Computer Graphics (210244)

Teaching Scheme

Lecture: 03 Hours/Week

Credit Scheme :03

Examination Scheme and Marks

Mid_Semester (TH): 30 Marks

End_Semester (TH): 70 Marks

Subject Teacher- Chetana Shravage

Course Objectives:

The Computer Graphics course prepares students for activities involving the design, development, and testing of modeling, rendering, and animation solutions to a broad variety of problems found in entertainment, sciences, and engineering.

- **Remembering:** To acquaint the learner with the basic concepts of Computer Graphics.
- **Understanding:** To learn the various algorithms for generating and rendering graphical figures.
- **Applying:** To get familiar with mathematics behind the graphical transformations.
- **Understanding:** To understand and apply various methods and techniques regarding projections, animation, shading, illumination and lighting.
- **Creating:** To generate Interactive graphics using OpenGL.

Course Outcomes:

On completion of the course, learner will be able to–

- CO1: **Identify** the basic terminologies of Computer Graphics and interpret the mathematical foundation of the concepts of computer graphics.
- CO2: **Apply** mathematics to develop Computer programs for elementary graphic operations.
- CO3: **Describe** the concepts of windowingand clipping and apply various algorithms to fill and clip polygons.
- CO4: **Understand** and apply the core concepts of computer graphics, including transformation in two and three dimensions, viewing and projection.
- CO5: **Understand** the concepts of color models, lighting, shading models and hidden surface elimination.
- CO6: **Describe** the fundamentals of and implement curves, fractals, animation and gaming.

Course Unit-06

SR.NO	UNIT TITLE	Mapping of Course Outcomes
1.	Graphics Primitives and Scan Conversion Algorithms (07 Hours)	CO1,CO2
2.	Polygon, Windowing and Clipping (07 Hours)	CO2,CO3
3.	2D, 3D Transformations and Projections (07 Hours)	CO2,CO4
4.	Light, Colour, Shading and Hidden Surfaces (07 Hours)	CO5
5.	Curves and Fractals (07 Hours)	CO2,CO6
6.	Introduction to Animation and Gaming (07 Hours)	CO6

Unit-01
Graphics Primitives and Scan Conversion Algorithms (07 Hours)

Course Contents:

- Introduction, graphics primitives - pixel, resolution, aspect ratio, frame buffer. Display devices, applications of computer graphics.

- **Introduction to OpenGL** - OpenGL architecture, primitives and attributes, simple modelling and rendering of two- and three-dimensional geometric objects, GLUT, interaction, events and call-backs picking. (Simple Interaction with the Mouse and Keyboard)

- **Scan conversion:** Line drawing algorithms: Digital Differential Analyzer (DDA), Bresenham. Circle drawing algorithms: DDA, Bresenham, and Midpoint.

Computer Graphics

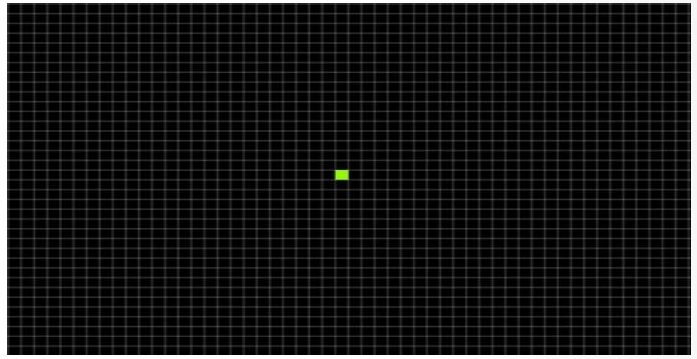
- Computer is a tool for storing, manipulating & correlating data.
- Computer is information processing machine.
- Graphics-Graph(Mathematical figures like line, circle etc.) + pics(Images)

- Computer Graphics is **an art of drawing pictures, lines, charts etc. on computer screen by using Programming language**.
- Computer Graphics is a process of **generating, manipulating, storing and displaying images**
- CG is one of the most **effective** and commonly used **way** to communicate the **processed information to the user**.
- Thus we can say that computer graphics makes it possible **to express data in pictorial form.**

Basics Computer Graphics

1) Pixel-

- In computer graphics objects are presented as a collection of discrete picture elements.
- **Discrete Picture Element** = Pixel
- The pixel is the smallest screen elements.



2) Resolution-

Resolution refers to **a number of dots** on the screen. It is expressed as a number of dots on horizontal line and the number of vertical lines.

Basics Computer Graphics..

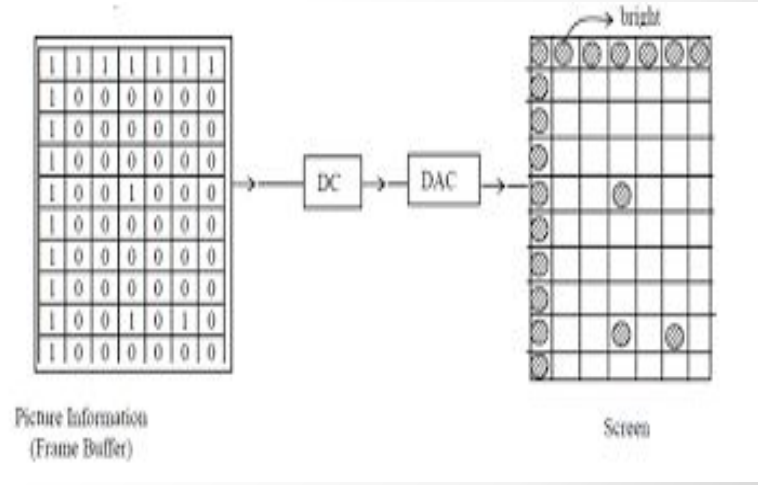
3) Aspect Ratio-

- Aspect ratio of **width to height** usually expressed as **x:y**
- When resizing picture it is important to maintain the aspect ratio to avoid stretching of the picture.

---

4) Frame Buffer-

- Frame memory is a memory area used to store picture information.
- Frame buffer is a large, contiguous piece of computer Memory(2D array structure).
- In frame buffer current image detail is stored in the Form of bits(each bit for single pixel)
- A block of memory dedicated to graphics output that Holds the content of what will be displayed.



Basics Computer Graphics..

- Graphics should be generated by controlling the pixel.
- The control is achieved by setting the intensity and colour of the pixel which compose screen.
- The process of determining 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”.

Basics Computer Graphics..

- What you can do with graphics before displaying it on screen?
- Graphics allows rotation, translation, scaling and performing various projections before displaying it.
- It also allows to add effects such as hidden surface removal, shading and transparency to the picture.

Basics Computer Graphics..

- User can edit (modify content, structure or appearance) graphics object with using keyboard, mouse or touch sensitive panel on the screen.

- There is close relationship between input devices and display devices.

- Graphics Devices = Input Devices + Display Devices
- **Input Devices-** Touch panel, Keyboard, Mouse, Light pen, graphics tablets, Joysticks, Film recorder.
- **Display Devices-** Cathode Ray Tube(CRT),Vector scan devices/Raster scan display, Colour monitor, LCD, LED, plasma

- **CG Software-** Photoshop, OpenGL, AutoCAD, Maya 3D, Corel Draw, Core graphics, Graphics Kernal system

Basic Computer Graphics..

- **Advantages of computer graphics :**
- High quality graphics displays on PC
- It provides tools for producing pictures
- Produce animation using static image with computer graphics
- Produce 1-D image in 2-D or 3-D using different simulators.
- Using motion dynamics tool, user can make object stationary and the viewer moving around them.
- Using update dynamics, it is possible to change the shape, colour or other properties of object.

Applications of Computer Graphics

- Graphics User Interface, Animation
- Entertainment ,Visual effects in Movies, TV Channels
- Car/Flight/Space simulation training, Virtual reality
- Auto CAD/CAM,PCB Designing
- Computer Art, Computer Games
- Map preparation(weather, geographical, population density)
- Art & commerce
- Image Processing, making charts
- Education,office automation, Desktop publishing(printing,scanning,designing)

Display Devices in Computer Graphics

- The display device is an **output device** used to represent the **information in the form of images (visual form).**
- Display systems are mostly called a **video monitor** or **Video display unit (VDU).**
- Display devices are designed to model, display, view, or display information.
- The **purpose** of display technology is to simplify information sharing.

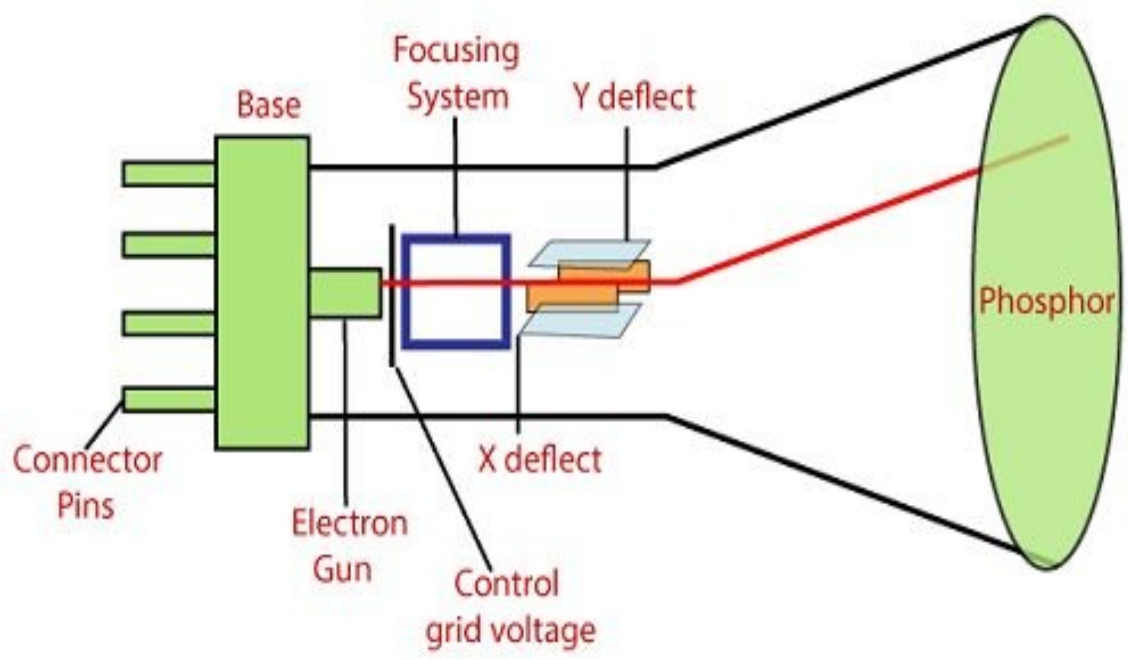
Display Devices in Computer Graphics..

