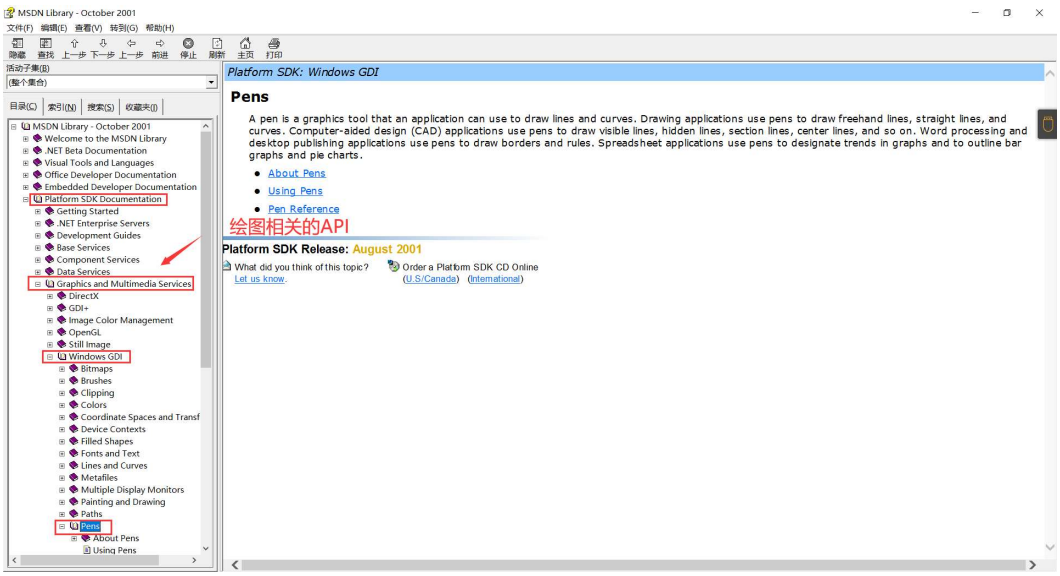
2020/07/07_MFC_第7课_CAD03_画笔画刷的使用

笔记本: MFC
创建时间: 2020/7/7 星期二 15:29
作者: ileemi
标签: 画笔画刷的使用

画笔



修饰画笔(cosmetic) -- 画位图
几何笔(geometric) -- 画矢量图

矢量图和位图的区别:
位图放大后会失真，矢量图放大不会失真。
MFC 提供了画笔的封装 -- CPen

主要API
CreatePen -- 创建具有指定样式、宽度和颜色的逻辑钢笔

为画笔，画刷添加属性时，要注意，绘制的图形是在内存DC中绘制的，更改画笔，画刷的相关属性需要告诉内存DC，使用 SelectObject API。
代码示例:

```cpp
#if 1
/*
测试画笔的使用
*/
CPen pen;
pen.CreatePen(PS_DASHDOTDOT, 1, RGB(255, 80, 255));
//LOGPEN logpen;
//logpen.lopnStyle = PS_DASHDOTDOT;
//pen.CreatePenIndirect(&logpen);
dcMem.SelectObject(&pen);  // 在内存中绘制，需要在内存中选入

/*
测试画刷的使用
*/
CBrush brush;
//dcMem.SelectObject(::GetStockObject(LTGRAY_BRUSH));

//brush.CreateSolidBrush(RGB(0, 0, 255));
brush.CreateHatchBrush(HS_FDIAGONAL, RGB(120, 120, 80));

//CBitmap bmp;
//bmp.LoadBitmap(BMP_SUNRISE);
//brush.CreatePatternBrush(&bmp);

//LOGBRUSH logbrush;
//logbrush.lbStyle = BS_NULL;
//brush.CreateBrushIndirect(&logbrush);

dcMem.SelectObject(&brush);
#endif // 测试画笔和画刷的使用
```

