# Department of CSE
# SSN College of Engineering

**Vishakan Subramanian - 18 5001 196 - Semester VII**

09 August 2021

---

## UCS 1712 - Graphics And Multimedia Lab

---

### Exercise 4: Midpoint Circle Drawing Algorithm in C++ using OpenGL

### Aim:

a) To plot points that make up the circle with center (xc,yc) and radius r using Midpoint circle drawing algorithm. Give atleast 2 test cases.

- Case 1: With center (0,0)
- Case 2: With center (xc,yc)

b) To draw any object using line and circle drawing algorithms.

## Code: Midpoint Circle Drawing Algorithm:

```
/* To draw a circle using the midpoint circle drawing algorithm with
   OpenGL */

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

GLint x, y, radius; //Variables for circle

const int WINDOW_WIDTH = 800;
const int WINDOW_HEIGHT = 800;

class screenPoint{
    /* Class for a coordinate point */

private:
    GLint x, y;

public:
    screenPoint(){
        //Default constructor, initialize to (0, 0)
        x = y = 0;
    }

    screenPoint(GLint xCoord, GLint yCoord){
        //Constructor with preset points
        x = xCoord;
        y = yCoord;
    }

    void setCoords(GLint xCoord, GLint yCoord){
        //Function to set coordinates
        x = xCoord;
        y = yCoord;
    }

    GLint getX() const{
        //Return x coordinate
        return x;
    }

    GLint getY() const{
        //Return y coordinate
        return y;
    }

    void incX(){
```

