图形编辑程序 Drawing.exe

–- LSH.Yangtze University 2017.12.3

MFC的“文档-视图”框架，是专门为“文档数据编辑与文件存储”类型的应用而设计的。该框架包括单文档（SDI）和多文档（MDI）等两种类型；两者的异同之处：

* 单文档程序只能操作一个文档，当打开新的文档时候，原来的文档必须关闭。
* 而多文档程序则可以同时打开多个文档，分别进行文档编辑操作。
* SDI和MDI均支持“一个文档多个视图”。

本程序可以采用SDI或MDI实现，SDI相对简单，本文采用了SDI。

1. 创建项目 — Drawing

在Microsoft VS 6.0及以上版本中，创建VC++语言的“单文档MFC应用程序”项目。

项目名称：Drawing，项目选项：“MFC应用程序”、“单文档”、“项目类型--MFC标准”等，另外，建议将CDrawingView的父类由CView改为CScrollView。

APPWizard向导会产生一个MFC应用程序代码框架，其中主要有4个类，如图1所示。

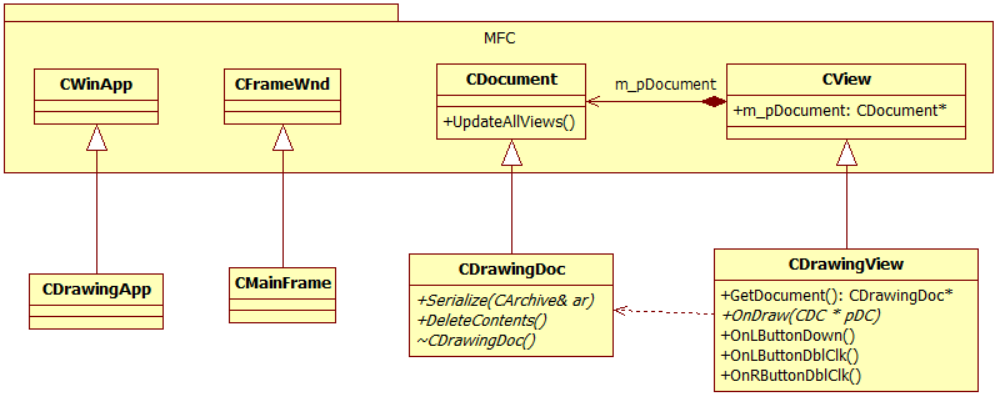


图1. AppWizard向导生成项目Drawing的类图**\***

\*注：（1）图中CView类可改为CScrollView，当图纸超出窗口大小时，以便滚动窗口进行显示；（2）CDrawingDoc和CDrawingView类的成员函数，除GetDocument()外，均需要编码实现，详见需求分析。

四个类的作用：

1. CDrawingApp应用程序类：负责启动和初始化Windows程序；创建CMainFrame类主框架窗口、CDrawingDoc类文档和CDrawingView类视图，以及框架窗口、文档、视图之间的连接；创建程序菜单，并将“文件-新建”、“文件-打开”、“帮助-关于”等菜单消息映射到对应的成员函数。
2. CMainFrame主框架窗口类：在SDI中，作为应用程序的主窗口，负责创建框架窗口的工具栏、状态栏。
3. CDrawingDoc文档类：负责存储文档的内部数据对象（本例为正方形、矩形、圆、椭圆、正三角形、文字等6种图形对象），并提供序列化**\***（Serialize）技术实现文档内部数据对象的文件读、写操作。
4. CDrawingView视图类：负责对文档数据对象的显示，以及对象数据的编辑等用户交互操作。

\*注：Serialize序列化由MFC的根类CObject提供，即序列化对象类必须由CObject派生而来。

文档-视图的交互：

1. CDocument::UpdateAllViews()： 当CDrawingDoc文档内的（图形）对象有变化时，应调用该函数。该函数作用是通知CDrawingView视图进行重绘，即调用CDrawingView::OnDraw()。
2. CDrawingView::GetDocument()：当在CDrawingView视图中需要访问CDrawingDoc文档内部的（图形）对象时，可调用该函数，从而得到CDrawingDoc文档对象的指针。
3. 需求分析与初步设计

