Assignment - 3

Illustrate Cohen-Sutherland line clipping algorithm.

Initially, every line endpoint in a picture is assigned a four-digit binary value, called a region code, and each bit position is used to indicate whether the point is inside or outside one of the clipping-window boundaries.

bit 4	bit 3	bit 2	bit 1
Top	Bottom	Right	Left.

There are 3 possibilities for the line :-

→ Line can be completely inside the window (this line should be accepted).

→ Line can be completely outside of the window (this line will be completely removed from the region).

→ Line can be partially inside the window.

Algorithm :-

S1 : Assign a region code for each end points

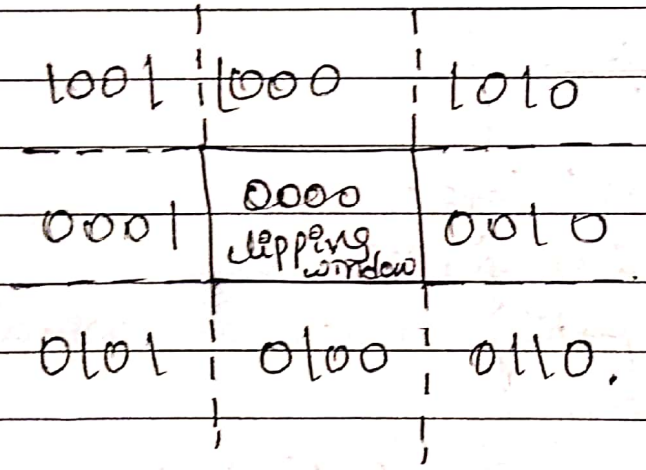
S2 : If both endpoints have a region code 0000 then accept this line.

S3 : Else, perform the logical AND operation for both region code.

S3.1 :- If the result is not 0000, then reject the line.

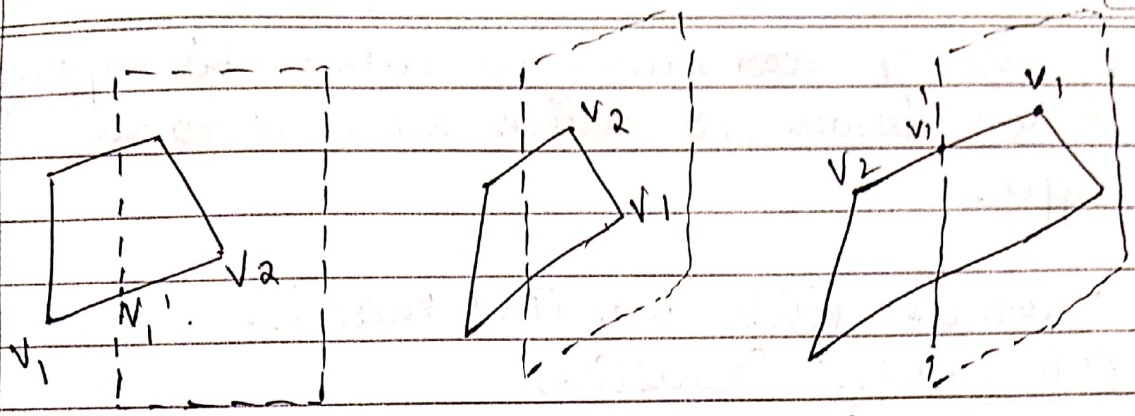
S3.2 : else you need clipping.

a) choose an endpoint of the line that is outside the window.
b) Find the intersection point at the window boundary
c) Replace endpoint with the intersection point & update the region code.
d) Repeat step 2 until we find a clipped line until totally accepted or rejected.
S4 :- Repeat step1 for other lines.



2) Illustrate Sutherland-Hodgman polygon clipping algorithm.

