KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Website: www.ktunotes.in

CST 304
Computer Graphics and Image Processing

Syllabus

Module – 1 (Basics of Computer graphics and Algorithms)

Basics of Computer Graphics and its applications. Video Display devices- Refresh Cathode Ray Tubes, Random Scan Displays and systems, Raster scan displays and systems. Line drawing algorithms- DDA, Bresenham's algorithm. Circle drawing algorithms- Midpoint Circle generation algorithm, Bresenham's algorithm.

SAINTGITS
LEARN.GROW.EXCEL

CST 304
Computer Graphics and Image Processing

Module – 2 (Filled Area Primitives and transformations)

Filled Area Primitives- Scan line polygon filling, Boundary filling and flood filling. Two dimensional transformations-Translation, Rotation, Scaling, Reflection and Shearing, Composite transformations, Matrix representations and homogeneous coordinates. Basic 3D transformations.

CST 304
Computer Graphics and Image Processing

Module - 3 (Clipping and Projections)

Window to viewport transformation. Cohen Sutherland Line clipping algorithm. Sutherland Hodgeman Polygon clipping algorithm. Three dimensional viewing pipeline. Projections- Parallel and Perspective projections. Visible surface detection algorithms- Depth buffer algorithm, Scan line algorithm.

CST 304
Computer Graphics and Image Processing

Module - 4 (Fundamentals of Digital Image Processing)

Introduction to Image processing and applications. Image as 2D data. Image representation in Gray scale, Binary and Colour images. Fundamental steps in image processing. Components of image processing system. Coordinate conventions. Sampling and quantization. Spatial and Gray Level Resolution. Basic relationship between pixels– neighbourhood, adjacency, connectivity. Fundamentals of spatial domain-convolution operation.

CST 304
Computer Graphics and Image Processing

Module - 5 (Image Enhancement in Spatial Domain and Image Segmentation)

Basic gray level transformation functions - Log transformations, Power-Law transformations, Contrast stretching. Histogram equalization. Basics of spatial filtering - Smoothing spatial filter- Linear and nonlinear filters, and Sharpening spatial filters-Gradient and Laplacian. Fundamentals of Image Segmentation. Thresholding - Basics of Intensity thresholding and Global Thresholding. Region based Approach - Region Growing, Region Splitting and Merging. Edge Detection - Edge Operators- Sobel and Prewitt.

Module 1 - Basics of Computer Graphics and Algorithms

Basics of Computer Graphics and its applications. Video Display devices- Refresh Cathode Ray Tubes, Random Scan Displays and systems, Raster scan displays and systems. Line drawing algorithms- DDA, Bresenham's algorithm. Circle drawing algorithms- Midpoint Circle generation algorithm, Bresenham's algorithm.

Basics of Computer Graphics

What is Computer Graphics?

- Computer Graphics includes all aspects of creating or synthesizing images using a computer.

- Creation of images, Storage, Modelling of geometric objects and Projecting 3D models onto 2D plane (Rendering) are part of Computer Graphics.

- Displaying these images on computer screen or other hardcopy output devices like printers, plotters etc. is also part of Computer Graphics.

Basics of Computer Graphics

Computer Graphics Vs Image Processing

- In Computer Graphics, computer is used to create a picture.

- In Image Processing, computer is used to modify or interpret existing pictures.

Basics of Computer Graphics

Types of Computer Graphics – broadly divided into two:

- Non interactive (Passive) Computer Graphics – observer has no control over the image. Example: Titles shown on TV
- Interactive Computer Graphics – involves two way communication between computer and user. Example: Video games

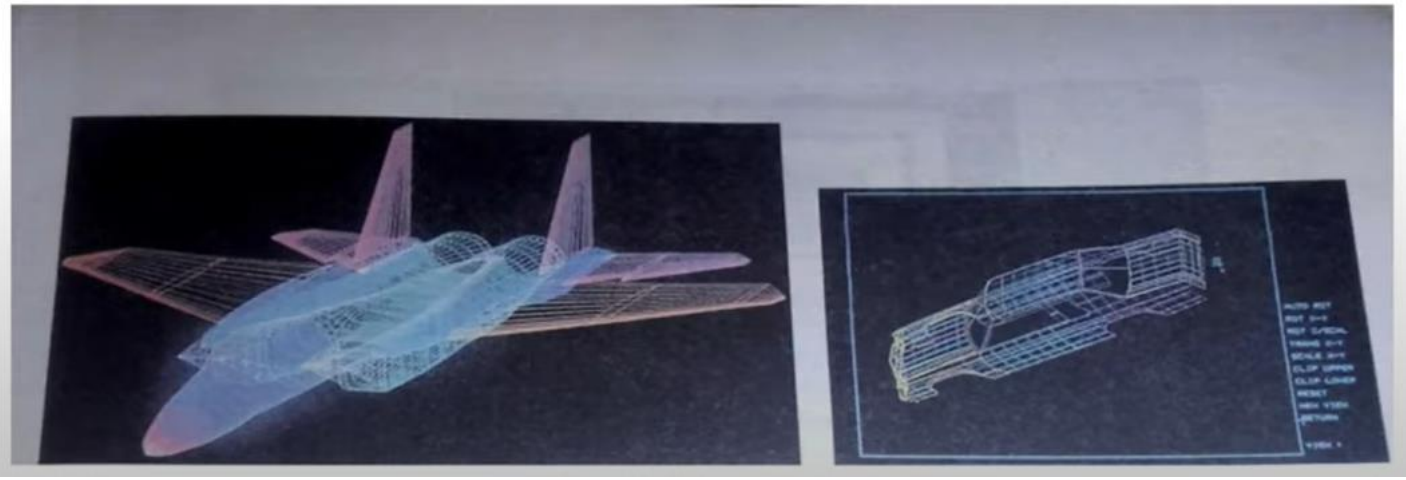
Applications of Computer Graphics

- Computer Aided Design (CAD)
- Presentation Graphics
- Computer Art
- Entertainment (Animation, Games etc.)
- Education and Training
- Visualization (Scientific and Business)
- Image Processing
- Graphical User Interfaces

Applications of Computer Graphics

1.Computer Aided Design (CAD)

- A major use of computer graphics is in **design processes**.
- Used in design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles & many other products
- Objects are displayed in **wire frame outline form** that shows the overall structure and internal features of objects. Wireframe displays also allow designers to quickly see the effects of interactive adjustments to design shapes.



Applications of Computer Graphics

Computer Aided Design

Animations – real time animations using wire frame displays are helpful in many industries

Animations in Virtual Reality environment

Architectural designs and visualization – 3D rendering and walkthroughs

Applications of Computer Graphics

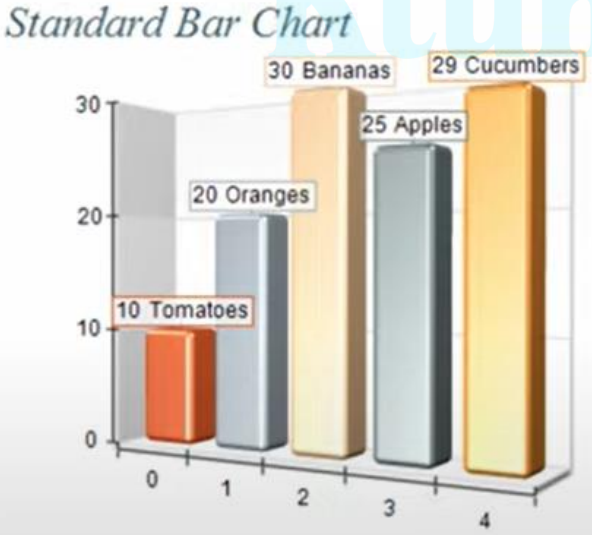
2.Presentation Graphics

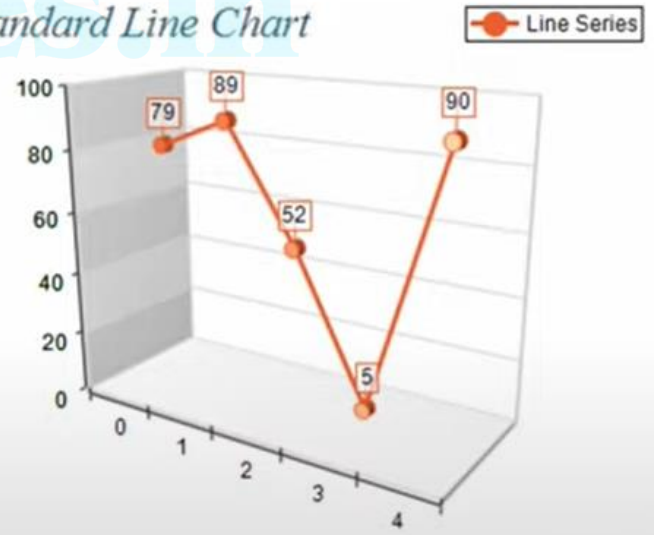
- Used to produce illustrations for reports or generate slides for use with projectors.

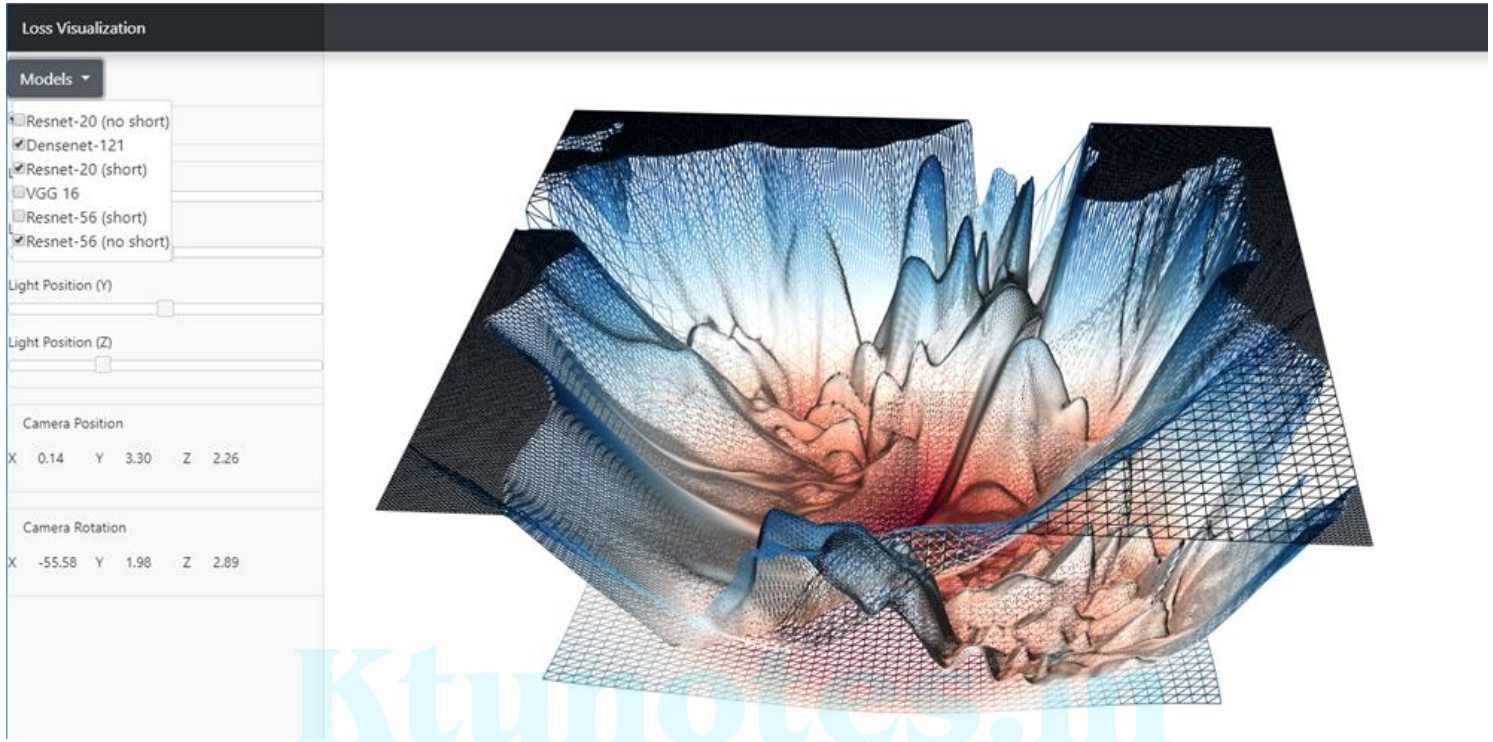
- Commonly used to summarize financial, statistical, mathematical, scientific, economic data for research reports, managerial reports & customer information bulletins.

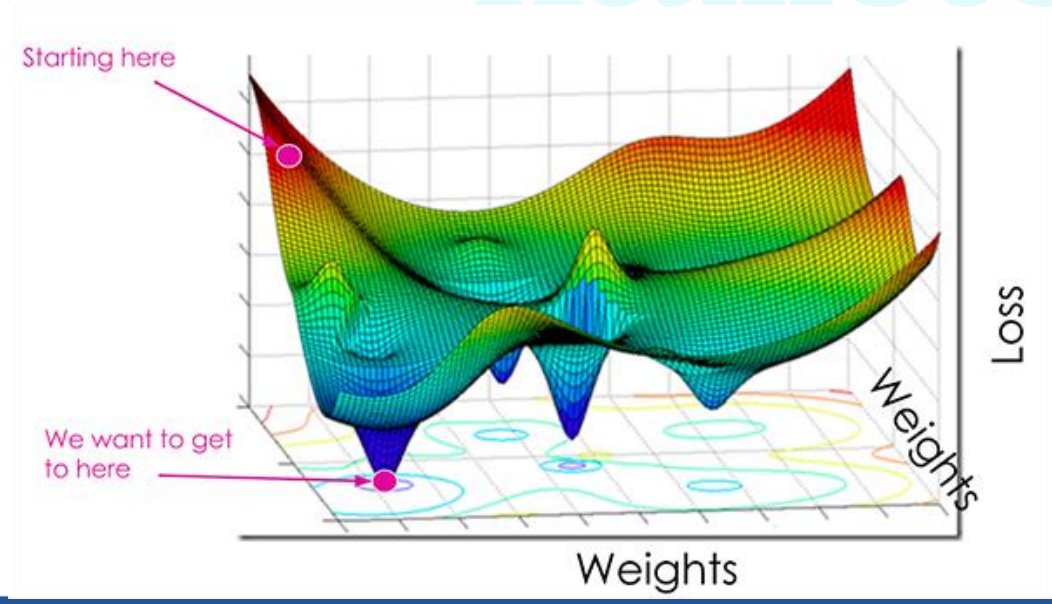
- Examples : Bar charts, line graphs, pie charts, surface graphs, time chart

Examples of presentation graphics









Neural Networks – Loss Visualization

Applications of Computer Graphics

3.Computer Art

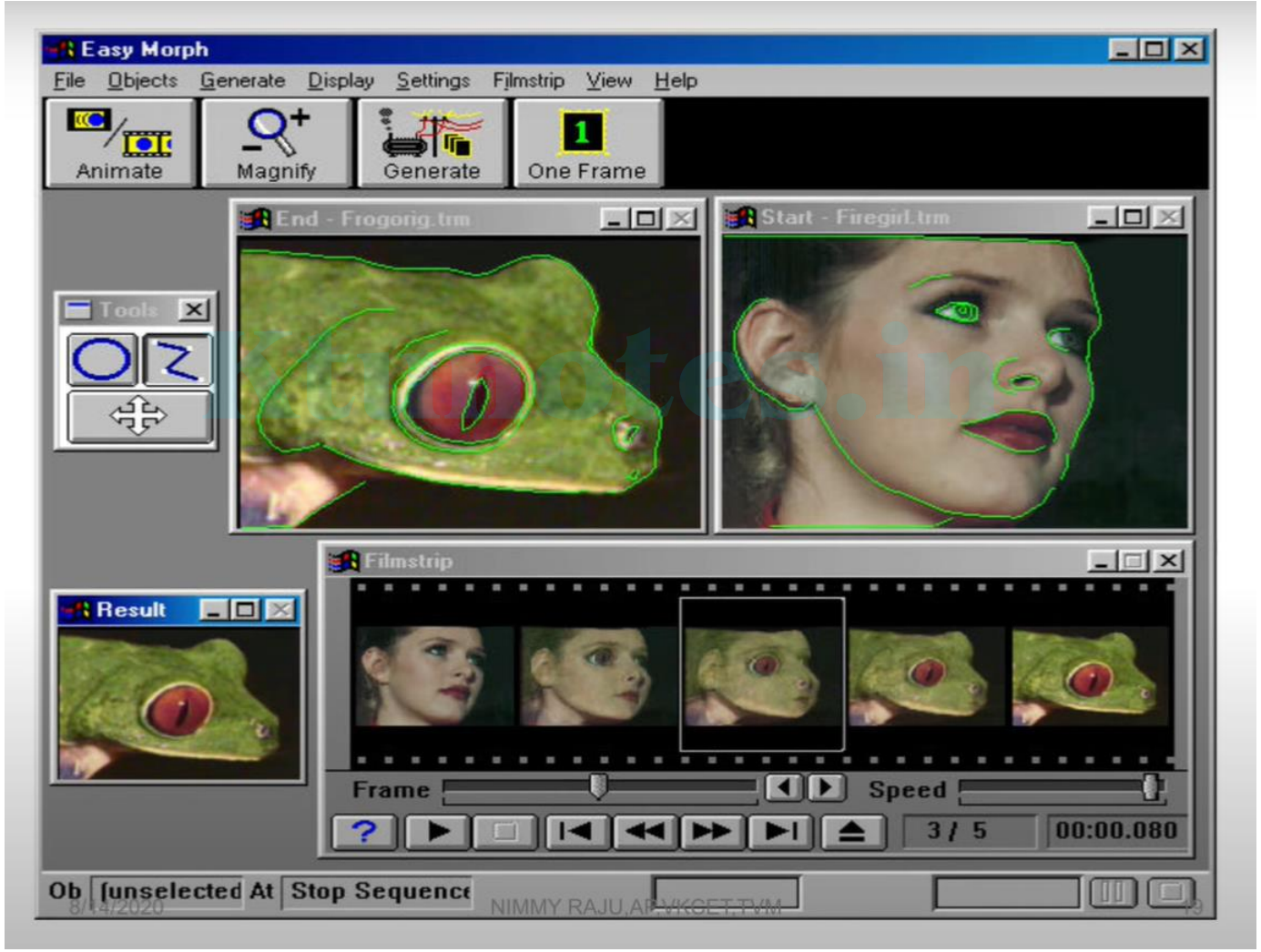
- Used in fine arts & commercial arts
  - Includes artist's paintbrush programs, paint packages, CAD packages and animation packages
  - These packages provides facilities for designing object shapes & specifying object motions.
  - Mathematical" art can also be developed to create a variety of three-dimensional and two-dimensional shapes and stereoscopic image pairs.
  - Examples : Cartoon drawing, paintings, product advertisements, logo design

