CGV Assignment

Md. Kaif
1BY20CS107

1) 2D - Pipeline

Construct world coo-ordinate using modeling co-ordinate transformation

Convert world co-ordinates to viewing co-ordinates

Transform viewing co-ordinates to normalized co-ordinates

Map normalized co-ordinates to device co-ordinates

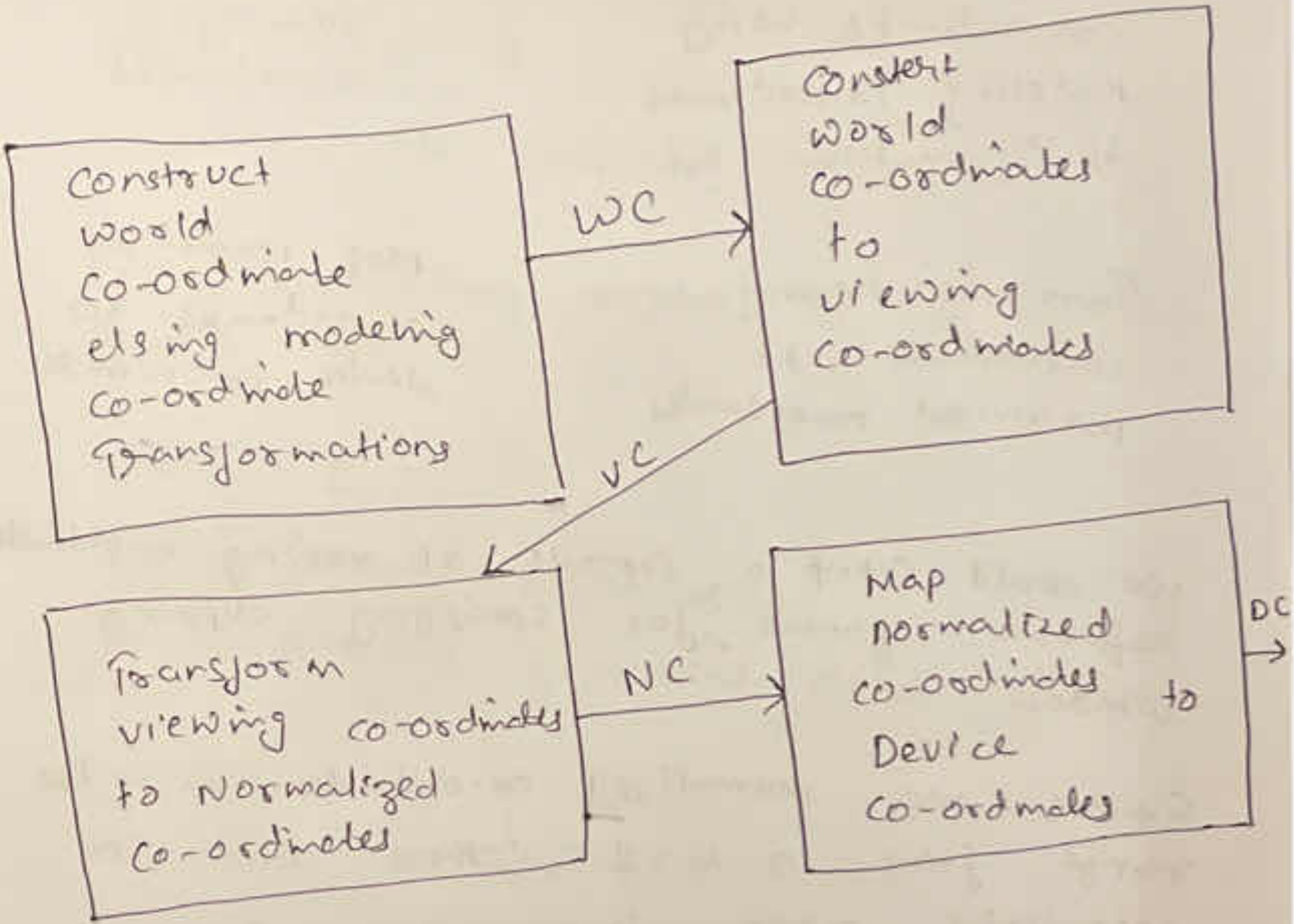
we could setup a seperate 2D viewing co-ordinate reference frame for specifying clipping window.

