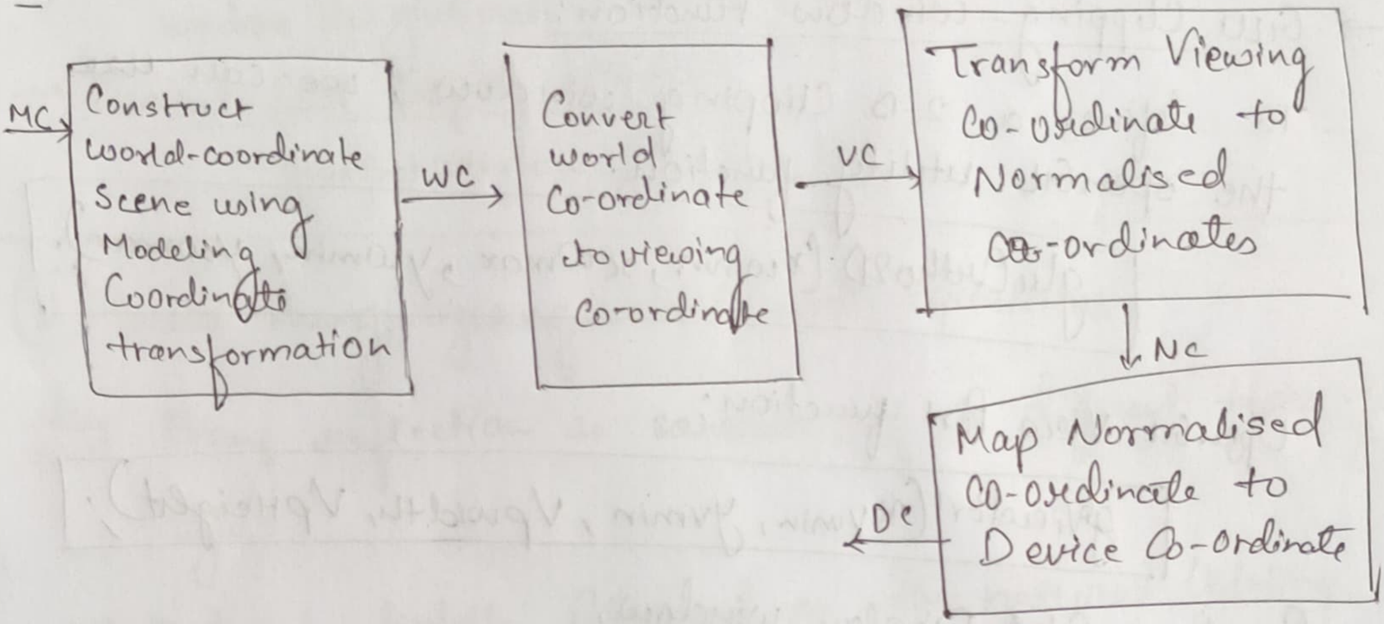
CGV Assignment

Palak Kumari
1BY20CS133
VI-B
~~CGV~~

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

Ans



2D viewing functions:

We can use these two dimensional routines along with the opengl viewport function, all the viewing operations we need.

OpenGL Projection Mode:

Before we select a clipping window and a viewport in OpenGL, we need to establish the appropriate mode for constructing the matrix to transform from world Co-ordinates to screen Co-ordinates.

glMatrixMode (GL-PROJECTION);

This designates the Projection matrix as the current matrix, which is originally set to identify matrix.

→ GLU Clipping-window function:

To define a 2-D Clipping window, we can use the openGL utility function.

gluOrtho2D (xwmin, xwmax, ywmin, ywmax);

OpenGL View Port function:

glViewPort(xvmin, yvmin, Vpwidth, VpHeight);

Create a Glut Display window.

glutInit (&argc, argv);

we have three functions in GLUT for definitions a display window and choosing its dimension and position.

glutInitWindowPosition (xTopLeft, yTopLeft);
glutInitWindowSize (dwidth, dHeight);
glutCreateWindow("Title of display Window");

→ Setting the GLUT Display window Mode & Color:-

Various display window Parameters are selected with the GLUT function:

glutInitDisplayMode (mode);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glClearColor (red, green, blue, alpha);
glClearIndex (index);

→ <u>GLUt Display-window identifier!</u>

windowID = glutCreateWindow ("A display window");

→ <u>Current GLUT Display Window</u>

glutSetWindow (window-ID);

---

(2) Build Phong lighting Model with equations.

