Study computer graphics

Computer Graphics

Image file formats:

The Image file formats are standardized means of Organizing and storing digital Images. Image files are composed of digital data in one of these formats that can be rasterized for use on a computer display or printer. It may store data in Uncompressed, Compressed or Vector formats. Once rasterized, an Image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.

Therefore the size of an Image is determined by the number of pixels plus the colour depth. These formats include, Jpgs, jpeg, Gif, Webp, SVG etc

(1) Image file sizes:

The size of raster Image files is positively Correlated with the number of pixels in the Image and the colour depth (bits per pixel). Images can be compressed in various ways however. A Compression Algorithm stores either an exact representation or an approximation of the Original Image in a smaller number of bytes that can be

LZW algorithm is used for lossless storage.

JPEG: Joint photographic Expert group. is lossy compression Method.

Expanded back to its Uncompressed form with a Corresponding decompression algorithm.

Images are often too large to fit into devices, therefore they have to be compressed and at this we make use of 2 Algorithms.

+ Lossy Compression: This reduces the Image size by cutting off some part of the details of the Image therefore reducing the sharpness of the Image. It reduces file size while preserving a perfect Copy of the Original uncompressed Image.

+ Lossless Compression: This compresses the Image without reducing or affecting the details and sharpness of the Image. This preserves a representation of the Original uncompressed Image that may appear to be a perfect Copy but is not a perfect Copy.

To calculate the size of the Image, multiply the pixels of the Image by the Colour depth of the Image ~~to get~~ and approximate to the nearest byte. eg 600 by 1200 Image which has a colour depth of 24 would be 600 x 1200 x 24 = 17,280,000 ⇒ 19.3Mb

Major graphics picture formats:

Pixels are the picture elements that make up a picture.

Images are of 2 types
- Raster
- Vector graphics

* Raster Image file formats are Images created using pixels. This is also refferred to as Bitmap. eg JPEG, JPG
* Vector Image file formats are ~~not~~ Images created using lines and are not made up of pixels.

Note: Metafile: These are formats that use portable formats which can include both raster and Vector Information.

Page Description language: This refers to formats used to describe the layout of a printed page containing text, Objects and Images. Eg. postscript, PDF and PCL

Image formats:

1. JPEG (Joint photographic Expert group): This supports the 8bits grayscale Images and 24-bit Color Images. The JPEG compressed Images are stored in JFIF (JPEG file Interchange format). it applies the lossy

Compression of images which can result to a significant reduction of the file size. However JPEG files suffers generational degredation when repeatedly edited and saved.

(2) JPEG 2000: This compression standard enables both the lossy and lossless storage. This however improves quality and compression ratios, but however requires more computational power to process. It adds features that are missing in the JPEG and is not as common as JPEG but is used for professional movie editing

(3) Exif (Exchangeable Image format): It is incorporated into JPEG writing software used in most cameras. Its purpose is to record and to standardize the exchange of images with image metadata between digital cameras and editing and viewing software.

(4) Tiff (Tagged Image file format): This is a flexible format that normally saves eight bits or sixteen bits

per color (red, green, blue) for 24-bits and 48-bits totals, respectively, usually using either the TIFF or TIF filename extension. This file format can be lossy or lossless depending on the technique chosen for storing the pixel data. This file format is not widely supported by web browsers but it remains ~~the~~ widely accepted as a photograph file standard in the printing business because it can handle device-specific color spaces.

(5) GIF (Graphics Interchange format): This is in normal use limited to an 8-bit palette, or 256 colors (although 24-bit color depth is technically possible). It is most suitable for storing graphics with few colors, such as simple diagrams, shapes, ~~images~~ logos, and cartoon style images as it uses LZW lossless compression, which is more effective when large areas have a single color, and less effective for photographic or dithered images.

(7) BMP (Windows Bitmap): This handles graphic files within the Microsoft Windows OS. Typically

Note: In photography JPEG is chosen for forced distribution since its compression technique technically yields smaller file sizes.

BMP files are uncompressed, and therefore large and lossless, their advantage is their simple structure and wide acceptance in windows programs.

(8) PNG (Portable Network Graphics): this was created as a free, Open-Source alternative to GIF. It supports eight-bit paletted images (with optional transparency for all palleted ~~images~~ colors) and 24-bit truecolor (16 million colours) or 48-bit truecolor with and without alpha channel - while GIF supports only 256 colours and a single transparent color. It uses the lossless compression technique.

Aside: LZW (Lempel Ziv Welch) is a simple basic form of file compression that squeezes out inefficiencies in data storage without loosing data or distorting the image. It is the best at compressing images with large fields of homogenous color, such as logos and diagrams.

10/2023 THREE DIMENSIONAL TRANSFORMATIONS:

In many application, changes in orientation, size,

and shape are accomplished with generic transformation that alter the coordinate description of objects. The basic geometric transformation are as follows.

- Translation
- Rotation
- Scaling.

While other transformations include the following:

- Reflection
- Shear

① Translation:

We translate a 2D point by adding translation distances, t_x and t_y, to the original coordinate position (x, y).

$$x' = x + t_x \text{ and } y' = y + t_y.$$

Alternatively, translation can also be specified by the following transformation matrix.

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore the above could be rewritten as follows

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

Example 1: To translate a triangle with vertices at original coordinates (10, 20), (10, 10), (20, 10) by $t_x = 5$, $t_y = 10$, we compute the values as follows.

1st coordinate Translation of vertex (10, 20)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\times10 + 0\times20 + 5\times1 \\ 0\times10 + 1\times20 + 10\times1 \\ 0\times10 + 0\times20 + 1\times1 \end{bmatrix} = \begin{bmatrix} 15 \\ 30 \\ 1 \end{bmatrix}$$

2nd coordinate Translation of vertex (10, 10)

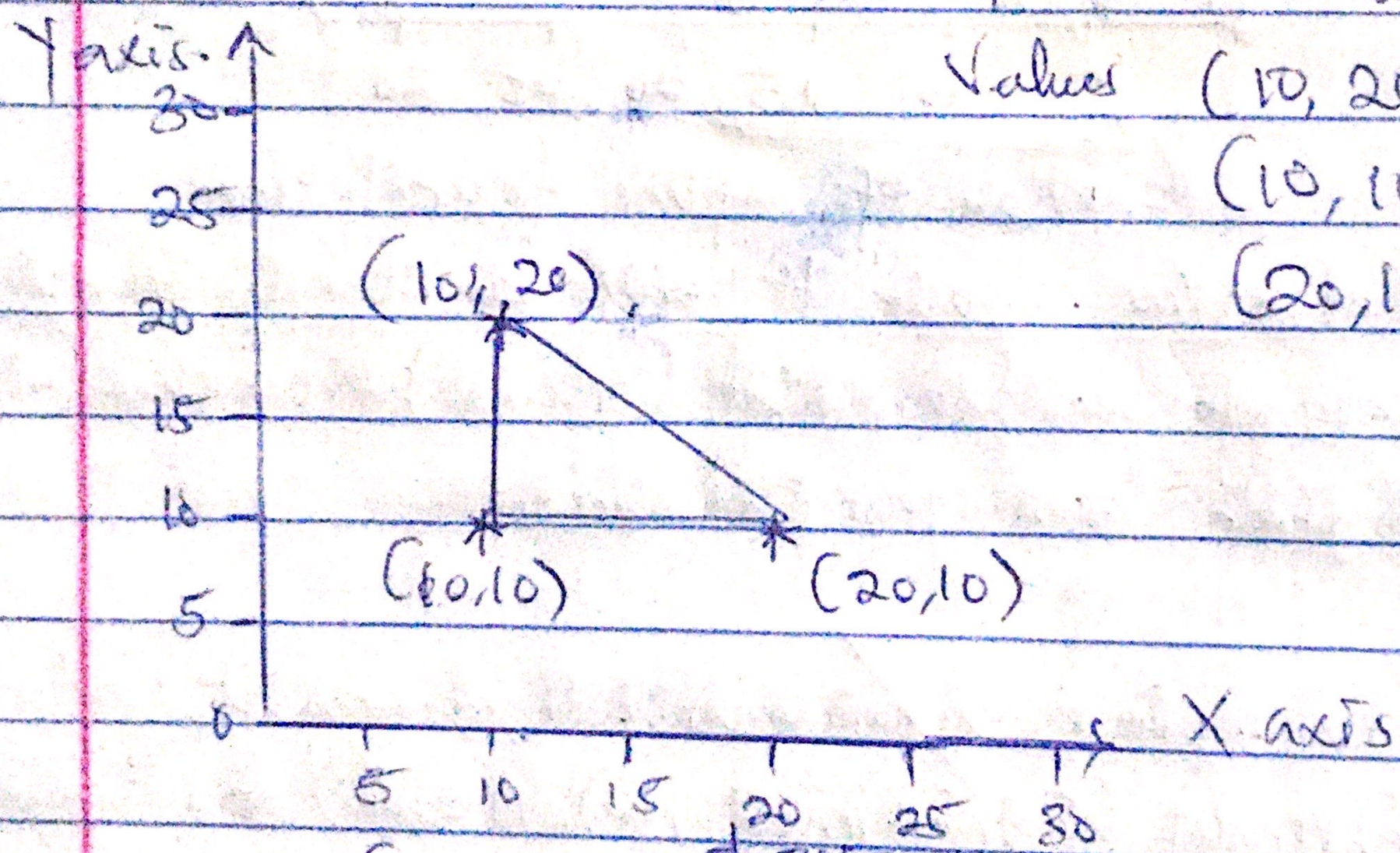
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\times10 + 0\times10 + 5\times1 \\ 0\times10 + 1\times10 + 10\times1 \\ 0\times10 + 0\times10 + 1\times1 \end{bmatrix} = \begin{bmatrix} 15 \\ 20 \\ 1 \end{bmatrix}$$

d coordinate Translation of vertex (20, 10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\times20 + 0\times10 + 5\times1 \\ 0\times20 + 1\times10 + 10\times1 \\ 0\times20 + 0\times10 + 1\times1 \end{bmatrix} = \begin{bmatrix} 25 \\ 20 \\ 1 \end{bmatrix}$$

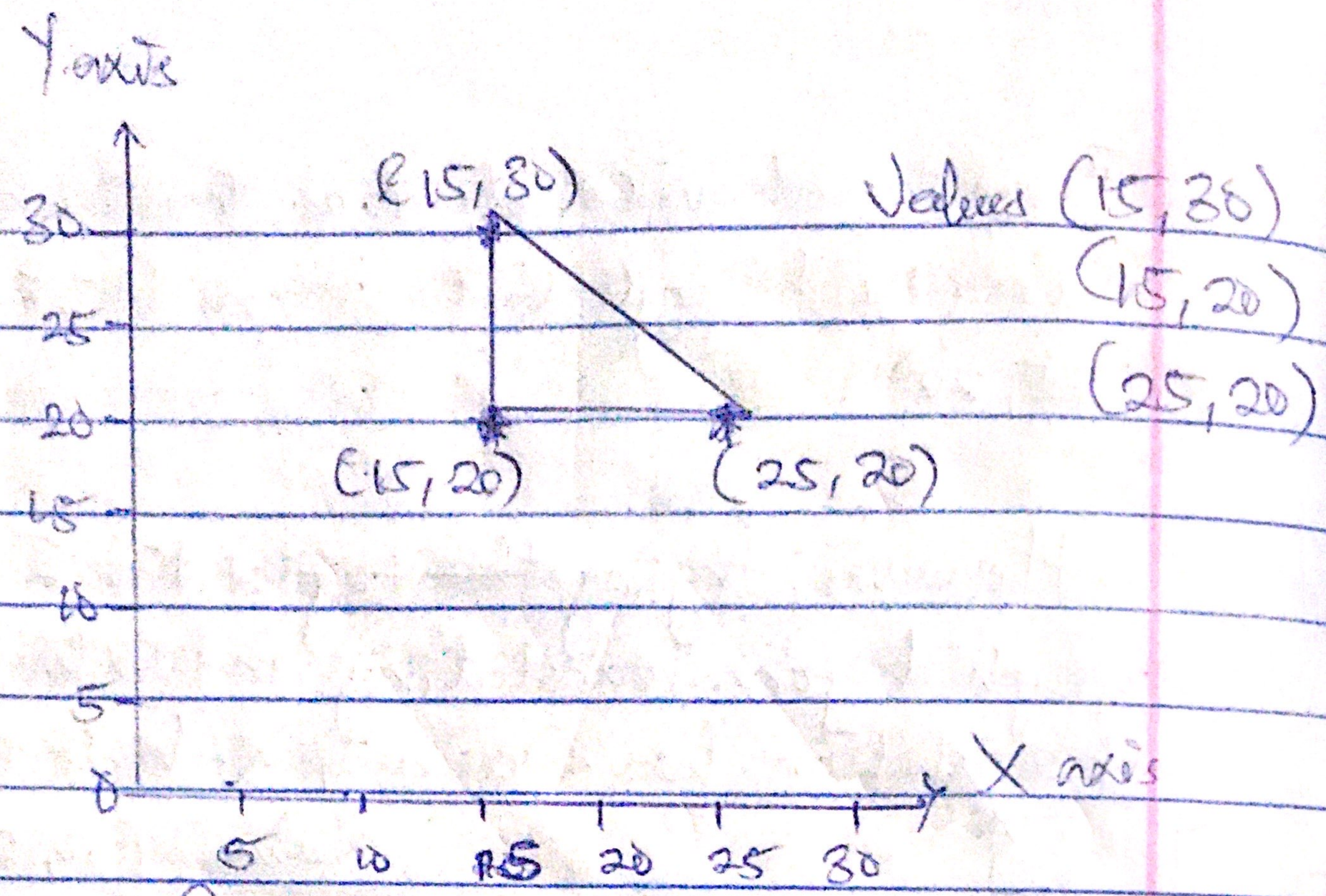
From the above calculations, the resultant coordinates of the triangle vertexes are (15, 30), (15, 20), (25, 20).

However, trying ~~the~~ to plot the 2 graphs, the graph before translating and after the translation has occured, we have the following values (10, 20) (10, 10) (20, 10)



Graph before translation

The graph above was ~~ts~~ plotted using the initial values (vertexes) of x and y to show the initial state of the triangle before translation. The graph is plotted by representing the points that were given in the coordinates where the first value is for x and the second value is for y for each coordinate.



