Name: Meruva Ashish

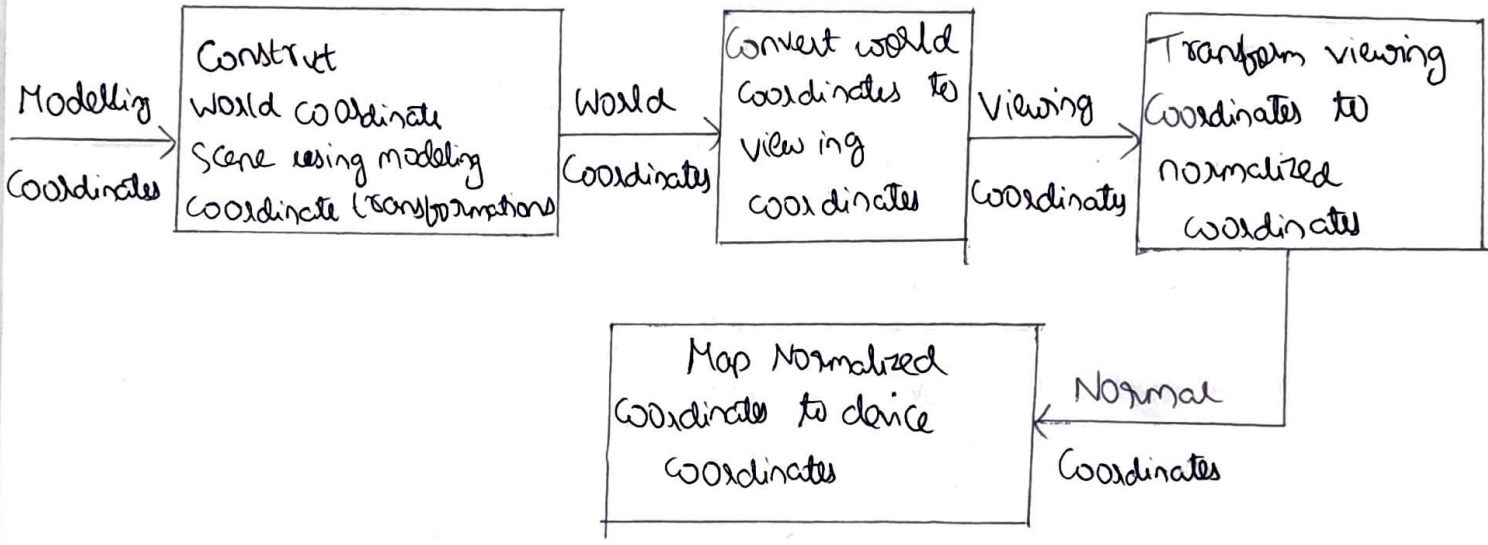
USN: 1BY20CS110

Sem-Sec: 6th - 'B'

CGV Assignment

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

⇒ We can describe the steps for two-dimensional viewing as the following pipeline.



This transformation is referred to as the window-to-viewport transformation or windowing transformation.

Once a world-coordinate scene has been constructed, we setup a separate 2D, viewing coordinate reference frame for specifying the clipping window.

A few openGL 2D viewing functions are:

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

ii) glLoadIdentity(); : This function ensures that each time we enter the projection mode, the matrix will be reset to the identity matrix so that the new viewing parameters are not combined with the previous ones

iii) glOrtho2D(xwmin, xwmax, ywmin, ywmax); : This function specifies an orthogonal projection for mapping the scene to the screen, the orthogonal projection has no effect to our 2-D scene other than to convert object positions to normalized coordinates

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

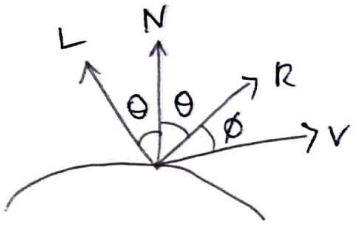
v) glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); : A background color for the display window is chosen in RGB mode with the OpenGL routine.

---

20) Build Phong Lighting model with equations.

⇒ Phong model deals ambient, diffusion and specular reflections. It's an empirical model for calculation of the specular reflection angle, developed by Phong Bui Tong.

It sets the intensity of specular reflection proportional to $\cos^{ns}\phi$



N - normal surface vector

R - unit vector in direction of ideal specular reflection

L - unit vector toward point light source

V - unit vector to the viewer from surface position

$I = I_a \times k_a$

Light source property — Material property

Ambient

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

Intensity of point light source — diffuse reflection coefficient

Diffusion

$I = I_p \times k_s \times \cos^n \alpha$

I_p: intensity of point light source

k_s: specular reflection coefficient

n: shininess

Specular

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

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

3Q) Apply homogenous coordinates for translation, rotation and scaling via matrix representation.

⇒ ~~Homo~~ Each Cartesian coordinate (x, y) with homogenous co-ordinate (x_h, y_h, h) where $x = x_h/h$, $y = y_h/h$

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

Set $h = 1$

$(x, y, 1)$

Homogenous co-ordinates representation for translation, scaling and rotation are as:

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

Scaling: $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Rotation: $\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

4Q) Outline the differences between raster scan displays and random scan displays.

