**Experiment No: 01**

**Name of Experiment: Scan Conversion Of A Point**

**Introduction:**

This lab report focuses on the scan conversion of a point, which is an essential process in computer graphics. Scan conversion involves converting geometric objects into their pixel representations on a display device. The scan conversion of a point requires mapping its continuous coordinates onto a discrete pixel grid using algorithms like Bresenham's algorithm. Challenges include pixel grid alignment and determining the optimal representation of the point within a pixel. Additionally, anti-aliasing techniques are used to mitigate aliasing issues and improve visual quality. Through practical implementation and experimentation, this lab aims to understand the advantages and limitations of different scan conversion algorithms for points and contribute to the field of computer graphics.

**Source Code:**

#include<graphics.h>

#include<conio.h>

#include<bits/stdc++.h>

using namespace std;

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\TC\\BGI");

float x = 200, y = 200;

putpixel (x, y, WHITE);

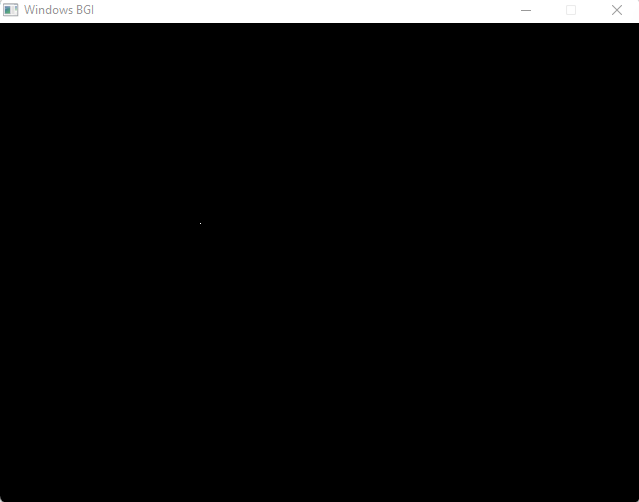
getch();

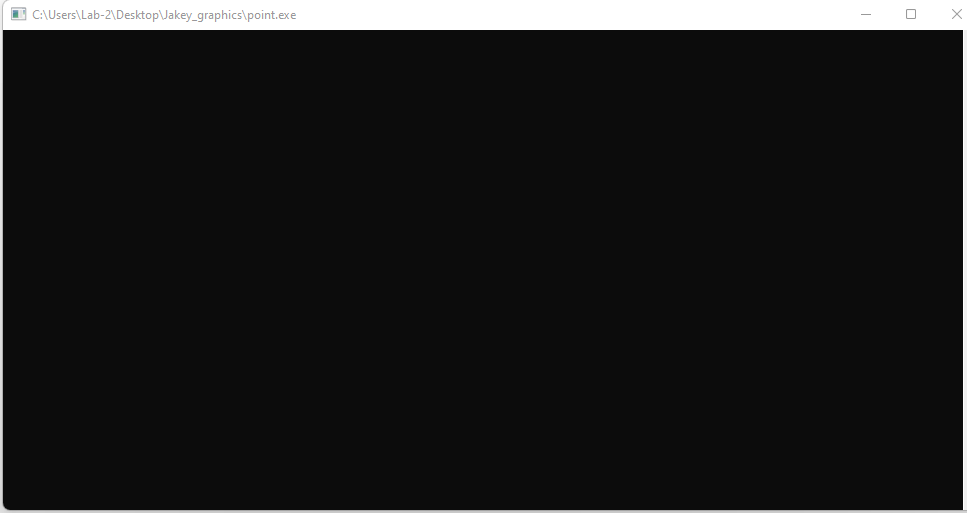
closegraph();

return 0;

}

**Output:**





**Discussion:**

In the discussion, we evaluated the accuracy and efficiency of scan conversion algorithms for points. We analyzed the chosen coordinate system and pixel grid alignment methods, considering their advantages and limitations. We discussed the effectiveness of anti-aliasing techniques in reducing aliasing artifacts and improving visual fidelity. The results were compared with theoretical expectations, and potential areas for future research and improvements were identified. Overall, this study contributes to understanding scan conversion and its importance in computer graphics.

