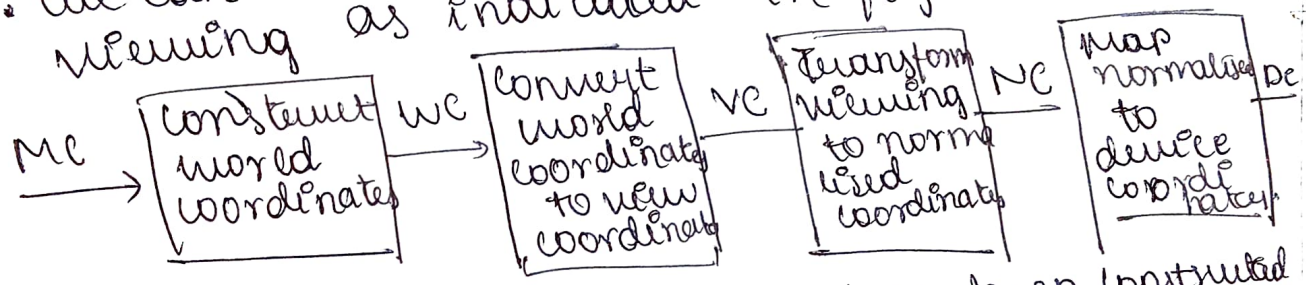
CG Assignment

Name: KEERTHANA.M
USN: 1BY20CS083

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

- The mapping of a two dimensional world coordinate scene description to device coordinate is called a two dimensional viewing transformation.
- This transformation is simply referred to as the window to viewport transformation or the windowing transformation.
- we can describe steps for 2 dimension viewing as indicated in fig.



- Once world coordinate scene has been constructed we could set up separate 2D viewing coordinate reference frame for clipping window
- To make viewing process independent of the requirements of any output devices, graphics system convert object description to normalised coordinates apply.
- At the final step of the viewing transformath contents of the viewport are transferred to positions within the display windows.
- clipping is usually performed in normalised coordinates.
- allows us to reduce computations by first contatenating various transformation matrices.

2D viewing functions

- function: glMatrixMode (GL_PROJECTION)
  set the initialisation as
  glLoadIdentity();
- to define two dimensional clipping window we can use GLUT func
  gluOrtho2D(xwmin, xwmax, ywmin, ywmax)
- we specify viewport parameters with OpenGL functions glViewport (xvmin, xvmax, xpwidth, xpheight);
  glutInit(&argc, argv);
- display-window parameters are selected with GLUT functions
  glutInitDisplayMode(mode);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glClearColor(red, green, blue, alpha);
  glClearIndex(index);

2. Build phong lighting model with equations

- A local illumination model that can be computed rapidly there are 3 components
  - Ambient,
  - Diffuse
  - Specular

Ambient lighting -> this produces a uniform ambient lighting that is same for the all objects and its approximates global diffuse reflections from the various illuminated surface. the component approximate the indirect lighting by a constant

$I = I_a k_a$.

where I_a: ambient light intensity (color)
k_a: ambient reflection coefficient

Diffuse reflection: Incident light on the surface is scattered with equal intensity in all

$\frac{dI}{dt}$

Outline differences between raster scan displays and random scan displays

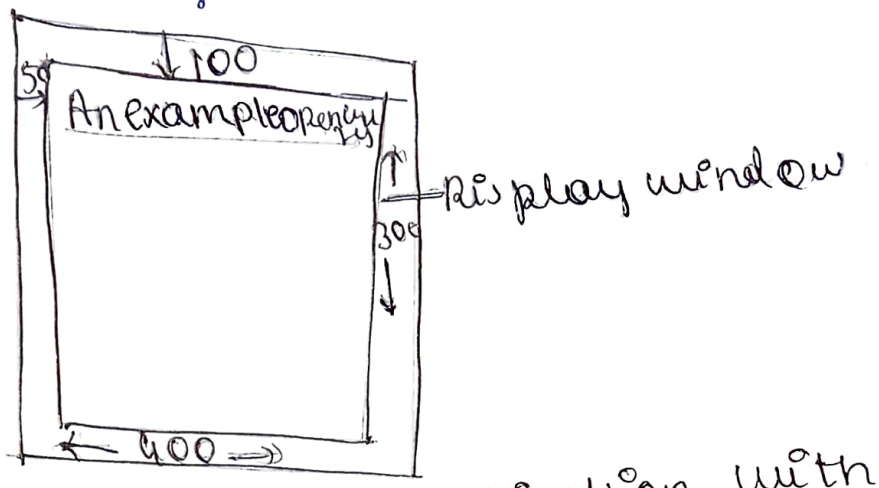
Raster scan displays

- Electron beam is swept across the screen one row at a time from top to bottom.
- As it moves across each row, beam intensity is turned on and off to create pattern of illuminated spots.
- Refreshing rate called frame rate normally 60 to 80 frames per second, describing as 60Hz to 80Hz.
- Picture definition is stored in a memory area called frame buffer. Stores intensity values for all screen points. Each screen point called pixel.

Random scan displays.

- When operated as a random scan display unit, CRT has electron beam directed only to those points of screen color a pictures to be displayed.
- Pictures are generated as line drawings with electron beam tracing out component lines. random scans monitors are also referred to as vector displays.
- A pen plotter operates in a similar way and it is an example of random scan, hard copy device.
- Refresh rate on a random scan depends on the number of lines to be displayed on that system.

5. Demonstrate OpenGL functions for display window management using GLUT.



- We perform GLUT initialisation with the Statement glutInit(&argc,argv);
- We can state that a display window is to be created on screen with a given caption for title bar.
  glutCreateWindow("An example OpenGL program");
  Single argument can be any character string
  → glutDisplayFunc(lineSegment);
glutMainLoop();
this func. must be last one in our prog.
it displays initial graphics and puts into infinite loop checks for input from devices such as mouse or keyboard.
glutInitWindowPosition(50,100);
following statement specifies that upper left corner display window should be placed 50 pixel to right of left edge of screen.
glutInitWindowSize(400,300)
glutInitWindowSize function is used to set initial pixel width and height of the display window.

5. Explain OpenGL visibility detection functions?

a) OpenGL Polygon culling functions
Back face removal is accomplished with functions

glEnable (GL-CULL-FACE)
glCullFace(mode);

where parameter mode is assigned the values GL-BACK, GL-FRONT, GL-BACK, GL-FRONT-AND-BACK.

By default parameter mode in glCullFace funcn has value GL-BACK.

culling routine is turned off with glDisable (GL-CULL-FACE).

b) OpenGL Depth buffer functions:
need to modify GLUT utility toolkit (GLUT) Initialisation function for the display mode to include a request for depth buffer, as well for refresh buffer.

glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH)

Depth buffer values can be initialised with glClear(GL-DEPTH-BUFFER-BIT).

glEnable (GL-DEPTH-TEST);

c) OpenGL uses frame surface visibility it displays of standard graphics object can be obtained in OpenGL.

glPolygonMode(GL-FRONT-BACK, GL-LINE)

d) OpenGL-DEPTH-cueing funcn.
brightness of object funcn of its distance

glEnable(GL-FOG);
glFogi(GL-FOG-MODE, GL-LINEAR);
glFogf(GL-FOG-START, minDepth);
glFogf(GL-FOG-END, maxDepth);

1. write special cases that we discussed wrt projection transformation.

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

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

Special cases

1. $z_{prp} = y_{prp} = 0$

$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right)$ $y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right)$ → ①

along z view axis

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

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

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

when projection reference point is fixed at coordinate origin.

3. $z_{vp} = 0$

$x_p = x\left(\frac{z_{prp}}{z_{prp} - z}\right) - x_{prp}\left(\frac{z}{z_{prp} - z}\right)$ → 3 (a)

$y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right) - y_{prp}\left(\frac{z}{z_{prp} - z}\right)$ → 3 (b).

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

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

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

with uv plane as the view plane & projection reference point or the z view axis.

8. Explain Bezier curve equation along with its properties.

- Developed by french engineer Bezier for use in design of Renault automobile design.
- Bezier curve seen can be fitted to many number of control points.

$P_k = (x_k, y_k, z_k)$ P_k = general $(n+1)$ control points positions.

P_k = positions vector which describes path of approx Bezier polynomial func P_0 and P_n.

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

$BEZ_{k,n}(u) = C(n,k) u^k (1-u)^{n-k}$ is Bernstein polynomial $C(n,k) = \frac{n!}{k!(n-k)!}$

Properties.

- Basic functions are real.
- Degree of polynomial defining curve one less than no of defining points.
- curve generally follows shape of defining polygon.
- curve connects first and last control points thus $P(0) = P_0$ $P(1) = P_n$.
- curve lies within the convex hull of the control points.

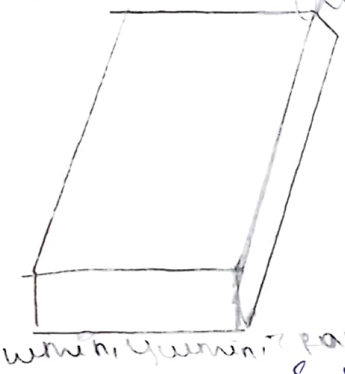
9. explain normalisation transformation for orthogonal projection.

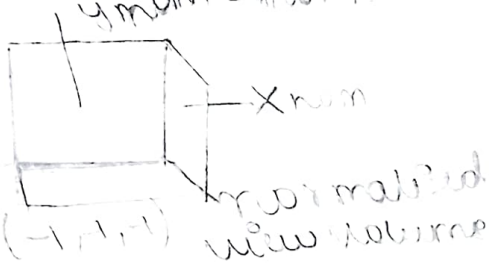
Normalisation transformation we assume orthogonal projection view volume is to be mapped into symmetric normalisation cube within a left handed reference frame.

this posn (xmin, ymin, znear) is mapped to normalization transformation for orthogonal view volume is

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

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





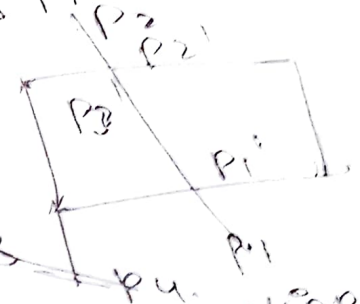
10. Explain cohen sutherland line clipping algorithm.

1001	1000	1010
0001	0000	0010
0101	0100	0110

every line end point in a picture is assigned four digit binary value called region code and each bit position is used to indicate whether point is inside or outside of one of clipping window boundaries. quickly determine which are completely within clip window & which are completely outside.

when the or operation between endpoint region code for a line segment is false (0000) this line is inside clipping window.

Lines that cannot be identified as being completely inside outside clipping window based on region codes for a line is true the line is completely outside clipping window lines cannot be identified as being completely inside or outside a clipping window by region code tests are next checked for intersection with boundary lines. Region codes says P_1 is inside and P_2 is outside



By checking region codes of P_3 & P_4 we find the remainder of line is below clipping window & can be eliminated.

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

where x is either x wmin or x wmax

slope is

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

∴ for intersection with horizontal border then x coordinate is

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