CGV ASSIGNMENT

LIKITHA NAVEEN
1BY20CS098
6 'B'

1. 2D Pipeline

MC

Construct world coordinate using modeling coordinate transformations

WC

Convert world coordinates to viewing coordinates

VC

Transform viewing coordinates to normalized coordinates

NC

Map normalized coordinates to device coordinates

DC

- We could setup 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
- Before we select a clipping window and a viewport in openGL, we need to establish the

the appropriate mode for constructing the matrix to transform from world coordinates to screen coordinates

glMatrixMode (GL_PROJECTION)

2. Build Phong highlighting model with equations.

Height consists of 3 different types of light.

→ Ambient lighting is referred to as the natural lighting

→ diffusion : The artificial light

→ Specular lighting : refers to the shininess of the object

$I_{amb} = L_a I_a$

L_a = ambient reflectivity

I_a = intensity of ambient light

Similarly,

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

$= k_d I_p (N.L)$

$I_{spec} = k I_s \cos^n(\theta)$

∴ 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. Apply homogeneous coordinates for translating, rotating and scaling via matrix representation.

→ The Matrix representations of translating, rotation and scaling are:

$$P' = P + T$$

↓

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

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

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	• More expensive
• Modification difficult	• Modification easy
• Resolution low	• Resolution high
• Solid pattern is easy to fill	• Solid pattern is difficult to fill

5. Demonstrate OpenGL functions for window management using (GLUT.

→ glutCreateWindow: Used to create a new window

- glutSetWindow - Used to set a particular id for the window
- glutGetWindow - Used to get window ID
- glutDestroyWindow - To display the window again and again, continuously until forcibly closed
- glutReshapeWindow - Used for transformation of world 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 or seen
- glutDisplayFunc - To display
- glutMainLoop()
- init()

6. OpenGL visibility Detection Functions

- glutCreateSubWindow - Used to create another window within the same window

a OpenGL polygon calling Functions

Remove backface, frontface of an object

glCullface (mode);

glEnable (GL_CULL_FACE);

glDisable (GL_CULL_FACE);

b. Depth buffer functions

glutInit displayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH)

glClear (GL_DEPTH_BUFFER_BIT)

This works as initializating function for depth buffer and refreek buffer

glDepthRange (nearNormDepth, farNormDepth)

glClear ~~Depth~~ (GL_DEPTH_BUFFER_BIT)

glClearDepth (MaxDepth)

glEnable (GL_DEPTH_TEST)

glDisable (GL_DEPTH_TEST)

c. OpenGl Window Wireframe Surface Visibility Methods

glPolygenMode (GL_FRONT_AND_BACK, GL_LINE) -

Visible and hidden edges displayed

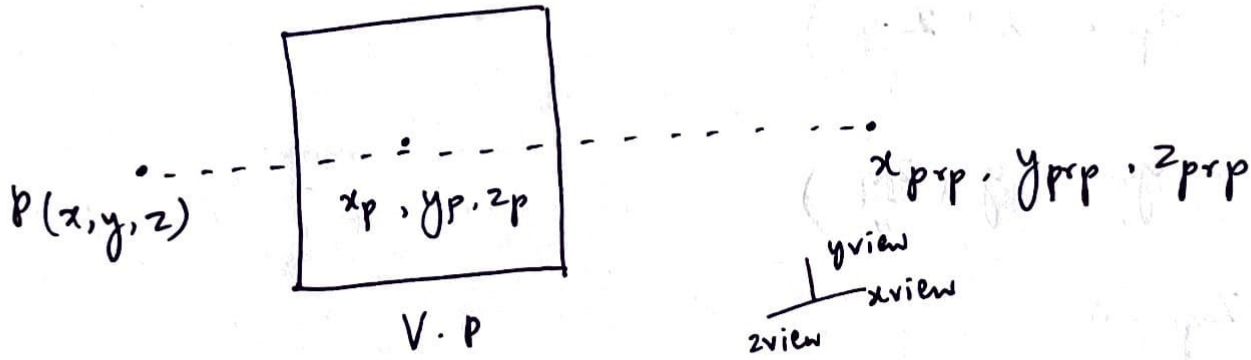
d Open GL Depth Curing Function

glFogi (GL_FOG_MODE , GL_UNCLEAR)

glEnable (GL_FOG)

→ To increase or decrease the brightness.

* Write special cases discussed with perspective Projection



Consider,

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

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

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

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

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

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

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. $z_{vp} = 0$

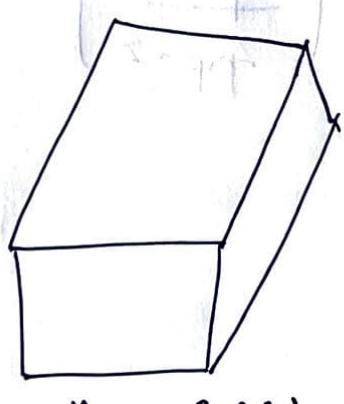
$$x_p = x\left[\frac{z_{prp}}{z_{prp} - z}\right] - x_{prp}\left(\frac{z}{z_{prp} - z}\right)$$

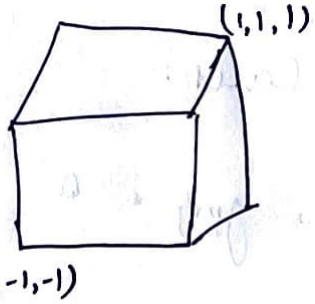
$$y_p = y\left[\frac{z_{prp}}{z_{prp} - z}\right] - y_{prp}\left[\frac{z}{z_{prp} - z}\right]$$

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

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

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 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 equation

→ 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's Polynomial.