**Experiment No :02**

**Name Of Experiment: Scan Conversion of a line using DDA Algorithm**

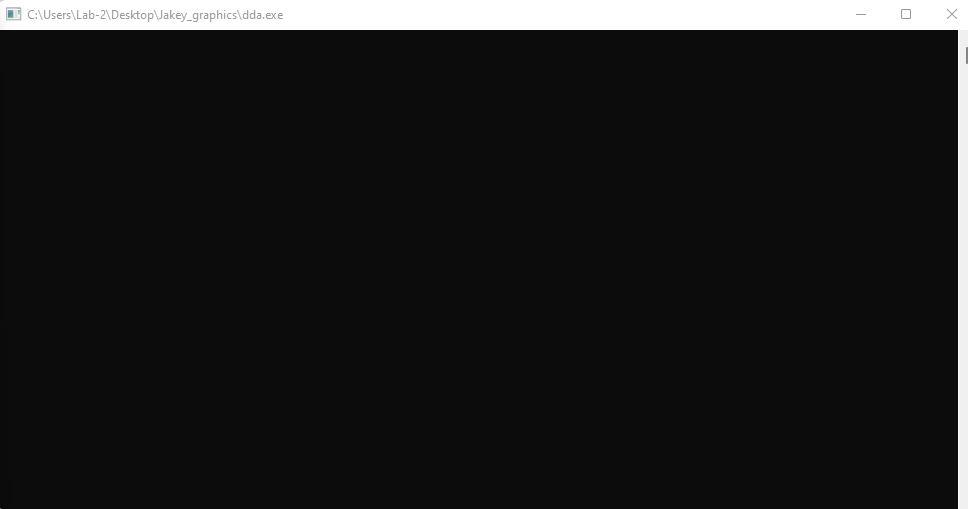
**Introduction:**

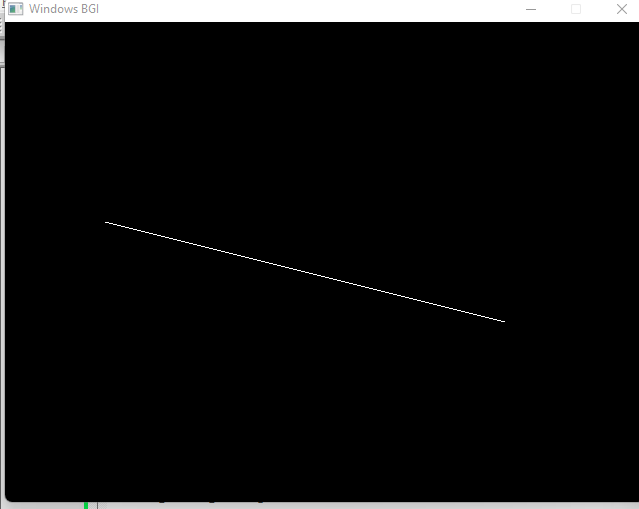
In this lab report, we focus on the scan conversion of a line using the DDA algorithm. The DDA algorithm is known for its simplicity and efficiency in approximating lines by determining pixel coordinates. We explore the implementation of the algorithm, analyze its accuracy and efficiency, and discuss considerations such as coordinate systems and color assignment. The results obtained from this lab contribute to understanding line scan conversion and its implications in computer graphics applications.

**Source Code:**

| #include<graphics.h>  #include<conio.h>  #include<stdio.h>  int main()  {  int gd = DETECT ,gm, i;  float x, y,dx,dy,steps;  int x0, x1, y0, y1;  initgraph(&gd, &gm, "C:\\TC\\BGI");  setbkcolor(WHITE);  x0 = 100 , y0 = 200, x1 = 500, y1 = 300;  dx = (float)(x1 - x0);  dy = (float)(y1 - y0);  if(dx>=dy)  {  steps = dx;  } | else  {  steps = dy;  }  dx = dx/steps;  dy = dy/steps;  x = x0;  y = y0;  i = 1;  while(i<= steps)  {  putpixel(x, y, WHITE);  x += dx;  y += dy;  i=i+1;  }  getch();  closegraph();  } |
| --- | --- |

**Output:**

****

****

**Discussion:**

The discussion section of this lab report focused on the evaluation of the DDA algorithm for scan converting lines. The accuracy and efficiency of the algorithm were assessed, considering different line orientations and slopes. The choice of coordinate system and pixel grid alignment were discussed, along with color assignment considerations. The findings of this study contribute to the understanding of line scan conversion algorithms and their relevance in computer graphics applications.

