Lab Report

Week 2

Jitendra Kumar , 1401CS19

30/01/2017

Title

- Draw a line in OpenGL and Matlab using :
  1). Basic Digital Differential Analyzer (DDA) Algorithm (both Simple and Symmetric DDA)
  2). Bresenham's mid-point Line Drawing Algorithm

Procedure

OpenGL

1). Draw a line using simple and symmetric DDA Line Algorithm.

- Create a C file and name it as *dda_lines.c*.
- Define global variables to store coordinates of points of a line and option to choose between Simple and Symmetric DDA.
- Following algorithm to compute incremental values of x and y in Simple DDA :

```
length = abs(x_2-x_1);
if(length < abs(y_2-y_1))
        length = abs(y_2-y_1);
x_increment = (x_2-x_1)/length;
y_increment = (y_2-y_1)/length;
```

- Following algorithm to compute incremental values of x and y in Symmetric DDA :

```
x_increment = x_2-x_1 ;
y_increment = y_2-y_1 ;
while(abs(x_increment)>=1 || abs(y_increment)>=1)
{
        x_increment /= 2;
        y_increment /= 2;
}
```

- Following algorithm to plot points in Simple and Symmetric DDA :

```
x = x_1;
y = y_1;
while x < x_2
begin
        plot(round(x),round(y));
        x = x + x_increment;
        y = y + y_increment;
end
```

- Following is the final code for Simple and Symmetric DDA Line Algorithm :

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

double x_1 , x_2 , y_1 , y_2 , x_increment , y_increment ;
double length ;
```

```
displayLine(void)
{
        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glBegin(GL_LINES);
                glColor3f(0.0f, 0.0f, 0.0f);
                glVertex2f(0.0f,1.0f);
                glVertex2f(0.0f,-1.0f);
                glVertex2f(1.0f,0.0f);
                glVertex2f(-1.0f,0.0f);
        glEnd();
        glBegin(GL_LINE_STRIP);
        double x = round(x_1);
        double y = round(y_1);
        while(x <= x_2)
        {
                double xf = floor(x);
                double yf = floor(y);
                xf/=100;
                yf/=100;
                glVertex3f(xf,yf,1.0f);
                x += x_increment;
                y += y_increment;
        }
        glEnd();
        glFlush();
        glutSwapBuffers();
}

int main(int argc, char const *argv[])
{
        int flag=0;
        printf("Enter  the  coordinates  of first point  : ");
        scanf("%lf %lf",&x_1,&y_1);
        printf("Enter  the  coordinates  of Second point : ");
        scanf("%lf %lf",&x_2,&y_2);
        printf("Enter choice\n\t1 : Draw Line using simple DDA\n\t2 : Draw Line using Symmetric DDA\n");
        printf("Your choice : ");
        scanf("%d\n",&flag);
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

        if(flag==1)
        {
                length = abs(x_2-x_1);
                if(length < abs(y_2-y_1))
                        length = abs(y_2-y_1);
                x_increment = (x_2-x_1)/length;
                y_increment = (y_2-y_1)/length;
                glutCreateWindow("Demonstrating Simple DDA");

        }
        else if(flag==2)
        {
                x_increment = x_2-x_1 ;
                y_increment = y_2-y_1 ;
                while(abs(x_increment)>=1 || abs(y_increment)>=1)
                {
                        x_increment /= 2;
                        y_increment /= 2;
                }
                glutCreateWindow("Demonstrating Symmetric DDA");
        }
        else
        {
                printf("Please try with correct input\n");
                exit(1);
        }
        glutDisplayFunc(displayLine);
        glutMainLoop();
        return 0;
}
```

- Compile and run the executable file in terminal by typing in the following commands :
  (a) *gcc dda_lines.c -lGL -lGLU -lglut -lm*
  (b) *./a.out*

2). Draw a line using Bresenham's Line Drawing Algorithm :

- Create a C file and name it as *bresenhams_line.c*.
- Define global variables to define coordinates of line and flag to find octant for line.
- Following algorithm to compute values for coordinates of line :

```
x := x_1;
y := y_1;
dx := x_2-x_1;
dy := y_2-y_1;
d := 2dy - dx;
while x < x_2
begin
        plot(x,y);
        if d<=0 /* Choose E */
                d = d + 2dy;
        else
        /* Choose NE */
                d = d + 2(dy - dx);
        y = y + 1;
        endif
        x = x + 1;
end
```

- Following is the final code for Bresenham's Line Drawing Algorithm :

