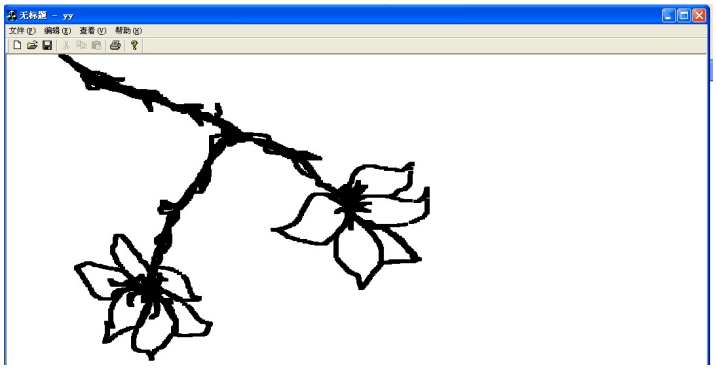
【计算机图形学课程】二.MFC鼠标响应函数模拟画图软件

上一篇文章我们讲述MFC绘制图形的基本函数，包括绘制直线、绘制矩形、绘制椭圆及绘制文字，同时通过绕圆旋转和矩形平移简单的理解了图形学知识。这篇文章我将介绍鼠标响应和键盘响应，通过这些事件让学生实现一个类似画图的简单软件，同时充分发挥学生想象，自己创作东西。

