Computer Graphics

Course Outline

This course is meant as an introduction to computer graphics , which covers a large body of work. The intention is to give a solid grounding in basic 2D computer graphics and introduce the concepts and some techniques required to implement 3D graphics. CURVES & SURFACES,PROJECTIONS & HIDDEN SURFACE REMOVAL,SHADING & COLOR ISSUES,FRACTALS & ANIMATION.

Introduction to Computer Graphics

- Computers graphics has become a powerful tool in most of the areas of science, engineering and education.
- The term Computer Graphics refers to the interface which helps a user to understand and control all the operations on a computer system.
- Computer Graphics has become a key technology for communicating information (data) and ideas etc. in modern world.

Factors

- How pictures or graphics objects are presented in computer graphics?
- How pictures or graphics objects are prepared for presentation?
- How previously prepared pictures or graphics objects are prepared?
- How interaction with the picture or graphics objects is accomplished?

Why Study Computer Graphics

- â Entertainment - computer animation;
- â User interfaces;
- â Interactive visualization - business and science;
- â Cartography;
- â Medicine;
- â Computer aided design;
- â Multimedia systems;
- â Computer games;
- â Image processing.

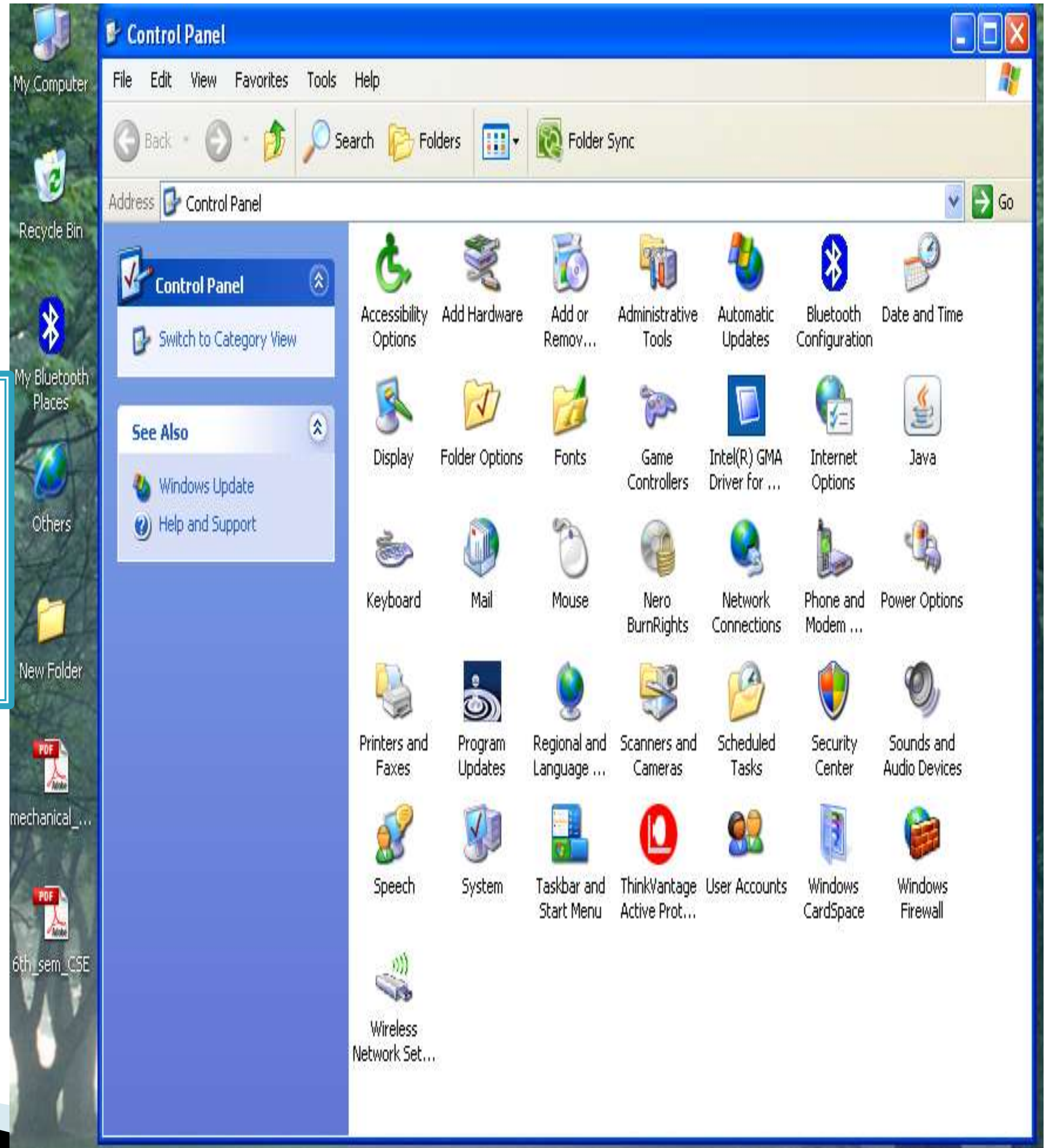
Advantages

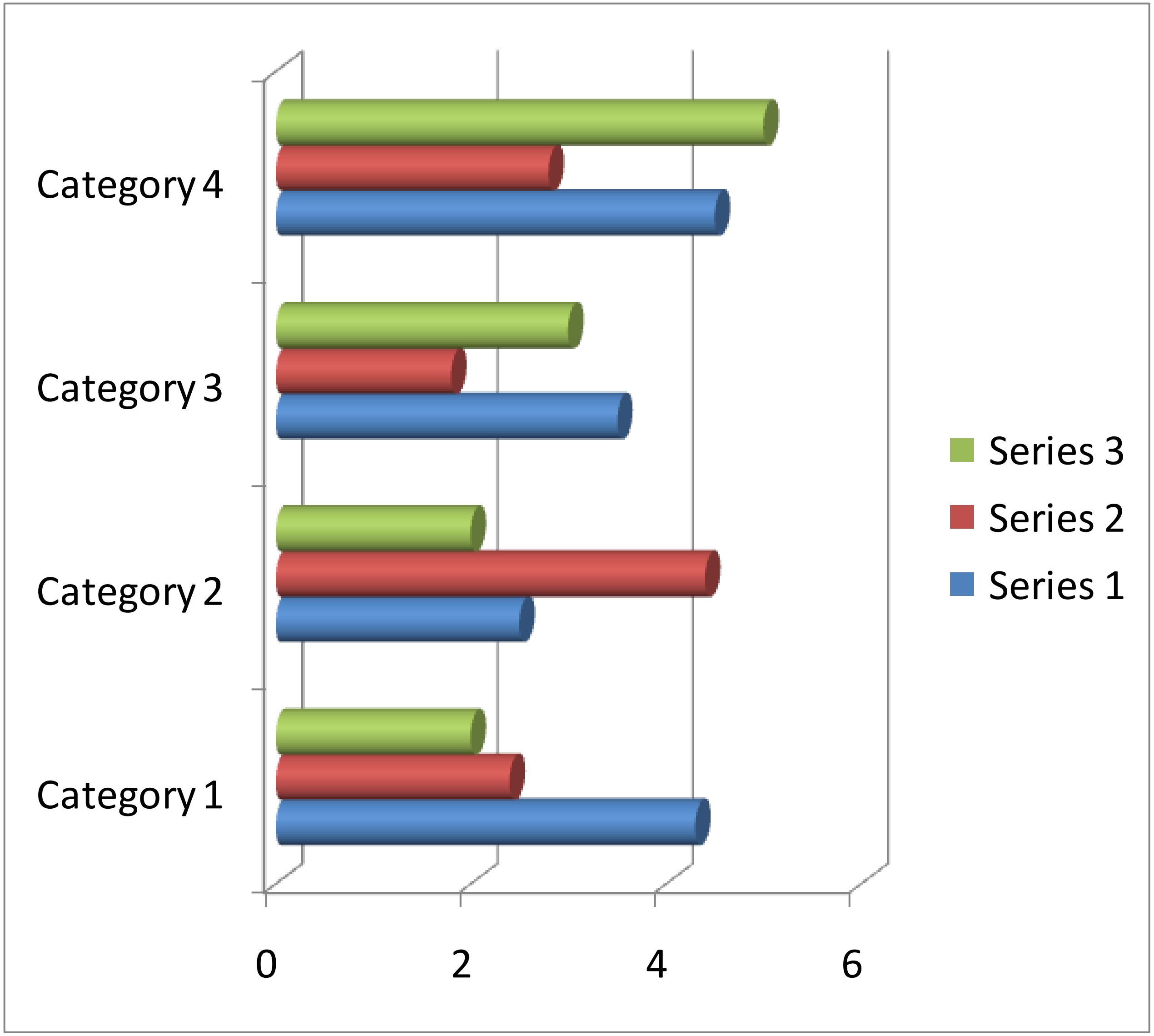
- A high quality graphics display.
- To show moving pictures.
- To produce animations.
- To control the animation.
- Motion dynamics.
- Update dynamics.
- More realistic
- To simulated environment.

Application of Computer Graphics

User Interface

Buttons
Menus
Icons
Scroll Bars

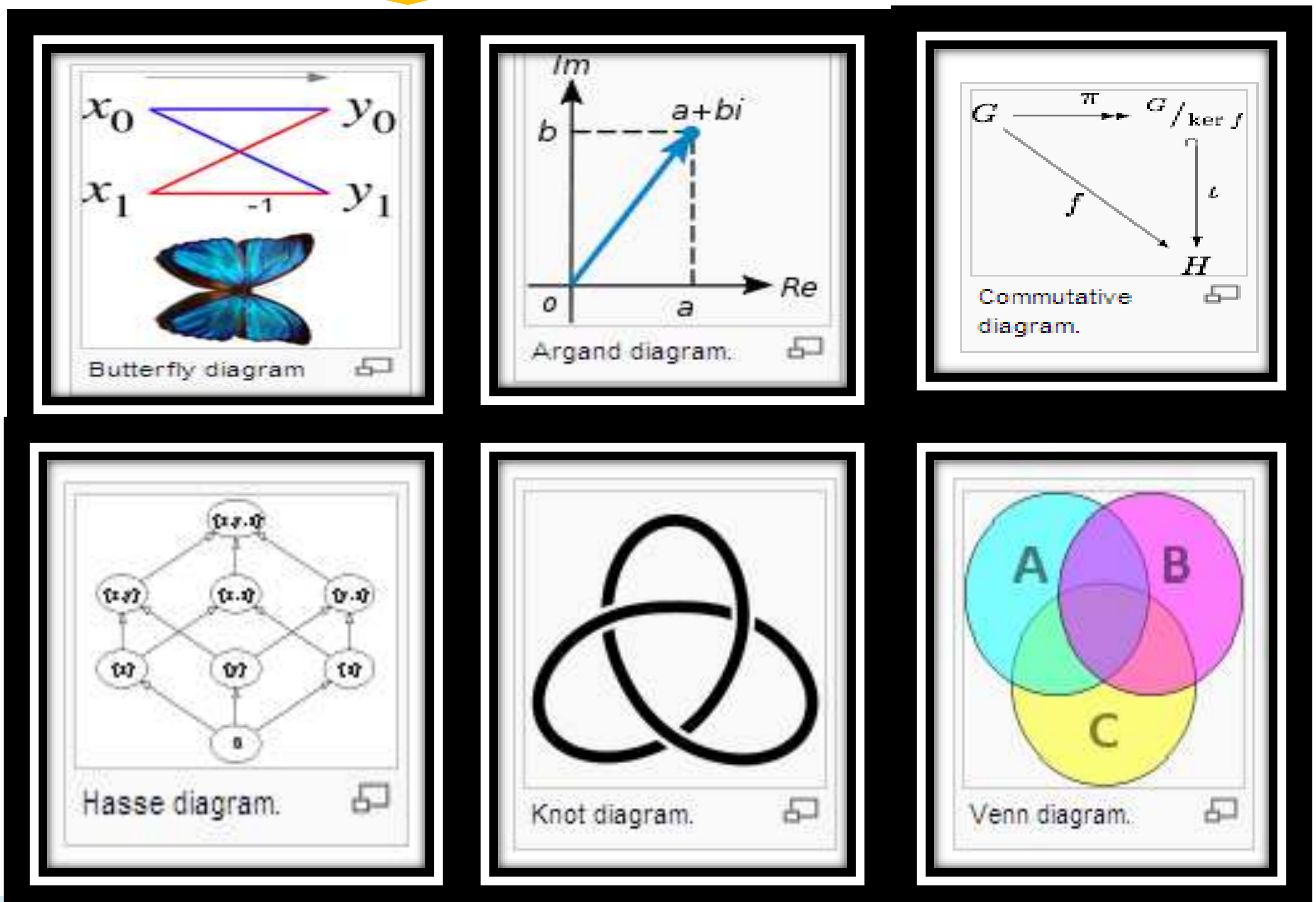



Category 4
Category 3
Category 2
Category 1
0
2
4
6
Series 3
Series 2
Series 1

Office Automation and Desktop Publishing

1. Creation
2. Printing
3. Tables
4. Forms of drawn
5. Scanned Images or Pictures

Simulation and Animation



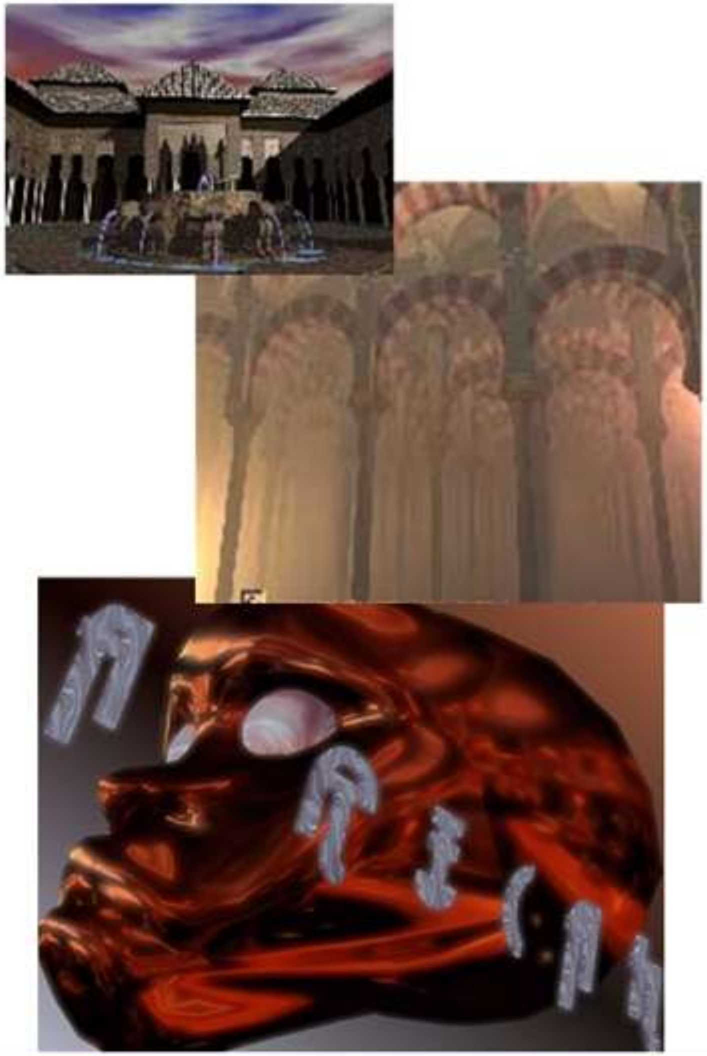
- Artistic field
  - Artistic and commercial objectives
  - *Logo design*
  - *Fine Arts*
- *Animations for advertising*
  - Techniques and software and software
  - *Programs like "PhotoShop",*

    "CorelDraw", "Freehand"

    ... • Animation programs

    - *Image processing techniques*

    •"*rendering*" techniques

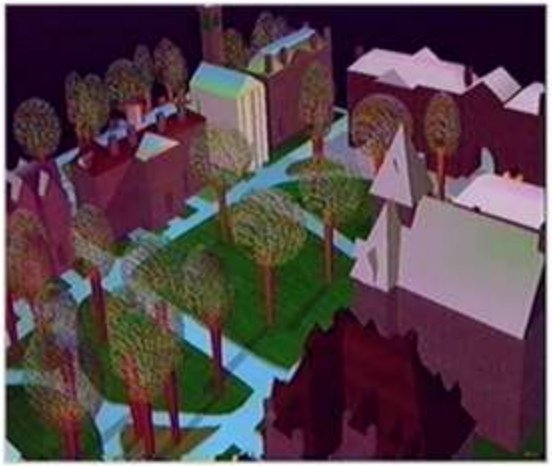


Entertainment

- Areas
  - *Movies: (Tron, Toy Story, etc.)*
  - *Television (transitions, headers, etc.)*
  - *Computer games* Techniques
  - *Animation*

- *Realistic visualization*
  - *Special effects (Ex. morphing)*
  - *Interactivity*

- Scientific and medical visualization
  - Graphics visualization of huge amount of data
  - Areas
    - *Medicine (Ex. resonnance)*
    - *Engineering (Ex. strengths in a mechanism)*
    - *Physics (Ex. Magnetic fields)*
    - *Chemistry (Ex. Molecular interaction)*
    - *Mathematics (Ex. equation solution)*
    - *Topography and oceanography (Ex Terrains and flows)*
  - Techniques
    - *Codification by color*
    - *Level curves*
- *Volume visualization*



Cartography

Geographical Map
Weather Map
Oceanographic Charts
Contour Maps
Population Density Maps

Other Applications

- Photography and printing
- Satellite image processing
- Machine Vision
- Medical image processing
- Face detection, feature detection, face identification
- Microscope image processing
- Car barrier detection

Fields of use

- The Architecture, Engineering, and Construction (AEC) Industry
  - Architecture
  - Architectural engineering
  - Interior Design
  - Interior Architecture
  - Building engineering
  - Civil Engineering and Infrastructure
  - Construction
  - Roads and Highways
  - Railroads and Tunnels
  - Water Supply and Hydraulic Engineering
  - Storm Drain, Wastewater and Sewer systems
  - Mapping and Surveying
  - (Chemical) Plant Design
  - Factory Layout
  - Heating, Ventilation and air-conditioning (HVAC)

contd.....

Fields of use

- Mechanical (MCAD) Engineering   Fully editable digital multi-CAD mockup
  - Automotive - vehicles
  - Aerospace
  - Consumer Goods
  - Machinery
  - Ship Building
  - Bio-mechanical systems
- Electronic design automation (EDA)
  - Electronic and Electrical (ECAD)
  - Digital circuit design
- Electrical Engineering
  - Power Engineering or Power Systems Engineering
  - Power Systems CAD
  - Power analytics
- Manufacturing process planning
- Industrial Design
- Software applications
- Apparel and Textile CAD
  - Fashion Design
- Garden design
- Lighting Design

HISTORY

- Prehistory
  - Whirlwind: Defensive radar system (1951). Computer graphics origin.
  - DAC-1: IBM & General Motors, 3D representation of a car.
- Advances in the 60's
  - Skechpad: Ivan Sutherland, considered as the father of computer graphics. created an interactive drawing program.(1961)
  - SpaceWar: Steve Russell (MIT) designed the first video-game on a DEC PDP-11. (1961)
  - First animation shorts to simulate physical effects (gravity, movement, etc.) (1963)

HISTORY

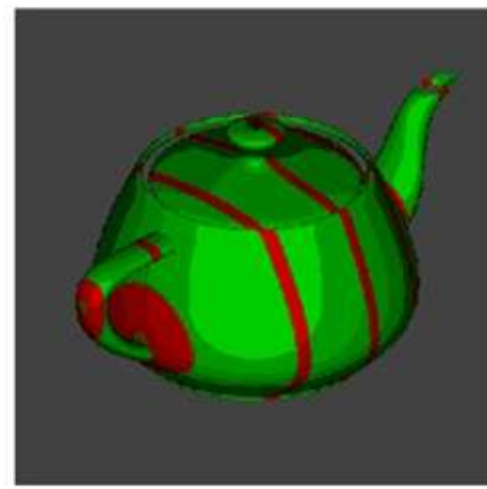
- Sutherland (MIT) made up the first head-mounted display with stereoscopy vision (1966)
- First algorithm of hidden surfaces. by Catmull et al. at the Utah University. At the end of 60’s.
- The same team began to have interest in surface shading using color.

