Bangladesh University of Business and Technology (BUBT)

Lab Report:02

Subject:- Computer Graphics Lab

Course Code:- CSE 342

Submitted by	**Submitted to**
Name: Mohasina Jannat Moon ID: 17182103092 Intake: 38 Section: 03 Program: B.Sc in CSE	Name: Sweety Lima Lecturer Department of Computer Science and Engineering (BUBT) Rupnagar R/A, Mirpur-2, Dhaka-1216

Date:21-01-2021

Lab No: 02

Lab Task Name: **Implementation of Midpoint Line Drawing Algorithm (for m>1) and Midpoint Circle Algorithm using OpenGL.**

Objective: Here we draw a line by intersecting some point using Midpoint Line Drawing Algorithm and OpenGL. And also draw a Circle using Midpoint Circle Drawing Algorithm and OpenGL.

Midpoint line: Given coordinate of two points A(x1, y1) and B(x2, y2) such that x1 < x2 and y1 < y2. The task to find all the intermediate points required for drawing line AB on the computer screen of pixels. Note that every pixel has integer coordinates.

We have discussed below algorithms for this task.

In this post, Mid-Point Line drawing algorithm is discussed which is a different way to represent Bresenham's algorithm introduced in previous post.

As discussed in previous post, for any given/calculated previous pixel P(X_p,Y_p), there are two candidates for the next pixel closest to the line, E(X_p+1, Y_p) and NE(X_p+1, Y_p+1) (**E** stands for East and **NE** stands for North-East).

Midpoint Circle: The **mid-point** circle drawing algorithm is an algorithm used to determine the points needed for rasterizing a circle.
We use the **mid-point** algorithm to calculate all the perimeter points of the circle in the first octant and then print them along with their mirror points in the other octants. This will work because a circle is symmetric about it's centre.

The algorithm is very similar to the Mid-Point Line Generation Algorithm. Here, only the boundary condition is different.
For any given pixel (x, y), the next pixel to be plotted is either (x, y+1) or (x-1, y+1). This can be decided by following the steps below.

Algorithm: OpenGL (Open Graphics Library) is a cross-platform, hardware-accelerated, language-independent, industrial standard API for producing 3D (including 2D) graphics. Modern computers have dedicated GPU (Graphics Processing Unit) with its own memory to speed up graphics rendering. OpenGL is the software interface to graphics hardware. In other words, OpenGL graphic rendering commands issued by your applications could be directed to the graphic hardware and accelerated.

Midpoint Line Drawing Algorithm:

Step1: Start Algorithm

Step2: Declare variables x_1,x_2,y_1,y_2,dx,dy,m,b,

Step3: Enter values of x_1,x_2,y_1,y_2.
The (x_1,y_1) are co-ordinates of a starting point of the line.
The (x_2,y_2) are co-ordinates of a ending point of the line.

Step4: Calculate $dx = x_2 - x_1$

Step5: Calculate $dy = y_2 - y_1$

Step6: Calculate $m = \frac{dy}{dx}$

Step7: Calculate $b = y_1 - m * x_1$

Step8: Set (x, y) equal to starting point, i.e., lowest point and x_{end}equal to largest value of x.

```
    If $dx < 0$
       then $x = x_2$
    $y = y_2$
            $x_{end} = x_1$
If $dx > 0$
    then $x = x_1$
       $y = y_1$
            $x_{end} = x_2$
```

Step9: Check whether the complete line has been drawn if $x=x_{end}$, stop

Step10: Plot a point at current (x, y) coordinates

Step11: Increment value of x, i.e., $x = x+1$

Step12: Compute next value of y from equation $y = mx + b$

Step13: Go to Step9.

