第 3 章　块和属性

读者会发现创建块与第 2 章中的创建基本对象非常相似。本章介绍块和属性的相关应用，包括创建和插入普通图块、创建和插入带有属性的图块，以及在对话框中预览块的图标。但是通过本章的知识了解块和属性与图形数据库的关系之后，

3.1 创建块定义

3.1.1 说明

本节的实例创建了一个块定义，也就是实现了 电子图板中创建块的操作。块定义中包含两条直线和一个圆，创建块定义之后，在电子图板 2011 中选择【绘图 / 块/插入】菜单项，就能将定义的块插入到图形中。

3.1.2 思路

必须深入理解数据库的结构，才能理解数据库中各种对象的操作。所有的实体都保存在块表记录中，而块表记录则存储在块表中。实际上，用户在电子图板中定义块相当于增加了一个块表记录，块表记录的名称就是块定义的名称。

在每个块表记录下都有 CRxDbBlockBegin 和 CRxDbBlockEnd 两个节点，两者之间的节点表示块表记录中存在的实体，例如“圆形”块定义包含了一条直线和一个圆。比较之后还可以发现，除了名称不同，用户自定义的“圆形”、“矩形”块表记录和图形数据库中的*Model_Space 块表记录在结构上没有任何区别。

于是，仿照在图形数据库中创建实体的步骤，给出创建块定义的一般步骤：

（1）获得当前图形数据库的块表，向其中添加一条新的块表记录。

（2）创建组成块定义的实体，将其添加到新的块表记录中。

（3）关闭块表、块表记录和新创建的实体。

3.1.3 步骤

在 Visual Studio 2010 中，使用 ObjectCRX 向导创建一个新工程，命名为 MakeBlkDef。注册一个新命令 AddBlk，其实现函数为：

```
void  CRXAddBlk()
{
```

```
    // 获得当前图形数据库的块表
    CRxDbBlockTable *pBlkTbl;
    crxdbHostApplicationServices()->workingDatabase()
        ->getBlockTable(pBlkTbl, CRxDb::kForWrite);
    // 创建新的块表记录
    CRxDbBlockTableRecord *pBlkTblRcd;
    pBlkTblRcd = new CRxDbBlockTableRecord();
    // 根据用户的输入设置块表记录的名称
    CxCHAR blkName[40];
        if (crxedGetString(CAXA::kFalse, _T("\n输入图块的名称："), blkName) !=RTNORM)
        {
            pBlkTbl->close();
            delete pBlkTblRcd;
            return;
        }
        pBlkTblRcd->setName(blkName);
        // 将块表记录添加到块表中
        CRxDbObjectId blkDefId;
        pBlkTbl->add(blkDefId, pBlkTblRcd);
        pBlkTbl->close();
        // 向块表记录中添加实体
        CRxGePoint3d ptStart(-10, 0, 0), ptEnd(10, 0, 0);
        CRxDbLine *pLine1 = new CRxDbLine(ptStart, ptEnd); // 创建一条直线
        ptStart.set(0, -10, 0);
        ptEnd.set(0, 10, 0);
        CRxDbLine *pLine2 = new CRxDbLine(ptStart, ptEnd); // 创建一条直线
        CRxGeVector3d vecNormal(0, 0, 1);
        CRxDbCircle *pCircle = new CRxDbCircle(CRxGePoint3d::kOrigin,
            vecNormal, 6);
        CRxDbObjectId entId;
        pBlkTblRcd->appendAcDbEntity(entId, pLine1);
        pBlkTblRcd->appendAcDbEntity(entId, pLine2);
        pBlkTblRcd->appendAcDbEntity(entId, pCircle);
        // 关闭实体和块表记录
        pLine1->close();
        pLine2->close();
        pCircle->close();
        pBlkTblRcd->close();
}
```

