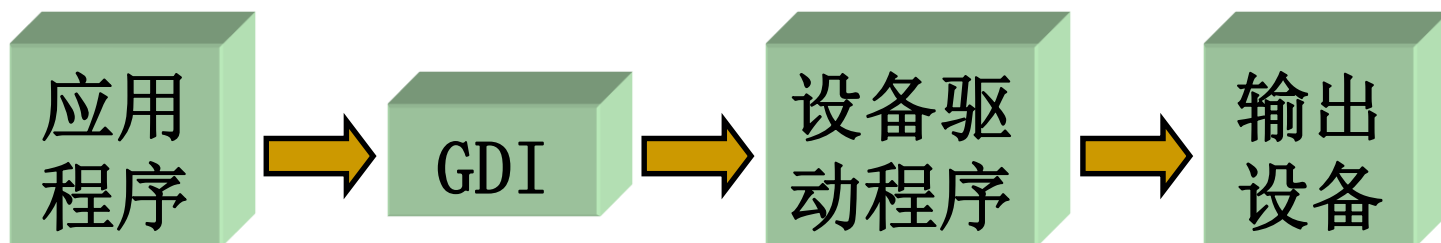


第4章 设备环境与视图显示

- 设备环境类
- 绘图工具与绘图区域
- 绘图模式与映射关系
- 绘图函数
- 文本与字体处理
- 位图与图标处理
- 光标处理

设备环境与GDI

- 设备环境用于Windows程序的可视化输出
- 设备环境提供一张画布，可书写文字，绘制点、图形、位图等
- 设备环境由图形设备接口（GDI）提供



设备环境类(1)

- CDC: 设备环境的基类
- CClientDC: 客户区(不包括边框、标题栏、菜单栏、状态栏)的设备环境类
- CWindowDC: 程序窗口的设备环境类
- CPaintDC: WM_PAINT专用的设备环境类

设备环境类(2)

例4-1

■ 画线程序的例子(CDC)

```
void CTestView::OnLButtonDown(UINT nFlags,
CPoint point)
{
    CDC *pDC=GetDC();
    pDC->MoveTo(start);
    pDC->LineTo(point);
    ReleaseDC(pDC);
}
```

设备环境类(3)

■ 画线程序的例子(CClientDC)

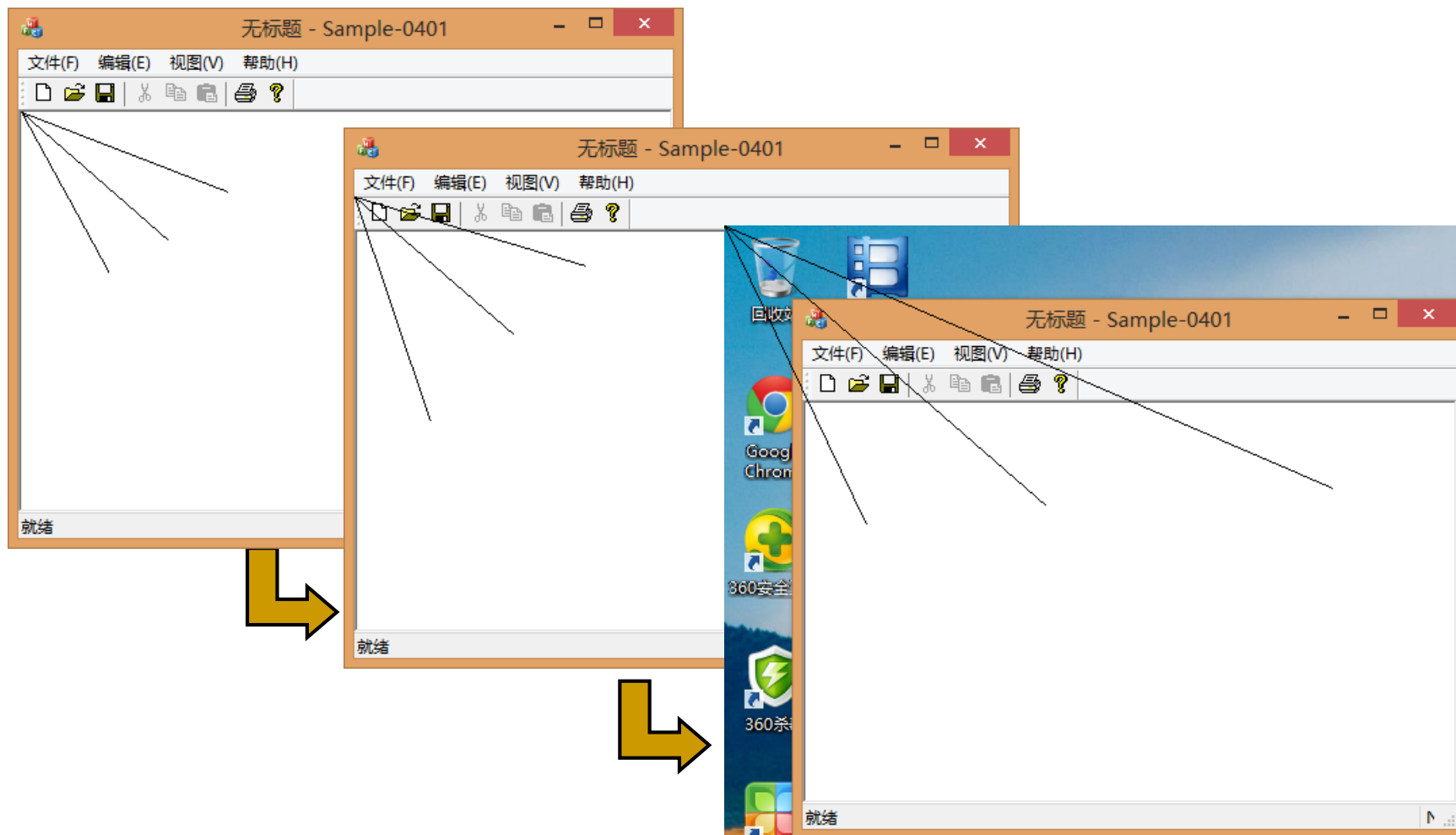
```
void CTestView::OnLButtonDown(UINT nFlags,
CPoint point)
{
    CClientDC dc(this);
    //CClientDC dc(GetParent());
    dc.MoveTo(start);
    dc.LineTo(point);
}
```

设备环境类(4)

■ 画线程序的例子(CWindowDC)

```
void CTestView::OnLButtonDown(UINT nFlags,
CPoint point)
{
    CWindowDC dc(this);
    //CWindowDC dc(GetParent());
    //CWindowDC dc(GetDesktopWindow());
    dc.MoveTo(start);
    dc.LineTo(point);
}
```

设备环境类(5)



绘图方法

- 通过CClientDC构造对象

```
CClientDC dc(this);  
dc.Ellipse(10, 10, 200, 200);
```

- 通过GetDC获得设备环境指针

```
CDC* pDC=GetDC();  
pDC->Ellipse(10, 10, 200, 200);  
ReleaseDC(pDC);
```


CPoint、CSize与CRect

- CPoint: 封装POINT结构的类
 - ✓ 定义一个点坐标, 成员为x、y
- CRect: 封装RECT结构的类
 - ✓ 定义一个矩形区域, 成员为left、right、top、bottom
- CSize: 封装SIZE结构的类
 - ✓ 定义一个矩形区域尺寸, 成员为cx、cy

绘图工具(1)

- 设备文本类(CDC)提供绘图工具，例如CPen、CBrush与CFont等，基类是CGdiObject
- 默认的画笔是黑色，宽度是一个像素，默认的画刷是白色
- 颜色由RGB值来指定
 - ✓ RGB (Red, Green, Blue)
 - ✓ 0x 00 FF FF FF
 B G R

绘图工具 (2)

■ 库存画刷与画笔

画刷类型	说明	画刷类型	说明
BLACK_BRUSH	黑色画刷	HOLLOW_BRUSH	透明画刷
DKGRAY_BRUSH	深灰画刷	NULL_BRUSH	空画刷
GRAY_BRUSH	灰色画刷	BLACK_PEN	黑色画笔
LTGRAY_BRUSH	浅灰画刷	WHITE_PEN	白色画笔
WHITE_BRUSH	白色画刷	NULL_PEN	空画笔

绘图工具(3)

例4-2

■ SelectStockObject的例子

```
pDC->MoveTo(10, 100);  
pDC->LineTo(550, 100);  
pDC->SelectStockObject(LTGRAY_BRUSH);  
pDC->Ellipse(50, 50, 150, 150);  
pDC->SelectStockObject(DKGRAY_BRUSH);  
pDC->Ellipse(200, 50, 350, 150);  
pDC->SelectStockObject(NULL_BRUSH);  
pDC->Rectangle(400, 50, 500, 150);
```

自定义画笔(1)

- 单步构造方法 是画笔样式，详细见下一页

```
CPen newPen(PS_SOLID, 5, RGB(0, 0, 255));
```

- 两步构造方法

```
CPen newPen;  
newPen.CreatePen(PS_SOLID, 5, RGB(0, 0, 255));
```

- 画笔风格是实线，宽度是5，颜色是蓝色

自定义画笔 (2)

■ 画笔样式

画笔样式	说明
PS_SOLID	实线画笔
PS_DASH	划线(虚线)画笔
PS_DASHDOT	点划线画笔
PS_DASHDOTDOT	双点划线画笔
PS_DOT	点线画笔
PS_NULL	空画笔

自定义画笔 (3)

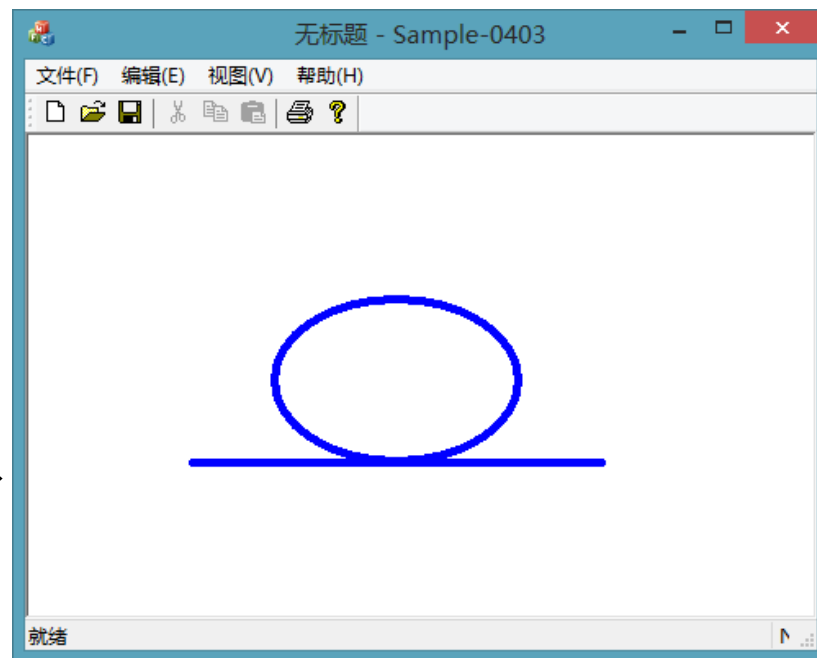
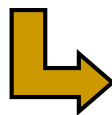
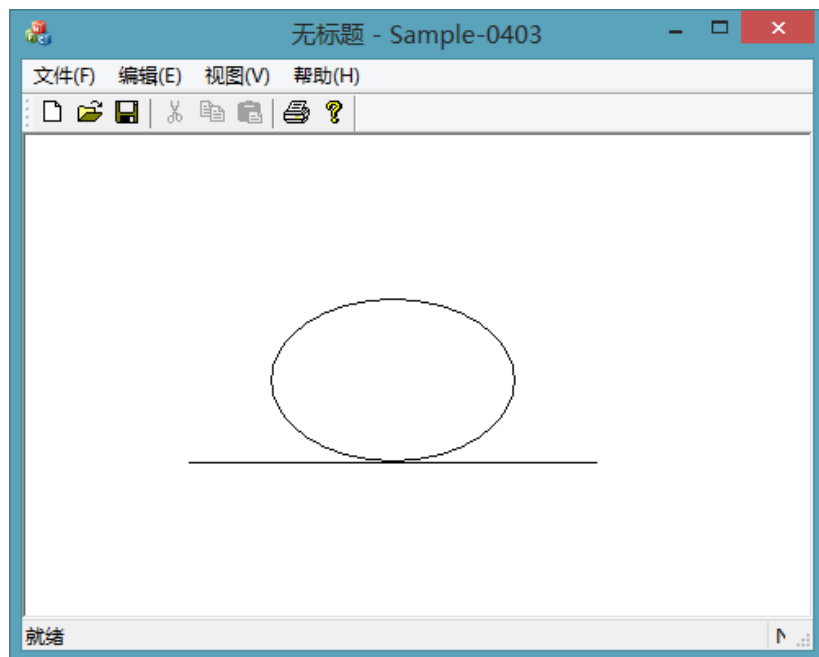
例4-3