注意：画笔的一些风格的使用，受其宽度的影响，一般当宽度为1的时候才有效。

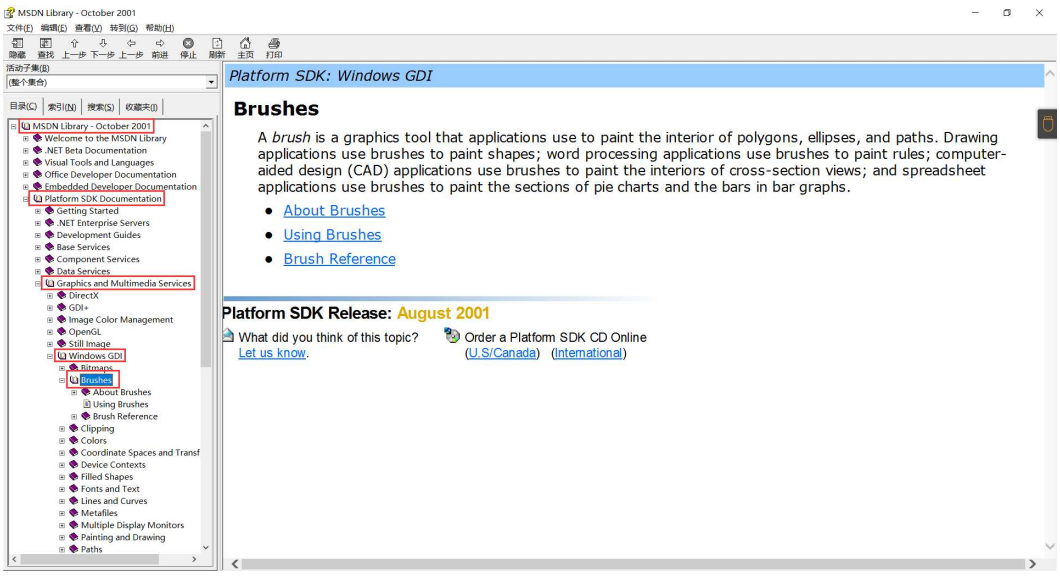
使用示例：

```cpp
/*
测试画笔的使用
*/
CPen pen;
pen.CreatePen(PS_DASHDOTDOT, 1, RGB(255, 0, 0));
dcMem.SelectObject(&pen);          // 在内存中绘制，需要在内存中选入

/*
fnPenStyle 可选值:
PS_SOLID            = 0;    {实线}
PS_DASH             = 1;    {段线; 要求笔宽<=1}
PS_DOT              = 2;    {点线; 要求笔宽<=1}
PS_DASHDOT          = 3;    {线、点; 要求笔宽<=1}
PS_DASHDOTDOT       = 4;    {线、点、点; 要求笔宽<=1}
PS_NULL             = 5;    {不可见}
PS_INSIDEFRAME      = 6;    {实线; 但笔宽是向里扩展}
*/
```

画刷

填充封闭图形内的区域，可填充颜色图案等



MFC进行了封装 -- **CBrush**

有两种类型的画刷：

- 逻辑画刷（**logical**）-- 逻辑画笔是应用程序用于绘制形状的理想位图的描述
- 物理画刷（**physical**）-- 物理画笔是设备驱动程序根据应用程序的逻辑画笔定义创建的实际位图

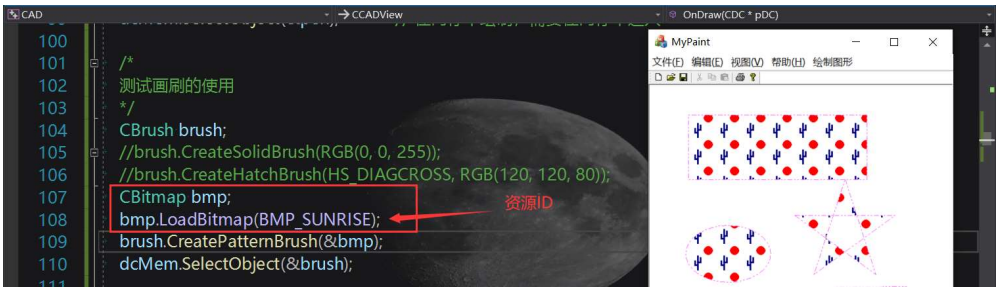
常用API：

- **CreateSolidBrush** -- 用指定的颜色填充封闭图形内的区域，边框依然是画笔的响应属性
- **CreateHatchBrush** -- 创建带有阴影模式和颜色的画笔 Hatch--影线刷

```
/*
测试画刷的使用
*/
CBrush brush;
//brush.CreateSolidBrush(RGB(0, 0, 255));
brush.CreateHatchBrush(HS_DIAGCROSS, RGB(120, 120, 80));
/*
Hatch Styles
#define HS_HORIZONTAL       0       -----
#define HS_VERTICAL         1       |||||
#define HS_FDIAGONAL        2       \\\\\
#define HS_BDIAGONAL        3       /////
#define HS_CROSS            4       +++++
#define HS_DIAGCROSS        5       xxxxx
#define HS_API_MAX          12
*/
dcMem.SelectObject(&brush);
```

