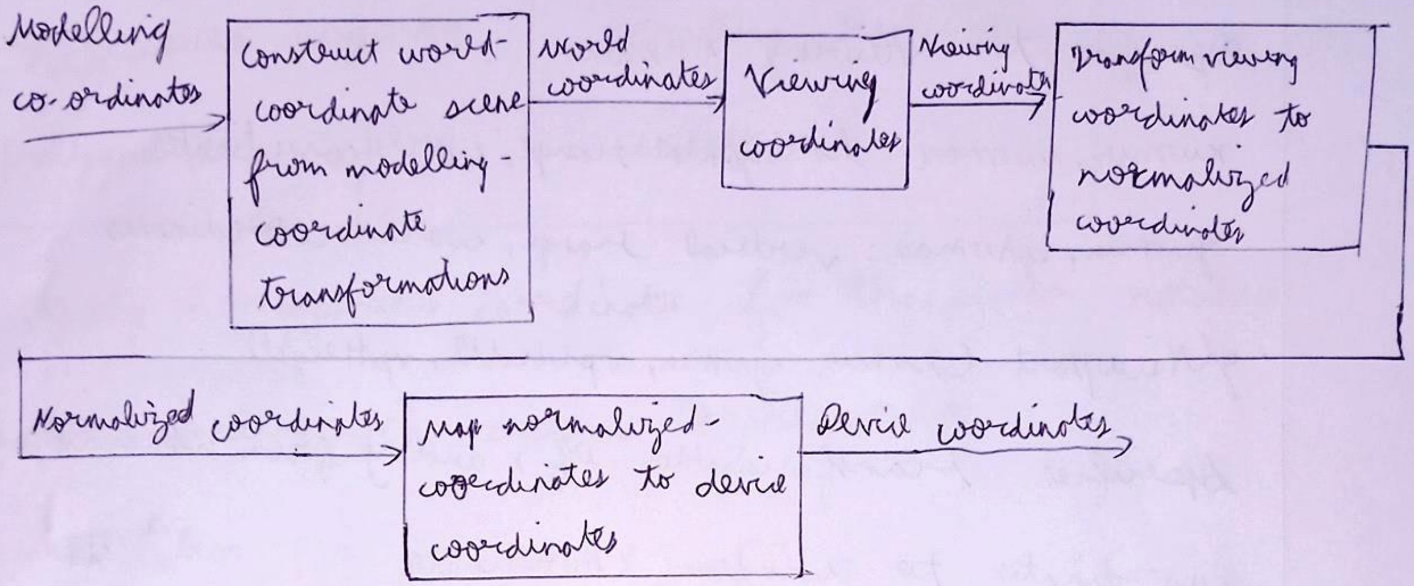
Mohammed Sulaiman Khan K
6-B , CSE
1BY20CS115

Assignment

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

A: 

Change modelling coordinates to world coordinates by applying modelling transformations. Change world coordinates to viewing coordinates ~~to~~ by determining visible parts, change viewing coordinates to normalized coordinates and further to device coordinates by clipping and determining pixels.

OpenGL 2D viewing functions

· glMatrixMode (~~GL~~ - PROJECTION)

It sets the current matrix mode.

It can assume one of the ~~four~~ two values.

GL-MODELVIEW

Applies subsequent matrix operations to modelview matrix stack.

GL_PROJECTION

Applies subsequent matrix operations to projection matrix stack

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

Specifies the viewing window

xwmin, xwmax: horizontal range, world coordinates

ywmin, ywmax: vertical range, world coordinates

- glViewport (xvmin, yvmin, vpWidth, vpHeight)

Specifies transformation of x and y from normalized coordinates to window coordinates

4. Outline the differences between raster scan displays and random scan displays

A:

Random scan	Raster scan
The resolution of random scan is higher than raster scan.	while the resolution of raster scan is lower than random scan
It is costlier than raster scan	Cost is lesser
~~raster scan~~ Alteration is easy in comparison of raster scan	Any alteration is not easy.