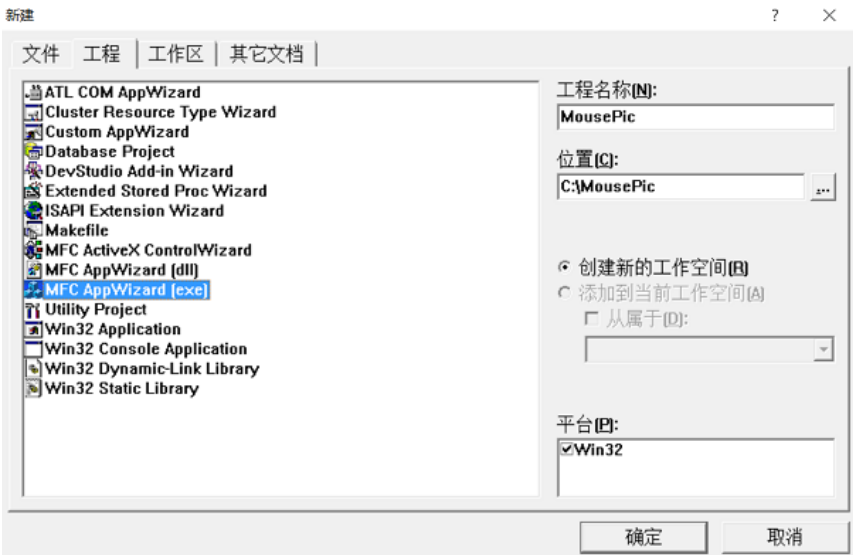
前文：

【计算机图形学课程】一.MFC基本绘图函数使用方法

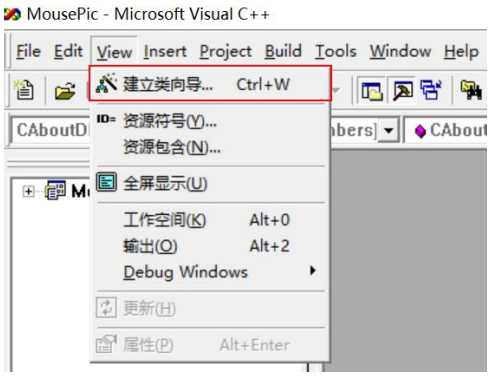


一. MFC工程创建及鼠标响应

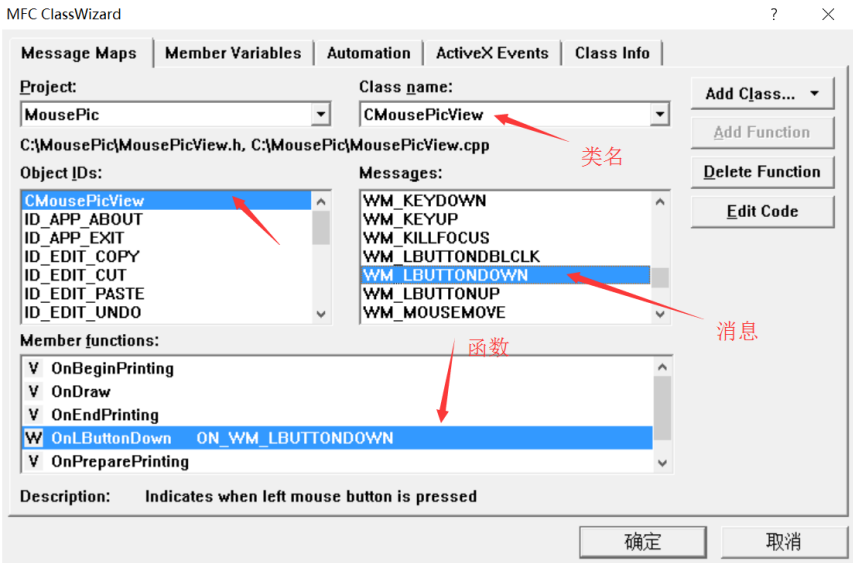
新建一个MFC 单文档的应用程序"MousePic"。



然后，选择"View（视图）"->"建立类向导"，快键键是Ctrl+W。这是MFC非常重要的一个知识点，对话框或单文档设置按钮操作、响应函数都是通过该操作实现。



在MFC ClassWizard中选择创建工程的"CMousePicView"类名，然后再"Message"中选择"WM_LBUTTONDOWN"，鼠标左键按下响应操作。同时，双击它添加函数OnLButtonDown()。



鼠标常见消息响应：

WM_LBUTTONDBCLK 双击鼠标左键

WM_LBUTTONDOWN 按下鼠标左键

WM_LBUTTONUP 释放鼠标左键

WM_MOUSEMOVE 在客户区移动鼠标

WM_RBUTTONDBCLK 双击鼠标右键

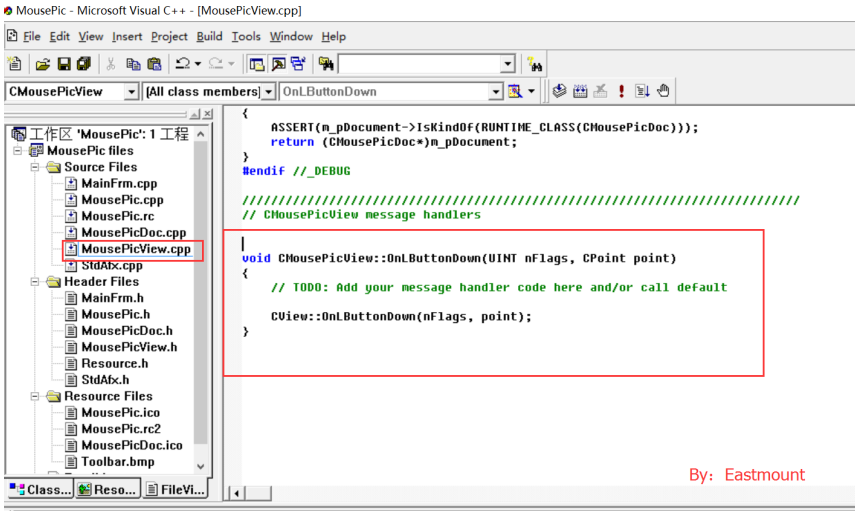
WM_RBUTTONDOWN 按下鼠标右键

WM_RBUTTONUP 释放鼠标右键

二. MFC实现鼠标响应操作

1.鼠标左键按下

双击函数会定位到"MousePicView.cpp"文件，现在可以对OnLButtonDown()函数进行编辑。其中CPoint point参数记录当前鼠标左键按下的位置，nFlags表示掩码。

