Computer Graphics, MID Question - 2018-19

Question - 4: (From book)

Here, The window parameters are,

$WX_{max} = 3$, $WX_{min} = 1$, $WY_{max} = 5$, $WY_{min} = 1$ and

the viewport parameters are,

$VX_{max} = \frac{1}{2}$, $VX_{min} = 0$, $VY_{max} = \frac{1}{2}$, $VY_{min} = 0$

Now calculating —

$$S_x = \frac{VX_{max} - VX_{min}}{WX_{max} - WX_{min}} = \frac{\frac{1}{2} - 0}{3 - 1} = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

$$S_y = \frac{VY_{max} - VY_{min}}{WY_{max} - WY_{min}} = \frac{\frac{1}{2} - 0}{5 - 1} = \frac{1}{2} \times \frac{1}{4} = \frac{1}{8}$$

and,

$$N = \begin{pmatrix} S_x & 0 & -S_x WX_{min} + VX_{min} \\ 0 & S_y & -S_y WY_{min} + VY_{min} \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & \frac{1}{8} & -\frac{1}{8} \\ 0 & 0 & 1 \end{pmatrix}$$

Question:- 01

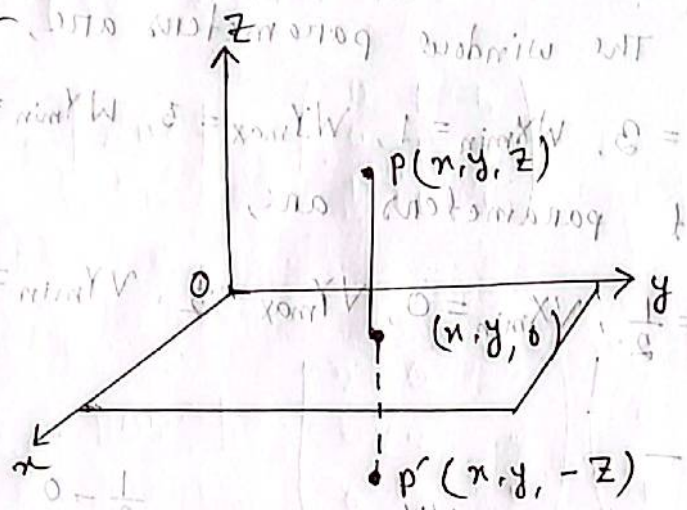


Fig:- a

From Fig-a, it is easy to see that the reflection of $P(x, y, z)$ is $P'(x, y, -z)$. The transformation that performs this reflection is -

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Question:- 06: [From book]

Anti-aliasing - There are techniques that can greatly reduce aliasing artifacts and improve the appearance of images without increasing their resolution. These techniques are collectively referred to as anti-aliasing techniques.

Some anti-aliasing techniques are designed to treat a

particular type of artifact.

Question: 07 [From books]

We can write the required transformation with $V = 5I + 2J$ as-

$$S_{2,2,C} = T_V \cdot S_{2,2} \cdot T_{-V}$$

$$= \begin{pmatrix} a & 0 & -ah+h \\ 0 & b & -bk+k \\ 0 & 0 & 1 \end{pmatrix}$$

Here,
$a = 2$
$b = 2$
$h = 5$
$k = 2$

$$= \begin{pmatrix} 2 & 0 & -5 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

Representing a point P with coordinates (x, y) by the column vector $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$, we have

$$S_{2,2,C} \cdot A = \begin{pmatrix} 2 & 0 & -5 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -5 \\ -2 \\ 1 \end{pmatrix}$$

$$S_{2,2,C} \cdot B = \begin{pmatrix} 2 & 0 & -5 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -3 \\ 0 \\ 1 \end{pmatrix}$$

$$S_{2,2,C} \cdot C = \begin{pmatrix} 2 & 0 & -5 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}$$

So, $A' = (-5, -2)$, $B' = (-3, 0)$, $C' = (5, 2)$

$$[ABC] = \begin{pmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

Now,

$$S_{2,2,c} \cdot [ABC] = \begin{pmatrix} 2 & 0 & -5 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} -5 & -3 & 5 \\ -2 & 0 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= [A'B'C']$$

Question: 02 [From slide]

Given that, $P_1(0, 0)$, $P_2 = (4, 6)$

First workout m and c →

$$m = \frac{6-0}{4-0} = \frac{3}{2}$$

$$c = (y_1 - m \cdot x_1) = 0 - \frac{3}{2} \cdot 0 = 0$$

Now, work for each x value workout the y value:

$$y(1) = \frac{3}{2} \cdot 1 + 0 = \frac{3}{2} \approx 2$$

$$y(2) = \frac{3}{2} \cdot 2 + 0 = \frac{6}{2} = 3$$

$$y(3) = \frac{3}{2} \cdot 3 + 0 = \frac{9}{2} \approx 5$$

The points that made up the line P1P2: (1,2), (2,3), (3,5)

Question: 05 [From AI]

An image's aspect ratio is the proportional relationship between its width and height, usually expressed as a ratio like 16:9, where the first number represents the width and the second number represents the height.

No, a simultaneous sheaning is not the same as sheaning in one direction followed by another because the order of operations in a simultaneous shear can create different results due to interactions between the sheaning forces along different axes.

Mid Question, 4th batch

Question : 01 [From book]

Major side effects of scan conversion:

① Staircase: A common example of aliasing effects is the staircase or jagged appearance, we see when scan-converting a primitive such as a line or a circle. we also see the "jaggies" along the border of a filled region.

② Unequal Brightness: Sometimes, lines at different angles can look uneven in brightness! A slanted line might seem dimmer compared to a straight horizontal or vertical line.

③ picket fence problem: The picket fence problem occurs when an object is not aligned with, or does not fit into, the pixel grid properly.

Rest of the questions are repeated - - - .

[illegible], 18-19

Final

Question-01

(a) "Midpoint circle Algorithm" was developed to address certain uncertainties and challanges in drawing circles.

Uncertain cases that motivated the midpoint circle algorithm:

(i) Pixel Ambiguity: The midpoint circle algorithm calculates the decision parameter to determine the appropriate pixel to color, reducing ambiguity.