There are some display devices given below:

1. Cathode-Ray Tube(CRT)
2. Random Scan (Vector Scan)
3. Raster scan Display
4. Color CRT Monitor
5. Liquid crystal display(LCD)
6. Light Emitting Diode(LED)
7. Plasma Display

1. Cathode-ray Tube (CRT)

- It is a technology which is used in traditional computer monitor and television.
- Cathode ray tube is a particular type of vacuum tube that displays images when an electron beam hit on the radiant surface.



□ **Advantages:**
- Real image
- Many colors to be produced
- Dark scenes can be pictured

□ **Disadvantages:**
- Less resolution
- Display picture line by line
- More costly

2.Random Scan (Vector scan)

- It is also known as stroke-writing display or calligraphic display.
- It uses an electron beam like a pencil to make a line image on the screen.
- The image is constructed from a sequence of straight-line segments.
- On the screen, each line segment is drawn by the beam to pass from one point on the screen to the other, where its x & y coordinates define each point.
- After compilation of picture drawing, the system cycle back to the first line and create all the lines of picture 30 to 60 times per second.

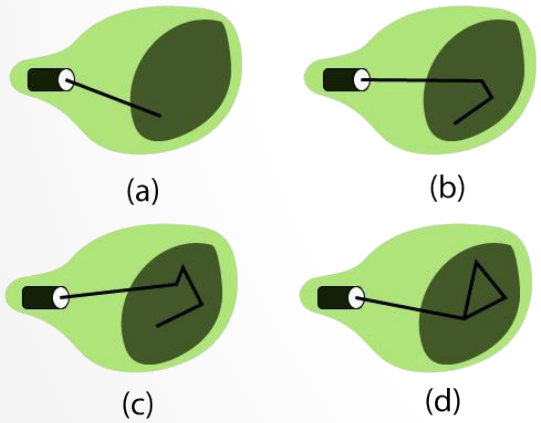
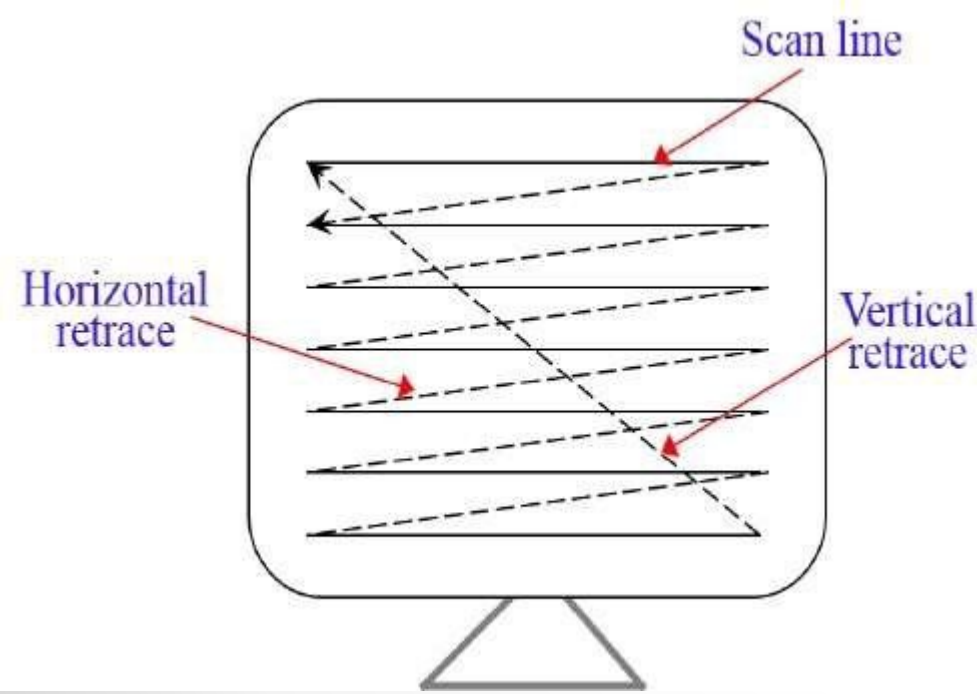


Fig: A Random Scan display draws the lines of an object in a specific order

- **Advantages:**
  - High Resolution
  - Draw smooth line Drawing
  - Less memory
- **Disadvantages:**
  - Can't handle complex or natural scenes
  - Costly

3.Raster Scan Display

- Raster system display the picture by drawing pixel in a row by row from left to right.
- Picture definition is stored in the memory called refresh buffer or frame buffer.
- After finishing each scan line electron guns are turned off and moved back to the first pixel of next scan line that horizontal movement is called **Horizontal retrace**.
- After reaching to the last pixel of the screen electron guns are moved at the first pixel of the first row, that vertical movement of electron guns is called **vertical retrace.**

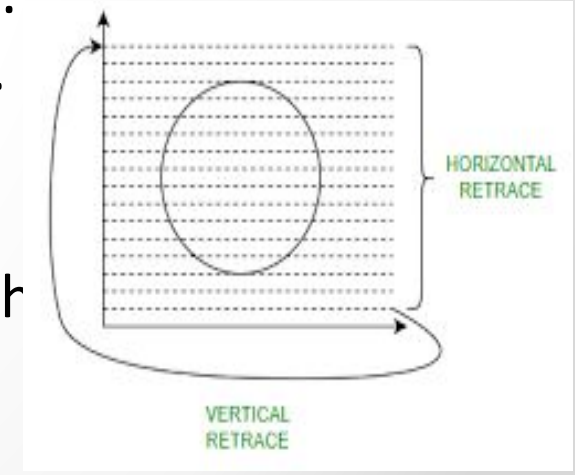


□ **Advantages:**

- Can handle complex,natural scenes,dynamic scenes.
- such displays economical.
- whole screen is scanned.

□ **Disadvantages:**

- Low resolution
- Modification is tough



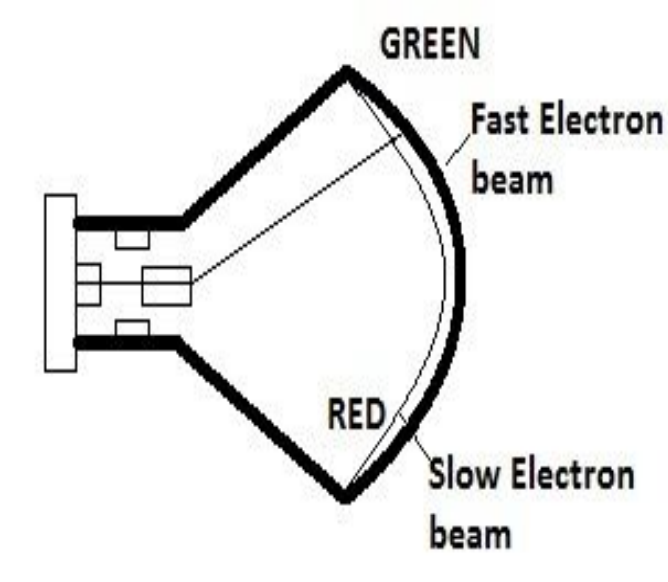
Difference Between Raster scan & Random scan display

Raster Scan Display	Random scan Display
1) The electron beam scan the entire screen to draw a picture	1) The electron beam scans only the part of the screen where picture information is present.
2) Low resolution	2) better Resolution
3) pixel location of the screen is used to draw an image.	3) Mathematical function are used to draw an image
4) Scan Conversion is required	4) Scan Conversion not required
5) Economical Devices	5) Costly
6) Used to display dynamic objects or scenes.	6) Used to display static objects.
7) While in raster scan, any modification is not so easy .	7) In random scan, any modification is easy in comparison of raster scan.
8) It stores picture definition as a set of intensity values of the pixels in the frame buffer.	8) It stores picture definition as a set of line commands in the Refresh buffer.

4. Color CRT Monitor

- It is similar to a CRT monitor.
- The basic idea behind the color CRT monitor is to combine three basic colors- Red, Green, and Blue.
- By using these three colors, we can produce millions of different colors.
- The two basic color display producing techniques are:

1. **Beam–Penetration Method-**Suitable for random scan display.



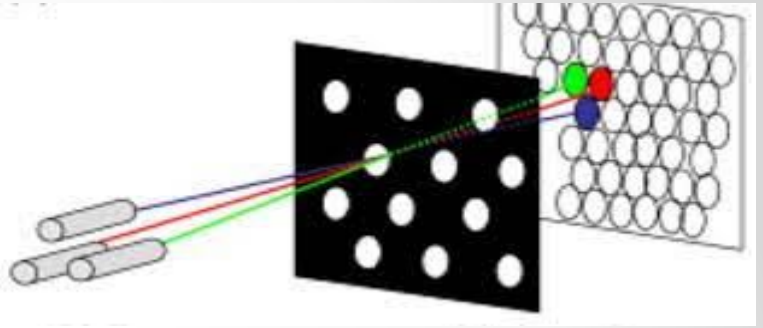
Advantages:

- Better Resolution
- Half cost
- Inexpensive

Disadvantages:

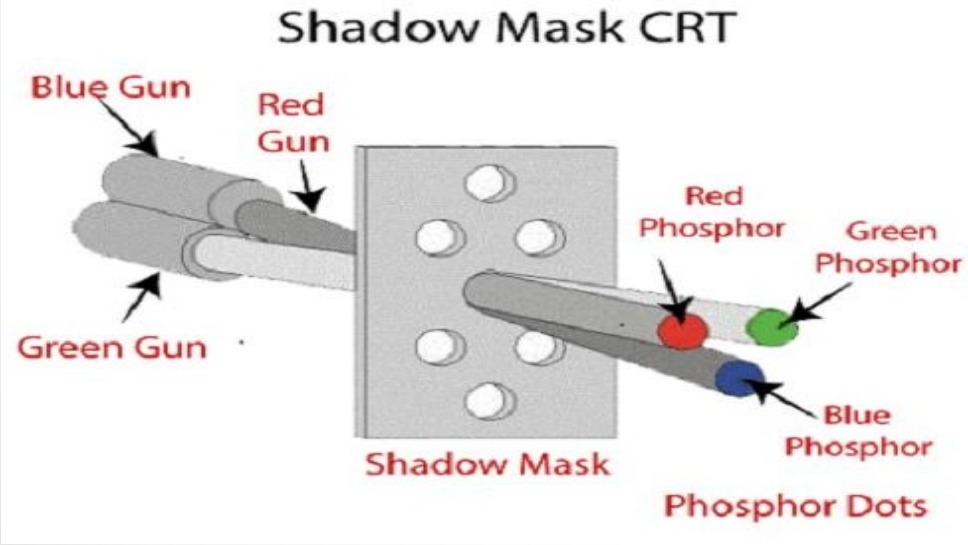
- Only four possible colors i.e. RGYO
- Time Consuming
- Quality of the picture is not as good as with other

Continue….