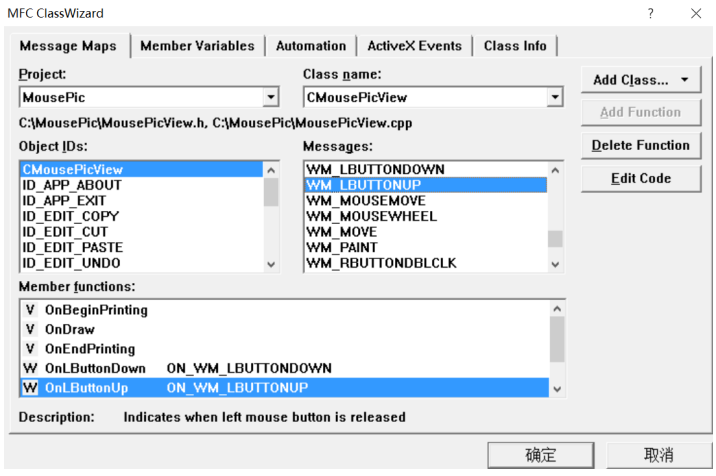


然后添加代码如下：

```
//定义一个点类型的变量,用来保存当用户点击界面时点击的位置
CPoint m_point;

//鼠标左键按下
void CMousePicView::OnLButtonDown(UINT nFlags, CPoint point)
{
	//把当前点击的点的位置赋给点m_point
	m_point = point;
	CView::OnLButtonDown(nFlags, point);
}
```

2.鼠标左键释放

通过同样的方法在"类向导"中实现鼠标左键释放函数，如下图所示。



添加代码主要是鼠标释放（弹起）：

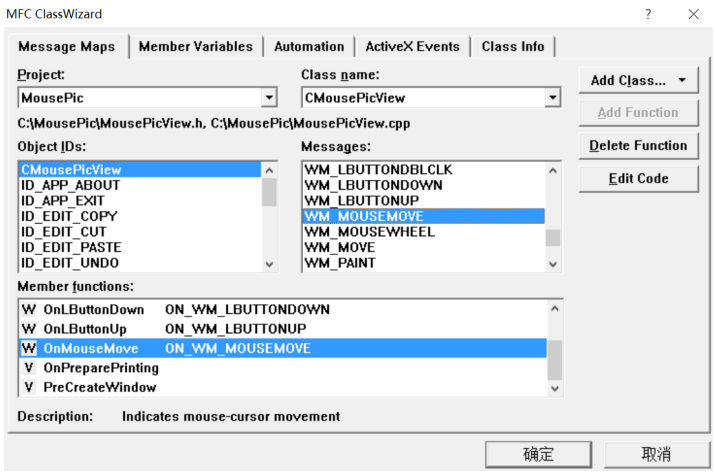
```
//鼠标释放：记录当前坐标
void CMousePicView::OnLButtonUp(UINT nFlags, CPoint point)
{
	//绘制图形
	CDC *p = GetDC();
	p->MoveTo(m_point);	//鼠标移动到左键按下点
	p->LineTo(point);	//绘制一条直线 终点为鼠标释放点
	CView::OnLButtonUp(nFlags, point);
}
```