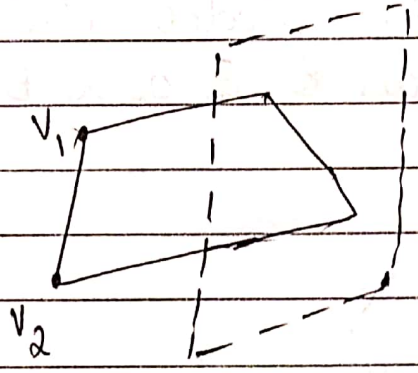
→ An efficient method for clipping a convex-polygon fill area, developed by Sutherland Hodgman. is to send the polygon vertices through each clipping stage so that clipped vector can be immediately processed to ~~ser~~ next stage.
• The final output is a list of vertices that describes the edges of the clipped polygon fill area the basic suther-land Hodgman algorithm is able to process concave polygon when the clipped fill area can be described with a single vertex list.



Out → in
output: V_1', V_2

in → in
o/p = V_2

in → out
output: V_1'.



out → out
output: none.

The selection of vertex edge of intersection of each clipped is given as follows:-

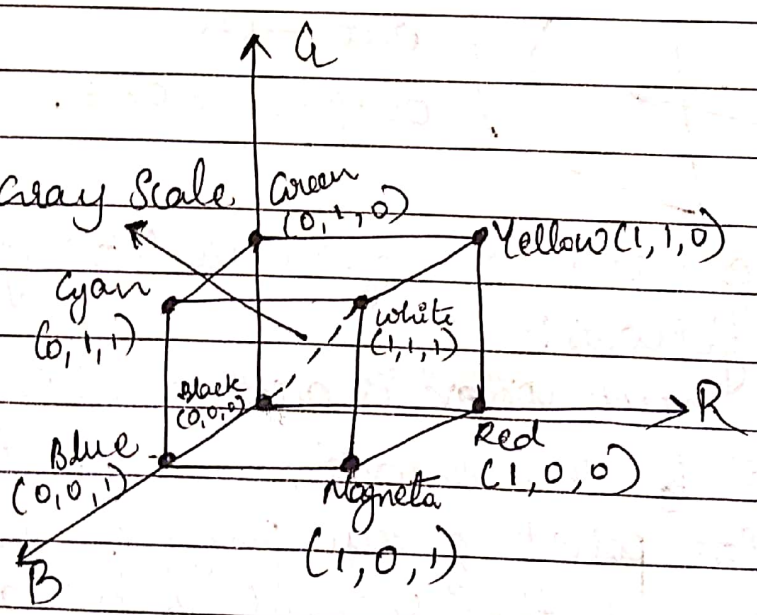
- If the first input vertex is outside this clipping-window ~~border~~ border & the second vertex is inside, both the intersection point of the polygon edge with the window border & the second vertex are sent to the next clipper.
- If both the input vertices are inside this clipping window border only the second vertex is sent to the next clipper.
- If the first vertex is inside this clipping window border and the second vertex is outside, only the polygon edge-intersection position with the clipping window border is sent to next clipper.

* If both i/p ~~vector~~ vertices are outside this clipping window border, no vertices are sent to the next clipper.

3) Illustrate RGB and CMY models.

→ RGB models. (additive)

- The three premiaries red, green, and blue, which is referred to as the RGB color model.
- We can represent this model using the unit cube defined on R, G and B axes.



The origin represents black & the diagonally opposite vertex, with coordinates (1,1,1) is white the RGB color scheme is an additive model.

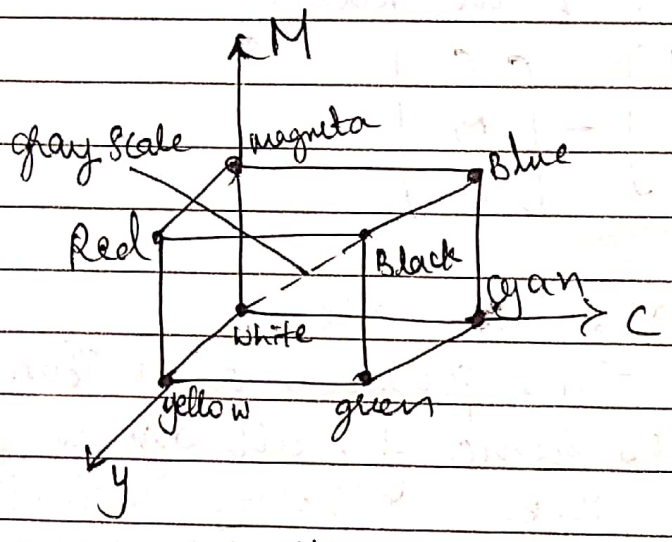
$$C(\lambda) = (R, G, B) = RR + GG + BB.$$

where parameters R, G and B are values in the range from 0 to 1.

