

University Of Anbar
Cs & It College
Computer Science Department
3rd Stage
First Course



Introduction to Computer Graphics

The phrase “Computer Graphics” was coined in 1960 by William Fetter, a graphic designer for Boeing. Computers have become very significant in today’s modern world. Computer graphics are pictures and movies created using computers usually referring to image data created by a computer specifically with help from specialised graphical hardware and software. It is a vast and recently developed area of computer science.

There are two types of graphics that are used nowadays, *interactive and passive*.

- 1- **Interactive Computer graphics (IGU):** it involves a two-way communication between the computer and user. Here, the displayed picture is modified appropriately to signals, the computer receives from input device e.g. (keyboard, mouse, joystick,..etc). It appears that picture is changing instantaneously in response to the observer’s commands.
Example: computer games (video game controller).
- 2- **Non-Interactive Computer graphics: (passive Computer graphics).** Here, the user does not have any kind of control over the image. The image is totally under the control of program instructions not under the user.
Example: screen savers.

The differences between active and passive graphics are shown in **Table 1**.

Table 1: The differences between Active and Passive Graphics.

	Passive Computer graphics	Active Computer graphics
Control	No control	Control (dynamic nature)
Communication	One way	Two way
Interaction	No interaction.	High interaction.
Motion and Updation	No facility	2-D,3-D transformations
Prone to Problems	Less	More

Choosing between interactive graphics and passive graphics depends on whether you are aiming for simplicity or features. Interactive graphics are more entertaining than passive graphics but is also more difficult to deal with. Some sites choose a compromise by creating two versions for their site; one with interactive graphics, and another with passive graphics.

There are many advantages of interactive graphics over passive graphics:

- ***Higher quality of images***
- ***Low cost***
- ***Higher productivity***
- ***Low analysis***

Application of Computer Graphics

Today, computers and computer-generated images touch many aspects of our daily life. Computer imagery is found on television, in newspapers, for example in their weather reports, or in all kinds of medical investigation and surgical procedures.

Some people even make computer graphics as art. **We can classify applications of computer graphics into four main areas:**

- Display of information
- Design
- User interfaces
- Simulation

CAD (Computer Aided Design):

use of a computer to aid in the design of product.

CAM (Computer Aided Manufacturing):

use of a computer to control the manufacturing of products.

CAI (Computer Aided Instructing):

use of computer to display animated pictures to illustrate educational concept.

CAE (Computer Aided Engineering):

use of a computer as engineer work.

Presentation Graphics:

To produce illustrations which summarize various kinds of data.

Entertainment:

Motion pictures, Music videos, and TV shows, Computer games.

Education and Training:

Training with computer-generated models of specialized systems such as the training of ship captains and aircraft pilots.

Visualization:

For analyzing scientific, engineering, medical and business data or behaviour.

Image Processing:

Image processing is to apply techniques to modify or interpret existing pictures.

Graphical User Interface:

Multiple window, icons, menus allow a computer setup to be utilized more efficiently.

Simulation:

use of a computer to experience circumstance that otherwise would be too expensive or catastrophic to experience in reality E.g.: flight simulators, simulating unclear reactors.

Graphics Display Devices

A display device is a device for visual or tactile presentation of images (including text) acquired, stored, or transmitted in various forms. As we are familiar that we can interact with the computer system through the Display Devices. These display devices enable us to communicate with the computer through the graphical interface. There are different types of Display Devices which are used to interact with the computer system or any other device. *Some of them are following*

- **Monitor**

This is mostly used as an output display device. There are further two types of Monitors available which are discussed following

1- Cathode Ray Tube (CRT):

A CRT monitor is heavier than flat-panel display (FPD) because it contains a large cathode ray tube. It uses electron guns to activate phosphors behind the screen, causing each pixel on the monitor to generate red, green, or blue.

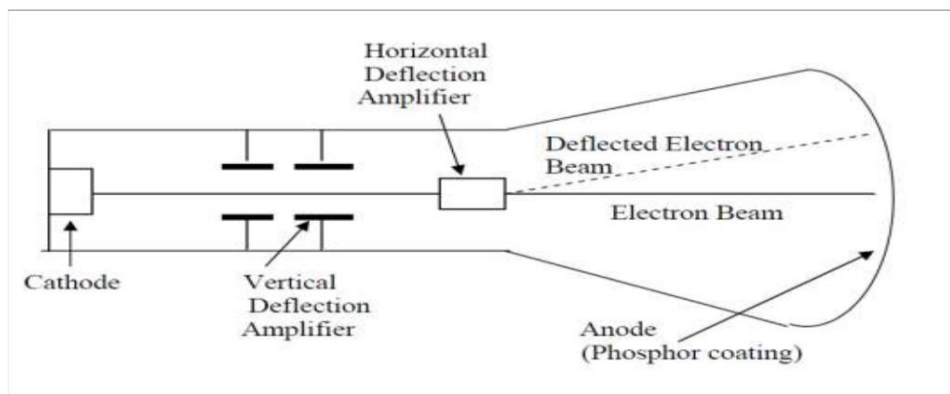


Figure 1: A CRT Monitor.

How the cathode ray tube (CRT) works:

A CRT is an evacuated glass tube. The CRT consists of three electron guns: red, green and blue. They emit a beam of negative charged electrons towards a positively charged phosphor-coated screen. Along the way the electron beam passes through the focusing system that concentrates the beam so that by the time the electrons reach the screen they have converged to a small dot. The beam then passes through the deflection system (horizontal and vertical) which deflect the beam to strike any point on the screen. When this focused electron beam strikes the screen, the phosphor emits as dot of visible light. There are two techniques used for producing image on the CRT screen: Vector scan / random scan and Raster scan.

The video display is divided in to very small dots called "**pixels**" (picture elements) each pixel is composed of a triangular pattern of red, green and blue phosphor dot. The CRT has three electron guns one for each of the three primary colors: red, green and blue each electron gun hits its corresponding phosphor dot causing that particular color to appear on the screen the light on the display screen starts to fade as soon as the beam moves to another location. So, the beam must refresh the screen by illuminating pixels 30 to 60 times each second.



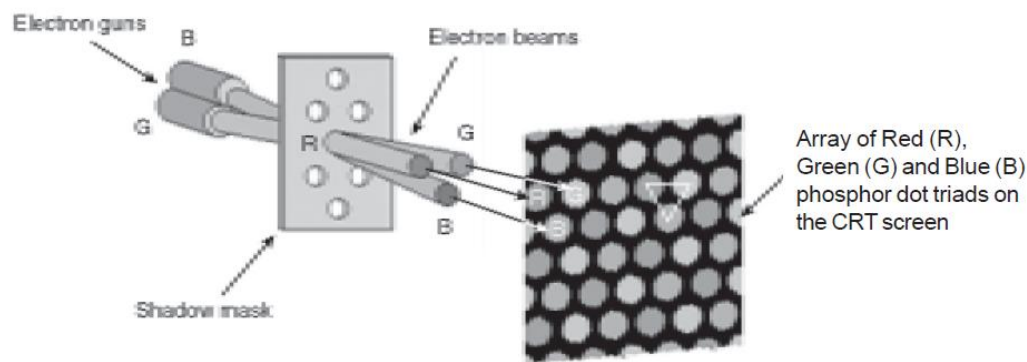


Figure 2: Electron Guns Passing Through CRT.

Generating Color on a *RGB* monitors:

Each electron gun in an **RGB** monitor, has an assigned number of bit that determines the intensities of the red, green and blue phosphors, with one bit per gun eight color are possible ($2^3=8$).

Table 2: Color Bit Distribution

<i>R</i>	<i>G</i>	<i>B</i>	<i>Binary value</i>	<i>Color name</i>
<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>Black</i>
<i>0</i>	<i>0</i>	<i>1</i>	<i>1</i>	<i>Blue</i>
<i>0</i>	<i>1</i>	<i>0</i>	<i>2</i>	<i>Green</i>
<i>0</i>	<i>1</i>	<i>1</i>	<i>3</i>	<i>Cyan</i>
<i>1</i>	<i>0</i>	<i>0</i>	<i>4</i>	<i>Red</i>
<i>1</i>	<i>0</i>	<i>1</i>	<i>5</i>	<i>Magenta</i>
<i>1</i>	<i>1</i>	<i>0</i>	<i>6</i>	<i>Brown</i>
<i>1</i>	<i>1</i>	<i>1</i>	<i>7</i>	<i>White</i>

If a fourth bit is used for the brightness' or intensity of the displayed color it results in ($2^4=16$) possible colors. When the brightness bit equals one another set of eight bright colors is produced.

Note: if True Color is 24 bit then displayed color = 0 to $2^{24}-1$.

In RGB 24 bit => Red (8 bit = 0 to 2^8-1), Green (8 bit = 0 to 2^8-1), Blue (8 bit = 0 to 2^8-1).

2- Flat Panel Display (FPD):

A flat-panel display (FPD) is very thin, lightweight, and very little power device. A flat panel display has common types like LCD (Liquid Crystal Display), LED (Light Emitting Diode) and OLED (Organic LED). LCD uses a liquid crystal molecule for displaying, LED uses light emitting diodes for display while OLED uses a special organic compound for display. **The main differences between CRT Display and Flat Panel Display (FPD) are shown in following Table3:**

Table 3: The Differences between CRT and FPD.