System use normalized co-ordinates in the range from 0 to 1, others used a normalized range from -1 to 1

clipping is usually performed in the normalized - co-ordinates

Before we select a clipping window and a viewport in open GL we need to establish the appropriate mode for constructing the matrix to transform from world - co-ordinates to screen co-ordinates

gl Matrix Mode (GL-PROJECTION)



2) Build phong lighlighting model with equations height consists of 3 different types of light.

Ambient lighting is reffered to as the natural lighting

diffusion The artificial light

Specular lighting referes to the shininess of the object.

$I_{amb} = La\, Za$

La = ambient reflectivity

Ja = intensity of ambient light

similarly

$I_{diff} = kd\, I_P \cos(\theta) \rightarrow (2)$

$= Kd\, 2p(N \cdot L)$

$= K I_s \cos^n(\theta)$

∴ The phong model gives us the eqn of all combined

Total intensity

$$I = K_a I_a + K_d I_p \cos\theta + K_s I_s \cos^n \phi$$

3.) Apply homogeneous co-ordinates for translating rotating and scaling via matrix representation

The matrix representation of translating, rotation and scaling are:

$P' = P + T$

$$\text{Translation } P' = \overset{P}{\begin{bmatrix} x \\ y \end{bmatrix}} + \overset{T}{\begin{bmatrix} +x \\ +y \end{bmatrix}}$$

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

$$\text{scaling } P' = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

generic equation = S · P

$P' = M_i * P + M_2$

13rd $x = \frac{xh}{h}$ $\quad y = \frac{yh}{h}$

Rotation $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

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

4) Difference between Roses and Randon Scan displays

Roster Scan	Randon Scan
• Produces jagged lines that are plated as a discrete point sets	Random system produces small lines drawing
Less expensive	more expensive
Modification diddicult	Modification easy
Resolution low	Resolution high
Solid pattern is easy to dill	Solid patton is diddicult to fill

5) Demonstrate OpenGL functions for window management using GLUT

* glutCreate window → used to create a new window

* glut Set window → used to set a particular id for the window

* glutSet window → used to get window ID.

* glut Reshape window → used for transformation of world co-ordinates to view co-ordinates and displaying it

* glut Hide window → To hide the window from being displayed or seen

* glut Main loop()
* init () .

6) Open GL visibility Detection function

* glut Create Sub window → used to create another window within the same window.

a Open GL polygon calling functions

Remove backface, frontface of an object

gl cullface (mode) ;

glEnable (GL_CULL - FACE) ;

gl Disable (GL - CULL - FACE) ;

b. Depth buffer function

glutInit display mode (GLUT _SINGLE) GLUT - DEPTH)

glclear (GL - DEPTH - BUFFER - BIT)

This works as initializating function for depth buffer and refresh buffer.

glDepth Range (near Norm Depth , for Norm Depth)

gl clear Depth (max Depth)

gl Disble (GL-DEPTH -TEST)

c. OpenGL window wire frame surface visibility methods glPolygonMode (GL_FRONT_AND BACK. GL-LINE)

Visible and hidden edges displayed.