CMY parameters

→ A subtractive color model can be formed with the three primary colors cyan, magenta & yellow.

- A unit cube representation for the CMY model is illustrated below:



- In CMY model, the spatial position (1,1,1) represents black, because all components of the incident light are subtracted.
- The origin represents white light.
- Equal amounts of each of the primary colors produce shades of gray along the main diagonal of the cube.
- A combination of cyan & magneta ink produces blue light, because the red and green components of the incident light are absorbed.
- Illy, a combination of cyan and yellow ink produces green light and a combination of magenta & yellow ink yields red light.

Conversion from RGB to CMY

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Conversion from CMY to RGB.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

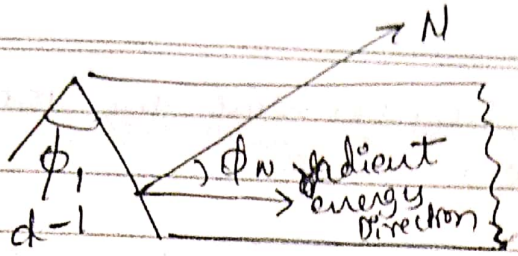
4) Illustrate illumination models.

→ Ambient light :- This produces a uniform ambient lighting that is same for all objects & it approximates the global diffuses reflections from the various illuminated surfaces.

- Reflections produced by ambient-light illumination are simply a form of diffuse reflections and they are independent of the viewing positions & the spatial orientation of a surface.

Diffuse reflection :- The incident light on the surface is scattered with equal intensity in all directions independent of the viewing position.

- Such surfaces are called ideal diffuse reflectors. They are also reffered to as lambertian reflectors because the reflected radiant light energy from any point on the surface is calculated with lambert's cosine law.



- The intensity of light in this direction can be completed as the ratio of the magnitude of the radiant energy per unit time divided by projection.

$$\text{Intensity} = \frac{\text{radiant energy per unit time}}{\text{projected area}}$$

$$= \frac{2\cos\phi_N}{2A\cos\phi_N} = \text{Constant}$$

This parameter is called diffuse reflection coefficient or diffuse reflectivity.

$$I_{ambdiff} = K_d I_a.$$

$$I_{incident} = I_1 \cos\theta.$$

$$I_{diff} = K_d I_{1,incident}$$

$$= K_d I_1 \cos\theta.$$

The diffuse reflections equation for single point source is

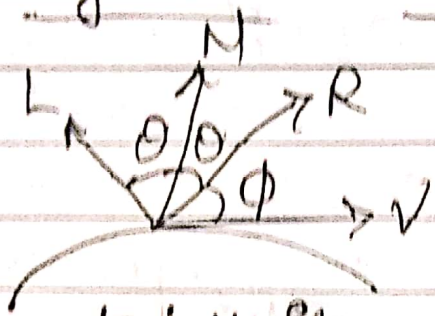
$$I_{1,diff} = \begin{cases} K_d I_1 (N.L, & \text{if } N.L > 0 \\ 0.0, & \text{if } N.L \leq 0. \end{cases}$$

$$L = \frac{P_{source} - P_{surf}}{|P_{source} - P_{surf}|}$$

Using parameter Ka, we can write the total diffuse equation

$$I_{diff} = \begin{cases} K_a I_a + K_d I_i (N \cdot L), & \text{if } N \cdot L > 0 \\ K_a I_a, & \text{if } N \cdot L \leq 0. \end{cases}$$

Specular reflection and the phong model.



- N represents : unit normal surface vector.
- R represent the unit vector.
- L is the unit vector directed toward the point light source.
- V is the unit vector pointing to the viewer, Angle ϕ is the viewing angle ~~ra~~ relative to specular reflection direction R.