上面的代码中要求用户输入块定义的名称，使用 crxedGetString 函数来获得用户输入的字符串，该函数定义为：

```
int crxedGetString(
int cronly,
const CxCHAR* * prompt,
CxCHAR* * result);
```

cronly 指定用户输入的字符串中是否可以包含空格，可以输入 CAXA::kTrue 或者 CAXA::kFalse；prompt 指定了在命令行提示用户输入的文本；result 则保存了用户输入的结果。

要想正确编译程序，必须添加对头文件“dbents.h”的包含。

3.1.4 效果

（1）编译运行程序，在电子图板 2011 中执行 AddBlk 命令，命令行会提示“输入图块的名称：”，输入 Center 作为图块的名称，按下 Enter 键完成操作，就在当前图形中创建了名为 Center 的块定义。

（2）在电子图板 2011 中，选择【绘图 / 块/插入】菜单项，系统会弹出如所示的【插入】对话框，单击【确定】按钮关闭该对话框。

图3.1设置插入块的参数

（3）在图形窗口中拾取一点作为块参照的插入点，能够得到如图 3.2 所示的块参照。

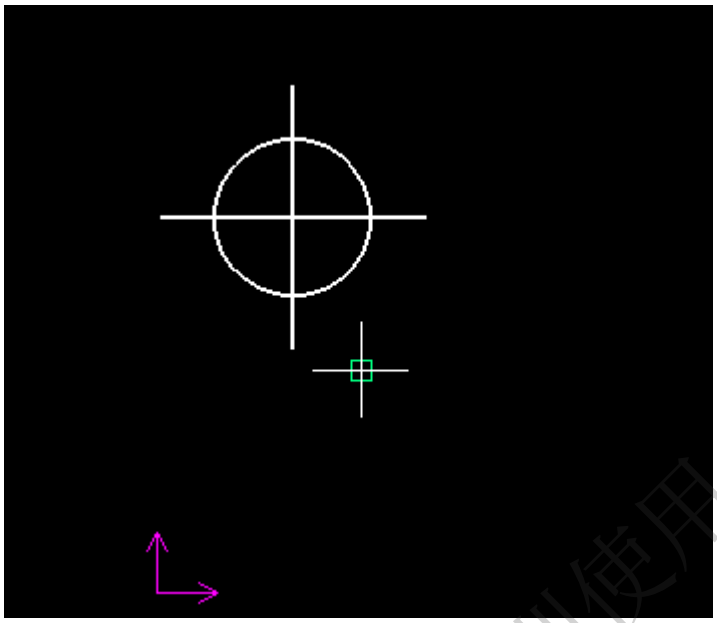

图3.2 插入的块参照

3.1.5 小结

学习本节内容之后，读者要掌握下面的知识点：

- 电子图板图形数据库中实体、块的存储方式。
- 电子图板数据库监视器的使用。
- 使用 crxedGetString 函数来获得用户输入的字符串。

3.2 插入块参照

3.2.1 说明

本实例允许用户输入一个块参照的名称，指定块参照的插入点等参数，然后将块参照插入到模型空间。

3.2.2 思路

如果对 电子图板的操作比较熟悉，就会知道电子图板有块定义和块参照两个概念。这两个概念有什么关系呢？块定义不是一个实体，而是一种对实体的描述，通过定义块获得；块参照则是一种实体，图形窗口中显示的“块”都是块参照，通过插入块获得。

在 ObjectCRX 编程中，块定义通过块表记录来保存，而块参照由 CRxDbBlockReference 类来表示。既然块参照是一种实体，那么创建块参照的过程与创建一条直线似乎不应该有什么区别，事实的确如此！