Interweaving is not used	Interweaving is used.
It is suitable for applications requiring polygon drawings	It is suitable for creating realistic scenes

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

A: Translation $P' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

Rotation $P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Scaling $P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Each cartesian co-ordinate (x, y) with homogeneous co-ordinate (x_h, y_h, h) where $x = x_h/h$, $y = y_h/h$

$(h * x, h * y, h)$

set $h = 1$

$(x, y, 1)$

Homogeneous coo-ordinate representation for translation, scaling and rotation are as follows:

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

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

8 Explain Bezier curve equation along with properties

A: For $n+1$ control-point positions, denoted as $p_k = (x_k, y_k, z_k)$, with k varying from 0 to n. These coordinate points are blended to ~~p~~ produce position vector $P(u)$, which describes the path of an approximating Bezier polynomial function between p_0 and p_n.

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

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

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

Eqn ~~B~~ $P(u)$ represents a set of three parametric equations for the individual curve coordinates.

$$x(u) = \sum_{k=0}^{n} x_k \, BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \, BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k \, BEZ_{k,n}(u)$$

In most cases, a Bezier curve is a polynomial of degree that is one less than the designated number of control points

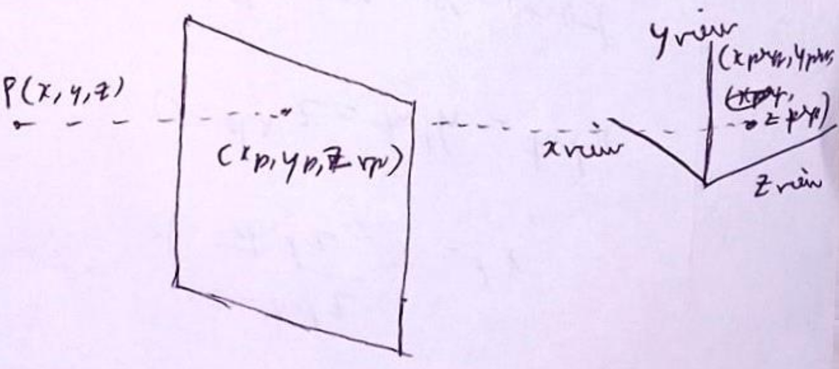
Three points generate a parabola, four points a cubic curve and so forth.

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

A. 1. If projection reference point is on z view, means $x_{prp} = y_{prp} = 0$

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

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



2. The projection reference point is fixed at the coordinate origin, and

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

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

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

3. If the view plane is the uv plane and there are no restrictions on the placement of the projection reference point, then we have

$z_{vp} = 0$

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

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

4. With uv plane as the view plane and the projection reference point on the z view axis, the perspective equations are

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

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

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

6. Explain OpenGL visibility detection functions

A: glEnable(GL_CULL_FACE)

It is used for turning culling on

glCullFace(mode)

It specifies what to cull

mode = GL_FRONT or GL_BACK

GL_BACK is default

glFrontFace(vertexOrder)

It is for order of vertices.

Orientation is changed.

vertexOrder = GL_CW or GL_CCW.

GL_CW is for clockwise direction (front)

GL_CCW is for counterclockwise direction (back)

GL_CCW is default

Create depth buffer by setting GLUT_DEPTH flag in glutInitDisplayMode() or the appropriate flag in the PIXELFORMATDESCRIPTOR.

Enable per-pixel depth ~~with~~ testing with glEnable(GL_DEPTH_TEST)

Clear depth buffer by setting GL_DEPTH_BUFFER_BIT in glClear().

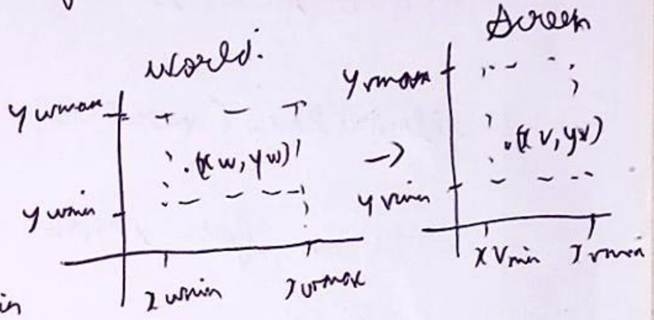
glDepthFunc(condition);

changes the test used

condition : GL_LESS [closer: visible (default)]

GL_GREATER [farther: visible]