- Advances in the 70’s
  - Introduction of computer graphics in television.
  - Gouraud presented his famous polygonal surface smoothing method.(1971)
  - Microprocessor on the market (1971)
  - Atari was born in 1972. It is the computer game pioneer.

HISTORY

- First uses of CG (Computer Graphics) in **movies**.
- Newell at the University of Utah create the famous teapot, a classical benchmark for visualization algorithms.

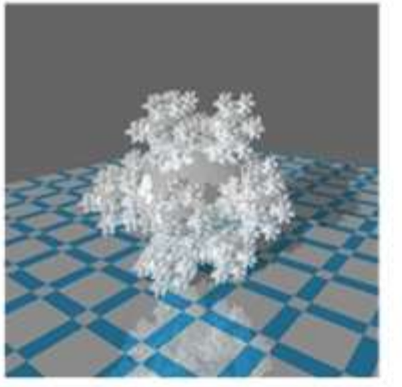
- Texturing and Z-Buffer: Catmull's thesis in 1974.
- Phong developed his polygonal surface smoothing method (1974).
- 1975 Baum and Wozniak founded Apple in a garage.
- Gates founded Microsoft (1975).
- Lucasfilm created the computer graphics division with the best gurus of the moment (1979).



HISTORY

Advances in the 80's

- SIGGRAPH is the most important event in this field.
- Whitted published an article about **ray tracing technique** (1980)
- Carpenter, at Lucasfilm, developed the first rendering engine: REYES, the Renderman precursor.(1981)
- TRON film by Lisberger and Kushner at Disney (beginning of the 80's)
- Massive sales of graphics terminals: IBM, Tektronix.
- The first ISO and ANSI standard for graphics libraries: GKS.
- IBM created the Personal Computer PC.



HISTORY

•Advances in the 90's and nowadays:

- Operative system based on windows for PC (Windows 3.0 at 1990).

-3D-Studio from Autodesk (1990).

- Massive use of computers to produce special effects: Terminator 2 (1991), Disney-Pixar (Toy Story, Bugs, Monsters, inc.), Forrest Gump, Jurassic Park, Lord of the Rings, Starwars episodes I, II and III etc.

- Internet success and 2D and 3D applications for the web.

  3D graphics cards for PC (Voodoo, Nvidia Gforce etc.).

  Unstoppable 3D games evolution.
- Virtual Reality. A reality.
- Nowadays: a must for any application.

Classification of Computer Graphics

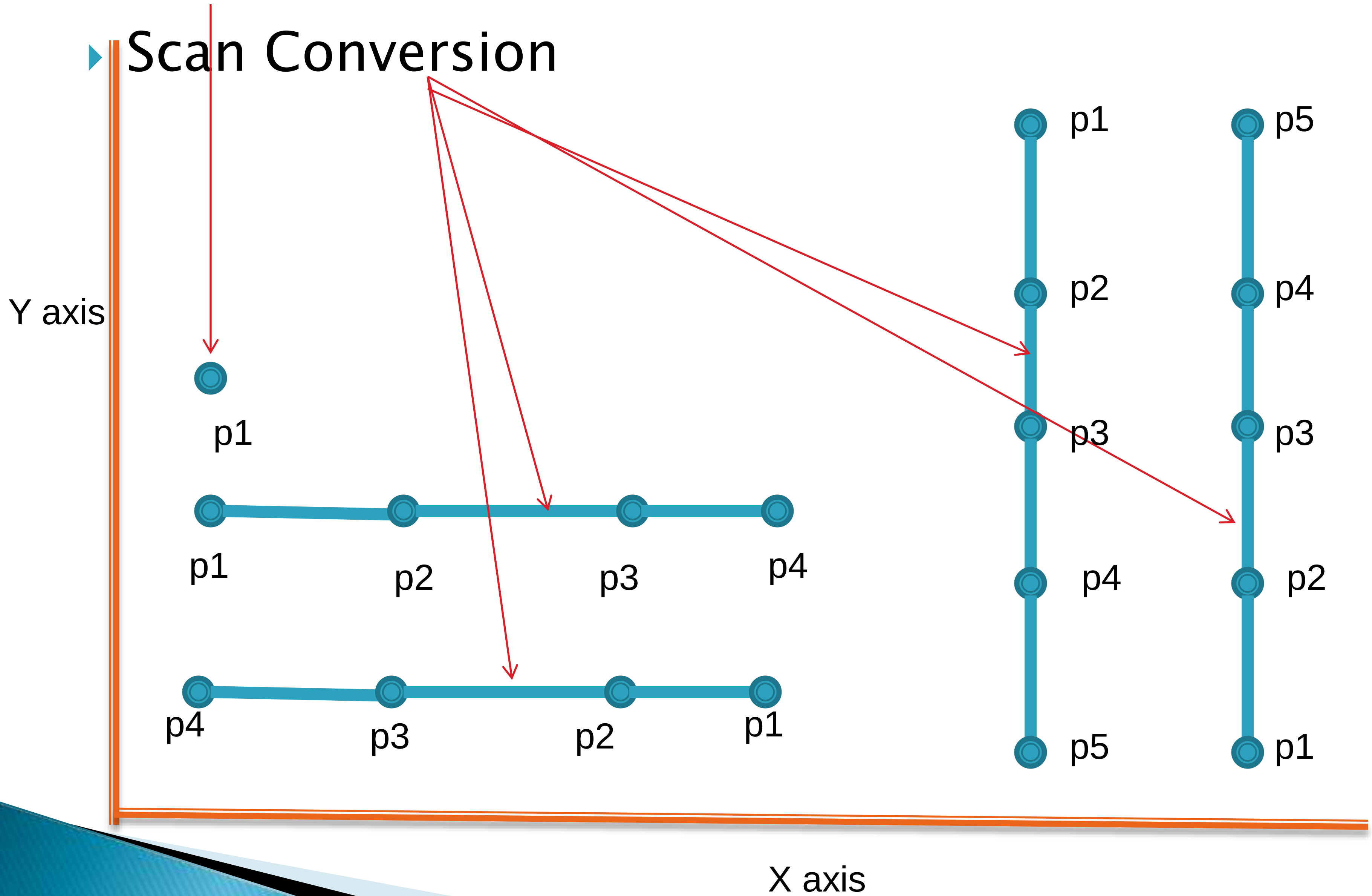
Types of Graphics Devices

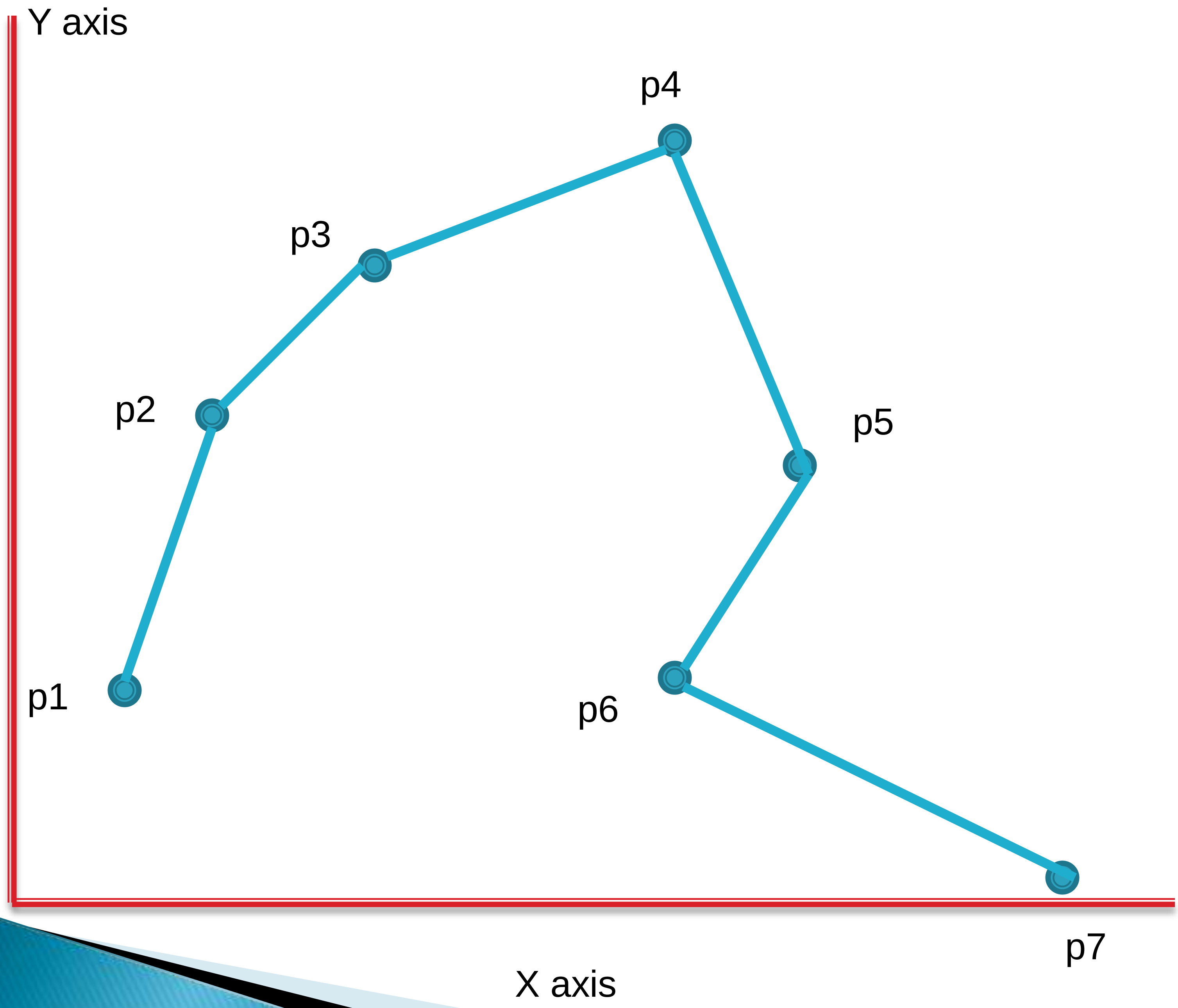
- Video Display Devices
  - Cathode Ray Tube
  - Vector Scan/Random Scan Display
  - Raster Scan Display
  - Colour CRT Monitors
  - Direct-View Storage Tubes
  - Flat Panel Display
  - Plasma Panel Display
  - Liquid Crystal Monitors

Types of Graphics Devices

- Input Devices
  - Keyboard
  - Mouse
  - Trackball and Space ball
  - Joysticks
  - Data Glove
  - Digitizer/Graphical Tablet
  - Image scanners
  - Touch Panels
  - Light Pan
  - Voice Systems

- Rasterization
- Scan Conversion




Y axis
p4
p3
p2
p5
p1
p6
p7
X axis

- The process of determine the appropriate pixels for representing picture or graphics object is known as rasterization.

- The process of representing continuous picture or graphics object as a collection of discrete pixels is called scan conversion.

Technologies for Generating Image

- Random Scan Display (Vector Scan Display)
- Raster Scan Display (Refreshing Scan Display)

Random Display

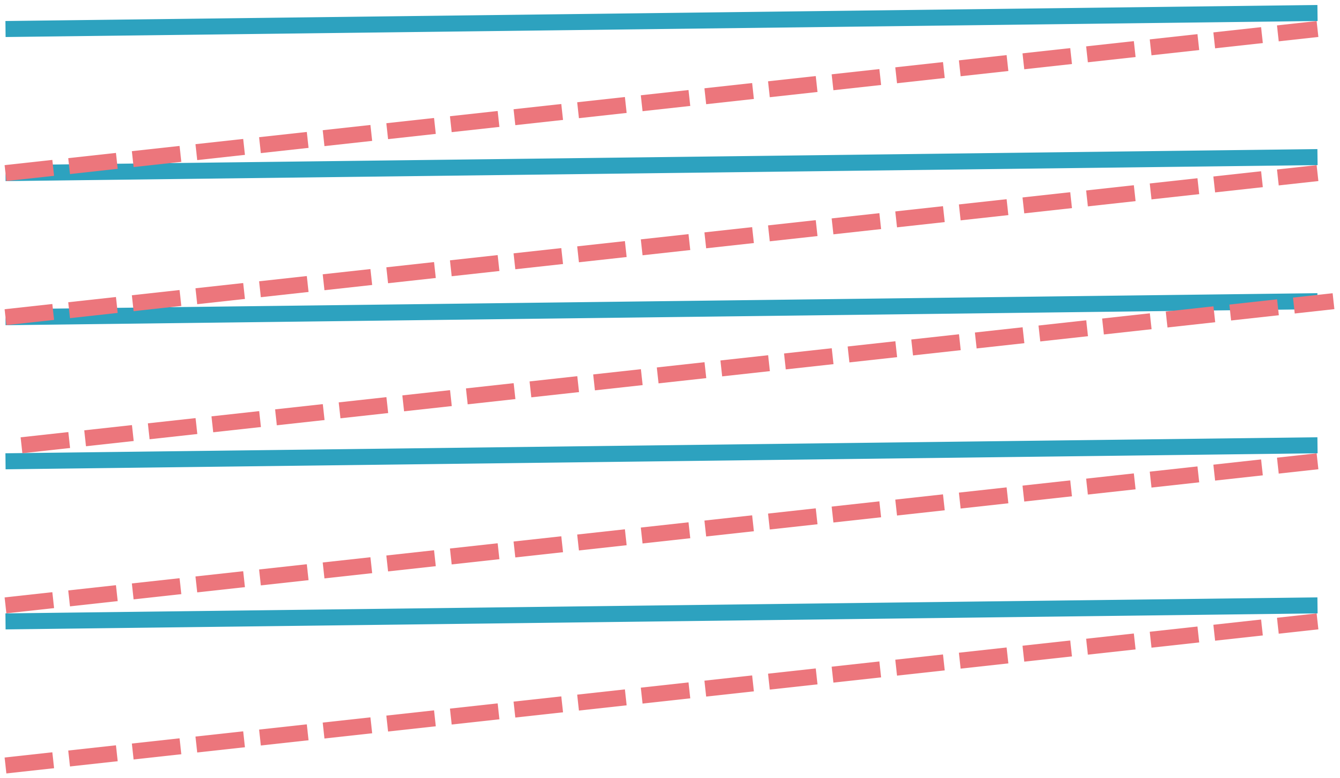
Advantages

- Basically used for line drawing command, produce smooth line drawing.
- Resolution of random display system is higher.
- Electron beam falls only those parts of the screen where a picture is to be drawn.

Disadvantages

- Main disadvantage of random display system is that they do not produce real and shadow images.
- Different colors are not possible with this approach.

Raster Display



Advantages

- You can also create shadow scenes.
- Millions of different colors can be displayed with this approach.
- Picture quality is good.
- It is popular in use because they generate realistic pictures.

Disadvantages

- It is expensive than random display.
- Low resolution.
- To draw the picture electron beam sweep across whole the screen.

Random Scan Display	Raster Scan Display
High Resolutions	Less Resolutions
The smooth lines are produced as the electron beam directly follows the line path.	The lines produced are ziz-zag as the plotted values are discrete.
realism is difficult to achieve.	high degree realism is achieved in picture with the aid of advanced shading and hidden surface technique.
random-scan system's are generally costlier.	decreasing memory costs.
Here CRT has the electron beam directly only to the parts of the screen where a picture is to be drawn.	In this case, the electron beam is swept across the screen, one row at a time from top to bottom.
Picture definition is stored as a set of line drawing commands in an area of memory referred to as refresh display file.	Picture definition is stored in a memory area called the refresh buffer/frame buffer.
Random scan systems are designed to draw all the component lines of a picture 30 to 60 times each second.	Refreshing on raster scan displays is carried out at the rate of 60 to 80 frames/second.

Resolution

- The maximum number of pixels that can be displayed per unit length in vertical as well as horizontal direction of the screen is known as resolution of the screen.

- Distance from one pixel to the next pixel.

- The total number of pixels along the entire height and width of the image.

Example

- Full screen image with resolution 800x600 means that there are 800 columns of pixels , each column comprising 600 pixels,
  i.e. a total of 800 X 600 = 480000 pixel in the image area.

ASPECT RATIO

- Aspect ratios gives the ratio of vertical points to horizontal point which produce equal to length line in both direction of screen .
- Aspect ratio can be measured in unit length of number of pixels.
- Standard PC have a display are with aspect ratio 4/3 where vertical line plotted with 4 pixels and horizontal line plotted with 3 pixel with same length.