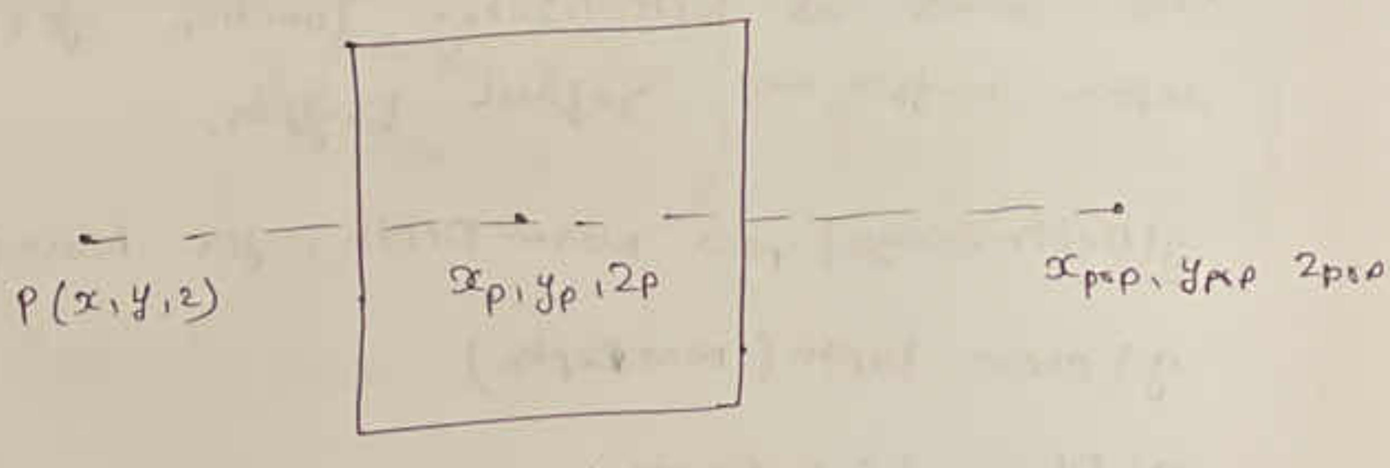
d. Open GL Depth curving function

glFogi (GL_FOG_MODE, GL_UNCLEAR)

glEnable (GL_FOG)

To increase or decrease the brightness

7) Write special cases discussed with perspective projection.



Consider

$$x' = x - (x - x_{prp})\,u$$

$$y' = y - (y - y_{prp})\,u$$

$$z' = z - (z - z_{prp})\,u$$

$$u = \frac{z_p - z}{z_{prp} - z}$$

$$x_p = x\left[\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right] + x_{prp}\left[\frac{z_{vp} - z}{z_{prp} - z}\right]$$

$$y_p = y\left[\frac{x_{prp} - z_{vp}}{z_{prp} - z}\right] + y_{prp}\left[\frac{z_{vp} - z}{z_{prp} - z}\right]$$

Special cases :-

1. $x_{prp} \cdot y_{prp} = 0$

$$x_p = x\left[\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right]$$

$$y_p = y\left[\frac{z_{prp} - z_{vp}}{z_{pvp} - z}\right]$$

2. $x_{prp}, y_{prp}, z_{prp} = (0,0,0)$

$$x_p = x\left(\frac{x_{vp}}{z}\right)$$

$$y_p = y\left(\frac{z_{vp}}{z}\right)$$

3. $z_{vp} = 0$

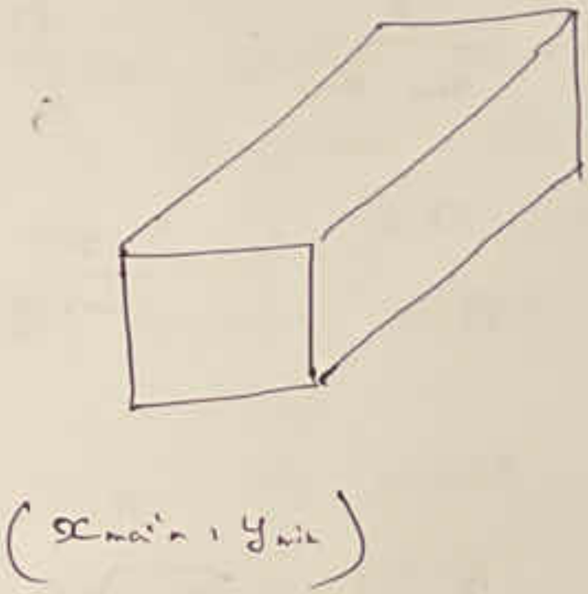
$$x_p = x\left[\frac{z_{prp}}{z_{prp} - z}\right] - x_{prp}\left(\frac{z}{z_{prp} - z}\right)$$

$$y_p = y\left[\frac{z_{prp}}{z_{prp} - z}\right] - y_{prp}\left[\frac{z}{z_{prp} - z}\right]$$