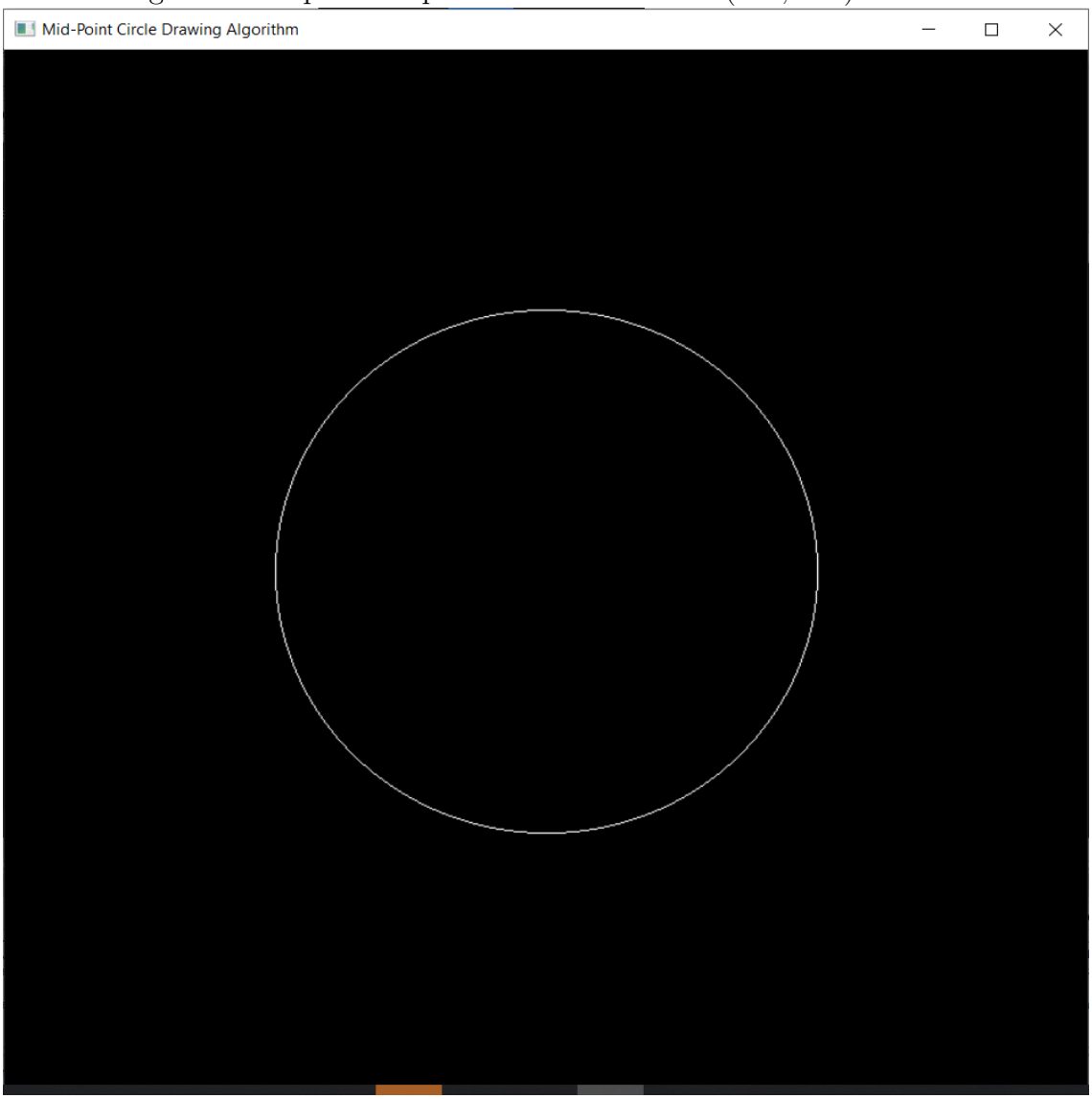
```
47          //Increment x coordinate
48          x++;
49      }
50
51      void decY(){
52          //Decrement y coordinate
53          y--;
54      }
55
56      void printCoords(){
57          //Print current coordinates
58          printf("\nX: %d\tY: %d", x, y);
59      }
60  };
61
62  void initializeDisplay();
63  void drawCircle();
64  void setPixel(GLint xCoord, GLint yCoord);
65  void circleMidPoint(GLint xc, GLint yc, GLint radius);
66  void circlePlotPoints(GLint xc, GLint yc, screenPoint circlePoint);
67
68
69  int main(int argc, char **argv){
70
71      printf("\n\t\tEnter the center coordinates of the circle\n");
72
73      printf("\nEnter the X coordinate: ");
74      scanf("%d", &x);
75
76      printf("\nEnter the Y coordinate: ");
77      scanf("%d", &y);
78
79      printf("\nEnter the radius of the circle: ");
80      scanf("%d", &radius);
81
82      glutInit(&argc, argv);
83      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
84      glutInitWindowPosition(100, 100);
85      glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
86      glutCreateWindow("Mid-Point Circle Drawing Algorithm");
87
88      initializeDisplay();
89      glutDisplayFunc(drawCircle);
90      glutMainLoop();
91
92      return 1;
93  }
94
95  void initializeDisplay(){
96      //Initialize the display parameters
97
```

```
98      glClearColor(0, 1, 1, 0);       //Display window color
99      glMatrixMode(GL_PROJECTION);    //Choose projection
100     gluOrtho2D(0, WINDOW_WIDTH, 0, WINDOW_HEIGHT);     //Set
        transformation
101 }
102
103 void setPixel(GLint xCoord, GLint yCoord){
104     //Draw a pixel at the given point
105
106     glBegin(GL_POINTS);
107     glVertex2i(xCoord, yCoord);
108     glEnd();
109 }
110
111 void circleMidPoint(GLint xc, GLint yc, GLint radius){
112     //Implementation of the midpoint circle drawing algorithm
113
114     screenPoint circlePoint;
115
116     GLint p = 1 - radius;   //Initial value for midpoint parameter
117
118     circlePoint.setCoords(0, radius);   //Set coordinates at top of circle
119
120     circlePlotPoints(xc, yc, circlePoint);  //Plot initial point
121     //circlePoint.printCoords();
122
123     //Calculate the next points while X < Y
124     while(circlePoint.getX() < circlePoint.getY()){
125         circlePoint.incX();
126
127         if(p < 0){
128             p += 2 * circlePoint.getX() + 1;
129         }
130
131         else{
132             circlePoint.decY();
133             p += 2 * (circlePoint.getX() - circlePoint.getY()) + 1;
134         }
135
136         //circlePoint.printCoords();
137         circlePlotPoints(xc, yc, circlePoint);
138     }
139
140     glFlush();  //Flush the completed circle output to display
141 }
142
143 void circlePlotPoints(GLint xc, GLint yc, screenPoint circlePoint){
144     //Plot points for all 8 octants of the circle
145
146     setPixel(xc + circlePoint.getX(), yc + circlePoint.getY());
147     setPixel(xc - circlePoint.getX(), yc + circlePoint.getY());
```

```
148     setPixel(xc + circlePoint.getX(), yc - circlePoint.getY());
149     setPixel(xc - circlePoint.getX(), yc - circlePoint.getY());
150     setPixel(xc + circlePoint.getY(), yc + circlePoint.getX());
151     setPixel(xc - circlePoint.getY(), yc + circlePoint.getX());
152     setPixel(xc + circlePoint.getY(), yc - circlePoint.getX());
153     setPixel(xc - circlePoint.getY(), yc - circlePoint.getX());
154 }
155
156 void drawCircle(){
157     //Driver function to call the circle drawing function
158
159     circleMidPoint(x, y, radius);
160 }
```