此时鼠标绘制图形如下所示，但是存在两个问题：绘制过程中不可见、绘制结果只是直线。

所以，需要借助鼠标移动函数实现，在鼠标移动过程中就进行绘制，同时引入bool类型的变量，判断鼠标按下或释放，按下的时候进行绘制操作。

3.鼠标左键移动

通过同样的方法在"类向导"中实现鼠标左键释放函数。

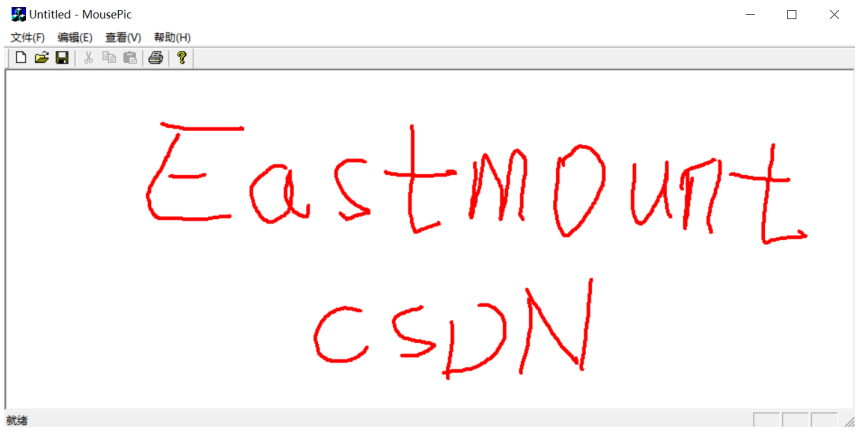


完整代码如下所示：

```
//定义一个点类型的变量,用来保存当用户点击界面时点击的位置
CPoint m_point;

//定义布尔型变量 m_click=true表示鼠标点击 false表示鼠标释放
bool m_click;

//鼠标左键按下
void CMousePicView::OnLButtonDown(UINT nFlags, CPoint point)
{
	//把当前点击的点的位置赋给点m_point
	m_point = point;
	m_click = true;
	CView::OnLButtonDown(nFlags, point);
}


//鼠标释放：记录当前坐标
void CMousePicView::OnLButtonUp(UINT nFlags, CPoint point)
{
	//绘制图形
	/*
	CDC *p = GetDC();
	p->MoveTo(m_point);	//鼠标移动到左键按下点
	p->LineTo(point);	//绘制一条直线 终点为鼠标释放点
	*/
	m_click = false;
	CView::OnLButtonUp(nFlags, point);
}

//鼠标移动绘制图形
void CMousePicView::OnMouseMove(UINT nFlags, CPoint point)
{
	//定义画笔并选择
	CDC *p=GetDC();
	CPen pen(PS_SOLID, 4, RGB(255,0,0));
	p->SelectObject(pen);
```

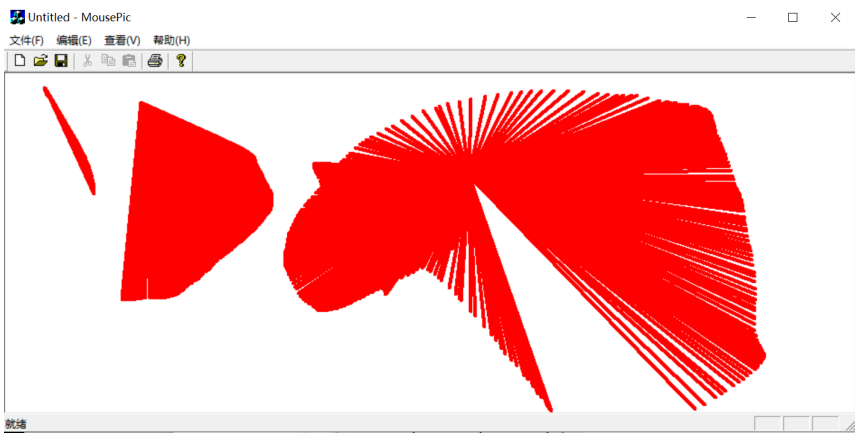
```
            //鼠标按下进行绘制
            if(m_click==true) {
                    p->MoveTo(m_point);
                    p->LineTo(point);
                    m_point = point;
            }
            CView::OnMouseMove(nFlags, point);
    }
```

绘制结果如下所示，相当于一个简单的画图软件。



4.补充知识

如果在OnMouseMove()鼠标移动函数if判断中缺少代码m_point = point，它会出现意想不到的效果，因为你需要每次绘制，鼠标移动当前点坐标point都需要赋值给下次绘制的起始坐标，供p->MoveTo(m_point)使用。



同时，你可以绘制圆形、矩形等相关形状，不仅仅限定于直线。

```
//鼠标移动绘制图形
void CMousePicView::OnMouseMove(UINT nFlags, CPoint point)
{
	//定义画笔并选择
	CDC *p=GetDC();
	CPen pen(PS_SOLID, 1, RGB(255,0,0));
	p->SelectObject(pen);

	//鼠标按下进行绘制
	if(m_click==true) {
		p->MoveTo(m_point);
		//p->LineTo(point);
		p->Rectangle(point.x, point.y,point.x+20, point.y+30);
		m_point = point;
	}
	CView::OnMouseMove(nFlags, point);
}
```

