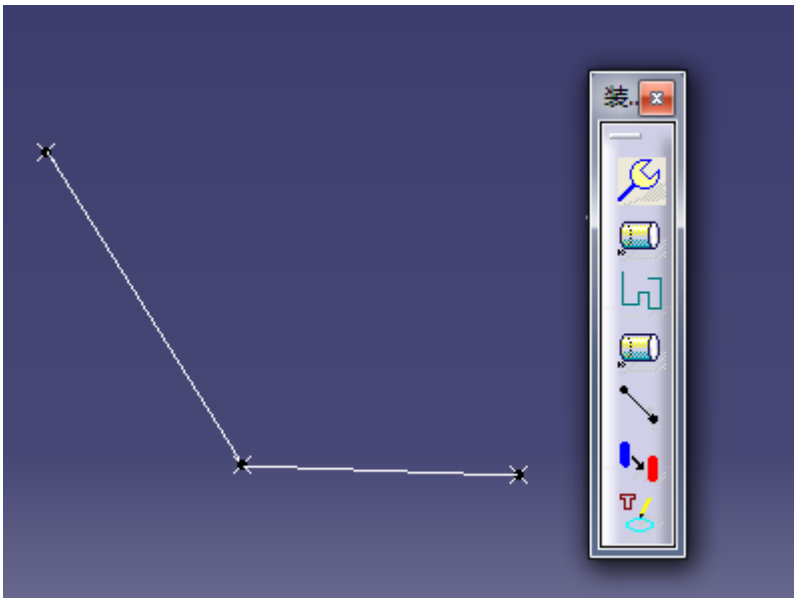
第 3 讲状态机和对话代理

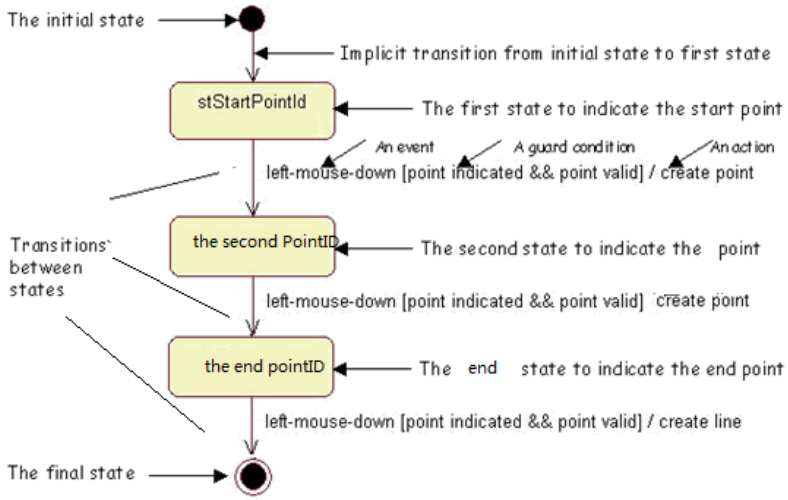
目录

1. 概述

目标 1：用鼠标在屏幕上点击的方式画点、线

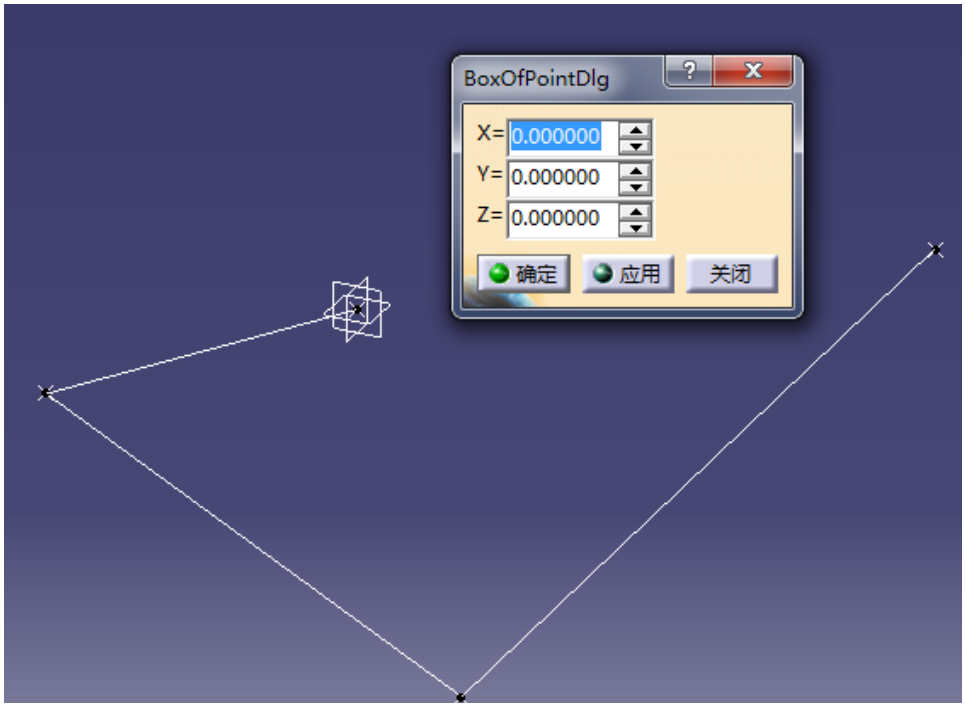


点击三次绘制两条直线的状态转移图

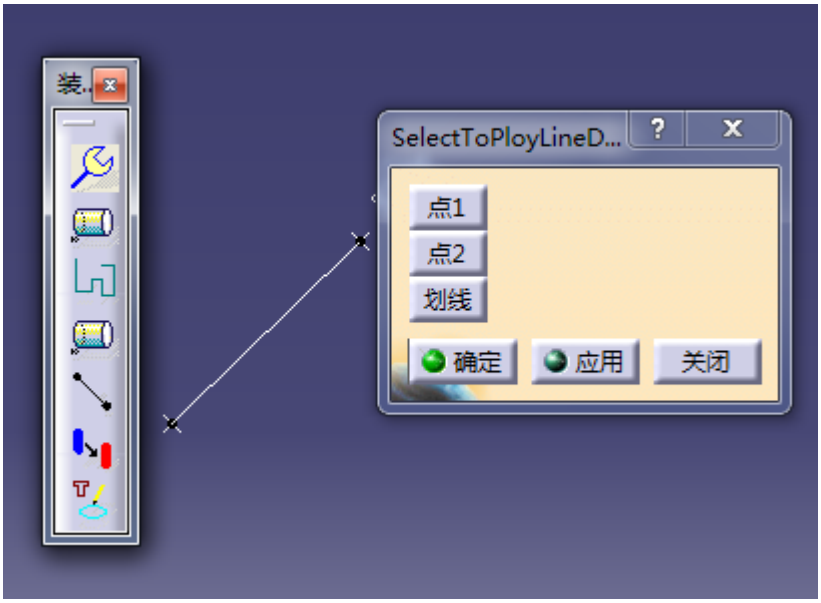


用鼠标在屏幕空白处随意点击，在点击处画点

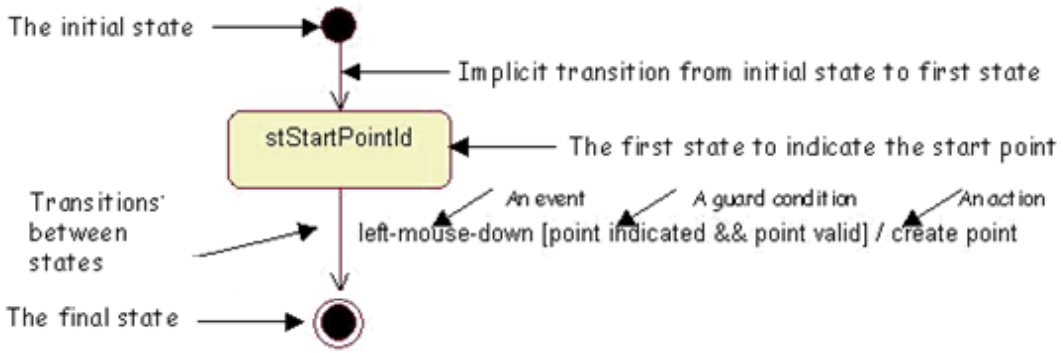
目标 2：用对话框输入方式维坐标画点、线



目标 3：屏幕上已经有两个点，用鼠标选取这两点画线



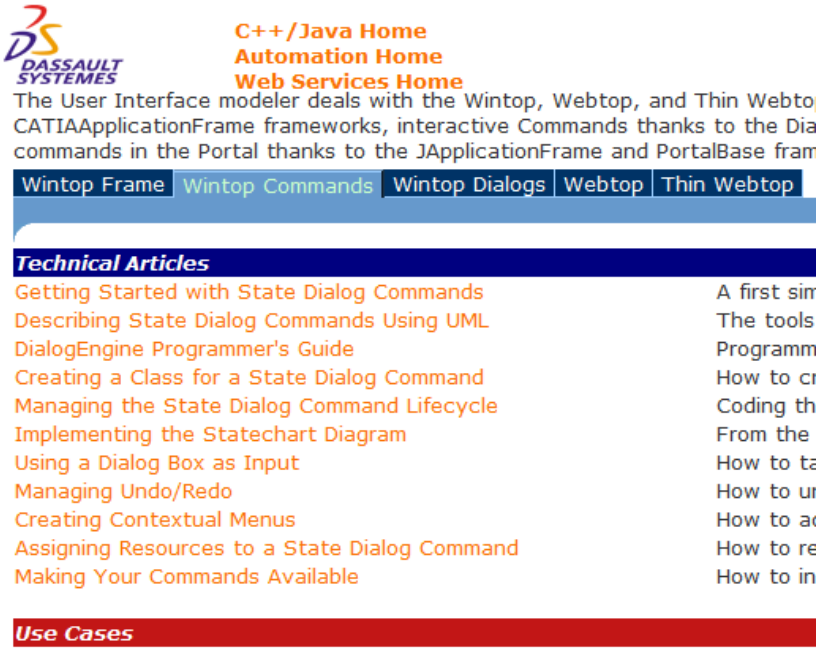
点击一次选择一个点的状态转移图



2. 简介

百科全书->User InterFace->WinTop Command

这里边所有的内容都得看。



3. CATIA 二次开发命令简介

命令是程序交互的基础，CAA 中命令主要分为三类：

（1）单步命令（One-shot Commands）：运行时用户不能有附加选项，从其开始运行直至其结束期间无法停止，该类命令派生自 CATCommand 类。

（2）对话框命令（Dialog Box Commands）：用户可以输入参数值或选择选项，对话框本身即命令，而不是其它命令的一部分，该类命令派生自 CATDlgDialog 类。