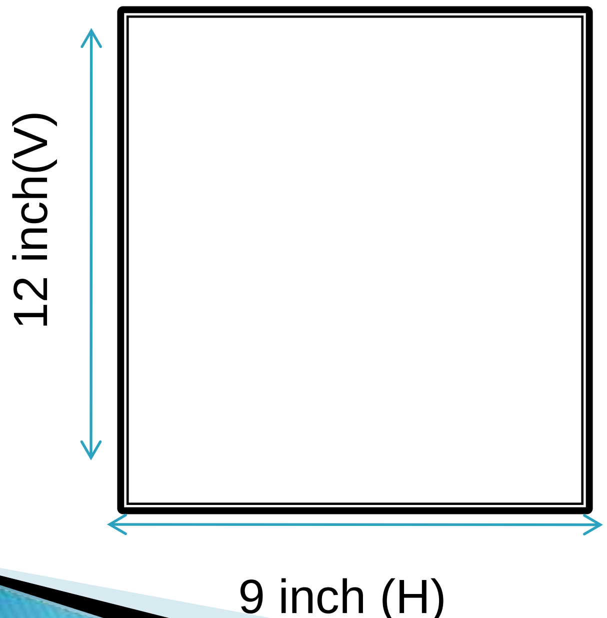


figure : Aspect ratio 4:3

figure : Aspect ratio 3:4

Resolution	Number of Pixels	Aspect Ratio
320x200	64,000	8:5
640x480	307,200	4:3
800x600	480,000	4:3
1024x768	786,432	4:3
1280x1024	1,310,720	5:4
1600x1200	1,920,000	4:3

Output Primitives

- POINT
- PIXELS
- PLANES
- VECTOR
- CHARACTER GENERATION
- FRAME BUFFER
- POINT PLOTTING TECHNIQUES(PPT)

POINT

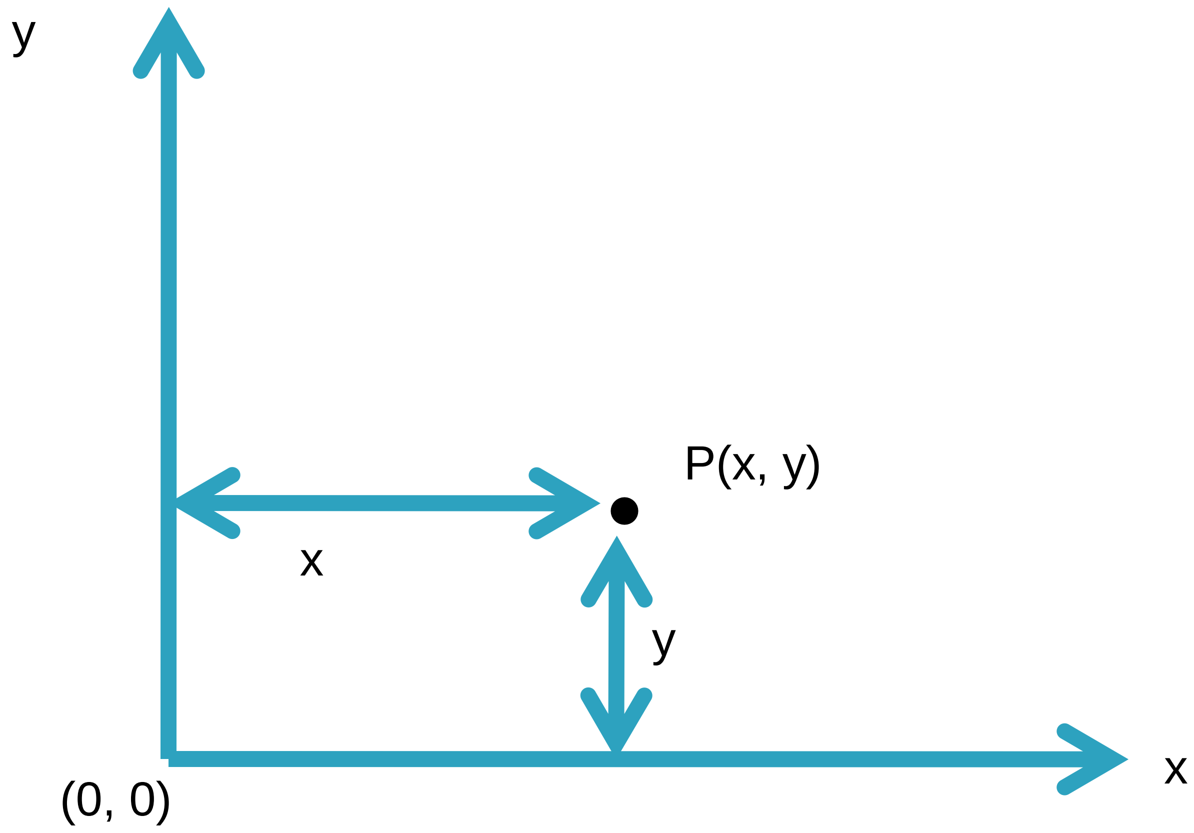


Fig: Position of a Point on Plane

PIXELS OR PEL

A pixel may be defined as the smallest size object or color spot that can be displayed and addressed on a monitor. Any image that is displayed on the monitor is made up of thousands of such small pixels (Picture Elements).

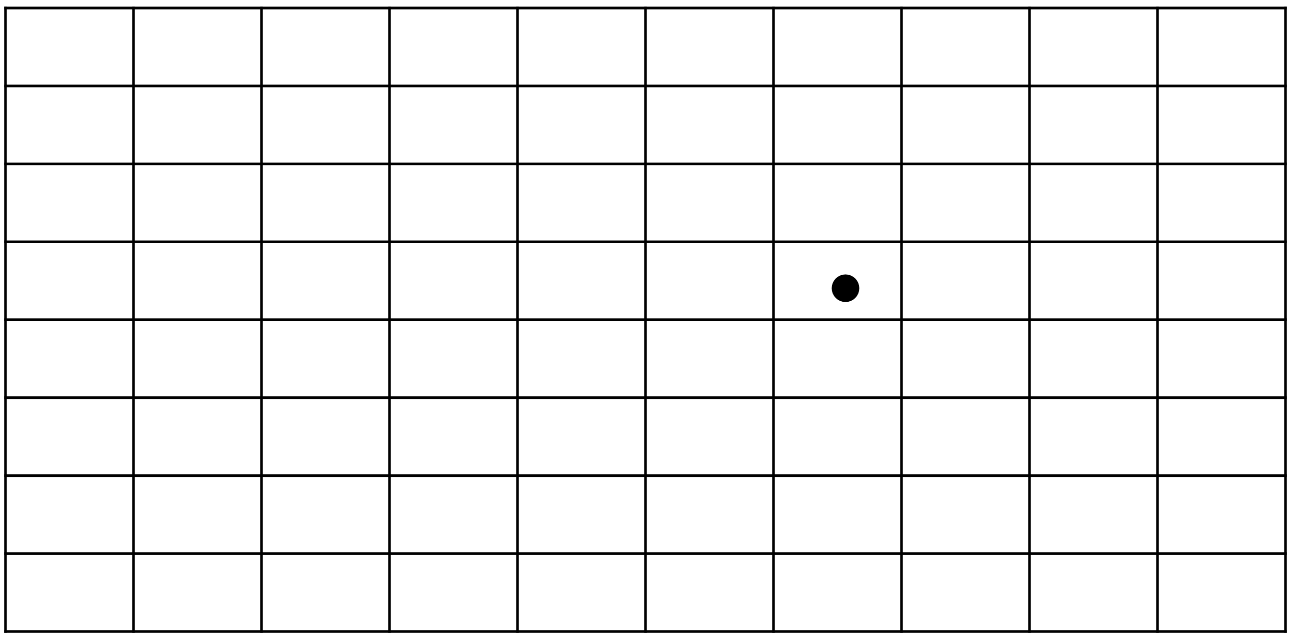
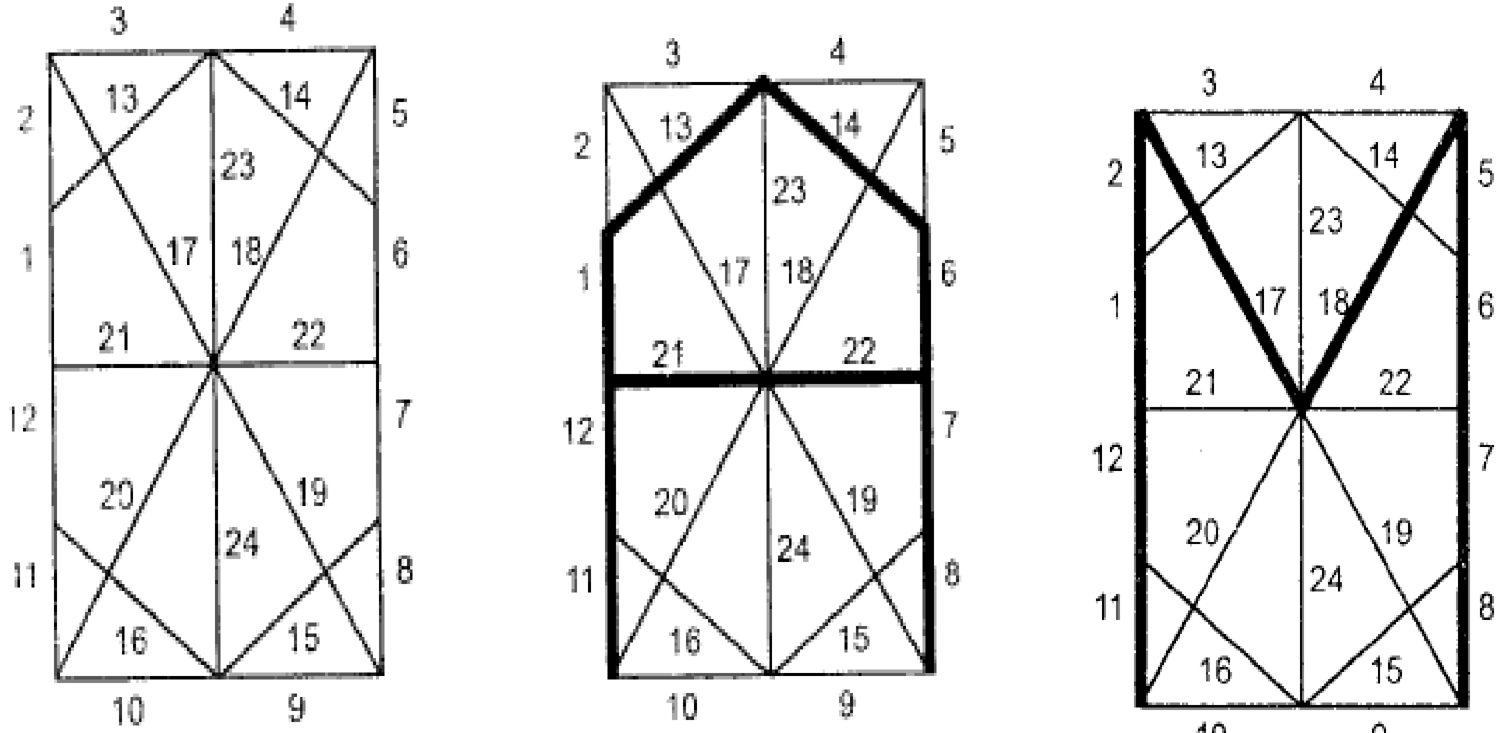


Fig: Pixel

Character Generation

1. Stroke Method
2. Starbust Method
3. Bitmap Method

Starbust Method



a) Star bust pattern of 24 line segments

b) Star bust pattern for character A

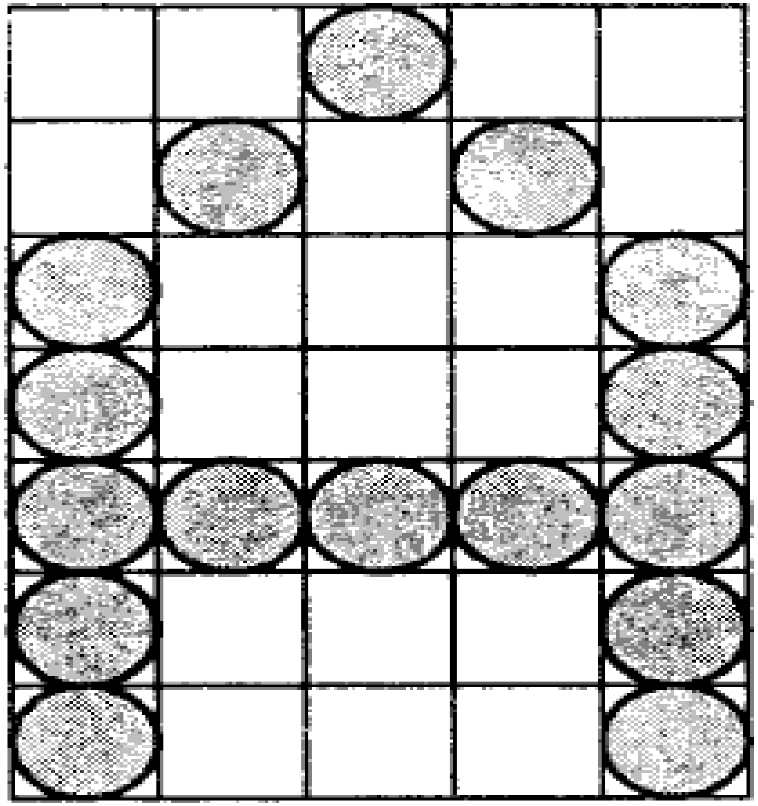
c) Star bust pattern for character M

Example : 24 bit code for						
Character A is	0011	0000	0011	1100	1110	0001
Character M is	0000	0011	0000	1100	1111	0011

Drawback

- The 24-bits are required to represent a character. Hence more memory is required.
- Requires code conversion s/w to display character from its 24-bits code.
- Character quality poor. It is worst for curve shaped character.

Bitmap Method



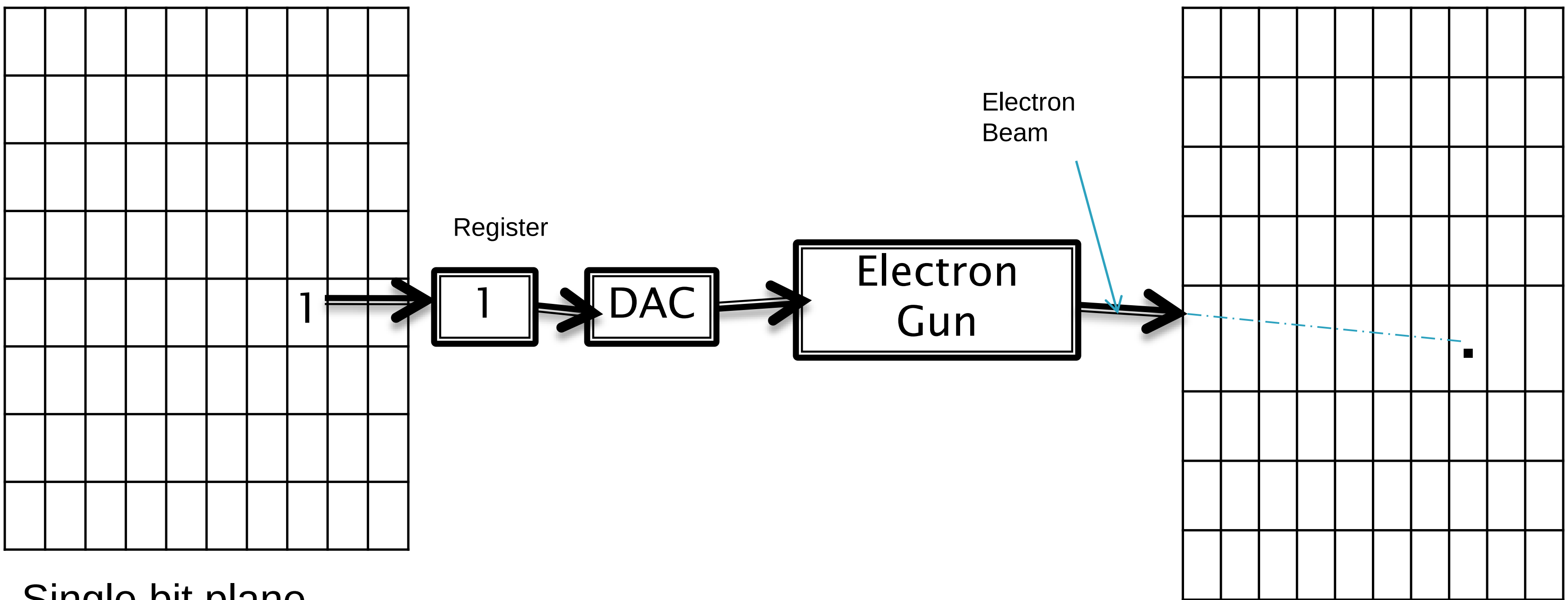
Note : Hardware device : Character Generation Chip

Frame Buffer

0	1	1	0	0	1	1	0	0	1
0	1	0	1	1	1	0	1	1	1
0	0	0	1	0	0	1	0	0	1
1	0	1	0	1	0	1	1	1	0
1	1	1	0	1	0	0	0	0	0
1	1	0	1	0	1	0	1	1	0
0	0	0	1	1	1	1	1	0	1
0	1	1	0	0	1	1	0	0	1

0 Represent – Black
1 Represent - White

One bit per pixel – Bitmap /Bit planes



Single bit plane

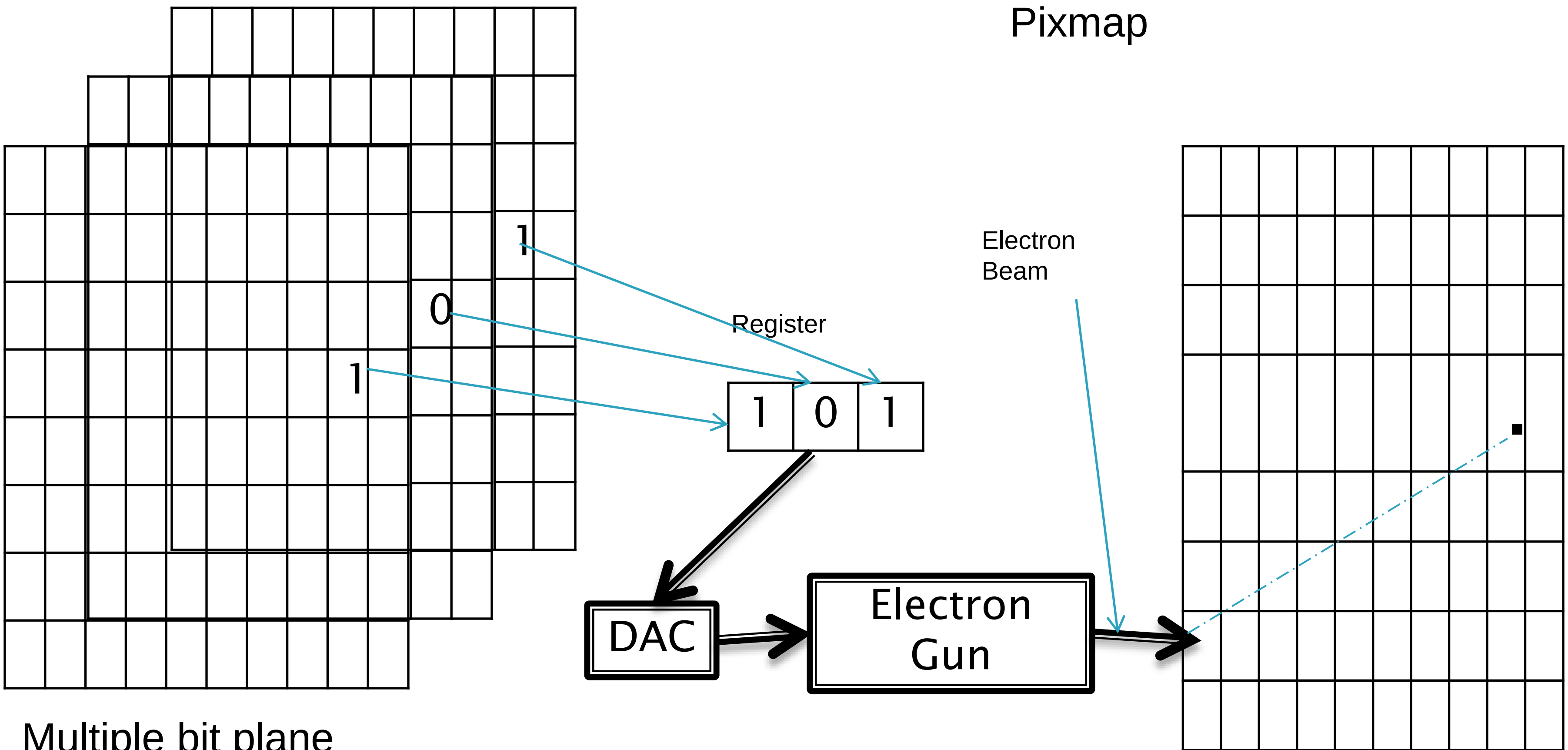
CRT Raster

Fig: For bit depth =1,

For 3 bit per pixel frame buffer

Color Code	Stored Color Values in Frame Buffer			Displayed Color
	Red	Green	Blue	
0	0	0	0	Black
1	0	0	1	Blue
2	0	1	0	Green
3	0	1	1	Cyan
4	1	0	0	Red
5	1	0	1	Magenta
6	1	1	0	Yellow
7	1	1	1	White

