**Assignment No. : 2**

2. Implement DDA and Bresenham line drawing algorithm to draw: i) Simple Line ii) Dotted Line

iii) Dashed Line iv) Solid line ;using mouse interface Divide the screen in four quadrants with

center as (0, 0). The line should work for all the slopes positive as well as negative.

|  |
| --- |
| **Aim** |
| Draw the following pattern using any Line drawing algorithms. |

|  |  |
| --- | --- |
| **Objective(s)** | |
| **1** | To Learn DDA line drawing algorithm |
| **2** | To learn Bresenham line drawing algorithm |

|  |
| --- |
| **Theory** |
| A line connects two points. It is a basic element in graphics. To draw a line, you need two points between which you can draw a line. In the following three algorithms, we refer the one point of line as X0,Y0X0,Y0 and the second point of line as X1,Y1X1,Y1.  **DDA Algorithm**  Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.  **Step 1** − Get the input of two end points (X0,Y0)(X0,Y0) and (X1,Y1)(X1,Y1).  **Step 2** − Calculate the difference between two end points.  dx = X1 - X0  dy = Y1 - Y0  **Step 3** − Based on the calculated difference in step-2, you need to identify the number of steps to put pixel. If dx > dy, then you need more steps in x coordinate; otherwise in y coordinate.  if (absolute(dx) > absolute(dy))  Steps = absolute(dx);  else  Steps = absolute(dy);  **Step 4** − Calculate the increment in x coordinate and y coordinate.  Xincrement = dx / (float) steps;  Yincrement = dy / (float) steps;  **Step 5** − Put the pixel by successfully incrementing x and y coordinates accordingly and complete the drawing of the line.  for(int v=0; v < Steps; v++)  {  x = x + Xincrement;  y = y + Yincrement;  putpixel(Round(x), Round(y));  }  #include<GL/glut.h>  #include <GL/gl.h>  #include<stdlib.h>  #define ROUND(x) ((int)(x+0.5))  int xa,xb,ya,yb;  void display (void)  {  int dx=xb-xa,dy=yb-ya,steps,k;  float xIncrement,yIncrement,x=xa,y=ya;  glClear (GL\_COLOR\_BUFFER\_BIT);  glColor3f (1.0, 0.0, 0.0);  if(abs(dx)>abs(dy))  steps=abs(dx);  else steps=abs(dy);  xIncrement=dx/(float)steps;  yIncrement=dy/(float)steps;  glBegin(GL\_POINTS);  glVertex2s(ROUND(x),ROUND(y));  for(k=0;k<steps;k++)  {  x+=xIncrement;  y+=yIncrement;  glVertex2s(ROUND(x),ROUND(y));  printf("%lf %lf\n",x,y);  }  glColor3f (1.0, 1.0, 1.0);  for(int i=-100 ; i<=100 ; i++)  {  glVertex2s(i,0);  glVertex2s(0,i);  }  for(int i=-2; i<=2 ; i++)  {  glVertex2s(95+i,4+i);  glVertex2s(95-i,4+i);  }  for(int i=0; i<=2 ; i++)  {  glVertex2s(4+i,95+i);  glVertex2s(4-i,95+i);  glVertex2s(4,95-i);  }  glEnd();  glFlush();  }  void init(void)  {  glClearColor (0.0, 0.0, 0.0, 0.0);  glOrtho(-100.0, 100.0, -100.0, 100.0, -1.0, 1.0);  }  int main(int argc, char\*\* argv)  {  printf("Enter the points\n");  scanf("%d %d %d %d",&xa,&ya,&xb,&yb);  glutInit(&argc, argv);  glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);  glutInitWindowSize (500, 500);  glutInitWindowPosition (100, 100);  glutCreateWindow ("Simple DDA ");  init ();  glutDisplayFunc(display);  glutMainLoop();  return 0;  }  **Difference Between DDA Line Drawing Algorithm and Bresenhams Line Drawing Algorithm**   |  |  |  | | --- | --- | --- | |  | **Digital Differential Analyzer** | Bresenhams Line Drawing Algorithm | | **Arithmetic** | DDA algorithm uses **floating points** i.e. **Real Arithmetic**. | Bresenhams algorithm uses**fixed points** i.e. **Integer Arithmetic**. | | **Operations** | DDA algorithm uses **multiplication** and **division** in its operations. | Bresenhams algorithm uses only **subtraction** and **addition** in its operations. | | **Speed** | DDA algorithm is rather**slow**ly than Bresenhams algorithm in line drawing because it uses real arithmetic (floating-point operations). | Bresenhams algorithm is faster than DDA algorithm in line drawing because it performs only addition and subtraction in its calculation and uses only integer arithmetic so it runs significantly**faster**. | | **Accuracy & Efficiency** | DDA algorithm is not as accurate and efficient as Bresenham algorithm. | Bresenhams algorithm is more efficient and much accurate than DDA