Applications of Computer Graphics

Computer Art

- Electronic painting
  - Picture painted electronically on a graphics tablet (digitizer) using a stylus.
  - Cordless, pressure sensitive stylus
- Morphing
  - A common graphics method employed in many commercials is morphing, where one object is transformed (metamorphosed) into another

Applications of Computer Graphics



Applications of Computer Graphics

4.Entertainment

- Computer graphics methods are now commonly used in making motion pictures, music videos, and television shows.

- Sometimes the graphics scenes are displayed by themselves, and sometimes graphics objects are combined with the actors and live scenes.

- Movie Industry
  - Used in making of cartoon animation films

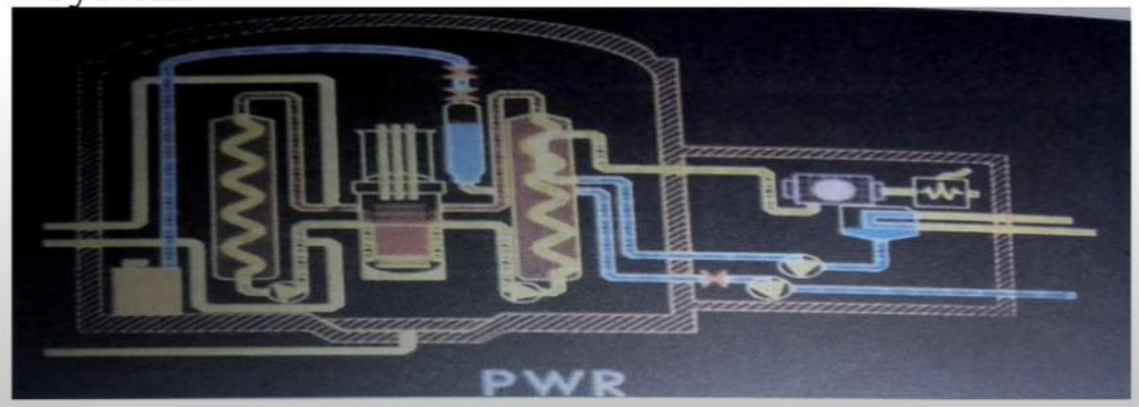
Applications of Computer Graphics

- Gaming
- VR experience

Applications of Computer Graphics

5.Education & Training

- Computer generated models of physical, financial and economic systems are used as educational aids.
- Models of physical systems, physiological systems, population trends, or equipment such as color-coded diagram help trainees understand the operation of the system



Applications of Computer Graphics

- Specialized systems used for training applications
  - simulators for practice sessions or training of ship captains
  - aircraft pilots
  - heavy equipment operators
  - air traffic-control personnel

Applications of Computer Graphics

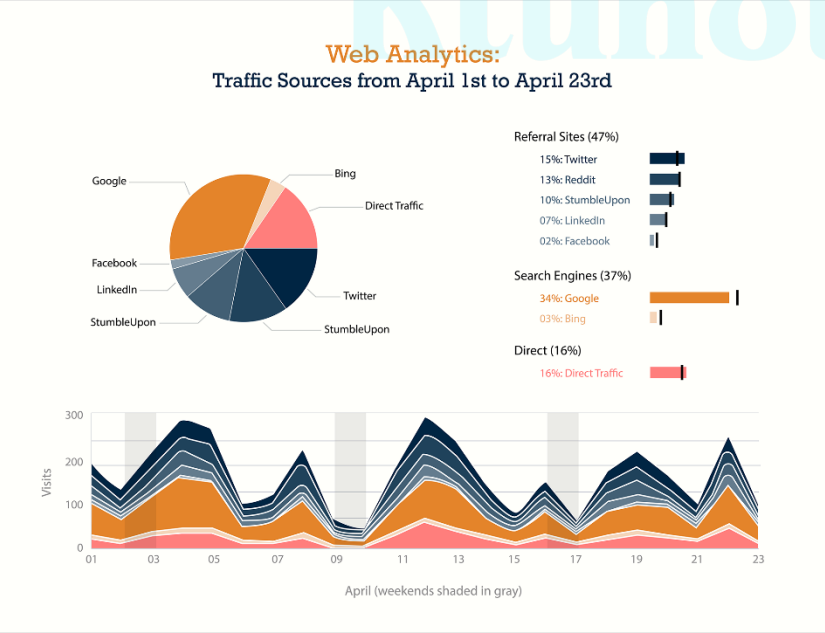
6.Visualization

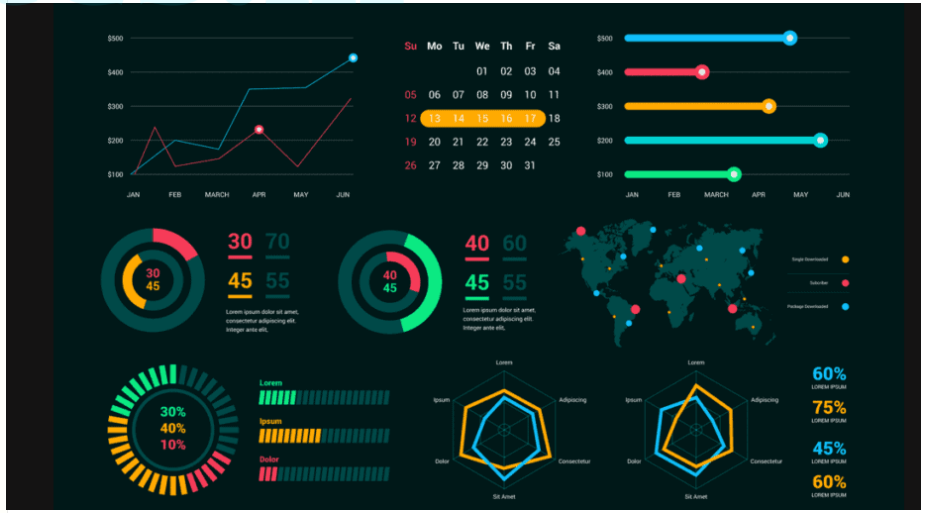
- Scientific Visualization
  - Producing graphical representations for scientific, engineering, and medical data sets.
  - To check the behaviour of certain process different fields like engineering, scientists business analysts etc need appropriate visualization.

Applications of Computer Graphics

- Business Visualization is used in connection with data sets related to commerce, industry and other non-scientific areas

- Image processing techniques are combined with computer graphics to produce many of the data visualizations





Applications of Computer Graphics

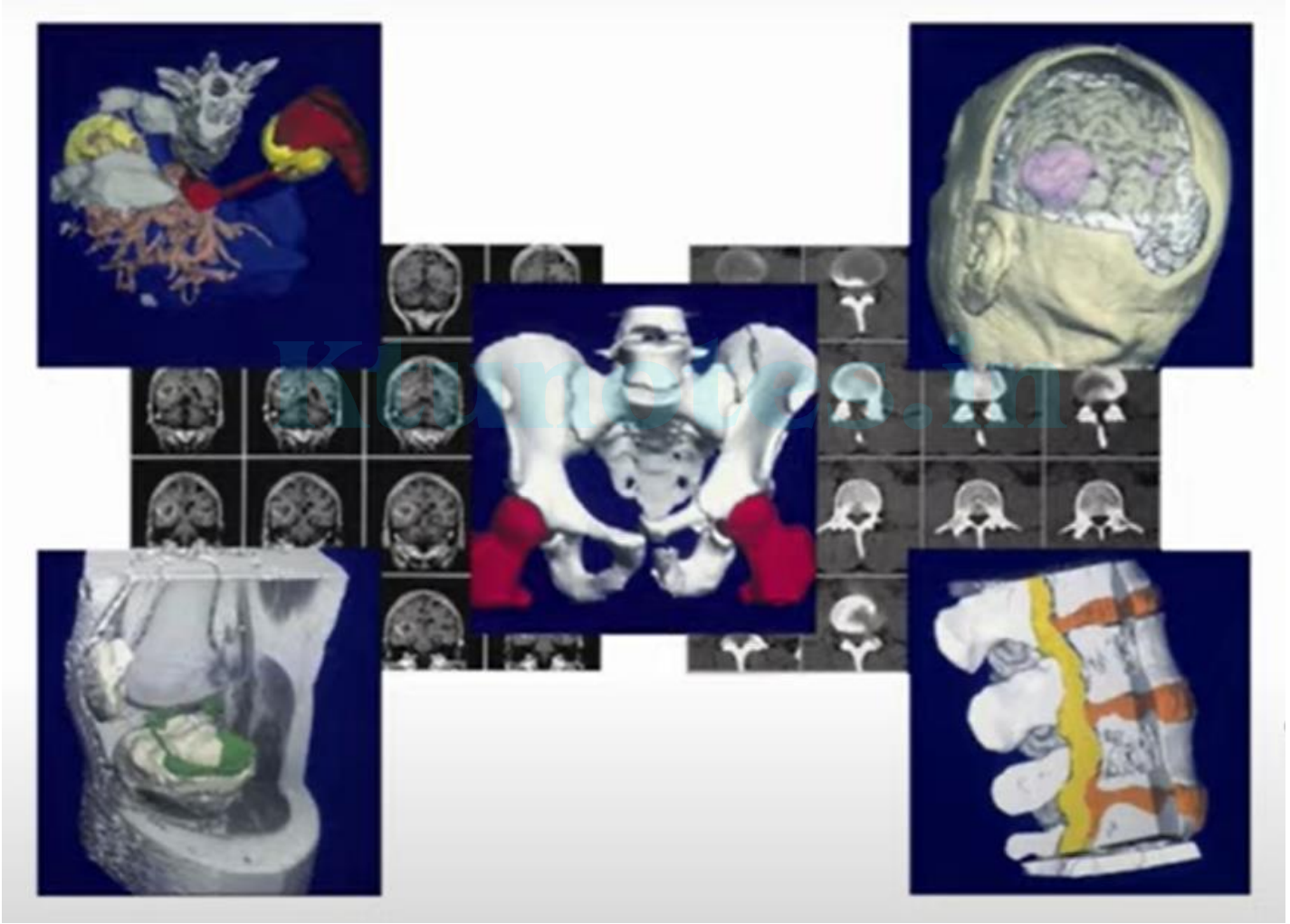
7. Image Processing

- CG- Computer is used to create a picture.
- Image Processing – applies techniques to modify or interpret existing pictures such as photographs and TV scans.
- 2 applications of image processing
  - Improving picture quality
  - Machine perception of visual information
- Medical applications
  - Picture enhancements
  - Tomography
  - Simulations of operations
  - Ultrasonics & nuclear medicine scanners

Applications of Computer Graphics

- To apply image processing methods
  - Digitize a photograph (or picture) into an image file
  - Apply digital methods to rearrange picture parts to
    - enhance color separations
    - Improve quality of shading.
  - Tomography – technique of X-ray photography that allows cross-sectional views of physiological systems to be displayed.
  - Computed X-ray tomography (CT) and position emission tomography (PET) use projection methods to reconstruct cross sections from digital data
  - Computer-Aided Surgery is a medical application technique to model and study physical functions to design artificial limbs and to plan & practice surgery

Applications of Computer Graphics



Applications of Computer Graphics

8.Graphical User Interfaces

- It is common for software packages to provide a graphical interface.

- A major component of a graphical interface is a window manager that allows a user to display multiple-window areas. To make a particular window active, simply click in that window using an interactive pointing device.

- Interfaces also display menus and icons for fast selection of processing options or parameter values.

Applications of Computer Graphics

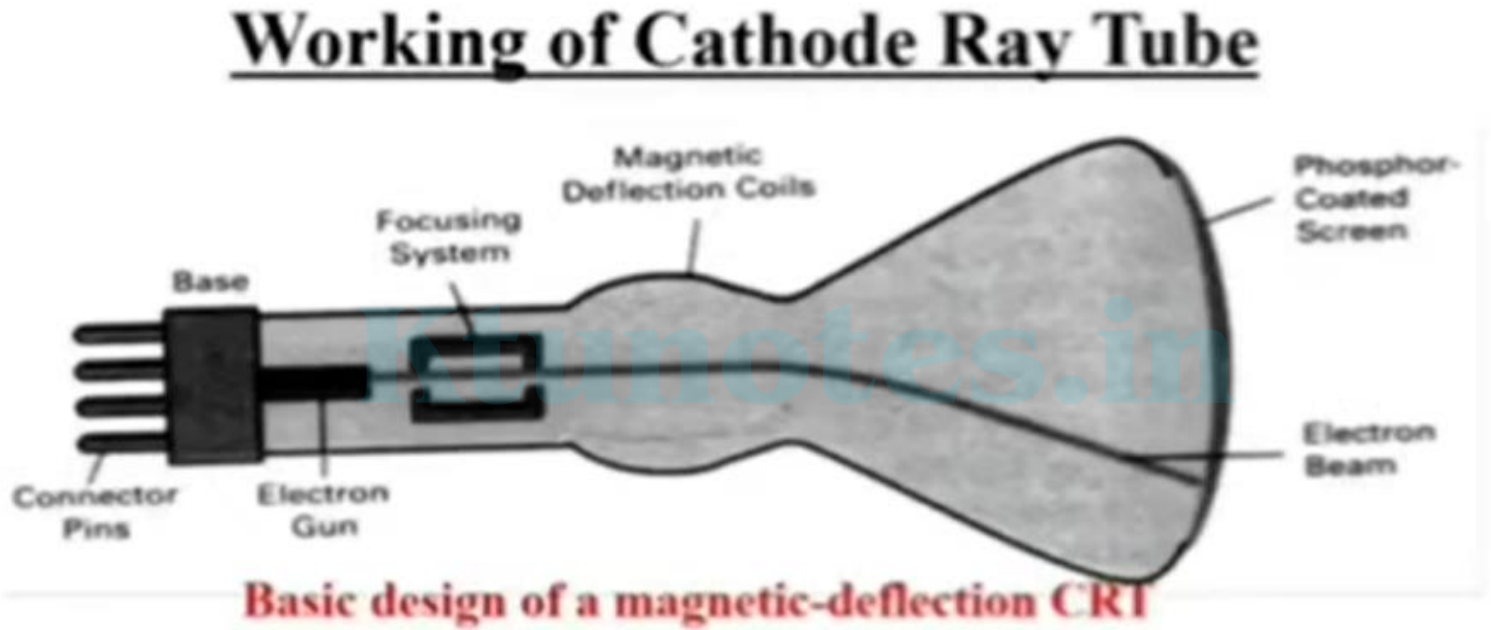
- An icon is a graphical symbol that is designed to look like the processing option it represents.
- The advantages of icons are that they take up less screen space than corresponding textual descriptions and they can be understood more quickly if well designed.
- Menus contain lists of textual descriptions and icons.

