

吉林大学

计算机科学与技术学院

《计算机图形学》

实验报告

班级： 计科 2 班

学号： 53160203

姓名： 倪畅

实验项目	边标志算法的实现		
实验性质	<input type="checkbox"/> 演示性实验 <input checked="" type="checkbox"/> 验证性实验 <input type="checkbox"/> 操作性实验 <input checked="" type="checkbox"/> 综合性实验		
实验地点	计算机楼B212	机器编号	
指导教师	徐长青	实验时间	2018年 11月5日 14时40分

一、实现的功能

采用鼠标输入顶点的方法确定待填充多边形（多边形最后一点双击）；实现边标志算法完成对该多边形的填充，并使用自己学号的后四位数字对多边形内部进行填充。

二、采用的图形学算法及实现

1.用一个顶点列表来记录多边形顶点：

CArray<CPoint, CPoint> pointList;

2. 实现DDA直线画法的函数为：

void DDALine(CDC *pDC, int x1, int y1, int x2, int y2, COLORREF color);

x1、y1代表起点，x2、y2代表终点，color为颜色。

3.绘图以鼠标双击结束，识别边界

```

else if(type==5)
{
    if (!m_LButtonDown && pointList.GetSize()>0)
    {
        ReleaseCapture();//释放鼠标
        MapObj* obj = new MapObj();
        obj->type = 5;
        for (int i=0;i<pointList.GetSize();i++){
            obj->points.Add(pointList.GetAt(i));
        }
        objList.Add(obj);
        CDC* pDC = this->GetDC();
        CPoint px = obj->points.GetAt(0);
        CPoint py;
        for (int j=1;j<obj->points.GetSize();j++){
            py = obj->points.GetAt(j);
            DDALine(pDC,px.x,px.y,py.x,py.y,RGB(r,g,b));
            px = py;
        }
        py = obj->points.GetAt(0);
        DDALine(pDC,px.x,px.y,py.x,py.y,RGB(r,g,b));
        CPoint p1,p2,p3;
        //取矩阵最小扫描界
        int maxX, minX, maxY, minY;
        p1 = pointList.GetAt(0);
        maxX = p1.x;
        minX = p1.x;
        maxY = p1.y;
        minY = p1.y;
        for (int i = 1; i < pointList.GetSize(); i++) {
            p1 = pointList.GetAt(i);
            if (p1.x > maxX) maxX = p1.x;
            if (p1.x < minX) minX = p1.x;
            if (p1.y > maxY) maxY = p1.y;
            if (p1.y < minY) minY = p1.y;
        }
        //改变极值点
        p1 = pointList.GetAt(pointList.GetSize()-1);
        p2 = pointList.GetAt(0);
        p3 = pointList.GetAt(1);
        if ((p1.y - p2.y)*(p2.y - p3.y) < 0)
        {
            pDC->SetPixel(p2.x, p2.y, RGB(255, 255, 255));
        }
    }
}

for (int ii = 1; ii < pointList.GetSize(); ii++) {
    p1 = pointList.GetAt(ii-1);
    p2 = pointList.GetAt(ii);
    if (ii != pointList.GetSize()-1)
    {
        p3 = pointList.GetAt(ii+1);
    }
    else
    {
        p3 = pointList.GetAt(0);
    }
    if ((p1.y-p2.y)*(p2.y-p3.y)<0)
    {
        pDC->SetPixel(p2.x, p2.y, RGB(255, 255, 255));
    }
}
//填充
bool inside = false; //是否在多边形中
bool flag = true; //是否需要改变
bool bder = false; //是否为边界值
for (int y = minY-1; y < maxY+1; y++)
{
    inside = false;
    flag = false;
    for (int x = minX-1; x < maxX+1; x++)
    {
        bder = (pDC->GetPixel(x, y) == RGB(0, 0, 0));
        if (bder && flag)
        {
            inside = !inside;
            flag = false;
        }
        else if (!bder && !flag)
        {
            flag = true;
        }
        if (inside && !bder) PatternFill(pDC,x,y);
    }
}
pointList.RemoveAll();
ReleaseCapture();//释放鼠标