- **CreatePatternBrush** -- 用位图模式创建画笔，需要添加位图，该API的参数为CBitmap*
  函数定义：BOOL CreatePatternBrush(CBitmap* pBitmap);
  通过从应用程序的可执行文件加载命名位图资源并将位图附加到对象来初始化对象，需要使用 CBitmap 类的成员函数 **LoadBitmap** 加载位图。
  成员函数定义：
  BOOL LoadBitmap(UINT nIDResource);
  nIDResource -- 资源的ID

使用示例：



- **CreateBrushIndirect** -- 创建具有指定样式、颜色和模式的逻辑画笔

参数：指向包含画笔信息的LOGBRUSH结构的指针

CreateBrushIndirect

The **CreateBrushIndirect** function creates a logical brush that has the specified style, color, and pattern.

```
HBRUSH CreateBrushIndirect(
  CONST LOGBRUSH *lplb   // brush information
);
```

Parameters

lplb
[in] Pointer to a **LOGBRUSH** structure that contains information about the brush.

Return Values

If the function succeeds, the return value identifies a logical brush.

If the function fails, the return value is NULL.

Windows NT/2000/XP: To get extended error information, call **GetLastError**.

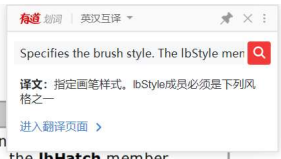
LOGBRUSH

The **LOGBRUSH** structure defines the style, color, and pattern of a physical brush. It is used by the **CreateBrushIndirect** and **ExtCreatePen** functions.

```
typedef struct tagLOGBRUSH {
  UINT     lbStyle;
  COLORREF lbColor;
  LONG     lbHatch;
} LOGBRUSH, *PLOGBRUSH;
```

Members

lbStyle
Specifies the brush style. The **lbStyle** member must be one of the following styles.



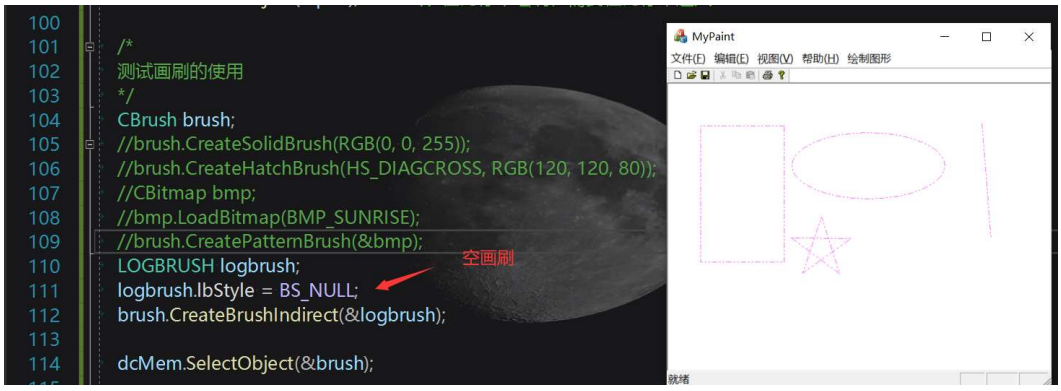
Members

lbStyle
Specifies the brush style. The **lbStyle** member must be one of the following styles.

