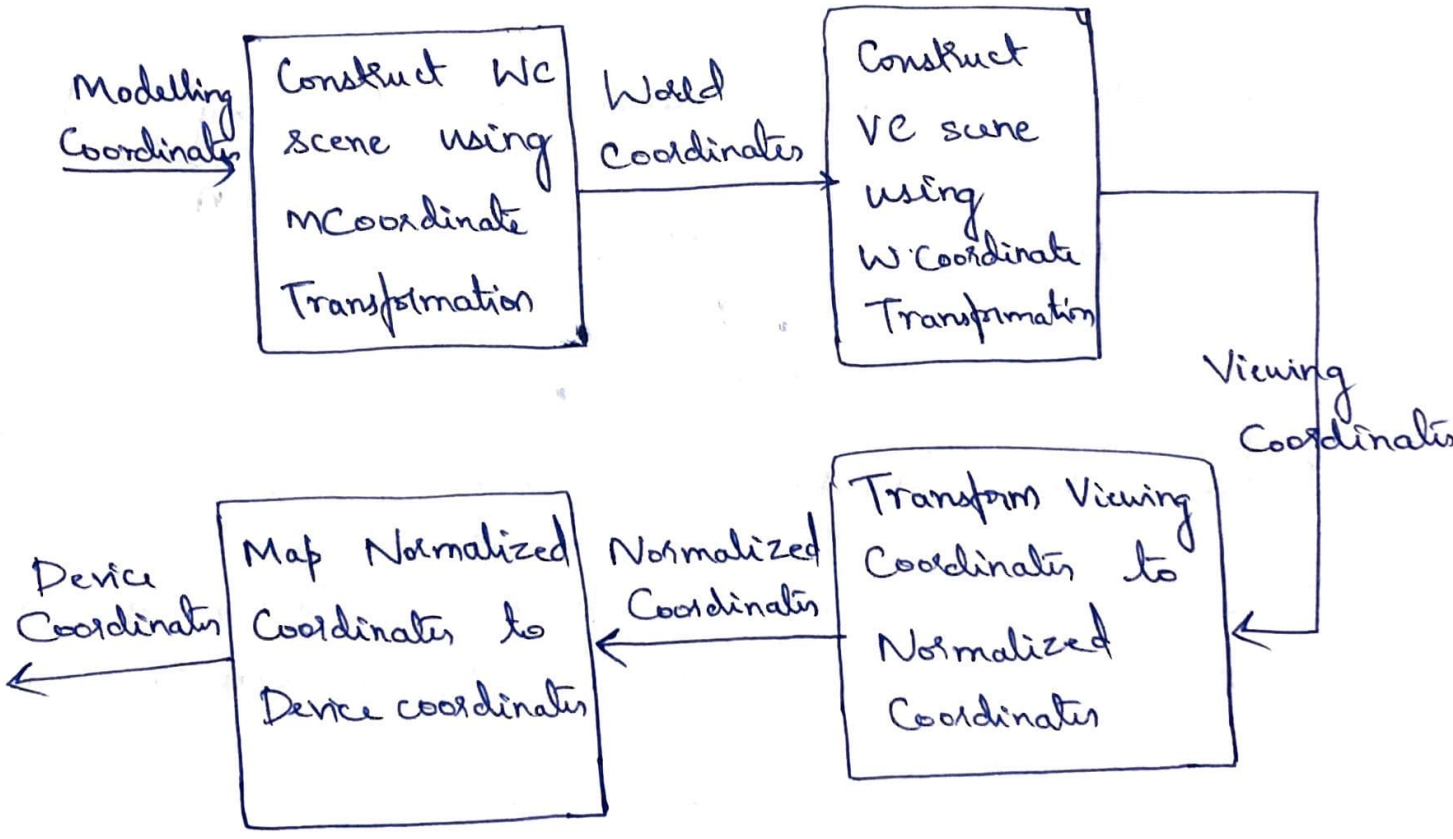
CGV ASSIGNMENT (18CS62)

Name: NAGASHREE L
Sem: 6th Sem, B sec
USN: 1BY20CS121

(1) Build a 2D Viewing pipeline and explain openGL 2D viewing function.

