Programming Exercise 2 : scan conversion of lines

Introduction

In this exercise, you will implement two algorithms for scan conversion of lines: DDA (Digital differential analyzer) and MidLine algorithm. Before starting on this programming exercise, we strongly recommend watching the lectures and glut tutorial.

To get started with the exercise, you will need to download the starter codes. There is a skeleton file "***ScanLine.cpp***" which provided all nessary code for you to start, you just need to finish the function *DDAline()* and *Midpointline()* to scan conversion of line with any slope.

We implemented the code partly for *Midpointline()* to draw a line with slope between 0 and 1. You must modify the code in this function to make it workable for lines with negative slope or slope larger than 1.

To run these programs ,the glut enviroment should be setup.You can get the glut from glut-3.7.6-bin.zip (117 KB)(http://user.xmission.com/~nate/glut/glut-3.7.6-bin.zip) ,unzip it and copy three files to different locations on your window as following:

- glut.h – This is the file you'll have to include in your source code. The common place to put this file is in the gl folder which should be inside the include folder of your system. for example ,your vc2010 include folder "*c:\program files\Microsoft Visual Studio 10.0\VC\include\gl*" or "*c:\program files(x86)\Microsoft Visual Studio 10.0\VC\include\gl*" in(my conputer
- glut32.lib (Windows version) – This file must be linked to your application so make sure to put it your lib folder. example ,your vc2010 lib folder "*c:\program files\Microsoft Visual Studio 10.0\VC\lib*" or "*c:\program files(x86)\Microsoft Visual Studio 10.0\VC\lib*" in(my conputer
- glut32.dll (Windows) – You could place the dll file in your exe's folder.for example,the sustem folder: "*c:\windows\system32*" or "*c:\windows\sysWOW64*" in my conputer

Scan conversion of lines

You could run the starter code " *ScanLine.cpp* " firstly and see the result as following(figure 1):

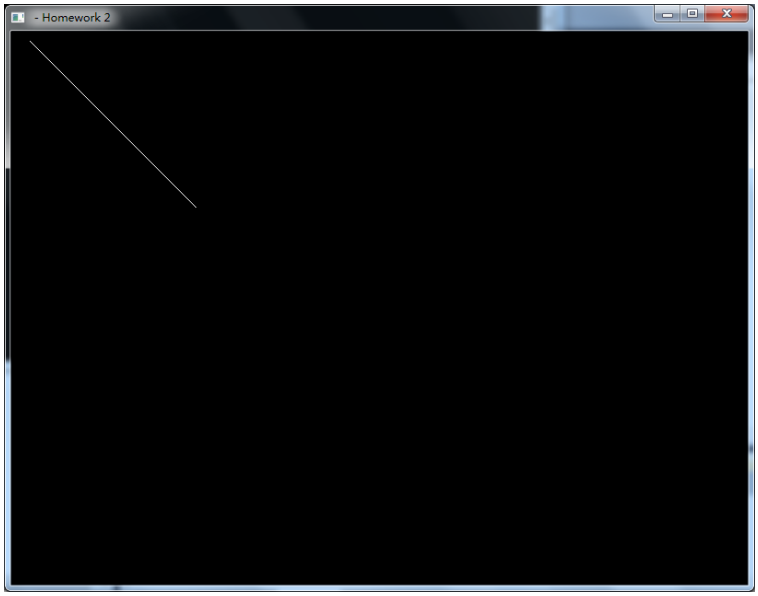

figure 1 a line from (20,10) to (200,400)

Please note that the origin of the screen coordinate frame is located on the top left.

Start a line from (xL, yL) to (xH, yH) with "xL< xH" and the slope m of the line is between 0 and 1, The DDA algorithm works as following:

float $x = xL$; $y = yL$;

for ($i = 0$; $i <= x_H - x_L$; i++)

DrawPixel (x, Round (y))

$x = x + 1$;

$y = y + m$;

The Round (y) is a helper cunction to find the integer nearest to the real number y. For example

std::cout <<round(1.57) <<" "<< round(1.49) ;

the code will output :

2 1

The DDA algorithm start from the start point and incrementely calculate the intersection of the line with row grid line or column grid line of pixel screen and draw the pixel nearest to the intersection point (figure 2).

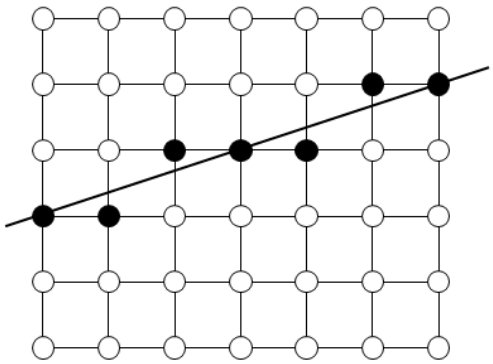

figure 2 DDA scan conversion of line

For lines with slope less than or equal to 1 sample at unit x intervals (dx=1) and compute successive y values as

$$y_{k+1} = y_k + m$$

For lines with slope greater than 1, we reverse the role of x and y i.e. we sample at dy=1 and calculate consecutive x values as

$$x_{k+1} = x_k + \frac{1}{m}$$

So the DDA code could be as following:

```
void DDAline(int x1,int y1,int x2,int y2,float R, float G, float B){
    float dX,dY,iSteps;
    float xInc,yInc,iCount,x,y;
    dX = x2 - x1;
    dY = y2 - y1;

    if (fabs(dX) > fabs(dY))
        iSteps = fabs(dX);
    else
        iSteps = fabs(dY);
    xInc = dX/iSteps;
    yInc = dY/iSteps;

    x = x1;      y = y1;
    setpixel(x,y,R,G,B);

    for (iCount=1; iCount<=iSteps; iCount++)    {
        // you should fill the code here
    }
    setpixel(x,y,R,G,B);
    return;
}
```

Your first task is to finish the code above and make it run to draw lines with any slope.

Your second task is: do the similar modification to code in MidLine() so that to draw lines with any slope.