《计算机图形学》实验报告

线画图元生成算法实验

姓名 王雨朦

学号 3012216083

专业 计算机科学与技术

班级 3 班

天津大学计算机科学与技术学院

2014 年 12 月 20 日

一、实验目的

1. 实现 DDA 直线生成算法
2. 实现 Bresenham 直线生成算法
3. 实现中点画圆算法（选做）

二、实验内容

1、采用 MFC 基于对话框实现；

2、直线绘制方法：在 picture control 控件中某位置按下鼠标，作为直线的起点，拖拽鼠标到直线终点，然后松开鼠标即可绘制一条直线；

3、由于为了让实验结果更清晰，实验采用像素思想，此实验设置的每个像素大小为 10*10，每个网格用矩形填充。

4、为解决在控件中画图的频闪问题，实验用到双缓冲技术；

5、算法核心思想：

1）对于不存在斜率的直线（端点横坐标相等），直接画竖线即可；

2）对于存在斜率 m 的直线，先判断斜率 m 是否小于零，如果是，则关于 Y 轴对称。再判断斜率 m 是否大于 1，如果是，关于 y=x 对称；

3）将所有的斜率都归一到斜率 m 属于 [0,1] 后，保证起点的 X 坐标小于终点的 X 坐标的情况下，从起点到终点，求出将要生成直线的 X、Y 的坐标。

DDA 直线生成算法：

```
x= x+1;y =(int)(y+m+0.5);
```

Bresenham 直线生成算法：

初始 Pk = Δy*2 - Δx。

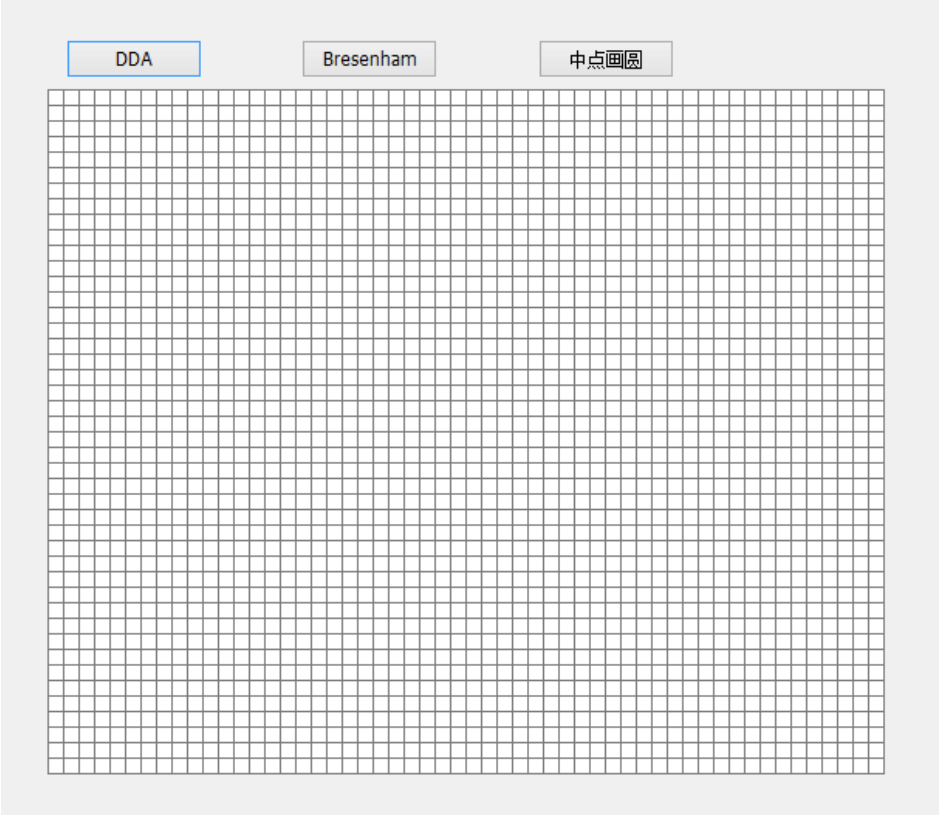
若 Pk>0,则 y = y + 1;Pk= Pk+Δy*2 - Δx*2;否则，y 不变，Pk= Pk+Δy*2;