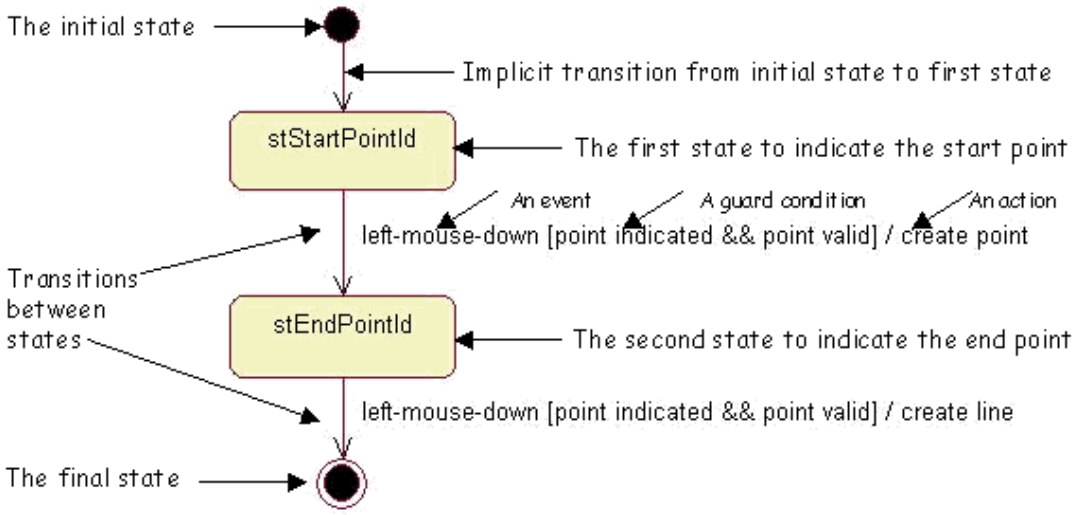
（3）状态对话命令（State Chat Commands）：状态对话命令被模拟为状态机，通过状态、迁移（或转换）的组合可构成高级对话命令。命令中可有数个状态，每个状态让用户选择对象、输入参数或选择选项。根据选择的对象、输入参数或选项判断是否满足条件，如果满足相应条件则触发迁移，跳转到下一状态执行，直到命令结束。对话框可用于状态对话命令作参数或选项输入界面。该类命令派生自 CATStateCommand 类。

3.1 状态机简介（state machine）

状态机是由状态和迁移组成的图，通常状态机附属于类，描述了类实例对接收事件的响应状态机是某个类的对象所有可能生命历史的模型，所有外部世界对对象的影响被总结为事件。

事件是具有时间和空间位置的显著发生的某件事，如鼠标在窗口某个位置点击、控件的某个操作等。当对象检测到事件，将对事件作出判断，并以相关于当前状态的方式来响应，这里的判断称为迁移条件或监控条件。响应可能包括动作的执行和改变到新的状态。状态即描述了对象生命周期中的一段时间，可通过三个方面来界定：某些性质上具有相似性的一系列对象值，对象等待某个或某些事件发生的一段时间，对象执行某些正在进行活动的一段时间。状态由迁移来连接，迁移定义了状态对象对某事件的响应，包括触发事件、监控条件、动作及目标状态。状态机对理解控制机制较实用，如用户交互界面和设备控制器等。