需求分析：

从程序外部看，需要实现的功能有：

1. 编辑图形，包括图形对象的新建、删除和修改等3项功能。
2. 文件操作，要求实现程序菜单中文件的新建、打开、保存、另存为、关闭等功能。

另外，需要对6种图形进行类的设计，包括：正方形、矩形、圆、椭圆、正三角形、文本等。需求分析过程与内容参见表1。

表1. 需求分析过程与内容

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能 | 操作 | 类/方法、属性 | 功能说明 | 隐含/相关功能 |
| 1.编辑图形 | 包括图形对象的新建、删除和修改等操作 | | | |
| 1.1新建图形 | Ctrl+鼠标左键单击 | CDrawingView::  OnLButtonDown() | 在鼠标点击处创建图形（6种） | 存储图形  显示图形  设置图形属性 |
| 1.2删除图形 | 鼠标左键双击 | CDrawingView::  OnLButtonDblClk() | 删除鼠标点击处所选中的图形 | 选中图形 |
| 1.3修改图形 | 鼠标右键双击 | CDrawingView::  OnRButtonDblClk() | 修改鼠标点击所选中图形的属性 | 选中图形  设置图形属性 |
| 1.4存储图形 | （隐含） | CDrawingDoc类 | 存储6种图形对象，应分配到文档类中存储 |  |
| 1.5显示图形 | （隐含） | CDrawingView::  OnDraw() | 绘制文档内存储的所有图形 | 6个图形类需分别实现各自的绘图方法Draw() |
| 1.6选中图形 | （隐含） | 6个图形类的  IsMatched() | 判断鼠标点击处是否落在图形内 | 6个图形类需分别实现IsMatched() |
| 1.7设置图形属性对话框 | （隐含） | CShapeDlg类 | 创建或修改图形时，设置图形属性 |  |
| 2.文件操作 | 实现程序菜单中文件的新建、打开、保存、另存为、关闭等功能。  CDocument已实现并封装了这些功能，只需实现2个函数：Serialize()、DeleteContents()。  另外，在图形的新建、删除和修改时，应调用CDocument:: SetModifiedFlag()方法，说明文档已被修改；这样，在文件的新建、打开时会提示：文件已修改，是否保存？ | | | |
| 2.1序列化 | （隐含） | CDrawingDoc::  Serialize() | 文档内图形对象的文件读、写操作 | 6个图形类需分别实现Serialize() |
| 2.2删除文档内容 | （隐含） | CDrawingDoc::  DeleteContents() | 在文件的新建、打开时会自动调用，该函数负责清空文档内容（即所有图形对象） |  |
| 2.3释放动态内存对象 | （隐含） | CDrawingDoc::  ~ CDrawingDoc() | 关闭程序时，delete删除由new分配的所有对象 |  |
| 3图形类 | 设计并实现正方形、矩形、圆、椭圆、正三角形、文本等6个图形类 | | | |
| 3.1正方形 | （隐含） | CSquare类 | 包括各自的属性以及构造函数、Draw()、Serialize()、IsMatched()等方法 |  |
| 3.2矩形 | （隐含） | CRectangle类 |  |
| 3.3圆 | （隐含） | CCircle类 |  |
| 3.4椭圆 | （隐含） | CEllipse类 |  |
| 3.5正三角形 | （隐含） | CTriangle类 |  |
| 3.6文本 | （隐含） | CText类 |  |