```
#include <stdio.h>
#include <math.h>
#include <GL/glut.h>
int x_1 , x_2 , y_1 , y_2 , flag , y_increment , d , d1 , d2 ;

void displayLine(void)
{
        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glBegin(GL_LINES);
                glColor3f(0.0f, 0.0f, 0.0f);
                glVertex2f(0.0f,1.0f);
                glVertex2f(0.0f,-1.0f);
                glVertex2f(1.0f,0.0f);
                glVertex2f(-1.0f,0.0f);
        glEnd();
        glBegin(GL_LINE_STRIP);
        glColor3f(0,0,0);
        double x = x_1, y = y_1;
        while(x < x_2)
        {
                glVertex3f(x/100.0,y/100.0,1.0f);
                if(d<=0)
                        d += d1;
                else
                {
                        d += d2;

                        if(flag)
                                y += y_increment;
                        else
                                x++;
                }
                if(flag)
                        x++;
                else
                        y += y_increment;
        }
        glEnd();
        glFlush();
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
        printf("Enter  the  coordinates  of first point  : ");
        scanf("%d %d",&x_1,&y_1);
        printf("Enter  the  coordinates  of Second point : ");
        scanf("%d %d",&x_2,&y_2);
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        double slope = (y_2-y_1)/(x_2-x_1);
        if(slope <=1 && x_1 >=0 && x_2 >= 0 && y_1 >=0 && y_2 >= 0)
```

```
        {
                int dx = x_2-x_1;
                int dy = abs(y_2-y_1);
                if(y_2 < y_1)
                        y_increment = -1;
                else
                        y_increment = 1;
                if(dy>dx)
                {
                        flag = 0;
                        d = 2*dx - dy;
                        d1 = 2*dx;
                        d2 = 2*(dx-dy);
                }
                else
                {
                        flag = 1;
                        d = 2*dy - dx;
                        d1 = 2*dy;
                        d2 = 2*(dy-dx);
                }
        }
        else
        {
                printf("Please enter the point  in first octant only\n");
                exit(1);
        }
        glutCreateWindow("Bresenham Line Drawing");
        glutDisplayFunc(displayLine);
        glutMainLoop();
        return 0;
}
```

- Compile and run the executable file in terminal by typing in the following commands :
  (a) gcc bresenhams_line.c -lGL -lGLU -lglut -lm
  (b) ./a.out

MatLab

1). Draw a line using Simple and Symmetric DDA Algorithm :

- Open a new Script and contruct a function dda_lines(flag). The flag is passed as an argument, which is 0 for Simple DDA and 1 for Symmetric DDA. The script prompts user for coordinates of the starting and ending points of the line.
- Following is the Matlab Script Code for DDA Algorithm :

```
function [] = dda_lines(flag)
        x1 = input("Enter x-coordinate of first Point  : ");
        y1 = input("Enter y-coordinate of first Point  : ");
        x2 = input("Enter x-coordinate of second Point : ");
        y2 = input("Enter y-coordinate of second Point : ");
        dx = x2-x1;
        dy = y2-y1;
        if flag==0

                length = abs(x2-x1);

                if length < abs(y2-y1)
                        length = abs(y2-y1);
                end

                dx = dx./length;
                dy = dy./length;
        end

        if flag==1
                while abs(dx)>=1 | abs(dy)>=1
                        dx = dx./2;
                        dy = dy./2;
                end
        end

        x = x1;
```

```
        y = y1;

        px = [x];
        py = [y];

        while x<x2
                x = x+dx;
                y = y+dy;
                px = [px round(x)];
                py = [py round(y)];
        end
        plot(px,py);
```

2). Draw a Line using Bresenham's Line Drawing Algorithm :

- Open a new Script and contruct a function bresenhams_line(). The script prompts user for input of the starting and ending pointś coordinates of the line.
- Following is the Matlab Script Code for Bresenham's Line Drawing Algorithm :

```
function [] = bresenhams_line()
        x1 = input("Enter x-coordinate of first Point  : ");
        y1 = input("Enter y-coordinate of first Point  : ");
        x2 = input("Enter x-coordinate of second Point : ");
        y2 = input("Enter y-coordinate of second Point : ");

        dx = abs(x2-x1);
        dy = abs(y2-y1);
        y_increment = 1;
        d = dy.*2 - dx;
        d1 = dy.*2;
        d2 = (dy-dx).*2;
        flag = 1;

        if y2 < y1
                y_increment = -1;
        end

        if dy > dx
                flag = 0;
                d = dx.*2 - dy;
                d1 = dx.*2;
                d2 = (dx-dy).*2;
        end

        x = x1;
        y = y1;
        px = [x];
        py = [y];

        while x < x2
                if d<=0
                        d = d + d1;
                else
                        d = d + d2;

                        if flag==1
                                y = y + y_increment;
                        end