algorithm. | | **Drawing** | DDA algorithm can draw circles and curves but that are not as accurate as Bresenhams algorithm. | Bresenhams algorithm can draw circles and curves with much more accuracy than DDA algorithm. | | **Round Off** | DDA algorithm round off the coordinates to integer that is nearest to the line. | Bresenhams algorithm does not **round off**but takes the incremental value in its operation. | | **Expensive** | DDA algorithm uses an enormous number of floating-point multiplications so it is expensive. | Bresenhams algorithm is less expensive than DDA algorithm as it uses only addition and subtraction. |   **Bresenham’s Line Generation**  The Bresenham algorithm is another incremental scan conversion algorithm. The big advantage of this algorithm is that, it uses only integer calculations. Moving across the x axis in unit intervals and at each step choose between two different y coordinates.  For example, as shown in the following illustration, from position (2, 3) you need to choose between (3, 3) and (3, 4). You would like the point that is closer to the original line.  Bresenham’s Line Generation  At sample position Xk+1,Xk+1, the vertical separations from the mathematical line are labelled as dupperdupper and dlowerdlower.  dupper and dlower  From the above illustration, the y coordinate on the mathematical line at xk+1xk+1is −  Y = m(XkXk+1) + b  So, dupperdupper and dlowerdlower are given as follows −  dlower=y−ykdlower=y−yk  =m(Xk+1)+b−Yk=m(Xk+1)+b−Yk  and  dupper=(yk+1)−ydupper=(yk+1)−y  =Yk+1−m(Xk+1)−b=Yk+1−m(Xk+1)−b  You can use these to make a simple decision about which pixel is closer to the mathematical line. This simple decision is based on the difference between the two pixel positions.  dlower−dupper=2m(xk+1)−2yk+2b−1dlower−dupper=2m(xk+1)−2yk+2b−1  Let us substitute *m* with *dy/dx* where *dx* and *dy* are the differences between the end-points.  dx(dlower−dupper)=dx(2dydx(xk+1)−2yk+2b−1)dx(dlower−dupper)=dx(2dydx(xk+1)−2yk+2b−1)  =2dy.xk−2dx.yk+2dy+2dx(2b−1)=2dy.xk−2dx.yk+2dy+2dx(2b−1)  =2dy.xk−2dx.yk+C=2dy.xk−2dx.yk+C  So, a decision parameter PkPk for the *k*th step along a line is given by −  pk=dx(dlower−dupper)pk=dx(dlower−dupper)  =2dy.xk−2dx.yk+C=2dy.xk−2dx.yk+C  The sign of the decision parameter PkPk is the same as that of dlower−dupperdlower−dupper.  If pkpk is negative, then choose the lower pixel, otherwise choose the upper pixel.  Remember, the coordinate changes occur along the x axis in unit steps, so you can do everything with integer calculations. At step k+1, the decision parameter is given as −  pk+1=2dy.xk+1−2dx.yk+1+Cpk+1=2dy.xk+1−2dx.yk+1+C  Subtracting pkpk from this we get −  pk+1−pk=2dy(xk+1−xk)−2dx(yk+1−yk)pk+1−pk=2dy(xk+1−xk)−2dx(yk+1−yk)  But, xk+1xk+1 is the same as (xk)+1(xk)+1. So −  pk+1=pk+2dy−2dx(yk+1−yk)pk+1=pk+2dy−2dx(yk+1−yk)  Where, Yk+1–YkYk+1–Yk is either 0 or 1 depending on the sign of PkPk.  The first decision parameter p0p0 is evaluated at (x0,y0)(x0,y0) is given as −  p0=2dy−dxp0=2dy−dx  Now, keeping in mind all the above points and calculations, here is the Bresenham algorithm for slope m < 1 −  **Step 1** − Input the two end-points of line, storing the left end-point in (x0,y0)(x0,y0).  **Step 2** − Plot the point (x0,y0)(x0,y0).  **Step 3** − Calculate the constants dx, dy, 2dy, and (2dy – 2dx) and get the first value for the decision parameter as −  p0=2dy−dxp0=2dy−dx  **Step 4** − At each XkXk along the line, starting at k = 0, perform the following test −  If pkpk < 0, the next point to plot is (xk+1,yk)(xk+1,yk) and  pk+1=pk+2dypk+1=pk+2dy  Otherwise,  pk+1=pk+2dy−2dxpk+1=pk+2dy−2dx  **Step 5** − Repeat step 4 (dx – 1) times.  For m > 1, find out whether you need to increment x while incrementing y each time.  After solving, the equation for decision parameter PkPk will be very similar, just the x and y in the equation gets interchanged. |

|  |
| --- |
| **Input** |
| Starting and ending point of line |

|  |
| --- |
| **Output** |
| Given pattern in the problem statement |

|  |
| --- |
| **Lab. Based FAQ** |
| Explain DDA  Explain Bresenham  Difference Between DDA and Bresenham |