Report : 01

Experiment No : 01

Experiment Name : Programs on scan conversion of the line using DDA, and Bresenham line drawing algorithm.

DDA Algorithm :

Aim : DDA Algorithm is the simplest line drawing algorithm. Given the starting and ending coordinates of a line, DDA Algorithm attempts to generate the points between the starting and ending coordinates. DDA Line Algorithm use floating point, i.e… Real Arithmetic.

Necessary Apparatus :

1. Hardware : Mouse, Monitor/Screen, RAM (Random Access Memory), CPU (Central Processing Unit), Hard Disk Drive (HDD), CD Rom.

2. Software: Codeblocks.

3. Compiler : MINGW.

4. Software Interface : OpenGL (Open Graphics Library).

5 . The OpenGL Utility Toolkit : GLUT.

Implementation :

Implementation of DDA line algorithm :

```
#include<stdio.h>
#include <GL/gl.h>
#include <GL/glut.h>
float x1,y1,x2,y2,m,i,j;
float dx,dy;
void display(void)
{
/* clear all pixels */
glClear (GL_COLOR_BUFFER_BIT);
/* draw white polygon (rectangle) with corners at
* (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0)
*/
glEnd();

glColor3f (0.0, 1.0, 0.0);
glBegin(GL_POINTS);
//write your code here

if(m>0 && m<=1)
{
```

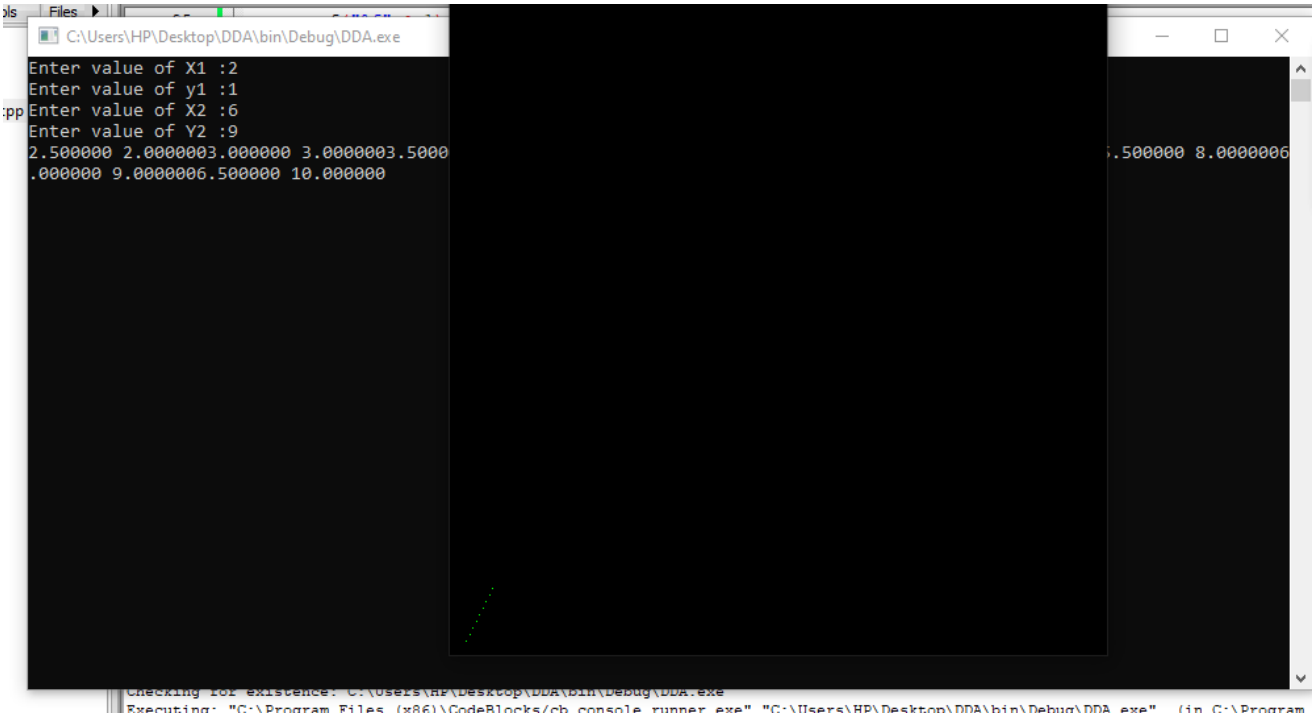
```
while(x1<=x2 && y1<=y2)
{
    x1=x1+1;
    y1=y1+m;
    glVertex3f(x1/100,y1/100,0.0);
   printf("%f %f",x1,y1);

}
}
else if(m>1)
{
    while(x1<=x2 && y1<=y2)
{
    x1=x1+(1/m);
    y1=y1+1;
   glVertex3f(x1/100,y1/100,0.0);