图形类及其存储设计：

设计一个抽象类CShape作为6个图形类的基类，设计类图参见图2。

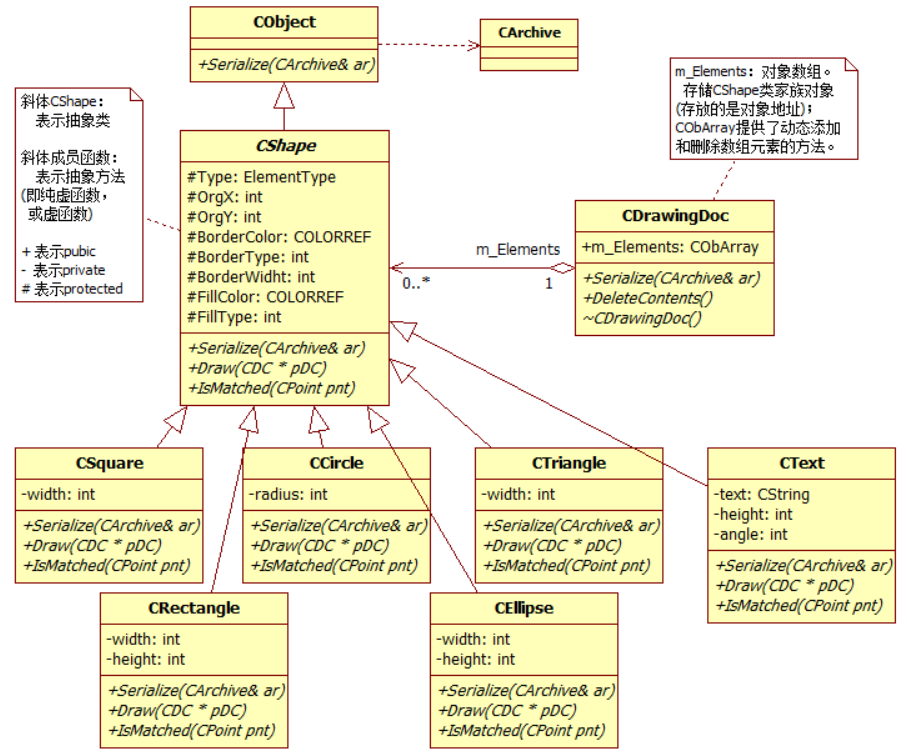


图2. 图形类及其在文档中的存储设计

图形类设计要点：

1. 采用new动态创建图形对象，因为图形对象由用户的操作创建，编程时不可能预知用户想要创建哪个图形和多少个图形；
2. 图形对象存储在CDrawingDoc类中，且可以动态添加和删除，故应该采用集合类（指MFC的CArray、CList、CMap等类）来存储（选择CObArray类较好），且应将6个图形类设计成一个类簇；
3. 支持序列化，即Serialize()方法，故应从MFC的CObject类派生；
4. Serialize()、Draw()、IsMatched()等方法应该采用动态联编，即虚函数。
5. 工作计划 — 编码、测试与阶段划分

当程序的工作量较大时，应该制定一个工作计划，将开发工作按阶段进行。每个开发阶段实现一些相关的功能，然后测试，这样可以避免问题积累过多，且可以及时纠正设计和编码过程中产生的问题。开发阶段可以按照工作量大小、功能相关性、便于测试等进行划分，这是一种能不断满足新功能需求的滚动式迭代开发方法。