Value	Meaning
BS_DIBPATTERN	A pattern brush defined by a device-indepen specification. If **lbStyle** is BS_DIBPATTERN, the **lbHatch** member contains a handle to a packed DIB. For more information, see discussion in **lbHatch**. **Windows 95:** Creating brushes from bitmaps or DIBs larger than 8 by 8 pixels is not supported. If a larger bitmap is specified, only a portion of the bitmap is used.
BS_DIBPATTERN8X8	Same as BS_DIBPATTERN.
BS_DIBPATTERNPT	A pattern brush defined by a device-independent bitmap (DIB) specification. If **lbStyle** is BS_DIBPATTERNPT, the **lbHatch** member contains a pointer to a packed DIB. For more information, see discussion in **lbHatch**.
BS_HATCHED	Hatched brush.
BS_HOLLOW	Hollow brush.
BS_NULL	Same as BS_HOLLOW.
BS_PATTERN	Pattern brush defined by a memory bitmap.
BS_PATTERN8X8	Same as BS_PATTERN.
BS_SOLID	Solid brush.

空画刷

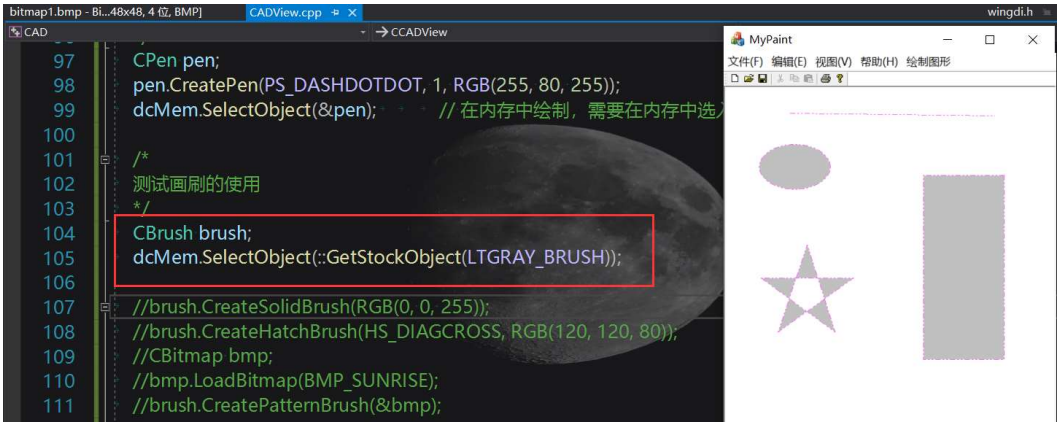
使用示例：



获取系统预定义好的画笔和画刷

使用API函数 -- GetStockObject
说明：此函数检索预定义的钢笔、画笔或字体的句柄。

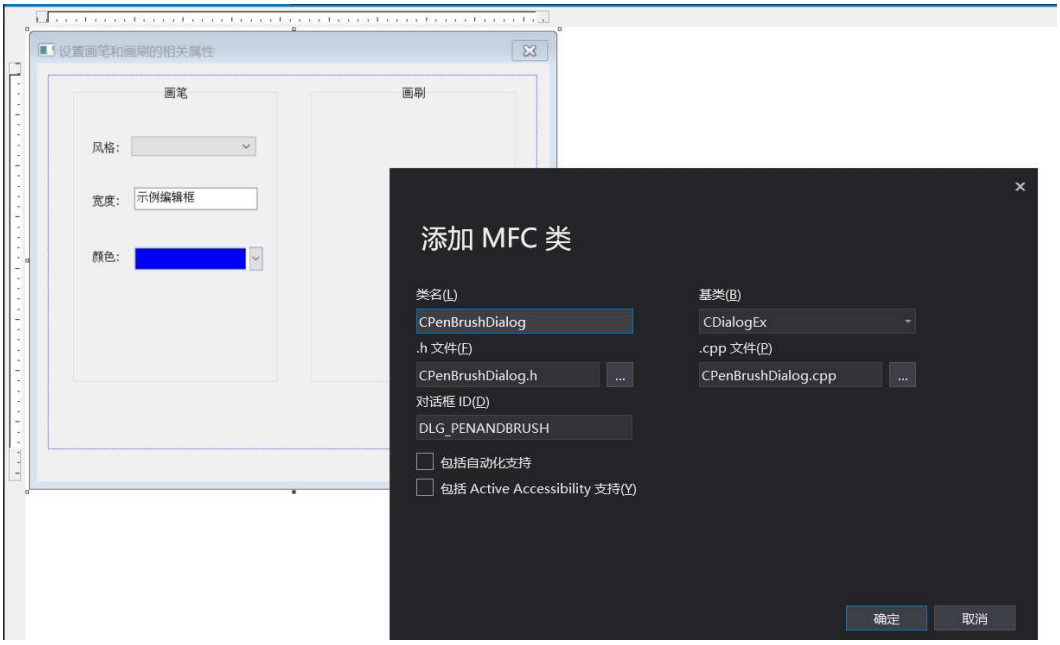
代码示例：



在CAD中设置画笔画刷

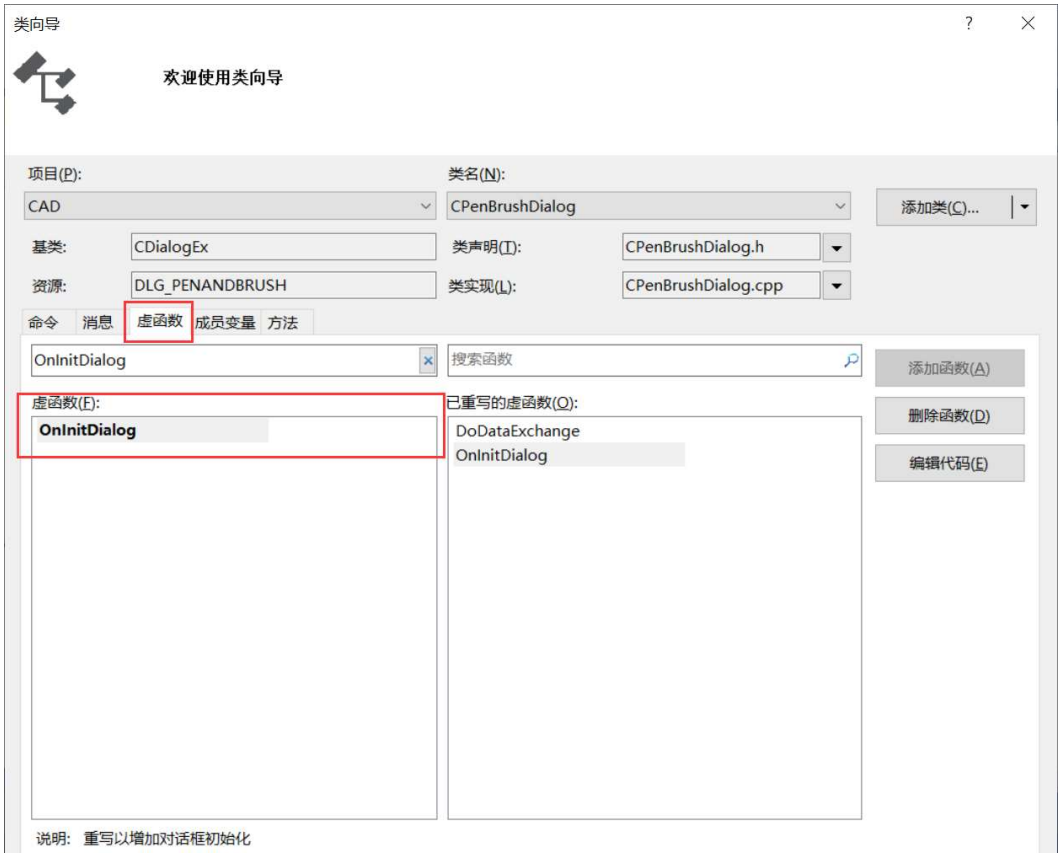
添加一个对应的画笔画刷对话框（在资源中进行添加即可）。
在MFC中想要操作对话框需要有一个对应的类，为对话框添加对应的类，只需要鼠标右键点击对话框，选择 -- **添加类** --> 输入类名， 即可。