■ 自定义画笔的例子

自定义画笔的应用

```
CPen newPen;  
newPen.CreatePen(PS_SOLID, 5, RGB(0, 0, 255));  
pDC->SelectObject(newPen);  
pDC->Ellipse(150, 100, 300, 200);  
pDC->MoveTo(100, 200);  
pDC->LineTo(350, 200);
```

自定义画笔(4)



自定义画刷(1)

画笔——线
画刷——要填充的！！！！

■ 实心(solid) 风格

✓ `CreateSolidBrush(COLORREF crColor);`

■ 网格(hatched) 风格

是网格样式，详细见下一页

✓ `CreateHatchBrush(int nIndex, COLORREF crColor);`

■ 模式(patterned) 风格

✓ `CreatePatternBrush(CBitmap *pBitmap);`

自定义画刷(2)

网格样式	说明
HS_CROSS	十字线填充
HS_HORIZONTAL	水平线填充
HS_VERTICAL	垂直线填充
HS_FDIAGONAL	斜线填充
HS_BDIAGONAL	反斜线填充
HS_DIAGCROSS	斜十字线填充

自定义画刷 (3)

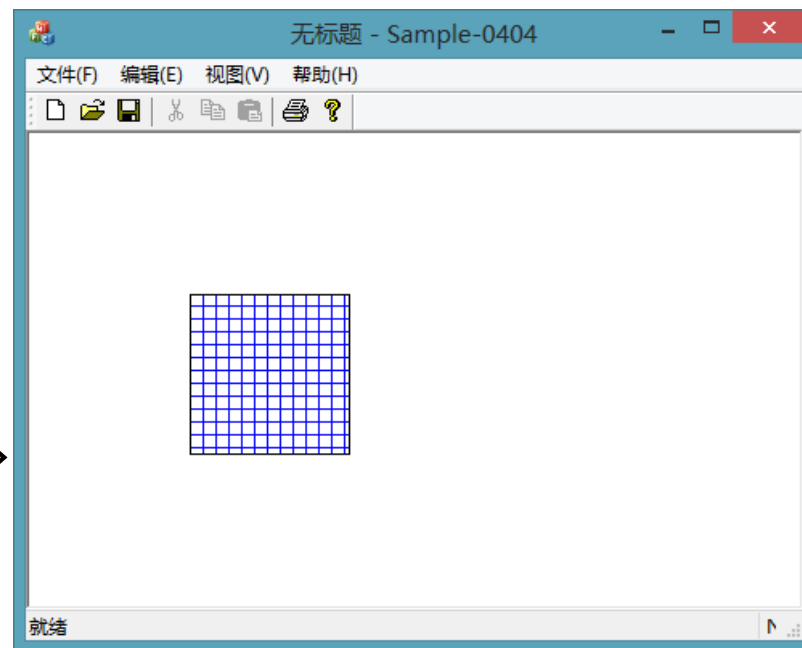
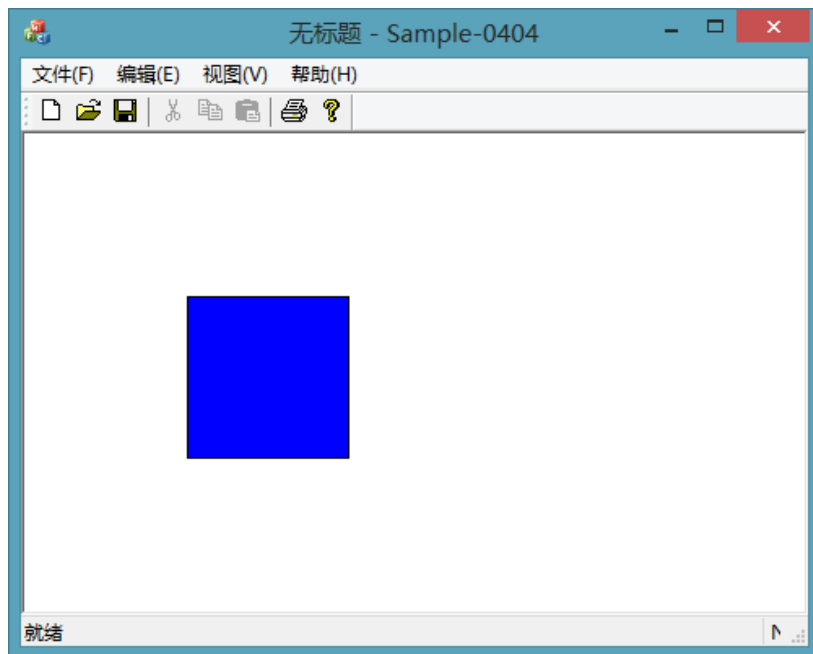
例4-4

■ 自定义画刷的例子

自定义画刷的应用

```
CBrush newBrush;  
newBrush.CreateSolidBrush( RGB(0, 0, 255) );  
newBrush.CreateHatchBrush( HS_CROSS, RGB  
(0, 0, 255) );  
pDC->SelectObject( newBrush );  
pDC->Rectangle( 100, 100, 200, 200 );
```

自定义画刷(4)



哪个函数将自定义绘图对象选入设备环境？

☐ A SelectStockObject

☒ B SelectObject

☐ C SelectClipRgn

☐ D SelectPalette

绘图函数(1)

- 绘图函数的坐标是逻辑单位。默认的左上角坐标为(0, 0)，逻辑单位为像素
- 绘图函数
 - ✓ 设置像素(SetPixel)、当前位置(Moveto)、画直线(Lineto)、画弧线(Arc)、画矩形(Rectangle)、画椭圆(Ellipse)、画饼图(Pie)、画多边形(Polygon)

绘图函数(2)

例4-5

■ 圆角矩形

```
pDC->Rectangle(100, 100, 300, 200);  
pDC->RoundRect(100, 100, 300, 200, 50, 40);
```

■ 弧线

```
pDC->Rectangle(100, 100, 300, 200);  
pDC->MoveTo(200, 150); pDC->LineTo(300, 175);  
pDC->MoveTo(200, 150); pDC->LineTo(125, 100);  
pDC->Arc(100, 100, 300, 200, 300, 175, 125, 100);
```

绘图函数(3)

■ 饼图

```
pDC->Rectangle(100, 100, 300, 200);  
pDC->Pie(100, 100, 300, 200, 300, 100, 100, 100);
```

■ 多边形

```
CPoint point[4];  
point[0].x=100;   point[0].y=100;  
point[1].x=200;   point[1].y=200;  
point[2].x=200;   point[2].y=100;  
point[3].x=100;   point[3].y=200;  
pDC->Polygon(point, 4);
```


绘图函数(4)

- 用画刷填充矩形，不绘制边框(FillRect)

```
CBrush newBrush;  
newBrush.CreateSolidBrush(RGB(255, 0, 0));  
pDC->FillRect(CRect(100, 100, 300, 200),  
&newBrush);
```

- 用画刷绘制边框，不填充内部(FrameRect)

```
CBrush newBrush;  
newBrush.CreateSolidBrush(RGB(255, 0, 0));  
pDC->FrameRect(CRect(100, 100, 300, 200),  
&newBrush);
```

绘图模式(1)

- 绘图模式指定绘图工具(画笔、画刷)颜色和显示颜色的处理方式
 - ✓ `SetROP2(int nDrawMode)`
- 绘图模式
 - ✓ `R2_COPYPEN`(绘图工具颜色)
 - ✓ `R2_NOT`(背景颜色取反)
 - ✓ `R2_XORPEN`(背景与绘图工具颜色异或)

绘图模式 (2)

例4-6

■ R2_NOT的例子

- ✓ 蓝色 (0x00FF0000) 取反为黄色 (0x0000FFFF)
- ✓ 白色 (0x00FFFFFF) 取反为黑色 (0x00000000)

■ 在CTestView::OnDraw() 中

```
CBrush newBrush;  
newBrush.CreateSolidBrush(RGB(0, 0, 255));  
pDC->SelectObject(newBrush);  
pDC->Rectangle(100, 100, 300, 200);
```

