

Department of Computer Science and Engineering

Shivanirudh S G, 185001146, Semester VII

31 July 2021

UCS1712 - Graphics and Multimedia Lab

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

Objective:

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 with centre as origin and elsewhere.

Code:

```
1 #ifndef LOPEGL_H
2 #define LOPEGL_H
3
4 #include <GL/freeglut.h>
5 #include <GL/gl.h>
6 #include <GL/glu.h>
```

```

7  #include <math.h>
8  #include <stdio.h>
9  #include<iostream>
10 #include<vector>
11 #include<ctime>
12 using namespace std;
13
14 #endif

1  #ifndef LUTIL_H
2  #define LUTIL_H
3
4  #include "Headers.h"
5
6  //Screen Constants
7  const int SCREEN_WIDTH = 640;
8  const int SCREEN_HEIGHT = 480;
9  const int SCREEN_FPS = 60;
10 const int POINT_SIZE=2;
11 int X0,Y0, radius;
12
13 const int coords[][3] ={{0, 0, 6},
14                          {2, 3, 6},
15                          };
16
17 bool initGL();
18
19 void update();
20
21 void render();
22
23 void y_axis();
24
25 void x_axis();
26
27 void selectCenter(int option);
28
29 vector<pair<int, int>> Midpoint();
30
31 #endif

1  #include "Signatures.h"
2
3  bool initGL(){
4      //Initialize Projection Matrix
5      glMatrixMode( GL_PROJECTION );

```

```

6      glLoadIdentity();
7      gluOrtho2D(0.0,640.0,0.0,480.0);
8
9      //Initialize Modelview Matrix
10     glMatrixMode( GL_MODELVIEW );
11     glLoadIdentity();
12
13     glTranslatef( SCREEN_WIDTH / 3.f, SCREEN_HEIGHT / 3.f, 0.
f );
14
15     //Initialize clear color
16     glClearColor( 0.f, 0.f, 0.f, 1.f );
17
18     glPointSize(POINT_SIZE);
19     glEnable(GL_POINT_SMOOTH);
20
21     //Check for error
22     GLenum error = glGetError();
23     if( error != GL_NO_ERROR )
24     {
25         printf( "Error initializing OpenGL! %s\n",
gluErrorString( error ) );
26         return false;
27     }
28
29     return true;
30 }
31
32 void update(){
33
34 }
35
36 void render(){
37     vector<pair<int, int>> points = Midpoint();
38     y_axis();
39     x_axis();
40     // glClear(GL_COLOR_BUFFER_BIT);
41     glColor3f(1,1,1);
42     glBegin(GL_POINTS);
43         for(pair<int, int> p: points){
44             glVertex2d(p.first, p.second);
45         }
46     glEnd();
47     glFlush();
48 }

```

```

49
50 void y_axis(){
51     // glClear(GL_COLOR_BUFFER_BIT);
52     glBegin(GL_LINES);
53         glVertex2d(0, -480.0);
54         glVertex2d(0, 480.0);
55     glEnd();
56     glFlush();
57 }
58
59 void x_axis(){
60     // glClear(GL_COLOR_BUFFER_BIT);
61     glBegin(GL_LINES);
62         glVertex2d(-640.0, 0);
63         glVertex2d(640.0, 0);
64     glEnd();
65     glFlush();
66 }
67
68 void selectCenter(int option){
69     X0 = coords[option-1][0]*20;
70     Y0 = coords[option-1][1]*20;
71     radius = coords[option-1][2]*10;
72 }
73
74 vector<pair<int, int>> Midpoint(){
75
76     int x = radius;
77     int y = 0;
78
79     int p = 1 - radius;
80
81     int point_x = x + X0;
82     int point_y = y + Y0;
83
84
85     vector<pair<int, int>> points;
86
87     points.push_back(pair<int, int>(point_x, point_y));
88
89     if( radius<0){
90         point_x = x + X0; point_y = -y + Y0;
91         points.push_back(pair<int, int>(point_x, point_y));
92
93         point_x = y + X0; point_y = x + Y0;