2D Viewing pipeline: The transformation is simply referred to as the window-to-viewport transformation or the windowing transformation.



Once a world coordinate scene has been constructed, we could set up a seperate 2D, viewing coordinate reference frame for specifying the clipping window.

OpenGL 2D-Viewing functions

i) glMatrixMode (GL_PROJECTION): To create a matrix with matrix mode as projection transformation (1)

ii) glLoadIdentity(): This ensures that each time we enter the projection mode, the matrix will be reset to identity matrix.

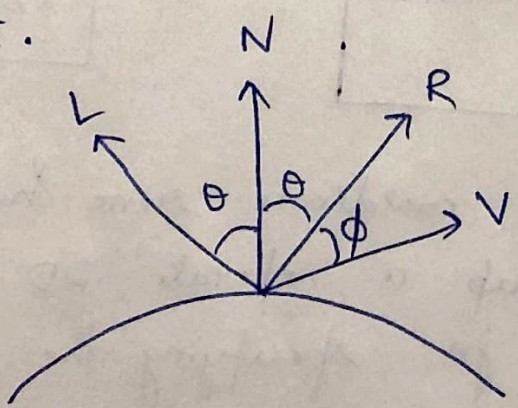
iii) gluOrtho2D(xwmin, xwmax, ywmin, ywmax): It specifies an orthogonal projection for mapping the scene to the screen the orthogonal projection has no effect on 2D scene.

iv) glViewport(xvmin, yvmin, vpWidth, vpHeight): To specify the viewport parameters with the OpenGL function.

v) glClearColor(red, green, blue, alpha): A background color for the display window is chosen in RGB mode with the OpenGL routine.

(2) Build Phong Lighting Model with Equations.

The bright spot, or specular reflection, is the result of total or near total reflection of the incident light.



N → unit normal vector
R → unit specular reflection
L → direction of light
V → vector pointing to viewer

An empirical model for calculating the specular reflection (direction R)

$I = I_a \times K_a$

Light Source property — Material property

$I = I_p \times K_d \times \cos\theta$

diffuse reflection coeff

I_p → Intensity of the point light source

$I = I_p \times K_s \times \cos^n\theta$

I_p → intensity of the point light source

K_s → specular reflection coefficient

n → shininess

$I = I_p \times K_d \times N.L$

$I = I_p \times K_s \times (R.V)^n$

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

Cartesian co-ordinate (x, y) with homogeneous coordinate (x_h, y_h, h) where $x = x_h/h$, $y = y_h/h$

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

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

Translation:

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

Scaling:

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

Rotation:

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

(4) Outline the differences between raster scan displays and random scan displays

Raster Scan Display	Random Scan Display
→ Electron beam is swept across the screen, one row at a time	→ Electron beam is directed only to the parts of screen.
→ Its resolution is poor because of zigzag lines	→ Resolution is good because of smooth lines.
→ It uses pixels, picture definition is stored as a set of intensity values.	→ Picture definition is stored as a set of line drawing instructions.
→ Screen points / pixels are used to draw images.	→ Mathematical functions are used to draw an image.

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

i) glutInitWindowPosition(): gives integer, screen coordin position for the top-left corner of the display window, relative to screen

ii) glutInitWindowSize(): the width and height has to be chosen for the display window in position integer dimensions.

iii) glutCreateWindow(): creates a display window, with the specified size and position and assigns depending on the windowing system.

iv) glutInitDisplayMode(mode): To choose the color mode, diffent buffer combinations, and the parameters.

v) glutearIndex(indux): sets the display window color using color-index mode, wher index is an integer value corresponding to the color table.

(6) Explain OpenGL visibility Detection functions.

i) OpenGL Culling function:

glEnable(GL_CULL_FACE): back-face removal is accomplished glCullFace(mode). By default, parameter mode in the glCullFace function.

ii) OpenGL Depth Buffer:

glutInitDisplayMode(mode): Initialization function for the display mode to include a request for the depth buffer.

iii) glClearDepth(maxDepth):

maxDepth is set to any value between 0 and 1. Projection words in openGL are normalized to range from -1.0 to 1.0.

iv) glDepthRange(near, far): By default, near is 0 and far is 1.0 but with this function, we can start them to any value between 0 and 1.

