



## 第 5 讲实体特征建模

### 目录

1	概述.....	3
1.1	简介.....	3
1.2	建模.....	3
1.3	文件管理.....	5
2	CATIA 零件文件操作.....	7
2.1	创建 Part 文件.....	7
2.2	打开 Part 文件.....	8
3	CATIA 几何特征简介.....	8
3.1	实体特征和曲面特征.....	8
3.2	形态特征和关系特征.....	9
4	CATIA 草图定义.....	9
4.1	创建草图.....	9
4.1.1	创建轴系.....	9
4.1.2	获取草图平面.....	10
4.1.3	创建草图.....	11
4.2	编辑草图.....	11
4.2.1	打开草图编辑.....	11
4.2.2	创建草图元素.....	12
4.2.3	创建草图约束.....	13
4.2.4	创建草图圆角.....	13
4.2.5	关闭草图编辑.....	14
5	实体建模.....	14
5.1	建模工厂（Factory）.....	14
5.2	创建几何体特征集.....	16
5.2.1	获取 CATIMechanicalRootFactory 接口.....	16
5.2.2	创建几何体.....	16
5.3	特征（Feature）的复制.....	16
5.3.1	复制方式一.....	16
5.3.2	复制方式二.....	17
5.4	属性（Property）修改.....	18
6	文件管理对话框实现.....	22
6.1	链表的使用.....	22
6.2	新建对话框.....	23
6.3	对话框功能实现.....	23
6.3.1	刷新按钮.....	23
6.3.2	向右  按钮.....	24
6.3.3	向上  按钮.....	25

6.3.4 输出文件按钮.....	25
7 零件实体建模实现.....	26

# 1 概述

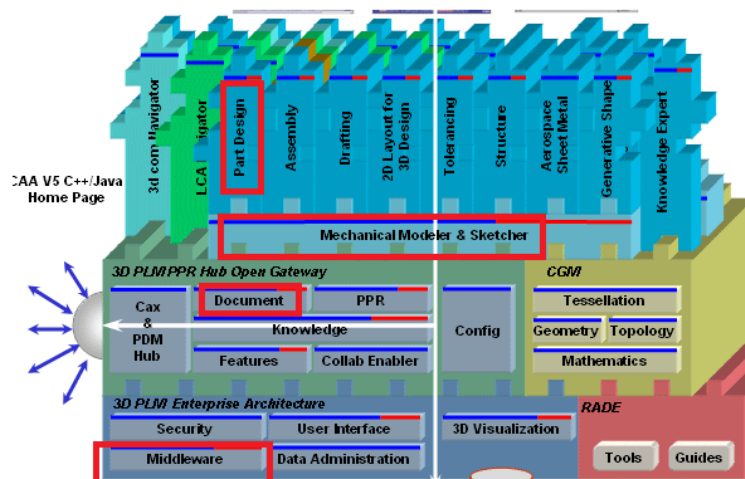
## 1.1 简介

文件和文件夹操作:Middleware->Middleware->(UserCase) Managing Directories and Files。

创建和加载文件: Document->File->(Use Case) Creating a New Document、Loading a Document 和 Exporting a Document Format Type

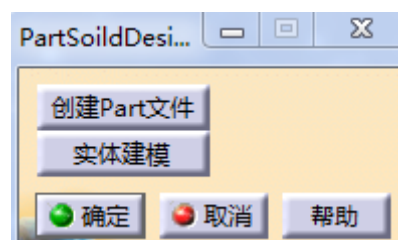
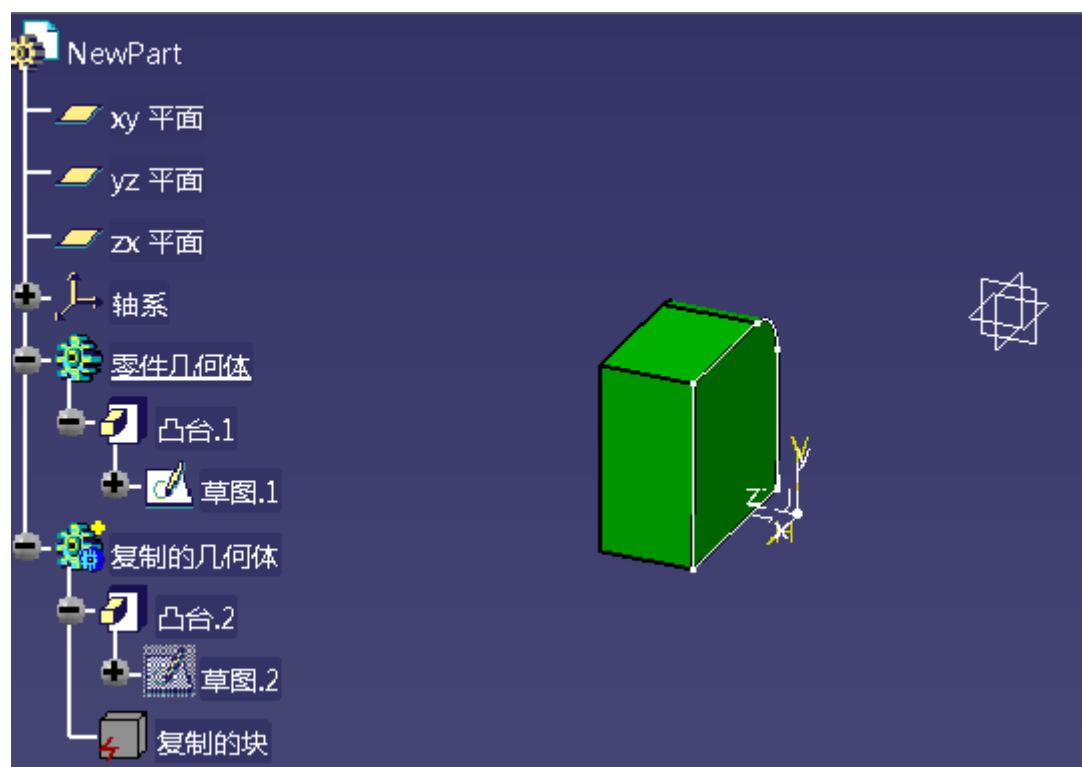
Part 零件设计: Part Design->(Use Case)Implementing the Cut/Copy/Paste Behavior for Mechanical Design Features.和 Creating and Modifying a Mechanical Design Feature

轴系和草图设计: Mechanical Modeler&Sketcher->(Use Case) Creating Axis System、Creating a Sketch on an Axis System Plane、Creating Sketching Elements

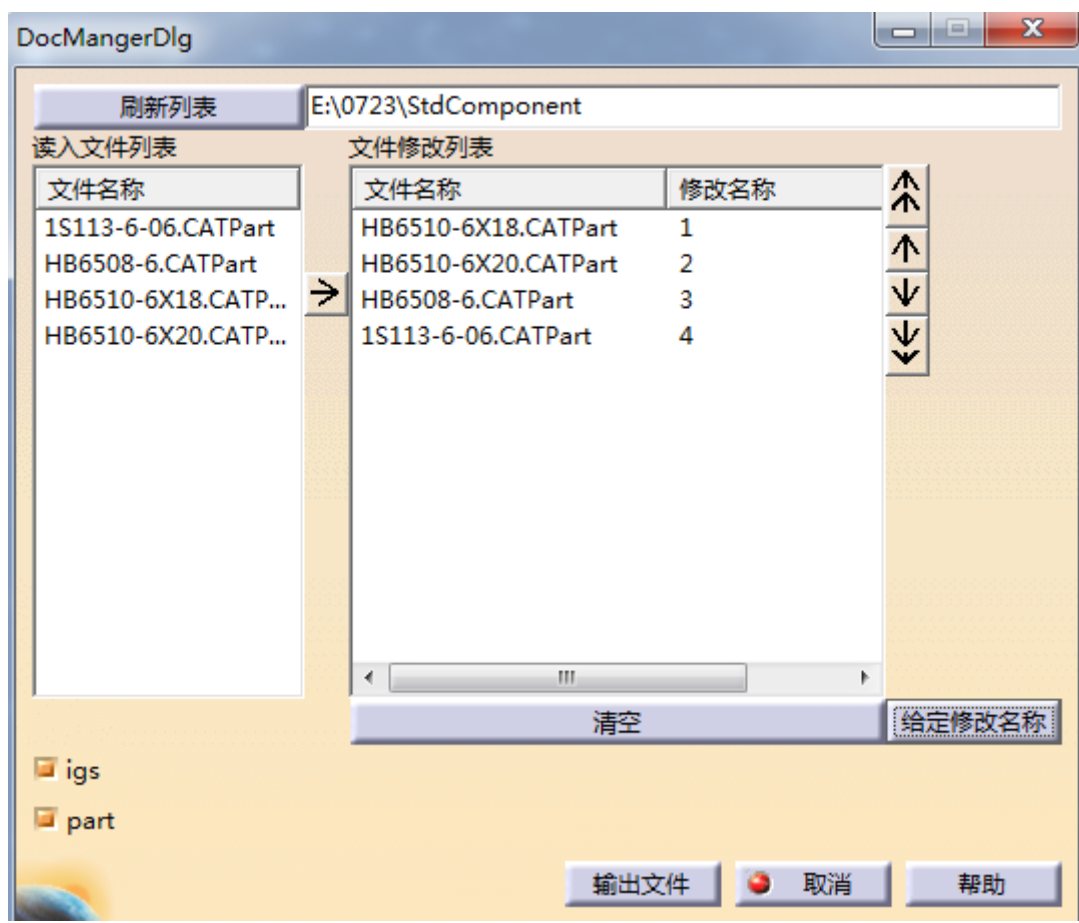


## 1.2 建模

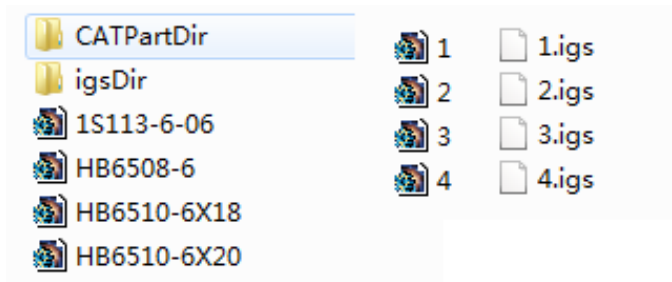
能够创建新 part 文件,在 part 文件中创建一个自定义的轴系,在轴系的 xy 面上创建草图,能够使用草图绘图,草图修饰,草图约束。完成草图后对草图进行拉伸操作,会设置拉伸参数。能够添加几何体,能够复制特征,能够修改特征颜色属性,显示和隐藏属性。能够给特征命名,能够给 part 特征命名。