Ans Phong reflection is consists of 3 different types of light.

→ Ambeint lightly Referred as the natural lighting

→ Diffusion - The artificial light

→ Specular lighting - Refers to the shininess of the object.

$I_{amb} = K_a I_a$ —— (1)

K_a = ambient reflectivity

I_a = Intensity of ambient light

Similarly

$I_{diff} = K_d I_p \cos(\theta)$ —— (2)

$= K_d I_p (N \cdot L)$

$$I_{spec} = k_s I_l \cos^n \phi$$

∴ 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 co-ordinates for translation, rotation and scalling via matrix representation.

<u>Ans</u> The three basic 2-D transformations are translation, Rotation and Scaling

$\boxed{P' = M_1 * P + M_2}$ P' x P represents column vectors

Matrix $M_1 \rightarrow$ 2x2 array containing multiplicative factor

$M_2 \rightarrow$ 2 elements column matrix containing translation term $\begin{bmatrix} x_b \\ y_b \end{bmatrix}$

For translation, M_1 is identity Matrix $P' = P + T$

where $T = M_2$

For rotation and scalling, M_1 is contains translational terms associated with pivot points or scaling.

HOMOGENOUS CO-ORDINATES:

A standard technique to expand the matrix representation for a 2D-coordinate (x, y) position to a 3-element representation for a 2D co-ordinates (x_h, y_h, h) → Called Homogeneous co-ordinates

h → homogeneous parameter h (non-zero value)

i.e (x,y) is Converted into new co-ordinate values as (x_h, y_h, h) $x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$ $x_h = x \cdot h$

$y_h = y \cdot h$

→ Translation

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

This translation operation can be written as

$$P' = T(t_x, t_y) \cdot P$$

3x3 translation matrix

→ Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = R(\theta) \cdot P$$

→ Scaling matrix

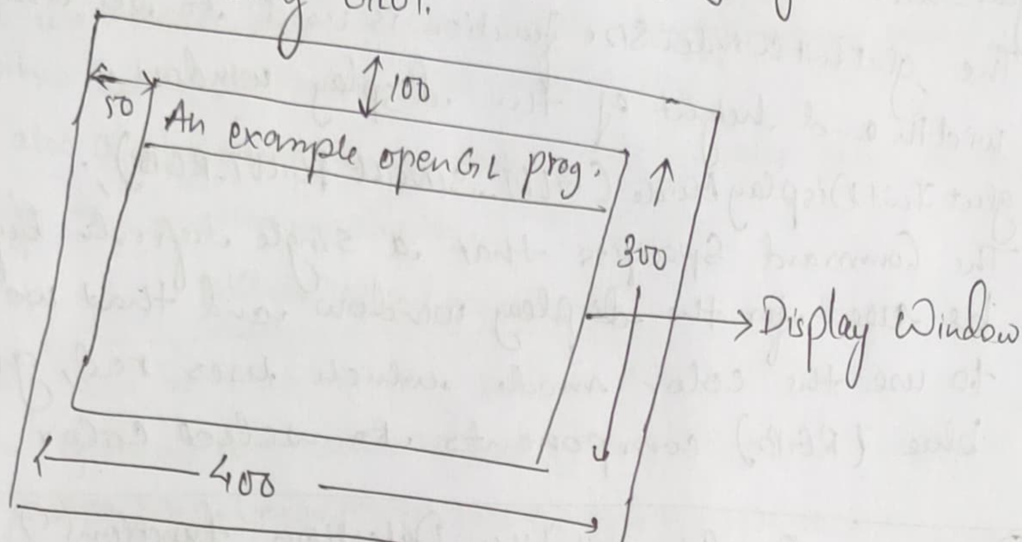
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = S(S_x, S_y) \cdot P$$

(4) Outline the diffrence between raster scan displays and random scan displays.

