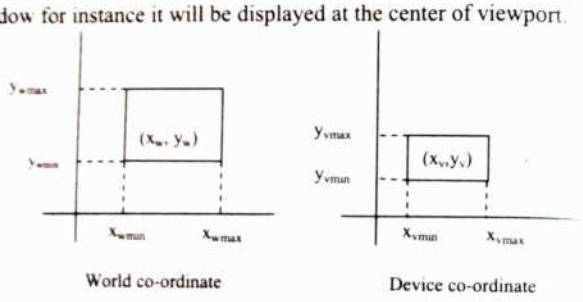


Figure 3.15: *Window to viewport mapping*

A window is specified by four world co-ordinates x_{wmin}, x_{wmax}, y_{wmin}, y_{wmax}. A viewport is described by four device co-ordinate x_{vmin}, x_{vmax}, y_{vmin}, y_{vmax}. To maintain the same relative placement in the viewport as in window, we require.

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

And

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

Solving the equation for viewport

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

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

where, scaling factor,

$$S_x = \frac{x_{v\,max} - x_{v\,min}}{x_{w\,max} - x_{w\,min}}$$

$$S_y = \frac{y_{v\,max} - y_{v\,min}}{y_{w\,max} - y_{w\,min}}$$

Steps (for Window to Viewport Transformation):

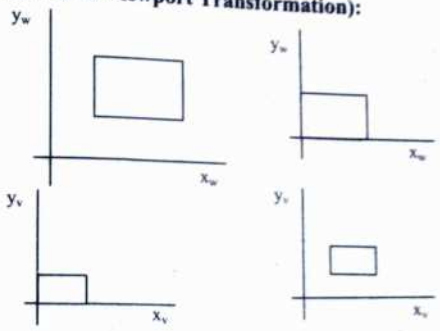


Figure 3.16: *Window to viewport transformation*

1. The object together with its window is translated until the lower left corner of the window is at origin.
2. Object and window are scaled until window has dimension of viewport.

   Perform a scaling transformation using a fixed-point position of (x_{wmin}, y_{wmin}) that scales the window area to the size of the viewport
3. Again, translate to move viewport to its correct position.

Viewing Transformation:

1. Translate window to origin by

   $T_x = -x_{wmin}$

   $T_y = -y_{wmin}$
2. Scale window such that its size is matched to viewport.

$$S_x = \frac{x_{v\,max} - x_{v\,min}}{x_{w\,max} - x_{w\,min}}$$

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

3. Retranslate it by

$T_x = x_{vmin}$

$T_y = y_{vmin}$

Composite matrix $(CM) = T_v \times S_{wv} \times T_w$

T_w = Translate window to origin $= \begin{bmatrix} 1 & 0 & -x_{wmin} \\ 0 & 1 & -y_{wmin} \\ 0 & 0 & 1 \end{bmatrix}$

S_{wv} = Scaling of window to viewport

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

T_V = Translate viewport to original position

$$= \begin{bmatrix} 1 & 0 & x_{vmin} \\ 0 & 1 & y_{vmin} \\ 0 & 0 & 1 \end{bmatrix}$$

Q. Consider the window is located from $X_{wmin} = 20$, $X_{wmax} = 80$, $Y_{wmin} = 40$, $Y_{wmax} = 80$, and a point is located in 30, 80. Identify the new location of the point in the view port considering viewport size $X_{vmin} = 30$, $X_{vmax} = 60$, $Y_{vmin} = 40$, $Y_{vmax} = 60$.

Solution:

$X_{wmin} = 20$ $\quad$ $X_{wmax} = 80$

$Y_{wmin} = 40$ $\quad$ $Y_{wmax} = 80$

$X_{vmin} = 30$ $\quad$ $X_{vmax} = 60$

$Y_{vmin} = 40$ $\quad$ $Y_{vmax} = 60$

$$S_x = \frac{X_{vmax} - X_{vmin}}{X_{wmax} - X_{wmin}} = \frac{60 - 30}{80 - 20} = \frac{3}{6} = \frac{1}{2}$$

$$S_y = \frac{Y_{vmax} - Y_{vmin}}{Y_{wmax} - Y_{wmin}} = \frac{60 - 40}{80 - 40} = \frac{2}{4} = \frac{1}{2}$$

$$X_v = X_{vmin} + (X_w - X_{wmin})S_x$$

$$= 30 + (30 - 20)\frac{1}{2} = 30 + 5 = 35$$

$$Y_v = Y_{vmin} + (Y_w - Y_{wmin})S_y$$

$$= 40 + (80 - 40)\frac{1}{2} = 60$$

By 2nd method

1. Translate window to origin

$T_x = -X_{wmin}$

$T_y = -Y_{wmin}$

2. Scale window such that its size is matched to viewport

$$S_x = \frac{X_{vmin} - X_{vmin}}{X_{wmax} - X_{wmin}}$$

$$S_y = \frac{Y_{vmax} - Y_{vmin}}{Y_{wmax} - Y_{wmin}}$$

3. Retranslate it

$T_x = X_{vmin}$

$T_y = Y_{vmin}$

$$\text{C.M.} = \begin{bmatrix} 1 & 0 & +X_{vmin} \\ 0 & 1 & +Y_{vmin} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -X_{wmin} \\ 0 & 1 & -Y_{wmin} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 30 \\ 0 & 1 & 40 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -20 \\ 0 & 1 & -40 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 30 \\ 0 & 1 & 40 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & -10 \\ 0 & \frac{1}{2} & -20 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & 0 & -10+30 \\ 0 & \frac{1}{2} & -20+40 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 20 \\ 0 & \frac{1}{2} & 20 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P^1 = C.M. \times P$$

