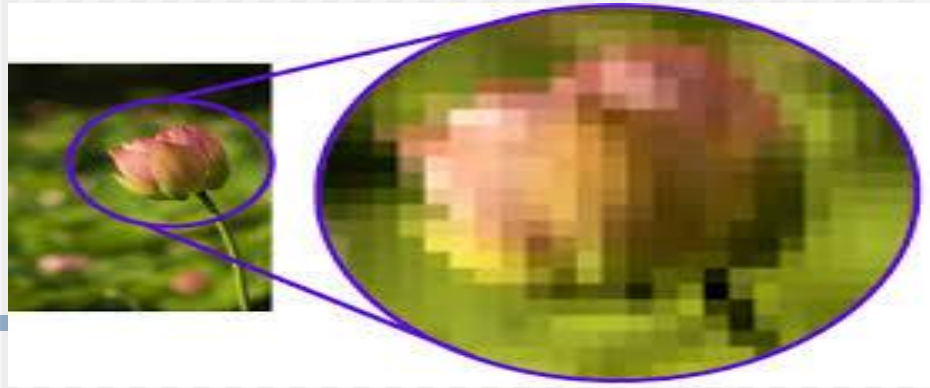
BCT 2405 Computer Graphical systems

Karanja Mwangi

Lesson objective

- **At the end of this class you will**
  - Appreciate the difference between vector and raster graphics
  - Identify the difference between vector and raster file formats
  - Understand the application areas of Vector and Raster graphics
  - Explore the 2D graphics Primitives – More on Raster graphics such Line drawing algorithms, Circle drawing algorithms

Recap: Pixels



- **Pixels**: we said it is a portmanteau of pix (from "**pictures**", shortened to "pics") and el (for "**element**") (picture elements) individual squares on a grid that makes up an image. Each square is made up of a color.
- A pixel is the smallest display element that makes up the images seen on a computer monitor or television

Raster Images vs. Vector Images

Raster (Bitmap) images

- A raster image is made up of **grid of pixels..** In raster images, the more pixels an image contains, the higher its resolution. For example, in a raster image a square is drawn as a grid of pixels (dots) and each of those pixels will have a specific color value.

- **Bitmap images lose their clarity when they are increased in size because the dots get bigger i.e cannot be scaled up without losing quality.**. (**resolution dependent**)

  - A raster image is resolution-dependant because it contains a fixed number of pixels that are used to create the image **.**

- **Common File Formats in Raster : Joint Photographers Expert Group (.JPEG, .JPG) Portable Network Graphics (.PNG)Graphic Interchange Format (.GIF).....TIFF, PSD etc**

- **Raster Editing Programs :AdobePhotoshop, PaintShop, and PhotoPaint all work with pixels (raster images). Gimp (opensource)**

Vector images

- A vector image is defined by objects which are made of lines and curves that are **defined mathematically** in the computer. Vectors can have various attributes such as line thickness, length and color. For example, in a vector image, a square is drawn as four lines connected at the corners. Those lines can be set to different thickness and colors..
- Vector graphics are **resolution-independent**. They can be made larger or smaller without any loss of quality to the image. Vectors can be printed at any size, on any output device, at any resolution, without losing detail and without altering the resolution of the image.
- Vector images are the best choice for typefaces, charts and graphs, drawings, and other graphics that must have sharp lines when scaled to various sizes.

- **Common File Formats in Raster llustrator (.AI), Encapuslated PostScript (.EPS), PostScript (.PS), Scalable vector Graphic .SVG etc**

- **Vectors Editing Programs : PowerPoint, Adobe Illustrator, Inkscape (open source) and Freehand**

Why and when to use Vector or Raster

- solid color objects, manipulated text or many small objects….then **vector** will be the choice
- drop shadows, or other 3D effects, texture or photographs …then **raster** is the best choice

- Both Illustrator AI and EPS formats allow users to place raster images within a vector file. The raster image is not converted to vector format; rather it is simply embedded in the vector file, and is rendered in raster format.

Why and when to use Vector or Raster -2

1) **Type**: does the file format store the type of image you want to use ?
2) **Portability**: can other people use images in this format?
3) **Color Depth**: does the format support the number of colors you need?
    1) **GIF supports 256 Colors (8-bit) , JPEG supports 16.7 M Colors – (24-bit), TIFF supports 16 M colors (24-bit)**
4) **Compression**: can make the file smaller, but it takes time to compress and decompress a file (**GIF supports Lossless compression**
5) **Transparency**: do parts of your image need to be transparent? **Like GIF, PSD supports transparency but JPEG do not**

1. Image Data Types [24-bit color]

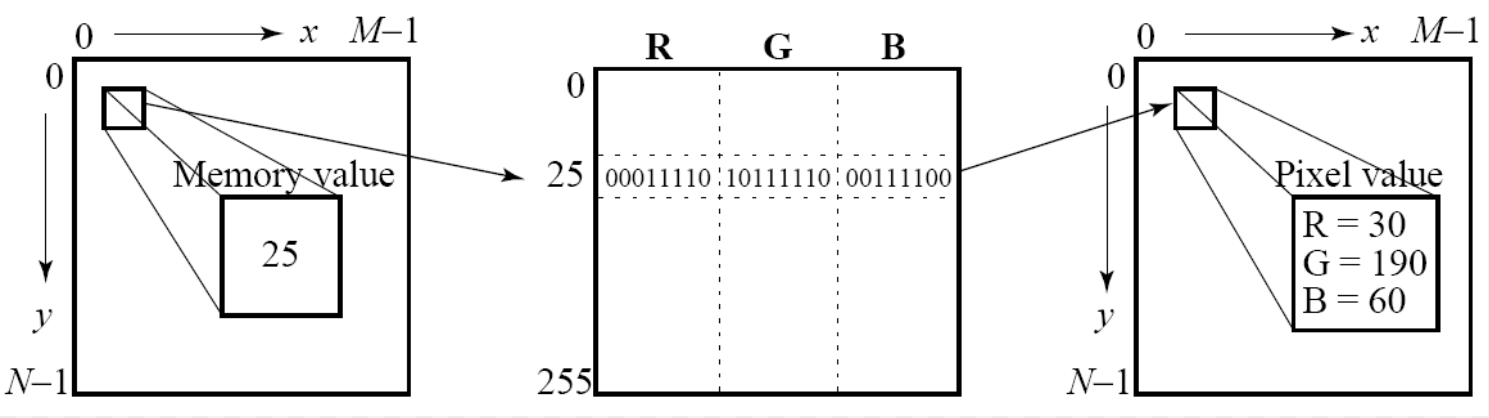
- In a color 24-bit image, each pixel is represented by three bytes, usually representing RGB.
  - - This format supports **256 x 256 x 256** possible combined colors, or a total of **16,777,216** possible colors.
- Many 24-bit color images are usually stored as 32-bit images, with the extra byte of data for each pixel used to store an ***alpha channel /value*** representing special effect information (e.g., transparency)
- Images now have two parts: The image and a mask, called the alpha channel
- GIMP and Photoshop support full transparency

1.Image Data Types -2 [8-bit color]

- 8 bits of color is also called (“256 colors”)
- In such images a concept of a **lookup table** to store color information is used.
  - - **Basically, the image stores not color, but instead just a set of bytes, each of which is actually an index into a table with 3-byte values that specify the color for a pixel with that lookup table index**.
- E.g Since humans are more sensitive to R and G than to B, we could shrink the R range and G range 0..255 into the 3-bit range 0..7 and shrink the B range down to the 2-bit range 0..3, thus making up a total of 8 bits.
- To shrink R and G, we could simply divide the R or G byte value by (256/8)=32 and then truncate. Then each pixel in the image gets replaced by its 8-bit index and the color LUT serves to generate 24-bit color.

