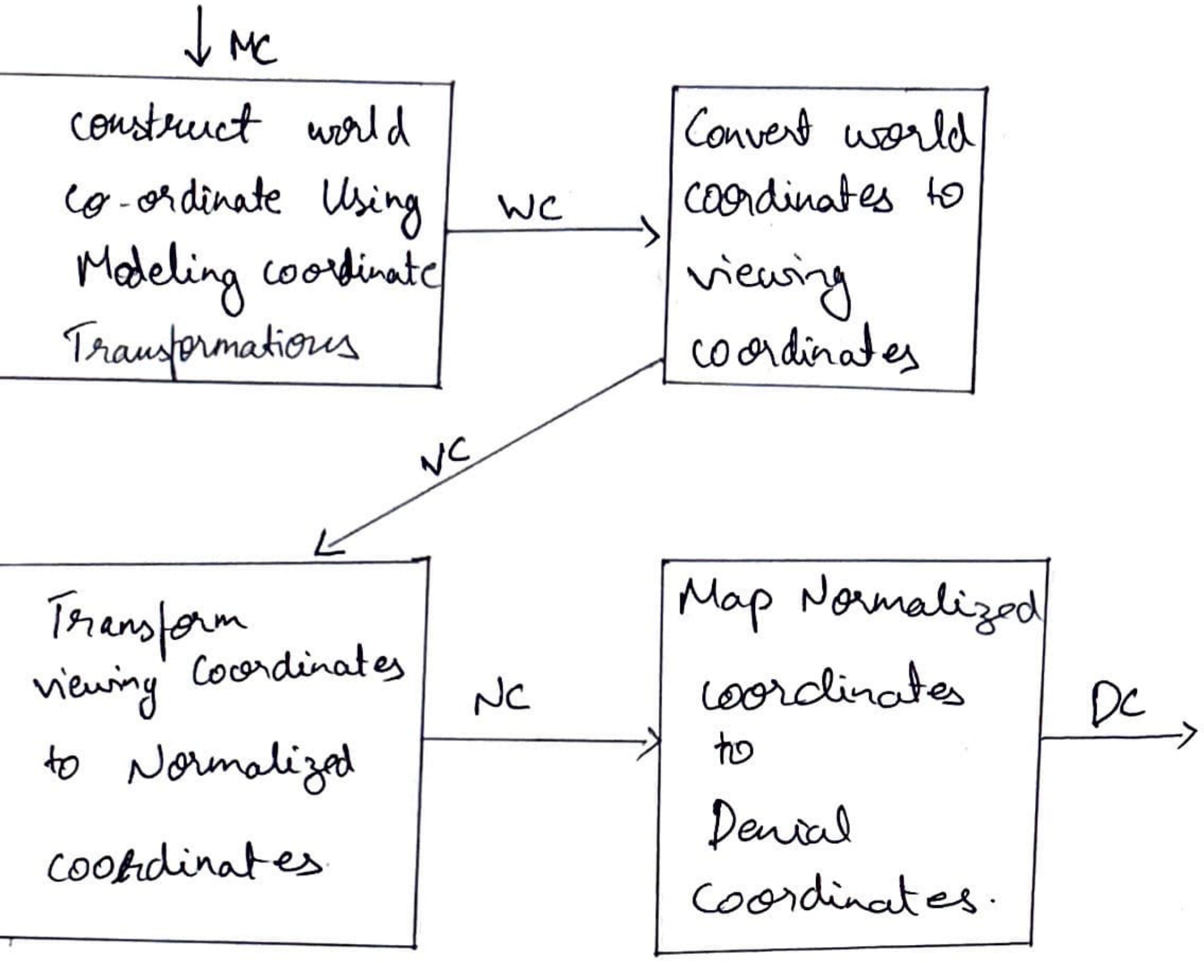
CGV Assignment.

Mithil Menon
1BY20CS111
'VI' B
CGV Assignment

1. 2D Pipeline



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

2. Phong lighting Model.

⟹ Ambient lighting refered as the natural lighting.

⟹ Diffusion - The Artificial light.

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

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

K_a = ambient ~~of~~ 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 $I = K_a I_a + K_d I_p \cos\theta + K_s I_l \cos^n \phi$

3. Homogeneous co-ordinates

The matrix Representations of Translation, Rotation and Scaling are

$P' = P + T$

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

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 $P' \Rightarrow \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Generic Eq $\Rightarrow$ S. P

$$P' = M_1 * P + M_2$$

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

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.

