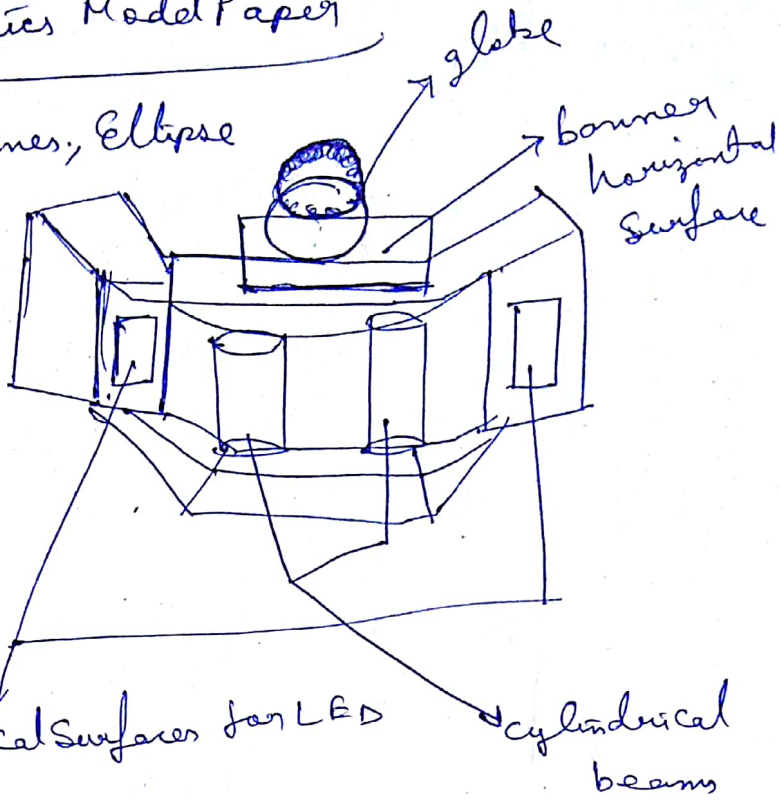
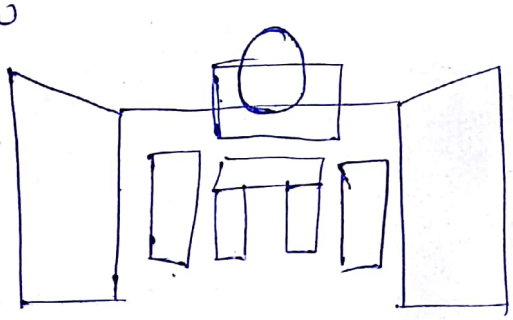
Computer Graphics Model Paper

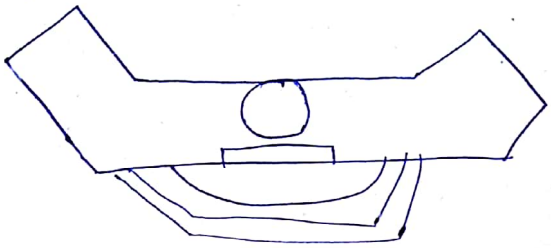
1) a) Circles, Lines, Ellipse



b) front-view



Top view



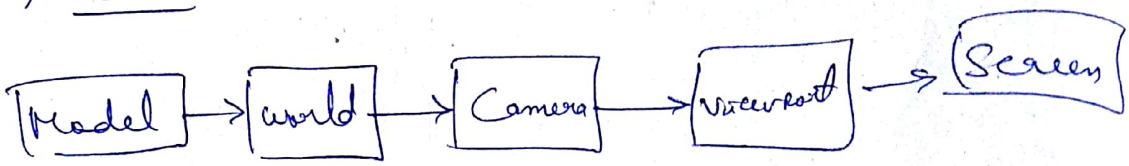
c) Model → First of all we will create a sphere and we need rectangle surfaces. We need circle for globe and lines for rectangles and ellipse and lines for beams

2) World: Then we place our model into the world and adjust it position accordingly.

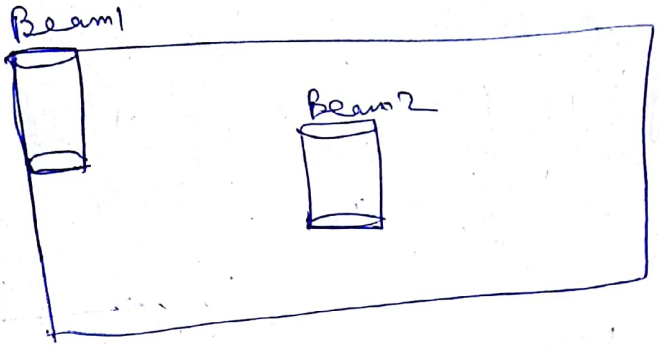
3) Camera: we place the camera to get the projection

4) Viewport: According to the position of the camera we get the viewport

5) Screen: we flush everything to the screen



d) i)



a) we use translation

ii) Since matrix is at origin we don't need any transformation for Beam1 but for beam 2 we use translation

$P' = (x', y')$ → new co-ordinates

$P = (x, y)$ → original co-ordinates

$T = (t_x, t_y)$ → translation matrix

$$P' = P + T$$

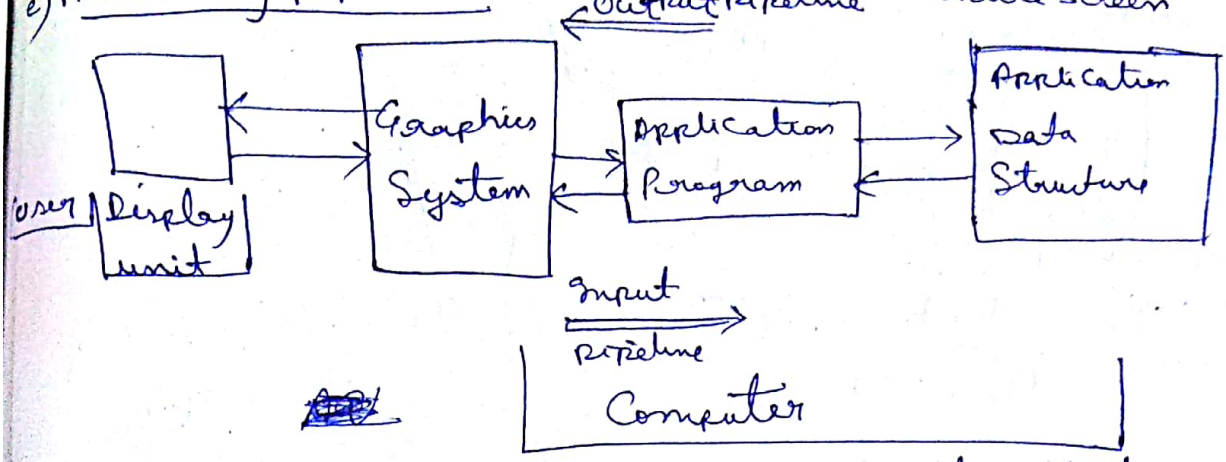
$$\begin{bmatrix} x' \\ y' \\ \cdot \end{bmatrix} = \begin{bmatrix} x \\ y \\ \cdot \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

in homogenous co-ordinates

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

iv) Since the frame is 600×600 the aspect ratio should be 1:1 So that we can use entire screen

e) Rendering pipeline



After the rendering pipeline, the image type that we get is vector image. Its composed of points and paths. and uses mathematical relationships between points and paths, connecting them to describe image.

The final output will be the building with all the requirements.

f) The factors which affect the image are.—

1) Field of View (FoV)
2) Depth of Field (DoF)
3) Exposure
4) Aspect Ratio

g) We will handle the exposure of light by using the technique of ray tracing. It is the rendering technique for generating an image by tracing the path of light as pixels in an image plane and simulating the effect of its encounters with vertical objects.

h) By using pushMatrix() and popMatrix() we can do this task. PushMatrix() saves the current co-ordinate system whereas popMatrix()

restores the co-ordinate system.

ii) M_{model} performs appropriate co-ordinate change

$$\begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix} = M_{model} \times \begin{pmatrix} x_{object} \\ y_{object} \\ z_{object} \\ 1 \end{pmatrix}$$

The next step is to transform the vertices from the world space to the eye or camera space

$$\begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix} = M_{modelview} \begin{pmatrix} x_{obj} \\ y_{obj} \\ z_{obj} \\ w_{obj} \end{pmatrix}$$

$$= M_{model} \times M_{view} \begin{pmatrix} x_{obj} \\ y_{obj} \\ z_{obj} \\ w_{obj} \end{pmatrix}$$

To project the object on the screen

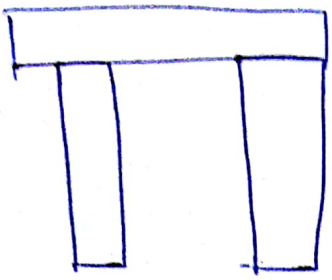
$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = M_{projection} \times \begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix}$$

---

d) No, the graphics system does not allow us to do so because currently the image is in vector form. It needs to rasterized before we can do any operation on the image because color correction and all can be applied only on pixels and not on lines or points.

The real time example is when we use photoshop and import some image and before we can apply any transformation on it we need to rasterize the layer.

k)



Clipping is the act of cutting out a portion of an object.

l) If we apply culling to beam1, nothing will happen to beam2 but beam1 will be completely excluded.

m) (R, G, B) stands for Red - Blue - Green. It is a color-coding method. It ranges from 0 to 255

i) (0, 0, 0) Black color will be filled in the two beams

ii) (255, 255, 255) white color will be filled in the two beams.