Applications of Computer Graphics

- Computer Aided Design (CAD)
- Presentation Graphics
- Computer Art
- Entertainment (Animation, Games etc.)
- Education and Training
- Visualization (Scientific and Business)
- Image Processing
- Graphical User Interfaces

Video Display Devices

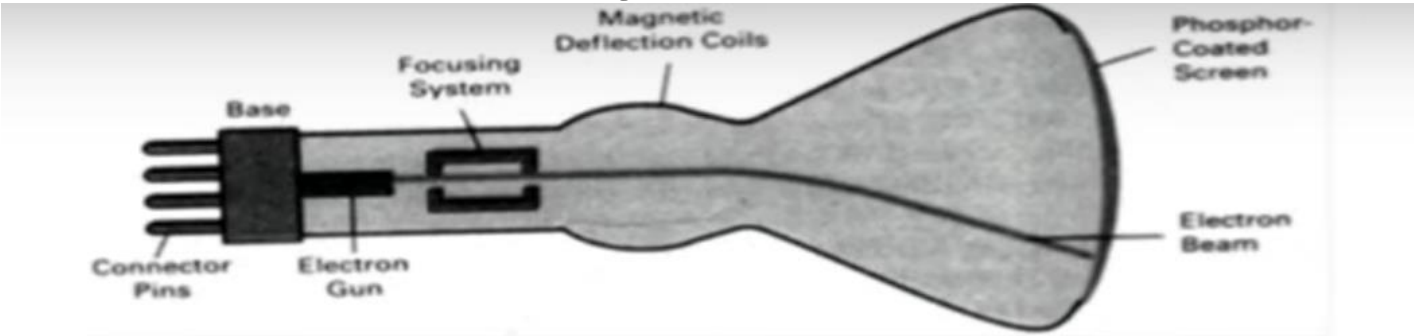
Refresh Cathode Ray Tube



- The primary output device in a graphics system is a video monitor.
- The operation of **most** video monitors is based on the standard **cathode-ray tube (CRT)** design

Video Display Devices

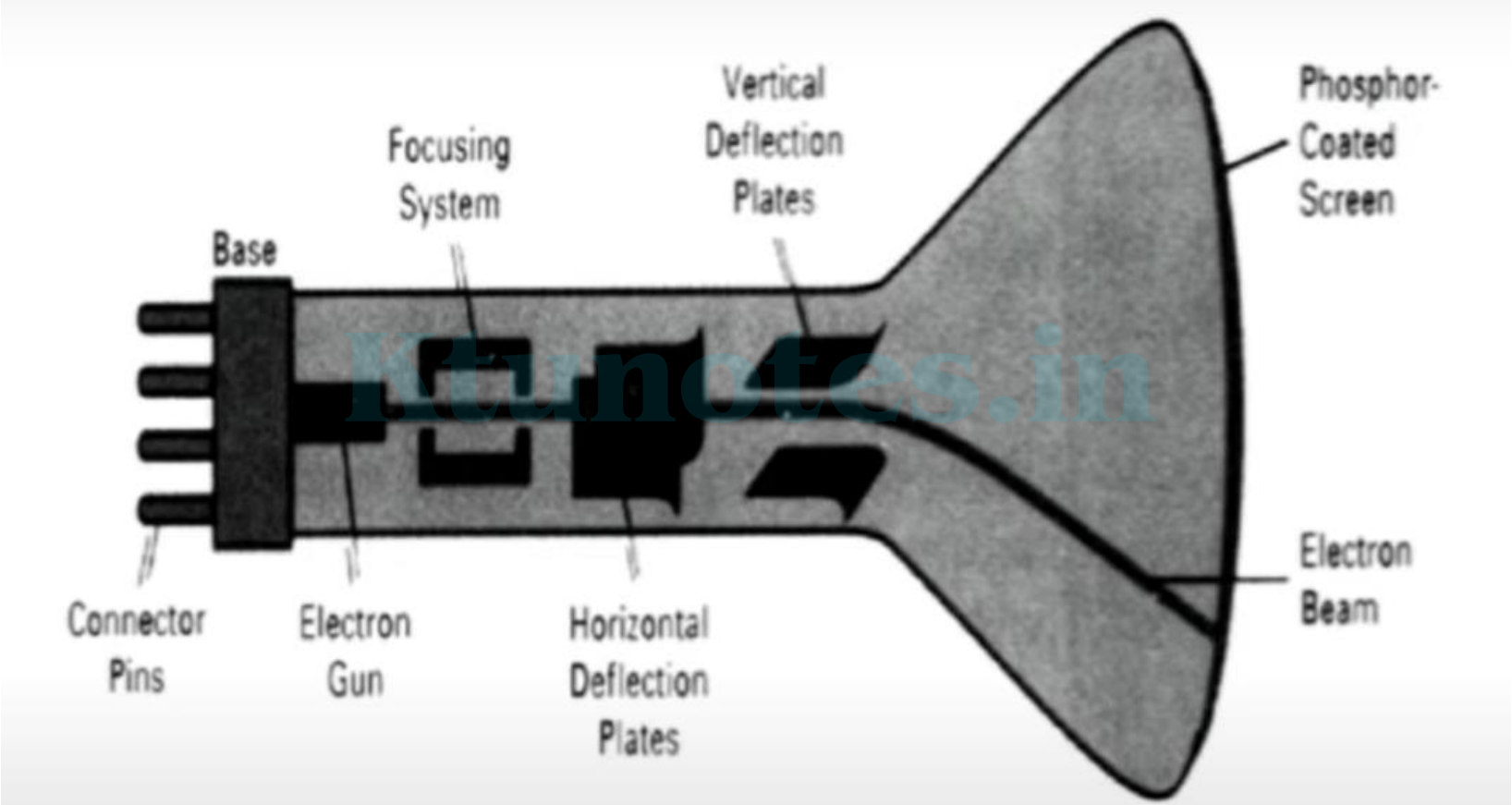
Refresh Cathode Ray Tube



- A beam of electrons (cathode rays), emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on the phosphor coated screen.
- The phosphor then emits a small spot of light at each position contacted by the electron beam.
- Because the light emitted by the phosphor fades very rapidly, some method is needed for maintaining the screen picture.
- One way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points.
- This type of display is called a refresh CRT.

Video Display Devices

– Refresh Cathode Ray Tube



Electrostatic deflection of the electron beam in a CI

Video Display Devices

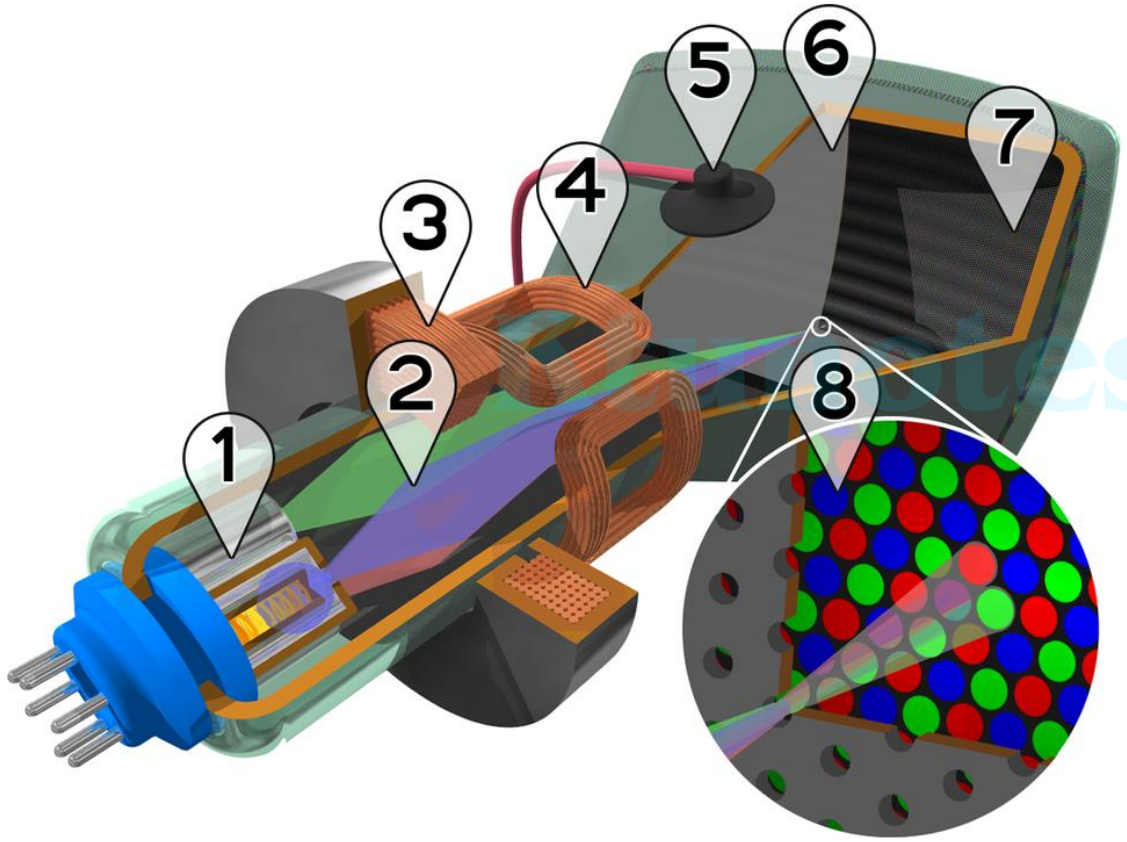
– Refresh Cathode Ray Tube

How are spots of light produced on the CRT Screen?

- Spots of light are produced on the screen by the transfer of the CRT beam energy to the phosphor.
- Part of the beam energy (kinetic) is converted by friction into heat energy and the remainder causes electrons in the phosphor atoms to move up to higher quantum energy levels.
- After a short time, the excited phosphor electrons begin dropping back to their stable ground state, giving up their extra energy as small amounts of light energy.
- What we see on the screen is the combined effect of all the electron light emissions and these spots of light quickly fading as the phosphor electrons return to ground energy level. Hence refreshing is done to retain the image.

Video Display Devices

– Refresh Cathode Ray Tube



Cutaway rendering of a color CRT:

1. Three electron emitters (for red, green, and blue phosphor dots)
2. Electron beams
3. Focusing coils
4. Deflection coils
5. Connection for final anodes
6. Mask for separating beams for red, green, and blue part of the displayed image
7. Phosphor layer (screen)with red, green, and blue zones
8. Close-up of the phosphor-coated inner side of the screen

Video Display Devices

– Refresh Cathode Ray Tube

The rear of a 14-inch color cathode-ray tube showing its deflection coils and electron guns

Video Display Devices

Some terms connected with the display screen:

- Persistence

Persistence is defined as the time it takes the emitted light from the screen to decay to one tenth of its original intensity.

- Resolution

The maximum number of points that can be displayed without overlap on a CRT is referred to as the resolution.

- Aspect Ratio

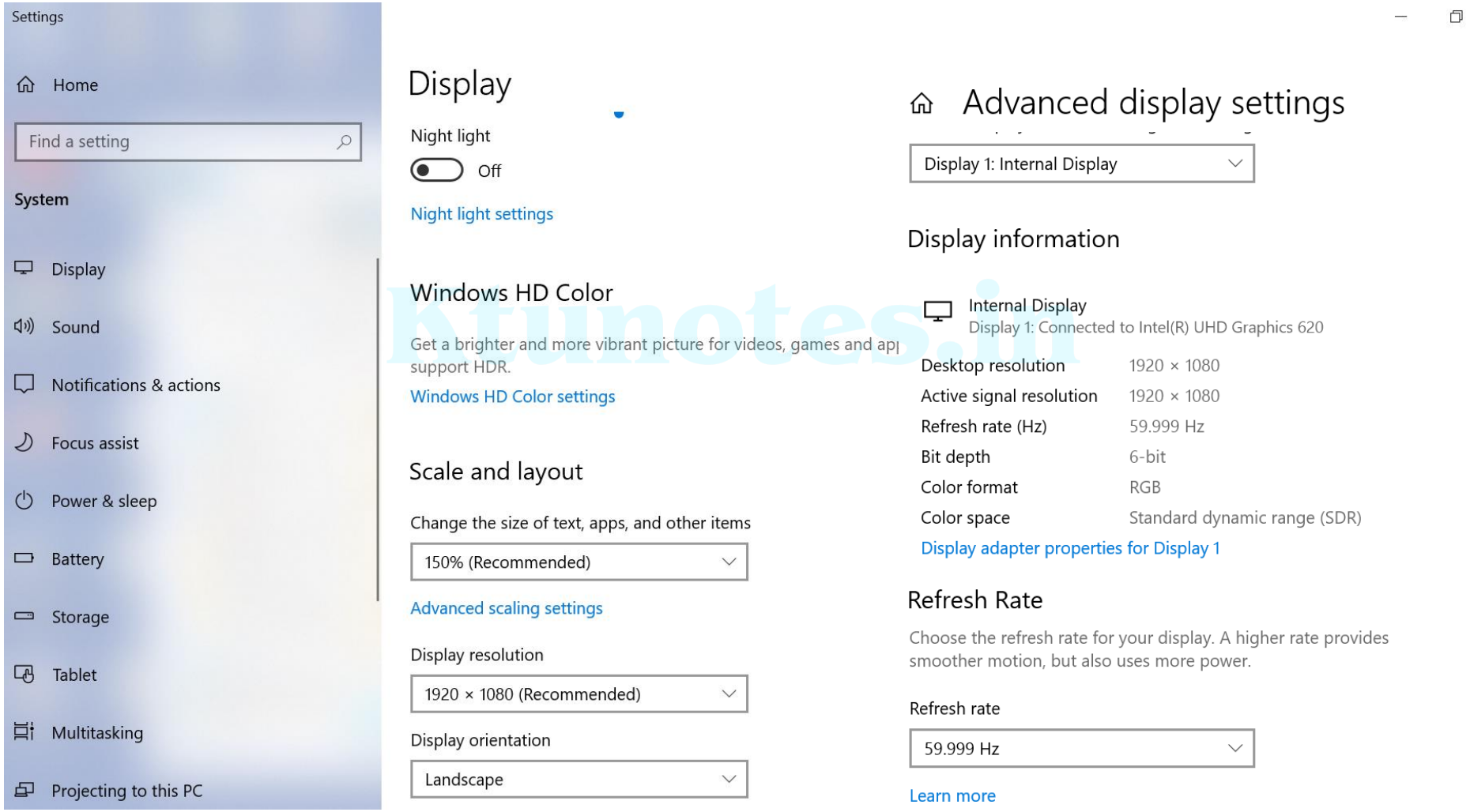
This number gives the ratio of horizontal points to vertical points (sometimes - vertical points to horizontal points).

Video Display Devices

Here's a breakdown of the different aspect ratios:

- 16:9 aspect ratio: (modern standard) 1366 x 768, 1600 x 900, 1920 × 1080, 3840 × 2160, 2560 × 1440
- 4:3 aspect ratio: (old standard) 1400 × 1050, 1440 × 1080, 1920 × 1440
- 21:9 aspect ratio: (ultrawide) 2560 × 1080, 3440 × 1440
- 32:9 aspect ratio: (superwide) 3840 × 1080

Video Display Devices



Video Display Devices

Raster Scan displays

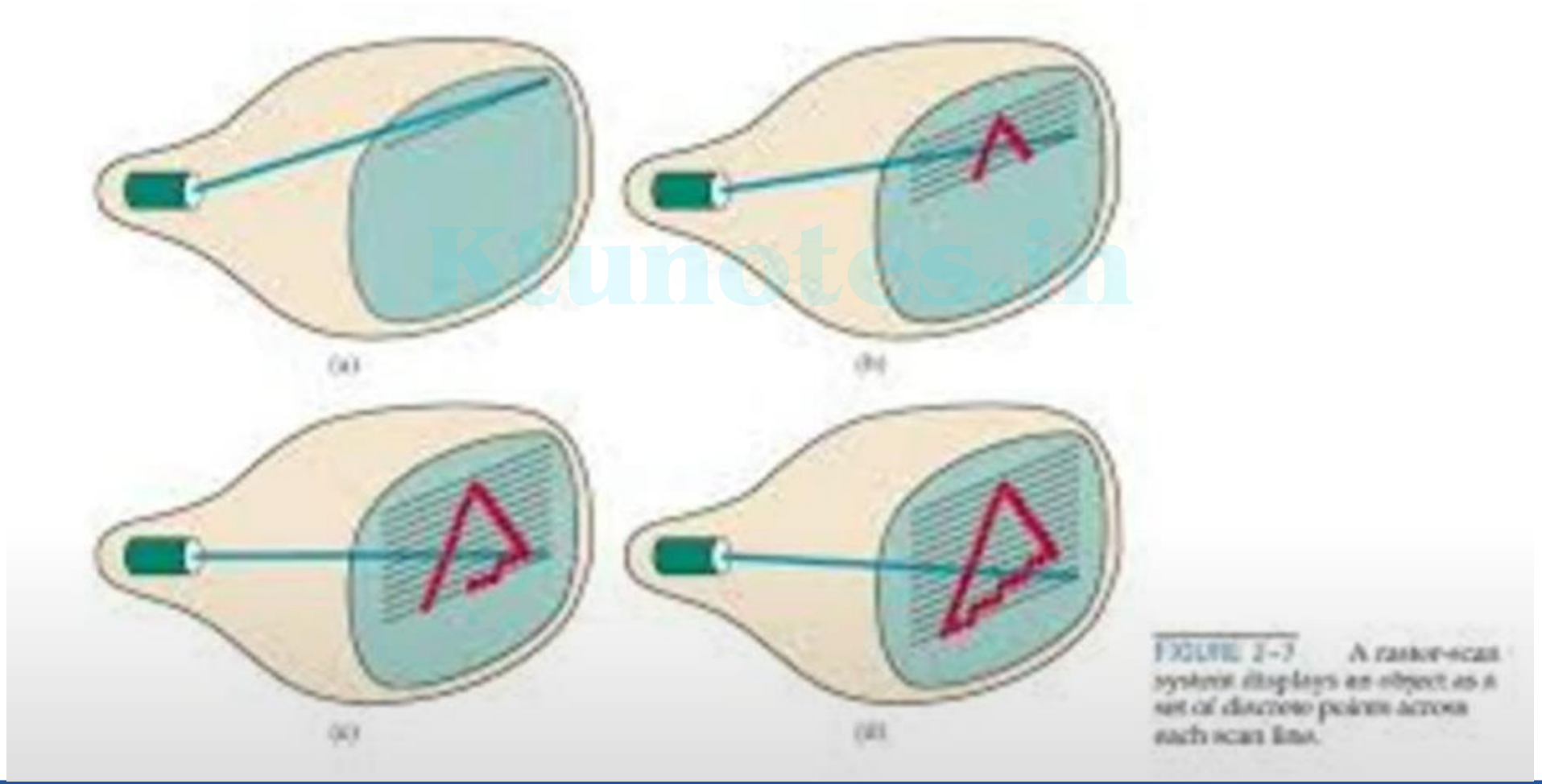


FIGURE 2-7 A raster-scan system displays an object as a set of discrete points across each scan line.