Using the spectral reflection function $w(\theta)$, we can write.

$$I_{spec} = w(\theta) I_i \cos^{n_s} \phi.$$

$$I_{spec} = \begin{cases} K_d I_i (V \cdot R)^{(n_s)} & \text{if } V \cdot R > 0 \text{ \& } N \cdot L > 0 \\ 0.0 & \text{if } V \cdot R \leq 0 \text{ or } N \cdot L \leq 0. \end{cases}$$

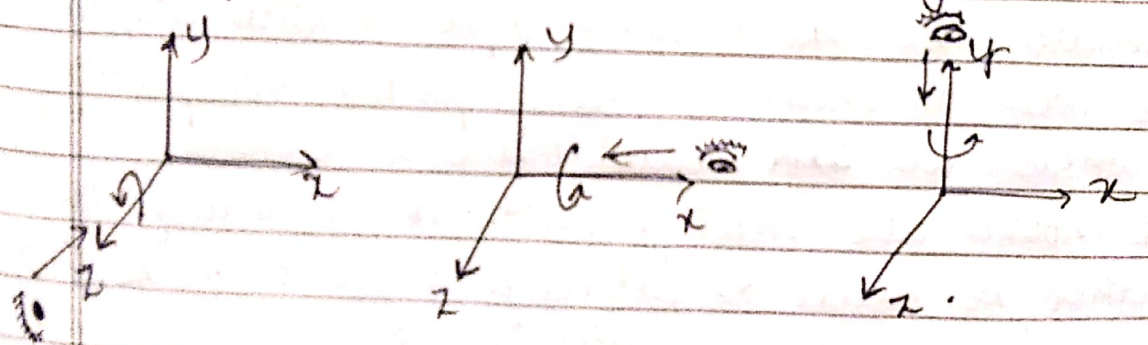
5) Illustrate depth buffer method.

→ • Initialize depth buffer and frame buffer so that all buffer position (x, y) depthBuff $(x, y) = 1.00$, frameBuffer = backgroundcolor.

• Process each polygon in a scene, one at a time.
→ For each projected (x, y) pixel position of a polygon, calculate the depth z.
→ If z < depthBuff(x, y), compute the surface color at that position & set
depthBuff(x, y) = z, frameBuff(x, y) = surfColor(x, y).
→ After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces and the frame buffer contains the corresponding color values for those surfaces.

6) Illustrate the three-dimensional rotation

→ • By convention, positive rotation angles produce counter clock wise rotations about a coordinates.
• Positive rotations about a coordinate axis are counter clockwise, when looking along the positive half of the axis towards the origin.



Three-Dimensional Coordinates Axis Rotations along z axis:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z.$$

In homogenous coordinate, the three dimensional z axis rotation equations are

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

Transformations equation for rotation about the other 2 coordinate axis can be

$x \rightarrow y \rightarrow z \rightarrow x$.

Along x axis :- $y' = y\cos\theta - z\sin\theta$

$z' = y\sin\theta + z\cos\theta$

$x' = x$.

Along y axis :- $z' = z\cos\theta - x\sin\theta$

$x' = z\sin\theta + x\cos\theta$

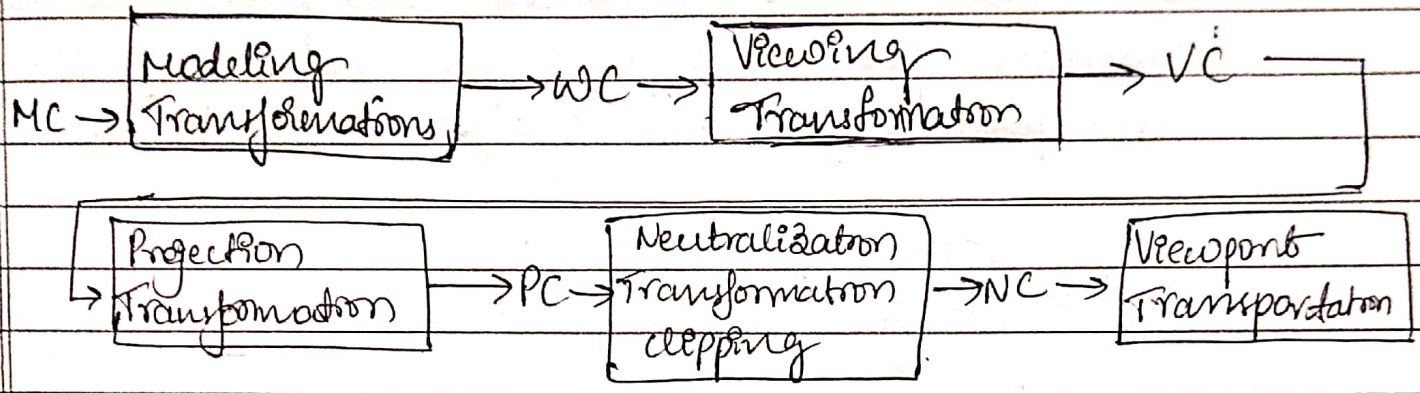
$y' = y$.