2. Shadow–Mask Method

- It is used with a **raster scan** monitor for displaying pictures.
- It has more range of color than the beam penetration method.
- It is used in television sets and monitors.



- **Advantages:**
  - Display a wider range picture.
  - Display realistic images.
  - In-line arrangement of RGB color.
- **Disadvantages:**
  - Difficult to cover all three beams on the same hole.
  - Poor Resolution.

5.Liquid crystal display (LCD)

- Used in Embedded system,Used to display status information,messages for user.
- The LCD flat panel display depends upon the light modulating properties of liquid crystals.
- LCD is used in calculator,watches and portable computers,TV,camera.
- LCD consumes less power than LED.
- Alphanumeric LCD & Graphical LCD

□ **Advantages:**

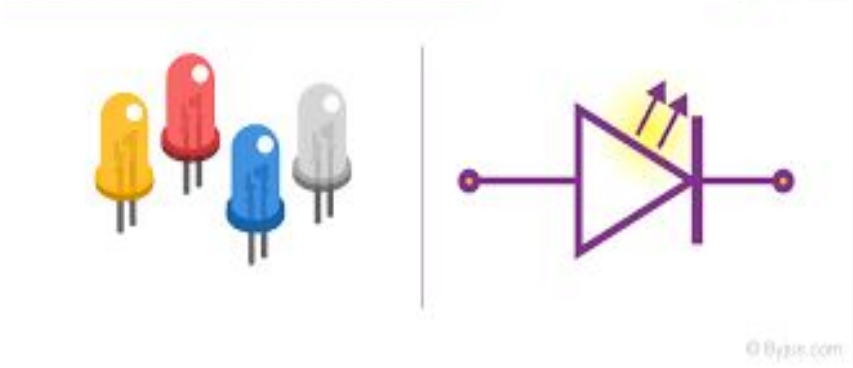
- Produce a bright image
- Energy efficient
- Completely flat screen

□ **Disadvantages:**

- Fixed aspect ratio & Resolution
- Lower Contrast
- More Expensive

6.Light Emitting Diode (LED)

- LED is output device in Embedded System.
- LED is a device which light emits when current passes through it.
- It is a semiconductor device.
- The size of the LED is small, so we can easily make any display unit by arranging a large number of LEDs.
- LED consumes more power compared to LCD. LED is used on TV, smartphones, motor vehicles, traffic light, etc.
- LED also works at high temperatures.



□ **Advantages:**

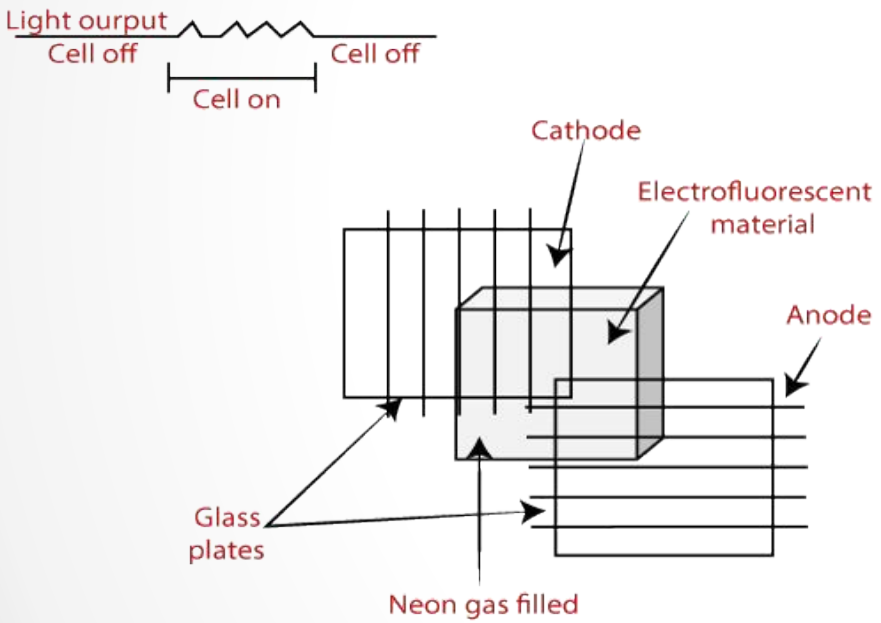
- The Intensity of light can be controlled.
- Low operational Voltage.
- Capable of handling the high temperature.

□ **Disadvantages:**

- More Power Consuming than LCD.

7.Plasma Display Panel(PDP)

- It is a type of flat panel display which uses tiny plasma cells,ionized gas that gas respond to electric field.
- Plasma Panel are made up of glass plates filled with a mixture of gases like neon.It is also known as **the Gas-Discharge display**.



□**Advantages:**

- Wall Mounted
- Slim
- Wider angle

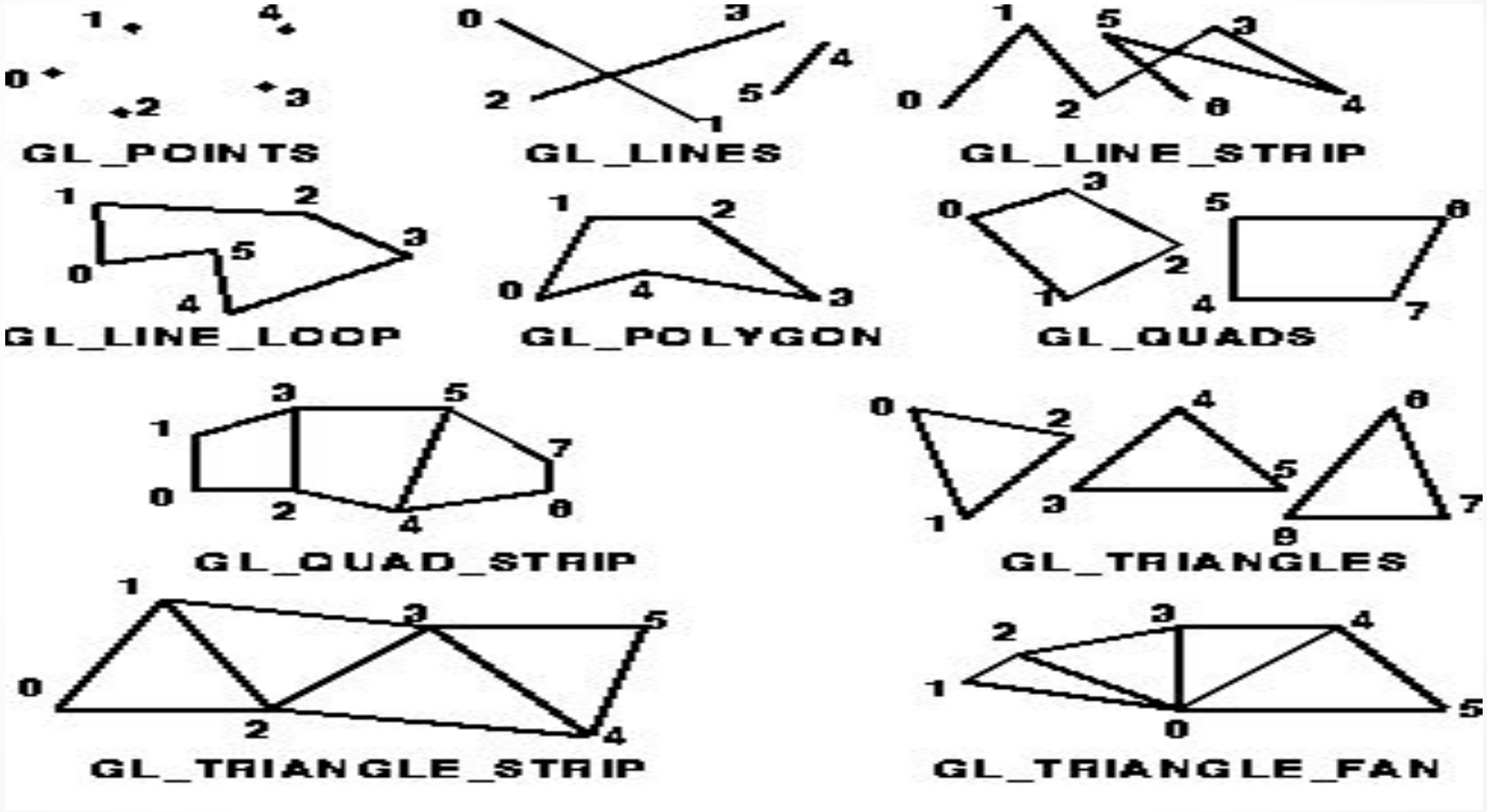
□ **Disadvantages:**

- Lower Brightness
- It consumes more electricity than LCD.
- screens are delicate.

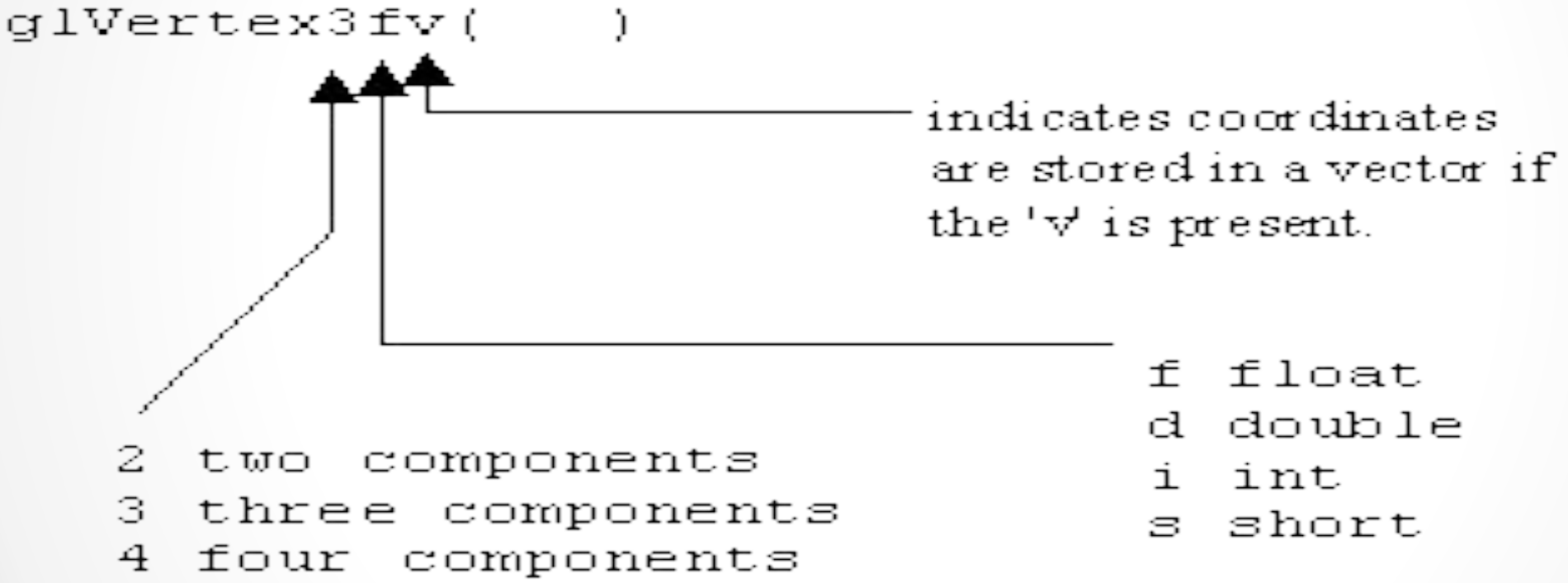
OpenGL(Open Graphics Library)

- OpenGL is a software Interface that allows the programmer to create 2D & 3D graphics.
- Cross Language- we can implement in any language.
- support client server protocol
- Open source
- Platform independence
- device independence
- openGL rendering commands however are the primitives. You can use commands(Functions) in the program to draw the points, lines & polygons and you have to build more complex entities upon these.
- Abstractions like GL(glColor,glVertex,glRotate), GLU(OpenGL Utility Library), GLUT(OpenGL Utility Toolkit)
- Softwares-CodeBlocks,Visual Studio,Dev C++ etc.