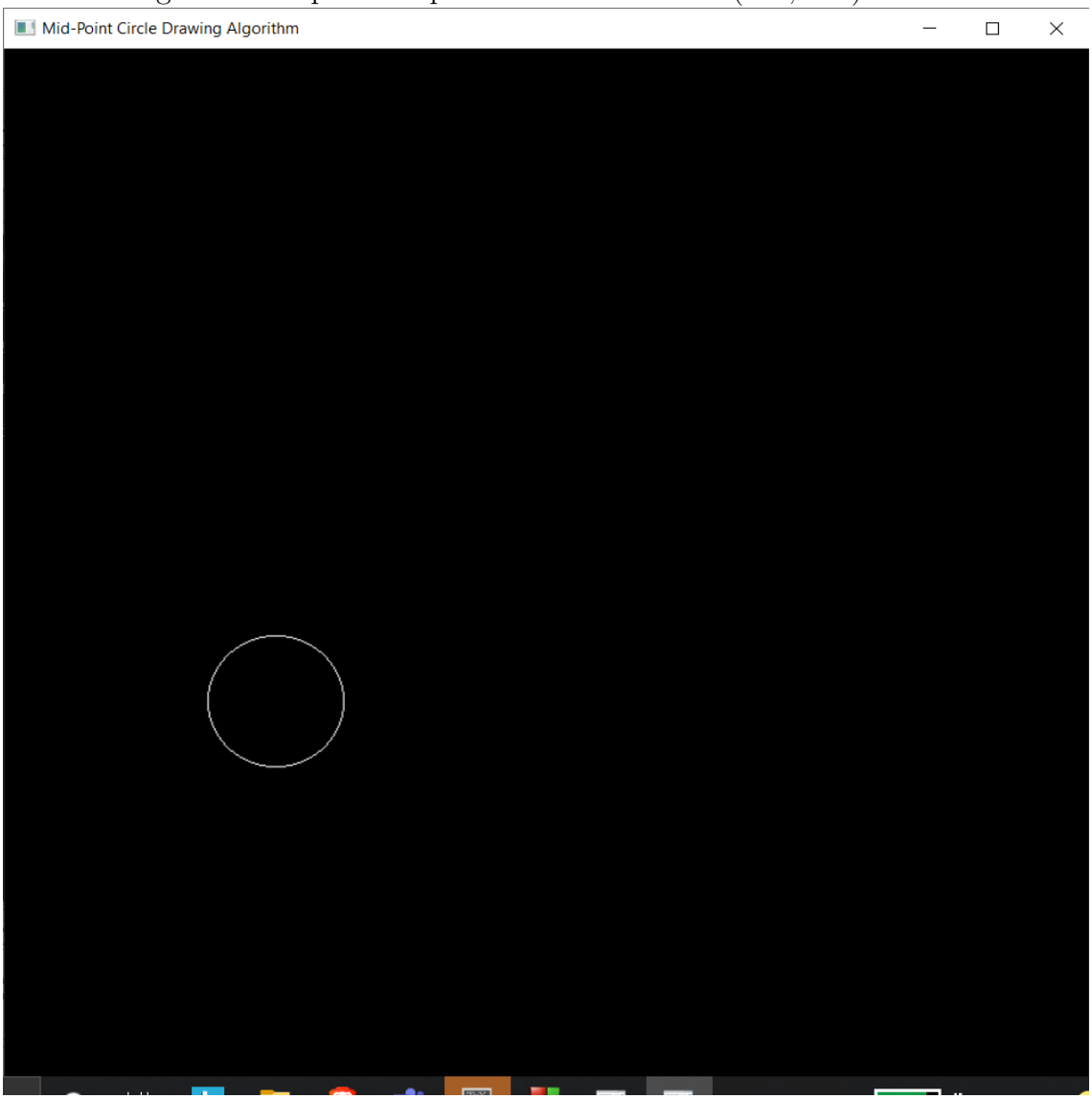
## Output: Midpoint Circle Case 1 - C(400, 400) R - 200:

Figure 1: Output: Midpoint Circle Case 1 - C(400, 400) R - 200.



## Output: Midpoint Circle Case 2 - C(200, 300) R - 50:

Figure 2: Output: Midpoint Circle Case 2 - C(200, 300) R - 50.



## Code: Object Using Circles and Lines:

```
/* To draw an object with circles and lines - Stick man */

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

const int WINDOW_WIDTH = 800;
const int WINDOW_HEIGHT = 800;

class screenPoint{
    /* Class for a coordinate point */

private:
    GLint x, y;

public:
    screenPoint(){
        //Default constructor, initialize to (0, 0)
        x = y = 0;
    }

    screenPoint(GLint xCoord, GLint yCoord){
        //Constructor with preset points
        x = xCoord;
        y = yCoord;
    }

    void setCoords(GLint xCoord, GLint yCoord){
        //Function to set coordinates
        x = xCoord;
        y = yCoord;
    }

    GLint getX() const{
        //Return x coordinate
        return x;
    }

    GLint getY() const{
        //Return y coordinate
        return y;
    }

    void incX(){
        //Increment x coordinate
        x++;
    }
```

```
48
49      void decY(){
50          //Decrement y coordinate
51          y--;
52      }
53
54      void printCoords(){
55          //Print current coordinates
56          printf("\nX: %d\tY: %d", x, y);
57      }
58  };
59
60  void drawStickman();
61  void initializeDisplay();
62  void setPixel(GLint xCoord, GLint yCoord);
63  void drawCircle(GLint xc, GLint yc, GLint radius);
64  void drawLine(GLint x1, GLint y1, GLint x2, GLint y2);
65  void circlePlotPoints(GLint xc, GLint yc, screenPoint circlePoint);
66
67
68  int main(int argc, char **argv){
69
70      glutInit(&argc, argv);
71      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
72      glutInitWindowPosition(100, 100);
73      glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
74      glutCreateWindow("Stickman using Circles & Lines");
75
76      initializeDisplay();
77      glutDisplayFunc(drawStickman);
78      glutMainLoop();
79
80      return 1;
81  }
82
83  void initializeDisplay(){
84      //Initialize the display parameters
85
86      glClearColor(0, 1, 1, 0);       //Display window color
87      glMatrixMode(GL_PROJECTION);    //Choose projection
88      gluOrtho2D(0, WINDOW_WIDTH, 0, WINDOW_HEIGHT);     //Set
        transformation
89  }
90
91  void setPixel(GLint xCoord, GLint yCoord){
92      //Draw a pixel at the given point
93
94      glBegin(GL_POINTS);
95      glVertex2i(xCoord, yCoord);
96      glEnd();
97  }
```

