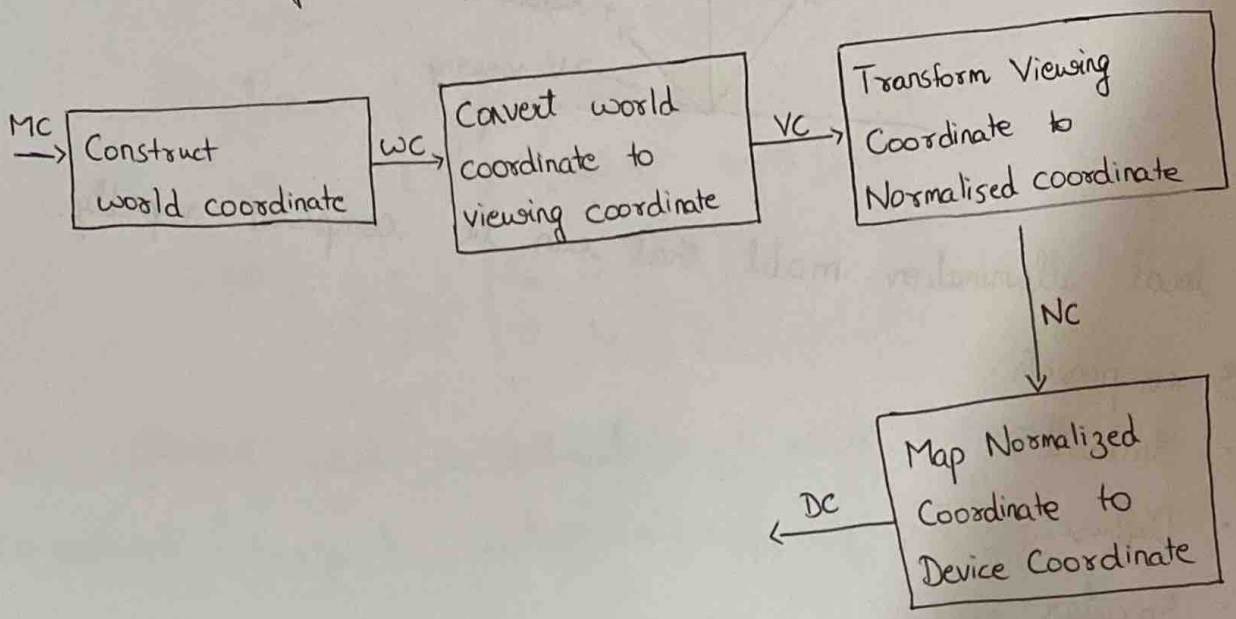
MOHAMMED HASEEB
1BY20CS114

CG Assignment.

1. Build a 2D viewing transformation pipeline and also explain OpenGL 2D viewing functions.

Sol: 2D - viewing pipeline

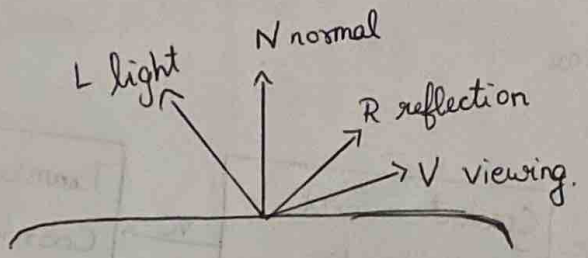


2D - viewing functions.

-> gluOrtho2D - Used to define a 2-D clipping window.
-> glutInit - Used to initalize GLUT
-> glutInitWindowSize - Used to set the size of viewing window
-> glutInitWindowPosition - Used to set the position of viewing window
-> glLoadIdentity()
-> glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB)
-> glutFullScreen()
-> glutPopWindow()
-> glutReshapeFunc(w, h)
-> glutMainLoop()

2. Build Phong Lighting Model with equations.

Sol: Phong Model and specular reflection.



-> A local illumination model that can be computed rapidly.

-> 3 components:
- Ambient
- Diffuse
- Specular

Ambient component

-> $I = I_a K_a$

where I_a = ambient light intensity
K_a = ambient reflection

Diffuse component

-> $I = I_p K_d \cos\theta$

where I_p = intensity of the point
K_d = diffuse reflection

Specular component

-> $I = I_p K_s \cos^n$

where I_p - intensity of the point
K_s - specular reflection
n - shininess.