指定两个点绘制一条直线的状态机（state machine）



状态机从初始状态（initial state）进行到结束状态（final state）。从初始状态到第一个状态（first state）的转移中没有任何响应函数调用，命令都会自动跳过初始状态从第一个状态开始执行；第一个状态是等待用户鼠标指定一个点（画点的位置），这个状态的命称是框中给出的 stStartPointId；只要用户指定一个有效的点，当鼠标左键按下去的时候，鼠标按键事件就已经被检测出，并判断满足创建点的条件，执行响应函数绘制了一个点，同时状态就从第一个状态转移到第二个状态（stEndPointId）,第二个状态也是等待用户用鼠标指定一个点。当鼠标第二次按下去的时候，鼠标按键事件已经被检测出，判断了绘制线的条件，执行了响应函数绘制了

一条线，同时状态从第二个状态转移到最终状态（final state）。

3.2 状态（State）和代理（Agent）

对话代理（Dialog Agent）是处理与界面命令相关的一个接口 CATDialogAgent,其中与人机交互有关的代理大致分为 2 类，选择和指定，他们分别对应接口 CATPathElementAgent 和 CATIndicationAgent。

一个在文件的对象显示在视图中，用户用左键点击它，这个对象可以作为后续工作的输入，比如通过 3 个点创建一个平面。选择两个点创建一条直线等。这种与选择有关的都要通过选择代理 CATPathElementAgent 来实现。

在 catia 中鼠标的每次选择对象都会产生路径，意思就是所选对象的位置路径，比如选中了某个点后会产生路径:Productee.1\part.1\几何图形集\点.1。若选择的是实体表面可能会产生路径：part.1\零件几何体\凸台.1\面。这种路径未必会按照 CATIA 中的结构树的上下级关系产生。它是按照几何元素对象的拓扑关系产生的。在选择后会自动产生路径，并将路径存放在 CATPathElementAgent 对象中。所以我们在选择了零件中的一个对象后从这个对象追溯它的路径中包含的对象。常用情形是在装配体中选择某个几何对象后获取几何对象所在的 part 名称。

使用选择代理要做到以下几个步骤：

3.2.1 声明代理（Agent）

一般而言代理要在命令执行期间都处于活动状态，因此申明为类的成员变量。

```
classCAADegCreatePlaneCmd : public CATStateCommand
{
…
private :
CATPathElementAgent * _daPathElement;
…
}
```

选择代理 CATPathElementAgent 是 CATPathElement 类的一个实例。

3.2.2 过滤代理（Agent）

选择代理实例化一般在命令的 BuildGraph 函数中进行如下所示：

```
voidCAADegCreatePlaneCmd::BuildGraph()
{
   ...
   _daPathElement = new CATPathElementAgent("GetPoint");
   _daPathElement->AddElementType(IID_CATIGSMPoint);
   ...
}
```

这里字符串“GetPoint”定义了一个选择代理在程序中的唯一识别符号。AddElementType 方法的参数给定了选择代理只能识别 CATIGSMPoint 类型的点对象，其他对象一概不能识别。

AddElementType 也可以通过字符串的方式给定，比如：

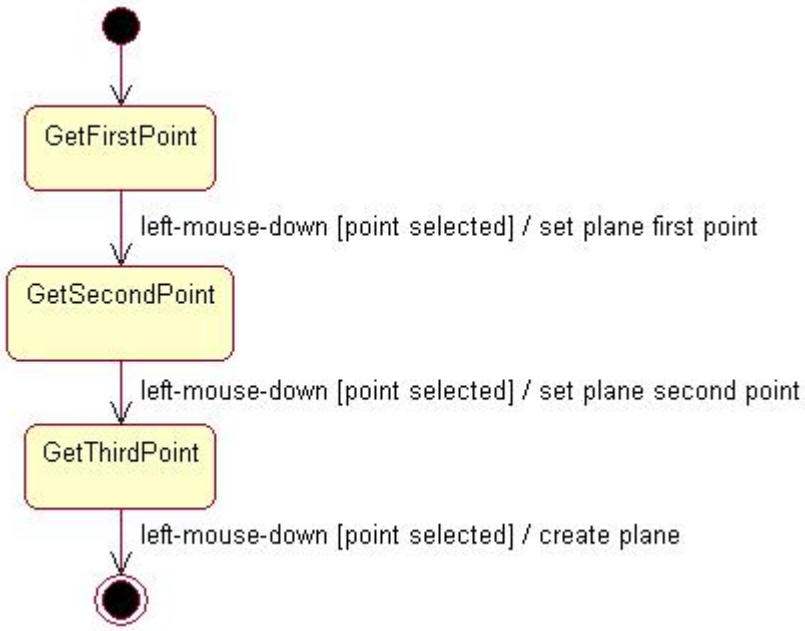
```
_daPathElement->AddElementType("CATIGSMPoint ");
```

若可以选择的对象较多是可以采用字符串链表的形式给定，如下所示：

```
CATListOfCATString Types;
Types.Append("CATPoint");
Types.Append("CATINewHole");
Types.Append("CATISketch");
Types.Append("CATIGSMLineNormal");
Types.Append("CATIGSMLinePtPt");
_daPathElement ->SetOrderedTypeList(Types);
```

3.2.3 为某个状态（State）添加代理（Agent）

选择三个点创建一个平面的状态机如下所示：



任然在BuildGraph方法中进行编程。状态机的初始状态一般不用设置代理，此处应该首先设置创建一个状态名为GetSecondPoint 的状态，并且将选择代理添加到GetSecondPoint状态中。这样就可以在该状态变成激活状态的时候给选择代理赋值。代码如下所示：

```
…
CATDialogState *stSecondState = AddDialogState("stSecondPointId");
stSecondState->AddDialogAgent(_daPathElement);
  …
```

3.2.4 定义状态（State）之间的转移

在两个状态之间创建转移，当选择代理被赋值时状态转移会触发，即就是当用户选择一个点时状态转移会被触发，并且若返回值是 true，那么状态转移函数就会执行。

状态转移代码也是在 BuildGraph 中编写。代码如下：

```
…
CATDialogTransition *pSecondTransition = AddTransition
```

```
    (
  stSecondState,
  stEndState,
  AndCondition(IsOutputSetCondition(_daPathElement),
  Condition((ConditionMethod) &CAADegCreatePlaneCmd::CheckPoint2)),
  Action((ActionMethod) &CAADegCreatePlaneCmd::CreatePoint,
                NULL, NULL, (void *) 2)
    );
```

状态转移的第一个参数为转移之前的状态，第二个参数为转移之后的状态，第三个参数为判断是否转移的条件，第四个参数为转移条件满足时执行的转移函数，这儿的转移函数为CreatePoint。

