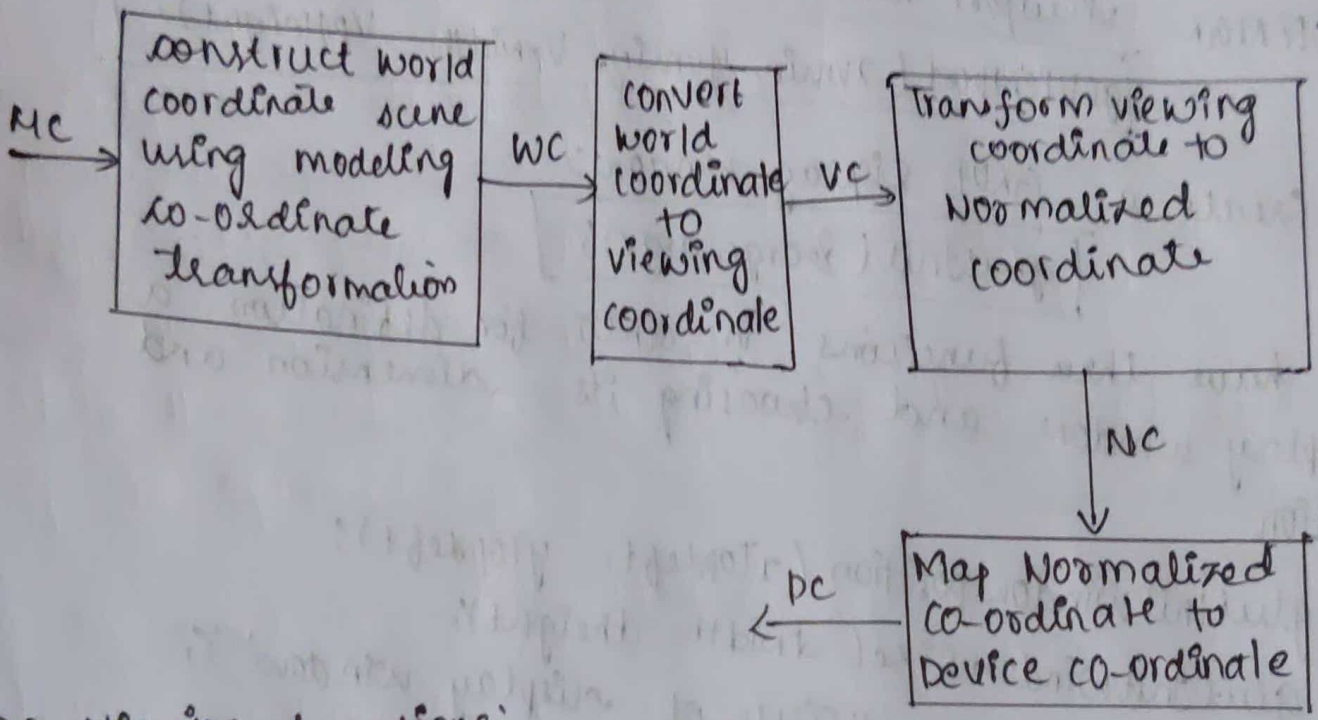
CG Assignment

NAME: Mythili N
USN: 1BY20CS119

1) Build a 2D viewing transformation pipeline and also explain open GL 2D viewing functions



2D-Viewing functions:

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

OPENGL Projection Mode:

Before we select a clipping window and a viewpoit in open GL, we need to establish the appropriate mode for constructing the matrix to transform from world coordinates to screen coordinates

```
gl Matrix Mode (GL-PROJECTION);
```

This designates the projection matrix as the current matrix, which is orginally set to Identity matrix

→ GLU clipping-window Function:

To define a 2D clipping window, we can use the OPENGL utility function.

```
gluOrtho2D ( xwmin, xwmax, ywmin, ywmax);
```

OPENGL viewport function

```
glViewport ( xvmin, yvmin, Vpwidth, VpHeight);
```

Create a GLUT Display window:-

```
glutInit ( &argc, argv);
```

We have three functions in GLUT for defination 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 parameter are selected with the GLUT function:-

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

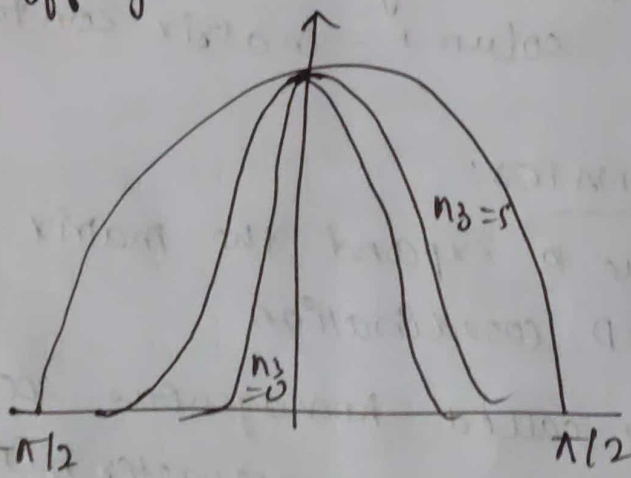
Current GLUT Display window:-

```
glutSetWindow (windowID);
```

(2.) Build phong lighting model with equations

Phong reflection is an empirical model of local illumination It decides the way a surface reflects light as a combination of the diffuse reflection of rough surfaces with the specular reflection of shiny surface.

It is based on the phongs uniform observation that shiny surfaces on p have small intense specular highlights while dull surfaces have large highlights. while dull surfaces have large highlights that fall more off gradually

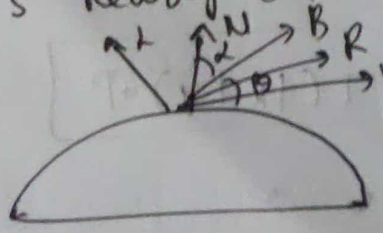


Phong model sets the intensity of specular reflection to $\cos^{n_s}\phi$

$I_{specular} = w(\theta) I_s \cos^{n_s}\phi$

$0 \le w(\theta) \le 1$ is called specular reflection coefficient

For most opeque materials specular reflection coefficent is nearly constant



$$I_{specular} = \begin{cases} K_s I_s (V \cdot R)^{n_s} & V \cdot R > 0 \ \& \ N \cdot L < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$R = (2N \cdot L)N - L$$

The Normal N may vary at each point To avoid N computation angle ϕ is replaced by angle α defined by halfway vector H between L and V

Efficient $\Rightarrow H = \frac{L+V}{|L+V|}$

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

Ans The three basics 2D transformation are translation, Rotation and scaling

$$P' = M_1 * P + M_2$$

$M_1 \rightarrow$ 2×2 array containing multiplicative factors.
$M_2 \rightarrow$ 2 elements column matrix containing translation item

HOMOGENOUS CO-ORDINATES:

A standard technique to expand the matrix 3 element representation for a 2D coordination

$(x_h, y_h, h) \rightarrow$ called homogenous coordinate
$h \rightarrow$ homogenous parameter h (non-zero value)

(x,y) is converted into new coordinate values (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:

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

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

Rotation:

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

Scaling Matrix:

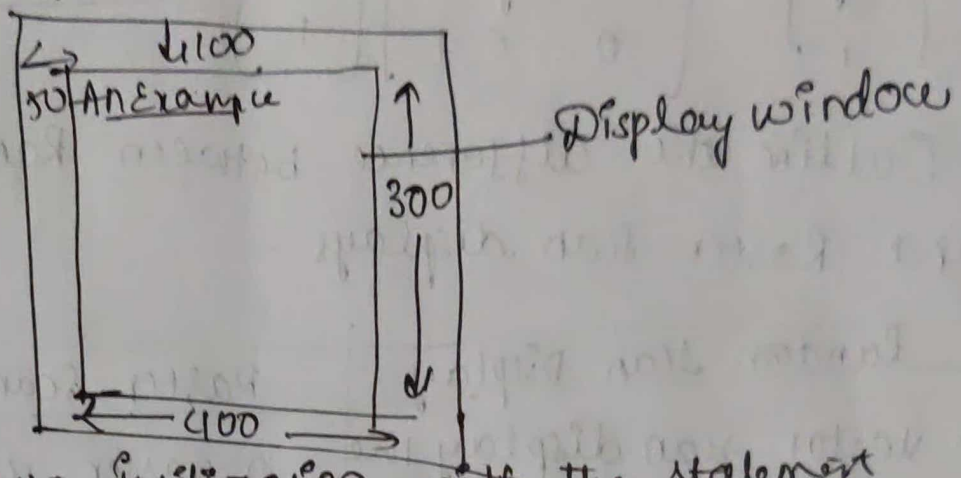
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \boxed{P' = S(sx, sy) \cdot P}$$

④ Outline the difference between Random scan Displays and Raster scan displays.

Random Scan Display	Raster-Scan Display
① In vector scan display the beam is moved between the endpoints of the graphics primitives	In raster scan display the beam is moved all once the screen and scanline at a time, from top to bottom and then break to top
2. Vector display flickers when the numbers of primitives in the buffer becomes too large	In raster display, the refresh process is independent of the complexity of the image
Scan conversion is not required	Graphics primitives are specified in terms of their endpoints and must be scan connected into their corresponding pixel in the frame buffers
Scan conversion is hardware is not required	because each primtive must be scan-converted, real-time dynamics is more

6. cost is more	cost is less
7. vector display only draws lines and characters.	Raster display has ability to display areas filled with solid colours or patterns

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



We perform the GLUT initilization with the statement

glutInit (&argc, argv);

next we can state that display window is to be created on the screen with a given captions for the title 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);

The following statement specifies that the upper left corner of the display window 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

* 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 visibilty detection Functions.

a] OpenGL - polygon calling Functions.

Back-face removal is accomplished with the functions

gl Enable (GL_CULL_FACE);

gl CullFace(mode);

* where parameter mode is assigned the value GL_BACK, GL_FRONT, GL_FRONT AND BACK

* By default, parameter mode in glcullface function has the value of GL_BACK

* the culling routine is turned off with glDisable (GL_CULL_FACE).

b] OpenGL Depth_Buffer Functions:

To use the opengl depth buffer visibity detection function we first need to modify the GL utility Toolkit (GLUT) initilization 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 0

→ These routines are activated with the following functions

glEnable(GL_DEPTH_TEST);

and we deactivate these depth-buffer routines with

glDisable(GL_DEPTH_TEST);

→ As an option, we can adjust normalization values with glDepthRange(nearNormDepth, farNormDepth);

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

glDepthFunc(test condition)

→ we can set the status of depth buffer so that if it is 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 by opengl by requesting

only this edges are to be generated.

glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)

But this displays both visible & hidden figures

d) OpenGL-DEPTH-Cuing Function

We can vary the brightness of an object as a function of its distance from the viewing position with

glEnable(GL_FOG);

glFogi(GL_FOG_MODE, GL_LINEAR)

→ This applies the linear depth function to object colours using dmin = 0.0 dmax = 1.0

glFogf(GL_FOG_END, maxDepth);

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

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

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

Special cases:

① $x_{prp} = y_{prp} = 0$

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

When projection point is limited to positions along zview axis

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

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

$$y_1 = y\left(\frac{z_{vp}}{z}\right) \text{ -(2)}$$

we get (2) when the projection reference point is fixed at co-ordinate origin

3] $z_{vp} = 0$

$$x_1 = x\left(\frac{z_{prp}}{z_{prp}-z}\right) - x_{prp}\left(\frac{z}{z_{prp}-z}\right) \text{ -(3)a}$$

$$y_1 = y\left(\frac{z_{prp}}{z_{prp}-z}\right) - y_{prp}\left(\frac{z}{z_{prp}-z}\right) \text{ -(3)b}$$

we get 3a & 3b if the view plane is the uv plane and there are no restrictions on the placement of projection of reference point

(4) $x_{prp} = y_{prp} = z_{vp} = 0$

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

(4) with the uv plane as the view plane and the projection reference points on the zview axis

⑧ Explain Bezier curve equation along with its properties

* Developed by French Engineer Pierre Bezier for use in design of Renault automobile bodies
* Bezier have a number of properties that make them highly useful for a curve and surface design They are also easy to implement.
* Bezier curve solution can be fulfilled to any number of control points.

Equation $P_k = (x_k, y_k, z_k)$

P_u = The position vector

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

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

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

Properties

* Basic functions are real
* Degree of polynomial defining the curve is one less than number of defining points
* Curve generally follows the shape of defining polygon
* Curve connects the first and last control points

thus $P(0) = P_0$

$P(1) = P_n$

(9) Explain normalization transformation for an orthogonal Projection.

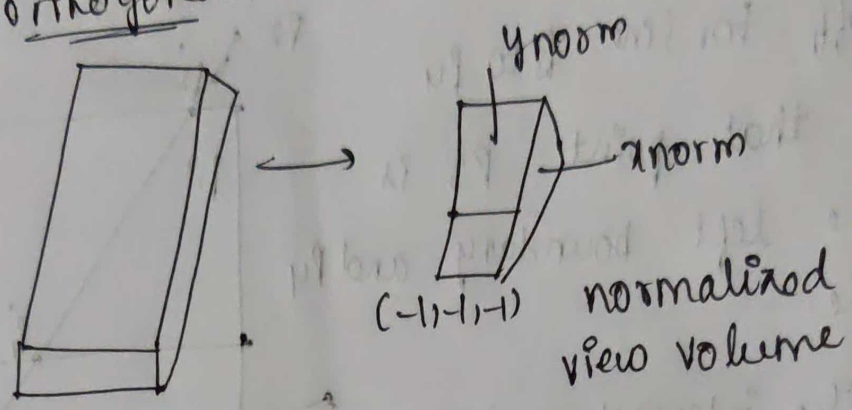
Sol The normalization transformation, we assume that the orthogonal projection view volume is to be mapped into the symmetric normalization cube within a left handed reference frame. Also z-coordinate position for the near and far planes denoted as znear & zfar this position (xmin, ymin, znear) is mapped to (-1,-1,-1) and (xmax, ymax, zfar) to (1,1,1)

The normalization transformation for the orthogonal view volume is

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

The matrix is multiplied on the right by the composible viewing transformation R.T to produce the complete transformation from world coordinates to normalize orthogonal projection coordinates.

orthogonal



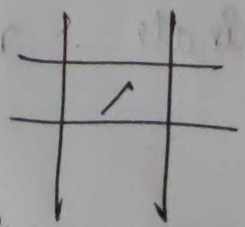
(10) Explain Cohen - Sutherland line clipping algorithm

Every line endpoint in a picture is assigned a four digit binary value called a region code and each bit position is used to indicate whether the point is inside or outside of one of the clipping window.

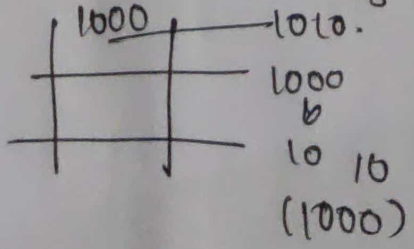
1001	1000	1010
0001	0000	0010
0101	0100	0110

once we have established codes we can determine which line is completely within window inside or outside.

when the OR operation is false then line is inside the clipping window

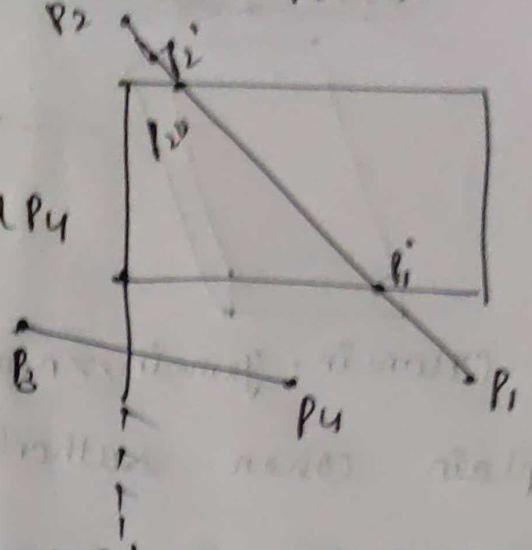


when AND operation is true then line is outside the clipping window.



The intersection to be P_2'' and P_2' to P_2'' is clipped off. For line P_3 to P_4 we find that point P_3 is outside the left boundary and P_4 is inside.

Therefore the intersection is P_3 & P_2 to is clipped off



By checking the region codes of P_3' & P_4, we find the remainder of the line or below the clipping window & can be eliminated. To determine a boundary intersection point with vertical clipping a border line can be obtained by

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

$x \rightarrow$ either x_{wmin} (or) x_{wmax}

slope $\rightarrow m = \dfrac{(y_{end} - y_0)}{(x_{end} - x_0)}$

for intersection with horizontal border, that x co-ordinate $x = x_0 + \left(\dfrac{y - y_0}{m}\right)$.