输出如下所示：

绘制中，定义了画笔Pen，正确的方法需要在绘制完成后，进行释放该画笔。核心代码如下：

```
	//定义画笔绘制矩形
	CPen MyPen, *OldPen;
	MyPen.CreatePen(PS_DASH, 2, RGB(0,0,255)); //虚线 粗2 蓝色
	OldPen = pDC->SelectObject(&MyPen);        //旧画笔赋值

	//绘制图形


	//清除
```

```
    pDC->SelectObject(OldPen);
                                    MyPen.DeleteObject();
```

三. MFC键盘响应函数

1.基础知识

Windows对每个按键定义了与设备无关的编码，这种编码叫做虚拟码。有了这个虚拟码，Windwos程序员可以使用该虚拟码进行编程。其中键盘上部分按键的虚拟码如下图所示：

虚拟码	虚拟码所对应的按键	虚拟码	虚拟码所对应的按键
VK_ADD	数字小键盘上的+键	VK_HOME	Home
VK_BACK	BackSpace	VK_INSERT	Insert
VK_CANCEL	Ctrl-Break	VK_LEFT	向左的箭头键
VK_CAPITAL	Caps Lock	VK_MENU	Alt
VK_CONTROL	Ctrl	VK_MULTIPLY	数字小键盘上的“*”键
VK_DECIMAL	数字小键盘上的“.”键	VK_NUMPAD0~9	数字小键盘上的0~9键
VK_DELETE	Delete	VK_RETURN	Enter
VK_DIVIDE	数字小键盘上的“/”键	VK_RIGHT	向右的箭头键
VK_DOWN	向下的箭头键	VK_SHIFT	Shift
VK_ESCAPE	Esc	VK_UP	向上的箭头键
VK_F1~VK_F12	F1~F12		

