*Course title: Computer Graphics Lab*

*Course code: CSE-304*

*3rd Year 1st Semester Examination 2022*

Date of Submission: 4 June 2023

Submitted to

Dr. Mohammad Shorif Uddin

*Professor*

Dr. Morium Akter

*Associate Professor*

*Department of Computer Science and Engineering Jahangirnagar University*

*Savar, Dhaka-1342*

| Name | Roll | Exam Roll |
| --- | --- | --- |
| Sudipta Singha | 408 | 202220 |

MIdpoint Circle

Introduction:

The Midpoint circle drawing algorithm is a popular method in computer graphics used to scan convert and draw circles on a 2D display. This algorithm efficiently determines which pixels to plot to approximate a circle by utilizing the midpoint between two adjacent pixels. This lab report explores the Midpoint circle drawing algorithm, its implementation in C++, and its effectiveness in drawing circles.

Algorithm:

The Midpoint circle drawing algorithm uses integer arithmetic and a decision parameter to determine the appropriate pixels to plot along the circumference of a circle. It iteratively calculates the midpoint between two adjacent pixels and selects the closest pixel based on the midpoint's position relative to the circle. The following steps outline the Midpoint circle drawing algorithm:

1. Obtain the center coordinates of the circle, (x0, y0), and its radius, r.
2. Set the initial pixel location to (0, r).
3. Calculate the decision parameter: p = 1 - r.
4. Repeat the following steps while the x-coordinate is less than or equal to the y-coordinate:
   * Plot pixels at symmetric positions around the circle's circumference based on the current pixel location.
   * If the decision parameter (p) is less than or equal to 0, update the decision parameter and the x-coordinate.
   * Otherwise, update the decision parameter, the x-coordinate, and the y-coordinate.

Code:

#include<graphics.h>

#include<math.h>

#include<stdio.h>

int main(){

float A=300;

float B=300;

int Gdriver=DETECT,Gmode;

initgraph(&Gdriver,&Gmode,"");

float R=200;

float x=R;

float y=0;

float P=4.0/5.0-R;

putpixel(0+A,R+B,RED);

putpixel(0+A,-R+B,RED); putpixel(R+A,0+B,RED); putpixel(A-R,0+B,RED); while(x>y){

y++;

if(P<=0){

P=P+2\*y+1;

}else{

x--;

P=P=2\*y-2\*x+1;

}

if(x<y)break;

putpixel(A+x,B+y,RED); putpixel(A-x,B+y,RED); putpixel(A+x,B-y,RED); putpixel(A-x,B-y,RED); putpixel(A+y,B+x,RED); putpixel(A-y,B+x,RED); putpixel(A+y,B-x,RED); putpixel(A-y,B-x,RED); }

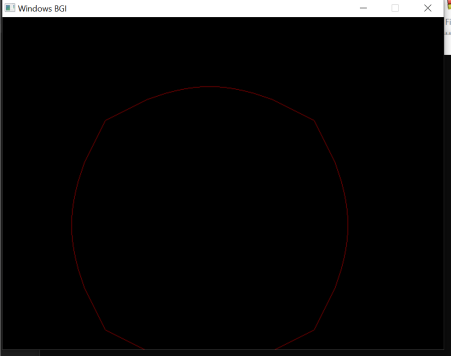
delay(10000);

closegraph();

return 0;

}

SCREENSHOT:



Conclusion:

The Midpoint circle drawing algorithm is an efficient method for scan converting and drawing circles in computer graphics. By utilizing integer arithmetic and a decision parameter, the algorithm determines the appropriate pixels to plot along the circumference of a circle. This lab report explored the implementation of the Midpoint circle drawing algorithm in C++ using the OpenGL library. The resulting circle was displayed in a graphical window, demonstrating the effectiveness and efficiency of the Midpoint circle drawing algorithm in drawing circles in computer graphics.

ELLIPSE:

Introduction:

The Midpoint ellipse drawing algorithm is a popular method in computer graphics used to scan convert and draw ellipses on a 2D display. This algorithm efficiently determines which pixels to plot to approximate an ellipse by utilizing the midpoint between two adjacent pixels. This lab report explores the Midpoint ellipse drawing algorithm, its implementation in C++, and its effectiveness in drawing ellipses.

Algorithm:

The Midpoint ellipse drawing algorithm uses integer arithmetic and a decision parameter to determine the appropriate pixels to plot along the boundary of an ellipse. It iteratively calculates the midpoint between two adjacent pixels and selects the closest pixel based on the midpoint's position relative to the ellipse. The following steps outline the Midpoint ellipse drawing algorithm:

1. Obtain the center coordinates of the ellipse, (x0, y0), and its major and minor axes lengths, a and b.
2. Calculate the decision parameters:
   * Calculate the squares of the major and minor axes lengths: a\_sqr = a \* a and b\_sqr = b \* b.
   * Set the initial pixel location to (0, b).
   * Calculate the decision parameter for the first region: p1 = b\_sqr - a\_sqr \* b + 0.25 \* a\_sqr.
   * Set the initial delta values for x and y: dx = 2 \* b\_sqr \* x and dy = 2 \* a\_sqr \* y.
3. Repeat the following steps while dx > dy:
   * Plot pixels at symmetric positions around the ellipse based on the current pixel location.
   * If the decision parameter (p1) is less than 0, update the decision parameter and the delta values for x and y.
   * Otherwise, update the decision parameter, the delta values for x and y, and decrement y.
4. Calculate the decision parameter for the second region: p2 = b\_sqr \* (x + 0.5) \* (x + 0.5) + a\_sqr \* (y - 1) \* (y - 1) - a\_sqr \* b\_sqr.
5. Repeat the following steps while y > 0:
   * Plot pixels at symmetric positions around the ellipse based on the current pixel location.
   * If the decision parameter (p2) is greater than 0, update the decision parameter and the delta values for x and y.
   * Otherwise, update the decision parameter, the delta values for x and y, and increment x and decrement y.

Code:

#include<graphics.h>

#include<iostream>

using namespace std;

int main(){

int Gdriver=DETECT,Gmode;

int x;scanf("%d",&x);

int x\_rad;scanf("%d",&x\_rad);

int y;scanf("%d",&y);

int y\_rad;scanf("%d",&y\_rad);

int start\_angle;scanf("%d",&start\_angle);

int end\_angle;scanf("%d",&end\_angle);

initgraph(&Gdriver,&Gmode,"");

ellipse(x,y,start\_angle,end\_angle,x\_rad,y\_rad);

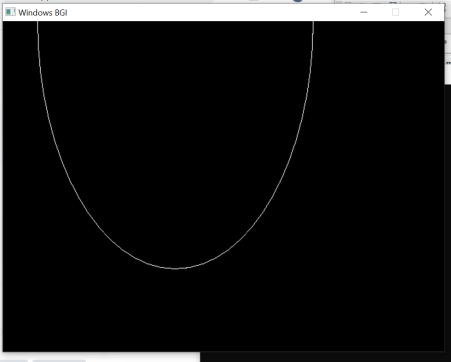
delay(10000);

closegraph();

return 0;

}

SCREENSHOT:



Conclusion:

The Midpoint ellipse drawing algorithm is an efficient method for scan converting and drawing ellipses in computer graphics. By utilizing integer arithmetic and decision parameters, the algorithm determines the appropriate pixels to plot along the boundary of an ellipse. This lab report explored the implementation of the Midpoint ellipse drawing algorithm in C++ using the OpenGL library. The resulting ellipse was displayed in a graphical window, demonstrating the effectiveness and efficiency of the Midpoint ellipse drawing algorithm in drawing ellipses in computer graphics.