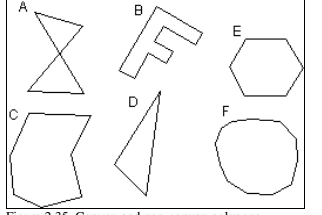


| **TITLE:** Draw the Following polygon/shape/curve.  a. Bresenham Circle Drawing Algorithm.  b. Other shapes |
| --- |

**AIM:**

a. Generate the Circle using Bresenham Circle Drawing Algorithm

b. Draw polygon shown in following figure



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**Understand the basic concepts of computer graphics and OpenGL**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

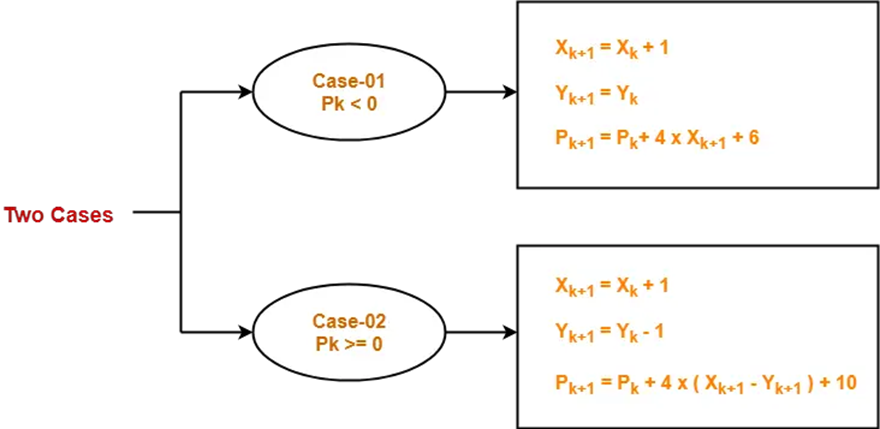
[**https://www.gatevidyalay.com/bresenham-circle-drawing-algorithm/**](https://www.gatevidyalay.com/bresenham-circle-drawing-algorithm/)

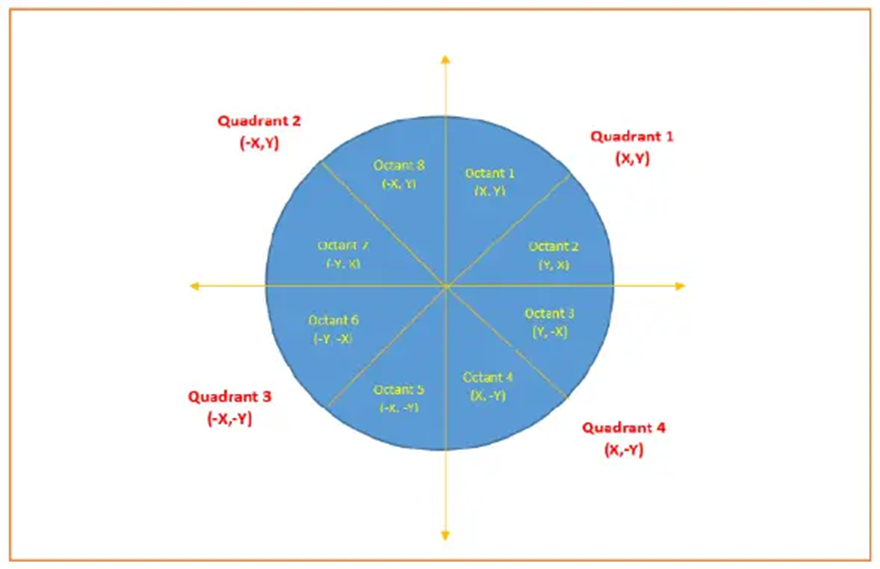
google

geeksforgeeks

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Algorithm:**

1. Bresenham’s circle drawing algorithm:
2. Start
3. Set initial values: (x, y) = (0, r) and p = 3 - 2 \* r.
4. Plot the eight symmetrical points for (x, y).
5. Adjust the decision parameter based on its value and update x and y accordingly.  
   
