computer graphics
ASSIGNMENT - 1

Name - Shivam Singh SAP: 500098301

(1). CRT stands for cathode Ray Tube which is an electronic display device and was widely used in older televisions and monitors.

The basic working of CRT involves the use of an electron gun to emit a beam of electrons which is then focused into narrow beam and directed towards fluroscent screen. The electrons are accelerated by the electric field towards the screen, where they strike the phosphor coating, causing it to emit light. By controlling the intensity and position of the electron beam, different areas of the screen can be illuminated.

In case of a colored CRT, there are three guns of each primary color (Red, green, and blue). The beams from these guns are focused into the screen through shadow mask which ensures that the electron only strike the appropriate colored phosphors on the screen. The intensity of an electron beam is controlled by varying the voltage of the electron guns.

Overall, the CRT technology was very successful from many decades, but it was largely superseded by newer display technologies like LCD and LED.

(2). There are several algorithms used for generating a circle on a computer screen.

Those algorithm's are — Midpoint circle Algo
— Bresenham circle Algo

Both these algorithm uses 8-way symmetry of a circle. i.e., it plots 1/8th of a circle and rest of a circle is plotted by reflection.

- Bresenham Algorithm -

1. Start Algorithm
2. Read radius of the circle (r)
3. Calculate decision parameter, $d = 3 - 2r$
4. Initial starting point $(x=0, y=r) \longrightarrow (x, y)$
5. Plot $(x, y) \longrightarrow$ current point
   Next point $\longrightarrow (x_{i+1}, y_{i+1})$

   Case I: If $d < 0$, then
   $x_{i+1} = x_i + 1$
   $y_{i+1} = y_i$
   $d_{i+1} = d_i + 4x_{i+1} + 6$

   Case II: If $d \geq 0$, then
   $x_{i+1} = x_i + 1$
   $y_{i+1} = y_i - 1$
   $d_{i+1} = d_i + 4(x_{i+1} - y_{i+1}) + 10$

6. When, $x_{i+1} \geq y_{i+1}$
   Stop.

- Mid-point circle algorithm.

1. Start Algorithm
2. Read radius (r)
3. Calculate decision parameter, $d = 1 - r$
4. Initially starting point $(x=0, y=r)$, $(x, y)$
5. current point $(x, y) \rightarrow$ Plot

   Next point - $(x_{i+1}, y_{i+1})$

   Case I: If $d < 0$

   $x_{i+1} = x_i + 1$

   $y_{i+1} = y_i$

   $d_{i+1} = d_i + 2x_{i+1} + 1$

   Case II: If $d \geq 0$

   $x_{i+1} = x_i + 1$

   $y_{i+1} = y_i - 1$

   $d_{i+1} = d_i + 2x_{i+1} - 2y_{i+1} + 1$

6. when, $x_{i+1} \geq y_{i+1} \rightarrow$ stop.

(5). The rendering pipeline in opengl refers to the sequence of operations that occur during the rendering process. The pipeline is divided into several stages, most are -

- Vertex Specification - This stage specifies the vertices of the objects to be rendered. It defines the attributes associated with each vertex, such as position, normal, color, etc.
- Vertex Shader - It applies transformations to the vertices, such as scaling, rotating and translating.
- Primitive Assembly - This stage groups the vertices into primitives, such as lines, points and triangles, based on the mode specified by the user.
- Tessellation - Its optional and can be used to divide the ~~os~~ primitives into smaller pieces, useful for generating more detailed geometry.
- Geometry Shader - It processes the primitive generated in the previous stages, it can create new primitives, discard primitives or modify existing primitives.
- Clipping - Discards primitive that lie outside the view frustum.
- Rasterization - It converts the primitive into fragments, which are individual pixels on the screen.
- Fragment Shader - It process each fragment and determine its final color based on lighting calculations, texture mapping, and other effects.

- Alpha Test :- It discard the fragments that have an alpha value below a specified threshold.
- Blending :- It combines the fragments with the contents of the framebuffer based on blending equations specified by the user.
- Scissor Test - Discards fragments that lie outside a specified rectangular region.
- Stencil Test - It discards fragments based on a stencil buffer value.
- Depth Test - It discards fragments based on their depth value compared to the depth buffer.
- Output Merger - It writes the final fragments to the framebuffer.

Overall, the rendering pipeline in OpenGL is a complex sequence of stages that work together to produce the final image on the screen.

(4). end points - (20, 10) and (30, 18).

