案例 3　DDA 画线算法

知识点

- 第一个八分象限的 DDA 算法。
- 直线的主位移方向。
- 增量算法。

一、案例需求

1. 案例描述

在窗口客户区内自定义二维坐标系。基于第一个八分象限的 DDA 算法，绘制起点为 $P_0(0,0)$、终点为 $P_1(300,100)$的一段直线。

2. 功能说明

(1) 自定义二维屏幕坐标系，原点位于客户区中心，x 轴水平向右为正，y 轴垂直向上为正。直线的起点和终点基于自定义坐标系定义。

(2) 绘制一段黑色直线，包括起点，不包括终点。

3. 案例效果图

绘制一段第一个八分象限内的直线，效果如图 3-1 所示。

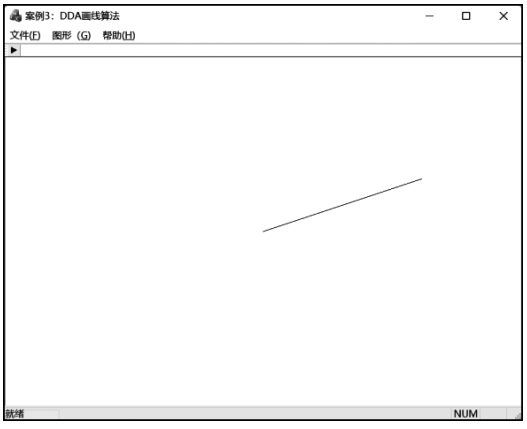


图 3-1　第一个八分象限内直线的效果图

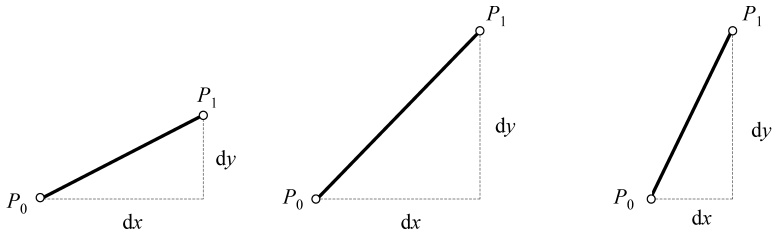
二、案例分析

1. 设计目的

学习用离散点表示理想直线的思想。光栅扫描显示器是画点设备，因此不能直接从一点到另一点绘制一段直线。直线扫描转换的结果是一组在几何上距离理想直线最近的离散像素点集。

2. 直线的主位移方向

扫描转换算法中，常根据直线在 x 轴和 y 轴上的投影长度确定主位移方向。在主位移方向上执行的是±1 操作，另一个方向上可以±1 或 0。如果 $dx > dy$，则取 x 方向为主位移方向，如图 3-2(a)所示；如果 $dx = dy$，取 x 方向为主位移方向或取 y 方向为主位移方向皆可，如图 3-2(b)所示；如果 $dx < dy$，则取 y 方向为主位移方向，如图 3-2(c)所示。



(a) x方向　　(b) x方向或y方向　　(c) y方向

图 3-2　主位移方向

3. DDA 算法

数值微分法(digital differential analyzer，DDA)是用数值方法求解微分方程的一种算法，根据直线的微分方程设计算法。

当直线的斜率满足 $0 \leqslant k \leqslant 1$ 时，有 $\Delta x \geqslant \Delta y$，所以 x 方向为主位移方向。取 $\Delta x = 1$，有 $\Delta y = k$。DDA 算法可简单表述为

$$\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = y_i + k \end{cases} \tag{3-1}$$

式(3-1)表示直线上的像素 P_{i+1} 与像素 P_i 的递推关系。可以看出，x_{i+1} 和 y_{i+1} 的值可以根据 x_i 和 y_i 的值推算出来，这说明 DDA 算法是一种增量算法。在一个迭代算法中，如果每一步的 x、y 值是用前一步的值加上一个增量获得的，那么，这种算法就称为增量算法。

x 方向的增量为 1，y 方向的增量为 k。x 是整型变量，y 和 k 是浮点型变量。DDA 算法使用宏命令 ROUND(y_{i+1})=int(y_{i+1} + 0.5)选择下一个像素，可表述为

$$y_{i+1} = \begin{cases} y_i + 1, & k \geqslant 0.5 \\ y_i, & k < 0.5 \end{cases} \tag{3-2}$$

三、算法设计

(1) 读入直线的端点坐标 $P_0(x_0, y_0)$和 $P_1(x_1, y_1)$。

(2) 计算直线沿 x 方向的位移量 dx 和沿 y 方向的位移量 dy。

(3) 计算直线的斜率 k。

(4) 沿 x 方向绘制点(x, y)，x 加 1，y 加 k。

(5) 如果 k 大于或等于 0.5，(x, y)更新为$(x+1, y+1)$；否则，(x, y)更新为$(x+1, y)$。

(6) 当直线没有绘制完，重复步骤(4)。

四、案例实现

1. 使用二维整数点类

MFC 提供了 CPoint 点类(派生自 tagPOINT 结构体),成员变量 x 和 y 取为整型。CPoint 类主要用于对屏幕上所绘制的像素点坐标取整。

2. 圆整函数

为了保证计算精度,点坐标的计算值是 double 型,而绘制到屏幕上是整型坐标,有两种取整方式,均用带参数的宏定义。

1) #define ROUND(d) int(d + 0.5)

