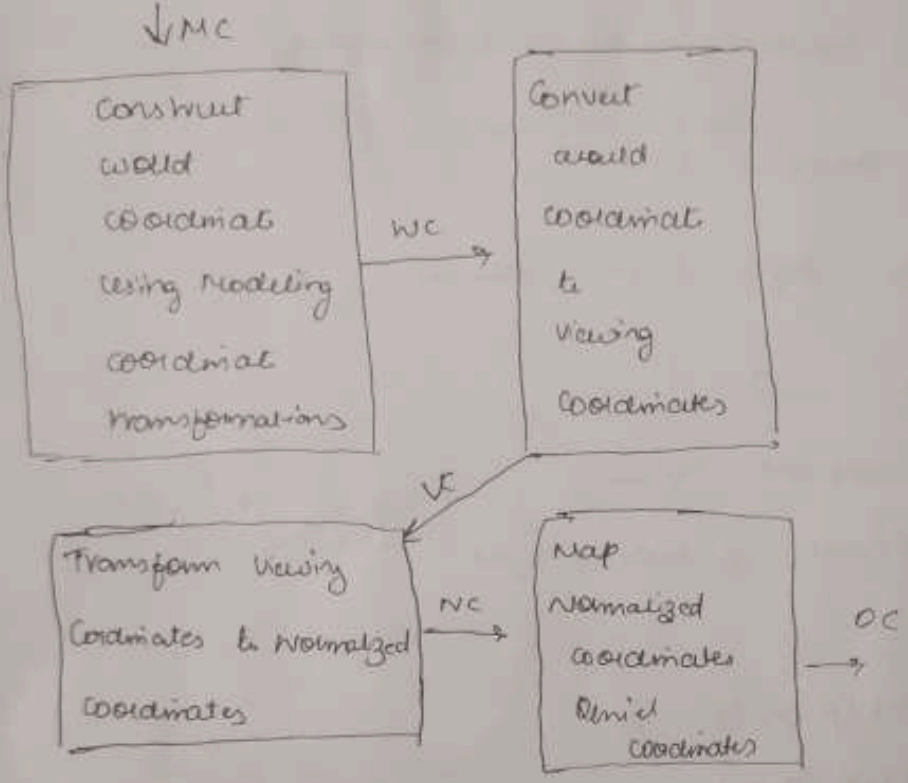
CGV Assignment

M. Mohit Krishna
IBY20CS105

① 2D Pipeline



→ We could set up a upto 2-d, viewing coordinate reference frame for specifying clipping window.

→ Systems use normalized coordinates in the range from 0 to 1, others used a normalized range from -1 to 1

→ Clipping is usually performed in the normalized coordinates.

(2) Build phong lighting Model with equations height consists of 3 different types of light

⇒ Ambient lighting refered as the natural lighting.

→ Diffusion - The artificial light

→ Specular lighting - Refers to the shininess of the object

$I_{amb} = k_a I_a$ — (1)

k_a : ambient reflexivity.

I_a : intensity of ambient light.

Similarly

$I_{diff} = k_d I_p \cos(\theta)$ — (2)

$= k_d I_p (N \cdot L)$

$I_{spec} = k_b I_l \cos^n \phi$

∴ The Phong Model gives the equation of the all combined

Total intensity $I = k_a I_a + k_d I_p \cos\theta + k_s I_l \cos^n \phi$

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

→ The Matrix Representations of

translation, Rotation & scaling

are

$P' = P + T$

↓

Translation $P' = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} +x \\ +y \end{bmatrix}$

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

Scaling $P' = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Generic eq $= S \cdot P$

$P' = M_1 * P + M_2$

But $x = \frac{xh}{h}$, $y = \frac{yh}{h}$

$h = 1$

∴ consider $(h*x, h*y, h)$

translation $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & +x \\ 0 & 1 & +y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Rotation $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

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

(2) (4) Difference between Raster and Random dcon displays.

