# Graphics Assignment No: 2

Shikhar Jaiswal
1601CS44

February 2, 2019

## 1 Assignment Description

Implement Bresenham's circle drawing algorithm and draw a circle using its center coordinates and radius.

## 2 Procedure

### Installation

Install the following packages from the Ubuntu repository:
- freeglut3-dev
- mesa-common-dev

```
sudo apt-get install freeglut3 freeglut3-dev mesa-common-dev
```

Check your $/usr/include/GL$ folder to verify the installation of the OpenGL headers that you intend to use.

### Compiling and Linking

We will have to use the $-lglut$ linker option with $gcc/g{+}{+}$ to compile a program with *glut* library.

For example, to compile the program, use the following to get the binary executable code:

```
g++ bresenham.cpp -lGL -lGLU -lglut -o bresenham
```

# 3 Discussion

The primary objective of the assignment is to implement an algorithm capable of determining the individual pixels needed for rasterizing a circle. For this purpose, Bresenham's Circle Drawing algorithm is implemented. The algorithm works by tracing individual pixels in one of the octants of the standard *xy*-plane and tracing the corresponding equidistant reflections in the other seven remaining octants. For a detailed explanation, please refer the code.

## OpenGL Code

```
#include <stdio.h>
#include <GL/glut.h>

// Global variables for circle's coordinates and radius.
int x_center, y_center, radius;

void plot(int x, int y)
{
    // Draw the points with respect to the center.
    glBegin(GL_POINTS);
        glVertex2i(x_center + x, y_center + y);
    glEnd();
}

void display()
{
    // Set the display area colour set using glClearColor().
    glClear(GL_COLOR_BUFFER_BIT);
    // Colour fill.
    glColor3ub(255, 255, 255);
    // Set point sizes.
    glPointSize(2.0);

    // Initial parameters for points.
    int x = 0;
    int y = radius;
    int h = 1 - radius;

    // Draw the points where the circle crosses the axes.
    plot(x, y);
    plot(x, -y);
    plot(y, x);
    plot(-y, x);

    // Main part of the algorithm.
    while (y > x)
    {
        if (h < 0)
        {
            h += (2 * x) + 3;
        } else
        {
            h += (2 * (x - y)) + 5;
            y -= 1;
        }

        x += 1;

        // Plot the points in counter-clockwise fashion.
        plot(x, y);
        plot(-x, y);
        plot(-y, x);
        plot(-y, -x);
        plot(-x, -y);
        plot(x, -y);
        plot(y, -x);
        plot(y, x);
    }

    // Begin execution of the above code.
    glFlush();
}

int main(int argc, char **argv)
{
    char c;

    printf("Bresenham's Circle Drawing Algorithm\n");
    printf("Please Enter The X-Coordinate of the Center: ");
    scanf("%d%c", &x_center, &c);
    printf("Please Enter The Y-Coordinate of the Center: ");
    scanf("%d%c", &y_center, &c);
    printf("Please Enter The Radius of the Circle: ");
    scanf("%d%c", &radius, &c);

    // Initialize to the command-line arguments.
    glutInit(&argc, argv);
    // Setup the colour depth of the window buffers.
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);

    // Assign the position, size and name to the window.
```

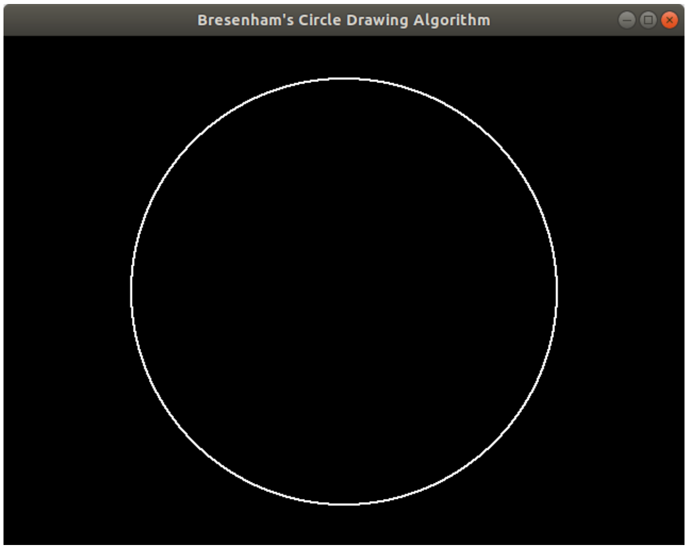
```c
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(640, 480);
    glutCreateWindow("Bresenham's Circle Drawing Algorithm");

    // Setup a black background.
    glClearColor(0.0, 0.0, 0.0, 0.0);
    // Setup viewing projection.
    glMatrixMode(GL_PROJECTION);
    // Initialize identity matrix.
    glLoadIdentity();
    // Setup a viewport.
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);

    // Pass the display function to generate the display.
    glutDisplayFunc(display);
    // Hand over the execution to the glut library.
    glutMainLoop();

    return 0;
}
```