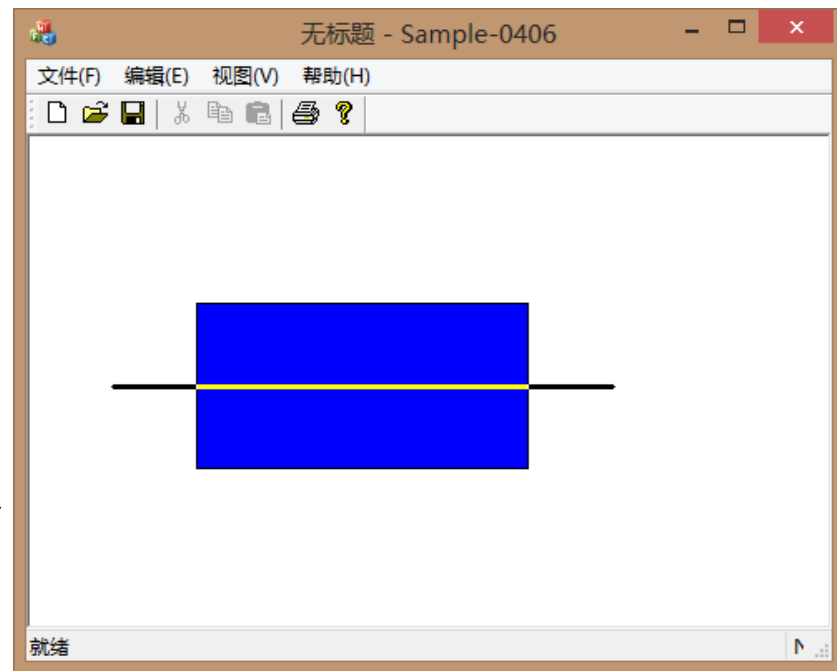
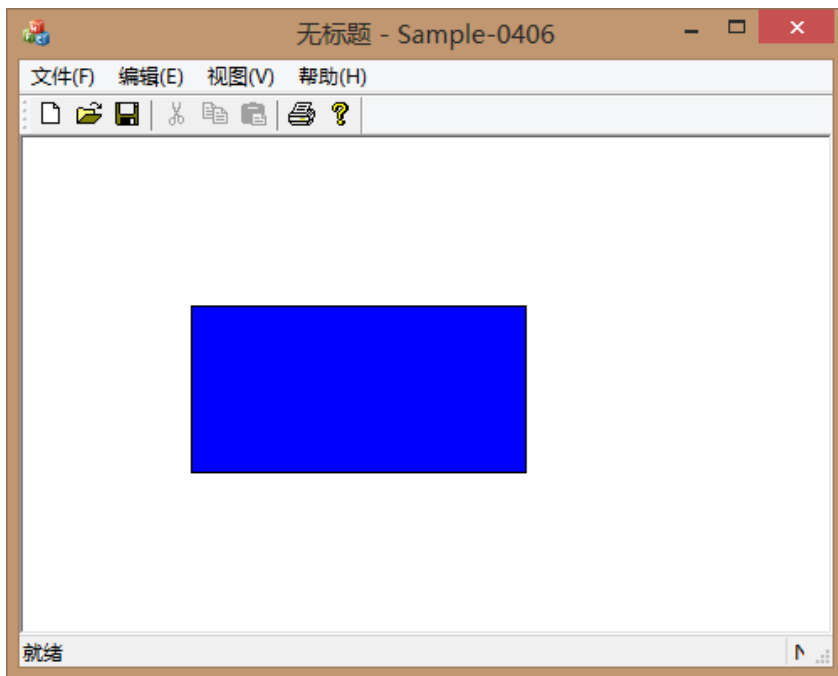
绘图模式 (3)

- 在CTestView::OnLButtonDown() 中

```
CDC* pDC=GetDC();  
CPen newPen;  
newPen.CreatePen(PS_SOLID, 3, RGB(0, 0, 255));  
pDC->SelectObject(newPen);  
pDC->SetROP2(R2_NOT);  
pDC->MoveTo(50, 150);  
pDC->LineTo(350, 150);  
ReleaseDC(pDC);
```

将自定义绘图对象（画笔）选入

绘图模式(4)



绘图模式 (5)

■ R2_XORPEN的例子

✓ $0x00FF0000 \wedge 0x00FF0000 = 0x00000000$ (黑色)

✓ $0x00FFFFFF \wedge 0x00FF0000 = 0x0000FFFF$ (黄色)

■ 在CTestView::OnDraw() 中

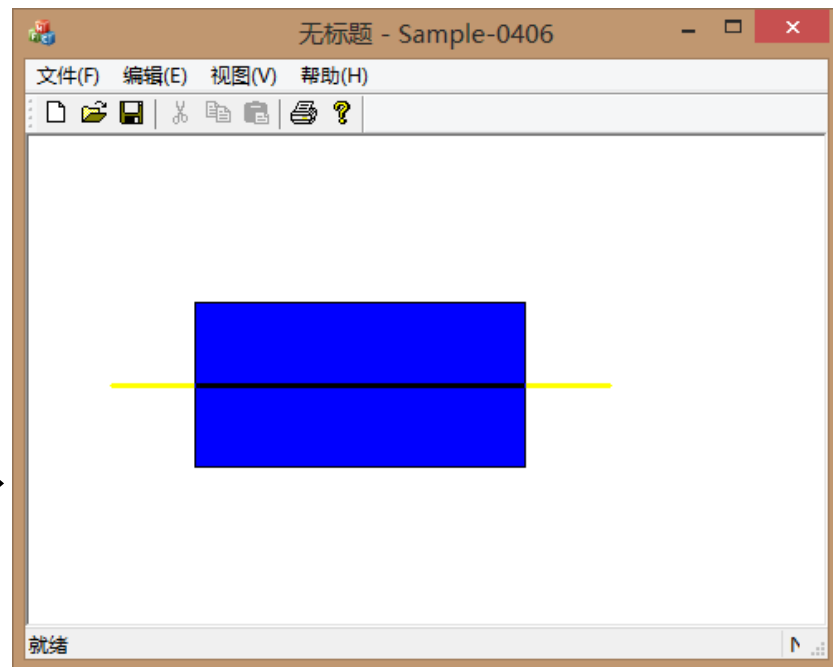
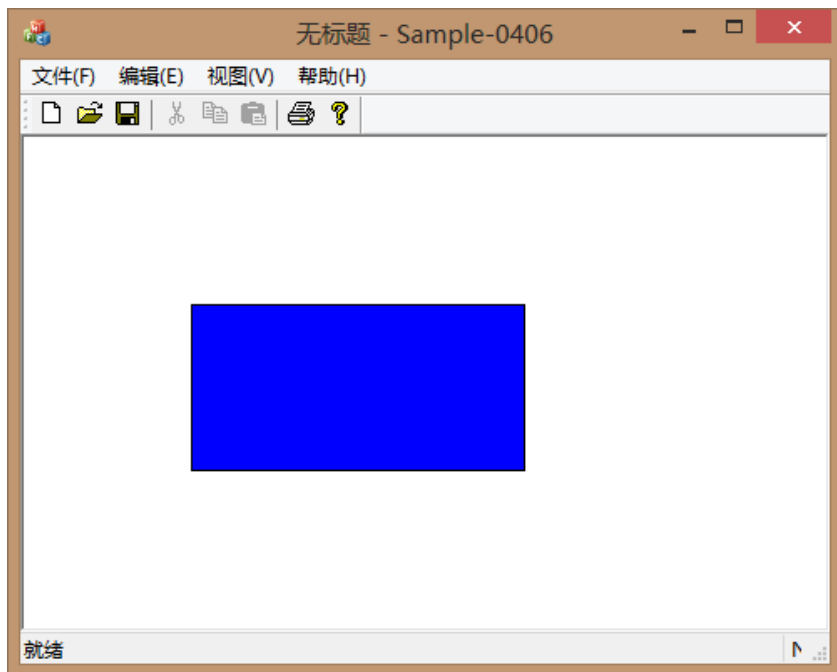
```
CBrush newBrush;  
newBrush.CreateSolidBrush( RGB(0, 0, 255) );  
pDC->SelectObject(newBrush);  
pDC->Rectangle(100, 100, 300, 200);
```

绘图模式 (6)

- 在CTestView::OnLButtonDown() 中

```
CDC* pDC=GetDC();  
CPen newPen;  
newPen.CreatePen(PS_SOLID, 3, RGB(0, 0, 255));  
pDC->SelectObject(newPen);  
pDC->SetROP2(R2_XORPEN);  
pDC->MoveTo(50, 150);  
pDC->LineTo(350, 150);  
ReleaseDC(pDC);
```

绘图模式(7)



CRgn与区域(1)

- CRgn类封装绘图区域
 - ✓ CreateRectRgn、CreateEllipticRgn、CreatePolygonRgn等
 - ✓ CombineRgn合并区域
- CDC类在区域中绘图操作
 - ✓ FillClipRgn填充区域，SelectClipRgn设置区域，SelectObject选入设备环境

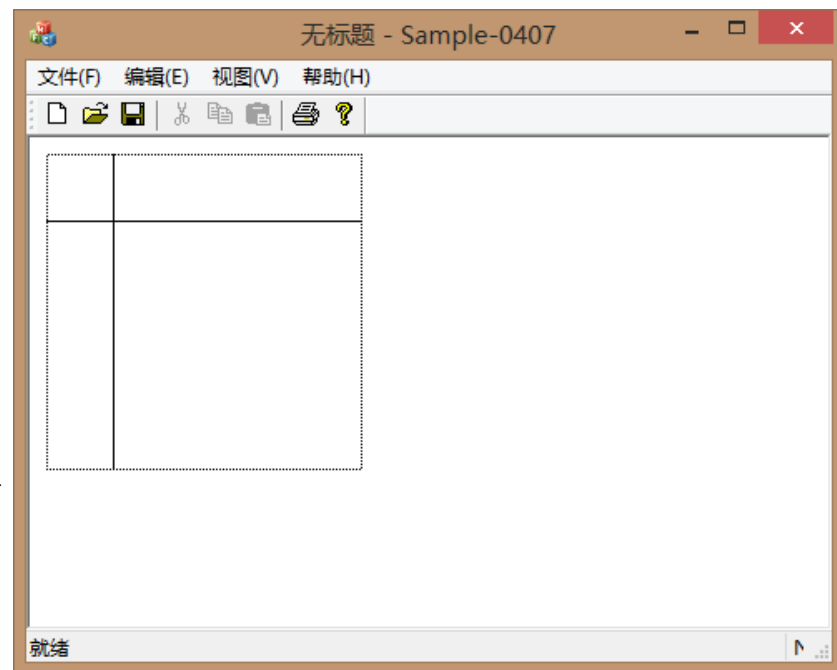
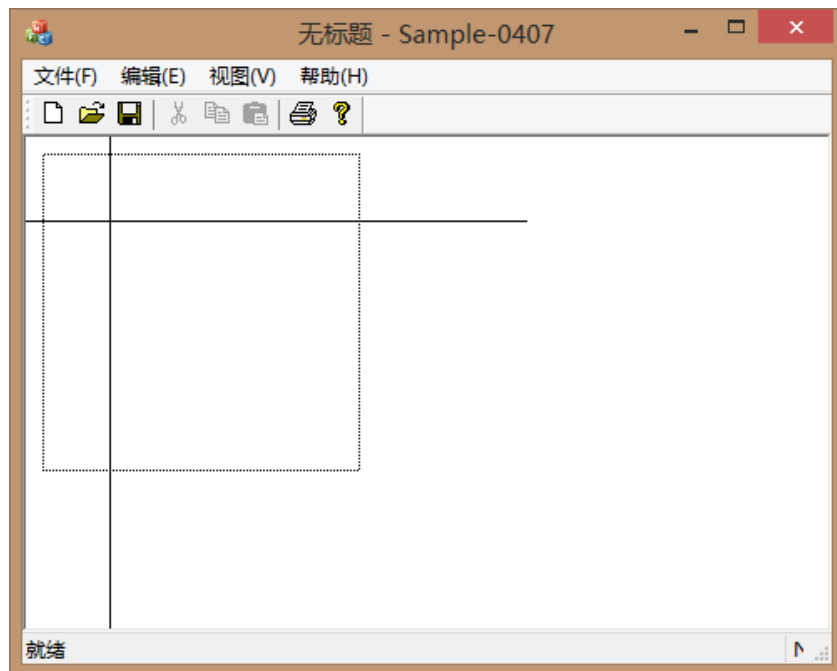
CRgn与区域(2)

例4-7

- 在CTestView::OnDraw()中 限制在这个矩形里

```
pDC->DrawFocusRect(CRect(10, 10, 200, 200));  
CRgn newRgn;  
newRgn.CreateRectRgn(10, 10, 200, 200);  
pDC->SelectObject(newRgn);  
pDC->MoveTo(0, 50);  
pDC->LineTo(300, 50);  
pDC->MoveTo(50, 0);  
pDC->LineTo(50, 300);
```

CRgn与区域(3)



映射模式(1)