### 1.3 文件管理



实现读取文件夹中的文件名称，按照新的名称另存文件。



管理文件夹和文件主要是指，检查给定路径和名称的文件或文件夹是否存在；创建给定路径的文件夹；复制文件和文件夹；删除文件和文件夹等文件操作。

1.创建文件和文件夹。创建文件和文件夹代码如下：

```
status = ::CATCreateDirectory(Path);
for (int i=0; i< 10; i++)
{
char name[14];
sprintf(name, "FILE%x", i);
charFilePath[1024];
```

```

        ::CATMakePath(Path, name, FilePath);
        FILE *fd = fopen(FilePath, "w+b");
fclose(fd);
    }

```

创建文件夹采用全局函数 `CATCreateDirectory`，`Path` 是路径参数，创建文件采用 c 语言二进制文件读写方法。如此就在 `Path` 文件夹中创建了 10 个文件。

## 2.浏览文件夹

现在可以用 Windows 资源管理器来对以上创建的一个文件夹和 10 个文件进行浏览，在 CATIA 程序中也可以进行相似的浏览操作。通过浏览可以将文件夹中的文件名称全部获取。

代码如下所示：

```

CATDirectoryDir;
status = ::CATOpenDirectory(Path, &Dir);
intEndOfDir=0;
CATDirectoryEntry Entry;
while ((EndOfDir != 1) && (CATLibSuccess == status))
{
    status = ::CATReadDirectory(&Dir, &Entry, &EndOfDir);
    if ((CATLibError == status) && (EndOfDir !=1))
    {
        cout<< " Can't read next entry in " << Path <<endl;
        exit (-1);
    }
    CATUnicodeStringfilename(Entry.name);//转化文件名称
    cout<< Entry.name <<endl;//输出文件名称
}

```

全局函数 `CATOpenDirectory` 可以打开已经创建了的文件夹，如果打开成功，可以采用 `CATReadDirectory` 全局函数进行读取文件夹中的内容，读取时返回文件名称（`Entry`）和是否到达文件结尾标志（`EndOfDir`），1 表示到达文件夹的最后，0 表示未到达。这样可以依次遍历出文件夹中的所有内容。

## 3.关闭文件夹

```
status = ::CATCloseDirectory(&Dir);
```

## 4 复制文件

全局函数 `CATFCopy(iFromName,iToName)`可以进行文件复制操作。

复制文件时需要提前创建好目标文件夹，比如 `iToName="D:\\New\\123.CATPart"`；复制文件成功的必要前提之一为 New 文件夹已经成功创建。

## 5.删除文件和文件夹

全局函数 `CATDeleteDirectory` 可以删除文件夹，但是前提为删除的文件夹必须为空。

全局函数 CATDeleteFile 可以删除文件。

## 2 CATIA 零件文件操作

### 2.1 创建 Part 文件

CATIA 中采用 CATDocumentServices 接口管理文件创建保存操作。

CATDocumentServices 接口提供了多种创建 CATIA 文件的方法。这些方法包括 CATDocumentServices::New();CATDocumentServices::NewFrom()等。新建不同的文件都是采用这个接口进行创建，不同类型文件的新建参数不同，比如新建一个 CATPart 零件，New()方法中对应的第一个参数字符串为”Part”,如果新建的是一个 CATProduct 文件，New()方法中对应的第一个参数字符串为”Product”。合理的字符串可以在打开 CATIA 后点击文件->新建弹出的对话框中找到，如图所示：



CATDocumentServices 接口同时提供了多种保存文件的方式，如 CATDocumentServices::Save(); CATDocumentServices:: SaveAs (); CATDocumentServices:: SaveAsNew ();

新建一个文件的一般步骤为新建文件，初始化，保存。但是 New()方法已经实现了自动初始化，我们也可以不用再添加初始化代码。新建一个 Part 文件的代码如下所示：

//创建新Part文件

```
CATDocument * pDoc = NULL;  
HRESULT rc = CATDocumentServices::New("Part", pDoc);  
if ( FAILED(rc) || NULL == pDoc ) return ;
```

//新Part文件初始化

```
CATInit * pInit = NULL;  
rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInit);  
if (FAILED(rc) || NULL == pInit ) return ;  
pInit->Init(TRUE);
```

//设置文件路径并保存

```
CATUnicodeString DocTitle="NewPart";  
CATUnicodeString DocName="NewPart.CATPart";
```

```
CATUnicodeString DocPath="D:\\";
```

```
CATDocumentServices::SaveAsNew(*pDoc, DocPath+ DocName);
```

这样就在 D 盘创建了一个名为 NewPart.CATPart 的文件。

## 2.2 打开 Part 文件

CATDocumentServices 接口不仅提供了新建和保存文件的操作方法，还提供了打开文件的操作方法 OpenDocument()。加载代码如下所示：

```
CATDocument *pDoc = NULL;
```

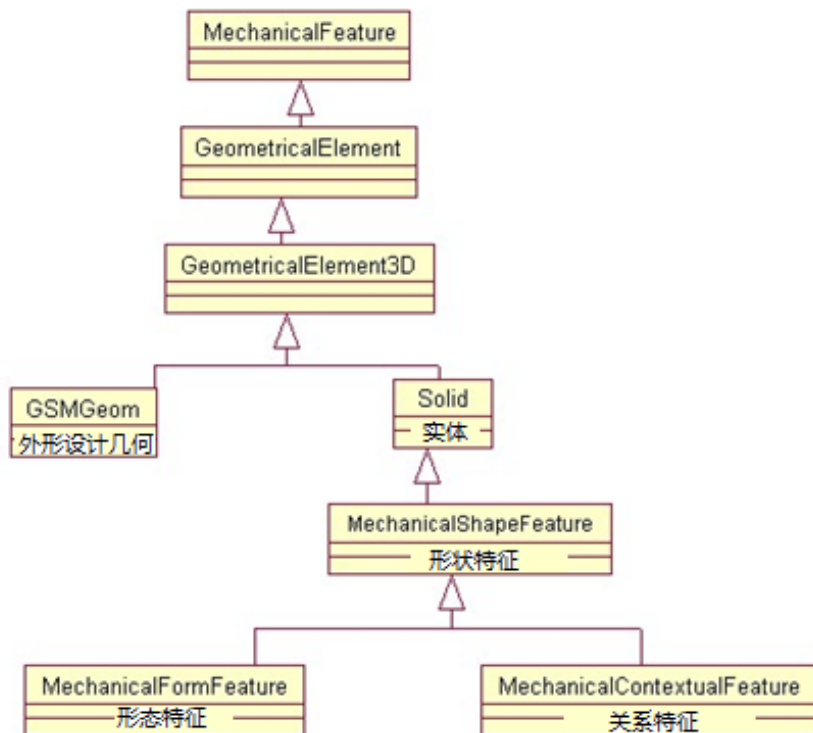
```
rc = CATDocumentServices::OpenDocument(argv[1], pDoc);
```

其中第一个参数是一个 CATIA 文件的完整名称，比如"D:\\ NewPart.CATPart"。它包括了文件路径，后缀名称，文件名称。

打开文件的同时获取了第二个参数，pDoc 是文件指针，在很多操作中都要用到。

## 3 CATIA 几何特征简介

几何特征形成了机械特征，机械特征通过 CATIBuild 接口更新并关联拓扑结果。几何特征从 GeometricalElement 启动项派生而来，下图所示为机械启动树和几何元素架构：



### 3.1 实体特征和曲面特征

实体特征(solid feature)是只与 3D 几何相关的特征，而曲面特征(surfacic feature)是包括 0D，1D，2D，3D 几何特征。点就是一个 0D 的曲面特征，线和草图属于 1D 曲面特征，曲面属于



2D 几何特征，体属于 3D 几何特征。

在一般情况下，从 `MechanicalShapeFeature` 驱动产生的是实体特征，而从 `GSMGeometry` 或者从 `GeometricalElement3D` 驱动产生的特征是曲面特征。但也有例外，比如边倒角特征，从 `Mechanical contextual feature` 驱动产生，但是却集成在了创成式外形设计工具中。

CAA 中从形状特征(`Mechanical shape feature`)派生而来的特征都是实体特征，而从外形设计几何 (`GSMGeom`) 和 `GeometricalElement3D` 派生而来的都是曲面特征。