$$= \begin{bmatrix} \frac{1}{2} & 0 & 20 \\ 0 & \frac{1}{2} & 20 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 30 \\ 80 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 15+20 \\ 40+20 \\ 1 \end{bmatrix} = \begin{bmatrix} 35 \\ 60 \\ 1 \end{bmatrix}$$

$$P^1 = (35, 60)$$

3.10 Clipping Operations

3.10.1 Introduction

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

3.10.2 Applications of clipping

Clipping is used for extracting part of a defined scene to view and identifying visible surfaces. It is used for drawing and painting operations that allow parts of a picture to be selected for copying, moving, erasing or duplicating. Clipping is used for creating objects using solid-modeling procedures. Clipping algorithm can be applied to window coordinates, so that only the content of the window interior is mapped to device coordinates. Alternatively, the complete world co-ordinate picture can be mapped first to device coordinate or NDC, then clipped against viewport boundaries.

WC clipping removes those primitives outside the window form further consideration, thus eliminating the processing necessary to transform those primitives to device space necessary to transform those primitives to device space. Viewport clipping on the other hand, can reduce calculations by allowing concatenation of viewing and geometric transformation matrices. But viewport clipping does require that the transformation to device co-ordinates

be performed for all objects, including those outside the window area.

3.10.3 Some Primitive Types of Clipping

- Point clipping
- Line clipping (straight line segment)
- Area clipping (polygon)
- Curve clipping
- Text clipping

1. Point Clipping

Assuming that the clip window is a rectangle in standard position, we save a point P(x, y) for display, if the following inequalities are satisfied.

$$xw_{min} \le x \le xw_{max}$$

$$yw_{min} \le y \le yw_{max}$$

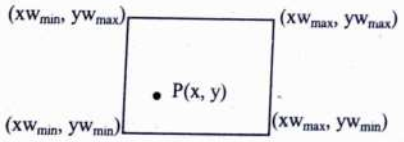
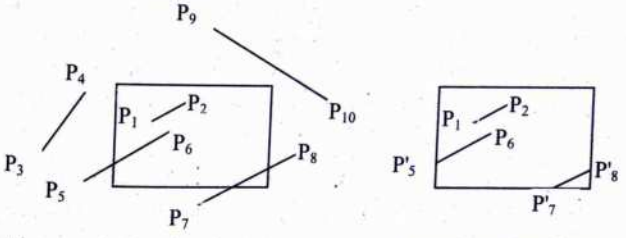


Figure 3.17: Point clipping

The edges of the clip window can be either WC window boundary or viewport boundaries. If any one of these four inequalities is not satisfied, the point is clipped (not saved for display).

2. Line Clipping



Fig(a): Before Clipping

Fig(b): After clipping

Figure 3.18: Line clipping

Line clipping against rectangular clip window

A line-clipping procedure involves several parts. First, a given segment is tested to determine whether it lies completely inside the clipping window. If it does not, it is tested to determine whether it lies completely outside the window. Finally, if we cannot identify a line as complete inside or outside, we must perform intersection calculation with one or more clipping boundaries.

We process line through the 'Inside-outside" test by checking the line endpoints. A line with both endpoints outside anyone of the clip boundaries (line P_1 to P_4) is outside the window. All other lines cross one or more clipping boundaries and may require calculation of multiple intersection points. For a line segment with end points (x_1, y_1) and (x_2, y_2) and one or both end points outside the clipping rectangle, the parametric representation could be used

$$x = x_1 + u(x_2 - x_1)$$

$$y = y_1 + u(y_2 - y_1), \text{ where } 0 \le u \le 1$$

If the value of u is outside the range 0 to 1, the line does not enter the interior of the window at that boundary. If the value of 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.

Cohen-Sutherland Line Clipping

Cohen-Sutherland line clipping algorithm is one of the oldest and most popular procedures. It uses bit operation to perform this test. It speeds up the processing of line segments by performing tests that reduce the no. of intersection that must be calculated.

For each end-point, a 4 digits binary code (called a region code) is assigned to identify the location relative to boundary. Lower order bit (bit1) set to '1' if end point is at the left side of the window, else set to '0' (by numbering the bit position in the region code as 1 to 4 from right to left. Bit 2 is set to '1' if end point lies at right side else set '0'. Bit 3 is set to '1' if end point lies bottom, else set '0'. Bit 4 is set to '1' if end point lies top, else set '0'.

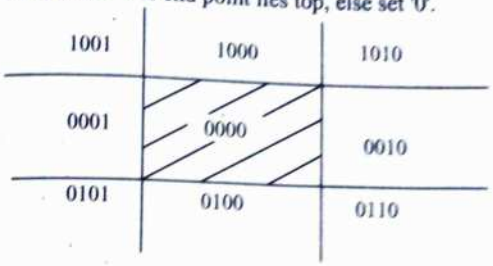


Figure 3.19: *Clip window and region code*

Algorithm

Step 1: Assign Region Code (TBRL)

Step 2: Establish region code for all line end points.

Bit 1 is set to '1' if $x < x_{min}$ else set to '0'.

Bit 2 is set to '1' if $x > x_{max}$ else set to '0'.