- Windows系统提供两类坐标：逻辑坐标与设备坐标(屏幕坐标、窗口坐标与客户区坐标)
- GDI函数的坐标为逻辑坐标，消息处理函数的坐标为设备坐标
- 映射模式定义逻辑坐标与设备坐标的关系
 - ✓ 约束映射模式：比例因子、轴向固定
 - ✓ 非约束映射模式：比例因子、轴向不固定

映射模式(2)

■ SetMapMode设置映射模式

- ✓ MM_TEXT: 默认映射模式, 每个单位映射为一个像素, X轴向右、Y轴向下
- ✓ MM_HIENGLISH、MM_LOENGLISH : 每个单位映射成0.001英寸, X轴向右、Y轴向上
- ✓ MM_HIMETRIC、MM_LOMETRIC : 每个单位映射成0.01毫米, X轴向右、Y轴向上

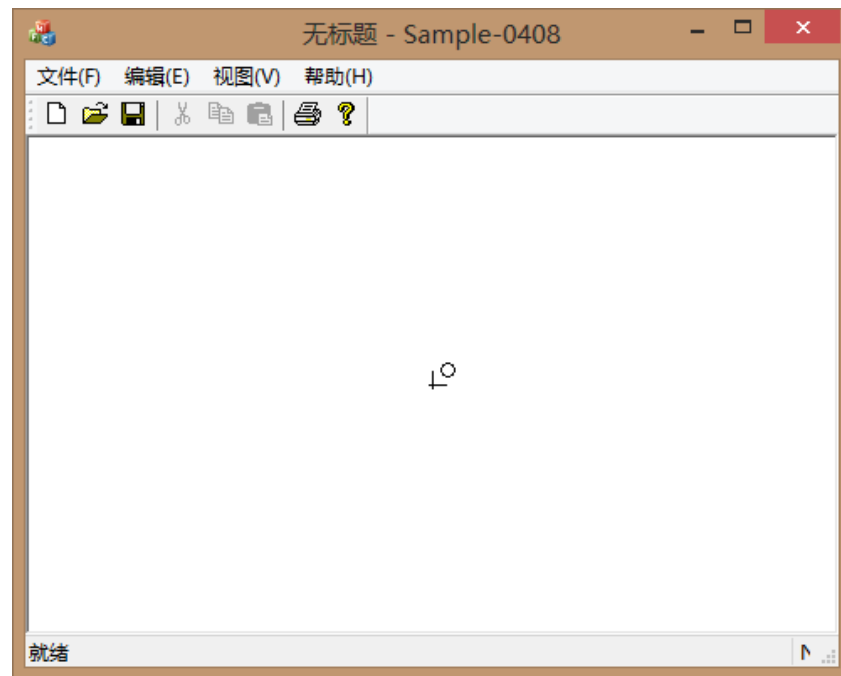
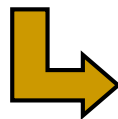
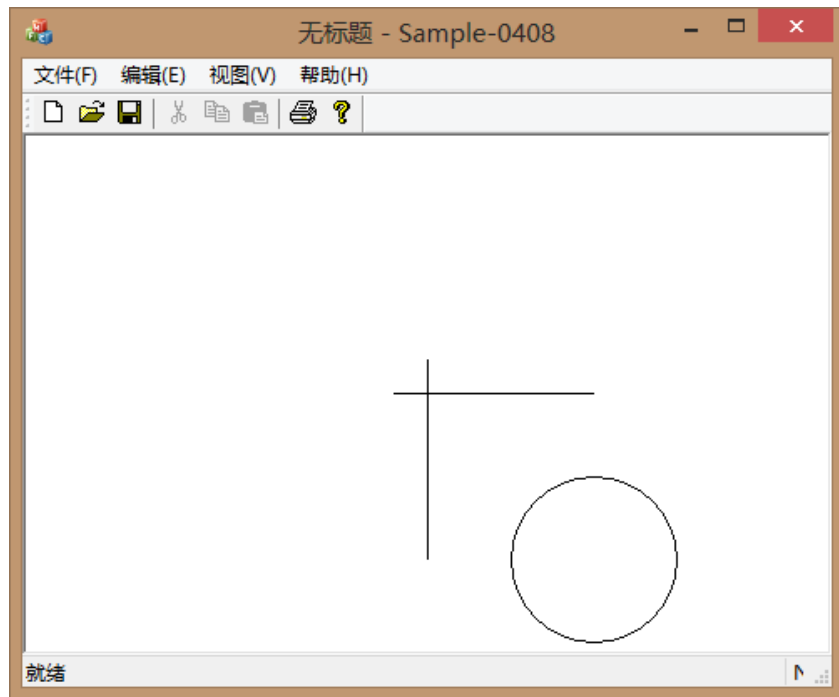
映射模式(3)

例4-8

■ MM_TEXT与MM_HIENGLISH的区别

```
CRect rect;  GetClientRect(&rect);  
pDC->SetMapMode(MM_HIENGLISH);  
pDC->SetViewportOrg(rect.right/2,  
rect.bottom/2);  
pDC->MoveTo(-20, 0);  pDC->LineTo(100, 0);  
pDC->MoveTo(0, -20);  pDC->LineTo(0, 100);  
CPoint pt(100, 100);  
pDC->Ellipse(pt.x-50, pt.y-50, pt.x+50,  
pt.y+50);
```

映射模式(4)



文本处理(1)

■ 文本输出函数

- ✓ `TextOutW`: 标准的文本输出
- ✓ `DrawTextW`: 矩形内文本输出
- ✓ `ExtTextOutW`: 扩展的文本输出
- ✓ `TabbedTextOutW`: 带制表符的文本输出

文本处理 (2)

■ 文本属性设置函数

- ✓ SetBkMode: 背景模式
- ✓ SetBkColor: 背景颜色
- ✓ SetTextColor: 文本颜色
- ✓ SetTextAlign: 对齐方式
- ✓ SetTextCharacterExtra: 字符间隔

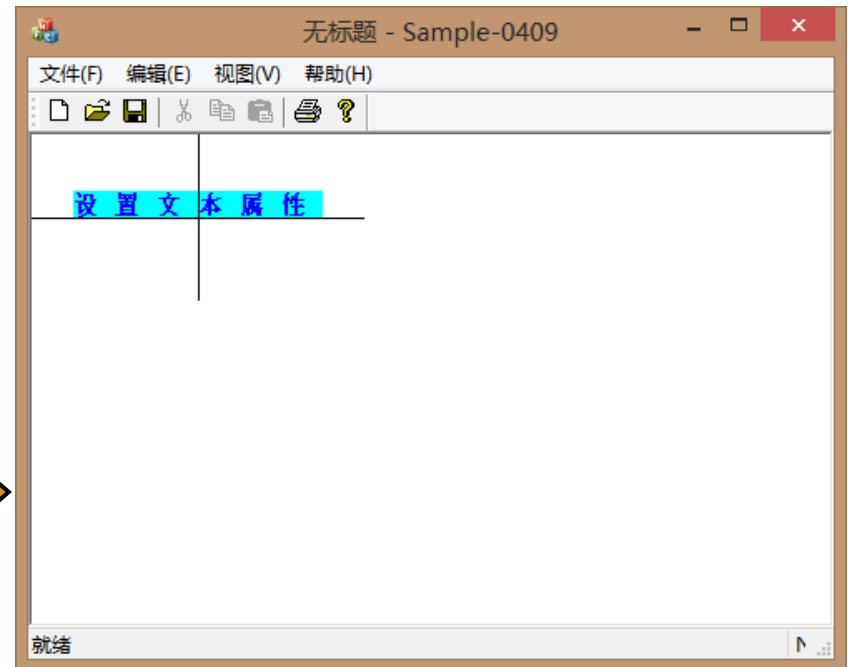
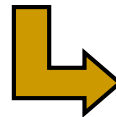
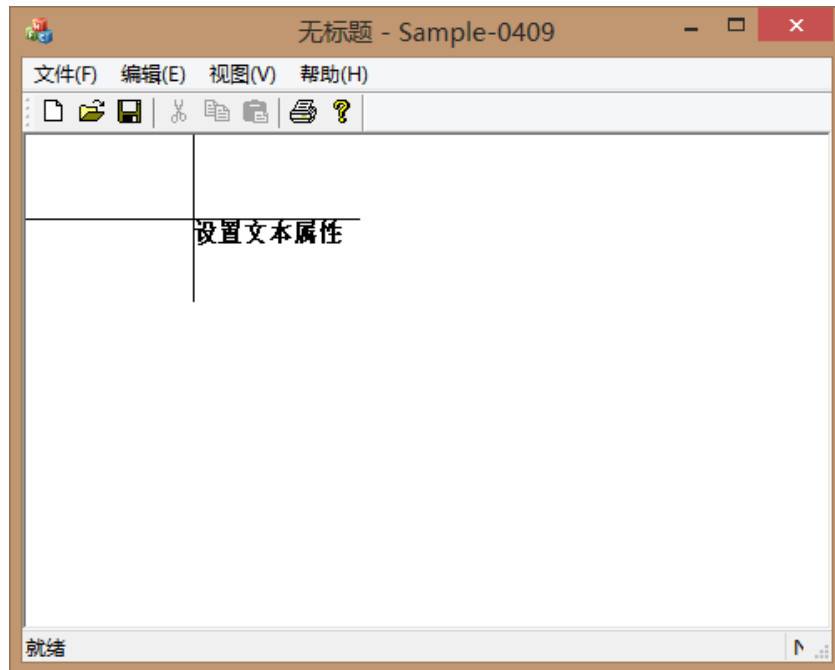
文本处理 (3)

例4-9

■ 文本属性设置的例子

```
pDC->SetBkMode(OPAQUE);  
pDC->SetBkColor(RGB(0, 255, 255));  
pDC->SetTextColor(RGB(0, 0, 255));  
pDC->SetTextAlign(TA_CENTER|TA_BOTTOM);  
pDC->SetTextCharacterExtra(10);  
pDC->TextOutW(100, 50, L"设置文本属性");  
pDC->MoveTo(0, 50); pDC->LineTo(200, 50);  
pDC->MoveTo(100, 0); pDC->LineTo(100, 100);
```

文本处理(4)



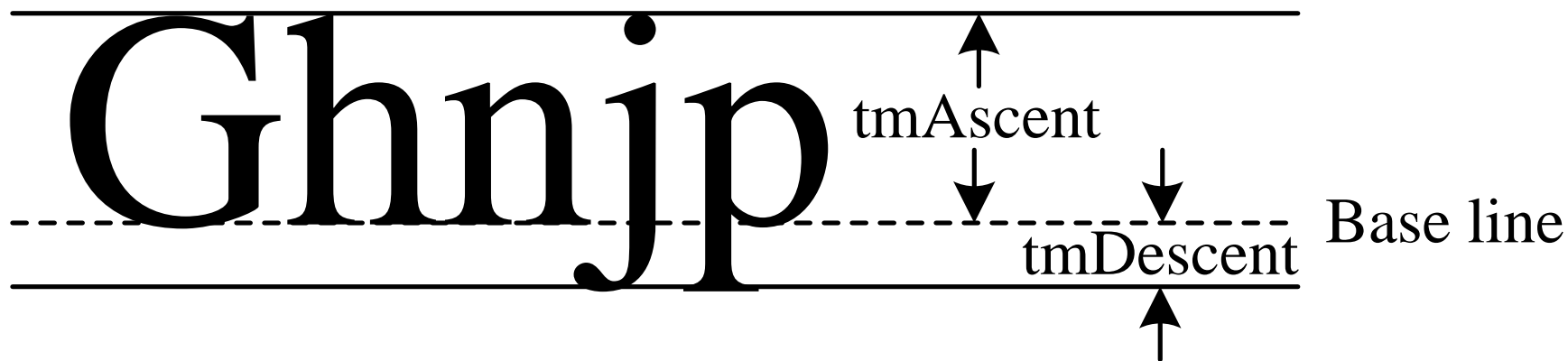
字符属性(1)