(ii) Optimization for simplicity: midpoint circle algorithm is designed for simplicity, making it easier to implement and more computationally efficient.

(iii) Error Correction: It utilizes the concept of a "decision parameter" that analizes the distance between the center and the next potential pixel. Based on this, it decides whether to move diagonally or stay horizontaly

vertical, effectively filling in the gaps and reducing discretization errors.

(iv) Vertical and Horizontal symmetry: Circles look the same from all sides. So we make bad pixel choices, the circle can look uneven or broken. picking the wrong pixel can ruin the shape. The algorithm draws part of the circle and, then mirrors the points to keep it even.

1(b) see mid Ques. : 2 (18-19)

2(a)

Let a Line with endpoint A(120,-110), B(-10,-20)

we have 3 clipping catagories:

(i) visible
(ii) Not visible
(iii) clipping candidate.

At first we have to find the categories to which AB belongs.

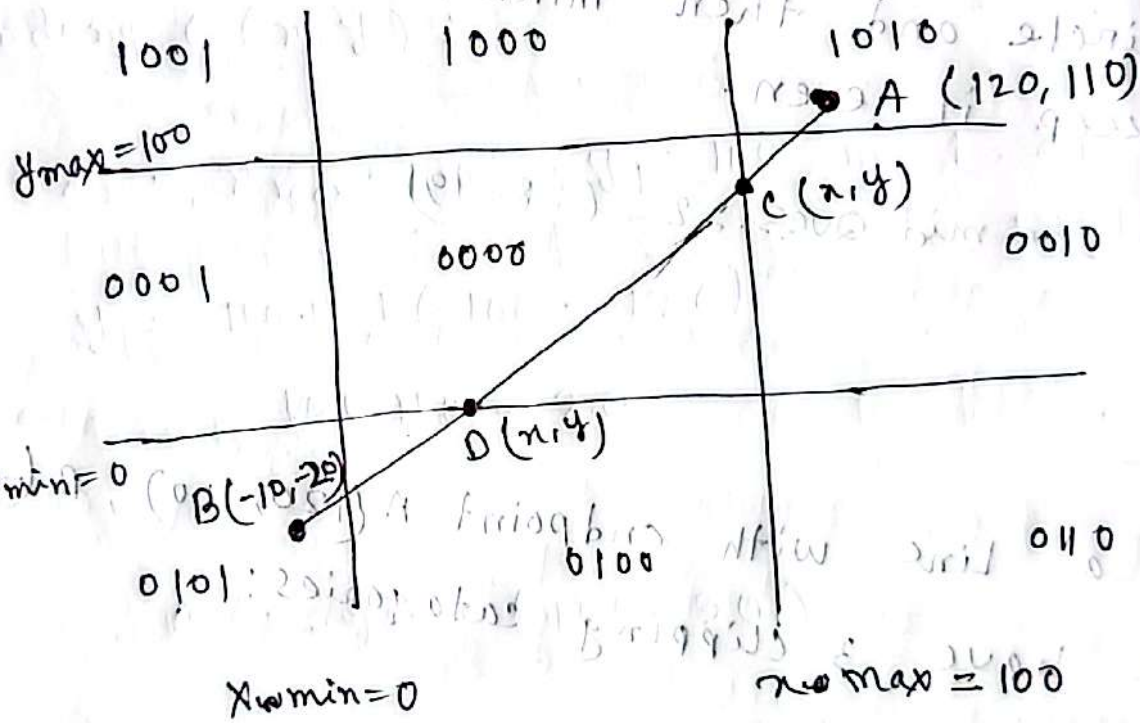
① At first assign 4 bit region code to each points.

Bit 1 = 1 $y > y_{max}$

Bit 2 = 1 $y < y_{min}$

Bit 3 = 1 $x > x_{max}$

Bit 4 = 1 $x < x_{min}$



② ⇒ completly inside if :

Bitwise or = 0000

⇒ completly outside if :

Bitwise And ≠ 0000

⇒ partially inside / outside (clipping candidate) if : Bitwise And = 0000

So, in this case taking Bitwise OR:

$$\begin{array}{r} 0010 \\ 0101 \\ \hline 0111 \end{array} \neq 0000$$ so not completly visible.

Now taking Bitwise And:

$$\begin{array}{r} 0010 \\ 0101 \\ \hline 0000 \end{array}$$

Here we can see that result is 0000.

So, it has clipping - candidate. Now, we will clip it using cohen - Sutherland algorithm:

Lets find the slope of AB:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-20 - 110}{-10 - 120} = 1$$

The clipping points will be C and D.

Intersection points:

C intersects with x_{max} and D intersects with y_{min}.

The equation that we will use to find intersection points

$$y_i = y_1 + m(x_i - x_1)$$
$$x_i = x_1 + (y_i - y_1)/m$$

$x_i = x_{min}$ or x_{max}

$y_i = y_{min}$ or y_{max}

Finding C(x, y):

$x_i = x_{max} = 100$, $y_1 = 110$, $m = 1$, $x_1 = 120$

$y_i = 110 + 1(100 - 120)$

$= 110 - 20$

$= 90$

$\therefore C(x, y) = (100, 90)$

Finding D(x, y):