Bit3 is set to '1' $y < y_{min}$ else set '0'

Bit 4 is set to '1' $y > y_{max}$ else set to '0'

Step 3: Determine whether line is completely inside or outside window using test.

a) If both end pint have region code ' 0000' line is completely inside.

b) If logical AND of end points of a line not '0000' line is completely outside.

Step 4. If both condition of step 2 fails. i.e. Logical AND give '0000' we need to find the intersection with window boundary.

Here,

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

a) If bit 1 is 1, line intersection with left boundary, so,

$y_i = y_1 + m(x - x_1)$ where $x = x_{min}$

b) If bit 2 is 1, line intersect with right boundary, so,

$y_i = y_1 + m(x - x_1)$ where $x = x_{max}$

c) If bit 3 is 1; line intersect with lower boundary, so,

$x_i = x_1 + \frac{1}{m}(y - y_1)$ where $y = y_{min}$

d) If bit 4 is 1, line intersects with upper boundary, so,

$x_i = x_1 + \frac{1}{m}(y - y_1)$ where $y = y_{max}$

Here, x_i and y_i are x, y intercepts for that line, update

Step 5: Repeat step 2 and 4 till completely accepted

Liang-Barsky Line Clipping

Faster line clippers have been developed that are based on analysis of the parametric equation of a line segment, which can be written in the form,

$x = x_1 + u\,\Delta x$

$y = y_1 + u\,\Delta y,\ 0 \le u \le 1$

where $\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$

Using these parametric equations, Cyrus and Beck developed an algorithm that is generally more efficient than the Cohen-Sutherland algorithm. Later, Liang and Barsky independently devised an even faster parametric line-clipping algorithm. Following the Liang-Barsky approach, we first write the point clipping condition in the parametric from,

$xw_{min} \le x_1 + u\,\Delta x \le xw_{max}$

$yw_{min} \le y_1 + u\,\Delta y \le yw_{max}$

Now, we can write

$-u\,\Delta x \le x_1 - xw_{min}$

$u\,\Delta x \le xw_{max} - x_1$

$-u\,\Delta y \le y_1 - yw_{min}$

$u\,\Delta y \le yw_{max} - y_1$

Each of these four inequalities can be expressed $u.p_k \le q_k$, k=1, 2, 3, 4where parameters p and q are defined as

k = 1 (is the line inside the left boundary) p1 = $-\Delta x$, q1 = $x_1 - x_{wmin}$

k = 2 (is the line inside the right boundary)p2 = Δx, q2 = $x_{wmax} - x_1$

k = 3 (is the line inside the bottom boundary)p3 = $-\Delta y$, q3 = $y_1 - y_{wmin}$

k = 4 (is the line inside the top boundary) p4 = Δy, q4 = $y_{wmax} - y_1$

when $P_k < 0$, the infinite extension of line proceeds from the outside to inside of the infinite extension of this particular clipping boundary. When $P_k > 0$, the line proceeds from inside to outside

Trivial Rejection

The line with $p_k = 0$ for some k and one $q_k < 0$ for these k is rejected. For line with $p_k = 0$ for some k and all $q_k >= 0$ for those k, line is parallel to one of the clipping boundary and some portion of the line is inside. For intersection with boundary, the parameters are supposed to be r_k, r_k is given by

$$r_k = \frac{q_k}{p_k}$$

Clipped line will be s

$x_1' = x_1 + u_1\Delta x$

$y_1' = y_1 + u_1\Delta y$

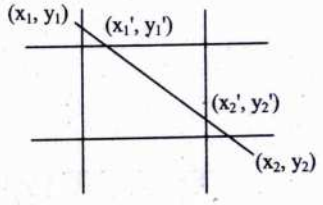


Figure 3.20: *Clip window, line and intersection points*

The value of r_k becomes candidate for $u_1$if pk < 0 . The value of u1 is greater or equal to 0. It is for intersection with the boundaries to which line enters the boundary = maximum value between 0 and r.

$x_2' = x_1 + u_2\Delta x$

$y_2' = y_1 + u_2\Delta y$

u_2 is lesser or equal to 1. It is for intersection with the boundaries to which line leaves the boundary. Its minimum value is between r and 1.

Algorithm

1) If $p_k = 0$ for some k, then the line is parallel to the clipping boundary now test q_k
   For these k, If one $q_k < 0$ then the line is completely outside. And can be eliminated.
2) For non zero p_k, calculate $r_k = \frac{q_k}{p_k}$, it gives the intersection point.
2) If $p_k<0$ then line proceeds from outside to inside boundary. Calculate $u_1 = \max(0, \{r_k : r_k = \frac{q_k}{p_k}\})$ to determine intersection point with the possible extended clipping boundary k and obtain a new starting point for the line at u_1.
3) If $p_k>0$ then line proceeds from inside to outside the boundary.Calculate $u_2 = \min(1, \{r_k : r_k = \frac{q_k}{p_k}\})$ to determine intersection point with extended clipping boundary k and obtain a new point at u_2
4) If $u_1>u_2$, then the line is outside and therefore rejected or line is discarded.
5) The line is now between $[u_1, u_2]$

SOLVED NUMERICALS AND DERIVATIONS

1. Rotate a triangle A(5,6), B(6,2) and C(4,1) by 45 degree about an arbitrary point (3,3). *[2076 Ashwin Back]*

Solution:

Composite matrix

$$= T_{(3,3)} R_{(45)} T_{(-3,-3)}$$

$$= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos45° & \sin45° & 0 \\ \sin45° & \cos45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, $P^1 = C.M. * P$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 3 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{-6}{\sqrt{2}}+1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 & 4 \\ 6 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{\sqrt{2}}+3 & -\frac{4}{\sqrt{2}}+3 & 3 \\ \frac{5}{\sqrt{2}}+1 & \frac{2}{\sqrt{2}}+1 & -\frac{1}{\sqrt{2}}+1 \\ 1 & 1 & 1 \end{bmatrix}$$

2. Consider a triangle A(0,0), B(1,1), C(5,2). The triangle has to be rotated by an angle 45° about the point P(–1,–1). What will be the co-ordinate of new triangle.

