第 7 章　操作图形数据库

CRxDbDatabase 类代表电子图板图形文件，每个 CRxDbDatabase 对象都包含许多系统变量、符号表、符号表记录、实体和组成图形的其他对象。

CRxDbDatabase 类提供了如下的几类成员函数：

- 访问所有符号表。
- 读写 DWG 文件。
- 获得和设置数据库的特性。
- 执行多种数据库层的操作，例如写块和深层克隆。
- 获得和设置系统变量。

一定要将图形数据库（CRxDbDatabase）和文档（CRxApDocument）区分开，前者代表了电子图板图形文件，后者仅仅是为了实现 MDI（多文档用户界面）而提供的一个接口而已。

每个打开的图形都有一个关联的 CRxApDocument 对象，CRxApDocument 对象包含了一些信息，例如文件名称、MFC 文档对象、当前数据库和当前图形的保存格式等。除了这些方面之外，在其他的任何情况下，请忘记 CRxApDocument。

本章要介绍的是与 CRxDbDatabase 有关的操作，例如创建图形数据库和访问图形数据库内容，通过写块、插入数据库实现两个图形数据库的内容传递，长事务处理实现在位编辑，以及读写图形的摘要信息等。

7.1 创建和访问图形数据库

7.1.1 说明

本节的实例创建一个新的图形数据库，添加两个圆并将其保存在 电子图板的安装目录下（以帮助系统中的 createDwg 函数为基础改写而成）；局部加载该图形文件，显示加载的所有实体的类名称（例如 CRxDbLine、CRxDbCircle 等）。

需要注意的是，本节所介绍的实例均不能在图形窗口中显示创建或打开的图形数据库.

7.1.2 思路

1．图形数据库的基本操作

可以使用下面的语句新建一个图形数据库：

```
CRxDbDatabase*pDb = new CRxDbDatabase();
```

CRxDbDatabase 类的构造函数为：

```
CRxDbDatabase(bool buildDefaultDrawing = true, bool noDocument = false);
```

其中，buildDefaultDrawing 指出是否创建一个空的图形数据库，也就是是否包含图形数据库的初始内容（默认的符号表、命名对象字典和一组系统变量）；noDocument 指出新建的图形数据库是否与当前文档相关联。saveAs 函数用于保存图形数据库，在指定文件名称时必须包含 dwg 扩展名。readDwgFile 函数将一个已经存在的图形文件的内容读入到当前图形数据库，但是调用该函数的图形数据库必须用下面的语句来创建：

```
CRxDbDatabase*pDb = new CRxDbDatabase(CAXA::kFalse);
```

2．获得电子图板的安装路径

获得 电子图板的安装路径可以使用三种方法：

 读取注册表 HKEY_LOCAL_MACHINE\ SOFTWARE\ CAXA\ 电子图板\ 11.0 项中 InstallDir 键的键值，如图 7.1 所示。其中，对应于不同的电子图板版本，InstallDir 的内容可能有所不同。

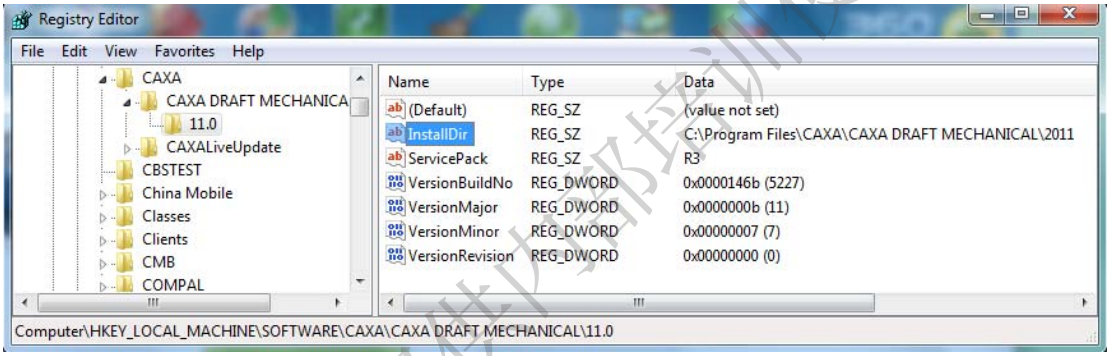

图7.1 注册表中保存的电子图板安装路径

 使用 COM 技术调用 ActiveX 模型中相应的属性，如果在 VBA 中描述应该是下面的形式：

```
Sub GetCDraftPath()
MsgBox Application.FullName
End Sub
```