$y_i = y_{min} = 0$, $x_1 = -10$, $y_1 = -20$, $m = 1$

$x_i = -10 + \{0 - (-20)\}/1$

$= -10 + 20$

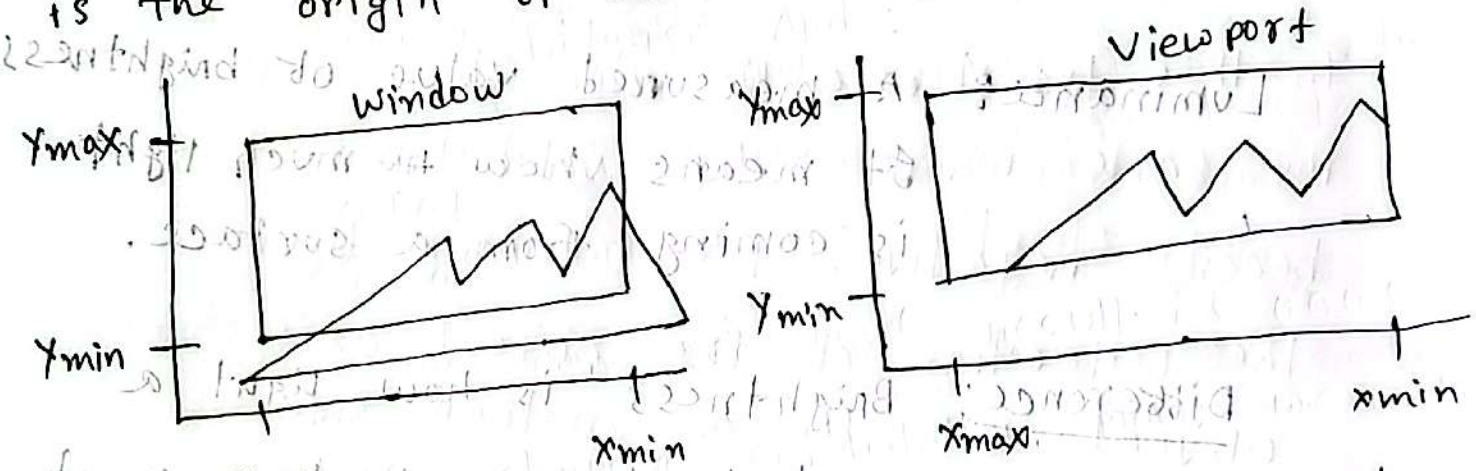
$= 10$

$\therefore D(x, y) = (10, 0)$

This is how we find intersection points.

2 (b)

To establish the viewing-coordinate reference frame, we first pick a world-coordinate position, called the view reference point. This point is the origin of our viewing-coordinate system.



Camera setup: Decide where the camera is and how it's looking at the scene.

Shift to camera's perspective: move everything so the camera is at the center.

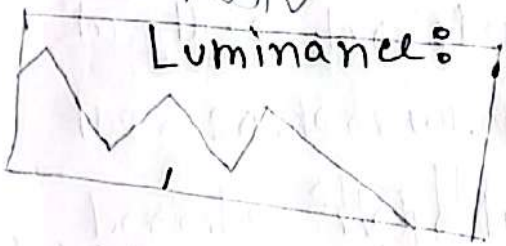
projection: project the 3D scene onto a 2D plane for display.

Adjust for screen: scale and move the projected scene to fit the screen.

3(a)

(i) Brightness / Luminance

Brightness: How light or dark a color appears to the human eye.



Luminance: A measured value of brightness. It means how much light is coming from a surface.

Difference: Brightness is how light a color feels, and luminance is how much light it gives off or reflects.

(ii) Hue / Dominant

Hue: This is what we usually mean by color like red, green, blue etc.

Dominant: This is the scientific reason for the hue.

Difference: Hue is the color name we say, and dominant is the light wave that create that color.

(iii) Saturation / Excitation purity-

Saturation: This tells us how pure or intense a color is.

⇒ A fully saturated color looks rich and vivid.

⇒ A less saturated color looks dull or faded!

Excitation purity: It tells us how far a color from gray on the color spectrum

Difference: Saturation is how rich a color looks, and excitation purity is how scientists measure that richness.

2(c)

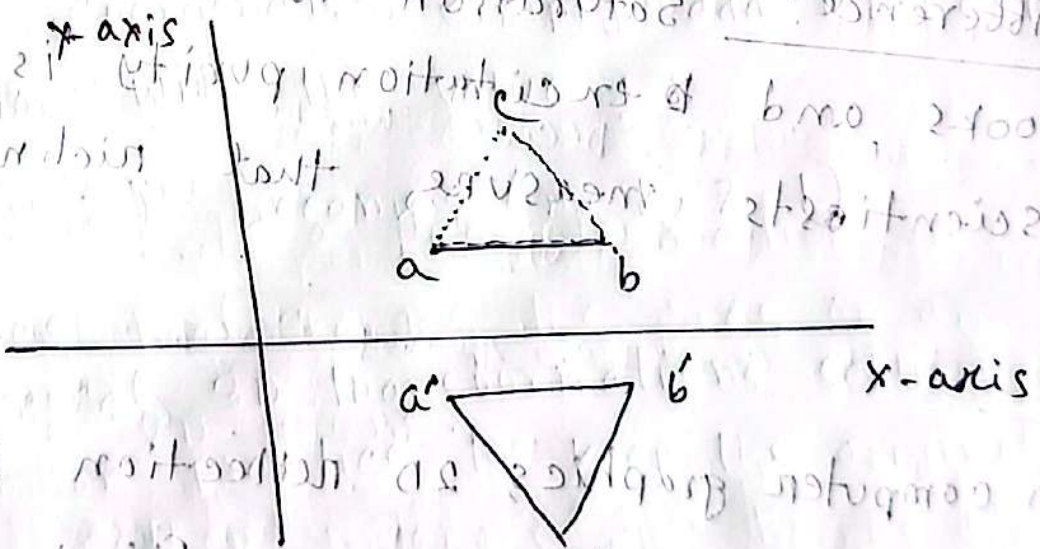
In computer graphics, 2D reflection is a technique that involves mirroring or flipping an object on coordinate system across a specific axis in a 2D plane.

(1) Reflection about the X-axis: The object can be reflected on the x-axis with the help

of the following matrix.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In this transformation value of x will remain the same whereas the value of y will become negative. The following figures show the reflection of the object axis. The object will lie on another side of the x-axis.



4 (a)

Geometric transformation	Coordinate transformation
① Changes the shape, size, or position of ~~the~~ objects	① Manipulates the coordinates of objects
② Collects object properties	② Does not preserve object properties
③ Deals with the objects as a whole	③ Deals with individual points and their transformations
④ Directly modifies the properties of objects	④ Manipulates the mathematical coordinates defining objects
⑤ Used in computer graphics, animation, and visual effects	⑤ Applied in navigation systems, robotics, and GPS technology.

