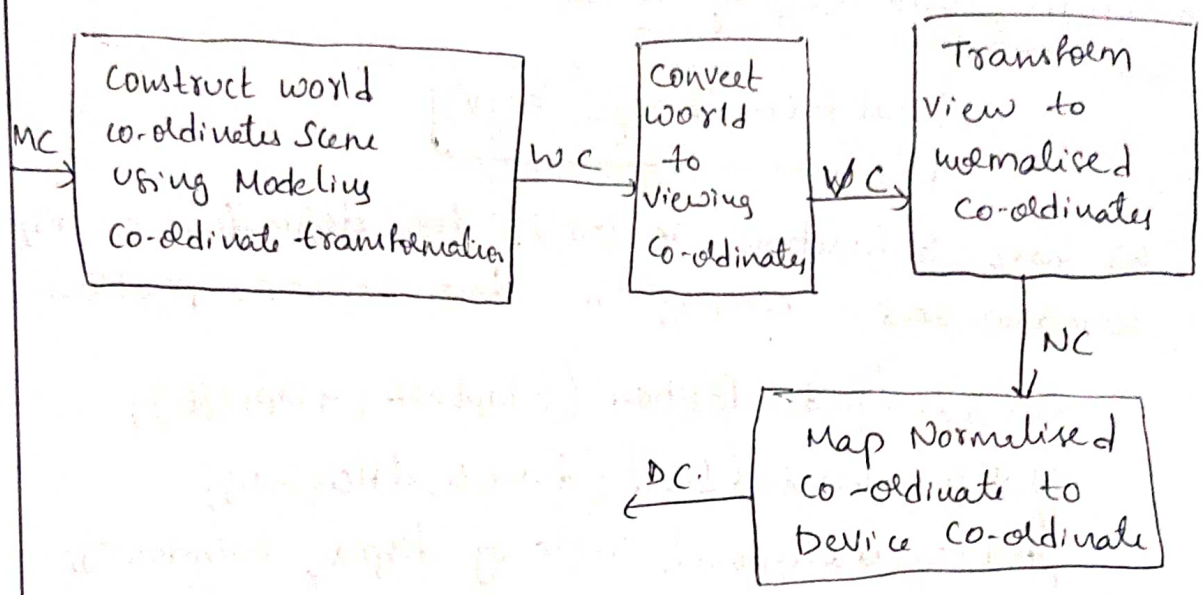
CG ASSIGNMENT

M. Hithyshi
1B420CS117
6th, CSE. B

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



2D-VIEWING FUNCTIONS

We can use these 2D routines, along with the openGL viewport function, all viewing operations we need.

OPENGL PROJECTION MODE :-

Before we select a clipping window and a viewport in openGL, we need to establish the appr. mode for openGL, contracting 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 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 3 functions in GLUT for definition a display window and choosing it's 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 GLUT function.

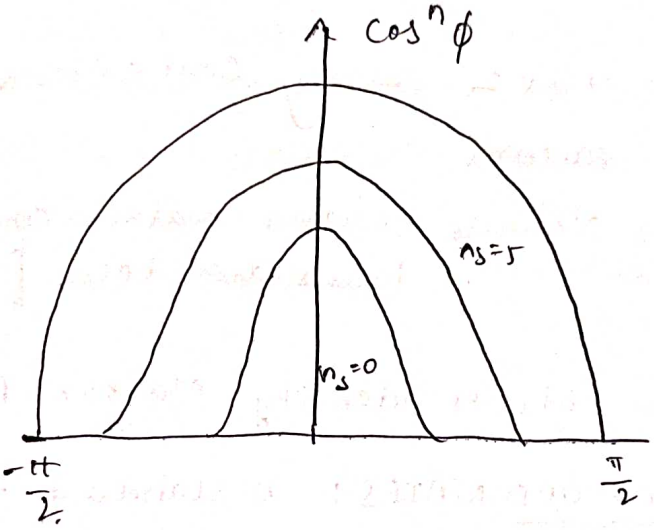
```
glutInitDisplayMode (mode);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glClearColor (red, green, blue, alpha);
glClearIndx (index);
```

GLUT display window Identifier:

```
window ID = glutCreateWindow ("A display window");
```

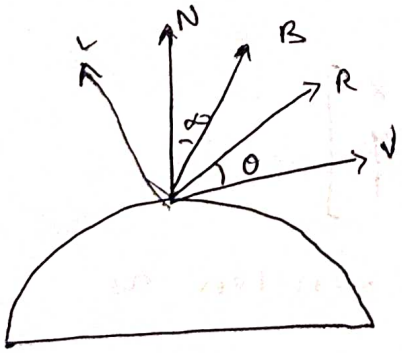
2. Build Phong Lighting Model with equations?

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 shiny surface. It is based on phong's informal observation that shiny surfaces have small intense specular highlights.



Phong model sets intensity of reflection.

If light direction L and viewing direction V are on the same side of normal N or if L is behind surface.



$$I_{specular} = \begin{cases} k_s I_l (V \cdot R)^{n_s}, & V \cdot R > 0 \text{ \& } N \cdot L > 0 \\ 0 \end{cases}$$

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

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

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

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

The 3 basic 2-D transformations are translation, rotation & scaling.

$$P' = M_1 \cdot P + M_2$$

P' & P represents column vectors.