Raster Scan	Random Scan
1) Produces jagged lines that are plotted as a discrete point set	1) Random System products smooth lines drawing
2) Less expensive	2) More expensive
3) Modification difficult	3) Modification easy.
4) Resolution low	4) Resolution high
5) Solid pattern is easy to fill	5) solid pattern is difficult to fill

5. Open GL functions.

- glutCreateWindow - Used to create a new window
- glut CreateSubWindow - Used to create another window within an existing one.
- glutSet Window - Used to set a particular id for the window.
- glut Get Window - Used to get the Window ID
- glut DestroyWindow - To delete the Window that was created.

- glut Post Redisplay - To display the window again & again, till forcibly closed.
- glut Full Screen - To represent Window as a full screen mode
- glut PopWindow / glut Push Window - Works just like a matrix in window.
- glut Display Func - To display
- glut Main Loop.
- init ()

6. Open GL visibility detection functions.

(a). Polygon functions.

```
gl Cull Face (mode);
gl Enable (GL_CULL_FACE);
gl Disable (GL_CULL_FACE);
```

(b). Depth Buffer.

```
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
glClear (GL_DEPTH_BUFFER BIT);
// This works as initiliazation for depth buffer.
```

P.T.O.

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

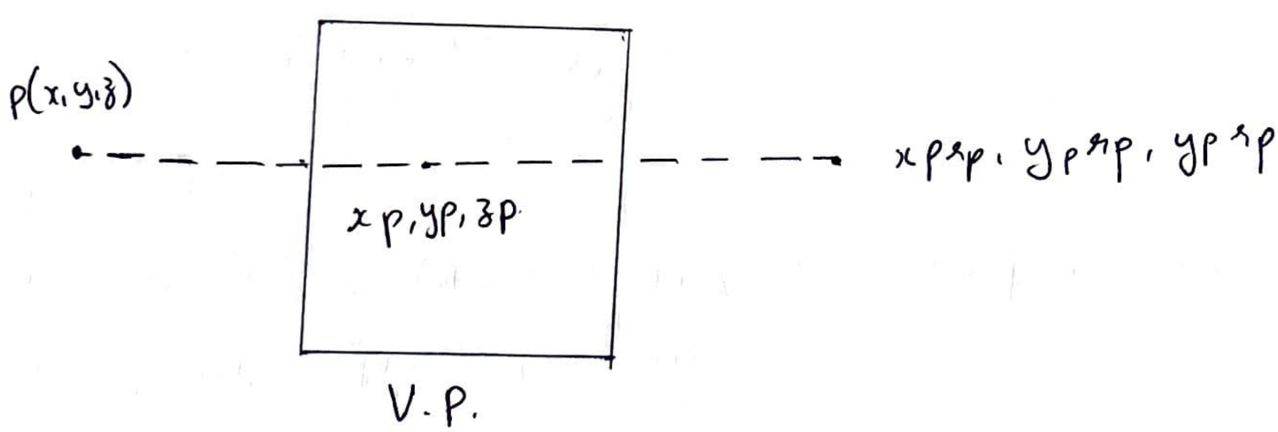
(c) OpenGL Wireframe methods.