Graph of after translation

From the above, it could be clearly seen that the triangle has translated (moved) from the initial position to a new position.

Exercise 2: Translate a triangle with vertices at Original Cordinates (20, 20), (20, 10), (10, 10) by $t_x = 0$, $t_y = 0$. Roughly plot the Original and resultant triangles.

Soln

1st Cordinate Translation of Vertex (20, 20)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

Example 2: Translate a triangle with vertices at original cordinates (10, 25), (5, 10), (20, 10) by $t_x = 15$, $t_y = 5$. Roughly plot the original and resultant triangles.

Soln

1st cordinate Translation of vertex (10, 25)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 15 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 25 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\times10 + 0\times25 + 15\times1 \\ 0\times10 + 1\times25 + 5\times1 \\ 0\times10 + 0\times25 + 1\times1 \end{bmatrix} = \begin{bmatrix} 25 \\ 30 \\ 1 \end{bmatrix}$$

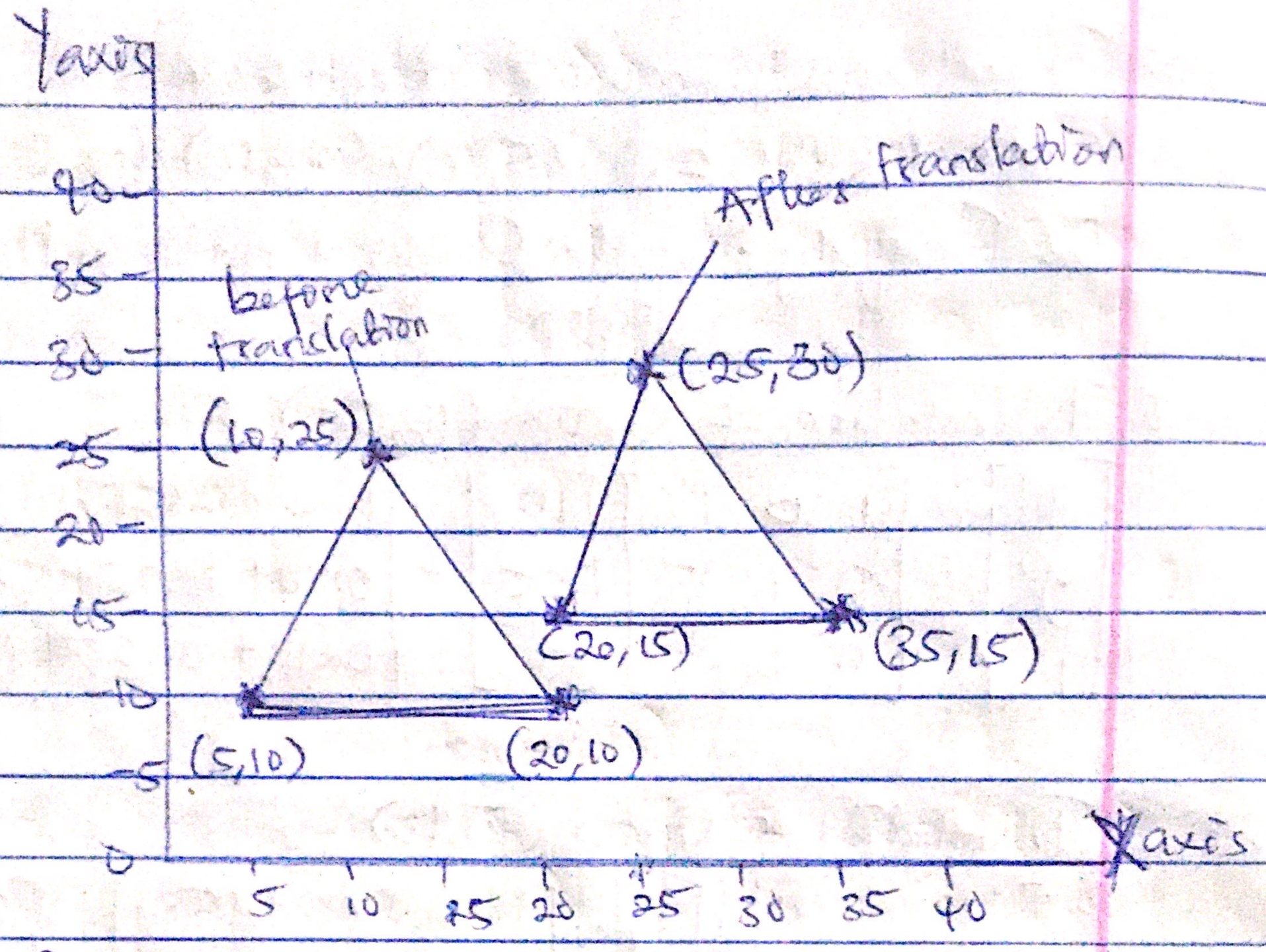
2nd cordinate Translation of vertex (5, 10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 15 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\times5 + 0\times10 + 15\times1 \\ 0\times5 + 1\times10 + 5\times1 \\ 0\times5 + 0\times10 + 1\times1 \end{bmatrix} = \begin{bmatrix} 20 \\ 15 \\ 1 \end{bmatrix}$$

3rd cordinate Translation of vertex (5, 10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 15 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\times20 + 0\times10 + 15\times1 \\ 0\times20 + 1\times10 + 5\times1 \\ 0\times20 + 0\times10 + 1\times1 \end{bmatrix} = \begin{bmatrix} 35 \\ 15 \\ 1 \end{bmatrix}$$

~~Transfer~~ Therefore the triangle has moved (translated) from the cordinates (10, 25), (5, 10), (20, 10) to (25, 30), (20, 5), (35, 15).



Graph Showing before & After translation.

ROTATION ABOUT THE ORIGIN:

To rotate an Object about the Origin (0,0), we Specify the rotation angle θ. Positive and negative values for the rotation angle define Counterclockwise and Clockwise rotations respectively. The following is the Computation of this rotation for a point

$$x' = x\cos\theta - y\sin\theta$$

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

Alternatively, this rotation can also be specified by

the following transformation matrix:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then we can rewrite the formular as

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

Example 1: To rotate a triangle about the Origin with Vertices at Origin Cordinates (10, 20), (10, 10), (20, 10) by 30 degrees we compute ~~the~~ as follows:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Cordinates Rotation of Vertex (10, 20).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866\times 10 + (-0.5)\times 20 + 0\times 1 \\ 0.5\times 10 + 0.866\times 20 + 0\times 1 \\ 0\times 10 + 0\times 20 + 1\times 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1.34 \\ 22.32 \\ 1 \end{bmatrix}$$

2nd Coordinates Rotation of Vertex (10, 10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 \times 10 + (-0.5) \times 10 + 0 \times 1 \\ 0.5 \times 10 + 0.866 \times 10 + 0 \times 1 \\ 0 \times 10 + 0 \times 10 + 1 \times 1 \end{bmatrix}$$

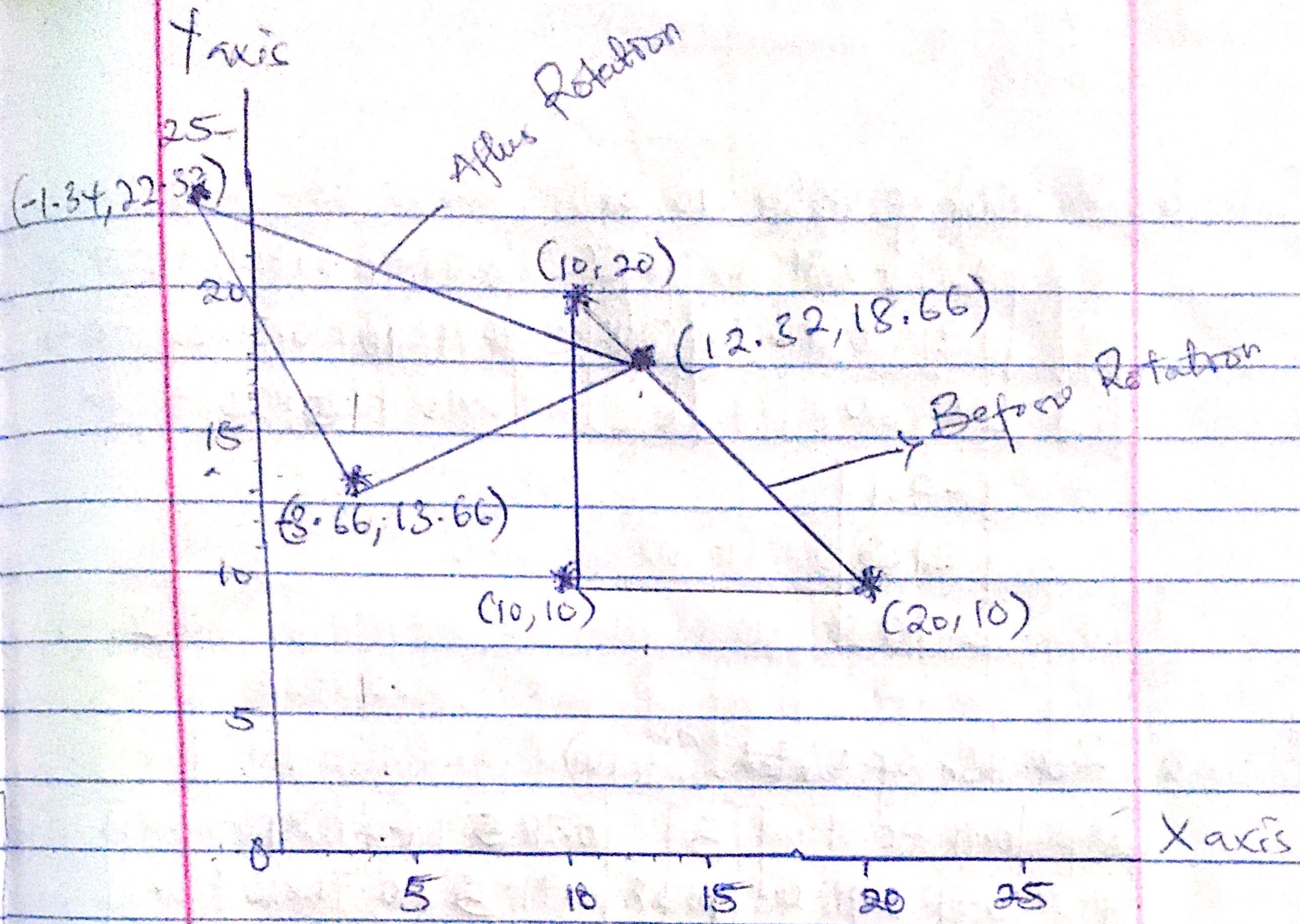
$$= \begin{bmatrix} 3.66 \\ 13.66 \\ 1 \end{bmatrix}$$

3rd coordinates Rotation of Vertex (20, 10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 \times 20 + (-0.5) \times 10 + 0 \times 1 \\ 0.5 \times 20 + 0.866 \times 10 + 0 \times 1 \\ 0 \times 20 + 0 \times 10 + 1 \times 1 \end{bmatrix}$$

$$= \begin{bmatrix} 12.32 \\ 18.66 \\ 1 \end{bmatrix}$$

From the above calculations, the triangle has been rotated from the initial position to a new position indicated by the change in the values of the coordinates indicating the new position of the new triangle.



The graph showing before & Aflus Rotation.

Example: Rotate a triangle with Vertices at Original Cordinates (10,20), (5,10), (20,10) by 45 degrees. Roughly plot the Original and Resultant triangles.

Soln

Using the formulas

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

$$= \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.71 & -0.71 & 0 \\ 0.71 & 0.71 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1st Coordinates Rotation of Vertex (10, 20)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.71 & -0.71 & 0 \\ 0.71 & 0.71 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.71\times10 + (-0.71)\times20 + 0\times1 \\ 0.71\times10 + 0.71\times20 + 0\times1 \\ 0\times10 + 0\times20 + 1\times1 \end{bmatrix}$$

$$= \begin{bmatrix} -7.1 \\ 21.3 \\ 1 \end{bmatrix}$$

2nd Coordinates Rotation of Vertex (5, 10)

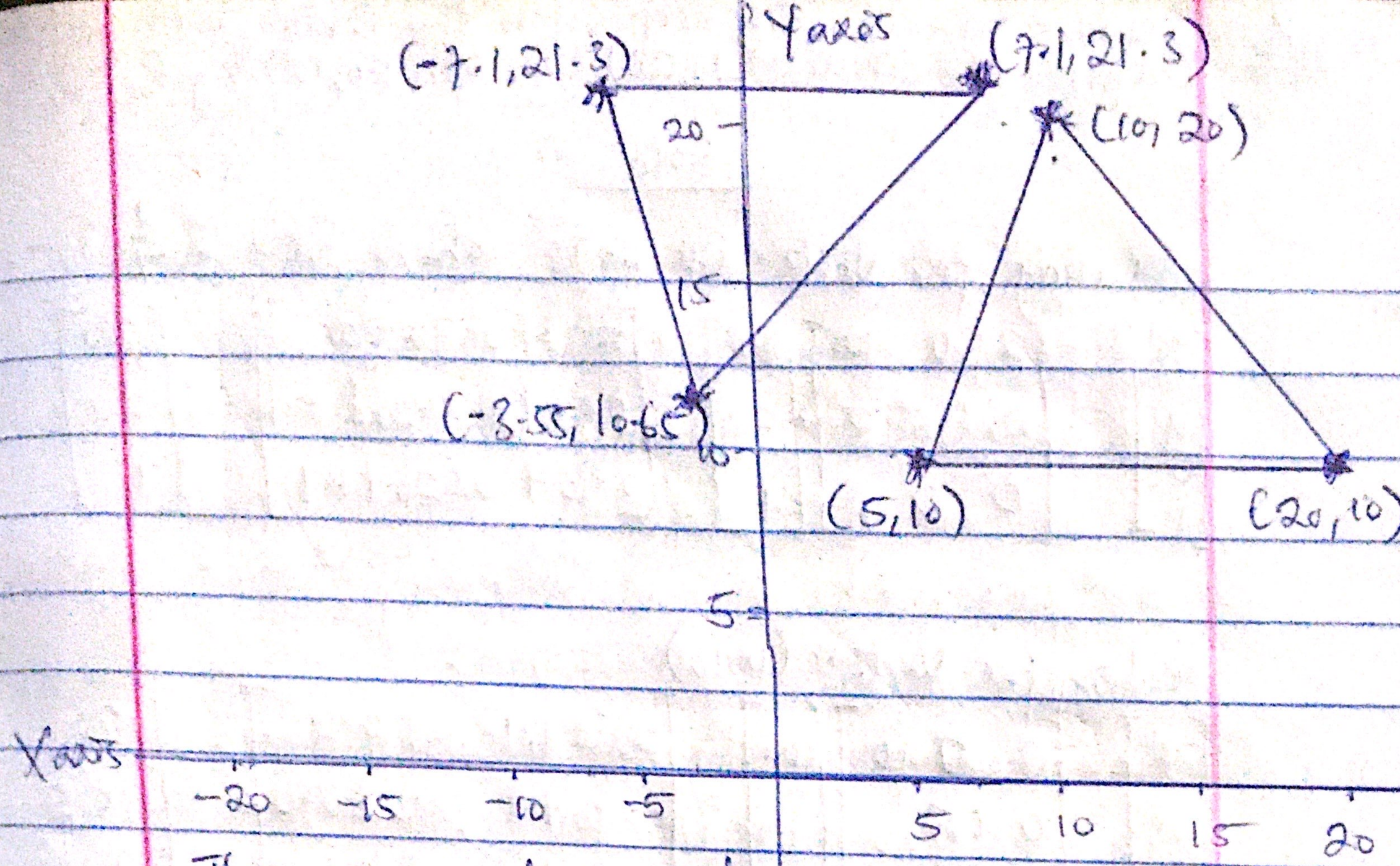
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.71 & -0.71 & 0 \\ 0.71 & 0.71 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.71\times5 + (-0.71)\times10 + 0\times1 \\ 0.71\times5 + 0.71\times10 + 0\times1 \\ 0\times5 + 0\times10 + 1\times1 \end{bmatrix}$$

