**Session 3: Implement DDA algorithm (forward) in OpenGL**

**Intended Learning Outcome:**

1. Students will be able to implement DDA algorithm (forward).
2. Students will be able to draw a line using two endpoints with DDA algorithm.

**Expected Skills:**

1. Calculating any point between two endpoints of a line.
2. Clear idea about drawing a line in first coordinate.

**Tools Required:**

1. CodeBlocks
2. OpenGL and GLUT using CodeBlocks.

**Session Detail:**

***Line DDA Algorithm:***

******

1. The digital differential analyzer(DDA) is a scan conversion line algorithm based on calculation

either Dy or Dx.

1. The line at unit intervals is one coordinate and determine corresponding integer values nearest line for the other coordinate.
2. Consider first a line with positive slope.

**Step: 1**

If the slope is less than or equal to 1, the unit x intervals Dx=1 and compute each successive y values.

Dx=1

m = Dy / Dx

m = ( y2-y1 ) / 1

m = ( yk+1 – yk ) /1

yk+1 = yk + m -------- ( 6 )

o Subscript k takes integer values starting from 1, for the first point and increment by 1 until

the final end point is reached.

1. m->any real numbers between 0 and 1
2. Calculate y values must be rounded to the nearest integer

**Step: 2**

If the slope is greater than 1, the roles of x any y at the unit y intervals Dy=1 and compute each successive x values.

Dy=1

m= Dy / Dx

m= 1/ ( x2-x1 )

m = 1 / ( xk+1 – xk )

|  |  |
| --- | --- |
| xk+1 = xk + ( 1 / m )------- | ( 7 ) |

1. Equation 6 and Equation 7 that the lines are to be processed from left end point to the right end point.

After getting DDA algorithm’s (forward) concept, students will write a program to implement DDA algorithm.

**Code for DDA Algorithm (forward) implementation:**

****

#include <GL/gl.h>

#include <GL/glut.h>

float x1,y1,x2,y2,m,i,j;

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) \*/

glColor3f (0.0, 1.0, 0.0);

glBegin(GL\_POINTS);

if(dx==1){

for(i=x1,j=y1;i<=x2,j<=y2;i=i+1,j=j+m){

glVertex3f ((i/100), (j/100), 0.0);

}

}

else if(dy==1){

for(i=x1,j=y1;i<=x2,j<=y2;i=(i+(1/m)),j=j+1){

glVertex3f ((i/100), (j/100), 0.0);

glVertex3f ((i/100), (j/100), 0.0);

}

}

Copyright © Daffodil International University, 2017 17

**Department of Computer Science and Engineering**

**CSE 422: Computer Graphics Lab Credits: 1.0** **Lab Manual v.2017.04**

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); /\*\*

gluOrtho2D(-300, 300, 0, 600);

\*/



}

/\*

* 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);

m=(y2-y1)/(x2-x1);

if((x1>x2)&&(y1>y2)){

dx=-1;

}

else{

dx=1;

}

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (500, 500);

glutInitWindowPosition (100, 100);

glutCreateWindow ("DDA algorithm implementation");

init ();

glutDisplayFunc(display);

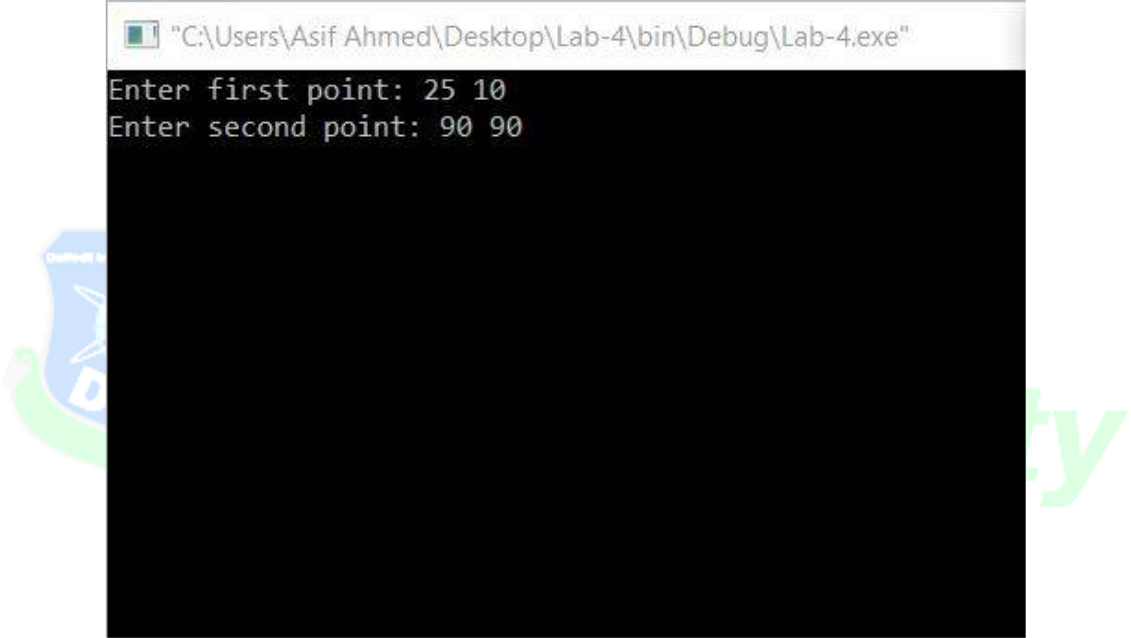
glutMainLoop();