Multiple bit per pixel – Pixmap



Multiple bit plane

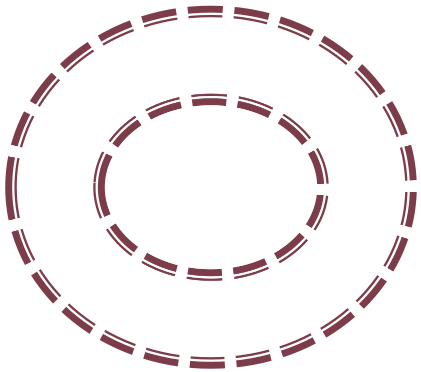
CRT Raster

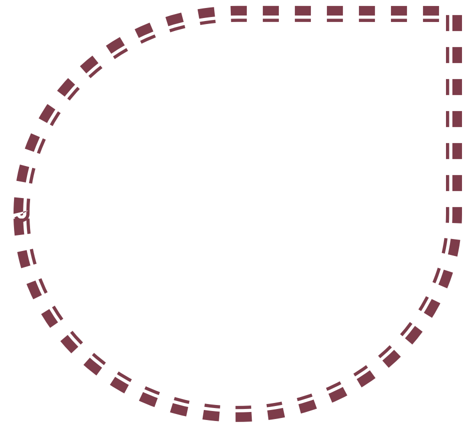
Fig: For bit depth =n,

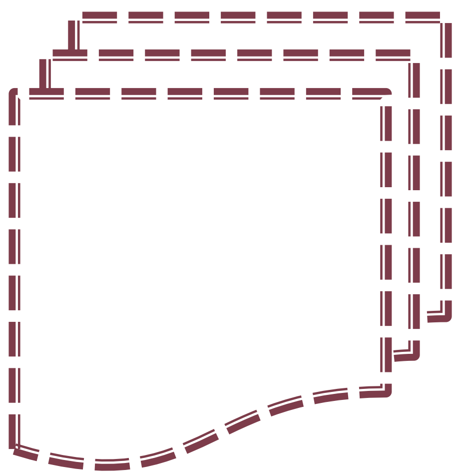
Common Color Depths Used in PCs

Color Depth	No.of Displayed Colors	Bits of storage Per Pixel	Displayed Color
4-Bit	16	4	Standard VGA
8-Bit	256	8	256-Color Mode
16-Bit	65,536	16	High Color
24-Bit	16,777,216	24	True Color

PPT : Point Plotting Techniques

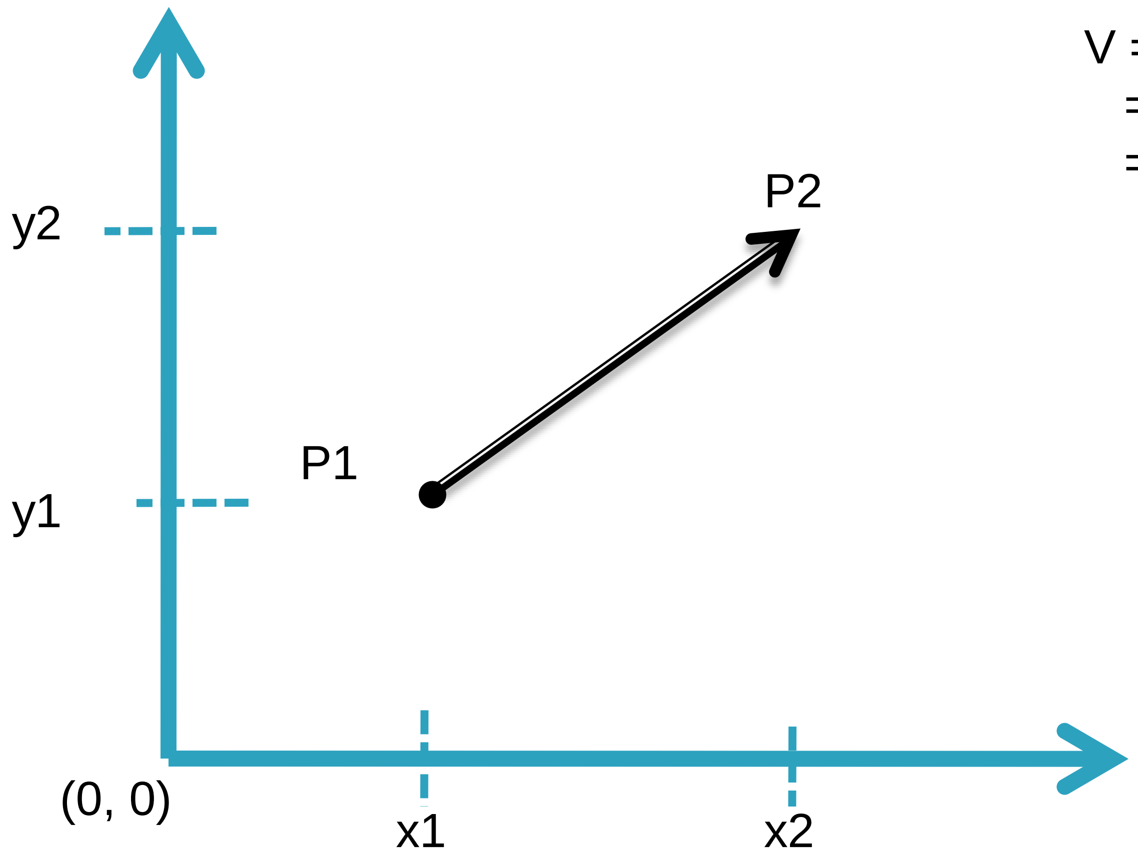






BOY

VECTOR



V = P2 - P1
= (x2 - x1,y2 - y1)
= (Vx,Vy)

Fig: Vector V in xy plane

It has two basic properties : Direction and Magnitude

Magnitude of any vector can be determine by using Pythagorean Theorm as

$|V| = \sqrt{V_x^2 + V_y^2}$

The Direction of any vector is given by

$\alpha = \tan^{-1}\left(\frac{V_y}{V_x}\right)$

3d Cartesian the magnitude of a vector

$$|V| = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

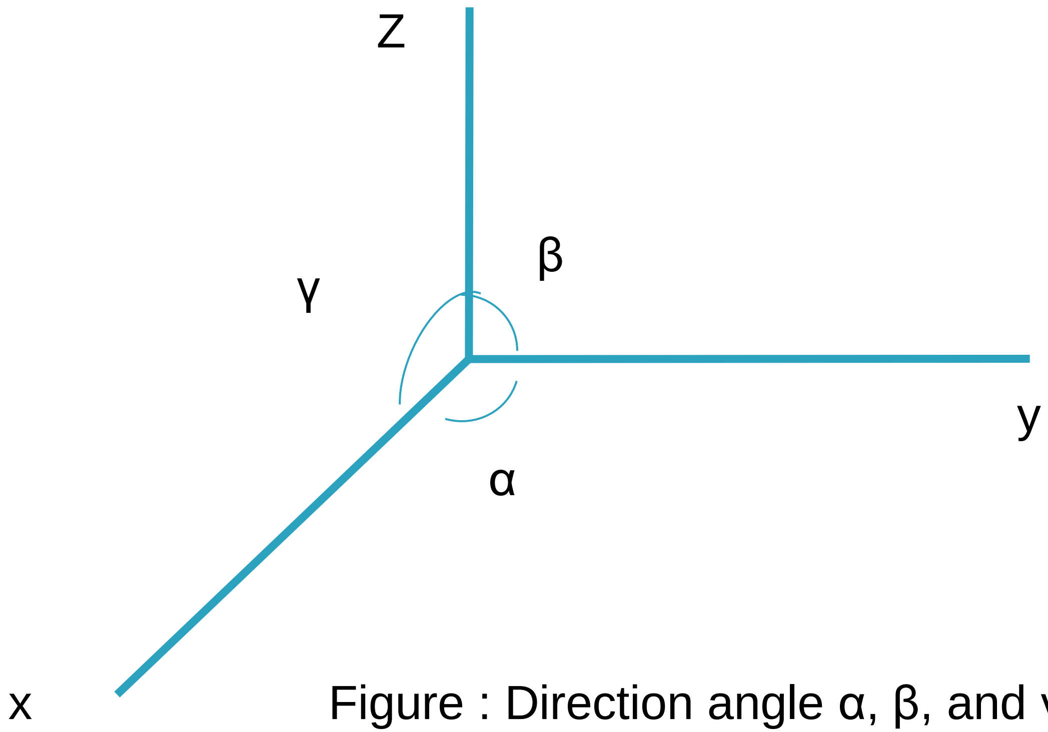
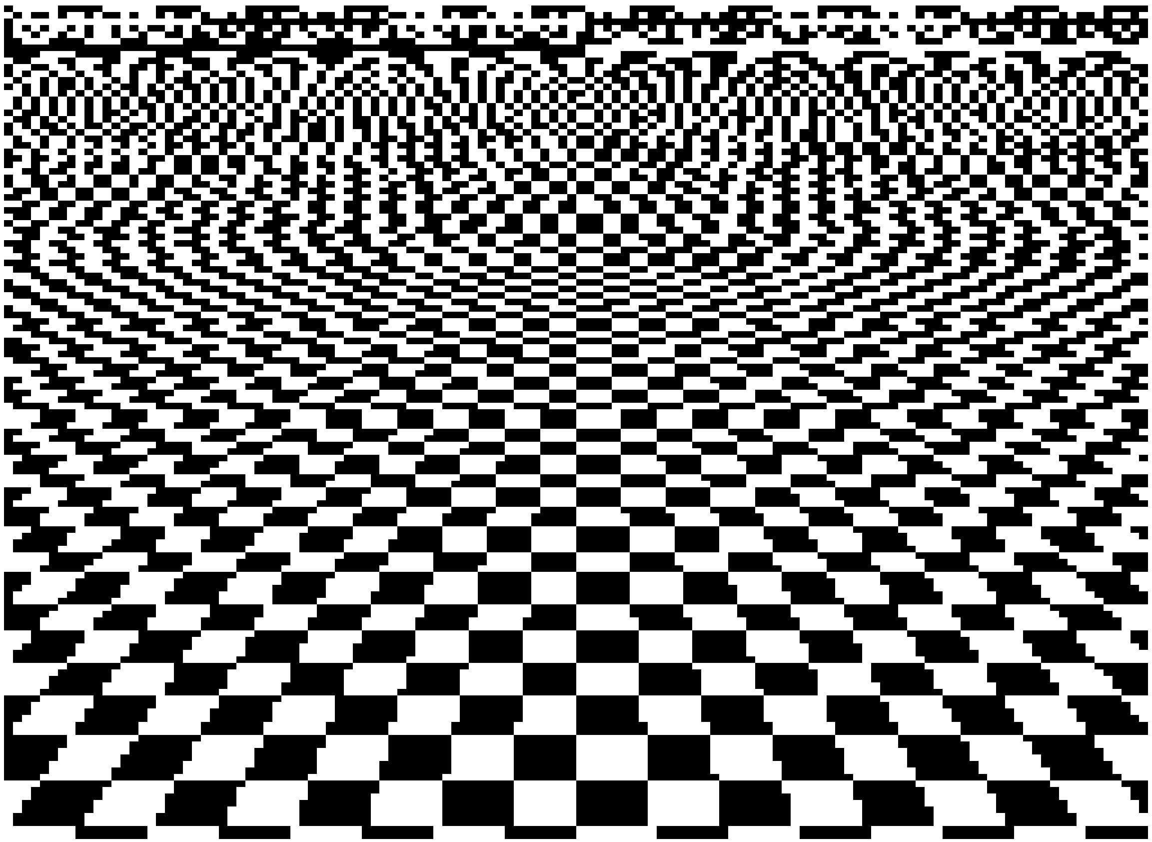


Figure : Direction angle α, β, and γ.

Different forms of Vector

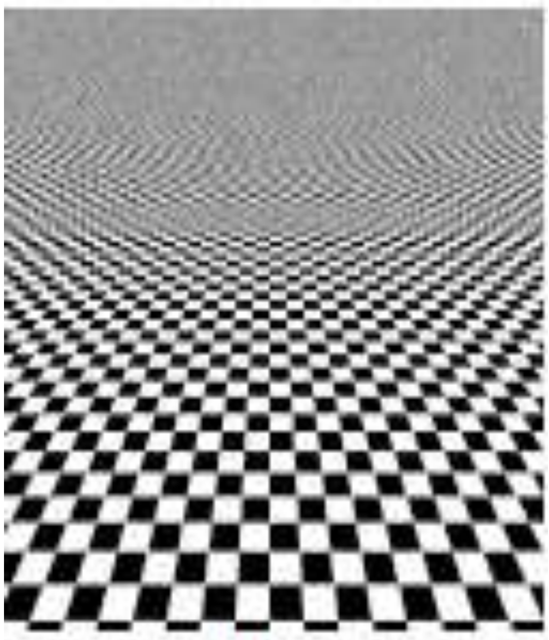
- Null Vector
- Unit Vector
- Vector Addition
- Vector Subtraction
- Scalar Product/Dot Product/Inner Product of Two Vector
- Vector product of Two Vectors
- Space Co-ordinate
- Resolution of Vectors

Aliasing

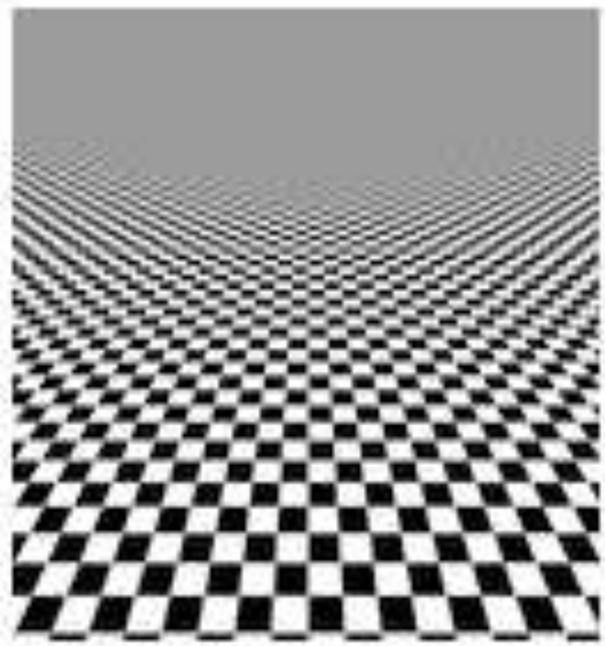


(a)

Antialiasing

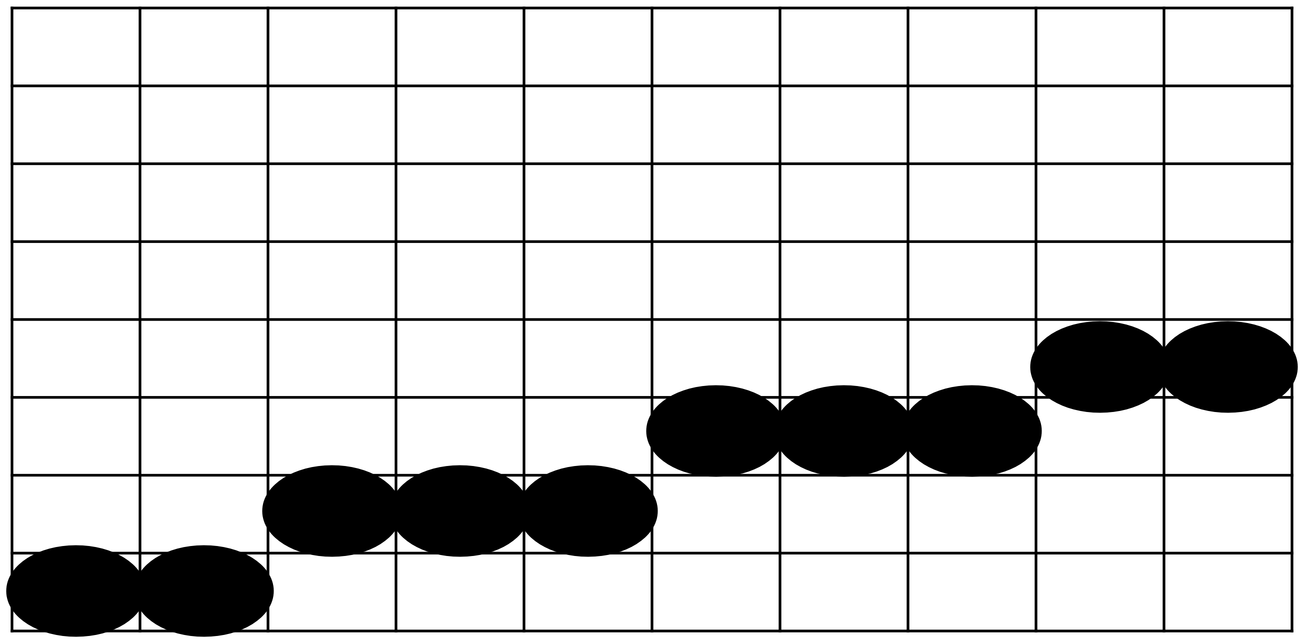


(b)



(c)

Antialiasing of lines



LINE DRAWING ALGORITHM

Criteria for good line drawing

- Line should be drawn rapidly.

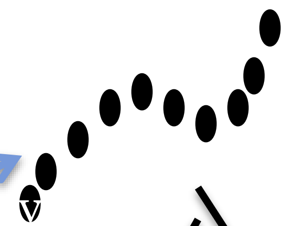


Fig: O/P from a poor line generating algorithm

- Line should be appearing straight.
- Line should terminate accurately.
- Line should have constant density.

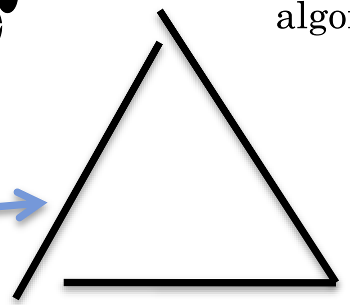


Fig: Line not terminate accurately

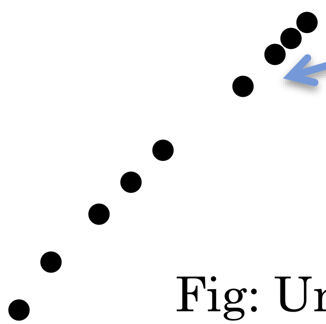


Fig: Uneven Line density

- DDA Algorithm
- Bresenham's Algorithm

DDA ALGORITHM

1. Calculate the horizontal difference between the two end points.

   dx = abs(x2-x1)

2. Calculate the Vertical difference between the two end points.

   dy = abs(y2-y1)

3. If dx>dy then the value of increment step is

   Step =dx

   Else

   Step =dy

4. Xinc = (X2-X1)/step and Yinc = (Y2-Y1)/step

5. X = X1 + 0.5

   Y = Y1 + 0.5

6. Set k=0

7. Plot(Round(X),Round(Y))

8. X=X+Xinc

   Y=Y+Yinc

9. k=k+1

10. If K<Step the goto step-7

DDA ALGORITHM

- Read the line end points (x1,y1) and (x2,y2) such that they are not equal.
- Δx = |x2 - x1|

  Δy = |y2 - y1|
- If (Δx >= Δy) then

  length = Δx

  else

  length = Δy
- Δx = (x2 - x1)/length //For Increment of X & Y)

  Δy = (y2 - y1)/length
- x = x1+ 0.5 * sign(Δx)

  y = y1+0.5 * sign(Δy)
- i=1

  while (i<=lengh)

  {

  plot(Integer(x),Integer(y))

  x= x+ Δx

  y=y+ Δy

  i=i+1;
- stop

Example 1 : Consider the line from (0,0) to (4,6). Use the simple DDA algorithm to rasterizing this line.

