Bangladesh University of Business & Technology (BUBT)

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

Submission Date:-07/01/2021

Lab No: 01

Lab Task Name: Implementation of Midpoint line drawing algorithm using openGL.

Objective:

OpenGL:- OpenGL is the most widely adopted 2D and 3D graphics API in the industry, bringing thousands of applications to a wide variety of computer platforms. It is window-system and operating-system independent as well as network-transparent. OpenGL enables developers of software for PC, workstation, and supercomputing hardware to create high-performance, visually compelling graphics software applications, in markets such as CAD, content creation, energy, entertainment, game development, manufacturing, medical, and virtual reality. OpenGL exposes all the features of the latest graphics hardware.

Midpoint line drawing algorithm:- 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.

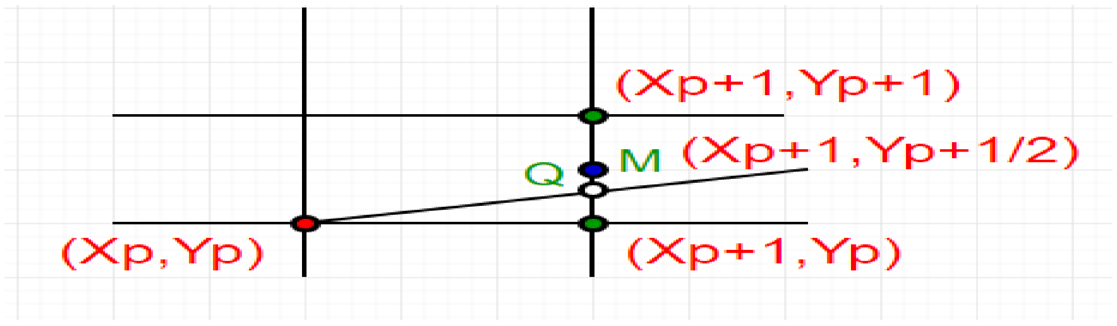
We have discussed below algorithms for this task.

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).
In Mid-Point algorithm we do following.

1. Find middle of two possible next points. Middle of $E(X_p+1, Y_p)$ and $NE(X_p+1, Y_p+1)$ is $M(X_{p+1}, Y_p+1/2)$.
2. If M is above the line, then choose E as next point.

3. If M is below the line, then choose NE as next point.



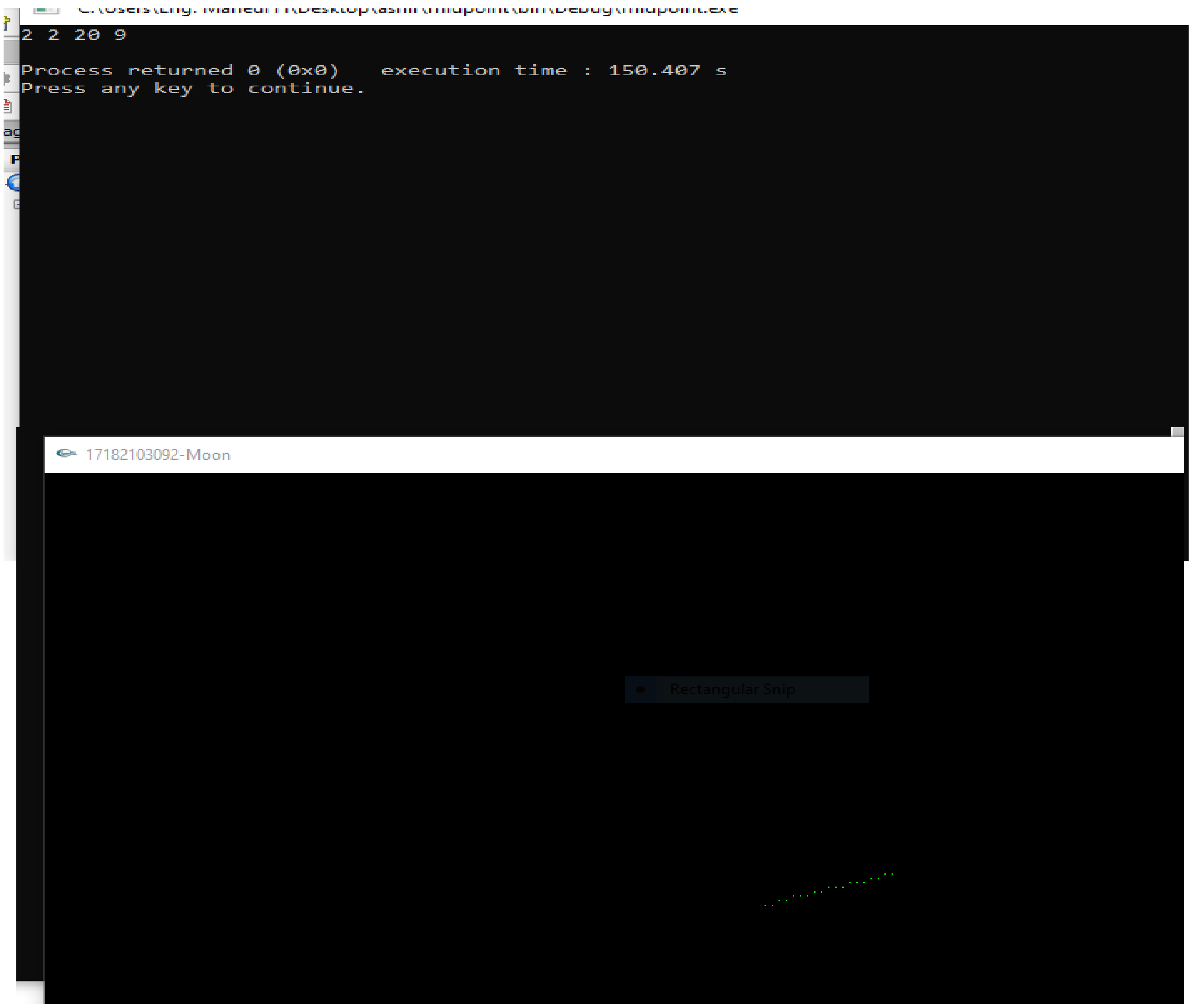
<u>Code:</u>

```
#include<windows.h>
#include<GL/glut.h>
#include<bits/stdc++.h>
using namespace std;
double X1,Y1,X2,Y2,x,y,m;
void plot(void)
{
glClear (GL_COLOR_BUFFER_BIT);
   glColor3f(0.0, 1.0, 0.0);
   int dx = X2 - X1;
   int dy = Y2 - Y1;
   int d = dy - (dx/2);
   int 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 (){
    cin>>X1>>Y1>>X2>>Y2;
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1000,1000);
    glutInitWindowPosition(100,00);
    glutCreateWindow("17182103092-Moon");
    init();
    glutDisplayFunc(plot);
    glutMainLoop();
}
```

Output:



```
2 2 20 9

Process returned 0 (0x0)   execution time : 150.407 s
Press any key to continue.
```