NISHANTHA I L S
1BY20CS127 CSE
BMSIT&M

# CGV ASSIGNMENT

2022-23.

Shankar R.

Questions:-

① Build 2D Viewing transformation pipeline and explain it. OpenGL 2D view functions.

② Build Phong lighting model with eqn.

③ Apply Homogenous co-ordination for translation, rotating, scaling via matrix representation.

④. Outline diff. blw raster scan display and random scan display.

⑤ Demonstrate openGL function for display window management using GLUT.

⑥- Explain OpenGL visibility detection functions.

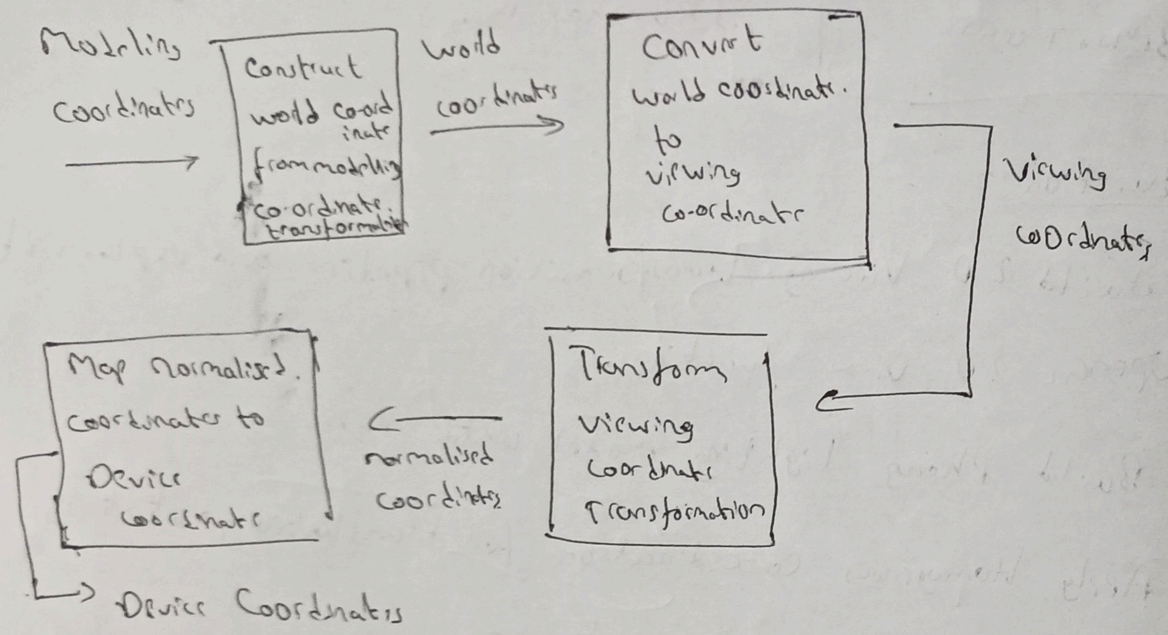
⑦. Write special cases that we discussed with r.t. perspective projection transformation co-ordinates.

⑧ Bezier curve equation along with properties.

⑨ Explain normalization transformation & orthogonal projection.

⑩. Explain Cohen-Sutherland Clipping Algorithm.

① 2D viewing transformation pipeline.




* First we apply model transformation on modeling coordinates to get world coordinates.
* Determine visible parts of world coordinates to get viewing coordinates.
* From that standard coordinates are applied to get normalized coordinates.
* Clip that Normalized Coordinate to get determined pixels we get Device coordinates.

This is the pipeline of 2D viewing.

OpenGL 2D view function.

① glMatrixMode(GL_PROJECTION);
② glLoadIdentity();
③ gluOrtho2D(xwmin, xwmax, ywmin, ywmax);
④ glViewport(xvmin, yvmin, vpwidth, vpheight);
⑤ glutInitWindowPosition(10,10)
⑥ glutInitWindowSize(500,500);
⑦ glutCreateWindow("My First");
⑧ glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
⑨ glutDisplayFunc(pictureDescrip);
⑩ glutPostRedisplay();

Q2) Phong lighting model with eqn:

Phong model clubs ambient, diffusion and specular reflection.

$I = I_a \times k_a$

$I_a$ - lighting source property. } ambient.