CRxDbBlockReference 类的构造函数定义为：

```
CRxDbBlockReference(
const CRxGePoint3d& position,
CRxDbObjectId blockTableRec);
```

position 是块参照的插入点；blockTableRec 是块参照所参照的块表记录（块定义）的 ID。

3.2.3 步骤

在 Visual Studio 2010 中，使用 ObjectCRX 向导创建一个新的项目，名称为 InsertBlkRef。注册一个名称为 InsertBlk 的命令，其实现代码为：

```
void CRXInsertBlk()
{
    // 获得用户输入的块定义名称
    CxCHAR blkName[40];
    if (crxedGetString(CAXA::kFalse, _T("\n输入图块的名称："), blkName) !=RTNORM)
    {
        return;
    }
    // 获得当前数据库的块表
    CRxDbBlockTable *pBlkTbl;
    crxdbHostApplicationServices()->workingDatabase()
        ->getBlockTable(pBlkTbl, CRxDb::kForWrite);
        // 查找用户指定的块定义是否存在
        CString strBlkDef;
    strBlkDef.Format(_T("%s"), blkName);
    if (!pBlkTbl->has(strBlkDef))
    {
        crxutPrintf(_T("\n当前图形中未包含指定名称的块定义！"));
        pBlkTbl->close();
        return;
    }
    // 获得用户输入的块参照的插入点
    crx_point pt;
    if (crxedGetPoint(NULL, _T("\n输入块参照的插入点："), pt) != RTNORM)
    {
        pBlkTbl->close();
        return;
```

```
    }
    CRxGePoint3d ptInsert = asPnt3d(pt);
    // 获得用户指定的块表记录
    CRxDbObjectId blkDefId;
    pBlkTbl->getAt(strBlkDef, blkDefId);
    // 创建块参照对象
    CRxDbBlockReference *pBlkRef = new CRxDbBlockReference();
    pBlkRef->setPosition(ptInsert);
    pBlkRef->setBlockTableRecord(blkDefId);

    // 将块参照添加到模型空间
    CRxDbBlockTableRecord *pBlkTblRcd;
    pBlkTbl->getAt(CRXDB_MODEL_SPACE, pBlkTblRcd,CRxDb::kForWrite);
    CRxDbObjectId entId;
    pBlkTblRcd->appendAcDbEntity(entId, pBlkRef);
    // 关闭数据库的对象
    pBlkRef->close();
    pBlkTblRcd->close();
    pBlkTbl->close();

}
```

为了增加函数的容错能力，增加了判断当前图形的块表中是否包含与指定名称匹配的块表记录的语句，使用 CRxDbBlockTable::has 函数进行检查。crxedGetPoint 函数能够暂停程序，等待用户输入一个点，然后获得该点的坐标。该函数定义为：

```
int crxedGetPoint(
const crx_point pt,
const CxCHAR* * prompt,
crx_point result);
```

pt 在当前 UCS 中指定了一个相对基点，如果指定了该参数，当用户移动鼠标选择要输入的点时，系统会在光标到基点之间创建一根临时直线，如果不需要基点，可以直接输入 NULL；prompt 将会显示在命令行中，给用户提供指导信息，如果不需要该参数，输入 NULL 即可；result 参数返回用户输入的点。关于 crxedGetPoint 函数返回值的问题，请参考第 5 章中关于获得用户输入信息的相关内容。

crx_point 编程中定义的一种数据类型，其定义为：

```
typedef crx_real crx_point[3];
```

而 crx_real 则被定义为：

```
typedef double crx_real;
```

可以看出，crx_point 实际上是一个三维浮点数组，它至今仍在与 ADS 相关的编程中使用。从 crx_point 转换到 CRxGePoint3d 类型的点，即可以通过数组元素直接赋值，也可以通过 asPnt3d 函数直接转化：

□ 通过数组元素交换：

```
ptInsert[X] = pt[X];
ptInsert[Y] = pt[Y];
ptInsert[Z] = pt[Z];
```

□ 使用 asPnt3d 函数（需要包含 geassign.h 头文件）：

```
CRxGePoint3d ptInsert = asPnt3d(pt);
```

之所以可以使用 X、Y 和 Z 直接作为数组的下标，是由于这三个字母在 ObjectCRX 有特殊的定义：

```
enum { X = 0, Y = 1, Z = 2 };
```