⇒ In Raster-scan

⇒

Raster Scan System	Random Scan System
• The electron beam is swept across the screen, one row at a time, from top to bottom	• The electron beam is directed only to the parts of screen where a picture is to be drawn
• It's resolution is poor because raster system in contrast produces zigzag lines that are plotted as discrete	• It's resolution is good because this system produces smooth lines drawing because CRT beam directly follows line path
• 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 for images.

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

⇒ i) <u>glutInitWindowPosition(xTopLeft, yTopLeft)</u> :

Gives the integer, screen-coordinate position for the top-left corner of the display window relative to the top-left corner of the screen.

ii) <u>glutInitWindowSize(dwWidth, dwHeight)</u>.

Allows us to choose the width & height for the display window in positive integer dimensions.

iii) <u>glutCreateWindow("Title of Display Window")</u>:

Creates a display window, with specified size & position and assigns it depending on windowing system

iv) glut Init Display Mode(mode):

We use this function to choose a color mode (RGB or index) and different buffer combinations, and the selected parameters are combined with the logical or operation

v) glutInit Display Mode(GLUT_SINGLE | GLUT_RGB):

The color mode specification GLUTRGB is equivalent to GLUT_RGBA

6Q) Explain OpenGL visibility Detection functions.

Polygon Culling:

=> i) glEnable (GL_CULL_FACE): Back-face removal is accomplished with these functions.
glCullFace (mode)

where parameter mode is assigned the value GL_BACK, GL_FRONT, GL_FRONT_AND_BACK.

ii) OpenGL Depth-Buffer functions:

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

~~Initial~~ Initialization function for the display mode to include a request for the depth buffer.

iii) glClearDepth (maxDepth): ~~Pa~~

Parameter MaxDepth can be set to any value b/w 0.0 and 1.0. Projection coords. in OpenGL are normalized to range from -1.0 to 1.0. And depth values b/w near & far clipping planes are further normalized to range from 0.0 to 1.0.

iv) glDepthRange (nearNormDepth, farNormDepth):

By default, nearNormDepth = 0.0 and farNormDepth = 1.0. but with this func., we can set them to any value with 0.0 to 1.0 including nearNormDepth > farNormDepth.

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

⇒ Case 1: To simplify persp calc., projection reference point could be limited to positions along z view axis, the

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

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

Case 2:

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

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

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

8Q) Explain Bezier curve equation along with it's properties

⇒ Bezier curve equation uses the following equation to draw curved lines with the help of perspective point.

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

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

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

A few properties of Bezier curve equation are:

- It uses three parametric eqns for individual curve

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

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

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

- 3 points forms a parabolic curve
- 4 points forms a quadratic curve
- It has degree less in polynomials:

Linear $x+1$

Quadratic x^2+x+1

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

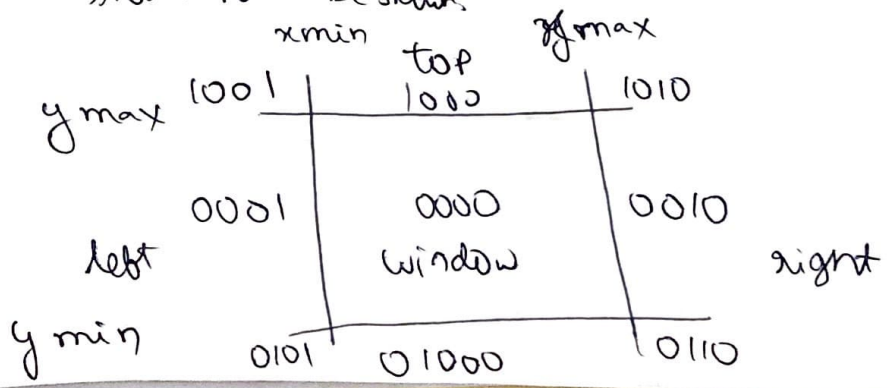
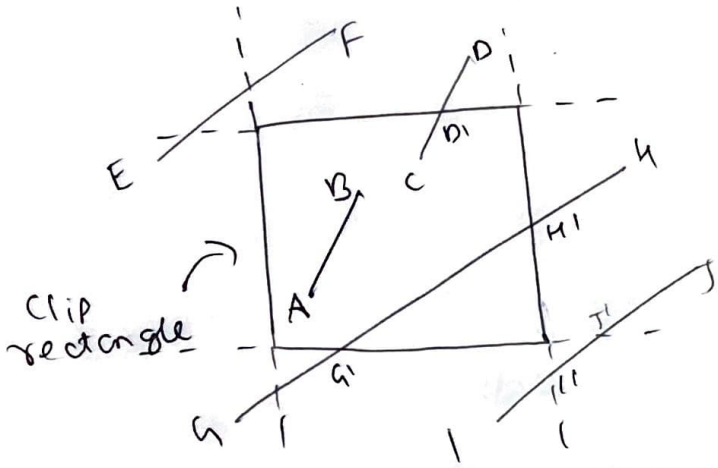
Quartic $x^4+x^3+x^2+x+1$

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

Q) Explain Cohen-Sutherland line-clipping algorithm

⇒ In Cohen-Sutherland line-clipping:

- There will be a rectangular window (clipping window) and an object
- We have to show only those pixels inside the rectangle and the ones outside should not be shown.



The first step in Cohen Sutherland and to first fix some boundaries so that we can know what to clip. After that, to clip pixels outside the window, we calculate the intersection point, then redraw the line from inner window point to this intersection point.