```

4. 填充学号的函数为:

```
void PatternFill(CDC *pDC, int x, int y);
```

x、y为当前坐标，矩阵取余实现学号显示。

[illegible]

三、采用的交互方式及实现

用鼠标绘制图形，首先需要确定鼠标的绘制动作。即每次按下鼠标左键时的点都填入pointList；同时使用左键双击来执行填补最后一条直线和学号填充操作。根据以上绘制方法，可知需要处理：

WM_OnLButtonDown (左键按下)、

WM_OnLButtonDblClk（左键双击）消息，

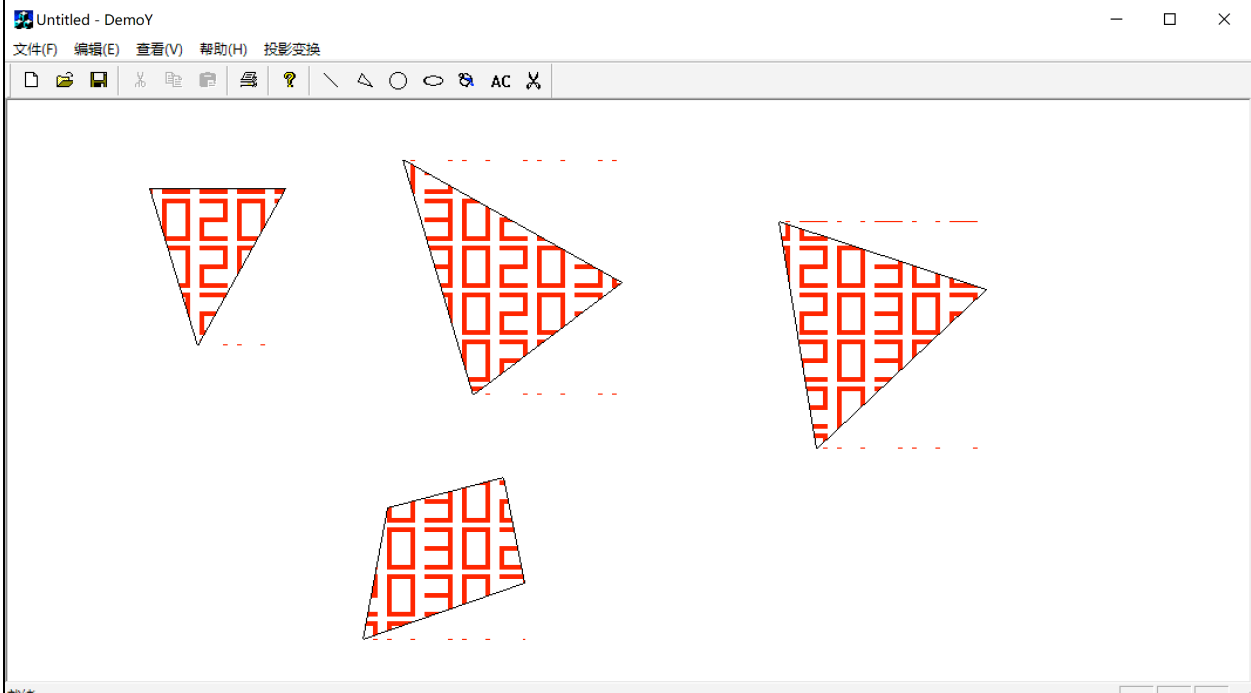
为了绘制橡皮线，还需处理WM MouseMove（鼠标移动）消息。

其中 WM_OnLButtonDblClk 里, 需要先取矩阵最小扫描界以减少扫描量, 其次改变极值点的边界标识, 最后再依据标志点取反扫描填充。

四、实验结果

（程序的运行结果）

选择椭圆右边那个油漆桶 



五、遇到的问题及解决办法

以边线颜色作为边界标志，在处理极值点的时候需要将顶点涂白。

另一方面，有时画线会一行不止占用一个像素，即出现了有宽度的线。

这时候，需要通过额外添加一个判断符，来标记扫描是否再图形中还是再边界上。以此增加可靠度。但是画的时候还是偶尔漏线，可能需要判断顶点。

实验项目	用矩形窗口对多边形进行裁剪		
实验性质	<input type="checkbox"/> 演示性实验 <input checked="" type="checkbox"/> 验证性实验 <input type="checkbox"/> 操作性实验 <input checked="" type="checkbox"/> 综合性实验		
实验地点	计算机楼B212	机器编号	
指导教师	徐长青	实验时间	2018 年 11 月 12 日 14 时 40 分
<p>一、实现的功能</p> <p>编写应用程序实现多边形裁剪。要求首先采用鼠标确定裁剪区域（矩形区域），然后用鼠标输入待裁剪的多边形（可分别使用鼠标左键和右键来确定裁剪区域和待裁剪的多边形）。多边形绘制完毕后进行裁剪，以不同颜色显示被裁剪对象位于窗口内（此部分应保证多边形的完整性）及外部的部分。</p>			
<p>二、采用的图形学算法及实现</p> <p>1. 裁剪矩形表示点</p> <p>CPoint EPoint;//end point</p> <p>CPoint BPoint;//begin point</p> <p>2. 折线计算结果数组</p> <p>CArray<CPoint, CPoint> pointCutList; 用来记录裁剪运算后的点集;</p> <p>3. 裁剪函数</p> <p>void CutTop();</p> <p>void CutLeft();</p> <p>void CutBottom();//代码和top类似，不贴了</p> <p>void CutRight();//代码和left类似，不贴了</p> <p>根据矩形的范围，进行4条边的裁剪。</p>			

```

void CDemoView::CutTop()
{
    CPoint F, P, S, I;
    int c1 = 0;
    int c2 = 0;
    int i = 0;
    pointCutList.RemoveAll();
    for (i = 0; i < pointList.GetSize(); i++)
    {
        P = pointList.GetAt(i);
        if (i != 0){
            c2 = (P.y < BPoint.y ? -1 : 1);
            if (c1 + c2 == 0) {
                I.y = BPoint.y;