OpenGL Drawing Primitives-

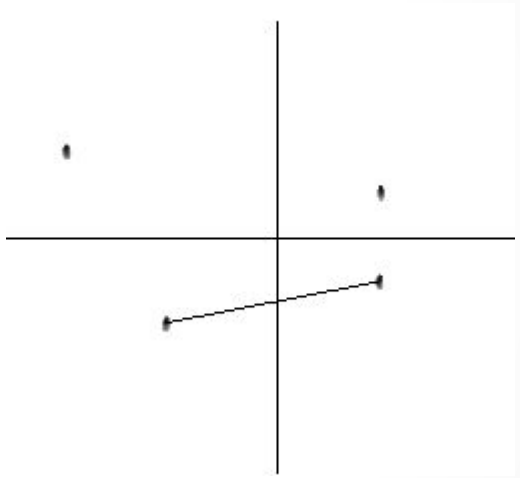


- Function names in the OpenGL basic library(OpenGL Core Library)are prefixed with gl exa.glVertex,glClear,glColor,glRotate,glEnd
- All constants begin with uppercase letters GL_POLYGON,GL_POINTS,
- Various forms of of glVertex() function calls are

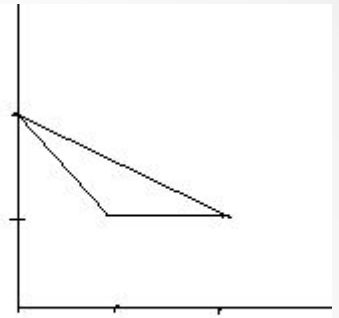


Drawing Primitives Shapes-

```
glBegin(GL_POINTS);     //Drawing Points
    glVertex2f(-3.0,2.0);
    glVertex2f(2.0,1.0);
glEnd();

glBegin(GL_LINES);      //Drawing Lines
    glVertex2i(-2,-2);
    glVertex2i(2,-1);
glEnd();
```



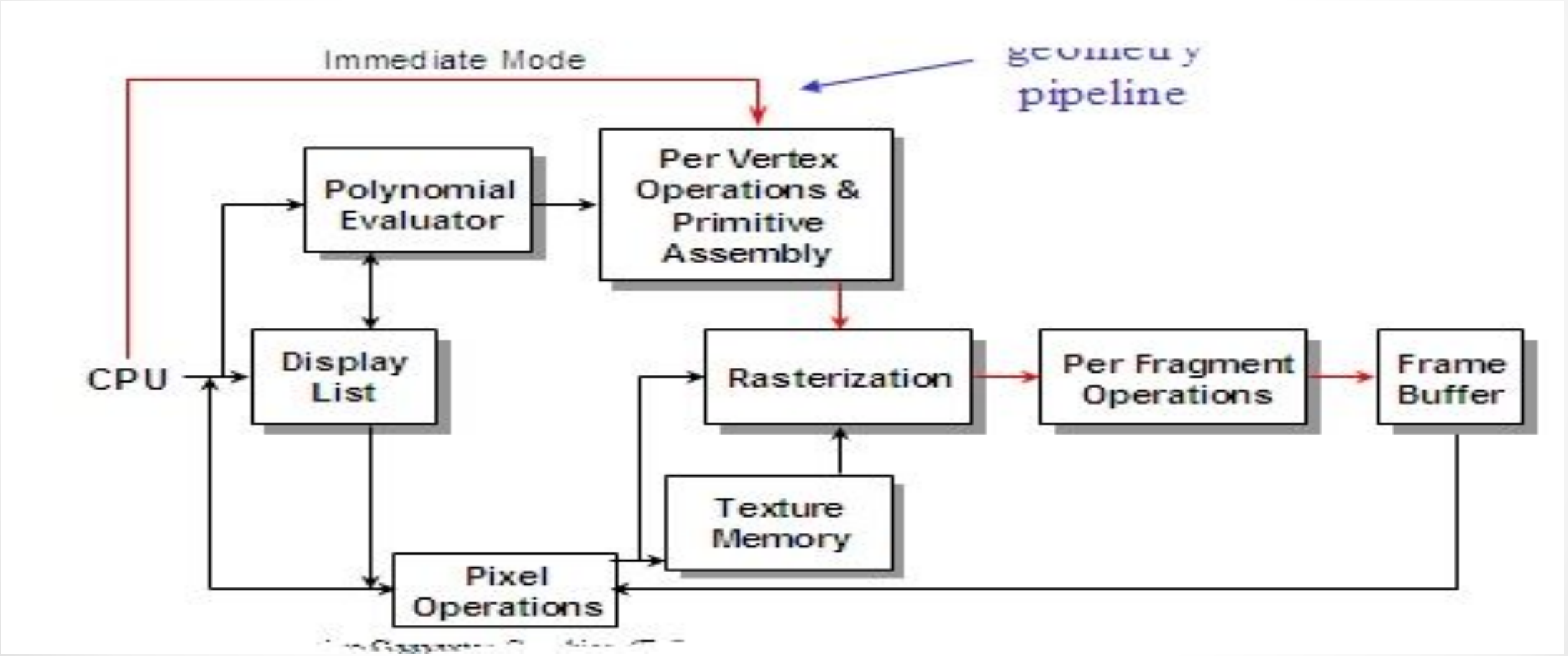
```
glBegin(GL_TRIANGLES);     //Drawing Triangle
    glVertex2i(2,1);
    glVertex2i(1,1);
    glVertex2i(0,2);
glEnd();
```



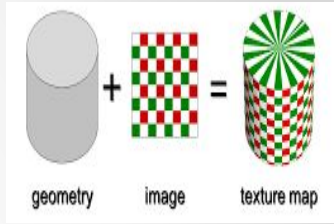
```
GLfloat pt[2]=[3.0,4.0];
glBegin(GL_POINTS);     //Draw several isolated points in 2D
    glVertex2f(1.0,2.0);
    glVertex2f(2.0,3.0);
    glVertex2fv(pt);
    glVertex2i(4,5);
glEnd();
```

```
GLfloat p1[3]=[0,0,1];
GLfloat p2[3]=[1,0,1];
GLfloat p3[3]=[1,1,1];
GLfloat p4[3]=[0,1,1];
glBegin(GL_POLYGON);                    //Draw 3D Ploygon
    glVertex3fv(p1);
    glVertex3fv(p2);
    glVertex3fv(p3);
    glVertex3fv(p4);
glEnd();
```

OpenGL Architecture



- **Display List**-List can accumulate some commands in a display list for processing later(make priority list) or can proceed immediately through pipeline
- **Pixel Operation**- Input data can be in the form of pixels (image for texture mapping) is processed in the pixel operational stage.
- **Polynomial Evaluator**- provides an efficient means for approximating curve & surface geometry by evaluating polynomial commands of input values.
- **Primitives Assembly**-Process geometric primitives points,line,polygon.
- **Rasterization**- Produces a series of frame buffer addresses & associated values using 2D description of a point,line etc
- **Texture Memory**- The images & geometry pass through separate pipeline & join at rasterizer.
- **Per fragment Operations**-optimize, find error,blending of incoming pixel colors with stored color and other logical operations.
- **Frame Buffer**-Final Result stored(sequence wise image)



GLUT: Interaction with mouse and keyboards

- The **OpenGL Utility Toolkit** (**GLUT**) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system.

- Functions performed include window definition, window control, and monitoring of keyboard and mouse input.
- **The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross-platform) and to make learning OpenGL easier.**

GLUT_KEY_UP	Up Arrow
GLUT_KEY_RIGHT	Right Arrow
GLUT_KEY_DOWN	Down Arrow
GLUT_KEY_PAGE_UP	Page Up
GLUT_KEY_PAGE_DOWN	Page Down
GLUT_KEY_HOME	Home
GLUT_KEY_END	End
GLUT_KEY_INSERT	Insert

Scan conversion:

The Process of representing continuous graphics object as a collection of discrete pixel called scan conversion.

Line drawing algorithms:

1. Digital Differential Analyzer (DDA)

1. Bresenham Line Drawing Algorithm

1. DDA Line Drawing Algorithm

- Line is a basic element in graphics.
- To draw a line, you need two end points between which you can draw a line.
- Digital Differential Analyzer (DDA) line drawing algorithm is the simplest line drawing algorithm in computer graphics.
- It works on incremental method.
- It plots the points from starting point of line to end point of line by incrementing in X and Y direction in each iteration.
- DDA line drawing algorithm works as follows:

DDA Algorithm-

- Step 1: Get coordinates of both the end points (X1, Y1) and (X2, Y2) from user.
- Step 2: Calculate the difference between two end points in X and Y

direction. dx = X2 – X1;

dy = Y2 – Y1;

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

```
if (absolute(dx) >
absolute(dy))       Steps =
absolute(dx);

else

Steps = absolute(dy);
```

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

```
Xincrement = dx /  steps;

Yincrement = dy / steps;
```

- Step 5: Plot the pixels by successfully incrementing x and y coordinates accordingly and complete the drawing of the line.

```
for(int i=0; i <= Steps; i++)

{

putpixel(Round(X1), Round(Y1), ColorName);

X1 = X1 +
Xincrement;

Y1 = Y1 +
```

1) **Draw a line from(2,2) to (9,2) using DDA Algorithm.**

(x1,y1)(x2,y2)

(2,2)(9,2)

dx=9-2=7

dy=2-2=0

steps=7

xincrement=7/7=1

yincrement=0/7=0

x	y
2	2
3	2
4	2
5	2
6	2
7	2
8	2
9	2

Horizontal Line

2)**Draw a line from(2,5) to (2,12) using DDA Algorithm.**

(x1,y1)(x2,y2)

(2,5)(2,12)

dx=2-2=0

dy=12-5=7

steps=7

xincrement=0/7=0

yincrement=7/7=1

x	y
2	5
2	6
2	7
2	8
2	9
2	10
2	11
2	12

Vertical Line

3)**Draw a line from(5,4) to (12,7) using DDA Algorithm.**

(x1,y1)(x2,y2)

(5,4)(12,7)

dx=12-5=7

dy=7-4=3

steps=7

xincrement=7/7=1

yincrement=3/7=0.4

x	y	Roundof of y
5	4	
6	4.4	4
7	4.8	5
8	5.2	5
9	5.6	6
10	6	6
11	6.4	6
12	6.8	7

Line having slop m<1

4)Draw a line from(5,7) to (10,15) using DDA Algorithm.

