MFC之三

周艳

西南交通大学电气工程学院

西南交通大学

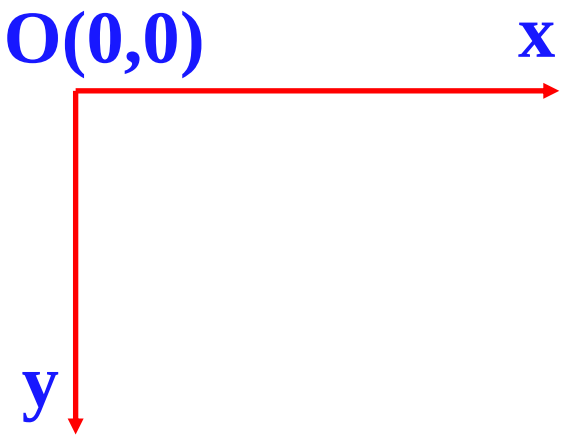
Windows绘图基础

- Windows提供图形设备接口GDI(Graphics Device Interface) 负责管理用户绘图操作时功能的转换。
- 用户通过调用GDI函数与设备打交道，GDI通过不同设备提供的驱动程序将绘图语句转换为对应的绘图指令，避免了直接对硬件进行操作，从而实现所谓的设备无关性。

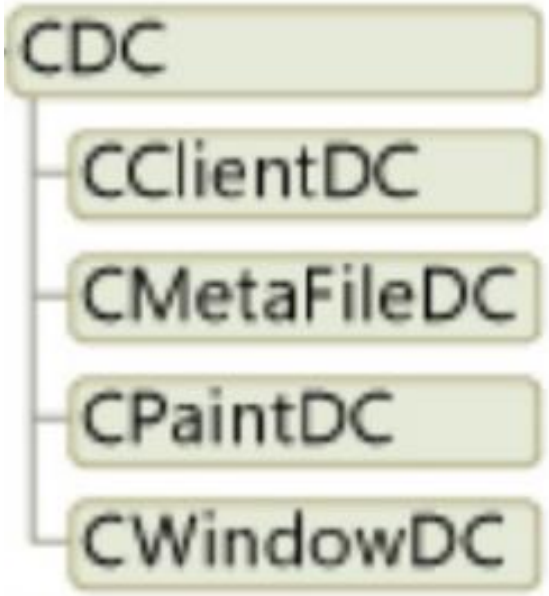
设备环境

- 为实现设备无关性，应用程序的输出不直接面向显示器等物理设备，而是面向一个称之为**设备环境DC**(Device Context)的虚拟逻辑设备。
- 设备环境也称**设备描述表**或**设备上下文**，它保存了绘图操作中一些共同需要设置的信息，如当前的画笔、画刷、字体和位图等图形对象及其属性，以及颜色和背景等影响图形输出的绘图模式。
- 设备环境提供了一张画布和一些绘画的工具，可使用不同颜色的工具在上面绘制点、线、圆和文本。

- 在Windows中不使用DC无法进行输出，在使用任何GDI绘图函数之前，必须建立一个设备环境。
- 设备坐标系统：*x*轴自左至右，*y*轴从上到下，坐标原点在屏幕左上角。



- **MFC封装了DC，提供CDC类及它的子类以访问GDI。CDC类派生自CObject类。**



设备环境类	功能描述
CDC	所有设备环境类的基类，对 GDI 的所有绘图函数进行了封装；可用来直接访问整个显示器或非显示设备(如打印机等)的上下文
CPaintDC	CPaintDC 用于响应窗口重绘消息(WM_PAINT)的绘图输出，不仅可对客户区进行操作，还可以对非客户区进行操作
CClientDC	代表窗口客户区的设备环境，一般在响应非窗口消息并对客户区绘图时要用到该类
CWindowDC	代表整个窗口的设备环境，包括客户区和非客户区；除非要自己绘制窗口边框和按钮，否则一般不用它

CPaintDC、 CPaintDC、 CWindowDC的区别

- 当窗口的某个区域需要重绘时激发窗口重绘消WM_PAINT，相应消息处理函数CWnd::OnPaint将被调用。CPaintDC一般只用于OnPaint函数中，在处理完窗口重绘后，CPaintDC对象的析构函数把WM_PAINT消息从消息队列中清除，避免不断地重绘操作。坐标原点(0，0)是客户区的左上角。

- Windows系统发送WM_PAINT消息的时机
  - 第一次创建一个窗口时
  - 改变窗口的大小时
  - 把窗口从另一个窗口背后移出时
  - 窗口显示数据变化时，应用程序引发重绘操作