A inverse 3D rotation matrix is obtained in the same by replacing θ with $-\theta$.

(7) Illustrate the 3D viewing pipeline with diagram.

→
- We need to choose a viewing position corresponding to where we would place a camera.
- We choose the viewing positions according to whether we want to display a front, back, side, top or bottom view of the scene.
- We could also pick a position in the middle of a group of objects or even inside a single object, such as a building or a molecule.
- Then we must decide on the camera orientation.
- Finally, when we snap the shutter, the scene is cropped to the size of a selected clipping.

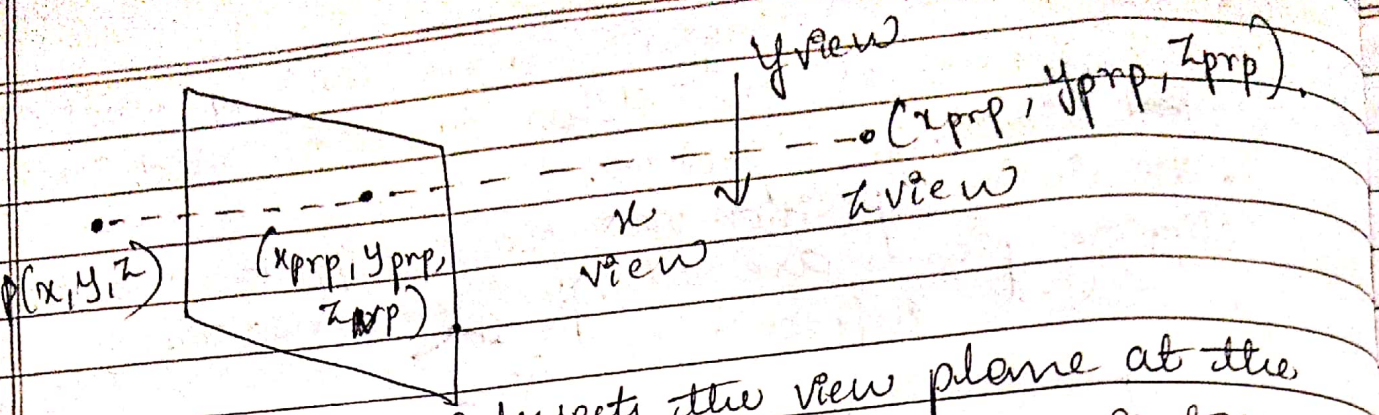
- Some of the viewing operations for a 3D scene are the same as or similar to those used in 2D viewing pipeline.
- The viewing position view plane clipping window & clipping planes are all specified within the viewing coordinate reference frame.



- A 2D dimensional clipping window, corresponding to a selected camera-lens.
- This clipping region is called the view volume.
- Projection operations are performed to convert the viewing coordinate descriptions of the scene to coordinate positions on the projection plane.
- The clipping operations can be applied after all device-independent coordinate transformations.
- The final step is to map viewport coordinate within a selected display window.

8) Illustrate perspective projection transformation Co-ordinate.

→ Figure below show the projection paths of a spatial position (x, y, z) to a general projection point at $(x_{prp}, y_{prp}, z_{prp})$.