(x1,y1)(x2,y2)

(5,7)(10,15)

Line having slop m>1

5)Draw a line from(12,9) to (17,14) using DDA Algorithm

(12,9)(17,14)

Line Having Slop m=1

DDA Algorithm

Advantages-

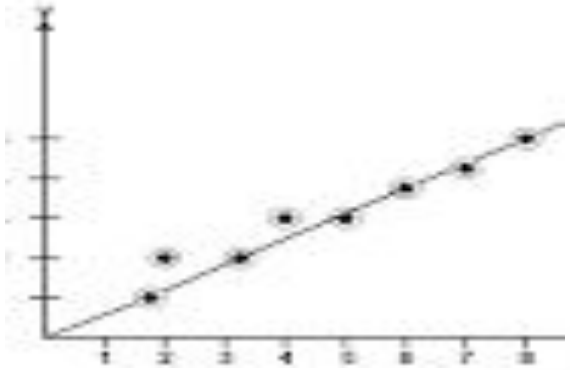
1. It is simple.
2. It is easy to understand.
3. It is not require any special skill to implement it.

Disadvantages-

1. Line will be not smooth line.
2. It involve floating point operation for each pixel.
3. It perform rounding off operation for each pixel.
4. Algorithm require extra time so algorithm becomes slower.

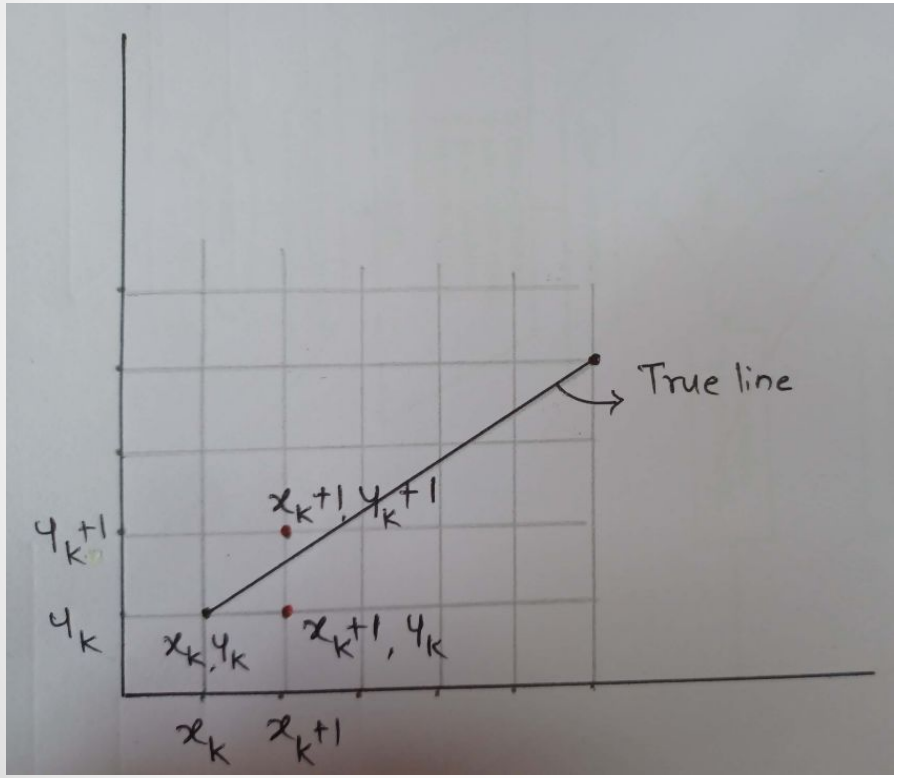
Bresenham's Line Algorithm

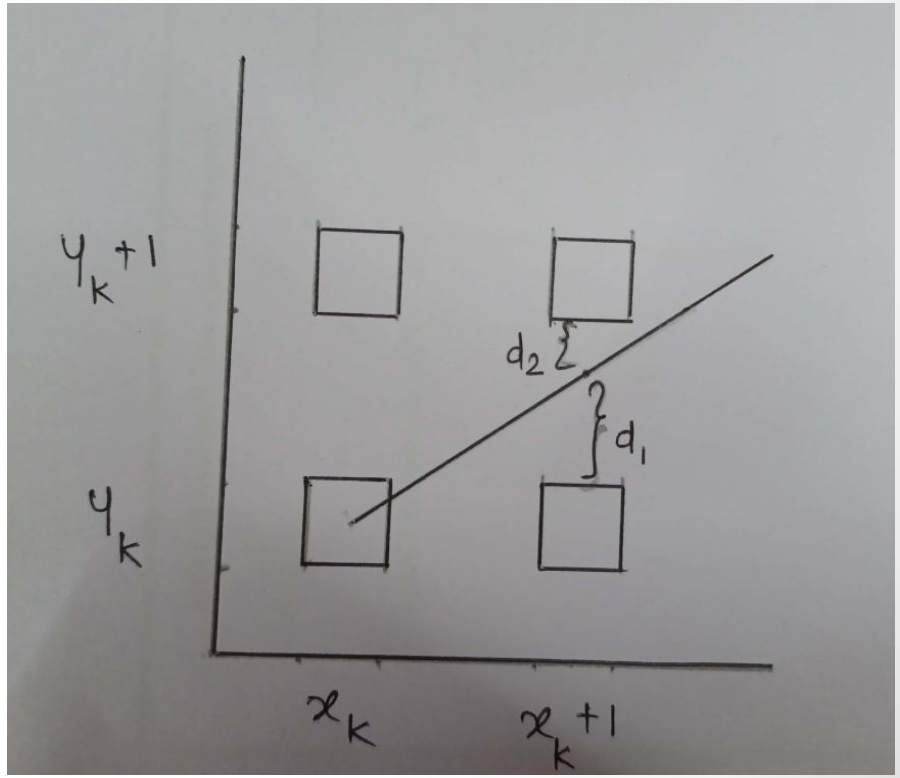
- This algorithm is used for scan conversion of a line.
- It was developed by Bresenham.
- It is an **efficient** method because it involves only integer addition, subtractions, and multiplication operations.
- These operations can be performed very **rapidly** so lines can be generated quickly.
- In this method, next pixel selected is that one who has the least distance from true line.
- It determines the points that should be selected in order to form close approximation to a straight line between two points.



Bresenham's Line Algorithm..

- From Fig 1 After displaying the first point (xk,yk) we have to select next point. there are 2 candidate pixel(xk+1,yk)(xk+1,yk+1). Out of these we have to select one pixel.
- From Fig 2 we can see d2 distance is lesser than d1 so (xk+1,yk+1) pixel selected.





Bresenham's Line Algorithm-

step 1- Read two endpoints (x1,y1) and (x2,y2).

step 2- Plot the first pixel (x1,y1)

step 3- Calculate Δx and Δy

Δx=x2-x1 Δy=y2-y1

step 4-Find Decision parameter pk(It is used to find exact point to draw line) pk=2Δy-Δx

step 5-Suppose current point(xk,yk) find next point depending on value find next point depending on value of decision parameter pk.

case 1- if pk<0

pk+1=pk + 2Δy

xk+1=xk +1

yk+1=yk

case 2- if pk >= 0

pk+1=pk+2Δy-2Δx

xk+1=xk+1

yk+1=yk+1

step 6-repeat 5 until end point is reached.

Examples

Exa.1 Consider the line from (20,10) to (30,18). Use Bresenham's line drawing algorithm to rasterize this line.

step 1- Read two endpoints (20,10)and (30,18)as a (x1,y1) and (x2,y2)

step 2- Plot the first pixel (20,10)

step 3- Calculate Δx and Δy

Δx=x2-x1=30-20=10 Δy=y2-y1=18-10=8

step 4-Find Decision parameter pk pk=2Δy-Δx=2(8)-10=16-10=6

step 5-

Examples..

		20	10
	pk	**xk**	**yk**
6>0	pk+1 = 6+2*8-2*10=6+16-20= 2	21	11
2>0	pk+1 = 2+2*8-2*10=2+16-20= -2	22	12
-2<0	pk+1 = -2+2*8 = -2+16 = 14	23	12
14>0	pk+1 = 14+2*8 -2*10= 14+16-20 = 10	24	13
10>0	pk+1 = 10+2*8 -2*10= 10+16-20 = 6	25	14
6>0	pk+1 = 6+2*8 -2*10= 6+16-20 = 2	26	15
2>0	pk+1 = 2+2*8 -2*10= 2+16-20 = -2	27	16
-2<0	pk+1 = -2+2*8 = -2+16 = 14	28	16
14>0	pk+1 = 14+2*8 -2*10= 14+16-20 = 10	29	17
10>0	pk+1 = 10+2*8 -2*10= 10+16-20 = 6	30	18

Examples

Exa.2 Consider the line from (1,1) to (6,4). Use Bresenham's line drawing algorithm to rasterize this line.

step 1- Read two endpoints (1,1)and (6,4)as a (x1,y1) and (x2,y2)

step 2- Plot the first pixel (1,1)

step 3- Calculate Δx and Δy

$\Delta x=x2-x1=6-1=5$ $\Delta y=y2-y1=4-1=3$

step 4-Find Decision parameter pk $pk=2\Delta y-\Delta x=2(3)-5=6-5=1$

step 5-

Examples..

		1	1
	pk	**xk**	**yk**
1>0	pk+1 = 1+2*3-2*5=1+6-10= -3	2	2
-3<0	pk+1 = -3+2*3=-3+6= 3	3	2
3>0	pk+1 = 3+2*3-2*5 = 3+6-10 = -1	4	3
-1<0	pk+1 = -1+2*3 = -1+6 = 5	5	3
5>0	pk+1 = 5+2*3 -2*5= 5+6-10 = 1	6	4

Examples

Exa.3 Consider the line from (0,0) to (6,6). Draw a line using Bresenham's line drawing algorithm.

step 1- Read two endpoints (0,0)and (6,6)as a (x1,y1) and (x2,y2)

step 2- Plot the first pixel (0,0)

step 3- Calculate Δx and Δy

Δx=x2-x1=6-0=6 Δy=y2-y1=6-0=6

step 4-Find Decision parameter pk pk=2Δy-Δx=2(6)-6=12-6=6

step 5-

Examples..