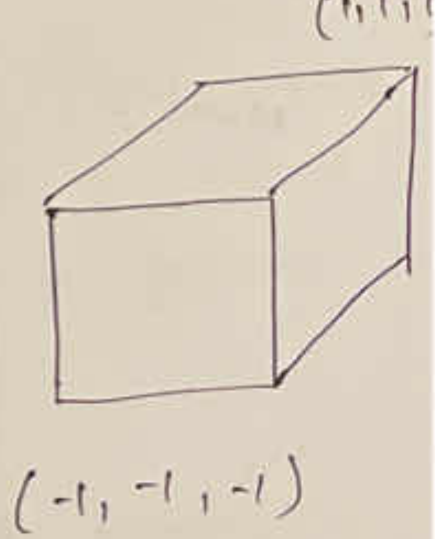
4. $x_{prp} = y_{prp} = z_{prp} = 0$

$$x_p = \left[\frac{z_{prp}}{z_{prp} - z}\right] \quad y_p = \left[\frac{z_{prp}}{z_{prp} - z}\right]$$

8) Explain normalization for an orthogonal projection.



orthogonal projection view volume



Normalized view volume.

* we consider a unit cube for this normalized view volume with each x, y, z co-ordinates normalised in the range 0 to 1

* Another normalization transformation approach is to used symmentric cube with co-ordinates -1 to 1.

∴ we get the normalization transformation for the orthogonal view volume.

$$M_{ort} = \begin{bmatrix} \frac{2}{x_{max} - x_{min}} & \frac{2}{y_{max} - y_{min}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{-2}{z_{max} - z_{far}} \\ 0 & & \end{bmatrix}$$

$$\begin{bmatrix} \frac{-x_{wmax} + x_{min}}{x_{wmax} + x_{min}} \\ \frac{-y_{max} + y_{min}}{y_{max} - y_{min}} \\ \frac{z_{max} + z_{far}}{z_{near} - z_{far}} \end{bmatrix}$$

9) Explain Bezier curve and its properties with equation

* Bezier curves are parameteric curves that are generated with the help of control points. It is widely used in graphics and other related industry.

* They are named after the french engineer pierre Bezier who discovered it

Bezier curves are represented as,

$$\sum_{k=0} P_i B_i^n (t)$$

$B_i^n(t)$ represents Brenstein Polynomial.

$$B_i^n(t) = \begin{bmatrix} n \\ i \end{bmatrix} (1-t)^{n-t} \; t^i$$

n - polynomial degree

t - variable

i - index.

They are of 2 - central points : linear curve
3 - central points : cubic curve
4 - central points : quadraatic curve

we used the above mentioned joomed of Bezier curve $= ({}^{n}C_{0}) * (1-t)^{n-t} * t^{i} *$ for every point

n = central points number - 1

t = 0 - 1 (Range)

10) Cohen Sutherland line clipping Algorithm

Cohen Sutherland Algorithm works on region code

Region code is a 4 - bit code.

(A B R L)

(T B R L) - Top Bottom Right Left

1001	1000	1010
0001	0000	0010
0101	0100	0110

For a line (x_0, y_0) to (x_{end}, y_{end})