这种圆整方式仅适用于对第一象限内浮点数 d 的圆整,取离 d 最近的整数点。定义形式简单,容易理解。

2) #define ROUND(d) int(floor((d) +0.5))

C++ 语言有两个取整函数 floor()和 ceil(),原型如下:

```
double floor( double x );
```

返回值是小于或等于 x 的最大浮点数,数学符号为"⌊⌋"。

```
double ceil( double x );
```

返回值是大于或等于 x 的最小浮点数,数学符号为"⌈⌉"。

例如,floor(2.8)的返回值为 2.000000,floor(−2.8)的返回值为−3.000000,ceil(2.8)的返回值为 3.000000,ceil(−2.8)的返回值为−2.00000。

floor()函数是向负无穷大方向取整,ceil()函数是向正无穷大方向取整。

ROUND (2.8)函数的返回值是 3.00000;ROUND (−2.8)函数的返回值是−3.00000。

3. 设计 OnDraw( )函数

在 OnDraw()函数中自定义二维坐标系,将屏幕划分为 4 个象限,x 坐标和 y 坐标出现了负值。鼠标点在设备坐标系中获得,只有正值,需要对二者进行转换。

```
void CTestView::OnDraw(CDC * pDC)
{
    CTestDoc * pDoc=GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;
    // TODO: 在此处为本机数据添加绘制代码
    CRect rect;                                              //定义客户区矩形
    GetClientRect(&rect);                                    //获得客户区矩形的信息
    pDC->SetMapMode(MM_ANISOTROPIC);                         //自定义二维坐标系
    pDC->SetWindowExt(rect.Width(), rect.Height());          //设置窗口范围
    pDC->SetViewportExt(rect.Width(), -rect.Height());       //设置视区范围
    pDC->SetViewportOrg(rect.Width()/2, rect.Height()/2);    //设置二维坐标系原点
    rect.OffsetRect(-rect.Width()/2, -rect.Height()/2);      //rect 矩形与客户区重合
    CPoint P0(0, 0), P1(300, 100);                           //确定直线的起点和终点
    DDALine(pDC, P0, P1);                                    //调用 DDA 算法
}
```

程序说明：在自定义坐标系内绘制起点位于原点，终点位于第一个八分象限内的直线。

4. DDA 算法

```
void CTestView::DDALine(CDC * pDC, CPoint P0, CPoint P1)
{
    int dx=P1.x-P0.x;                              //直线水平方向位移
    int dy=P1.y-P0.y;                              //直线垂直方向位移
    double k=double(dy)/dx;                        //直线斜率
    int x;
    double y=P0.y;
    COLORREF crColor=RGB(0, 0, 0);
    for (x=P0.x; x<P1.x; x++)
    {
        pDC->SetPixelV(x, ROUND(y), crColor); //绘制离散像素点
        y+=k;
    }
}
```

程序说明：SetPixelV()函数不需要返回值，绘图速度要快于 SetPixel()函数。

五、案例小结

(1) DDA 算法计算一个像素点时，需要做除法运算，因此计算时采用了浮点数。

(2) DDA 算法需要对计算结果进行圆整处理，不适合硬件实现。

六、案例拓展

给定直线的起点坐标为 $P_0(x_0, y_0)$、终点坐标为 $P_1(x_1, y_1)$，容易计算出直线的斜率 k。当 $|k| \leqslant 1$ 时，DDA 算法的递推公式为 $\begin{cases} x_{i+1} = x_i \pm 1 \\ y_{i+1} = y_i \pm k \end{cases}$；当 $|k| \geqslant 1$ 时，DDA 算法的递推公式为 $\begin{cases} x_{i+1} = x_i \pm \dfrac{1}{k} \\ y_{i+1} = y_i \pm 1 \end{cases}$。假定直线的间隔角度是 5°，试编程绘制图 3-3 所示的图形。

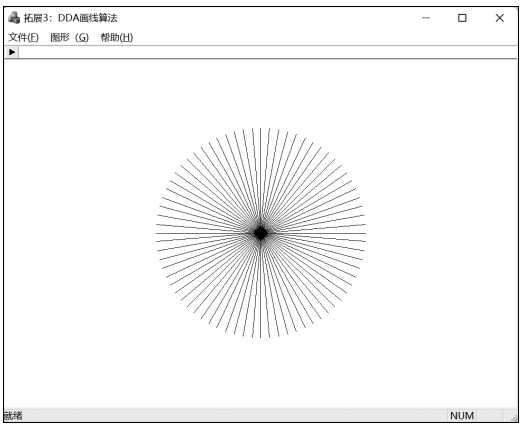


图 3-3 放射直线