                I.x = (P.x - S.x) * (I.y - S.y) / (P.y - S.y) + S.x;
                pointCutList.Add(I);
            }
        }
        else{
            F = P;
            S = P;
            if (S.y < BPoint.y) {
                c1 = -1;
            }
            else {
                c1 = 1;
                pointCutList.Add(S);
            }
        }
        c2 = (F.y < BPoint.y ? -1 : 1);
        if (c1 + c2 == 0) {
            I.y = BPoint.y;
            I.x = (F.x - S.x) * (I.y - S.y) / (F.y - S.y) + S.x;
            pointCutList.Add(I);
        }
        pointList.RemoveAll();
        for (i = 0; i < pointCutList.GetSize(); i++)
        {
            pointList.Add(pointCutList.GetAt(i));
        }
    }
}

```

```

void CDemoView::CutLeft()
{
    CPoint F, P, S, I;
    int c1 = 0;
    int c2 = 0;
    pointCutList.RemoveAll();
    for (int i = 0; i < pointList.GetSize(); i++)
    {
        P = pointList.GetAt(i);
        if (i != 0)
        {
            c2 = (P.x < BPoint.x ? -1 : 1);
            if (c1 + c2 == 0) {
                I.x = BPoint.x;
                I.y = (P.y - S.y) * (I.x - S.x) / (P.x - S.x) + S.y;
                pointCutList.Add(I);
            }
        }
        else{
            F = P;
        }
        S = P;
        if (S.x < BPoint.x){
            c1 = -1;
        }
        else{
            c1 = 1;
            pointCutList.Add(S);
        }
    }
    c2 = (F.x < BPoint.x ? -1 : 1);
    if (c1 + c2 == 0) {
        I.x = BPoint.x;
        I.y = (F.y - S.y) * (I.x - S.x) / (F.x - S.x) + S.y;
        pointCutList.Add(I);
    }
    pointList.RemoveAll();
    for (i = 0; i < pointCutList.GetSize(); i++)
    {
        pointList.Add(pointCutList.GetAt(i));
    }
}