3.2.5 取值后重新初始化代理（Agent）

当用户鼠标选择了点，状态转移条件达成，转移函数开始执行。代码如下所示：

```
...
CATBooleanCAADegCreatePlaneCmd::CreatePoint(void *iPointIndex)
{
int index = (int ) iPointIndex -1;
   // Gets x,y,z from the selected point
floatx,y,z ;
CATPathElement * pModelPath = _daPathElement->GetValue();
CATBaseUnknown * pExpectedPoint = NULL ;
if ( pModelPath&&pModelPath->GetSize() )
   {
intElementCountInPath = pModelPath->GetSize() - 1;
pExpectedPoint = (*pModelPath)[ElementCountInPath];
   }
   ...
}
```

选择代理在用户鼠标点击时已经赋值，第一件事就是用 GetValue 方法从_daPathElement中将选择的值获取出来。正如前边所述，选择代理的值就是从根对象到选择对象的一系列路径表，所选对象就在路径的最后。

从所选对象获取 CATIGSMPoint 的代码如下所示：

```
if ( pExpectedPoint )
   {
CATIGSMPoint* pExpectedPointAsISysPoint = NULL;
      HRESULT rc = pExpectedPoint->QueryInterface(IID_CATIGSMPoint,

(void**)&pExpectedPointAsISysPoint);
if (SUCCEEDED(rc))
      {
```

```
        …
pExpectedPointAsISysPoint->Release();
      }
   }
```

这儿采用了 QueryInterface 来获取对象，也可以采用 Search 方法获取对象。比如从所选对象获取 CATISpecObject 的代码如下所示：

```
    CATISpecObject_varspObj = NULL_var;//CATISpecObject 是选择对象
    pPath->Search(IID_CATISpecObject, (void**)&spObj );//从路径中获得当前对象
```

若这个选择代理在这次选择之后还要使用，那么需要重置选择代理，代码如下：

```
…
   _daPathElement->InitializeAcquisition();
return TRUE;
}
```

3.2.6 代理（Agent）指针释放

命令结束的时候应该在析构函数或者取消中添加释放选择指针的代码。如下所示：

```
CAADegCreatePlaneCmd::~CAADegCreatePlaneCmd()
{
   …
if (_daPathElement) _daPathElement->RequestDelayedDestruction();
   _daPathElement = NULL ;
   …
```

3.3 鼠标点击代理（Agent）

指定代理是指鼠标在屏幕上点击或者移动的时候发生的事件，这点击或者移动的过程中产生临时 2D 点。指定代理能够获取文件中不存在的临时 2D 点的坐标，这 2D 点是由用户在 2D 或 3D 视图中屏幕上点击鼠标。

3.3.1 创建 2D 代理（Agent）

与选择代理类似，一个指定代理在声明之后可以采用如下代码创建：

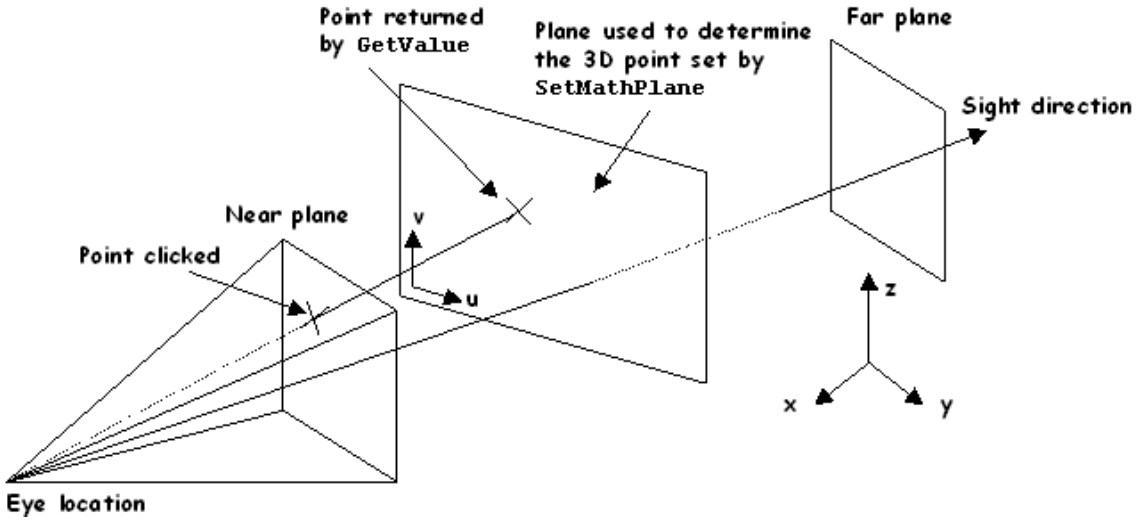
```
_daIndicationAgent = new CATIndicationAgent("2DIndicationAgentId");
```

3.3.2 在 2D 屏幕上指定一个 3D 点

一个鼠标指针代理致力于怎么从 2D 屏幕所在的平面中获取 2D 点坐标，并将坐标转化在绝对坐标系。鼠标在屏幕上的点击是不确定的（只给定了 2 个坐标值），如果给定一个投影平面，那么在屏幕上的点击会根据视线方向（当前视角给定）投影到投影平面。（默认的投影平面就是与屏幕平行的平面，也可给指定代理提供一个平面）

注意，为了获得投影点，投影平面不能与近处或远处的平面垂直。

如图所示：



指定代理与状态机的结合如同选择代理一样过程。

3.3.3 获取指定的点

用户指定的点可以直接以 CATMathPoint2D 的实例形式获取 2D 视图中的点，还需要将改 2D 点转化为 3D 视图中的点。代码如下：

```
//2D 视图中的点
...
CATMathPoint2D IndPoint = _daIndicationAgent->GetValue();
double X = IndPoint.GetX();
double Y = IndPoint.GetY();
// OR
double X, Y;
IndPoint.GetCoord(X, Y);
...
//3D 视图中的点
...
CATMathPoint2D IndPoint2D = _daIndicationAgent->GetValue();
CATMathPoint IndPoint3D;
Plane.EvalPoint(IndPoint2D.GetX(),IndPoint2D.GetY(), IndPoint3D);
double X = IndPoint3D.GetX();
double Y = IndPoint3D.GetY();
double Z = IndPoint3D.GetZ();
// OR
double, X, Y, Z;
IndPoint3D.GetCoord(X, Y, Z);
...
```

如果没有定义投影平面，也可以采用 GetMathPlane 方法获取默认的平面。代码如下：

```
...
CATMathPlane Plane = _daIndicationAgent->GetMathPlane();
```

Plane.EvalPoint(IndPoint2D.GetX(),IndPoint2D.GetY(), IndPoint3D);

