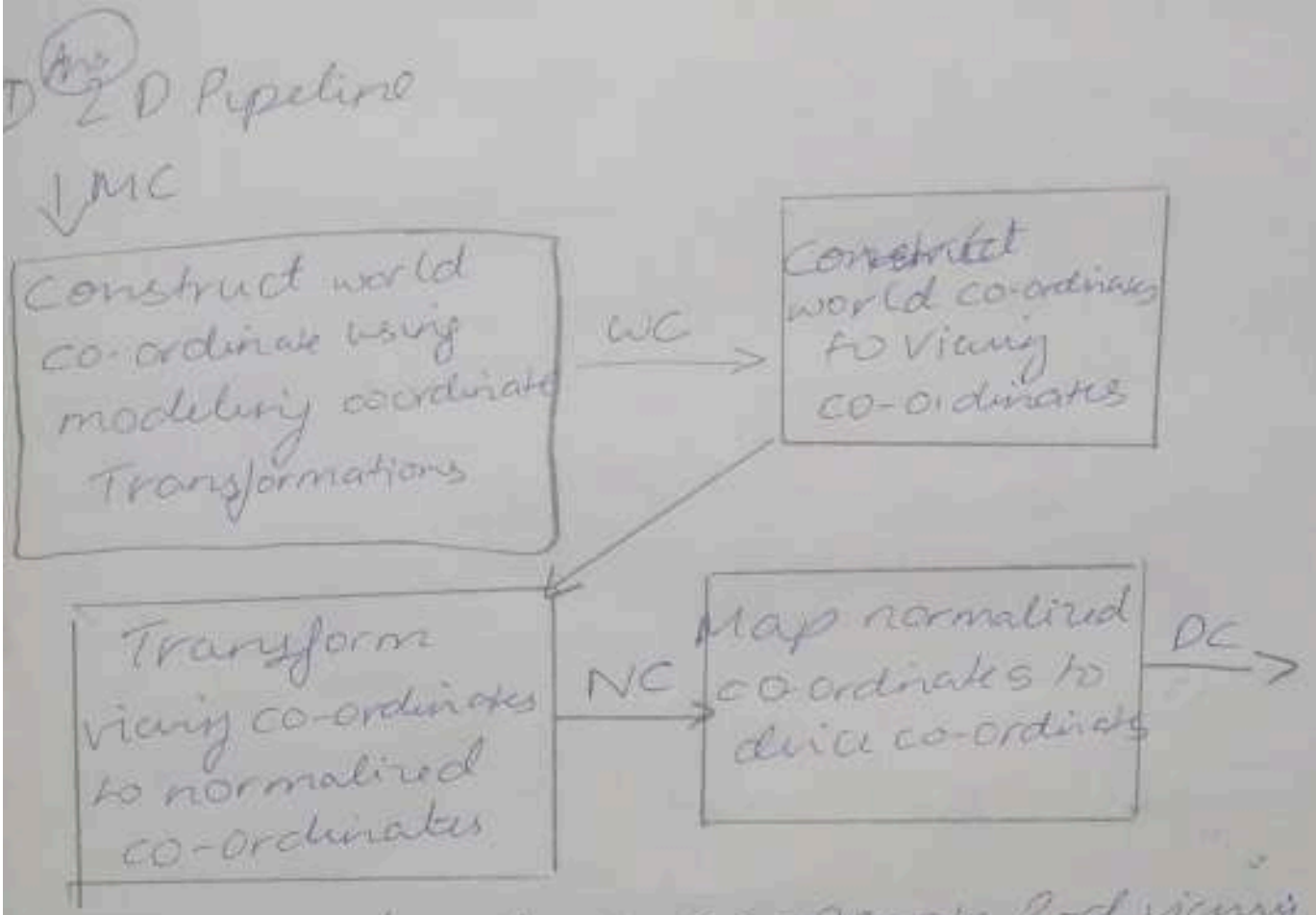
KUNAAL SHIVAKUMAR 1BY20CS092
CGV-ASSIGNMENT 21-B

1) 2D Pipeline



- We could set up a separate 2-d viewing co-ordinate reference frame for specifying clipping window.
- Systems 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.

(2) Build Phong lighting Model with Equations
Light consists of 3 different types of light

(Ans) Ambient lighting refered as the natural lighting.

→ Diffusion → The artificial light

→ Specular lighting - Refers to the shininess of the object

$I_{amb} = I_a R_a$ — (1)

R_a = ambient reflectivity

I_a = intensity of ambient light

Similarly

$I_{diff} = I_d I_p \cos(\theta)$ — (2)

$= K_d I_p (N.L)$

$I_{spec} = K_s I_l \cos^n \phi$

∴ The phong Model gives us the equation of all combined.