```


三、采用的交互方式及实现

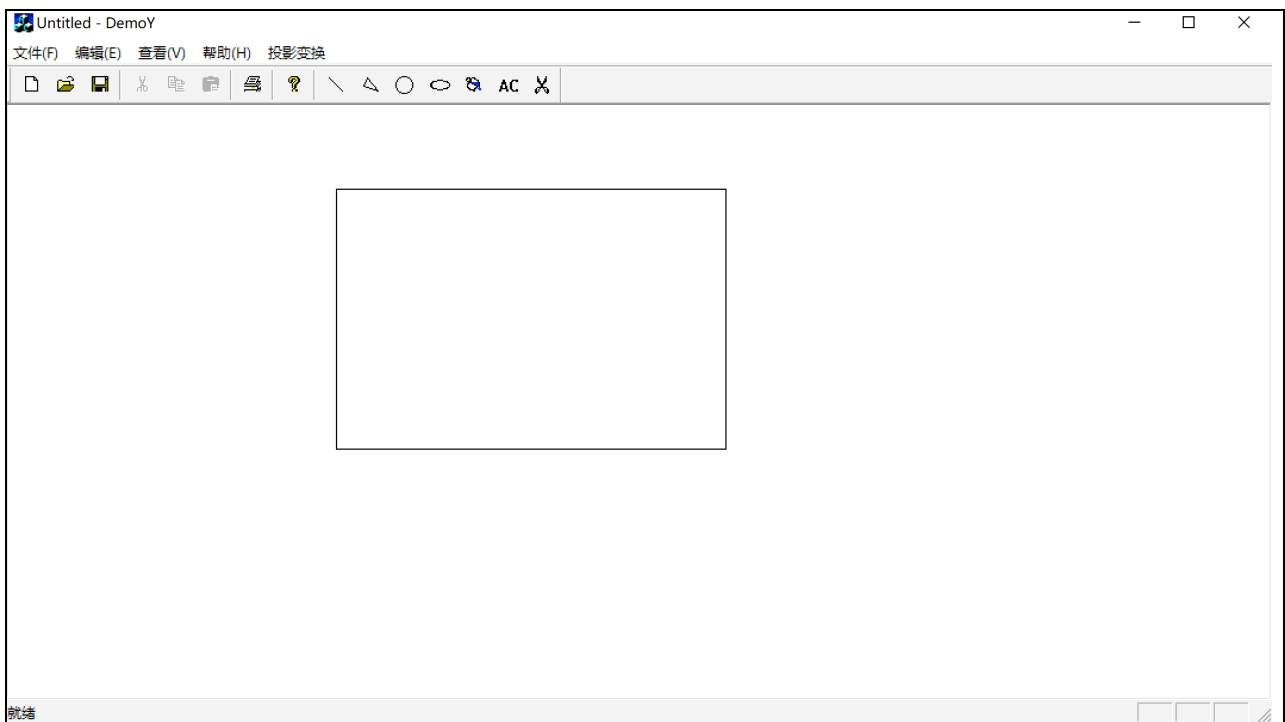
用鼠标绘制图形，首先需要确定鼠标的绘制动作。即每次按下鼠标左键时的点都填入pointList；同时使用右键双击来执行矩形操作。根据以上绘制方法，可知需要处理：

WM_OnLButtonDown（左键按下）、WM_OnLButtonDblClk（左键双击），为了绘制橡皮线，还需处理 WM_MouseMove（鼠标移动）消息。