9. Explain normalization transformation for an orthogonal projection



Relative position is same

A.
$$\frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}$$

$$x_v - x_{vmin} = (x_{vmax} - x_{vmin})\left(\frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}\right)$$

$$x_v - x_{vmin} = (x_w - x_{wmin})\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right)$$

$$x_v = x_w\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) + x_{vmin} + \frac{x_{wmin}x_{vmin} - x_{wmin}x_{vmax}}{x_{wmax} - x_{wmin}}$$

$$x_v = x_w\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) + \left(\frac{x_{wmax}x_{vmin} - x_{wmin}x_{vmax}}{x_{wmax} - x_{wmin}}\right)$$

$$x_v = x_w s_x + t_x$$

where $s_x = \dfrac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$

$$t_x = \frac{x_{wmax}\, x_{vmin} - x_{wmin}\, x_{vmax}}{x_{wmax} - x_{wmin}}$$

Similarly,

$$y_v = y_w S_y + t_y$$

where $S_y = \dfrac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$

$$t_y = \frac{y_{wmax}\, y_{vmin} - y_{wmin}\, y_{vmax}}{y_{wmax} - y_{wmin}}$$

$$\left.\begin{aligned} x_v &= S_x x_w + t_x \\ y_v &= S_y y_w + t_y \end{aligned}\right\} \text{can be written as}$$

$$M_{window,normview} = T \cdot S = \begin{bmatrix} S_x & 0 & t_x \\ 0 & S_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

For normalized coordinates,

-1 ~~0~~ for x_{vmin} and y_{vmin}

~~0~~ 1 for x_{vmax} and y_{vmax}

$$M_{window,normsquare} = \begin{bmatrix} \dfrac{2}{x_{wmax} - x_{wmin}} & 0 & -\dfrac{x_{wmax} + x_{wmin}}{x_{wmax} - x_{wmin}} \\ 0 & \dfrac{2}{y_{wmax} - y_{wmin}} & -\dfrac{y_{wmax} + y_{wmin}}{y_{wmax} - y_{wmin}} \\ 0 & 0 & 1 \end{bmatrix}$$

Similarly for 3D,

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

5. Demonstrate openGL functions for displaying window management using GLUT.

A. glutInit(&argc, argv)

It is used to initialize GLUT library.

glutInitWindowPosition(xTopLeft, yTopLeft);

Position of ~~a~~ display window on screen.

glutInitWindowSize (dwWidth, dwHeight);

Size of window

dwWidth is width of display

dwHeight is height of display

glutCreateWindow ("~~Tit~~ ~~String~~ String");

It is used to create display window with name.

glutDisplayFunc ();

It sets the ~~initial~~ display ~~mode~~ callback for current window

glutInitDisplayMode();

It sets the initial display mode.

glutReshapeFunc();

Its sets the reshape callback for current window

glutSetCursor ();

It changes the cursor image of current window.

10 Explain cohen-sutherland line clipping algorithm

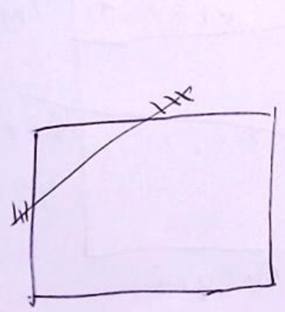
A. · There will be a rectangular window (clipping window)

· There will be an object (exi: line

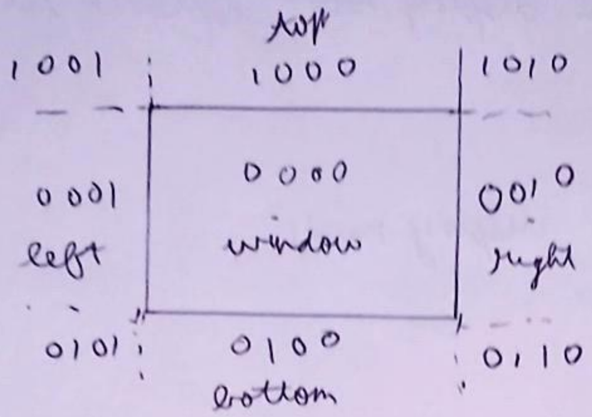
· ~~Pixels~~ Only pixels inside the rectangle ~~are~~ must be shown.