## Python Code

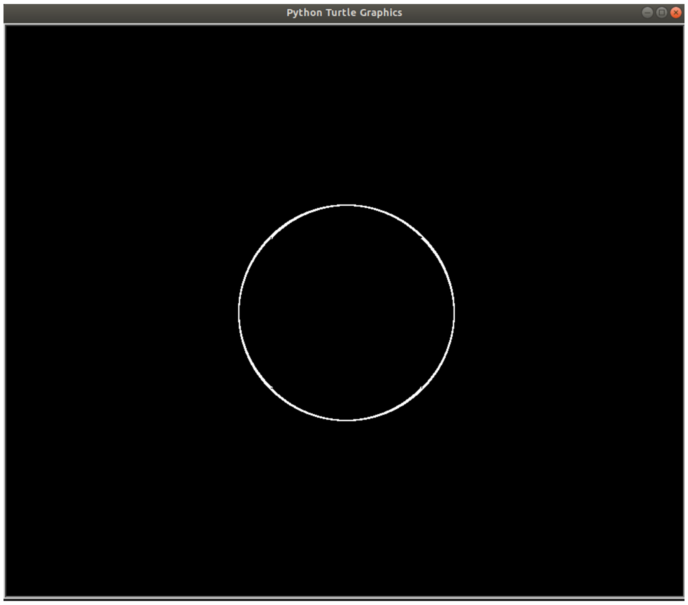
```python
# Using turtle graphics library.
import turtle

# Function to draw a single point.
def draw_point(x_center, y_center, x, y):
    turtle.goto(x_center + x, y_center + y)
    turtle.dot(5, "white")

# Function to draw the whole circle.
def draw_circle(x_center, y_center, radius):

    # Initial arguments.
    x_center = int(x_center)
    y_center = int(y_center)
    x = int(0)
    y = int(radius)
    h = int(1 - y)

    # Draw the points where the circle crosses the axes.
    draw_point(x_center, y_center, x, y);
    draw_point(x_center, y_center, x, -y);
    draw_point(x_center, y_center, y, x);
    draw_point(x_center, y_center, -y, x);

    # Main part of the algorithm.
    while y > x:
        if h < 0:
            h += (2 * x) + 3;
        else:
            h += (2 * (x - y)) + 5;
            y -= 1;

        x += 1;

        # Plot the points in counter-clockwise fashion.
        draw_point(x_center, y_center, x, y);
        draw_point(x_center, y_center, -x, y);
        draw_point(x_center, y_center, -y, x);
        draw_point(x_center, y_center, -y, -x);
        draw_point(x_center, y_center, -x, -y);
        draw_point(x_center, y_center, x, -y);
        draw_point(x_center, y_center, y, -x);
        draw_point(x_center, y_center, y, x);

# Initial input.
print("Bresenham's Circle Drawing Algorithm\n")
x_center = input("Please Enter The X-Coordinate of the Center: ")
y_center = input("Please Enter The Y-Coordinate of the Center: ")
radius = input("Please Enter The Radius of the Circle: ")

# Initialization and background colour.
turtle.setup()
turtle.bgcolor("black")

# Set the fill colour to black/
turtle.fillcolor("black")
# Draw the circle.
draw_circle(x_center, y_center, radius)

# Exit on click.
turtle.exitonclick()
```

## 4 Result

### OpenGL



### Python



# 5 References

[1] How to install OpenGL/GLUT libraries

[2] An Introduction to OpenGL Programming

[3] Turtle Graphics - The Python Standard Library

[4] Wikipedia - Midpoint Circle Algorithm