                        if flag==0
                                x = x + 1;
                        end
                end

                if flag==1
                        x = x + 1;
                end

                if flag==0
                        y = y + y_increment;
                end

                px = [px x];
                py = [py y];
        end
plot(px,py);
```

Output

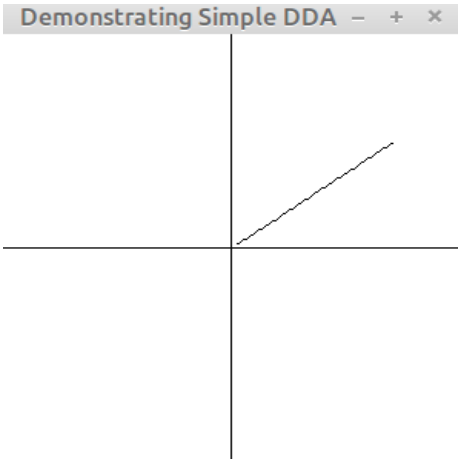

FIGURE 1 – Draw Line using Simple DDA in OpenGL

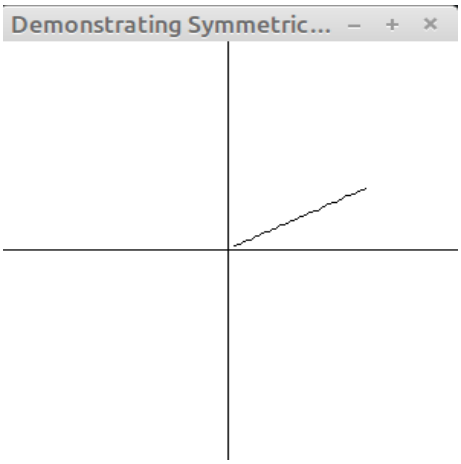

FIGURE 2 – Draw Line using Symmetric DDA in OpenGL

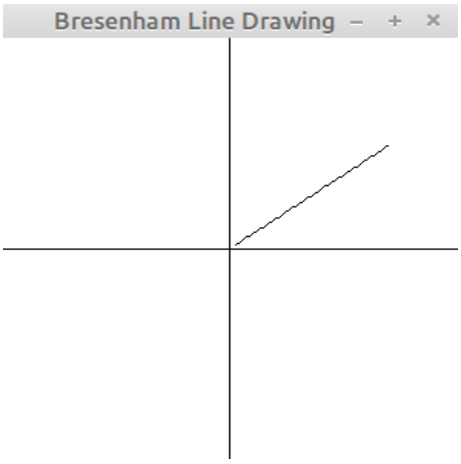

FIGURE 3 – Draw Line using Bresenham Algorithm in OpenGL

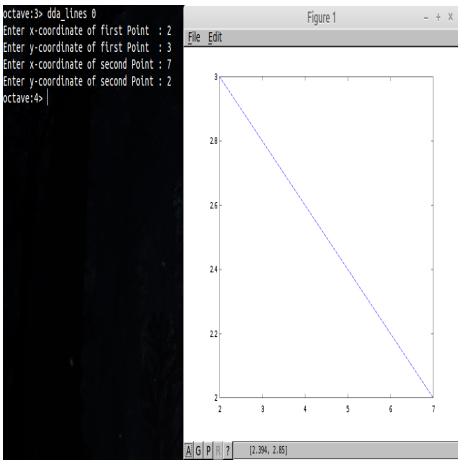

FIGURE 4 – Draw Line using Simple DDA in Matlab

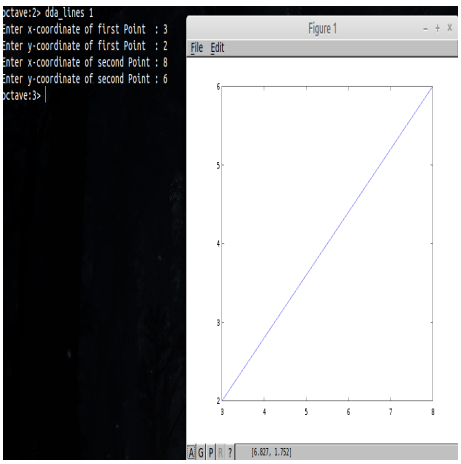

FIGURE 5 – Draw Line using Symmetric DDA in Matlab

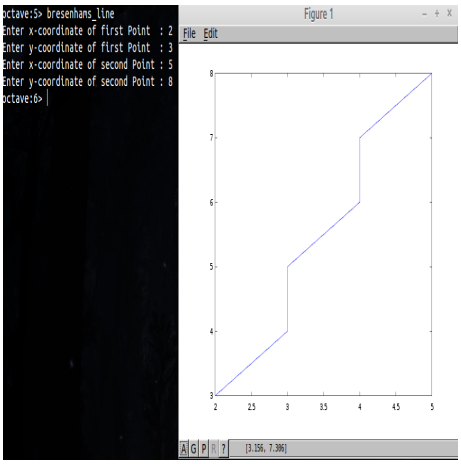

FIGURE 6 – Draw Line using Bresenham Algorithm in Matlab