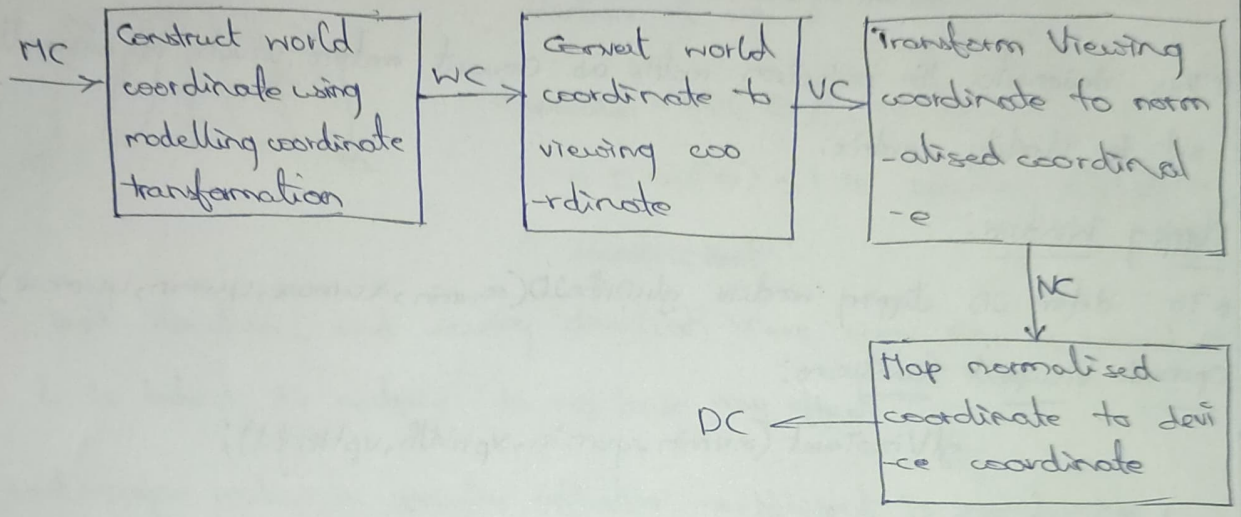
CG ASSIGNMENT

RANGI SARAN SRIPADH

6TH B, CSE

1BY20CS146

1. Build a 2D transformation pipeline and also explain Open GL 2D Viewing functions.



* A section of 2-D scene that is selected for display is called clipping window and all parts outside are clipped off.

* Mapping 2-D world coordinate scene description to device coordinate is called 2D viewing transformation or window to viewport transformation.

* Once the world-coordinate scene has been constructed, we could set 2D viewing coordinate reference frame for specifying clipping window.

* To make viewing process, independent of requirement of output

device, system cannot convert description to normalized coordinate from 0 to 1 and others use -1 to 1

*Finally normalized coordinate map to device coordinate.

Viewing Functions:

OpenGL projection mode:

```
glMatrixMode(GL_PROJECTION);
```

*This designates the projection matrix as current matrix, which is originally set to identity matrix.

Clipping Window:

*To define 2D clipping window gluOrtho2D(xwmin, xwmax, ywmin, ywmax)

OpenGL Viewport functions:

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

Create GlutDisplay Window:

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

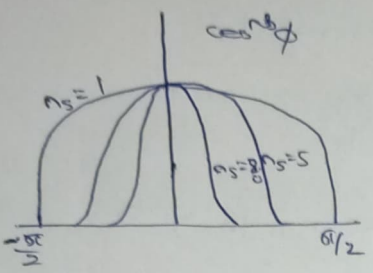
Display Window:

```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitDivideSize(500, 500);
glutInitWindowPosition(0, 0);

glClearColor(r, g, b, a);
glClearIndex(index);
```

(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 combination of diffuse reflection of rough surface with specular of shiny surface. It is based on Phong's informal observation that shiny surface have small intense specular highlights while dull surface have larger highlights that off gradually.



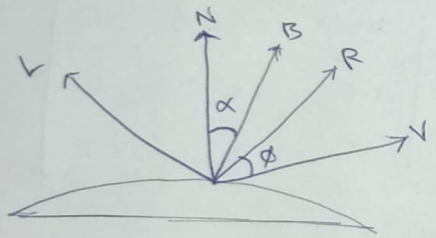
* Phong model sets the intensity of specular reflection of $\cos^{ns}\phi$

$$I_{specular} = w(\theta) I_l \cos^{ns}\phi$$

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

* If light direction L and viewing direction V are same side of normal, or if L is behind the surface do not have any effect.