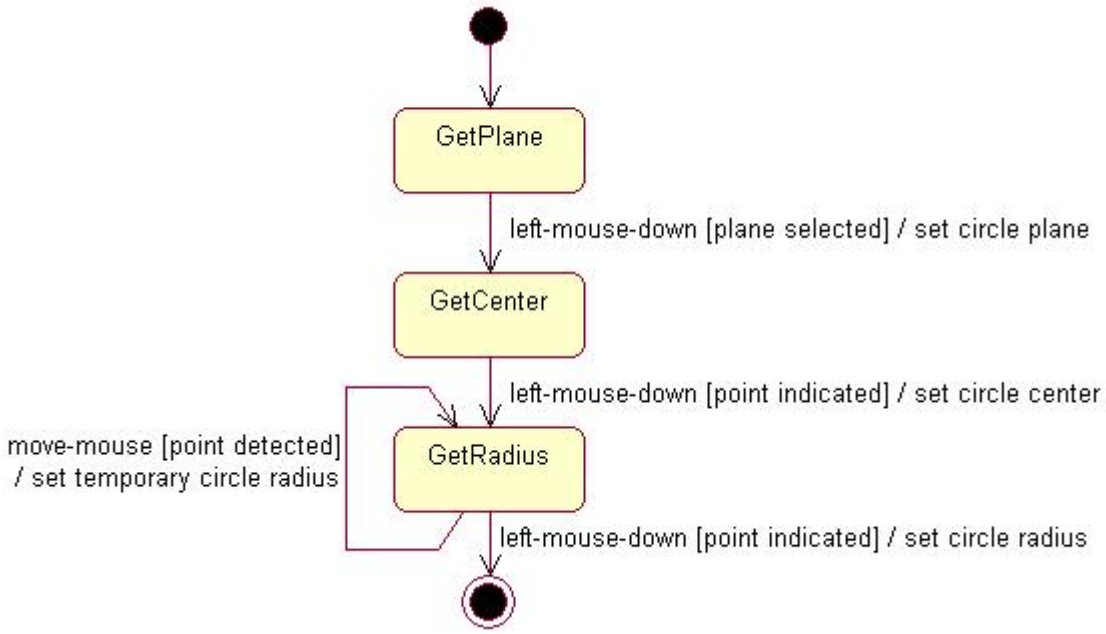
...

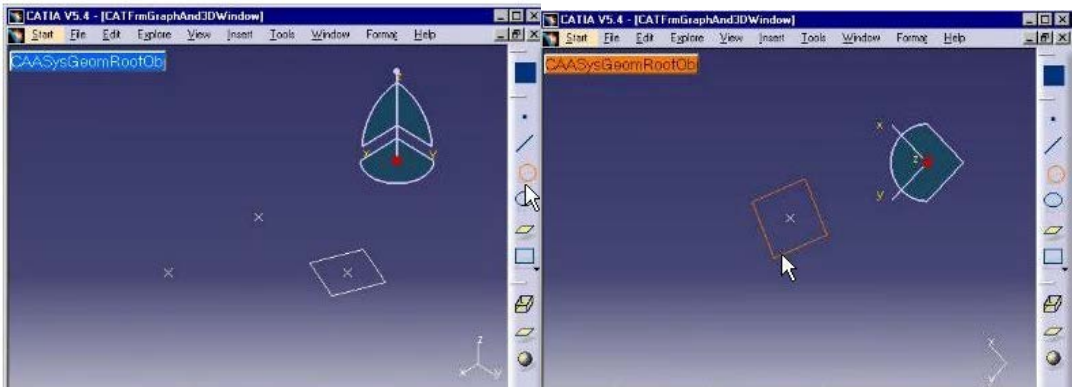
3.4 鼠标移动事件

当鼠标在执行命令时可以获取鼠标移动的轨迹，就称之为鼠标移动事件，在鼠标在屏幕上点击画点的过程中用鼠标在当前屏幕上的位置转化为 3D 空间中的坐标，用这个 3D 坐标绘制一个 3D 点，并且创建了一个临时对象来显示，在检测到鼠标移动事件的时候这个临时对象会更新以跟着鼠标移动，就好像临时对象和鼠标粘在一起。

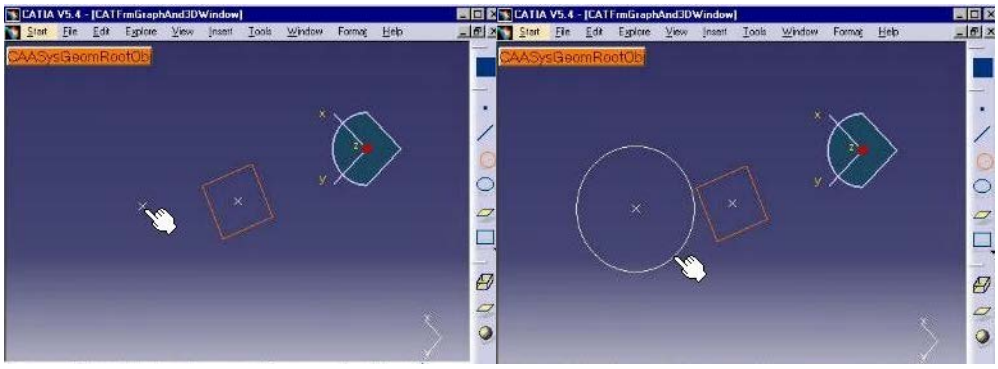
这个在下边的例子中临时对象是一个临时的圆，能够帮助用户看清当点击鼠标时创建当前圆的大小。临时对象不会存储在文件中，它的显示对象使用了 ISO（交互对象集），把对象添加到 ISO 后会自动显示，这个 ISO 是由 CATI3DGeoVisu 接口实现的。

采用鼠标在 3D 空间绘制一个圆，它的状态图如下所示。





GetPlane 状态　　getcenter 状态



GetRadius 状态　　GetRadius 状态迁移

指一个点（indicating a point）意思是用鼠标在屏幕上希望的位置用左键点击，命令从这次点击获取绝对坐标系中的坐标并创建一个几何点。这样做的原因是状态中有 Indication agent 参与，它用来询问指一个点。这个点是鼠标在屏幕平面上的点击投影到具体平面上而来。

鼠标在移动过程中需要定义一个从当前状态转移到当前状态的一个循环转移。循环转移的定义代码为：

```
CATDialogTransition *pRubberTransition = AddTransition
(
   stGetPoint,//当前状态
   stGetPoint,//转移后还是当前状态
   IsLastModifiedAgentCondition(_daIndicpoint),
  Action((ActionMethod) & MouseMovePoint::UpdatePointCoor)
    );
CATDialogTransition *pThirdTransition = AddTransition
(
   stGetPoint,
   NULL,//终止状态
   IsOutputSetCondition(_daIndicpoint),
  Action((ActionMethod) & MouseMovePoint::CreatePoint)
);
```