■ 字符属性：字符大小、行间距

```
typedef struct tagTEXTMETRIC
{
    LONG  tmHeight;           //字符高度
    LONG  tmAscent;           //基线以上高度
    LONG  tmDescent;          //基线以下高度
    LONG  tmExternalLeading;   //行间距
    .....
} TEXTMETRIC;
```

字符属性 (2)

- tmAscent与tmDescent的含义



字符属性 (3)

例4-10

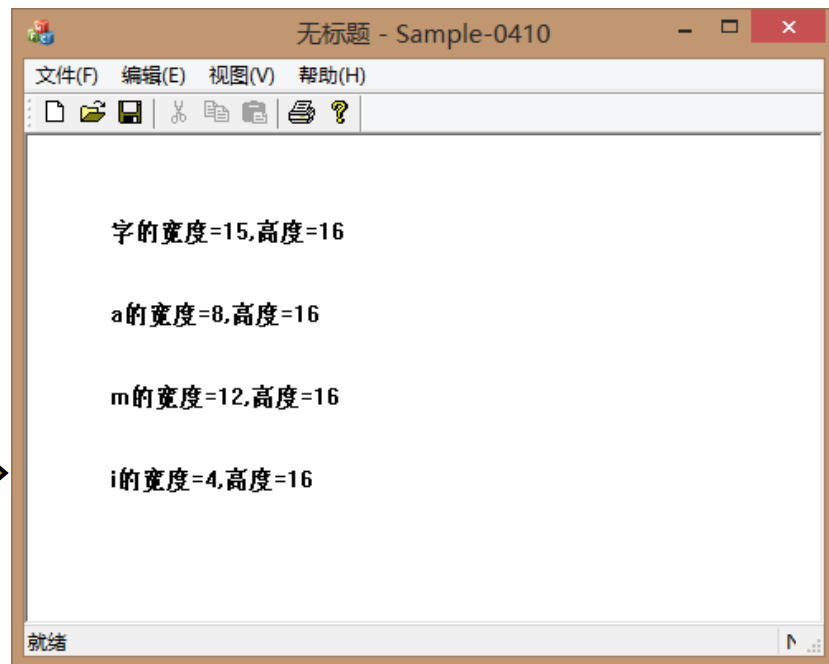
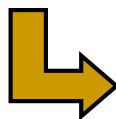
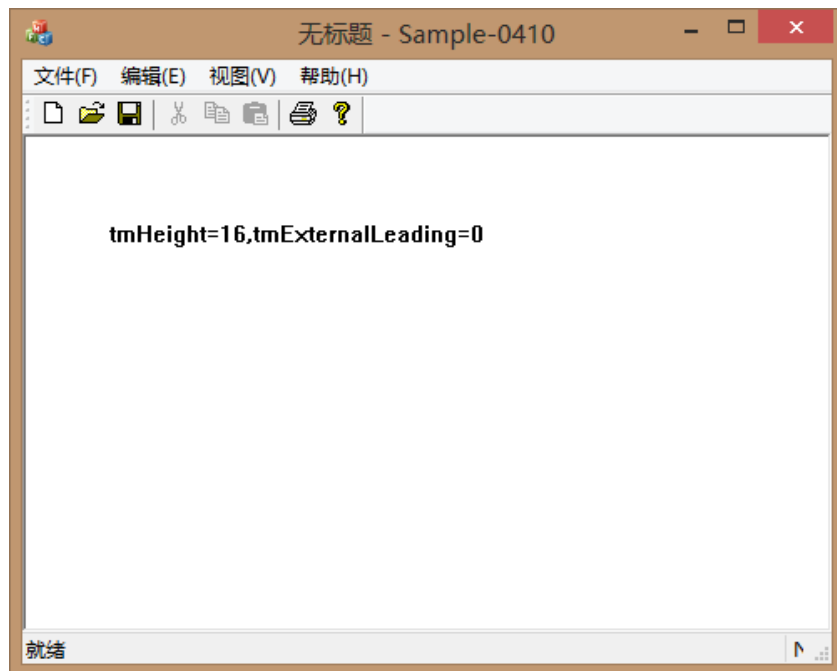
■ 字符属性的例子

```
TEXTMETRIC tm;  
pDC->GetTextMetrics(&tm); 将TEXTMETRIC选入  
CString str;  
str.Format(L"tmHeight=%d, tmExternalLeading  
=%d", tm.tmHeight, tm.tmExternalLeading);  
pDC->TextOutW(50, 50, str);
```

字符属性(4)

```
CString str; CSize sz; 单个字符的宽度、高度  
sz=pDC->GetTextExtent(L"字");  
str.Format(L"字的宽度=%d, 高度=%d", sz.cx, sz.cy);  
pDC->TextOutW(50, 50, str);  
sz=pDC->GetTextExtent(L"a");  
str.Format(L"a的宽度=%d, 高度=%d", sz.cx, sz.cy);  
pDC->TextOutW(50, 100, str);  
sz=pDC->GetTextExtent(L"m");  
str.Format(L"m的宽度=%d, 高度=%d", sz.cx, sz.cy);  
pDC->TextOutW(50, 150, str);  
sz=pDC->GetTextExtent(L"i");  
str.Format(L"i的宽度=%d, 高度=%d", sz.cx, sz.cy);  
pDC->TextOutW(50, 200, str);
```

字符属性 (5)



字体操作(1)

■ 库存字体的类型

字体类型	说明
SYSTEM_FONT	系统字体
SYSTEM_FIXED_FONT	固定宽度系统字体
ANSI_FIXED_FONT	ANSI固定宽度字体
ANSI_VAR_FONT	ANSI可变宽度字体
DEVICE_DEFAULT_FONT	设备缺省字体
OEM_FIXED_FONT	OEM固定宽度字体

字体操作 (2)

例4-11

■ 库存字体的例子

```
pDC->TextOutW(50, 50, L"DEFAULT字体");  
pDC->SelectStockObject(ANSI_FIXED_FONT);  
pDC->TextOutW(50, 100, L"ANSI_FIXED_FONT字  
体");  
pDC->SelectStockObject(SYSTEM_FONT);  
pDC->TextOutW(50, 150, L"SYSTEM_FONT字体");
```

字体操作 (3)

■ 字体类型

- ✓ 逻辑字体与物理字体

■ 两步构造方法

- ✓ 在LOGFONT结构中定义逻辑字体
- ✓ 调用CreateFontIndirect() 函数

■ 单步构造方法

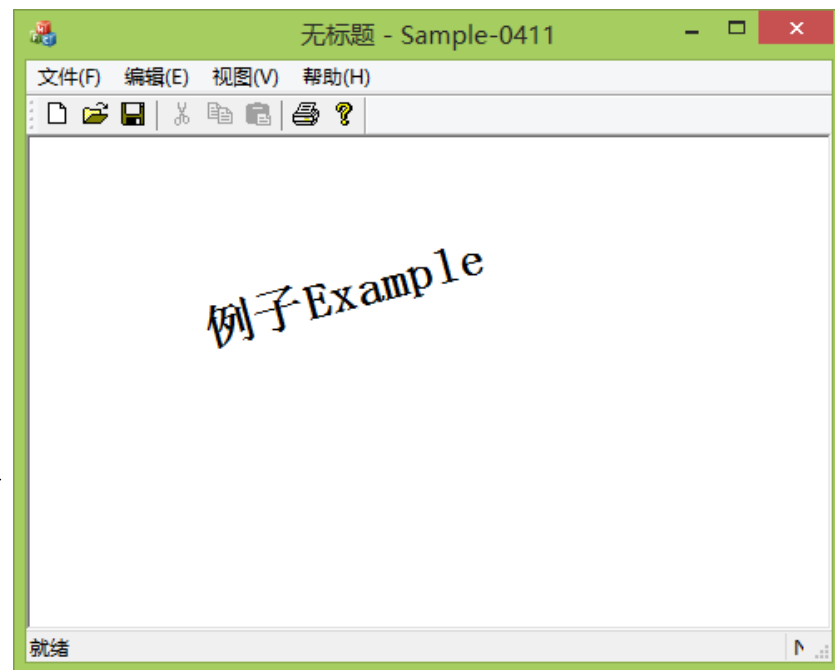
- ✓ 直接调用CreateFont() 函数

字体操作(4)

```
LOGFONT lf; LOGFONT是设计字体的一个结构
lf.lfHeight=30; //高度
lf.lfWidth=0; //宽度
lf.lfEscapement=150; //与水平线角度
lf.lfWeight=FW_BOLD; //粗体
lf.lfUnderline=false; //无下划线
lf.lfItalic=false; //非斜体
lf.lfStrikeOut=false; //无删除线
lf.lfCharSet=GB2312_CHARSET; //字符集
CFont newFont;
newFont.CreateFontIndirect(&lf);
pDC->SelectObject(newFont);
pDC->TextOutW(100, 100, L"例子Example");
```

把字体对象lf用在newFont上，下一行中选入设备环境

字体操作 (5)



位图操作(1)

BMP这种格式的特点是包含的图像信息较丰富，几乎不进行压缩，但由此导致了它与生俱来的缺点--占用磁盘空间过大。

- BMP是与硬件无关图像格式，采用位映射存储方式，除图像深度可选，不用其它压缩
- 图像深度包括：1位(单色)、4位(16色)、8位(256色)、24位(16M色)
- BMP文件结构
 - ✓ 文件头：文件类型、大小、起始位置等
 - ✓ 信息头：图像大小、压缩方法等
 - ✓ 颜色表与位图数据

位图操作 (2)

- 位图以位模式形成图像，`CBitmap`类定义位图，`LoadBitmap()`从资源装载位图
- CDC提供传输图形数据的函数
 - ✓ `PatBlt()`：用选定画刷填充矩形
 - ✓ `BitBlt()`：将图像输出到兼容设备环境
 - ✓ `StretchBlt()`：与`BitBlt()`类似，可改变图像大小

位图操作 (3)

■ 画刷填充模式

填充参数	说明
BLACKNESS	黑色填充目标区域
WHITENESS	白色填充目标区域
PATCOPY	画刷复制到目标区域
PATINVERT	画刷异或到目标区域
DSTINVERT	目标区域取反

位图操作(4)