Algorithm -

1. Accept start and end points.
2. Calculate: $dx = x_n - x_0$
   $dy = y_n - y_0$
3. Calculate decision parameter
   $P_k = 2\Delta y - \Delta x$
4. Suppose -
   current point - $(x_k, y_k)$
   Next point - $(x_{k+1}, y_{k+1})$.

   To find next point, depending on decision parameter.

   $P_k = 2\Delta y - \Delta x$

| case I: If $P_k < 0$ | case II: If $P_k \geq 0$ |
|---|---|
| $P_{k+1} = P_k + 2\Delta y$ | $P_{k+1} = P_k + 2\Delta y - 2\Delta x$ |
| $x_{k+1} = x_k + 1$ | $x_{k+1} = x_k + 1$ |
| $y_{k+1} = y_k$ | $y_{k+1} = y_k + 1$ |

5. Repeat step (4). until end points is reached.

Calculation -

$dx = 30 - 20 = 10$

$dy = 18 - 10 = 8$

$P_k = 16 - 10 = 6$

Now, generating table.

and $y_0 \rightarrow (20, 10)$ is already plotted. (1)

| $p_k$ | $x_{k+1}$ | $y_{k+1}$ | $p_{k+1}$ |
|---|---|---|---|
| 6 | 21 | 11 | 2 |
| 2 | 22 | 12 | -2 |
| -2 | 23 | 12 | 14 |
| 14 | 24 | 13 | 10 |
| 10 | 25 | 14 | 6 |
| 6 | 26 | 15 | 2 |
| 2 | 27 | 16 | -2 |
| -2 | 28 | 16 | 14 |
| 10 | 29 | 17 | 6 |
| 6 | 30 | 18 | 2 |

Stop.

The points are -

(20, 10)
(21, 11)
(22, 12)
(23, 12)
(24, 13)
(25, 14)
(26, 15)
(27, 16)
(28, 16)
(29, 17)
(30, 18)

Done.

(5). To translate the square we have to add 2 in both x and y direction

$x' = x + t_x$

$y' = y + t_y$

A (0, 0) → $x' = 0 + 2 = 2$

$y' = 0 + 2 = 2$

B (3, 0) → $x' = 3 + 2 = 5$

$y' = 0 + 2 = 3$

C (3, 3) → $x' = 3 + 2 = 5$

$y' = 3 + 2 = 5$

D (0, 3) → $x' = 0 + 2 = 2$

$y' = 3 + 2 = 5$

New coordinates of the square are -

A' (2, 2), B' (5, 3), C' (5, 5) and D' (2, 5)

Now scaling -

$x' = 1.5x$

$y' = 0.5y$

A' (2, 2) → $x' = 2 \times 1.5 = 3$

$y' = 2 \times 0.5 = 1$

B' (5, 3) → $x' = 5 \times 1.5 = 7.5$

$y' = 3 \times 0.5 = 1.5$

C' (5, 5) → $x' = 5 \times 1.5 = 7.5$

$y' = 5 \times 0.5 = 2.5$

D' (2, 5) → $x' = 2 \times 1.5 = 3$

$y' = 5 \times 0.5 = 2.5$

(6). (a) about origin. , $\theta = 45°$

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

In matrix form

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

$A(0,0) \rightarrow x' = 0 \times 1/\sqrt{2} - 0 \times 1/\sqrt{2} = 0 \rightarrow (0,0)$
$y' = 0 \times 1/\sqrt{2} + 0 \times 1/\sqrt{2} = 0$

$B(2,2) \rightarrow x' = \sqrt{2} - \sqrt{2} = 0 \rightarrow (0, 2\sqrt{2})$
$y' = \sqrt{2} + \sqrt{2} = 2\sqrt{2}$

$C(4,2) \rightarrow x' = 4 \times \frac{1}{\sqrt{2}} - 2 \times \frac{1}{\sqrt{2}} = 2\sqrt{2} - \sqrt{2} = \sqrt{2}$

$y' = 4 \times \frac{1}{\sqrt{2}} + 2 \times \frac{1}{\sqrt{2}} = 3\sqrt{2} \rightarrow (\sqrt{2}, 3\sqrt{2})$

(b) about $P(-2,-2)$

We will translate it to the origin and then back to the original position.