- The projection line intersects the view plane at the coordinate position (x_p, y_p, z_{vp}), where z_{vp} is some selected position for the view plane on the zview axis.

$$x' = x - (x - x_{prp})u$$
$$y' = y - (y - y_{prp})u \qquad 0 \le u \le 1$$
$$z' = z - (z - z_{prp})u$$

on the view plane $z' = z_{vp}$ & we can solve z' eqn. for parameter u at this position.

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

Substituting this value of a u into the eqn. of x' & y'. we obtain the general perspective-transformation eqns.

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

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

9) Illustrate perspective projection transformation matrix.

→ We can use a 3D, in homogenous coordinate representation to emphasize the perspective equ. in the form

$$x_p = \frac{x_h}{h}, \quad y_p = \frac{y_h}{h}.$$

Where the homogenous parameter has the value

$$h = z_{prp} - z$$

$$x_h = x(z_{prp} - z_{vp}) + x_{prp}(z_{vp} - z)$$

$$y_h = y(z_{prp} - z_{vp}) + y_{prp}(z_{vp} - z).$$

- The perspective-projection transformation of a viewing coordinate position is then accomplished in 4 steps
- First, we calculate the homogenous coordinates using the perspective transformation matrix

$$P_h = M_{pers} \cdot P.$$

where P_h is the column matrix representation of the homogenous point (x_h, y_h, z_h, h) and P is the column matrix representation of the coordinate position $(x, y, z, 1)$.

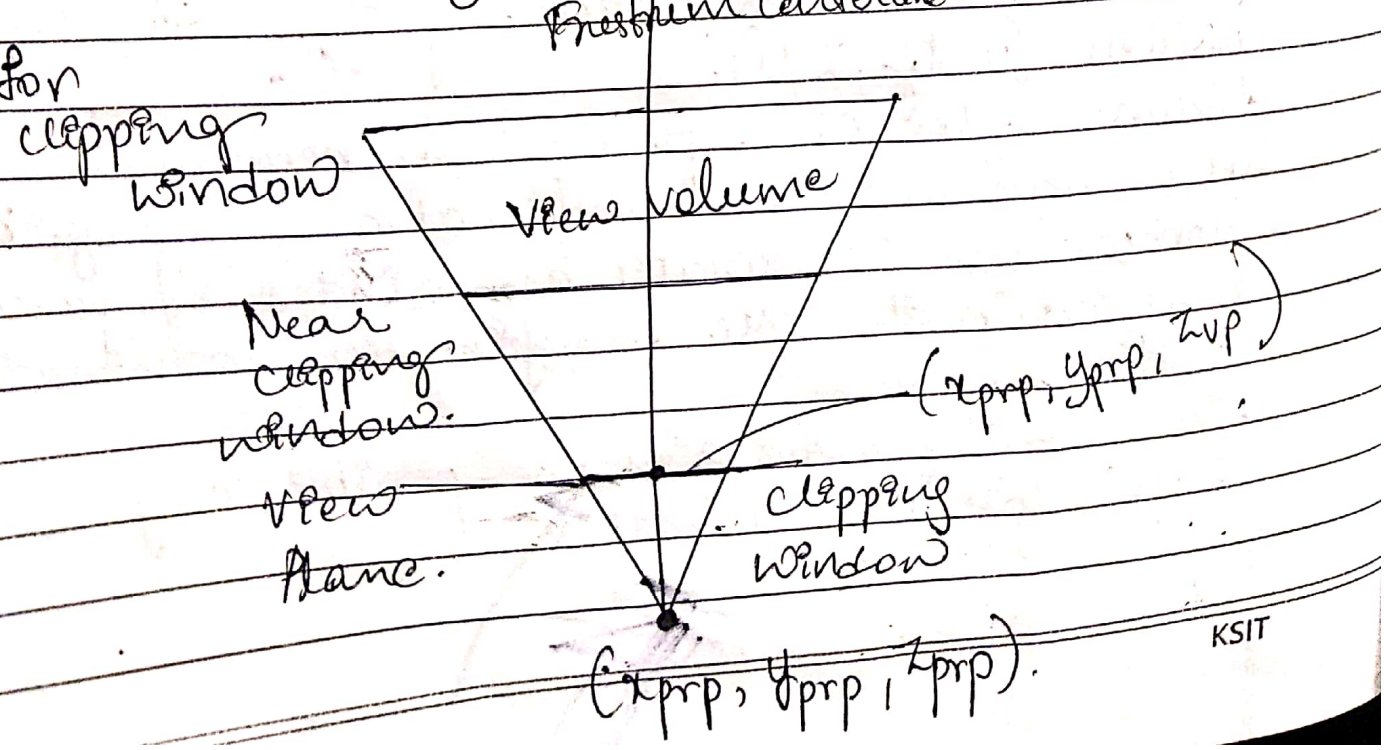
- Second, after other process have been applied, such as the normalization transformation & clipping routines, homogenous coordinates are divided by parameter h to obtain the true transformation coordinate positions.
- The following matrix gives one possible way to formulate a perspective projection matrix.

