C Bhavesh Kumar
S20190010034

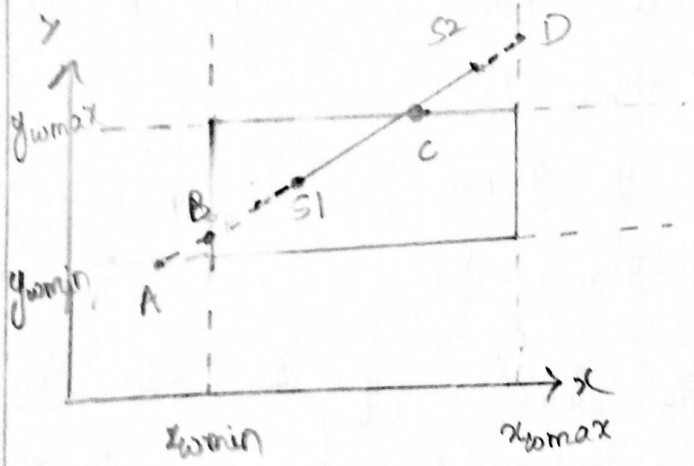
5/12/21

CGiM END EXAM

Question 2

a) Clipping is to display only the region of interest which is defined by a rectangle ($x_{min}, x_{max}, y_{min}, y_{max}$)

Liang-Barsky line clipping algorithm is used to clip line segments. It is a faster line clipping algorithm and is based on parametric line equations.



consider the following example, the main idea of Liang-Barsky is to find nearest point to S_2 from A, B, S_1 (S_1 in Example) and to find nearest point to S_1 from C, S_2, D (C in example).

we consider the parametric equations using which we decide whether a given line segment is visible inside a clipping window or not and if visible what part of it is visible.

$x = x_1 + u\Delta x$ where $S_1(x_1, y_1)$, $S_2(x_2, y_2)$

$y = y_1 + u\Delta y$ $\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$, $u \in [0, 1]$

Point clipping condition in parametric form is given by

$xw_{min} \leq x_1 + u\Delta x \leq xw_{max}$, $yw_{min} \leq y_1 + u\Delta y \leq yw_{max}$

$\Rightarrow$ $u \times (-\Delta x) \leq x_1 - xw_{min}$

$u \times (\Delta x) \leq xw_{max} - x_1$

$u \times (-\Delta y) \leq y_1 - yw_{min}$

$u \times (\Delta y) \leq yw_{max} - y_1$

In another form we can write this as,

$up_k \leq q_k$ where,

$p_1 = -\Delta x$ $\quad q_1 = x_1 - xw_{min}$ and $k = 1, 2, 3, 4$

$p_2 = \Delta x$ $\quad q_2 = xw_{max} - x_1$

$p_3 = -\Delta y$ $\quad q_3 = y_1 - yw_{min}$

$p_4 = \Delta y$ $\quad q_4 = yw_{max} - y_1$

we need to decide the direction of the given line segment to find the starting and ending edge. we do it based on Δx, Δy.

	Starting edge	Ending Edge
$\Delta x >= 0$	x_{min}	x_{max}
$\Delta x < 0$	x_{max}	x_{min}
$\Delta y >= 0$	y_{min}	y_{max}
$\Delta y < 0$	y_{max}	y_{min}

If $p_k = 0$ for $k = 1, 2$, the line segment is parallel to the clipping edge and

a) If $q_k < 0$ $(k=1,2)$ the line is completely outside of clipping region and we discard it.

b) If $q_k > 0$ $(k=1,2)$ line segment is visible and we get intersections with windows edges and decide the visible part.

Similarly for $p_k = 0$ given $k = 3, 4$.

The parameters of intersections of S_1S_2 with the starting edges are referred to as u_1', u_2'' and

$u_1 = \max\{u_1', u_1'', 0\}$

where u_1 is the nearest clipping point to S_2.

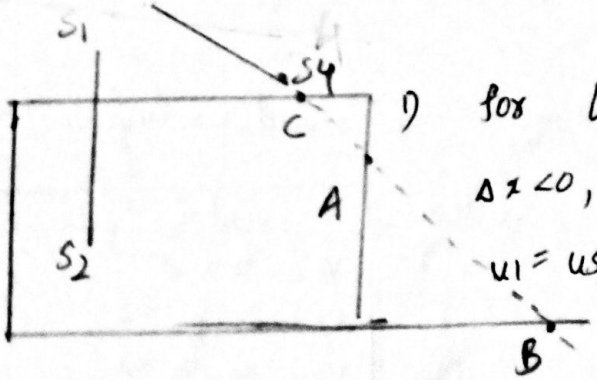
Similarly S_1S_2 intersection with ending edges are referred to as u_2', u_2'' and $u_2 = \min\{u_2', u_2'', 0\}$

1. If $u_1 < u_2$, $x = x_1 + \Delta x \times u$
   $y = y_1 + \Delta y \times u$ $(u_1 < u < u_2)$

   this defines the visible part.