3.2.4 效果

编译运行程序，在 电子图板 2011 中创建一个块定义，然后执行 InsertBlk 命令，按照命令提示进行操作：

命令: insertblk

输入图块的名称：line

输入块参照的插入点：

完成操作后，就能在图形窗口中插入指定的块参照，其在 X、Y 和 Z 方向的插入比例均为 1。

如果用户输入的图块名称未在当前图形中定义，系统则会在命令行提示“当前图形中未包含指定名称的块定义！”，并且自动退出。

3.2.5 小结

学习本节实例之后，需要掌握下面的知识点：

□ 结合图形数据库的基本结构，熟悉创建块参照的过程。

□ crxedGetPoint 函数的使用。

□ crx_point 和 CRxGePoint3d 的转化。

3.3 创建带有属性的块定义

3.3.1 说明

本实例创建一个带有属性的块定义，当用户使用 电子图板中的 INSERT 命令向图形窗口中插入该图块时，系统会提示用户输入属性值。

3.3.2 思路

与创建普通的块定义相比，创建带有属性块定义的步骤完全相同，唯一的不同点就是在块定义中包含了一个属性定义的对象。

在 ObjectCRX 中，CRxDbAttributeDefinition 类表示属性定义对象，属性定义是电子图板的一种图形对象（对应于电子图板中的“属性”），可以直接创建该类的一个对象，然后将其添加到块表记录中。

3.3.3 步骤

在 Visual Studio 2010 中，使用 ObjectCRX 向导创建一个名为 BlkWithAttribute 的项目，注册一个命令 AddBlk，其实现函数为：

```
void CRXAddBlk()
{
    // 获得当前图形数据库的块表
    CRxDbBlockTable *pBlkTbl;
    crxdbHostApplicationServices()->workingDatabase()
        ->getBlockTable(pBlkTbl, CRxDb::kForWrite);
    // 创建新的块表记录
    CRxDbBlockTableRecord *pBlkTblRcd;


        pBlkTblRcd = new CRxDbBlockTableRecord();
    // 根据用户的输入设置块表记录的名称
    CxCHAR  blkName[40];
    if (crxedGetString(CAXA::kFalse,_T("\n输入图块的名称："), blkName) !=RTNORM)
    {
        pBlkTbl->close();
        delete pBlkTblRcd;
        return;
    }
    pBlkTblRcd->setName(blkName);
    // 将块表记录添加到块表中
    CRxDbObjectId blkDefId;
    pBlkTbl->add(blkDefId, pBlkTblRcd);
    pBlkTbl->close();
    // 向块表记录中添加实体
    CRxGePoint3d ptStart(-10, 0, 0), ptEnd(10, 0, 0);
    CRxDbLine *pLine1 = new CRxDbLine(ptStart, ptEnd); // 创建一条直线
    ptStart.set(0, -10, 0);
    ptEnd.set(0, 10, 0);
    CRxDbLine *pLine2 = new CRxDbLine(ptStart, ptEnd); // 创建一条直线
```

```
    CRxGeVector3d vecNormal(0, 0, 1);
    CRxDbCircle *pCircle = new CRxDbCircle(CRxGePoint3d::kOrigin,
        vecNormal, 6);
    // 创建一个属性输入直径
    CRxDbAttributeDefinition *pAttDef = new CRxDbAttributeDefinition(ptEnd, _T("20"),
_T("直径"), _T("输入直径"));
    CRxDbObjectId entId;
    pBlkTblRcd->appendAcDbEntity(entId, pLine1);
    pBlkTblRcd->appendAcDbEntity(entId, pLine2);
    pBlkTblRcd->appendAcDbEntity(entId, pCircle);
    pBlkTblRcd->appendAcDbEntity(entId, pAttDef);
    // 关闭实体和块表记录

        pLine1->close();
    pLine2->close();
    pCircle->close();
    pAttDef->close();
    pBlkTblRcd->close();
}
```