   printf("%f %f",x1,y1);
}
}

else if(m>-1 && m<=0)
{
    while(x1>=x2 && y1>=y2)
```

```
{
    x1=x1-1;
    y1=y1-m;
   glVertex3f(x1/100,y1/100,0.0);
   printf("%f %f",x1,y1);
}
}
else if(m<-1)

  {

    while(x1>=x2 && y1>=y2)
{
    x1=x1-(1/m);
    y1=y1-1;
    glVertex3f(x1/100,y1/100,0.0);
    printf("%f %f",x1,y1);
}
  }

glEnd();
```

```
/* don't wait!
* start processing buffered OpenGL routines
*/
glFlush ();
}
void init (void)
{
/* select clearing (background) color */
glClearColor (0.0, 0.0, 0.0, 0.0);
/* initialize viewing values */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
/*
* Declare initial window size, position, and display mode
* (single buffer and RGBA). Open window with "hello"
* in its title bar. Call initialization routines.
* Register callback function to display graphics.
* Enter main loop and process events.
*/
int main(int argc, char** argv)
{
```

```
    //glVertex3f(x1/100,y1/100,0.0);write your code here
    printf("Enter value of X1 :");
    scanf("%f",&x1);
    printf("Enter value of y1 :");
    scanf("%f",&y1);
    printf("Enter value of X2 :");
    scanf("%f",&x2);
    printf("Enter value of Y2 :");
    scanf("%f",&y2);
    dx=x2-x1;
    dy=y2-y1;
    m=dy/dx;



glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100, 100);
glutCreateWindow ("hello");
init ();
glutDisplayFunc(display);
glutMainLoop();
```

```
return 0; /* ISO C requires main to return int. */
}
```