$$M_{pp} = \begin{bmatrix} z_{prp} - z_{vp} & 0 & -x_{prp} & -x_{prp} z_{prp} \\ 0 & z_{prp} - z_{vp} & -y_{prp} & -y_{prp} z_{prp} \\ 0 & 0 & -s_z & t_z \\ 0 & 0 & -1 & z_{prp} \end{bmatrix}$$

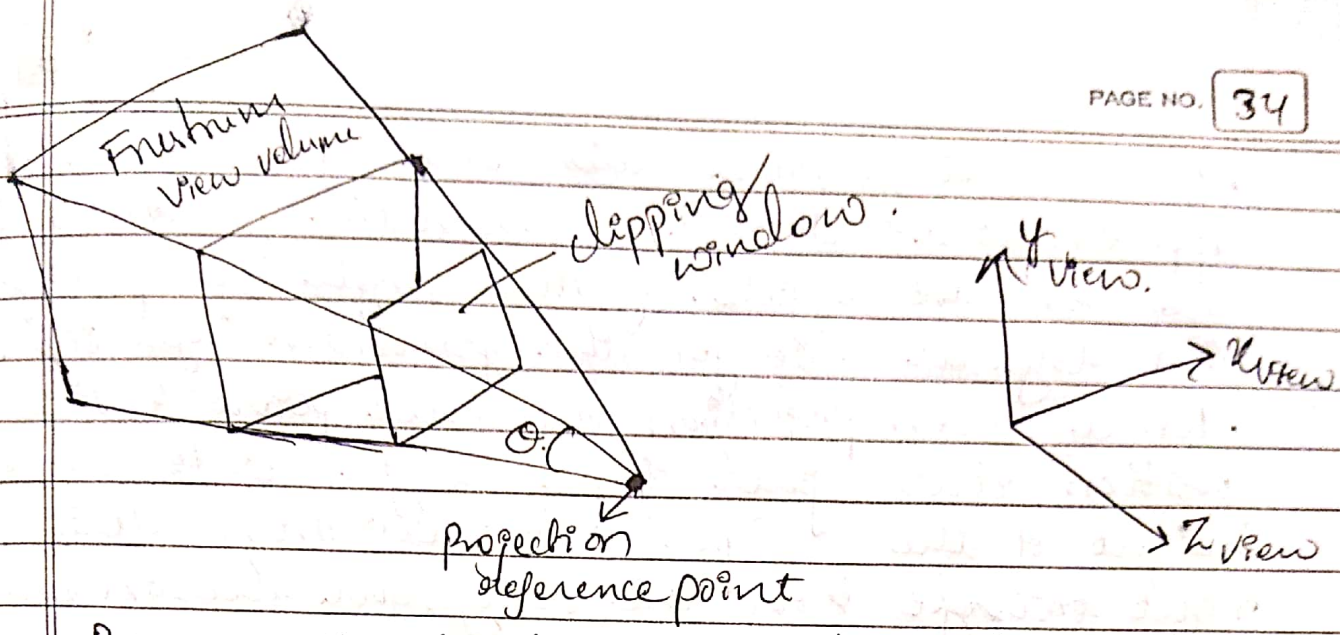
- Parameters s_z & t_z are the scaling and translation factors for normalizing the projected values of Z-Coordinate.
- Specific values for s_z & t_z depend on the normalization range we select.

10) Illustrate perspective projection frustrum. (frustrum viewing volume)

=> • The line from the projection reference point through the centre of the clipping window and on through the view volume is the centre line for a perspective projection frustrum.
• If this centre line is $\perp^r$ to the viewplane, we have a symmetric frustrum (wrt to its centreline).



KSIT



- Because the frustum centre line intersects the view plane at the coordinate location $(x_{prp}, y_{prp}, z_{vp})$ we can express the corner position for the clipping window in terms of the window dimensions.

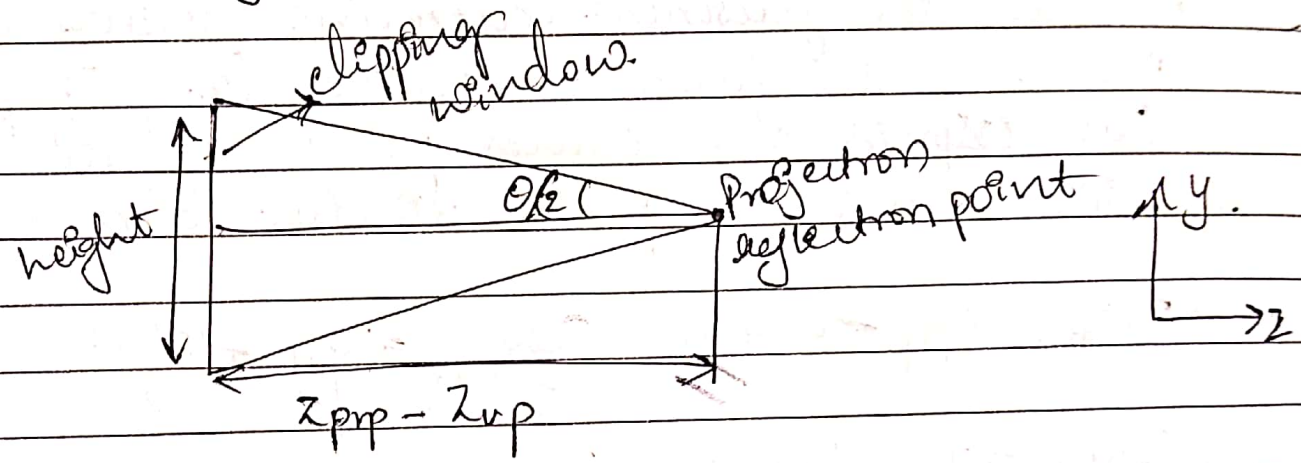
$$xw_{min} = x_{prp} - \frac{width}{2}, \quad xw_{max} = x_{prp} + \frac{width}{2}$$

$$yw_{min} = y_{prp} - \frac{height}{2}, \quad yw_{max} = y_{prp} + \frac{height}{2}$$

- Another way to specify a symmetric perspective projection is to use parameters that approximate the properties of a camera lens.
- A photograph is produced with a symmetric perspective projection of a scene onto the film plane.
- Reflected light rays from the objects in a scene are collected on the film plane from the "cone of vision" of the camera.
- This cone of vision can be referred with a field of view angle, which is a measure of the size of the camera lens.
- A large field of view angle, for example corresponds to a wide-angle lens.

• In Computer graphics, the cone of vision is approximated with a symmetric frustrum and we can use a field of view angle to specify an angular size for the frustrum refer fig.

• For a given projection reference point & view plane position, the field of view angle determines the height of the clipping window from the right triangle in the diagram shown below



$$\tan(\theta/2) = \frac{\text{height}/2}{z_{prp} - z_{vp}}$$

So that the clipping window height can be calculated as $\text{height} = 2(z_{prp} - z_{vp}) \tan(\theta/2)$

$$z_{prp} - z_{vp} = \frac{\text{height}}{2} \cot(\theta/2)$$

$$= \frac{\text{width} - \cot(\theta/2)}{2 \cdot \text{aspect}}$$