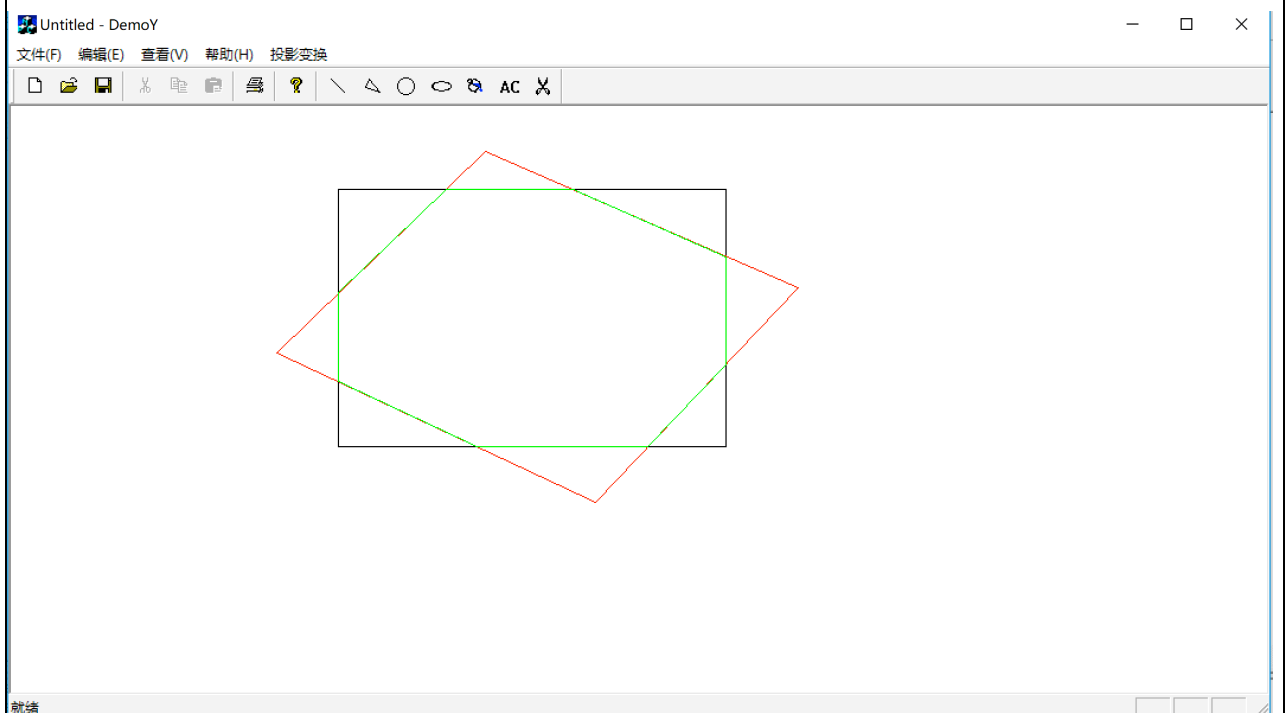
WM_OnRButtonDown（右键按下）、WM_OnRButtonUp（右键弹起）消息绘制矩形，同样为了绘制橡皮线，也需处理 WM_MouseMove（鼠标移动）消息。

四、实验结果

（程序的运行结果）选择 AC 右边的小剪刀
 右键绘制矩形表示裁剪框 



然后左键绘制要裁剪的图形，左键双击结束



五、遇到的问题及解决办法

是从一开始的三角形样例程序一直添加各种功能来完成前四个实验的，但是直到这个实验才用到右键，一开始没有加右键监听，怎么画都画不出矩形，卡了很长时间，后来才发现，马上添加了