Solution : Evaluating steps 1 to 5 the DDA Algorithm we have

$x1 = 0$ $y1 = 0$

$x2 = 4$ $y2 = 6$

:: Length = y2 -y1 = 6 - 0 = 6

$\Delta x = (x2 - x1)/Length = (4-0)/6 = 4/6$ and

$\Delta y = (y2 - y1)/Length = (6 - 0)/6 = 1$

Initial Value for $x = 0+0.5*sign(4/6)=0.5$

$y =0+0.5*sign(1) = 0.5$

Tabulating the results of the each iteration in step 6 we get:

i	Plot	x	y
		0.5	0.5
1	(0,0)		
		1.167	1.5
2	(1,1)		
		1.833	2.5
3	(1,2)		
		2.5	3.5
4	(2,3)		
		3.167	4.5
5	(3,4)		
		3.833	5.5
6	(3,5)		
		4.5	6.5

Example2:Consider the line from (0,0)to(6,6) Use the simple DDA algorithm to rasterizing this line.

Example 3 : Consider the line from (10,15) to (20,21).Use the simple DDA algorithm to rasterizing this line.

Example 4: Scan Convert a straight line whose end points are (5,10) and (15,35) using DDA Algorithm.

Example 5: Scan Convert a straight line whose end points are (-1,- 2) and (+4,+8) using DDA Algorithm.

Advantages of DDA algorithm

- It is the simplest algorithm that it does not require special skills for implementation.
- It is a faster method for calculating pixel positions than the direct use of equation y=mx+b.

DISADVANTAGES OF DDA ALGORITHM

- Floating point arithmetic in DDA algorithm is still time consuming.
- The algorithm is orientation (Direction) dependent. Hence end point accuracy point is poor.

Bresenham's line algorithm

- It uses only integer addition and subtraction and multiplication by 2, and we know that computer can perform the operation of integer addition and subtraction very rapidly.
- The computer is also time-efficient when performing integer multiplication by powers of 2. Therefore, it is an efficient method for scan converting straight lines.
- The basic principle of Bresenham's line algorithm is to select the optimum raster locations to represent a straight line.
- To accomplish this the algorithm always increments either x or y one unit depending on the slope of line.
- The increment in the other variable is determined by examining the distance between the actual line location and the nearest pixel. This distance is called **decision variable** or the **error**.

BRESENHAM'S LINE DRAWING ALGORITHM

1. Input the two line end-points, storing the left end-point in (x_0, y_0)
2. Plot the point (x_0, y_0)
3. Calculate the constants Δx, Δy, $2\Delta y$, and $(2\Delta y - 2\Delta x)$ and get the first value for the decision parameter as:

$$p_0 = 2\Delta y - \Delta x$$

4. At each x_k along the line, starting at $k = 0$, perform the following test. If $p_k < 0$, the next point to plot is (x_k+1, y_k) and:

$$p_{k+1} = p_k + 2\Delta y$$

The Bresenham Line Algorithm (cont…)

Otherwise, the next point to plot is (x_k+1, y_k+1) and:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

or

$$p_{k+1} = p_k - 2\Delta x$$

5. Repeat step 4, (Δx) times.

- Read the line end point (x1,y1) and (x2,y2) such that they are not equal.
- $\Delta x = |x2 - x1|$ and $\Delta y = |y2 - y1|$
- Initialize starting point

  x=x1

  y=y1 and then Plot the first pixel.
- p = 2 * Δy -Δx (Initialize value of decision variable or error to compensate for nonzero intercepts).
- i=1
- while(p>=0)

```
{
        y=y+1
        p=p -2* Δx
}
        x=x+1
        p=p+2* Δy
```

- Plot(x,y)
- i=i+1
- if(i<= Δx) then go to step 6
- Stop

Example 1 : Consider the line from (5,5) to (13,9). Use the Bresenham's Algorithm to rasterizing the line.

Solution : Evaluating steps 1 through 4 in the Bresenham's Algorithm we have,

$\Delta x = |13 - 5| = 8$

$\Delta y = |9 - 5| = 4$

x= 5 and y = 5

$p = 2 * \Delta y - \Delta x = 2 * 4 - 8$

$= 0$

Tabulating the result of each iteration in the step 5 through 10.

i	Plot	x	y	p
	(5,5)	5	5	0
1	(6,6)	6	6	-8
2	(7,6)	7	6	0
3	(8,7)	8	7	-8
4	(9,7)	9	7	0
5	(10,8)	10	8	-8
6	(11,8)	11	8	0
7	(12,9)	12	9	-8
8	(13,9)	13	9	0

Basic concept of circle drawing

- Polynomial Method
- Trigonometric Method

A Simple Circle Drawing Algorithm

- The equation for a circle is:

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

- where r is the radius of the circle
- So, we can write a simple circle drawing

algorithm by solving the equation for y at unit x intervals using:

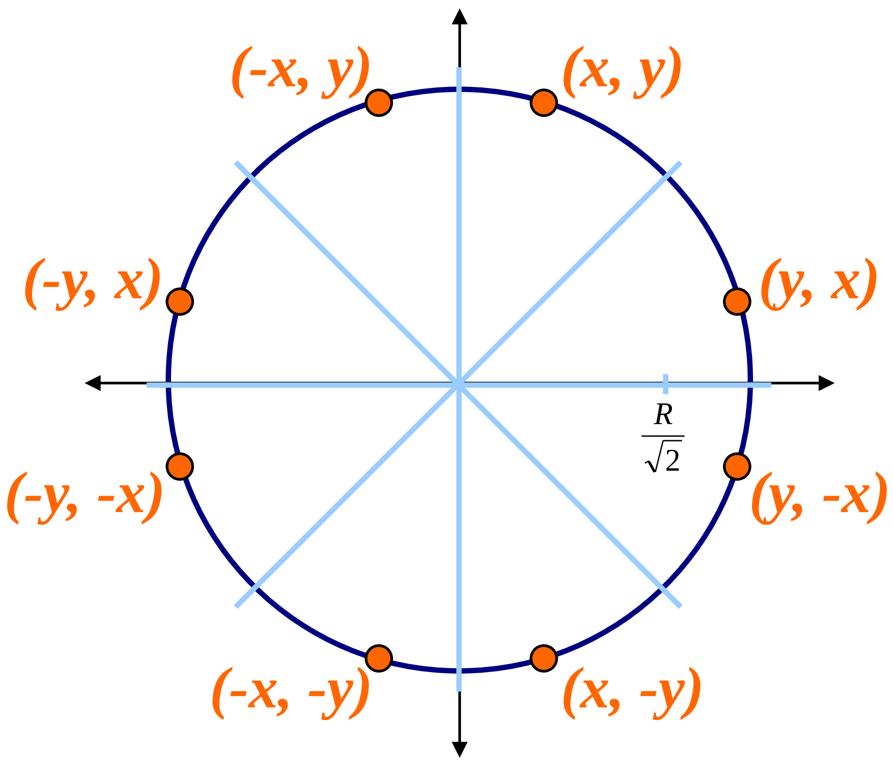
$$y = \pm\sqrt{r^2 - x^2}$$

A Simple Circle Drawing Algorithm (cont…)

- 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

EIGHT-WAY SYMMETRY

- 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*



BRENSENHAM'S CIRCLE ALGORITHM

- It is based on the following function for testing the spatial relationship between an arbitrary ponit (x,y) and a circle of radius r centred at the the origin :

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

- 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 point between the candidate pixels we can make our decision

BRENSENHAM'S CIRCLE ALGORITHM

1. 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:

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

2. Calculate the initial value of the decision parameter as:

$$d_1 = 3 - 2r$$

3. If T is the chosen pixel (Meaning that d_i<0) then y_i+1=y_i and so

$$d_{i+1} = d_i + 4x_i + 6$$

Circle Algorithm (cont…)

4. On the other hand, if S is the chosen pixel(di<0) then $y_i+1 = y_i-1$ and so

$$d_{i+1} = d_i + 4(x_i - y_i) + 10$$

5. Repeat steps 3 to 5 until (x<y)

- The D provides a relative measurement of distance from the center of a pixel to true line. Since D(T) will always be positive (T is outside the true circle) and D(S) will always be negative (S in inside the true circle). A decision variable d_i may be defined as follows :

$$d_i = D(T) + D(S)$$

As the coordinates of T = (Xi +1 ,Yi)

As the coordinates of S = (Xi+1 ,Yi-1)

$D(T) = (x_i+1)^2 + y^2_i - r^2$

$D(S) = (x_i+1)^2 + (y_i-1)^2 - r^2$

$d_i = D(T) + D(S)$

$\mathbf{d_i = 2(x_i + 1)^2 + y_i^2 + (y_i - 1) - 2r^2}$

When $d_i <= 0$, we have $|D(T) < D(S)|$ and pixel T is chosen.

When $d_i >= 0$, we have $|D(T) > D(S)|$ and pixel S is chosen.

For next step , decision variable d_{i+1}

$\mathbf{d_{i+1} = 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1) - 2r^2}$

Hence $\mathbf{d_{i+1} - d_i = 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1) - 2r^2 -}$

$\mathbf{2(x_i + 1)^2 - y_i^2 - (y_i - 1) + 2r^2}$

$= 2(x_{i+1}+1)^2 + y^2_{i+1} + (y_{i+1} - 1)^2 - 2(x_i+1)^2 - y_i^2 - (y_i - 1)^2$

Since $x_{i+1} = x_i + 1$

$d_{i+1} = d_i + 4x_i + 2(y^2_{i+1} - y^2_i) - 2(y_{i+1} - y_i) + 6$

If T is the chosen pixel ($d_i < 0$) then $y_{i+1} = y_i$

$$d_{i+1} = d_i + 4x_i + 6$$

If S is the chosen pixel ($d_i > 0$) then $y_{i+1} = y_i - 1$

$$d_{i+1} = d_i + 4(x_i - y_i) + 10$$

Hence We have

$$d_{i+1} = \begin{cases} d_i + 4x_i + 6 & (d_i < 0) \\ d_i + 4(x_i - y_i) + 6 & (d_i > 0) \end{cases}$$

Bresenham's circle drawing algorithm

- Read the radius (r) of the circle.
- d=3-2r [Initialize the decision variable]
- x = 0, y = r [Initialize the starting point]
- do {

```
        plot(x,y)
        if( d<0) then
                {  d=d +4x+6 }
        else
                { d = d+4(x-y) +10
                  y=y-1 }
        x = x+1
        } while (x<y)
```

- Stop

Mid-Point Circle Algorithm

- Similarly to the case with lines, there is an incremental algorithm for drawing circles – the *mid-point circle algorithm*
- In the mid-point circle algorithm we use eight-way symmetry so only ever calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points

The mid-point circle algorithm was developed by Jack Bresenham, who we heard about earlier. Bresenham's patent for the algorithm can be viewed here.

Mid-Point Circle Algorithm (cont…)

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

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

- 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 (cont…)

- 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:

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

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

- If $p_k < 0$ the midpoint is inside the circle and 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 (cont…)

- To ensure things are as efficient as possible we can do all of our calculations incrementally
- First consider:

$$p_{k+1} = f_{circ}\left(x_{k+1}+1, y_{k+1}-\tfrac{1}{2}\right)$$
$$=[(x_k+1)+1]^2+\left(y_{k+1}-\tfrac{1}{2}\right)^2-r^2$$

- or:

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

- where y_{k+1} is either y_k or y_k-1 depending on the sign of p_k

Mid-Point Circle Algorithm (cont…)

- The first decision variable is given as:

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

$$= 1 + (r - \tfrac{1}{2})^2 - r^2$$

$$= \tfrac{5}{4} - r$$

- Then if $p_k < 0$ then the next decision variable is given as:

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

- If $p_k > 0$ then the decision variable is:

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

The 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:

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

- Starting with $k = 0$ at each position x_k, perform the following test. If $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and:

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

The Mid-Point Circle Algorithm (cont…)

- Otherwise the next point along the circle is (x_k+1, y_k-1) and:

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

4. Determine symmetry points in the other seven octants
5. Move each calculated pixel position (x, y) onto the circular path centred at (x_c, y_c) to plot the coordinate values:

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

6. Repeat steps 3 to 5 until $x >= y$

Mid-Point Circle Algorithm Example

- To see the mid-point circle algorithm in action lets use it to draw a circle centred at (0,0) with radius 10

MIDPOINT CIRCLE DRAWING ALGORITHM

- Read the radius (r) of the circle.
- Initialize starting position as

  x= 0 and y = r
- Calculate initial value of decision parameter as

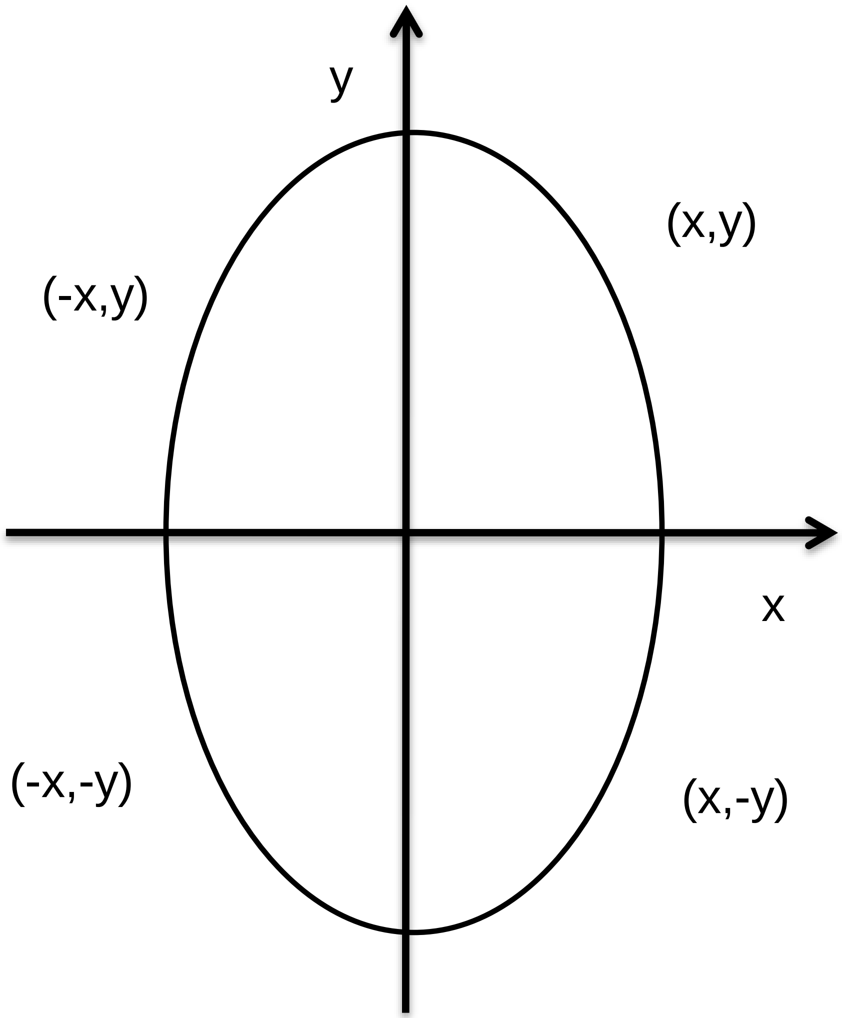
  d = 1.25 – r or (5/4-r)
- do {

```
        plot (x,y)
        if (d<0)
        {       x = x+1
                y = y
                d = d+2x +1 }
        else
        {       x = x+1
                y = y-1
                d = d+2x-2y+1
}while (x<y)
```

- Determine symmetry points
- Stop.

ELLIPSE DRAWING ALGORITHM

- The midpoint ellipse drawing algorithm uses the four way symmetry of the ellipse to generate it.



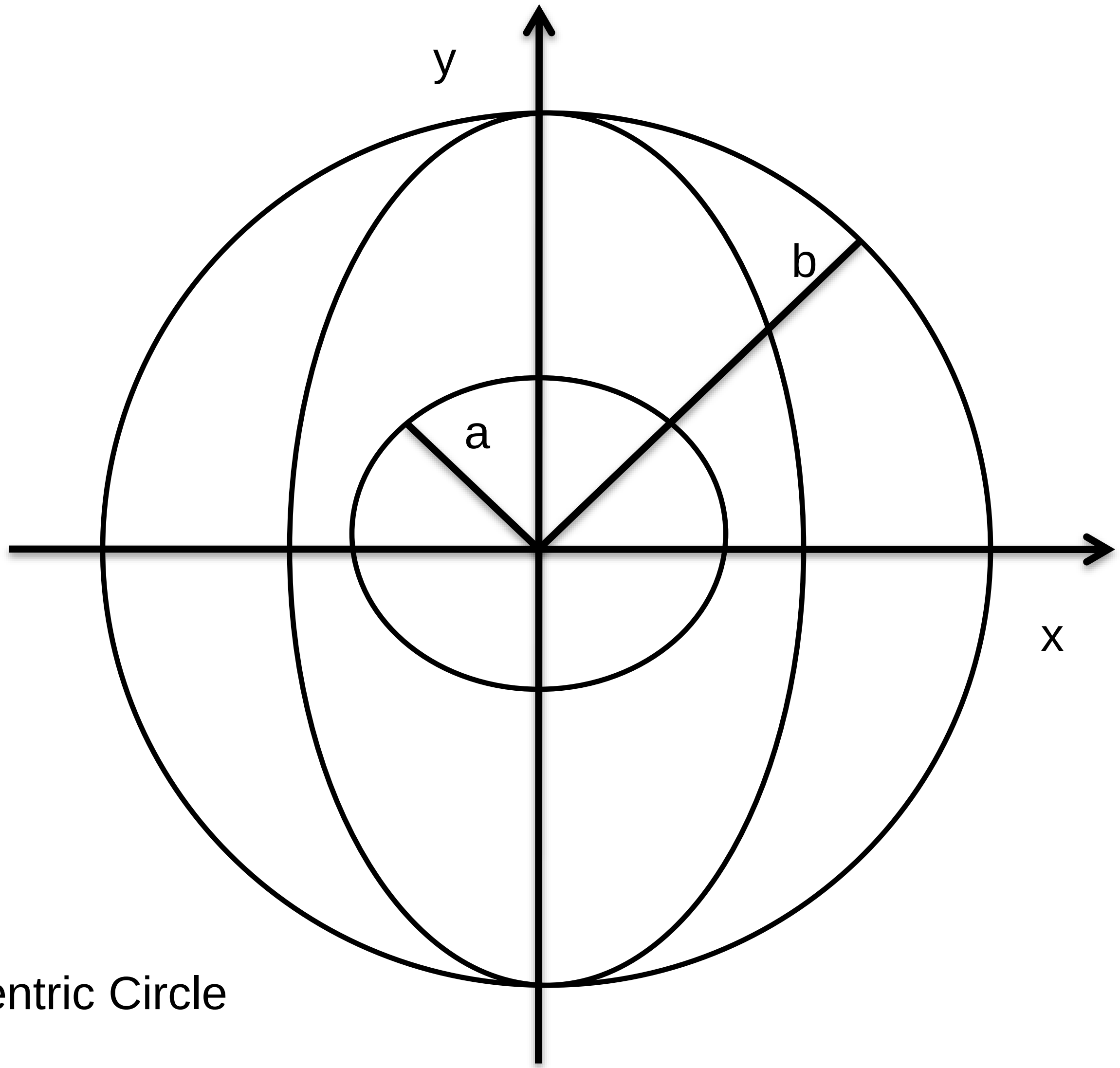
MIDPOINT ELLIPSE ALGORITHM

- The ellipse equation and define function f that can be used to decide if the midpoint between two candidate pixels in inside or outside the ellipse:

$$f(x,y) = b^2x^2+a^2y^2 -a^2b^2 \begin{cases} <0 & (x,y) \text{ inside the ellipse} \\ =0 & (x,y) \text{ on the ellipse} \\ >0 & (x,y,) \text{ outside the ellipse} \end{cases}$$