$k_a$ - Natural Property.

$I = I_p \times k_d \times \cos\alpha$

$I_p$ - intensity of point light source. } diffusion.

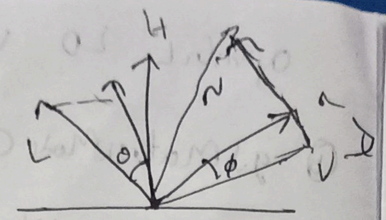
$k_d$ - diffuse reflection co-efficient.

$I = I_p \times k_y \times \cos^n \alpha$

$I_p$ - intensity of point light source.

$k$ - specular reflection coefficient.

$n$ - shininess




but. I can also be written as $I = I_p \times k_y \times \cos\theta$. $N.L$

Diffusion eqn

specular eqn can also be written as $I = I_p \times k_s \times \cos^n \alpha$.

$= I_p \times k_s \times (R.V)^n$

Ambient - Combination of light reflection from various surfaces to produce uniform illumination.

Diffuse - uniform light scattering on surface. proportional to amount of light. surface normal and light vector.

Specular - light which gets reflected. light ray, viewing angle, surface normal.

Q3) Homogenous co-ordinates for transformation via matrix representation. also for rotation and scaling.

translation $P' = P + T$ : $\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$.

rotation $P' = RP$ : $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$.

scaling $P' = S.P$ : $\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$.

also can be written as.

$P'_x = 1.P + T$

$P'_y = R.P + 0$ } - $P' = M_1 * P + M_2$.

$P' = S.P + 0$

$x = x_h / h$

$y = y_h / h$.

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

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

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

| Q(b) Raster screen display | Random screen display |
|---|---|
| * Resolution is lower than random. | resolution of random scan high. |
| * while cost of raster scan is less. | Costlier than raster scan. |
| * alteration is not easy. | alteration is easy. |
| * interlacing is used. | interlacing is not used. |
| * raster scan uses pixels. suitable realistic. | mathematical function for rendering polygon drawing. |
| * Picture definition as frame buffer. | Picture definition as refresh buffer. |
| * refreshrate 60 to 80 frame. | 30 to 60 frame |

Visibility Detection.

Q(5). OpenGL function for display window management using GLUT.

i) glEnable(GL_CULL_FACE) - Turn culling on.

ii) glCullFace(mode); - Specify what to cull

iii) mode = GL_FRONT or GL_BACK - GL_BACK is default.

iv) glFrontFace(vertexOrder); - order of vertices

v) vertexOrder = GL_CW or GL_CCW - clockwise, counter.

vi) glut_depth - depth buffer.

vii) glutInitDisplayMode() - appropriate flag in pixelformat descriptor.

viii) glEnable(GL_DEPTH_TEST) - Enable per pixel depth tests

ix) glDepthFunc(condition); - Change test used.

condition: GL_LESS GREATER - closer, farther

Q

(6). Display window management using GLUT.

i) glutInit(int *argc, char **argv) - initialize command line arguments.

ii) glutInitDisplayMode(unsigned int mode) - set display mode for window.

iii) glutInitWindowSize(int width, int height) - set initial size of window.

iv) glutInitWindowPosition(int x, int y) - sets initial position

v) glutCreateWindow(const char *title) - create top-level window title.

vi) glutDisplayFunc(void(*func)(void)) - set display callback function.

vii) glutMainLoop() - Enter GLUT event processing loop.

Q7) Special case w.r.t. perspective project co-ordinates.

① If projection reference point is on z_view means $x_{prp} = y_{prp} = 0$.

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

② projection reference point is fixed at coordinate origin.

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

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

③ If view plane is the uv plane, no restriction on placement of projection reference point. $z_{vp} = 0$.

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

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

④ With uv plane as view plane and projection reference point on z_view also perspective eqn. are: $x_{prp} = y_{prp} = z_{vp} = 0$

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

Reason:-

$x' = x - (x - x_{prp})u.$

$y' = y - (y - y_{prp})u.$

$z' = z - (z - z_{prp})u.$

$0 \le u \le 1$

viewplane $z' = z_{vp}$.

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

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

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

Q8) Bezier curve eqn along with properties

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

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

Bezier blending functions $BEZ_{k,n}(u)$ - Bernstein Polynomial.