Solution:

$$CM = T_{(-1,-1)} R_{45} T_{(1,1)}$$

$$= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos45° & -\sin45° & 0 \\ \sin45° & \cos45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \sqrt{2}-1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P^1 = C.M \times P$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \sqrt{2}-1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & -1 & \frac{3}{\sqrt{2}}-1 \\ \sqrt{2}-1 & 2\sqrt{2}-1 & \frac{9}{\sqrt{2}}-1 \\ 1 & 1 & 1 \end{bmatrix}$$

$A' = (-1, \sqrt{2} - 1)$

$B' = (-1, 2\sqrt{2} - 1)$

$C' = \left(\frac{3}{\sqrt{2}} - 1, \frac{9}{\sqrt{2}} - 1\right)$

3. ***Scale an object (4, 4), (3, 2), (5, 2) about a fixed point (4, 3) by 2.***

Solution:

$P' = C.M. \times P$

$$C.M. = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & -8 \\ 0 & 2 & -6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & -8-4 \\ 0 & 2 & -6-3 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 4 & 3 & 5 \\ 4 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$P' = C.M. \times P$

$$= \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & -3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & 5 \\ 4 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 8-4 & 6-4 & 10-4 \\ 8-3 & 4-3 & 4-3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 6 \\ 5 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

So, scaled points are (4, 5), (2, 1) and (6, 1)

4. ***Rotate the triangle (5,5), (7,3), (3,3) about fixed points (5,4) in counter clockwise by 90°.***

Solution:

$C.M. = T_{(5,4)} R_{90} T_{(-5,-4)}$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90^0 & -\sin 90^0 & 0 \\ \sin 90^0 & \cos 90^0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Now $P' = C.M.^* P$

$$= \begin{bmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 7 & 3 \\ 5 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -5+9 & -3+9 & -3+9 \\ 8-1 & 7-1 & 3-1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 6 & 6 \\ 4 & 6 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$\therefore$ New co-ordinates are (4,4), (6,6) (6,2)

5. ***Reflect an object (2, 3), (4, 3), (4, 5) about line $y = x + 1$.***

Solution:

Here, m = 1

c = 1

$C.M. = T' R_\theta' Rf_x . R_\theta T$

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_\theta = \begin{bmatrix} \cos 45^0 & \sin 45^0 & 0 \\ -\sin 45^0 & \cos 45^0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{f_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_\theta' = \begin{bmatrix} \cos 45^0 & -\sin 45^0 & 0 \\ \sin 45^0 & \cos 45^0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$P' = C.M. \times P$

$$= \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 4 \\ 3 & 3 & 5 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 4 \\ 3 & 5 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

Required points are(2,3), (2,5), (4,5)

6. ***Reflect an object (2,3), (4,3) and (4,5) about line y = 2x+1***

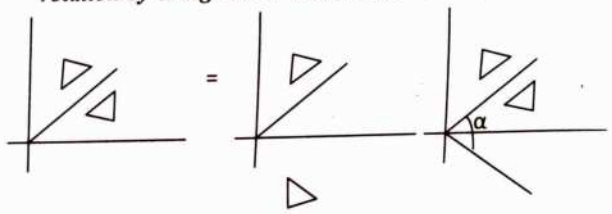
Solution:

Hints:

Here, m = 2, c = 1

1. Translate with value of c
2. Rotate by angle $\tan^{-1}(m)$
3. Reflect about x axis
4. Again re-rotate
5. Translate

7. ***The reflection along the line y = x is equipment to the reflection along the x axis followed by counter clockwise rotation by α degree. Find the angle α.*** *[2071 Chaitra]*



Solution:

Steps for reflection about line y = x

i) Rotate about origin in clockwise direction by 45°

ii) Reflection about x –axis

iii) Rotate in anticlockwise direction in 45°

$Rf_{(y=x)} = R(\theta)^{-1} \times Rf_x \times R(\theta)$

$$= \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(i)}$$

Again, the matrix for the reflection along x-axis followed by counter clockwise rotation by α degree

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

$$= \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ \sin\alpha & -\cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(ii)}$$

Now equating matrix (1) and (2)

$\cos\alpha = 0$

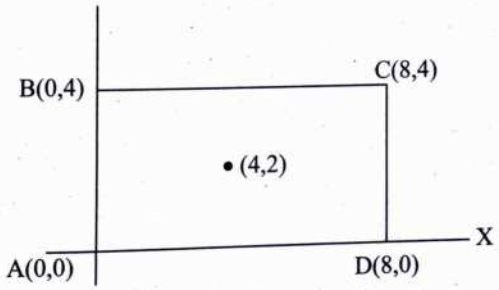
$\sin\alpha = 1$

$-\cos\alpha = 0$

$\because \alpha = 90$

8. ***Find the transformation matrix that transforms the rectangle ABCD whose center is at (4, 2) is reduced to half of its size, the center will remain same. The co-ordinate of ABCD are A(0, 0), B(0, 4), C(8, 4) and D(8, 0). Find co-ordinate of new square. Also derive the transformation matrix to convert this rectangle to square.*** *[2072 kartik]*