```

```

94         points.push_back(pair<int, int>(point_x, point_y));
95
96         point_x = -y + X0; point_y = x + Y0;
97         points.push_back(pair<int, int>(point_x, point_y));
98     }
99
100     while(x > y){
101         y++;
102         if(p <=0){
103             p += ((2*y) + 1);
104         }
105         else{
106             x--;
107             p += ((2*y) - (2*x) + 1);
108         }
109         if(x < y)
110             break;
111
112         point_x = x + X0; point_y = y + Y0;
113         points.push_back(pair<int, int>(point_x, point_y));
114
115         point_x = -x + X0; point_y = y + Y0;
116         points.push_back(pair<int, int>(point_x, point_y));
117
118         point_x = x + X0; point_y = -y + Y0;
119         points.push_back(pair<int, int>(point_x, point_y));
120
121         point_x = -x + X0; point_y = -y + Y0;
122         points.push_back(pair<int, int>(point_x, point_y));
123
124         if( x != y ){
125             point_x = y + X0; point_y = x + Y0;
126             points.push_back(pair<int, int>(point_x, point_y)
127 );
128
129             point_x = -y + X0; point_y = x + Y0;
130             points.push_back(pair<int, int>(point_x, point_y)
131 );
132
133             point_x = y + X0; point_y = -x + Y0;
134             points.push_back(pair<int, int>(point_x, point_y)
135 );
136
137             point_x = -y + X0; point_y = -x + Y0;

```

```

135         points.push_back(pair<int, int>(point_x, point_y)
136     );
137     }
138     return points;
139 }

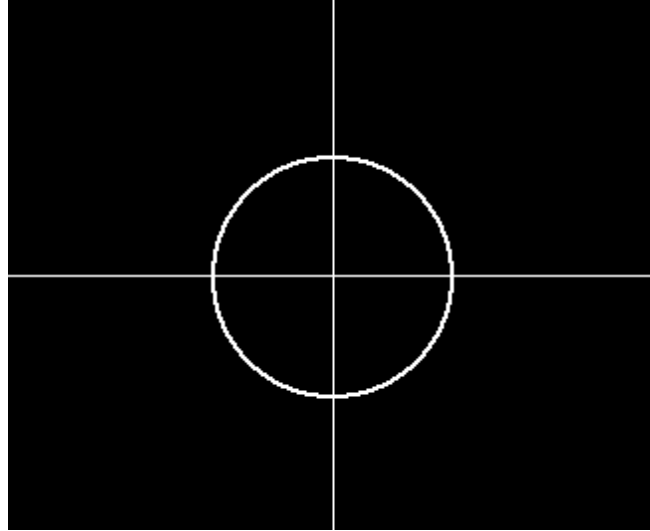
1 #include "Helpers.h"
2
3 void runMainLoop(int val);
4
5 int main( int argc, char* args[] ){
6
7     glutInit( &argc, args );
8
9     glutInitContextVersion( 2, 1 );
10
11     glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
12     glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
13     glutCreateWindow( "OpenGL" );
14
15     int option=0;
16     cout<<"Choose center: (1 for origin, 2 for elsewhere): ";
17     cin>>option;
18
19     selectCenter(option);
20     cout<<"Center: ("<<X0<<", "<<Y0<<)"<<endl;
21     cout<<"Radius: "<<radius<<endl;
22
23     if( !initGL() )
24     {
25         printf( "Unable to initialize graphics library!\n" );
26         return 1;
27     }
28
29     vector<pair<int, int>> points = Midpoint();
30     int count=0;
31     cout<<"Points plotted: "<<endl;
32     for(pair<int, int> p: points){
33         cout<<"("<<p.first<<", "<<p.second<<)"<<" ";
34         count++;
35         if(count==4){
36             count=0;
37             cout<<endl;
38         }
39     }

```

```
40
41     glutDisplayFunc( render );
42
43     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, 0 );
44
45     glutMainLoop();
46
47     return 0;
48 }
49
50 void runMainLoop( int val ){
51     update();
52     render();
53
54     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, val );
55 }
```

Output:

Center at origin:



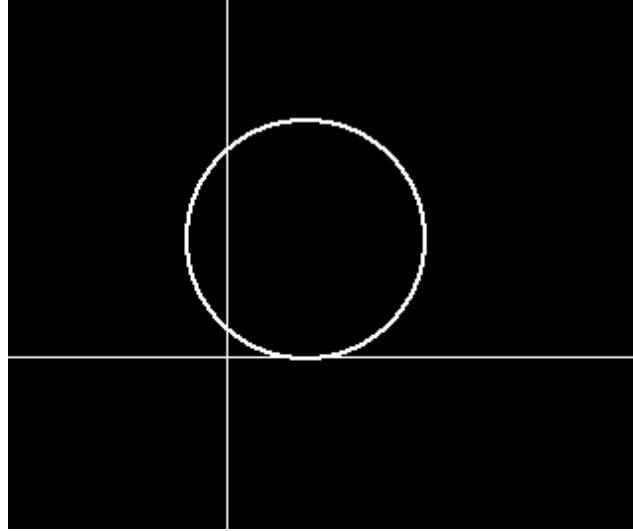
Choose center: (1 for origin, 2 for elsewhere): 1

Center: (0, 0) Radius: 60

Points plotted: (60, 0) (60, 1) (-60, 1) (60, -1) (-60, -1) (1, 60) (-1, 60) (1, -60) (-1, -60) (60, 2) (-60, 2) (60, -2) (-60, -2) (2, 60) (-2, 60) (2, -60) (-2, -60) (60, 3) (-60, 3) (60, -3) (-60, -3) (3, 60) (-3, 60) (3, -60) (-3, -60) (60, 4) (-60, 4) (60, -4) (-60, -4) (4, 60) (-4, 60) (4, -60) (-4, -60) (60, 5) (-60, 5) (60, -5) (-60, -5) (5, 60) (-5, 60) (5, -60) (-5, -60) (60, 6) (-60, 6) (60, -6) (-60, -6) (6, 60) (-6, 60) (6, -60) (-6, -60) (60, 7) (-60, 7) (60, -7) (-60, -7) (7, 60) (-7, 60) (7, -60) (-7, -60) (59, 8) (-59, 8) (59, -8) (-59, -8) (8, 59) (-8, 59) (8, -59) (-8, -59) (59, 9) (-59, 9) (59, -9) (-59, -9) (9, 59) (-9, 59) (9, -59) (-9, -59) (59, 10) (-59, 10) (59, -10) (-59, -10) (10, 59) (-10, 59) (10, -59) (-10, -59) (59, 11) (-59, 11) (59, -11) (-59, -11) (11, 59) (-11, 59) (11, -59) (-11, -59) (59, 12) (-59, 12) (59, -12) (-59, -12) (12, 59) (-12, 59) (12, -59) (-12, -59) (59, 13) (-59, 13) (59, -13) (-59, -13) (13, 59) (-13, 59) (13, -59) (-13, -59) (58, 14) (-58, 14) (58, -14) (-58, -14) (14, 58) (-14, 58) (14, -58) (-14, -58) (58, 15) (-58, 15) (58, -15) (-58, -15) (15, 58) (-15, 58) (15, -58) (-15, -58) (58, 16) (-58, 16) (58, -16) (-58, -16) (16, 58) (-16, 58) (16, -58) (-16, -58)