$P(u)$ - coordinate points. from 0-n Polynomial function $P_0 - P_n$

General case of n+1 control point position.

$P_k = (x_k, y_k, z_k)$. k varying from 0 to n.

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

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

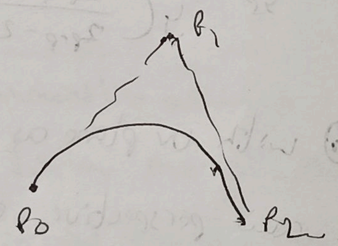
$$y(u) = \sum_{k=0}^{n} y_k BEZ_{k,n}(u).$$

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

Recursive calculation.

$$C(n,k) = \frac{n-k+1}{k} C(n,k-1) \text{ for } n \ge k$$

$$BEZ_{k,n}(u) = (1-u) BEZ_{k,n-1}(u) + u BEZ_{k-1,n-1}(u).$$




Properties

1. Control points.
2. Degree.
3. Interpolation.
4. Tangents
5. Convex Hull Property
6. Parametric representation
7. Affine invariance.
8. Subdivision.
9. Higher degree curves.

Q(a):- normalising transformation of orthogonal projection mapping coordinates into a normalised view volume with coordinates in the range -1 to 1.

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

reason:- $xv = s_x x_w + t_x$ where scaling factor.
$yv = s_y y_w + t_y$.

$$s_x = \frac{x_{vmax}-x_{vmin}}{x_{wmax}-x_{wmin}} \qquad s_y = \frac{y_{vmax}-y_{vmin}}{y_{wmax}-y_{wmin}}.$$

Transformation factor's

$$t_x = \frac{x_{wmax}x_{vmin} - x_{wmin}x_{vmax}}{x_{wmax}-x_{wmin}} \qquad t_y = \frac{y_{wmax}y_{vmin} - y_{wmin}y_{vmax}}{y_{wmax}-y_{wmin}}.$$

to normalise co-ordinates substitute -1 for $x_{wmin}$ & $y_{wmin}$.

1 for $x_{wmax}$ & $y_{wmax}$.

finally we get $M_{window,normalised} = \begin{bmatrix} \frac{2}{x_{wmax}-x_{wmin}} & 0 & -\frac{x_{wmax}+x_{wmin}}{x_{wmax}-x_{wmin}} \\ 0 & \frac{2}{y_{wmax}-y_{wmin}} & -\frac{y_{wmax}+y_{wmin}}{y_{wmax}-y_{wmin}} \\ 0 & 0 & 1 \end{bmatrix}$

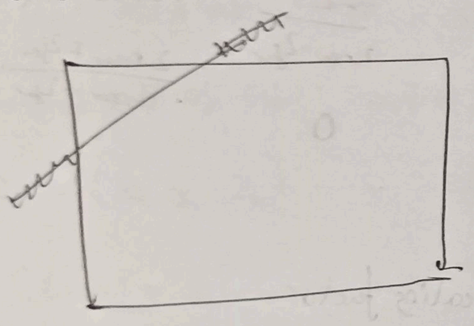
similar for 3D.

Q10) Cohen Sutherland Algorithm (Clipping window)

There will be rectangular window (Clipping window)

There will be object.

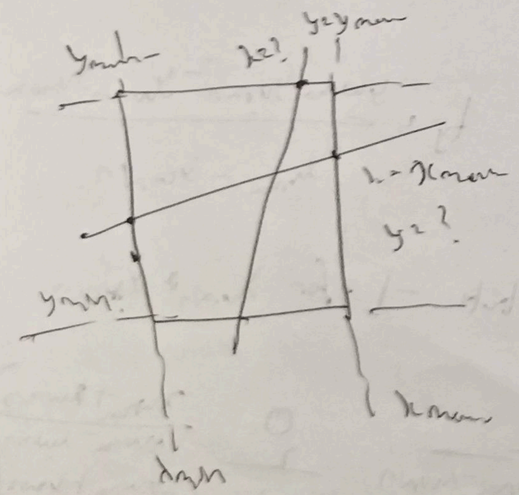
only pixel inside rectangle must be shown, the pixels outside should be clipped.



Boundaries in top

| 1001 | 1000 | 1010 |
|---|---|---|
| 0001 left. | 0000 window | 0010 right. |
| 0101 | 0100 bottom. | 0110 |

Consider.




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

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

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

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

2)