$$= \begin{bmatrix} -3.55 \\ 10.65 \\ 1 \end{bmatrix}$$

3rd Coordinates Rotation of Vertex (20, 10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.71 & -0.71 & 0 \\ 0.71 & 0.71 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.71\times20 + (-0.71\times10) + 0\times1 \\ 0.71\times20 + 0.71\times10 + 0\times1 \\ 0\times20 + 10\times0 + 1\times1 \end{bmatrix}$$

$$= \begin{bmatrix} 7.1 \\ 21.3 \\ 1 \end{bmatrix}$$



The graph of before & after Rotation by 45°

SCALLING WITH RESPECT TO THE ORIGIN

We scale a 2D object with respect to the Origin by setting the scalling factors, S_x and S_y which are multiplied to the Original vertex cordinate positions (x, y)

$$X' = X * S_x \text{ and } Y' = Y * S_y.$$

Alternatively, this scalling can also be specified by the following transformation matrix

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus we rewrite the formulas as follows

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $S_x =$

Example: Scale a triangle with respect to the origin, with vertices at original coordinates (10,20), (10,10), (20,10) by $S_x = 2$, $S_y = 1.5$

Soln

Scaling for Vertex (10, 20); where $S_x = 2$ & $S_y = 1.5$

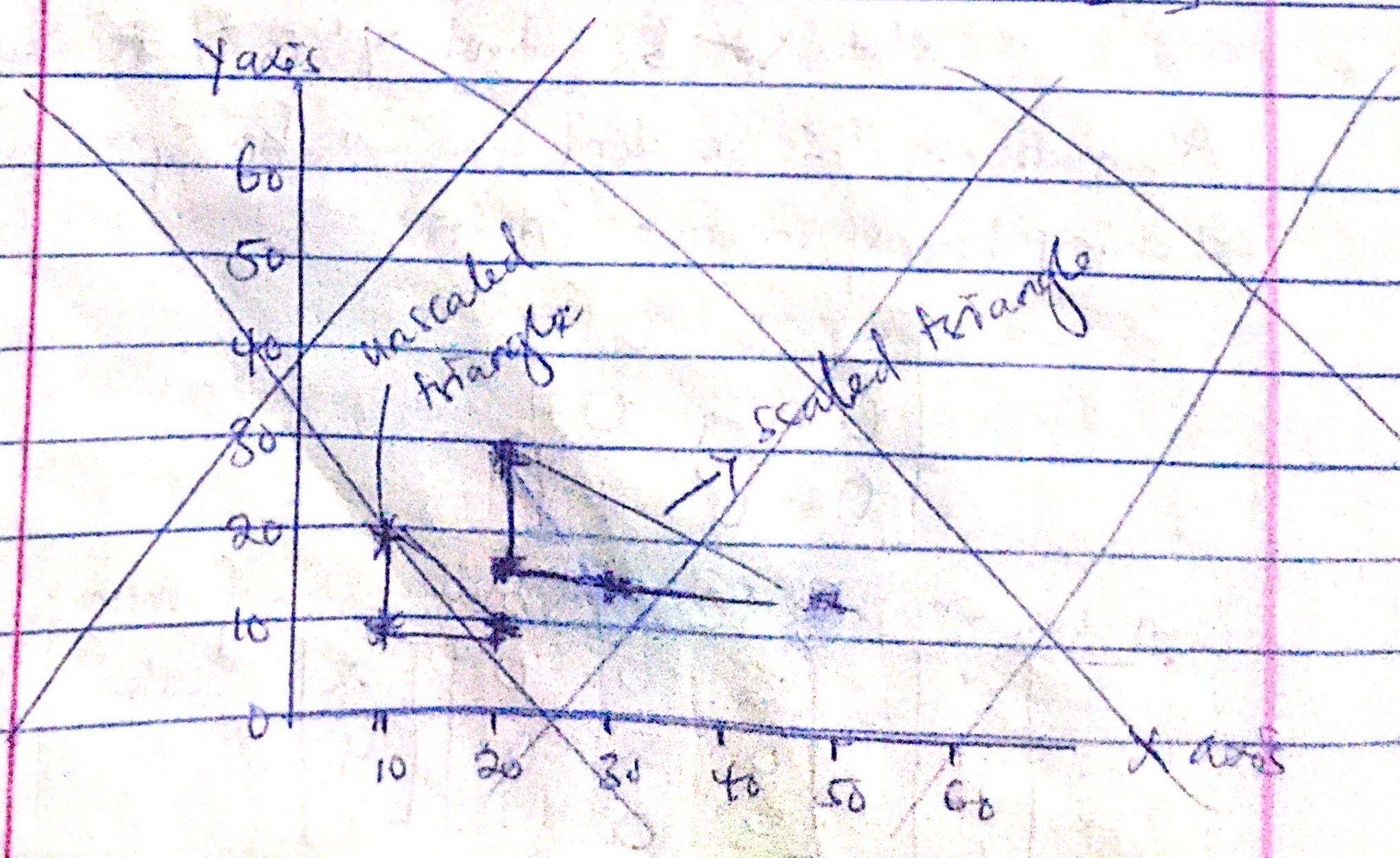
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 2\times10 + 0\times20 + 0\times1 \\ 0\times10 + 1.5\times20 + 0\times1 \\ 0\times10 + 0\times20 + 1\times1 \end{bmatrix} = \begin{bmatrix} 20 \\ 30 \\ 1 \end{bmatrix}$$

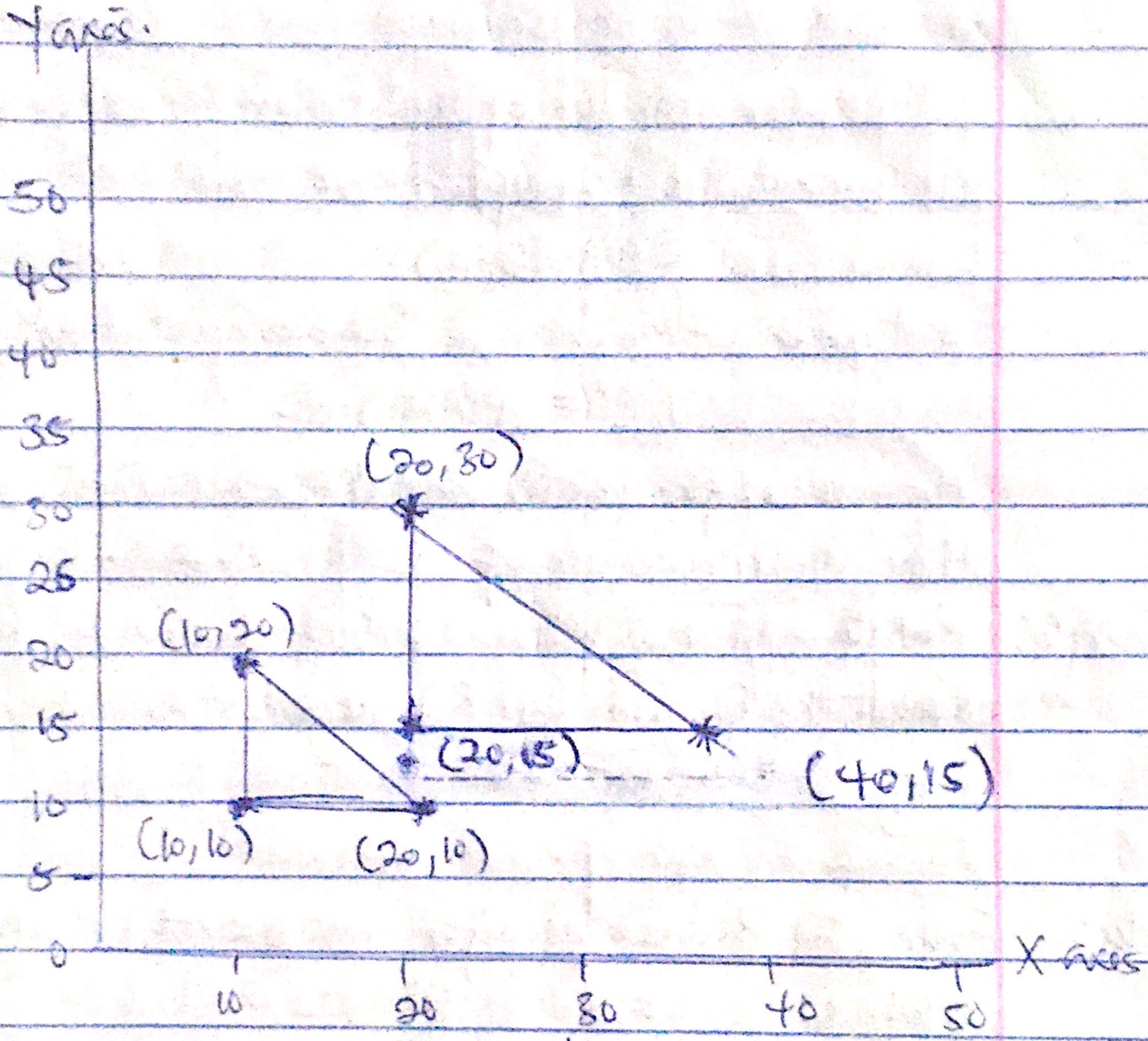
Scaling for Vertex (10,10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 2\times10 + 1\times10 + 0\times0 \\ 0\times10 + 1.5\times10 + 0\times0 \\ 0\times10 + 1\times0 + 1\times1 \end{bmatrix} = \begin{bmatrix} 20 \\ 15 \\ 1 \end{bmatrix}$$

Scaling for Vertex (20,10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 40 + 0 + 0 \\ 0 + 15 + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 40 \\ 15 \\ 1 \end{bmatrix}$$





Graph Showing Before & and After Scaling

CONCATENATION PROPERTIES OF COMPOSITE MATRIX:

(1) MATRIX MULTIPLICATION IS ASSOCIATIVE

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

Therefore we can evaluate matrix product using these associative groupings. For Example, we have a triangle. We want to rotate it with the matrix B, then we

Note. ~~from the Multiplied~~

translate it with matrix A.

Thus, for a vector of that triangle represented as C. We compute its transformation as

$$C = A \cdot (B \cdot C)$$

But we can also change the computation method as

$$C'' = (A \cdot B) \cdot C$$

However, for faster and more efficient computing time we make use of $C'' = (A \cdot B) \cdot C$

Example: Rotate a triangle with vertices (10, 20), (10, 10), (20, 10) about the origin by 30° degrees and then translate it by $t_x = 5$, $t_y = 10$.

Note: From the Rules above for concatenation of composite Matrix,

A = Scale factor (translation value of t_x & t_y)
B = Location (Rotation)
C = Coordinates
C'' = The Object Represented.

Soln

Compute the rotation Matrix

$$B = \begin{bmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compute the translation Matrix

$$A = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$

Then we Compute the values of $M = A \cdot B$

$$M = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1\times 0.866 + 0\times 0.5 + 5\times 0 & 1\times -0.5 + 0\times 0.866 + 5\times 0 & 1\times 0 + 0\times 0 + 5\times 1 \\ 0\times 0.866 + 1\times 0.5 + 10\times 0 & 0\times -0.5 + 1\times 0.866 + 10\times 0 & 0\times 0 + 1\times 0 + 10\times 1 \\ 0\times 0.866 + 0\times 0.5 + 1\times 0 & 0\times -0.5 + 0\times 0.866 + 1\times 0 & 0\times 0 + 0\times 0 + 1\times 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0.866 & -0.5 & 5 \\ 0.5 & 0.866 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$

Note: The three by 3 matrix is multiplied by going through the Matrix A horizontally and multiplying the respective values with the vertical values of matrix B.