Solution:



The transformation matrix that transformations the rectangle ABCD whose center is (4, 2) is reduced to half of its size keeping the center same is,

P' = C.M. *P

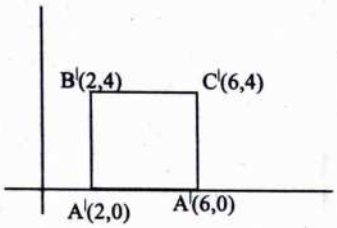
$$C.M. = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C.M = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now to find new coordinate of new square,

A' = C.M*A

$$\begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$



B' = C.M*B

$$\begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix}$$

C' = C.M. × C

$$\begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 1 \end{bmatrix}$$

D' = C.M. × D

$$= \begin{bmatrix} 1/2 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \\ 1 \end{bmatrix}$$

The coordinates of new square are (2,0), (2,4), (6,4), and (6,0)

Steps to get the transformation matrix to convert this rectangle to square are;

i. Translate by (-4, -2)

ii. Scale by S_x=1/2

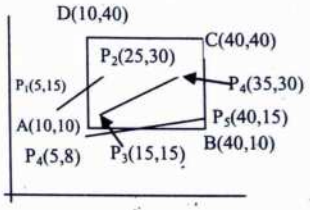
iii. Rotate by (4, 2)

$$C.M. = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

9. ***Given a clipping window A(10, 10), B(40,10), C(40, 40) and D(10, 40). Using Cohen-Sutherland line clipping algorithm find region code of each end points of lines P_1P_2, P_3P_4 and P_5P_6 where co-ordinates are P_1(5,15), P_2(25, 30), P_3(15, 15), P_4(35, 30), P_5(5, 8) and P_6(40, 15). Also find clipped lines using above parameters. ?*** *[2071 Shrawan]*

Solution:

Step 1: Assign the region code.



For P_1 (5, 15)

5 < 10 true so bit 1= 1

5 > 40 False so bit 2 = 0

15 < 10 false so bit 3 = 0

15 > 40 false so bit 4 = 0

Region code for

P_1 (5,15) = 0 0 0 1

For P_2 (25,30)

25 < 10 false so bit 1= 0

25 > 40 False so bit 2= 0

30 < 10 false so bit 3= 0

30 > 40 false so bit 4= 0

Region code for P_2 (25,30)= 0 0 0 0

For P_3 (15,15)

15 < 10 false so bit 1= 0

15 > 40 False so bit 2= 0

15 < 10 false so bit 3= 0

15 > 40 false so bit 4= 0

Region code for P_3 (15,15)= 0 0 0 0

For P_4 (35,30)

35 < 10 false so bit 1= 0

35 > 40 False so bit 2= 0

30 < 10 false so bit 3= 0

30 > 40 false so bit 4= 0

Region code for P_4 (35,30)= 0 0 0 0

For P_5 (5,8)

5 < 10 true so bit 1 = 1

5 > 40 False so bit 2 = 0

8 < 10 true so bit 3 = 1

8 > 40 false so bit 4 = 0

Region code for P_5 (5,8)= 0 1 0 1

For P_6 (40,15)

40 < 10 false so bit 1= 0

40 > 40 False so bit 2= 0

15 < 10 false so bit 3= 0

15 > 40 false so bit 4= 0

Region code for P_6 (40,15)= 0 0 0 0

Now, for line P_1 P_2, P_1(5, 15), P_2(25,3 0)

Region code for P_1= 0001

Region code for P_2 = 0000

i. Both end point have not region code '0000', line is not completely inside.

ii. The logical AND of end points of the line P_1 P_2 is '0000', so line is not completely outside.

So now,

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{30 - 15}{25 - 5} = \frac{15}{20} = \frac{3}{4}$$

If bit1 is 1, line intersects with left boundary.

So,

$$y_i = y_1 + m(x_{min} - x_1)$$

$$= 15 + \frac{3}{4}(10 - 5)$$

$$= 18.75$$

$y_i = 18.75 = 19$

$x_i = 10$

Region code is,

10 < 10 false bit 1 =0

10 > 40 false bit 2 = 0

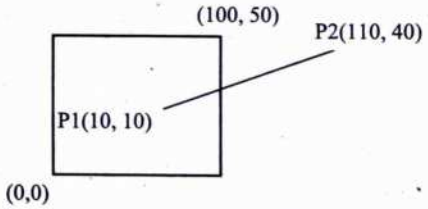
19 < 10 false bit 3= 0

19 > 40 false bit 4= 0

So, the required line is P_1' (10, 19) and P_2' (25,30)

10. Use Liang Barsky clipping method to clip a line starting from P1 (10, 10) and ending at P2(110, 40) against the window having its lower corner at (0, 0) and upper right corner at (100, 50)

Solution:



K	p_k	q_k	r_k
1	$-\Delta x$ $= -(110-10)$ $= -100$ i. e., $p_k<0$	$x_1 - x_{wmin}$ $= 10-0$ $= 10$	r1 = 10/-(100) = -1/10 = candidate for u_1