4) 求出将要生成直线的 XY 坐标后，如果前面有关于 y=x 对称的，再对称回去；有关于 Y 轴对称的也再对称回去;

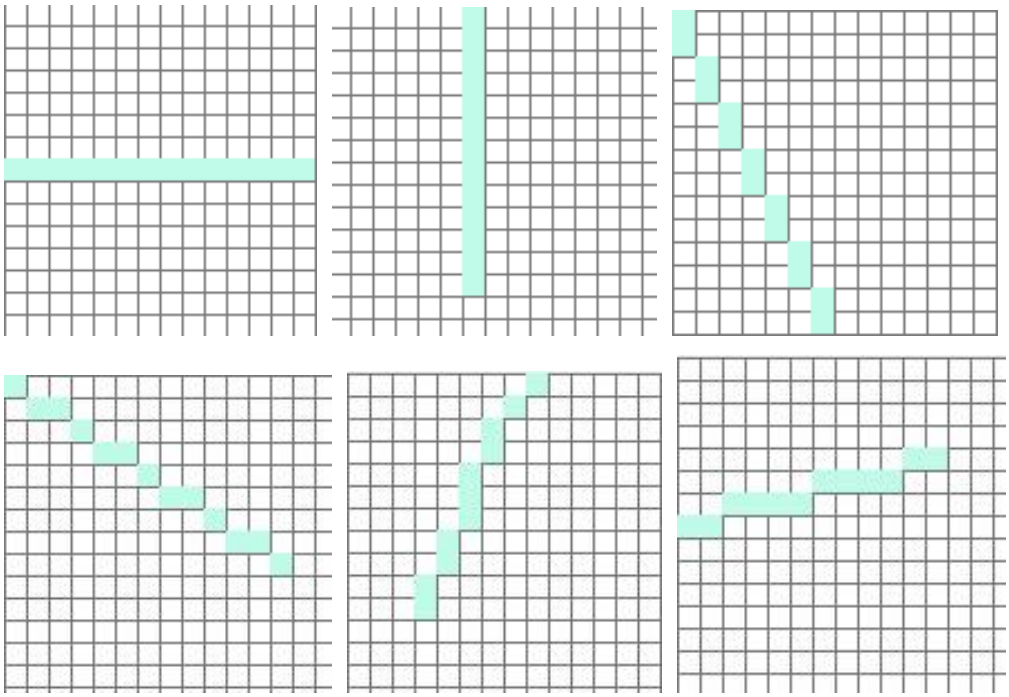
5) 按照求出得到的 XY 坐标，从起点到终点填充像素，绘制直线。

三、 实验结果

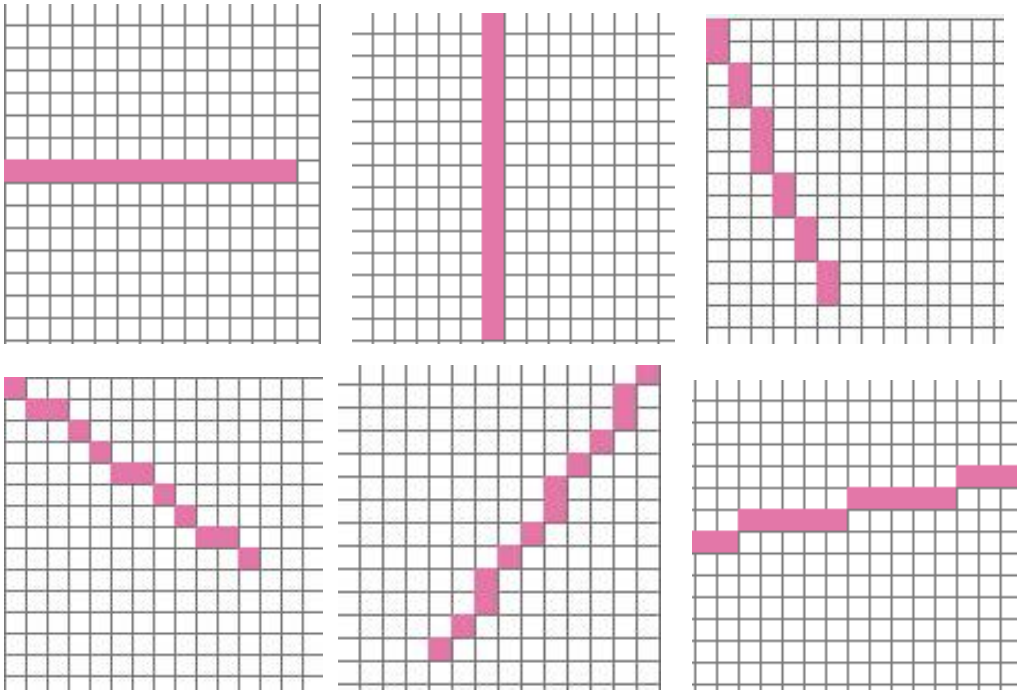
1. 界面



2. DDA 直线生成算法：



3. Bresenham 直线生成算法：



四、实验分析和总结

1. 本次实验较好的完成了 DDA 直线生成算法和 Bresenham 直线生成算法，实现了画图工具中直线绘制的功能。
2. 为了实验结果的清晰，我们采用网格表示像素，使得生成每条直线的图元更加清晰可见。两种算法的图元差别也对比明显。
3. 通过实验我们更加深入的理解了直线元生成思想

五、源代码

1. 头文件

```
public:
    int m_algorithm = 0;//标识鼠标按的哪按钮; //1, dda;2,bresenham;3,中点画圆
    afx_msg void OnBnClickedButton1();//DDA
    afx_msg void OnBnClickedButton2();
    afx_msg void OnBnClickedButton3();
    void drawGrid(CDC *pDc);//网格背景
    CPoint p1, p2;//鼠标点击画线的端点
    bool flag = false;//鼠标画线的标志
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    int x1, y1, x2, y2;//单击点所在像素的左上角点坐标
    double m;
    void ddaLine(CDC *pDc);
    void bresenhamLine(CDC *pDc);
    void xySwap(int &x, int &y);
    bool reverse = false,xnot = false;//关于y=x对称，关于y轴对称
```

