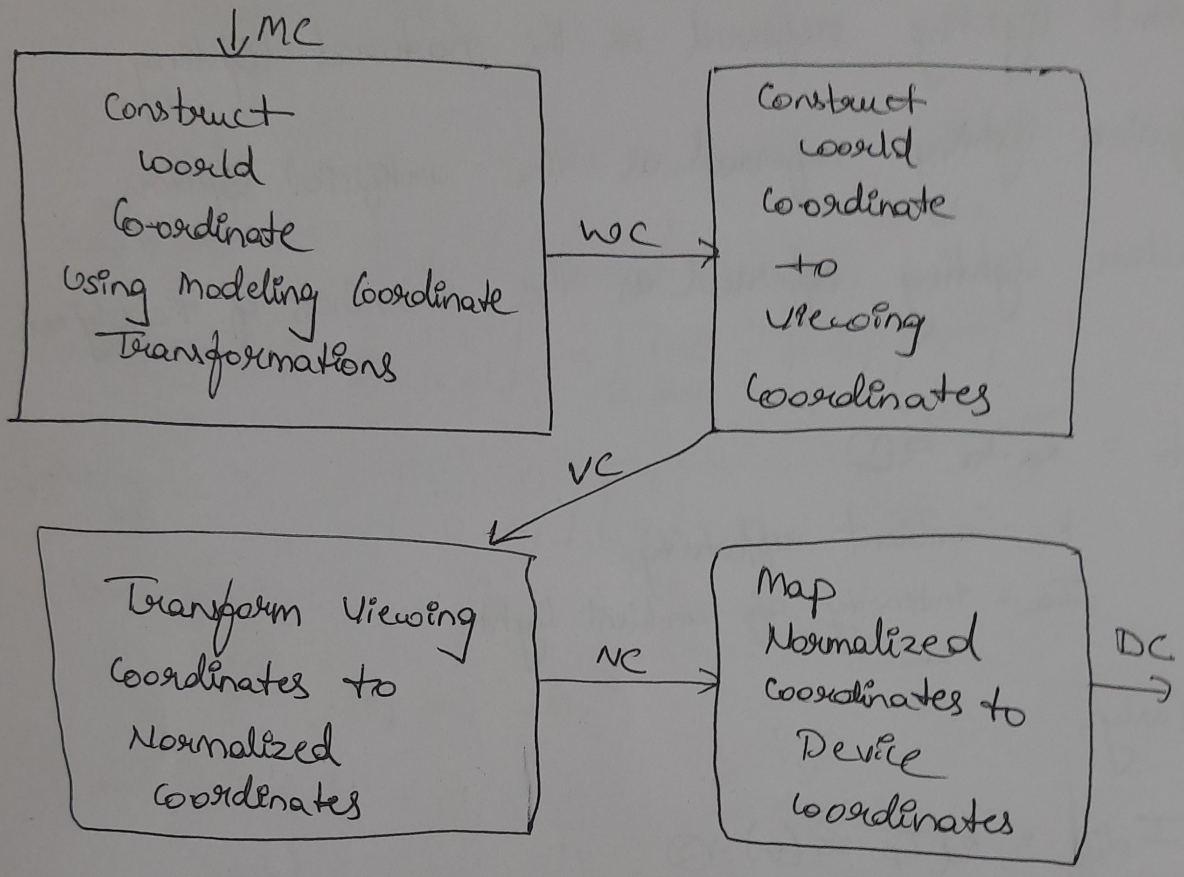
CGV ASSIGNMENT

Nithin R Swamy
1BY20CS129
VI 'B'
CSE

① 2D Pipeline

⇒



- We could set up a separate 2D, 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 co-ordinates.

② Build Phong Lighting model with Equations.

⇒

Light consists of 3 different types of lighting.

- Ambient lighting referred as the natural lighting.
- Diffusion lighting referred as the artificial lighting.
- Specular lighting referred as the shinniness of the obje

$I_{amb} = k_a I_a \rightarrow ①$

k_a = ambient reflectivity.

I_a = intensity of ambient light.

Similarly,

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

$= k_d I_p (N.L)$

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

∴ The Phong model gives the equation of all combined.