4. 实例：用鼠标在屏幕上点击的方式画点线

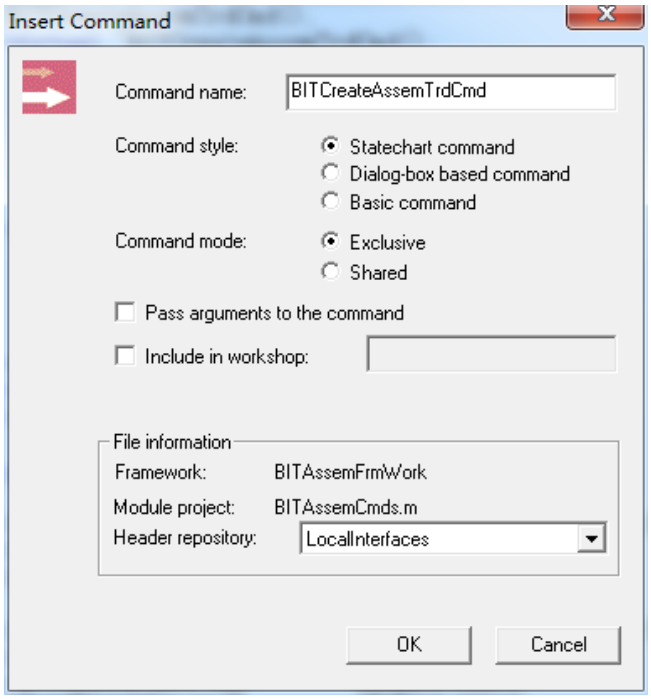
4.1 创建工作空间（Workspace）和框架（Frame）

根据第一章《CAA 简介和框架》，创建工作空间和框架，并添加相应的模块（module），用来存放命令（Command）和对话框（Dialog），此处不再赘述。

4.2 创建命令（Command）

单击：文件->Add CAAV5 Itrem->CATIA resource->Command

Command name：BITCreateAssemTrdCmd；Command style：StateChart Command



单击 OK 后自动生成两个文件 BITCreateAssemTrdCmd.h 和 BITCreateAssemTrdCmd.cpp

4.3 修改和添加代码

打开 BITCreateAssemTrdCmd.cpp

4.3.1 BuildGraph()

找到 BuildGraph(){……}

将大括号中的代码用以下代码覆盖：

```
_StartPointIndication = new CATIndicationAgent ("StartPointIndication");

 CATDialogState * stStartState = GetInitialState("stStartPointId");//第一个初始状态时用GetInitialState

stStartState -> AddDialogAgent (_StartPointIndication);

_EndPointIndication = new CATIndicationAgent ("EndPointIndication");

 CATDialogState * stEndState = AddDialogState("stEndPointId");//后边的状态时用AddDialogState

stEndState -> AddDialogAgent (_EndPointIndication);

  _ThrPointIndication = new CATIndicationAgent ("ThrPointIndication");

 CATDialogState * stThrState = AddDialogState("stThrPointId");

stThrState -> AddDialogAgent (_ThrPointIndication);
```

```
AddTransition(stStartState,stEndState,          //从初始状态到第二个状态
                IsOutputSetCondition (_StartPointIndication),   //ActionOne执行的条件
                Action ((ActionMethod) &BITCreateAssemTrdCmd::ActionOne));//执行ActionOne

AddTransition(stEndState, stThrState ,
                IsOutputSetCondition (_EndPointIndication),
                Action ((ActionMethod) &BITCreateAssemTrdCmd::ActionTwo));

AddTransition( stThrState,NULL ,
                IsOutputSetCondition (_ThrPointIndication),
                Action ((ActionMethod) &BITCreateAssemTrdCmd::ActionThree));
```

以上代码是用来创建状态，判断状态。在与屏幕的互动（Indication）中必不可少。注意最后一个 AddTransition 中的 NULL，意思是没有下一个状态了，如果将此处的 NULL 改为 stStartState 那么程序将会循环进行，也就是说，你可以在屏幕上点击无数次。

4.3.2 ActionOne

找到

```
CATBoolean BITCreateAssemTrdCmd::ActionOne( void *data )
{
return TRUE ;
}
```

在大括号中的 return TRUE ;前添加以下代码：

```
////////////////// 创建第一个点
    CATMathPoint2D point2D = _StartPointIndication->GetValue();//获得一个D的点
    CATMathPoint Point3D;
    CATMathPlane Plane = _StartPointIndication->GetMathPlane();
    Plane.EvalPoint(point2D.GetX(),point2D.GetY(),Point3D);      //将D点转换为D点
////////////////设置Container
    CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();    //获得Editor
```

```
CATDocument *pDoc = pEditor->GetDocument();                    //获得Document
CATIContainerOfDocument_var spConODocs = pDoc;
_pContainer = NULL;
HRESULT hr = spConODocs->GetSpecContainer(_pContainer);        //获得SpecContainer
//SpecContainer:It is the container which contains the mechanical features. It is those that you can see in the specification tree.
spGSMFactory = NULL_var;
spGSMFactory = _pContainer;                                    //设置工厂

//////////////创建一个CATISpecPoint并将其显示出来
CATIGSMPoint_var spPoint1 = spGSMFactory->CreatePoint(Point3D); //Creates a point
spSpecPoint1 = spPoint1;                    //Casts the point as a CATISpecObject
CATIGSMProceduralView_var spSndPntObj = spSpecPoint1;
spSndPntObj->InsertInProceduralView();                         //将点显示在屏幕上
_StartPointIndication->InitializeAcquisition();
```

以上代码的作用是在屏幕上创建一个点

4.3.3 创建并编辑 ActionTwo 和 ActionThree

在 ActionOne 的后边编写 ActionTwo。

继续添加代码

```
CATBoolean BITCreateAssemTrdCmd::ActionTwo( void *data )
{
//////////////////创建点；方法与ActionOne一样
    CATMathPoint2D point2D = _EndPointIndication->GetValue();
    CATMathPoint Point3D;

    CATMathPlane Plane = _EndPointIndication->GetMathPlane();
    Plane.EvalPoint(point2D.GetX(),point2D.GetY(),Point3D);
    //此处spGSMFactory是全局变量，所以在ActionOne中赋值之后可以使用
    CATIGSMPoint_var spPoint1 = spGSMFactory->CreatePoint(Point3D);
```

```
        spSpecPoint2 = spPoint1; //Casts the point as a CATISpecObject


        CATIGSMProceduralView_var spSndPntObj = spSpecPoint2;
        spSndPntObj->InsertInProceduralView();
        _EndPointIndication->InitializeAcquisition();


        //////////用两个SpecPoint划线
        CATIGSMLinePtPt_var spLine1 = spGSMFactory->CreateLine(spSpecPoint1,
spSpecPoint2);
        CATISpecObject_var spSpecLine1 = spLine1;//将线转换为SpecObject型
        //////////将画好的线显示在屏幕上
        spSpecLine1->Update();
        CATIGSMProceduralView_var spCurObj = spLine1;
        spCurObj->InsertInProceduralView();
    return TRUE ;
    }
```