示例：



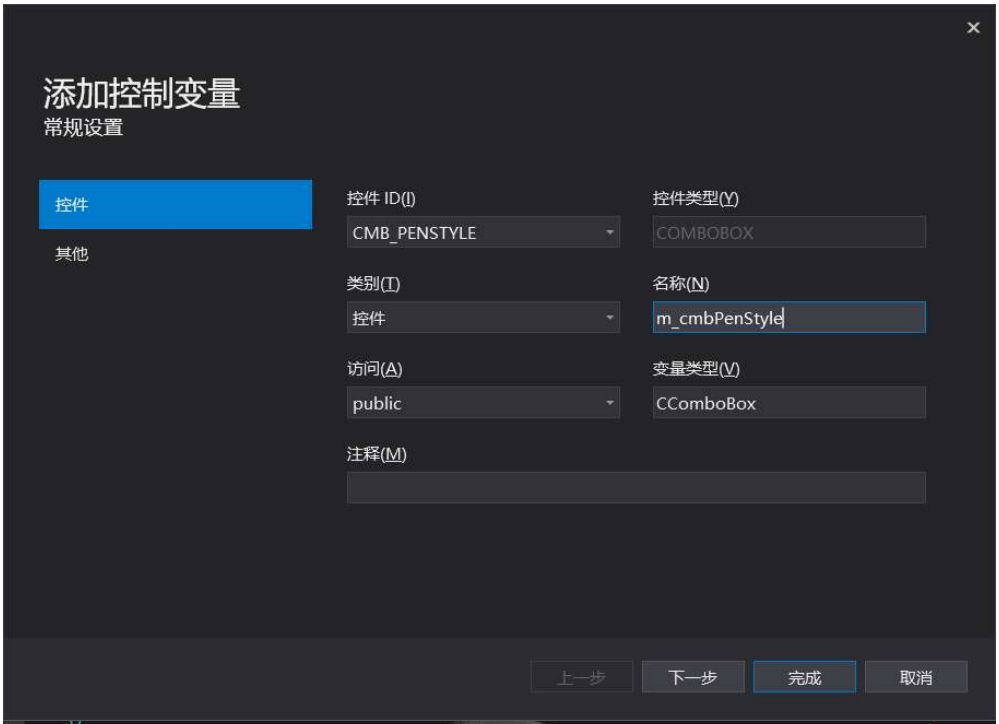
弹出一个对话框需要使用API -- DoModal
窗口类 xxx;
xxx.DoModal();