- CCIientDC用于特定窗口客户区（窗口中除边框、标题栏、菜单栏、状态栏外的中间部分）的输出，其构造函数中包含了GetDC，析构函数中包含了ReleaseDC，不需要显式释放DC资源。一般用于响应非重绘消息（如键盘和鼠标消息）的绘图操作。坐标原点(0，0)是客户区的左上角。
- CWindowDC在整个应用程序窗口上画图，而CClientDC和CPaintDC只能在客户区绘制图形；除非要自己绘制窗口边框和按钮，否则一般不用它。坐标原点(0，0)是屏幕的左上角。

OnDraw和OnPaint的区别

- OnDraw是CView类的成员函数，不响应消息；
- OnPaint是CWnd类的成员函数，响应WM_PAINT消息。
- CView默认调用的OnPaint函数如下：

```
void CView::OnPaint()
{
    CPaintDC dc(this);
    OnPrepareDC(&dc);
    OnDraw(&dc);//调用了OnDraw
}
```

CView默认调用的OnPaint函数调用了OnDraw函数。因而一般在OnDraw函数内添加绘图代码，完成绘图任务。

CDC类常用绘图函数

Arc：绘制一段弧

Ellipse：绘制椭圆或圆

MoveTo：将当前位置移动到指定位置

LineTo：从当前位置到指定位置画一条直线

Polyline：画连接指定点的折线段

PolyBezler：根据两个端点和两个控制点画贝塞尔曲线

Polygon：根据两个或两个以上的顶点绘制一个多边形

Rectangle：根据指定的左上角和右下角坐标绘制一个矩形

RoundRect：画圆角矩形

SetPixel：画一个点

……

画线LineTo

绘制一条从当前绘图位置到指定坐标点的直线段。

BOOL LineTo(int x, int y);

BOOL LineTo(POINT point);

参数为坐标点
默认从(0,0)开始画

移动MoveTo

将当前绘图位置移到指定的坐标点处。

CPoint MoveTo(int x, int y);

CPoint MoveTo(POINT point);

参数指定新的当前绘图位置坐标。返回值是CPoint对象实例(新的当前绘图位置坐标)

绘制矩形

参数为坐标点或矩形区域结构指针，CRect类也可以

BOOL Rectangle(int x1,int yl,int x2,int y2）；

BOOL Rectangle(LPCRECT lpRect);

- CRect类是对Windows结构RECT的封装。结构RECT表示一个矩形，定义：

```
Typedef   struct  tagRECT
{    LONG  left;  //矩形左上角顶点的横坐标
     LONG  top;  //矩形左上角顶点的纵坐标
     LONG  right;  //矩形右下角顶点的横坐标
     LONG  bottom;  //矩形右下角顶点的纵坐标
}RECT;
```

绘制椭圆或者圆

- BOOL Ellipse(int x1, int yl, int x2， int y2);

xl和yl指定椭圆或者圆的外接矩形的左上角点坐标，参数x2和y2指定椭圆或者圆的外接矩形的右下角点坐标。

- BOOL Ellipse(LPCRECT IpRect);

使用矩形区域结构来存放外接矩形的左上角和右下角坐标。 CRect类也可以。

绘图基础示例

编写一个单文档MFC应用程序MFCDRAW1，完成以下要求：

（1）在客户区中画最大的椭圆

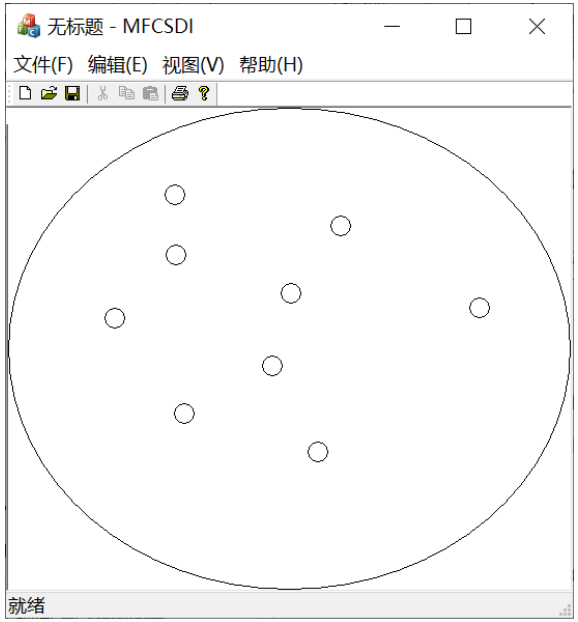
（2）当点击鼠标左键时，以鼠标左键点击的位置为中心，画一个半径为20个像素的圆。