Total Intensity $I = R_a I_a + K_d I_p \cos\theta + K_s I_l \cos^n \phi$

(2) Apply Homogenous co-ordinates for translation, rotation and scaling via matrix representation

(Ans) The Matrix Representations of Translation, Rotation & Scaling

$P' = P + T$

P T

↓

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

R

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

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

Generic eqn $\rightarrow S \cdot P$

$P' = M_1 * P + M_2$

But $x = \frac{xh}{h}$, $y = \frac{yh}{h}$

$h = 1$

$\therefore$ consider $(h * x, h * y, h)$

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

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

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) Differences between Raster and Random scan displays

Raster Scan	Random Scan
1) Produces jagged lines that are plotted as discrete point sets • Less expensive • Modification difficult • Resolution low • Solid pattern is easy to fill.	• Random System produces smooth lines drawing • More expensive • Modification easy • Resolution high • Solid pattern is difficult to fill.

(5) Demonstrate open GL functions for window management using GLUT.

Ans) glutCreateWindow → used to create another window within same window

• glutCreateSubWindow → used to create another window within new window

• glutSetWindow - used to set on particular id for the window

- glutGetWindow - Used to get the window ID.
- glutDestroyWindow - To delete the window that was created
- glutPostRedisplay - To display the window again & again, continously until forcibly closed.
- glutReshapeWindow - Used for transformation of world co-ordinates to view co-ordinates and displaying it.
- glutFullScreen → To represent window in full screen.
- glutPopWindow / glutPushWindow - Works just like a matrix in window.
- glutHideWindow - To hide the window from been displayed on screen.
- glutDisplayFunc - To display
- glutMainLoop() -
- init()

⑥ OpenGL visibility detection functions

(a) OpenGL polygon culling functions

Remove backface, frontface of an object

```
glCullFace(mode);
glEnable(GL_CULL_FACE);
glDisable(GL_CULL_FACE);
```

(b) Depth Buffer Functions

```
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB |
GLUT_DEPTH);
glClear(GL_DEPTH_BUFFER_BIT);
```

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

```
glDepthRange(nearNormDepth, farNormDepth);
glClear(GL_DEPTH_BUFFER_BIT);
glClearDepth(maxDepth);
glEnable(GL_DEPTH_TEST);
glDisable(GL_DEPTH_TEST);
```

(c) OpenGL wireframe surface Visibility Methods

```
glPolygonMode(GL_FRONT_BACK, GL_LINE);
```

visible & hidden edges displayed

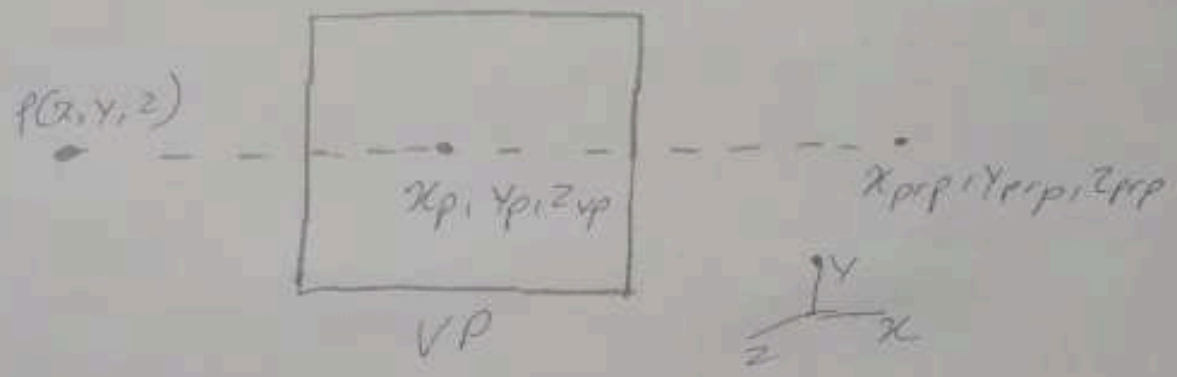
(d) OpenGL Depth cueing Fns

```
glFogi(GL_FOG_MODE, GL_LINEAR);
glEnable(GL_FOG);
```

To increase decrease brightness

(7) Write special cases discussed with perspective projections.