Ans

Random Scan Display	Raster Scan Display
1. In vector scan display the beam is moved between the end points of the graphics primitives.	1. In raster scan display the beam is moved all once the screen one scanline at a time from top bottom and then break to top.
2. Vector display flickers when the numbers of primitives in the buffer becomes too large	2. In raster display, the refresh process is independent of the complexity of the image.
3. Scan Conversion is not required.	3. Graphics primitives are specified in terms of their endpoints and must be scan connected into their corresponding pixel in the frame buffers.
4. Scan Conversion hardware is not requird	4. Because each primitive must be scan-converted, real-time dynamics is far more computational and required separate scan conversion hardware.
5. Vector display derives a continuous and smooth lines.	5. Raster display can display mathematically smooth lines polygons. and boudries of curved primitives only by approxmating them with pixel on the raster grid
6. Cost is more	6. Cost is low

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



→ we perform the GLUT initialization with the statement

$\boxed{\text{glutInit (\&argc, argv);}}$

→ next, we can state that a display window is to be created on the screen with a given caption for the little bar this is accomplished with the function.

glutCreateWindow("An Example OpenGL program").

where the single argument for this function can be any character string.

→ the following function call the line segment description to the display window

glutDisplayFunc(lineSegment);

→ glutMainLoop();

this function must be the last one in our program It displays the initial graphics and puts the program into an infinite loop that checks for input from devices such as mouse or keyboard.

→ glutInitWindowPosition(50,100);

This following statement specifies that the upper-left cover of the displaywindow should be placed 60 pixel to the right of the left edge of the screen and 100 pixels down from the top edge of the screen.

→ glutInitWindowSize (400, 300);

The glutInitWindowSize function is used to set the init pixel width and height of the display window.

→ glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);

The Command Specifies that a single refresh buffer is to be used for the display window and that we convert to use the color mode which uses red, green and blue (RGB) components to select color values.

---

(6) Explain OpenGL visibility Detection functions?

Ans

a) OpenGL polygon - Culling function:

Back face removal is accomplised with the function glEnable (GL_CULL_FACE); glCullFace (mode);

- where parameter mode is assigned the value GL_BLACK, GL_FRONT, GL_FRONT_AND_BACK.
- By default, parameter mode in glCullFace function has the value GL_BACK.
- The culling routine is turned off with glDisable (GL_CULL_FACE).

b) OpenGL - Depth-Buffer - function:

To use the openGL depth buffer visibility detection function, we first need to modify the GL utility Toolkit (GLUT) initialization function for the display mode to include a request for the depth buffer, as well as for the refresh buffer.

glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH).

→ Depth buffer values can be initialized with

glClear (GL_DEPTH_BUFFER_BIT)

By default it is set to 1.0

→ These routines are activiated with the following functions:

glEnable (GL-DEPTH TEST);

And we deactivates those depth-buffer using some routines with glDisable (GL_DEPTH_TEST);

→ we can also apply depth-buffer testing using some other initial value for the maximumdepth.

glClearDepth (maxDepth);

It can be set to any value b/w 0 & 1

As an option, we can adjust normalization values with glDepthRange (near NormDepth, far NomDepth);

→ we specify a test condition for the depth buffer routines using the following function

glDepthFunc (test condition);

→ we can set the status of the depth buffer so that if is in a read-only state or in a read write state

glDepthMask (write status);

c) OpenGL wire-frame Surface visibility Methods

→ A wire-frame displays of a standard graphics object can be obtained in OpenGL by requesting that only its edges are to be generated

glPolygonMode (GL-FRONT_AND_BACK, GL_LINE)

But this displays both visible and hidden edges.

d) open depth-curing function

glFogi (GL-FOG-MODE, GL-LINEAR)

glEnable (GL_FOG)

To increase or decrease the brightness.

7) Write the special cases that we discussed with respect to perspective projection transformation Co-ordinate

Ans

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

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

Special cases:

(i) $z_{prp} = y_{prp} = 0$

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

We get (i) when the projection reference point is limited to positions along the z_{view} axis.