4(b)

In computer graphics, scaling can be done without a fixed point by uniformly adjusting the size of the entire object. While certain scaling operations may involve fixed points for specific effects or transformation, a general scaling process does not strictly require a fixed point.

4(c)

To convert 3D coordinates (x, y, z) to homogeneous coordinates, simply add an extra dimension with the value 1:

(x, y, z) becomes $(x, y, z, 1)$

For typical 3D transformations, setting $w = 1$ is common practice to keep the original size. Homogeneous coordinates with $w = 0$ represent points at infinity.

Therefore, the standard conversion is:

$$(x, y, z) \rightarrow (x, y, z, 1)$$

(4) To convert homogeneous coordinates to standard 3D coordinates, ~~you~~ simply perform division by the last component (w). Here's how it works:

(1) Divide each component by w:

$$x = x/w$$
$$y = y/w$$
$$z = z/w$$

(2) The resulting values (x, y, z) represent the standard 3D coordinates.

4(d) (15-16)

Blobby objects are objects which change their shape and size on the basis of their states. Blobby objects are shapes that look soft, round, and a bit undefined, like blobs. They don't have clean edges.

Model :

One simple model for representing blobby objects is the Metaball model. Imagine

these blobs as flexible, stretchable, and soft materials like jelly. metaballs are like blobs that can merge or separate based on their proximity. when blobs come close, they blend together, creating a smooth, continuous surface. This model is often used in computer graphics for things like water droplets, clouds, or other amorphous shapes.

4(d) (18-19)

2D scaling

2D scaling is a transformation in computer graphics that changes the size of an object in the 2D plane.

⇒ It either enlarges or shrinks the objects

⇒ scaling is done with respect to the origin unless otherwise specified.

Scaling matrix

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

S_x = Scaling factor along the X-axis.

S_y = Scaling factor along the Y-axis.

Scaling an object to 1/2 with respect to X-axis.

$S_x = \frac{1}{2}$

$S_y = 1$

Scaling matrix in this case:

$$S = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{bmatrix}$$

If we multiply the matrix with point (x, y) we get:-

$$\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{x}{2} \\ y \end{bmatrix}$$

5(a)

We want to convert a point from the world coordinate system to screen coordinates.

Let's assume:

world coordinate window –

$$x_{min} \leq x \leq x_{max}, \quad y_{min} \leq y \leq y_{max}$$

screen coordinate viewport –

$$x_v \leq x_s \leq x_v + w_v, \quad y_v \leq y_s \leq y_v + h_v$$

where,

x, y = point in world coordinates

x_s, y_s = corresponding screen coordinates

x_v, y_v = viewport origin

W_v, h_v = width and height of the viewport

Normalize the world coordinate to $[0,1]$

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}}$$

$$y_n = \frac{y - y_{min}}{y_{max} - y_{min}}$$

Now convert the normalized value to actual screen coordinates :-

$$x_s = x_v + x_n \cdot w_v = x_v + \left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) \cdot w_v$$

$$y_s = y_v + y_n \cdot h_v = y_v + \left(\frac{y - y_{min}}{y_{max} - y_{min}}\right) \cdot h_v$$

So, World-to-screen transform vector :-

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x_v + \left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) \cdot w_v \\ y_v + \left(\frac{y - y_{min}}{y_{max} - y_{min}}\right) \cdot h_v \end{bmatrix}$$

5(b)

Given that,

$x_{min} = -3$

$y_{min} = -3$

$x_{max} = 2$

$y_{max} = 1$

$U_{min} = 30$

$U_{max} = 80$

$V_{min} = 10$

$V_{max} = 30$

window : $(x_{min}, y_{min}) = (-3, -3)$

$(x_{max}, y_{max}) = (2, 1)$

viewport : $(V_{min}, V_{min}) = (30, 10)$

$(V_{max}, V_{max}) = (80, 30)$

plugging the values into the equation :

$$P' = \begin{bmatrix} (x - x_{min}) \cdot \frac{V_{max} - V_{min}}{x_{max} - x_{min}} + V_{min} \\ (y - y_{min}) \cdot \frac{V_{max} - V_{min}}{y_{max} - y_{min}} + V_{min} \end{bmatrix} \begin{matrix} x \\ y \end{matrix}$$

$$\Rightarrow P' = \begin{bmatrix} (x+3) \cdot \frac{80-30}{2-(-3)} + 30 \\ (y+3) \cdot \frac{30-10}{1-(-3)} + 10 \end{bmatrix} \begin{matrix} x \\ y \\ 1 \end{matrix}$$

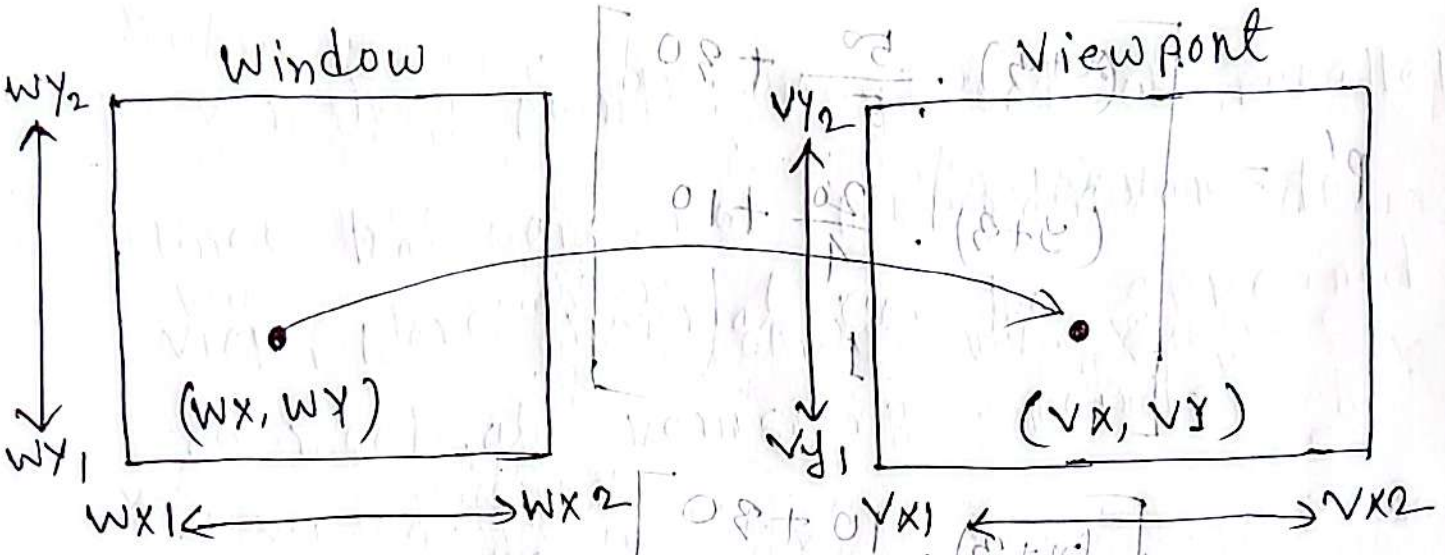
$$\Rightarrow P' = \begin{bmatrix} (x+3)\cdot\frac{50}{5}+30 \\ (y+3)\cdot\frac{20}{4}+10 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} (x+3)\cdot 10+30 \\ (y+3)\cdot 5+10 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 10x+60 \\ 5y+25 \\ 1 \end{bmatrix}$$