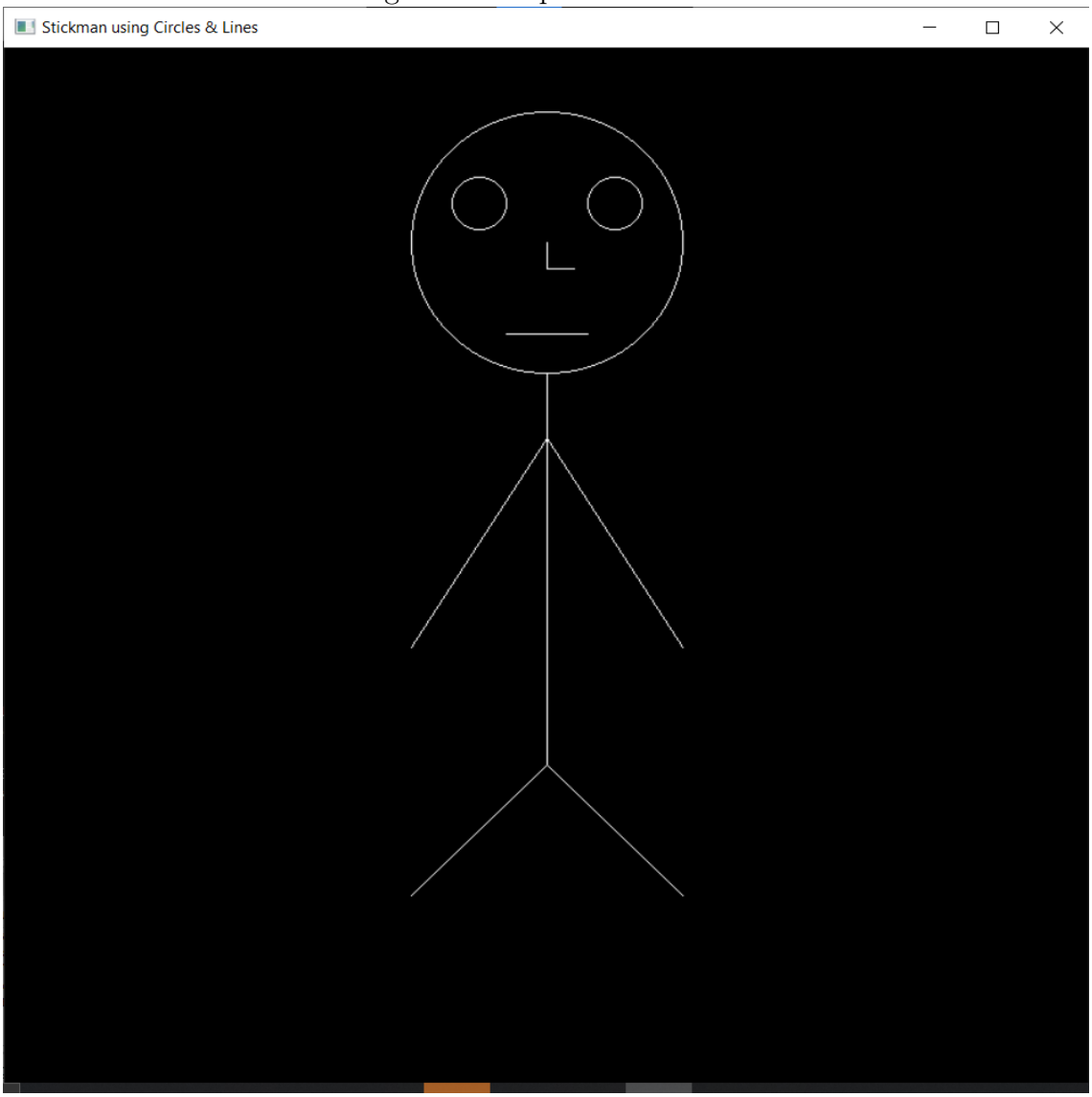
```
98
99  void drawCircle(GLint xc, GLint yc, GLint radius){
100     //Implementation of the midpoint circle drawing algorithm
101
102     screenPoint circlePoint;
103
104     GLint p = 1 - radius;   //Initial value for midpoint parameter
105
106     circlePoint.setCoords(0, radius);   //Set coordinates at top of circle
107
108     circlePlotPoints(xc, yc, circlePoint);  //Plot initial point
109     //circlePoint.printCoords();
110
111     //Calculate the next points while X < Y
112     while(circlePoint.getX() < circlePoint.getY()){
113         circlePoint.incX();
114
115         if(p < 0){
116             p += 2 * circlePoint.getX() + 1;
117         }
118
119         else{
120             circlePoint.decY();
121             p += 2 * (circlePoint.getX() - circlePoint.getY()) + 1;
122         }
123
124         //circlePoint.printCoords();
125         circlePlotPoints(xc, yc, circlePoint);
126     }
127
128     glFlush();  //Flush the completed circle output to display
129 }
130
131 void circlePlotPoints(GLint xc, GLint yc, screenPoint circlePoint){
132     //Plot points for all 8 octants of the circle
133
134     setPixel(xc + circlePoint.getX(), yc + circlePoint.getY());
135     setPixel(xc - circlePoint.getX(), yc + circlePoint.getY());
136     setPixel(xc + circlePoint.getX(), yc - circlePoint.getY());
137     setPixel(xc - circlePoint.getX(), yc - circlePoint.getY());
138     setPixel(xc + circlePoint.getY(), yc + circlePoint.getX());
139     setPixel(xc - circlePoint.getY(), yc + circlePoint.getX());
140     setPixel(xc + circlePoint.getY(), yc - circlePoint.getX());
141     setPixel(xc - circlePoint.getY(), yc - circlePoint.getX());
142 }
143
144 void drawLine(GLint x1, GLint y1, GLint x2, GLint y2){
145     //Bresenham's Line Drawing Algorithm Implementation
146
147     GLint dx, dy, x, y, p, inc_x1, inc_y1, inc_p1, inc_p2;
148     GLint x_sign, y_sign;
```

```
149
150     dx = x2 - x1;
151     dy = y2 - y1;
152
153     //note the sign for directionality
154     x_sign = dx/abs(dx);
155     y_sign = dy/abs(dy);
156
157     //increment is by 1, and in the direction of +/-
158     inc_x1 = 1 * x_sign;
159     inc_y1 = 1 * y_sign;
160
161     //change the differences to absolute values (crucial step)
162     dx = abs(dx);
163     dy = abs(dy);
164
165     //initial coordinates
166     x = x1;
167     y = y1;
168
169     glBegin(GL_POINTS);
170
171     if(abs(dx) > abs(dy)){
172         //X difference > Y difference