		0	0
	pk	**xk**	**yk**
6>0	pk+1 = 6+2*6-2*6=6+12-12= 6	1	1
6>0	pk+1 = 6+2*6-2*6=6+12-12= 6	2	2
6>0	pk+1 = 6+2*6-2*6=6+12-12= 6	3	3
6>0	pk+1 = 6+2*6-2*6=6+12-12= 6	4	4
6>0	pk+1 = 6+2*6-2*6=6+12-12= 6	5	5
6>0	pk+1 = 6+2*6-2*6=6+12-12= 6	6	6

Examples

Exa.4 Consider the line from (2,5) to (8,8). Draw a line using Bresenham's line drawing algorithm.

step 1- Read two endpoints (2,5)and (8,8)as a (x1,y1) and (x2,y2)

step 2- Plot the first pixel (2,5)

step 3- Calculate Δx and Δy

$\Delta x=x2-x1=8-2=6$ $\Delta y=y2-y1=8-5=3$

step 4-Find Decision parameter pk $pk=2\Delta y-\Delta x=2(3)-6=6-6=0$

step 5-

Examples..

		2	5
pk		**xk**	**yk**
0>=0	pk+1 = 0+2*3-2*6=0+6-12= -6	3	6
-6<0	pk+1 = -6+2*3 =-6+6= 0	4	6
0>=0	pk+1 = 0+2*3-2*6=0+6-12= -6	5	7
-6<0	pk+1 = -6+2*3 =-6+6= 0	6	7
0>=0	pk+1 = 0+2*3-2*6=0+6-12= -6	7	8
-6<0	pk+1 = -6+2*3 =-6+6= 0	8	8

Examples

Exa.5 Write and explain Bresenham's line drawing algorithm and find out which pixel would be turn on for the line with endpoints (3,2) to (7,4) using the same.

step 1- Read two endpoints (3,2)and (7,4)as a (x1,y1) and (x2,y2)

step 2- Plot the first pixel (3,2)

step 3- Calculate Δx and Δy

$\Delta x=x2-x1=7-3=4$ $\Delta y=y2-y1=4-2=2$

step 4-Find Decision parameter pk $pk=2\Delta y-\Delta x=2(2)-4=4-4=0$

step 5-

Examples..

		3	2
	pk	**xk**	**yk**
0>=0	pk+1 = 0+2*2-2*4=0+4-8= -4	4	3
-4<0	pk+1 = -4+2*2 =-4+4= 0	5	3
0>=0	pk+1 = 0+2*2-2*4=0+4-8= -4	6	4
-4<0	pk+1 = -4+2*2 =-4+4= 0	7	4

Differentiate between DDA Algorithm and Bresenham's Line Algorithm

Sr No	DDA Line Drawing Algorithm	Bresenham's Line Drawing Algorithm
1	DDA Algorithm use floating point, i.e., Real Arithmetic.	Bresenham's Line Algorithm use fixed point, i.e., Integer Arithmetic
2	DDA Algorithms uses multiplication & division its operation	Bresenham's Line Algorithm uses only subtraction and addition its operation
3	DDA Algorithm is slowly than Bresenham's Line Algorithm in line drawing because it uses real arithmetic (Floating Point operation)	Bresenham's Algorithm is faster than DDA Algorithm in line because it involves only addition & subtraction in its calculation and uses only integer arithmetic.
4	To display pixel we need to use floor and ceil operation for round up.	No need to use floor and ceil operation.
5	DDA Algorithm is not accurate and efficient as Bresenham's Line Algorithm	Bresenham's Line Algorithm is more accurate and efficient at DDA Algorithm.

Circle Drawing Algorithm

1. DDA Circle Drawing Algorithm

1. Bresenham's Circle Drawing Algorithm

1. Mid point Circle drawing Algorithm

1.DDA Circle Drawing Algorithm-

- This algorithm uses the incremental method to draw **each point.**
- The algorithm calculate starting point & by the value of ϵ the value of x & y coordinate will be increases & decreases respectively.
- The equation of circle $x^2 + y^2 = r^2$ is use to draw circle.
- To calculate the value of ϵ we have formula

  $2^{n-1} <= r < 2^n$ where r=radius of circle
- $\epsilon = 2^{-n} = 1/2^n$

Exa. if r=50

$2^5 <= 50 < 2^6$

2^1 2^2 2^3 2^4 2^5 2^6

2 4 8 16 32 64

- $\epsilon = 2^{-6} = 1/2^6 = 1/64 = 0.0156$
- ϵ is use to calculate the value of circle point which help to draw circle as

  $x_2 = x_1 + \epsilon y_1$ for x coordinate

  $y_2 = y_1 - \epsilon x_2$ for y coordinate

1.DDA Circle Drawing Algorithm..

step 1- Read the radius(r) of the circle and calculate value of ϵ.

step 2- start_x=0 //Initially starting point

start_y=r //setting the radius

step 3- x_1=start_x //setting initial value of x_1 & y_1

y_1=start_y

step 4- do

{

$x_2 = x_1 + \epsilon\, y_1$ //x_1 represent x_n

$y_2 = y_1 - \epsilon\, x_2$ //x_2 represent x_{n+1}

plot(int(x_2),int(y_2))

$x_1 = x_2$ //swapping points to calculate next closest pixel

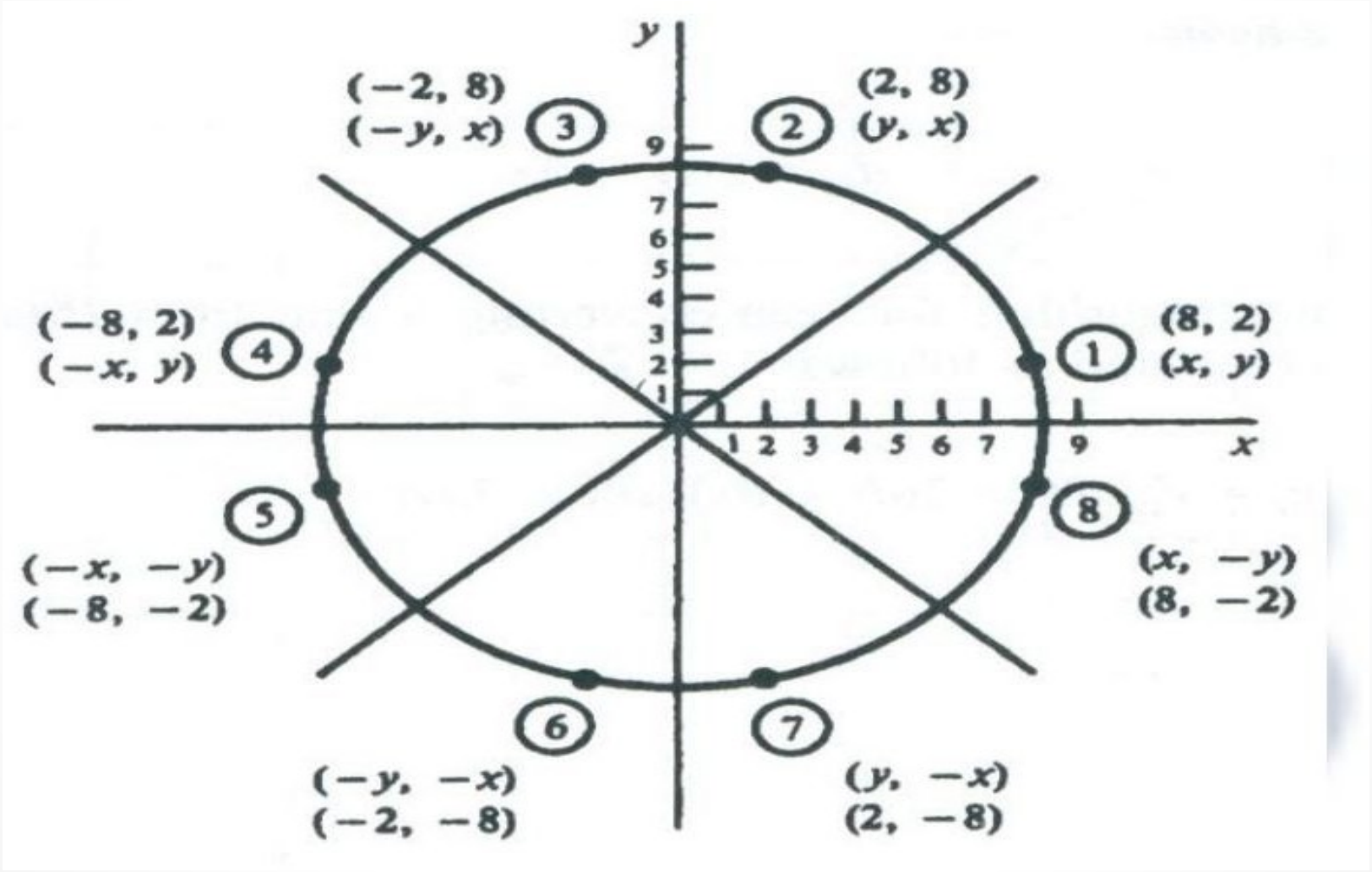
$y_1 = y_2$ //Initialise current point

}while((y_1 - start_y)$<\epsilon$ or (start_x - x_1)$>\epsilon$)