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
// visible and hidden edges displayed.
```

(d) OpenGL Depth using Functions.

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

7. Special cases w.r.t. Perspective Projections.



Consider,

$$x' = x - (x - x_{prp})u$$
$$y' = y - (y - y_{prp})u$$
$$z' = z - (z - z_{prp})u$$

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

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

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

<u>Special Cases:</u>

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 $x_p \wedge p, y_p \wedge p, z_p \wedge p = (0,0,0)$;

$$x_p = x\left(\frac{z \wedge p}{z}\right)$$

$$y_p = y\left(\frac{z \wedge p}{z}\right)$$

③ $z \wedge p = 0$.

$$x_p = x\left(\frac{z_p \wedge p}{z_p \wedge p - z}\right) - x_p \wedge p\left(\frac{z}{z_p \wedge p - z}\right)$$

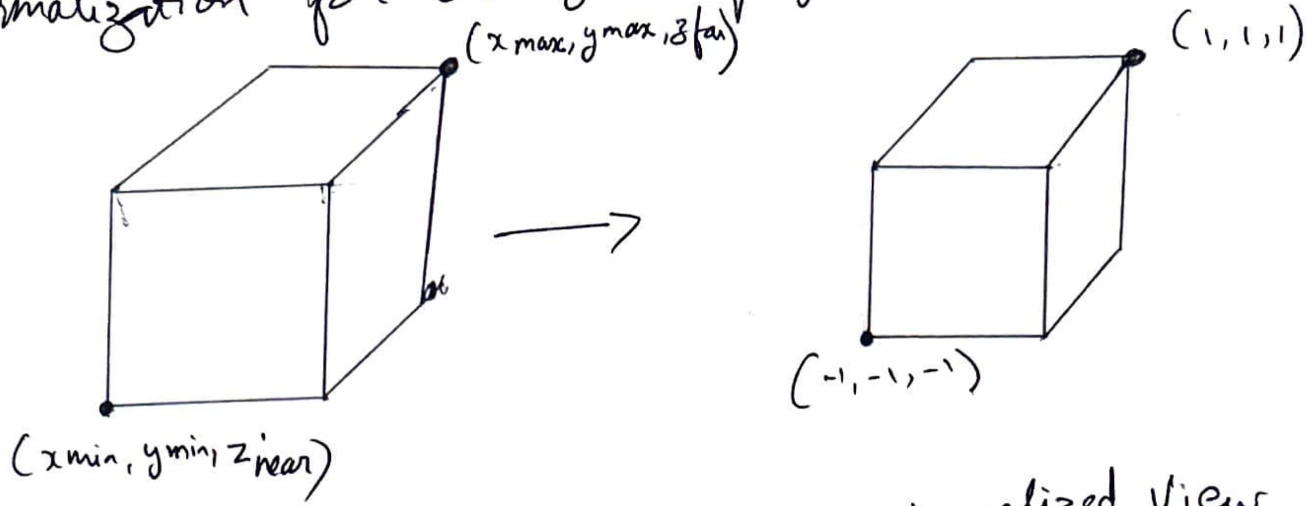
$$y_p = y\left(\frac{z_p \wedge p}{z_p \wedge p - z}\right) - y_p \wedge p\left(\frac{z}{z_p \wedge p - z}\right)$$

④ $x_p \wedge p = y_p \wedge p = z \vee p = 0$.

$$x_p = x\left(\frac{z_p \wedge p}{z_p \wedge p - z}\right)$$

$$y_p = y\left(\frac{z_p \wedge p}{z_p p - z}\right)$$

8. Normalization for orthogonal projection.



Orthogonal projection view

Normalized View.

Consider a unit cube for normalized view with each x, y, z co-ordinate normalized in the range 0-1

Another approach is to use the range -1 to 1

Orthogonal view ~~view~~ volume is.

$$M_{ortho} = \begin{bmatrix} \frac{2}{x_{max} - x_{min}} & 0 & 0 & -\frac{x_{max} + x_{min}}{x_{max} - x_{min}} \\ 0 & \frac{2}{y_{max} - y_{min}} & 0 & -\frac{y_{max} + y_{min}}{y_{max} - y_{min}} \\ 0 & 0 & \frac{-2}{z_{near} - z_{far}} & \frac{z_{near} + z_{far}}{z_{near} - z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

9. Bezier curve.

- Bezier curves are parametric curves that are generated with the help of control points. It is widely used in graphics and other related industry.
- It is named ~~of~~ after French engineer, Pierre Bezier.

Bezier curves are represented as.

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

$B_i^n(t)$ is Bernstein Polynomial.

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

n = polynomial degree
t = variable
i = index.

Bezier curves can be of 2-control point (linear), 3-control point (cubic), 4 control point (quadratic)

we use, $\quad {}^nC_r * (1-t)^{n-t} t^i \quad$ for every point.

10. Cohen - Sutherland line clipping.

- This algorithm works on Region Code.

(TBRL) - Top, Bottom, Right, Left.

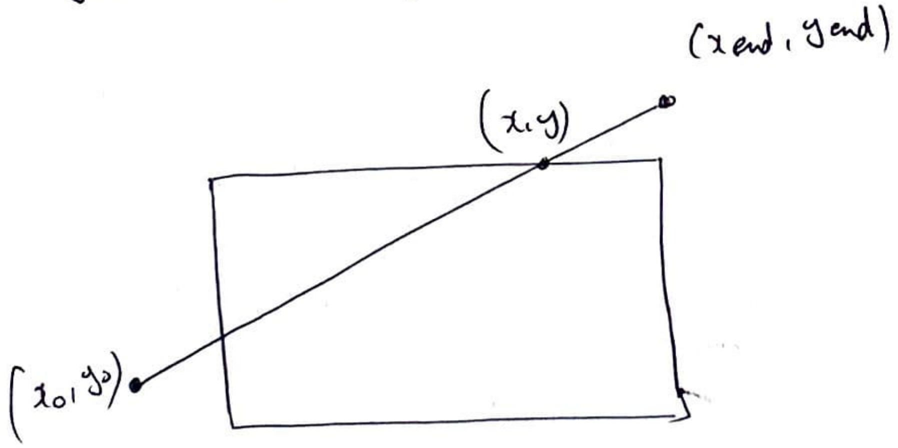
1001	1000	1010
0001	0000 Clipping window	0010
0101	0100	0110

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

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

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

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



There are four formulae that can be used.

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

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

- $y = y_0 + m(x_{max} - x_0)$

- $y = y_0 + \frac{1}{m}(x_{min} - x_0)$