5(c)

Viewing transformation: is the process in computer graphics that converts objects from the world coordinate system to the screen coordinates.



world-coordinates screen coordinates

$$VX = VX_1 + (Wx - WX1) * (VX2 - VX1) / (WX2 - WX1);$$

$$Vy = V_{y1} + (W_y - W_{y1}) * (W_{y2} - V_{y1}) / (W_{y2} - W_{y1});$$

6 (a)

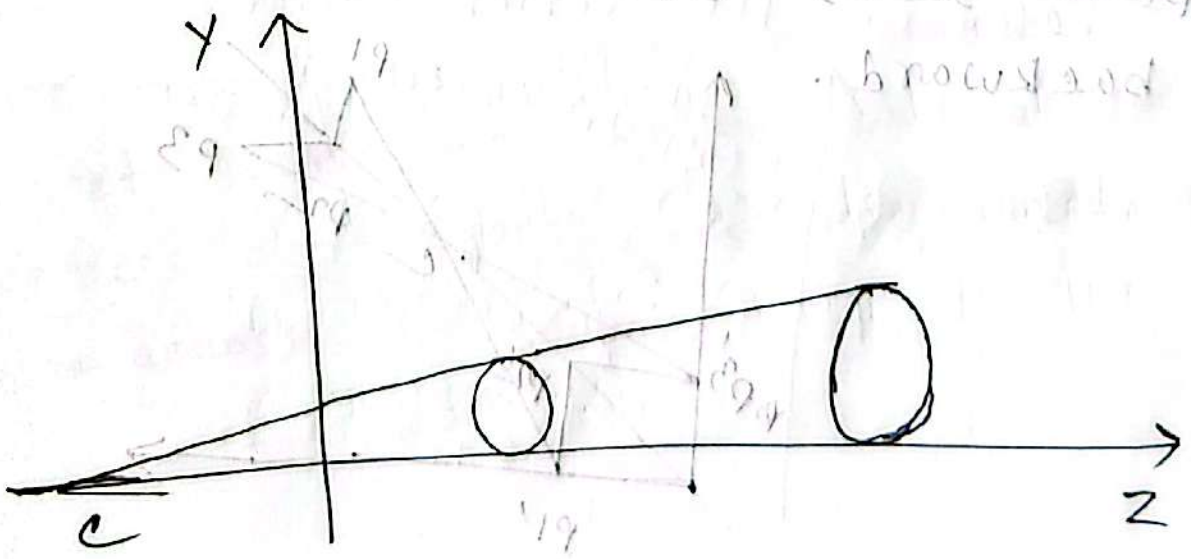
Our human visual system is most sensitive to three specific colors: red, green, and blue. This sensitivity is because our eyes have special cells called 'cones' that are tuned to respond to different wavelengths of light. Red light stimulates the cones that are most sensitive to longer wavelengths.

Green light stimulates the cones sensitive to medium wavelengths.
Blue light stimulates the cones sensitive to shorter wavelengths.

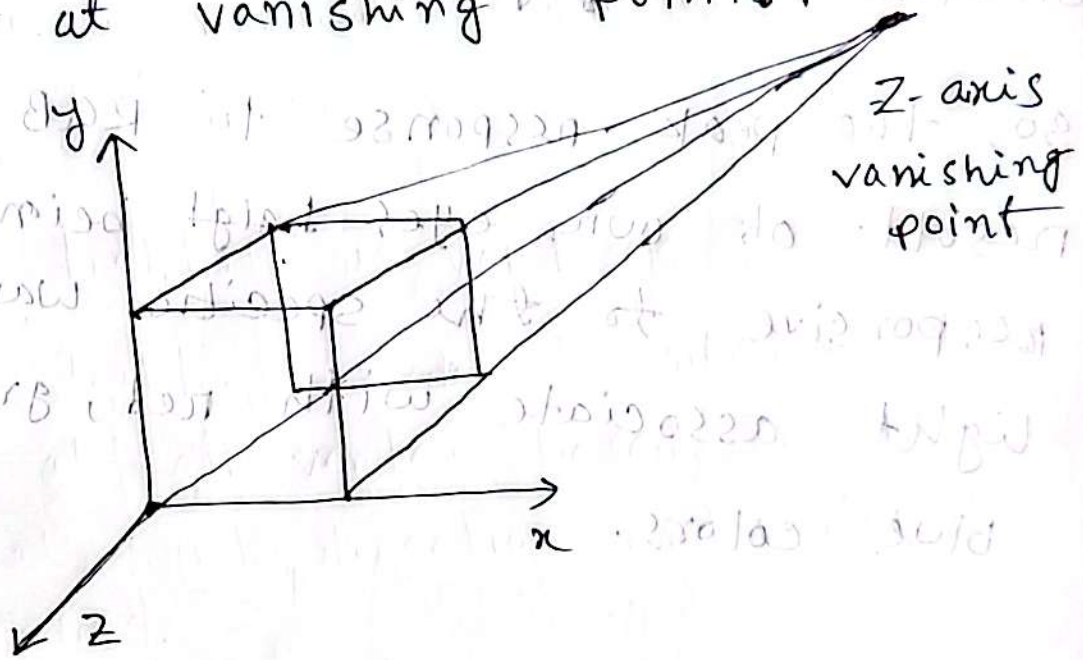
So, the peak response to RGB colors is a result of our eyes being most responsive to the specific wavelengths of light associate with red, green, and blue colors.