2. Cpp 文件

```
void CdrawLineDlg::OnPaint()
{
    if ……
    else
    {
        CRect rect;
        CWnd *pWin = GetDlgItem(IDC_STATIC);//获取该控件的指针
```

```
        pWin->GetClientRect(rect);//把控件的长宽、坐标等信息保存在rect里
        int width = rect.Width();//获取宽和高
        int height = rect.Height();
        CString str;
        //str.Format("高度为：%d,宽度为：%d", height, width);
        //AfxMessageBox(str, MB_YESNO | MB_ICONSTOP, 0);

        CDC *pDc = pWin->GetDC();//获取该控件的画布
        //pDc->Rectangle(rect);//填充矩形
        //pDc->SelectStockObject(BLACK_BRUSH);

        //双缓冲技术
        CDC MemDC; //定义显示设备对象
        CBitmap MemBitmap;//定义位图对象
        MemDC.CreateCompatibleDC(NULL); //这时还不能绘图，因为没有地方画
        //建立与屏幕显示兼容的位图，位图的大小这里用窗口的大小
        MemBitmap.CreateCompatibleBitmap(pDc, width, height); //将位图选入内存设备中
        CBitmap *pOldBit = MemDC.SelectObject(&MemBitmap);//先用背景色将位图清除干净
        MemDC.FillSolidRect(0, 0, width, height, RGB(255,255,255));//背景色置白色
        // 画图
        x1 = (int)(p1.x / GRIDLEN)*GRIDLEN;
        y1 = (int)(p1.y / GRIDLEN)*GRIDLEN;
        x2 = (int)(p2.x / GRIDLEN)*GRIDLEN;
        y2 = (int)(p2.y / GRIDLEN)*GRIDLEN;
        m = (y2 - y1)*1.0 / (x2 - x1);//!类型转换
        drawGrid(&MemDC);//网格
        if (m_algorithm == 1){
            ddaLine(&MemDC);
        }
        if (m_algorithm == 2){
            bresenhamLine(&MemDC);
        }
        //将内存中的图拷贝到屏幕上进行显示
        pDc->BitBlt(0, 0, width, height, &MemDC, 0, 0, SRCCOPY);//rect.left
        //绘图完成后的清理
        MemDC.SelectObject(pOldBit);//把前面的pOldBit选回来.在删除MemBitmap之前要先从
设备中移除它
        MemBitmap.DeleteObject();
        MemDC.DeleteDC();
        CBrush brRed(RGB(128, 0, 0));
        CBrush brGreen(RGB(0, 128, 0));
        CBrush brBlue(RGB(0, 0, 128));
        //pDc->FillRect(rect, &myBrush);
    }
```

```
}
//交换两点坐标
void CdrawLineDlg::xySwap(int &x, int &y)
{
    int mid;
    mid = x;
    x = y;
    y = mid;
}
//dda画直线
void CdrawLineDlg::ddaLine(CDC *pDc)
{
    if (m_algorithm == 1)
    {
        if (flag){
            CBrush myBrush;
            myBrush.CreateSolidBrush(RGB(192, 250, 233));//蓝绿色
            if (x1 == x2){//斜率不存在
                if (y1 > y2)
                {
                    xySwap(x1, x2);
                    xySwap(y1, y2);
                }
                for (int y = y1; y <= y2; y = y + GRIDLEN)
                {
                    CRect rect(x1, y, x1 + GRIDLEN, y + GRIDLEN);
                    //CRect rect(pt1, pt2);
                    pDc->FillRect(&rect, &myBrush);
                }
            }
            else{//斜率存在
                if (m <0){//若斜率小于零，关于y轴对称
                    x1 = -x1;
                    x2 = -x2;
                    m = -m;
                    xnot = true;
                }
                if (m > 1){//若斜率大于一，关于y = x对称
                    xySwap(x1, y1);
                    xySwap(x2, y2);
                    m = 1 / m;
                    reverse = true;
                }
                if (x1 > x2){ //保证x1<x2
```

```
                    xySwap(x1, x2);
                    xySwap(y1, y2);
                }
                if (m >= 0 && m <= 1){
                    double y_initial = (y1 / GRIDLEN) *1.0;
                    int x, y = y1;
                    for (x = x1; x <= x2; x = x + GRIDLEN){
                        if (reverse){//m>1
                            xySwap(x, y);
                        }
                        if (xnot){  //m<0
                            x = -x;
                        }
                        CRect rect(x, y, x + GRIDLEN, y + GRIDLEN);
                        pDc->FillRect(&rect, &myBrush);
                        if (xnot){  //m<0
                            x = -x;
                        }
                        if (reverse){//m>1;换完了应该换回去;
                            xySwap(x, y);
                        }
                        //改变y的值
                        y_initial += m;
                        y = (int)(y_initial + 0.5) *GRIDLEN;
                        //dc.SetPixel(int(x), int(y + 0.5), RGB(255, 0, 0));
                    }//for