Midpoint Circle Drawing Algorithm:

```
Step1: Put x =0, y =r in equation 2
          We have p=1-r

Step2: Repeat steps while x ≤ y
          Plot (x, y)
          If (p<0)
Then set p = p + 2x + 3
Else
          p = p + 2(x-y)+5
          y =y - 1 (end if)
          x =x+1 (end loop)

Step3: End
```

Source Code:

Midpoint Line Drawing:

```
#include<windows.h>
#include<GL/glut.h>
#include<bits/stdc++.h>
using namespace std;
double X1,Y1,X2,Y2,x,y,m,d;
void plot(void)
{
glClear (GL_COLOR_BUFFER_BIT);

   glColor3f(0.0, 1.0, 0.0);
   int dx = X2 - X1;
   int dy = Y2 - Y1;
   m=dy/dx;
   if(m>1){
   d = dx - (dy/2);
   x = X1;
   y = Y1;
   glBegin(GL_POINTS);
   glVertex2i(x,y);
```

```
glEnd();
while (x < X2)
{
    y++;
    if (d < 0)
        d = d + dx;
    else
    {
        d += (dx - dy);
        x++;
    }
    glBegin(GL_POINTS);
    glVertex2i(x,y);
    glEnd();
}
}
else{
d = dy - (dx/2);
x = X1;
y = Y1;
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
while (x < X2)
{
    x++;
    if (d < 0)
        d = d + dy;
    else
    {
        d += (dy - dx);
        y++;
    }
    glBegin(GL_POINTS);
    glVertex2i(x,y);
```

```
            glEnd();
        }
        }

    glFlush();
    }

    void init(void)
    {
        glClearColor(0.0,0.0,0.0,0.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-100,100,-100,100);
    }
    int main ()
    {
        cout<<"Enter The start point:"<<endl;
        cin>>X1>>Y1;
        cout<<"Enter The end point:"<<endl;
        cin>>X2>>Y2;
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(1000,1000);
        glutInitWindowPosition(100,00);
        glutCreateWindow("Mohasina Jannat Moon");
        init();
        glutDisplayFunc(plot);
        glutMainLoop();
    }
```