Then we Compute the transformation of the 3 vertices
Transformation of the vertex (10, 20)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 5 \\ 0.5 & 0.866 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 \times 10 + (-0.5) \times 20 + 5 \times 1 \\ 0.5 \times 10 + 0.866 \times 20 + 10 \times 1 \\ 0 \times 10 + 0 \times 20 + 1 \times 1 \end{bmatrix}$$

$$= \begin{bmatrix} 3.66 \\ 32.32 \\ 1 \end{bmatrix}$$

Transformation of Vertex (10,10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 5 \\ 0.5 & 0.866 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 8.66 + 5 + 5 \\ 5 + 8.66 + 10 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 18.66 \\ 23.66 \\ 1 \end{bmatrix}$$

Transformation of Vertex (20,10)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 5 \\ 0.5 & 0.866 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 17.32 + 5 + 5 \\ 10 + 8.66 + 10 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 27.32 \\ 28.66 \\ 1 \end{bmatrix}$$

Note: If you are asked (the question says) to rotate before you translate, then your implementation would be to translate before rotating and you do this by using the formular

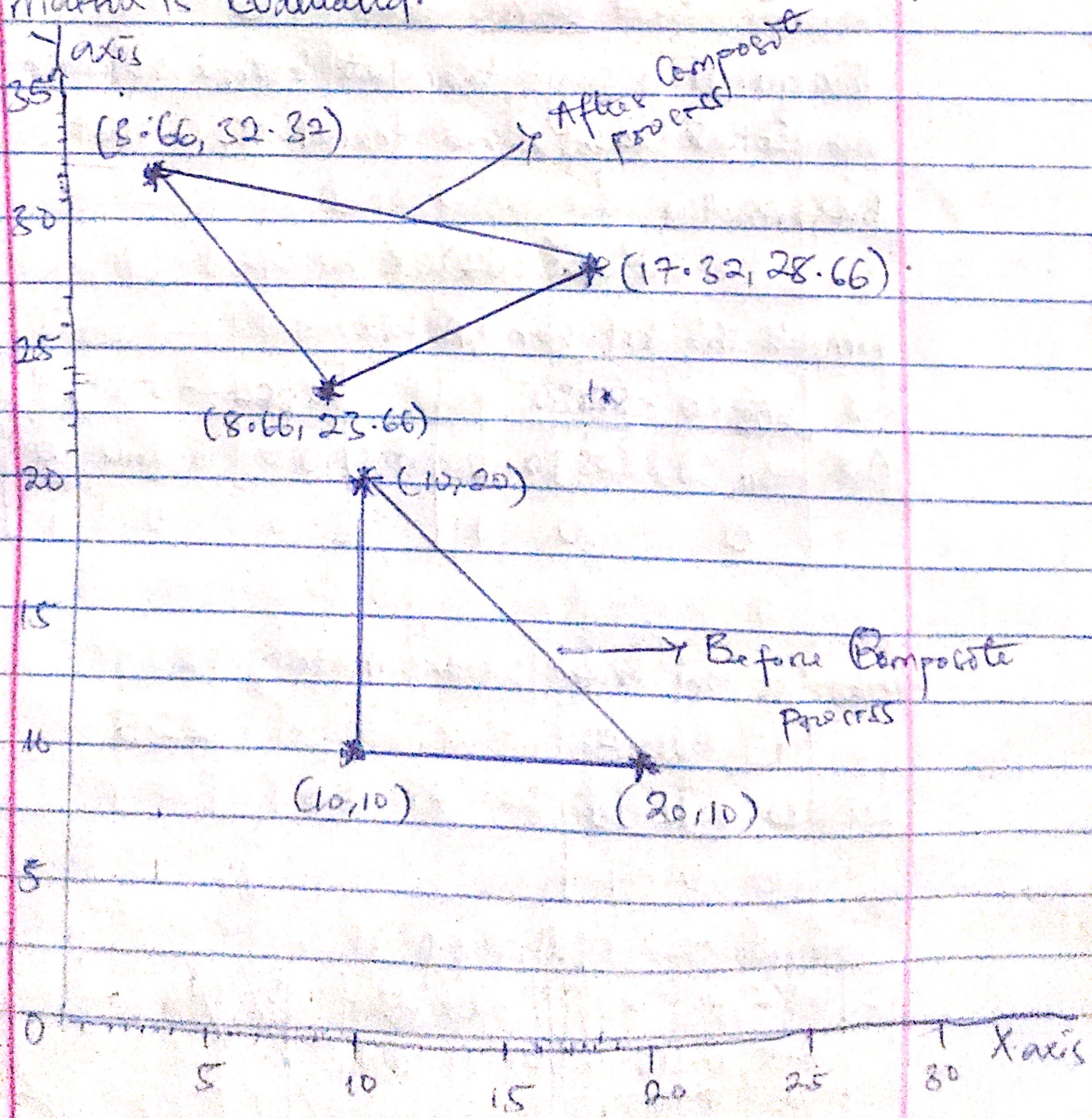
$$C' = (B \cdot A) \cdot C$$

But if you are asked to translate and then rotate

The object, you will have to implement solution before translation and then makes use of the formulas

$$C' = (A \cdot B) \cdot C$$

However, it is also important to note that $A \cdot B$ may not be equal to $B \cdot A$ therefore we must be careful about the order in which the composite matrix is evaluated.



Note: In Exam Scenario's, plot the graphs differently to show the before and after.

Assignment:

Suppose you given rectangle at vertexes (10,10), (10,15) (15, 10) and (15, 20) find the new position & In translation about (5,10). Rotation about 30° and Scaling at $S_x = 2$, $S_y = \cancel{18}\ 1.5$ all about the Origin. Plot the graph of the result.

Soln

The question says translate before rotating therefore we rotate before we translate, In that case we use the formular

$$C' = (A \cdot B) \cdot C$$

Compute the rotation Matrix

$$A = \begin{bmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compute the translation Matrix

$$B = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$

Compute the Matrix of M

$$M \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} = \cancel{\begin{bmatrix} 0.866 \end{bmatrix}}$$

$$\begin{bmatrix} 0.866+0+0 & 0+(-0.5)+0 & 4.33+5+0 \\ 0.5+0+0 & 0+0.866+0 & 2.5+8.66+0 \\ 0+0+0 & 0+0+0 & 0+0+1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0.866 & -0.5 & -0.67 \\ 0.5 & 0.866 & 11.16 \\ 0 & 0 & 1 \end{bmatrix}$$

TRANSLATION

Compute the translation for the vertexes (10,10)

$$\begin{bmatrix} 0.866 & -0.5 & -0.67 \\ 0.5 & 0.866 & 11.16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 8.66+5-0.67 \\ 5+8.66+11.16 \\ 0+0+1 \end{bmatrix} = \begin{bmatrix} 2.99 \\ 24.82 \\ 1 \end{bmatrix}$$

Compute the translation for the vertexes (10, 15)

$$\begin{bmatrix} 0.866 & -0.5 & -0.67 \\ 0.5 & 0.866 & 11.16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 15 \\ 1 \end{bmatrix} = \begin{bmatrix} 8.66-7.5-0.67 \\ 5+12.99+11.16 \\ 0+0+1 \end{bmatrix} = \begin{bmatrix} 0.49 \\ 29.15 \\ 1 \end{bmatrix}$$

Compute the translation for the vertexes (15, 10)

$$\begin{bmatrix} 0.866 & -0.5 & -0.67 \\ 0.5 & 0.866 & 11.16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 15 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 12.99-5-0.67 \\ 7.5+8.66+11.16 \\ 0+0+1 \end{bmatrix} = \begin{bmatrix} 7.32 \\ 27.32 \\ 1 \end{bmatrix}$$

Note: There is no point scalling after finding the coordinates, ~~fi~~ scale before finding the coordinates. So you have 2 graphs alone

Compute the translation for the Vertex (15, 20)

$$\begin{bmatrix} 0.866 & -0.5 & -0.67 \\ 0.5 & 0.866 & 11.16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 15 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 12.99 + 10 - 0.67 \\ 7.5 + 17.32 + 11.16 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 2.32 \\ 35.98 \\ 1 \end{bmatrix}$$

SCALING OF THE ACHIEVED RESULT

where $S_x = 2$ and $S_y = 1.5$

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compute the ~~translation~~ scalling for the Vertex (10, 10)

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \times 10 + 0 + 0 \\ 0 + 15 + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 20 \\ 15 \\ 1 \end{bmatrix}$$

Compute the scalling for the Vertex (10, 15)

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 15 \\ 1 \end{bmatrix} = \begin{bmatrix} 20 + 0 + 0 \\ 0 + \cancel{22.5}\ 22.5 + 0 \\ 0 + 0 + 0 \end{bmatrix} = \begin{bmatrix} 20 \\ 22.5 \\ 0 \end{bmatrix}$$

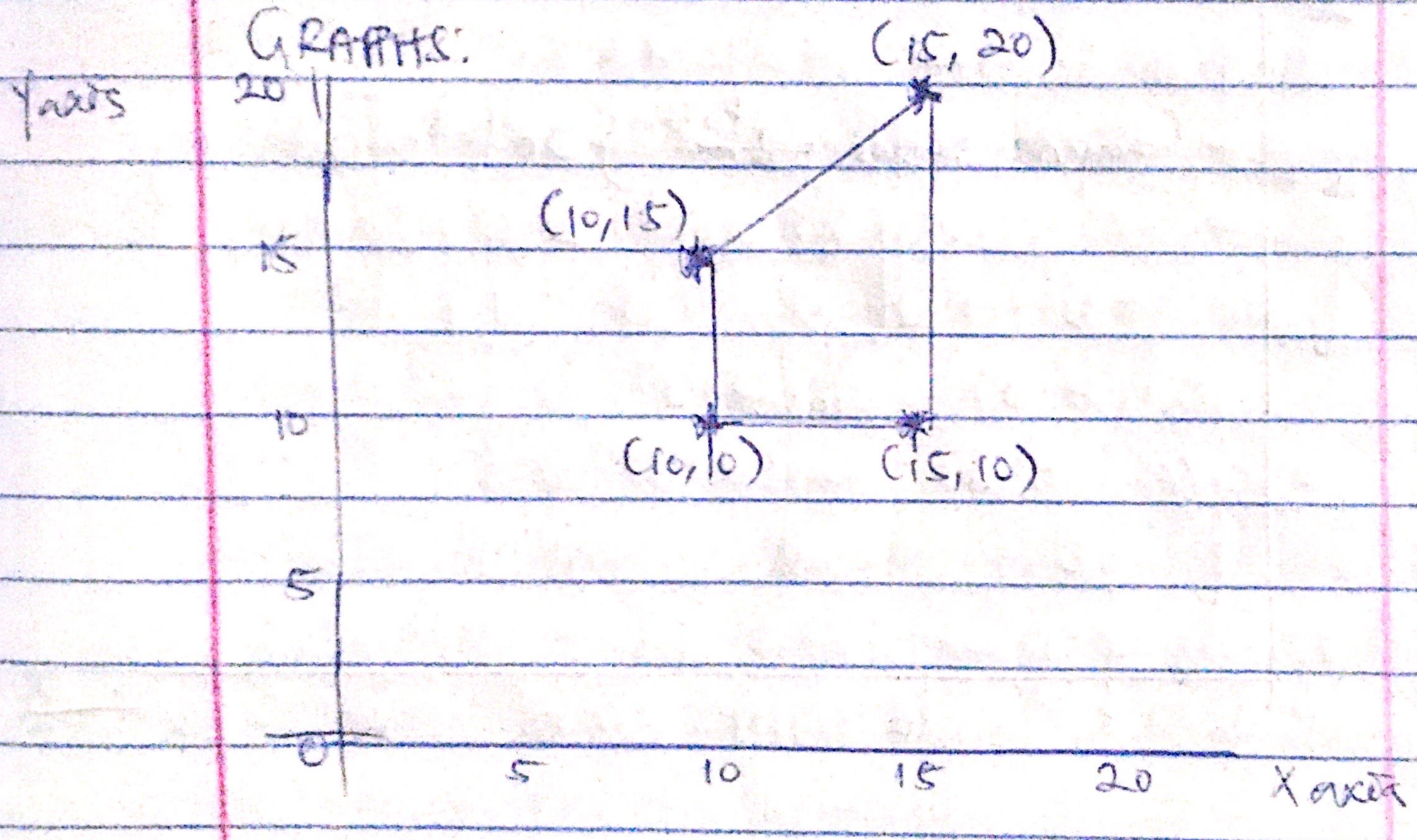
Compute the Scalling for the Vertex (15, 10)

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 15 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 30 + 0 + 0 \\ 0 + 15 + 0 \\ 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 30 \\ 15 \\ 1 \end{bmatrix}$$

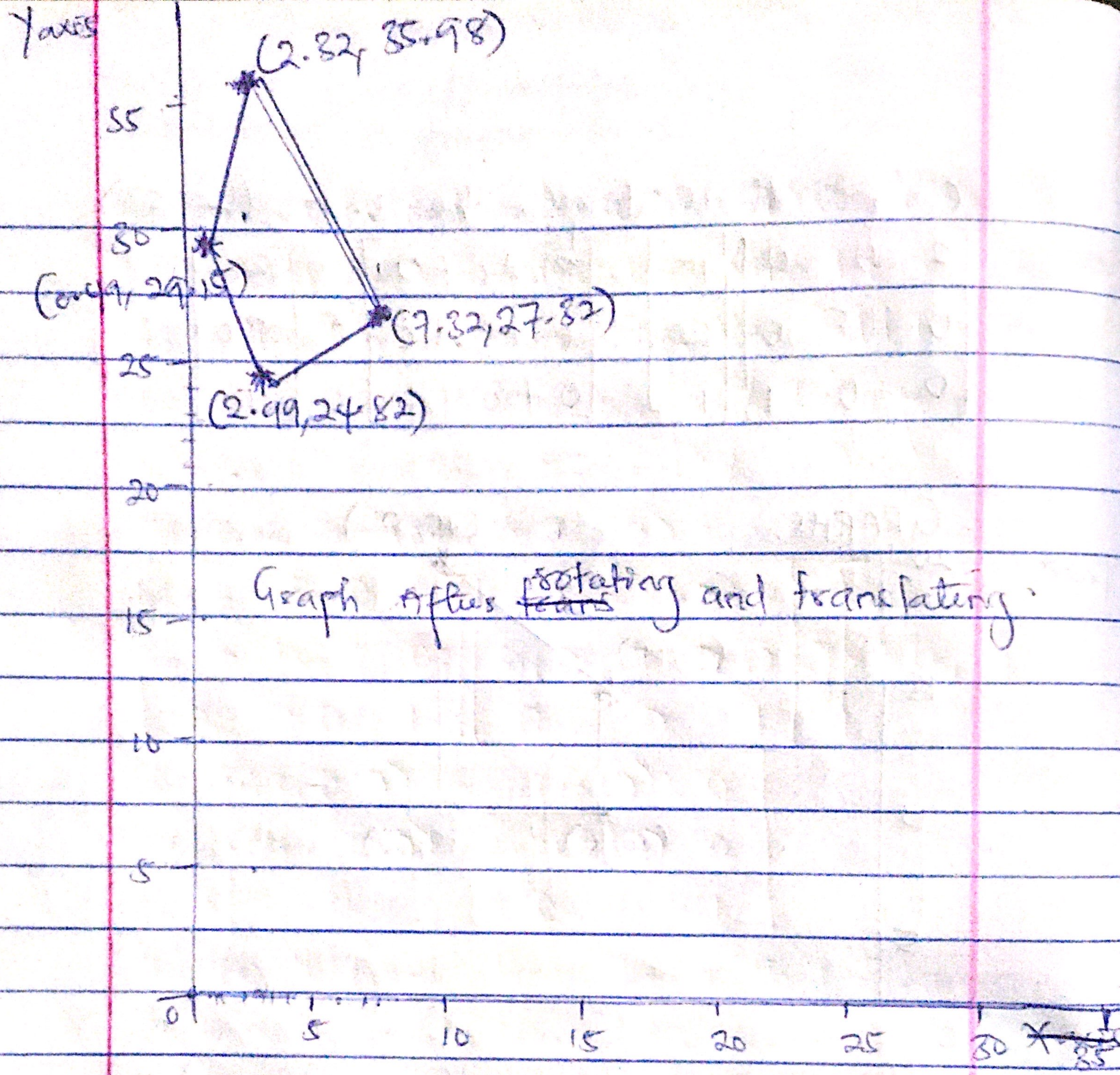
Compute the scaling for the vertex (15, 20)