· Pixels outside the rectangle should not be shown (clipped).

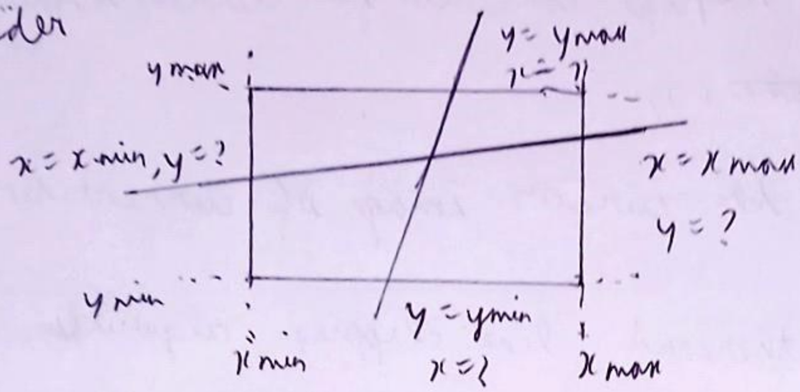
Example



Boundaries



Consider



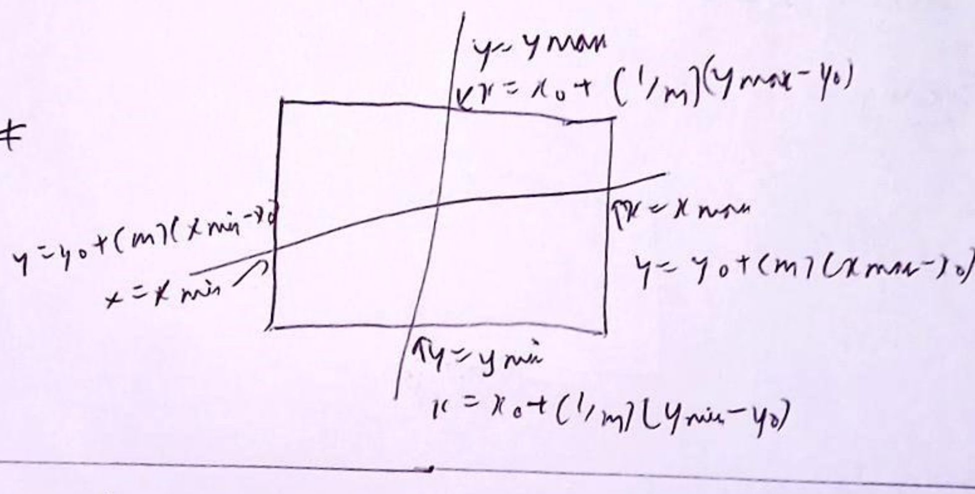
$$m = (y - y_0)/(x - x_0)$$

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

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

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

$\therefore$ ~~$x = x_0 +$~~

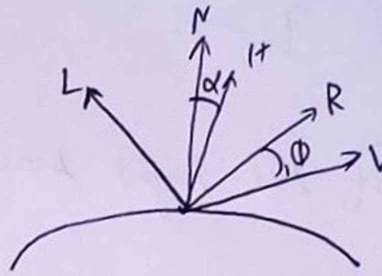


2. Build Phong Lighting model with equations

A: A shiny surface has a ~~noo~~ narrow specular reflection range

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

n_s = shininess



$I_{l,\,specular} = W\theta\ I_l \cos^{n_s}\phi$

I = intensity

$0 \le W\ \theta \le 1$ is called 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

For most opaque materials specular-reflection coefficient is nearly content K_s.

$$I_{l,\,specular} = \begin{cases} K_s I_l\ V \cdot R^{n_s}, & V \cdot R > 0 \text{ and } N \cdot L > 0 \\ 0.0, & \text{otherwise} \end{cases}$$

R can be calculated from L and N. $R = 2N \cdot L\ N - L$.

The normal N may vary at each point; To avoid N computations, angle ϕ is replaced by an angle α defined by a halfway vector H between L and V.

Efficient computations

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

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

H is the direction yielding maximum specular reflection in viewing direction V if the surface normal N would coincide with H.

If V is coplanar with R and L (and hence with N too) $\alpha = \phi/2$.