绘制椭圆：修改CView类的虚函数OnDraw

```
void CMFCSDIView::OnDraw(CDC* pDC)
{
    ……
    CRect rect;
    this->GetClientRect(rect);//得到当前客户区域的大小
    pDC->Ellipse(rect);
}
```

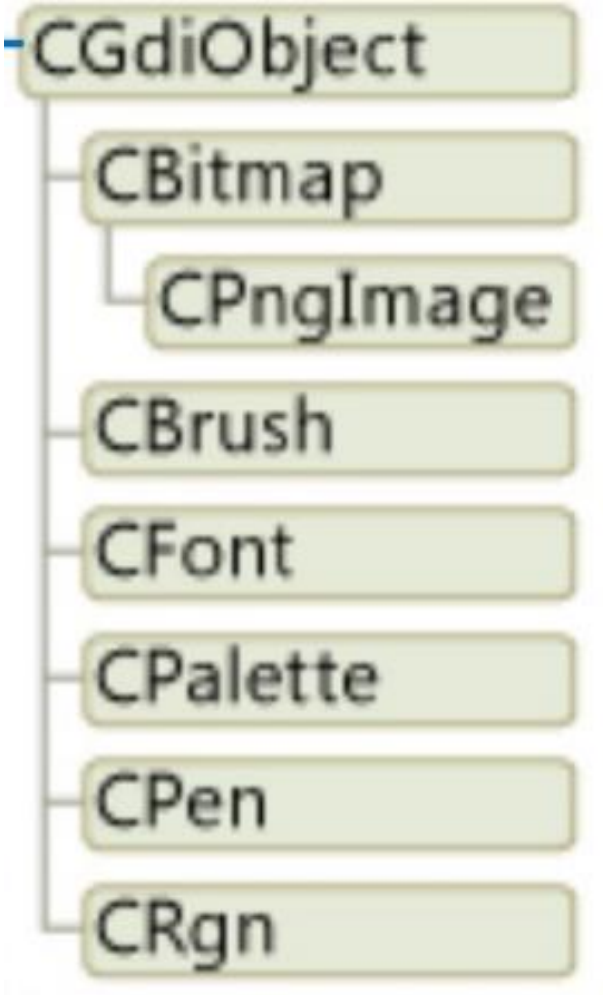
鼠标点击消息处理函数

```
void CMFCSDIView::OnLButtonDown(UINT nFlags, CPoint point)
{   ……..
    CClientDC dc(this);
    dc.Ellipse(point.x - 10, point.y - 10, point.x + 10, point.y + 10);
}
```



绘图工具类CGdiObject

绘图工具类主要包括画笔CPen、画刷CBrush、字体CFont、位图CBitmap和调色板CPalette等。



- CPen类：画笔，用于画线。默认的画笔用于绘制与一个像素等宽的黑色实线。
- CBrush类：画刷，填充一个封闭图形对象的内部区域，默认的画刷颜色是白色。
- CFont类：字体，用来绘制文本，可设置文字的大小、是否加粗、是否斜体、是否加下划线等。
- CBitmap类：位图，用于填充区域。
- CPalette类：调色板，包含系统可用的色彩信息，是应用程序和彩色输出设备环境(如显示器)的接口

选择GDI对象

成员函数SelectObject() 有多种重载形式，可以选择用户已定制好的画笔、画刷、字体和位图等不同类型的GDI对象。

```
CPen* SelectObject(CPen* pPen);

CBrush* SelectObject(CBrush* pBrush);

virtual CFont* SelectObject(CFont* pFont);

CBitmap* SelectObject(CBitmap* pBitmap);
```

绘图颜色

- DWORD类型的COLORREF数据用于存放颜色值"0x00bbggrr”
  - 低位字节存红色强度值，第2个字节存绿色强度值，第3个字节存蓝色强度值，高位字节存0
- 可用RGB宏设置颜色值，将红、绿、蓝分量值转换为COLORREF类型的颜色数据
  - COLORREF RGB(

    BYTE byRed,//red component of color

    BYTE byGreen，//green component of color

    BYTE bvBlue//blue component of color);

- **RGB宏的使用：**

很多涉及到颜色的GDI函数都需要使用COLORREF类型的参数，如

```
COLORREF rgbBkClr=RGB(192,192,192);//定义灰色
pDC->SetBkCorlor(rgbBkClr);//背景色为灰色
pDC->SetTextColor(RGB(0,0,255)); //文本颜色为蓝色
```