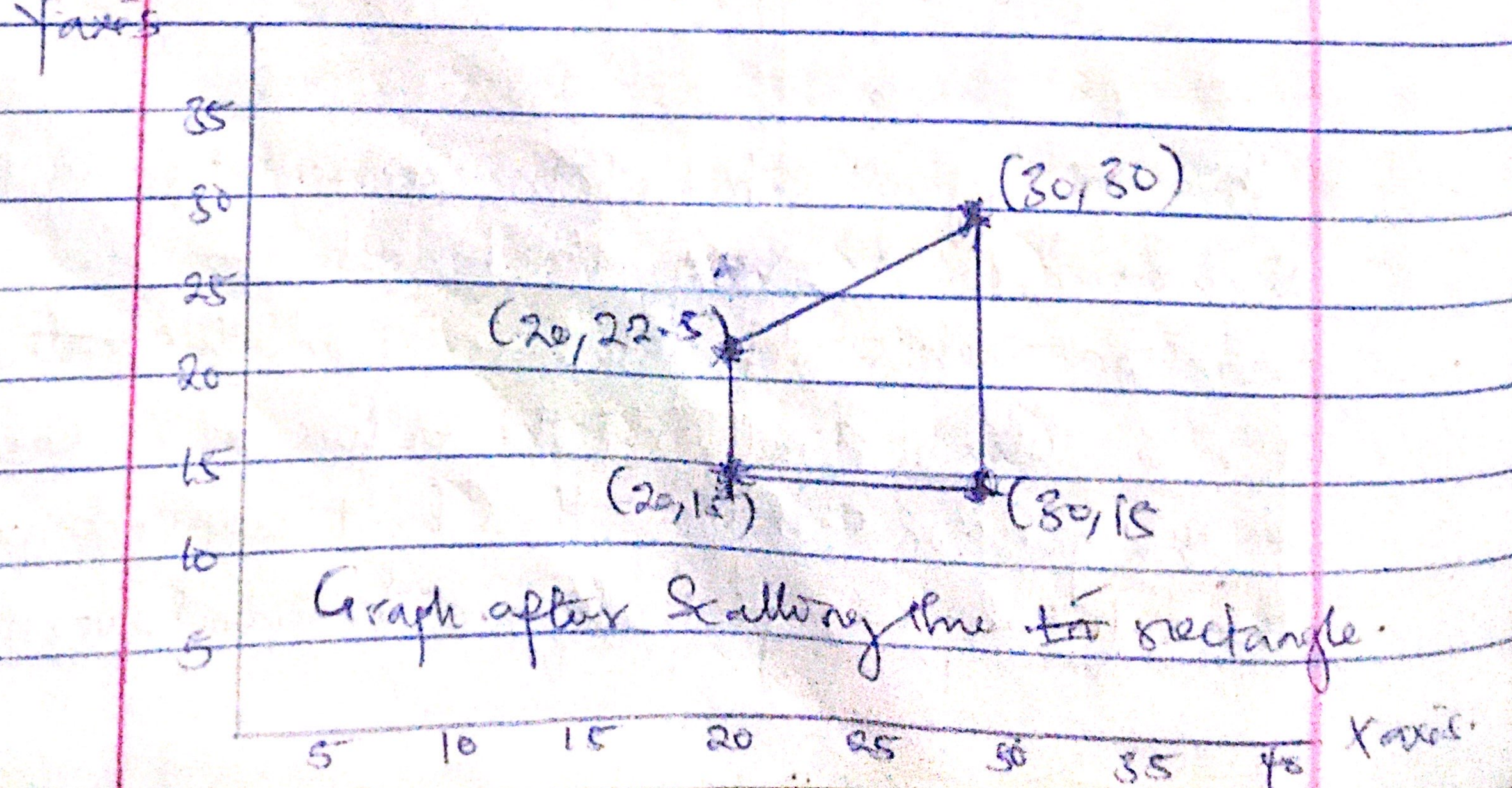
$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 15 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 30+0+0 \\ 0+30+0 \\ 0+0+1 \end{bmatrix} = \begin{bmatrix} 30 \\ 30 \\ 1 \end{bmatrix}$$

GRAPHS:



The above graph shows the representation of the vertexes before applying the translation and rotation techniques. However, from the ~~vati~~ graph we can clearly denote that the vertex asert correct as the question state that the object is a rectangle but the values depics a parallelogram. but we solves regardless


Y axis
(2.82, 35.98)
(7.32, 27.82)
(2.99, 24.82)
Graph after ~~rotars~~ rotating and translating.
X axis


Y axis
(30, 30)
(20, 22.5)
(20, 15)
(30, 15
Graph after scalling the ~~tr~~ rectangle.
X axis

Note: In 2D transformation, there is no point scaling after finding the coordinates of the translation, scale before translating or rotating to prevent going through the long method.

COMPOSITE TRANSFORMATION MATRIX

TRANSLATIONS

By common sense, if we translate a shape with 2 successive translation vectors (t_{x_1}, t_{y_1}) and (t_{x_2}, t_{y_2}) it is equal to a single translation of $(t_{x_1}+t_{x_2}, t_{y_1}+t_{y_2})$. The additive property can be demostrated by composite transformation matrix.

Example:

$$\begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} = \cancel{\begin{bmatrix} 1\times1+0\times0+t_{x_1}\times0 & & \\ & & \\ & & \end{bmatrix}}$$

$$= \begin{bmatrix} 1\times1+0\times0+t_{x_1}\times0 & 1\times0+0\times1+t_{x_1}\times0 & 1\times t_{x_2}+0\times t_{y_2}+t_{x_1}\times1 \\ 0\times1+1\times0+t_{y_1}\times0 & 0\times0+1\times1+t_{y_1}\times0 & 0\times t_{x_2}+1\times t_{y_2}+t_{y_1}\times1 \\ 0\times1+0\times0+1\times0 & 0\times0+0\times1+1\times0 & 0\times t_{x_2}+0\times t_{y_2}+1\times1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & t_{x_2}+t_{x_1} \\ 0 & 1 & t_{y_2}+t_{y_1} \\ 0 & 0 & 1 \end{bmatrix}$$

However, this indicates that 2 successive translations are additive.

ROTATIONS:

By common sence, if we rotate a shape with 2

Successive rotation angles θ and α, about the Origin, It is equal to rotating the shape Once by an angle $\theta + \alpha$ about the Origin. Similarly this additive property can be demonstrated by composite transformation matrix.

Mathematically

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = \cancel{\begin{bmatrix} \cos \end{bmatrix}}$$

$$= \begin{bmatrix} \cos(\theta+\alpha) & -\sin(\theta+\alpha) & 0 \\ \sin(\theta+\alpha) & \cos(\theta+\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The result from above shows that 2 successive rotations are additive.

SCALINGS WITH RESPECT TO THE ORIGIN:

By common sense, If we scale a shape with 2 successive scaling factors (Sx_1, Sy_1) and (Sx_2, Sy_2) with respect to the Origin. It is equal to a single scalling of $(Sx_1 * Sx_2, Sy_1 * Sy_2)$ with respect to the Origin. This multiplicative property can be demonstrated by composite transformation matrix

Mathematically:

$$\begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} Sx_1 \times Sx_2 & 0 & 0 \\ 0 & Sy_1 \times Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This demonstrates that 2 successive scalings with respect to the Origin are multiplicative.

A JAVA PROGRAM TO DO MATRIX MULTIPLICATION

```
Import Java util Scanner
Class Matmult {
    public static void main (String [] args) {
        Scanner In = new Scanner (System.In);
        ~~for int~~ int a[i][j] = 0
        for (int i = 0; i <= 2; i++) {
            for (int j = 0; <= 2; j++) {
                Con << a [i][j] <<;
            }
        }

        for (int i = 0; i <= 2; i++) {
            for (int j = 0; j <= 2, j++) {
                Con << [i][j] <<;
```

```
a[i][j] = In.nextInt()
    }
  }

// Multiplication Engine
for (int i=0; i<=2; i++) {
  for (int j=0; j=2, j++) {
    for (int k=0; k<=2; k++) {
      C[i][j] = C[i][j] + a[i,k] x b[k,j]
    }
  }
}.
```

Assignment:

Scale a triangle with vertices as (10, 15), (10, 20) and (15, 20) with scalling factor $S_x = 5$ and $S_y =$ and Rotate at an angle of 30° about the Origin Using any language of your choice to calculate the coordinates.

A JAVA PROGRAM TO THE ABOVE STATED QUESTION

```
import java.awt.geom.Point2D
public class CompositeTransformation {
```

```
Static class point {
    double x;
    double y;
    Point (double x, double y) {
        this.x = x;
        this.y = Y;
    }
}

Public Static point Scale (point point, double ScaleX,
double ScaleY) {
    double newX = point.X * ScaleX;
    double newY = point.Y * ScaleY;
    return new Point (newX, newY);
}

Public Static point rotate (point point, double angle)
{ double radians = Math.toRadians (angle);
    double newX = point.X * Math.Cos (radians) -
    Point.Y * Math.Sin (radians);
    double newY = point.X * Math.Sin (radians) +
    Point.Y * Math.Cos (radians);
    return new point (newX, newY);
}
```

```
Public static void main (String [] args) {
// define the Vertices of the triangle.
Point Vertex1 = new point (10, 15);
Point Vertex2 = New point (10, 20);
Point Vertex3 = new Point (15, 20);

// Scalling factors
double scalex = 5;
double scaleY = 10;

// Rotate angle in degrees
double rotateAngle = 30;

// Apply Scaling to the triangle Vertices.
Vertex1 = Scale (Vertex1, Scale x, Scale Y);
Vertex2 = Scale (Vertex2, Scale X, Scale Y);
Vertex 3 = Scale (Vertex 3, Scale x, Scale Y);

// Apply rotation to the Scaled Vertices
Vertex1 = rotate (Vertex1, rotate Angle);
Vertex2 = rotate (vertex 2, rotate Angle);
Vertex3 = rotate (Vertex 3, rotateAngle);
```

```
// Display the new vertices after composite transformation
System.Out.Println("Vertex 1: (" + Vertex1.X + "," +
Vertex1.Y + ")");
System.Out.Println("Vertex2: (" + Vertex2.X + "," +
Vertex2.Y + ")");
System.Out.Println("Vertex 3: (" + Vertex3.X + "," +
Vertex3.Y + ")");
} // End of main function
} // End of the program.
```

2 DIMENSIONAL VIEWING:

IMAGES ON THE SCREEN:

When showing an image of an Object on the Screen, we use a screen Coordinate system that defines the location of the Object in the same relative position as in the real world. After we select the screen Coordinate System, we change the picture to display interior Screen that means change it into Screen Coordinate System.

WINDOWS AND CLIPPING:

The world Coordinate system is used to define the

position of objects in the natural world. The intern of numbers can either be (positive, negative or decimal). However, the whole picture of the image can fit in to the screen, therefore we need to show only some part of the object.

Aside: WINDOWING: This is the capability ~~that~~ to show some part of an object internal a specify window.

Definition WINDOW: A window is a rectangular region in the world coordinate system. This is the coordinate system used to locate an object in the natural world. The world coordinate system does not depend on a display device, so the units of measure can be positive, negative or decimal numbers.

Definition VIEWPORT: This is the section of the screen where the image encompassed by the window on the world coordinate system will be drawn. A coordinate transformation is required to display the image, encompassed by the window, in the viewport. The viewport uses the "Screen Cordinates System" so this transformation is from the world cordinate system to

the screen cordinate system.

Definition CLIPPING: The process of "cutting off" parts of the image from view of the world when a window is "placed" on the world. here we examine each line to determine whether or not it is completely inside the window, completely outside the window, or crosses a window boundary. If inside the window, the line is displayed, if outside the window, the lines and points are not displayed. If a line crosses the boundary, we must determine the point of intersection and display only the part which ~~line~~ lies inside the window.