Output of DDA Algorithm

Bresenham Algorithm :

Aim : Bresenham's line algorithm is a line drawing algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations.

Necessary Apparatus :

1. Hardware : Mouse, Monitor/Screen, RAM (Random Access Memory), CPU (Central Processing Unit), Hard Disk Drive (HDD), CD Rom.

2. Software: Codeblocks.

3. Compiler : MINGW.

4. Software Interface : OpenGL (Open Graphics Library).

5 . The OpenGL Utility Toolkit : GLUT.

Implementation of Bresenham line algorithm :

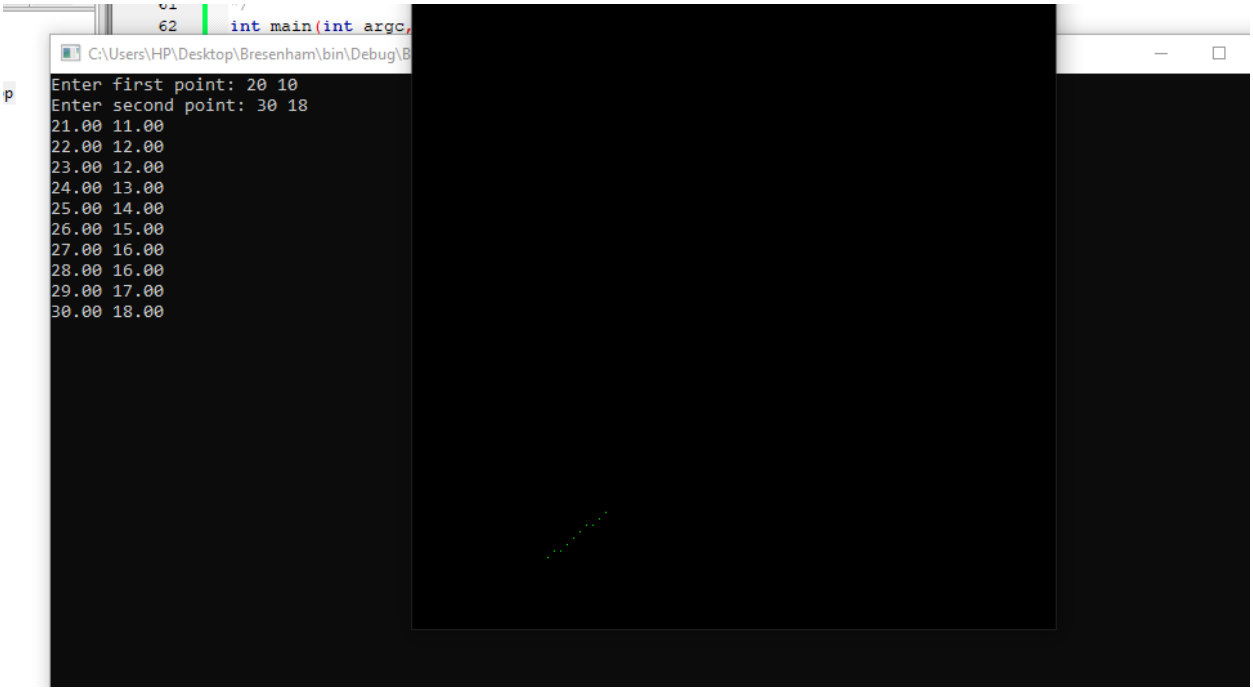
Implementation :

```
#include <stdio.h>
#include <GL/gl.h>
#include <GL/glut.h>
float x1,y1,x2,y2,m,i,j,p;
int dx=0,dy=0;
void display(void)
{
/* clear all pixels */
glClear (GL_COLOR_BUFFER_BIT);
/* draw white polygon (rectangle) with corners at
* (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0)
*/
glEnd();

    glColor3f (0.0, 1.0, 0.0);
    glBegin(GL_POINTS);
    p=(2*dy)-dx;
    for(i=x1,j=y1;i<=x2,j<=y2; ){
```

```
        if(p>=0){
            i=i+1;
            j=j+1;
            if((i>x2)||(j>y2)){
                break;
            }
            printf("%0.2f %0.2f\n",i,j);
            glVertex3f ((i/100), (j/100), 0.0);
            p=p+(2*dy)-(2*dx);
            }
            else if(p<0){
            i=i+1;
            if((i>x2)||(j>y2)){
                break;
            }
            printf("%0.2f %0.2f\n",i,j);
            glVertex3f ((i/100), (j/100), 0.0);
            p=p+(2*dy);
            }
        }
glEnd();

/* don't wait!
```

```
* start processing buffered OpenGL routines
*/
glFlush ();
}
void init (void)
{
/* select clearing (background) color */
glClearColor (0.0, 0.0, 0.0, 0.0);
/* initialize viewing values */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
/*
* Declare initial window size, position, and display mode
* (single buffer and RGBA). Open window with "hello"
* in its title bar. Call initialization routines.
* Register callback function to display graphics.
* Enter main loop and process events.
*/
int main(int argc, char** argv)
{
```

```
    printf("Enter first point: ");
    scanf("%f %f",&x1,&y1);
    printf("Enter second point: ");
    scanf("%f %f",&x2,&y2);
    dx=x2-x1;
    dy=y2-y1;
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100, 100);
glutCreateWindow ("hello");
init ();
glutDisplayFunc(display);
glutMainLoop();
return 0; /* ISO C requires main to return int. */
}
```



Output of Bresenham Algorithm