~~$A(0,0) \rightarrow (0-(-2), 0-(-2)) = (2,2)$~~
~~$x' = 0$~~
~~$y' = 2\sqrt{2} \rightarrow (0, 2\sqrt{2})$~~

The rotation matrix is given by

$$R_{45} \cdot @P = T_V \cdot R_{45} \cdot T_{-V}$$

$$R_{45} \cdot P = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$[A'B'C'] = R_{45} \cdot P \cdot [ABC]$$

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

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

Therefore coordinates are

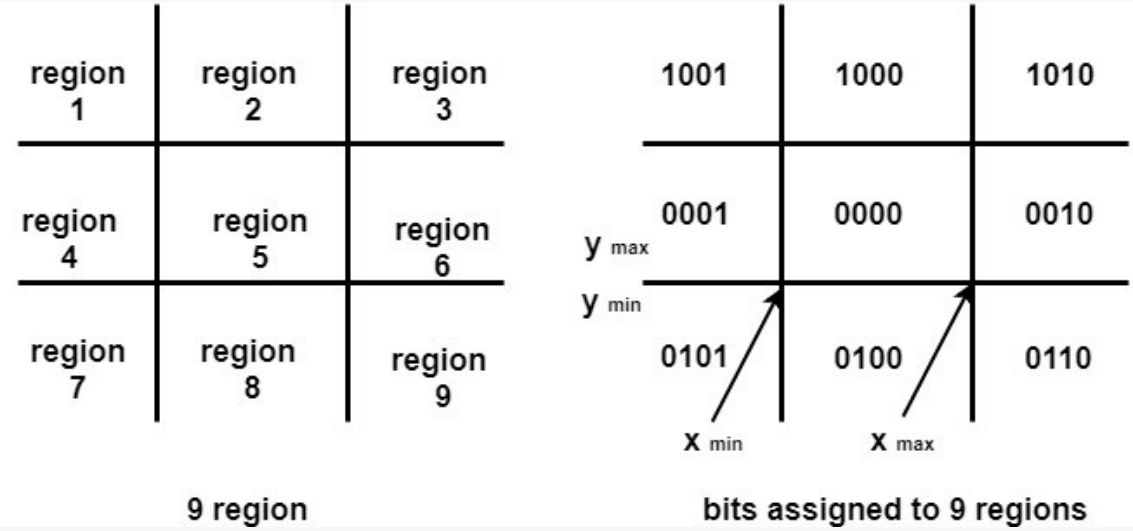
$A' \left(-2, 2\sqrt{2} - 2\right)$

$B' \left(-2, 4\sqrt{2} - 2\right)$

$C' \left(\sqrt{2} - 2, 5\sqrt{2} - 2\right)$

**7. Explain working of Cohen Sutherland Algorithm.**

Cohen-Sutherland algorithm is a line clipping algorithm used to clip a line against a rectangular viewing window or viewport. The algorithm divides the viewport into 9 regions using 4 lines, which form a rectangular area known as the clipping window.




The algorithm uses binary codes to represent the regions of the line endpoints and the clipping window. These codes are known as the outcodes and are defined as follows:

Top = 1000 Bottom = 0100 Right = 0010 Left = 0001

For each endpoint of the line, the algorithm computes its outcode by testing its position with respect to the clipping window. If the endpoint is inside the clipping window, its outcode is 0000. If it is outside the clipping window, its outcode is determined by the region in which it lies. For example, if the endpoint is above the top edge of the clipping window, its outcode will be 0001.

The algorithm then checks if both endpoints of the line lie inside the clipping window (i.e., both outcodes are 0000). If this is the case, the line is visible and is drawn in its entirety.

If one or both endpoints lie outside the clipping window, the algorithm computes the intersection points of the line with the clipping window. To do this, it uses the slope of the line to determine which edge(s) of the clipping window the line intersects. If the slope of the line is positive, it will intersect the left and/or right edges of the clipping window. If the slope is negative, it will intersect the top and/or bottom edges.

The algorithm then updates the outcodes of the endpoints based on their new positions. If an endpoint is moved outside the clipping window, its outcode is updated accordingly. The algorithm then repeats the above steps until both endpoints are inside the clipping window or the line is determined to be completely outside the clipping window (i.e., both outcodes have a common bit set).

Once the line has been clipped, the visible portion of the line is drawn using any line-drawing algorithm, such as Bresenham's algorithm.