(58, 17) (-58, 17) (58, -17) (-58, -17) (17, 58) (-17, 58) (17, -58) (-17, -58)
 (57, 18) (-57, 18) (57, -18) (-57, -18) (18, 57) (-18, 57) (18, -57) (-18, -57)
 (57, 19) (-57, 19) (57, -19) (-57, -19) (19, 57) (-19, 57) (19, -57) (-19, -57)
 (57, 20) (-57, 20) (57, -20) (-57, -20) (20, 57) (-20, 57) (20, -57) (-20, -57)
 (56, 21) (-56, 21) (56, -21) (-56, -21) (21, 56) (-21, 56) (21, -56) (-21, -56)
 (56, 22) (-56, 22) (56, -22) (-56, -22) (22, 56) (-22, 56) (22, -56) (-22, -56)
 (55, 23) (-55, 23) (55, -23) (-55, -23) (23, 55) (-23, 55) (23, -55) (-23, -55)
 (55, 24) (-55, 24) (55, -24) (-55, -24) (24, 55) (-24, 55) (24, -55) (-24, -55)
 (55, 25) (-55, 25) (55, -25) (-55, -25) (25, 55) (-25, 55) (25, -55) (-25, -55)
 (54, 26) (-54, 26) (54, -26) (-54, -26) (26, 54) (-26, 54) (26, -54) (-26, -54)
 (54, 27) (-54, 27) (54, -27) (-54, -27) (27, 54) (-27, 54) (27, -54) (-27, -54)
 (53, 28) (-53, 28) (53, -28) (-53, -28) (28, 53) (-28, 53) (28, -53) (-28, -53)
 (53, 29) (-53, 29) (53, -29) (-53, -29) (29, 53) (-29, 53) (29, -53) (-29, -53)
 (52, 30) (-52, 30) (52, -30) (-52, -30) (30, 52) (-30, 52) (30, -52) (-30, -52)
 (51, 31) (-51, 31) (51, -31) (-51, -31) (31, 51) (-31, 51) (31, -51) (-31, -51)
 (51, 32) (-51, 32) (51, -32) (-51, -32) (32, 51) (-32, 51) (32, -51) (-32, -51)
 (50, 33) (-50, 33) (50, -33) (-50, -33) (33, 50) (-33, 50) (33, -50) (-33, -50)
 (49, 34) (-49, 34) (49, -34) (-49, -34) (34, 49) (-34, 49) (34, -49) (-34, -49)
 (49, 35) (-49, 35) (49, -35) (-49, -35) (35, 49) (-35, 49) (35, -49) (-35, -49)
 (48, 36) (-48, 36) (48, -36) (-48, -36) (36, 48) (-36, 48) (36, -48) (-36, -48)
 (47, 37) (-47, 37) (47, -37) (-47, -37) (37, 47) (-37, 47) (37, -47) (-37, -47)
 (46, 38) (-46, 38) (46, -38) (-46, -38) (38, 46) (-38, 46) (38, -46) (-38, -46)
 (46, 39) (-46, 39) (46, -39) (-46, -39) (39, 46) (-39, 46) (39, -46) (-39, -46)
 (45, 40) (-45, 40) (45, -40) (-45, -40) (40, 45) (-40, 45) (40, -45) (-40, -45)
 (44, 41) (-44, 41) (44, -41) (-44, -41) (41, 44) (-41, 44) (41, -44) (-41, -44)
 (43, 42) (-43, 42) (43, -42) (-43, -42) (42, 43) (-42, 43) (42, -43) (-42, -43)

Center at (xc, yc):



Choose center: (1 for origin, 2 for elsewhere): 2
Center: (40, 60) Radius: 60

Points plotted: (100, 60) (100, 61) (-20, 61) (100, 59) (-20, 59) (41, 120) (39, 120) (41, 0) (39, 0) (100, 62) (-20, 62) (100, 58) (-20, 58) (42, 120) (38, 120) (42, 0) (38, 0) (100, 63) (-20, 63) (100, 57) (-20, 57) (43, 120) (37, 120) (43, 0) (37, 0) (100, 64) (-20, 64) (100, 56) (-20, 56) (44, 120) (36, 120) (44, 0) (36, 0) (100, 65) (-20, 65) (100, 55) (-20, 55) (45, 120) (35, 120) (45, 0) (35, 0) (100, 66) (-20, 66) (100, 54) (-20, 54) (46, 120) (34, 120) (46, 0) (34, 0) (100, 67) (-20, 67) (100, 53) (-20, 53) (47, 120) (33, 120) (47, 0) (33, 0) (99, 68) (-19, 68) (99, 52) (-19, 52) (48, 119) (32, 119) (48, 1) (32, 1) (99, 69) (-19, 69) (99, 51) (-19, 51) (49, 119) (31, 119) (49, 1) (31, 1) (99, 70) (-19, 70) (99, 50) (-19, 50) (50, 119) (30, 119) (50, 1) (30, 1) (99, 71) (-19, 71) (99, 49) (-19, 49) (51, 119) (29, 119) (51, 1) (29, 1) (99, 72) (-19, 72) (99, 48) (-19, 48) (52, 119) (28, 119) (52, 1) (28, 1) (99, 73) (-19, 73) (99, 47) (-19, 47) (53, 119) (27, 119) (53, 1) (27, 1) (98, 74) (-18, 74) (98, 46) (-18, 46) (54, 118) (26, 118) (54, 2) (26, 2) (98, 75) (-18, 75) (98, 45) (-18, 45) (55, 118) (25, 118) (55, 2) (25, 2) (98, 76) (-18, 76) (98, 44) (-18, 44) (56, 118) (24, 118) (56, 2) (24, 2) (98, 77) (-18, 77) (98, 43) (-18, 43) (57, 118) (23, 118) (57, 2) (23, 2) (97, 78) (-17, 78) (97, 42) (-17, 42) (58, 117) (22, 117) (58, 3) (22, 3) (97, 79) (-17, 79) (97, 41) (-17, 41) (59, 117) (21, 117) (59, 3)