实体特征只能在实体和曲面混合特征集中，比如零件几何体(`partBody`)和几何体(`Body`)，而曲面特征可以在任何特征集中比如几何特征集，有序几何特征集，零件几何体，几何体。

### 3.2 形态特征和关系特征

形态特征(`Mechanical Form Feature`)具有一个共同特点：具有形态，所以称之为形态特征，以开槽特征为例，形状有其自己的输入参数：草图和曲线，可以轻松通过沿着曲线滑动草图获得开槽特征。实际上这种特征可以用先定义形态，然后通过操作几何图形集中的草图来操作形态特征，这种特征的形态有时候也被称为脚印。

相比形态特征，关系特征(`Mechanical Contextual Feature`)没有独立的脚印，倒角，拔模等其他的修饰特征都是这类，这种特征的定义取决于输入参数和实体拓扑逻辑关系。

在零件几何体中的第一个特征不能是关系特征，因为建模时没有实体拓扑关系作为输入。

## 4 CATIA 草图定义

在 `Train1` 中就有创建草图，绘制一个圆，将圆拉伸成为一个圆柱的例子。在此将从更一般的情况入手说明草图的绘制过程。

### 4.1 创建草图

#### 4.1.1 创建轴系

##### (1) 获取轴系工厂

在一般的情况下需要获取两个工厂：轴系工厂和草图工厂，这两个工厂在结构 `container` 中，通过获取文档，获取 `container`，获取结构 `container` 得到。获取工厂的代码如以下：

```
//设置草图工厂，设置轴系工厂
```

```
CATIPrtContainer* pIPrtContOnDocument = pPrtCon;
```

```
CATIMf3DAxisSystemFactory * pIMf3DAxisSystemFactoryOnFeatCont = NULL ;
```

```
rc=pIPrtContOnDocument->QueryInterface(IID_CATIMf3DAxisSystemFactory,
```

```

        (void **) & pIMf3DAxisSystemFactoryOnFeatCont);
CATISketchFactory * pISketchFactoryOnFeatCont = NULL ;
rc=pIPrtContOnDocument->QueryInterface(IID_CATISketchFactory,
        (void **) &pISketchFactoryOnFeatCont);

```

其中 pIPrtContOnDocument 是 Part 文档的结构 container。

## (2) 创建轴系

轴系由一个点（原点），三个边（三个轴）和三个平面组成。CATIA 支持的轴系也有多种，包括右手直角轴系，欧拉角轴系，标准角轴系。

CATIMf3DAxisSystemFactory 接口可以用来创建轴系，创建轴系可以让草图的位置和坐标系更恰当。此处轴系原点在绝对坐标系的坐标为(100,0,0)，坐标轴方向可以用数学向量设置，给定了前两个轴向的方向向量 xdir 和 ydir，那么第三个方向按照右手坐标系可以自动产生。创建直角坐标系的轴系的代码为：

```

CATMathPoint Origin (100.0,.0,.0);
CATMathVectorXdir (1.0,0.0,.0);
CATMathVectorYdir (0.0,0.0,1.0);
CATIMf3DAxisSystem_var NewAxisSystem ;
rc = pIMf3DAxisSystemFactoryOnFeatCont->
CreateAxisSystem(Origin,Xdir,Ydir,NewAxisSystem);

```

此处 pIMf3DAxisSystemFactoryOnFeatCont 是获取的轴系工厂。

一旦创建了轴系，可以通过使用更新功能将轴系添加到结构树中。代码如下：

```

CATISpecObject * pSpecObjectOnAxisSystem = NULL ;
rc=NewAxisSystem->QueryInterface(IID_CATISpecObject,(void *)&pSpecObjectOnAxisSystem);
CATTrypSpecObjectOnAxisSystem->Update();
CATCatch(CATError,error)
{
cout<< error->GetNLSMessage().CastToCharPtr() <<endl;
Flush(error);
return 1 ;
}
CATEndTry

```

## 4.1.2 获取草图平面

### (1) 获取 BRep 面

轴系创建并更新后就可以使用轴系中的子元素了。获取轴系 xy 平面的代码为：

```

CATIBRepAccess_varPlaneBRep ;
rc = NewAxisSystem->GetPlaneBRepAccess(CATAxisSystemZNumber,PlaneBRep);

```

此处 CATAxisSystemZNumber 是指 xy 平面；CATAxisSystemYNumber 是指 zx 平面，

CATAxisSystemXNumber 是指 zy 平面;

同样可以获取其他轴系元素供后续使用, 如获取坐标原点采用方法 GetOriginPointBRepAccess(); 获取坐标轴采用 GetAxisBRepAccess();

## (2) 特征化 BRep

PlaneBRep 是一个拓扑概念不能直接用来建模, 具体为 CATCell, 不能直接作为输入参数构建特征、草图等。

使用 CAA 进行 CATIA 二次开发时经常需要通过 CATCell 构造对象, 但是 CATCell 无法直接用来作为输入参数建模, 这是就需要使用 CATBRepDecodeCellInBody 函数得到对应的以 CATIBRepAccess 表示的 Brep 对象, 同样 Brep 对象也是无法作为特征的输入参数, 这时就需要使用 CATIFeaturize 接口的 Featurize 函数实现 Brep 对象的特征化, 特征化以后就可以作为特征的输入参数了。

将 PlaneBRep 特征化的具体代码如下:

```
CATIFeaturize * pIFeaturizeOnPlane = NULL ;  
rc = PlaneBRep->QueryInterface(IID_CATIFeaturize,(void **) &pIFeaturizeOnPlane);  
CATISpecObject_varMFPlane = pIFeaturizeOnPlane->FeaturizeF();
```

此处 MFPlane 称为 Brep 特征。

### 4.1.3 创建草图

在 train1 中采用的是基于绝对坐标系 xy 平面创建的草图, 此处创建基于 Brep 平面特征的草图。代码如下:

```
CATISpecObject_varNewSketch = pISketchFactoryOnFeatCont->CreateSketch(MFPlane);
```

## 4.2 编辑草图

### 4.2.1 打开草图编辑

刚刚新建的草图类型为 CATISpecObject, CATISpecObject 是一个基类, 大部分 CATIA 的对象都可以用 CATISpecObject 表示, 可以用强制转化和查询接口函数 QueryInterface()两种方式从中获取具体对象。此处 NewSketch 是一个草图对象, 可以采用强制转化为 CATISketch 类型。代码如下:

```
CATISketch_varspSketch(NewSketch); if ( NULL_var == spSketch ) return 7;
```

也可以采用查询接口方法 QueryInterface();

```
CATISketch_varspSketch=NULL_var;
```

```
NewSketch->QueryInterface(IID_CATISketch,(void**)&spSketch);
```

```
if ( NULL_var == spSketch ) return 7;
```

然后打开草图编辑器，代码如下：

```
spSketch->OpenEdition();
```

#### 4.2.2 创建草图元素

2D 几何工厂可以用草图实现。然后用 2D 几何工厂绘制图形。代码如下：

```
CATI2DWFFactory_var sketch2DFactory(spSketch); // 获取 2D 工厂来创建图形
```

```
CATI2DPoint_var spPt_bottom_left, spPt_bottom_right, spPt_top_right, spPt_top_left;
```

```
CATI2DLine_var spLine1, spLine2, spLine3, spLine4;
```

```
doublept_bottom_left[2] = {10., 10.};
```

```
doublept_bottom_right[2] = {50., 10.};
```

```
doublept_top_right[2] = {50., 50.};
```

```
doublept_top_left[2] = {10., 50.};
```

```
spPt_bottom_left = sketch2DFactory->CreatePoint(pt_bottom_left);
```

```
spPt_bottom_right = sketch2DFactory->CreatePoint(pt_bottom_right);
```

```
spPt_top_right = sketch2DFactory->CreatePoint(pt_top_right);
```

```
spPt_top_left = sketch2DFactory->CreatePoint(pt_top_left);
```

```
spLine1 = sketch2DFactory->CreateLine(pt_bottom_left,pt_bottom_right);
```

```
spLine2 = sketch2DFactory->CreateLine(pt_bottom_right,pt_top_right);
```

```
spLine3 = sketch2DFactory->CreateLine(pt_top_right,pt_top_left);
```

```
spLine4 = sketch2DFactory->CreateLine(pt_top_left,pt_bottom_left);
```

```
// 将线连接起来
```

```
CATI2DCurve_var spCurve1 (spLine1);
```

```
CATI2DCurve_var spCurve2 (spLine2);
```

```
CATI2DCurve_var spCurve3 (spLine3);
```

```
CATI2DCurve_var spCurve4 (spLine4);
```

```
spCurve1->SetStartPoint(spPt_bottom_left);
```

```
spCurve1->SetEndPoint(spPt_bottom_right);
```

```
spCurve2->SetStartPoint(spPt_bottom_right);
```

```
spCurve2->SetEndPoint(spPt_top_right);
```

```
spCurve3->SetStartPoint(spPt_top_right);
```

```
spCurve3->SetEndPoint(spPt_top_left);
```

```
spCurve4->SetStartPoint(spPt_top_left);
```

```
spCurve4->SetEndPoint(spPt_bottom_left);
```

此处创建了 4 个点，然后通过点创建了 4 条线，这四条线组成了一个矩形，最后创建了线的首尾链接性，不再是单独的 4 条线。

当然，可以用 CATI2DWFFactory 接口的方法创建圆，椭圆，等其他草图图形。

创建圆的方法 CreateCircle();创建圆弧的方法 CreateArc();创建倒角方法 CreateChamfer();创建椭圆 CreateEllipse();创建样条曲线 CreateSplineCurve();从几何图形投影 ProjectGeometry();等。