使用画笔

- 当用户创建一个用于绘图的设备环境时，该设备环境自动提供了一个宽度为一个像素单位、风格为实黑线(BLACK_PEN)的缺省画笔。
- 在设备环境使用自己的画笔绘图
  - 首先需要创建一个指定风格的画笔
  - 然后将创建的画笔选入设备环境
  - 最后，在使用该画笔绘图结束后需要释放该画笔。

1、创建画笔

- 最简单的方法是调用CPen类的一个带参数的构造函数来构造一个CPen类画笔对象。

```
CPen PenNew(PS_DASH,1,RGB(255,0,0))；//创建红色虚线画笔
```

- 第二种方法：先构造一个没有初始化的CPen类画笔对象，然后调用成员函数创建定制的画笔工具

```
Cpen PenNew;
PenNew.CreatePen(PS_DASH,1,RGB(255,0,0));
```

参数与上面的CPen类构造函数完全一样

当画笔对象的声明与创建不在同一个地方时(如需要多次改变画笔)只有采用这种方法。

画笔样式

样　式	说　明	样　式	说明
PS_SOLID	实线	PS_DASHDOTDOT	双点划线
PS_DOT	点线	PS_NULL	空的边框
PS_DASH	虚线	PS_INSIDEFRAME	边框实线
PS_DASHDOT	点划线		

2、选择创建的画笔

- 创建画笔后必须调用成员函数SelectObject将创建的画笔选入当前设备环境。如果选择成功，SelectObject将返回以前画笔对象的指针。选择新的画笔时应该保存以前的画笔对象。

```
CPen*pPenOld;
pPenOld=pDC->SelectObject(&PenNew);
```

3、还原画笔

- 绘图完成后，应通过调用成员函数SelectObject恢复设备环境以前的画笔工具，并通过调用成员函数DeleteObject释放GDI对象所占的内存资源。

```
pDC->SelectObject(pPenOld);
//恢复设备环境DC中原来的画笔
PenNew.DeleteObject();    //删除底层的GDI对象
```

每个图形设备接口对象要占用一个HDC句柄，而可用的句柄数量有限，如果用完后未及时释放，积累下去将导致严重的运行错误。

```
CPen *pNewPen = new CPen;
pNewPen->CreatePen(PS_DASHDOT,3,RGB(255,0,0));
CPen *pOldPen = pDC->SelectObject(pNewPen);
// 用新创建的画笔绘图
pDC->MoveTo(10,20);
pDC->LineTo(200,20);
pDC->LineTo(200,80);
pDC->SelectObject(pOldPen); //恢复设备描述表中原有的笔
delete pNewPen;
```