这段代码的功能是再绘制一个点，通过这个点和 ActionOne 中绘制的点创建线段。

在 ActionTwo 的后边继续添加 ActionThree。ActionThree 的功能和 ActionTwo 的功能完全一样，唯一的区别在于创建的点的名称同。

```
    CATBoolean BITCreateAssemTrdCmd::ActionThree( void *data )
    {
    // Creation of the first 3D point
        //道理与前边ActionOne一样，不再详细注释
        CATMathPoint2D point2D = _ThrPointIndication->GetValue();
        CATMathPoint Point3D;


    // Projection of the 2D point in the 3D space
        CATMathPlane Plane = _ThrPointIndication->GetMathPlane();
        Plane.EvalPoint(point2D.GetX(),point2D.GetY(),Point3D);
```

```
        CATIGSMPoint_var spPoint1 = spGSMFactory->CreatePoint(Point3D); //Creates a point
        spSpecPoint3 = spPoint1; //Casts the point as a CATISpecObject
        CATIGSMProceduralView_var spSndPntObj = spSpecPoint3;
        spSndPntObj->InsertInProceduralView();

        _StartPointIndication->InitializeAcquisition();

        CATIGSMLinePtPt_var spLine1 = spGSMFactory->CreateLine(spSpecPoint2,
spSpecPoint3);
        CATISpecObject_var spSpecLine1 = spLine1;
        spSpecLine1->Update();
        CATIGSMProceduralView_var spCurObj = spLine1;
        spCurObj->InsertInProceduralView();
    return TRUE ;
    }
```

添加头文件

打开 BITCreateAssemTrdCmd.h 文件

现在光有这些代码是不能正确运行的。首先，没有添加头文件，其次，没有在 BITCreateAssemTrdCmd.h 中添加 ActionTwo、ActionThree、全局变量的声明

在 BITCreateAssemTrdCmd.h 中添加以下头文件：

```
#include"CATMathPoint.h"
#include"CATISO.h"
#include"CATFrmEditor.h"
#include"CATDocument.h"
#include"CATIContainerOfDocument.h"
#include"CATIGSMFactory.h"
#include"CATIPrtFactory.h"
#include"CATICkeParmFactory.h"
```

```
#include"CATIGSMPoint.h"
#include"CATIGSMProceduralView.h"
#include"CATIGSMFactory.h"
#include"CATIDescendants.h"
#include"CATIGSMLinePtPt.h"
```

这些头文件未必全都会用，但是添加之后也不会有错。

4.3.4 添加函数和变量声明

继续在.h文件里边添加，在类的大括号中添加以下声明：

```
public:
virtual CATBoolean  ActionTwo(void * data);       //对ActionTwo的声明
virtual CATBoolean  ActionThree(void * data);//对ActionThree的声明

private:

  CATIndicationAgent   * _StartPointIndication; //三个Indication
  CATIndicationAgent   * _EndPointIndication;
  CATIndicationAgent   * _ThrPointIndication;

  CATISpecObject_var       spSpecPoint1;       //三个SpecObject类型的点
  CATISpecObject_var       spSpecPoint2;
  CATISpecObject_var       spSpecPoint3;

  CATIContainer*            _pContainer;       //Container
  CATIGSMFactory_var        spGSMFactory;      //GSM 工厂
```

在这儿不用将类的私有成员（private: ）和公共成员（public: ）分得特别清楚。

4.3.5 将命令（Command）和菜单（Addin）关联

和 Train1 中所述一样，打开 BITAssemAddin.cpp，

在void BITAssemAddin::CreateCommands(){.....}中添加代码：

```
new BITAssemHeader("BITAssemThrHdr", "BITAssemCmds", "BITCreateAssemTrdCmd", (void*)NULL);
```

在CATCmdContainer * BITAssemAddin::CreateToolbars(){.....}中添加代码：

```
//////////////////////////////////////////////////////////创建第三个按钮
    //3.1。添加按钮
        NewAccess(CATCmdStarter, pCAATPMSCreateLineThrStr, CAATPMSCreateLineThrStr);
        //3.2。给按钮关联命令
        SetAccessCommand(pCAATPMSCreateLineThrStr,"BITAssemThrHdr");
    //3.3。将按钮添加至工具条中
    SetAccessNext(pCAATPMSCreateLineSndStr, pCAATPMSCreateLineThrStr);
```

再在下边找到相应位置继续添加：

```
NewAccess(CATCmdStarter,pCAATPMuAixsAutoThrStr,CAATPMuAixsAutoThrStr);
SetAccessCommand(pCAATPMuAixsAutoThrStr,"BITAssemThrHdr");
SetAccessNext(pCAATPMuAixsAutoSndStr,pCAATPMuAixsAutoThrStr);
```

这样就将程序和工具条相关联。

（注意：这个地方写的不好，不详细，最好仔细看源代码，源代码的打开方法见“一些注意事项”）

4.4 运行

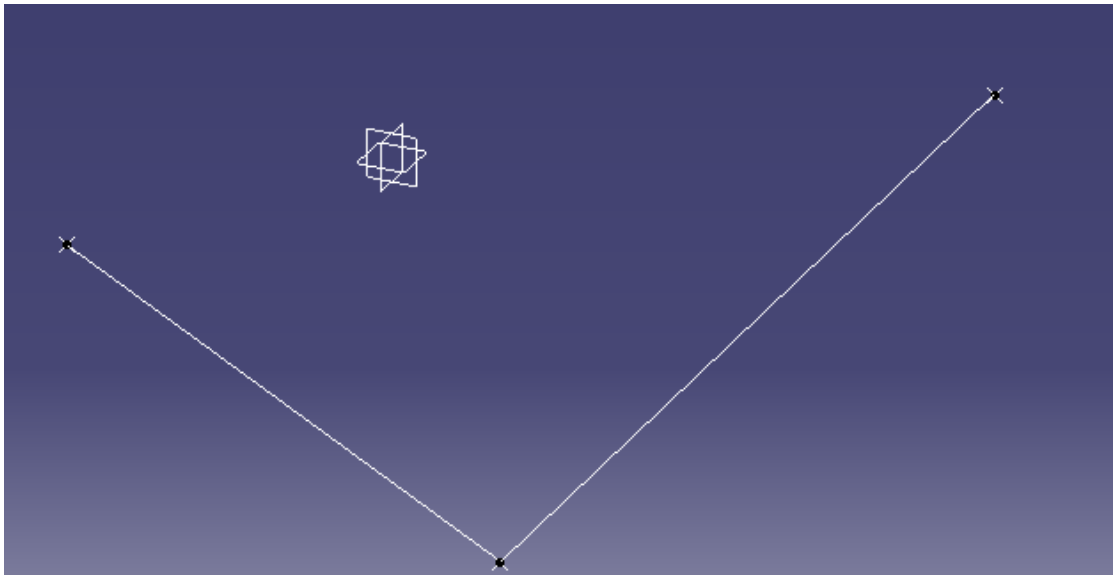
CAAV5 Workspace->Create/Update Runtime View->Ok