### 4.2.3 创建草图约束

同样的，2D 约束工厂可以用草图实现。然后用 2D 约束工厂添加图形约束。代码如下：

```
CATI2DConstraintFactory_var spConstraint2DFactory(spSketch);
spConstraint2DFactory->CreateConstraint( spLine1, NULL, NULL, NULL, NULL, NULL,
NULL,
Cst2DType_Horizontal, 0, 0 );//水平约束
spConstraint2DFactory->CreateConstraint( spLine2, NULL, NULL, NULL, NULL, NULL,
NULL,
Cst2DType_Vertical, 0, 0 );//垂直约束
spConstraint2DFactory->CreateConstraint( spLine3, NULL, NULL, NULL, NULL, NULL,
NULL,
Cst2DType_Horizontal, 0, 0 );//水平约束
spConstraint2DFactory->CreateConstraint( spLine4, NULL, NULL, NULL, NULL, NULL,
NULL,
Cst2DType_Vertical, 0, 0 );//垂直约束
spConstraint2DFactory->CreateConstraint( spLine2, NULL, NULL, NULL, NULL, NULL,
NULL,
Cst2DType_Length, 0, 0 );//长度约束
spConstraint2DFactory->CreateConstraint( spLine2, NULL, spLine4, NULL, NULL, NULL,
NULL,
Cst2DType_Distance, 0, 0 );//距离约束
```

此处创建了 2 个水平约束，2 个垂直约束，一个距离约束，一个长度约束。

### 4.2.4 创建草图圆角

采用 CATI2DTopologicalOperators 接口可以进行倒圆角等修饰草图的操作，代码如下：

```
double radius = 10.;
double pt_center[2] = {70., 40.};
CATI2DCurve_var spCurve5 = sketch2DFactory->CreateCorner(spCurve3, spCurve4, pt_center,
&radius);
CATI2DTopologicalOperators_var spOperateur = spSketch;
spOperateur->InsertCorner(spCurve5, spLine3, 1, spLine4, 1);
spConstraint2DFactory->CreateConstraint( spLine3, NULL, spCurve5, NULL, NULL, NULL,
```

NULL,

Cst2DType\_Tangent, 0, 0);

spConstraint2DFactory->CreateConstraint( spCurve5, NULL, spLine4, NULL, NULL, NULL,  
NULL,

Cst2DType\_Tangent, 0, 0);

spConstraint2DFactory->CreateConstraint( spCurve5, NULL, NULL, NULL, NULL, NULL,  
NULL,

Cst2DType\_Radius, 0, 1);

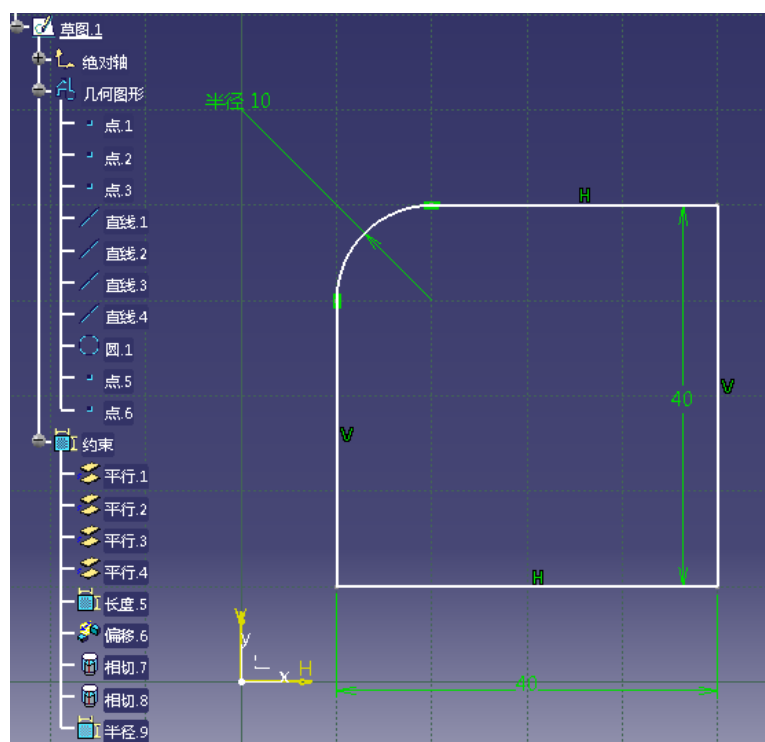
此处首先用 2D 工厂创建了一个倒角，然后用 CATI2DTopologicalOperators 接口将倒角插入到草图中，再添加了两个相切约束，一个半径约束。

#### 4.2.5 关闭草图编辑

spSketch->CloseEdition();

完成草图编辑后关闭草图编辑器。

完成的草图如图所示：



## 5 实体建模

### 5.1 建模工厂（Factory）

对机械设计而言，最有效果的步奏就是实体建模步骤，在 CAD/CAM 系统实体模型代表了真实的材料实体，而在曲面造型中就没有材料的概念，因为去曲面没有厚度。

在实体建模的过程中，设计者需要设计形状，尺寸和基本组件的位置，所以设计者们经常

使用创建、修改、修饰实体的方法。这 3 种方法可以归类为：

- 1.创建实体的方法。比如凸台，旋转体，肋，厚曲面，多截面等。
- 2.两种实体结合而产生另外的实体。凸台，旋转体，肋，厚曲面，开槽，凹槽等。
- 3.采用修饰的手段获得实体。比如倒角，拔模，箱体，缝合，分割等。

前两种属于形态特征，三属于关系特征。

创建基于草图的特征一般过程为设置建模工厂，创建模型，修改模型三步完成。代码如下所示：

```
CATMathDirectiondirZ(0., 0., 1.); //凸台拉伸的方向
doublefirstLimit = 20.;
doublesecondLimit = 0.;
// 获取机械设计工厂
CATIPrtFactory_varspPrtFactOnPrtCont(piPrtCont);
piPrtCont->Release();
CATISpecObject_varspSpecObj = spPrtFactOnPrtCont->CreatePad(spSketch); //创建凸台
CATIPad_varspPadOnSpecObj(spSpecObj);
//修改凸台参数
spPadOnSpecObj->ModifyDirection(dirZ);
spPadOnSpecObj->ModifyEndType(catOffsetLimit);
spPadOnSpecObj->ModifyEndOffset(firstLimit);
spPadOnSpecObj->ModifyStartType(catOffsetLimit);
spPadOnSpecObj->ModifyStartOffset(secondLimit);
spSpecObj->Update(); //将凸台插入到视图中
```

上述代码中出现的 piPrtCont 是结构 container。现在，基于草图的拉伸凸台建模完成。ModifyDirection 的默认方向是草图的法向方向。

在建模工厂 CATIPrtFactory 中可以找的很多建模方法，不管是形态特征还是关系特征都可以找到创建方法：

创建槽特征 CreateGroove();创建拔模特征 CreateLoft();创建镜像特征 CreateMirror();创建凹槽特征 CreatePocket();创建肋特征 CreateRib();创建旋转体特征 CreateRotate();创建缝合体特征 CreateSewing();创建抽壳特征 CreateShell();开槽特征 CreateSlot();打孔特征 CreateHole();加厚特征 CreateThickness();

## 5.2 创建几何体特征集

### 5.2.1 获取 CATIMechanicalRootFactory 接口

在一个 part 文档中只有在有序几何图形集合特征和 Part 特征下才可以创建几何体。

用 CATIMechanicalRootFactory 接口的方法 CreatePRTTool 可以创建一个几何体特征。首先为设置 MechanicalRootFactory 代码:

```
CATIPrtContainer* ipPrtCont = pPrtCon;//结构 Container
CATIMechanicalRootFactory *pSetToolFactory=NULL;
rc=ipPrtCont->QueryInterface(IID_CATIMechanicalRootFactory,(void **)&pSetToolFactory);
if(FAILED(rc)||pSetToolFactory ==NULL) return;
CATIPrtPart_var spPart=ipPrtCont->GetPart();
if(spPart==NULL_var) return ;
```

### 5.2.2 创建几何体

```
CATISpecObject_var spPRTTool
rc= pSetToolFactory->CreatePRTTool(“复制的几何体”,spPart,spPRTTool);
```

## 5.3 特征 (Feature) 的复制

可以采用两种方式完成特征复制，分别采用 CATMmrInterPartCopy 接口和 CATICutAndPastable 接口完成复制。

### 5.3.1 复制方式一

CATMmrInterPartCopy 接口:

在零件环境中复制一个特征一般需要以下五步完成。将 Pad.1 复制到零件几何体中，新特征为 Pad.2。

1.创建 CATMmrInterPartCopy 类的实例。代码如下:

```
CATISpecObject_var SourceToCopy = spSpecObj;
CATISpecObject_var Target = spPRTTool;
CATMmrInterPartCopy * ptCATMmrInterPartCopy =
newCATMmrInterPartCopy (SourceToCopy,spPartBodyset) ;
```