(Ans)



Consider,

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

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

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

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

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

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) + y_{prp}\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$

Special cases

(1) $x_{prp}, y_{prp} = 0$

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

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

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

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

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

(3) $z_{vp} = 0$

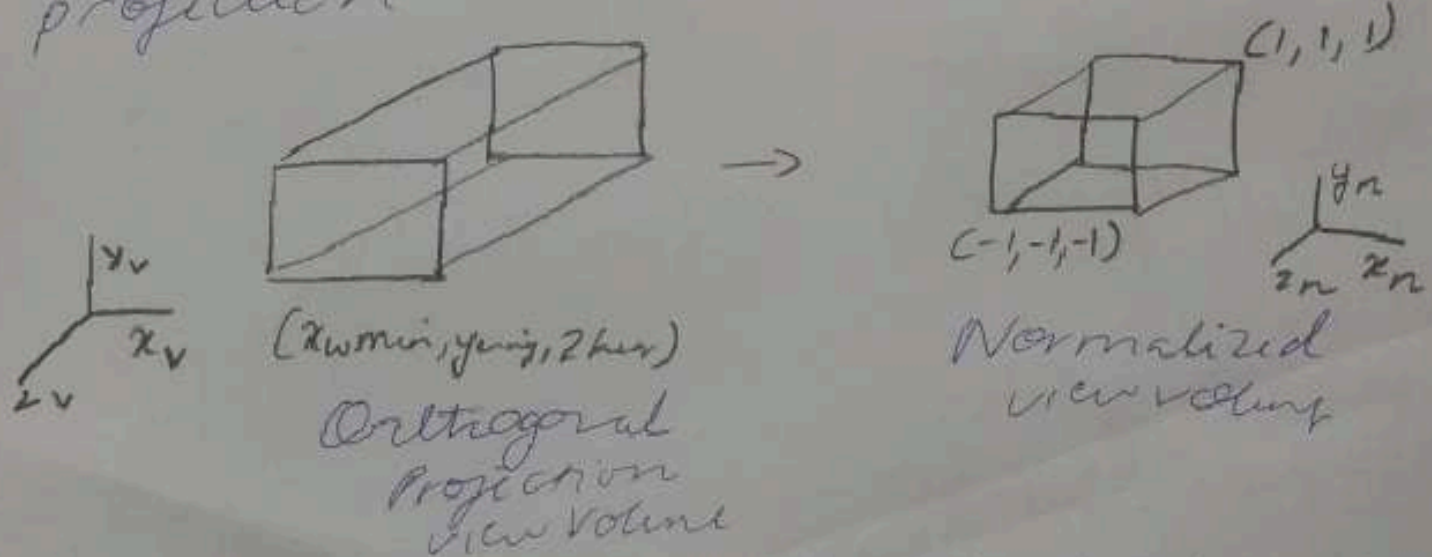
$$x_p = x\left(\frac{z_{prp}}{z_{prp} - z}\right) - x_{prp}\left(\frac{z}{z_{prp} - z}\right)$$

$$y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right) - y_{prp}\left(\frac{z}{z_{prp} - z}\right)$$

(4) $x_{prp} = y_{prp} = z_{vp} = 0$

$$x_p = x\left(\frac{z_{prp}}{z_{prp} - z}\right), \quad y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right)$$

(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 normalized in range 0 to 1

Another normalization transformation approach is to use symmetric cube co-ordinates -1 to 1.

∴ We get normalization transformation for orthogonal view volume

$$M_{ortho,norm} = \begin{bmatrix} \frac{2}{x_{wmax}-x_{wmin}} & 0 & 0 & -\frac{x_{wmax}+x_{wmin}}{x_{wmax}-x_{wmin}} \\ 0 & \frac{2}{y_{wmax}-y_{wmin}} & 0 & -\frac{y_{wmax}+y_{wmin}}{y_{wmax}-y_{wmin}} \\ 0 & 0 & -\frac{2}{z_{near}-z_{far}} & \frac{z_{near}+z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(9) Explain Bezier curves with eqns

- Bezier curves are parametric curves that are generated with help of control pts
It is widely used in graphics and other related industry.
- They are named after French engineer Pierre Bezier who discovered it

Bezier curves are represented as

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

$B_i^n(t)$ represent Bernstein Polynomial

n -> polynomial degree
t -> variable
i -> index

They are be of 2 - control points - linear curve
3 - control points - cubic curve
4 - control points - quadratic curve

We used the above mentioned formulas are :-

Bezier curve = $\left(n C_i\right) t^i (1-t)^{n-t}$ * for every $p \in$

n = control pts - 1
t = 0 - 1 (Range)

(10) Cohen Sutherland line clipping algorithm

- Cohen Sutherland algorithm works on Region code
- Region code is 4bit code
  (A B R L)
  (T B R L) - TOP, Bottom, Right, Left