(7) Write the special cases that we discussed with respect to perspective projection

Case 1: $x_{prp} = y_{prp} = 0$

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

Case 2:

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

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

Case 3:

$$z_{vp} = 0$$

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

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

Case 4:

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

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

(8) Explain Bezier curve equation along with its properties.

There are $n+1$ control points, $p_k = (x_k, y_k, z_k)$

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

Bezier blending functions $BEZ_{k,n}(u)$ is:

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

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

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

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

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

Bezier's curve :

→ 3 points forms a parabolic curve

→ 4 points forms a quadratic curve

→ It has one degree less in polynomials :

Linear $x+1$

Quadratic $x^2 + x + 1$

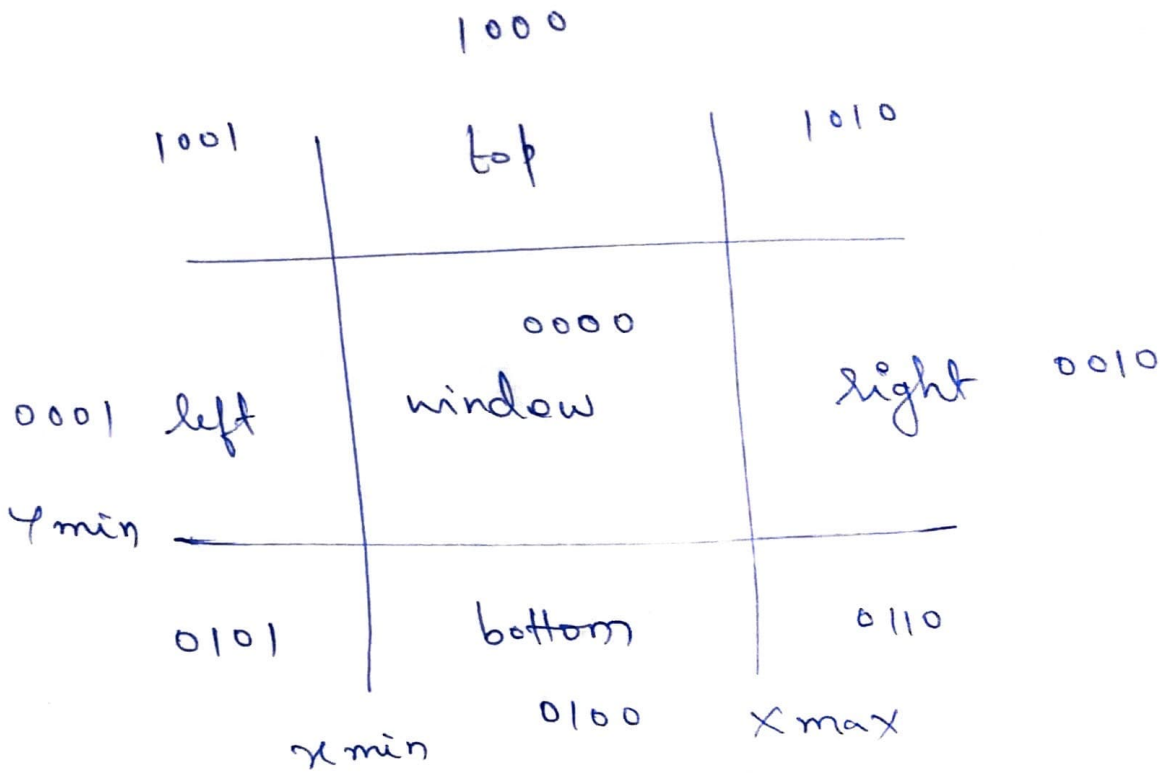
Cubic $x^3 + x^2 + x + 1$

Quantic $x^4 + x^3 + x^2 + x + 1$

Quintic $x^5 + x^4 + x^3 + x^2 + x + 1$

(10) Explain Cohen - Sutherland line clipping algorithm

There will be a rectangular window called the clipping window and an object.



→ Fix boundries so that we can know what to clip.

→ To clip pixels outside the window, lets find and first calculate the intersection point, then redraw the line line from inner window.