(21, 3) (97, 80) (-17, 80) (97, 40) (-17, 40) (60, 117) (20, 117) (60, 3) (20, 3)
 (96, 81) (-16, 81) (96, 39) (-16, 39) (61, 116) (19, 116) (61, 4) (19, 4) (96, 82)
 (-16, 82) (96, 38) (-16, 38) (62, 116) (18, 116) (62, 4) (18, 4) (95, 83) (-15,
 83) (95, 37) (-15, 37) (63, 115) (17, 115) (63, 5) (17, 5) (95, 84) (-15, 84) (95,
 36) (-15, 36) (64, 115) (16, 115) (64, 5) (16, 5) (95, 85) (-15, 85) (95, 35)
 (-15, 35) (65, 115) (15, 115) (65, 5) (15, 5) (94, 86) (-14, 86) (94, 34) (-14,
 34) (66, 114) (14, 114) (66, 6) (14, 6) (94, 87) (-14, 87) (94, 33) (-14, 33) (67,
 114) (13, 114) (67, 6) (13, 6) (93, 88) (-13, 88) (93, 32) (-13, 32) (68, 113)
 (12, 113) (68, 7) (12, 7) (93, 89) (-13, 89) (93, 31) (-13, 31) (69, 113) (11,
 113) (69, 7) (11, 7) (92, 90) (-12, 90) (92, 30) (-12, 30) (70, 112) (10, 112)
 (70, 8) (10, 8) (91, 91) (-11, 91) (91, 29) (-11, 29) (71, 111) (9, 111) (71, 9)
 (9, 9) (91, 92) (-11, 92) (91, 28) (-11, 28) (72, 111) (8, 111) (72, 9) (8, 9) (90,
 93) (-10, 93) (90, 27) (-10, 27) (73, 110) (7, 110) (73, 10) (7, 10) (89, 94) (-9,
 94) (89, 26) (-9, 26) (74, 109) (6, 109) (74, 11) (6, 11) (89, 95) (-9, 95) (89,
 25) (-9, 25) (75, 109) (5, 109) (75, 11) (5, 11) (88, 96) (-8, 96) (88, 24) (-8,
 24) (76, 108) (4, 108) (76, 12) (4, 12) (87, 97) (-7, 97) (87, 23) (-7, 23) (77,
 107) (3, 107) (77, 13) (3, 13) (86, 98) (-6, 98) (86, 22) (-6, 22) (78, 106) (2,
 106) (78, 14) (2, 14) (86, 99) (-6, 99) (86, 21) (-6, 21) (79, 106) (1, 106) (79,
 14) (1, 14) (85, 100) (-5, 100) (85, 20) (-5, 20) (80, 105) (0, 105) (80, 15) (0,
 15) (84, 101) (-4, 101) (84, 19) (-4, 19) (81, 104) (-1, 104) (81, 16) (-1, 16)
 (83, 102) (-3, 102) (83, 18) (-3, 18) (82, 103) (-2, 103) (82, 17) (-2, 17)

Objective:

To draw any object using line and circle drawing algorithms.