Region 1 : Vertical
Region2 : Horizontal

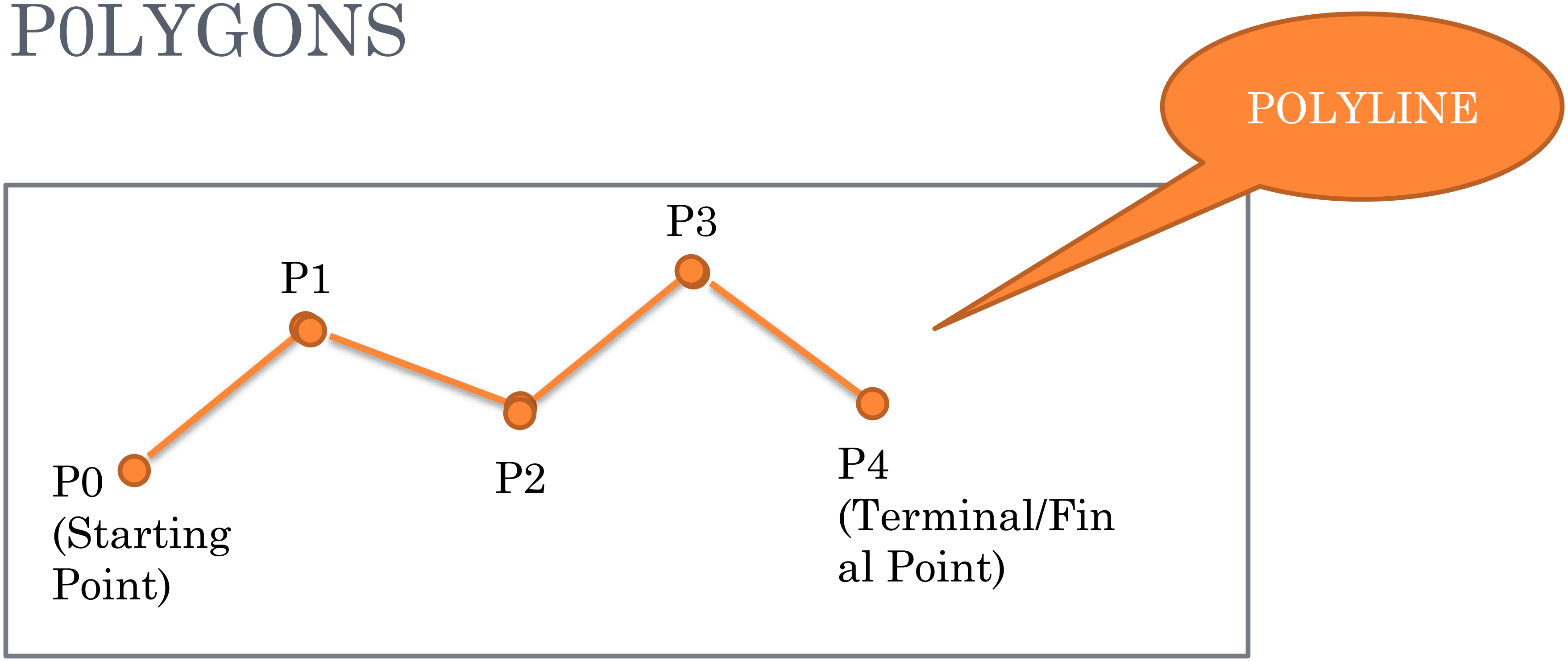
$$x^2/a^2 + y^2/b^2 = 1$$

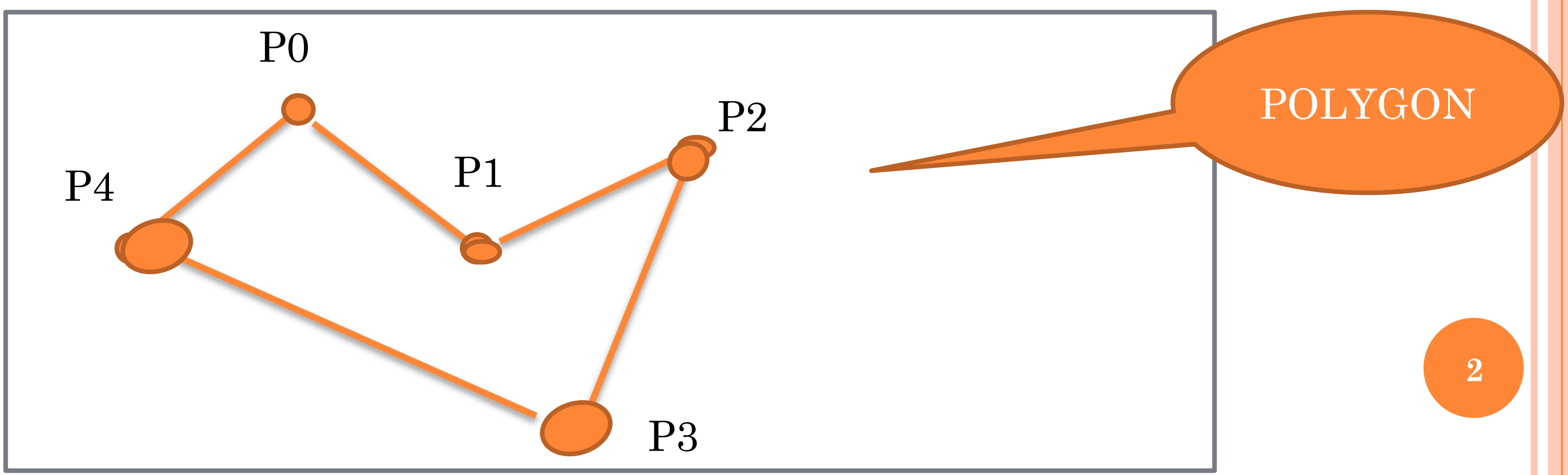


Concentric Circle

FILLED AREA PRIMITIVES

P0LYGONS

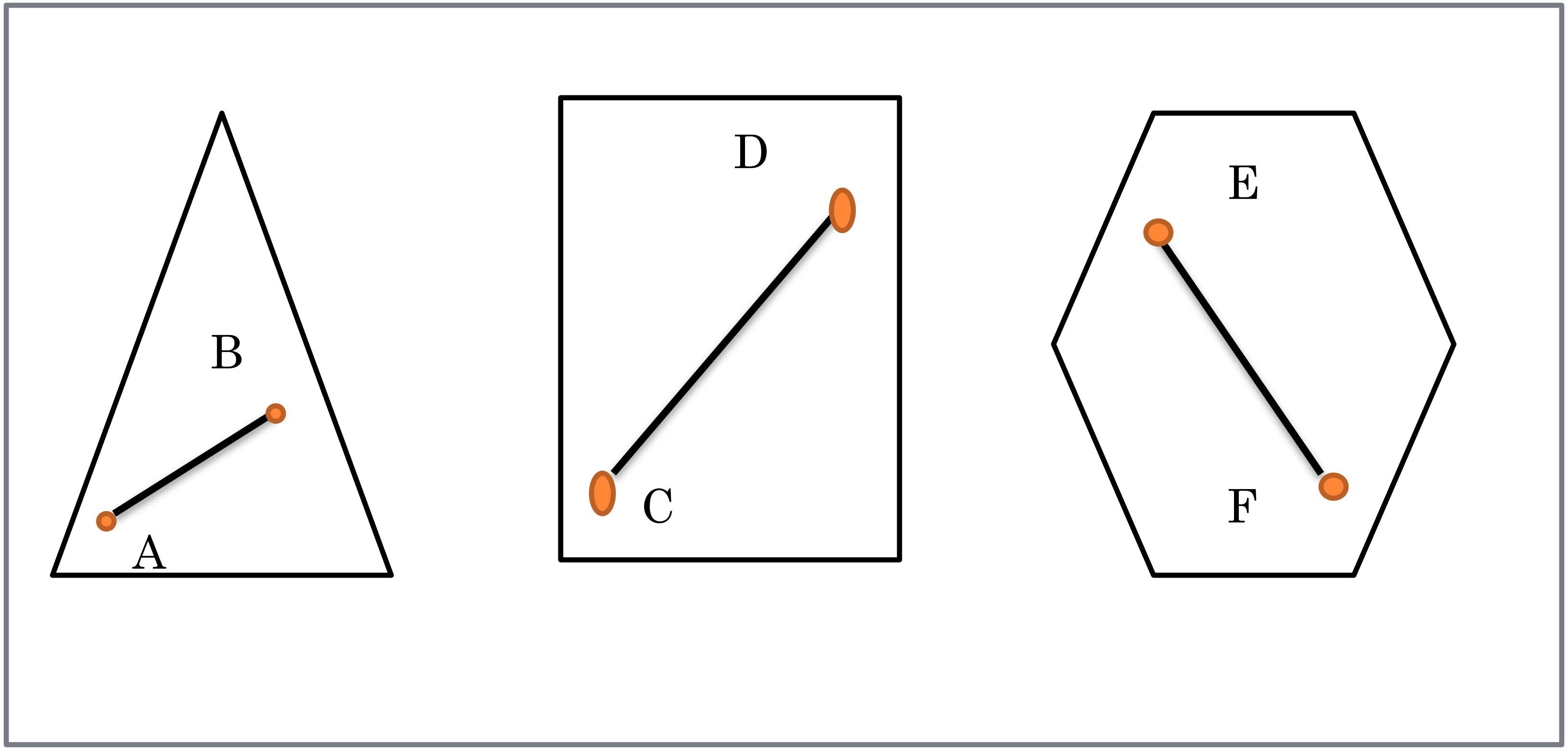

POLYLINE
P0
(Starting
Point)
P1
P2
P3
P4
(Terminal/Fin
al Point)


POLYGON
P0
P1
P2
P3
P4

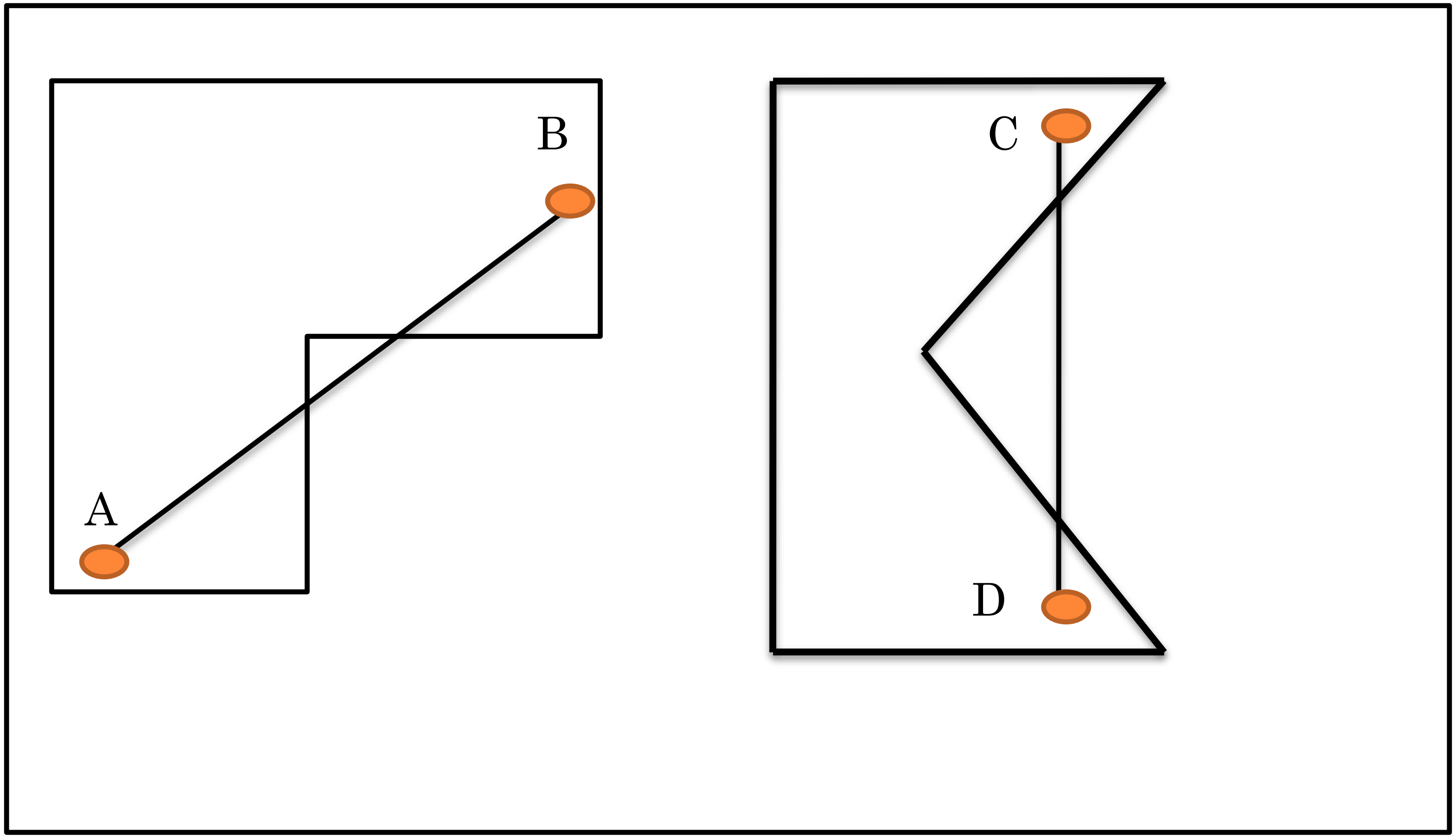
Types of polygons

- Convex Polygons
- Concave Polygons

CONVEX POLYGONS



Concave polygons



Representation of polygons

- Polygon drawing primitive Approach.
- Trapezoid primitive Approach.
- Line and Point Approach

EXAMPLES

Fig : Polygon

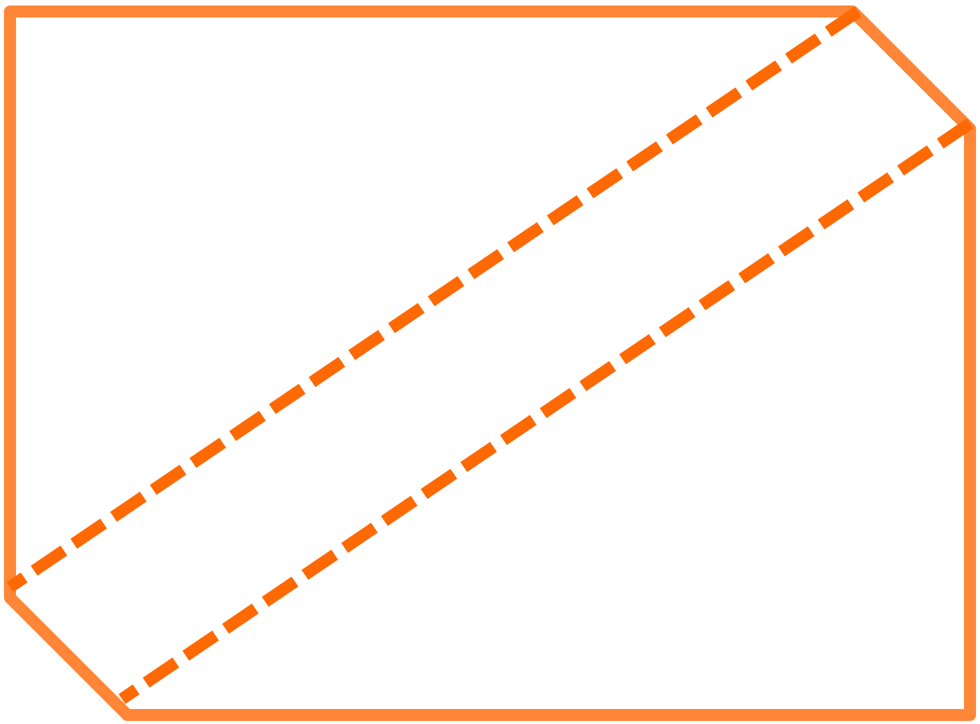
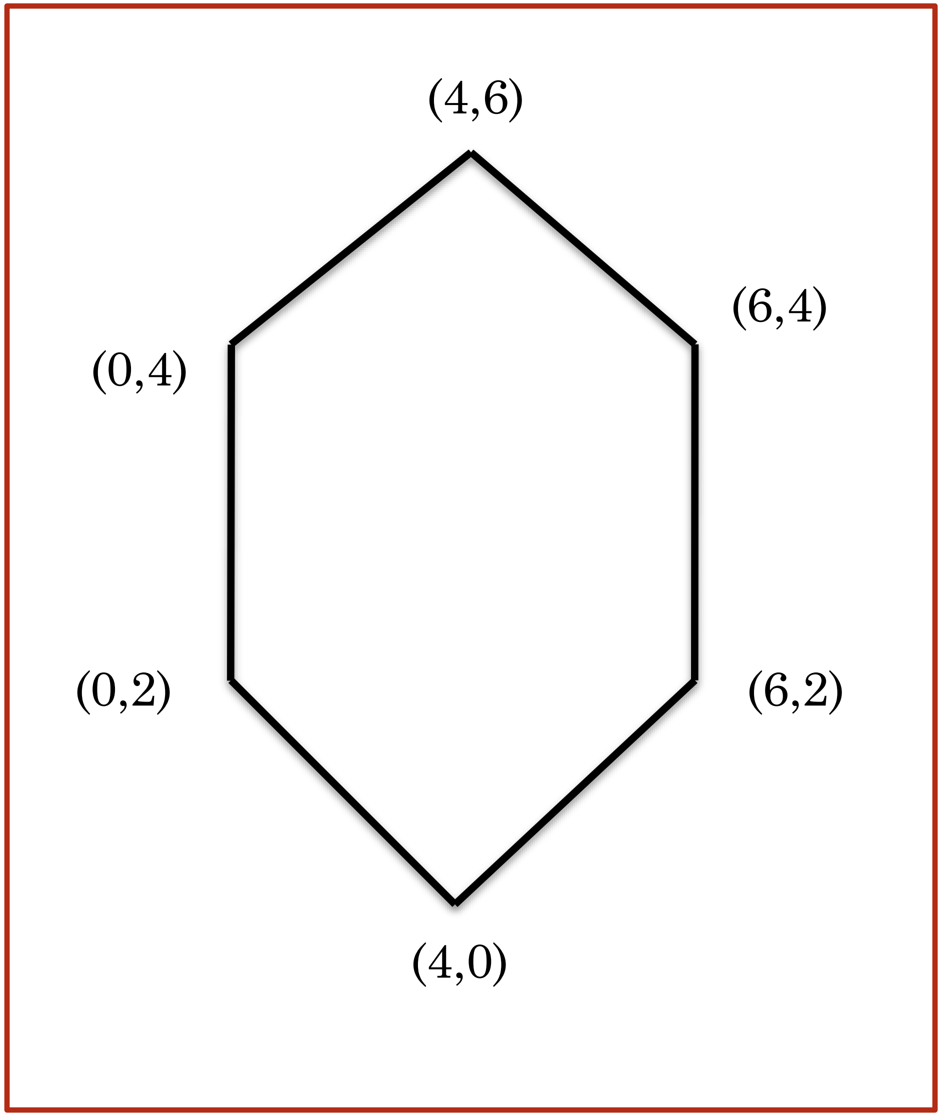


Fig : Representations a series of trapezoids

Lines and points approach



DF_OP	DF_x	DF_y
6	0	2
2	0	4
2	4	6
2	6	4
2	6	2
2	4	0
2	0	2

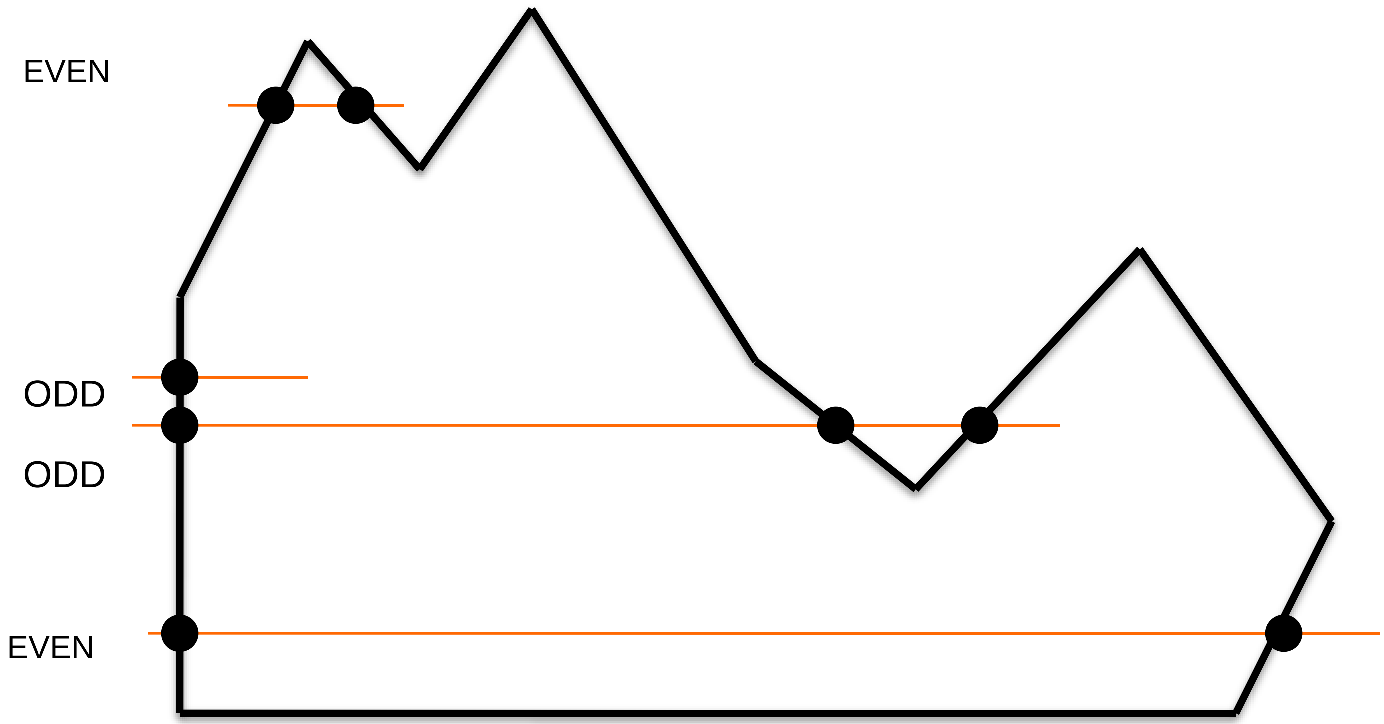
Fig: Polygon and its representation using display file

Algorithm : Entering the polygon into the display file

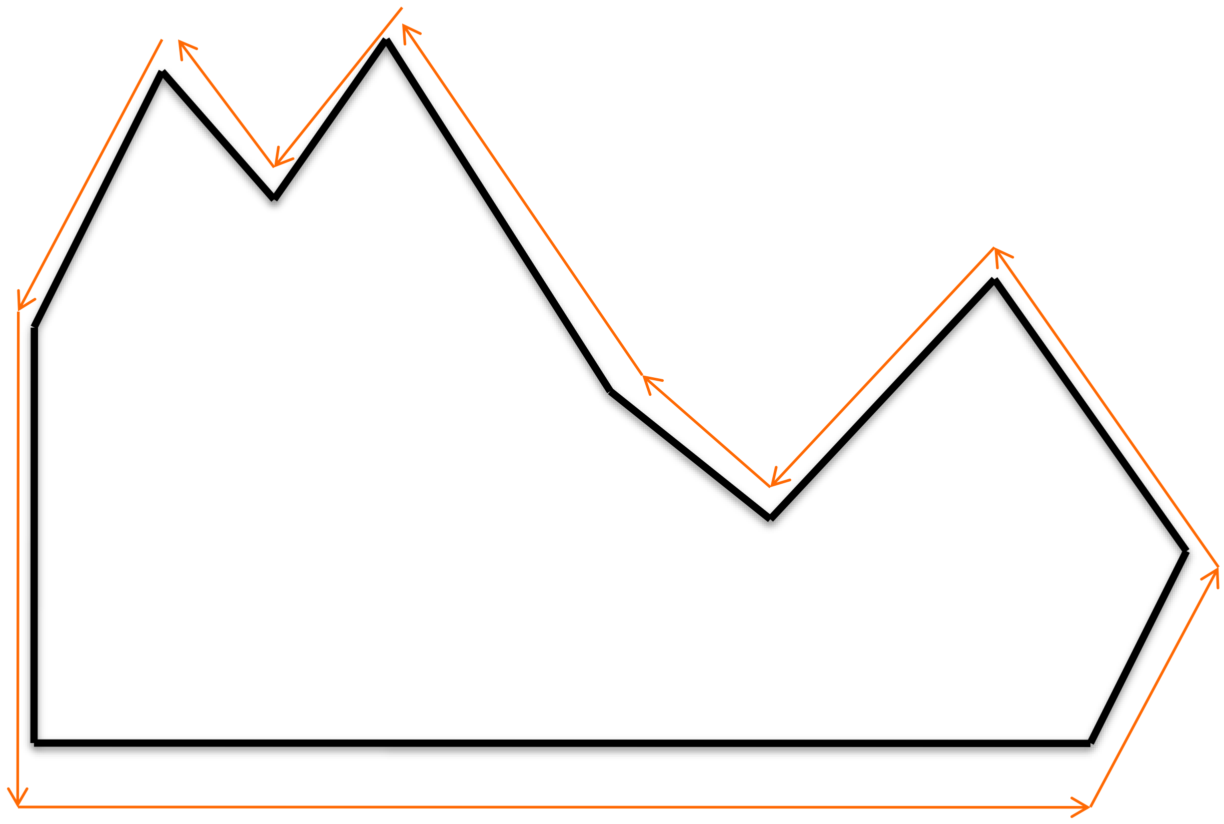
1. Read AX and AY of Length N
2. i=0
   DF_OP[i] ⟵ N
   DF_x[i] ⟵ AX[i]
   DF_y[i] ⟵ AY[i]
   i=i+1
   [Load Polygon Command]
3. do {
   DF_OP[i] ⟵ 2
   DF_x[i] ⟵ AX[i]
   DF_y[i] ⟵ AY[i]
   i ⟵ i+1
   } while(i<N) [Enter line commands]
4. DF_OP[i] ⟵ 2
   DF_x[i] ⟵ AX[0]
   DF_y[i] ⟵ AY[0]
   [Enter last line command]
5. Stop