	<i>CRT</i>	<i>FPD</i>
<i>Stands For</i>	For “Cathode Ray Tube “.	For “Flat Panel Display “.
<i>Examples</i>	Televisions and Was Used in Old Computer Monitors.	LCD, Plasma, OLED, And LED
<i>Cost</i>	It Is Less Expensive.	It Is More Expensive.
<i>Power Consumption</i>	It Consumes High Power.	It Consumes Low Power.
<i>Weight</i>	Heavier	Less Heavy
<i>Image Retention</i>	Not There in CRT.	There in FPD.
<i>Size</i>	Cannot Be Used for Smaller Displays Ex: Watches	Can Be Used for Smaller Displays
<i>Thickness</i>	Much Larger	Less Large
<i>Dead Pixels</i>	Do Not	Suffer

All computer art is digital, but there are two very different ways of drawing digital images on a computer screen, known as **raster and vector graphics**.

Raster Graphics:

Describes a picture using many small dots of color. Each dot is called a *pixel*, which is an abbreviation for “picture element”. If the dots are small enough and close enough together, a person does not see the individual dots, but rather sees a “picture”.

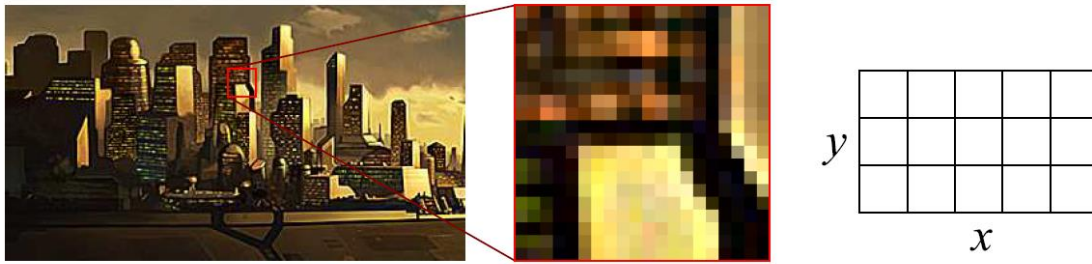


Figure 3: Raster graphics lose quality as they are scaled up (or down!).

One of 2^N intensities or colors are associated with each pixel, where N is the number of bits per pixel. Greyscale typically has one byte per pixel, for $2^8 = 256$ intensities. Color often requires one byte per channel, with three color channels per pixel: red, green, and blue.

A **bitmap** (also called "**raster**") graphic is created from rows of different colored pixels that together form an image. In their simplest form, bitmaps have only two colors, with each pixel being either black or white. With increasing complexity, an image can include more colors; photograph-quality images may have millions. Examples of bitmap graphic formats include GIF, JPEG, PNG, TIFF, XBM, BMP, and PCX as well as bitmap (i.e., screen) fonts.

Color data is stored in a **frame buffer**. This is sometimes called an image map or bitmap.

Scan conversion is the process of converting basic, low level objects into their corresponding pixel map representations. "The process of representing continuous graphics objects as a collection of discrete pixel is called scan conversion".

- **Vector Graphics:**

Describes objects as geometric shapes using mathematical equations. A picture is created from the mathematical descriptions through a process called *rendering*. The results of a rendering is a 2-dimensional raster image.

In below figure we adopt a standard notation and show vector quantities in **bold** type scalar quantities (like real number) in standard type. Thus, the arrow from A to B will be written \vec{AB} and the arrow from B to A will be written \vec{BA} ; in both cases

the length of the line (the size of the vector) is the same. (in handwriting, we usually underline vector quantities).

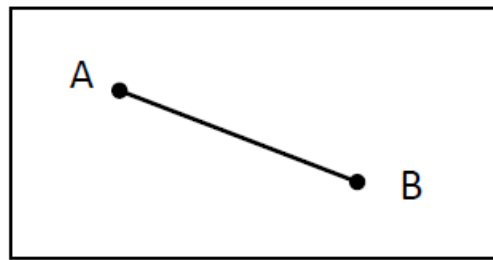


Figure 4: lines Connection.

As shown in above figure, vector scan CRT display directly traces out only the desired lines on CRT i.e. If we want a line connecting point A with point B on the vector graphics display, we simply drive the beam deflection circuitry, which will cause beam to go directly from point A to B. if we want to move the beam from point A to point B without showing a line between points, we can blank the beam as we move it. To move the beam across CRT, the information about, magnitude and direction is required. This information is generated with the help of vector graphics generator. The figure bellow shows the typical vector display architecture.

Table 4: The differences, advantage (pros), and disadvantages (cons) between Raster and vector Graphics.

Raster Graphics	Vector Graphics
Pixel-based	Shapes based on mathematical calculations
Comprised of pixels, arranged to form an image	Comprised of paths, dictated by mathematical formulas
Constrained by resolution and dimensions	Infinitely scalable
Capable of rich, complex color blends	Difficult to blend colors without rasterizing
Large file sizes (but can be compressed)	Small file sizes
File types include .JPG, .GIF, .PNG, .TIF, .BMP, .PSD; PLUS .EPS AND .PDF when created by raster programs	File types include .AI, .CDR, .SVG; PLUS .EPS AND .PDF when created by vector programs
Common raster programs: photo editing / paint programs such as Photoshop & Paint Shop, GIMP (free)	Common vector programs: drawing programs such as Illustrator, CorelDraw, Inkscape (free)
Capable of detailed editing	Less detailed, but offers precise paths
Do not scale up optimally - Image must be created/scanned at the desired usage size or larger	Can be scaled to any size without losing quality

Important Characteristics of Video Display Devices:

Persistence: The major difference between phosphors is their persistence. It decides how long they continue to emit light after the electron beam is removed.

Resolution: Resolution indicates the maximum number of points that can be displayed without overlap on the CRT. Resolution depends on the type of phosphor, the intensity to be displayed and the focusing and deflection systems used in the CRT.

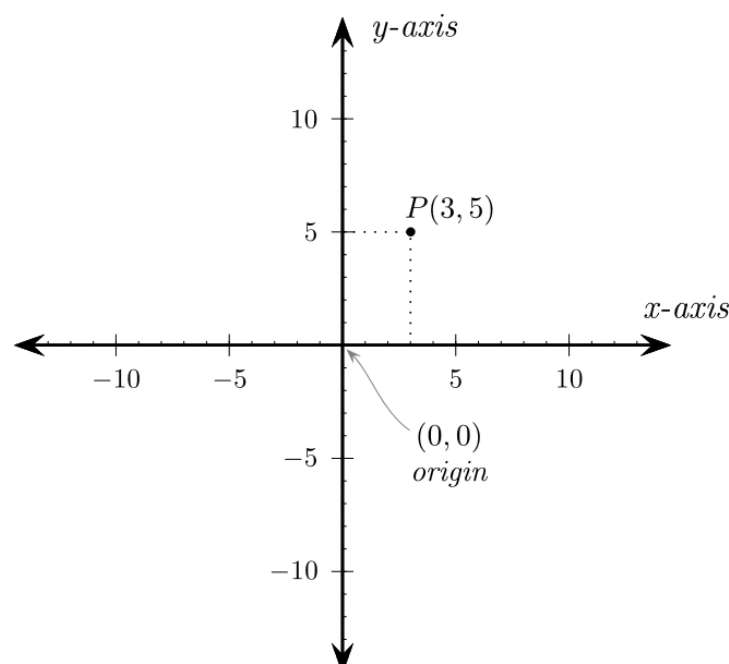
Aspect Ratio: It is the ratio of vertical points to horizontal points to produce equal length lines in both directions on the screen. An aspect ratio of 4/5 means that a vertical line plotted with four points has the same length as a horizontal line plotted with five points.

Line Drawing Algorithms

1- Plotting Points:

To be able to plot points, one needs to first decide on a point in plane which will be called the *origin*, and a couple of perpendicular lines, called the x and y axes, as well as a preferred direction on each of the lines. Usually one chooses the x axis pointing right and the y axis pointing up, and these will be named the *positive* directions. Also, one picks a segment in the plane which is declared to be of unit length. Using rotated versions of this segment, one can measure distances along the x and y axes.

Having the origin and the axes in place, given a pair (x, y) of real numbers, one considers the point on the x axis at distance $|x|$ from the origin and along the positive direction if $x \geq 0$, and the other direction otherwise. In the same way one picks the point on the y axis corresponding to the number y . The line parallel to the y axis going through the first point and the line parallel to the x axis going through the second point will intersect at precisely one point, which will be called the point with coordinates (x, y) .



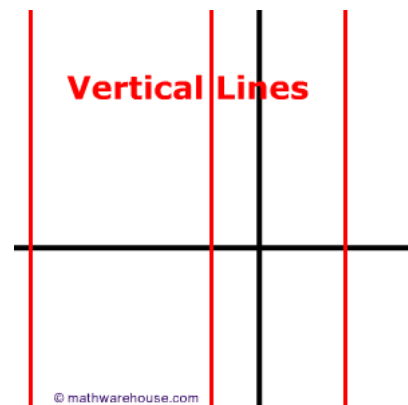
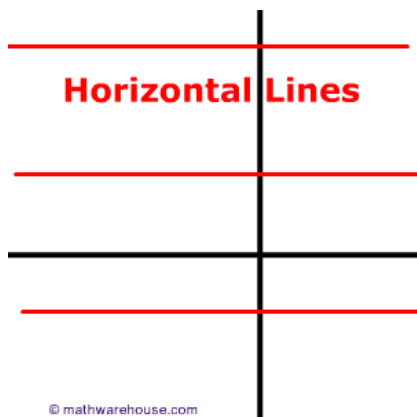
2- Drawing Lines

Requirements for an acceptable line drawing algorithm

1. The line should appear to be straight.
2. The line should terminate accurately.
3. The line should have constant density.
4. The line density should be independent of line length and angle.
5. Line should be drawn rapidly.

Horizontal and Vertical Lines

Vertical lines are lines that go straight up and down, parallel to the y-axis of the coordinate plane. All points on the line will have the same x-coordinate. While, A horizontal line is one which runs left-to-right across the page.



A line parallel to the x -axis is called a **horizontal line**. A line parallel to the y -axis is called a **vertical line**.

Example:

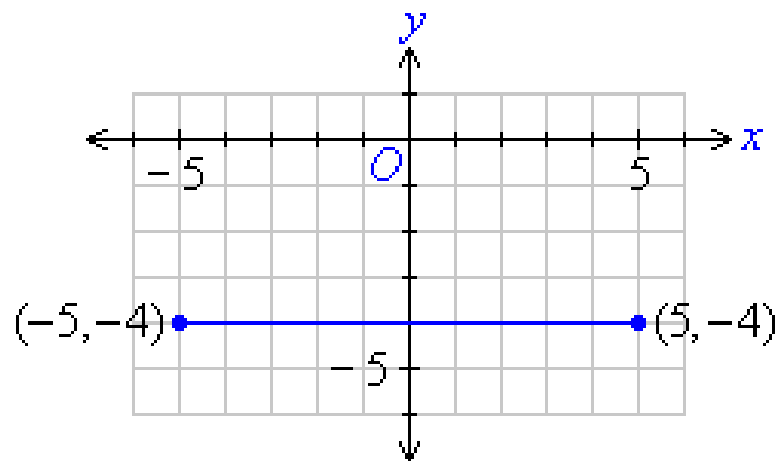
Plot the graph of each of the following relations:

a. $y = -4$

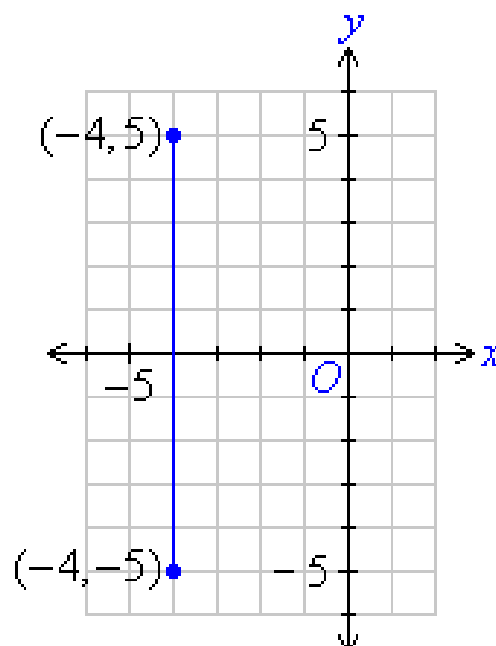
b. $x = -4$

Solution:

a. The graph of the relation $y = -4$ is a line parallel to the x -axis because it passes through points such as $(-5, -4)$ and $(5, -4)$.



b. The graph of the relation $x = -4$ is a line parallel to the y -axis because it passes through points such as $(-4, -5)$ and $(-4, 5)$.



The following lines of **MATLAB** code draw a horizontal line between the two points,

```
x=1:12;  
y=5;  
plot (x, y*ones(size(x)))
```

Diagonal lines

The following lines of **MATLAB** code draw a diagonal line between the two points,

```
x = linspace(0,12);  
y = linspace(0,12);  
Plot(x, y);
```

A **line drawing algorithm** is a graphical algorithm for approximating a line segment on discrete graphical media. On discrete media, such as pixel-based displays and printers, line drawing requires such an approximation (in nontrivial cases). Basic algorithms rasterize lines in one color. A better representation with multiple color gradations requires an advanced process, spatial anti-aliasing.

Line drawing on discrete graphics devices such as raster video displays, plotters, and image printers is one of the fundamental operations in computer graphics. Real-time interactive applications or high-speed image output (such as on a Postscript laser printer) may require line drawing speeds in the millions of pixels

per second. Such demands, which are ever increasing, push the efficiency of line generation.

A line connects two points. It is a basic element in graphics. To draw a line, you need two points between which you can draw a line. In the following three algorithms, we refer the one point of line as X_0, Y_0 and the second point of line as X_1, Y_1 . *The following is a partial list of line drawing algorithms:*

A Naive Line-Drawing Algorithm

The simplest method of screening is the direct drawing of the equation defining the line.

```
dx = x2 - x1
dy = y2 - y1
for x from x1 to x2
{
    y = y1 + dy * (x - x1) / dx
    plot(x, y)
}
```

It is here that the points have already been ordered so that $x_2 > x_1$. This algorithm works just fine when $dx \geq dy$ (i.e., slope is less than or equal to 1), but if $dx < dy$ (i.e., slope greater than 1), the line becomes quite sparse with lots of gaps, and in the limiting case of $dx=0$, only a single point is plotted.

The naïve line drawing algorithm is inefficient and thus, slow on a digital computer. Its inefficiency stems from the number of operations and the use of floating-point calculations. Line drawing algorithms such as Bresenham's or Wu's are preferred instead.

A Using Line Equation 'Y=mX+b'

This method is advantaged the line equation to finds point where they belongs in line. The constant **b** is found by if it assigns line point in this equation then you find the value of **b**. For example, if the endpoints of line are (x₁, y₁) (x₂, y₂) then the **b = y₁-mx₁** or **b=y₂-mx₂**. The following algorithm is used this method below:

Step 1: dx=x₂-x₁: dy=y₂-y₁

Step 2: if (dx=0) that lead draw the *vertical line* & exit

Step 3: if (dy=0) that lead draw the *horizontal line* & exit

Step 4: m=dy/dx: b=y₁-mx₁ {OR b=y₂-mx₂}

Step 5: for x=x₁ to x₂

Step 5.1: y=m*x+b

Step 5.2: plot point (x, y), color

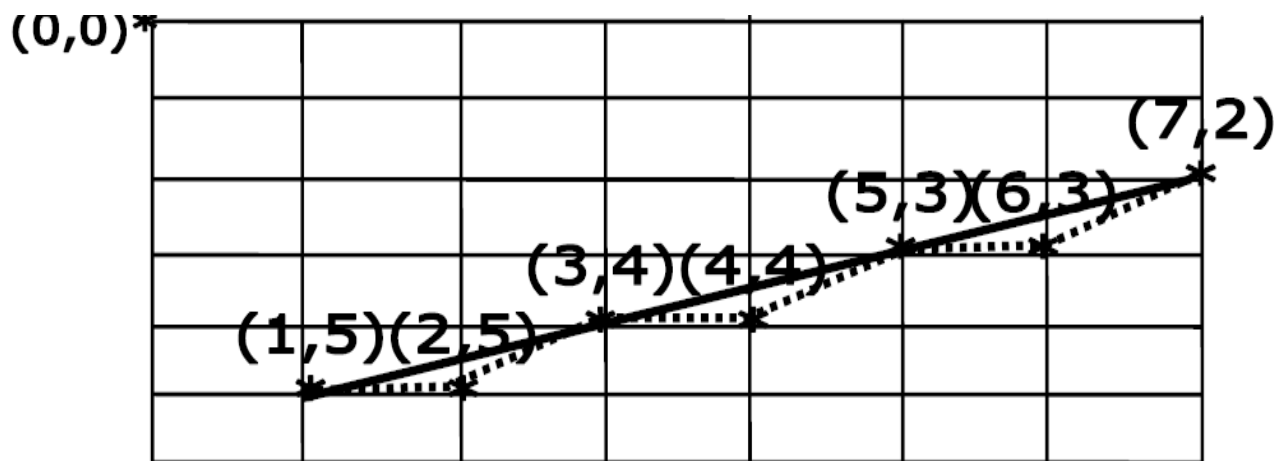
Step 6: next x {goto Step 5 where un-finish}

Example: trace the line equation to draw the line that endpoints are (1, 5), (7, 2), and draw the line in screen coordinate.

dx=7-1=6; dy=2-5= -3; m= -3/6= -1/2 =-0.5; b=5- (-0.5)*1=5.5

$dx > 0$ then increment of X

X	Y	Point (x, y)	Plot in screen
1	$1 * -0.5 + 5.5$	(1,5)	(1,5)
2	$2 * -0.5 + 5.5$	(2,4.5)	(2,5)
3	$3 * -0.5 + 5.5$	(3,4)	(3,4)
4	$4 * -0.5 + 5.5$	(4,3.5)	(4,4)
5	$5 * -0.5 + 5.5$	(5,3)	(5,3)
6	$6 * -0.5 + 5.5$	(6,2.5)	(6,3)
7	$7 * -0.5 + 5.5$	(7,2)	(7,2)



H.W/

1. Tracing same example but endpoints (7, 2) (1, 5).
2. Tracing the line (-8, -2), (4, 7) and draw line in coordinate system (4quartered)

DDA Algorithm

The simple **Digital Differential Analyzer** is a line drawing algorithm that generates line from their differential equations. It work on the principle of simultaneously incrementing x and y by small steps proportional to the first derivatives of x and y. The slop a line between the two points (x1,y,) ,(x2,y2) is given by $m=(y_2-y_1)/(x_2-x_1)$. The algorithm starts with the initial values $x=x_1$ & $y=y_1$, the coordinates are then incremented by. y and .x respectively to find the next points coordinates. The value of x and y are rounded to integers and a pixel is set at that point. This step is repeated until the second end point (x2, y2) is reached.

Step 1: Get the input of two end points (X1, Y1) and (X2, Y2)

Step 2: Calculate the difference between two end points.

$$dx = X_2 - X_1$$

$$dy = Y_2 - Y_1$$

Step 3: Based on the calculated difference in step-2, you need to identify the number of steps to put pixel. If $dx > dy$, then you need more steps in x coordinate; otherwise in y coordinate.

if (abs(dx)>abs(dy)) then length=abs(dx)

else length=abs(dy)

Step 4: Calculate the increment in x coordinate and y coordinate.

$$X_{\text{increment}} = dx / \text{length};$$

$$Y_{\text{increment}} = dy / \text{length};$$

Step 5: $x=x_1$; $y=y_1$

Step 6: Put the pixel by successfully incrementing x and y coordinates accordingly and complete the drawing of the line.

for i=0 to length

Step 6.1: plot pixel (x, y)

Step 6.2: $x = x + x_{inc}$: $y = y + y_{inc}$

Step 7: next i

Example:

Trace the simple DDA algorithm to draw a line between the two points (23, 33), (29,40).

Solution:

$$dx = 29 - 23 = 6$$

$$dy = 40 - 33 = 7$$

$$\text{Length} = 7$$

$$X_{inc} = 6/7 = 0.857$$

$$Y_{inc} = 7/7 = 1.$$

<i>X</i>	<i>Y</i>	<i>Plot(X)</i>	<i>Plot(Y)</i>
23	33	23	33
23.857	34	24	34
24.714	35	25	35
25.571	36	26	36
26.429	37	26	37
27.286	38	27	38
28.143	39	28	39

H.W/

Trace the simple DDA algorithm to draw a line between the two point (29, 40), (23, 33).

Bresenham's Line Algorithm

This algorithm is designed so that each iteration changes one of the coordinate values by ± 1 . The other coordinate may or may not change depending on the value of an error term maintained by the algorithm. This error term record the distance measure perpendicular to the axis of greatest movement, between the exact path of the line and the actual dots generated.

If the x-axis is the axis of greatest movement, then at each iteration the x coordinates of the line is incremented, and the slop of the line by dy/dx is added to the error term. Whether to increment the y coordinate of the current point. A positive e value indicates that the exact path of the line lies above the current point, therefore they coordinate is incremented, and 1 is decremented from e. If e is negative the y coordinate value is left unchanged. And verse vise if y-axis is greatest movement.

Impartment note: -If X-axis is greatest movement the initial $e = dy/dx$ But If Y-axis is greatest movement the initial $e = dx/dy$

This algorithm is X-axis greatest movement and slop is positive

Step 1: $dx=x_2-x_1$; $dy=y_2-y_1$; $e=(dy/dx)-0.5$; $x=x_1$; $y=y_1$;

Step 2: For $i=0$ to $|dx|$

Step 2.1: Plot pixel (x, y)

Step 2.2: If $(e>0)$ then if $(y_1>y_2)$ $y=y-1$

Else $y=y+1$

$e=e-1$

Step 2.3: if $(x_1>x_2)$ then $x=x-1$

else $x=x+1$

Step 2.4: $e=e+ (dy/dx)$

Step 3: next i

Example:

Trace the Bresenham algorithm to draw a line between the two points $(50, 65)$, $(59, 68)$.

Solution:

$$dx= 59-50= 9$$

$$dy= 68-65= 3$$

$$m= dy/dx =3/9 = 0.333$$

$$e=0.333 - 0.5 = -0.167$$

X	Y	e	
50	65	-0.167	$+m$
51	65	0.166	$-1+m$
52	66	-0.501	$+m$
53	66	-0.168	$+m$
54	66	0.165	$-1+m$
55	67	-0.502	$+m$
56	67	-0.169	$+m$
57	67	0.164	$-1+m$
58	68	-0.503	

Note: this algorithm uses in **slop** of line positive but in **negative** slop replaces $e > 0$ to $e < 0$, and $e = e - 1$ to $e = e + 1$ and coordinate $y = y + 1$ to $y = y - 1$ or $x = x + 1$ to $x = x - 1$ to obtain algorithms:

Step 1: $dx = x_2 - x_1$; $dy = y_2 - y_1$; $e = (dy/dx) + 0.5$; $x = x_1$; $y = y_1$;

Step 2: For $i = 0$ to $|dx|$

Step 2.1: Plot pixel (x, y)

Step 2.2: If ($e < 0$) then if ($y_1 > y_2$) $y = y - 1$

Else $y = y + 1$

$e = e + 1$

Step 2.3: if ($x_1 > x_2$) then $x = x - 1$

else $x = x + 1$

Step 2.4: $e = e + (dy/dx)$

Step 3: next i

Example:

Trace the Bresenham's algorithm to draw a line between the two points (1, 5), (7, 2).

Solution:

$dx = 7 - 1 = 6$;

$dy = 2 - 5 = -3$

$m = dy/dx = -3/6 = -0.5$

{Negative slope} Uses algorithm modify of Bresenham's

$e = -0.5 + 0.5 = 0$

Note:- try $e = -0.5$

X	Y	e
1	5	0 $+m$
2	5	-0.5 $+1+m$
3	4	0 $+m$
4	4	-0.5 $+1+m$
5	3	0 $+m$
6	3	-0.5 $+1+m$
7	2	0

Example:

Trace the line where end points (0, 3), (2, -2) by Bresnham's Method

Solution:

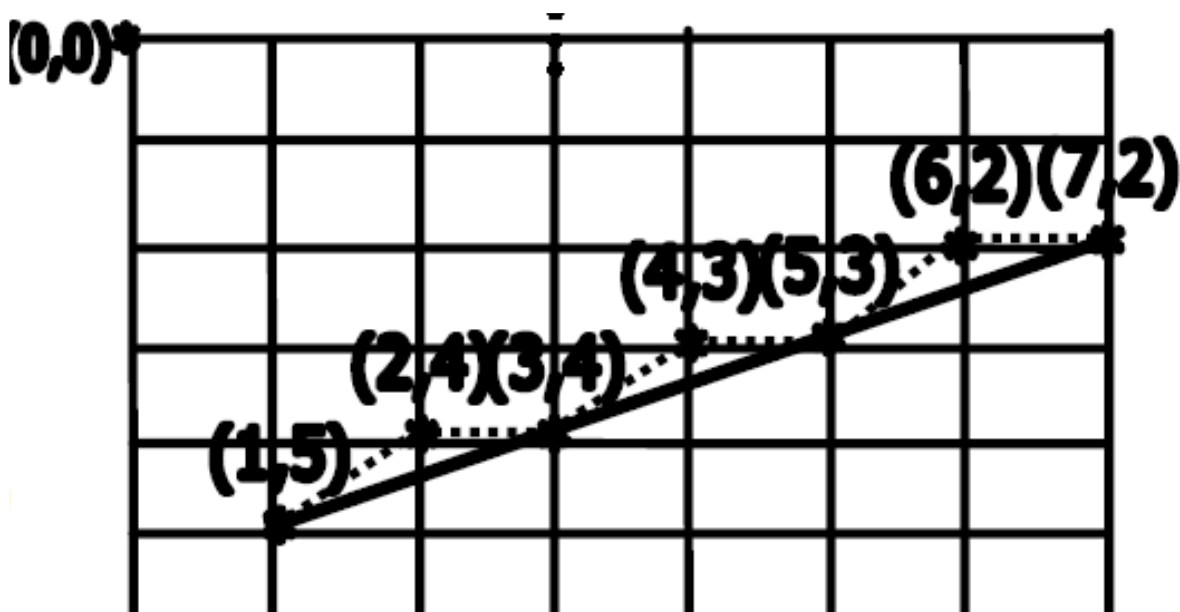
$$dx = 2 - 0 = 2, dy = -2 - 3 = -5,$$

$|dy| > |dx| \Rightarrow Y$ is Greatest move: Change for each step,

$$m = (dx/dy) = 2/-5 = -0.4 \rightarrow$$

$m < 0$ then $e < 0$ then change X otherwise un-change X because less move,

$$e = (dx/dy) + 0.5 = 0.1, X = 0, Y = 3$$



$dx > 0$ then increment of X
 $dy < 0$ then decrement of Y

X	Y	e	Status of e
0	3	0.1	$+(dx/dy)$
0	2	-0.3	$+1+(dx/dy)$
1	1	0.3	$+(dx/dy)$
1	0	-0.1	$+1+(dx/dy)$
2	-1	0.5	$+(dx/dy)$
2	-2	0.1	Finish

Note:- need Trace by $| \text{different Greatest move} | + 1$

Circle Drawing Algorithms

1- Direct Algorithm

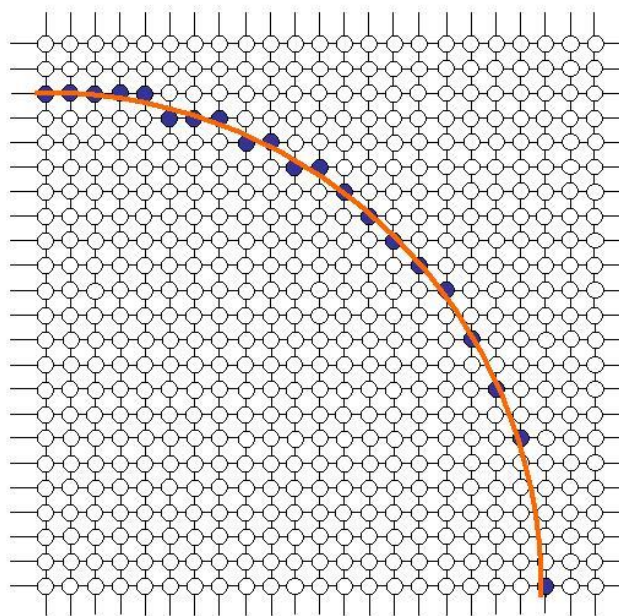
A circle is specified by the coordinate of its centre (x_c, y_c) and its radius(r). If the centre of the circle is at the origin (0,0) the equation is:

$$x^2 + y^2 = r^2$$

Where (0, 0) is the origin r is the **radius** of the circle and (x, y) is any point on the circle. So, we can write a direct circle drawing algorithm by solving the equation for y at unit x intervals using:

$$y = \pm \sqrt{R^2 - x^2}$$

Note: To draw a quarter circle, we can increment x by one unit form $-r$ to r and so the above equation to solve for the two y value at each step (convert to integer).



The algorithm:

1: **x:** = -**r**: **r** (increment unit in circle)

2: **While** ($x \leq r$)

2.1: $y1 = +\text{sqrt}(r*r - x*x)$; $y2 = -\text{sqrt}(r*r - x*x)$;

2.2: *plot pixel* ($x_c + x, y_c + y1$): *put pixel* ($x_c + x, y_c + y2$)

2.3 $x = x + 1$

3: **go** to 2

Example 1// Trace the Direct Algorithm to generate the points of the circle centred at (100,-20) with a radius equal to 6 unit.

$X(-6 \text{ to } 6)$	$Y1 = +\text{Sqrt}(r^2 - x^2)$	$Y2 = -\text{Sqrt}(r^2 - x^2)$	$X_C + X$	$Y_C + Y1$	$Y_C + Y2$
-6	0	0	94	-20	-20
-5	$Y1 = +\text{Sqrt}(6^2 - (-5)^2)$	$Y2 = -\text{Sqrt}(6^2 - (-5)^2)$	95	$-20 + \text{Sqrt}(11)$	$-20 - \text{Sqrt}(11)$
	H.W				
	H.W				
6	H.W				

H.W// Trace the Direct Algorithm to generate the points of the circle centred at (20, 10) with a radius equal to 8 unit.

2- DDA Algorithm

To write an algorithm to generate a circle of the form $(x-a)^2+(y-b)^2=r^2$ by the help of digital differential analyser where (a, b) is the centre of the circle and r is the radius.

The Algorithm:

1. START

2. Get the values of the centre (a, b) and radius (r) of the circle.

3. Find the polar co-ordinates by

$$x=a+r\cos\theta$$

$$y=b+r\sin\theta$$

4. Plot the points (x, y) corresponding to the values of θ , where θ lies between 0 and 360.

5. STOP

Example 2// Trace the DDA algorithm to generate the points of the circle centred at (100, 20) with a radius equal to 5 unit and theta between 0 & 7. (**Rad**)

<i>theta</i>	<i>X</i>	<i>Y</i>	<i>Plot (X,Y)</i>
0	105	20	(100,20)
1	102.7	24.2	(103,24)
2	97.7	24.5	(98,25)
3	95	20.7	(95,21)
4	96.7	16.2	(97,16)
5	101.4	15.2	(101,15)
6	104.8	18.6	(105,19)
7	103.7	23.2	(104,23)

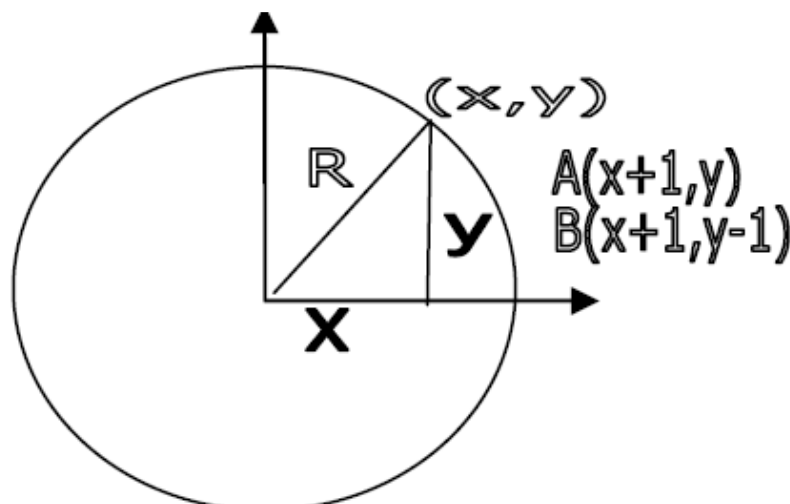
3- Bresenham Circle Algorithm:

The values of a circle centered at the origin are computed in a 45 sector from $x=0$ to $x=y$ the remaining seven sectors are obtained from the eight point symmetry of the circle. The values for this sector decrease as the x values increase if $(0, r)$ is the starting point of the algorithm, then as x increase by one unit the y value either remains the same or is decrease by one unit. If (x, y) is a pixel on the circle, the next pixel is either A or B.

A: $(x+1, y)$; to the right of previous point

B: $(x+1, y-1)$; down and to the right of the previous point.

The algorithm proceeds to choose Pixel A or B by finding and comparing the distance from each pixel to the point on the circle that has x value of $(x+1)$ these distances measure how far from the circle each pixel is the pixel with the smallest distances the best approximation on the circle. The square of the distance of pixel A from the center of the circle is $(x+1)^2 + y^2$.



The difference between this squared distance and the squared distance to the closest point on the circle is:

$$d(A) = (x+1)^2 + y^2 - r^2$$

for pixel B the distance is : $d(B) = (x+1)^2 + (y-1)^2 - r^2$.

A point lies on the circle if its d value equals 0, in order to determine which pixel A or B has the smallest d value we examine the sum $S = d(A) + d(B)$; if $S > 0$ we chose pixel B otherwise we choose pixel A.

The Algorithm:

1: $x=0; y=r;$

2: While($x \leq y$)

2.1: plot pixel($x_c + x, y_c + y$), 1 AND Symmetric 7 point($x_c + x, y_c + y$)

2.2: $da = (x+1)^2 + (y)^2 - (r)^2$

2.3: $db = (x+1)^2 + (y-1)^2 - (r)^2$

2.4: $s = da + db$

2.5: if ($s > 0$) $y = y - 1$

2.6 $x = x + 1$

3. Wend 'goto 2

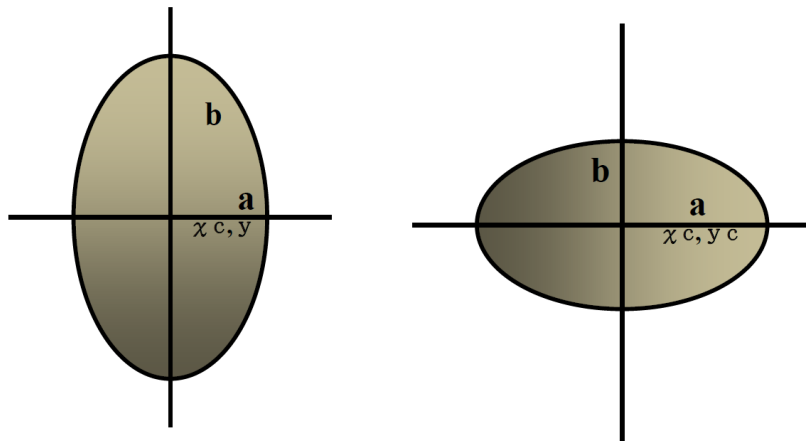
Example 3// Trace the Bresenham algorithm to generate six points of the circle centred at (300,150) with a radius equal to 9 unit.

Sol. / $da = (x+1)^2 + y^2 - R^2$, $db = (x+1)^2 + (y-1)^2 - R^2$ $R^2 = 81$

x	y	x +xc	y +yc	d(A)	d(B)	s
0	9	300	159	1	-16	-15
1	9	301	159	4	-13	-9
2	9	302	159	9	-8	1
3	8	303	158	-1	-16	-17
4	8	304	158	8	-7	1
5	7	305	157	4	-9	-5

Drawing Ellipses

An ellipse is a variation of a circle. Stretching a circle in one direction produces an ellipse, we shall examine ellipse that are stretched in the X or Y direction.



DDA Algorithm

To write an algorithm to generate an ellipse using the Digital Differential Analyzer Algorithm (DDA).

Equation to the ellipse is

$$((x-x_c)/r_x)^2 + ((y-y_c)/r_y)^2 = 1$$

where (x_c, y_c) - center of the ellipse.

r_x - x radius of ellipse, r_y - y radius of ellipse.

The Algorithm:**1. START**

2. Get the centre (xc,yc),x radius (rx) and y radius (ry) of the ellipse.

3. The polar co-ordinates on an ellipse are

$$x = xc + rx \cos \theta$$

$$y = yc + ry \sin \theta$$

4. Plot the point (x, y) corresponding to the values of θ where $0 \leq \theta \leq 360$

5. STOP

Example 1// Trace the DDA algorithm to generate the points of the ellipse centred at (300, 150) with xr=10, yr=5 and theta between 0 & 7. **(Rad)**

Sol:

Get the centre (300,150)

x radius (rx)=10 and y radius (ry)=5

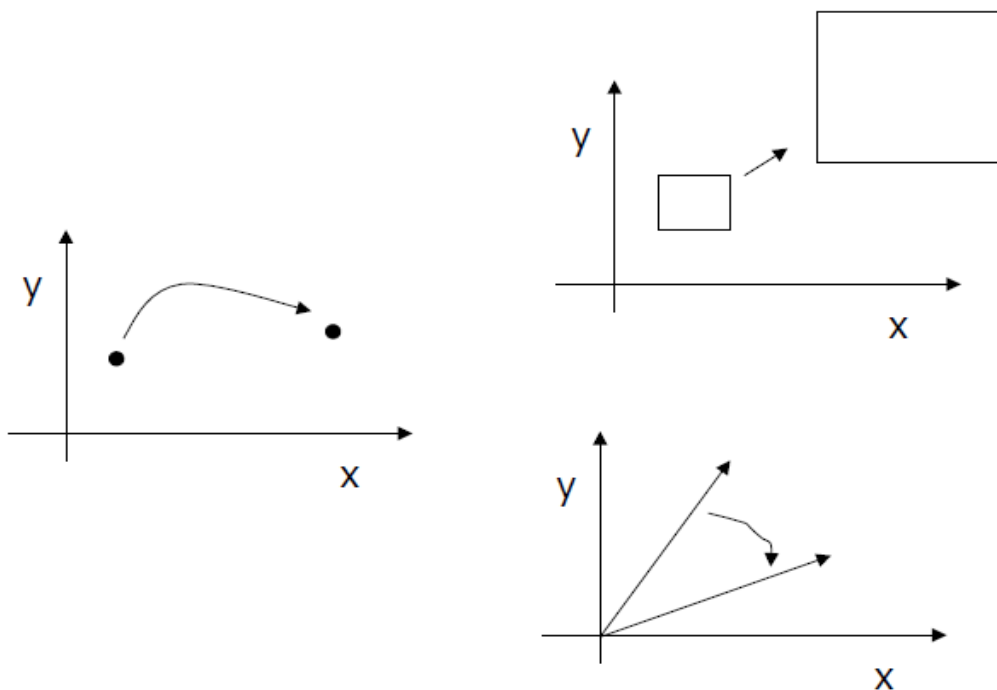
$$x = xc + rx \cos \theta$$

$$y = yc + ry \sin \theta$$

<i>theta</i>	<i>X</i>	<i>Y</i>	<i>Plot (X,Y)</i>
0	310	150	(310,150)
1	305.4	154.2	(305,154)
2	295.8	154.5	(300,155)
3			
4			
5			
6			
7			

Two Dimensional Geometric Transformations

Two-dimensional transformation is the basic transformation that can be done on the image. Performing appropriate geometric or coordinate transformation on the object does the manipulation of image. Using transformation, we can change the size of polygon or any object, change its position, and rotate it easily. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called 2D transformation. Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.



Fundamental Transformation

The 5 common transformations that are used in computer graphics are:

- 1. Translation (shift OR move).**
- 2. Size (Scaling).**
- 3. Orientation (Rotation).**
- 4. Reflection.**
- 5. Shapes (Shear).**

1. Translation

Consider a point $P(x, y)$. We can translated it means shift it to new position $p'(x', y')$ by adding tx and ty in y where T_x and T_y are translating factor.

Mathematically this can be represented as:

$$x' = x + T_x$$

$$y' = y + T_y$$

Note1: Using coordinate system the translating factor are

If $T_x > 0$ then point moves to the right.

If $T_x < 0$ then point moves to the left.

If $T_y > 0$ then point moves to the up.

If $T_y < 0$ then point moves to the down.

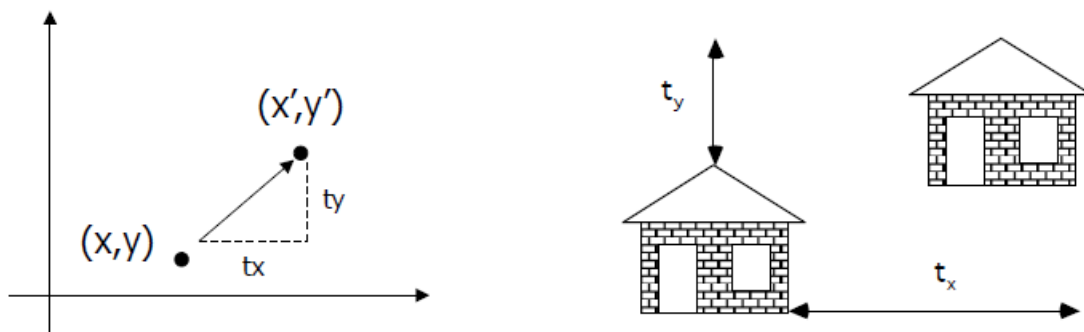
Note2: using coordinate system in screen then translating factor are

If $T_x > 0$ then point moves to the right.

If $T_x < 0$ then point moves to the left.

If $T_y > 0$ then point moves to the down.

If $T_y < 0$ then point moves to the up.



The pair (t_x, t_y) is called the translation vector or shift vector. The above equations can also be represented using **the column vectors**.

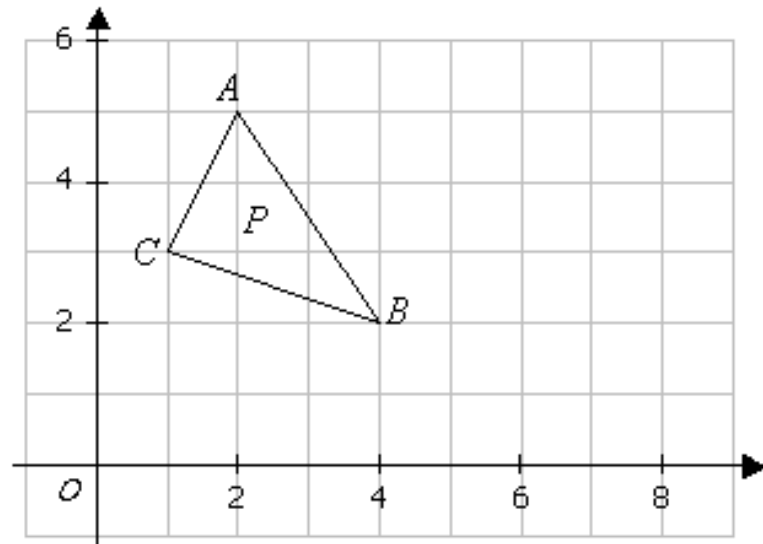
In general, a translation can be represented by a **column matrix or column vector** $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$ where t_x is the number of units to move right or left along the x -axis and t_y is the number of units to move up or down along the y -axis.

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

where $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$ is the translation matrix and $\begin{bmatrix} x' \\ y' \end{bmatrix}$ is the image of $\begin{bmatrix} x \\ y \end{bmatrix}$

Example 2//

The triangle P is mapped onto the triangle Q by the translation $\begin{pmatrix} 4 \\ -1 \end{pmatrix}$.



a) Find the coordinates of triangle Q .

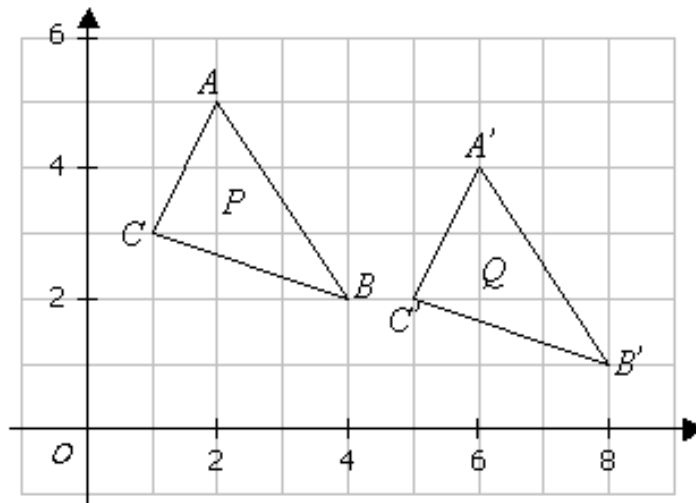
Solution:

$$A' = \begin{pmatrix} 2 \\ 5 \end{pmatrix} + \begin{pmatrix} 4 \\ -1 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

$$B' = \begin{pmatrix} 4 \\ 2 \end{pmatrix} + \begin{pmatrix} 4 \\ -1 \end{pmatrix} = \begin{pmatrix} 8 \\ 1 \end{pmatrix}$$

$$C' = \begin{pmatrix} 1 \\ 3 \end{pmatrix} + \begin{pmatrix} 4 \\ -1 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$$

b) On the diagram, draw and label triangle Q .



As a mathematical notation, we may write: $T(A) = B$, to mean object A is mapped onto B under the transformation T .

2D Translation using a 3x3 Matrix

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

Inverse of 2D Translation is inverse of 3x3 matrix

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

Example 3//

to translate a triangle with vertices at original coordinates (10,20), (10,10), (20,10) by $t_x=5$, $t_y=10$, we compute as followings:

Translation of vertex (10, 20):

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

Translation of vertex (10, 10):

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

Translation of vertex (20, 10):

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

The resultant coordinates of the triangle vertices are (15, 30), (15,20), and (25,20) respectively.

H.W//

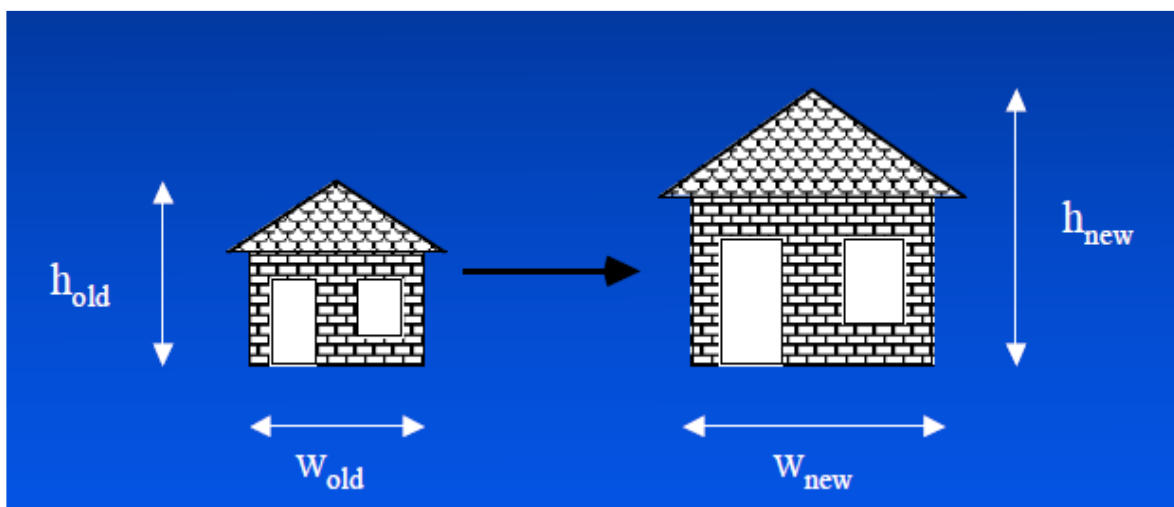
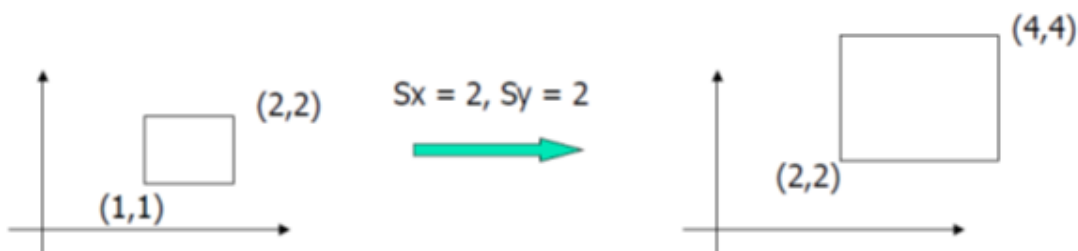
Translate a triangle with vertices at original coordinates (10, 25), (5,10), (20,10) by $t_x=15$, $t_y=5$. Roughly plot the original and resultant triangles.

2. Scaling

A scaling transformation alters size of an object. In the scaling process, we either compress or expand the dimension of the object. Scaling operation can be achieved by multiplying each vertex coordinate (x, y) of the polygon by scaling factor s_x and s_y to produce the transformed coordinates as (x', y') . So, $x' = x * s_x$ and $y' = y * s_y$. The scaling factor s_x, s_y scales the object in X and Y direction respectively. So, the above equation can be represented in matrix form:

$$\begin{array}{l} x' = x \cdot S_x \\ y' = y \cdot S_y \end{array} \Rightarrow \begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$

Example:



2D Scaling using a 3x3 Matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example 4//

For example, to scale a triangle with respect to the origin, with vertices at original coordinates (10, 20), (10, 10), (20, 10) by $s_x=2$, $s_y=1.5$, we compute as followings:

Scaling of vertex (10, 20):

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

Scaling of vertex (10, 10):

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

Scaling of vertex (20, 10):

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

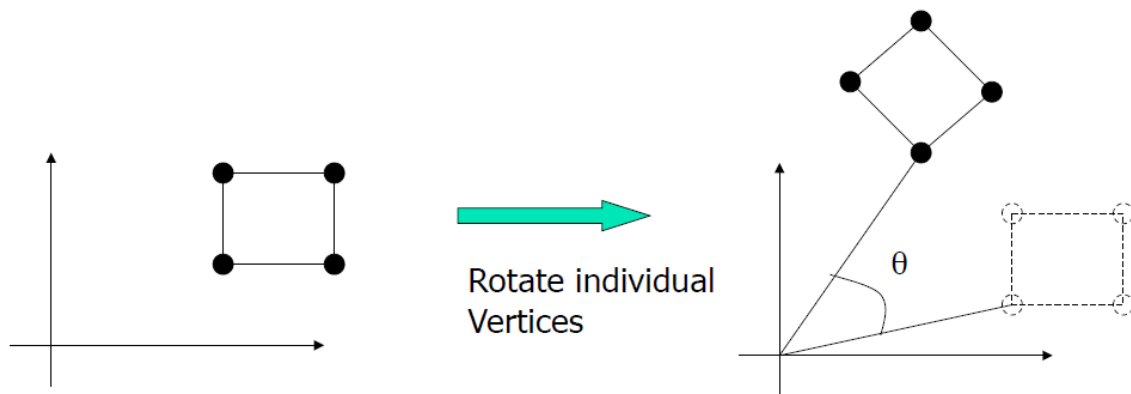
The resultant coordinates of the triangle vertices are (20, 30), (20, 15), and (40, 15) respectively.

H.W:

Scale a triangle with vertices at original coordinates (10, 25), (5,10), (20,10) by $s_x=1.5, s_y=2$, with respect to the origin. Roughly plot the original and resultant triangles.

3. Rotation

Rotation can be performed about the origin or about a pivot point. But the rotate direction either positive oriented (anticlockwise) where angle θ (theta) is positive or negative oriented (clockwise) where angle is negative

**A. Rotation about the origin**

Mathematically the counter anticlockwise rotation of a point (x, y) about the origin an angle θ . To produce the point (x^R, y^R) is represented by:

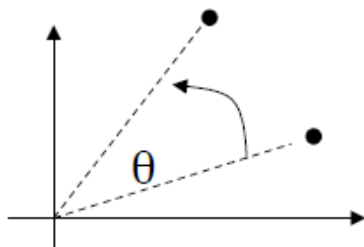
$$X_R = X \cos(\theta) - Y \sin(\theta)$$

$$Y_R = Y \cos(\theta) + X \sin(\theta)$$

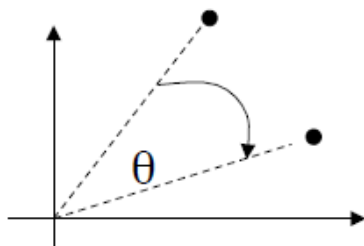
Mathematically, counter anticlockwise is considered the positive rotation direction, to rotate in a clockwise direction change to angle θ to $-\theta$. Then:
 $\cos(-\theta) = \cos(\theta)$, $\sin(-\theta) = -\sin(\theta)$; clockwise rotation can be represent by:

$$X_R = X \cos(\theta) + Y \sin(\theta)$$

$$Y_R = Y \cos(\theta) - X \sin(\theta)$$



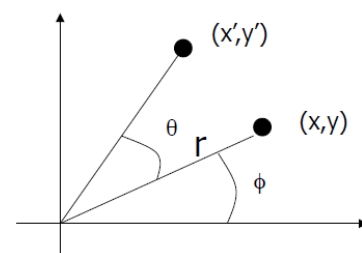
$\theta > 0$: Rotate counter clockwise



$\theta < 0$: Rotate clockwise

2x2 2D Rotation Matrix:

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



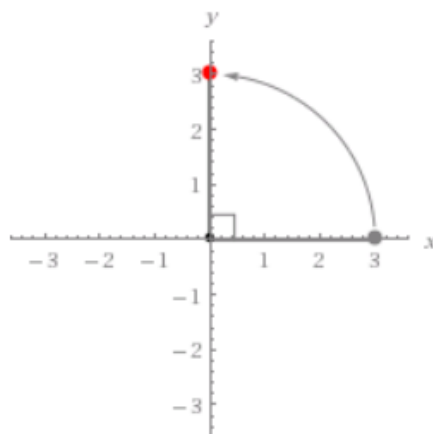
Example 5//

To rotate a vector 90 degrees counterclockwise is done as follows:

$$\vec{u} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

**2D Rotation using a 3x3 Matrix:**

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

Example 6//

To rotate a triangle about the origin with vertices at original coordinates (10, 20), (10, 10), (20, 10) by 30 degrees, we compute as followings:

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

Rotation of vertex (10, 20):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 * 10 + (-0.5) * 20 + 0 * 1 \\ 0.5 * 10 + 0.866 * 20 + 0 * 1 \\ 0 * 10 + 0 * 20 + 1 * 1 \end{bmatrix} = \begin{bmatrix} -1.34 \\ 22.32 \\ 1 \end{bmatrix}$$

Rotation of vertex (10, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 * 10 + (-0.5) * 10 + 0 * 1 \\ 0.5 * 10 + 0.866 * 10 + 0 * 1 \\ 0 * 10 + 0 * 10 + 1 * 1 \end{bmatrix} = \begin{bmatrix} 3.66 \\ 13.66 \\ 1 \end{bmatrix}$$

Rotation of vertex (20, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 * 20 + (-0.5) * 10 + 0 * 1 \\ 0.5 * 20 + 0.866 * 10 + 0 * 1 \\ 0 * 20 + 0 * 10 + 1 * 1 \end{bmatrix} = \begin{bmatrix} 12.32 \\ 18.66 \\ 1 \end{bmatrix}$$

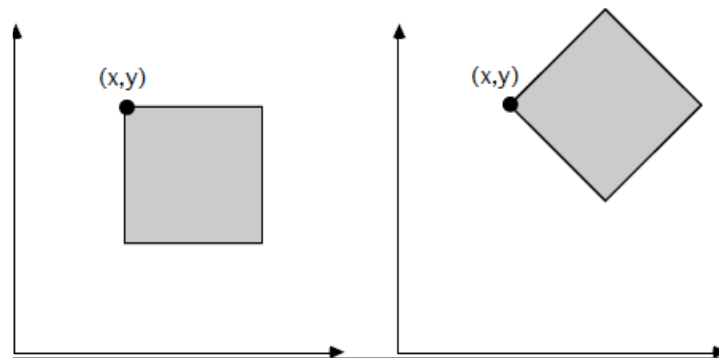
The resultant coordinates of the triangle vertices are (-1.34, 22.32), (3.6,13.66), and (12.32,18.66) respectively.

H.W:

Rotate a triangle with vertices at original coordinates (10, 20), (5,10), (20,10) by 45 degrees. Roughly plot the original and resultant triangles.

B. Rotation about a pivot point

After an object is rotated about a specified pivot point, it is still the same distance away from the pivot point but its orientation has been changed:



To rotate a point an angle θ about a pivot point three steps are required:-

1. Translate the pivot point (x_p, y_p) to the origin $(0, 0)$ this is done by translating the point (x, y) to the new point (x', y') where :-

$$x' = x - x_p, y' = y - y_p ;$$

2. Rotate the translated point (x', y') θ degree about the origin to obtain the new point (x^R, y^R) where $\mathbf{X}^R = \mathbf{X}' * \cos(\theta) - \mathbf{Y}' * \sin(\theta)$.

$$\mathbf{Y}^R = \mathbf{Y}' * \cos(\theta) + \mathbf{X}' * \sin(\theta).$$

3. Translate the center of rotation back to the pivot point (x_p, y_p) thus the point (x_g, y_g) is translated to: $\mathbf{x}^g = \mathbf{x}^R + x_p ; \mathbf{y}^g = \mathbf{y}^R + y_p ;$

These three steps can be combined in the following equation for the counterclockwise rotation of a point by an angle Θ about pivot point (x_p, y_p) .

$$\rightarrow X^g = (x - x_p) * \cos(\Theta) - (y - y_p) * \sin(\Theta) + x_p.$$

$$Y^g = (y - y_p) * \cos(\Theta) + (x - x_p) * \sin(\Theta) + y_p$$

4. Shearing

Shearing transformations makes the objects to distort their shapes in either x or y or both the directions. It can be imagined as the object is made up of very thin layers and they are slid over each other fixing the base in case of single directional shearing.

1. To **shear** in x direction only use shearing matrix in the equation as:

$$x' = x + shx * y$$

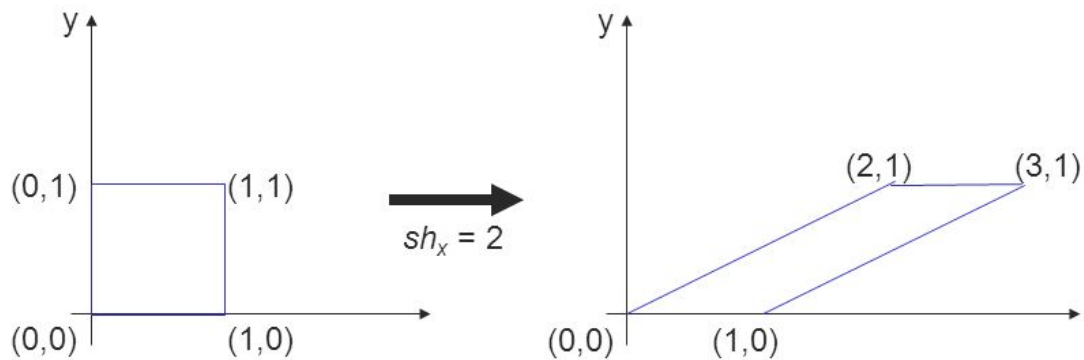
$$y' = y$$

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



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

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & -a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

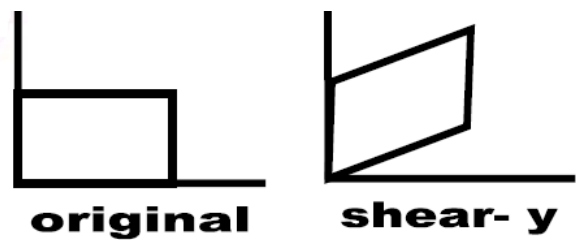
Example 7//

2. To **shear** in y direction only use shearing matrix in the equation as:

$$x' = x$$

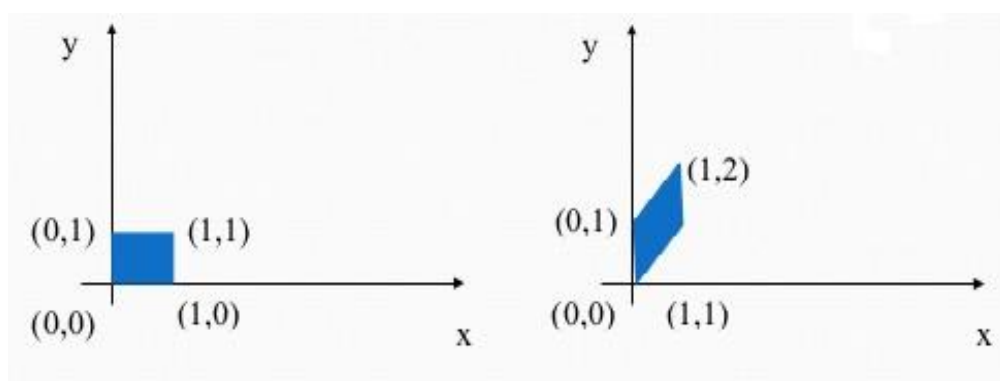
$$y' = x * sh_y + y$$

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



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


$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Example 8//

3. To shear in x

and y directions both :

$$x' = x + shx * y, \quad y' = x * shy + y;$$

$$\begin{bmatrix} 1 & shy & 0 \\ shx & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$


original shear- xy

5. Mirror reflection about an axis

Reflection is the mirror image of original object. In other words, we can say that it is a rotation operation with 180° . In reflection transformation, the size of the object does not change. The following figures show reflections with respect to X and Y axes, and about the origin respectively.

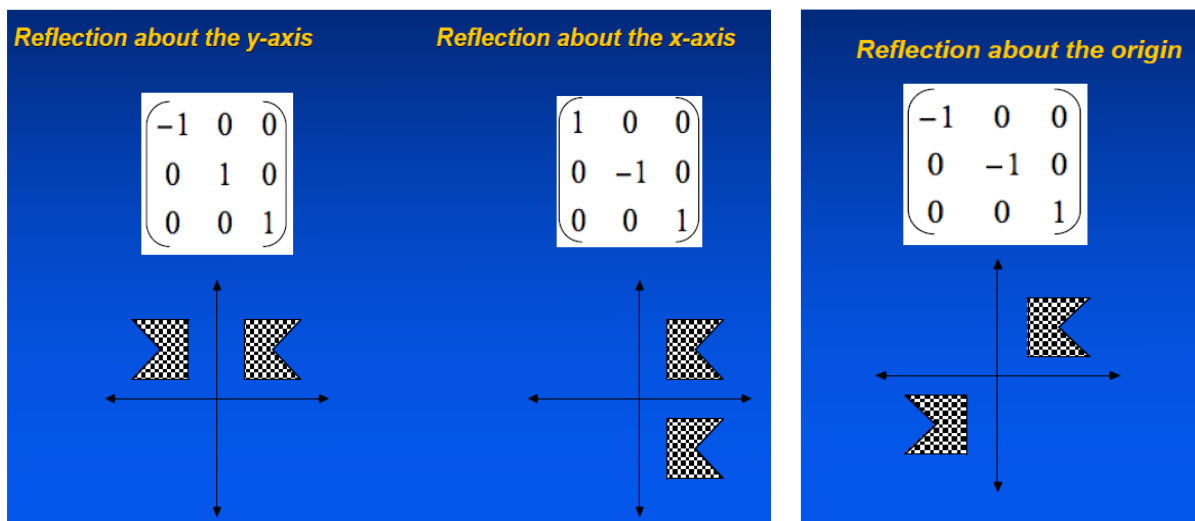
The **mirror** reflection about the x axis is: $xm=x$; $ym=-y$;

The **mirror** reflection about the y axis is: $xm=-x$; $ym=y$;

The **mirror** reflection about the origin point is: $xm=-x$; $ym=-y$;

The **mirror** reflection about the line $y=x$: $xm=y$; $ym=x$;

The **mirror** reflection about the line $y=-x$: $xm=-y$; $ym=-x$;



Matrix representation of transformations

Each of two dimensional transformations can be represented as a product of the row vector (x y 1) and a 3*3 matrix. The following are the matrix representation of the transformation.

- Translation:
$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} x \\ y \end{vmatrix} + \begin{vmatrix} tx \\ ty \end{vmatrix}$$

- Rotation:
$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} * \begin{vmatrix} x \\ y \end{vmatrix}$$

- Scaling:
$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} Sx & 0 \\ 0 & Sy \end{vmatrix} * \begin{vmatrix} x \\ y \end{vmatrix}$$

- Translation:
$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Rotation:
$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Scaling:
$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- *mirror reflection in x-axis:* $(x^{mx} \ y^{mx} \ 1) = (x \ y \ 1) * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- *mirror reflection in y-axis:* $(x^{my} \ y^{my} \ 1) = (x \ y \ 1) * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- *mirror reflection in origin point:* $(x^{mo} \ y^{mo} \ 1) = (x \ y \ 1) * \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Example 9//

Rotate the object defined by (43, 88), (84,50), (66,72) in a rotate at anticlock wise by angle 63 then scale it with the scaling factor $S_x=1$, $S_y=2$. where (65,40) is fixed point,

1. **Translate** the pivot point to the origin.

$$\begin{bmatrix} 43 & 88 & 1 \\ 84 & 50 & 1 \\ 66 & 72 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -65 & -40 & 1 \end{bmatrix} = \begin{bmatrix} -22 & 48 & 1 \\ 19 & 10 & 1 \\ 1 & 32 & 1 \end{bmatrix}$$

2. **Rotate** in an anticlockwise direction by 63.

$$\begin{bmatrix} -22 & 48 & 1 \\ 19 & 10 & 1 \\ 1 & 32 & 1 \end{bmatrix} * \begin{bmatrix} \cos(63) & \sin(63) & 0 \\ \sin(63) & \cos(63) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. **Scale** with $S_x=1$, $S_y=2$. (H.W)

4. **Translate** the origin back to the pivot point. (H.W)

H.W:

Rotate the object define by (54,68), (104,66), (70,102) by counter anticlockwise by 37 degree after scaling it with the scaling factor $S_x=3$, $S_y=2$ using the fixed point(54,68) and mirror reflection in origin point.

Thanks