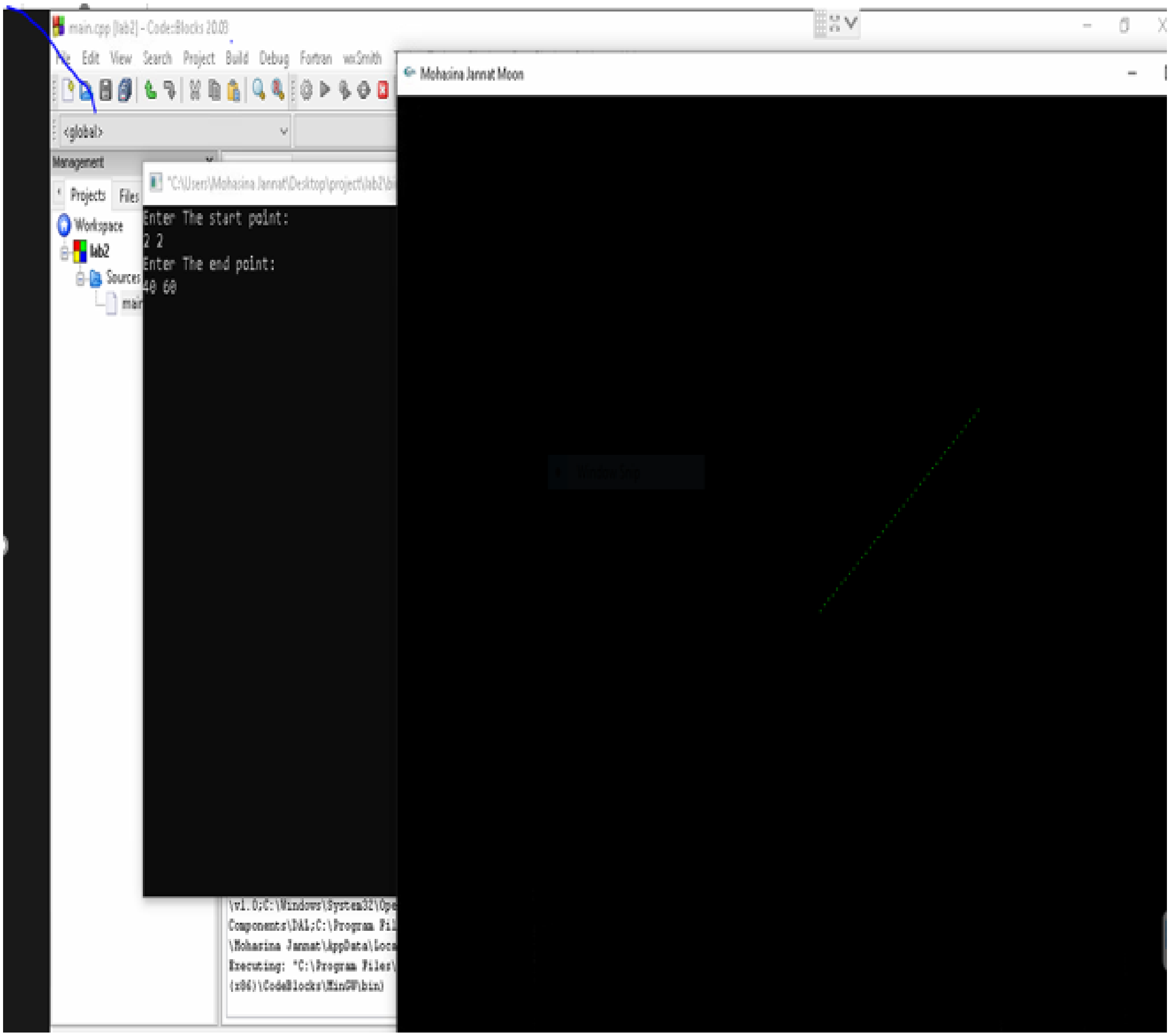
Midpoint Circle Drawing:

```
#include <iostream>
#include <GL/gl.h>
#include <GL/glut.h>
#include<bits/stdc++.h>
```

```
using namespace std;
float x=0,y,p;
int r;
void display(void)
{

glClear (GL_COLOR_BUFFER_BIT);

glEnd();

  glColor3f (1.0,0.0,0.0);
  glBegin(GL_POINTS);
  p=1-r;
  while((x<=y)){
    if(p<0){
     x=x+1;
     y=y;
     p=p+(2*x)+1;
    }



      else{
      x=x+1;
      y=y-1;
      p=p+(2*x)+1-(2*y);
      }
      glVertex3f (((x/100)), ((y/100)),0.0);
      glVertex3f (((y/100)), ((x/100)),0.0);
      glVertex3f ((-(x/100)), (-(y/100)),0.0);
      glVertex3f ((-(x/100)), ((y/100)),0.0);
      glVertex3f (((x/100)), (-(y/100)),0.0);
      glVertex3f (((y/100)), (-(x/100)),0.0);
      glVertex3f ((-(y/100)), (-(x/100)),0.0);
      glVertex3f ((-(y/100)), ((x/100)),0.0);
    }
glEnd();


glFlush ();

}



void init (void)
{glClearColor (1.0, 0.5, 1.0,0.0);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

}

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