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

$$x_p = x\left(\frac{z_{vp}}{z}\right) \quad y_p = y\left(\frac{z_{vp}}{z}\right) \quad \text{— (ii)}$$

We get (ii) when the projection reference point is fixed at co-ordinate origin.

(iii) $z_{vp} = 0$

$$x_p = x\left(\frac{z_{prp}}{z_{prp} - z}\right) - x_{prp}\left(\frac{z}{z_{prp} - z}\right) \quad \text{— (iii) a}$$

$$y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right) - y_{prp}\left(\frac{z}{z_{prp} - z}\right) \quad \text{— (iii) b}$$

We get (iii) a & (iii) b if the view plane as the UV plane & there are no restrictions on the placement of the projection reference point.

(iv) $x_{prp} = y_{prp} = z_{rp} = 0$

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

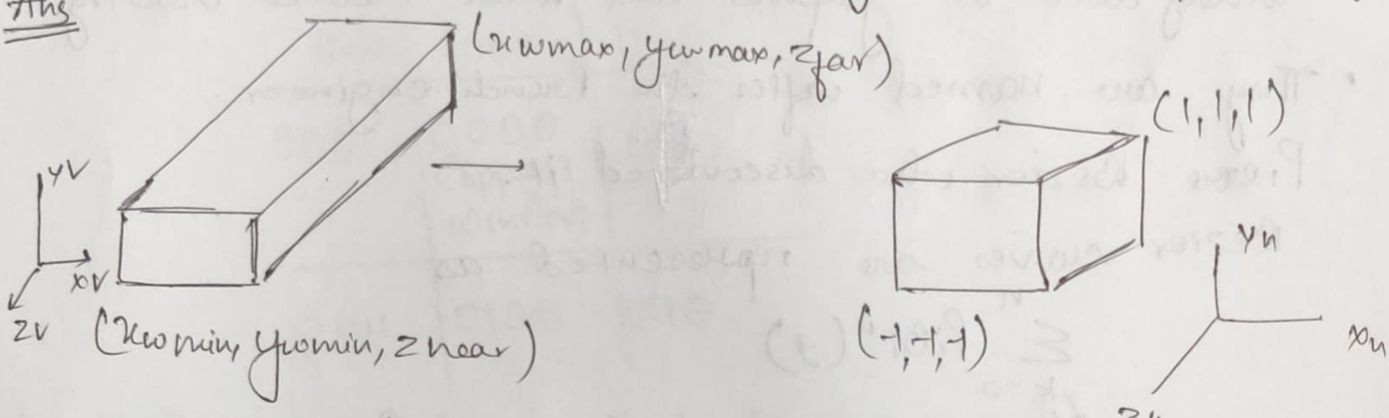
$$y_p = y\left[\frac{z_{prp}}{z_{prp} - z}\right]$$

we get (iv) with the uv plane as the view plane & the projection references point on the z view, axis.

---

(8) Explain Bezer Curve Equation along with its properties ?

Ans



Orthogonal projection View Volume

Normalized View Volume

we consider a unit cube for this normalized view volumn with each x, y, z coordinats 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

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

---

(9) Explain Bezier Curve and its properties with equations

Ans

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 Polynomial

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

n - polynomial degree

t - variable

i - index

They are be of 2 control-points - Linear Curve 3-control points - cubic curve

4 control points - quadratic curve

we used the above mentioned formadaas Bezier

$$\text{curve} = \left({}^nC_r\right) * (1-t)^{n-t} t^i \ * \text{for every point.}$$

n = control point number -1

t = 0-1 (Range)

---

(10) Explain Cohen - Sutherland Line Clipping algorithm.

Cohen Suther algorithm works on Region code

· Region code is 4- bit Code

(ABRL)

(TBRL) TOP - Bottom - Right - Left

1001	1000	1010
0001	0000 Clipping window	0010
0101	0100	0110

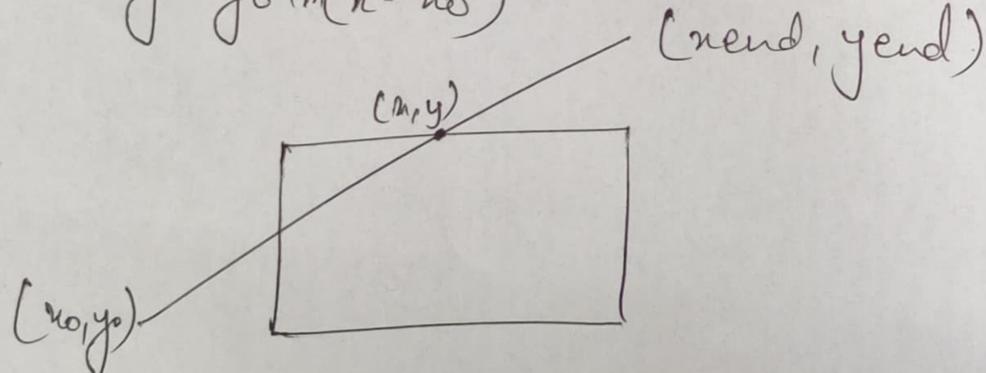
For a line - (x_0, y_0) to (x_{end}, y_{end})

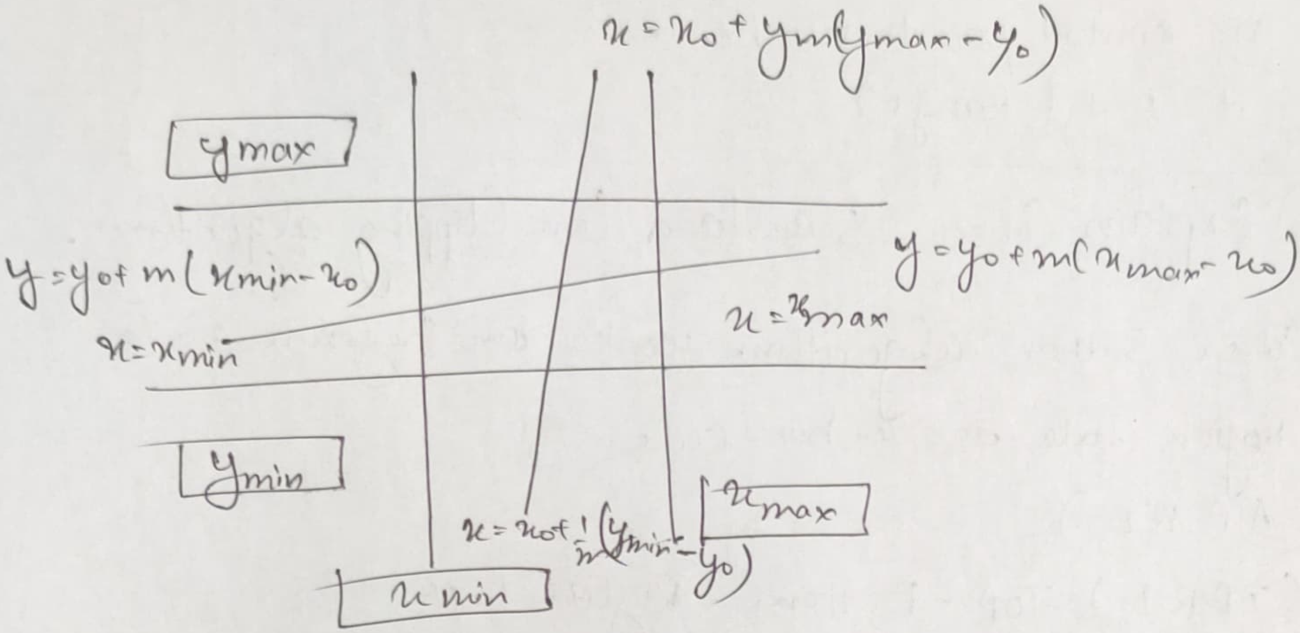
$$m = (y - y_0)/(x - x_0)$$

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

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

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





These the above formulas to be applied when a particular line needs to be clipped.