下面给出了一个开发计划，供参考。该计划分为四个阶段，并给出前2个阶段的实现代码，参见表1：

表1. 详细开发计划

| **阶段** | **类/类型** | **成员** | **内容及说明** | **备注** |
| --- | --- | --- | --- | --- |
| 1 | 目标：实现1~2个图形类的绘制功能。（下面以正方形类CSquare为例） | | | |
|  | ElementType |  | enum ElementType { NOTSET, SQUARE, RECTANGLE, CIRCLE, ELLIPSE, TRIANGLE, TEXT }; | 枚举，代表各种图形类的编号 |
|  | CShape | 图2中所有成员 |  |  |
|  | CSquare | CSquare() | 除原点和宽度外，其它属性使用默认值 | 构造函数后期完善 |
|  | CSquare | Serialize() | void CSquare::Serialize(CArchive &ar)  { } | 空函数体，  阶段2实现 |
|  | CSquare | Draw() | void CSquare::Draw(CDC\* pDC)//绘制图形函数  {  CPen pen, \*pOldPen;  pen.CreatePen(BorderType, BorderWidth, BorderColor);  pOldPen = (CPen\*)pDC->SelectObject(&pen);  CBrush brush, \*pOldBrush;  if ( FillType >= HS\_HORIZONTAL && FillType <= HS\_DIAGCROSS)  brush.CreateHatchBrush(FillType, FillColor);  else  brush.CreateSolidBrush(FillColor);  pOldBrush = (CBrush\*)pDC->SelectObject(&brush);  pDC->Rectangle(OrgX - width / 2, OrgY - width / 2, OrgX + width / 2, OrgY + width / 2);  pDC->SelectObject(pOldPen);  pDC->SelectObject(pOldBrush);  } | 完整实现。  画正方形使用：pDC->Rectangle()；  包括笔、刷及类型、颜色等的选择。 |
|  | CSquare | IsMatched() | bool CSquare::IsMatched(CPoint pnt)//图元匹配函数  { return false; } | 阶段4实现。  鼠标点是否落在了正方形区域之内 |
|  | CDrawingDoc | m\_Elements | 加public属性：CObArray m\_Elements; |  |
|  | CDrawingView | OnDraw() | void CDrawingView::OnDraw(CDC\* pDC)  {  CDrawingDoc\* pDoc = GetDocument();  ASSERT\_VALID(pDoc);  if (!pDoc)  return;  // TODO: 在此处为本机数据添加绘制代码  for (int i = 0; i < pDoc->m\_Elements.GetCount(); i++)  {  CShape\* p = (CShape\*)pDoc->m\_Elements[i];  p->Draw(pDC);  }  } | 完整实现。  遍历m\_Elements的数组元素，调用Draw()方法。 |