绘图示例：菜单控制

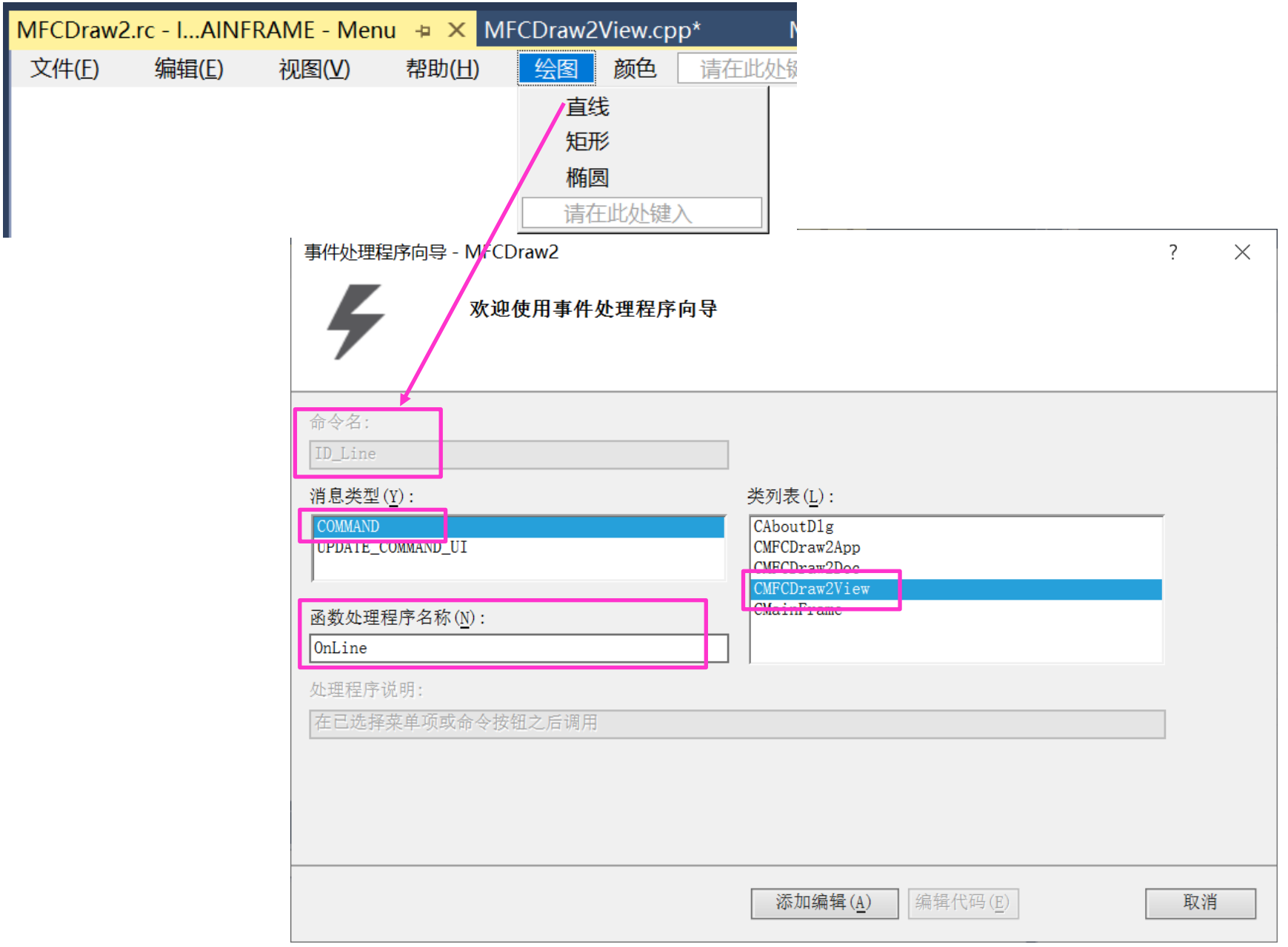
可以先在视图类中添加两个变量并初始化

MFCDraw2.rc - I...AINFRAME - Menu　MFCDraw2View.cpp*　MFCDraw2View.h

CMFCDraw2View　OnEcli()