173         glVertex2i(x, y);
174
175         p = (2 * dy) - dx;
176         inc_p1 = 2 * (dy - dx);
177         inc_p2 = 2 * dy;
178
179         //plot for dx number of points
180         for(GLint i = 0; i < dx; i++){
181             if(p >= 0){
182                 y += inc_y1;
183                 p += inc_p1;
184             }
185             else{
186                 p += inc_p2;
187             }
188
189             x += inc_x1;
190
191             glVertex2i(x, y);
192         }
193     }
194     else{
195         //X difference <= Y difference
196         glVertex2i(x, y);
197
198         p = (2 * dx) - dy;
199         inc_p1 = 2 * (dx - dy);
```

```
        inc_p2 = 2 * dx;

        //plot for dy number of points
        for(GLint i = 0; i < dy; i++){
            if(p >= 0){
                x += inc_x1;
                p += inc_p1;
            }
            else{
                p += inc_p2;
            }

            y += inc_y1;

            glVertex2i(x, y);
        }
    }

    glEnd();
    glFlush();
}


void drawStickman(){
    //Draw a stick man using the drawLine() and drawCircle() algorithms

    //Face
    drawCircle(400, 650, 100);

    //Eyes
    drawCircle(350, 680, 20);
    drawCircle(450, 680, 20);

    //Nose
    drawLine(400, 650, 400, 630);
    drawLine(400, 630, 420, 630);

    //Mouth
    drawLine(370, 580, 430, 580);

    //Body
    drawLine(400, 550, 400, 250);

    //Arms
    drawLine(400, 500, 300, 340);
    drawLine(400, 500, 500, 340);

    //Legs
    drawLine(400, 250, 300, 150);
    drawLine(400, 250, 500, 150);
}
```

## Output: Stickman:

Figure 3: Output: Stickman.



## Learning Outcome:

- I understood the **Midpoint Circle Drawing Algorithm**'s working.
- I implemented the Midpoint Circle Drawing algorithm using an OpenGL program.
- I understood how the algorithm makes use of **octant symmetry** and plots points on the second octant and mirrors it to other octants.
- I understood how the parameter **p** is calculated in each iteration to determine the decrement of Y coordinate.
- I learnt about **classes and member functions** in C++ to implement methods to facilitate the circle drawing algorithm process.
- I was able to output all different test cases appropriately to verify the correctness of my program to implement the Midpoint Circle Drawing Algorithm.
- I was able to draw and output a **Stickman** using the **Midpoint Circle Drawing Algorithm** and the **Bresenham's Line Drawing Algorithm**.