其中 spSpecObj 和 spPartBodyset 是 pad.1 和 “零件几何体” 的智能指针。

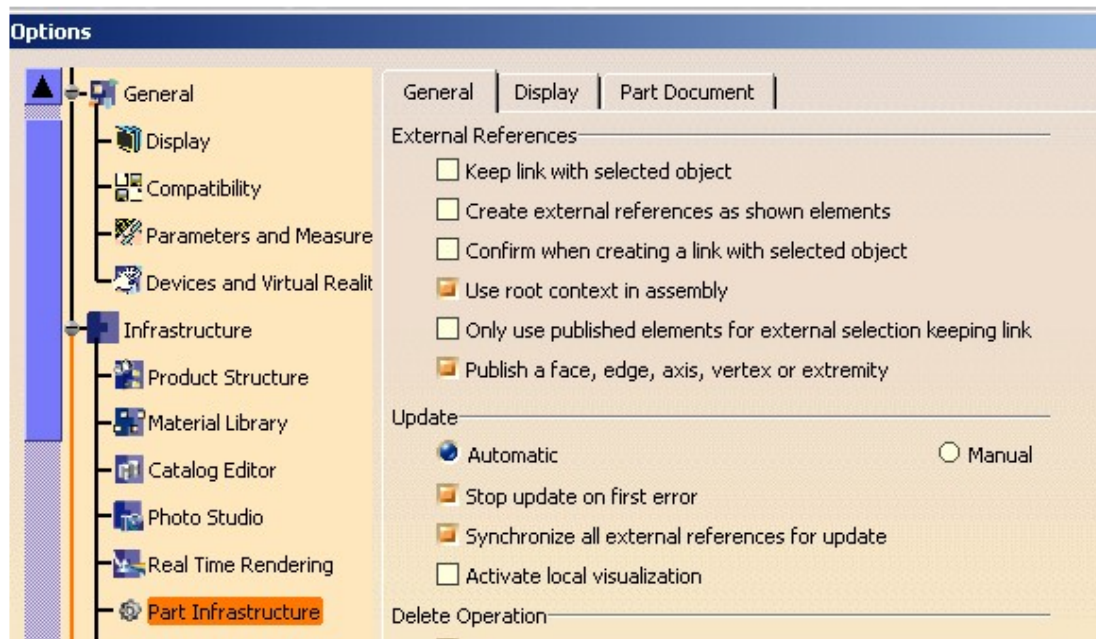
2.用 SetLinkMode 方法设置复制选项。代码如下:

```
ptCATMmrInterPartCopy ->SetLinkMode(CopyWithLink);
```

其中 CopyWithLink 是一个 CATBoolean 布尔值。如果值是 1, CopyWithLink 的值是 TRUE,



否则值是 FALSE，如果不用 CopyWithLink 方法，那么复制选项取决于工具选项中的 keep link with selected object 选项选择情况，如图所示：



3.采用 run()方法执行复制任务。代码如下：

```
ErrorMsg = "" ;  
rc = ptCATMmrInterPartCopy ->Run(&ErrorMsg);
```

4.采用 GetResult 方法获取复制结果。代码如下：

```
CATISpecObject_var Result;  
rc = ptCATMmrInterPartCopy ->GetResult(Result);
```

5.将复制结果添加到视图中

```
Result->Update();
```

### 5.3.2 复制方式二

CATICutAndPastable 接口：

CATICutAndPastable 接口不仅可以用来在同一个文件内进行操作，还可以在不同文档的 Container 之间操作。它可以进行删除、粘贴，复制操作。

CATICutAndPastable::Remove()可以进行删除和剪切操作；

CATICutAndPastable:: Paste ()可以进行粘贴操作；

复制 pad.1 特征的代码如下所示：

```
//复制凸台特征用接口 CATICutAndPastable  
if (spSpecObj == NULL_var) return;
```

```

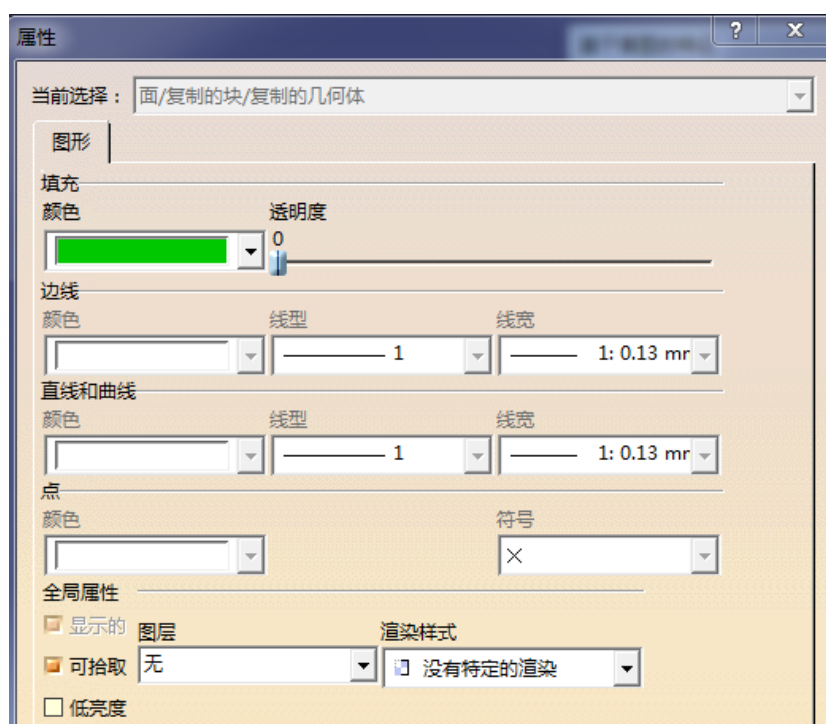
//获取特征容器
CATIContainer_var spContainer = spSpecObj->GetFeatContainer();
if (spContainer == NULL_var) return;
//由容器查找到剪切粘贴接口
CATICutAndPatable *pCutAndPatable = NULL;
rc = spContainer->QueryInterface(IID_CATICutAndPatable, (void
**) &pCutAndPatable);
if (FAILED(rc) || pCutAndPatable==NULL) return ;
//添加删除元素
CATLISTV(CATBaseUnknown_var) listOfUnknown;
listOfUnknown.Append(spSpecObj);
pCutAndPatable->Paste(listOfUnknown);

```

## 5.4 属性（Property）修改

### （1）设置颜色属性

在 CATIA 中可以通过接口 CATIVisProperties 更改特征的图形属性。



具体的更改颜色步骤为：

1. 采用 CATIMfGeometryAccess 接口获取每个特征的拓扑表示结构。

这里可以通过 GetCells 方法和 GetBreps 两种方法获取拓扑结构。这两种方法其实是从不同的层面反应同一个事物，拓扑层和视图层。视图层管理机械世界和拓扑世界联通的对象。视图由节点组成（有时也称为选中的对象），节点与拓扑世界相关联。

GetCells 提供了一个直接访问拓扑世界单元的方法，而 GetBreps 获取的是与之相对的视图

节点（或者称为选中的对象）。

GetBreps 有多个重载，可以获取对象的所有节点，也可以在某些情形下只获取个别节点。

代码如下所示：

```
CATIMfGeometryAccess *pPadAsGeomAccess = NULL;//定义拓扑几何元素接口
```

```
rc= Result->QueryInterface(IID_CATIMfGeometryAccess, (void**)&pPadAsGeomAccess) ;//
```

获取拓扑几何元素接口

```
if (FAILED(rc) ) return ;
```

```
CATLISTV(CATBaseUnknown_var) PadBREps;
```

```
pPadAsGeomAccess->GetBREps(PadBREps);//获取所有的拓扑元素 BRep 面
```

## 2. 设置每个拓扑元素的颜色

对于每个节点，我们可以改变它们的图形属性，这些属性包括颜色，线宽，透明度，线型，图层等图形属性。具体的控制图形属性的接口为 CATVisProperties。CATVisProperties 本身没有任何函数可以进行操作，它的所有函数都是继承自接口 CATVisPropertiesAbstract。对于 CATVisPropertiesAbstract::SetPropertiesAtt(CATVisPropertiesValues&iValues,

```
CATVisPropertyTypeiPropertyType,
```

```
CATVisGeomTypeiGeomType,
```

```
unsignedintiPropertyNumber,
```

```
intifromSetProperties
```

```
)
```

其中 CATVisPropertiesValues 可以用 new 创建得到，然后用 CATVisPropertiesValues 的方法修改具体内容；

CATVisPropertyType 是一个枚举，其合法的变量都有 CATVPColor//颜色 CATVPOpacity, CATVPSymbol, CATVPLineType,// 线型 CATVPWidth,// 线宽 CATVPIheritance, CATVPLayer,// 图层 CATVPShow,// 显示隐藏 CATVPPick, CATVPLowInt, CATVPRenderingStyle,

CATVisGeomType 也是枚举类型，合法的变量有 CATVPGlobalType, CATVPMesh, CATVPEdge, CATVPLine, CATVPPoint, CATVPAsm

在这里我们用 new 新建了一个 CATVisPropertiesValues 类，并用 SetColor 方法设置了颜色，最后针对 CATVPMesh 类型的几何拓扑元素修改了颜色。

具体代码如下：

```
//替 BRep 面更改颜色
```

```

for(int currentBRep=1; currentBRep<=PadBReps.Size(); currentBRep++)
{
CATIVisProperties *pPadBrepAsGraphics = 0;
const CATBaseUnknown_var&currentPadBRep = PadBReps[currentBRep];
if (NULL_var != currentPadBRep)
rc
currentPadBRep->QueryInterface(IID_CATIVisProperties,(void**)&pPadBrepAsGraphics) ;// 获取
面的属性接口
if ( SUCCEEDED(rc) )
{
int R,G,B;
R=0;
G=200;
B=0;
CATVisPropertiesValues color;//定义颜色属性
color.SetColor(R, G, B); // green
pPadBrepAsGraphics->SetPropertiesAtt(color, CATVPColor, CATVPMesh);//设置颜色属性
pPadBrepAsGraphics->Release();
pPadBrepAsGraphics = NULL ;
}
}