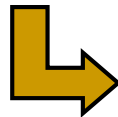
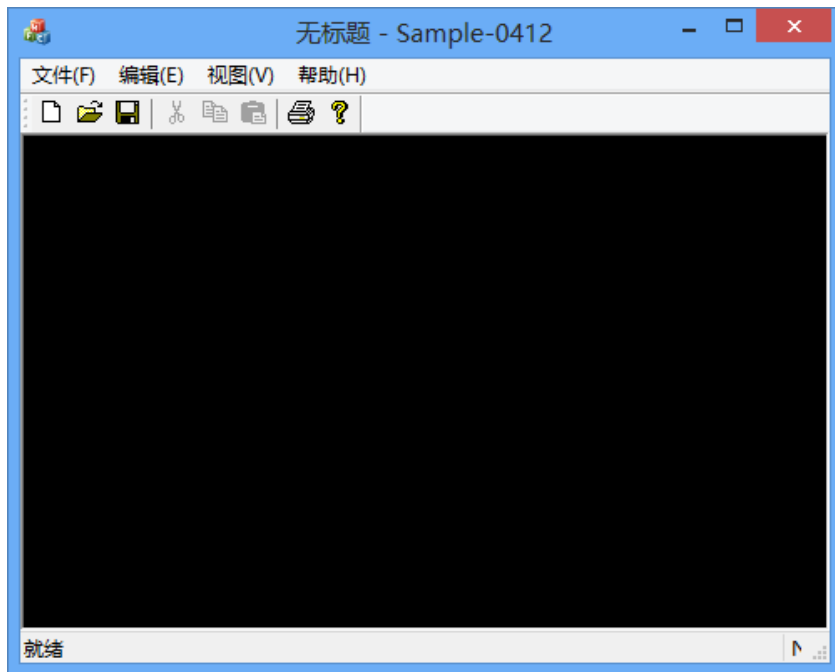
例4-12

■ PatBlt的例子

通过画刷复制到目标区域

```
CBitmap bm;  bm.LoadBitmap(IDB_MYBITMAP);  
CBrush newBrush;  
newBrush.CreatePatternBrush(&bm);  
pDC->SelectObject(newBrush);  
CRect rect;  GetClientRect(&rect);  
pDC->PatBlt(0, 0, rect.right, rect.bottom,  
PATCOPY);  
bm.DeleteObject();
```

位图操作 (5)



位图操作 (6)

■ BitBlt与StretchBlt的填充模式

填充参数	说明
SRCCOPY	来源区域直接填充目标区域
SRCAND	来源区域与目标区域“与”
SRCPAINT	来源区域与目标区域“或”
SRCINVERT	来源区域与目标区域“异或”

位图操作(7)

引入info的作用只是为了确定位图的大小 (获取info. bmWidth&info. bmHeight)

■ BitBlt的例子

通过CDC

```
CBitmap bm;  bm.LoadBitmap(IDB_MYBITMAP);  
CDC memDC;  memDC.CreateCompatibleDC(pDC);  
memDC.SelectObject(&bm);  
BITMAP info;  bm.GetBitmap(&info);  
pDC->BitBlt(0, 0, info.bmWidth,  
info.bmHeight, &memDC, 0, 0, SRCCOPY);  
bm.DeleteObject();
```

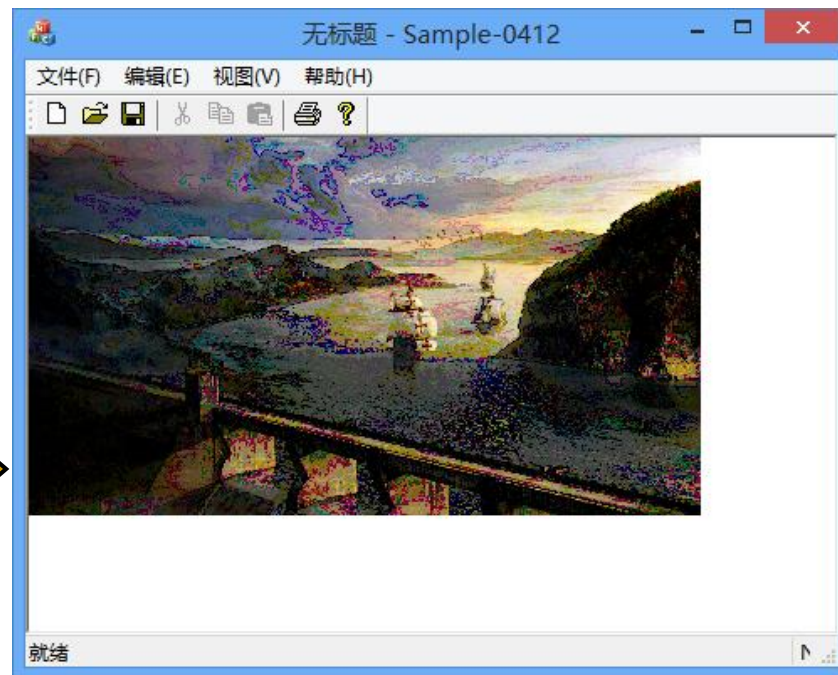
位图操作 (8)

StretchBlt() : 与BitBlt()类似，可改变图像大小

■ StretchBlt的例子

```
CBitmap bm;  bm.LoadBitmap(IDB_MYBITMAP);  
CDC memDC;  memDC.CreateCompatibleDC(pDC);  
memDC.SelectObject(&bm);  
BITMAP info;  bm.GetBitmap(&info);  
pDC->StretchBlt(0, 0, info.bmWidth/2,  
info.bmHeight/2, &memDC, 0, 0, info.bmWidth,  
info.bmHeight, SRCCOPY);  
bm.DeleteObject();
```

位图操作 (9)



图标操作(1)

- 图标(Icon)是一种特殊的位图，与位图的区别是固定大小
- CWinApp提供LoadStandardIcon()，加载系统预定义的图标
- CWinApp提供LoadIcon()，加载图形编辑器创建的图标

图标操作 (2)

■ 系统预定义的图标

预定义图标宏	说明
IDI_APPLICATION	默认图标
IDI_ASTERISK	信息图标
IDI_EXCLAMATION	惊叹号图标
IDI_HAND	严重警告图标
IDI_QUESTION	问号图标

图标操作 (3)

上一页中的预设的图标

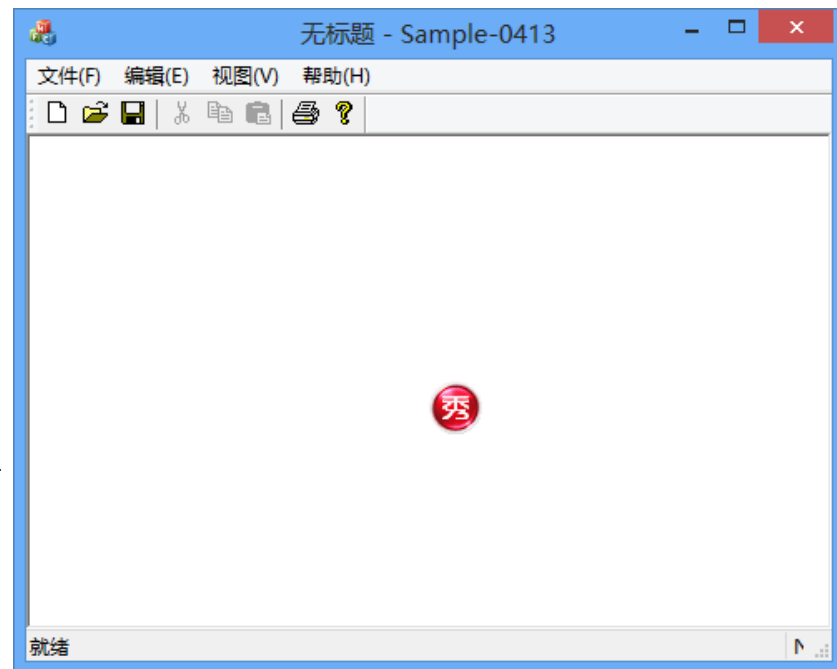
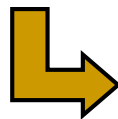
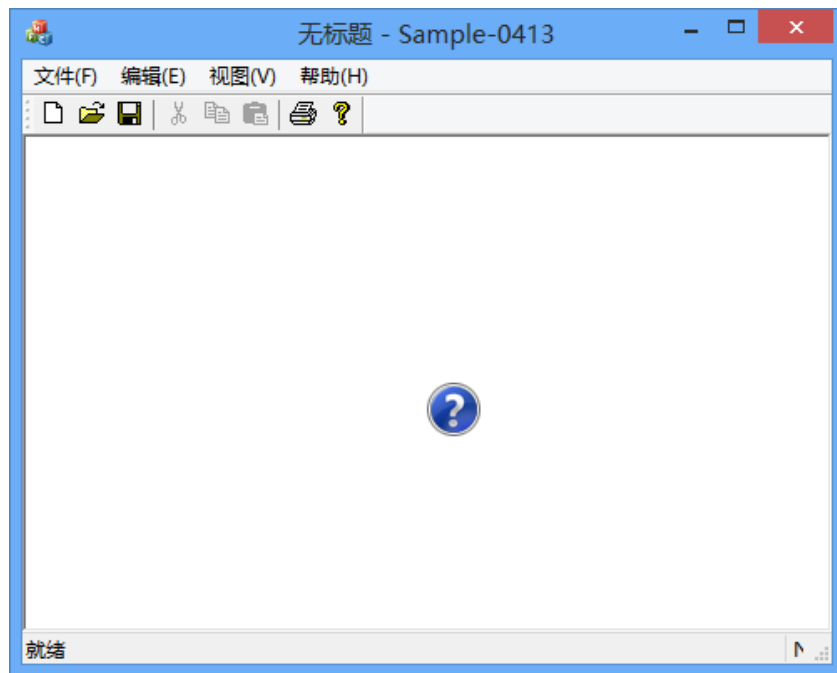
例4-13

自定义的添加的图标

■ 显示图标的例子

```
HICON icon;  
icon=AfxGetApp()->LoadStandardIcon  
(IDI_QUESTION);  
icon=AfxGetApp()->LoadIconW(IDI_MYICON);  
CRect rect;  
GetClientRect(&rect);  
pDC->DrawIcon(rect.right/2, rect.bottom/2,  
icon);
```

图标操作(4)



光标操作(1)

- 光标(Cursor)用于显示鼠标操作时，鼠标所在位置与显示形状
- CWinApp提供LoadStandardCursor()，加载系统预定义的光标
- CWinApp提供LoadCursor()，加载图形编辑器创建的光标
- SetCursor()用于设置光标形状

光标操作(2)

■ 系统预定义的光标

光标类型	说明	光标类型	说明
IDC_ARROW	箭头	IDC_UPARROW	垂直箭头
IDC_CROSS	十字光标	IDC_SIZEALL	四向箭头
IDC_WAIT	沙漏光标	IDC_SIZENWSE	左上右下双箭头
IDC_IBEAM	输入光标	IDC_SIZENESW	右上左下双箭头
IDC_SIZE	装入方框	IDC_SIZEWE	水平双箭头
IDC_ICON	空肖像	IDC_SIZENS	垂直双箭头

光标操作 (3)

例4-14

- 在CTestView::OnLButtonDown() 中

```
HCURSOR cursor;  
cursor=AfxGetApp()->LoadStandardCursorW  
(IDC_CROSS);  
cursor=AfxGetApp()->LoadCursorW  
(IDC_MYCURSOR);  
SetCapture(); SetCursor(cursor);  
CRect rect; GetClientRect(&rect);  
ClientToScreen(&rect); ClipCursor(&rect);
```

光标操作 (4)

- 在CTestView::OnLButtonUp() 中

```
ReleaseCapture();  
ClipCursor(NULL);
```

- 在CMainFrame::PreCreateWindow() 中

```
cs. cx=500;  
cs. cy=400;
```

鼠标画线的例子(1)

例4-15

- 在CTestView类定义中

```
private:  
    int m_draw;  
    HCURSOR m_cursor;  
    CPoint m_old, m_origin;
```