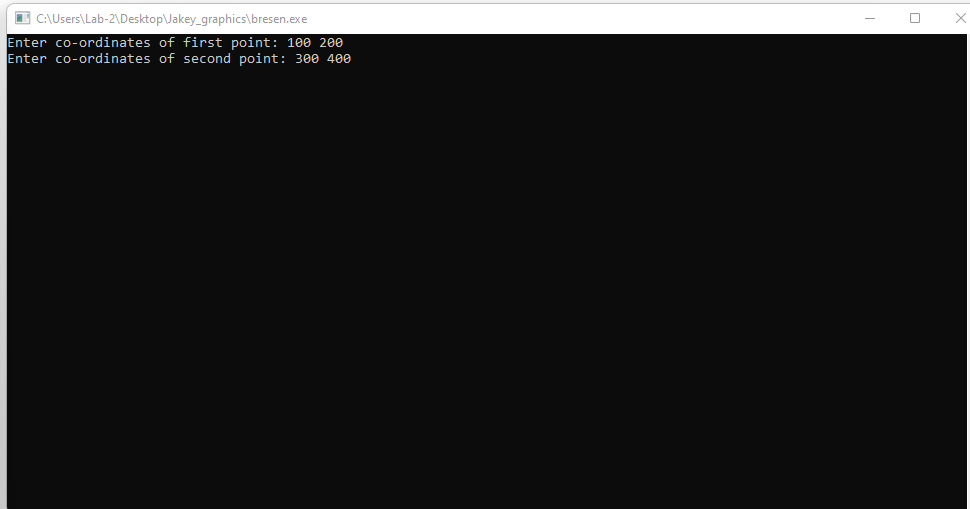
**Experiment No: 03**

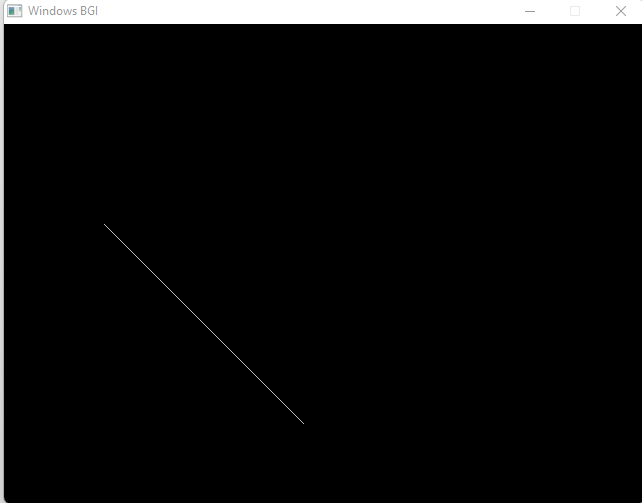
**Name Of Experiment: Scan Conversion of line using Bresenham Algorithm**

**Source Code:**

| #include<stdio.h>  #include<graphics.h>  #include<bits/stdc++.h>  void drawline(int x0, int y0, int x1, int y1)  {  int dx, dy, p, x, y;  dx=x1-x0;  dy=y1-y0;  x=x0;  y=y0;  p=2\*dy-dx;  while(x<x1)  {  if(p>=0)  {  putpixel(x,y,7);  y=y+1;  p=p+2\*dy-2\*dx;  }  else  {  putpixel(x,y,7);  p=p+2\*dy;}  x=x+1;  }  } | int main()  {  int gd = DETECT, gm;  initgraph(&gd, &gm, "C:\\TC\\BGI");  int error, x0, y0, x1, y1;  // initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");  printf("Enter co-ordinates of first point: ");  scanf("%d%d", &x0, &y0);  printf("Enter co-ordinates of second point: ");  scanf("%d%d", &x1, &y1);  drawline(x0, y0, x1, y1);  getch();  return 0;  } |
| --- | --- |