For most opaque materials specular reflection coefficient is nearly k_s



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

Normal N may vary at each pt. To compute it's.

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

If the distance b/w light source and viewer are relatively far then α is constant.

it is direction yielding maximum specular reflection in viewing direction V if surface normal N would coincide with H. If V is coplanar with R and L then $\alpha = \phi/2$

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

Translation: It moves all points in an object along same straight line path to new position.

Path is represented by vector

$$P_x' = P_x + t_x$$

$$P_y' = P_y + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

In homogenous coordinates:

(x_h, y_h, h)

$$x = \frac{x_h}{h}$$

$x_h = x * h$; $y_h = y * h$

consider $h = 1$

Rotation: It repositions all points in an object along a circular path in plane centered at pivot center.

$\cos\phi = x/r$ $\quad \sin\phi = y/r$

$x = r\cos\phi$ $\quad y = r\sin\phi$

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

$P' = R \cdot P$ where $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

In homogenous coordinate:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling: It is used to change the size of an object and involve two scaling factor s_x & s_y

$$P_x' = S_x \cdot P_x$$

$$P_y' = S_y \cdot P_y$$

$$P' = S \cdot P$$

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

In Homogenous coordinate:

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4. Outline the difference b/w raster and random scan displays.

Raster Scan Display:

* Electron beam is swept across one row data from top to bottom.
* As it moves across each row, beam intensity is turned on & off to create a pattern of illuminated spots.
* Scanning process called refreshing. Complete scanning of screen is normally called frame.
* Refreshing rate is frame rate, It is 60 to 80 Hz
* Picture definition is stored in a memory area called frame buffer. This stores the intensity values for all screen points called pixel.

Random Scan Display:

* When operated as random scan display unit CRT has electron beam distributed only to those pairs of screen where the picture is to be displayed.
* Pictures are generated as line drawings with electron beam tracing out the component lines one after the other.
* Also known as vector displays. Component line of picture can be drawn refreshed by a random scan system in any specified order.
* Refresh rate depends on the number of lines to be displayed on that system.

Raster Scan Display

Random Scan Display

5- Display Window Management using GLUT

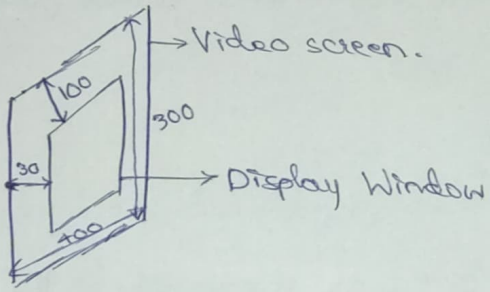
Step 1: Initialization of GLUT

* We use GLUT
* We perform initialization - glutInit(&argc, argv);

Step 2: Title.

glutCreateWindow("An example");

There is single argument for this function that can be any character string that we want character string that we want to display to window title.



glutDisplayFunc(lineSegment);

It is used to call the display function repeatedly. Name of the function to be called line segment.

glutMainLoop();

It is last line. It displays the initial graphic and put program into infinite loop.

glutInitWindowPosition();

It is used to specify upper left corner of the display window.

glutInitWindowSize;

It is used to set initial pixel with height & width of display window.

glutInitDisplaymode(GLUT-SINGLE|GLUT-RGB):

It is used for color modes like red, green & blue.

6. Explain OpenGL visibility detection functions.

a. OpenGL polygon filling functions:

Backfall removal with functions.

```
glenable (GL - CULL-FACE);
glCullFace (Mode);
```

mode can be GL_BACK, GL-FRONT, GL_FRONT_AND_BACK

Disable with.

```
gl Disable (GL-CULL-FACE);
```

b. OpenGL DepthBuffer functions:

To use openGL depthBuffer visibility detection function, we need to modify GLUT initialization function.

```
glutInitDisplayMode (GLUT_SINGLE | GLUT-RGB | GLUT_DEPTH);
glutClear (GL_DEPTH_BUFFER_BIT);
```