COHEN - ~~SUT~~ SUTHERLAND LINE CLIPPING:

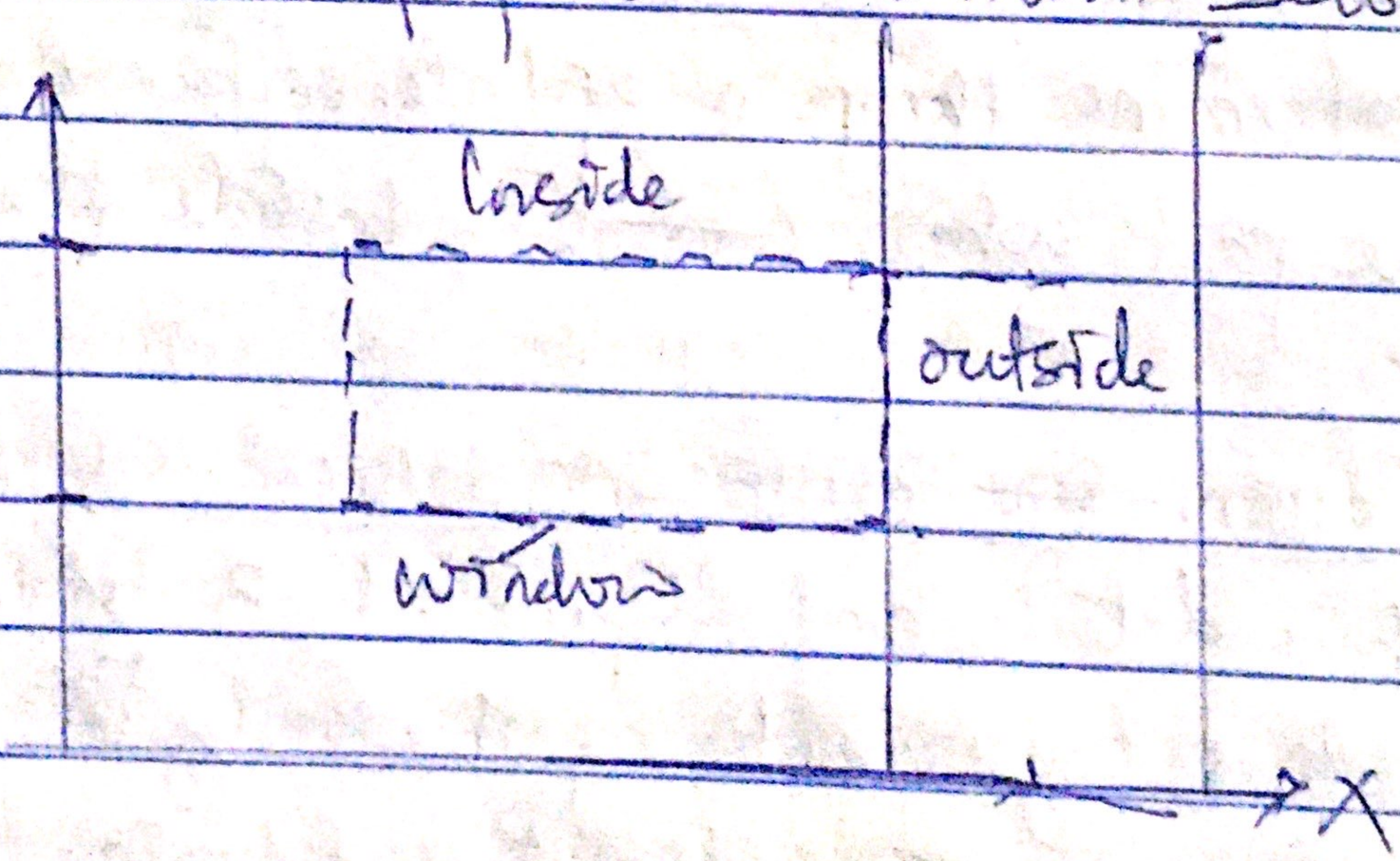
This detects and dispenses in 2 trivial cases which it does by considering the only ~~its~~ the endpoints of the line, and judging using the following criteria.

1. If both endpoints of a line lie inside the window, the entire line lies inside the window. It is trivially accepted and needs no clipping.
2. If both endpoints of a line lie entirely to one side of the window, the line must lie entirely outside the window

Moreover it is totally and needs to be neither clipped nor displayed.

INSIDE-OUTSIDE WINDOW CODES:

This algorithm setup a half-space code for each endpoint to determine whether the endpoints are inside or outside a window. Each edge of the window defines an infinite line that divides the whole space into two half-space, the inside half-space and the Outside-half-space. as shown below.



As we proceed around the window, extending each edge and defining an inside half-space and an Outside half-space. Nine regions are created - The eight "Outside" regions and one "Inside" region. Each of the nine regions associated with the window is assigned a 4 bit

code to identify the region. Each bit in the code is set to either 1 (true) or 0 (false). If the region is to the left of the window, the first bit of the code is set to 1, if the region is to the ~~top~~ right of the window, the second bit of the code is set to 1, if the region is set to the bottom of the window the third bit ~~is set~~ ~~to~~ of the code is set to 1 and if the region is set to the top of the window, the fourth bit of the code is set to 1. The 4 bits of the code then identify each of the nine regions as shown below.

1001	0001	0101
1000	0000 window.	0100
1010	0010	0110

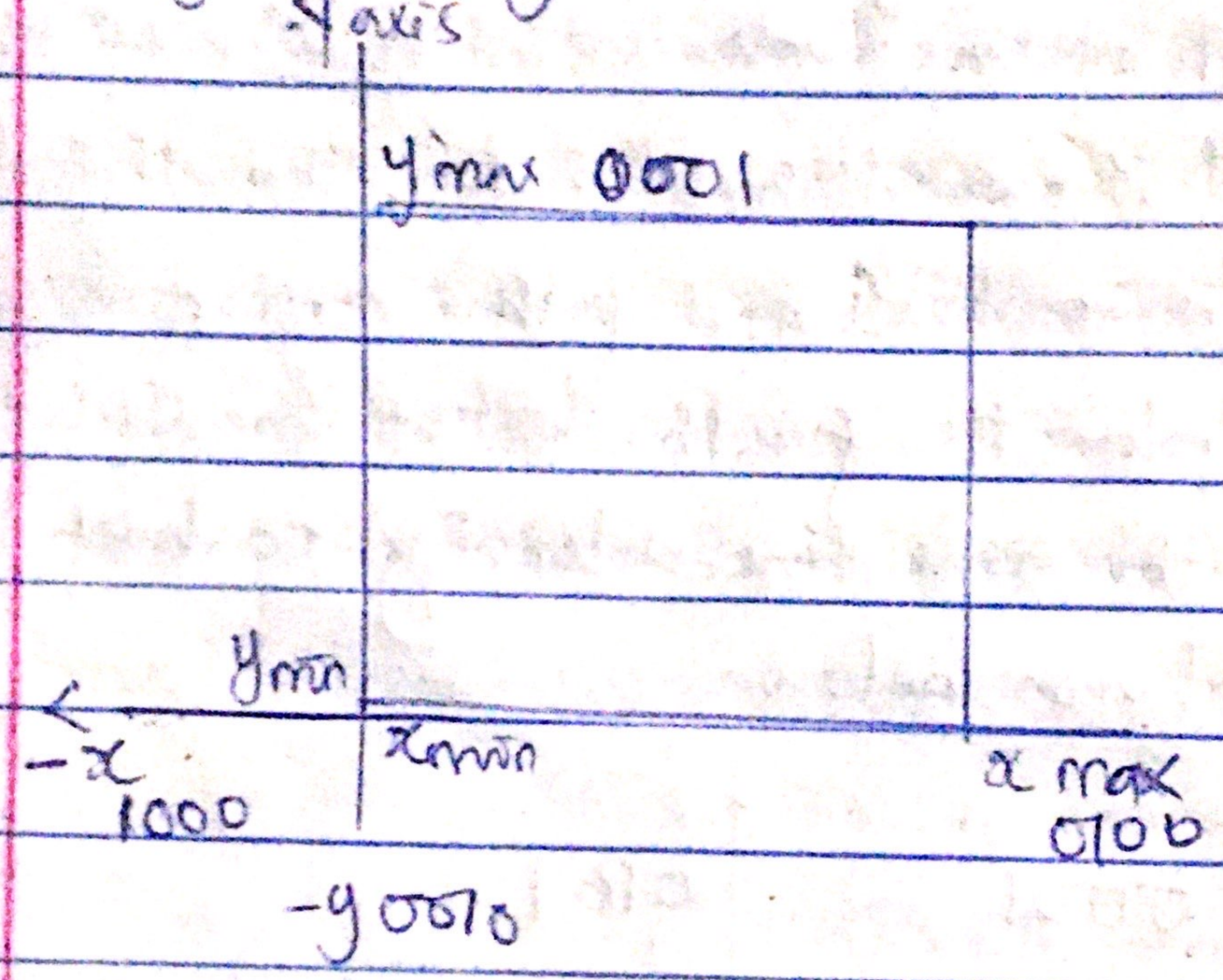
However, for any endpoint (x, y) of a line, the code can be determined that identifies which region the endpoint lies. The code's bit are set according to the following conditions: ~~Algorithm~~.

Asc
1. First bit set 1 : point lies to left of window $X < X_{min}$
2. Second bit set 1 : point lies to right of window $X > X_{max}$

③ The third bit set 1: point lies below (bottom) window $Y < Y_{min}$

④ fourth bit set 1: point lies above (top) window $Y > Y_{max}$

Diagrammatically:



The sequence of scanning the code becomes LRBT (left, right, bottom, top).

Note: However, once the code for each endpoint of a line are determined, the logical AND operation of the codes determines if the line is completely outside of the window. If the logical AND of the endpoint code is not zero (0000), the line can be trivially rejected. eg a line with endpoints 1001 and 1010,

$$\begin{array}{r} 1001 \\ 1010 \\ \hline 1000 \end{array}$$

The output is 1000 which shows that the line segment lies outside of the window

On the other hand a line with endpoints 1001 and 0110, the result would be 0010 which shows that the line is inside the window and could not be trivially rejected.

The Logical OR of the endpoint codes determines if the line is completely inside the window. If the logical OR zero (0000). The line can be trivially accepted. for Eg line with endpoints 0000 and 0000, the result would be 0000 meaning it would be trivially accepted but for a line with endpoints 0000 and 0110, the logical OR is 0110 and the line cannot be trivially accepted.

COHEN - SUTHERLAND ALGORITHM (ADC)

This makes use of the divide and conquer algorithm to know if a line can be trivially accepted or rejected.

To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions. Each endpoint of the line segment is then assigned the code of the region in which it lies.

① Given a line segment with endpoint $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$

(2) Compute the 4-bit codes for each endpoint.
If both codes are 0000 (bitwise OR of the code yields 0000) line lies completely inside the window: pass the endpoints to the draw routine.
If both codes have a 1 in the same bit position (bitwise AND of the code is not 0000), the line lies outside the window. It can be trivially rejected.

(3). If line cannot be trivially accepted or rejected, at least one of the two endpoints must be outside the window and the line segment crosses a window edge. The line must be clipped at the window edge before being passed to the drawing routine.

(4) Examine one of the endpoints, say $P_1 = (x_1, y_1)$ Read P_1's 4-bit code in order: Left to Right, Bottom to top

(5) When a set bit (1) is found, compute the intersection I of the corresponding window edge with the line from P_1 to P_2. Replace P_1 with I and repeat. The algorithm

3D TRANSFORMATIONS:

The ~~germ~~ generic transformations of and object modelling in 3D are extended from 2D methods by

Note $X_{min} = X_{Left}$ $\quad Y_{max} = Y_{Top}$
$X_{max} = X_{Right}$ $\quad Y_{min} = Y_{Bottom}$

HOW TO CALCULATE INTERSECTIONS

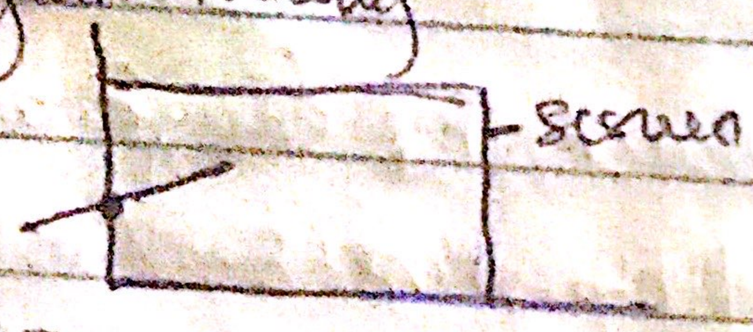
Assume that we're clipping a polygon's edge with vertices at (x_1, y_1) and (x_2, y_2) against a clip window with vertices at (X_{min}, Y_{min}) and (X_{max}, Y_{max}).

(a) The location (IX, IY) of the intersection of the edge with the left side of the window is:

i. $IX = X_{min}$

ii. $IY = Slope * (X_{min} - X_1) + y_1$, where the slope $= \frac{(y_2 - y_1)}{(x_2 - x_1)}$

diagrammatically

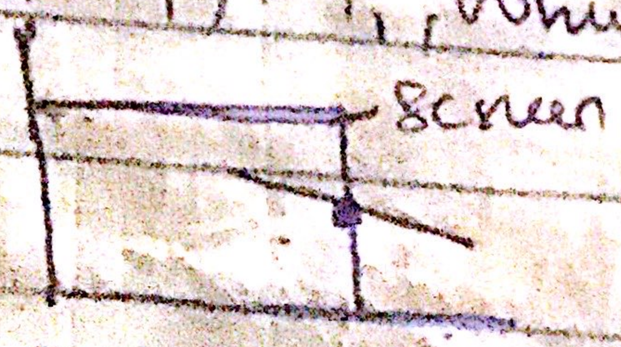


(b) The location of intersection of the edge with the right side of the window is:

i. $IX = X_{max}$

ii. $IY = Slope * (X_{max} - X_1) + Y_1$, where the slope $= \frac{(y_2 - y_1)}{(x_2 - x_1)}$

diagrammatically:



(c) The intersection of the polygon's edge with the top side of the window is

i. $IX = X_1 + \frac{(Y_{max} - y_1)}{Slope}$

ii. $IY = Y_{max}$

diagrammatically

(D) Finally the intersection of the edge with the bottom side of the window is:

i $IX = X_1 + \frac{(Y_{min} - Y_1)}{Slope}$

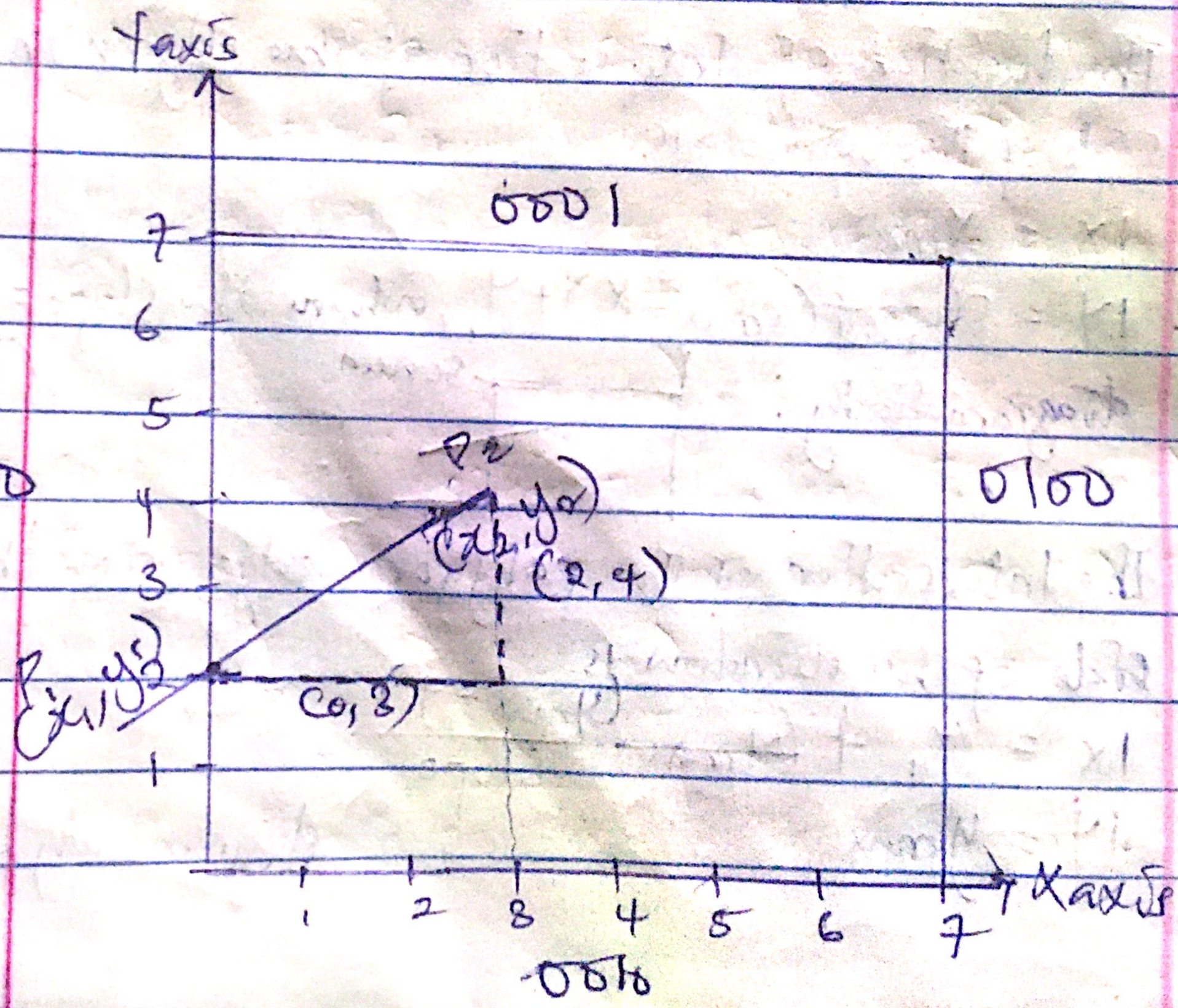
ii $IY = Y_{min}$

PROBLEMS WITH THIS ALGORITHM

(1) This algorithm does not work if the clip window isn't convex

(2) If the polygon is not also convex, there may be some dangling edges.

Example: Given the diagram below clip the line on the to fit into the screen and calculate the intercept



from the diagram above

$x_{min} = 0$ (the Minimum Value of x)

$Y_{min} = 0$ (The Minimum Value of Y)

$x_1 = 0 \quad x_2 = 2$

$y_1 = 2 \quad y_2 = 4$

~~Sol~~ Slope: $= \frac{y_2 - y_1}{x_2 - x_1} = \frac{4-2}{3-0} = \frac{2}{3}$

However if we compute the bitwise AND and OR of the line to know if it is completely inside, Outside or crosses the edge - we have the following:

~~Thang~~ The line cuts accross the screen from the left into the screen therefore:

Left code = 1000

Center Code = 0000

AND bitwise Operation =
$$\begin{array}{r} 1000 \\ 0000 \\ \hline 0000 \end{array}$$

~~AND~~ OR Bitwise Operation =
$$\begin{array}{r} 1000 \\ 0000 \\ \hline 1000 \end{array}$$

One of the bits is 1 meaning the line is not completely visible on the screen and needs to be clipped. Therefore we perform the clipping.

Since the line cuts into the screen from the left, we make use of the following formulae

$$IX = X_{min}$$

$$IY = Slope * (X_{min} - X_1) + Y_1$$

where $X_{min} = 0$

$Slope = 2/3$

$X_1 = 0$

$Y_1 = 2$

$$IY = 2/3 \times (0 - 0) + 2$$

$$2/3 \times 0 + 2$$

$$IY = 2$$

Therefore the point of intersection becomes 0 and 2.

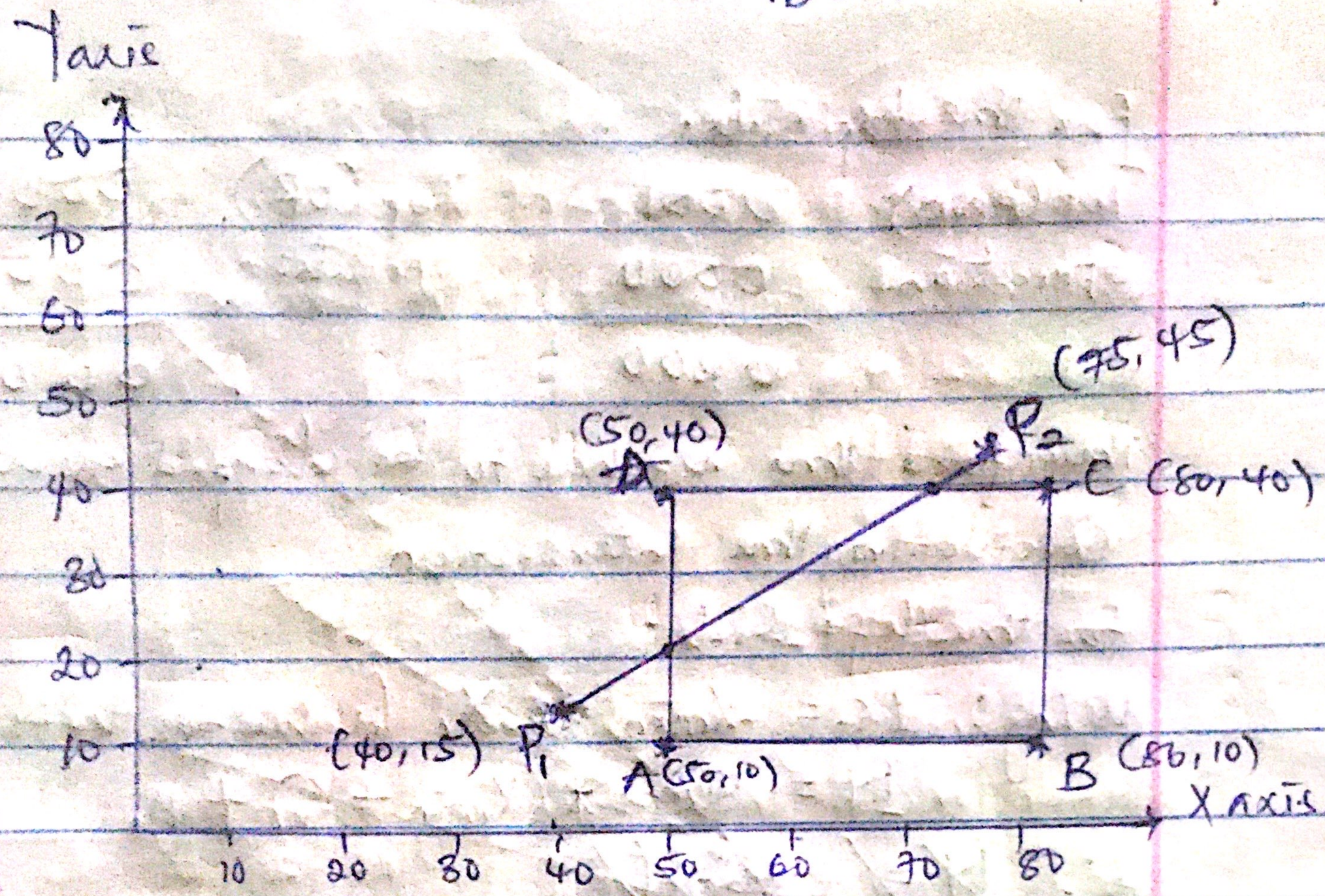
Example 2: Use Cohen Sutherland Algorithm to clip line $P_1(40, 15)$ and $P_2(75, 45)$ against a window $A(50, 10)$, $B(80, 10)$ $C(80, 40)$ and $D(50, 40)$

Soln

Plotting the graph we have the following.

$Y_T = 45$ $X_L = 40$

$Y_B = 15$ $X_R = 75$.



from the graph above we calculate the slope as follows.

$$\text{slope } (m) = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{where } y_2 = 45 \quad x_2 = 75$$

$$y_1 = 15 \quad x_1 = 40$$

$$m = \frac{45 - 15}{75 - 40} = \frac{30}{35} = 0.857 \simeq \underline{\underline{0.86}}$$

We therefore check to know if the line is to be clipped. Since the line cuts from top accross to the screen to the left we compute as follows

left =	1000	left (OR)	1000
(AND operation)	0000	operation	0000
	0000		1000

Therefore the left side has to be clipped.

For the top side.

top (AND)	0001	Top (OR)	0001
operation	0000	operation	0000
	0000		0001

Therefore the top has to be clipped as well

Calculating the intersection

for the left side:

Using: $I_X = X_{min}$ or X_L (value for left side of X)

$I_Y = \text{Slope} * (X_{min} - X_1) + y_1$

$\therefore I_Y = 0.86 * 0$

Note: The X_L = The minimum point of the line on X axis

X_R = The maximum value of the line on X axis.

Y_b = The minimum value of the line on the Y axis

Y_T = The maximum value of the line on the Y axis.

for the left side.

$I_Y = 0.86 * (40 - 40) + 15$

$I_Y = 0.86 \times (0) + 15$

$I_Y = 15.86$

$I_X = 40$

for the top of the screen we solve ~~the~~ as follows:

$I_X = X_1 + \frac{(y_{max} - y_1)}{\text{Slope}}$

$I_Y = y_{max}$

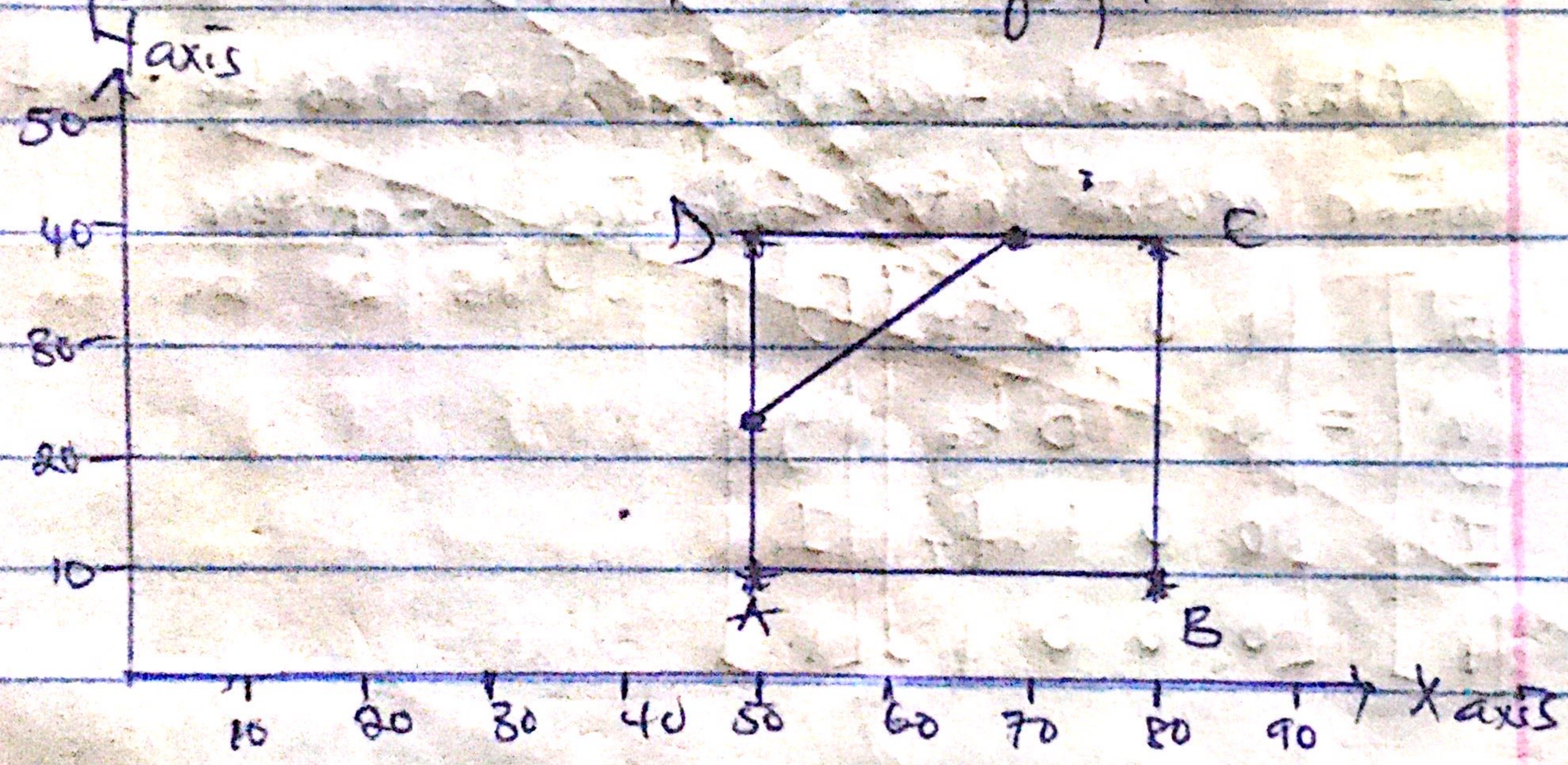
$$I_x = 40 + \frac{(45-15)}{0.86}$$

$$I_x = 40 + 34.88$$

$$I_x = 74.88$$

$$I_y = 45$$

After the calculation, the new graph become:



3D TRANSFORMATIONS:

This generic transformations and object modelling in 3D are extended from 2D methods by including the considerations for the X-Cordinates, Basic geometric transformations are as follows:

- Translation
- Rotation.
- Scalling.

