Name - Kumar Kushal
USN → 1BV20CS222
Subject → CGV, 18CS62
~~Class & S~~
Sem & Sec → VI/B

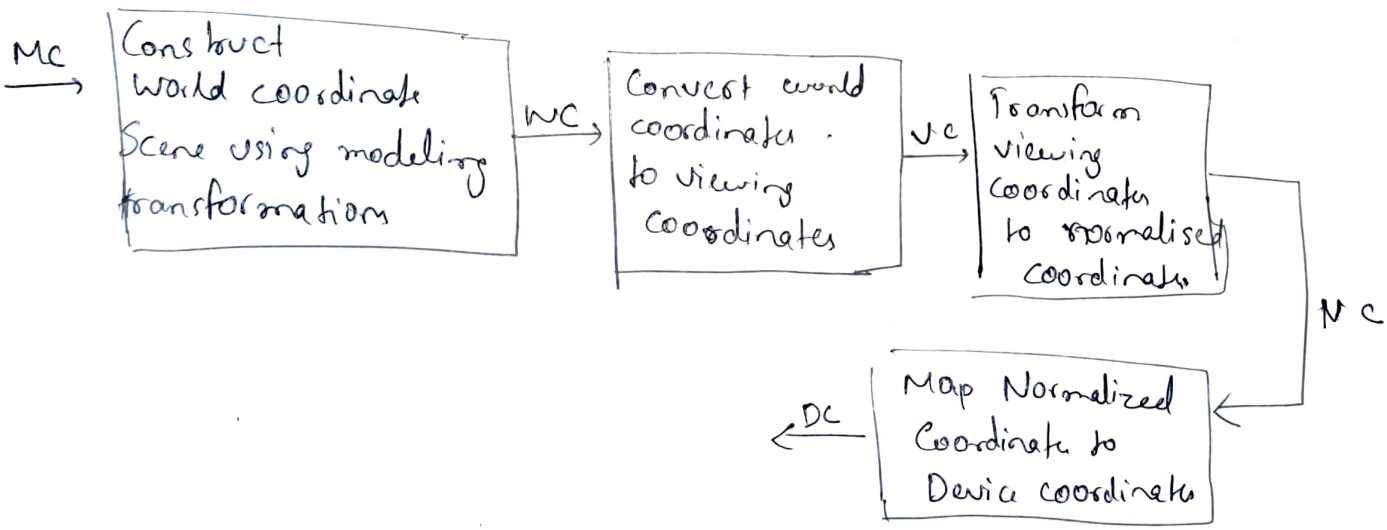
CGV Assignment

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

= Viewing pipeline means the mapping of a two dimensional world-coordinate scene description to device coordinates is called a two dimensional viewing transformation.

This transformation is simply referred to as the window to viewport transformation or the windowing transformation

We can describe the steps for 2D viewing pipeline as indicated in figure;



Once a world coordinate scene has been constructed, we would set up a separate two dimensional viewing coordinate reference frame for specifying the clipping window.

To make the viewing process independent of the requirements of any output device, graphics system convert object descriptions to normalized coordinates.

OpenGL 2D Viewing Functions

The GLU library provides a functions for specifying a two-dimensional clipping window, and we have GLUT library functions for handling display windows

(i) OpenGL projection mode

We must set the parameters for the clipping window as part of the projection transformation function.

```
glMatrixMode (GL_PROJECTION);
```

We can also set the initialization as

```
glLoadIdentity ();
```

(ii) GLU clipping window function:

To define 2D clipping window, we can use the GLU function

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

The normalized coordinates in the range from −1 to 1 are used in the openGL clipping routine

(iii) OpenGL viewport function

We specify the viewport parameter with the OpenGL function

```
glViewPort (xvmin, yvmin, vpWidth, vpHeight);
glGetIntegerv (GL_VIEWPORT, vpArray);
```

2. Build phong lighting Model with equations.

- A phong lighting model is a local model that can be computed rapidly. It's an empirical model for calculating the specular reflection range, developed by phong.

Angle ϕ can be assigned values in the range 0° to 90°, so that $\cos\phi$ varies from 0 to 1.0.

It has 3 components:

(i) Ambient Component

The component approximates the indirect lighting by a constant

$$I = I_a * K_a$$

where, I_a = Ambient light intensity (color)
K_a = ambient reflection const (cons)

(ii) Diffuse Component

It describes the diffuse reflection of rough surfaces.