2. If $u_1 > u_2$ S_3S_4 then the line is invisible.

Example



1) for line S_3, S_4.

$\Delta x < 0$, $\Delta y < 0 \Rightarrow$ starting $= (x_{wmax}, y_{wmin})$

$u_1 = u_{S3} = 0$ and $u_2 = u_c < 0$

$\therefore u_1 > u_2$

S_3S_4 is invisible.

2) for the line S1S2, $P_1 = P_2 = 0$ but $q_1 > 0, q_2 > 0$

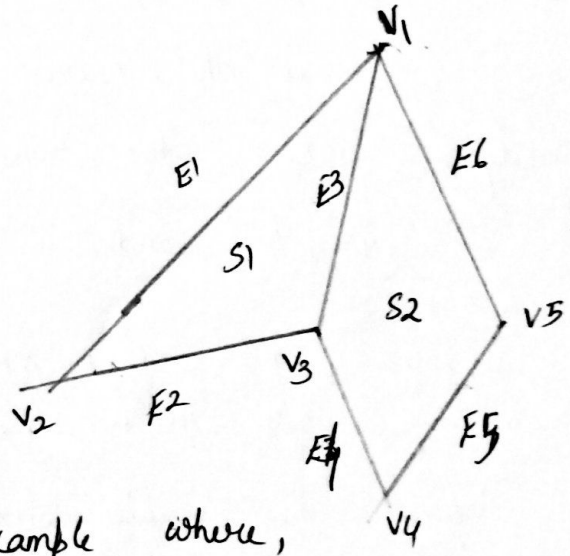
∴ the line is visible and we find out the visible part by considering the intersections.

b) Polygon surfaces ~~are the surfaces~~ are the surfaces that are used for Boundary representation of a 3D object.

Polygon tables are used to represent polygon surfaces.

The Polygon tables contains three tables.

1) vertex table
2) Edge table
3) Polygon - surface table



* consider the following example where, the polygon consists of two surfaces. S1, S2.

* The vertex table stores the vertex information, i.e, x, y, z co-ordinates for each vertex.

vertex table
V1 : x_1, y_1, z_1
V2 : x_2, y_2, z_2
V3 : x_3, y_3, z_3
V4 : x_4, y_4, z_4
V5 : x_5, y_5, z_5

* Edge table is used to store information of the edges in terms of vertices.

Edge table
E1 : V1, V2
E2 : V2, V3
E3 : V3, V1
E4 : V3, V4
E5 : V4, V5
E6 : V5, V1

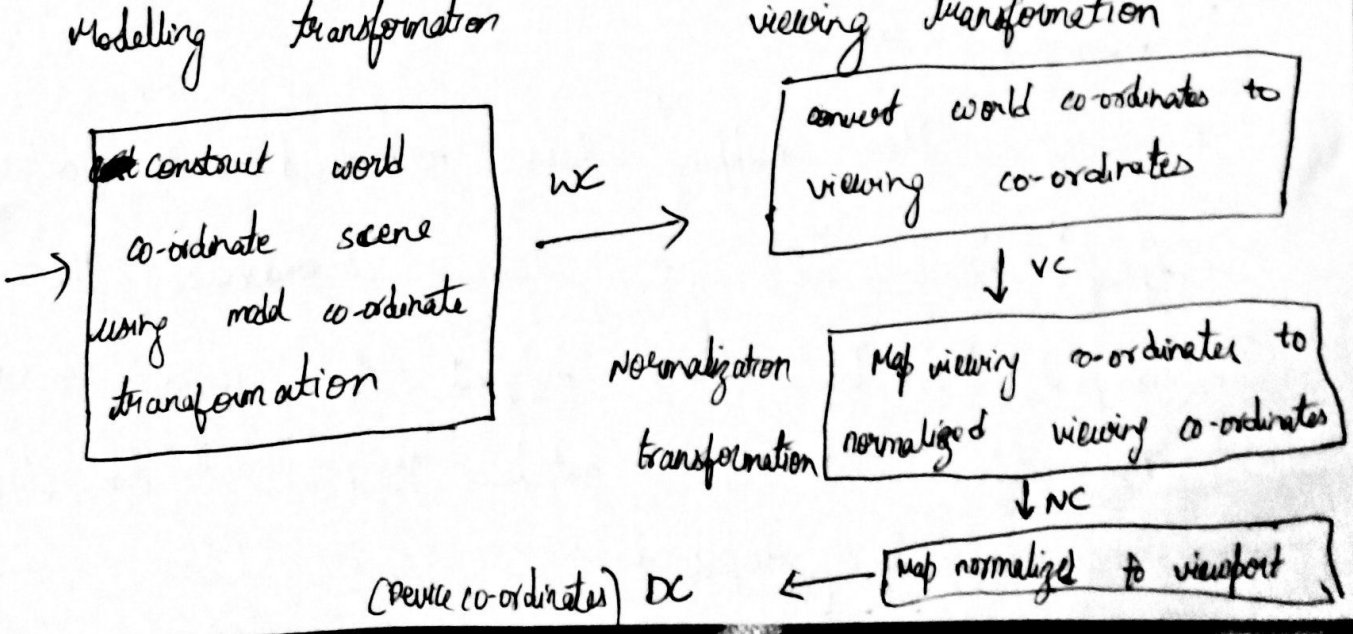
* The polygon surface tables which describes the surface stores the surface information for each surface present in the polygon.

