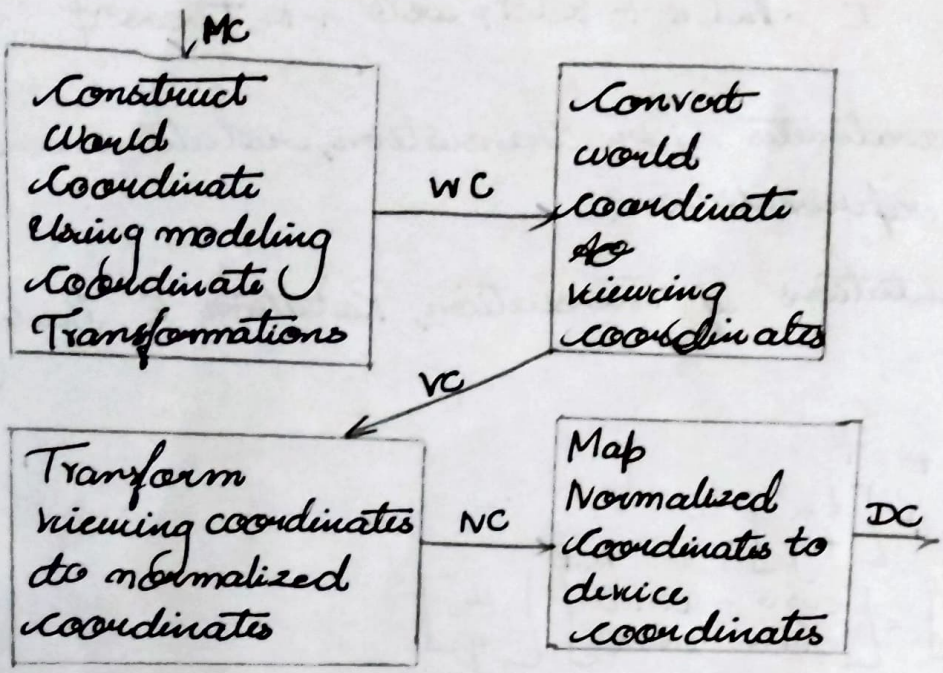
CGV Assignment

RAJATH S
1BY20CS143
VI 'B'

① 2D Pipeline



- We could set up a seperate 2-d, viewing coordinate reference frame for specifying clipping window.
- Systems use normalized coordinates in the range from 0 to 1, others used a normalized range from -1 to 1
- Clipping is usually performed in the normalized coordinates.

② Build Phong lighting model with equations light consists of 3 different types of light.

⟹ Ambient lighting referred as the natural lighting.

⟹ Diffusion - The artificial light.

⟹ Specular lighting - Refers to the shininess of the object.

$$I_{amb} = k_a I_a \rightarrow (1)$$

k_a = ambient reflectivity

I_a = intensity of ambient light

Similarly,

$$I_{diff} = k_d I_p \cos(\theta) \rightarrow (2)$$
$$= k_d I_p (N \cdot L)$$

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

∴ The Phong model gives us the equation of all combined

∴ Total intensity $I = V_a I_a + k_d I_p \cos\theta + k_s I_l \cos^n \phi$

③ Apply homogeneous coordinates for translation, rotation and Scaling via matrix representation.

Sol The Matrix representations of Translation, Rotation & Scaling are

$P' = P + T$

↓

$P + T$

Translation $P' = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$

R.P

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

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

Generic equation $= S \cdot P$

⇓

$P' = M_1 * P + M_2$

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

$h = 1$

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

Translation $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

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

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
⇒ Produces jagged lines that are plotted as a discrete point sets.	Random system produces smooth lines drawing
• Less expensive • Modification difficult • Resolution low • Solid pattern is easy to fill	• More expensive • Modification easy • Resolution high • Solid pattern is difficult to fill.

⑤ Demonstrate openGL functions for window management using GLUT.