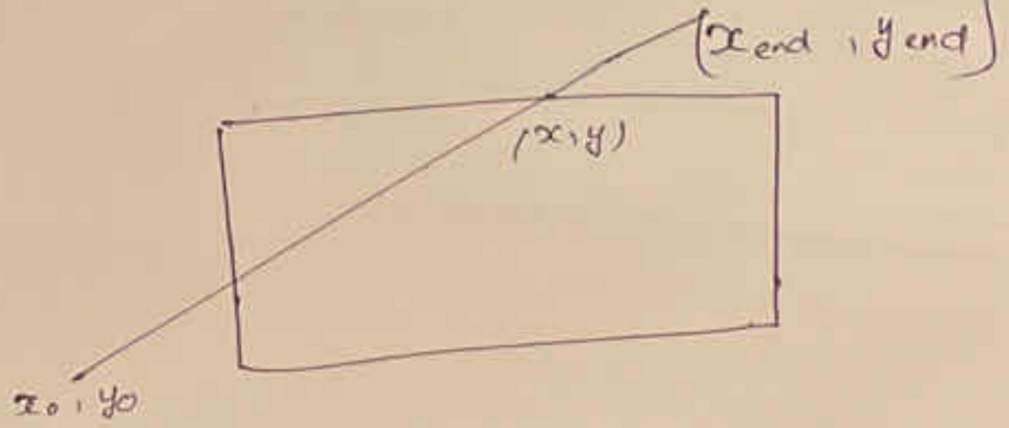
$$m = (y - y_0)(x - x_0)$$

$$m(x - x_0) = (y - y_0)$$

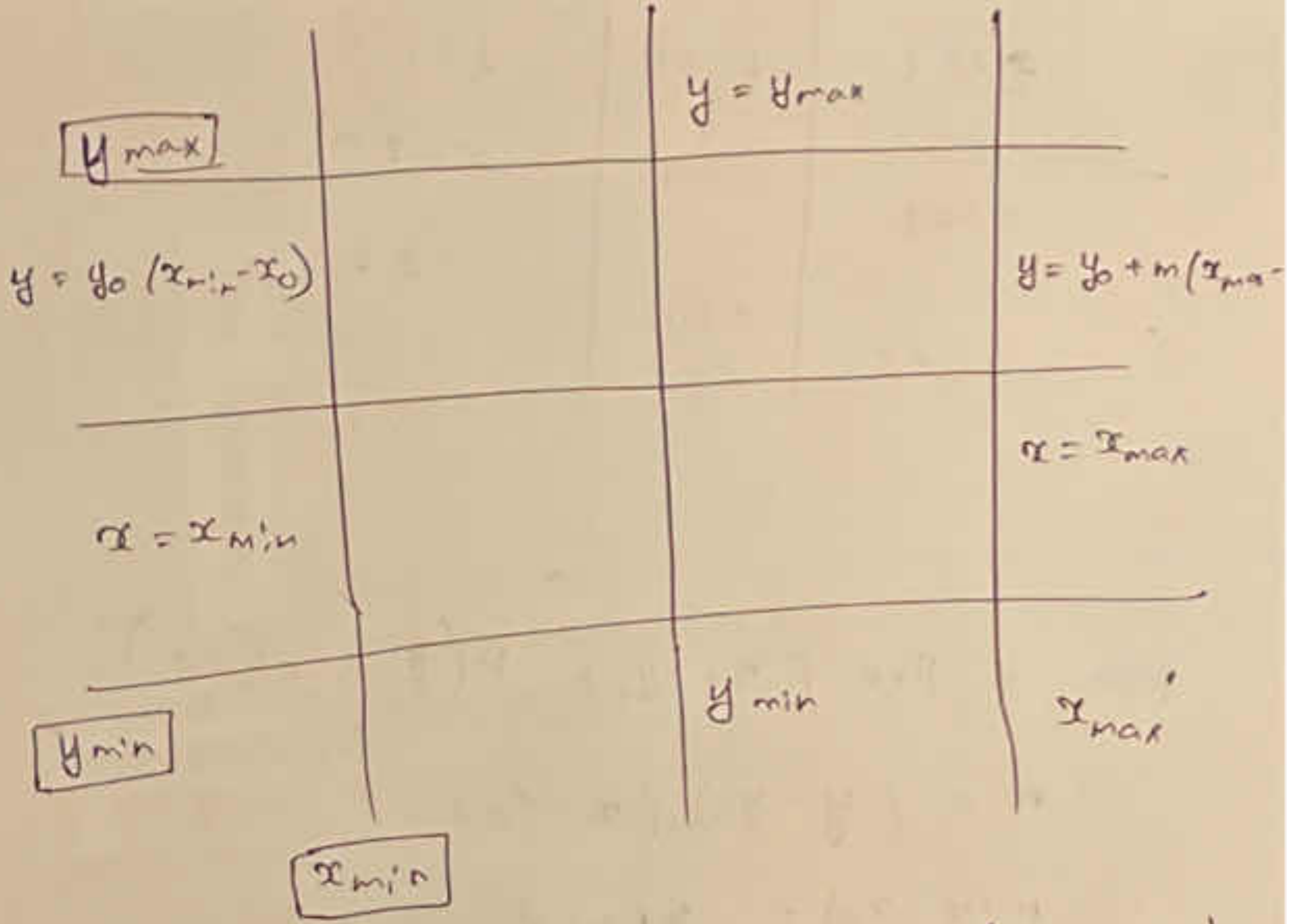
$$x - x_0 = \frac{(y - y_0)}{m}$$

$$x = x_0 + \frac{(y - y_0)}{m}$$

$$y = y_0 + m(x - x_0)$$



$$x = x_0 + y_m (y_{max} - y_0)$$



$$x = x_0 + \frac{1}{m}(y_{min} - y_0)$$

Thus the above formulae to be applied when a particular pipe needs to be clipped.