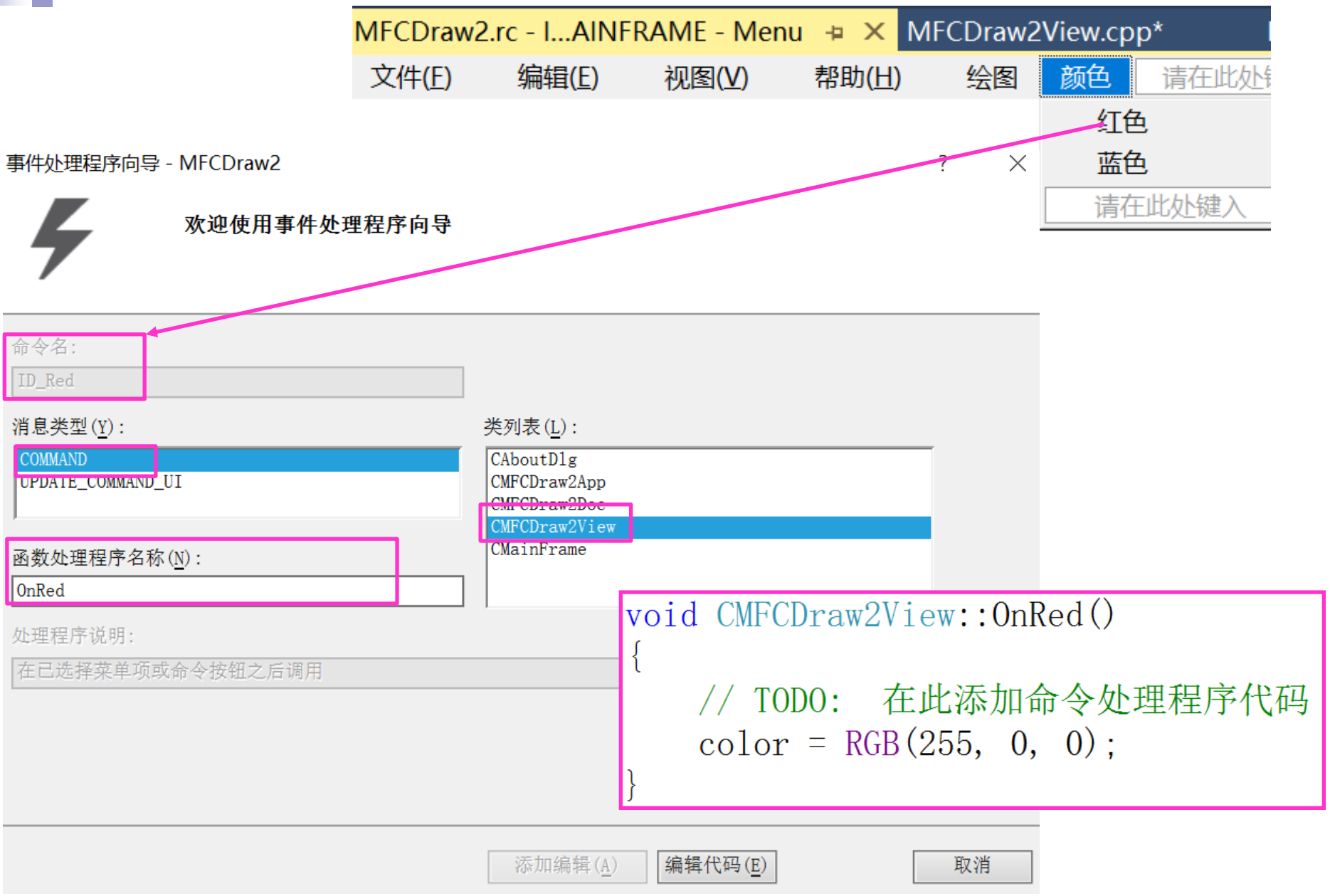
```
public:
    int shape;
    COLORREF color;
```

```
CMFCDraw2View::CMFCDraw2View()
: shape(0)
{
    // TODO:  在此处添加构造代码
    color = RGB(0, 255, 0);
}
```