3D TRANSLATIONS:

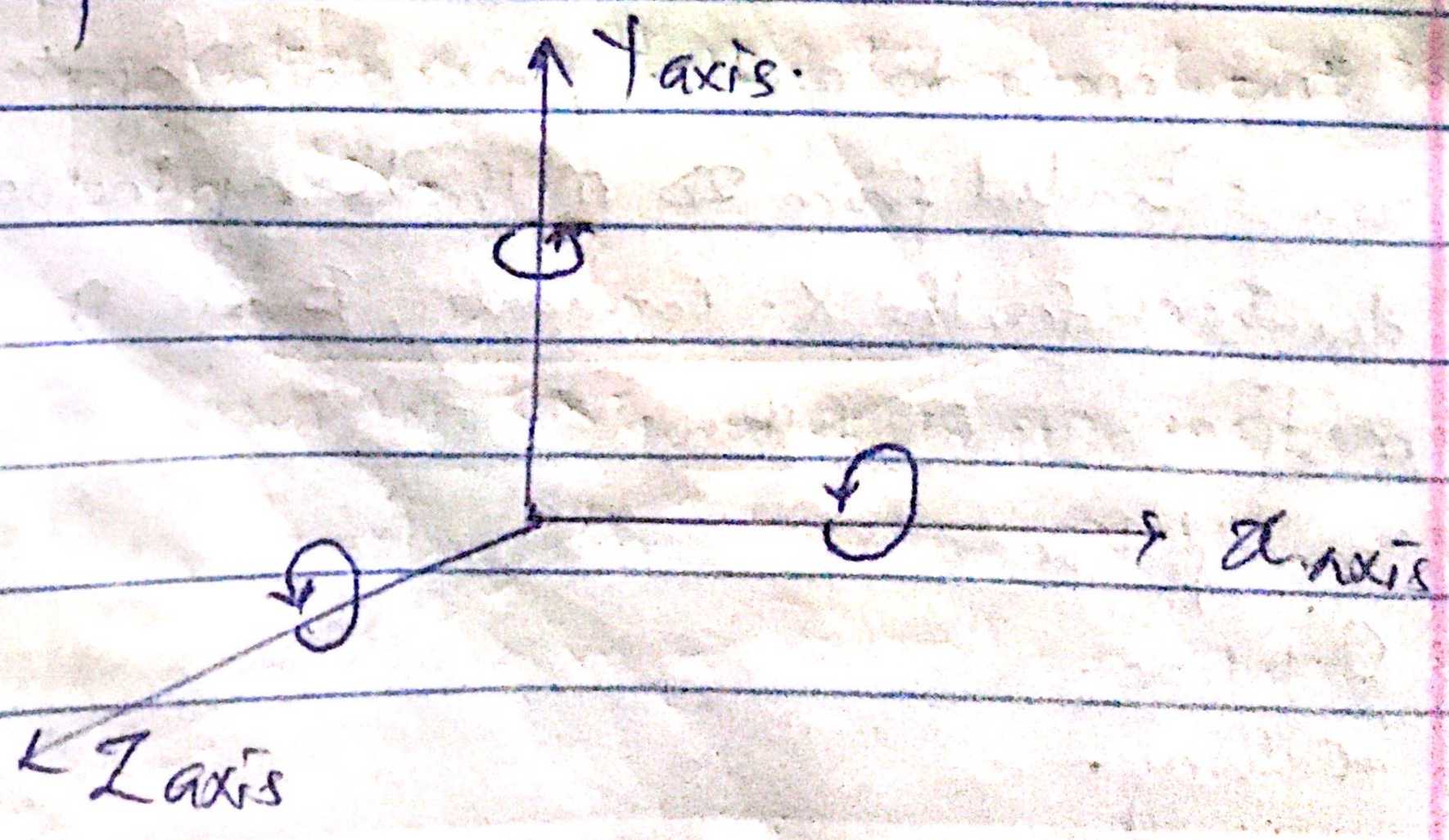
We translate a 3D point by adding translation distances, t_x, t_y and t_z to the Original Coordinate position. (x, y, z).

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

Alternatively translation can also be the transformation matrix in the following formulas

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

The 3D transformation makes use of 3 points $(x, y$ and $z)$ represented as shown below.



Example 1: Translate a triangle with Vertices at Original Cordinates (10, 25, 5), (5, 10, 5), (20, 10, 10) by $t_x = 15$, $t_y = 5$ and $t_z = 5$

Note: When performing 3D transformations, you plot the X and Y Values and therefore try imagining that Z position to portray the 3D View in Memory.

Soln

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 15 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 10 \\ 25 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 10+0+0+15 \\ 0+25+0+5 \\ 0+0+5+5 \\ 0+0+0+1 \end{bmatrix} = \begin{bmatrix} 25 \\ 30 \\ 10 \\ 1 \end{bmatrix}$$

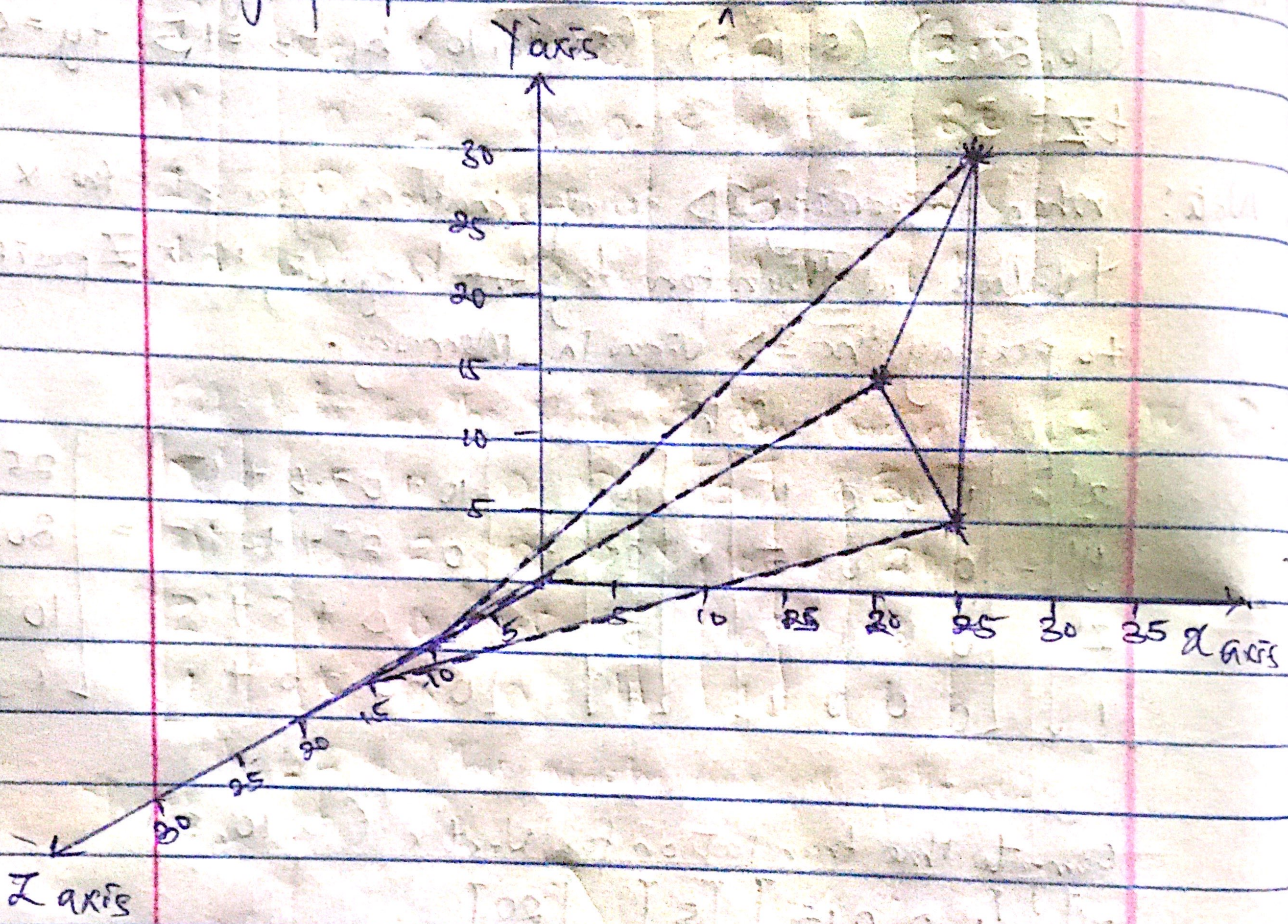
Compute the translation for Vertex (5, 10, 5)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 15 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 5 \\ 10 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 20 \\ 15 \\ 20 \\ 1 \end{bmatrix}$$

Compute the translation for Vertex (20, 10, 10)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 15 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 20 \\ 0 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 25 \\ 5 \\ 15 \\ 1 \end{bmatrix}$$

The graph from the above Resultant Matrix



However, you can choose not to represent the Z values and just imagine its positions or you can as well choose to represent it using dotted lines to depict how it will appear on 3D.

SCALING IN 3D WITH RESPECT TO A SELECTED FIXED POSITION

Scaling involves the increase in the size of an image on the screen based on some cordinate values.

The Scalling formulae is given as follows:

$x' = x * S_x$

$y' = y * S_y$

$z' = z * S_z$

and the Matrix is given as follows

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Example: ~~3D Rotation~~ Scale a triangle with Vertexes at Original Coordinates (10, 25, 5), (5, 10, 5), (20, 10, 10) by $S_x = 1.5$, $S_y = 2$, and $S_z = 0.5$ with respect to the Centre of the triangle: Try sketching the graph of before and after the scalling Operations.

Sol:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 10 \\ 25 \\ 5 \\ 0 \end{bmatrix} = \begin{bmatrix} 15 \\ 50 \\ 2.5 \\ 1 \end{bmatrix}$$

Values

Calculate for the ~~Coordinates~~ for the Coordinates (5, 10, 5)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 5 \\ 10 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 7.5 \\ 20 \\ 2.5 \\ 1 \end{bmatrix}$$

Calculate for the Values for the Coordinates (20, 10, 10).

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 20 \\ 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 30 \\ 20 \\ 5 \\ 1 \end{bmatrix}$$

Therefore the graphs are as follows:

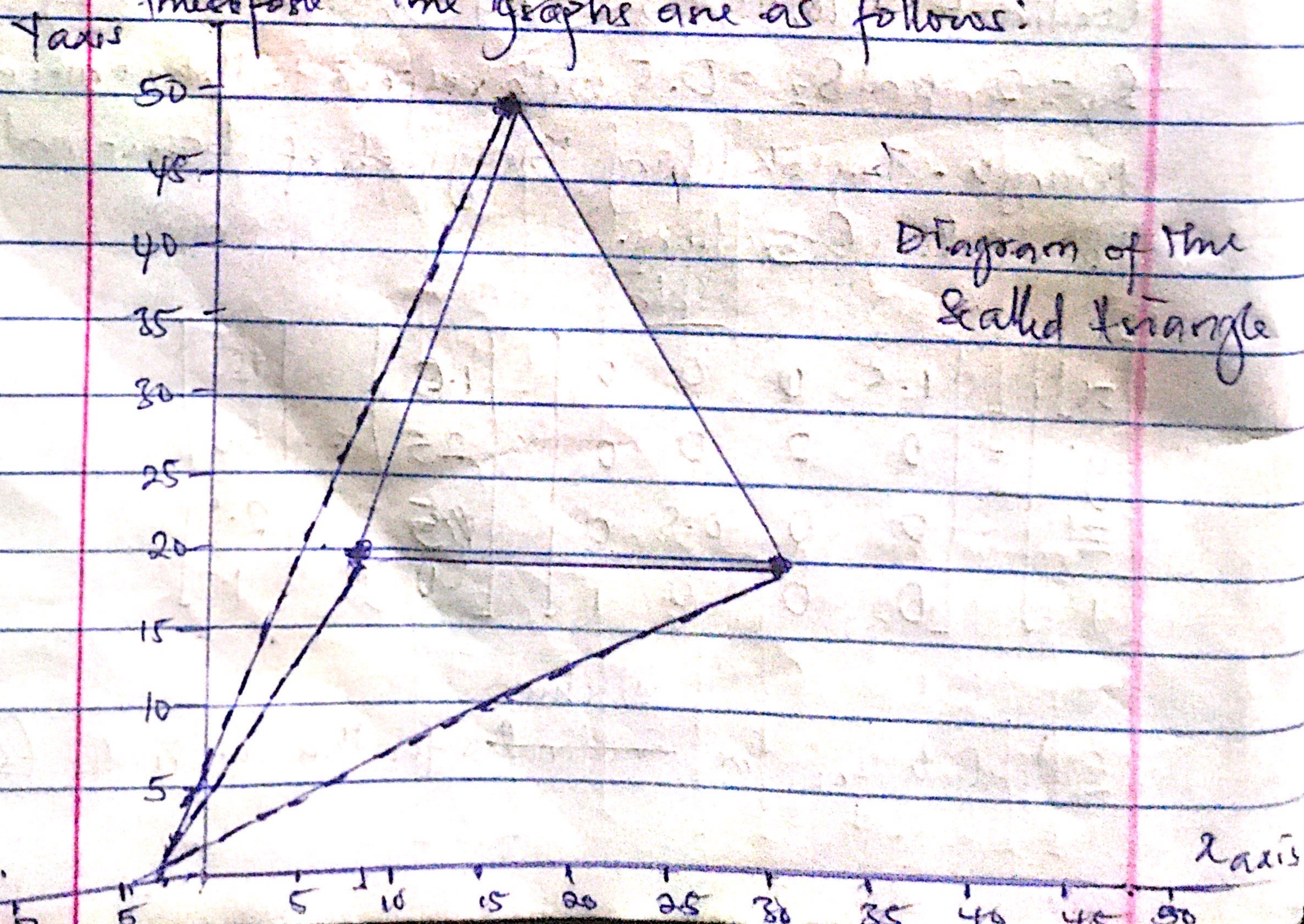


Diagram of the Scaled triangle

CORDINATES AXES ROTATION:

A 3D rotation can be specified around any line in space. The easiest rotation axes to handle are the coordinate axes.

i Z-axis rotation

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

Matrix of Z

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

~~tab~~

ii X-axis rotation.

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

matrix of X

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

iii Y-axis rotation

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

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