3. Apply homogeneous coordinate for translation, rotation, scaling via matrix representation.

Sol.

$$\text{Translation} = \underset{P}{\begin{bmatrix} x \\ y \end{bmatrix}} + \underset{T}{\begin{bmatrix} t_x \\ t_y \end{bmatrix}}$$

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

$$\text{Scaling} = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

We represent each co-ordinate (x, y) with homogeneous co-ordinate (x_h, y_h, h) where $x = x_h/h$, $y = y_h/h$

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

$$\text{set } h = 1$$

$$= (x, y, 1)$$

Homogeneous representation for translation, rotation, scaling are:-

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

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

4. Outline difference between raster scan and random scan display.

Sol:

Raster scan	Random scan
→ The resolution of raster scan is lower than random scan.	→ The resolution is higher than raster scan.
→ It is affordable	→ It is expensive.
→ It is difficult to do modifications	→ It is easy to proceed with modification.
→ The entire screen is scanned	→ Only area of the screen with a picture is displayed.
→ Example :- TV sets	→ Example :- Pen plotter.

5. Demonstrate OpenGL function for displaying window management using GLUT

Sol:
→ glutCreateWindow(name) - Creates a top-level window.
→ gluteCreateSubWindow() - Creates a sub window of size width and height at location x and y within current window
→ glutDestroyWindow() - destroys the window specified by win.

→ glutPostRedisplay() - makes the current window as needing to be redisplayed.

→ glutSwapBuffer() - swaps the buffer of the current window if double buffered.

6. Explain OpenGL Visibility Detection Function.

Sol: Z-buffer method.

→ In this method, each surface is processed seperately one pixel position at a time across the surface. The depth value for the pixel are compared and the smallest z surface determines the color to be displayed in frame buffer.

Back-Face detection.

→ A fast and simple object-space method for identifying the back faces of a polyhedron is based on the "inside-outside" test.

→ When an inside point is along the line of sight to the surface the polygon must be a back face.

→ In general, if V is a vector in the viewing direction of eye, then this polygon is a back face if

$$V \cdot N > 0$$

→ If your viewing direction is parallel to viewing z-axis then

$V = (0, 0, V_z)$ and $V \cdot N = V_z C$

So we need to consider the sign of C.

→ Thus, in general we can label any polygon as a back face if its normal vector has a z-component value -

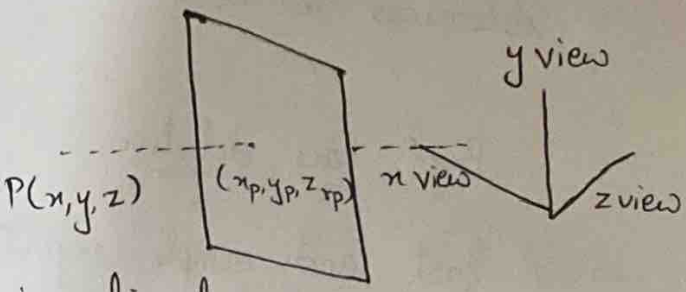
$$C <= 0$$

7. Write the special cases with respect to Perspective Projection transformation coordinates.

Sol. 1. If projection reference point is on z-view, means $x_{prp} = y_{prp} = 0$

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

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



2. The projection reference point is fixed at the co-ordinate origin and $(x_{prp}, y_{prp}, z_{prp}) = (0, 0, 0)$

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

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

3. If the view plane is the uv plane and there are no restrictions on the placement of the projection reference point, then we have.

$$z_{vp} = 0$$

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

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

4. With uv plane as the view plane and the projection reference point on the z view axis, the perspective equations are.

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

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

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

8. Explain Bezier curve equation along with properties.

Sol: For $n+1$ control-point positions, denote as $P_K = (x_K, y_K, z_K)$ with K varying from 0 to n. These coordinate points are blended to produce position vector $P(u)$, which describes the path of an approximating Bezier polynomial function

between p_0 and p_n

$$P(u) = \sum_{k=0}^{n} p_k \, BEZ_{k,n}(u), \quad 0 \le u \le 1$$

$$BEZ_{k,n}(u) = C(n,k)\, u^k (1-u)^{n-k}$$

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

$$x(u) = \sum_{k=0}^{n} x_k \, BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \, BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k \, BEZ_{k,n}(u)$$

Bezier curve is a polynomial of degree that is one less than the number of control points.