Video Display Devices

– Raster Scan Displays

- In a raster-scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.
- Picture definition is stored in a memory area called the **refresh buffer or frame buffer**. This memory area holds the set of intensity values for all the screen points. Stored intensity values are then retrieved from the refresh buffer and "painted" on the screen one row (scan line) at a time .

Video Display Devices

– Raster Scan Displays

- Each screen point is referred to as a **pixel or pel** (shortened forms of picture element).
- It provides realistic display of scenes.
- Home television sets and printers are examples using raster-scan methods.
- In a simple black-and-white system, each screen point is either on or off, so only one bit per pixel is needed to control the intensity of screen positions.
- For a bilevel system, a **bit value of 1** indicates that the electron beam is to be **turned on** at that position, and a **value of 0** indicates that the beam **intensity is to be off**.

Video Display Devices

– Raster Scan Displays

- Additional bits are needed when colour and intensity variations can be displayed.

- **A** system with **24** bits per pixel and a screen resolution of **1024 by 1024** requires 3 megabytes of storage for the frame buffer

- On a black-and-white system with one bit per pixel, the frame buffer is commonly called a **bitmap**.
- For systems with multiple bits per pixel, the frame buffer is often referred to as a **pixmap**.

- Refreshing on raster-scan displays is carried out at the rate of 60 to 80 frames per second, which are described in units of cycles per second, or Hertz (Hz), where a cycle corresponds to one frame.

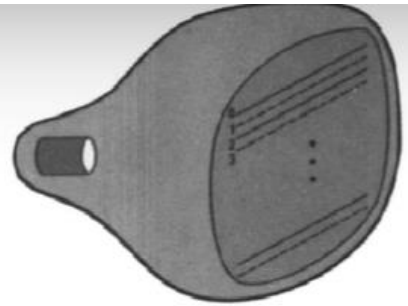
Video Display Devices

– Raster Scan Displays

- At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line. The return to the left of the screen, after refreshing each scan line, is called the **horizontal retrace** of the electron beam.

- And at the end of each frame (displayed in 1/80th to 1/60th of a second), the electron beam returns (**vertical retrace**) to the top left comer of the screen to begin the next frame.

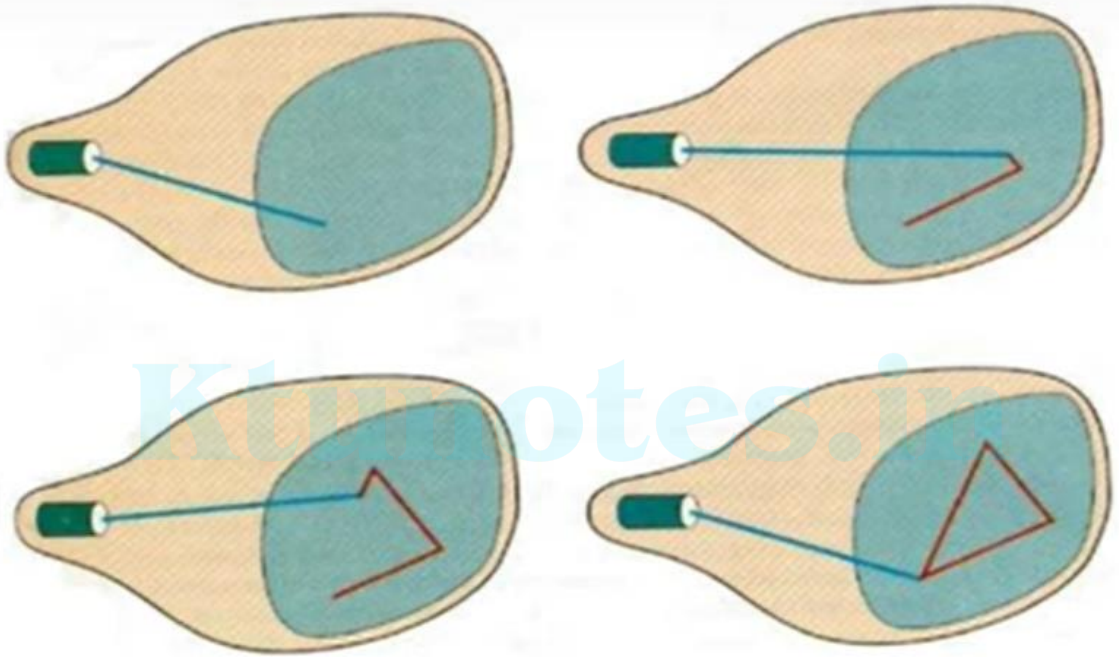
Video Display Devices
– Raster Scan Displays



- On some raster-scan systems , each displayed in two passes using an interlaced refresh
- In the first pass, the beam sweeps across every other scan line from top to bottom.
- Then after the vertical retrace, the beam sweeps out the remaining scan lines .
- Interlacing of the scan lines in this way allows us to see the entire screen displayed in one-half the time it would have taken to sweep across all the lines at once from top to bottom.
- Interlacing is primarily used with slower refreshing rates

Video Display Devices

Random-Scan Displays



- In **random-scan display unit**, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn.
- Random scan monitors draw a picture one line at a time and for this reason are also referred to as **vector displays** ***(or*** stroke-writing or calligraphic displays).

Video Display Devices

– Random Scan Displays

- Refresh rate on a random-scan system depends on the number of lines to be displayed.
- Picture definition is now stored as a set of line drawing commands in an area of memory referred to as the **refresh display file**. Sometimes the refresh display file is called the **display list, display program, or simply the refresh buffer**.
- To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all line drawing commands have been processed, the system cycles back to the first line command in the list.
- Random-scan systems are designed for line drawing applications and cannot display realistic shaded scenes.

Line Drawing algorithms

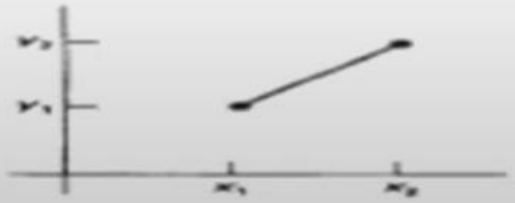
DDA Line drawing Algorithm

General equations of line:

- The Cartesian ***slope-intercept equation*** for a straight line is with m representing the slope of the line and b as they intercept. **y=m.x +b**
- For the line segment with two endpoints that are specified at positions **(x1,y1) and (x2,y2)** determine values for the slope m and y intercept b with the following calculations:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - m \cdot x_1$$



Line Drawing algorithms

- For any given x interval x along a line, compute the corresponding y interval y *as*

$$\Delta y = m \Delta x$$

- Similarly, the x interval x corresponding to a specified y **as**

$$\Delta x = \frac{\Delta y}{m}$$

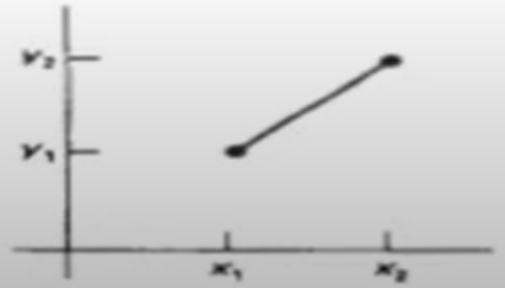
Line Drawing algorithms

DDA Algorithm

- The **digital differential analyzer (DDA)** is a scan-conversion line algorithm based on calculating either Δx *or* Δy.

- The line should be sampled the line at unit intervals in one coordinate and determine corresponding integer values nearest the line path for the other coordinate.

- Consider first a line with positive slope



Line Drawing algorithms- DDA

1. If the **slope is less than or equal to 1,** sample at unit x intervals $(\Delta x = 1)$ and compute each successive **y** value as

$$y_{k+1} = y_k + m$$

$m = (y_{k+1} - y_k) / (x_{k+1} - x_k)$

Here we increment x by 1. So $x_{k+1} - x_k = 1$

Therefore, $m = (y_{k+1} - y_k)/1$

$Y_{k+1} = y_k + m$

2. For lines with a **positive slope greater than 1**, reverse the roles of x and y. That is, sample at unit y intervals $(\Delta y = 1)$ and calculate each succeeding x value as

$$x_{k+1} = x_k + \frac{1}{m}$$

Here we increment y by 1 . So $y_{k+1} - y_k = 1$

Hence: $m = 1 / (x_{k+1} - x_k)$

$(x_{k+1} - x_k) = 1/m$

$x_{k+1} = x_k + 1/m$

These Equations are based on the assumption that lines are to be processed from the **left endpoint to the right endpoint .**

Line Drawing algorithms - DDA

- This algorithm is summarized in the following procedure as

1. **Input the two endpoint pixel positions.**
2. **Horizontal and vertical differences between the endpoint positions are assigned to parameters dx and dy.**

   dx=x2-x1; dy=y2-y1;
3. **The difference with the greater magnitude determines the value of parameter steps.**

   X_{inc} = dx/steps Y_{inc} = dy/steps. When slope m<1, steps = dx and X_{inc} = dx/dx =1. We need to increment x by 1 and calculate corresponding y.
4. **Starting with pixel position (x_0,y_0), determine the offset needed at each step to generate the next pixel position along the line path.**

   When slope m>1, steps =dy and Y_{inc} = dy/steps =1 and X_{inc} = dx/steps = dx/dy = 1/m We need to increment y by 1 and calculate corresponding x value.

   x_{inc}=dx/steps; y_{inc}=dy/steps;
5. **Loop through this process steps times.**

   x=x+ x_{inc}; y=y+ y_{inc}

when m<1,
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k + m$

when m>1,
$y_{k+1} = y_k + 1$
$x_{k+1} = x_k + 1/m$

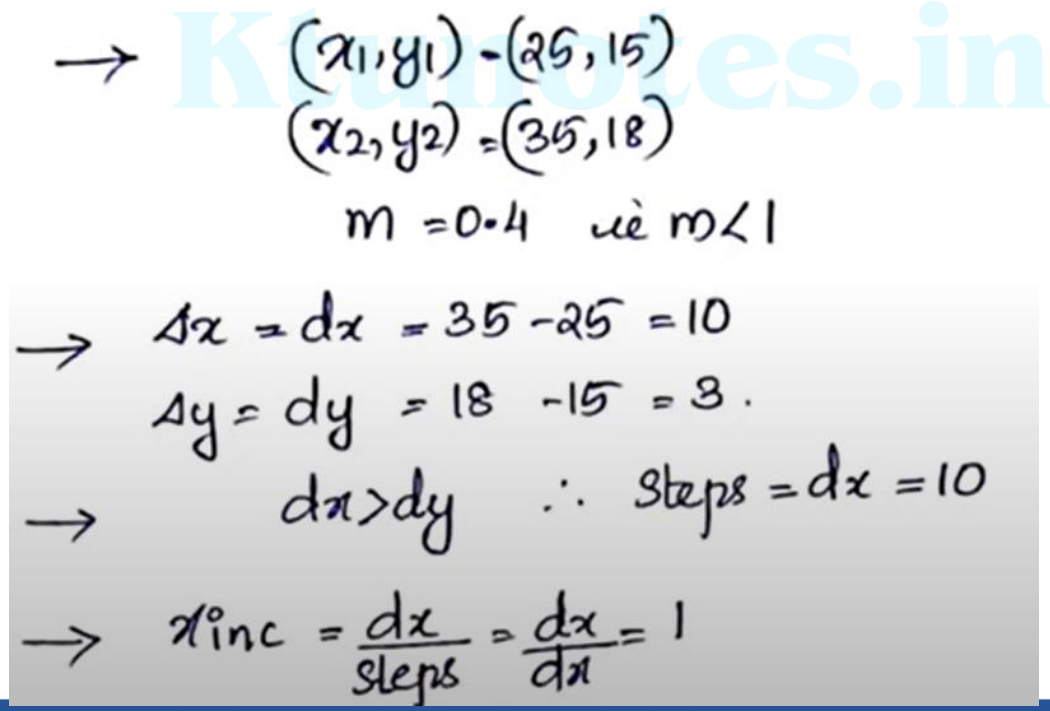
Line Drawing algorithms - DDA

```
#include "device.h"

#define ROUND(a) ((int)(a+0.5))

void lineDDA (int xa, int ya, int xb, int yb)
{
  int dx = xb - xa, dy = yb - ya, steps, k;
  float xIncrement, yIncrement, x = xa, y = ya;

  if (abs (dx) > abs (dy)) steps = abs (dx);
  else steps = abs (dy);
  xIncrement = dx / (float) steps;
  yIncrement = dy / (float) steps;

  setPixel (ROUND(x), ROUND(y));
  for (k=0; k<steps; k++) {
    x += xIncrement;
    y += yIncrement;
    setPixel (ROUND(x), ROUND(y));
  }
```

Line Drawing algorithms - DDA

- Problem:

Use DDA line drawing algorithm to plot the line with end points (25,15) and (35,18).