* In the given example, S1 is covered by Edges E1, E2 and E3 and hence is represented as S1: E1, E2, E3.

Polygon surface table
S1 : E1, E2, E3
S2 : E3, E4, E5, E6

Question 1

a) 2D viewing pipeline



* In the first component, the co-ordinates in which individual objects are created are called model co-ordinates when several objects are assembled into a single scene, they are described by world co-ordinates. so, we set up the co-ordinates of a scene to display.
* In the second component, viewing transformation is done in which world co-ordinates which we created earlier are transformed into the co-ordinate system of camera and viewer. the world co-ordinates become viewing co-ordinates (VC).
* In the third step, these viewing co-ordinates are normalized to get normalized viewing co-ordinates (NC). This is done to ensure that the viewing process is independent of any output devices. Clipping is generally done for normalised viewing co-ordinates.
* finally, after mapping these normalized co-ordinates to a specific device we get device co-ordinates. The normalised viewport co-ordinates are mapped to device co-ordinates. The objects that lie outside viewport will be clipped and will not be displayed.

b)
```
glMatrixMode ( GL_MODELVIEW ); // defining matrix mode.
glLoadIdentity (); // loads identity matrix
glColor3f (0.0, 0.0, 1.0); // color set to blue.
glRecti (50, 100, 200, 150); // displaying blue rectangle
glColor3f ( 0.0, 1.0, 0.0) ; // color set to green
glRotatef (45.0, 0.0, 0.0, 1.0); // rotate 45° around z-axis
glRecti ( 300, 100, 450, 150) ; // display green rotated rectangle
                                   side by side.
```

Question 3

a) There are many visible surface detection methods one of the methods is Depth (z) buffer method. To represent a non-transparent object on 2D plane to make it realistic we need to remove the non-visible planes from diagram. Every point on polygon (3D) is represented on a 2D new plane; each point is compared with depth (z) values.

We will use normalized z values (0,1). Here we use two buffer areas Depth (z) buffer, Refresh Buffer.

Depth buffer contains depth value of each (x, y) position

Refresh buffer contains intensity value at each (x, y) position

Steps:-

1) Initialising Intensity and depth values

$\forall (x,y)$ depth $(x, y) = z_{max}$ $(z_{max} = 1)$

Refresh $(x, y) = I_B$ (I_B is Background Colour)

2) calculating on every portion of polygon surface.

a) calculate z-depth for each position (x, y) of the polygon.

b) Here we will consider least value so that upper or closer is visible to us.

If $z <$ depth(x,y) then:

depth $(x, y) = z$

refresh $(x, y) = I_S(x, y)$

where I_S is the projection intensity value of the surface at (x, y) which has minimal value of z at current current iteration

$$z = \frac{-Ax - By - D}{C}$$ for plane

$Ax + By + Cz + D = 0$

b) 3D Geometric transformations

i) 3D translation: same as 2D, add z-axis and z-co-ordinate

Homogeneous co-ordinates contains four co-ordinates and 4×4 matrix is multiplied to apply transformations.

A position $P = (x, y, z)$ in 3D is translated to a location $P' = (x', y', z')$ by adding translation distances t_x, t_y, t_z.

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

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

$$P' = TP.$$

(ii) 3D rotation.

co-ordinate axes rotation - x axis, y axis, z axis rotation.

Rotation about an axis that is is parallel to one of co-ordinate axis.

General 3D rotation

Rotation about an arbitrary axis.

→ translate P to origin.

→ Rotate so the arbitrary axis is aligned with one of co-ordinate axes and perform desired rotation and rotate back.

3D Scaling - scale objects relative to (0, 0, 0)

All vectors are ~~scaled~~ scaled from origin.

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