```

## （2）设置显示和隐藏属性

根据上节所述，可以设置草图为显示状态。代码如下所示：

```

CATIVisProperties *pVisProperties = NULL;
rc = pSpecObjectOnSketch->QueryInterface(IID_CATIVisProperties,(void**)&pVisProperties);
if( SUCCEEDED(rc) && pVisProperties!=NULL)
{
CATVisPropertiesValues MyProp;
pVisProperties->GetPropertiesAtt(MyProp, CATVPShow);
MyProp.SetShowAttr(CATShowAttr);
pVisProperties -> SetPropertiesAtt(MyProp,CATVPShow ,CATVPGlobalType);//
CATVPShow CATVPGlobalType
pVisProperties -> Release();
pVisProperties = NULL ;
}
pSpecObjectOnSketch->Update();

```

## （3）更改名称

对于大部分 SpecObject 对象，可以通过 CATIAlias 接口进行修改名称操作。具体代码如下所示：

```

//重命名特征
if ( NULL_var == Result ) return;
CATIAlias* pAlias = NULL;

```

```

rc = Result->QueryInterface(IID_CATIAAlias, (void**)&pAlias); //获取名称接口
if ( SUCCEEDED(rc) )
{
    pAlias->SetAlias("复制的块"); //设置名称
    pAlias->Release();
    pAlias = NULL;
}

```

#### (4) 更改Part特征名称

采用 **Automation** 编程中的内容更改 **Part** 特征名称。

//更改Part特征名称为NewPart

```

CATIAPartDocument* pCATIAPrtDoc = NULL;
CATIAProductDocument * pCATIAPrdtDoc = NULL;
rc = pDoc->QueryInterface(IID_CATIAPartDocument, (void**)&pCATIAPrtDoc);
rc = pDoc->QueryInterface( IID_CATIAProductDocument, (void**)&pCATIAPrdtDoc );
if ( NULL == pCATIAPrdtDoc && NULL == pCATIAPrtDoc ) return;
CATIAProduct* pCATIAPrtRoot = NULL;
if ( NULL != pCATIAPrtDoc )
{
    pCATIAPrtDoc->get_Product(pCATIAPrtRoot);
}
elseif ( NULL != pCATIAPrdtDoc )
{
    pCATIAPrdtDoc->get_Product(pCATIAPrtRoot);
}
if (NULL == pCATIAPrtRoot )return;
CATUnicodeString PartNumber = "NewPart";
CATBSTR bstr;
PartNumber.ConvertToBSTR(&bstr);
pCATIAPrtRoot->put_PartNumber(bstr);
pCATIAPrtRoot->Update();
CATBSTR newBstr;
pCATIAPrtRoot->get_PartNumber(newBstr);
CATUnicodeString s;
s.BuildFromBSTR(newBstr);
if ( NULL != pCATIAPrtDoc )
{
    pCATIAPrtDoc->Release();    pCATIAPrtDoc = NULL;
}
if ( NULL != pCATIAPrdtDoc )
{
    pCATIAPrdtDoc->Release();    pCATIAPrdtDoc = NULL;
}

```

## 6 文件管理对话框实现

### 6.1 链表的使用

链表常用来存储多个同类型的对象，在 C++ 中就有多种方式可以存储同类对象，比如数组，vector；list 等；在 CAA 开发中也定义了几种不同形式的链表。它们分别是 CATListOfXXXX；CATLISTV(XXXX)；CATLISTP(XXXX)；比如 CATListOfCATUnicodeString；CATLISTV(CATBaseUnknown\_var)；CATLISTP(CATBaseUnknown)；

这里以 CATListOfCATUnicodeString 为例说明链表使用。它是一个用来存储多个 CATUnicodeString 类型字符串的链表。

链表声明：CATListOfCATUnicodeString Strs；

给链表最后添加一个元素采用 Append 方法：

Strs.Append(“sss”);

链表中插入一个元素可以用 InsertAfter；InsertBefore：

Strs.InsertAfter (“sss”);

删除链表中的某个元素用方法 RemovePosition，RemoveValue，Remove 等；

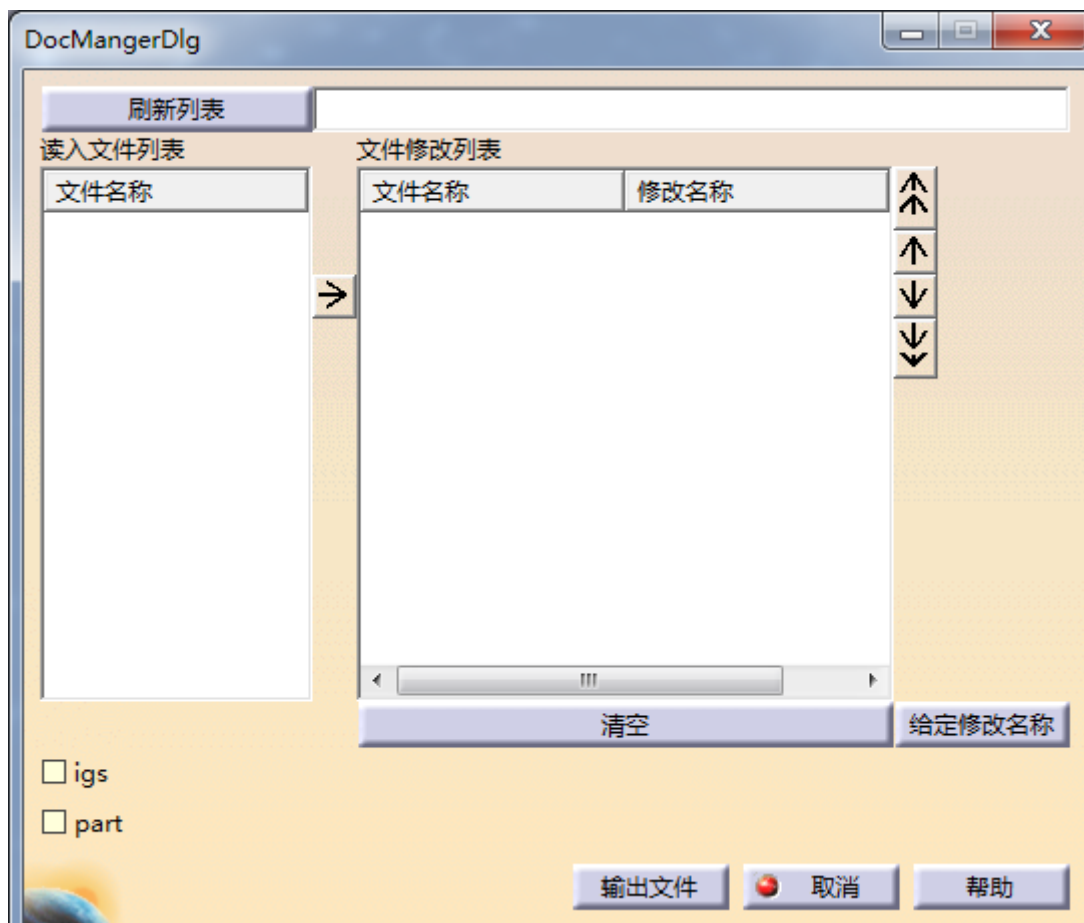
删除链表中的全部元素用方法 RemoveAll；

获取链表长度 Size；

获取链表中的值用操作符 “[]”；

一般而言链表的其实标号为 1，而不是 0；

## 6.2 新建对话框



新建对话框，修改对话框属性，添加控件事件，保存对话框。

## 6.3 对话框功能实现

### 6.3.1 刷新按钮

```
// Callback _PushButtonOpenFolder//打开文件夹
void DocMangerDlg::OnPushButtonOpenFolderPushBActivateNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
{
    m_FilesName.RemoveAll();
    _MultiList001->ClearLine();
    m_InPath= _EditorInPath->GetText(); //从Editor中获取文件夹路径
    CATUnicodeString iFromFolderPath = m_InPath;
    CATDirectory oDir;
    CATLibStatus
libstatus=CATOpenDirectory( iFromFolderPath.ConvertToChar(), &oDir) ; //打开文件夹
    if(CATLibSuccess ==libstatus)
    {
```

```

int oEndSign=0;
while(oEndSign!=1)
{
    CATDirectoryEntry oEntry;
    libstatus= CATReadDirectory( &oDir, &oEntry, &oEndSign);//逐个读取文件
    if(CATLibSuccess ==libstatus)
    {
        CATUnicodeString FileName(oEntry.name);
        int check = FileName.SearchSubString(".CATPart");//获取CATPart文件
        if (FileName == "."||FileName==".."||FileName=="")continue;
        if(check != -1)//当前的FileName是CATPart文件
        {
            m_FilesName.Append(FileName); //CATPart文件放到链表最后
        }
    }
    CATCloseDirectory(&oDir); //文件夹关闭
}
//添加到multilist
for(int i=1;i<=m_FilesName.Size();i++)
    _MultiList001->SetColumnItem(0,m_FilesName[i]);

// Add your code here
}