Windows按键消息常见如下：

WM_CHAR 敲击键盘上的字符键时，产生该消息

WM_KEYDOWN 任意键（包括字符键）被按下时都产生该消息，如果被按下的是字符键，在产生消息的同时还产生字符消息

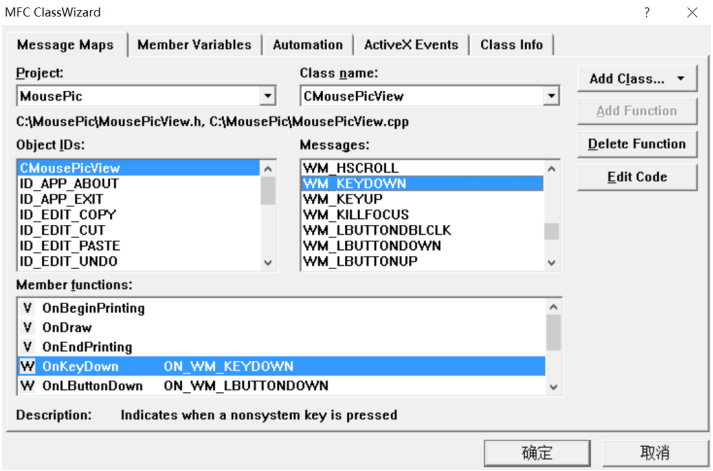
WM_KEYUP 任意角（包括字符键）被释放都产生该消息

WM_SYSKEYDOWN F10被按下或者Alt与另一个键被同时按下

WM_SYSKEYUP F10被释放或者Alt与另一个键被同时释放

2.按键响应操作

同样，通过类向导建立按键按下函数。



然后添加如下代码，按下任意一个键，绘制的矩形向右平移40距离。

```
//鼠标按键
void CMousePicView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	// TODO: Add your message handler code here and/or call default

	CDC *p = GetDC();
	p->MoveTo(m_point);		//键盘按下
	m_point.x += 40;		//水平平移40
	p->Rectangle(m_point.x, m_point.y, m_point.x+20, m_point.y+30);
	CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

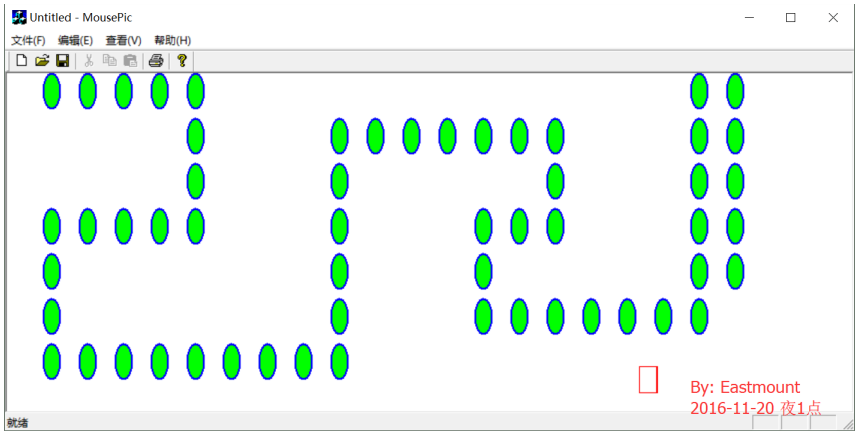
绘制图形如下所示：

3.响应不同键盘的操作

需要将UINT nChar转换为Char字符型，然后进行盘，WASD进行上下左右移动绘制椭圆。

```
//鼠标按键
void CMousePicView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        // TODO: Add your message handler code here and/or call default

        CDC *p = GetDC();
        char cChar;               //当前被按下的字符
    cChar = char(nChar);      //将按下的键转换为字符

        //定义画笔
    CPen MyPen, *OldPen;
    MyPen.CreatePen(PS_DASH, 2, RGB(0,0,255));  //虚线 粗2 蓝色
    OldPen = p->SelectObject(&MyPen);           //旧画笔赋值

    //画刷
    CBrush MyBrush, *OldBrush;
    MyBrush.CreateSolidBrush(RGB(0,255,0));
    OldBrush = p->SelectObject(&MyBrush);


    if (cChar == 'D') {
                p->MoveTo(m_point);       //D键按下
                m_point.x += 40;          //水平向右平移40
                p->Ellipse(m_point.x, m_point.y, m_point.x+20, m_point.y+40);
        }
        if (cChar == 'A') {
                p->MoveTo(m_point);       //A键按下
                m_point.x -= 40;          //水平向左平移40
                p->Ellipse(m_point.x, m_point.y, m_point.x+20, m_point.y+40);
        }
        if (cChar == 'S') {
                p->MoveTo(m_point);       //S键按下
                m_point.y += 50;          //竖直向下平移50
                p->Ellipse(m_point.x, m_point.y, m_point.x+20, m_point.y+40);
        }
        if (cChar == 'W') {
                p->MoveTo(m_point);       //W键按下
                m_point.y -= 50;          //竖直向上平移50
                p->Ellipse(m_point.x, m_point.y, m_point.x+20, m_point.y+40);
        }


        //清除
    p->SelectObject(OldPen);
    MyPen.DeleteObject();
    p->SelectObject(OldBrush);
    MyBrush.DeleteObject();

        CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

绘制如下图所示：



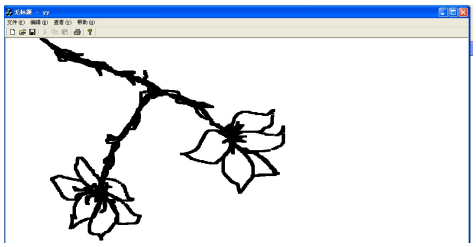
4.按键光标选择

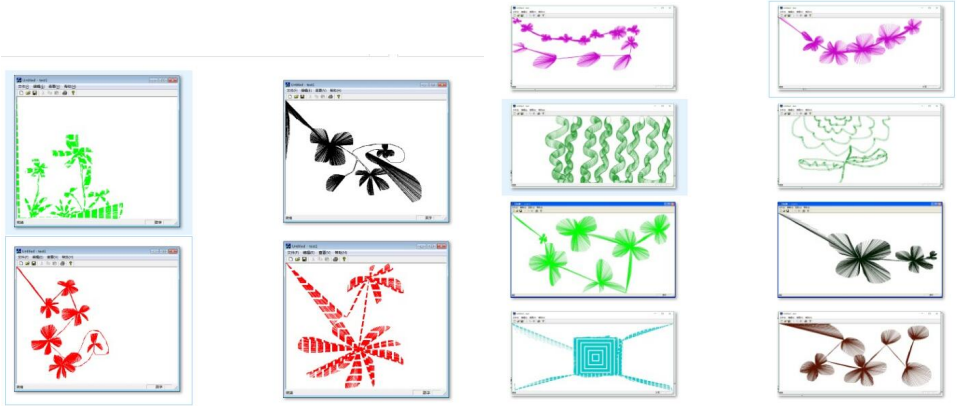
```
//鼠标按键
void CMousePicView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        //光标操作
    char cChar;               //当前被按下的字符
    HCURSOR hCursor = 0;      //显示光标句柄
    HCURSOR hPrevCursor = 0;  //以前的光标句柄
    cChar = char(nChar);      //将按下的键转换为字符
    if (cChar == 'A'){
        //加载箭头光标
        hCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
    }
    if (cChar == 'B'){
        //加载箭头光标
        hCursor = AfxGetApp()->LoadStandardCursor(IDC_IBEAM);
    }
    if (cChar == 'C'){
        //加载箭头光标
        hCursor = AfxGetApp()->LoadStandardCursor(IDC_WAIT);
    }
    if (cChar == 'X'){
        hCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
        hPrevCursor = SetCursor(hCursor);
        if (hPrevCursor)
            DestroyCursor(hPrevCursor);
    }
    else{
        if (hCursor){
            hPrevCursor = SetCursor(hCursor);
            if (hPrevCursor)
                DestroyCursor(hPrevCursor);
        }
```

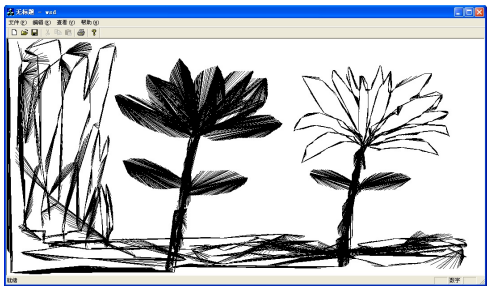
```
    }

        CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

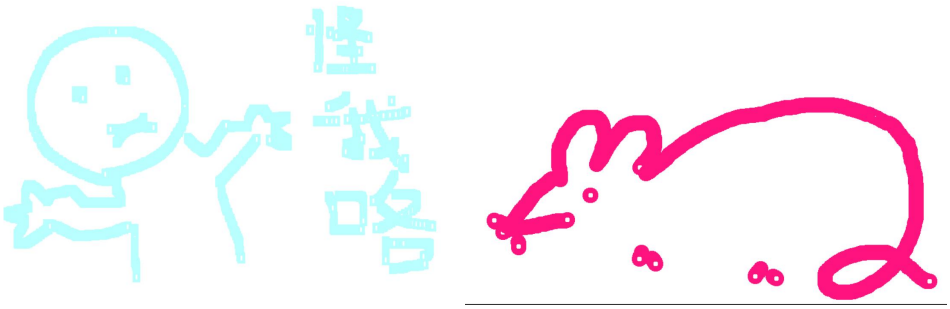
四. MFC鼠标绘制-学生作业展示

最后展示学生做的成果，虽然代码非常简单，原理也很简单，但是学生做得真的挺好的，原来编程还可以这样上啊，一方面提升学生的学习兴趣，另一方面增加他们的编程能力。









还是那句话，非常佩服学生的创造力及想象力吧！而且编程课原来可以这么进行，提升学生的编程能力的同时也培养了学生的兴趣。希望文章对你有所帮助~

(By:Eastmount 2016-11-20 半夜2点半 http://blog.csdn.net/eastmount/)