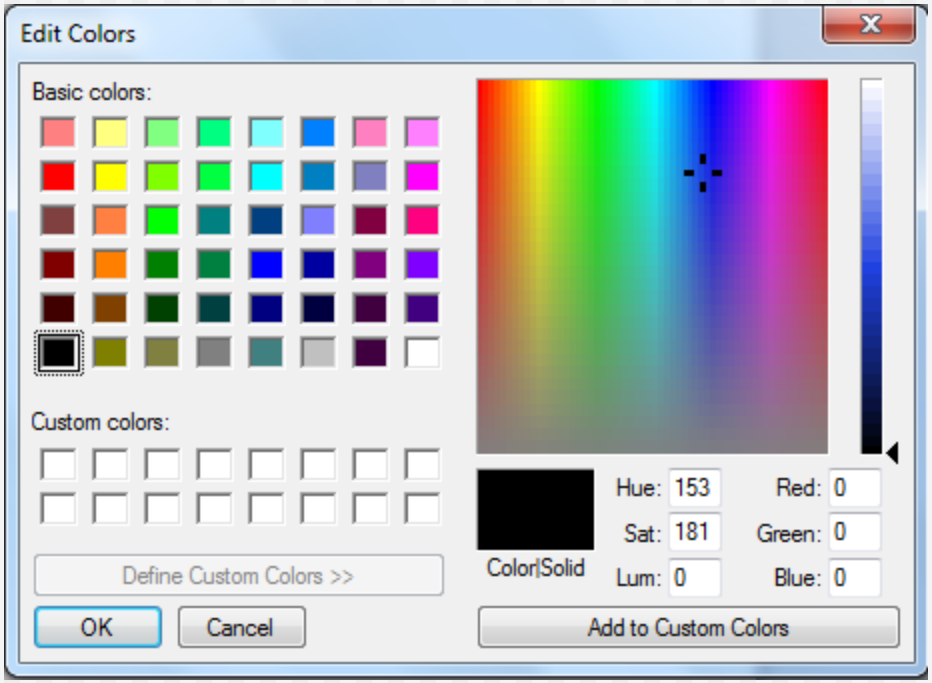
Color Look-up Tables (LUTs)

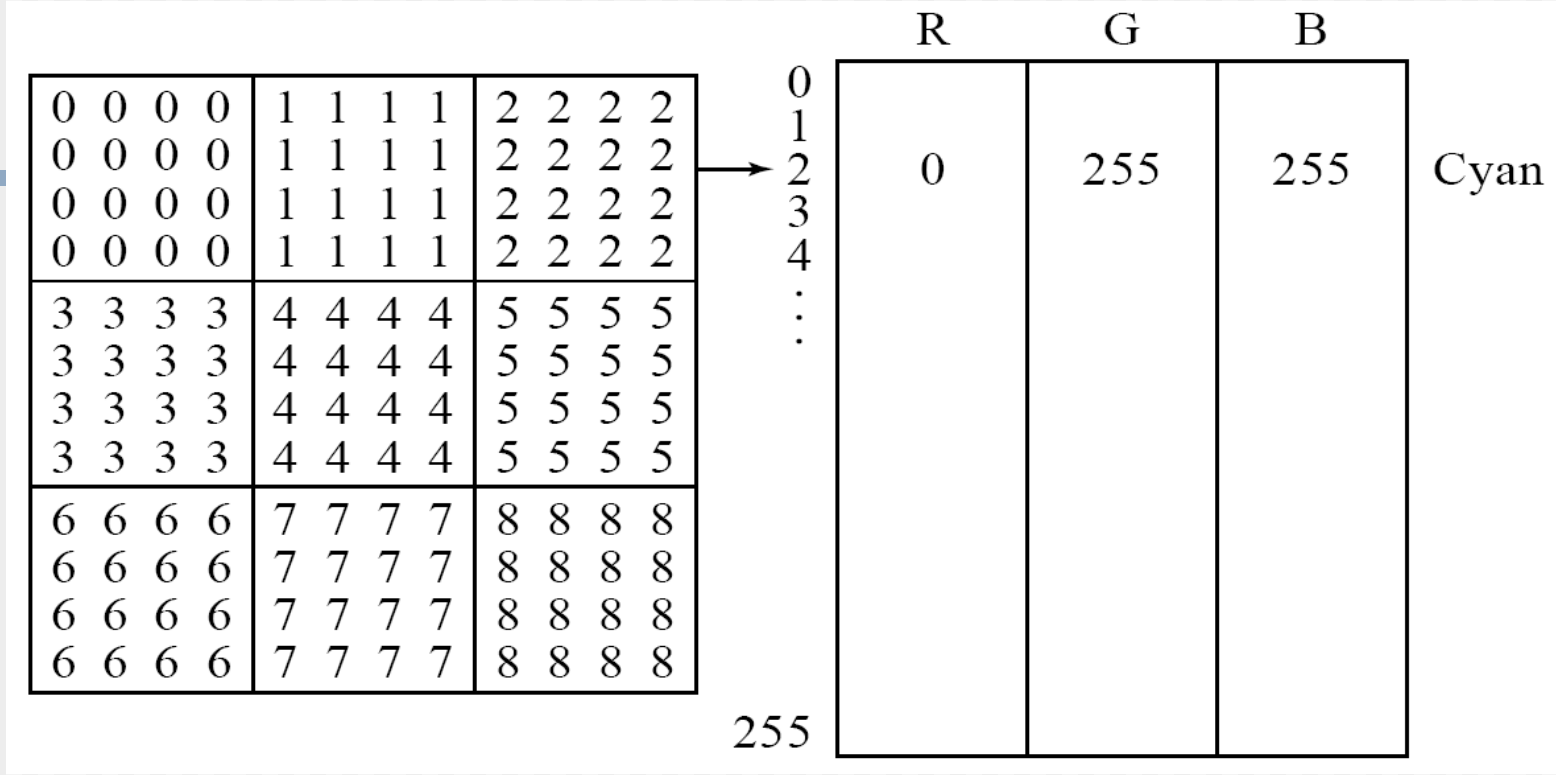
- The idea used in 8-bit color images is to store only the index, or code value, for each pixel. Then, e.g., if a pixel stores the value 25, the meaning is to go to row 25 in a color look-up table (LUT).



Color LUT for 8-bit color images.

- A **Color-picker** consists of an array of fairly large blocks of color (or a semi-continuous range of colors) such that a mouse-click will select the color indicated.
  - In reality, a color-picker displays the palette colors associated with index values from 0 to 255.
  - Figure below displays the concept of a color-picker: if the user selects the color block with index value 2, then the color meant is cyan, with RGB values (0, 255, 255).





Color-picker for 8-bit color: each block of the color-picker corresponds to one row of the color LUT

1.Image Data Types [24-bit color VS 8-bit color]

24-bit:

- Each pixel is represented by three bytes (e.g., RGB)
- Supports 256 x 256 x 256 possible combined colors (16,777,216)
- A 640 x 480 24-bit color image would require 921.6 KB of storage
- Many 24-bit color images are stored as 32-bit images, the extra byte of data for each pixel is used to store an alpha value representing special effect information

8-bit:

- One byte for each pixel
- Supports 256 out of the millions colors possible, acceptable color quality
- Requires Color Look-Up Tables (LUTs) -- *Pallete*
- A 640 x 480 8-bit color image requires 307.2 KB of storage (the same as 8-bit grayscale)

3.Color Depth

- Image formats will only allow a certain number of different colors for each pixel
- The format stores the level of each of the three values for RGB to specify color
- R,G,and B are referred to as *components*
  - ***Red Green Blue***
- *Bit depth* describes the number of colors that can be used in the image

3.Color Depth :Monochrome vs. Grayscale

Monochrome:

- Each pixel is stored as a single bit (0 or 1)
- A 640 x 480 monochrome image requires 37.5 KB of storage.

Grayscale:

- Each pixel is usually stored as a byte (value between 0 to 255)
- A 640 x 480 grayscale image requires over 300 KB of storage.

3.Color Depth :Color codes -RGB-Code

- Black: (0,0,0)
- White: (255,255,255)
- Red: (255,0,0)
- Green: (0,255,0)
- Blue: (0,0,255)

0-255 ….as intensity increases

Hexadecimal RGB-code of significant colours:

Black: #000000
White: #FFFFFF
Red: #FF0000
Green: #00FF00
Blue: #0000FF

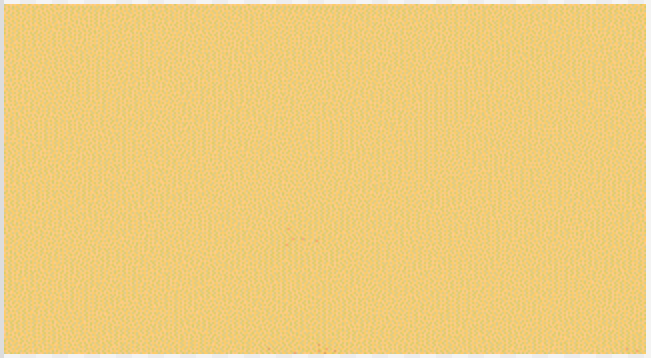
3.Color Depth -When image has more colors than the color depth?

- If the color is not in the palette , the Computer graphics software can choose a separate, close color, or do ***dithering***
- ***Dithering*** is where the program fools your eye by creating a pattern of pixels in two colors which makes you see a color between the two
  - *There are various forms of dithering and applications such as Photoshop allows user to select the type of dithering he wants*

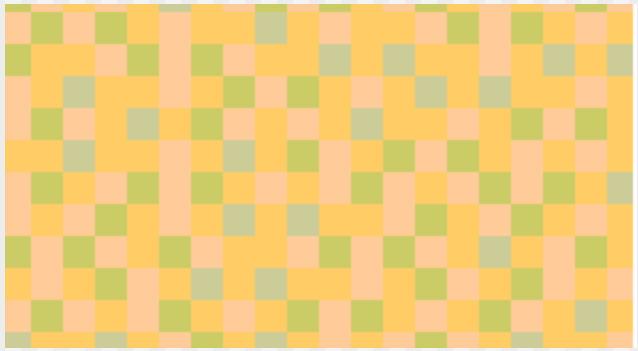
3.Color Depth -Example : Dithering with Color

Color I want to produce

Dithering with only 2 colors, orange and white

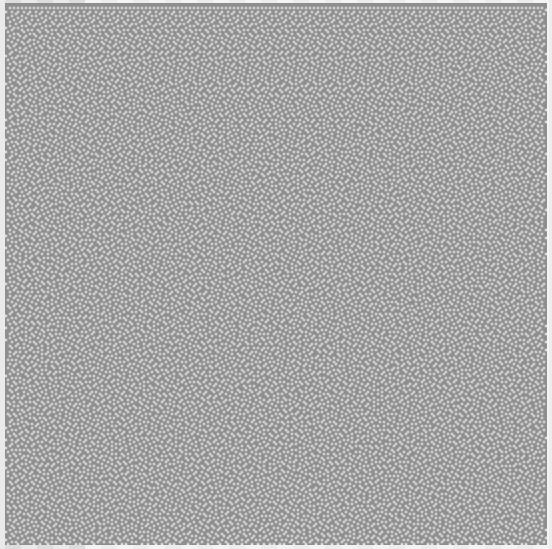


Dithering with all web-safe colors enabled

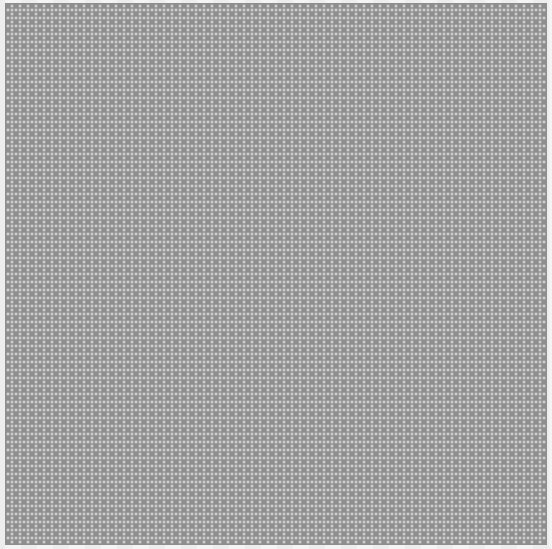


Zoomed View of dithering

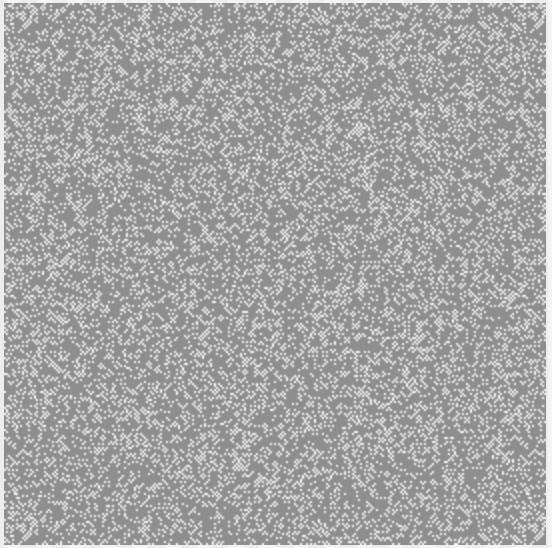
3.Color Depth- Photoshop Dithering Example



Diffusion



Pattern



Noise

3.Color Depth - Bit Depth

- Refers to the number of colors that have been used in an image.
- 24-bit image allows you to specify up to 2^{24} different colors in your image
  - For each of the RGB values you can specify 2^{8} different values (256 values)
  - 2^{8*3} values for R, for G, and for B gives us 2^{24} different colors
- The “24” in 24-bit says how many bits it takes to store a single color

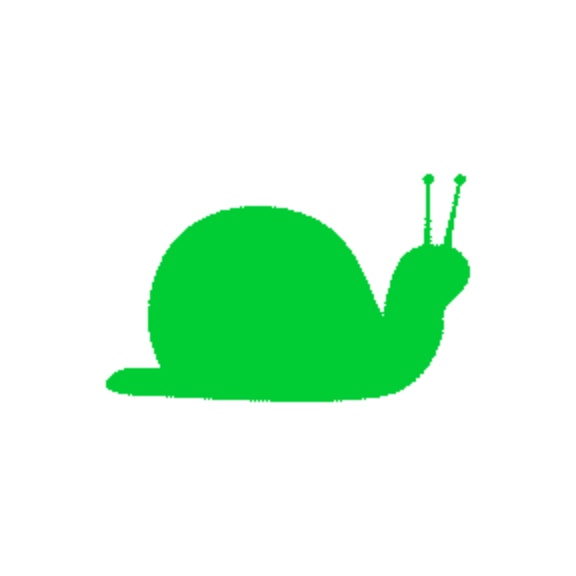
4. Compression

- *Compression* is used to make files smaller
- Bitmap images, especially, can be quite large if stored without compression
  - Ex. A 640 x 480 pixel image with 24-bit color depth would take:
    - 640 * 480*24 bits = 7372800 bits = 900kB
- Windows bmp format is uncompressed not good for usage in websites online as it takes time to load

4. Compression: Lossy vs. Lossless Compression

- There are two main types of compression:
  - *Lossy* and *Lossless*
- When an image is compressed and later uncompressed with a *lossless algorithm*, the image is exactly the same
  - Most common type
  - No information is lost
- *Lossy algorithms* produce images which may be slightly different before compression and after uncompressing
  - Some information may be lost
  - Can produce sometimes produce much smaller files than a lossless algorithm
  - Can be used when the difference isn't very noticeable

5. Transparency : Example of GIF Transparency



- GIF with a white background.
- Unsightly for web pages

- **GIF with a transparent background.**
- **Image is the same size as image on left**
- **But, background shows through**

The difference between Raster and Vector Graphics

Raster Graphics	Vector Graphics
They are composed of pixels.	They are composed of paths.
In Raster Graphics, refresh process is independent of the complexity of the image.	Vector displays flicker when the number of primitives in the image become too large.
Graphic primitives are specified in terms of end points and must be scan converted into corresponding pixels.	Scan conversion is not required.
Raster graphics can draw mathematical curves, polygons and boundaries of curved primitives only by pixel approximation.	Vector graphics draw continuous and smooth lines.
Raster graphics cost less.	Vector graphics cost more as compared to raster graphics.
They occupy more space which depends on image quality.	They occupy less space.
File extensions: .BMP, .TIF, .GIF, .JPG	File Extensions: .SVG, .EPS, .PDF, .AI, .DXF

Source :https://tinyurl.com/taa5ju7

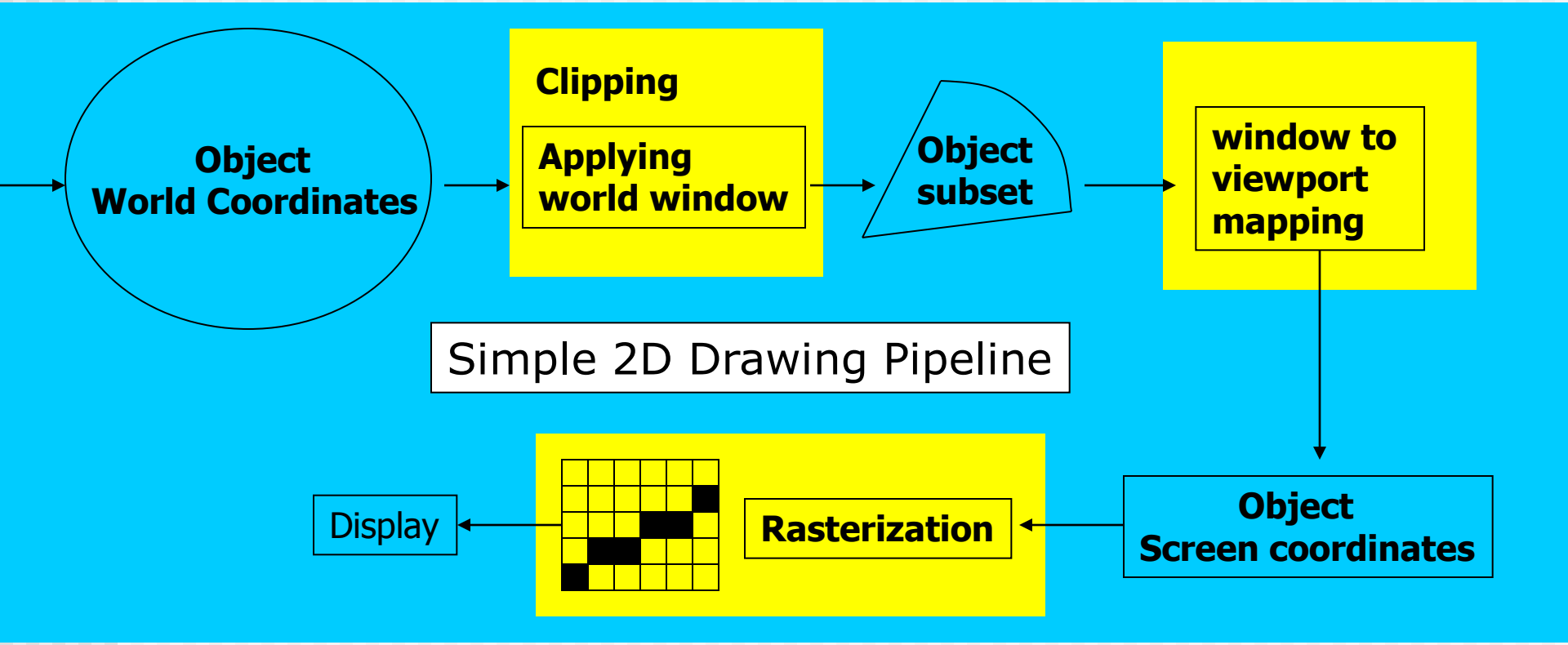
Recap: Resolution

- Resolution refers to the number of pixels in an image, and is a measurement of the output quality of an image, usually in terms of samples, pixels, dots, or lines per inch. Images are displayed on your computer screen at display (or screen) resolution (72 or 96 ppi)..

- Resolution terminology varies according to the intended output device**. PPI (pixels per inch) refers to screen resolution (monitors), DPI (dots per inch) refers to print resolution, SPI (samples per inch) refers to scanning resolution, and LPI (lines per inch) refers to halftone (often newspapers) resolution.**

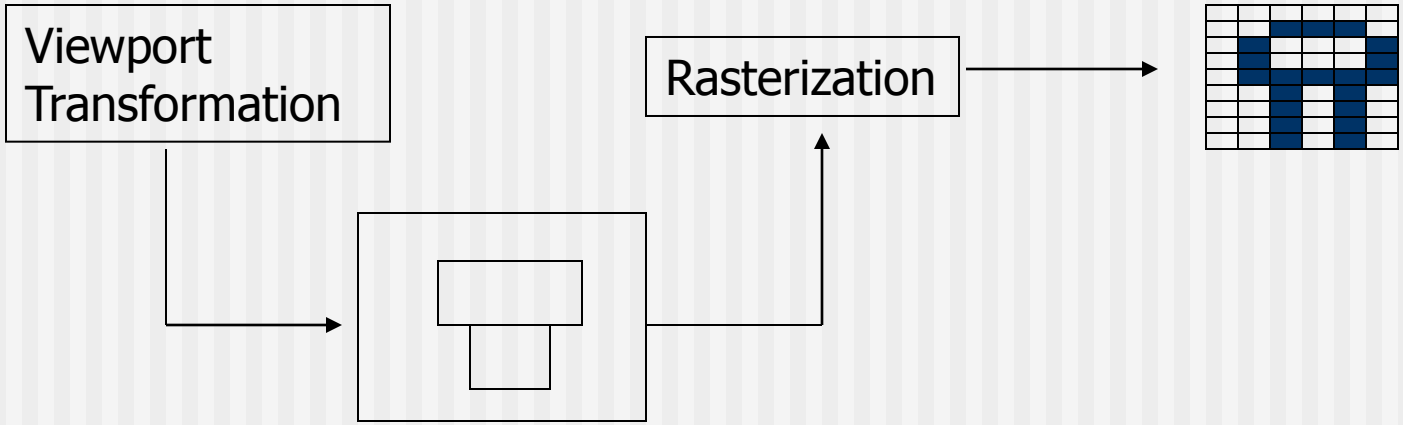
- Sometimes images are referred to as high resolution (hi-res) or low resolution (lo-res). High resolution would be an image intended for print, generally having at least 300 pixels per inch. Low resolution refers to images only intended for screen display, generally having 72-96 pixels per inch. An image for use on the internet should only be 72 dpi (the minimum display resolution) to minimize download time.

2D Graphics Pipeline



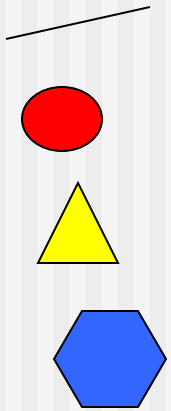
Rasterization (Scan Conversion)

- Convert high-level geometry description to pixel colors in the frame buffer
- Example: given vertex x,y coordinates determine pixel colors to draw line
- Two ways to create an image:
  - Scan existing photograph
  - Procedurally compute values (rendering)



Rasterization

- A fundamental computer graphics function
- Determine the pixels' colors, illuminations, textures, etc.
- Implemented by graphics hardware
- Rasterization algorithms
  - Lines
  - Circles
  - Triangles
  - Polygons



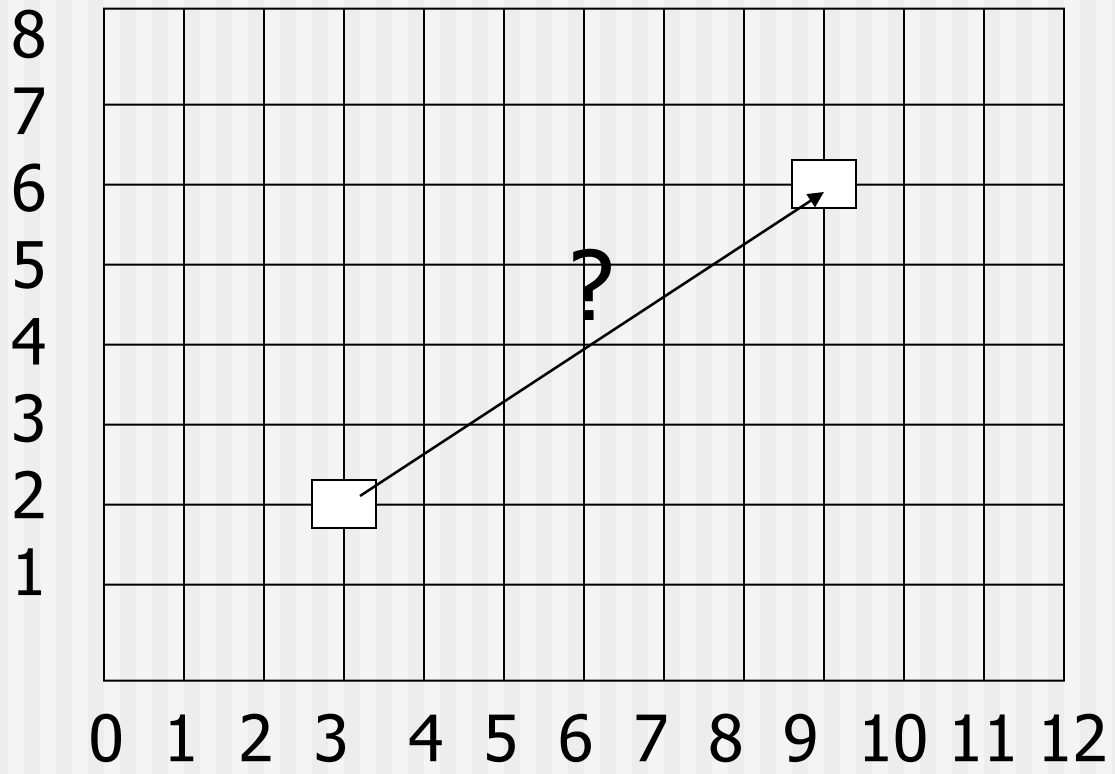
Rasterization Operations

- Drawing lines on the screen
- Manipulating pixel maps (pixmaps): copying, scaling, rotating, etc
- Compositing images, defining and modifying regions
- Drawing and filling polygons
- Aliasing and antialiasing methods

Concept: Line drawing algorithm

- Programmer specifies (x,y) values of end pixels
- Need algorithm to figure out which intermediate pixels are on line path
- Pixel (x,y) values constrained to integer values
- Actual computed intermediate line values may be floats
- Rounding may be required. E.g. computed point (10.48, 20.51) rounded to (10, 21)
- Rounded pixel value is off actual line path (jaggy!!)
  - Jaggies are stair-like lines that appear where there should be "smooth" straight lines or curves
- Sloped lines end up having jaggies
- Vertical, horizontal lines, no jaggies

Concept: Line Drawing Algorithm



Line: (3,2) -> (9,6)

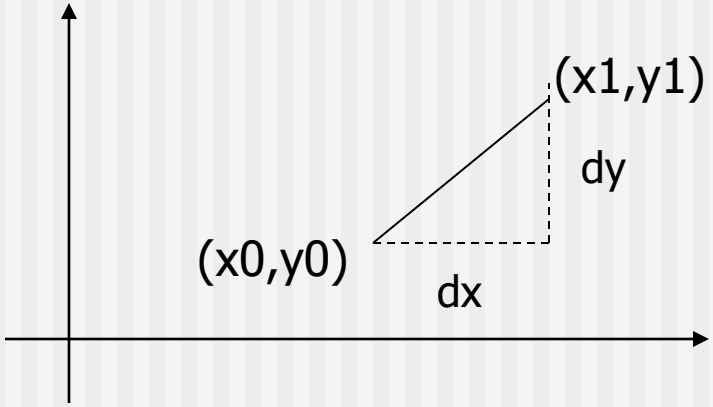
Which intermediate pixels to turn on?

Concept: Line Drawing Algorithm

- Slope-intercept line equation
  - y = mx + b
  - Given two end points (x0,y0), (x1, y1), how to compute m and b?

$$m = \frac{dy}{dx} = \frac{y1 - y0}{x1 - x0}$$

$$b = y0 - m * x0$$



Line Drawing Algorithm -Gradient

- Numerical example of finding slope m:
- (Ax, Ay) = (23, 41), (Bx, By) = (125, 96)

$$m = \frac{By - Ay}{Bx - Ax} = \frac{96 - 41}{125 - 23} = \frac{55}{102} = 0.5392$$

The various Line drawing Algorithms

- **DDA(Digital Differential Analyzer) Line Drawing Algorithm**
- **Mid Point Line Drawing Algorithm**
- **Bresenham Line Drawing Algorithm**

Read about (Important to do so)

- Xiaolin Wu's line algorithm
- Gupta-Sproull algorithm

There are many others but we limit ourselves to these

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -1

- Given the starting and ending coordinates of a line,
- DDA Algorithm attempts to generate the points between the starting and ending coordinates.

DDA Procedure-

If we have the

Starting coordinates = (X_0, Y_0)

Ending coordinates = (X_n, Y_n)

The points generation using DDA Algorithm is as follows

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -2

Step-1:

Calculate ΔX, ΔY and M from the given input. These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

$M = \Delta Y / \Delta X$

For example to **calculate the points between the starting point (5, 6) and ending point (8, 12).**

 # Starting coordinates = $(X_0, Y_0) = (5, 6)$

Ending coordinates = $(X_n, Y_n) = (8, 12)$

#Calculate ΔX, ΔY and M from the given input.

$\Delta X = X_n - X_0 = 8 - 5 = 3$

$\Delta Y = Y_n - Y_0 = 12 - 6 = 6$

$M = \Delta Y / \Delta X = 6 / 3 = 2$

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -3

Step-2:

Find the number of steps or points in between the starting and ending coordinates.

```
if (absolute (ΔX) > absolute (ΔY))
        Steps = absolute (ΔX);
    else
        Steps = absolute (ΔY);
```

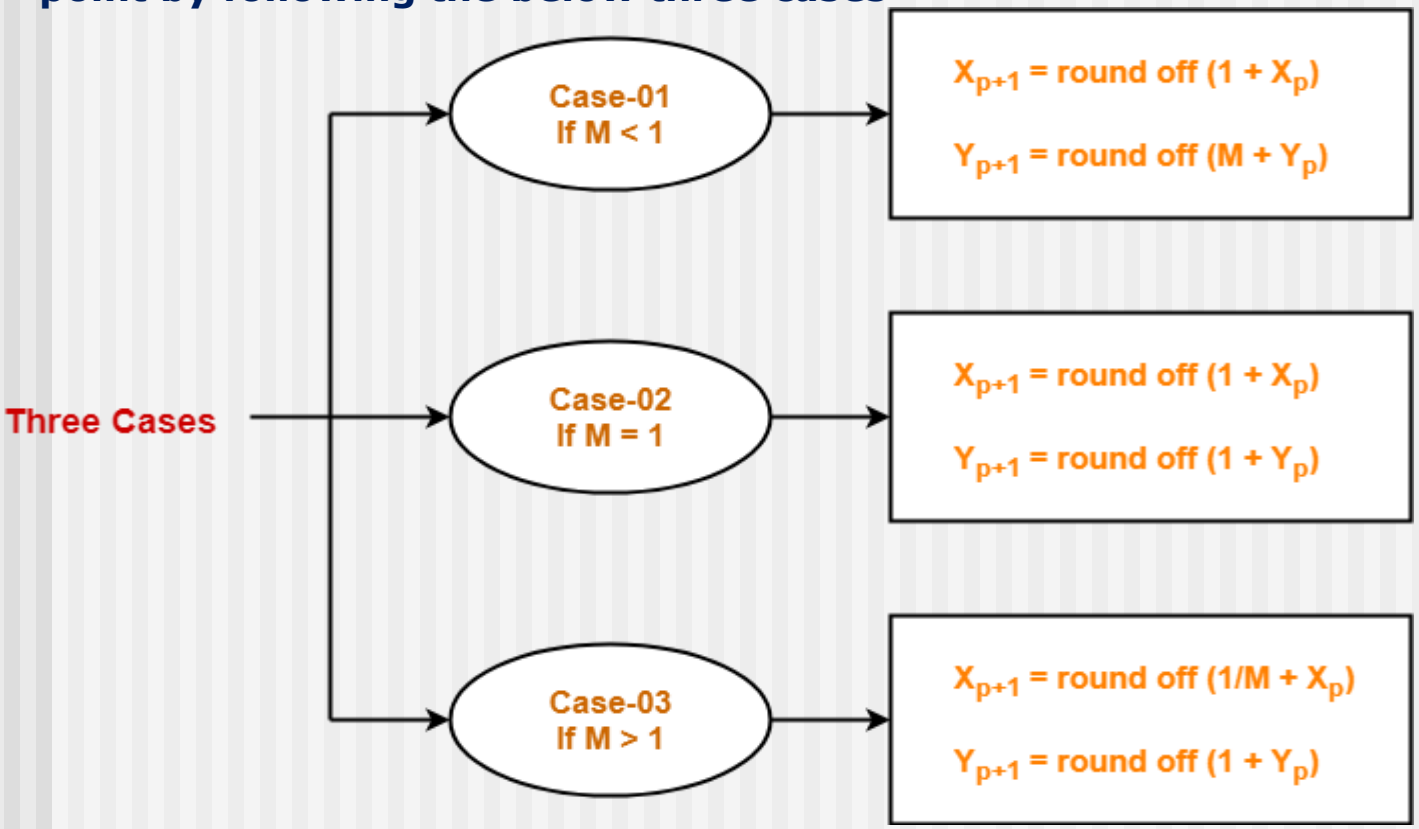
 # in our case example of starting point (5, 6) and ending point (8, 12). The number of steps are

|ΔX| < |ΔY| = 3 < 6, so number of steps = ΔY = 6

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -4

Step-3:

If the current point is (X_p, Y_p) and the next point is (X_{p+1}, Y_{p+1}). Find the next point by following the below three cases-



Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -5

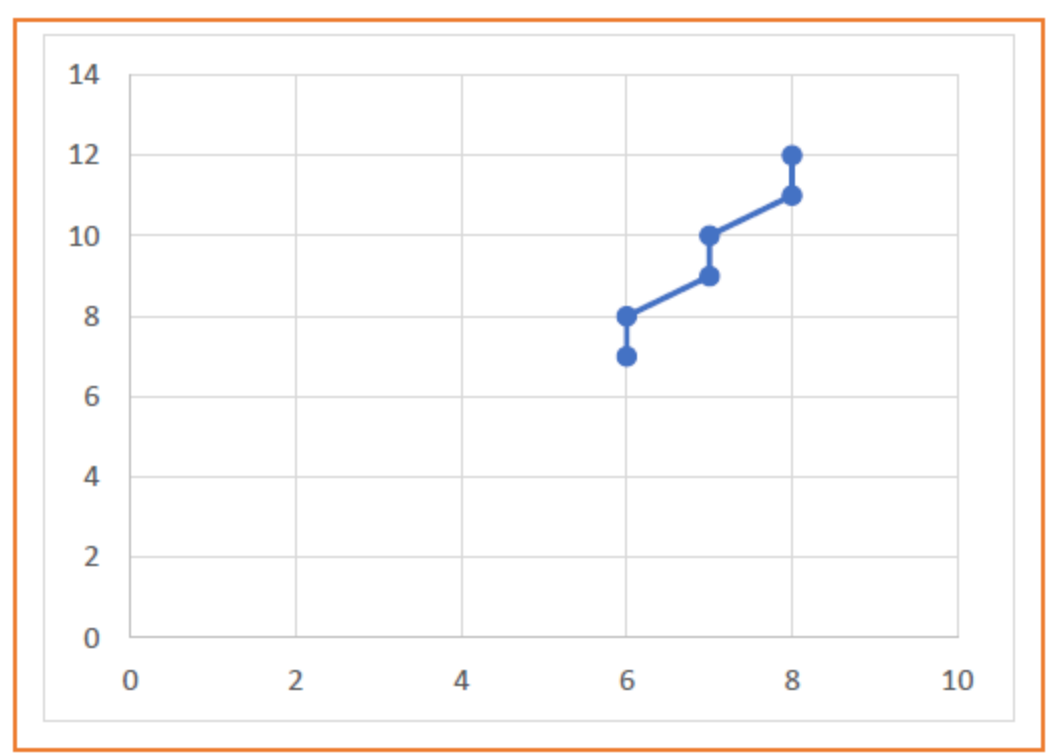
Step-3: continued

 # As M > 1, so case-03 is satisfied in our example
Now, Step-03 is executed until Step-04 is satisfied.

X_p	Y_p	X_{p+1}	Y_{p+1}	Round off (X_{p+1}, Y_{p+1})
5	6	5.5	7	(6, 7)
		6	8	(6, 8)
		6.5	9	(7, 9)
		7	10	(7, 10)
		7.5	11	(8, 11)
		8	12	(8, 12)

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -5

Step-3: The plot



DDA Line Drawing Algorithm advantages

- DDA is the simplest line drawing algorithm
- It is easy to implement.
- It avoids using the multiplication operation which is costly in terms of time complexity.

DDA Line Drawing Algorithm Drawbacks

- DDA is the simplest line drawing algorithm
  - Not very efficient
  - Round operation is expensive
  - Using round off( ) function increases time complexity of the algorithm.
  - Resulted lines are not smooth because of round off( ) function.
  - The points generated by this algorithm are not accurate.

Exercise Gauge yourself with the tasks given in Print

Line Drawing Algorithm: Mid Point Line Drawing Algorithm The Notion -1

Given the starting and ending coordinates of a line, Mid Point Line Drawing Algorithm attempts to generate the points between the starting and ending coordinates.

Midpoint Procedure-

If we have the

Starting coordinates = (X_0, Y_0)

Ending coordinates = (X_n, Y_n)

The points generation using midpoint Algorithm is as follows

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -2

Step-1:

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

For example to **calculate the points between the starting point (20, 10) and ending coordinates (30, 18).**

Starting coordinates = $(X_0, Y_0) = (20, 10)$

Ending coordinates = $(X_n, Y_n) = (30, 18)$

#**Calculate ΔX and ΔY from the given input.**

$\Delta X = X_n - X_0 = 30 - 20 = 10$

$\Delta Y = Y_n - Y_0 = 18 - 10 = 8$

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -3

Step-2:

Calculate the value of initial decision parameter and ΔD.

These parameters are calculated as-

$$D_{initial} = 2\Delta Y - \Delta X$$

$$\Delta D = 2(\Delta Y - \Delta X$$

Starting coordinates = $(X_0, Y_0) = (20, 10)$

Ending coordinates = $(X_n, Y_n) = (30, 18)$

#Calculate ΔX and ΔY from the given input.

$$\Delta X = X_n - X_0 = 30 - 20 = 10$$

$$\Delta Y = Y_n - Y_0 = 18 - 10 = 8$$

Calculate $D_{initial}$ and ΔD as-

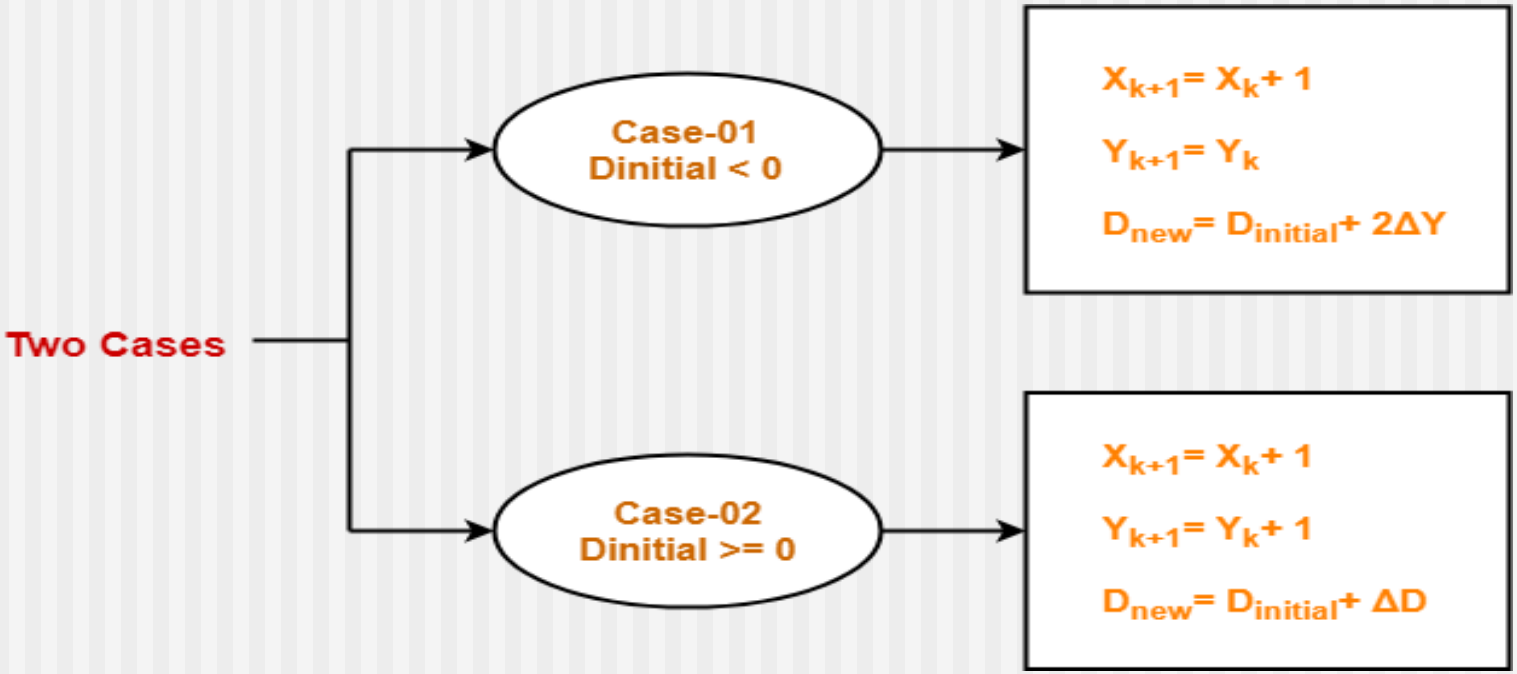
$$D_{initial} = 2\Delta Y - \Delta X = 2 \times 8 - 10 = 6$$

$$\Delta D = 2(\Delta Y - \Delta X) = 2 \times (8 - 10) = -4$$

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -4

Step-3:

The decision whether to increment X or Y coordinate depends upon the flowing values of $D_{initial}$.
Follow the below two cases-



Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -4

Step-3: continued

 # As $D_{initial}$ >= 0, so case-02 is satisfied.

Thus,

$X_{k+1} = X_k + 1 = 20 + 1 = 21$

$Y_{k+1} = Y_k + 1 = 10 + 1 = 11$

$D_{new} = D_{initial} + \Delta D = 6 + (-4) = 2$

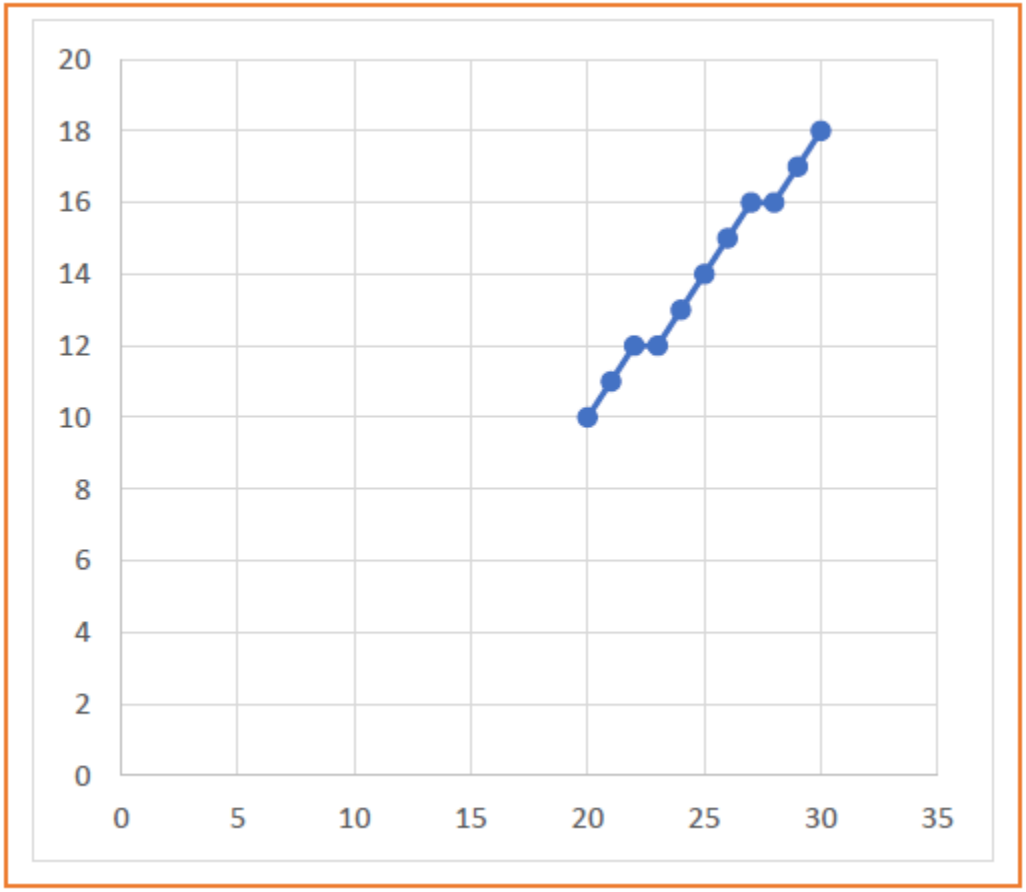
Similarly, Step 3 is executed until the end point is reached.

.

$D_{initial}$	D_{new}	X_{k+1}	Y_{k+1}
		20	10
6	2	21	11
2	-2	22	12
-2	14	23	12
14	10	24	13
10	6	25	14
6	2	26	15
2	-2	27	16
-2	14	28	16
14	10	29	17
10		30	18

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -5

Step-3: The plot



Mid Point Line Drawing Algorithm : Advantages

- Accuracy of finding points is a key feature of this algorithm.
- It is simple to implement.
- It uses basic arithmetic operations.
- It takes less time for computation.
- The resulted line is smooth as compared to other line drawing algorithms

Mid Point Line Drawing Algorithm : Drawbacks

- This algorithm may not be an ideal choice for complex graphics and images.
- In terms of accuracy of finding points, improvement is still needed.
- There is no any remarkable improvement made by this algorithm.

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -1

- Given the starting and ending coordinates of a line, Bresenham's algorithm attempts to generate the points between the starting and ending coordinates.

Bresenham Procedure-

If we have the

- Starting coordinates = (X_0, Y_0)
- Ending coordinates = (X_n, Y_n)
- The points generation using Bresenham's algorithm is as follows

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -2

Step-1:

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

For example to **calculate the points between the starting point (9, 18) and ending coordinates (14, 22).**

Starting coordinates = $(X_0, Y_0) = (9, 18)$

Ending coordinates = $(X_n, Y_n) = (14, 22)$

#Calculate ΔX, ΔY from the given input.

$\Delta X = X_n - X_0 = 14 - 9 = 5$

$\Delta Y = Y_n - Y_0 = 22 - 18 = 4$

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -3

Step-2:

Calculate the decision parameter P_k.

It is calculated as-

$$P_k = 2\Delta Y - \Delta X$$

.

 # in our case example of starting point (9, 18) and ending coordinates (14, 22). Where $\Delta X = X_n - X_0 = 14 - 9 = 5$ and $\Delta Y = Y_n - Y_0 = 22 - 18 = 4$

The decision parameter will be

$$P_k = 2\Delta Y - \Delta X$$
$$= 2 \times 4 - 5$$
$$= 3$$

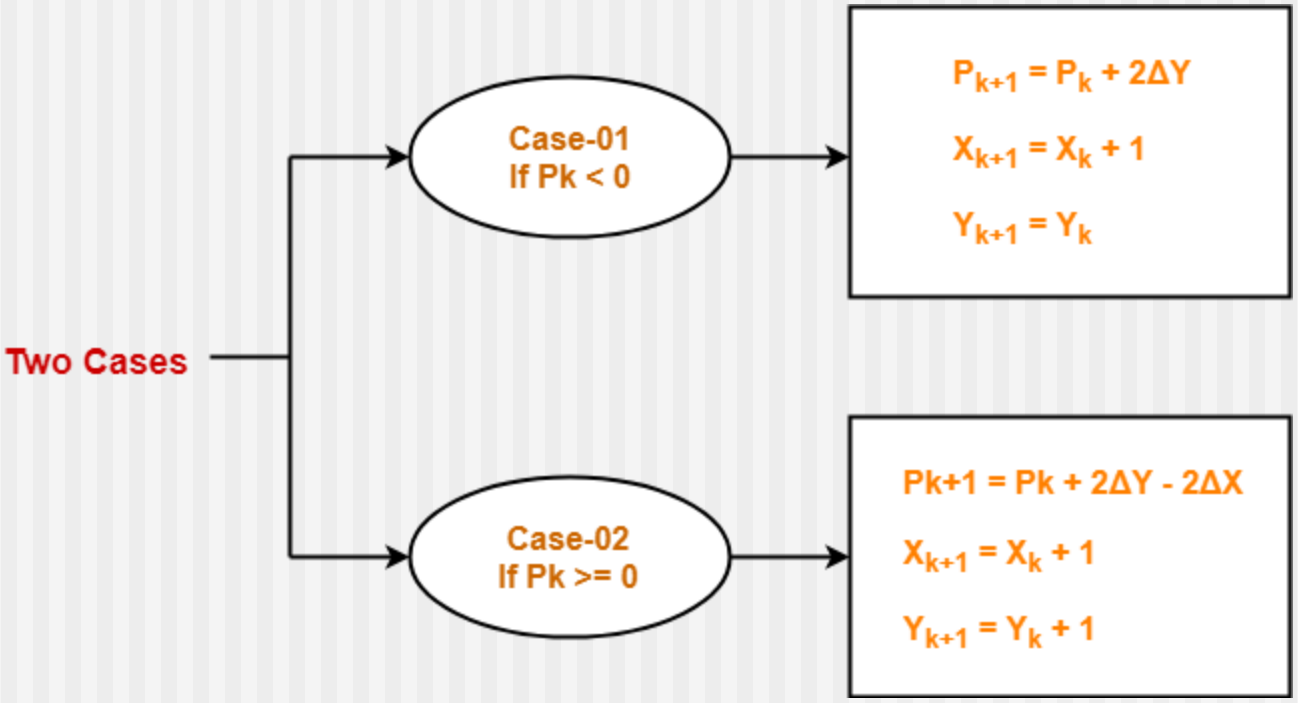
So, decision parameter $P_k = 3$

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -4

Step-3:

If the current point is (X_p, Y_p) and the next point is (X_{p+1}, Y_{p+1}). Find the next point the next point depending on the value of decision parameter P_k.

Follow the below two cases-



Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -5

Step-3: continued

 # As $P_k >= 0$, so case-02 is satisfied. Thus,

$P_{k+1} = P_k + 2\Delta Y - 2\Delta X = 3 + (2 \times 4) - (2 \times 5) = 1$

$X_{k+1} = X_k + 1 = 9 + 1 = 10$

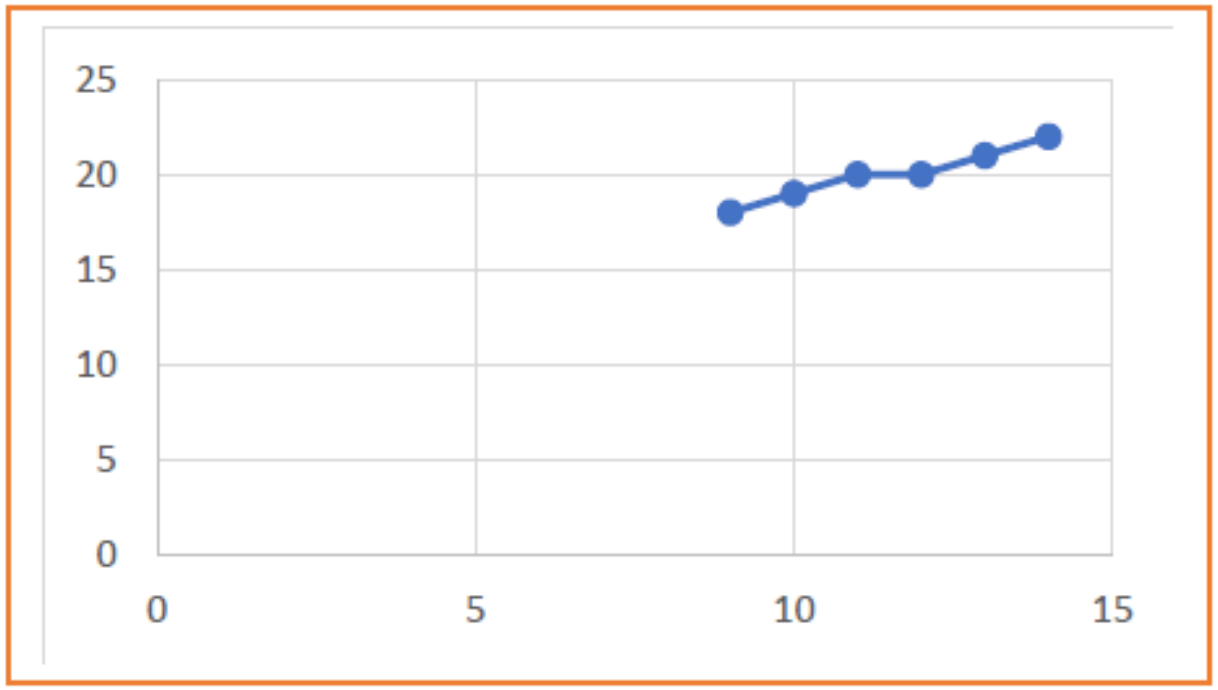
$Y_{k+1} = Y_k + 1 = 18 + 1 = 19$

Similarly, **Step-3** is executed until the end point is reached or number of iterations equals to 4 times. (Number of iterations = $\Delta X - 1 = 5 - 1 = 4$)

P_k	P_{k+1}	X_{k+1}	Y_{k+1}
		9	18
3	1	10	19
1	-1	11	20
-1	7	12	20
7	5	13	21
5	3	14	22

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -5

Step-3: The plot



Bresenham's Line-Drawing Algorithm: Advantages

- It is easy to implement.
- It is fast and incremental.
- It executes fast but less faster than DDA Algorithm.
- The points generated by this algorithm are more accurate than DDA Algorithm.
- This uses only integer calculations.

Bresenham's Line-Drawing Algorithm: Drawbacks

- Cannot effectively handle "jaggies"
  - Remember: WHAT ARE jaggies - **Jaggies** are stair-like lines that appear where there should be "smooth" straight lines or curves
    - **We will deal with them when dealing with aliasing effect**
- Improves line accuracy in generation of points but not yet smooth

Assignment (Important to do so)

- 1. Read on various image formats such as Ai, wmf, Cmx, cgm,svg ,odg, eps , dxf , bmp, jpeg ,Gif ,Tiff,PICT and png
  - **Explain what the abbreviation stand for and some history on the format**
  - **State whether each of the graphic format above is raster or vector**
  - **Briefly explain a typical application or area of usage of each of the format**
- **2. Read about Xiaolin Wu's line algorithm and**
- **Gupta-Sproull algorithm**

  Can you implement all the Five algorithms in a programming language of your choice?

3 .**How is Rasterization carried for other 2D primitives? Circles , Polygon, Ellipses , Triangles etc ?**

Installing OpenGL

- **Run**: GLinfo2.exe to see your graphic card capabilities(will tell you what OpenGL version your card supports and a list of extensions your card supports)
- **Install Microsoft Visual Studio or your favorite**

IDE (Eclipse, Intellij, Netbeans ….)

- **Install Open GL**
- **Install Glew and Glut**
- **Set project directory settings to link Glut and the IDE**

Installing OpenGL -2

You can get online guidance such as

- https://www.absingh.com/opengl/
- https://sites.google.com/site/gsucomputergraphics/educational/set-up-opengl
- https://www3.ntu.edu.sg/home/ehchua/programming/opengl/HowTo_OpenGL_C.html
- http://cacs.usc.edu/education/cs596/OGL_Setup.pdf
- https://content.byui.edu/file/2315e65e-a34a-48d3-814d-4175a2b74ed5/1/intro/165-opengl-visualStudio2017.html
- http://www.cs.uky.edu/~cheng/cs633/OpenGLInstallGuideWindows.pdf