Code:

```
1 #ifndef LOPENGL_H
2 #define LOPENGL_H
3
4 #include <GL/freeglut.h>
5 #include <GL/gl.h>
6 #include <GL/glu.h>
7 #include <math.h>
8 #include <stdio.h>
9 #include <iostream>
10 #include <vector>
11 #include <ctime>
12 using namespace std;
13
14 #endif
15
16 #ifndef LUTIL_H
17 #define LUTIL_H
18
19 #include "Headers.h"
20
21 //Screen Constants
22 const int SCREEN_WIDTH = 640;
23 const int SCREEN_HEIGHT = 480;
24 const int SCREEN_FPS = 60;
25 const int POINT_SIZE=2;
26
27 bool initGL();
28
29 void update();
30
31 void render();
32
33 void circle(int X0, int Y0, int radius);
```

```

20
21 void line(int x1, int y1, int x2, int y2);
22
23 void lineloop(int x1, int y1, int x2, int y2);
24
25 void solidQuad(int x1, int y1, int x2, int y2);
26
27 void platform();
28
29 void human();
30
31 void head();
32
33 void body();
34
35 void brush();
36
37 void canvas();
38
39 void art();
40
41 void base();
42
43 void snowman();
44
45 #endif

1 #include "Signatures.h"
2
3 bool initGL(){
4     //Initialize Projection Matrix
5     glMatrixMode( GL_PROJECTION );
6     glLoadIdentity();
7     gluOrtho2D(0.0,640.0,0.0,480.0);
8
9     //Initialize Modelview Matrix
10    glMatrixMode( GL_MODELVIEW );
11    glLoadIdentity();
12
13    // glTranslatef( SCREEN_WIDTH / 3.f, SCREEN_HEIGHT / 3.f,
14    0.f );
15
16    //Initialize clear color
17    glClearColor( 0.f, 0.f, 0.f, 1.f );
18
19    glPointSize(POINT_SIZE);

```

```

19     glEnable(GL_POINT_SMOOTH);
20
21     //Check for error
22     GLenum error = glGetError();
23     if( error != GL_NO_ERROR )
24     {
25         printf( "Error initializing OpenGL! %s\n",
26             gluErrorString( error ) );
27         return false;
28     }
29     return true;
30 }
31
32 void update(){
33
34 }
35
36 void render(){
37
38     glClear(GL_COLOR_BUFFER_BIT);
39     platform();
40     human();
41     canvas();
42     art();
43     glFlush();
44 }
45
46 void line(int x1, int y1, int x2, int y2) {
47
48     glBegin(GL_LINES);
49
50     glVertex2d(x1,y1);
51     glVertex2d(x2,y2);
52
53     glEnd();
54 }
55
56 void lineloop(int x1, int y1, int x2, int y2) {
57
58     glBegin(GL_LINE_LOOP);
59
60     glVertex2d(x1,y1);
61     glVertex2d(x2,y1);
62     glVertex2d(x2,y2);

```

```

63     glVertex2d(x1,y2);
64
65     glEnd();
66 }
67
68 void solidQuad(int x1, int y1, int x2, int y2) {
69
70     glBegin(GL_QUADS);
71
72     glVertex2d(x1,y1);
73     glVertex2d(x2,y1);
74     glVertex2d(x2,y2);
75     glVertex2d(x1,y2);
76
77     glEnd();
78 }
79
80 void circle(int X0, int Y0, int radius){
81     int x = radius;
82     int y = 0;
83
84     int p = 1 - radius;
85
86     int point_x = x + X0;
87     int point_y = y + Y0;
88
89
90     glBegin(GL_POINTS);
91
92     glVertex2d(point_x, point_y);
93
94     if( radius<0){
95         point_x = x + X0; point_y = -y + Y0;
96         glVertex2d(point_x, point_y);
97
98         point_x = y + X0; point_y = x + Y0;
99         glVertex2d(point_x, point_y);
100
101         point_x = -y + X0; point_y = x + Y0;
102         glVertex2d(point_x, point_y);
103     }
104
105     while(x > y){
106         y++;
107         if(p <=0){