Line Drawing algorithms - DDA

→ $\Delta x = dx = 35 - 25 = 10$

$\Delta y = dy = 18 - 15 = 3.$

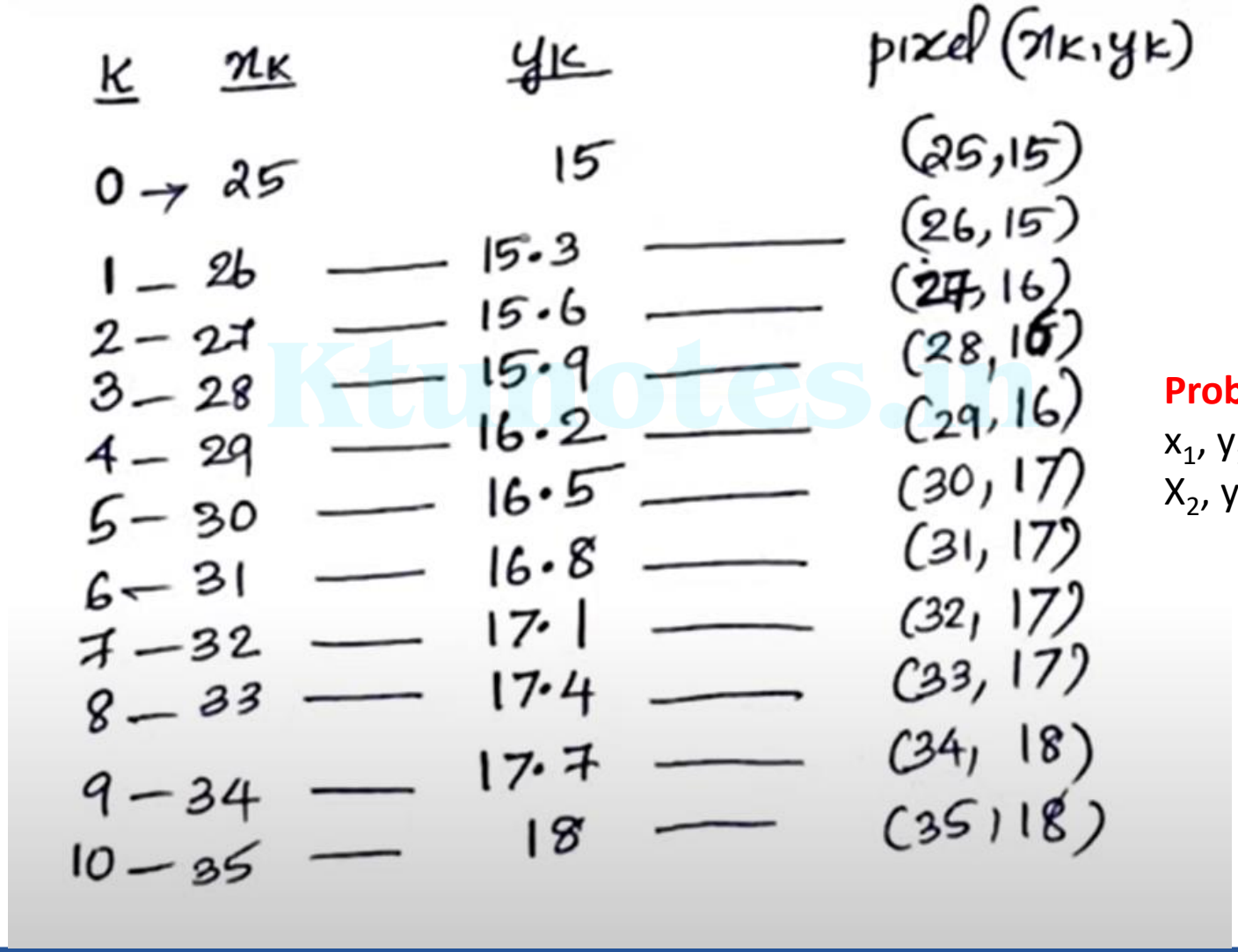
→ $dx > dy \quad \therefore \text{ Steps} = dx = 10$

→ $x_{inc} = \frac{dx}{\text{Steps}} = \frac{dx}{dx} = 1$

$y_{inc} = \frac{dy}{\text{Steps}} = \frac{dy}{dx} = \frac{3}{10} = 0.3$

→ $x = x + x_{inc} \qquad y = y + y_{inc}$

Line Drawing algorithms - DDA



Problem 2:

$x_1, y_1 = (15,25)$

$X_2, y_2 = (18,35)$

SAINTGITS
LEARN.GROW.EXCEL

Bresenham's Line Algorithm

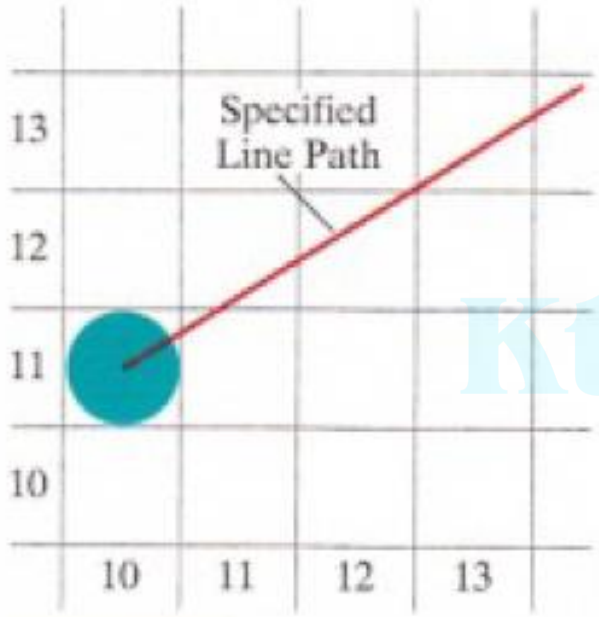


FIGURE 3-8 A section of a display screen where a straight-line segment is to be plotted, starting from the pixel at column 10 on scan line 11.

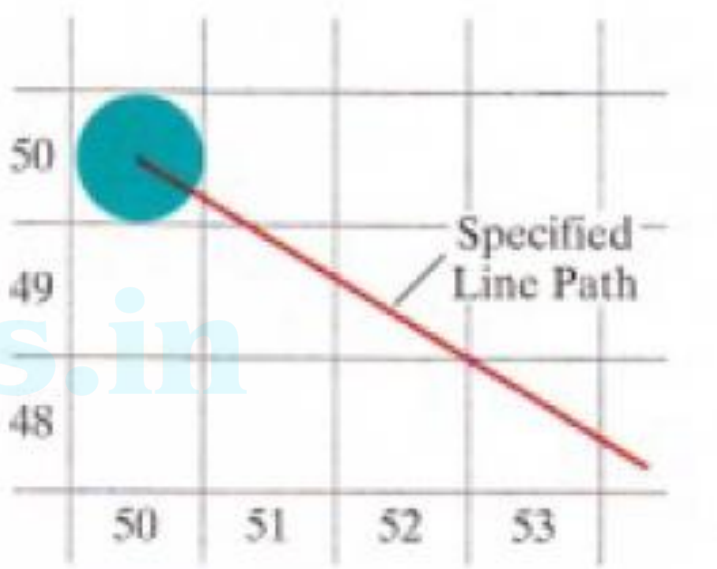
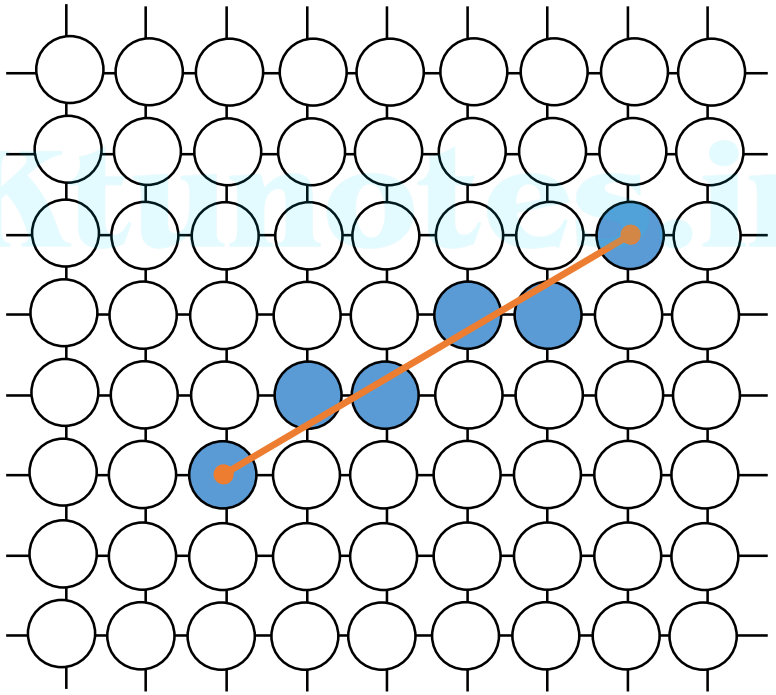


FIGURE 3-9 A section of a display screen where a negative slope line segment is to be plotted, starting from the pixel at column 50 on scan line 50.

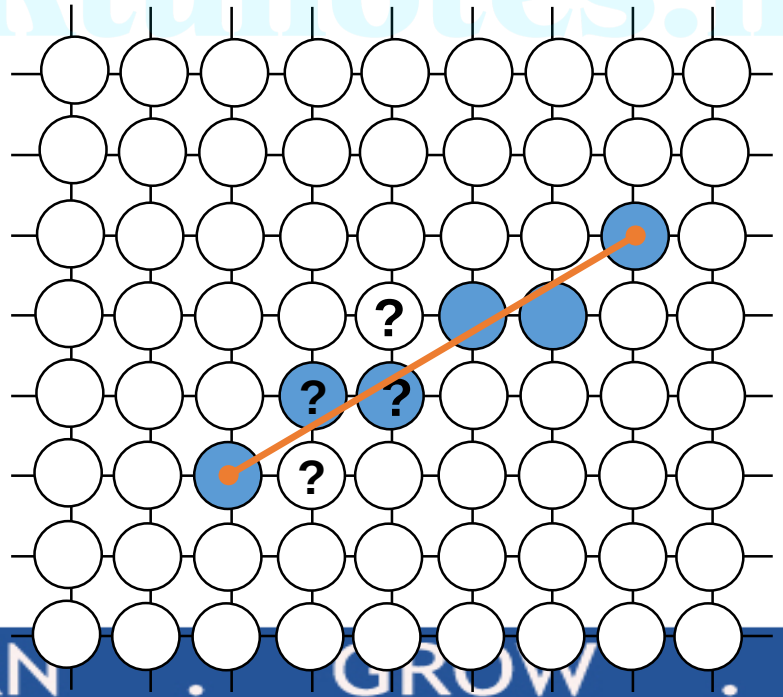
The Problem (cont…)

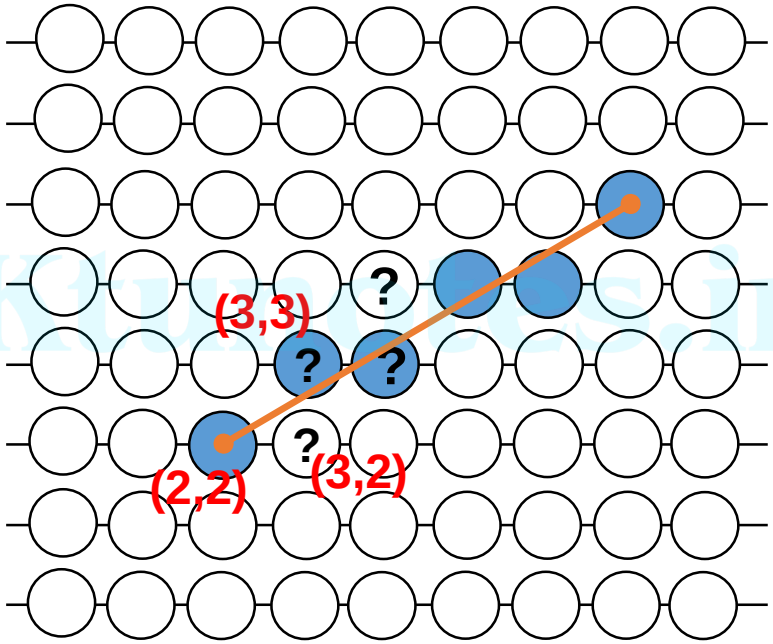
- What happens when we try to draw this on a pixel based display?



How do we choose which pixels to turn on?

- In Bresenham's line drawing algorithm the incremental integer calculations are used to scan convert the lines, that can be adapted to display circles and the other curves as well.





- .The next sample positions can be plotted either at (3,2) or (3,3) ?

SAINTGITS
LEARN.GROW.EXCEL

Bresenham's line algorithm

Jack Bresenham worked for 27 years at IBM before entering academia. Bresenham developed his famous algorithms at IBM in the early 1960s

Line Drawing algorithms – Bresenham's

Bresenham's Line drawing Algorithm

- An accurate and efficient raster line-generating algorithm, developed by Bresenham.
- First consider the scan-conversion process for lines with positive slope less than 1.
- Pixel positions along a line path are then determined by sampling at unit ***x*** intervals.
- Starting from the left endpoint (x_0, y_0) of a given line, step to each successive column (***x*** position) and plot the pixel whose scan-line y value is closest to the line path.

To illustrate Bresenham's approach, we first consider the scan-conversion process for lines with positive slope less than 1.0. Pixel positions along a line path are then determined by sampling at unit x intervals. Starting from the left endpoint (x_0, y_0) of a given line, we step to each successive column (x position) and plot the pixel whose scan-line y value is closest to the line path. Figure 3-10 demonstrates the kth step in this process. Assuming we have determined that the pixel at (x_k, y_k) is to be displayed, we next need to decide which pixel to plot in column $x_{k+1} = x_k + 1$. Our choices are the pixels at positions $(x_k + 1, y_k)$ and $(x_k + 1, y_k + 1)$.

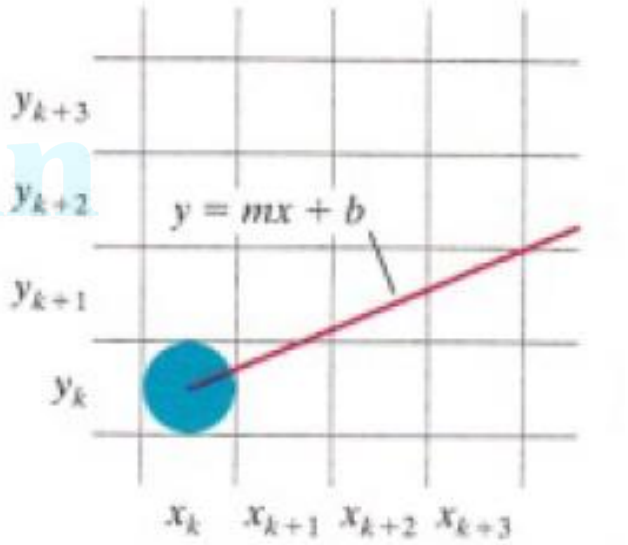


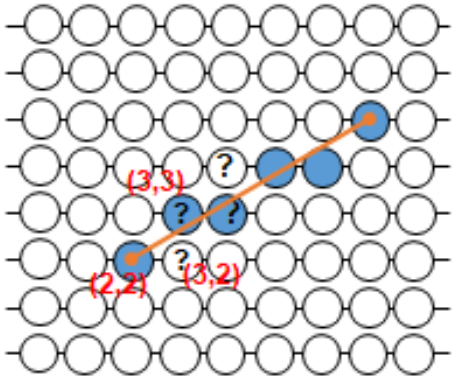
FIGURE 3-10 A section of the screen showing a pixel in column x_k on scan line y_k that is to be plotted along the path of a line segment with slope $0 < m < 1$.

For lines with positive slope m<1

- The pixel positions on line can be identified by doing sampling along x axis at unit intervals.
- The process of sampling begins from the pixel position (X_0,Y_0) and proceeds by plotting the pixels whose 'Y' value is nearest to the line path.

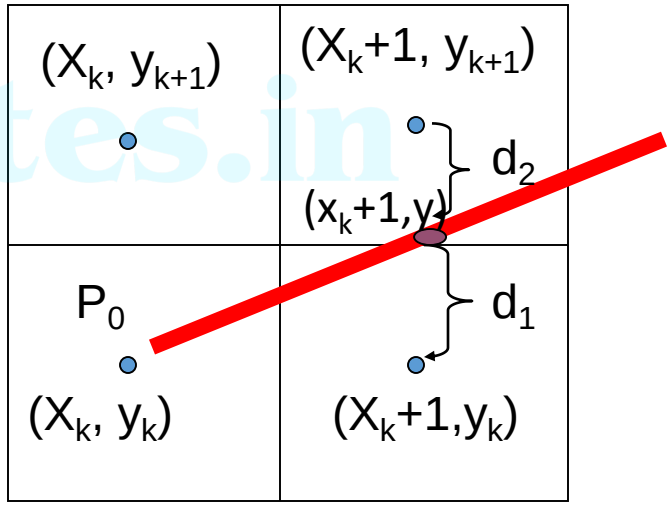
- If the pixel to be displayed occurs at a position (X_k, Y_k) then the next pixel is either at (X_k+1, Y_k) or (X_k+1, Y_{k+1}) i.e, (3,2) or (3,3)
- The 'Y' coordinate at the pixel position X_{k+1} can be obtained from

$Y=m(X_k+1)+b$ . . . . . eq 1



Bresenham's - for lines with positive slope m<1

- the separation between (X_k+1,Y_k) and (X_k+1,Y) is d1 and the separation between (X_k+1,Y) and (X_k+1, Y_{k+1}) is d2 then
- $d1 = y - y_k$ and
- $d2 = (Y_{k+1}) - Y$



Bresenham's - for lines with positive slope m<1

$y=m(X_k+1)+b$ . . . . eqn 1

$d1=y - y_k$ .....eqn 2

$d1=m(x_k+1)+b - Y_k$ (from eqn (1) .....(2)

And $d2= (Y_k+1) - Y$

$=(Y_K+1) - [m(X_k+1)+b]$

$=(Y_K+1) - m(X_k+1) - b$ .....(3)

The difference is given as

d1-d2 =

$= m(X_k+1)+b-Y_k-[(Y_k+1)-m(X_k+1)-b]$

$=m(X_k+1)+b-Y_k-(Y_k+1)+m(X_k+1)+b$

$d1-d2 = 2m(X_k+1)-2Y_k+2b - 1$ .....(4)

Contd..

A decision parameter P_k can be obtained by substituting m= dy/dx in equation 4

$$d1 - d2 = 2m(X_k+1)-2Y_k+2b - 1$$

$$= 2\, dy/dx\, (X_k+1) - 2Y_k + 2b - 1$$

$$= \frac{2\, dy(X_k+1)-2\, dx.Y_k + 2b.dx - dx}{dx}$$

$$dx(d1-d2) = 2\, dy(X_k+1)-2\, dx.Y_k + 2b.dx - dx$$

$$= 2\, dyX_k+2\, dy-2\, dx.Y_k + 2b.dx - dx$$

$$= 2\, dyX_k- 2\, dx.Y_k + c$$

Where, $dx(d1-d2) = P_k$ and $c= 2\, dy+ dx(2b-1)$

$$P_k = 2\, dyX_k- 2\, dx.Y_k + c \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

The value of c is constant and is independent of the pixel position. It can be deleted in the recursive calculations, of for P_k

if $d1 < d2$ then, P_k is negative (i.e, Y_k is nearer to the line path than Y_{k+1})

If P_k is –ve, a lower pixel (Y_k)is plotted else, an upper pixel (Y_k+1)is plotted.

At k+1 step, the value of Pk is given as

$$P_{K+1} = 2\, dyX_{k+1}- 2\, dx.Y_{k+1} + c \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6) \text{ (from 5)}$$

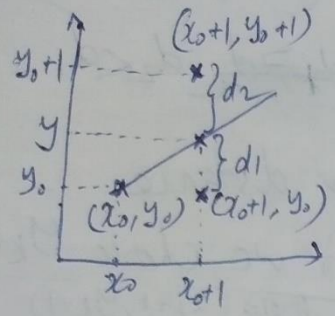
- Eq 6 – eq 5
- $P_{k+1} - P_k = (2\,dyX_{k+1} - 2\,dx.Y_{k+1} + c) - (2\,dyX_k + 2\,dx.Y_k + c)$
- $= 2dy(X_{k+1} - X_k) - 2\,dx(Y_{k+1} - Y_k)$ .....**(7)**

- Since $X_{k+1} = X_k + 1$ The eqn 7 becomes

- $P_{k+1} - P_k = 2dy(X_k + 1 - X_k) - 2\,dx(Y_{k+1} - Y_k)$
- $= 2dy - 2\,dx(Y_{k+1} - Y_k)$
- $P_{k+1} = P_k + 2dy - 2\,dx(Y_{k+1} - Y_k)$ .....**(8)**

- Where $(Y_{k+1} - Y_k)$ is either 0 or 1 based on the sign of P_k (if P_k is –ve, Y_k is not incremented: $Y_{k+1} = Y_k$)
- **The starting parameter P_0 at the pixel position (X_0,Y_0) is given as**
- **$P_0 = 2dy - dx$** .....**(9)**

Col 1	Col 2
(X_0, y_0+1)	(X_0+1, y_0+1)
P_0 (X_0, y_0)	(x_0+1,y) d_2 d_1 (X_0+1,y_0)



$d_1 = y - y_0$, $d_2 = y_0+1-y$

We know that $P_k = \Delta x(d_1 - d_2)$

The line pass through (x_0, y_0) and (x_0+1, y)

$P_0 = \Delta x(y - y_0 - y_0 - 1 + y)$

$= \Delta x[2y - 2y_0 - 1]$

$= \Delta x[2(y - y_0) - 1]$

From line equation, $y_0 = mx_0 + c$

$y = m(x_0+1) + c$

$\therefore P_0 = \Delta x[2(y - y_0) - 1]$

$= \Delta x[2(mx_0 + m + c - mx_0 - c) - 1]$

$= \Delta x[2m - 1]$

$= \Delta x\left[2\frac{\Delta y}{\Delta x} - 1\right]$

$m = \Delta y / \Delta x$

$\boxed{P_0 = 2\Delta y - \Delta x}$

Bresenham's algorithm

- Step 1: Enter the 2 end points for a line and store the left end point in (X_0,Y_0).
- Step 2: Plot the first point by loading (X_0,Y_0) in the frame buffer.
- Setp 3: determine the initial value of the decision parameter by calculating the constants dx, dy, 2dy and 2dy-2dx as
- $P_0 = 2dy - dx$
- Step 4: for each X_k, conduct the following test, starting from k= 0

- If Pk <0, then the next point to be plotted is at (X_{k+1}, Y_k) and
- $P_{k+1} = P_k + 2dy$
- Note: $P_{k+1} = P_k + 2dy - 2\,dx(Y_{k+1} - Y_k)$ ....here y values remain the same. Hence $Y_{k+1} - Y_k = 0$
- Else, the next point is (X_{k+1}, Y_{k+1}) and
- $P_{k+1} = P_k + 2dy - 2dx$ (step 3)
- Step 5: iterate through step (4) dx times.

Example

- Let the given end points for the line be (30,20) and (40, 28)
- $m = \frac{dy}{dx} = \frac{y2 - y1}{x2 - x1} = \frac{28 - 20}{40 - 30} = \frac{8}{10}$
- $m = 0.8$

  $dy = 8$ and $dx = 10$

- The initial decision parameter P_0 is
- $P_0 = 2dy - dx = 2(8) - 10 = 16 - 10 = 6$

  $P_0 = 6$

- The constants 2dy and 2dy-2dx are
- $2dy = 2(8) = 16$ $\qquad$ $2dy-2dx = 2(8)- 2(10) =16 - 20 = -4$

  $2dy = 16$ $\qquad$ $2dy - 2dx = -4$

The starting point $(x_0, y_0)=(30,20)$ and the successive pixel positions are given in the following table

dy = 8 and dx = 10 If P_k is +ve, $P_{k+1} = P_k + 2dy - 2dx = P_k - 4$ If P_k is –ve, $P_{k+1} = P_k + 2dy = P_k + 16$

K	P_k	(X_{k+1}, Y_{k+1})
0	6	(31,21)
1	2	(32,22
2	-2	33,22
3	14	34,23
4	10	35,24
5	6	36,25
6	2	37,26
7	-2	38,26
8	14	39,27
9	10	40,28

- End points (30,20) (40,28)
- dx=x2-x1=40-30=10 dy= y2-y1=28-20=8
- P_0=2dy-dx=2X8-10 = 6 >0 pt(31,21)
- P_{k+1}= p_k+2dy-2dx= 6+16-20 =2 >0 (32,22)
- P_{k+1}=2+16-20 =-2 <0 (33,22)
- P_{k+1}= p_k+2dy =-2+16 = 14 >0 (34,23)
- P_{k+1}= p_k+2dy-2dx=14+16-20=10>0 (35,24)
- P_{k+1}=10+16-20=6 >0 (36,25)
- P_{k+1}= 6+16-20=2>0 (37,26)
- P_{k+1}=2+16-20=-2 <0 (38,26)
- P_{k+1}= p_k+2dy=-2+16=14>0 (39,27)
- P_{k+1}=p_k+2dy-2dx= 14+16-20=10>0 (40,28)

- In bresenham's algorithm, if the positive slope of a line is greater than 1, the roles of x and y are interchanged.
- For positive slope lines:

1. If the initial position of a line is the right end point, then both x and y are decremented as we move from right to left.
2. If d1 = d2 then select the upper or the lower candidate pixel.

- For negative slope lines:
- One coordinate increases and the other coordinate decreases.
- Special cases:
- the vertical lines dx = 0, horizontal lines dy = 0 and diagonal lines |dx| = |dy| can be directly loaded into the frame buffer.

```
Void LineBres( int x1, int y1, int x2, int y2)
{
 int dx =abs(x2-x1), dy= abs(y2-y1)
 int p=2* dy- dx ;
 int x, y, Xend; // Xend is similar to steps in DDA
 // Determine which point to start and which as end
If(x1>x2)
  {
      x=x2;y=y2;Xend=x1;
   }
 else
  {
    x=x1;y=y1;Xend=x2;
  }
 setPixel( x, y );
```

```
While( x < Xend)
{
   x++;
   if( p < 0)
      p=p+2dy;
    else
     {
       y++;
       p=p+2dy-2dx;
     }
    setPixel(x,y);
  }   // End While

}   //End LineBres
```

Advantages of Bresenham's Alg.

- It uses only integer calculations
- It is faster than DDA

Circle Drawing Algorithms

- Circle – As et of all points in a plane which are at a fixed distance from a fixed point
- Circle Equation:

(Polynomial) -

When center is at (x_c, y_c) => $(x-x_c)^2 + (y-y_c)^2 = r^2$

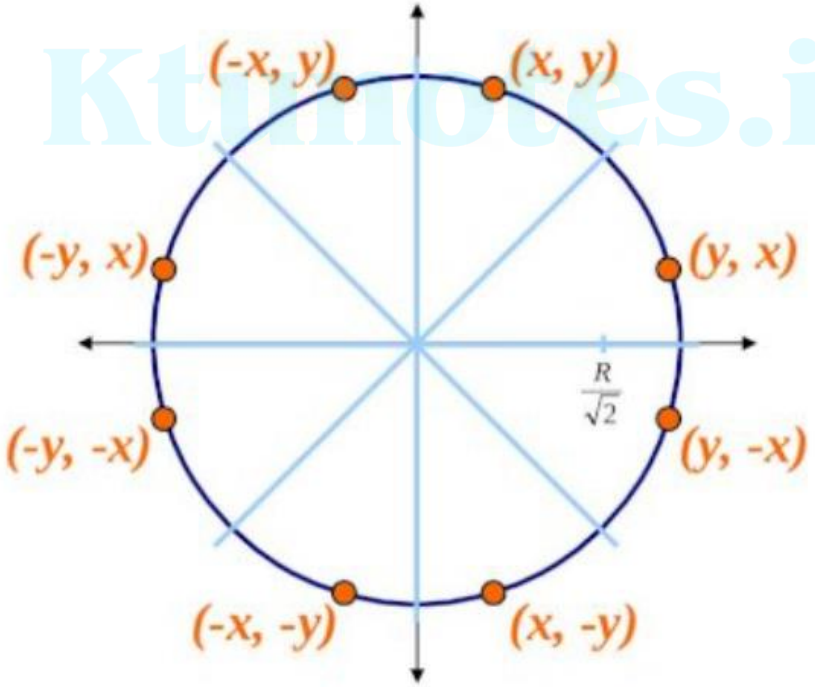
$(0,0)$ => $x^2 + y^2 = r^2$

Trigonometric (Polar coordinate) –

$x = r \cos\Theta$ and $y = r \sin\Theta$

Circle Drawing Algorithms

The first thing we can notice to make our circle drawing algorithm more efficient is that circles centred at $(0, 0)$ have *eight-way symmetry*



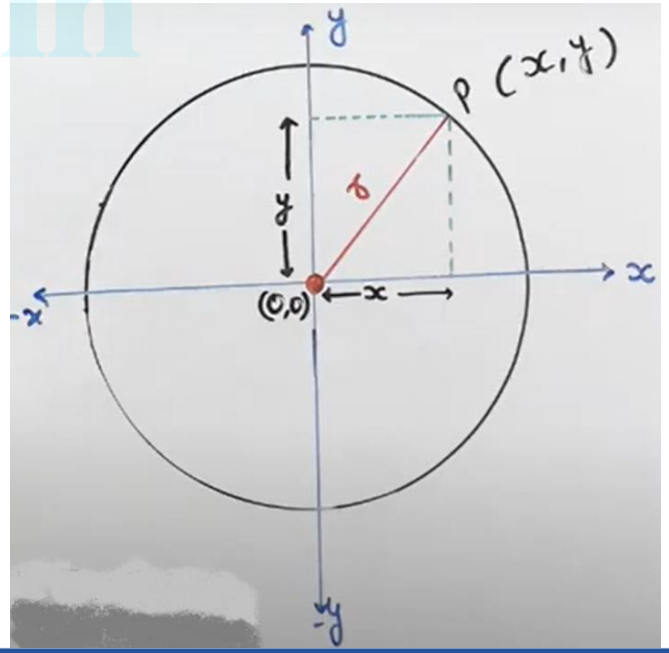
Circle Drawing Algorithms

Polynomial Method of Circle representation

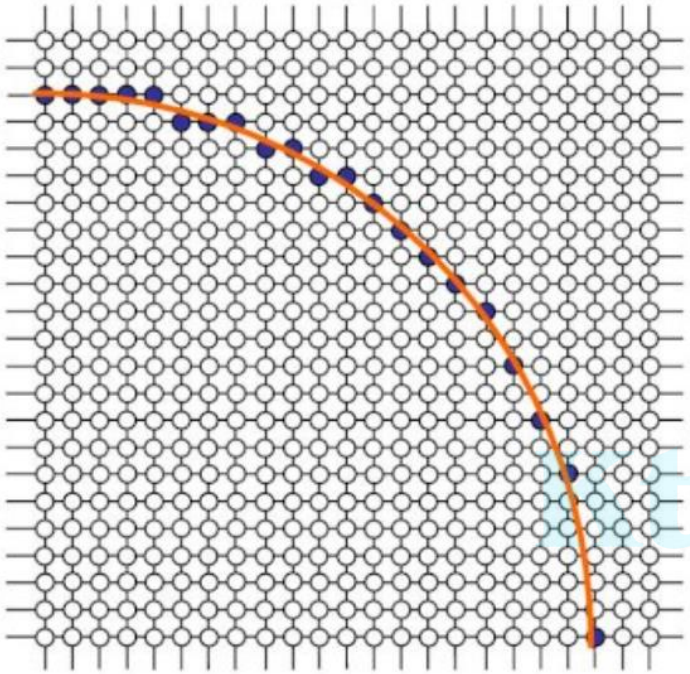
In this method, Circle is represented by a Polynomial Equation: $x^2 + y^2 = r^2$
Here, this equation can be used to find 'y' coordinates for the known 'x' coordinates – 'r' is also known.

The scan converting circle using this method is achieved by stepping 'x' from 0 to r/√2 and each 'y' coordinate value Is found by evaluating $\sqrt{(r^2-x^2)}$ for each step of 'x'.

This will generate 1/8 portion (90^0 to 45^0) of the circle. Remaining for other octants are generated the circle symmetry principle.



Circle Drawing Algorithms



$$y_0 = \sqrt{20^2 - 0^2} \approx 20$$

$$y_1 = \sqrt{20^2 - 1^2} \approx 20$$

$$y_2 = \sqrt{20^2 - 2^2} \approx 20$$

$$\vdots$$

$$y_{19} = \sqrt{20^2 - 19^2} \approx 6$$

$$y_{20} = \sqrt{20^2 - 20^2} \approx 0$$

However, unsurprisingly this is not a brilliant solution!

Firstly, the resulting circle has large gaps where the slope approaches the vertical

Secondly, the calculations are not very efficient

- The square (multiply) operations
- The square root operation – try really hard to avoid these!

We need a more efficient, more accurate solution

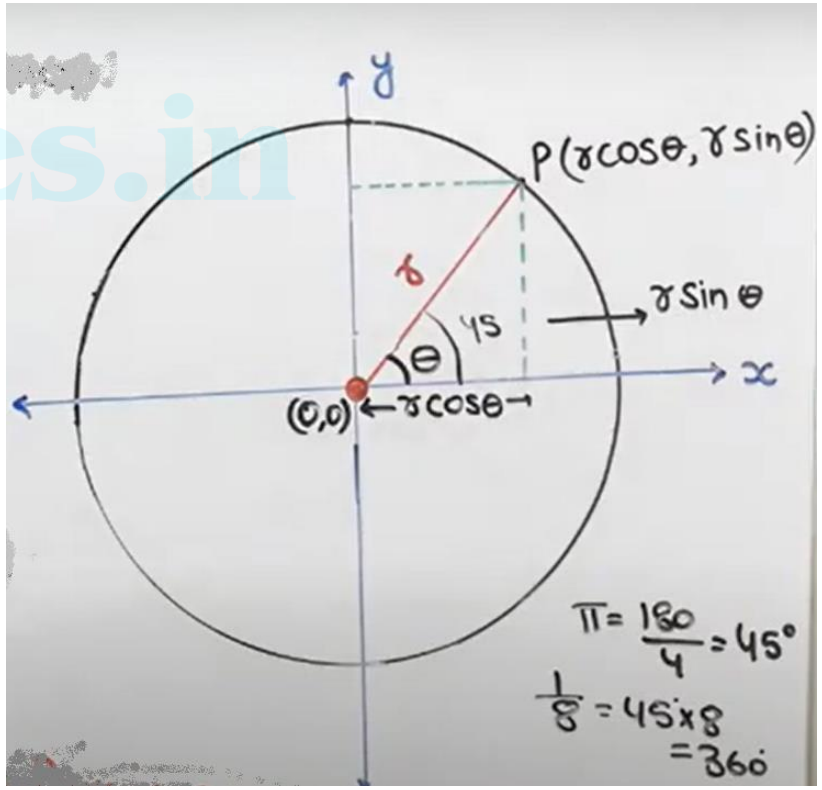
Circle Drawing Algorithms

- Trigonometric / Polar Coordinate Method:

$x = r \cos\Theta$ and $y = r \sin\Theta$

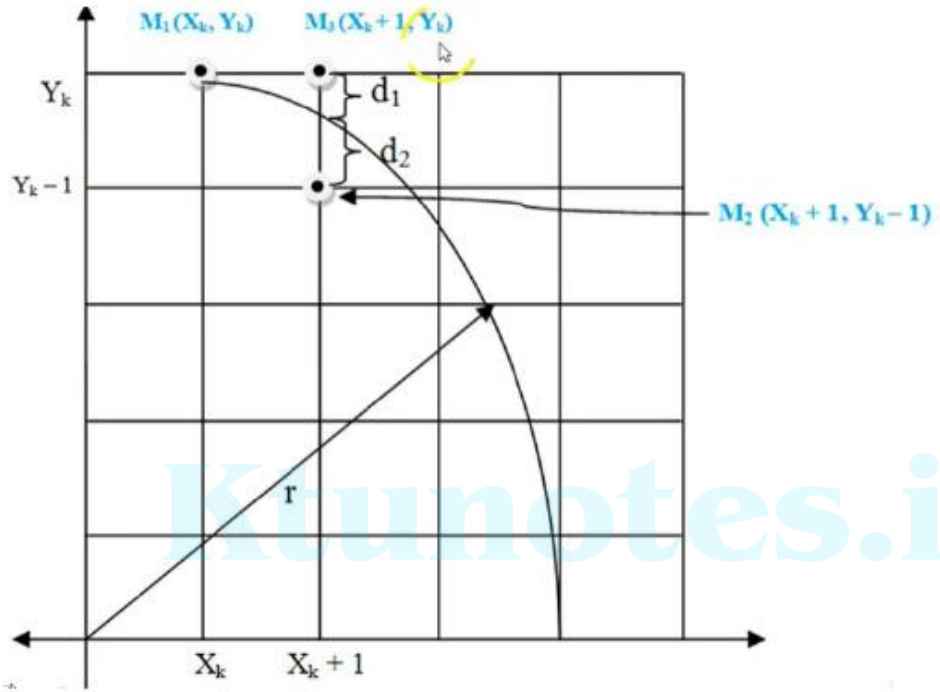
where Θ is the current angle and x and y are the respective x and y coordinates.

The scan converting circle using Polar coordinate method is achieved by stepping Θ from 0 to 45^0 and calculating each x and y value. Using 1/8 symmetry concept remaining points in other octants are found out.



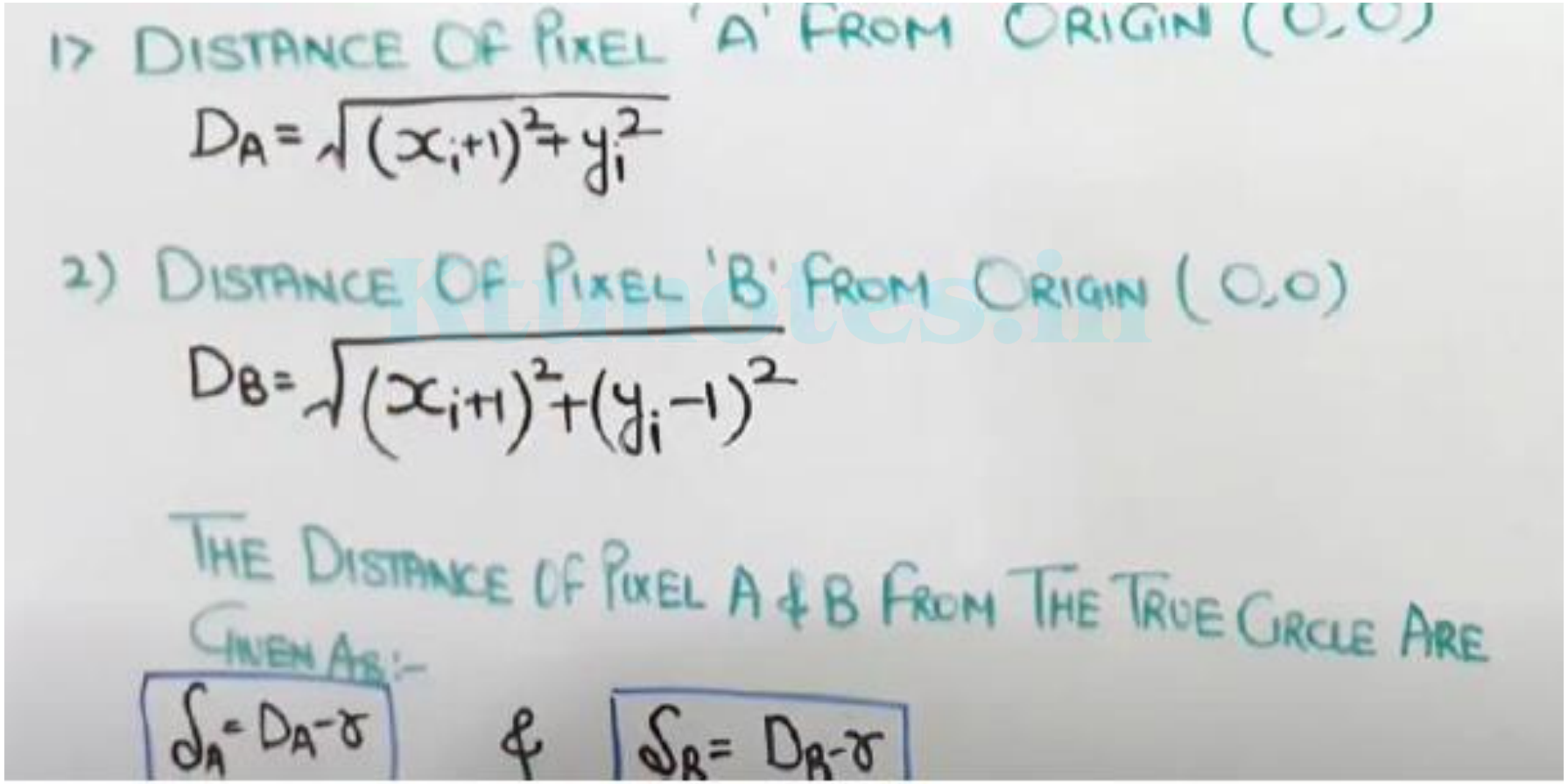
This method is also quite inefficient and time consuming because of the repeated computation of sinΘ and cosΘ.

Bresenham's Circle Drawing Algorithm



This is an efficient circle drawing algorithm because it avoids square root and trigonometric calculations. It involves only integer operations. $1/8^{th}$ of a circle is plotted from 90^o to 45^o where x moves in the positive direction and y in the negative direction. 8 way symmetry of the circle is considered to generate the remaining points.

Bresenham's Circle Drawing Algorithm



Circle equation can also be applied at A and B to get d_a and d_b respectively.

Bresenham's Circle Drawing Algorithm

- Circle function is,

$$f_{circle}(X, Y) = X^2 + Y^2 - r^2$$

- Any point (X, Y) on the boundary of the circle with radius r satisfies the equation $f_{circle}(X, Y) = 0$

- In general the relative position of any point (X, Y) can be determined by sign of the circle function

$$f_{circle}(X, Y) \begin{cases} < 0, (x, y) \text{ inside circle} \\ = 0, (x, y) \text{ on the circle} \\ > 0. (x, y) \text{ outside circle} \end{cases}$$

Bresenham's Circle Drawing Algorithm

d_1 = distance of M_3 from circle
d_2 = distance of M_2 from circle

Circle eqn. can be applied at points A and B to get d1 and d2 respectively.

$$\therefore d_1 = (X_k + 1)^2 + Y_k^2 - r^2 \quad \text{[always +ve]}$$
$$\therefore d_2 = (X_k + 1)^2 + (Y_k - 1)^2 - r^2 \quad \text{[always –ve]}$$

- Decision parameter $P_k = d_1 + d_2$

$$P_k = (X_k + 1)^2 + Y_k^2 - r^2 + (X_k + 1)^2 + (Y_k - 1)^2 - r^2$$
$$= 2(X_k + 1)^2 + Y_k^2 + (Y_k - 1)^2 - 2r^2 \quad \text{---------(1)}$$

- So, if $P_k < 0$ then circle is closer to M_3(above point),
  $(X_{k+1}, Y_k) = (X_k+1, Y_k)$

- Otherwise, $P_k \geq 0$ then circle is closer to M_2(below point),
  $(X_{k+1}, Y_k) = (X_k+1, Y_k-1)$

Bresenham's Circle Drawing Algorithm

- Now, we will see how to calculate the next pixel location from a previously known pixel location (x, y).
- In Bresenham's algorithm at any point (x, y) we have two option either to choose the next pixel as (x+1, y) or (x+1, y-1).
- this can be decided by using the decision parameter p_k as:

- If $p_k > 0$, then (x+1, y-1) is to be chosen as the next pixel as it will be closer to the arc.
- else (x+1, y) is to be chosen as next pixel.

Bresenham's Circle Drawing Algorithm

- To find next decision parameter, P_{k+1}

$$P_{k+1} = 2(X_{k+1} + 1)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 - 2r^2 \quad \text{------(2)}$$

- Difference between equation (2) – (1)

$$P_{k+1} - P_k = 2(X_{k+1} + 1)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 - 2r^2 - 2(X_k + 1)^2 - Y_k^2 - (Y_k - 1)^2 + 2r^2$$

$$= 2(X_k + 1 + 1)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 - 2(X_k + 1)^2 - Y_k^2 - (Y_k - 1)^2 \quad (\text{where } X_{k+1} = X_k + 1)$$

$$= 2(X_k + 2)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 - 2(X_k + 1)^2 - Y_k^2 - (Y_k - 1)^2$$

$$= 2(X_k^2 + 4X_k + 4) + Y_{k+1}^2 + Y_{k+1}^2 - 2Y_{k+1} + 1 - 2(X_k^2 + 2X_k + 1) - Y_k^2 - Y_k^2 + 2Y_k - 1$$

$$= 2X_k^2 + 8X_k + 8 + 2Y_{k+1}^2 - 2Y_{k+1} - 2X_k^2 - 4X_k - 2 - 2Y_k^2 + 2Y_k$$

$$= 4X_k + 6 + 2Y_{k+1}^2 - 2Y_{k+1} - 2Y_k^2 + 2Y_k$$

$$P_{k+1} = P_k + 4X_k + 2(Y_{k+1}^2 - Y_k^2) - 2(Y_{k+1} - Y_k) + 6$$

Bresenham's Circle Drawing Algorithm

- If $P_k < 0$ (M_3 is select) then $Y_{k+1} = Y_k$

$$P_{k+1} = P_k + 4X_k + 2(Y_k^2 - Y_k^2) - 2(Y_k - Y_k) + 6$$

$$P_{k+1} = P_k + 4X_k + 6$$

- If $P_k \geq 0$ (M_2 is select) then $Y_{k+1} = Y_k - 1$

$$\begin{aligned} P_{k+1} &= P_k + 4X_k + 2[(Y_k - 1)^2 - Y_k^2)] - 2[(Y_k - 1) - Y_k)] + 6 \\ &= P_k + 4X_k + 2[Y_k^2 - 2Y_k + 1 - Y_k^2] - 2[Y_k - 1 - Y_k] + 6 \\ &= P_k + 4X_k - 4Y_k + 2 + 2 + 6 \\ &= P_k + 4X_k - 4Y_k + 10 \end{aligned}$$

Bresenham's Circle Drawing Algorithm

$$d_1 = (X_k + 1)^2 + Y_k^2 - r^2$$
$$d_2 = (X_k + 1)^2 + (Y_k - 1)^2 - r^2$$

- Here initial decision parameter P_0,

$(x_k, y_k) = (0,r)$

$$P_0 = d_1 + d_2$$
$$= [1^2 + r^2 - r^2] + [1^2 + (r-1)^2 - r^2]$$
$$= 1 + 1 + r^2 - 2r + 1 - r^2$$
$$P_0 = 3 - 2r$$

Bresenham's circle drawing algorithm

Algorithm:

Step-1: Input radius r and circle centre (X_c, Y_c) obtained the first point $(X_0, Y_0) = (0, r)$

Step-2: Calculate the initial value of decision parameter as

$$P_0 = 3 - 2r$$

Step-3: At each X_k position, starting at k = 0, perform, the following test:

If $P_k < 0$, the next point is $(X_k + 1, Y_k)$

$$P_{k+1} = P_k + 4X_k + 6$$

otherwise the next point is $(X_k + 1, Y_k - 1)$

$$P_{k+1} = P_k + 4(X_k - Y_k) + 10$$

Step-4: Determine the symmetry points in other seven octant

Step-5: Move each pixel position (X, Y) into circular path

$$X = X + X_c \quad \text{and} \quad Y = Y + Y_c$$

Step-6: Repeat step 3 to 5 until $X \geq Y$.

Bresenham's Circle Drawing Algorithm

Example-1: Given a circle radius r = 10, we demonstrate the Bresenham circle drawing algorithm by determining position along the circle octant in the first quadrant from X= 0 to X = Y.

Solution:

– The initial decision parameter P_0

$\Rightarrow P_0 = 3 - 2r = 3 - 20 = -17 \quad \therefore \mathbf{P_0 < 0} \Rightarrow (X_1, Y_1) = \mathbf{(1, 10)}$

– For the circle centred on the coordinator origin, the initial point is $(X_0 , Y_0) = (0, 10)$ and initial increment terms for calculating the decision parameter are

$\Rightarrow P_1 = P_0 + 4X_0 + 6$

$= -17 + 0 + 6$

$= -11 \qquad \therefore \mathbf{P_1 < 0} \Rightarrow (X_2, Y_2) = \mathbf{(2, 10)}$

Bresenham's Circle Drawing Algorithm

$\Rightarrow P_2 = P_1 + 4X_1 + 6$

$= -11 + 4 + 6$

$= -1$ $\quad \therefore P_2 < 0 \Rightarrow (X_3, Y_3) = (3, 10)$

$\Rightarrow P_3 = P_2 + 4X_2 + 6$

$= -1 + 8 + 6$

$= 13$ $\quad \therefore P_3 > 0 \Rightarrow (X_4, Y_4) = (4, 9)$

$\Rightarrow P_4 = P_3 + 4(X_3 - Y_3) + 10$

$= 13 - 28 + 10$

$= -5$ $\quad \therefore P_4 < 0 \Rightarrow (X_5, Y_5) = (5, 9)$

$\Rightarrow P_5 = P_4 + 4X_4 + 6$

$= -5 + 16 + 6$

$= 17$ $\quad \therefore P_5 > 0 \Rightarrow (X_6, Y_6) = (6, 8)$

$\Rightarrow P_6 = P_5 + 4(X_5 - Y_5) + 10$

$= 17 - 16 + 10$

$= 11$ $\quad \therefore P_6 > 0 \Rightarrow (X_7, Y_7) = (7, 7)$

K	d_k	New Point (X_{k+1}, Y_{k+1})
0	–17	(1, 10)
1	–11	(2, 10)
2	–1	(3, 10)
3	13	(4, 9)
4	–5	(5, 9)
5	17	(6, 8)
6	11	(7, 7)

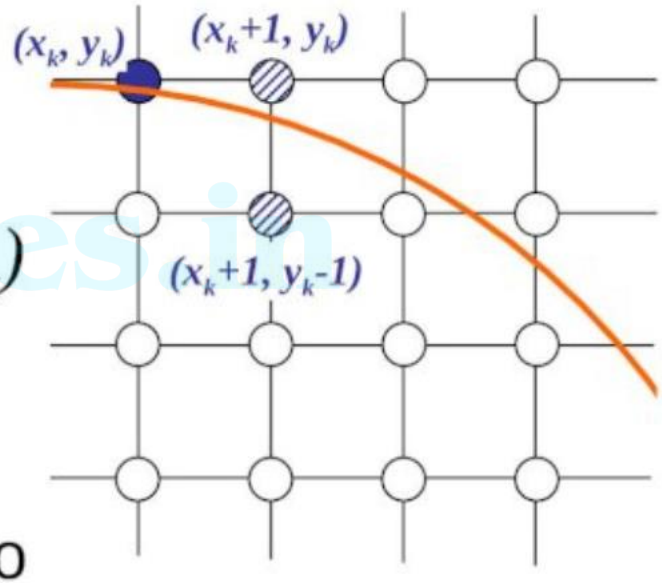
Mid-Point Circle Algorithm

Assume that we have just plotted point (x_k, y_k)

The next point is a choice between (x_k+1, y_k) and (x_k+1, y_k-1)

We would like to choose the point that is nearest to the actual circle

So how do we make this choice?



Mid-Point Circle Algorithm

- In the midpoint circle drawing algorithm, the midpoint between two pixels along the circle path is first found out.
- Using a decision variable, it is then checked whether the midpoint is inside or outside the circle path. If the midpoint is inside, the outside pixel is chosen and if it is outside, the inside pixel is chosen.
- Circle is plotted from 90 to 45 degrees and ‘x’ moves in the positive direction, while ‘y’ moves in the negative direction.
- Points are identified for one octet and the remaining points are found out using 8 way symmetry of the circle.

Mid-Point Circle Algorithm

Let's re-jig the equation of the circle slightly to give us:

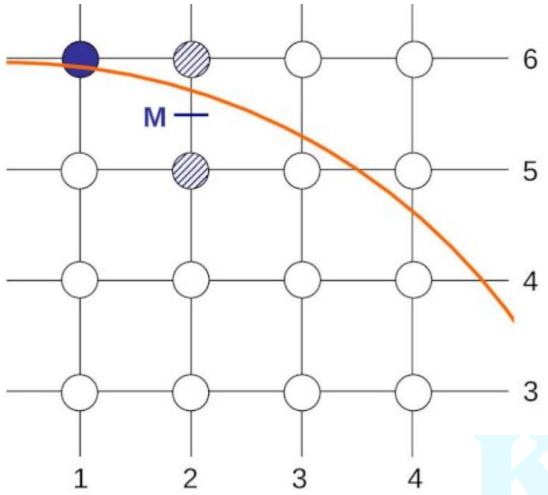
$$f_{circ}(x, y) = x^2 + y^2 - r^2 \quad \ldots(1)$$

The equation evaluates as follows:

$$f_{circ}(x, y) \begin{cases} < 0, \text{ if } (x, y) \text{ is inside the circle boundary} \\ = 0, \text{ if } (x, y) \text{ is on the circle boundary} \\ > 0, \text{ if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

By evaluating this function at the midpoint between the candidate pixels we can make our decision

Mid-Point Circle Algorithm



Assuming we have just plotted the pixel at (x_k,y_k) so we need to choose between (x_k+1,y_k) and (x_k+1,y_k-1)

Our decision variable can be defined as: mid point b/w 2 points (x_k+1,Y_k) and $(Xk+1, Y_k-1)$ is $[x_k+1, y_k-1/2]$

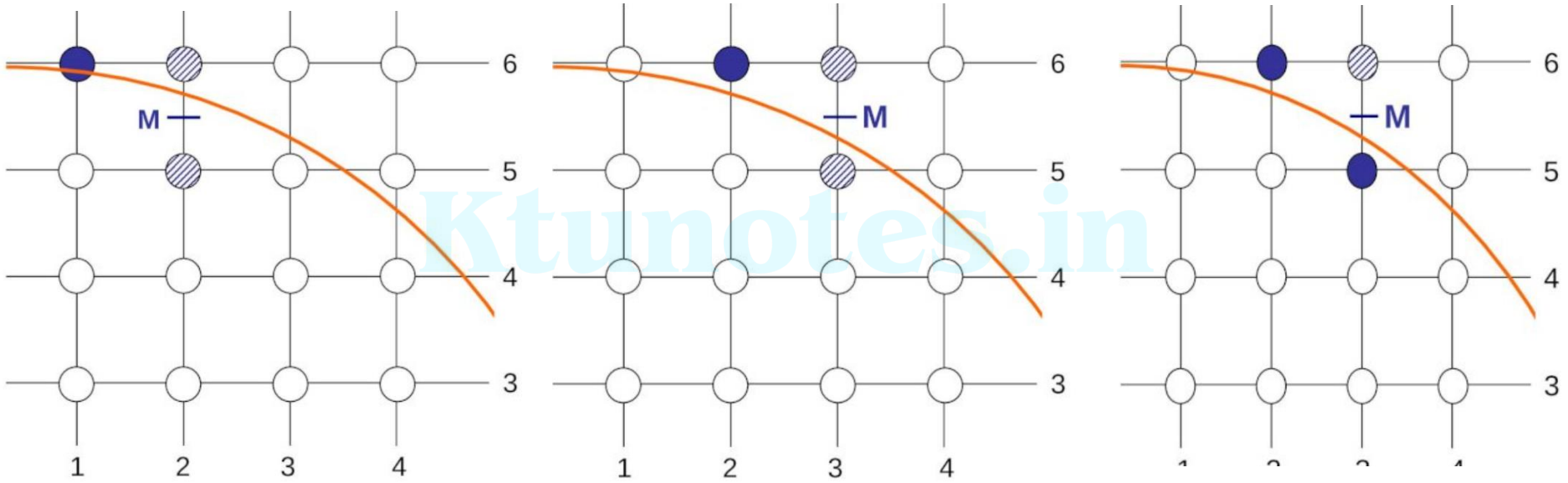
$$p_k = f_{circ}(x_k+1, y_k - \tfrac{1}{2})$$

$$=(x_k+1)^2+(y_k-\tfrac{1}{2})^2-r^2 \qquad \ldots 2$$

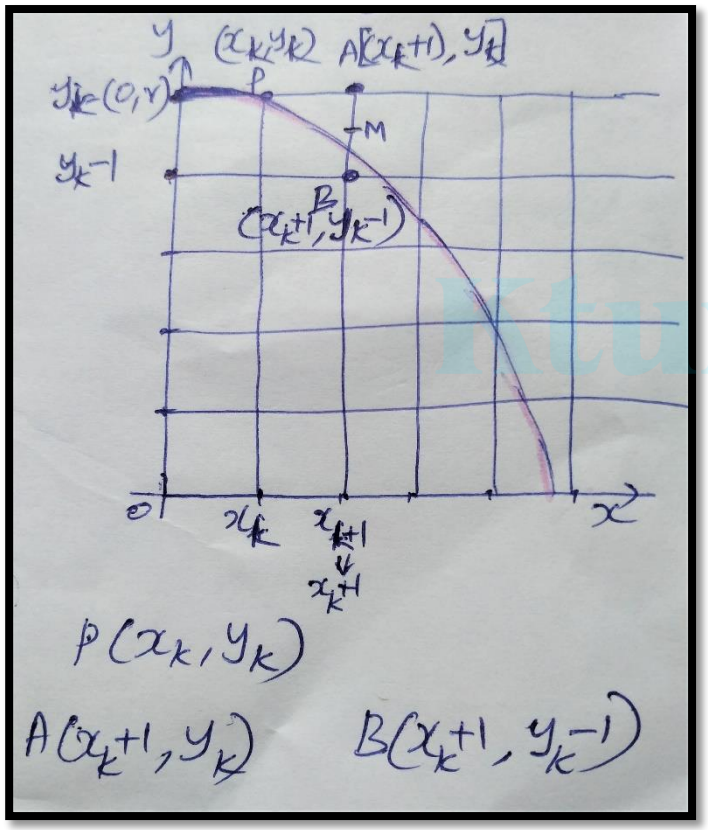
If $p_k < 0$ the midpoint is inside the circle and the pixel at y_k is closer to the circle

Otherwise the midpoint is outside and y_k-1 is closer

Mid-Point Circle Algorithm



Mid-Point Circle Algorithm



$$F_{circle}(x,y) \begin{cases} <0, & \text{if } (x,y) \text{ is inside the circle boundary} \\ =0; & \text{if } (x,y) \text{ is on the circle boundary} \\ >0; & \text{if } (x,y) \text{ is outside the circle boundary.} \end{cases}$$

$F_{circle}(x,y) = x^2 + y^2 - r^2$

Circle equation at midpoint 'm':

$$d_k = F_{circle}(x_k+1, y_k-\tfrac{1}{2}) = (x_k+1)^2 + (y_k-\tfrac{1}{2})^2 - r^2$$

$$\boxed{d_k = (x_k+1)^2 + y_k^2 - y_k + \tfrac{1}{4} - r^2}$$

The initial decision variable 'd_0' is the decision variable at (0,r). We start from 90° and move to 45°.

$$d_0 = F(0,r) = (0+1)^2 + r^2 - r + \tfrac{1}{4} - r^2 = 1 + \tfrac{1}{4} - r = 1.25 - r$$

$$\boxed{d_0 = 1.25 - r} \Rightarrow \text{If } r \text{ is an integer, it is rounded off as } \boxed{d_0 = 1 - r}$$

Next decision variable d_{k+1}:

$$d_k = (x_k+1)^2 + y_k^2 - y_k + \tfrac{1}{4} - r^2$$

$$d_{k+1} = (x_{k+1}+1)^2 + y_{k+1}^2 - y_{k+1} + \tfrac{1}{4} - r^2$$

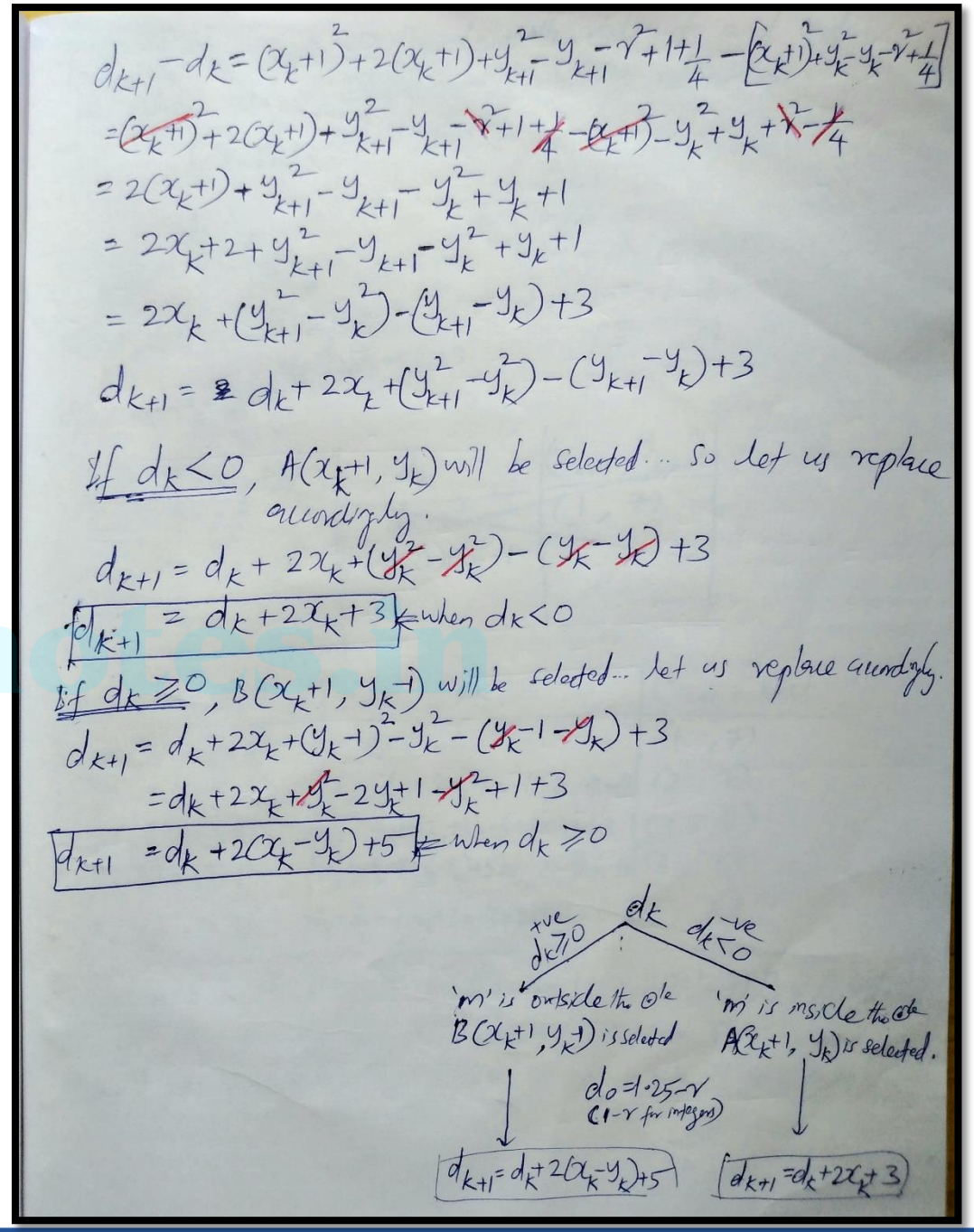
$$= [(x_k+1)+1]^2 + y_{k+1}^2 - y_{k+1} + \tfrac{1}{4} - r^2$$

$$= (x_k+1)^2 + 2(x_k+1) + 1 + y_{k+1}^2 - y_{k+1} + \tfrac{1}{4} - r^2$$

Now we have to calculate $d_{k+1} - d_k$ to find the increment for the next decision ~~parameter~~ variable.

SAINTGITS
LEARN.GROW.EXCEL

Mid-Point Circle Algorithm



Mid-Point Circle Algorithm

the initial value of Pk is given by the circle function at the position (0,r) as,

$$p_k = f_{circ}(x_k + 1, y_k - \tfrac{1}{2})$$
$$= (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2$$

Substituting k=0,Xk=0,Yk=r in the above function results in,

The first decision variable is given as:

$$p_0 = f_{circ}(1, r - \tfrac{1}{2})$$
$$= 1 + (r - \tfrac{1}{2})^2 - r^2$$
$$= \tfrac{5}{4} - r$$

if r is an integer, then Po can be rounded to P0= 1 – r.

P_o = (1.25 – r) or (**1 – r**) when r is an integer

Then if $p_k < 0$ then the next decision variable is given as: $p_{k+1} = p_k + 2x_{k+1} + 1$

$p_{k+1} = p_k + 2x_k + 3$

If $p_k > 0$ then the decision variable is: $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_k + 1$

$p_{k+1} = p_k + 2(x_k - y_k) + 5$

SAINTGITS
LEARN.GROW.EXCEL

MID-POINT CIRCLE ALGORITHM

- Input radius r and circle centre (x_c, y_c), then set the coordinates for the first point on the circumference of a circle centred on the origin as:

$$(x_0, y_0) = (0, r)$$

- Calculate the initial value of the decision parameter as:

$$p_0 = \frac{5}{4} - r$$

$P_0 = (1.25 - r)$
~ (1 - r) for integer r

- Perform the test, Starting with $k = 0$ at each position x_k, perform the following test.

(i) If $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and: $p_{k+1} = p_k + 2x_{k+1} + 1$

$p_{k+1} = p_k + 2x_k + 3$

(ii) If Pk >0 then the next point along the circle is (x_k+1, y_k-1) and:

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

$p_{k+1} = p_k + 2(x_k - y_k) + 5$

where $2x_{k+1} = 2X_k + 2$ and $2y_{k+1} = 2Y_k - 2$

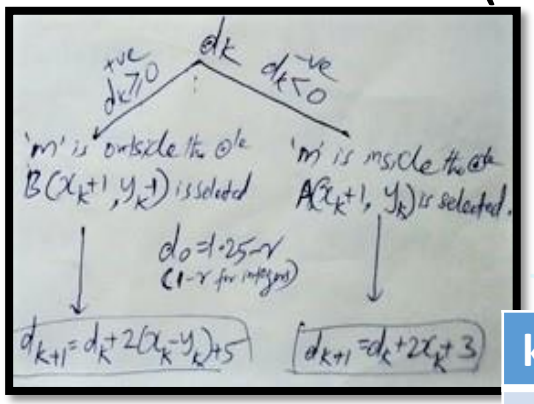
- Identify the symmetry points in the other seven octants
- Move (x, y) according to:

$$x = x + x_c \qquad y = y + y_c$$

- Repeat steps 3 to 5 until $x >= y$

Mid-Point Circle Algorithm

- Calculate the points of a circle with radius 7 and center at (10,10) using midpoint circle algorithm.



$(x_0, y_0) = (0,r) = (0,7)$
center $(x_c, y_c) = (10,10)$
$d_0 = 1-r = 1-7 = -6$

k	(x_k, y_k)	d_k	$(x_{k+1}+x_c, y_{k+1}+y_c)$
0	$(x_0,y_0)=(0,7)$	$d_0 = -6$	(1,7) => (11,17)
1	$(x_1,y_1)=(1,7)$	$d_1 = d_0+2x_0+3 = -6+2x0+3 = -3$	(2,7) => (12,17)
2	$(x_2,y_2)=(2,7)$	$d_2 = d_1+2x_1+3 = -3+2x1+3 = 2$	(3,6) => (13,16)
3	$(x_3,y_3)=(3,6)$	$d_3 = d_2+2(x_2-y_2)+5 = 2+2(-5)+5 = -3$	(4,6) => (14,16)
4	$(x_4,y_4)=(4,6)$	$d_4 = d_3+2x_3+3 = -3+2x3+3 = 6$	(5,5) => (15,15)

The points in this octet (90^o to 45^o) are (10,17), (11,17), (12,17), (13,16), (14,16), (15,15). At 45^o, x=y and we stop here. (We can stop when 'y' gets equal or less than x – or continue while y >= x)