6. Continue the process until x > y.
7. Stop

**  
 Taken x and y co-ordinates as (0,0) to get it in center.**

**Implementation details:**

**a )** Bresenham Circle Drawing Algorithm

**#include <GL/glew.h>**

**#include <GLFW/glfw3.h>**

**#include <iostream>**

**using namespace std;**

**void plotCirclePoints(GLint xc, GLint yc, GLint x, GLint y) {**

**glBegin(GL\_POINTS);**

**glVertex2i(xc + x, yc + y);**

**glVertex2i(xc - x, yc + y);**

**glVertex2i(xc + x, yc - y);**

**glVertex2i(xc - x, yc - y);**

**glVertex2i(xc + y, yc + x);**

**glVertex2i(xc - y, yc + x);**

**glVertex2i(xc + y, yc - x);**

**glVertex2i(xc - y, yc - x);**

**glEnd();**

**}**

**void drawCircle(GLint xc, GLint yc, GLint r) {**

**GLint x = 0, y = r;**

**GLint p = 3 - 2 \* r;**

**plotCirclePoints(xc, yc, x, y);**

**while (x <= y) {**

**x++;**

**p = (p < 0) ? (p + 4 \* x + 6) : (p + 4 \* (x - y) + 10);**

**if (p >= 0) y--;**

**plotCirclePoints(xc, yc, x, y);**

**}**

**}**

**int main() {**

**GLint r;**

**cout << "Enter the radius (r) of the circle:" << endl;**

**cin >> r;**

**glfwInit();**

**GLFWwindow\* window = glfwCreateWindow(400, 400, "Bresenham Circle Drawing", NULL, NULL);**

**glfwMakeContextCurrent(window);**

**glewInit();**

**glViewport(0, 0, 400, 400);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**glOrtho(0, 400, 400, 0, -1, 1);**

**glPointSize(0.2f);**

**while (!glfwWindowShouldClose(window)) {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**drawCircle(200, 200, r);**

**glfwSwapBuffers(window);**

**glfwPollEvents();**

**}**

**glfwTerminate();**

**return 0;**

**}**

b) Polygons:

#include <bits/stdc++.h>

#include <GL/glut.h>

void A() {

glBegin(GL\_LINE\_LOOP);

glVertex2i(30, 90);

glVertex2i(70, 80);

glVertex2i(22, 30);

glVertex2i(75, 32);

glVertex2i(30, 90);

glEnd();

}

void B() {

glBegin(GL\_LINE\_LOOP);

glVertex2i(120, 120);

glVertex2i(170, 110);

glVertex2i(165, 100);

glVertex2i(135, 108);

glVertex2i(128, 95);

glVertex2i(145, 90);

glVertex2i(140, 80);

glVertex2i(120, 85);

glVertex2i(110, 45);

glVertex2i(100, 55);

glVertex2i(120, 120);

glEnd();

}

void C() {

glBegin(GL\_LINE\_LOOP);

glVertex2i(190, 80);

glVertex2i(250, 78);

glVertex2i(240, 50);

glVertex2i(252, 30);

glVertex2i(215, 20);

glVertex2i(180, 35);

glVertex2i(180, 60);

glVertex2i(190, 80);

glEnd();

}

void D() {

glBegin(GL\_LINE\_LOOP);

glVertex2i(260, 30);

glVertex2i(310, 70);

glVertex2i(280, 110);

glEnd();

}

void Hexagon(int xc, int yc, int r) {

glBegin(GL\_LINE\_LOOP);

for (int i = 0; i < 6; ++i) {

glVertex2i(xc + r \* cos(i \* M\_PI / 3), yc + r \* sin(i \* M\_PI / 3));

}

glEnd();

}

void IrregularCircle(int xc, int yc, int r) {

glBegin(GL\_LINE\_LOOP);

for (int i = 0; i < 10 ; ++i) {

glVertex2i(xc + r \* cos(i \* M\_PI / 5), yc + r \* sin(i \* M\_PI / 5));

}

glEnd();

}

void display(void) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 1.0, 0.0);

A();

B();

Hexagon(360, 70, 40);

C();

D();

IrregularCircle(460, 70, 40);

glFlush();

}

void init(void) {

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, 500, 0, 150);

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 150);

glutInitWindowPosition(100, 100);

glutCreateWindow("Shapes");

init();

glutDisplayFunc(display);

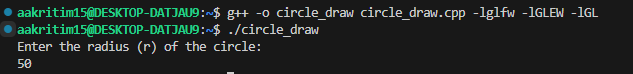
glutMainLoop();

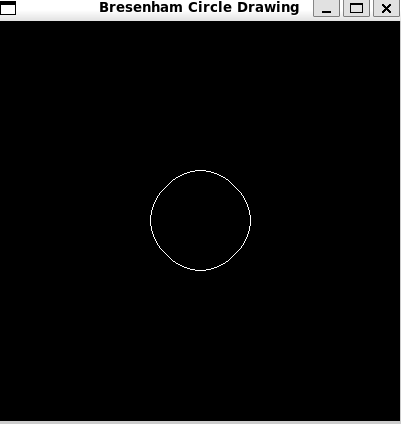
return 0;

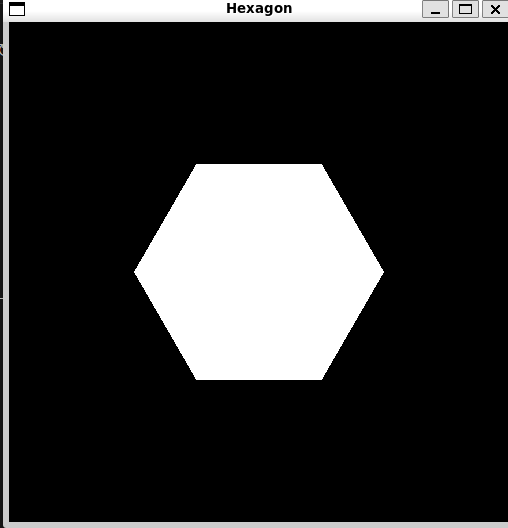
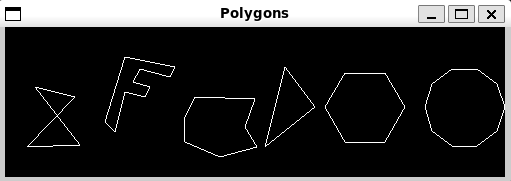
}

**Output(s) (final edited screen shot):**

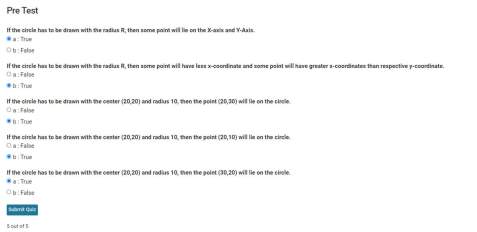
**a)Bresenham’s Circle Drawing Algorithm**

****

****

**Polygon:  
  
  
**

**Screenshots from vlab:**

****

**Conclusion and discussion:**

**Learned Bresenham’s Circle Darwing Algorithm and how to draw different polygons.**

**Date: 19/8/24**

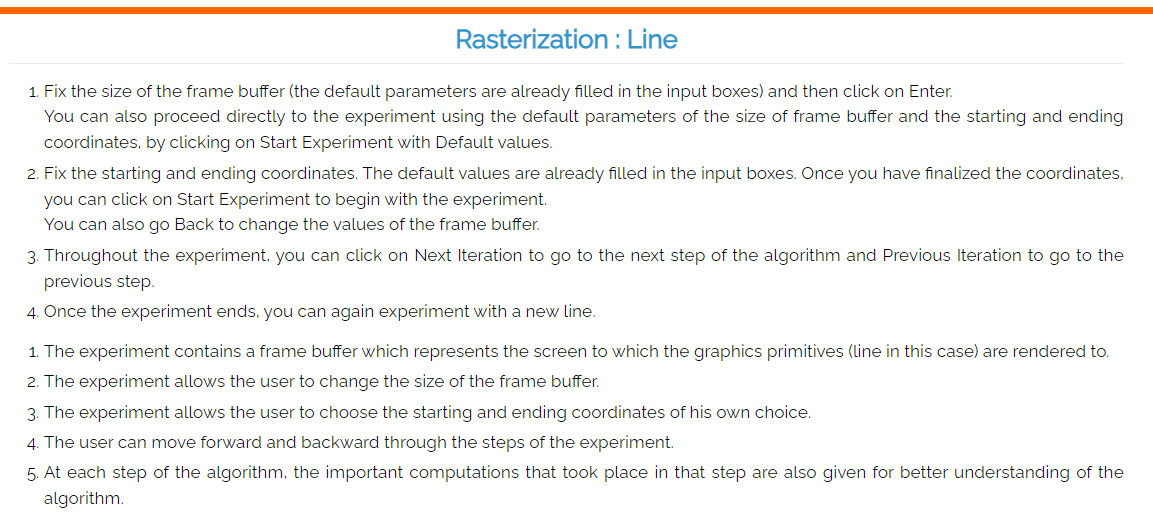
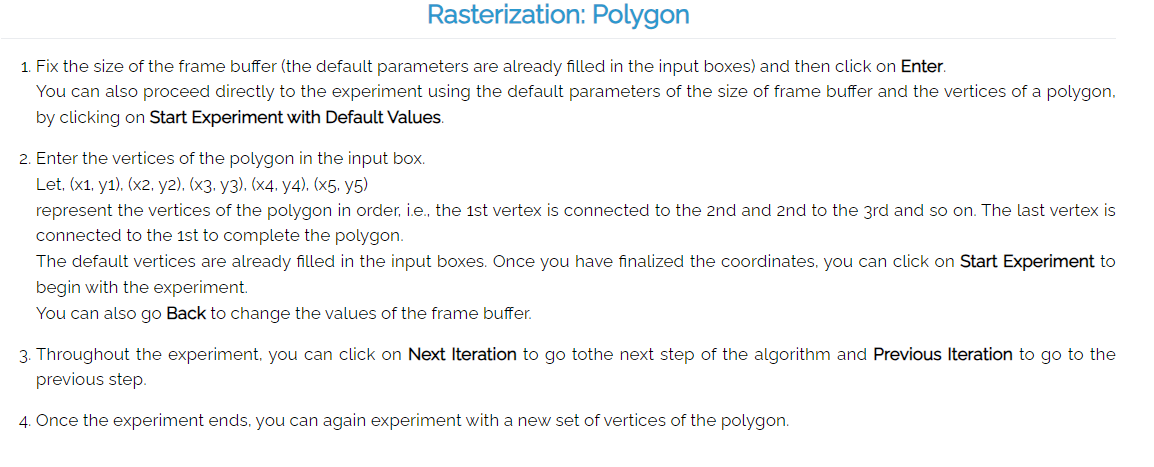
**Signature of faculty in-charge**

**Post lab**

**Visit and explore and paste your screenshot**

[**https://cse18-iiith.vlabs.ac.in/exp/rasterization-line/**](https://cse18-iiith.vlabs.ac.in/exp/rasterization-line/)

[**https://cse18-iiith.vlabs.ac.in/exp/rasterization-polygon/**](https://cse18-iiith.vlabs.ac.in/exp/rasterization-polygon/)

**  
**