```

```

108         p += ((2*y) + 1);
109     }
110     else{
111         x--;
112         p += ((2*y) - (2*x) + 1);
113     }
114     if(x < y)
115         break;
116
117     point_x = x + X0; point_y = y + Y0;
118     glVertex2d(point_x, point_y);
119
120     point_x = -x + X0; point_y = y + Y0;
121     glVertex2d(point_x, point_y);
122
123     point_x = x + X0; point_y = -y + Y0;
124     glVertex2d(point_x, point_y);
125
126     point_x = -x + X0; point_y = -y + Y0;
127     glVertex2d(point_x, point_y);
128
129     if( x != y ){
130         point_x = y + X0; point_y = x + Y0;
131         glVertex2d(point_x, point_y);
132
133         point_x = -y + X0; point_y = x + Y0;
134         glVertex2d(point_x, point_y);
135
136         point_x = y + X0; point_y = -x + Y0;
137         glVertex2d(point_x, point_y);
138
139         point_x = -y + X0; point_y = -x + Y0;
140         glVertex2d(point_x, point_y);
141     }
142 }
143 glEnd();
144 }
145
146
147 void platform(){
148     solidQuad(100, 50, 600, 100);
149 }
150
151 void human(){
152     head();

```



```

153     body();
154     brush();
155 }
156
157 void head(){
158     circle(300, 300, 20);
159 }
160
161 void body(){
162     //torso
163     line(300, 280, 300, 175);
164
165     //arm
166     line(300, 260, 275, 235);
167     line(275, 235, 310, 190);
168
169     line(300, 260, 325, 235);
170     line(325, 235, 360, 250);
171
172     //legs
173     line(300, 175, 325, 100);
174     line(300, 175, 275, 100);
175 }
176
177 void brush(){
178     line(345, 258, 385, 230);
179     line(346, 259, 386, 231);
180     line(347, 260, 387, 232);
181
182     line(345, 258, 347, 260);
183     line(385, 230, 387, 232);
184
185     //bristles
186     line(385, 230, 388, 228);
187     line(387, 232, 388, 228);
188 }
189
190
191 void canvas(){
192     lineloop(375, 150, 575, 340);
193
194     line(475, 150, 445, 100);
195     line(475, 150, 505, 100);
196 }
197

```

```

198 void art(){
199     base();
200     snowman();
201 }
202
203 void base(){
204     line(395, 170, 555, 170);
205 }
206
207 void snowman(){
208     //body
209     circle(475, 210, 40);
210     circle(475, 215, 2);
211     circle(475, 225, 2);
212     circle(475, 235, 2);
213
214     line(512, 220, 550, 218);
215     line(532, 219, 549, 208);
216     line(532, 219, 549, 228);
217
218     line(438, 220, 400, 218);
219     line(418, 219, 401, 208);
220     line(418, 219, 401, 228);
221
222     //head
223     circle(475, 270, 20);
224
225     //eyes
226     circle(467, 275, 4);
227     circle(483, 275, 4);
228
229     // circle(475, 265, 2);
230
231     //mouth
232     for(int x=465, y=260; x<475; x+=3, y-=1){
233         circle(x, y, 0);
234     }
235     for(int x=485, y=260; x>475; x-=3, y-=1){
236         circle(x, y, 0);
237     }
238
239
240 }

1 #include "Helpers.h"
2

```

```

3 void runMainLoop(int val);
4
5 int main( int argc, char* args[] ){
6
7     glutInit( &argc, args );
8
9     glutInitContextVersion( 2, 1 );
10
11     glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
12     glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
13     glutCreateWindow( "OpenGL" );
14
15     if( !initGL() )
16     {
17         printf( "Unable to initialize graphics library!\n" );
18         return 1;
19     }
20
21     glutDisplayFunc( render );
22
23     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, 0 );
24
25     glutMainLoop();
26
27     return 0;
28 }
29
30 void runMainLoop( int val ){
31     update();
32     render();
33
34     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, val );
35 }

```

Output:

