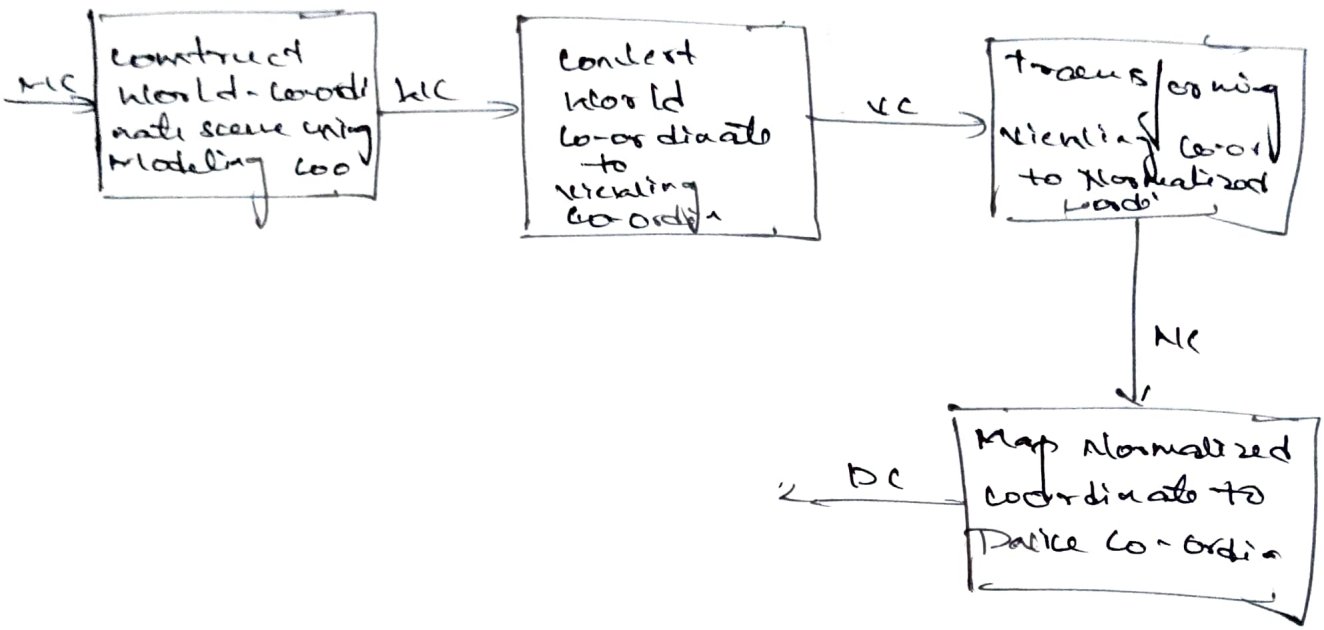
CG Assignment

NAME: Jeevitha.M
USN: 1BY21CS407

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



2D-viewing functions:

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

OpenGL Project 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 identity matrix

GLU 'Clipping-Window' function:

To define 2-D clipping window, we can use the Opengl utility function

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

Open GL Viewport function:

```
glViewport (xvmin, yvmin, vpWidth, vpHeight);
```

Create a Glut Display window:

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

We have three functions in GLUT for definiation a display window and choosing its dimension & 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);
glutClearColor (red, green, blue, alpha);
glClearIndex (index);
```

Glut Display Window identifier

windowID = glutCreateWindow ("A display window");

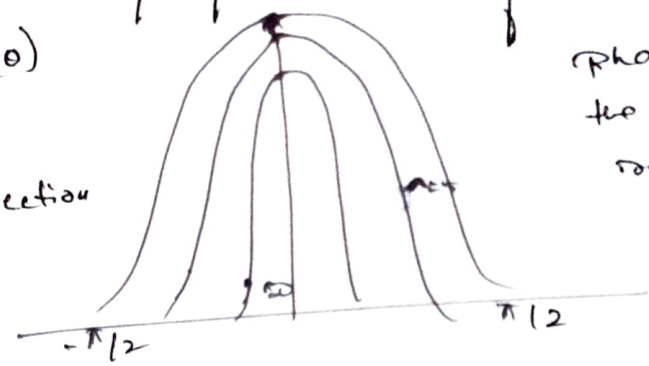
Current Glut Display Window:

glutSetWindow (windowID);

9. Build Phong Lighting Model with equations.

→

Phong reflection is an empirical model of local illumination. It describes the way a surface reflects light as a combination of the diffuse reflection of rough surfaces with the specular reflection of rough surfaces with the specular shine of surface. It is based on Phong's uniform observation that shiny structure have small intense specular highlights, while dull have small intense specular highlights that fall off more gradually



Phong model sets the intensity of specular reflection to con-

$I_{l,specular} = w(\theta)$

$0 \le w(\theta) \le 1$

Specular reflection coefficient

If light direction L and viewing direction V are on the same side of the normal N, or if L is behind the surface, specular effects do not exist

$I_{specular} = \begin{cases} k_s I_l (V \cdot R)^{n_s}, & V \text{ is on the same side as } N \text{ and } L \\ 0 & \text{otherwise} \end{cases}$

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

* The normal N may vary at each point. To avoid N computation angle ϕ is replaced by an angle α defined by a half way vector H between L and V

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

If the light source and viewer are relatively far from the object, α is constant

3] Apply homogeneous co-ordinates for translation, rotation and scaling via matrix representation.

→ The three basic 2-D transformation are

* Translation.
* Rotation
* Scaling

$P' = M_1 * P + M_2$ $P' \times P$ represents column vector

Matrix M_1 → 2×2 array containing multiplicative factors.

M_2 → 2 elements column matrix containing translation term $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$

For translation, M_1 is identity Matrix $P' = P + T$ where $T = M_2$ associated with pivot points scaling

Homogenous Co-ordinates

* A standard technique to expand the matrix representation for a 2D co-ordinate (x, y) position to a 3-element represent for a 2D coordinate (x_h, y_h, h) – is called Homogenous co-ordinates.

h – homogenous parameter h (non-zero value)

(x, y) — converted into new co-ordinate values as (x_h, y_h, h) $x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$ $x_h = x.h$ $y_h = y.h$

Translation

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

* This translation operation can be written as $P' = T(t_x, t_y) \cdot P$

Rotation

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

Scaling Matrix

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

A: Outline the difference between raster scan display & Random scan display

⟶

Random Scan Display

* In vector scan display the beam is moved between the end points of the graphics primitives.
* Vector display flicker when the number of primitives in the buffer becomes too large.
* Scan conversion is not required.

Raster Scan Display.

* In raster scan display the beam is moved all over the screen one scan line at a time, from top to bottom & then back to top.
* In Raster display the refresh process is independent of the complexity of the image.
* Graphics primitive are specified in terms of their endpoints & must be scan converted into their corresponding pixel in the frame buffer.

b) Open GL Depth-Buffer Functions:

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

glutInitDepth
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 routine are activated with the following function

glEnable (GL_Depth Test);

* And we deactivated these depth-buffer routine with glDisable (GL_DEPTH_TEST)
* It can be set to any value b/w 0 & 1

c) Open GL Wire-Frame Surface Visibility methods.

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

glPolygonMode (GL_FRONT_AND_BACK, GL_LINE)

* Vector display draws a continuous & smooth lines.

* Raster display can display mathematics smooth lines, polygon & curved primitives only by approximating them into pixels on the raster.

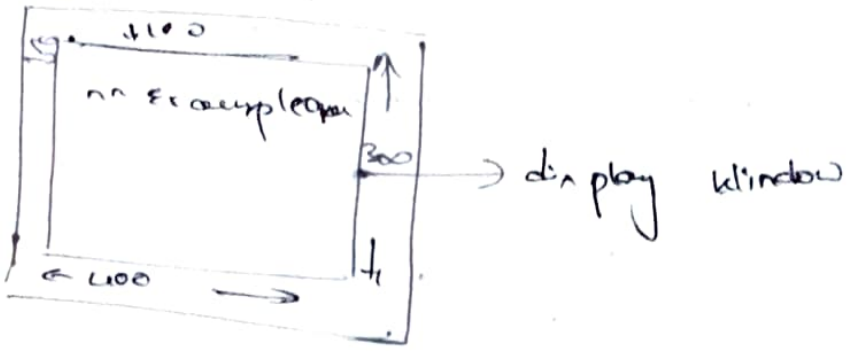
* Cost is low.

* Cost is more

* Vector display only draws lines & characters.

* Raster display has ability to display area filled with solid colours or patterns

5. Demonstrate Open GL for displaying window management using GLUT



the open GLUT initialization with the statement

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

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

→ glut CreateWindow ("An Example OpenGL Program");

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

* 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.

* glutInitWindowPosition(50,100);
it is specify that the upper left corner of the display window should be placed 50 pixel to the right of the left edge of the screen & 100 pixels down from the top edge of the screen

* glutInitWindowSize(400, 300)
- the glutInitWindowSize function is used to set the initial pixel width and height of the display window.

* glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
- the command specifies that a single refresh button buffer is to be used for the display window and that we want to use the color mode which uses red, green & blue (RGB) components to select color values.

(6) explain OpenGL visible Detection functions?

→ a] OpenGL polygon-culling functions

* Back-face removal accomplished with the function elimination. glEnable (GL_CULL_FACE)
glCullFace (Mode);

* Where parameter mode is assigned the values GL_BACK, GL_FRONT, GL_FRONT_AND_BACK

* By default, parameter mode is glCullFace function has the value GL_BACK.

* The culling routines is turned off with glDisable (GL_CULL_FACE).

4) Write a special cases that is discussed with respect to perspective projection transformation co-ordinates.

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

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

Special cases:

1. $z_{prp} = y_{prp} = 0$

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right), \quad y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) \rightarrow (1)$$

We get (1) when the projection reference point is limited to position along the z-axis i.e.

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

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

$$y_p = y\left(\frac{z_{vp}}{z}\right) \rightarrow (2)$$

We get (2) when the projection reference point is fixed at co-ordinates origin.

3] $z_{vp} = 0$

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

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

we get 3 axis the view plane is the UV plane & there are no restriction on the placement of the projection reference point.

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

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

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

we get (4) with the UV plane as the view plane & the projection reference point on the z view axis

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 makes them highly useful for curve & surface design they are also easy to implement.

* Bezier curve selection can be filled to any number of control points.

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

P_k = General $(n+1)$ Control-point position

P_u = the position vector which describes the path of an appropriate Bezier polynomial function between P_0 and P_n

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

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

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

Properties :- Basic function are real.

* Curve lies within the convex hull of the control points
* Curve connects the first and last control points
  thus $P(0) = P_0$
  $P(1) = P_n$
* Curve generally follows the shape of defining polygon

9. Explain normalization transformation for an Orthogonal projection.

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

Transforming the rectangular-parallel clipped view volume to a normalized cube is similar to the method for to a normalized cube is clipping window. Symmetric square.

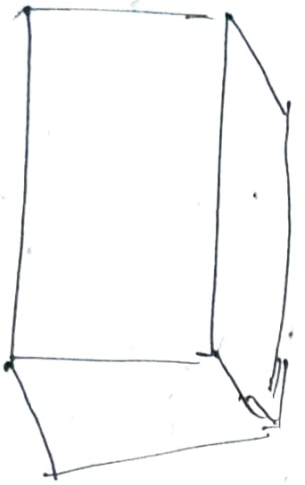
The normalized transformation for the Ortho gonal view volume is

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

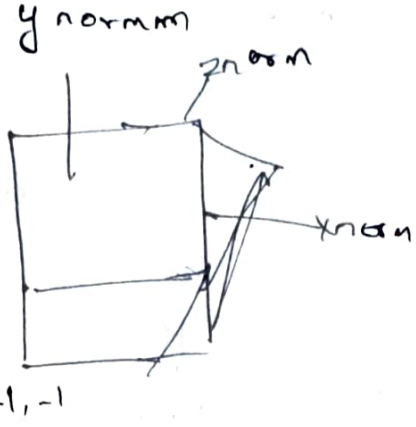
* The matrix is multiplied on the right by the composite viewing transformation R.T to produce the completely transformation from world co-ordinates to normalize orthogonal-projection co-ordinates

orthogonal - projection

(xmax, ymax, zfar)

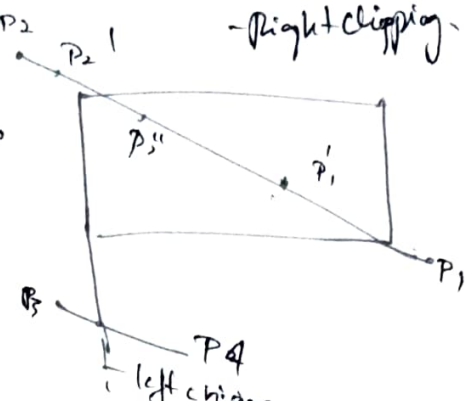


(xmin, ymin, zfar)



normalized visual volume.

9. lines that cannot be identified as being completely inside (or) completely outside a clipping window by the region codes tests are next checked for intersection with window border lines



- Right clipping.

* The intersection to the P_2' to P_2'' is clipped off the lines $P_3 = P_4$ we find that point P_3 is outside the left boundary & P_4 is inside. Therefore the intersection is $P_3 \& P_3$ if P_3' is clipped window can be eliminated. To determine a boundary intersection point with vertical clipping border we can create, obtained by

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

where x is either x_{wmin} (or) x_{wmax} & slope is

$$m = \frac{(y_{end} - y_0)}{(x_{end} - x_0)}$$

∴ for intersection with horizontal border, the x co-ordinates is $x = x_0 + \frac{(y - y_0)}{m}$

10. Explain Cohen-Sutherland line clipping algo.

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

1001	1000	1010
0001	0000 (clipping window)	0010
0101	0100	0110

Once we have established region codes for all the endpoints, we can quickly determine which line are completely within clipping window & which are clearly outside.

When the OR operation between 2 endpoints region codes for a line segment is false (0000). the line is inside the clipping window.

When AND operation between 2 endpoints region codes for a line is true, the line is completely outside clipping window