   cout<<"Enter  the radius of circle: ";
   cin>>r;
   y=r;
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100, 100);
glutCreateWindow ("Mohasina Jannat Moon");
init ();
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

<u>Output:</u>

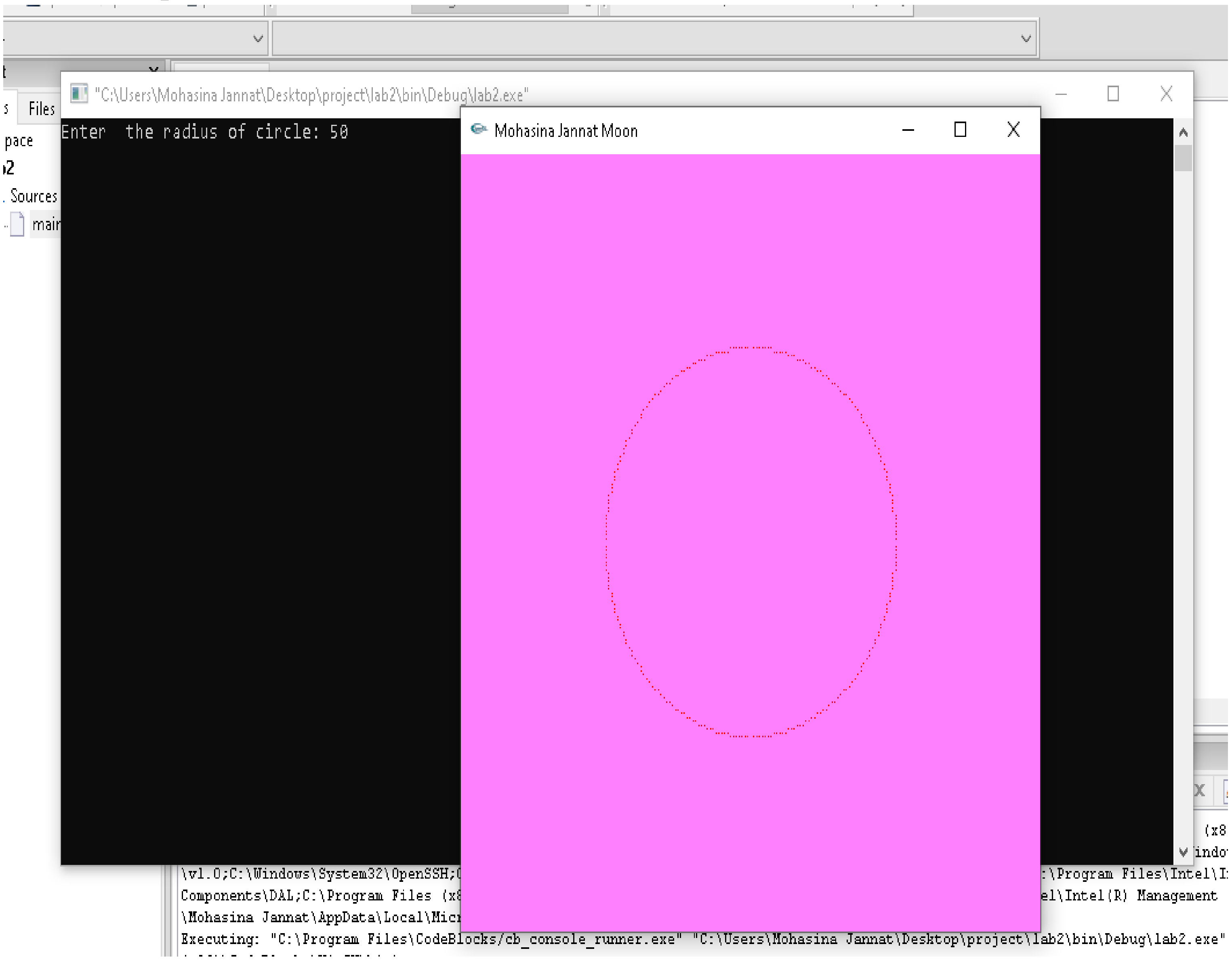
Midpoint line:


main.cpp [lab2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith
<global>
Management
Projects Files
Workspace
lab2
Sources
main
"C:\Users\Mohasina Jannat\Desktop\project\lab2\b
Mohasina Jannat Moon

```
Enter The start point:
2 2
Enter The end point:
40 60
```

```
\v1.0;C:\Windows\System32\Ope
Components\DAL;C:\Program Fil
\Mohasina Jannat\AppData\Loca
Executing: "C:\Program Files\
(x86)\CodeBlocks\MinGW\bin)
```

Midpoint Circle:



```
"C:\Users\Mohasina Jannat\Desktop\project\lab2\bin\Debug\lab2.exe"
Enter  the radius of circle: 50
```

```
\v1.0;C:\Windows\System32\OpenSSH;
Components\DAL;C:\Program Files (x8
\Mohasina Jannat\AppData\Local\Mic
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\Mohasina Jannat\Desktop\project\lab2\bin\Debug\lab2.exe"
```