6 (b)

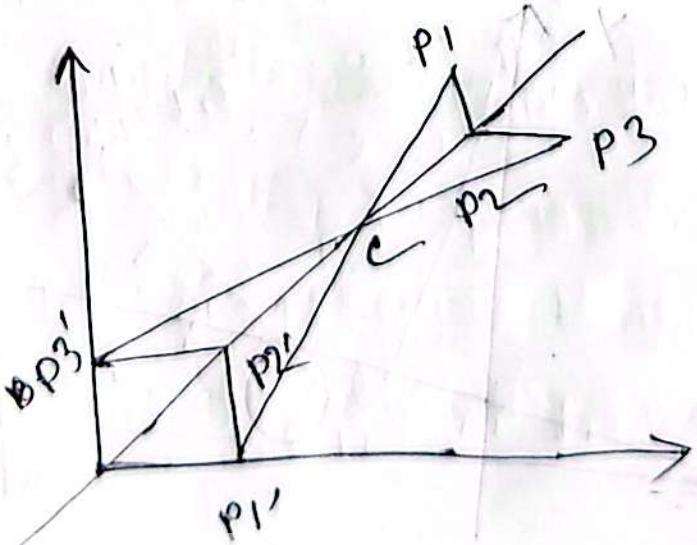
(i) Foreshortening: An object appears smaller if it further from center of projection (COP).



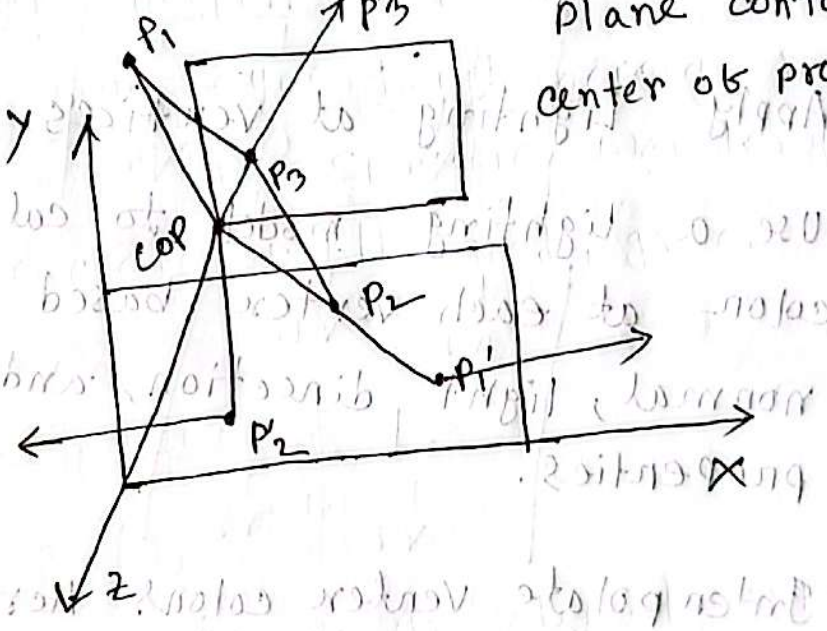
(II) Vanishing points: Any set of parallel lines that are not perpendicular to view plane normal, can be appeared to meet at vanishing points.



(III) View confusion: If any object exist behind the cop (center of projection), then it can be projected onto the view-plane seems like upside down and backward.



(iv) Topological distortion: Consider all points on a plane of these points are parallel to view plane and passes through the cop, then these points are projected to a broken line of infinite degree.



plane containing center of projection.

6(c) Gouraud shading is a surface rendering technique used in computer graphics to produce smooth color transitions across the surface of a 3D object. It computes lighting only at the vertices of polygons, and interpolates the color across the surface.

Gouraud Shading Algorithm - Step by step

(i) Calculate Normal at each vertex

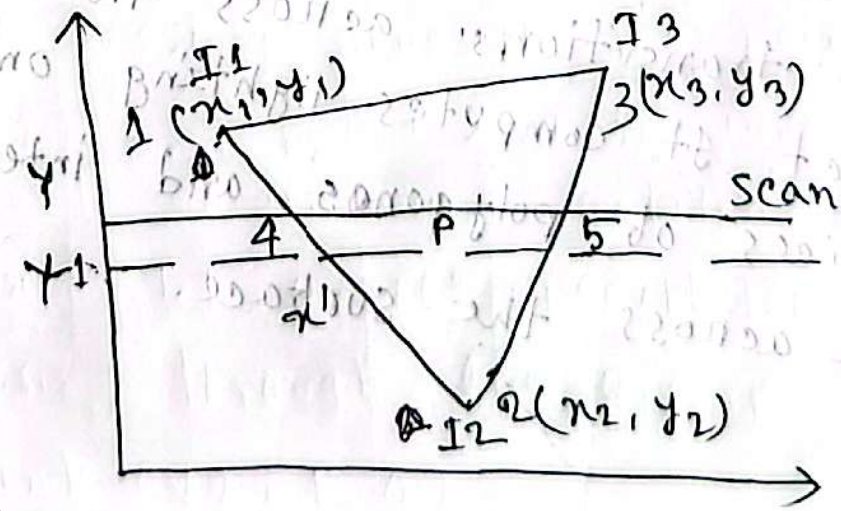
⇒ For a curved or complex surface, compute a normal vector at each vertex.

(ii) Apply Lighting at vertices

→ Use a lighting model to calculate the color at each vertex based on the normal, light direction, and material properties.

(iii) Interpolate vertex colors across surface

→ Use linear interpolation to fill in the color of each pixel inside the triangle based on the vertex color.



$$I = \left(\frac{y - y_2}{y_1 - y_2}\right) * I1 + \left(\frac{y_1 - y}{y_1 - y_2}\right) * I2$$

2 (a)

Translate the line to the origin

$P_1 = (1, 3, 2)$, $P_2 = (2, 4, 3)$

$P_1' = (0, 0, 0)$

$P_2' = P_2 - P_1$

$= (2-1, 4-3, 3-2)$

$= (1, 1, 1)$

(b) rotate the line about y-axis, $\theta = 45°$

$$Ry(\theta) = \begin{bmatrix} \cos\frac{\pi}{4} & 0 & \sin\frac{\pi}{4} \\ 0 & 1 & 0 \\ -\sin\frac{\pi}{4} & 0 & \cos\frac{\pi}{4} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Apply the rotation to P_2' —

$$P_2'' = R_y\left(\frac{\pi}{4}\right) \cdot P_2'$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} \\ \\ -\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{2} \\ 1 \\ 0 \end{bmatrix}$$

So, the line passing through the points $P_1 = (1, 3, 2)$ and $P_2 = (2, 4, 3)$, when translated to origin and rotated about the y axis by $0-45°$, result in a line passing through the origin, and the point $(\sqrt{2}, 1, 0)$

7(b)

Lighting is a crucial element of computer animation. It can enhance the mood, realism, and expression of a scene. Lighting can also be used to:

(I) create a 3D effect by separating the foreground from the background.

(II) merge the foreground and background to create a flat 2D effect.

7(c)

YIQ color model

→ Y : Luminance (brightness)

→ I : In-phase (orange - blue hue information)

→ Q : Quadrature (purple - green hue information)

The YIQ model was used in the NTSC TV broadcasting system because it separates brightness (Y) from color (I and Q).

RGB to YIQ conversion formula:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where, R, G, B are values in the range [0, 1] or [0, 255]

8(a)

Gouraud shading	phong shading
(i) It was introduced by Henni gouraud in 1970	(i) Introduced by phong Bui ♦ Tuong in 1973
(ii) Evaluate the illumination at borden vertices and interpolates	(ii) Evaluate the illumination at every point of the polygon surface
(iii) Cheapen than the phong shading	(iii) More costly than Gouraud shading

(iv) Lighting equation used at each vertex	(v) Lighting equation Used at each pixel.
(v) Requires average processing and less time	(v) Requires high processing and more time
(vi) It gives less accurate results	(vi) It gives more accurate results.
(vii) Smooth surfaces	(vii) smooth and shining surfaces

8(b)

From triangle ABe and A'B'e

$$\frac{AB}{Be} = \frac{A'B'}{B'e}$$

$$\Rightarrow \frac{x}{z} = \frac{x'}{-d} \Rightarrow x' = \frac{x}{-(z/d)}$$

similarly, $y' = \frac{y}{-(z/d)}$ and,

$z' = -d$

$$(x', y', z', 1) \Rightarrow \left(\frac{x}{-(z/d)}, \frac{y}{-(z/d)}, -d, 1\right)$$

projective transformation

$$\begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{-(z/d)} \\ \frac{y}{-(z/d)} \\ -d \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -\frac{z}{d} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -\frac{z}{d} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} \xrightarrow{\text{perspective division}} \begin{bmatrix} \frac{x}{-(z/d)} \\ \frac{y}{-(z/d)} \\ -d \\ 1 \end{bmatrix}$$

12-18 final

(1) (b)

DDA Line Algo.	Bresenham Line Algo.
DDA stands for digital differential Analyzer	While it has no full form
less efficient	more efficient
calculation speed is less than Bresenham line algorithm	Faster than DDA algo.
Has less precision or accuracy	Has more precision or accuracy

(3)

3(a)

The line joining the viewpoint C(0, 0, -10) and point $P_1(1, 2, 0)$ is $x = t$, $y = 2t$, $z = -10 + 10t$

To determine whether $P_2(3, 6, 20)$ lies on this line, we see that $x = 3$ when $t = 3$, and then at $t = 3$, $x = 3$, $y = 6$, and $z = 20$. So, P_2 lies on the projection line through C and P_1.

Next, we determine which point is in front with respect to C. Now, C occurs on the line at $t=0$, P_1 occurs at $t=1$, and P_2 occurs at $t=3$.

Thus, comparing t values, P_1 is in front of P_2 with respect to C; that is, P_1 obscures P_2.

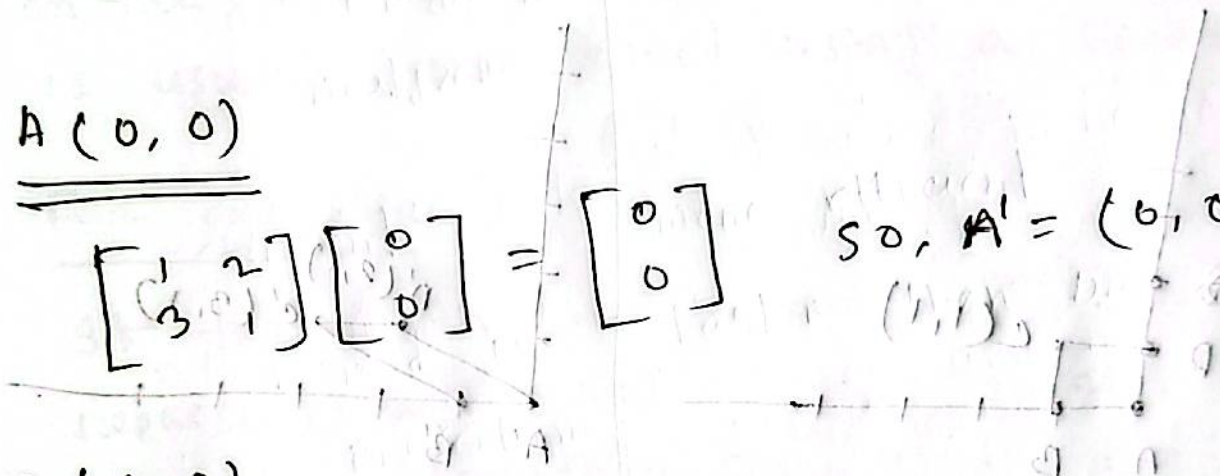
We now determine whether $P_3(2,4,6)$ is on the line. Now, $x=2$ when $t=2$ and then $y=4$, and $z=10$.