- 在CTestView构造函数中

```
m_draw=0;  
m_cursor=AfxGetApp()->LoadStandardCursor  
(IDC_CROSS);  
m_old=m_origin=CPoint(0,0);
```

鼠标画线的例子(2)

- 在CTestView::OnLButtonDown() 中

```
m_old=m_origin=point;  
m_draw=1;  
SetCapture(); SetCursor(m_cursor);  
CRect rect; GetClientRect(&rect);  
ClientToScreen(&rect); ClipCursor(&rect);
```

- 在CTestView::OnLButtonUp() 中

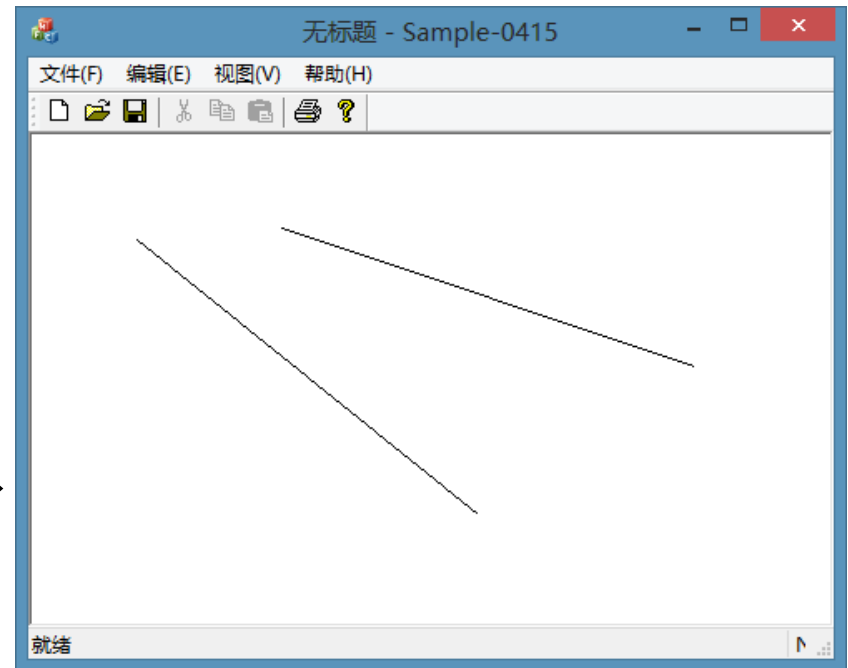
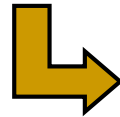
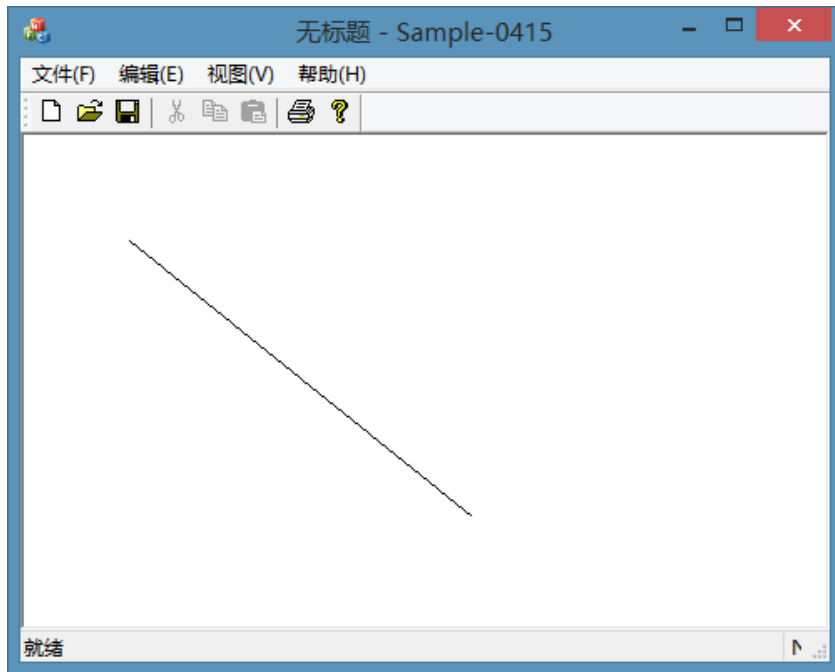
```
m_draw=0;  
ReleaseCapture(); ClipCursor(NULL);
```


鼠标画线的例子(3)

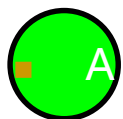
- 在CTestView::OnMouseMove() 中

```
CClientDC dc(this);  
dc.SetROP2(R2_NOT);  
if(m_draw==1)  
{  
    dc.MoveTo(m_origin);    dc.LineTo(m_old);  
    dc.MoveTo(m_origin);    dc.LineTo(point);  
    m_old=point;  
}
```

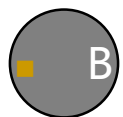
鼠标画线的例子(4)



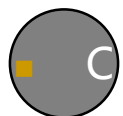
哪个函数可获得客户区矩形区域?



GetClientRect



GetDC



GetDocument



InvalidateRect

背景与贴图的例子(1)

例4-16

■ 在CTestView类定义中

```
private:  
    CBitmap m_back;  
    CBitmap m_bird0;  
    CBitmap m_bird1;
```

■ 在CTestView构造函数中

```
m_back.LoadBitmap(IDB_BACK);  
m_bird0.LoadBitmap(IDB_BIRD0);  
m_bird1.LoadBitmap(IDB_BIRD1);
```

背景与贴图的例子(2)

- 在CMainFrame::PreCreateWindow() 中

```
cs.cx=534;   cs.cy=432;  
cs.style&=~WS_MAXIMIZEBOX;
```

- 在CTestView::OnDraw() 中

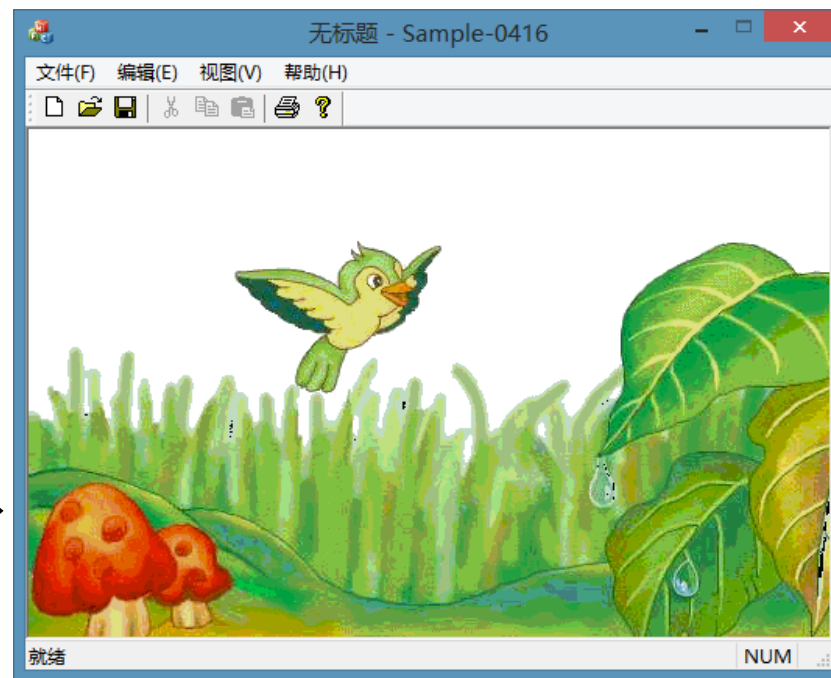
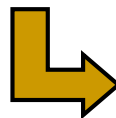
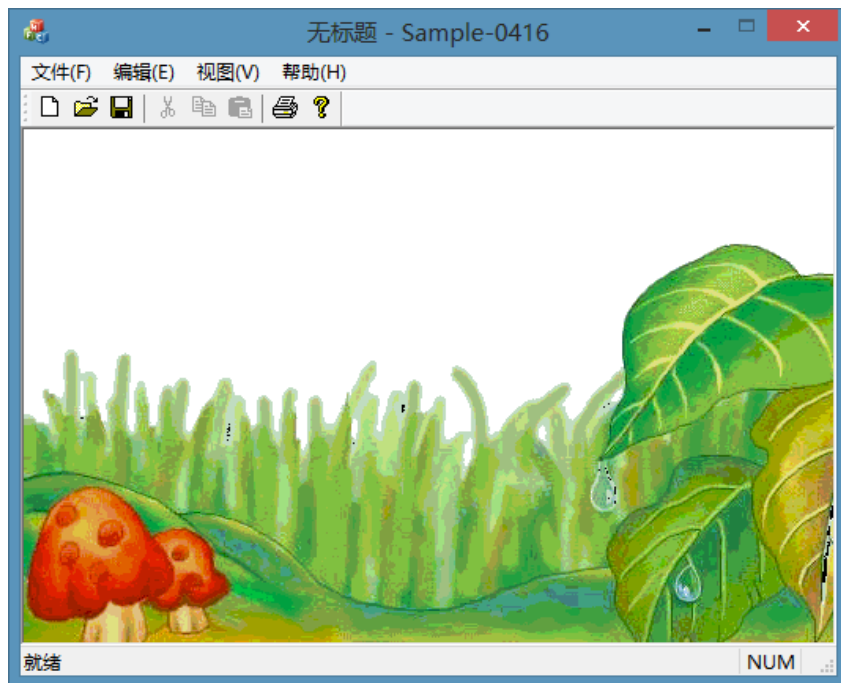
```
CDC memDC;  
memDC.CreateCompatibleDC(pDC);  
memDC.SelectObject(&m_back);  
pDC->BitBlt(0, 0, 534, 432, &memDC, 0, 0,  
SRCCOPY);
```

背景与贴图的例子(3)

- 在CTestView::OnLButtonDown() 中

```
CDC *pDC=GetDC();  
CDC memDC;  memDC.CreateCompatibleDC(pDC);  
memDC.SelectObject(&m_bird0);  
pDC->BitBlt(point.x, point.y, 137, 99, &memDC,  
0, 0, SRCAND);  
memDC.SelectObject(&m_bird1);  
pDC->BitBlt(point.x, point.y, 137, 99, &memDC,  
0, 0, SRCPAINT);
```

背景与贴图的例子(4)



扇面效果的例子(1)

例4-17

- 在CTestView类中

```
private:  
    BOOL m_draw;  
    CPoint m_old, m_origin;
```

- 在CTestView构造函数中

```
m_draw=false;
```

- 在CTestView::OnLButtonDown中

```
m_draw=true;  
m_old=m_origin=point;
```


扇面效果的例子(2)