To Disable the depth-test:

```
glDisable (GL_DEPTH-TEST);
```

c. OpenGL wireframe surface visibility method:

A wireframe display can be obtained in OpenGL by requesting that only its

```
glPolygonModel (GL-FRONT_AND-BACK, GL-LINE);
```

d. OpenGL_DEPTH-CUEING-function:

It is used to vary the brightness of an object as a function of its distance from viewing position with.

```
glEnable (GL_FOG);
glFogi (GL_FOG_MODE|GL_LINEA);
```

7. Write special cases that are discussed with respect to perspective projection.

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

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

Cases:

1. Projection reference point is limited along z view axis – $x_{prp} = y_{prp} = 0$

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

2. When projection reference point is at coordinate origin.

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

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

3. If view point is uv plane and no restriction on placement of projection.

$$z_{vp} = 0 \qquad x_p = x\left[\frac{z_{prp}}{z_{prp} - z}\right] - x_{prp}\left[\frac{z}{z_{prp} - z}\right]$$

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

4. If uv plane is uv projection reference point on z view axis.

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

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

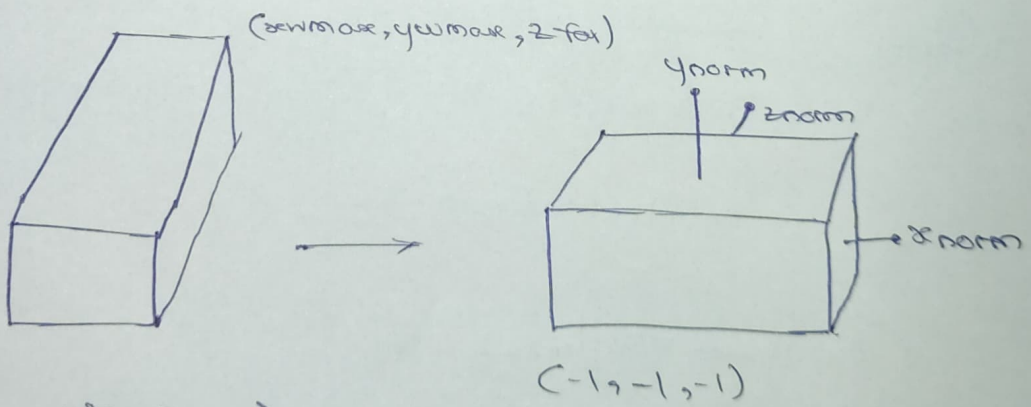
7. Explain normalisation transformation for orthogonal projection.

We assumed that orthogonal projection view volume to mapped into symmetric normalisation cube within left-handed reference frame. Also z-coordinate positions for handed reference frame. Also z-coordinate for near & far positions. Is denoted as z_{near} & z_{far} respectively. This position $(x_{min}, y_{min}, z_{near})$ is mapped to normalized position $(-1,-1,-1)$ & position $(x_{max}, y_{max}, z_{max})$ is mapped to $(1,1,1)$

Normalisation transformation for view volume.

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

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

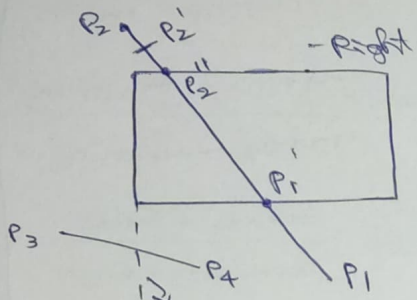


10. Explain cohen sutherland's line clipping algorithm.

Every line endpoint in picture is designed with four digit binary code called region code and each bit is used to indicate whose point lies inside or outside once we establish region code for all lines endp-oint, we determine where they lie completely inside or not.

1001	1000	1010
0001	0000	0010
0101	0100	0110

When OR opn b/w 2 endpoints is false, line is inside window. AND opn b/w 2 endpoints is true, completely outside clipping window. If it is not completely inside, it lies partially outside.



Intersection 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 left boundary & P_4 is inside. Therefore intersection P_3, P_3' to P_3' is clipped off.

To determine a boundary for line eq, the y-coordinate intersecti-on point with vertical clipping line can be obtained by.

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

$$m = y_{end} - y_0 / x_{end} - x_0$$

$$x = x_0 + \left(\frac{y - y_0}{m}\right)$$