```

### 6.3.2 向右 按钮

```

// Callback on _PushButtonRight//向右
void DocMangerDlg::OnPushButtonRightPushBActivateNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
{
    for(int i=1;i<=m_FilesName.Size();i++)
    {
        int check =0;
        for(int j=1;j<=m_FilesIn004.Size();j++)
        {
            CATUnicodeString Exist;
            Exist = m_FilesIn004[j];
            if(Exist==m_FilesName[i])
            {
                check = 1;
            }
        }
    }
}

```



```

        if(check == 0)
            m_FilesIn004.Append(m_FilesName[i]);
    }
    refresh004();
}

```

### 6.3.3 向上 按钮

```

// Callback on _PushButtonUPUP//向上
void DocMangerDlg::OnPushButtonUPUPPushBActivateNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
{
    if(_MultiList004->GetLineCount()==0) return;
    CATUnicodeString tempStr;
    int select[1]={0};
    _MultiList004->GetSelect (select,1);
    tempStr = m_FilesIn004[select[0]+1];
    m_FilesIn004.RemoveValue(tempStr);
    m_FilesIn004.InsertBefore(1,tempStr);
    refresh004();
    int selectafter[1]={0};
    _MultiList004->SetSelect(selectafter,1);
}

```

上述自定义函数 refresh004(); 功能为刷新 MultiList004; 实现代码为:

//刷新列表

```

void DocMangerDlg::refresh004()
{
    _MultiList004->ClearLine(); //清空列表
    for(int i=0;i<m_FilesIn004.Size();i++)
        _MultiList004->SetColumnItem(0,m_FilesIn004[i+1]);
}

```

### 6.3.4 输出文件按钮

```

// Callback on _DocMangerDlg//输出文件
void DocMangerDlg::OnDocMangerDlgDiaOKNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
{
    //新建一个文件夹
    if(m_NewFileName.Size()==0) return;
    CATUnicodeString Suffex_1=".CATPart";
    CATUnicodeString Suffex_2=".igs";
    CATLibStatus libstatus;
    HRESULT rc;
    for(int i=1;i<=m_FilesIn004.Size();i++)

```

```

{
    CATUnicodeString CurrentFileName;
    CurrentFileName = m_FilesIn004[i];
    CATDocument *pDoc = NULL;
    rc = CATDocumentServices::OpenDocument(m_InPath+"\\ "+CurrentFileName, pDoc);
    if (FAILED(rc) ) return;
    if(pDoc==NULL) return;
    //保存CATPart
    if(_CheckButtonPart->GetState()==CATDlgCheck)
    {
        CATUnicodeString tempPathPart = m_InPath+"\\CATPartDir";
        //新建一个CATPartDir文件夹
        libstatus=CATCreateDirectory(tempPathPart.ConvertToChar());
        if(CATLibSuccess !=libstatus) return;
        rc = CATDocumentServices::SaveAsNew (*pDoc,
tempPathPart+"\\ "+m_NewFileName[i]+Suffex_1);
    }
    //保存igs
    if(_CheckButtonIGS->GetState()==CATDlgCheck)
    {
        CATUnicodeString tempPathIgs = m_InPath+"\\igsDir";
        //新建一个igsDir文件夹
        libstatus=CATCreateDirectory(tempPathIgs.ConvertToChar());
        if(CATLibSuccess !=libstatus) return;
        rc = CATDocumentServices::SaveAsNew (*pDoc,
tempPathIgs+"\\ "+m_NewFileName[i]+Suffex_2);
    }
}
}

```

## 7 零件实体建模实现

给按钮创建 Part 文件添加响应代码。

//创建新的Part文件

```

CATDocument * pDoc = NULL;
HRESULT rc = CATDocumentServices::New("Part", pDoc);
if ( FAILED(rc) || NULL == pDoc ) return ;

```

//新Part文件初始化

```

CATInit * pInit = NULL;
rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInit);
if (FAILED(rc) || NULL == pInit ) return ;
pInit->Init(TRUE);

```

//获取结构container

```

CATContainerOfDocument_var spNewDocContainer = pDoc;

```

```

CATIContainer *piNewContainer = NULL;
rc = spNewDocContainer->GetSpecContainer(piNewContainer);
if (FAILED(rc) || piNewContainer == NULL) return ;

CATIPrtContainer* pPrtCon = NULL;
rc= piNewContainer->QueryInterface( IID_CATIPrtContainer, (void **)&pPrtCon);
if (FAILED(rc) || pPrtCon == NULL) return ;
//设置草图工厂，设置轴系工厂
CATIPrtContainer* pIPrtContOnDocument = pPrtCon;
CATIMf3DAxisSystemFactory * pIMf3DAxisSystemFactoryOnFeatCont = NULL ;
rc=pIPrtContOnDocument->QueryInterface(IID_CATIMf3DAxisSystemFactory,
                                         (void **) & pIMf3DAxisSystemFactoryOnFeatCont);
CATISketchFactory * pISketchFactoryOnFeatCont = NULL ;
rc=pIPrtContOnDocument->QueryInterface(IID_CATISketchFactory,
                                         (void **) & pISketchFactoryOnFeatCont);
//创建轴系
CATMathPoint Origin (100.0,.0,.0);
CATMathVector Xdir (1.0,0.0,.0);
CATMathVector Ydir (0.0,0.0,1.0);
CATIMf3DAxisSystem_var NewAxisSystem ;
rc =
pIMf3DAxisSystemFactoryOnFeatCont->CreateAxisSystem(Origin,Xdir,Ydir,NewAxisSystem);
//添加轴系到视图中
CATISpecObject * pSpecObjectOnAxisSystem = NULL ;
rc=NewAxisSystem->QueryInterface(IID_CATISpecObject, (void
***)&pSpecObjectOnAxisSystem);
CATTry pSpecObjectOnAxisSystem->Update();
CATCatch(CATError, error)
{
    Flush(error);
    return ;
}
CATEndTry
//获取轴系的xy平面
CATIBRepAccess_var PlaneBRep ;
rc = NewAxisSystem->GetPlaneBRepAccess(CATAxisSystemZNumber, PlaneBRep);
//Brep对象特征化
CATIFeaturize * pIFeaturizeOnPlane = NULL ;
rc = PlaneBRep->QueryInterface(IID_CATIFeaturize, (void **)
&pIFeaturizeOnPlane);
CATISpecObject_var MFPlane = pIFeaturizeOnPlane->FeaturizeF();
//创建草图
CATISpecObject_var NewSketch =

```

```

pISketchFactoryOnFeatCont->CreateSketch(MFPlane);
    //启动草图编辑器
    CATISketch_var spSketch(NewSketch); if ( NULL_var == spSketch ) return ;
    spSketch->OpenEdition();
    //开始草图绘图
    CATI2DWFFactory_var sketch2DFactory(spSketch); // 获取2D工厂来创建图形
    CATI2DPoint_var spPt_bottom_left, spPt_bottom_right, spPt_top_right,
    spPt_top_left;
    CATI2DLine_var spLine1, spLine2, spLine3, spLine4;
    double pt_bottom_left[2] = {10., 10.};
    double pt_bottom_right[2] = {50., 10.};
    double pt_top_right[2] = {50., 50.};
    double pt_top_left[2] = {10., 50.};
    spPt_bottom_left = sketch2DFactory->CreatePoint(pt_bottom_left);
    spPt_bottom_right = sketch2DFactory->CreatePoint(pt_bottom_right);
    spPt_top_right = sketch2DFactory->CreatePoint(pt_top_right);
    spPt_top_left = sketch2DFactory->CreatePoint(pt_top_left);
    spLine1 = sketch2DFactory->CreateLine(pt_bottom_left, pt_bottom_right);
    spLine2 = sketch2DFactory->CreateLine(pt_bottom_right, pt_top_right);
    spLine3 = sketch2DFactory->CreateLine(pt_top_right, pt_top_left);
    spLine4 = sketch2DFactory->CreateLine(pt_top_left, pt_bottom_left);
    // 将线连接起来
    CATI2DCurve_var spCurve1 (spLine1);
    CATI2DCurve_var spCurve2 (spLine2);
    CATI2DCurve_var spCurve3 (spLine3);
    CATI2DCurve_var spCurve4 (spLine4);
    spCurve1->SetStartPoint(spPt_bottom_left);
    spCurve1->SetEndPoint(spPt_bottom_right);
    spCurve2->SetStartPoint(spPt_bottom_right);
    spCurve2->SetEndPoint(spPt_top_right);
    spCurve3->SetStartPoint(spPt_top_right);
    spCurve3->SetEndPoint(spPt_top_left);
    spCurve4->SetStartPoint(spPt_top_left);
    spCurve4->SetEndPoint(spPt_bottom_left);
    //设置草图约束
    CATI2DConstraintFactory_var spConstraint2DFactory(spSketch);
    spConstraint2DFactory->CreateConstraint( spLine1, NULL, NULL, NULL, NULL, NULL,
    NULL,
                                     Cst2DType_Horizontal, 0, 0 );//水平约束
    spConstraint2DFactory->CreateConstraint( spLine2, NULL, NULL, NULL, NULL, NULL,
    NULL,
                                     Cst2DType_Vertical, 0, 0 );//垂直约束
    spConstraint2DFactory->CreateConstraint( spLine3, NULL, NULL, NULL, NULL, NULL,
    NULL,