$I = I_p * K_d * \cos\theta$ where, I_p = intensity of point light source
K_d = diffuse reflection coefficient
$\cos\theta$ = lambertian cosine law

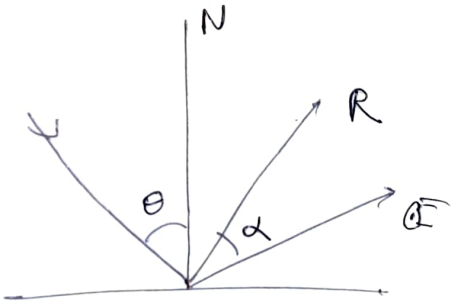
(iii) Specular Component

It describes the specular reflection of smooth (shiny) surfaces.

$I = I_p * K_s * \cos^n\alpha$, n = shininess

where I_p = intensity at the point light source
K_s = specular reflection coefficient (0-1)

Similarly from the figure,



$\cos\theta = N \cdot L$
$\cos\alpha = RV$

So,

$I = I_p * K_d * NL$ (Diffusion)
$I = I_p * K_s * (RV)^n$ (Specular)

Therefore, Phong model is

I = Ambient + Diffusion + Specular

$\Rightarrow I = I_a K_a + I_p K_d \cos\theta + I_p K_s \cos^n\alpha$

It also can be written in term of normal reflection & view vector as

$$I = I_a K_a + I_p K_d NL + I_p K_s (RV)^n$$

This gives the perfect phong model for a object under the illumination

3. Apply Homogeneous coordinate for translation, rotation, and scaling via matrix representation.

= A 2D point P in represented in homogenous coordinate by a 1 dim

vector $P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ or $P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

We use the homogeneous coordinate its eliminate addition. Hence, we use a point homogenous coordinate, we don't close anything, it is easier its compose of everything in matrix multiplication

(i) Translation:

In 2D: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix} = P' \begin{bmatrix} x' \\ y' \end{bmatrix}$

In 3D: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

(ii) Rotation:

$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow$ clockwise

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

(iii) Scaling:

$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

In all transformation x, y are the current points of x', y' are the receiving points after the transition, rotation & scaling.

The 3D scaling can done as below in homogenous coordinate.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

4. Outline the difference between raster scan display and random scan displays.

=

Random Scan Display	Raster Scan Display
1. The resolution is higher than raster scan display	The resolution of raster scan is lower than random scan display.
2. It is more expensive & less affordable	It is affordable as compared to random scan display.
3. In a random scan display, it is easy to proceed with modification.	In it, it is difficult to do any modifications.
4. In it, we don't prefer interacting	In raster scan display, we prefer interlacing.
5. When it comes to image rendering, we prefer mathematic functions	When it comes to image rendering, we prefer pixels then it is good for continuous lifelike scenes.
6. eg: Pen Plotter	eg: TV sets

5. Demonstrate OpenGL function for displaying window management window management using GLUT

= Since we are using the OpenGL utility toolkit, we need its initialize GLUT. Steps 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 be created on the screen with a given option for the title bar.
• glutCreateWindow ("An example");

→ Then the following function add scene the line segment description to the display window.
• glutDisplayFun (lineSegment);

eg:-
```
#include <GL/glut.h>
void display()
{
    glClearColor (0,0,0,1);
    glClear (GL_COLOR_BUFFER_BIT);
    glBegin (GL_TRIANGLES);
    glColor3f (1,0,0);
    glVertex2f (-0.8, -0.8);
    glColor3f (0,1,0);
    glVertex2f (-0.8, -0.8)
    glColor3f (0,0,1);
    glVertex2f (0.0, 0);
    glEnd();
    glutSwapBuffers();
}
```

```
int main (int argc , char ** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500,500);
    glutInitWindowPosition (100,100);
    glutCreateWindow ("GL RGB triangle");
    glutDisplayFunc (display);
    glutMainLoop ();
}
```

6. Explain the OpenGL visibility detection function

= The OpenGL has the different function for the visibility detection in GLUT library. They are:

(i) OpenGL polygon culling function

This function is used to remove back face, front face or both face of the object.

```
glCullFace (mode);
```

* parameter mode is;

```
glEnable (GL_CULL_FACE);
glEnable (GL_CULL_FACE);
```

(ii) OpenGL Depth - buffer function

```
• glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH)
```

This initialization function will request for depth buffer and refresh buffer.

```
glClear (GL_DEPTH_BUFFER_BIT);
glEnable (GL_DEPTH_TEST);
glEnable (GL_DEPTH_TEST);
```

(iii) OpenGL transform surface visibility function

This function is used for wire frame display of the object, But display both visible and hidden edges.

