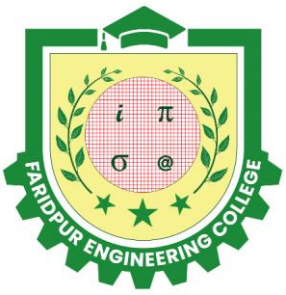
FARIDPUR ENGINEERING COLLEGE



Dept. of Computer Science & Engineering

Lab Report

Subject Name : Computer Graphics (Sessional)

Subject Code : CSE 802

Lab Report On : Implement Bresenham's circle drawing algorithm

Submitted By:

Name : Md. Rakibul Islam Nahid

ID No. : 2537

Reg No. : 1797

Semester : 8th

Session : 2017-18

Submitted To:

Md. Rony Ahmed

Lecturer, Dept of CSE

Faridpur Engineering College, Faridpur

REMARKS

Date of submission

Experiment No: 04

Experiment Name: Implement Bresenham’s circle drawing algorithm.

Aim: The aim of this lab report is to implement Bresenham's circle drawing algorithm, which is a fast and efficient algorithm for drawing circles in a digital space. The report will provide a detailed analysis of the algorithm's implementation, including its strengths and weaknesses, and compare it with other circle drawing algorithms. The report will also present the results of the implementation, including the accuracy and efficiency of the algorithm, and discuss its practical applications in computer graphics.

Description: The algorithm works by determining the points on a circle's circumference that are closest to a pixel's center and uses these points to draw the circle. Bresenham's algorithm uses an iterative process to determine the next pixel to be drawn based on the error accumulated in previous iterations. This approach results in a highly accurate circle and significantly reduces the number of calculations required to draw the circle.

One of the main advantages of Bresenham's circle drawing algorithm is its speed. The algorithm requires only integer arithmetic operations, making it efficient and ideal for low-powered devices. This speed also makes it suitable for real-time applications such as video games and simulations.

Despite its advantages, Bresenham's algorithm is not without its limitations. The algorithm is primarily designed for circles that have their center at the origin and have a radius of less than one-half the screen width or height. This limitation can be addressed by transforming the circle into a more convenient position or by scaling the circle to fit within these bounds.

Requirements:

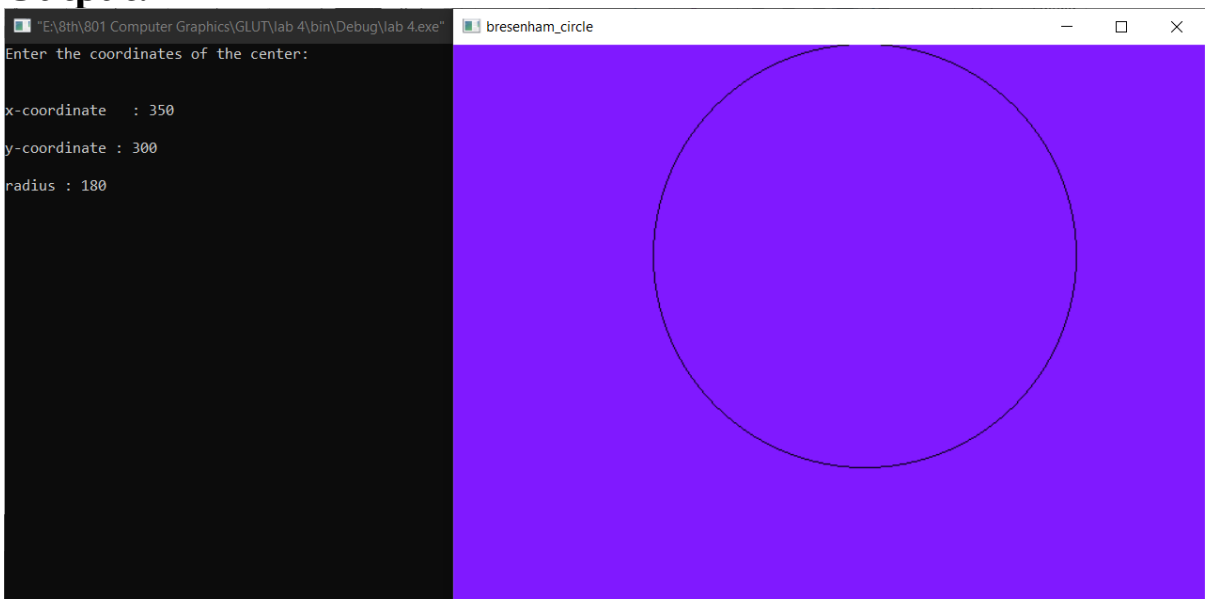
1. Codeblocks
2. OpenGL
3. Glut

Algorithm:

1. Define the circle's center point (xc, yc) and its radius r.
2. Set the initial point (x, y) to (0, r).
3. Calculate the initial value of the decision parameter as P = 3 - 2r.
4. While x <= y: a. Draw the circle's symmetric points using the current point (x, y) and the circle's center point (xc, yc). b. If the decision parameter is less than or equal to 0 (P <= 0), increment x and update the decision parameter as P = P + 4x + 6. c. If the decision parameter is greater than 0 (P > 0), increment x and y, and update the decision parameter as P = P + 4(x - y) + 10.
5. Repeat the above steps until all points on the circle's circumference have been drawn.

The above algorithm uses only integer arithmetic operations, making it fast and efficient for drawing circles in a digital space. The algorithm also ensures that the circle's center point can be located anywhere on the screen and that the circle's radius can be any positive integer value.

Output:



Discussion:

1. Write the code properly, according to the algorithm.
2. Make the syntax and function correct.
3. Put the input value properly to express the graphical representation according to display size.
4. The unique part of this algorithm is that it uses only integer arithmetic.