与创建普通的块定义相比，上面的函数多创建了一个属性定义，并将其添加到新建的块表记录中。创建属性定义和向块表记录中添加属性定义的语句，在代码中用粗体表示。

CRxDbAttributeDefinition 类是 CRxDbText 类的一个派生类，其构造函数定义为：

```
CRxDbAttributeDefinition(
const CRxGePoint3d& position,
const CxCHAR* text,
const CxCHAR* tag,
const CxCHAR* prompt,
CRxDbObjectId style = CRxDbObjectId::kNull);
```

position 是属性定义的插入点；text 是属性定义默认的显示文字；tag 是属性定义的标记文字；prompt 是属性定义的提示文字；style 是文字样式表记录的 ID，用来指定属性定义所使用的文字样式。

3.3.4 效果

编译运行程序，在电子图板 2011 中执行 AddBlk 命令，输入 blk 作为要创建的块定义的名称，完成带属性的块定义的创建。

在电子图板 2011 中，选择【绘图 / 块/插入】菜单项，系统会弹出如图 3.3 所示的对话框。选择 blk 块定义，然后单击【确定】按钮。

图3.3 选择要插入的块定义

按照命令行的提示进行操作：

命令: INSERT

指定插入点或 [比例(S)/X/Y/Z/旋转(R)/预览比例(PS)/PX/PY/PZ/预览旋转(PR)]:

输入属性值

输入直径 <20>: 40

完成操作后，得到如图 3.4 所示的结果。

图3.4插入块参照的结果

3.3.5 小结

学习本节内容之后，读者需要掌握下面的知识点：

- 创建带属性的块与普通块定义的区别。
- 创建属性定义的方法。

3.4 插入带有属性的块参照

3.4.1 说明

前面一节已经介绍了在 电子图板中插入带属性的块参照的方法，本节将要介绍在程序中插入带属性块的方法。

3.4.2 思路

与插入普通的块定义相比，插入带有属性的块定义比较复杂。插入带有属性的块参照，实际上包含块参照和属性两个部分，属性不能直接存在于图形窗口中，而必须依附于块参照存在。