| 测试 | CDrawingView | OnLButtonDown() | void CDrawingView::OnLButtonDown(UINT nFlags, CPoint point)  {  // TODO: 在此添加消息处理程序代码和/或调用默认值  if ((nFlags&MK\_CONTROL) == MK\_CONTROL)//Ctrl键按下  {  CDrawingDoc\* pDoc = GetDocument();  ASSERT\_VALID(pDoc);  if (!pDoc) return;  CClientDC dc(this);  CPoint pntLogical = point;  OnPrepareDC(&dc);  dc.DPtoLP(&pntLogical);//DP->LP进行转换  // ----- 测试代码 begin -----  CShape \* p = new CSquare(pntLogical.x, pntLogical.y, 100);  pDoc->m\_Elements.Add(p);  pDoc->SetModifiedFlag();  pDoc->UpdateAllViews(NULL);  // ----- 测试代码 end -----  }  CScrollView::OnLButtonDown(nFlags, point);  } | 测试代码。  在鼠标点击处new创建CSquare对象，并添加到m\_Elements中 |
| 测试 | 测试内容：在窗口不同位置试几次“Ctrl+鼠标左键单击”，看是否正确绘出了OnLButtonDown()创建的图形； | | | |
| 2 | 目标：实现序列化和文件打开与保存等功能。 | | | |
|  | CSquare | Serialize() |  | 完整实现。 |
|  | CDrawingDoc | Serialize() | void CDrawingDoc::Serialize(CArchive& ar)  {  m\_Elements.Serialize(ar);  } | CObArray类有自己的Serialize()方法，它会对数组中的每个元素（图形）调用Serialize()方法 |
|  |  | DeleteContents() | void CDrawingDoc::DeleteContents()  {  // TODO: 在此添加专用代码和/或调用基类  for (int i = 0; i < m\_Elements.GetSize(); i++)  {  CShape\* p = (CShape\*)m\_Elements[i];  delete(p);  }  m\_Elements.RemoveAll();  CDocument::DeleteContents();  } | 使用类向导添加该函数，然后在函数内添加代码。 |
|  |  | ~CDrawingDoc() | CDrawingDoc::~CDrawingDoc()  {  this->DeleteContents();  } | 程序关闭前的清理工作。 |
| 测试 | 测试内容：文件的新建、打开、保存、另存为、关闭等功能。 | | |  |
| 3 | 目标：实现全部6个图形类，以及绘制Draw()、序列化Serialize()等方法 | | | |
|  | CRectangle | 参照CSquare | 参照阶段1的CSquare。画矩形pDC->Rectangle() |  |
|  | CCircle | 参照CSquare | 参照阶段1的CSquare。画圆pDC->Ellipse() |  |
|  | CEllipse | 参照CSquare | 参照阶段1的CSquare。画椭圆pDC->Ellipse() |  |
|  | CTriangle | 参照CSquare | 参照阶段1的CSquare。画多边形pDC->Polygon() |  |
|  | CText | 参照CSquare | 参照阶段1的CSquare。创建字体、输出文本 |  |
| 测试 | CDrawingView | OnLButtonDown() | new几个CSquare、CRectangle、CCircle、CEllipse、CTriangle、CText对象，并添加到m\_Elements中 | 测试代码。 |
| 测试 | 测试内容：同阶段1、2的测试内容。 | | | |
| 4 | 目标：实现6个图形类的IsMatched()方法，和图形属性对话框CShapeDlg类。 | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | | | |

第4阶段

程序的设计类图，参见图3。

程序运行效果，参见图4。

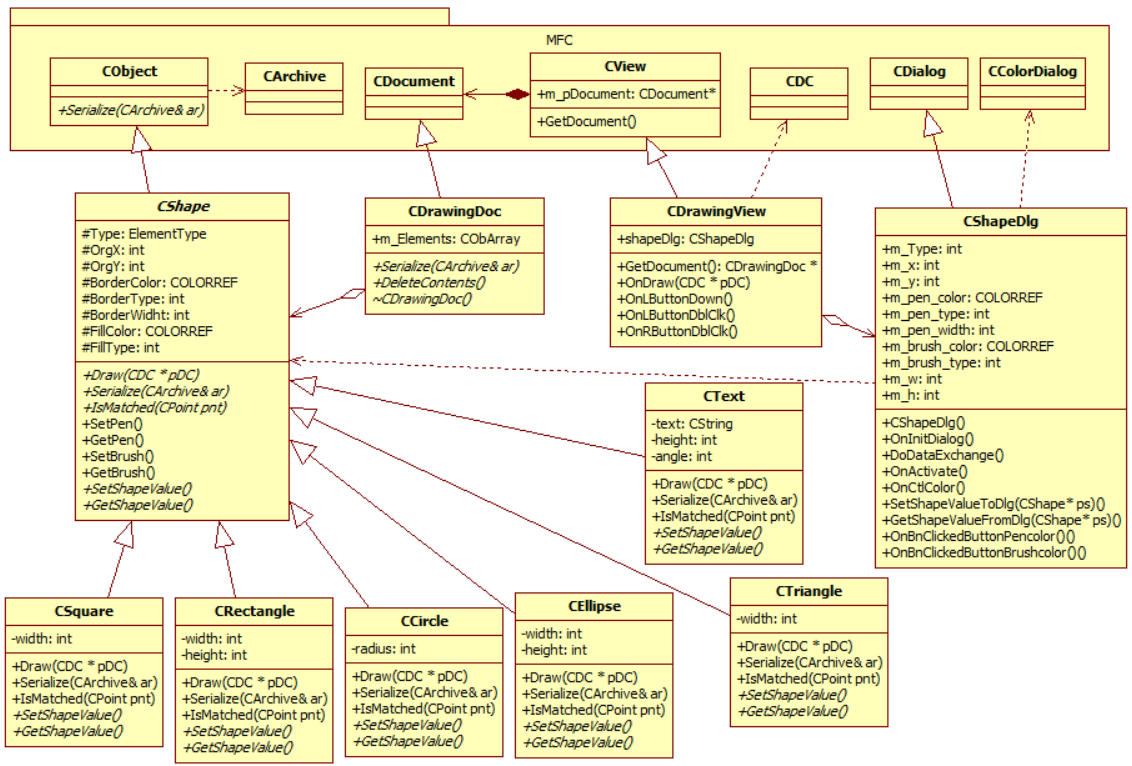


图3. 设计类图（仅供参考）

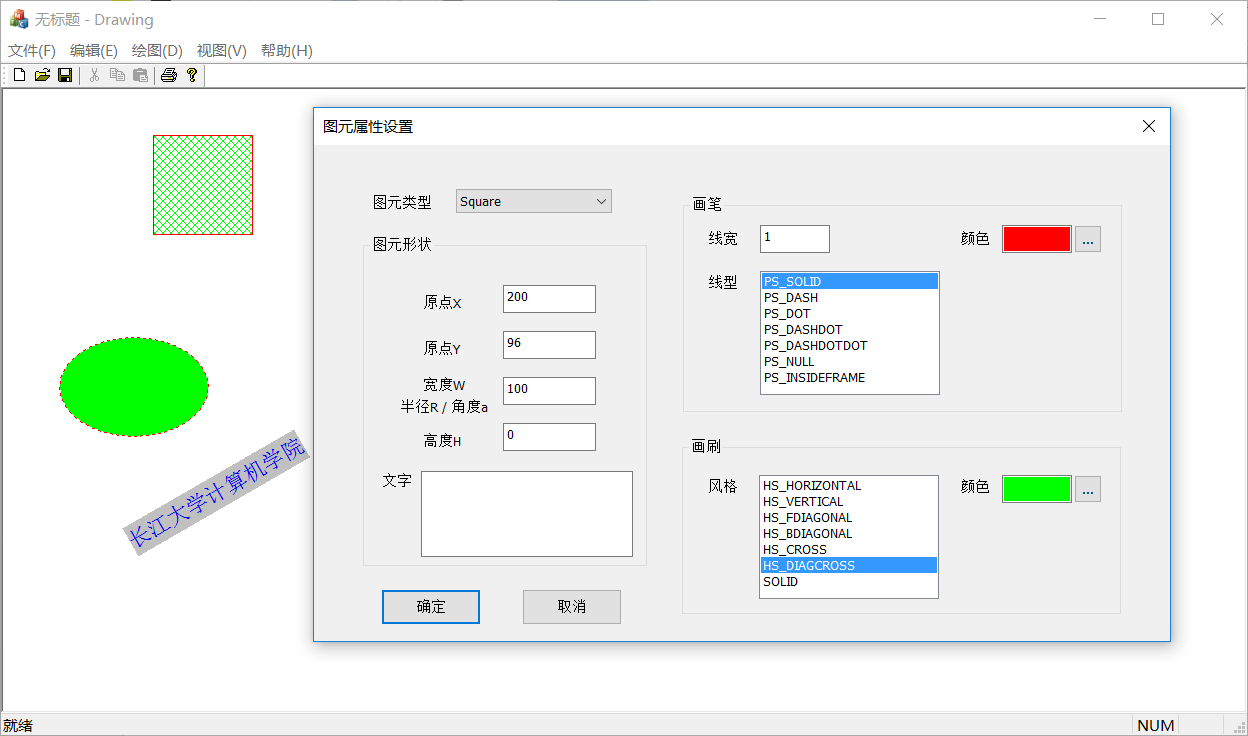


图4. 程序运行效果