//while condition is to check the current point is starting point or not if current point is not starting point then repeat step 4

step 5- stop

8 way symmetry of a circle



8 way symmetry of a circle…

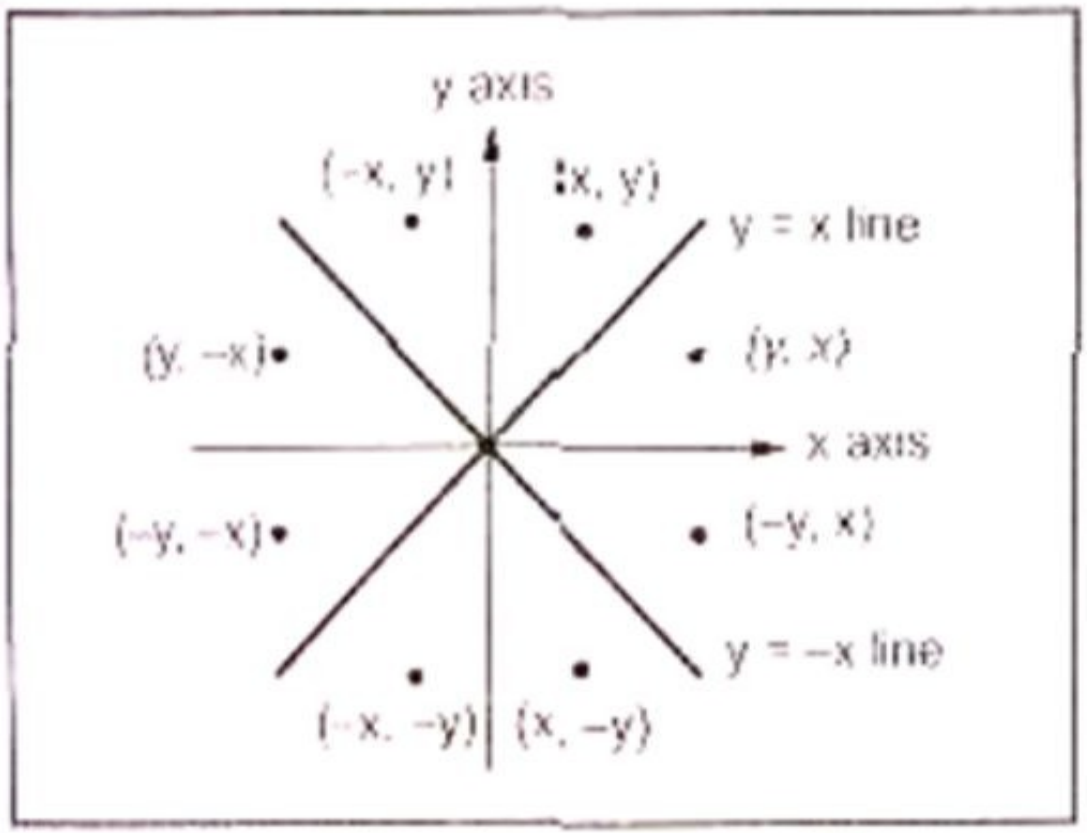


Fig. 1.29 Eight-way symmetry of the circle

plot (y, x)

plot (y, – x)

plot (x, – y)

plot (– x, – y)

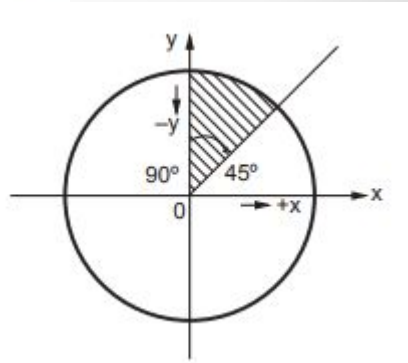
plot (– y, – x)

plot (– y, x) and

plot (– x, y)

2.Bresenham's Circle Drawing algorithm

- The bresenham's circle drawing algorithm considers the eight-way symmetry of the circle to generate it.
- It plots 1/8th part of circle, i.e. from 90° to 45° as shown in figure.
- As circle is drawn from 90° to 45° the x moves in positive direction and y moves in negative direction.



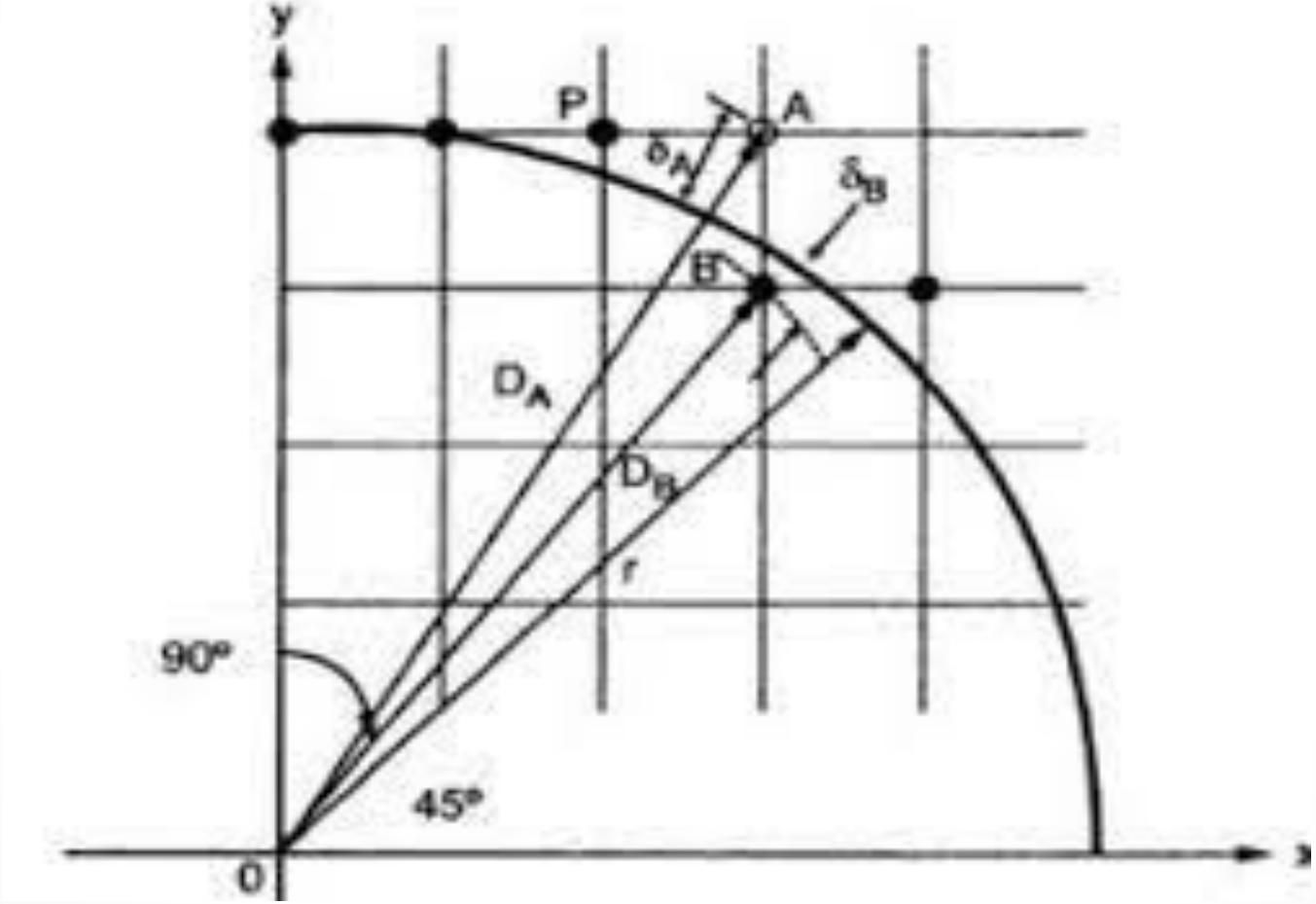
2.Bresenham's algorithm…

- The bresenham's circle drawing algorithm considers the eight-way symmetry of the circle to generate it.
- It plots $1/8^{th}$ part of circle, i.e. from 90° to 45° as shown in figure.
- As circle is drawn from 90° to 45° the x moves in positive direction and y moves in negative direction.

2.Bresenham's algorithm…

- To achieve best approximation to the true circle we have to select those pixels in the raster that fall the least distance from the true circle can be found by applying either of the two options:
- Increment in positive x direction by one unit or remain y as it is
- Increment in positive x direction and negative y direction by one unit.

2.Bresenham's algorithm…



Mathematical expression

- Let us assume point P (x,y)as a last scan converted pixel. Now we have two options either to choose pixel A or pixel B. The closer pixel amongst these two can be determined as follows:
- $fcircle(x,y)=x^2+y^2-r^2$
- This function determines a circle with radius r around the origin in the following way

Mathematical expression

- For point A the equation of circle will be
- $f(A)=f(x_n+1,y_n)$

  $= (x_n+1)^2+(y_n)^2-r^2$

- For point B the equation of circle will be

  $f(B)=f(x_n+1,y_n-1)$

  $= (x_n+1)^2+(y_n-1)^2-r^2$

- We are going to choose the point A or point B which is nearer to expected circle.

Mathematical expression

- So decision variable d is calculated by making the sum of function.
- d =f(A)+f(B) f(A) is always positive
  and f(B) is negative
- If d variable will contain either -ve value or 0 then A point is closer to circle

  Point A when d<=0
- If d variable will contain either +ve value then B point is closer to circle

  Point B when d>0

Mathematical expression

- $d = f(A) + f(B)$

  $= (x_n + 1)^2 + (y_n)^2 - r^2 + (x_n + 1)^2 + (y_n - 1)^2 - r^2$

  $d = 2x_n^2 + 4x_n + 3 + 2y_n^2 - y_n - 2r^2$

- Put initial value (x0,y0) is (0,r)
- $f(A) = f(x_n + 1, y_n)$

  $= (x_o + 1)^2 + (y_o)^2 - r^2$

  $= (0 + 1)^2 + (r)^2 - r^2$

  $= 1$

Mathematical expression

- $f(B)=f(x_n+1,y_n-1)$

$= (x_0+1)^2+(y_0-1)^2-r^2$

$=(0+1)^2+(r-1)^2-r^2$

$=2-2r$

$d= 1+2-2r$

$\boxed{d=3-2r}$

Mathematical expression

- Case1: d <= 0
- Previous point is A either select C or D which is nearer
- $f(C)=f(x_n+2,y_n)$
  $= (x_n+2)^2+(y_n)^2-r^2$
- $f(D)=f(x_n+2,y_n-1)$
  $= (x_n+2)^2+(y_n-1)^2-r^2$
- $d = f(C)+f(D)$
  $= (x_n+2)^2+(y_n)^2-r^2 +(x_n+2)^2+(y_n-1)^2-r^2$
  $= 2x_n^2+4x_n+3+ 2y_n^2-y_n-2r^2+4x_n+6$
- **$d=d+4x_n+6$**

Mathematical expression

- Case2: d >0
- Previous point is B Next point will be either D or E.
- $f(D)=f(x_n+2, y_n-1)$

  $= (x_n+2)^2+(y_n-1)^2-r^2$

- $f(E)=f(x_n+2,y_n-2)$

  $= (x_n+2)^2+(y_n-2)^2-r^2$

- $d = f(D)+f(E)$

  $= (x_n+2)^2+(y_n-1)^2-r^2 +(x_n+2)^2+(y_n-2)^2-r^2$

$d=d+4(x_n-y_n)+10$

Mathematical expression

- For $d <= 0$ $\mathbf{d=d+4x_n+6}$ and
- For $d > 0$ $\mathbf{d=d+4(x_n-y_n)+10}$

2.Bresenham's Algorithm-

step 1- Read the radius(r) of the circle.