Raster dcon	Random dcon
1) Produces jagged lines that are plotted as a discrete point sets.	1) Random system produces smooth lines drawing
2) Less expensive	2) More expensive
3) Modification difficult	3) Modification easy
4) Resolution low	4) Resolution high.
5) Solid pattern is easy to fill.	5) Solid pattern is difficult to fill.

(5) Demonstrate OpenGL functions for window Management using GLUT.

- glut Create window - used to create a new window.
- glut Createsub window - used to create another window within done window.
- glut Set window - used to set an particular id for the window.

glut Get window - used to get the window ID.

- glut Destroy window - To delete the window that was created.
- glut Post Redisplay - To display the window again & again, continuously until forcibly closed.
- glut Reshape window - used for transformation of world coordinate view coordinates and displaying it.
- glut Full screen - To represent window in full screen mode.
- glut pop window / glut use window - works just like it is a matrix on window.
- glut Hide window - To hide the window from being displayed on screen.
- glut pre display Fare - To display.
- glut Main loop
- init()
- OpenGL visibility Detection Functions

(A) OpenGL polygon culling functions.

Remove back face, front face of an object.

glCullFace(mode);

glEnable(GL_CULL_FACE);

glDisable(GL_CULL_FACE);

(B) Depth-Buffer function.

glutInitDisplayMode(GLUT_SINGLE |

GLUT_RGB | GLUT_DEPTH);

glClear(GL_DEPTH_BUFFER_BIT)

This works as initialization function for depth buffer and refresh buffer.

glDepthRange(nearNormDepth, farNormDepth)

glClear(GL_DEPTH_BUFFER_BIT)

glClearDepth(maxDepth)

glEnable(GL_DEPTH_TEST)

glDisable(GL_DEPTH_TEST)

(C) OpenGL wireframe surface visibility methods

glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)

Visible & hidden edges display.

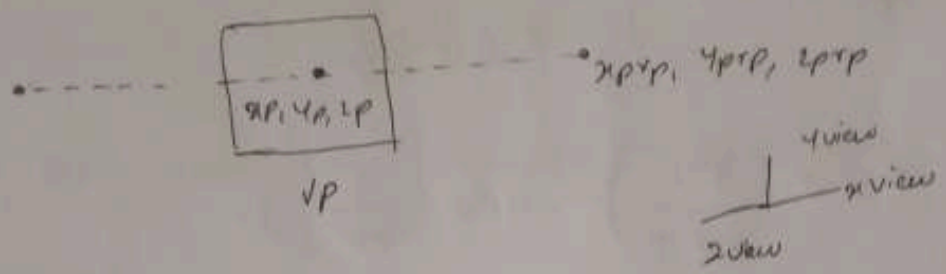
Open GL Depth Cuing Ins.

glFogi (GL_FOG_MODE, GL_LINEAR)

glEnable (GL_FOG)

- To increase & decrease the brightness.

three special cases discussed with perspective projection.



Consider

$$x' = x - (x - x_{PRP})u \qquad u = \frac{z_{VP} - z}{z_{PRP} - z}$$

$$y' = y - (y - y_{PRP})u$$

$$z' = z - (z - z_{PRP})u$$

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

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

special cases:

① $x_{PRP}, y_{PRP} = 0$

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

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

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

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

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

3) $z_{vp} = 0$

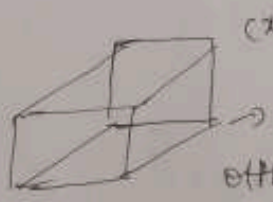
$$x_p = x\left(\frac{z_{prp}}{z_{prp}-z}\right) - z_{prp}\left(\frac{x}{z_{prp}-z}\right)$$

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

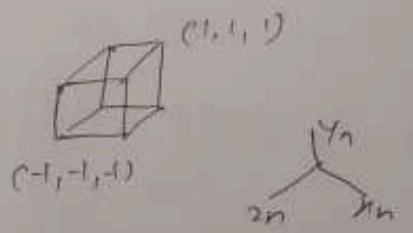
4) $x_{prp} = y_{prp} = z_{vp} = 0$

$$x_p = x\left(\frac{z_{prp}}{z_{prp}-z}\right), \quad y_1 = y\left(\frac{z_{prp}}{z_{prp}-z}\right)$$

(8) Explain normalization for an orthogonal projection



Orthogonal projection view volume



Normalized view volume.