- glutCreateWindow – used to create a new window.
- glutCreateSubWindow – used to create another window within same window.
- glutSetWindow – used to set an particular id for the window.
- glutGetWindow – used to get the windows ID.
- glutDestroyWindow – to delete the window that was created.
- glutPostRedisplay – To display the window again and again, continuously until forcibly closed.
- glutReshapeWindow – Used for transformation of world coordinates view coordinates and displaying it.
- glutFullScreen – To represent window in full screen mode.
- glutPopWindow / glutPushWindow – works just like a matrix or window.
- glutHideWindow – To hide the window from being displayed on screen.
- glutDisplayFunc – To display
- glutMainLoop
- init()

(6) OpenGL visibility detection functions

a) openGL polygon culling functions

Remove back face, front face of an object.

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

b) Depth-Buffer Functions

```
glutInitDisplayMode(GLUT_SINGLE/GLUT_RGB/GLUT_DEPTH)
glutInit(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(GC_DEPTH_TEST)
glDisable(GC_DEPTH_TEST)
```

c) openGL wireframe surface visibility methods

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE),
```

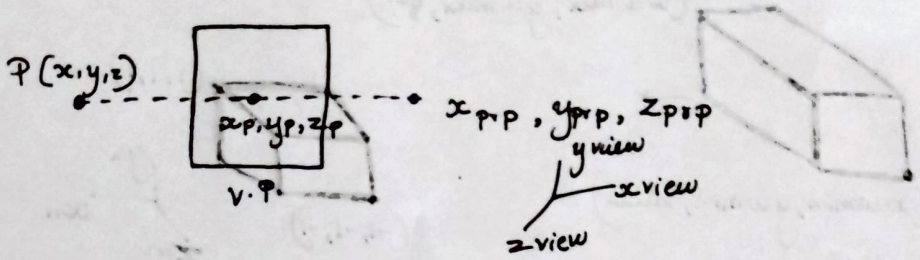
Visible & hidden edges displayed.

d) openGL Depth cueing Fns

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

To increase or decrease the Brightness.

④ Write Special cases discussed with Perspective Projections. [5]



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 $z_{vp} = 0$