插入带有属性的块参照，可以按照下面的步骤：

（1）插入一个普通的块参照；

（2）使用遍历器遍历块参照对应的块表记录，如果找到一个属性定义，就创建一个属性，并且附加到块参照上。

3.4.3 步骤

打开 BlkWithAttribute 项目，在其中注册 InsertBlk 命令，该命令的实现函数为：

```
void CRxInsertBlk()
{
    // 获得用户输入的块定义名称
    CxCHAR blkName[40];
    if (crxedGetString(CAXA::kFalse, _T("\n输入图块的名称："), blkName) !=RTNORM)
    {
        return;
    }
    // 获得当前数据库的块表
    CRxDbBlockTable *pBlkTbl;
    crxdbHostApplicationServices()->workingDatabase()->getBlockTable(pBlkTbl,
CRxDb::kForWrite);
    // 查找用户指定的块定义是否存在
    CString strBlkDef;
    strBlkDef.Format(_T("%s"), blkName);
    if (!pBlkTbl->has(strBlkDef))
    {
        crxutPrintf(_T("\n当前图形中未包含指定名称的块定义！"));
        pBlkTbl->close();
        return;
    }
    // 获得用户输入的块参照的插入点
    crx_point pt;
    if (crxedGetPoint(NULL, _T("\n输入块参照的插入点："), pt) != RTNORM)
    {
        pBlkTbl->close();
        return;
    }
    CRxGePoint3d ptInsert = asPnt3d(pt);
```

```
// 获得用户指定的块表记录
CRxDbObjectId blkDefId;
pBlkTbl->getAt(strBlkDef, blkDefId);
// 创建块参照对象
CRxDbBlockReference *pBlkRef = new CRxDbBlockReference(ptInsert,blkDefId);
// 将块参照添加到模型空间
CRxDbBlockTableRecord *pBlkTblRcd;
pBlkTbl->getAt(CRXDB_MODEL_SPACE, pBlkTblRcd,
    CRxDb::kForWrite);
pBlkTbl->close();
CRxDbObjectId entId;
pBlkTblRcd->appendAcDbEntity(entId, pBlkRef);
// 判断指定的块表记录是否包含属性定义
CRxDbBlockTableRecord *pBlkDefRcd;
crxdbOpenObject(pBlkDefRcd, blkDefId, CRxDb::kForRead);
if (pBlkDefRcd->hasAttributeDefinitions())
{
    CRxDbBlockTableRecordIterator *pItr;
    pBlkDefRcd->newIterator(pItr);
    CRxDbEntity *pEnt;
    for (pItr->start(); !pItr->done(); pItr->step())
    {
        pItr->getEntity(pEnt, CRxDb::kForRead);
        // 检查是否是属性定义
        CRxDbAttributeDefinition *pAttDef;
        pAttDef = CRxDbAttributeDefinition::cast(pEnt);
        if (pAttDef != NULL)
        {
            // 创建一个新的属性对象
            CRxDbAttribute *pAtt = new CRxDbAttribute();
            // 从属性定义获得属性对象的对象特性
            pAtt->setPropertiesFrom(pAttDef);
            // 设置属性对象的其他特性
            pAtt->setInvisible(pAttDef->isInvisible());
            CRxGePoint3d ptBase = pAttDef->position();
            ptBase += pBlkRef->position().asVector();
            pAtt->setPosition(ptBase);
            pAtt->setHeight(pAttDef->height());
            pAtt->setRotation(pAttDef->rotation());
```

```
                    // 获得属性对象的Tag、Prompt和TextString
                    CxCHAR *pStr;
                pStr = pAttDef->tag();
                pAtt->setTag(pStr);
                free(pStr);
                pStr = pAttDef->prompt();
                crxutPrintf(_T("%s%s"), _T("\n"), pStr);
                free(pStr);
                pAtt->setFieldLength(30);
                pAtt->setTextString(_T("40"));
                // 向块参照追加属性对象
                pBlkRef->appendAttribute(pAtt);
                pAtt->close();
            }
            pEnt->close();
        }
        delete pItr;
    }
    // 关闭数据库的对象
    pBlkRef->close();
    pBlkTblRcd->close();
    pBlkDefRcd->close();
}
```

遍历块表记录的方法在上一章已经介绍，如果查找到一个属性定义，就创建一个新的属性，属性的各项特性与属性定义有关：

- 属性的图层特性与属性定义保持一致，使用 setPropertiesFrom 函数实现。
- 属性的可见性与属性定义一致。
- 属性的高度和角度与属性定义一致。
- 属性的插入点：属性定义的插入点与块参照插入点的矢量和。
- 属性的标记文字、提示文字与属性定义保持一致。

CRxDbAttributeDefinition 类的 tag 和 prompt 函数返回值均是指向字符串的指针，那么在使用时可以遵照下面的方法：

```
char *pStr;
// 使用pStr
……
free(pStr);
```

可以用 ObjectCRX 的全局函数 crxutDelString 来代替 free 函数。

3.4.4 效果

编译运行程序，在电子图板 2011 中首先执行 AddBlk 命令，输入 blk 作为块定义的名称。执行 InsertBlk 命令，按照命令提示进行操作：

命令: insertblk

输入图块的名称：blk

输入块参照的插入点： 〖拾取一点作为块参照的插入点〗

输入直径

完成操作后，就能将带有属性的块参照插入到当前图形中。

3.4.5 小结

学习本节内容之后，读者需要掌握下面的知识点：

- 使用 setPropertiesFrom 函数复制另一个实体的对象特性。
- 根据属性定义和块参照插入点计算属性的插入点。