```

```

        Cst2DType_Horizontal, 0, 0 ); //水平约束
    spConstraint2DFactory->CreateConstraint( spLine4, NULL, NULL, NULL, NULL, NULL,
    NULL,

        Cst2DType_Vertical, 0, 0 ); //垂直约束
    spConstraint2DFactory->CreateConstraint( spLine2, NULL, NULL, NULL, NULL, NULL,
    NULL,

        Cst2DType_Length, 0, 0 ); //长度约束
    spConstraint2DFactory->CreateConstraint( spLine2, NULL, spLine4, NULL, NULL,
    NULL, NULL,

        Cst2DType_Distance, 0, 0 ); //距离约束

    //创建草图修饰——圆倒角
    double radius = 10.;
    double pt_center[2] = {70., 40.};
    CATI2DCurve_var spCurve5 = sketch2DFactory->CreateCorner(spCurve3, spCurve4,
    pt_center, &radius);
    CATI2DTopologicalOperators_var spOperateur = spSketch;
    spOperateur->InsertCorner(spCurve5, spLine3, 1, spLine4, 1);
    spConstraint2DFactory->CreateConstraint( spLine3, NULL, spCurve5, NULL, NULL,
    NULL, NULL,

        Cst2DType_Tangent, 0, 0);
    spConstraint2DFactory->CreateConstraint( spCurve5, NULL, spLine4, NULL, NULL,
    NULL, NULL,

        Cst2DType_Tangent, 0, 0);
    spConstraint2DFactory->CreateConstraint( spCurve5, NULL, NULL, NULL, NULL, NULL,
    NULL,

        Cst2DType_Radius, 0, 1);

    //关闭草图编辑器
    spSketch->CloseEdition();
    //添加草图到视图中
    CATISpecObject * pSpecObjectOnSketch = NULL ;
    rc=spSketch->QueryInterface(IID_CATISpecObject, (void **)&pSpecObjectOnSketch);
    CATTry pSpecObjectOnSketch->Update();
    CATCatch(CATError, error)
    {
        Flush(error);
        return ;
    }
    CATEndTry
    //创建凸台特征
    CATMathDirection dirZ(0., 0., 1.); //凸台拉伸的方向
    double firstLimit = 20.;
    double secondLimit = 0.;
    // 获取机械设计工厂
    CATIPrtFactory_var spPrtFactOnPrtCont (pPrtCon);

```

```

    CATISpecObject_var spSpecObj = spPrtFactOnPrtCont->CreatePad(spSketch); //创建
凸台
    CATIPad_var spPadOnSpecObj(spSpecObj);
    //修改凸台参数
    //spPadOnSpecObj->ModifyDirection(dirZ);
    spPadOnSpecObj->ModifyEndType(catOffsetLimit);
    spPadOnSpecObj->ModifyEndOffset(firstLimit);
    spPadOnSpecObj->ModifyStartType(catOffsetLimit);
    spPadOnSpecObj->ModifyStartOffset(secondLimit);
    CATTry spSpecObj->Update(); //将凸台插入到视图中
    CATCatch(CATError, error)
    {
        Flush(error);
        return ;
    }
    CATEndTry
//创建几何特征集
    CATIPrtContainer* ipPrtCont = pPrtCon; //结构Container
    CATIMechanicalRootFactory *pSetToolFactory=NULL; //设置MechanicalRoot工厂
    rc=ipPrtCont->QueryInterface(IID_CATIMechanicalRootFactory, (void **)&
pSetToolFactory);
    if(FAILED(rc) || pSetToolFactory ==NULL) return;
    CATIPrtPart_var spPart=ipPrtCont->GetPart();
    if(spPart==NULL_var) return;
    CATISpecObject_var spPRTTool;
    spPRTTool= pSetToolFactory->CreatePRTTool("复制的几何体", spPart);
//复制凸台特征用接口 CATICutAndPastable
    if (spSpecObj == NULL_var) return;
    //获取特征容器
    CATIContainer_var spContainer = spSpecObj->GetFeatContainer();
    if (spContainer == NULL_var) return;
    //由容器查找到剪切粘贴接口
    CATICutAndPastable *pCutAndPastable = NULL;
    rc = spContainer->QueryInterface(IID_CATICutAndPastable, (void
**)&pCutAndPastable);
    if (FAILED(rc) || pCutAndPastable==NULL) return;
    //添加删除元素
    CATLISTV(CATBaseUnknown_var) listOfUnknown;
    listOfUnknown.Append(spSpecObj);
    pCutAndPastable->Paste(listOfUnknown);
//复制凸台特征到 复制的几何体用接口CATMmrInterPartCopy
    CATISpecObject_var SourceToCopy = spSpecObj;
    CATISpecObject_var Target = spPRTTool;
    //创建CATMmrInterPartCopy类的实例

```

```

CATMmrInterPartCopy * ptCATMmrInterPartCopy = new CATMmrInterPartCopy
(SourceToCopy, Target) ;
//用SetLinkMode 方法设置复制选项
ptCATMmrInterPartCopy ->SetLinkMode(FALSE);
//采用run()方法执行复制任务
CATUnicodeString ErrorMsg = "" ;
rc = ptCATMmrInterPartCopy ->Run(&ErrorMsg);
//采用GetResult方法获取复制结果。代码如下：
CATISpecObject_var Result;
rc = ptCATMmrInterPartCopy ->GetResult(Result);
//复制结果添加到视图
Result->Update();
//更改实体的颜色属性
CATIMfGeometryAccess *pPadAsGeomAccess = NULL;//定义拓扑几何元素接口
rc= Result->QueryInterface(IID_CATIMfGeometryAccess,
(void**)&pPadAsGeomAccess) ;//获取拓扑几何元素接口
if (FAILED(rc) ) return ;
CATLISTV(CATBaseUnknown_var) PadBReps;
pPadAsGeomAccess->GetBReps(PadBReps);//获取所有的拓扑元素 BRep面
// 6-3 替BRep面更改颜色
for(int currentBRep=1; currentBRep<=PadBReps.Size(); currentBRep++)
{
    CATIVisProperties *pPadBrepAsGraphics = 0;
const CATBaseUnknown_var& currentPadBRep = PadBReps[currentBRep];
if (NULL_var != currentPadBRep)
    rc =
currentPadBRep->QueryInterface(IID_CATIVisProperties, (void**)&pPadBrepAsGraphics)
; //获取面的属性接口
if ( SUCCEEDED(rc) )
{
    int R, G, B;
    R=0;
    G=200;
    B=0;
    CATVisPropertiesValues color;//定义颜色属性
    color.SetColor(R, G, B); // green
    pPadBrepAsGraphics->SetPropertiesAtt(color, CATVPColor, CATVPMesh);//设置颜色属性
    pPadBrepAsGraphics->Release();
    pPadBrepAsGraphics = NULL ;
}
}
//重命名特征
if ( NULL_var == Result ) return;

```

```

CATIAlias* pAlias = NULL;
rc = Result->QueryInterface(IID_CATIAlias, (void**)&pAlias); //获取名称接口
if ( SUCCEEDED(rc) )
{
    pAlias->SetAlias("复制的块"); //设置名称
    pAlias->Release();
    pAlias = NULL;
}
//更改Part特征名称为NewPart
CATIArtDocument* pCATIArtDoc = NULL;
CATIArtProductDocument * pCATIArtPrdtDoc = NULL;
rc = pDoc->QueryInterface(IID_CATIArtDocument, (void**)&pCATIArtDoc);
rc = pDoc->QueryInterface( IID_CATIArtProductDocument, (void**)&pCATIArtPrdtDoc );
if ( NULL == pCATIArtPrdtDoc && NULL == pCATIArtDoc ) return;
CATIArtProduct* pCATIArtPrtRoot = NULL;
if ( NULL != pCATIArtDoc )
{
    pCATIArtDoc->get_Product(pCATIArtPrtRoot);
}
elseif ( NULL != pCATIArtPrdtDoc )
{
    pCATIArtPrdtDoc->get_Product(pCATIArtPrtRoot);
}
if (NULL == pCATIArtPrtRoot )return;
CATUnicodeString PartNumber = "NewPart";
CATBSTR bstr;
PartNumber.ConvertToBSTR(&bstr);
pCATIArtPrtRoot->put_PartNumber(bstr);
pCATIArtPrtRoot->Update();
CATBSTR newBstr;
pCATIArtPrtRoot->get_PartNumber(newBstr);
CATUnicodeString s;
s.BuildFromBSTR(newBstr);
if ( NULL != pCATIArtDoc )
{
    pCATIArtDoc->Release();    pCATIArtDoc = NULL;
}
if ( NULL != pCATIArtPrdtDoc )
{
    pCATIArtPrdtDoc->Release();    pCATIArtPrdtDoc = NULL;
}
//设置草图为显示状态
CATIVisProperties *pVisProperties = NULL;
rc =

```



```

pSpecObjectOnSketch->QueryInterface(IID_CATVisProperties, (void**)&pVisProperties
);
    if( SUCCEEDED(rc) && pVisProperties!=NULL)
    {
        CATVisPropertiesValues MyProp;
        pVisProperties->GetPropertiesAtt(MyProp, CATVPShow);
        MyProp.SetShowAttr(CATShowAttr);

        pVisProperties -> SetPropertiesAtt(MyProp, CATVPShow , CATVPGlobalType);//
CATVPShow CATVPGlobalType
        pVisProperties -> Release();
        pVisProperties = NULL ;
    }
    pSpecObjectOnSketch->Update();
//设置文件路径
CATUnicodeString DocTitle="NewPart";
CATUnicodeString DocName="NewPart.CATPart";
CATUnicodeString DocPath="D:\\";
//保存文件
CATDocumentServices::SaveAsNew(*pDoc, DocPath+ DocName);

```