9. Explain normalized transformation for an orthogonal projection.

Sol.

$$\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}$$

$$x_v - x_{vmin} = (x_{vmax} - x_{vmin}) \left(\frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}} \right)$$

$$x_v = x_w \left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}} \right) + x_{vmin} + \frac{x_{wmin}\, x_{vmin} - x_{wmin}\, x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$x_v = x_w\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) + \left(\frac{x_{wmax}\,x_{vmin} - x_{wmin}\,x_{vmax}}{x_{wmax} - x_{wmin}}\right)$$

$$x_v = x_w S_x + t_x.$$

where $S_x = \dfrac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$

$$t_x = \frac{x_{wmax}\,x_{vmin} - x_{wmin}\,x_{vmax}}{x_{wmax} - x_{wmin}}$$

$x_v = S_x x_w + t_x$
$y_v = S_y y_w + t_y$ } can be written as.

$$\therefore T.S. = \begin{bmatrix} S_x & 0 & t_x \\ 0 & S_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

For normalized coordinates,

-1 for x_{vmin} and y_{vmin}

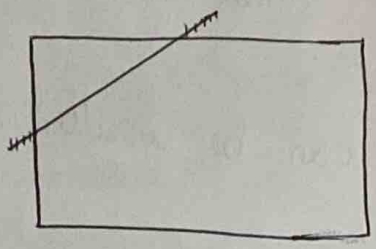
1 for x_{vmax} and y_{vmax}.

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

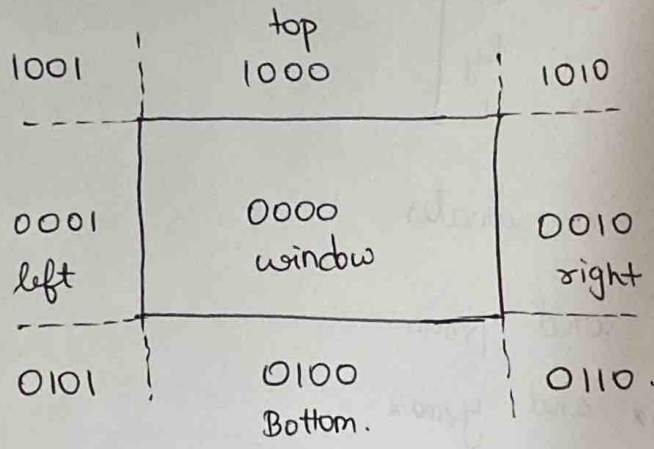
10. Explain Cohen-Sutherland line clipping algorithm.

Sol. · There will be a rectangular window (clipping window)
· There will be an object
· Only pixels inside the rectangle must be shown.
· Pixels outside the rectangle should be clipped.

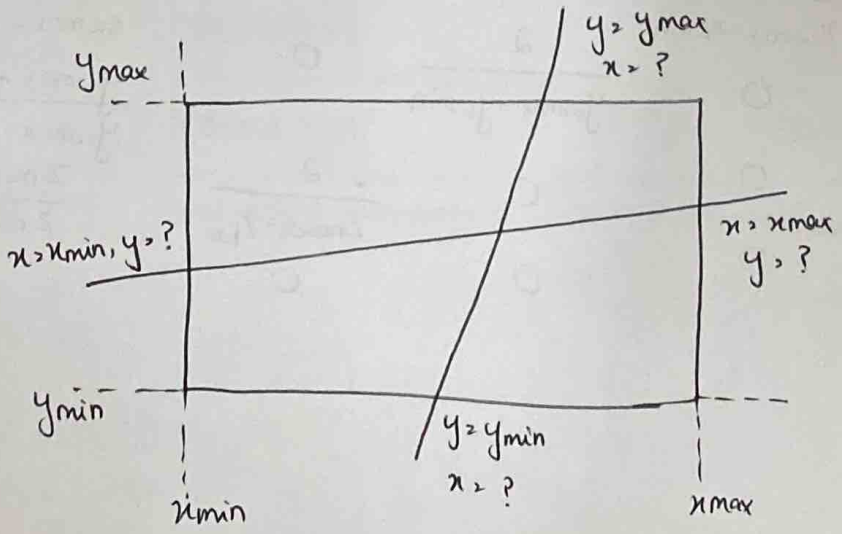
Example :-



Boundaries



Consider



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

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

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

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

$\therefore$