CGV Assignment

Name: Nishanth Y
USN: 1BY20CS126

① Build a 2D viewing transformation pipelines & also explain openGL 2D Viewing function

MC → Construct world co-ordinate scope using Modeling →(WC) Convert word co-ordinate for viewing co-ordinate →(VC) Transform viewing co-ordinates to Normalize →(NC) Map normalized co-ordinates do Device Coordinates →(DC)

A section of 2D scene that is slected for display is called a clipping window because all parts of scene outside the selected section are "clipped off"
The mapping of a 2D world-co-ordinate (WC) description to device co-ordinates (DC) is called 2D viewing transformation
Once the world co-ordinate scene has been constructed we could set up a seperate 2D-viewing co-ordinates reference frame for specifying the clipping window.
Depending upon graphics library, the viewport is defined in normalized co-ordinates or screen co-ordinates
At the final step of viewing transformation the contents of viewport are transformed to partions within the display window.

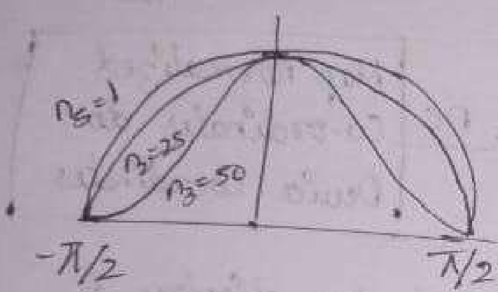
The openGL 2D viewing function in OpenGL projection Mode.

GLU clipping window function:

glViewPort (Xvmin, Yvmin, Vp width, Vpheight);

② Build phong lighting model with equations?

Phong reflection is an empirical mode of local illumination, It describe the way a surface reflects light as a combination of diffuse reflection of rough surface with the specular reflection of shiny reflection



Phong model sets the intensity of specular reflection to $\cos^n\phi$

I_L, specular $= w(\theta) I_l \cos^n \phi$.

$0 \leq w(\theta) \leq 1$ is called specular reflection co-efficient. If light direction & viewing direction, V are on same side of normal V, or if L is behind the surface, specular effects do not exist.

We have 3 function in GLUT for display window

glutInitWindowPosition (x Topleft, Y Topleft);

glutInitWindowSize (dwidth, dheight);

glutCreateWindow ("Title of Window");

③ Apply Homogenious Co-ordinates for translation, rotation and scaling via matrix representation

A standard technique for accomplish 2D or 3D Transformation is to expand each two-dimensional co-ordinate-position representation (x,y) to 3D (x_h, y_h, z_h)

where, $x = \frac{x_h}{h}$; $y = \frac{y_h}{h}$

Translation

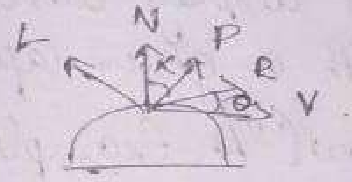
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \\ 1 \end{bmatrix} \quad P' = T(t_x, t_y).P$$

Scaling

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

Ration

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



I_c, specular = $\{ k_s I_l (V.R)^{n_s}, \; VR > 0 \; \& \; N.L > 0 \}$

$R = (2N \cdot L)N - L$. The normal N may vary at each point. To avoid N computations, angle ϕ is replaced by angle α defined by a halfway vector H between α & V

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

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

Random Scan	Raster Scan
• It has high resolution	• Its resolution is low.
• It is more expensive.	• It is less expensive.
• Easier to modify.	• Modification is tough.
• Solid pattern tough to fill	• Easy to fill solid pattern
• Refresh rate depends on resolution	• Does not depend on pictures

⑤ Demonstrate OpenGL function for displaying window management using GLUT.

A:- We perform the GLUT initialization with statement

glutInit (&argc, argv);

Next, we can state that display window is to be created on screen with a given caption for title.

glutCreateWindow ("An Example");

when the signal argument for this function can be any character string that we want to use.
The following must be last one in program, It puts the device in infinite loops the checks for inputs

glutMainLoop()

The function must be last one in program. It puts the device specific upperleft corner.

glutInitWindowPosition (50, 100);

⑥ Explain OpenGL visibility Detection function

a). OpenGL polygon Culling functions -

Back face removal with functions

glEnable (GL_CULL_FACE); glCullFace (mode);

Mode can be GL_Black, GL_FRONT, GL_FRONT_BACK

Disable with glDisable (GL_CULLFACE);

b) OpenGL Depth Buffer Function

To use OpenGL depth Buffer visibility detection functions. We need to modify GLUT initialization function.

glutInitDisplayMode (GLUT_SINGLE | GLUT_DEPTH);

glClear (GL_DEPTH_BUFFER_BIT);

c) OpenGL wireframe surface visibility method

A wireframe display can be obtained in OpenGL by representing that only its edges are generated

glutInitDisplayMode (GLUT_SINGLE | GLUT_DEPTH)

d) OpenGL Depth Cueing Function

It is used to vary the brightness of object as function of its distance from viewing

glEnable (GL_FOG);

applies to depth f^n d min = 0.0 and dmax = 1.0

and set different values for dmin

glFogf (GL_FOG_START, minDepth);

glFogf (GL_FOG_END, maxDepth);

7) Write special cases that we discuss to respective projection.

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

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

Cases

(i) projection reference point is limited along z-view axis

$x_{prp} = y_{prp} = 0$ $\quad x_p = x\left[\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right]$ $\quad y_p = y\left[\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right]$

(ii) when projection reference point is at co-ordinate

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

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

(iii) If view plane is uv plane and no restriction on placement of projection reference point.

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

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

(8) Explain Bezier curve equation along with equation along with properties

Developed by french engineer pure Bezier for use in design. It can be filled to any number of control points

Equation:- $P_k = (x_k, y_k, z_k)$ P_k = generate (n+1) control point position.

P_k = position vector that describe path

$P(u) = \sum_{k=0}^{n} P_k BEZ_{k,n}(u)$ $BEZ_{k,n}(u)$ (n, k)

is Bezier polynomial.

9) Explain Normalization transformation for Orthogonal projection.

We assume that orthogonal projection view volume to mapped into symmetric normalization cube within left-handed reference frame.

Also Z-co-ordinate position for handed reference frame.. This position ($X_{min}, Y_{min}, Z_{near}$) is mapped to (1, 1, 1)

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

10) Explain Cohen-Sutherland line clipping.

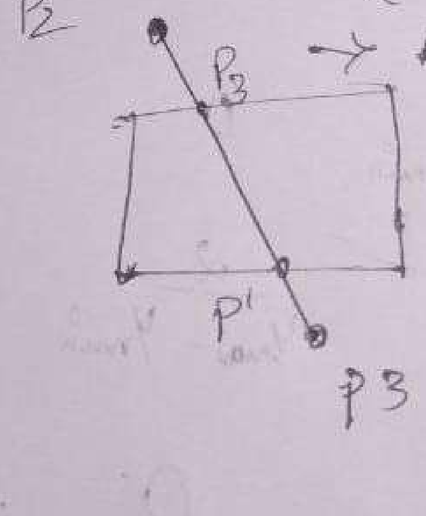
Every line end point in picture is designed with 4 digits binary code called region code & each bits is used to indicate where point lie

Once we established region code for all line end-point we determine where the completly inside or not.

Intersection P_2' & P_2^2 P_2^4 is clipped off for line P_0 to P_4 we find that point ~~to~~ is outside left boundary P_4 is inside therefore intersect P_3 & P_3' to P_3' clipped off.

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

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



→ Right clipping @