step 2- x=0

y=r

step 3- d=3 - 2r //calculate initial decision parameter

step 4- Repeat till x<=y

case 1- if d <= 0

d=d+4x+6

x=x +1

y=y

case 2- if d> 0

d=d+4(x-y)+10

x=x+1

y=y-1

step 5- plot (x,y)

step 6- Determine and plot the symmetric point in other octant as well.

step 7- stop

Examples

Exa.1 Calculate the pixel position along the circle path with radius r=7 centered on the origin using Bresenham's circle drawing algorithm from point (0,7) to point x=y.

step 1- radius(r) of the circle r=7

step 2- x=0 y=7

step 3- d=3 - 2r= 3- 2(7)=3-14= -11

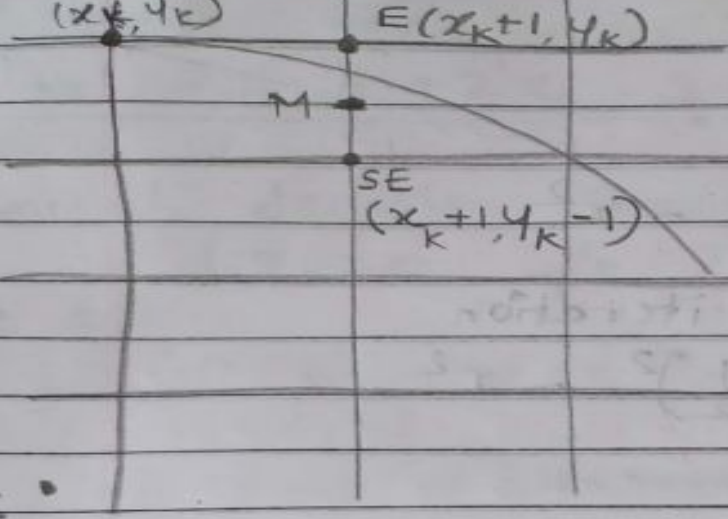
Step	Decision Variable		x	y
1	-11<0	d=d+4x+6= -11+4(0)+6 = -11+6= -5	1	7
2	-5<0	d=d+4x+6= -5+4(1)+6 = -5+4+6 = 5	2	7
3	5>0	d=d+4(x-y)+10= 5+4(2-7)+10 = 5+4(-5)+10 = 15-20=-5	3	6
4	-5<0	d=d+4x+6= -5+4(3)+6 = -5+12+6 = 13	4	6
5	5>0	d=d+4(x-y)+10= 13+4(4-6)+10 = 5+4(-2)+10 = 15-8=7	5	5

3. Mid Point Circle Drawing Algorithm-

- The Mid-point circle drawing algorithm considers the eight-way symmetry of the circle to generate it.
- It plots $1/8^{th}$ part of circle, i.e. from 90° to 45° as shown in figure.
- As circle is drawn from 90° to 45° the x moves in positive direction and y moves in negative direction.

Mathematical expression

- Let us assume point P (x,y)as a last scan converted pixel. Now we have two options either to choose pixel A or pixel B. The closer pixel amongst these two can be determined as follows:
- $fcircle(x,y)=x^2+(y-½)^2-r^2$



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

$x^2 + y^2 - r^2 = 0$

(x', y')

Result $\{(x')^2 + (y')^2 - r^2\}$

→ 0 → Point lies ON circle

→ <0 → Lies INSIDE circle boundary

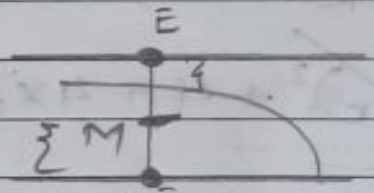
→ >0 → Lies OUTSIDE +11−

+11−

< 0

$(x')^2 + (y')^2 - r^2 < 0$

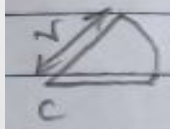
$< r^2$

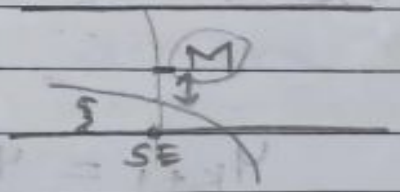


(E) Near selected

(SE) far

> 0 $(x')^2 + (y')^2 - r^2 > 0$





(E) far

(SE) Next selected

Mid-point Circle Algorithm:

- (x_k, y_k) is plotted
- Next pixel position is either(x_k+1, y_k) or (x_k+1, y_k-1)?
- Decision Parameter
  - p_k : $f(x_k+1, y_k)$ or
    : $f(x_k+1, y_k-1)$
- Midpoint of these two points:
  - $$p_k = f(x_k + 1, y_k - ½)$$
    $$= (x_k + 1)^2 + (y_k - ½)^2 - r^2$$

Mid-point Circle Algorithm:

- Our decision parameter is the earlier circle function evaluated at the mid point between the 2 pixels

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

P_k
- < 0: midpoint is inside the circle; plot (x_k+1, y_k)
- +ve: midpoint is outside the circle; plot (x_k+1, y_k-1)

- Successive decision parameters are obtained using incremental calculation

Mid-point Circle Algorithm:

- Initial decision parameter is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0,r)$

$$p_0 = f(1, r-1/2) = 1^2 + (r - 1/2)^2 - r^2$$

$$p_0 = 5/4 - r$$

PO = 1.25 - r;

3.Mid Point Circle Drawing Algorithm-

step 1- Read the radius(r) of the circle.

step 2- x=0

y=r

step 3- p=1 - r //calculate initial decision parameter

step 4- Repeat till x<=y

case 1- if p < 0

p=p+2x+ 3

x=x +1

y=y

case 2- if p>= 0

p=p+2(x-y)+5

x=x+1

y=y-1

step 5- plot (x,y)

step 6- Determine and plot the symmetric point in other octant as well.

step 7- stop

Examples

Exa.1 Plot the First octant of a circle centered at origin having radius 10 units by using mid point circle drawing algorithm.

step 1- radius(r) of the circle r=10

step 2- x=0 y=10

step 3- p=1 - r= 1- 10= -9

Examples

Step	Decision Variable	x	y
1	-9<0 p=p+2x+3= -9+2(0)+3 = -9+3= -6	1	10
2	-6<0 p=p+2x+3= -6+2(1)+3 = -6+5= -1	2	10
3	-1<0 p=p+2x+3= -1+2(2)+3 = -1+7= 6	3	10
4	6>=0 p=p+2(x-y)+5= 6+2(3-10)+5 =6-14+5=11 -14= -3	4	9
5	-3<0 p=p+2x+3= -3+2(4)+3 = -3+11= 8	5	9
6	8>=0 p=p+2(x-y)+5= 8+2(5-9)+5 =8-8+5=5	6	8
7	5>=0 p=p+2(x-y)+5= 5+2(6-8)+5 =5-4+5=6	7	7

Sample graphics code

```
#include<graphics.h>
#include<conio.h>
int main()
{
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "C:\\TC\\BGI");

  getch();
  closegraph();
  return 0;
}
```

- Let me tell you what the output of this program is, this program initializes graphics mode and then closes it after a key is pressed.

- To begin with we have declared two variables of int type gd and gm for graphics driver and graphics mode respectively, you can choose any other variable name as well.

- DETECT is a macro defined in "graphics.h" header file, then we have passed three arguments to initgraph function first is the address of gd, second is the address of gm and third is the path where your BGI files are present (you have to adjust this accordingly where you Turbo C compiler is installed).

- Initgraph function automatically decides an appropriate graphics driver and mode such that maximum screen resolution is set, getch helps us to wait until a key is pressed, closegraph function closes the graphics mode, and finally return statement returns a value 0 to main indicating successful execution of the program.

- After you have understood initgraph function then you can use functions to draw shapes such as circle, line , rectangle, etc., then you can learn how to change colors and fonts using suitable functions, then you can go for functions such as getimage, putimage, etc., for doing animation.

Draw a line in C++ graphics

Explanation : The header file graphics.h contains line() function which is described below :
Declaration : **void line(int x1, int y1, int x2, int y2);**

ine function is used to draw a line from a point(x1,y1) to point(x2,y2) i.e. (x1,y1) and (x2,y2) are end points of the line.The code given below draws a line.

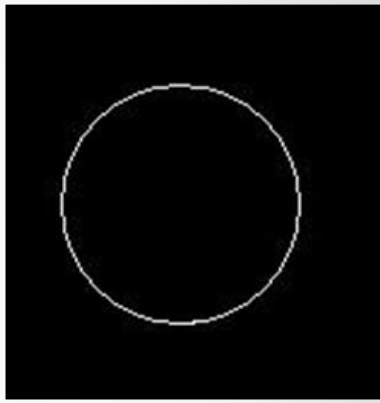
```
// C++ Implementation for drawing line
#include <graphics.h>

// driver code
int main()
{
   // gm is Graphics mode which is a computer display
   // mode that generates image using pixels.
   // DETECT is a macro defined in "graphics.h" header
file

int gd = DETECT, gm;

   // initgraph initializes the graphics system
   // by loading a graphics driver from disk
   initgraph(&gd, &gm, “c://TC/BGI ");

   // line for x1, y1, x2, y2
   line(150, 150, 450, 150);

 // line for x1, y1, x2, y2
   line(150, 200, 450, 200);

// line for x1, y1, x2, y2
   line(150, 250, 450, 250);

   getch();


   // by graphics system .
   closegraph();
    return 0;
}
```

Output

Below is the implementation to draw circle :

```
// C++ Implementation for drawing
circle #include <graphics.h>

//driver code
int main()
{
    // gm is Graphics mode which is
    // a computer display mode that
    // generates image using pixels.
    // DETECT is a macro defined in
    // "graphics.h" header file
    int gd = DETECT, gm;

    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
    initgraph(&gd, &gm, "      ");

    // circle function
    circle(250, 200, 50);

    getch();

    // closegraph function closes
the
    // graphics mode and
deallocates
    // all memory allocated by
    // graphics system .
    closegraph();

    return 0;
}
```

Output :



```
// C++ program for drawing a triangle
#include <graphics.h>
#include <iostream>

// Driver code
int main()
{
    // gm is Graphics mode which
    // is a computer display
    // mode that generates
    // image using pixels.
    // DETECT is a macro
    // defined in "graphics.h"
    // header file
    int gd = DETECT, gm;

    // initgraph initializes
    // the graphics system
    // by loading a graphics
    // driver from disk
    initgraph(&gd, &gm, "");

    // Triangle

    // line for x1, y1, x2, y2
    line(150, 150, 450, 150);

    // line for x1, y1, x2, y2
    line(150, 150, 300, 300);

    // line for x1, y1, x2, y2
    line(450, 150, 300, 300);

    // closegraph function closes
    // the graphics mode and
    // deallocates all memory
    // allocated by graphics system
    getch();

    // Close the initialized gdriver
    closegraph();
}
```

Output: