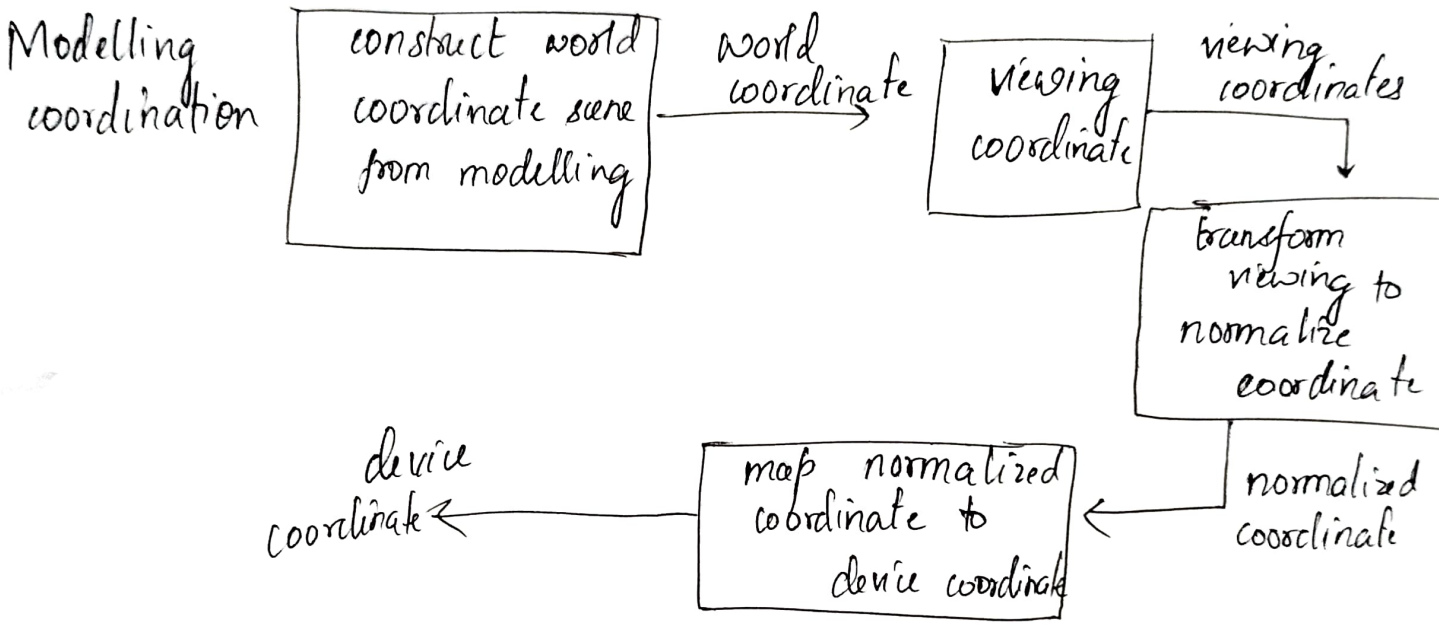
CGV ASSIGNMENT

PRANAV B.C
1BY20CS224
6th 'B'

) Build 2D viewing transformation pipeline and also explain opengl 2D viewing function



Change modelling coordinate to world coordinate by applying modelling transformation. Change world coordinate to viewing coordinate to by determing visible parts. Change viewing coordinates to normalized coordinates & further to device coordinate by clipping and determining pixels.

Open 2D viewing fnc^n

glMatrixMode ()

gf sets current matrix mode

gf assume one of 2 values.

GL_MODELVIEW

Applies subsequent matrix operations to modelview matrix stack.

GL_PROJECTION

Applies subsequent matrix operation to projection matrix stack

gluOrtho2D (xwmin, xwmax, ywmin, ywmax)

specifies the viewing window

xwmin, xwmax, horizontal range, worldcoordinate

ywmin, ywmax : vertical range, world coordinate

glViewport (xmin, ymin, wpwidth, vpright)

2) Outline the differences between raster scan display and random scan display

Random scan	Raster scan
The resolution of random scan is higher than raster scan.	While the resolution of raster scan is lower than random scan
It is costlier than raster scan	cost is less
Alteration is easy in comparison of raster scan	Any alteration is not easy
Interlacing is not used	Interlacing is used
It is suitable for application requiring polygon drawing	It is suitable for creating realistic scenes.

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

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

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

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

Each cartesian coordinates (x, y) with homogeneous coordinates (x, y, h) where $x = x_h/h$, $y = y_h/h$

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

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

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

4) Explain Bezier curve equation.

for n+1 control points positions denoted as $p_k = (x_k, y_k, z_k)$, 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 fn blw fo 4 pn.

$$p(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u), \quad 0 \le n < 4$$

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

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

Eqn P(u) represents a set of three parametric equation for the individual curve coordinate

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

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

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

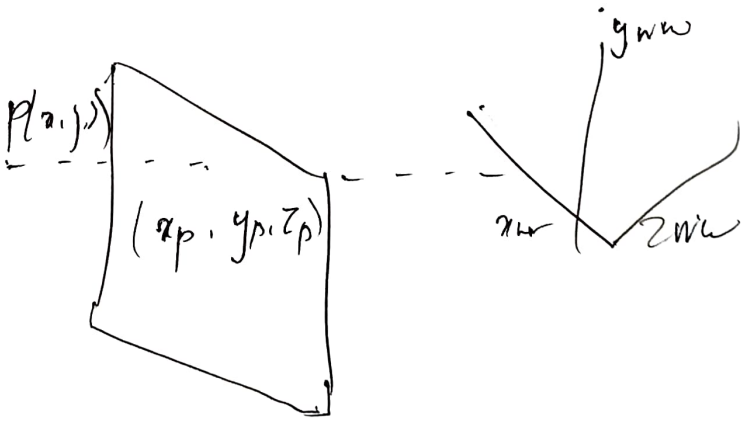
In most cases, a Bezier curve is a polynomial of degree that is one less than the designated number of control points. Three pts generate a parabola, four pts a cubic curve and so forth.

---

7) Write the special cases that we discussed with respect to perspective projection transformation coordinate.

i) If projection reference point is on z view, means $x_{pr} = y_{pr} = 0$

$$x_p = x\left(\frac{z_p - z_p}{z_{ppp} - z_p}\right)$$



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

ii) The projection reference point is fixed at the coordinates origin is

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

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

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

6) Explain opengl visibility detection fns.

glEnable (GL_CULL_FACE)

It is used for turning culling on

glCullFace (mode)

It specifies what to cull

mode = GL_FRONT or GL_BACK

GL_BACK is default

glfrontface (vertex order)

It is for order of vertices

Orientation is changed

vertex Order = GL_CW or GL_CW

GL_CW is for clockwise direction (front)

GL_CCW is for counterclockwise (back)

GL_CCW is default

Create depth buffer by setting GLUT_DEPTH

flag in glutInitDisplayMode() or the appropriate

flag in the PIXELFORMATDESCRIPTOR

7) Demonstrate opengl fn for displaying window management using GLUT

glutInit (&argc, argv)

It is used to initialize GLUT library

glutInitWindowPosition (x_topleft, y_topleft)

position of display window on screen

glutInitWindowSize (dwwidth, dwheight)

size of window

dwwidth is width of display

dwheight is height of display

glutCreateWindow ("String");

It is used to create display window with name.

glutDisplayFun ();

It sets the display callback for current window. ~~glutInitDisplay~~

9) Explain normalization transformation for a orthogonal projection

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

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

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

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

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

$$x_v = x_w s_x + t_x$$

where $s_x = \frac{x_{vmax} - x_{min}}{x_{wmax} - x_{wmin}}$.

M window normsquare =

$$\begin{bmatrix} \frac{2}{x_{wmax} - x_{wmin}} & 0 & 0 & -\frac{x_{max} + x_{min}}{x_{max} - x_{min}} \\ 0 & \frac{2}{y_{max} - y_{wmin}} & 0 & -\frac{y_{max} + y_{min}}{y_{max} - y_{min}} \\ 0 & 0 & \frac{-2}{z_{max} - z_{far}} & \frac{z_{max} + z_{far}}{z_{max} - z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

10) Explain Cohen - Sutherland line clipping algorithm

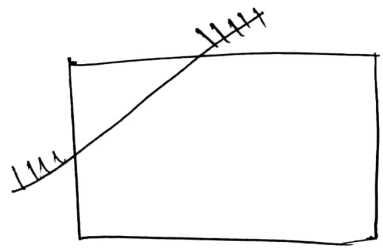
There will be rectangular window (clipping window)

There will be an object (ex. line)

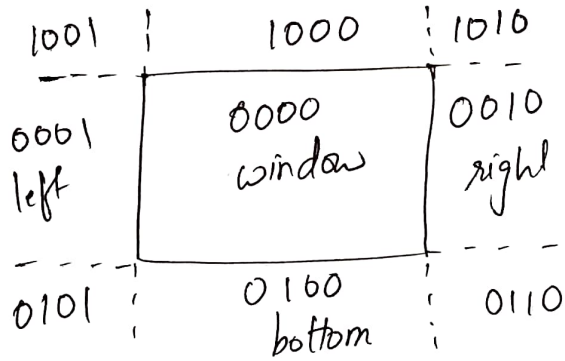
Only pixel inside the rectangle must be shown

Pixel outside the rectangle should not be shown

Example



Boundaries



consider