Thus $P_3(2,4,6)$ in not on this projection line and so it neither obscures nor is obscured by P_1 and P_2.

3(b)

The transformation matrix is given as $\begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix}$. when, $a=2$ and $b=3$ the matrix becomes $\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$.

Finding the transformed coordinates of the square A(0, 0), B(1, 0), C(1, 1) and D(0, 1).



A(0, 0)

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{So, } A' = (0, 0)$$

B(1, 0)

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \text{So, } B' = (1, 3)$$

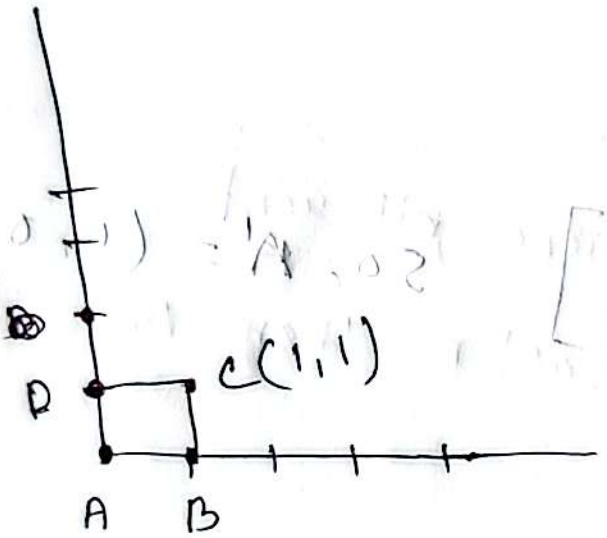
C(1, 1)

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \quad \text{So, } C' = (3, 4)$$

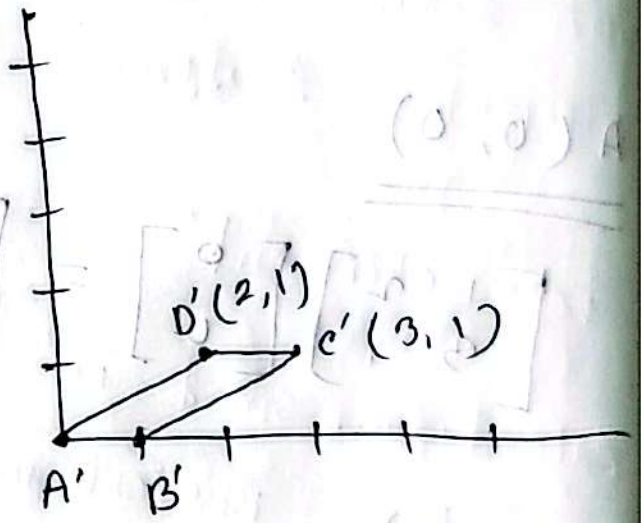
D(0, 1)

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \text{So, } D' = (2, 1)$$

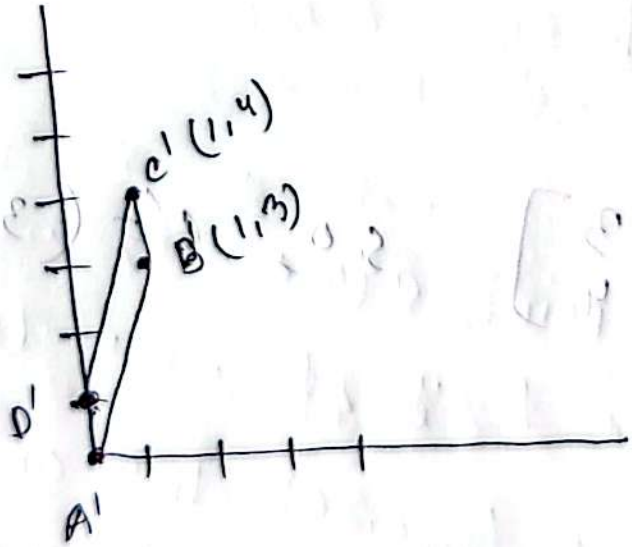
Thus, the transformed coordinates of the square are A'(0,0), B'(1,3), C'(3,4) and D'(2,1).



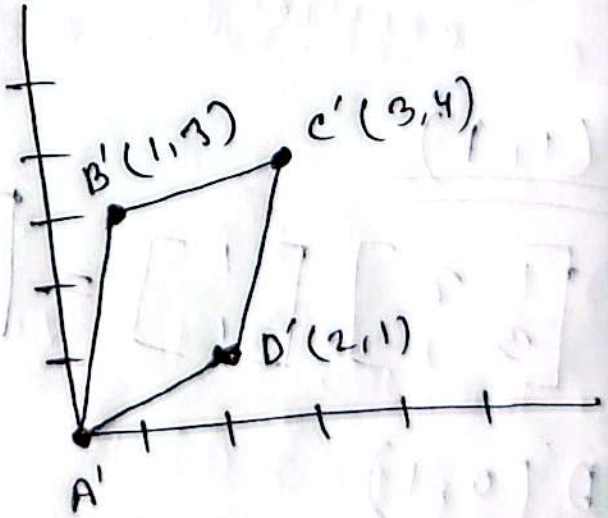
(a) Original square



(b) Shearing in the X direction



(c) Shearing in the Y directions



(d) Shearing in both directions

3(c)

The maximum number of objects that can be presented by the Z-buffer algorithm depends on the precision of the depth buffer, which is usually represented using a fixed point.

For example:

If the depth buffer uses 16 bits to represent depth values, this means that the Z-buffer algorithm can handle up to 65,536 different objects without depth precision issues.