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

ns.

ghting.

lighting.

l lighting.

of the obje

of

$k_s I \cos^n \phi$

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

The matrix Representations of Translation, Rotation and Scaling are

$$P' = P + T$$

$P + T$

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

$R \cdot P$

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

Generic Eqn $= S \cdot P$

$\downarrow$

$P' = M_1 * P + M_2$

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

$h = 1$

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

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

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

⇒

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

⑤ Demonstrate OpenGL functions for displaying 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 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 coordinates to view coordinates and displaying it.

glutFullScreen - To represent window in Full screen mode.

glutPopWindow/glutPushWindow - works just like a matrix in window.

glutHideWindow - To hide the window from being displayed on screen.

glutDisplayFunc - To display.

glutMainLoop

init().

⑥ Explain OpenGL Visibility Detection functions.

⇒

- OpenGL polygon Culling Functions

Remove backface, frontface of an object

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

- Depth-Buffer Functions

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

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

```
glDepthRange(nearNormDepth, farNormDepth)
glClear(GL_DEPTH_BUFFER_BIT)
```

- OpenGL wireframe surface Visibility Methods

glPolygonMode(GL_FRONT_AND_BACK, GL_LINE),
visible & hidden edges displayed.

- OpenGL Depth Functions

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

To increase or decrease the Brightness.

write special cases that we discussed with respect to perspective projection transformation coordinates.

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

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

Cases

Projection reference point is limited along z-view axis $x_{prp} = y_{prp} = 0$

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

when projection reference point is at co-ordinate origin

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

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

) If view plane is uv plane and no restriction on placement of projection reference point.

$$z_{vp} = 0 \quad x_p = x\left[\frac{z_{prp}}{z_{prp} - z}\right] - x_{prp}\left[\frac{z}{z_{prp} - z}\right] \quad y_p = [y]\left[\frac{z_{prp}}{z_{prp} - z}\right] - y_{prp}\left[\frac{z}{z_{prp}}\right]$$

) If view plane is in uvw plane and restriction on reference point

$$x_{prp} = y_{prp} = z_{vp} = 0$$

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

⑧ Explain Bezier curve equation along with equation along with properties.

⇒ Developed by French engineer Pierre Bezier for use in design.

It can be fitted to any number of control points.

Equation : $P_k = (x_k, y_k, z_k)$ P_k = generate $(n+1)$ control point position lies inside.

P_k = position vector that describes path.

$$P(u) = \sum_{k=0}^{n} P_k \; BEZ_{k,n}(u)$$

$BEZ_{k,n}(u)$ $C(n,k)\, u^k (1-u)^{n-k}$ is Bezier polynomial.

Explain Cohen-Sutherlan

Every line end poi

4 digits binary cod

each bits is used

1001	1000
0001	000
0101	010

re we established

e determine where

---

⑨ Explain Normalization transformation for orthogonal projecti

⇒ We assume that orthogonal projection view volume to mapped into symmetric normalization cube cube find that point left handed reference frame. Also z-coordinate positon fo handed reference frame. Also x-co-ordinate for near & far position is denoted at z_{near} & z_{far} respectively. This position $(x_{min}, y_{min}, z_{near})$ is mapped to $(1,1,-1)$ Normalization transformation for view volume.

intersection P_2' & P_3 before Intersection

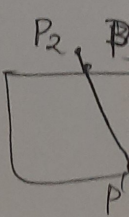
clicking the leg

e is called clipping

determine

$y = y_0$

$x = x_0$



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

) Explain Cohen-Sutherland line clipping algorithm

Every line end point in picture is designed with 4 digits binary code called region code & each bits is used to indicate where point lies inside.

1001	1000	1010
0001	0000	0010
0101	0100	0110

Once we established region code for all line end-point we determine where the complexity inside are not.

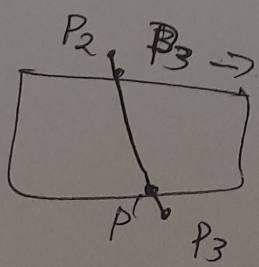
Intersection P_2' & P_3' P_2'' is clipped off for line P_0 to P_4 with we find that point is outside left boundary P_1 is inside. Therefore Intersection P_3 & P_3' to P_3' clipped off.

By clicking the begun code P_3' to P_4, we find remainder of line is below clipping window & can be determined & eliminated.

So determine

$$Y = Y_0 + m(x - x_0)$$

$$x = x_0 + \left(\frac{Y - Y_0}{m}\right)$$



P_2 P_3 → Right Clipping.