对话框设置初始值，需要在对话框类中添加虚函数 **OnInitDialog** 。
示例：



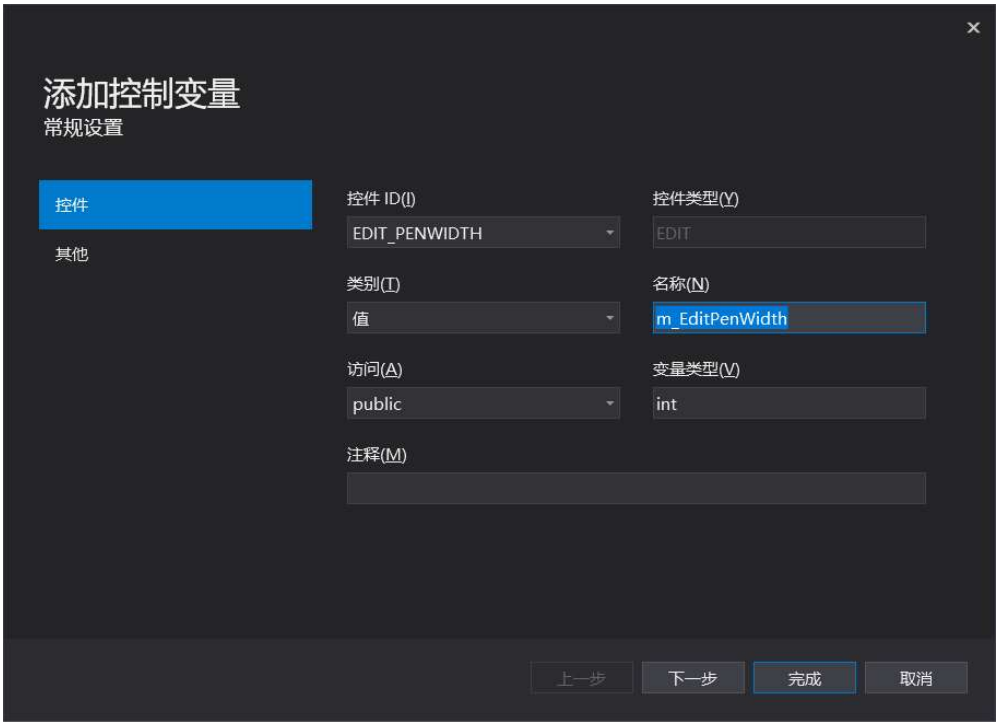
为对话框的控件绑定(添加)对应的变量

- 对对话框内的控件进行初始化时，使用DDX，将其绑定到对应的对应的类型上，鼠标右键添加变量，如下图所示：

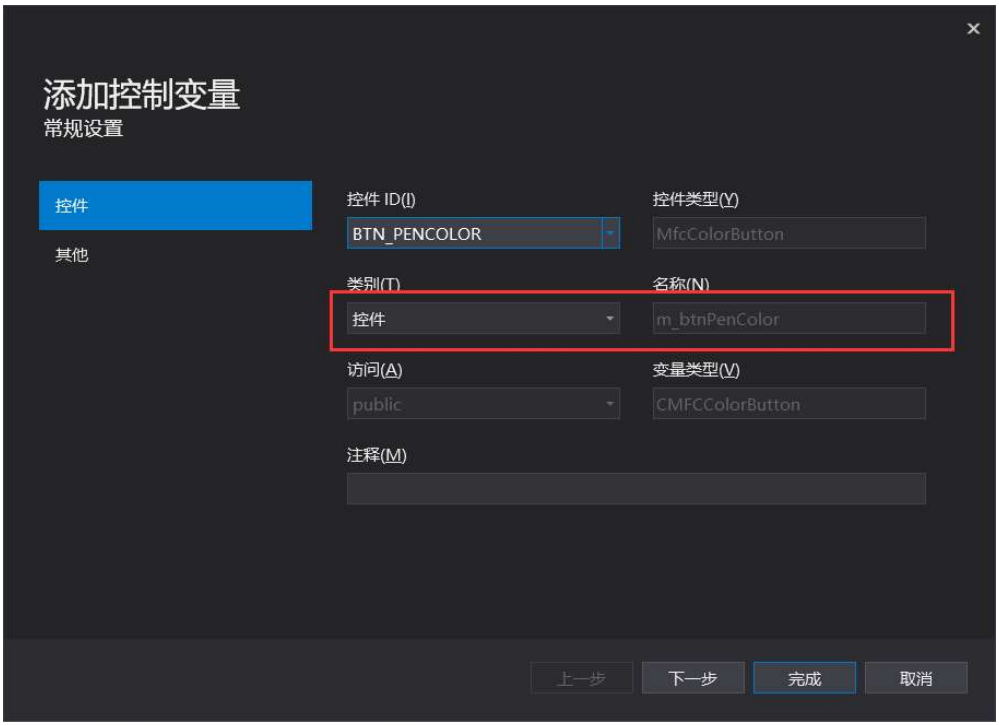



```
38  BOOL CPenBrushDialog::OnInitDialog()
39  {
40      CDialogEx::OnInitDialog();
41
42      // TODO:  在此添加额外的初始化
43      // 根据参数宏的作用为画笔的风格设定初始值
44      m_cmbPenStyle.AddString("_______");
45      m_cmbPenStyle.AddString("--------");
46      m_cmbPenStyle.AddString(".........");
47      m_cmbPenStyle.AddString("_._._._.");
48      m_cmbPenStyle.AddString("_.._.._._");
49      m_cmbPenStyle.AddString("         ");
50
51      return TRUE;  // return TRUE unless you set the focus to a control
52                    // 异常: OCX 属性页应返回 FALSE
53  }
```

- 绑定文本编辑框中的画笔宽度，将其绑定到一个int值上即可



- 复杂的控件绑定到MFC对应对象的类型上，



CMFCColorButton 控件

代码示例：

```
139     // 保存当前画笔和画刷的设置属性
140     m_cmbPenStyle.SetCurSel(m_nPenStyle);
141     m_nEdtPenWidth = m_nPenWidth;
142     m_btnPenColor.SetColor(m_clrPenColor);
143     m_btnBrushColor.SetColor(m_clrBrushColor);
144     m_cmbHatchStyle.SetCurSel(m_nHatchStyle);
145
146     // 解决CMFCColorButton控件点击other...不弹窗问题
147     m_btnPenColor.EnableOtherButton(_T("Other..."), FALSE, TRUE);
148     m_btnBrushColor.EnableOtherButton(_T("Other..."), FALSE, TRUE);
149
150     UpdateData(FALSE);          // 更新控件数据
151
```