 使用 Windows API 函数 GetModuleFileName，例如下面的代码可以在 CString 类型的变量 cdraftPath 中保存电子图板的安装路径(实际上准确说应该是 CDRAFT_M.exe 文件的路径）：

```
DWORD dwRet
= ::GetModuleFileName(acedGetCDraftWinApp()->m_hInstance,
cdraftPath.GetBuffer(_MAX_PATH) , _MAX_PATH);
cdraftPath.ReleaseBuffer();
```

其中，_MAX_PATH 是 Windows 标准库中定义的一个常量，保存 Windows 的最大路径长度。

> 提示：选择 Windows 开始菜单中的【运行】菜单项，在弹出的【运行】窗口中输入 regedit 然后单击【确定】按钮即可打开注册表编辑器。

3．局部加载

要实现图形的局部加载，应该在调用 readDwgFile 函数之后执行 CRxDbDatabase 类的 applyPartialOpenFilters 函数，最后还要执行 closeInput 函数。

applyPartialOpenFilters 函数的定义为：

```
CDraft::ErrorStatus applyPartialOpenFilters(
const AcDbSpatialFilter* pSpatialFilter,
const AcDbLayerFilter* pLayerFilter);
```

其中，pSpatialFilter 指定了模型空间中的一个三维区域进行空间过滤，pLayerFilter 则指定了所要进行过滤的图层。

7.1.3 步骤

（1）启动 Visual Studio 2010，使用 ObjectCRX 向导创建一个新工程，其名称为 CreateDatabase。在创建工程的过程中，注意要选择【 MFC Extention Support】选项，如图 7.2 所示。