1001	1000	1010
0001	0000	0010
0101	0100	0110

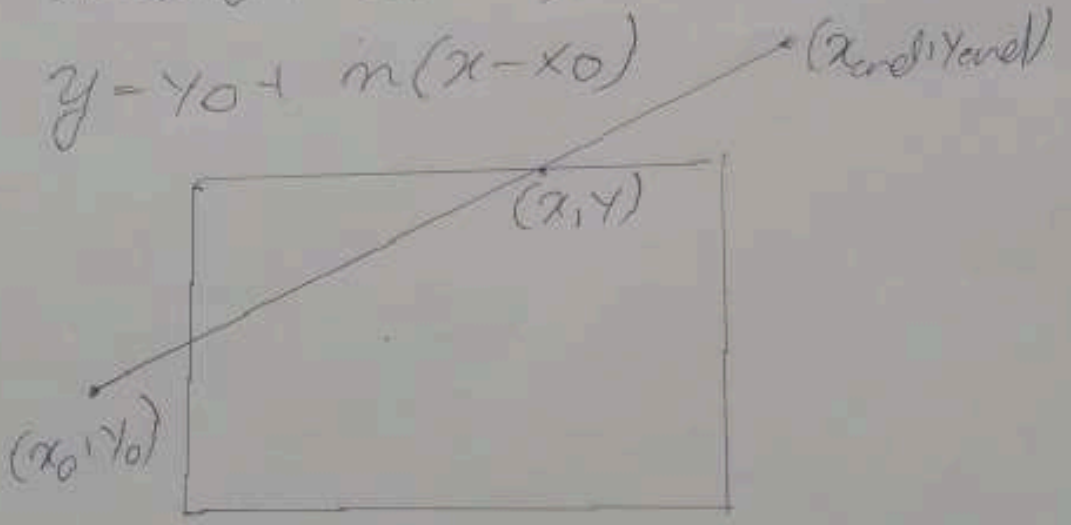
For a line - (x_0, y_0) to (x_{end}, y_{end})

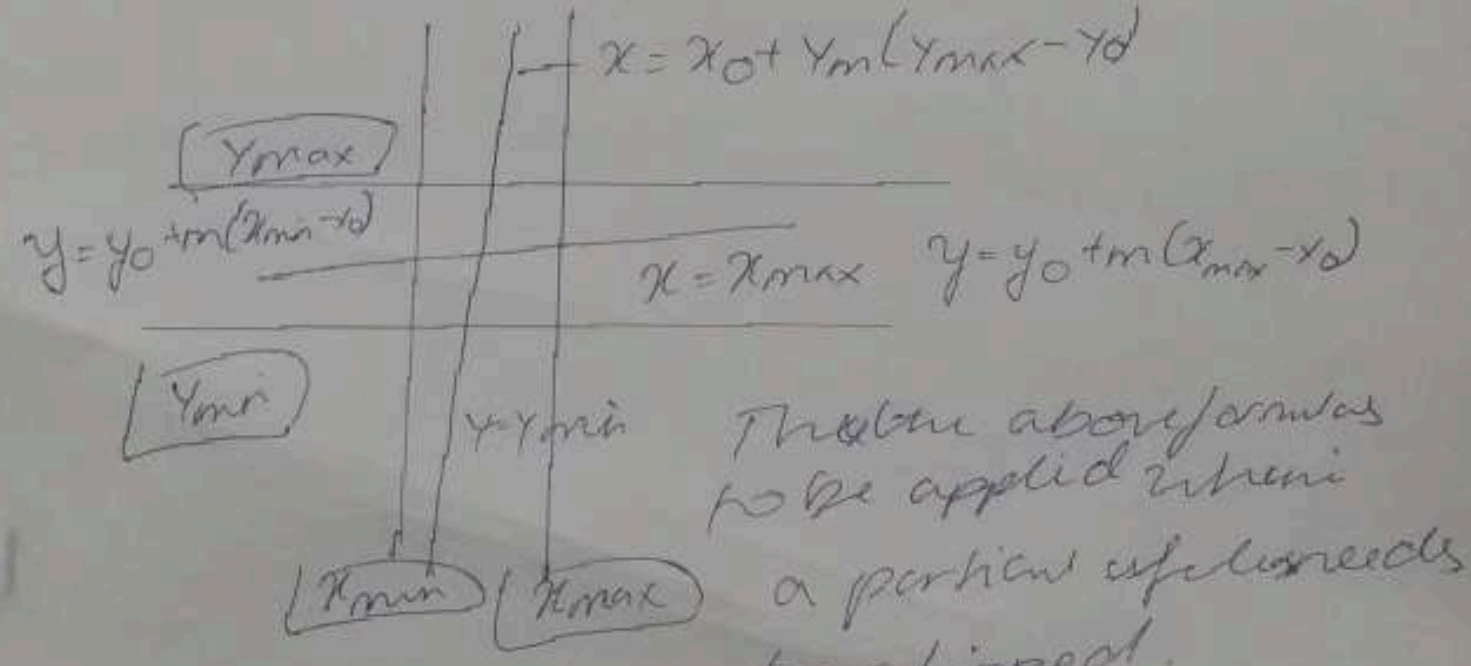
$$m = (y - y_0)/(x - x_0)$$

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

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

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





The above formulas to be applied when a portion of a line needs to be clipped.