WM_OnRButtonDown 和 WM_OnRButtonUp，运行成功。

实验项目	4 阶 3 次等距 B 样条曲线绘制及变换		
实验性质	<input type="checkbox"/> 演示性实验 <input checked="" type="checkbox"/> 验证性实验 <input type="checkbox"/> 操作性实验 <input checked="" type="checkbox"/> 综合性实验		
实验地点	计算机楼B212	机器编号	
指导教师	徐长青	实验时间	2018 年 12 月 10 日 14 时 40 分
<p>一、实现的功能</p> <p>编写应用程序，让用户用鼠标左键单击的方式顺序输入控制点，鼠标左键双击代表结束，在工具条中添加三个按钮，使之分别具有如下功能：</p> <p>1) 按钮1的功能：绘制以鼠标输入的的点为控制点的4阶3次等距B样条曲线，要求该曲线要以控制点的第一个点和最后一点为起点和终止点，每段曲线用红、蓝、绿三种颜色之一绘制，相邻的曲线段颜色不能相同，并且要绘制出控制多边形。</p> <p>2) 按钮2的功能：假设控制点中第一个点的坐标为（x_1, y_1），绘制出以$x = x_1$直线为对称轴的按钮1所绘制的曲线的对称曲线，要求原曲线和对称曲线都要画出来。</p> <p>3) 按钮3的功能：设2)中所显示图形在空间中$z=0$平面内，以$x = x_1$直线为旋转轴，将该图形旋转360度，每次旋转步长为10度，可构成一个立体图形。绘制该图形的斜二测投影图。</p> <p>要求每个按钮功能执行之前都要先清屏，保证视图区每次只显示一个按钮功能所绘制的图形。</p> <p>二、 采用的图形学算法及实现</p> <p>计算 B 样条基函数:</p> <pre>double CFinal07View::Base(int i, int k, double knot[], int num, double u)</pre> <p>计算 B 样条曲线在参数为 U 的坐标值:</p> <pre>CPoint CFinal07View::BSpline(double knot[], int num, double u, int t)</pre> <p>简单的清屏函数，重新开了个工程重写了下 void</p> <pre>CFinal07View::ClearScreen()</pre> <p>画出三色 B 样条曲线: void CFinal07View::Draw3ColorBSpline()</p> <p>坐标变换 void CFinal07View::From3DTo2D(Point3D *c, int n)</p> <pre>void CFinal07View::Onone()void CFinal07View::Ontwo() void CFinal07View::Onthree()//三个按钮</pre>			


```

double CFinal07UView::Base(int i, int k, double knot[], int num, double u)
{
    double temp1 = 0;
    double temp2 = 0;
    if (k == 1)
    {
        if (i == (num - 2))
        {
            if ((u >= knot[i]) && (u <= knot[i + 1])) return 1;
            else return 0;
        }
        else
        {
            if ((u >= knot[i]) && (u < knot[i + 1])) return 1;
            else return 0;
        }
    }
    else if (k > 1)
    {
        if (knot[i + k - 1] != knot[i])
        {
            temp1 = Base(i, k - 1, knot, num, u);
            temp1 = (u - knot[i]) * temp1 / (knot[i + k - 1] - knot[i]);
        }
        if (knot[i + k] != knot[i + 1])
        {
            temp2 = Base(i + 1, k - 1, knot, num, u);
            temp2 = (knot[i + k] - u) * temp2 / (knot[i + k] - knot[i + 1]);
        }
        return temp1 + temp2;
    }
    return 0;
}

void CFinal07UView::Onone()
{
    // TODO: Add your command handler code here
    TypeEXP = 7;
    ClearScreen();

    typeEXP7 = 1;
    N = 3;
    K = 4;
    nPoints = 30;
    pointList.RemoveAll();
    pointPrintList.RemoveAll();
}

void CFinal07UView::Ontwo()
{
    // TODO: Add your command handler code here
    typeEXP7 = 2;
    ClearScreen();
    pointPrintList.RemoveAll();
    Draw3ColorBSpline();
    CPoint tempP = pointList.GetAt(0);
    int xcenter = tempP.x;
    for (int i = 1; i < pointList.GetSize(); i++)
    {
        tempP = pointList.GetAt(i);
        tempP.x = xcenter * 2 - tempP.x;
        pointList.SetAt(i, tempP);
    }
}