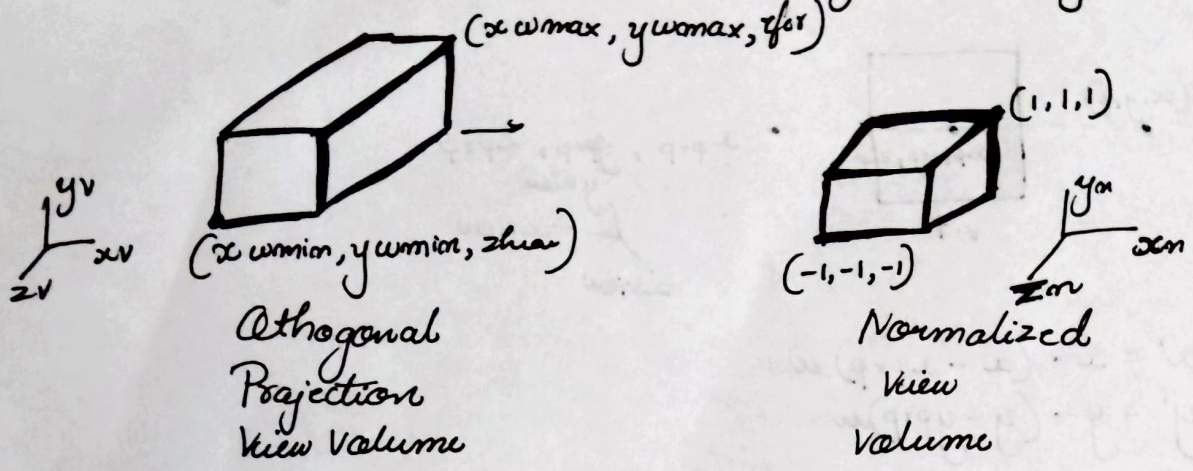
$$x_p = x\left(\frac{z_{prp}}{z_{prp} - z}\right) - x_{prp}\left(\frac{z}{z_{prp} - z}\right)$$

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

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 coordinates normalized in the range 0 to 1.

Another normalization transformation approach is to use Symmetric cube with coordinates -1 to 1.

∴ We get the normalization transformation for the 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 curve and its properties with equations.

Bezier curves are parametric 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

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

$B_i^n(t)$ represents Bernstein Polynomial

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-t} t^i$$

n - polynomial degree
t - variable
i - index

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

We used the above mentioned formulas

Bezier curve = $\binom{n}{r} * (1-t)^{n-t} t^i$ * for every point

n = control points number - 1

t = 0 - 1 (Range)

⑩ Cohen - Sutherland line clipping algorithm

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

1001	1000	1010
0001	0000 (Clipping window)	0010
0101	0100	0110

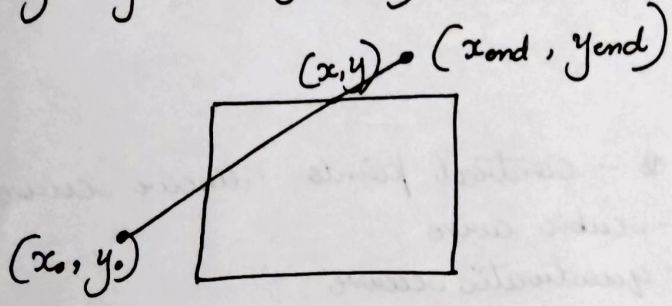
For a line - (x_0, y_0) to (x_{end}, y_{end})

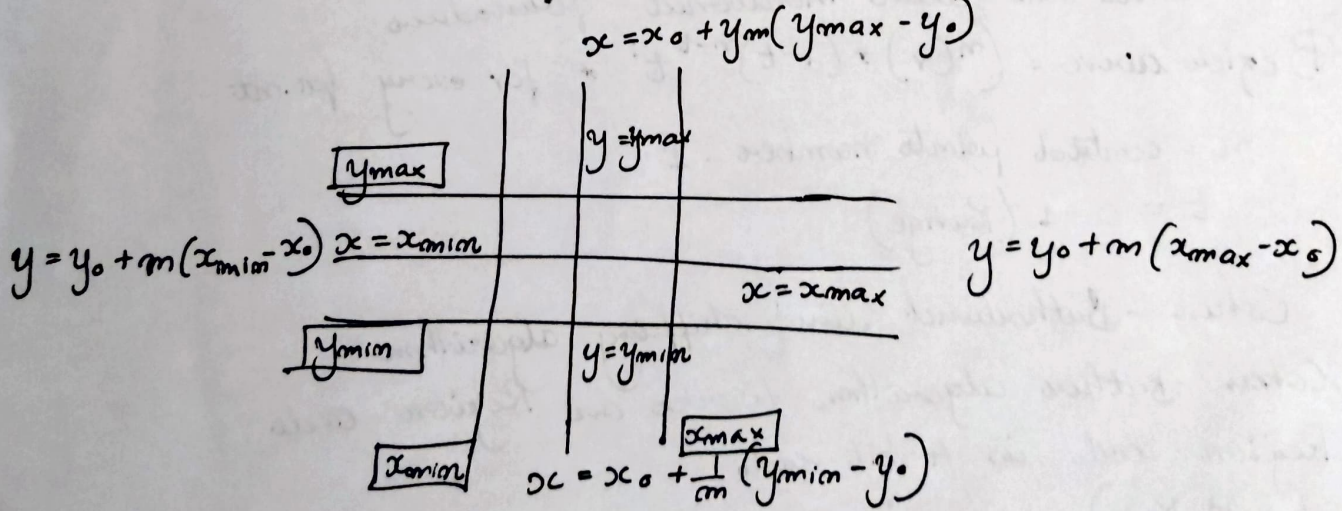
$$m = \frac{(y - y_0)}{(x - x_0)}$$

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

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

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





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