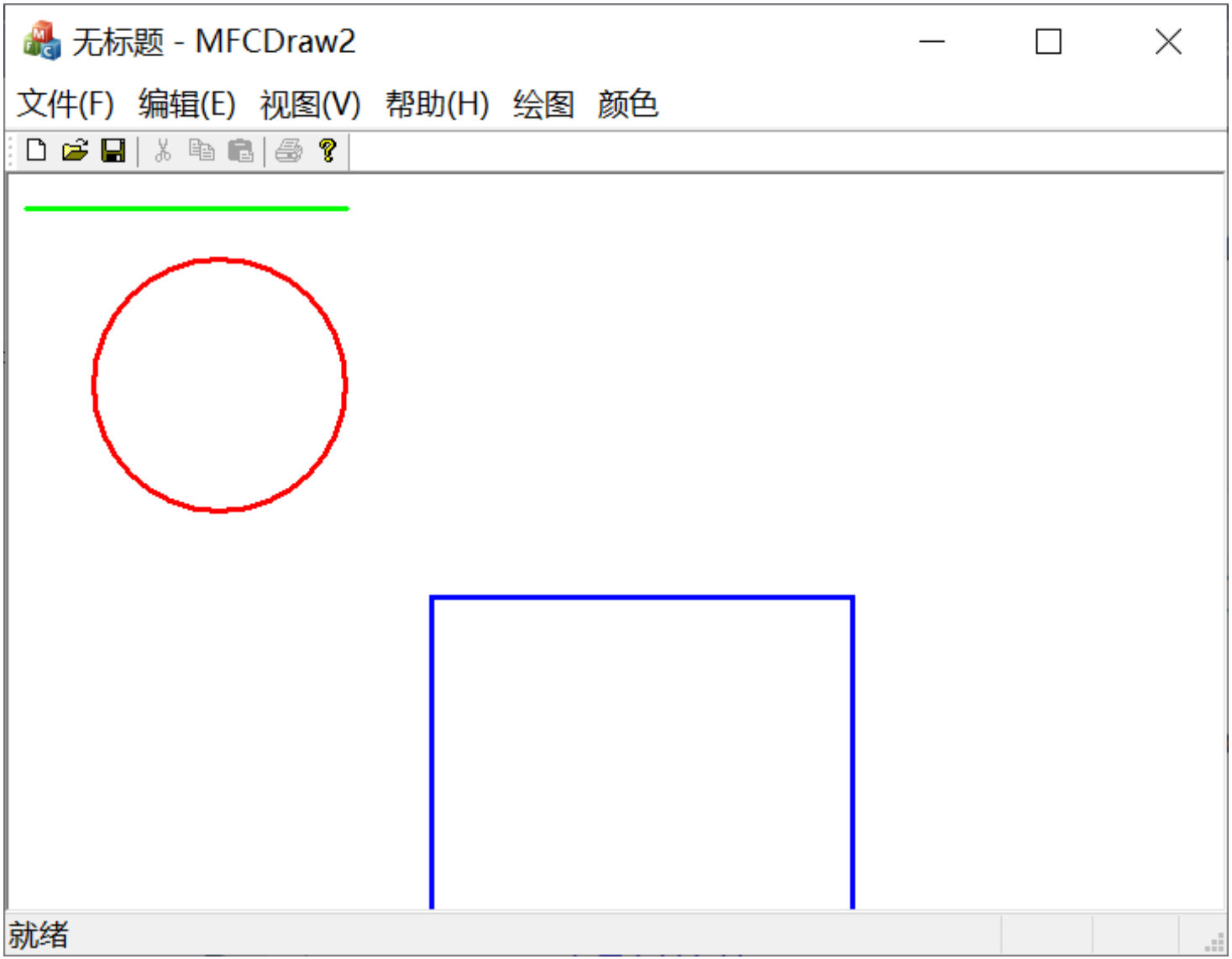

MFCDraw2.rc - I...AINFRAME - Menu
MFCDraw2View.cpp*
文件(F)
编辑(E)
视图(V)
帮助(H)
绘图
颜色
请在此处键
直线
矩形
椭圆
请在此处键入
事件处理程序向导 - MFCDraw2
欢迎使用事件处理程序向导
命令名:
ID_Line
消息类型(Y):
COMMAND
UPDATE_COMMAND_UI
类列表(L):
CAboutDlg
CMFCDraw2App
CMFCDraw2Doc
CMFCDraw2View
CMainFrame
函数处理程序名称(N):
OnLine
处理程序说明:
在已选择菜单项或命令按钮之后调用
添加编辑(A)
编辑代码(E)
取消

```
void CMFCDraw2View::OnLine()
{
    // TODO:  在此添加命令处理程序代码
    shape = 1;
    Invalidate(0);
}
```

- 使整个窗口客户区无效，此时需要重绘
  - Invalidate(false)：不擦除背景，直接绘图
  - Invalidate(true)：擦除背景后再绘图