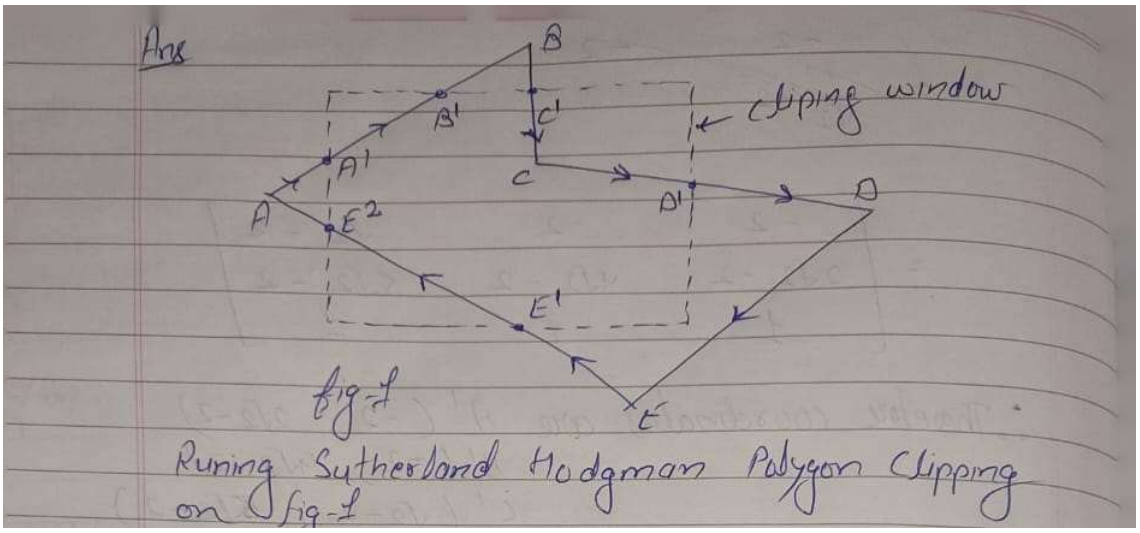
Overall, the Cohen-Sutherland algorithm is a simple and efficient way to clip lines against a rectangular viewport, and it forms the basis for more advanced clipping algorithms such as the Sutherland-Hodgman algorithm.

**8. Using Sutherland Hodgman Polygon Clipping Algorithm clip the following polygon.**

Ans- It is performed by processing the boundary of polygon against each window corner or edge. First of all entire polygon is clipped against one edge, then resulting polygon is considered, then the polygon is considered against the second edge, so on for all four edges.

Four possible situations while processing

- If the first vertex is an outside the window, the second vertex is inside the window. Then second vertex is added to the output list. The point of intersection of window boundary and polygon side (edge) is also added to the output line.
- If both vertexes are inside window boundary. Then only second vertex is added to the output list.
- If the first vertex is inside the window and second is an outside window. The edge which intersects with window is added to output list.
- If both vertices are the outside window, then nothing is added to output list.



Ans

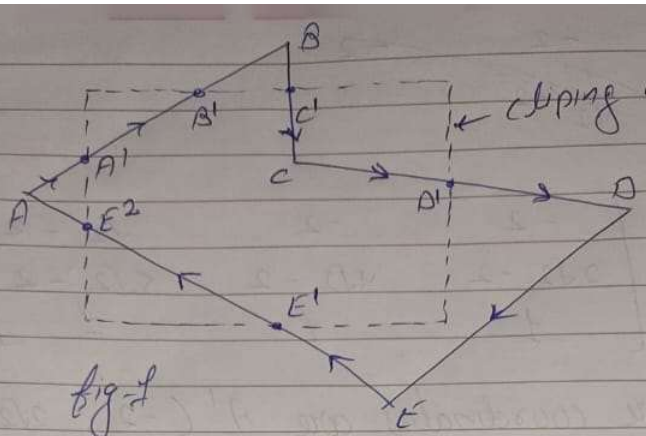



fig-1

Runing Sutherland Hodgman Polygon Clipping on fig-1

List → vertices of inside and intersect

① Outside → inside
A' save

② inside → outside
B' save

③ outside → inside
C' save

④ ~~outside~~ inside → outside
D' save

⑤ outside → inside
E' save

⑥ inside → outside
E² save

List [A' | B' | C' | D' | E' | E²]

Then after cliping the ~~fig~~ figure will appeare

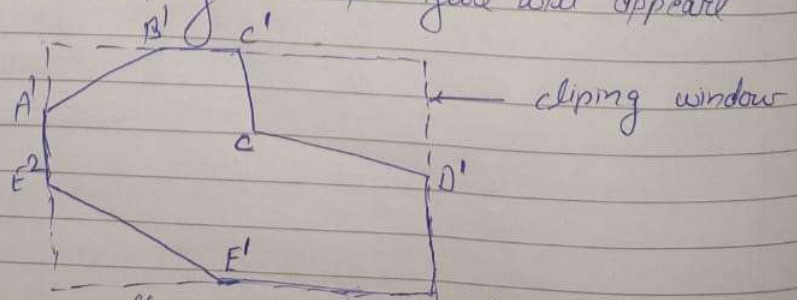



Image after cliping the window