                }//if m=[0,1]
            }//else 斜率存在的情况
        }
    }//if m_algerithm button
}//DDA
//bresenham画直线
void CdrawLineDlg::bresenhamLine(CDC *pDc)
{
    if (m_algorithm == 2)
    {
        if (flag)
        {
            CBrush myBrush;
            myBrush.CreateSolidBrush(RGB(227, 119, 168));//粉紫色

            if (x1 == x2){//斜率不存在
                if (y1 > y2)
                {
```

```
            xySwap(x1, x2);
            xySwap(y1, y2);
        }
        for (int y = y1; y <= y2; y = y + GRIDLEN)
        {
            CRect rect(x1, y, x1 + GRIDLEN, y + GRIDLEN);
            //CRect rect(pt1, pt2);
            pDc->FillRect(&rect, &myBrush);
        }
    }
    else{//斜率存在
        if (m <0){//若斜率小于零，关于y轴对称
            x1 = -x1;
            x2 = -x2;
            m = -m;
            xnot = true;
        }
        if (m > 1){//若斜率大于一，关于y = x对称
            xySwap(x1, y1);
            xySwap(x2, y2);
            m = 1 / m;
            reverse = true;
        }
        if (x1 > x2){ //保证x1<x2
            xySwap(x1, x2);
            xySwap(y1, y2);
        }

        if (m >= 0 && m <= 1){
            int dy = y2 - y1, dx = x2 - x1;
            int p = dy * 2 - dx;
            int x, y = y1;
            for (x = x1; x <= x2; x += GRIDLEN){
                if (reverse){//m>1
                    xySwap(x, y);
                }
                if (xnot){  //m<0
                    x = -x;
                }
                CRect rect(x, y, x + GRIDLEN, y + GRIDLEN);
                pDc->FillRect(&rect, &myBrush);
                if (xnot){  //m<0
                    x = -x;
                }
```

```
                        if (reverse){//m>1;换完了应该换回去;
                            xySwap(x, y);
                        }
                        //改变y的值
                        if (p > 0){
                            y = y + GRIDLEN;
                            p = p + dy * 2 - dx * 2;
                        }
                        else
                        {
                            p = p + dy * 2;
                        }
                        //dc.SetPixel(int(x), int(y + 0.5), RGB(255, 0, 0));
                    }//for
                }//if m=[0,1]
            }//else 斜率存在的情况
        }//flag
    }//if_algorithm
}//bresenham
//画网格
void CdrawLineDlg::drawGrid(CDC *pDc)
{
    CPen myPen;
    myPen.CreatePen(PS_SOLID, 0.1, RGB(122, 122, 122));
    CPen *Oldpen = pDc->SelectObject(&myPen);
    int i;
    int n = (GRIDNUM - 1)* GRIDLEN;
    for (i = 0; i <= n; i = i + GRIDLEN)
    {
        pDc->MoveTo(0, i);
        pDc->LineTo(n, i);
        pDc->MoveTo(i, 0);
        pDc->LineTo(i, n);
    }
}
void CdrawLineDlg::OnBnClickedButton1()
{
    m_algorithm =1;//dda
}
void CdrawLineDlg::OnBnClickedButton2()
{
    m_algorithm = 2;//bresenham
}
void CdrawLineDlg::OnBnClickedButton3()
```

```
{
    m_algorithm = 3;//中点画圆
}
//鼠标响应函数
void CdrawLineDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    CRect rectP;//获取控件的大小
    GetDlgItem(IDC_STATIC)->GetWindowRect(&rectP);//获取控件相对于屏幕的位置
    ScreenToClient(rectP);//转化为对话框上的相对位置
    if ((point.x >= rectP.left && point.x <= rectP.right) && (point.y >= rectP.top && point.y
<= rectP.bottom))
    {
        point.x =point.x- rectP.left;
        point.y =point.y- rectP.top;
        p1 = point;
        flag = false;//点下时不画线
    }
    else{//控件外，不做响应
        flag = false;
    }
    reverse = false; xnot = false;//每一次都是否对称需要重置
    CDialogEx::OnLButtonDown(nFlags, point);
}
void CdrawLineDlg::OnLButtonUp(UINT nFlags, CPoint point)
{
    CRect rectP;//获取控件的大小
    GetDlgItem(IDC_STATIC)->GetWindowRect(&rectP);//获取控件相对于屏幕的位置
    ScreenToClient(rectP);//转化为对话框上的相对位置
    //保证画图在控件内
    if ((point.x >= rectP.left && point.x <= rectP.right) && (point.y >= rectP.top && point.y
<= rectP.bottom))
    {
        //转化成相对于控件的坐标，在控件上绘图
        point.x = point.x - rectP.left;
        point.y = point.y - rectP.top;
        p2 = point;
        flag = true;
    }
    else{//控件外，不做响应
        flag = false;
    }
    //UpdateWindow();
    CDialogEx::OnLButtonUp(nFlags, point);
}
```