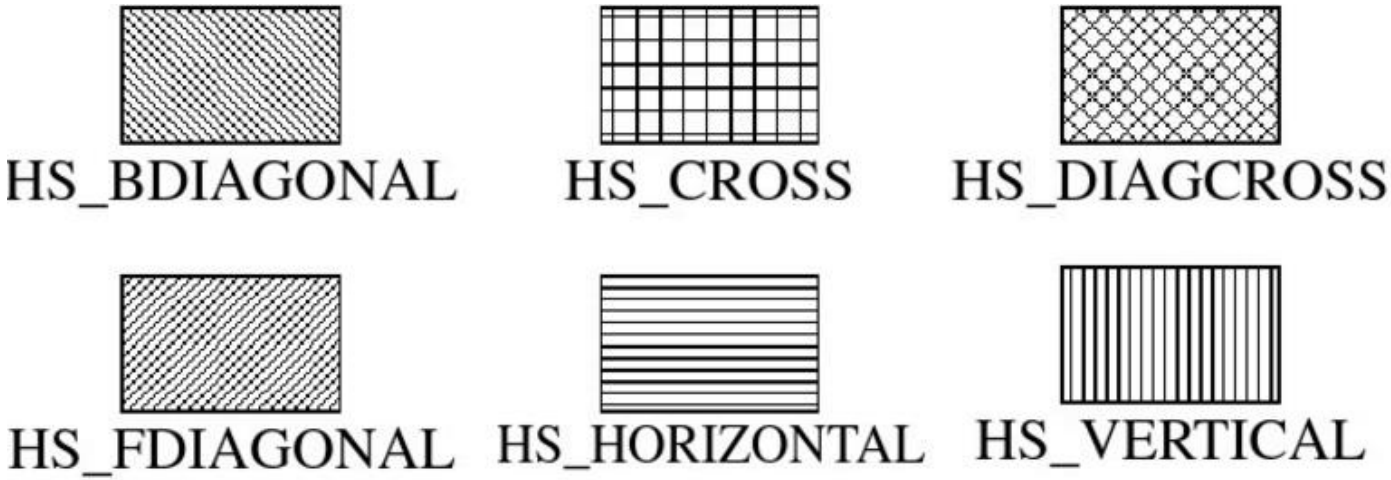

MFCDraw2.rc - I...AINFRAME - Menu
MFCDraw2View.cpp*
文件(F)
编辑(E)
视图(V)
帮助(H)
绘图
颜色
请在此处键入
红色
蓝色
请在此处键入
事件处理程序向导 - MFCDraw2
欢迎使用事件处理程序向导
命令名:
ID_Red
消息类型(Y):
COMMAND
UPDATE_COMMAND_UI
类列表(L):
CAboutDlg
CMFCDraw2App
CMFCDraw2Doc
CMFCDraw2View
CMainFrame
函数处理程序名称(N):
OnRed
处理程序说明:
在已选择菜单项或命令按钮之后调用
添加编辑(A)
编辑代码(E)
取消
void CMFCDraw2View::OnRed()
{
// TODO: 在此添加命令处理程序代码
color = RGB(255, 0, 0);
}

```
void CMFCDraw2View::OnDraw(CDC* pDC)
{……
        CPen *pNewPen = new CPen;
        pNewPen->CreatePen(PS_DASHDOT, 3, color);
        CPen *pOldPen = pDC->SelectObject(pNewPen);
        switch (shape)
        {
        case 1: pDC->MoveTo(10, 20);
                pDC->LineTo(200, 20);
                break;
        ……
        }
        pDC->SelectObject(pOldPen); //恢复设备描述表中原有的笔
        delete pNewPen;
```


无标题 - MFCDraw2
文件(F) 编辑(E) 视图(V) 帮助(H) 绘图 颜色
就绪

使用画刷

- 画刷是填充图形的工具。画刷有三种属性：
  - 填充色和画笔颜色一样，使用COLORREF颜色类型
  - 填充图案通常是用户定义的8X8位图
  - 填充样式往往是以HS_为前缀的标识，如图所示：



```
CBrush *pNewBrush = new CBrush;//使用前初始化画刷
pNewBrush->CreateSolidBrush(RGB(180, 70, 230));
//创建纯色画刷
CBrush*pOldBrush = pDC->SelectObject(pNewBrush);
//选择新画刷
pDC->Rectangle(40,60,200,100);//绘制矩形
pDC->SelectObject(pOldBrush);//恢复设备描述表中原有的画刷
delete pNewBrush;//删除新画刷
```

模拟时钟

涉及指针运动算法、屏幕重绘方法、定时器消息、鼠标消息、菜单命令、对话框、画笔/画刷、显示文字等

```
CBrush bushHour(RGB(255, 0, 0));
CBrush bushMinute(RGB(255, 255, 255));
int Redius = 200;//时钟半径
CPoint middle(250,250);//时钟表盘中心点位
for (int i = 0; i < 60; i++)
{    CPoint pt;
     double angle = i*3.14 / 30;
     pt.x = middle.x + (int)(Redius * cos(angle));
     pt.y = middle.y + (int)(Redius * sin(angle));
     if (i % 5 == 0)//绘制时钟点位
    {      pDC->SelectObject(&bushHour);
           pDC->Rectangle(pt.x - 5, pt.y - 5, pt.x + 5, pt.y + 5);    }
    else//绘制分针位置
    {      pDC->SelectObject(&bushMinute);
           pDC->Ellipse(pt.x - 3, pt.y - 3, pt.x + 3, pt.y + 3);      }
}
//绘制中心点位
pDC->SelectObject(&bushHour);
pDC->Rectangle(middle.x - 5, middle.y - 5, middle.x + 5, middle.y + 5);
```

绘制表盘标记

示例：绘制指针，参数为长度，角度，和颜色

```
void CMFCSDIView::DrawPt(CDC* pDC, int Length, int Degree,
COLORREF Color)
{
    CPoint point;
    CPoint middle(250, 250);//时钟表盘中心点位
    point.x = (int)(middle.x + sin((3.14 / 30) * Degree) * Length);
    point.y = (int)(middle.y - cos((3.14 / 30) * Degree) * Length);

    CPen pen(PS_SOLID, 0, RGB(0, 0, 255));
    CPen* pOldPen = pDC->SelectObject(&pen);
    pDC->SelectObject(&pen);
    pDC->MoveTo(middle.x, middle.y);
    pDC->LineTo(point.x, point.y);
    pDC->SelectObject(pOldPen);
}
```

示例：画当前时间的时针和分针

```
CClientDC dc(this);
int Minute, Hour;
CTime time = CTime::GetCurrentTime();
Minute = time.GetMinute();
Hour = time.GetHour() % 12;

DrawPt(&dc, 90, Hour * 5 + (int)(Minute / 12), RGB(255, 0, 0));
DrawPt(&dc, 150, Minute, RGB(0, 0, 255));
```