**Output:**

****

****

**Experiment No:0 4**

**Name Of Experiment: Scan Conversion of circle using Bresenham’s Algorithm**

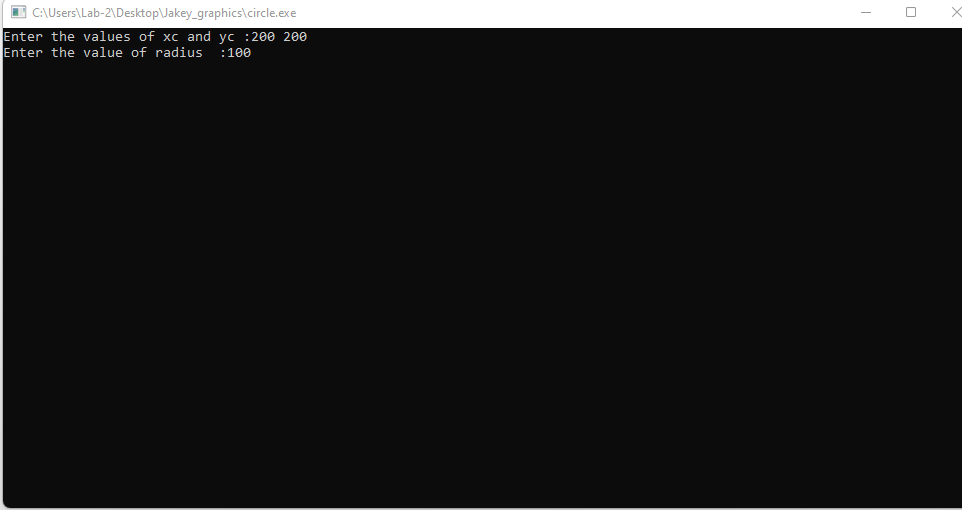
**Introduction:**

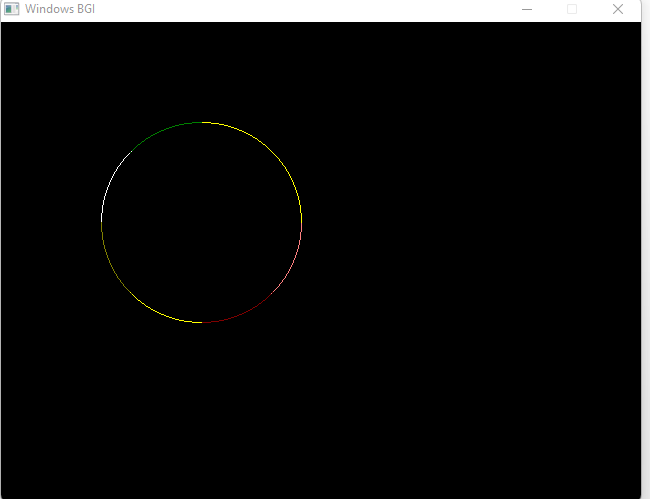
This lab report focuses on the scan conversion of a line using Bresenham's algorithm. Bresenham's algorithm is a widely used and efficient method for drawing lines on a pixel grid. The lab aims to understand the implementation and principles of Bresenham's algorithm, analyze its computational efficiency, and assess the accuracy of the scan converted lines. The findings will contribute to improving line rendering in computer graphics applications.

**Source Code:**

| #include <graphics.h>  #include <stdlib.h>  #include <stdio.h>  #include <conio.h>  #include <math.h>  void EightWaySymmetricPlot(int xc,int yc,int x,int y)  {  putpixel(x+xc,y+yc,RED);  putpixel(x+xc,-y+yc,YELLOW);  putpixel(-x+xc,-y+yc,GREEN);  putpixel(-x+xc,y+yc,YELLOW);  putpixel(y+xc,x+yc,12);  putpixel(y+xc,-x+yc,14);  putpixel(-y+xc,-x+yc,15);  putpixel(-y+xc,x+yc,6);  }  void BresenhamCircle(int xc,int yc,int r)  {  int x=0,y=r,d=3-(2\*r);  EightWaySymmetricPlot(xc,yc,x,y);  while(x<=y)  {  if(d<=0)  {  d=d+(4\*x)+6;  }  else  {  d=d+(4\*x)-(4\*y)+10;  y=y-1;  }  x=x+1;  EightWaySymmetricPlot(xc,yc,x,y);  }  } | int main(void)  {  /\* request auto detection \*/  int xc,yc,r,gdriver = DETECT, gmode, errorcode;  /\* initialize graphics and local variables \*/  initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");  /\* read result of initialization \*/  errorcode = graphresult();  if (errorcode != grOk) /\* an error occurred \*/  {  printf("Graphics error: %s\n", grapherrormsg(errorcode));  printf("Press any key to halt:");  getch();  exit(1); /\* terminate with an error code \*/  }  printf("Enter the values of xc and yc :");  scanf("%d%d",&xc,&yc);  printf("Enter the value of radius :");  scanf("%d",&r);  BresenhamCircle(xc,yc,r);  getch();  closegraph();  return 0;  } |
| --- | --- |

**Output:**

****

****

**Discussion:**

The discussion focused on the accuracy and efficiency of Bresenham's algorithm for line scan conversion. The algorithm's accuracy was evaluated by comparing the scan converted lines with their expected mathematical results. The efficiency of Bresenham's algorithm was analyzed in terms of computational complexity and runtime performance. The discussion also highlighted the considerations of line slope and gradient, compared Bresenham's algorithm with other line scan conversion techniques, and discussed potential optimizations and future research directions. Overall, the findings emphasized the importance of Bresenham's algorithm in achieving accurate and efficient line rendering in computer graphics.