Matrix M_1 ⟶ 2 x 2 array containing multiplicative factors

M_2 ⟶ elements column matrix containing translation terms $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$

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

HOMOGENOUS CO-ORDINATES: A standard technique to expand the matrix representation for a 2D co-ordinate (x, y) position position to a 3 element representation.

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

Rotation :-

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

Scaling Matrix :

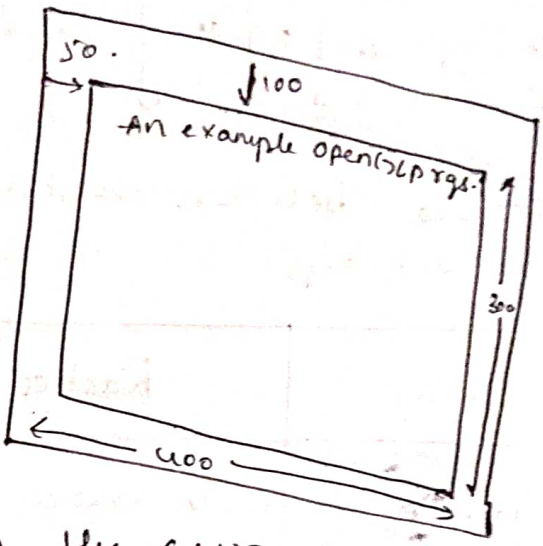
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = S(S_x, S_y) \cdot P$$

6. Ep. Outline the difference between raster scan displays and random scan displays.

Random Scan Display	Raster Scan Display.
1. In vector scan display the beam is moved between the end pts 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 no. of primitives in the buffer becomes too large.	2. In raster display, the refresh process is independent of the complexity of image
3. Scan conversion is not required.	3. Graphics primitives are specified in terms of their endpoints and must be scan connected.
4. Scan conversion h/w is not required.	4. Because each primitive must be scan-converted, real time dynamics is lost more conversion h/w.
5. Cost is more	5. Cost is low.
6. Vector display only draws lines and characters	6. Raster display has ability to display areas filled with solid colors or patterns.

Transformation for an Orthogonal

5. Demonstate opengl functions for displaying window management using GLUT.



* we perform the GLUT initialization with the statement

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

* next, we can state that a display window is to be created on the screen that a given caption for the title bar.

→ 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 to an infinite loop that checks for input from devices such as mouse or keyboard.

* glutInitWindowPosition (50,100);
The following statement specifies that corner of the display window should be placed 50 pixel to the right of the left-edge.

normalization transformation for an Orthogonal

* 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 buffer is to be used for the display window and that we convert to use the color mode which uses red, green to select Color values.

6. Explain OpenGL Visibility Detection Functions?

a) OpenGL Polygon - Culling Functions

Back-face removal is accomplished with the functions

glEnable (GL_CULL_FACE);
glCullFace (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 GL_BACK.
* The culling routine is turned off with glDisable (GL_CULL_FACE).

b) OpenGL Depth-Buffer Functions:

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 initialised with

glClear (GL_DEPTH_BUFFER_BIT)

* By default it is set to 1.0.

=> These routines are activated with following functions:

```
glEnable(GL_DEPTH_TEST);
```

And we de-activate these depth-buffer routines with glDisable (GL_DEPTH_TEST).

=> We can also apply depth-buffer testing using some other initial value for max depth.

```
glClearDepth(maxDepth);
```

3. Write special cases that we 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 projection reference point is limited to positions along the Z axis.

1. For an Orthogonal

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

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

$$y_p = y\left[\frac{z_{vp}}{z}\right] \longrightarrow (1).$$

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

3] $z_{vp} = 0$

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

$$y_p = y\left[\frac{z_{prp}}{z_{prp} - z}\right] - y_{prp}\left[\frac{z}{z_{prp} - z}\right] \longrightarrow (3)\ b.$$

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

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

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

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

We get (4) with the uv plane as the view plane & projection references point on the z view. axis.

9. 8. Explain Bezier Curve Eqn along with its properties.

* Developed by French engineer Pierre Bezier for use in design of Renault automobile bodies.
* Beziers have a number of properties that make them highly useful for curve and surface design. They are also easy to Implement.
* Bezier Curve section can be fitted to any no. of control points.

Equation:-

$P_k = (x_k, y_k, z_k)$ P_k = General $(n+1)$ Control-point positions.

P_U = the position vector which describes the path of an approximate Bezier polynomial function between P_0 and P_n.

$$P[U] = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(U) \qquad 0 \le U < 1$$

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

Properties:-

* Basic functions are real.
* Degree of polynomial defining the curve is one less than no. of defining points.
* Curve generally follows the shape of defining polygon.
* Curve connects the first and last control points

thus $P(0) = P_0$

$P(1) = P_n$

* Curve lies within the convex hull of control points.

9. Explain normalization transformation for an Orthogonal Projection.

The normalization transformation, we assume that the orthogonal-projection view volume is to be mapped into the symmetric normalisation cube within a left handed reference frame. Also, z-coordinate positions for the near and far planes are denoted as z_{near} and z_{far}.

Transforming the rectangular-parallelepiped view volume to a normalised cube is similar to the method for converting the clipping window into the normalised symmetric square.

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

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