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
```

(iv) OpenGL Depth Fuzzy function

```
glFogi(GL_FOG_MODE, GL_LINEOF);
```

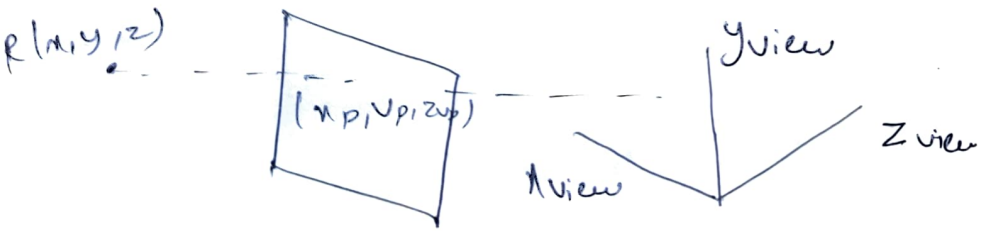
This function is used to vary the brightness of an object. It applies linear depth function to object colors using $d_{min} = 0.0$ & $d_{max} = 1.0$ by default.

Thereby using above function, we can detect the visibility of objects.

7. Write the special cases that we encountered with respect to perspective projection transformation coordinates.

= The projection line intersects the view plane at the coordinates position (x_p, y_p, z_{vp}) where z_{vp} is some selected position for the view plane on the zview axis.

Fig. below shows the projection path of a spatial position (x, y, z) to a general projection reference point at $(x_{prp}, y_{prp}, z_{prp})$:



We can write equation describing coordinate position along this perspective projection line in parametric form as

$$x' = x - (x - x_{prp})u$$
$$y' = y - (y - y_{prp})u$$
$$z' = z - (z - z_{prp})u$$

On the view plane, $z' = z_{vp}$, we can write u as,

$$u = \frac{z_{vp} - z}{z_{prp} - z}$$

By substituting this value of u into the equations for x' & y' we obtain the general perspective transformation equations

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

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

Special cases:

Case I: The projection reference point could be limited its position along the z view axis, then

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

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

Case II: Sometimes the projection reference point is fixed at the coordinate origin as $(x_{prp}, y_{prp}, z_{prp}) = (0, 0, 0)$ then,

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

Case III: if the view plane is the u-v plane and there are no restriction on the placement of the projection reference point then we have,

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

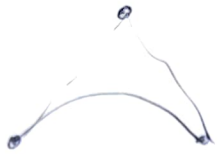
$$y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right) - y_{prp}\left(\frac{z}{z_{prp} - z}\right)$$

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

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

8. Explain Bezier curve equation along with its properties (10)

= Bezier curve is drawn by considering control points. Approximate target is drawn by using control point



3 control points of Bezier curve

Simplest form of bezier curve is connection line endpoints

→ Bezier curve eqn is also known as Bezier spline curve it is developed by french engineer Pierre Bezier.

Bezier curve can be fitted to any number of control points although some graphic package limit to four control point.

→ Bezier curve equation

$$C(u) = \sum_{K=0}^{n} P_K \cdot BEZ_{K,n}(u)$$

where P_K = position vector coordinate $K=0,\dots n$

$BEZ_{K,n}(u)$ = bezier blending function

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

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

→ A set of 3 parametric equations for the individual curve coordinates can be represented as

$$x(u) = \sum_{K=0}^{n} x_K \cdot BEZ_{K,n}(u)$$

$$y(u) = \sum_{K=0}^{n} y_K \cdot BEZ_{K,n}(u)$$

$$z(u) = \sum_{K=0}^{n} z_K \cdot BEZ_{K,n}(u)$$

Bezeir Curve properties

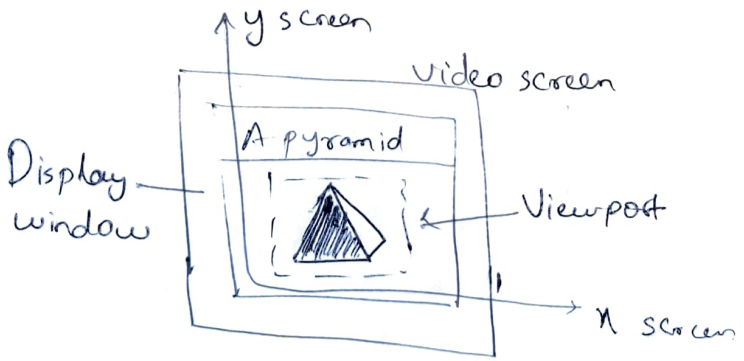
→ depends on number of control points.
→ curve will pass through end points, but not all control point.
→ polynomial degree of curve depends on no. of control point
n → control points (4)
n-1 → degree of polynomial equation (3)

9. Explain normalization transformation for an orthogonal projection

= Once we have established the limits of the view volume, coordinate description inside this rectangular parallelepiped are the projection coordinates and they can be mapped into a normalized view volume without any further projection processing.

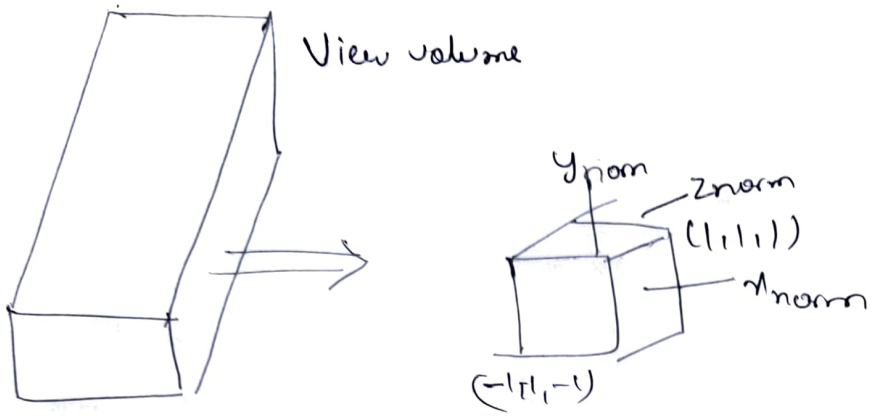
Some graphics use a unit cube for this normalized view volume with each of the x, y and z coordinates normalized in the range from 0 to 1.

Another normalization transformation approach is to use a symmetric cube, with coordinates in the range from -1 to 1.



To illustrate the normalization, 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 positions for the near and far planes are denoted as z_{near} and z_{far} respectively.



The normalization transformation for the orthogonal view volume is

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

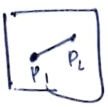
10. Explain cohen-sutherland line clipping algorithm.

= Cohen-Sutherland algorithm works on Region code.

Region code is 4 Bit code (ABRL; above below right left)

1001	1000	1010
0001	(Window) 0000	0010
0101	0100	0110

Case 1: if both endpoint region code is zero, completely inside & visible



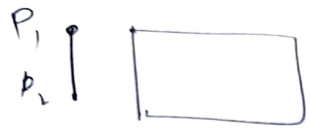
$P_1 = 0000$ (zero)

$P_2 = 0000$ (zero)

AND 0000 (zero)

Case 2: If both end point region code is non zero, apply the logical AND and result is non zero

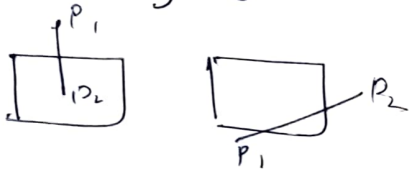
Completely outside & invisible



$P_1 = 0001$
$P_2 = 0001$
AND 0001

Case 3: a) if one of P_1 & P_2 is zero
b) Both non-zero
} logical AND is zero

Find the intersection point



Finding intersection points

(1) Find y value of vertical line

Consider a line segment (x_1, y_1) (x_2, y_2) & intersection point (x, y)

⇒ Find slope of (x_1, y_1) & (x, y)

$$m = \frac{y - y_1}{x - x_1}$$

then $(y - y_1) = m(x - x_1)$

$$\therefore \boxed{y = y_1 + m(x - x_1)}$$

If $x = x_{wmin}$ & $x = x_{wmax}$

(2) Find x value of horizontal line

Firstly slope of (x_1, y_1) & (x, y)

$$m = \frac{y - y_1}{x - x_1}$$

$$m(x - x_1) = y - y_1$$

$$\therefore \boxed{x = x_1 + \frac{y - y_1}{m}}$$