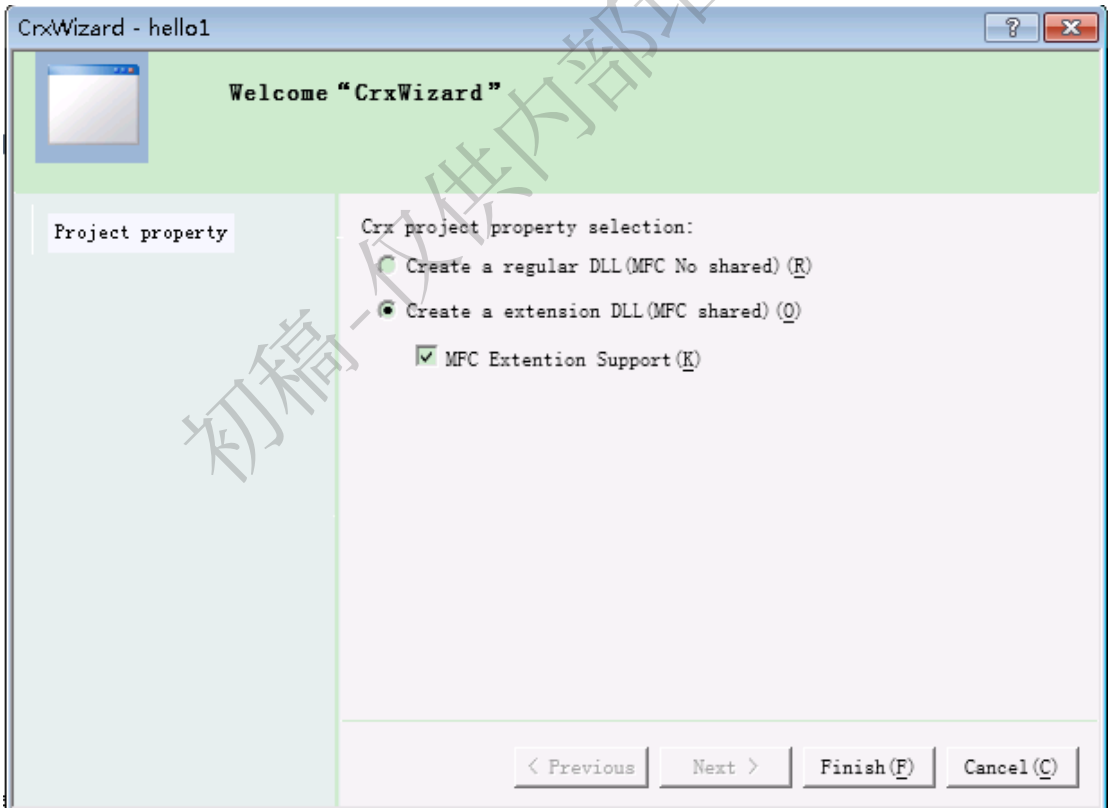

图7.2 在向导中选择“使用MFC”的选项

（2）注册一个新命令 CreateEXB，用于创建一个新的图形文件，并保存在电子图板的

安装路径中，其实现函数为：

```
void CRXCreateEXB()
{
    // 创建新的图形数据库，分配内存空间
    CRxDbDatabase*pDb = new CRxDbDatabase();
    CRxDbBlockTable *pBlkTbl;
    pDb->getSymbolTable(pBlkTbl, CRxDb::kForRead);
    CRxDbBlockTableRecord *pBlkTblRcd;
    pBlkTbl->getAt(ACDB_MODEL_SPACE, pBlkTblRcd,
        CRxDb::kForWrite);
    pBlkTbl->close();
    // 创建两个圆
    CRxDbCircle *pCir1 = new CRxDbCircle(CRxGePoint3d(1,1,1),
        CRxGeVector3d(0,0,1), 1.0);
    CRxDbCircle *pCir2 = new CRxDbCircle(CRxGePoint3d(4,4,4),
        CRxGeVector3d(0,0,1), 2.0);
    pBlkTblRcd->appendAcDbEntity(pCir1);
    pCir1->close();
        pBlkTblRcd->appendAcDbEntity(pCir2);
    pCir2->close();
    pBlkTblRcd->close();
    CString cdraftPath=_T("C:\\");
CString cdraftPath;
    GetCDraftPath(cdraftPath); // 获得CDRAFT_M.exe的位置
    // 去掉路径最后的"CDRAFT_M.exe"字符串，得到电子图板安装路径
    cdraftPath = cdraftPath.Left(cdraftPath.GetLength() - 8);
    CString filePath = cdraftPath + _T("test.exb");
    // 使用saveAs成员函数时，必须指定包含dwg扩展名的文件名称
    pDb->saveAs(filePath.GetBuffer(0));
    filePath.ReleaseBuffer();
    delete pDb; // pDb不是数据库的常驻对象，必须手工销毁
}
```

上面的代码中使用了 MFC 的 CString 类，其 GetLength 函数用于获得其中包含的字符串的长度。使用 CString 类连接字符串特别方便，可以直接使用+运算符与字符串连接，返回一个新的 CString 类对象。如果你愿意，当然可以使用“cdraftPath = cdraftPath + "test.exb";”，这里为了加强变量的可读性使用了 filePath 来作为文件的名称。

CRxDbDatabase 类的 saveAs 函数接受一个 const CxCHAR*类型的变量(字符串常量)，可以直接使用 CString 类的 GetBuffer(0)来获得一个指向保存在 CString 对象中的字符指针，以此作为 saveAs 函数的参数。每次执行完毕 GetBuffer 函数之后，应尽快使用 ReleaseBuffer 函数来释放分配的缓冲区。代码中的 GetCDraftPath 是一个自定义函数，用

于获得当前运行的电子图板程序的 CDRAFT_M.exe 的位置，其实现代码为：

7.1.4 效果

（1）编译链接程序，启动电子图板 2011，加载生成的 CRX 文件。执行 CreateEXB 命令，然后选择【文件 / 打开】菜单项，打开生成的 test.exb 文件，执行 ZOOM 命令并选择 E 选项，观察图形中已经创建的两个圆。

（2）选择【格式 / 图层】菜单项，打开图层特性管理器，创建两个新图层 Circle 和 Line, 并将其颜色修改为红色和蓝色，如图 7.3 所示。

图7.3 创建新图层

（3）删除图形中的两个圆，使用 Rectangle 命令创建一个角点为（0，0）和（100，100）的矩形，然后分别使用 Line 和 Circle 命令创建若干个圆和直线，确保有一条直线位于矩形的外部，并分别将两条直线放置在 Line 层上，将一个圆放置在 Circle 层上，如图 7.4 所示。保存并关闭 test.exb 图形。

图7.4 创建测试实体

（4）执行 ReadEXB 命令，能够在命令窗口得到如图 7.5 所示的结果。

图7.5 显示图形数据库中所有实体的类名称

这个结果与预期是一致的，因为局部加载包含了两个过滤条件：（1）在矩形内部；（2）在图层 Circle 或 Line 上。满足条件的实体仅有两个，因此被加载到图形数据库中的实体仅有两个。

7.1.5 小结

学习本节的内容之后，读者需要掌握下面的知识点：

- 获得 电子图板安装路径。
- 创建新的图形数据库和读取已经存在的图形文件。
- 局部加载的实现。
- CString 类型变量的使用。