K	p_k	q_k	r_k
2	Δx = (110-10) = 100 i. e., $p_k>0$	$x_{max}-x_1$ = 100-10 = 90	r_2 = 90/100 = 0.9 = candidate for u_2
3	$-\Delta y$ = -(40-10) = -30 i. e.$Pk<0$	y_1-y_{min} = 10-0 = 10	r_3 = 10/-30 = -1/3 = candidate for u_1
4	Δy = (40-10) = 30 i.e. $Pk > 0$	$y_{max}-y_1$ = 50-10 = 40	r_4 = 40/30 = 4/3 = candidate for u_2

We take $u_1 = 0$ and $u_2 = 0.9$

Clipped line

$x_1' = 10 + 0 \times 100 = 10$

$y_1' = 10 + 0 \times 30 = 10$

$x_2' = x1 + u_2 \times 100 = 10 + 0.9 \times 100 = 100$

$y_2' = y1 + u_2 \times 100 = 10 + 0.9 \times 30 = 37$

11. State the condition of point clipping perform clipping operation for the following using Liang Barskey line clipping algorithm. *[2070 Chaitra]*

Solution:

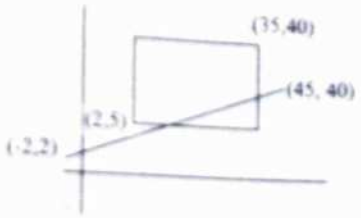
Clipping window: $(x_{min}, y_{min}) = (2,5)$

And $(x_{max}, y_{max}) = (35,50)$

Line $(x_1, y_1) = (-2,2)$ and $(x_2, y_2) = (45,40)$

$\Delta x = x_2 - x_1 = 45 - (-2) = 47$

$\Delta y = y_2 - y_1 = 40 - 2 = 38$



k	p_k	q_k	$R_k = \frac{q_k}{p_k}$
0	$\Delta x = -47$, $p_k<0$	$x_1 - x_{min} = -4$	$0.0851(u_1)$
1	$\Delta x = 47$	$x_{max} - x_1 = 37$	$0.787(u_2)$
2	$-\Delta y = -38$	$y_1 - y_{min} = -3$	$0.0789(u_1)$
3	$\Delta y = 38$	$y_{max} - y_1 = 48$	$1.263(u_2)$

$u_1 = \max(0, r_k)$

$= \max(0, 0.0851, 0.0789)$

$= 0.0851$

$u_2 = \min(1, r_k) = \min(1, 0.787, 1.263) = 0.787$

$x_1' = x_1 + u_1\Delta x = -2 \times 0.051 \times 47 = 1.997 \approx 2$

$y_1' = y_1 + u_1\Delta y = -2 + 0.0851 \times 38 = 5$

$x_2' = x_1 + u_2\Delta x = -2 + 0.787 \times 47 = 35$

$y_2' = y_1 + u_2\Delta y = 2 + 0.787 \times 38 = 32$

Required points are (2, 5) and (35, 32)

12. Write down the condition for point clipping. Find the clipped region in window of diagonal vertex (10, 10) and (100, 100) for line P1(5, 120) and P2(80, 7) using Liang-Barsky line clipping method. *[2072 kartik]*

Solution:

Assuming that the clip window is a rectangle in standard position, we save a point P(x, y) for display, if the following inequalities are satisfied.

K	p_k	q_k	r_k
2	Δx = (110-10) = 100 i. e., $p_k>0$	$x_{k\,max}-x_1$ = 100-10 = 90	r_2 = 9/100 = 0.9 = candidate for u_2
3	$-\Delta y$ = -(40-10) = -30 i. e.$P_k<0$	$y_1-y_{k\,min}$ = 10-0 = 10	r_3 = 10/-30 = -1/3 = candidate for u_1
4	Δy = (40-10) = 30 i.e. $P_k>0$	$y_{k\,max}-y_1$ = 50-10 = 40	r_4 = 40/30 = 4/3 = candidate for u_2

We take $u_1 = 0$ and $u_2 = 0.9$

Clipped line :

$x_1' = 10 + 0 \times 100 = 10$

$y_1' = 10 + 0 \times 30 = 10$

$x_2' = x1 + u_2 \times 100 = 10 + 0.9 \times 100 = 100$

$y_2' = y1 + u_2 \times 100 = 10 + 0.9 \times 30 = 37$

11. State the condition of point clipping perform clipping operation for the following using Liang Barsky line clipping algorithm. *[2070 Chaitra]*

Solution:

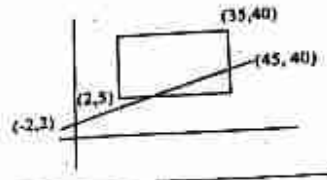
Clipping window: $(x_{min}, y_{min}) = (2,5)$

And $(x_{max}, y_{max}) = (35,50)$

Line $(x_1 y_1) = (-2,2)$ and $(x_2, y_2) = (45,40)$

$\Delta x = x_2 - x_1 = 45 - (-2) = 47$

$\Delta y = y_2 - y_1 = 40 - 2 = 38$



k	p_k	q_k	$u_k = \frac{q_k}{p_k}$
0	$\Delta x = -47$, $p_k<0$	$x_1 - x_{min} = -4$	0.0851(u_1)
1	$\Delta x = 47$	$x_{max} - x_1 = 37$	0.787(u_2)
2	$-\Delta y = -38$	$y_1 - y_{min} = -3$	0.0789(u_1)
3	$\Delta y = 38$	$y_{max} - y_1 = 48$	1.263(u_2)