保存画笔画刷风格

点击关闭按钮，将修改的数据传出，需要在 WM_CLOSE 消息里保存修改的数据，在对话框类的类向导里添加 WM_CLOSE 消息。

每个图形都需要有自己的画笔和画刷，没有的话修改画笔画刷属性后，原来的画过的图形的相关属性也会受到影响。为每个图像都使用自己的画笔相关的属性，在自己的类中 OnDraw 的时候，使用自己的画笔相关属性。

不能再 CxxView 类的成员函数 OnDraw 里绘制画笔，会导致绘制的图形画笔属性一样。

在绘制的时候（当图形创建的时候）就应该显示当前绘制图形的画笔相关属性，在 LBUTTONDOWN 里进行设置当前图形的画笔属性。
代码示例：

```
222  // 鼠标按下保存当前图形的起点
223 void CCADView::OnLButtonDown(UINT nFlags, CPoint point)
224 {
225 #if 0 非活动预处理器块
230 #endif // 使用函数指针
231
232  //使用类工厂
233  if (m_pShapeFactory != NULL)
234  {
235  // 获取当前正在绘制的图形（创建图形）
236  m_pCurrentShape = m_pShapeFactory->CreateShape();
237  }
238  // 检查图形是否创建成功
239  if (m_pCurrentShape != NULL)
240  {
241  // 图像创建成功，就设置对应的画笔属性
242  m_pCurrentShape->myPen.SetPenStyle(m_nPenStyle);
243  m_pCurrentShape->myPen.SetPenWidth(m_nPenWidth);
244  m_pCurrentShape->myPen.SetPenColor(m_clrPenColor);
245
246  // 设置对应的画刷属性
247  m_pCurrentShape->myPen.SetBrushColor(m_clrBrushColor);
248  m_pCurrentShape->myPen.SetHatchStyle(m_nHatchStyle);
249
250  // 鼠标按下保存当前图形的起点
251  m_pCurrentShape->SetPointBegin(point);
252  }
253
254  // 鼠标左键按下，开始捕获窗口外的鼠标消息
255  SetCapture();
256
257 #if 0 非活动预处理器块
262 #endif // 测试pair的使用
263
264  CView::OnLButtonDown(nFlags, point);
265 }
```

将每次更改的画笔，画刷的属性进行保存，方便下次使用，再对画笔画刷对话框类的初始化函数内，保存当前画笔，画刷的相关属性。
代码示例：

```
139  // 保存当前画笔和画刷的设置属性
140  m_cmbPenStyle.SetCurSel(m_nPenStyle);
141  m_nEdtPenWidth = m_nPenWidth;
142  m_btnPenColor.SetColor(m_clrPenColor);
143  m_btnBrushColor.SetColor(m_clrBrushColor);
144  m_cmbHatchStyle.SetCurSel(m_nHatchStyle);
145
146  // 解决CMFCColorButton控件点击other...不弹窗问题
147  m_btnPenColor.EnableOtherButton(_T("Other..."), FALSE, TRUE);
148  m_btnBrushColor.EnableOtherButton(_T("Other..."), FALSE, TRUE);
149
```

弹出画笔画刷属性对话框之前，应该将当前绘制图形画笔画刷的相关属性显示到对应的属性数值上。

代码示例：

```
// 设置画笔画图
void CCADView::OnSetPenbrush()
{
// 对话框关闭后，窗口已经销毁，不能通过消息的方式获取数据
//int nPenWidth = penbrudlg.GetDlgItemInt(EDIT_PENWIDTH);

CPenBrushDialog penbrudlg;
// 画笔画刷风格对话框内显示当前图形的风格属性数值
penbrudlg.SetPenStyle(m_nPenStyle);
penbrudlg.SetPenWidth(m_nPenWidth);
penbrudlg.SetPenColor(m_clrPenColor);
penbrudlg.SetBrushColor(m_clrBrushColor);
penbrudlg.SetHatchStyle(m_nHatchStyle);
```