```

```

void CFinal07View::Onthree()
{
    // TODO: Add your command handler code here
    typeEXP7 = 3;

    double L = 0.5;
    double Alpha = 45;
    Alpha *= 3.1415926 / 180;
    double c = cos(Alpha);
    double s = sin(Alpha);
    mPM[0][0] = 1;
    mPM[0][1] = 0;
    mPM[0][2] = L * c;
    mPM[0][3] = 0;
    mPM[1][0] = 0;
    mPM[1][1] = 1;
    mPM[1][2] = L * s;
    mPM[1][3] = 0;
    mPM[2][0] = 0;
    mPM[2][1] = 0;
    mPM[2][2] = 0;
    mPM[2][3] = 0;
    mPM[3][0] = 0;
    mPM[3][1] = 0;
    mPM[3][2] = 0;
    mPM[3][3] = 1;

    Point3D *curve = new Point3D[pointPrintList.GetSize()];
    CPoint tempP = pointList.GetAt(0);
    CClientDC pDC(this);
    CPen *oldpen;
    COLORREF color[3] = { RGB(255,0,0), RGB(0,255,0), RGB(0,0,255) };
    int xcenter = tempP.x;
    for (int i = 0; i < 36; i++)
    {
        Alpha = i * 10 * 3.1415926 / 180;
        for (int j = 0; j < pointPrintList.GetSize(); j++)
        {
            tempP = pointPrintList.GetAt(j);
            curve[j].x = (tempP.x - xcenter) * cos(Alpha) + xcenter;
            curve[j].y = tempP.y;
            curve[j].z = (tempP.x - xcenter) * sin(Alpha);
        }

        From3DTo2D(curve, pointPrintList.GetSize());

        pDC.MoveTo(pointPrint3DList.GetAt(0));
        for (j = 1; j < pointPrint3DList.GetSize(); j++)
        {
            CPen pen;
            pen.CreatePen(PS_SOLID, 2, color[j / (nPoints + 1) % 3]);
            oldpen = pDC.SelectObject(&pen);
            pDC.LineTo(pointPrint3DList.GetAt(j));
            pDC.SelectObject(oldpen);
        }
    }
}

void CFinal07View::Draw3ColorBSpline()
{
    CClientDC pDC(this);
    CPen *oldpen;
    COLORREF color[3] = { RGB(255,0,0), RGB(0,255,0), RGB(0,0,255) };
    BSplineToPoints();
    pDC.MoveTo(pointPrintList.GetAt(0));

    for (int i = 1; i < pointPrintList.GetSize(); i++)
    {
        CPen pen;
        pen.CreatePen(PS_SOLID, 3, color[i / (nPoints + 1) % 3]);
        oldpen = pDC.SelectObject(&pen);
        pDC.LineTo(pointPrintList.GetAt(i));
        pDC.SelectObject(oldpen);
    }
}