- Consider a unit cube for this normalized view volume with each x, y, z co-ordinates normalized in the range 0 to 1

normalization transformation approach is to use symmetric cube with coordinates -1 to 1.

∴ we get the normalization transformation for the orthogonal view volume.

$M_{ortho,norm}$ =

$$\begin{bmatrix} \frac{2}{x_{max}-x_{min}} & 0 & 0 & -\frac{x_{max}+x_{min}}{x_{max}-x_{min}} \\ 0 & \frac{2}{y_{max}-y_{min}} & 0 & -\frac{y_{max}+y_{min}}{y_{max}-y_{min}} \\ 0 & 0 & \frac{-2}{z_{near}-z_{far}} & -\frac{z_{near}+z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(9) Explain Bezier curve and its properties with equation.

Bezier curves are parametric curves that are generated with the help of control points. It is widely used in graphics and other related industry.

- they are named after the french engineer pierre Bezier who discovered it.

$$\sum_{i=0}^{n} P_i B_i^n(t)$$

n - polynomial degree
t = variable
i = index

$B_i^n(t)$ represents Bernstein polynomial

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-t} t^i$$

they are be of 2 - control points - linear curve

3 - control points - cubic curve.

4 control points - quadratic curve.

we used the above mentioned formed as Bezier curve

$= (n_{(r)} * (1-t)^{n-t} t^i$ ✱ for every point.

n - control points number - 1

t = 0 - 1 (Range)

Sutherland line clipping algorithm.

Then - Sutherland algorithm works on Region code

- Region code is 4- bit code.

(A B R L)

(T B R L) - Top, Bottom, Right, Left

1001	1000	1010
0001	0000	0010
0101	0100	0110

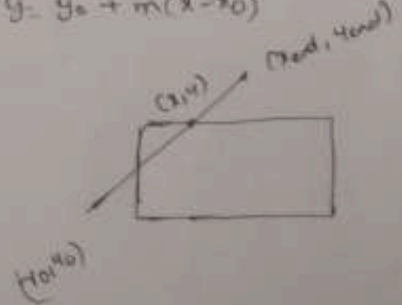
for a line - (x_0, y_0) & (x_{end}, y_{end})

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

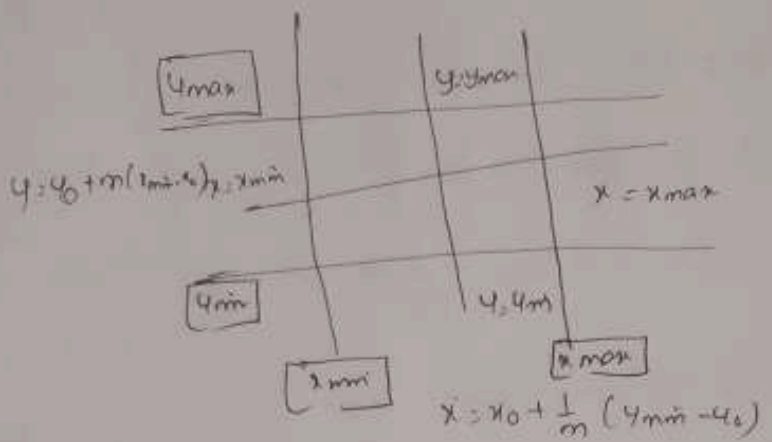
$$m(x - x_0) = (y - y_0)$$

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

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



$x = x_0 + y_m(y_{max} - y_0)$ --



$y = y_0 + m(y_{max} - y_0)$

Thus, the above formulas to be applied when a particular needs to be clipped.