$u_1 = \max(0, r_k)$

$= \max(0, 0.0851, 0.0789)$

$= 0.0851$

$u_2 = \min(1, r_k) = \min(1, 0.787, 1.263) = 0.787$

$x_1' = x_1 + u_1 \Delta x = -2 \times 0.051 \times 47 = 1.997 = 2$

$y_1' = y_1 + u_1 \Delta y = -2 + 0.0851 \times 38 = 5$

$x_2' = x_1 + u_2 \Delta x = -2 + 0.787 \times 47 = 35$

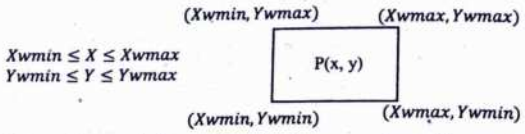
$y_2' = y_1 + u_2 \Delta y = 2 + 0.787 \times 38 = 32$

Required points are (2,5) and (35, 32)

12. Write down the condition for point clipping. Find the clipped region in window of diagonal vertex (10, 10) and (100, 100) for line P_1(5, 120) and P_2(80, 7) using Liang-Barsky line clipping method. *[2072 kartik]*

Solution:

Assuming that the clip window is a rectangle in standard position, we save a point P(x, y) for display, if the following inequalities are satisfied.



- The edges of the clip window can be either we window boundary or viewport boundaries.
- If any one of these four inequalities is not satisfied, the point is clipped i.e., not saved for display.

Here,

$x_{wmin} = 10$

$y_{wmin} = 10$

$x_{wmax} = 100$

$y_{wmax} = 100$

$(x_1, y_1) = (5, 120)$

$(x_2, y_2) = (80, 7)$

Now, we have to find out the value of u_1 and u_2 by calculating p_k, q_k, r_k from k = 1 to 4.

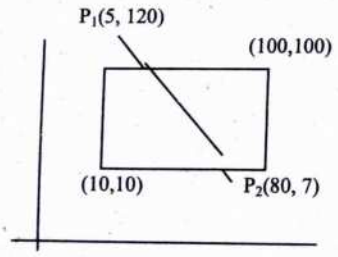
Then, the clipped line will be

$x_1' = x_1 + u_1\Delta x$

$y_1' = y_1 + u_1\Delta y$

$x_2' = x_1 + u_2\Delta x$

$y_2' = y_1 + u_2\Delta y$



K	p_k	q_k	$r_k = \frac{q_k}{p_k}$
1	$-\Delta x = -(80-5) = -75$ i.e., $p_k<0$	$x_1 - x_{wmin}$ $(5-10) = -5$	$r_1 = \frac{-5}{-75} = \frac{1}{15}$ (u_1)
2	Δx $= -(80-5)$ $= 75$ i.e., $p_k>0$	$x_{wmax} - x_1$ $= 100 - 5$ $= 95$	$r_2 = \frac{95}{75}$ (u_2)
3	Δy $= -(7-120)$ $= 113$ i.e., $p_k>0$	$y_1 - y_{wmm}$ $= (120-10)$ $= 110$	$r_3 = \frac{110}{113}$ (u_2)
	Δy $= -113$ i.e., $p_k<0$	$y_{wmax} - y_1$ $= 100 - 120$ $= -20$	$r_4 = \frac{-20}{-113} = \frac{20}{113}$ (u_1)

Now,

$u_1 = \max(0, r_k)$

$= \max(0, \frac{1}{15}, \frac{20}{113}) = \frac{20}{113}$

$u_2 = \min(1, r_k)$

$= \min(1, \frac{95}{75}, \frac{110}{113}) = \frac{110}{113}$

$x_1' = x_1 + u_1\Delta x$

$= 5 + \frac{20}{113} \times 75 = 18.27$

$y_1' = y_1 + u_1\Delta y$

$= 120 + \frac{20}{113} \times (-113) = 100$

$x_2' = x_1 + u_2\Delta x$

$= 5 + \frac{110}{113} \times 75 = 78$

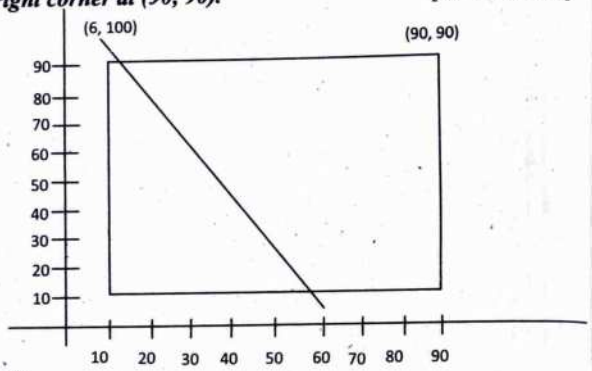
$y_2' = y_1 + u_2\Delta y$

$= 120 + \frac{110}{113} \times (-113) = 10$

$P_1'(x_1', y_1') = P_1'(18, 27, 100)$

$P_2'(x_2', y_2') = P_2'(78, 10)$

13. ***Use Liang Barsky line clipping algorithm to clip a line starting from (6, 100) and ending at (60, 5) against the window having its lower left corner at (10, 10) and upper right corner at (90, 90).*** *[2075 Ashwin]*



Solution:

Clipping window $(x_{min}, y_{min}) = (10, 10)$

$(x_{max}, y_{max}) = (90, 90)$

$(x_1, y_1) = (6, 100)$ and $(x_2, y_2) = (60, 5)$

$\Delta x = x_2 - x_1 = 60 - 6 = 54$

$\Delta y = y_2 - y_1 = 5 - 100 = -95$

K	p_k	q_k	$r_k = q_k/p_k$
0	$-\Delta x = -54 (p_k < 0)$	$x_1 - x_{min} = 6 - 10 = -4$	$0.074(u_1)$
1	$\Delta x = 54$	$x_{max} - x_1 = 90 - 6 = 84$	$1.55(u_2)$
2	$-\Delta y = 95$	$y_1 - y_{min} = 100 - 10 = 90$	$0.947(u_2)$
3	$\Delta y = -95$	$y_{max} - y_1 = 90 - 100 = -10$	$0.105(u_1)$

$u_1 = \max(0, r_k) = 0.105$

$u_2 = \min(0, r_k) = 0.947$

$x_1' = x_1 + u_1\Delta x = 6 + 0.105 \times 54 = 11.67 \approx 12$

$y_1' = y_1 + u_1\Delta y = 100 + 0.105 \times (-94) = 90.025 \approx 90$

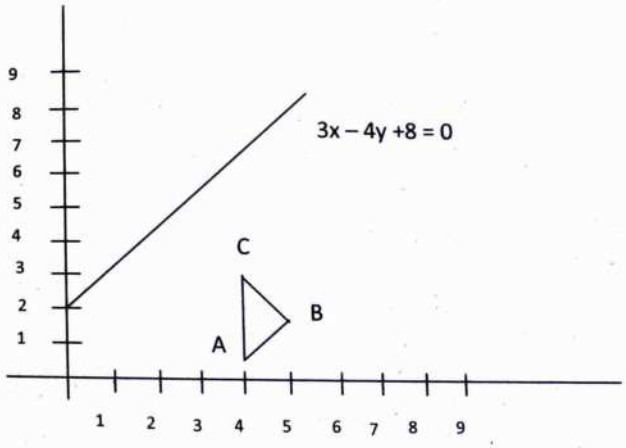
$x_2' = x_1 + u_2\Delta x = 6 + 0.947 \times 54 = 57.118 \approx 57$

$y_2' = y_1 + u_2\Delta y = 100 + 0.947 \times (-94) = 10.035 \approx 10$

Required points are (12, 90) and (57, 10)

14. ***Reflect the triangle ABC about the line $3x - 4y + 8 = 0$ the position vector of coordinate ABC as A(4, 1), B(5, 2) and C(4, 3).*** *[2075 Ashwin]*

Solution:



The arbitrary line about which the triangle ABC has to be reflected is $3x - 4y + 8 = 0$

i.e., $y = \frac{3}{4}x + 2$

$m = \frac{3}{4}$

$c = 2$

$$\theta = \tan^{-1}(m) = \tan^{-1}\frac{3}{4} = 36.8698^{\circ}$$

$$C.M. = T^{-1}R^{-1}{}_{\theta}R_{fx}R_{\theta}T$$

$$T=\begin{bmatrix}1 & 0 & t_x\\ 0 & 1 & t_y\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & -2\\ 0 & 0 & 1\end{bmatrix}$$

$$R_{\theta}=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}\cos(36.87) & \sin(36.87) & 0\\ -\sin(36.87) & \cos(36.87) & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$R_{fx}=\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$R(-\theta)=\begin{bmatrix}\cos(-\theta) & \sin(-\theta) & 0\\ -\sin(-\theta) & \cos(-\theta) & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}\cos(-36.87) & \sin(-36.87) & 0\\ -\sin(-36.87) & \cos(-36.87) & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$T^{-1}=\begin{bmatrix}1 & 0 & t_x\\ 0 & 1 & t_y\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 2\\ 0 & 0 & 1\end{bmatrix}$$

$$C.M.=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 2\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0.8 & -0.6 & 0\\ 0.6 & 0.8 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}0.8 & 0.6 & 0\\ -0.6 & 0.8 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & -2\\ 0 & 0 & 1\end{bmatrix}$$

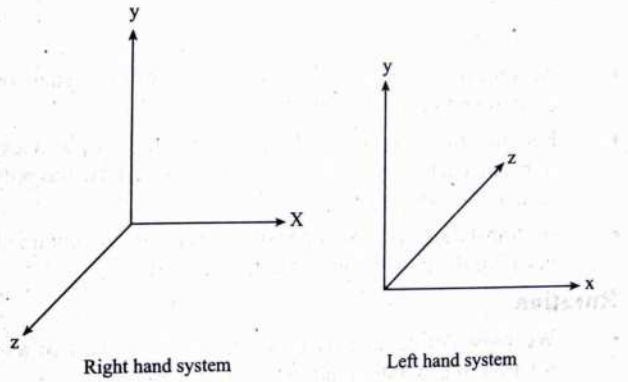
$$\begin{bmatrix}x'\\ y'\\ 1\end{bmatrix}=C.M.\begin{bmatrix}x\\ y\\ 1\end{bmatrix}$$

Chapter 4

Three-Dimensional Transformations

Three Dimensional Graphics

Three dimensional graphics uses three dimensional representations of geometric data. 3D adds the depth (z) dimension in length (x) and breadth (y) dimension in 2D. 3D is more complex than 2D because in 3D relatively more co-ordinate points are needed, object boundaries can be constructed with various combination of plane and curved surfaces, viewing direction, position in space, orientation, projection consideration, visible surface detections etc. matters in displaying the graphics.



Right hand system Left hand system

4.1 Three-Dimensional Transformations

- Translation
- Rotation
- Scaling
- Reflection
- Shear

Matrix used in 3d transformation is of order 4×4 homogeneous co-ordinate for 3D is (x, y, z, 1)