```

三、采用的交互方式及实现

用鼠标绘制图形，首先需要确定鼠标的绘制动作。即每次按下鼠标左键时的点都填入pointList；同时使用左键双击来结束操作。

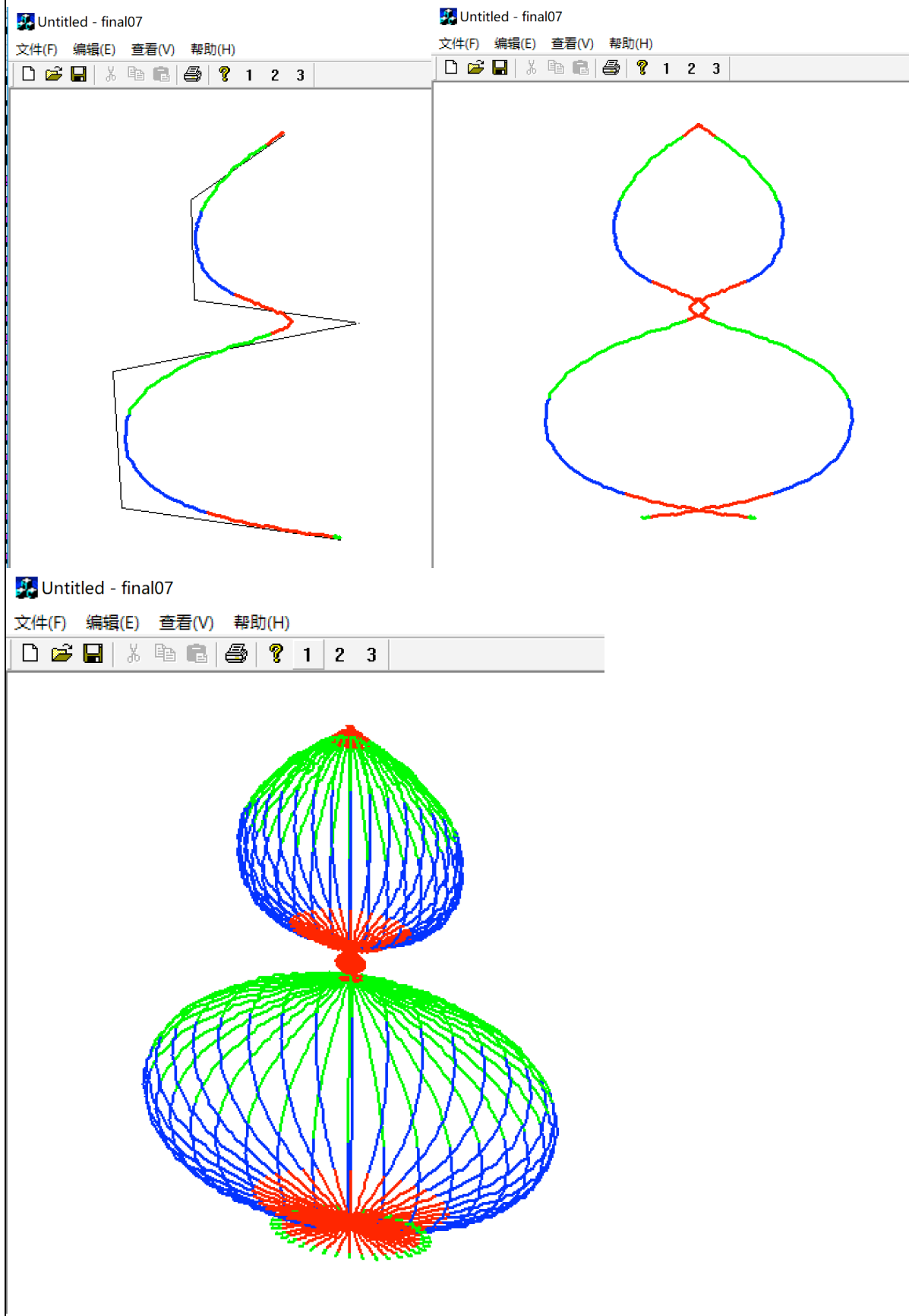
根据以上绘制方法，可知需要处理：

WM_OnLButtonDown（左键按下）、WM_OnLButtonUp（左键抬起）

WM_OnLButtonDblClk（左键双击）消息，

为了绘制橡皮线，还需处理WM_MouseMove（鼠标移动）消息。

四、实验结果



五、遇到的问题及解决办法

B 样条本来头尾和首尾点不连着，然后老师指导要将首尾点当作三个点，因此这样处理，成功运行

```
void CFinal07View::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call de
    switch (typeEXP7)
    {
    case 1:
    {
        this->SetCapture();
        startPoint = point;
        endPoint = point;
        boolLButtonDown = true;

        pointList.Add(point);
        if(pointList.GetSize()<2){
            pointList.Add(point);
            pointList.Add(point);
        }

    }
    break;
    default:
        break;
    }
    CView::OnLButtonDown(nFlags, point);
}

void CFinal07View::OnLButtonDb1C1k(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    switch (TypeEXP)
    {
    case 7:
    {
        if (boolLButtonDown)
        {
            pointList.Add(point);
            pointList.Add(point);
            ReleaseCapture();
            boolLButtonDown = false;
            boolRButtonDown = false;
            Draw3ColorBSpline();
        }

    }
    break;
    default:
        break;
    }
    CView::OnLButtonDb1C1k(nFlags, point);
}
```

