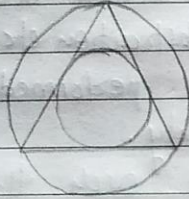


- # Title: Draw the pattern using Bresenham's line & circle drawing algorithm.

Problem Statement:

Write a C++ program to draw the following pattern, Use DDA & Bresenham's line drawing algorithm.



Objectives

- learn Bresenham's line drawing algorithm
- learn Bresenham's Circle drawing algorithm

Outcomes:

- Implement Bresenham's line drawing & circle drawing algorithm in C++.

Hardware Requirements:

Intel core i5 8th gen processor.
8 GB RAM, 512 GB SSD

Software Requirements:

VS creator 4.12.4 on windows 10 operating system.

Theory

→ DDA: DDA stands for Digital Difference Analysis.

It is an incremental method of scan conversion of line.

Adv: 7 faster method

→ easy to implement & understand

→ plotting same point twice isn't possible

Disadv: \Rightarrow It involves float calculations.

\Rightarrow Rounding operations due to floats consume lot of processing time.

\Rightarrow Bresenham's line drawing Algorithm:

- This algorithm is used for scan line converting. It is an efficient method because it involves only integer addition, subtraction etc.
- These operations can be performed very rapidly, so lines are drawn quickly.
- Next pixel is selected such that it is at least distance from true line.

Advantage: \Rightarrow Involves only integer arithmetic hence simple & fast algorithm.

\Rightarrow Avoids generation of duplicate points

\Rightarrow Faster as compared to DDA as it doesn't use floating point calculations.

* Disadvantage: \Rightarrow meant for basic line drawing
 \Rightarrow smooth line is not possible.

\Rightarrow Bresenham's Circle Drawing Algorithm:

- Points are generated from 0° to 45° only & then those points are mirrored to get the full circle.
- For this the moves will be made only in x & y dir.
- The best approximation of true circle will be described by those pixels in the raster that falls the least distance from true circle.

Algorithm (Pseudo Code)

Algorithm drawpattern (length of triangle side a)
 // in draw pattern function we use dda & bresenham
 algorithm to draw pattern

$$x_1 = 200, y_1 = 400$$

$$x_2 = 200 + a, y_2 = 400$$

$$\text{if } (\text{abs}(x_2 - x_1) \geq \text{abs}(y_2 - y_1)) \quad l = \text{abs}(x_2 - x_1)$$

$$\text{else } l = \text{abs}(y_2 - y_1)$$

$$dx = (x_2 - x_1) / l$$

$$dy = (y_2 - y_1) / l$$

$$i = 0, x = x_1, y = y_1$$

while ($i < l$) {

 setPixel(x, y)

$i \leftarrow i + 1$

$x \leftarrow x + dx$

$y \leftarrow y + dy$

// Repeat above procedure for $(200, 400), (200 + \frac{a}{2}, 400 - \frac{a\sqrt{3}}{2})$

// for circle

$$x_c = 200 + \frac{a}{2}, y_c = 400 - \frac{a\sqrt{3}}{2}$$

$$r = \frac{a\sqrt{3}}{6}$$

$$d = 3 - 2 * r$$

$$x = 0, y = r$$

while ($x \leq y$) {

 plot($x_c + x, y_c + y$)

 plot($x_c - y, y_c - x$)

 plot($x_c + y, y_c - x$)

 plot($x_c - y, y_c + x$)

 plot($x_c + y, y_c + x$)

 plot($x_c - x, y_c - y$)

 plot($x_c + x, y_c - y$)

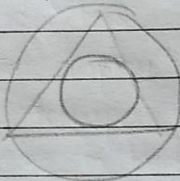
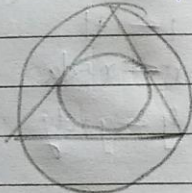
 plot($x_c - x, y_c + y$)


```

if (d < 0) d = d + 4 * x + 6;
else if (d > 0) d = d + 4 * x - 4 * y + 10;
        y = y - 1;
    }
    x = x + 1;
}
// for circumcircle repeat same for  $z = \frac{a\sqrt{3}}{3}$ 

```

Test cases

Input	Expected output	Actual output	Results
$a = 100$			Pass

Conclusion

- learned to use basic widgets of Qt
- implemented the line drawing & circle drawing algorithms.