AN INSIDE-OUTSIDE TEST

Odd-Even Rule



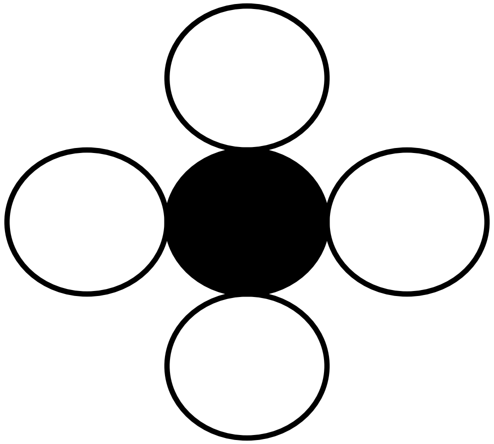
WINDING NUMBER RULE



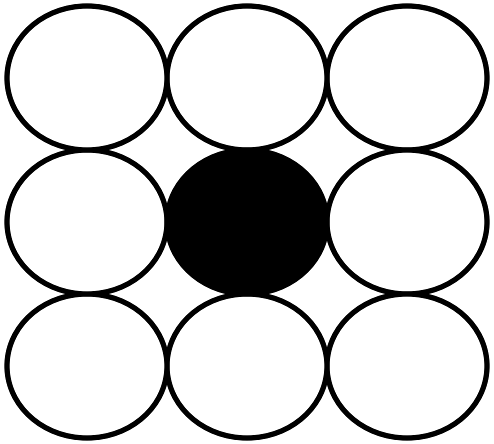
Polygon Filling

- Boundary Fill Algorithm
- Flood Fill Algorithm

Boundary Fill algorithm

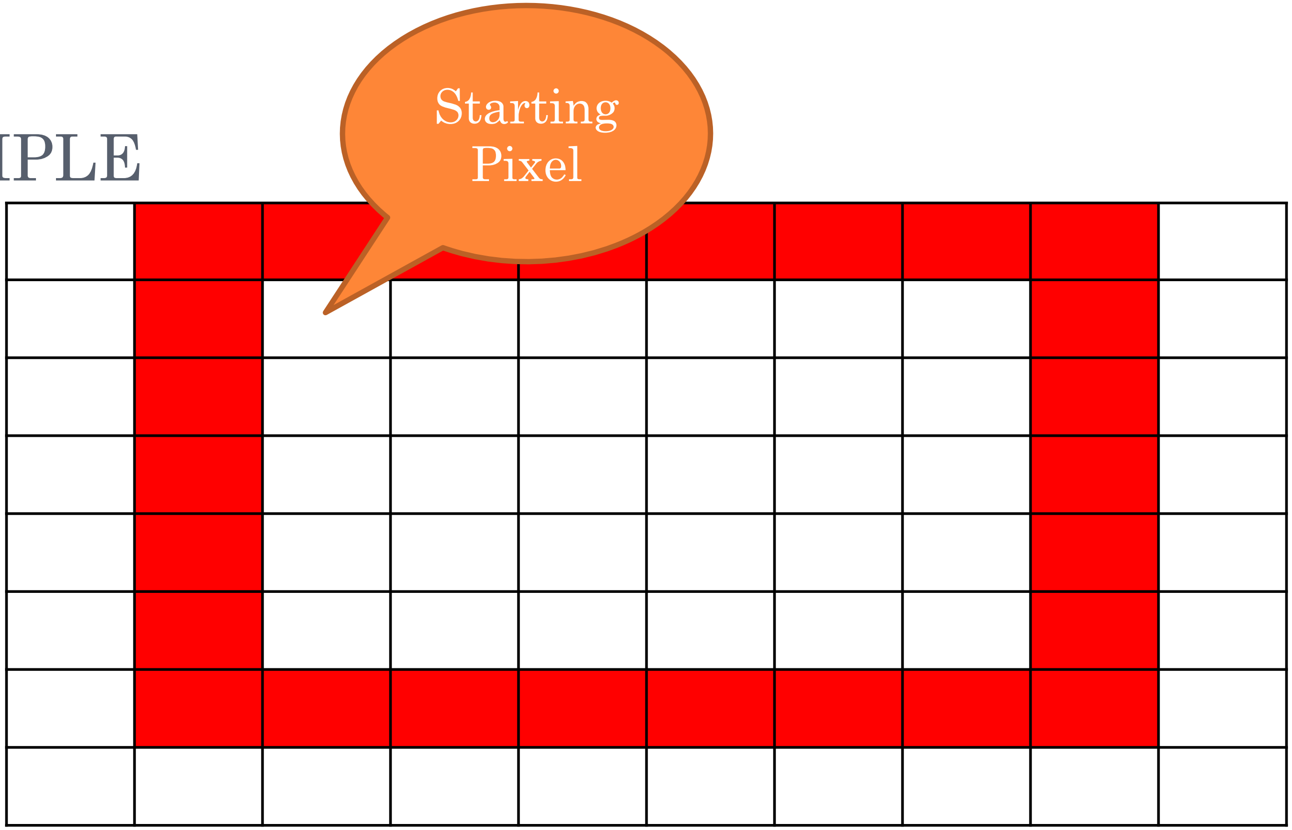


4 Connected Region

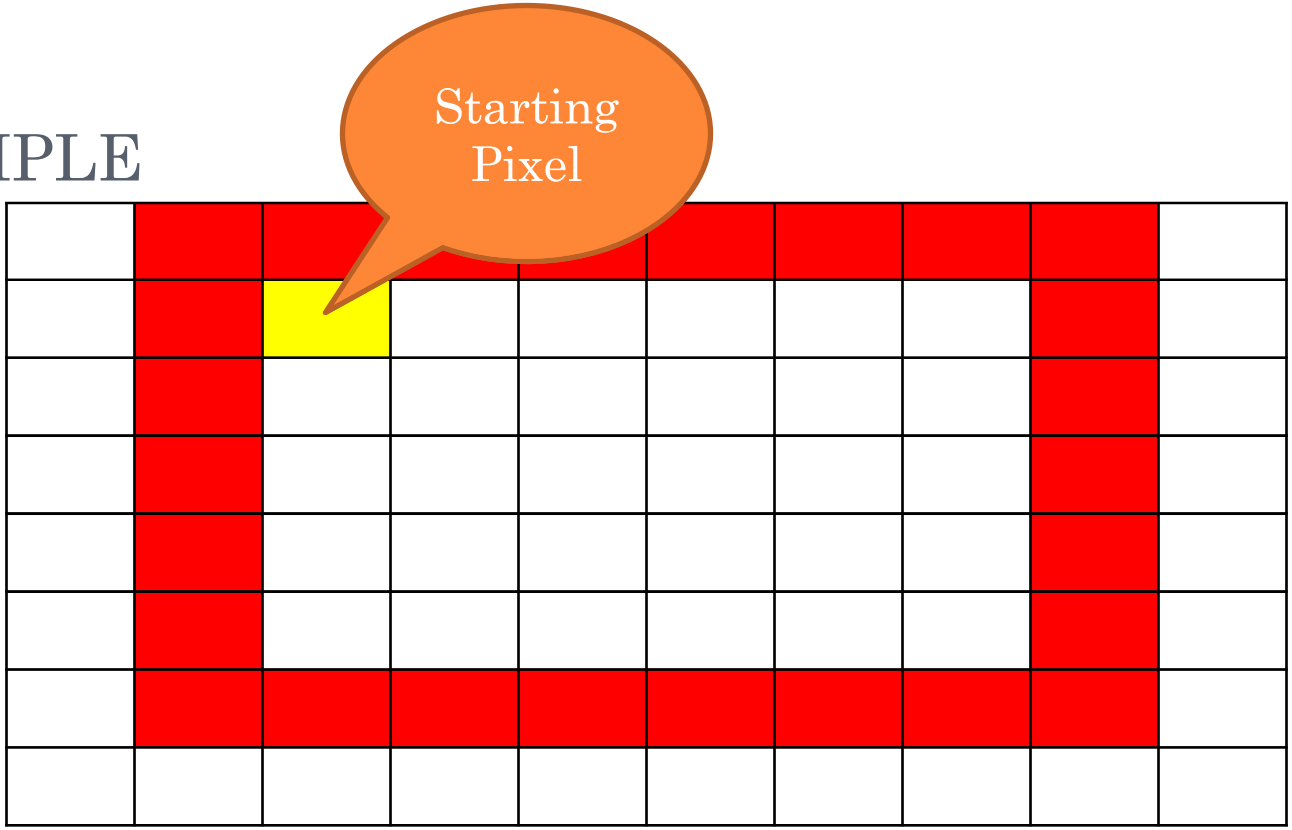


8 Connected Region

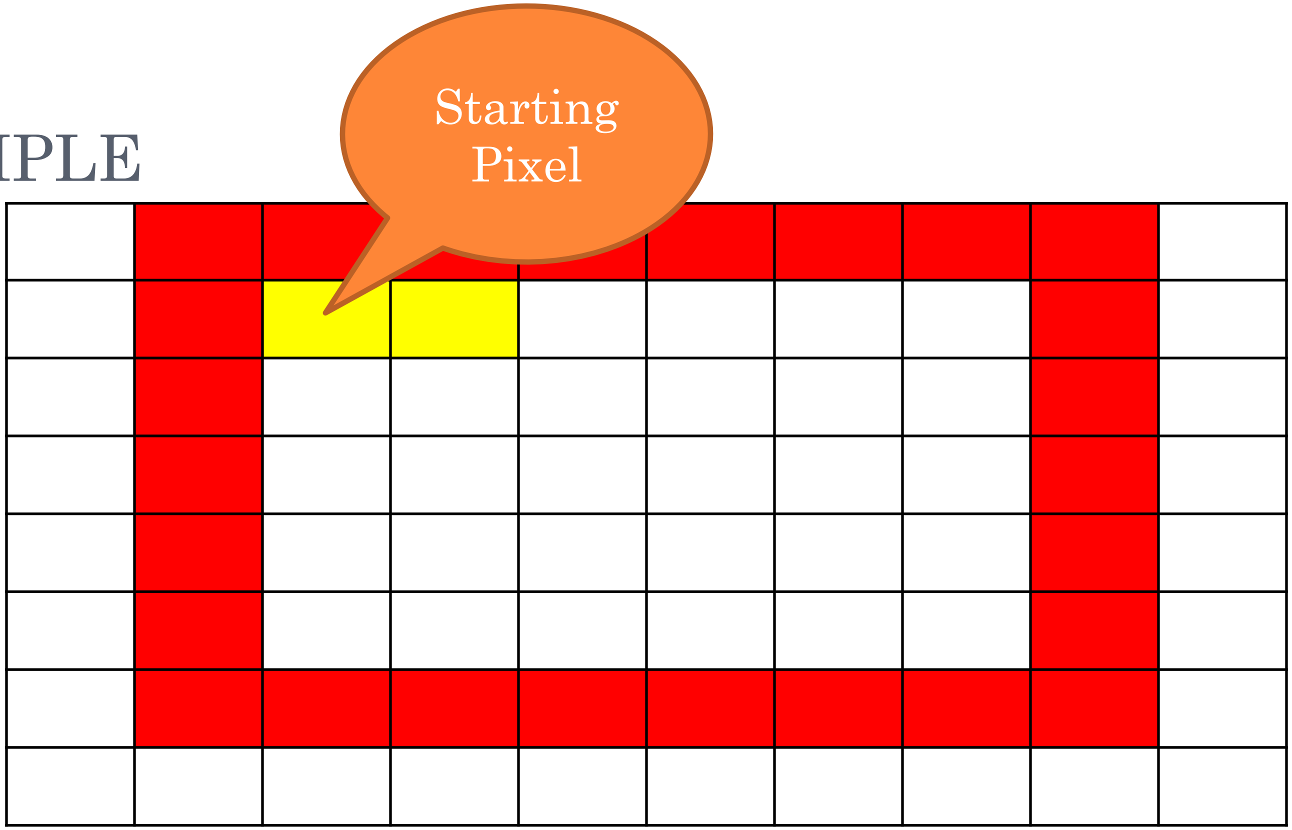
EXAMPLE



Example

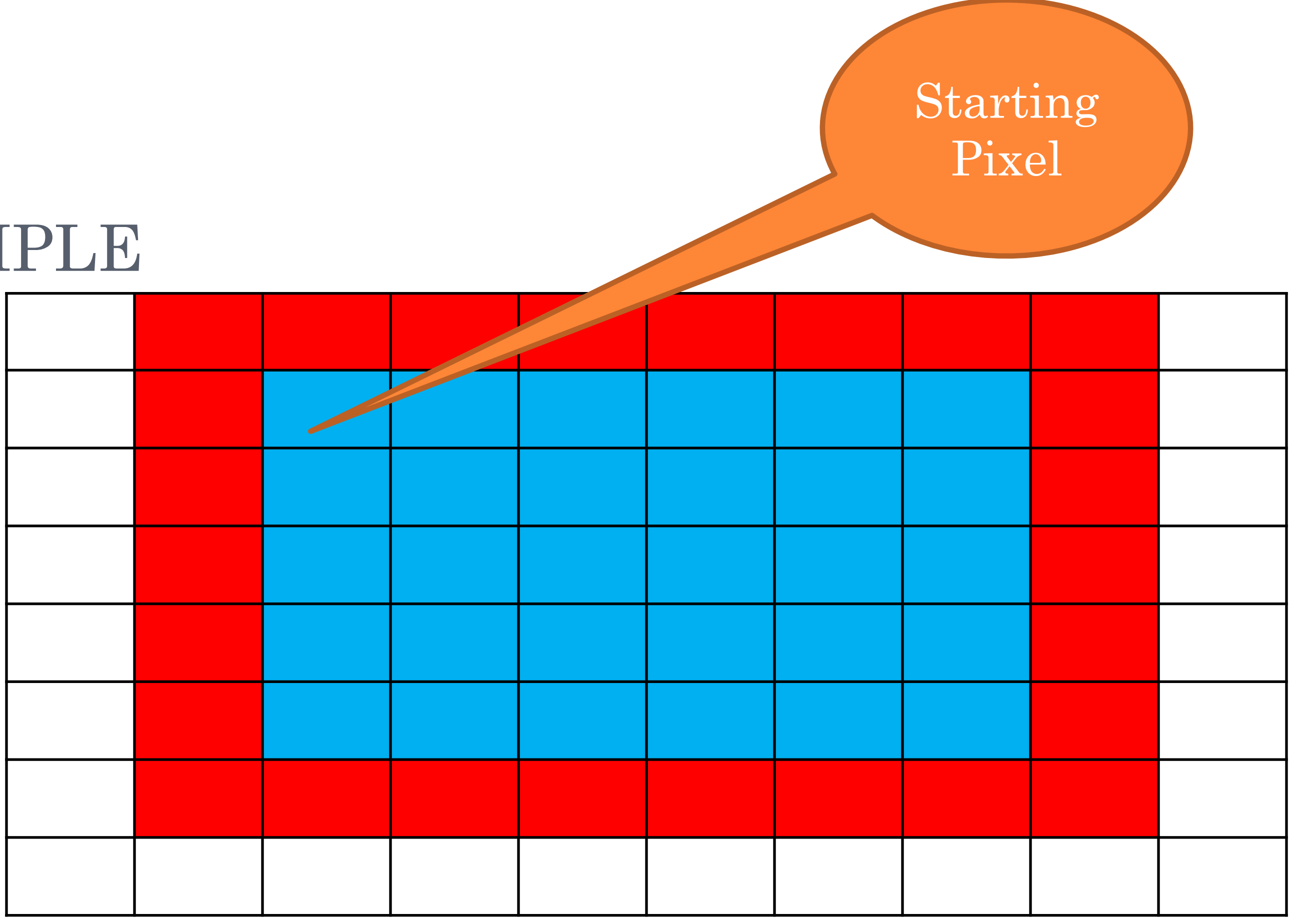


EXAMPLE



```
Procedure : boundary_fill(x,y,f_colour,b_colour)
{
  if (getpixel(x,y)!=b_colour && getpixel(x,y)!=f_colour)
        {
              putpixel(x,y,f_colour);
              boundary_fill(x+1,y,f_colour,b_colour);
              boundary_fill(x,y+1,f_colour,b_colour);
              boundary_fill(x-1,y,f_colour,b_colour);
              boundary_fill(x,y-1,f_colour,b_colour);
        }
}
```

EXAMPLE

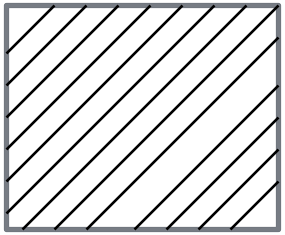


Flood fill algorithm

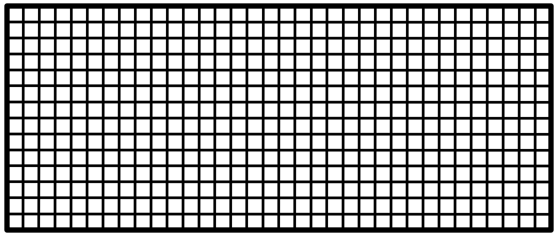
```
Procedure : flood_fill(x,y, fill_color,old_color)
{
        if (getpixel(x,y)==old_color)
                {
                        putpixel(x,y,fill_color);
                        flood_fill (x+1,y,fill_color,old_color);
                        flood_fill (x,y+1,fill_color,old_color);
                        flood_fill (x-1,y,fill_color,old_color);
                        flood_fill (x,y-1,fill_color,old_color);
                }
}
```

FILLING PATTERN

Name	Value	Result
EMPTY_FILL	0	Background coloe
SOLID_FILL	1	Solid fill
LINE_FILL	2	Line fill -----
LTSLASH_FILL	3	////
SLASH_FILL	4	///// thick line
BKSLASH_FILL	5	////// thick line
LTSLASH_FILL	6	\\\\\

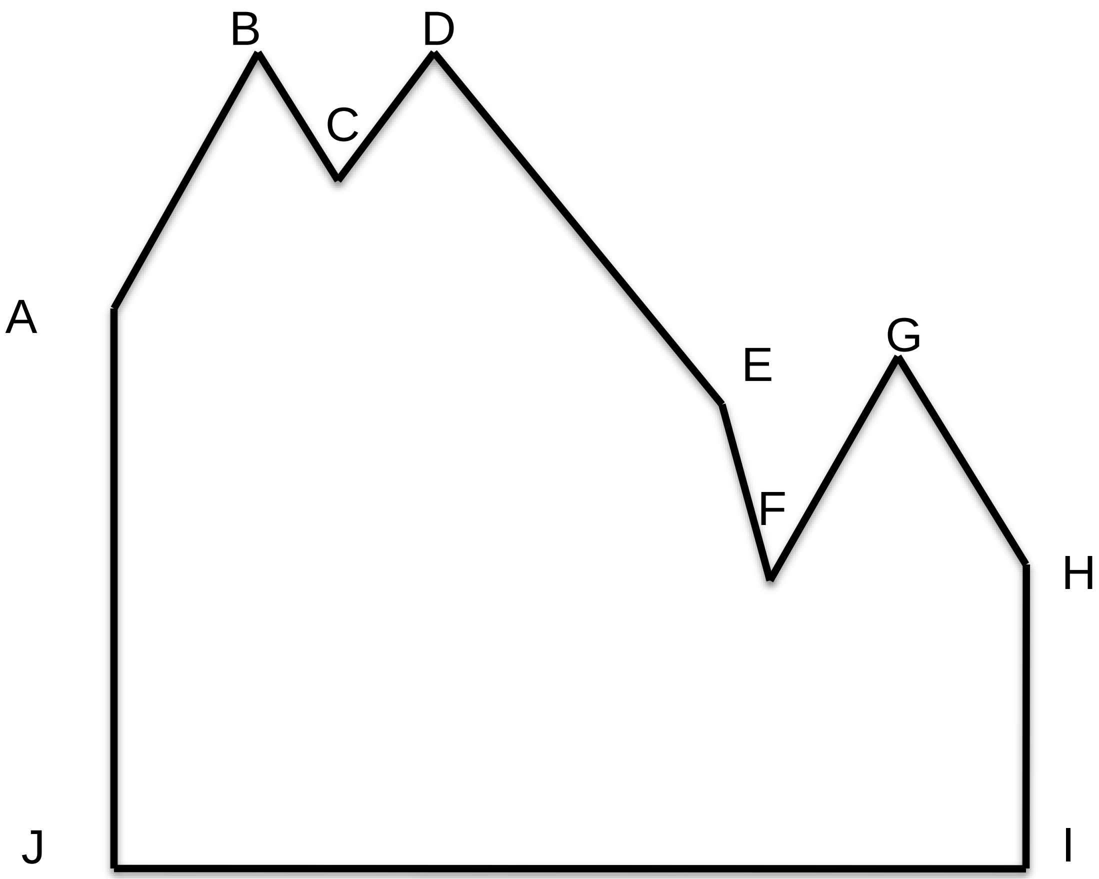


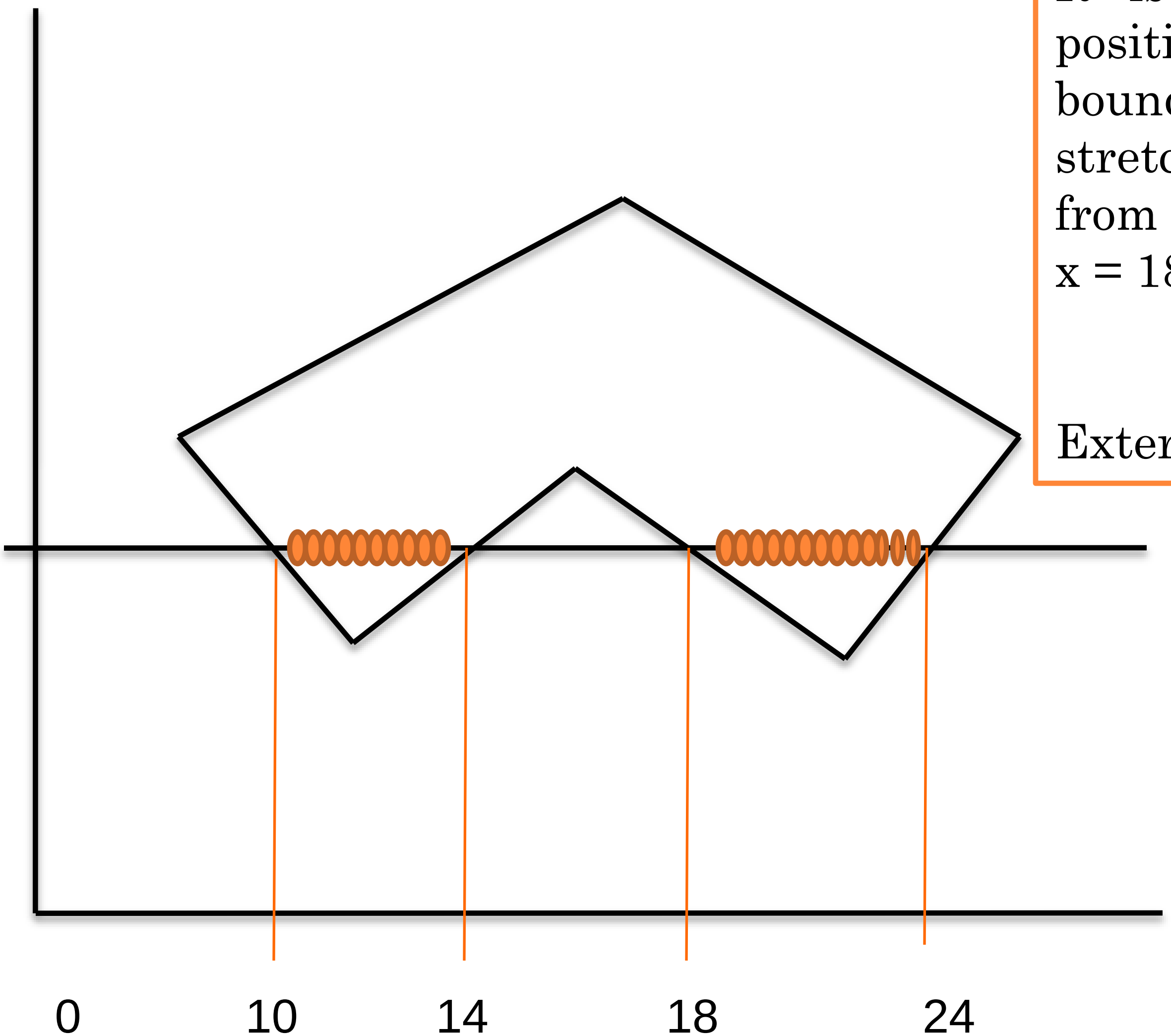
Name	Value	Meaning
HATCH_FILL	7	Light Hatch
XHATCH_LINE	8	Heavy Hatch
INTERLEAVE_FILL	9	Interleaving lines
WIDE_DOT_FILL	10	Widely Spaced dots
CLOSE_DOT_FILL	11	Closely Spaced dots
USE_FILL	12	User-defined fill pattern



```
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
void main()
{
 int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\tc\\bgi");
 setcolor(1);
 rectangle(100,100,200,150);
 setfillstyle(SOLID_FILL,4); or setfillstyle(SOLID_FILL,4);
 floodfill(103,103,1);
}
```

Scan line algorithm for filling polygon





It is four pixel intersection positions with the polygon boundaries define two stretches of interior pixels from x =10 to x= 14 and from x = 18 to x = 24.

Exterior Pixels : from 14 to 18

Polygon filling algorithms

1. Plot one octant of a circle of radius 7 pixels with the origin at the origin.
2. Plot all octant of a circle having radius of 14 pixels with its origin at the centre.

Polygon filling algorithms

Objectives

- Categorize the two basic approaches for area filling on raster systems.
- List out the applications of the two approaches.
- Boundary fill algorithm.
- Flood fill algorithm
- Scan line fill algorithm.

Region Filling

Seed Fill Approaches .

- **Start from a given interior position and paint outward from this point until the specified boundary condition is encountered.**
  - **2 algorithms:**
  - **Boundary Fill and Flood Fill**
  - **works at the pixel level**
    - **suitable for interactive painting applications**

Boundary Fill Algorithm

- **Start at a point inside a region.**
- **Paint the interior outward to the boundary.**
- **The edge must be specified in a single color.**
- **Fill algorithm proceeds outward pixel by pixel until the boundary color is encountered.**

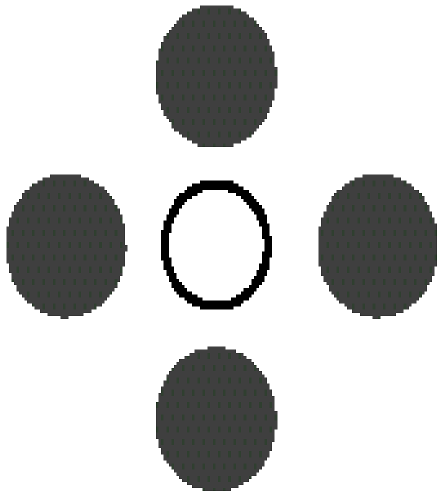
- The procedure accepts as input the coordinates of an interior point (x, y), fill color and a boundary color .
- Starting from (x, y) the procedure tests neighbouring positions to determine whether they are of the boundary color.
- If not paint them with fill color and test their neighbours and process continues until all pixels up to the boundary color for the area have been tested.

There are 2 methods for proceeding to neighbouring pixels from the current test positions.

- **4 connected method.**
- **8 connected method.**

- **4-connected region**: From a given pixel, the region that you can get to by a series of 4 way moves (N, S, E and W).
- The **neighbouring 4 pixel positions are tested.**
- If the selected pixel is **(x, y)** the neighbouring pixels are **(x+1, y) , (x-1, y) (x, y+1) , (x, y-1)**
- 4-connected fill is faster,

  but can have problems



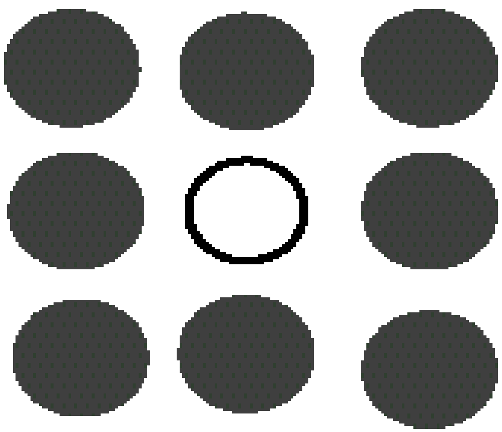
4-connected

- **8-connected region**: **From a given pixel, the region that you can get to by a series of 8 way moves (N, S, E, W, NE, NW, SE, and SW), the 4 diagonal pixels are also included.**
- **If the selected pixel is (x, y) the 8 neighbouring pixels are**

(x+1, y) ,(x-1, y), (x, y-1), (x, y+1)

(x+1, y+1) ,(x-1, y+1)

(x-1, y-1), (x+1, y-1)



8-connected

Boundary Fill Algorithm (cont.)

```
void BoundaryFill4(int x, int y,
                   color newcolor, color edgecolor)
{
  int current;
  current = ReadPixel(x, y);
  if(current != edgecolor && current != newcolor)
  {
     BoundaryFill4(x+1, y, newcolor, edgecolor);
     BoundaryFill4(x-1, y, newcolor, edgecolor);
     BoundaryFill4(x, y+1, newcolor, edgecolor);
     BoundaryFill4(x, y-1, newcolor, edgecolor);
  }
}
```

Flood Fill Algorithm

- Used when an area defined with multiple color boundaries.
- Start at a point inside a region
- Replace a specified interior color (old color) with fill color instead of searching for a boundary color value and the method is called flood fill.

- Start from a specified interior point (x, y) and reassign all pixel values that are set to a given interior color with the desired fill color.
- Fill the 4-connected or 8-connected region until all interior points being replaced.

Flood Fill Algorithm (cont.)

```
void FloodFill4(int x, int y, color newcolor,
color oldColor)
{
  if(ReadPixel(x, y) == oldColor)
  {
     FloodFill4(x+1, y, newcolor, oldColor);
     FloodFill4(x-1, y, newcolor, oldColor);
     FloodFill4(x, y+1, newcolor, oldColor);
     FloodFill4(x, y-1, newcolor, oldColor);
  }
}
```

Scan line Scan Line Polygon Fill Algorithms

Scan line Fill Approaches.

Fill an area by determining the overlap intervals for scan lines that cross that area.

- works at the polygon level
- used in general graphics packages to fill polygons, circles etc.
- better performance.