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

n - polynomial degree

t - variable

i - index

They are of 2 - control points : linear curve

3 - control points : cubic curve

4 - control points : quadratic curve

We used the above mentioned formed as

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

n = control points number - 1

t = 0 - 1 (Range)

10. Cohen Sutherland line clipping Algorithm

→ Cohen 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 Clipping Window	0010
0101	0100	0110

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

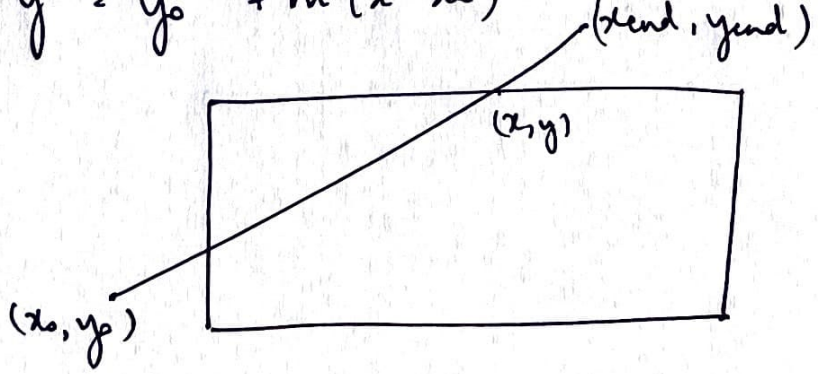
$$m = (y - y_0) / (x - x_0)$$

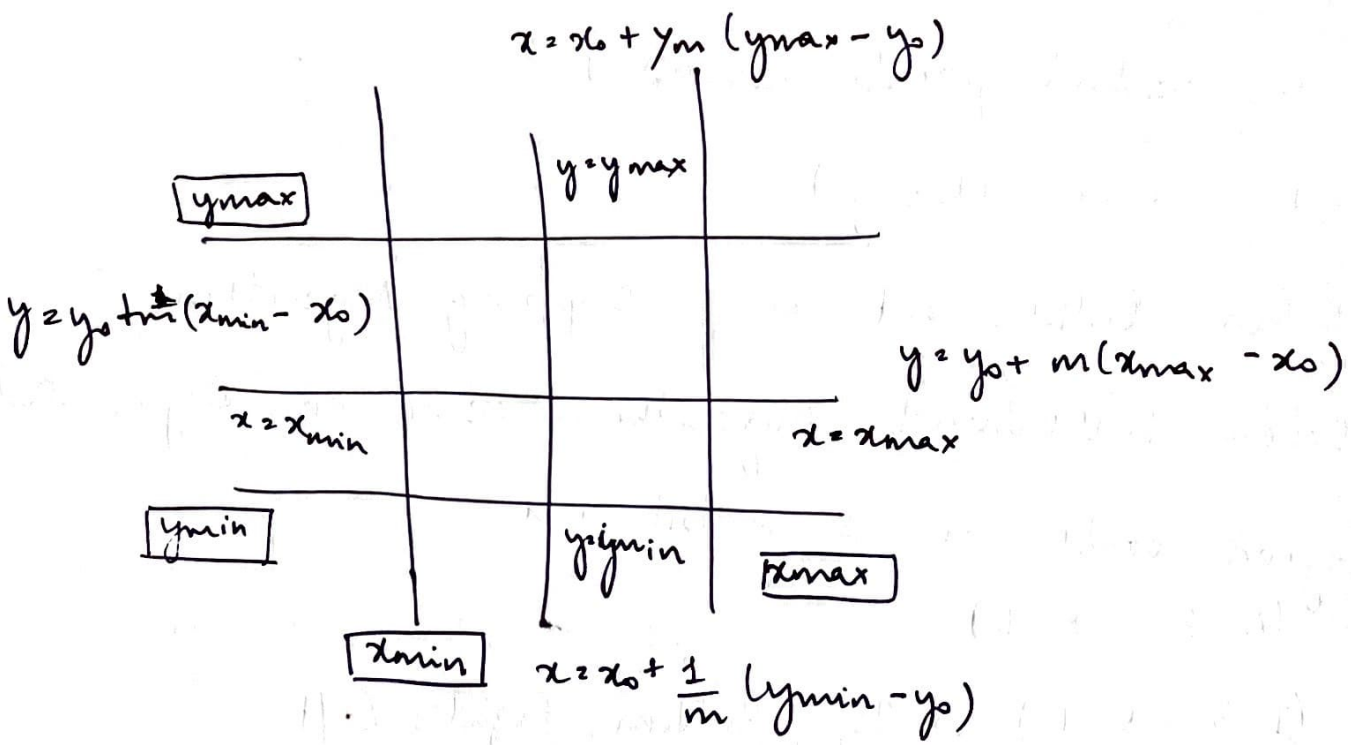
$$m(x - x_0) = (y - y_0)$$

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

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

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





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