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

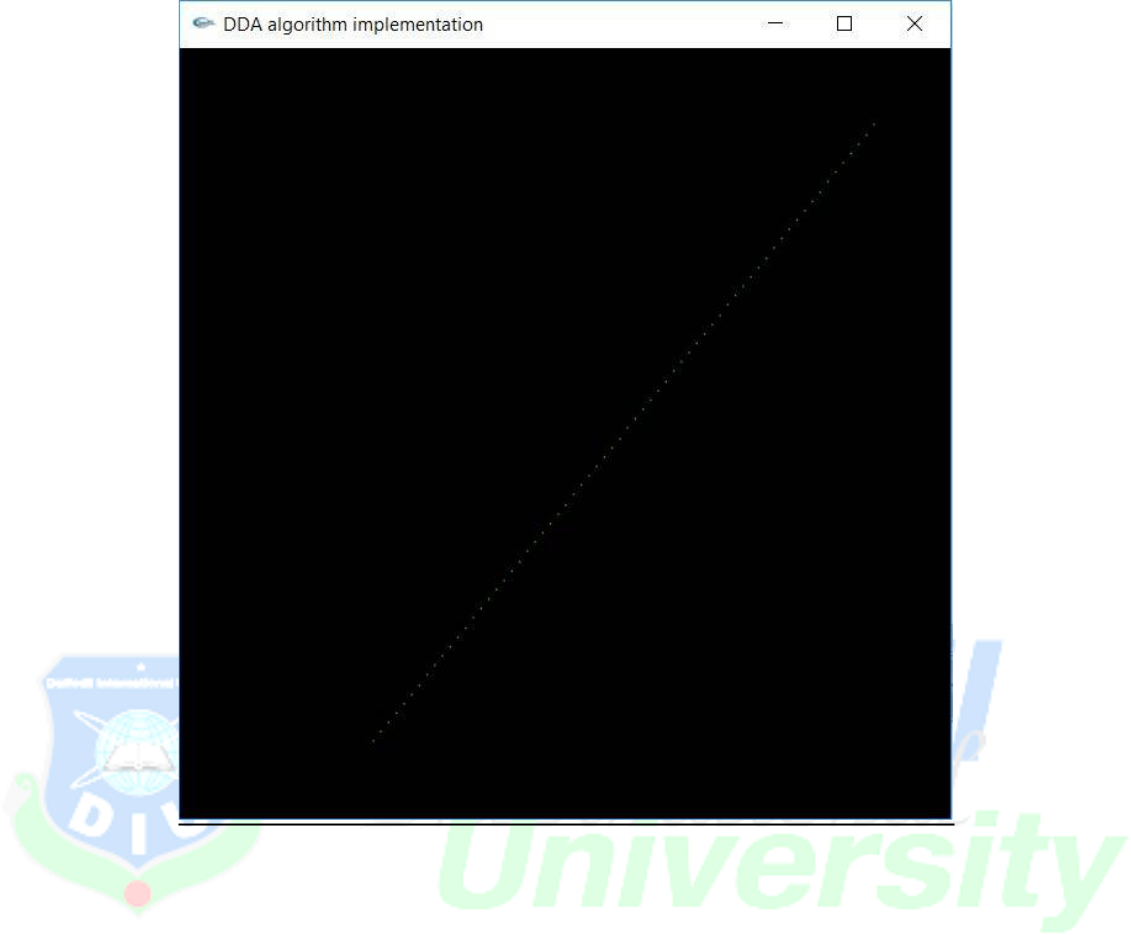
}

**Sample input:**

N.B.: (0<=x<100) and (0<=y<100).



**Sample output:**

****