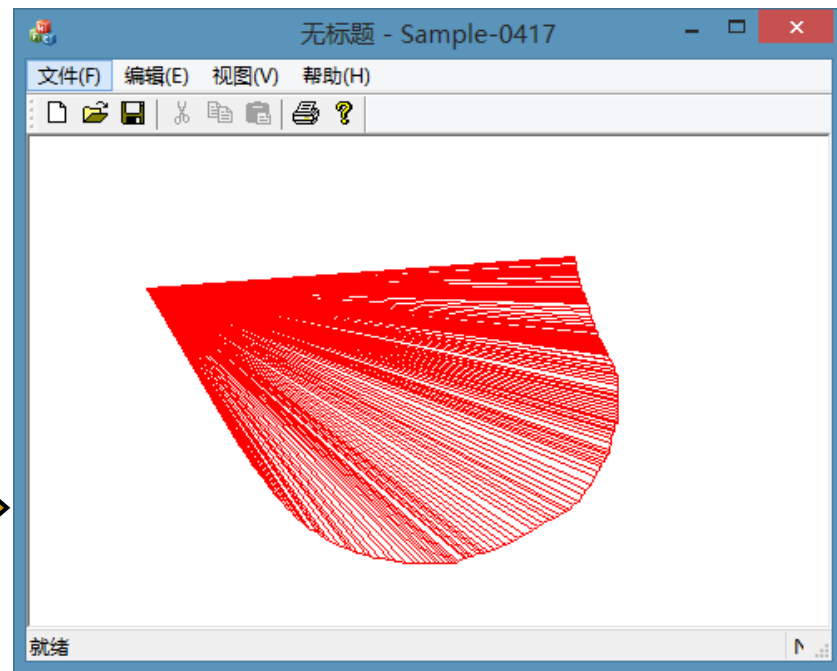
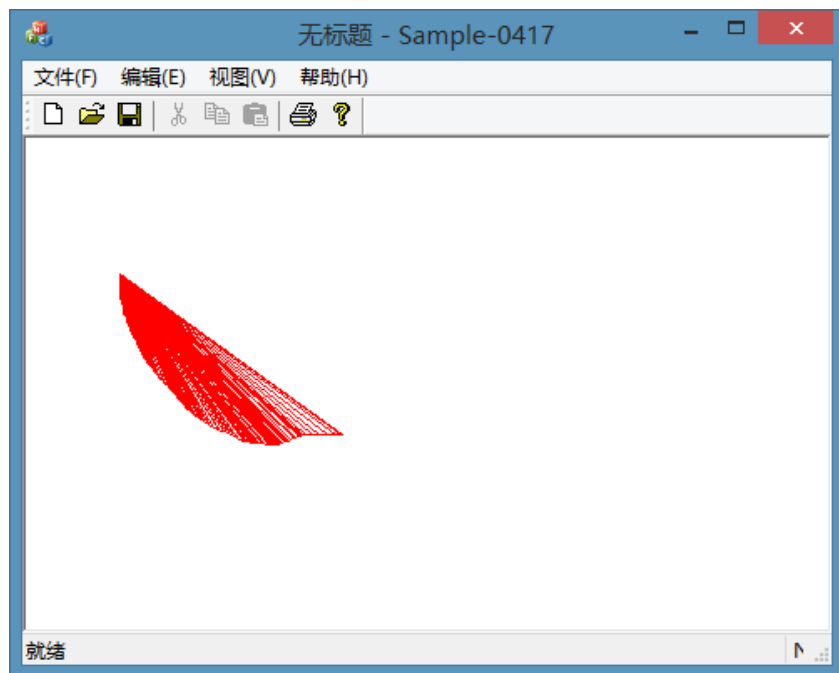
- 在CTestView::OnMouseMove() 中

```
CClientDC dc(this);  
CPen newPen(PS_SOLID, 1, RGB(255, 0, 0));  
dc.SelectObject(newPen);  
if(m_draw==true)  
{ dc.MoveTo(m_origin); dc.LineTo(point);  
  dc.LineTo(m_old); m_old=point; }
```

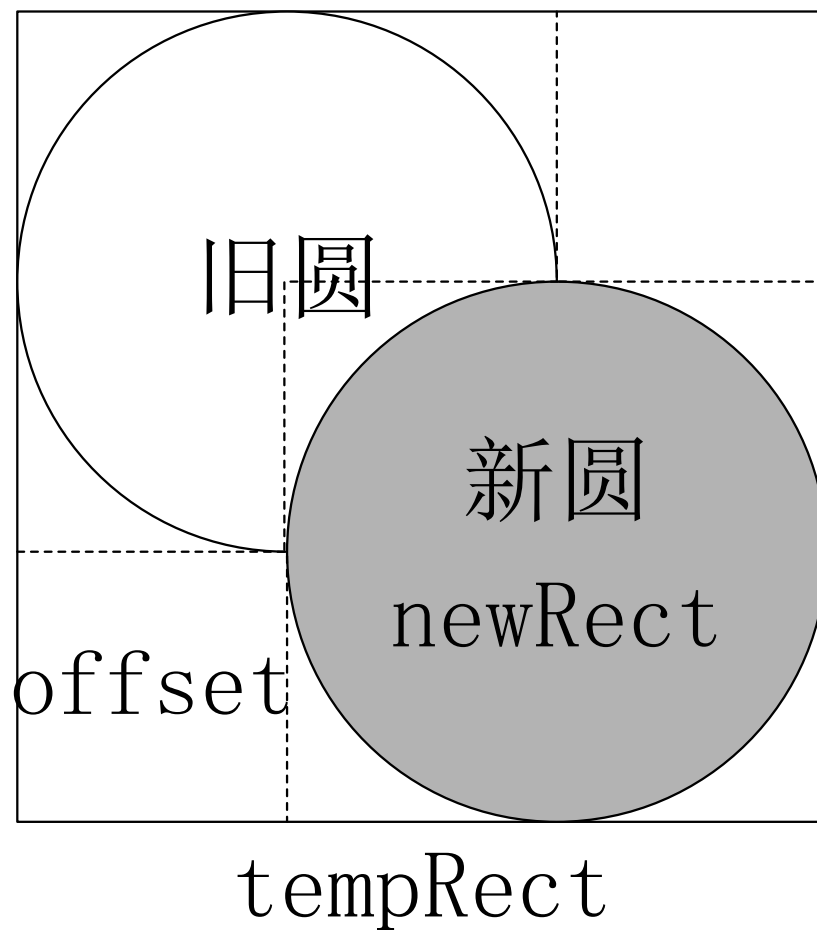
- 在CTestView::OnLButtonUp() 中

```
m_draw=false;
```

扇面效果的例子(3)



鼠标拖动圆的例子(1)



鼠标拖动圆的例子(2)

例4-18

- 在CTestView类定义中

```
private:  
    CRect m_ellipse;  
    CPoint m_pos;  
    bool m_capture;
```

- 在CTestView构造函数中

```
m_ellipse=CRect(0, 0, 60, 60);  
m_capture=false;
```

鼠标拖动圆的例子(3)

- 在CTestView::OnDraw() 中

画刷，填充矩形为浅灰色

```
pDC->SelectStockObject(LTGRAY_BRUSH);  
pDC->Ellipse(m_ellipse);
```

- 在CTestView::OnLButtonUp() 中

```
m_capture=false;  
ReleaseCapture();
```

鼠标拖动圆的例子(4)

- 在CTestView::OnLButtonDown() 中

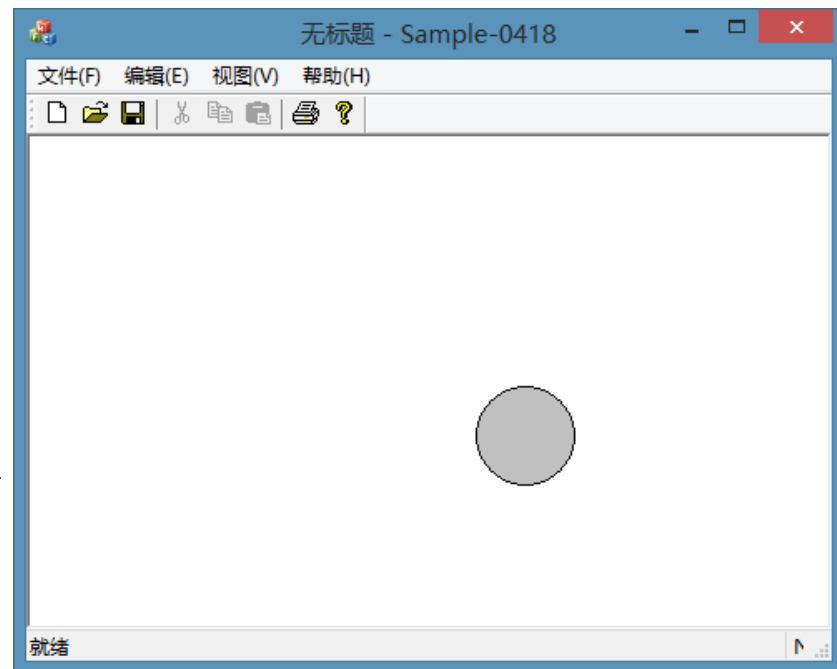
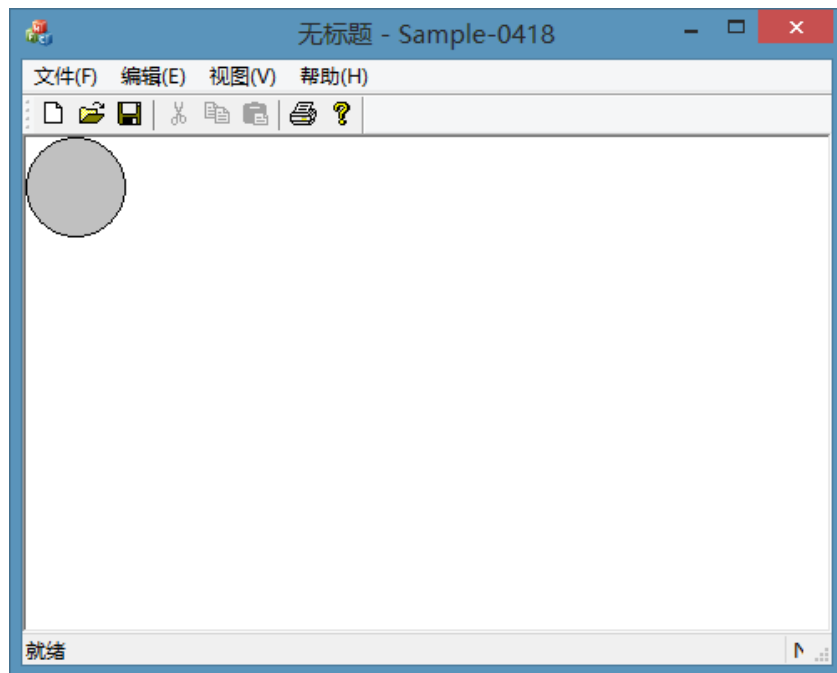
```
CRgn rgn; 限制区域
rgn.CreateEllipticRgnIndirect(m_ellipse);
if(rgn.PtInRegion(point)) 如果point在该区域内
{ m_capture=true;
  m_pos=point;
  SetCapture();
  SetCursor(AfxGetApp()->LoadCursor
(IDC_CROSS)); }
```

鼠标拖动圆的例子(5)

- 在CTestView::OnMouseMove()中

```
CPoint offset;  offset=point-m_pos;
CRect clientRect,newRect,tempRect;
GetClientRect(&clientRect);
if(m_capture==true)
{ if(clientRect.PtInRect(point))
  { newRect=m_ellipse+offset;
    tempRect.UnionRect(m_ellipse,newRect);
    InvalidateRect(tempRect,true);
    m_pos=point; m_ellipse=newRect; } }
```

鼠标拖动圆的例子(6)



第4次作业

- 设计一个单文档边框窗口程序，要求具有以下几个功能：
 - ✓ 弹出快捷菜单(Line、Rectangle)
 - ✓ 点击菜单项输出相应图形
 - ✓ 通过工具栏按钮保存与打开图形
 - ✓ 按“Ctrl+V”键，一个位图(自行设计)沿图形边框移动



谢谢大家