生成->mkmk->OK

窗口->Open runtime Window-输入”Cnext”->回车

开始->机械设计->零件设计->装配a->连续划线1

在屏幕空白处点击鼠标->在屏幕空白处点击鼠标->在屏幕空白处点击鼠标



5. 实例：输入坐标画点、线

5.1 创建命令（Command）

单击：文件->Add CAAV5 Itrem->CATIA resource->Command

在弹出 Insert Command 对话框中输入和选择：

Command name：FourCmd（注意：此处名字必须和 BITAssemAddin.cpp 中 new BITAssemHeader()中的倒数第二个参数相同。BITAssemAdding.cpp 是根据这个参数运行 FourCmd 的）

Command style 选择 Basic Command

Command Mode 选择 Exclusive（默认值）

点击 OK 键后生成两个文件 FourCmd.cpp 和 FourCmd.h

5.2 创建对话框（Dialog）

5.2.1 创建文件

单击：文件->Add CAAV5 Itrem->CATIA resource->Dialog

在弹出 Insert Dialog 对话框中输入和选择：

Dialog object type：Dialog box（默认值）

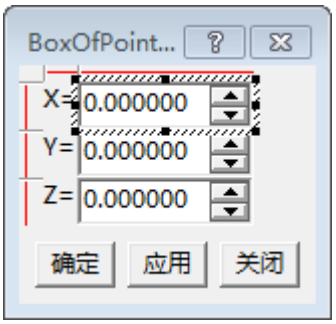
Dialog Class name：BoxOfPointDlg

点击 OK 键后生成三个文件 BoxOfPointDlg.cpp 和 BoxOfPointDlg.h 和 BoxOfPointDlg.CATDlg

5.2.2 编辑对话框

打开 BoxOfPointDlg.CATDlg

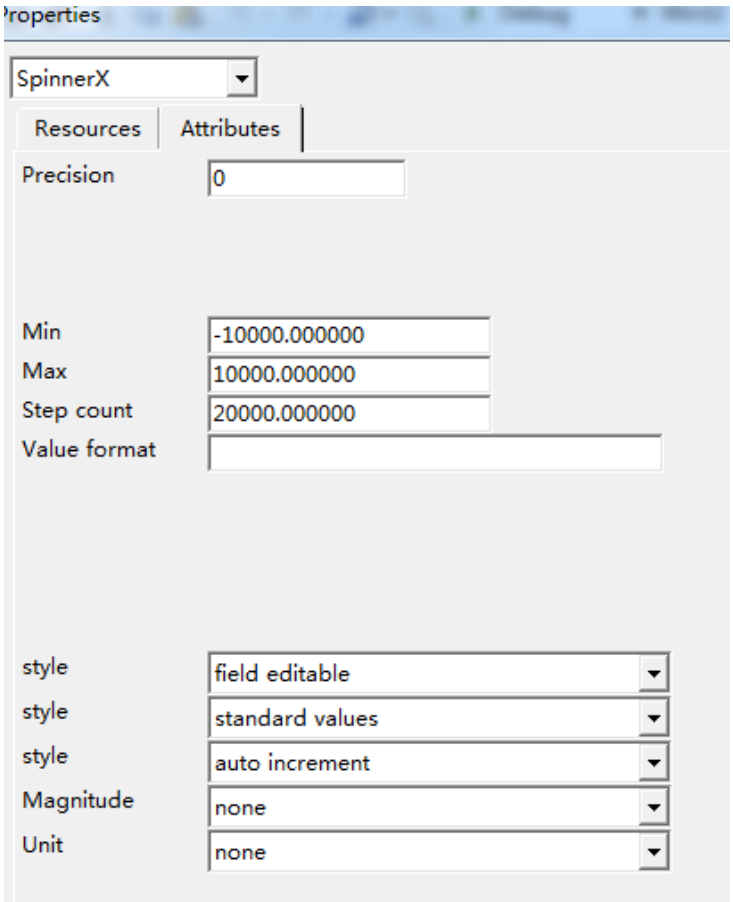
添加控件：形成如下图所示对话框。



3 个 Label 的 Title 分别为“X=”、“Y=”、“Z=”

3 个 Spinner 的 Name 分别为“SpinnerX”、“SpinnerY”、“SpinnerZ”

在每个 Spinner 的 Attributes 选项卡中按照下图修改属性：



在 Attributes 选项卡中 Min 和 Max 意识是最大最小值，Step Count 意识是吧-10000-10000 之间的数分成多少份。此处 20000 是分成 20000 份，那么每按一次黑色三角形变化值为 1

第一个 Style 选择 Field editable

记着点击应用。

5.3 添加代码

5.3.1 将 FourCmd.cpp 和 BITAssemAddin.cpp 文件关联

和 Train1 中所述一样，打开 BITAssemAddin.cpp。

在void BITAssemAddin::CreateCommands(){.....}中添加代码：

```
new BITAssemHeader("BITAssemFurHdr", "BITAssemCmds", "FourCmd", (void*)NULL);
```

在CATCmdContainer * BITAssemAddin::CreateToolbars(){.....}中添加代码：

```
//////////////////////////////////////////////////////////////////////创建第四个按钮
    //添加按钮
        NewAccess(CATCmdStarter, pCAATPMSCreateLineFurStr, CAATPMSCreateLineFurStr);
        //给按钮关联命令
        SetAccessCommand(pCAATPMSCreateLineFurStr,"BITAssemFurHdr");
    //将按钮添加至工具条中
    SetAccessNext(pCAATPMSCreateLineThrStr, pCAATPMSCreateLineFurStr);
```

在后边找到相应位置继续添加代码：

```
                    //3.添加菜单项
            NewAccess(CATCmdStarter,pCAATPMuAixsAutoFurStr,CAATPMuAixsAutoFurStr);
            SetAccessCommand(pCAATPMuAixsAutoFurStr,"BITAssemFurHdr");
            SetAccessNext(pCAATPMuAixsAutoThrStr,pCAATPMuAixsAutoFurStr);
```

（注意：这个地方写的不好，不详细，最好仔细看源代码，源代码的打开方法见“一些注意事项”）

5.3.2 将 FourCmd 和 BoxOfPointDlg 关联

打开 FourCmd.Cpp

找到CATStatusChangeRC FourCmd::Activate(CATCommand * iFromClient, CATNotification * iEvtDat){}

在大括号中添加

```
    BoxOfPointDlg * _pPointEditor = new BoxOfPointDlg(); //新建一个BoxOfPointDlg对话
```

框

```
_pPointEditor->Build();                                    //运行Build（）函数
_pPointEditor->SetVisibility(CATDlgShow);
```

添加头文件：

```
#include"BoxOfPointDlg.h"
```

5.3.3 添加 APPLY 按钮响应程序

打开BoxOfPointDlg.cpp文件

找到函数：

```
void BoxOfPointDlg::OnBoxOfPointDlgDiaAPPLYNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
```

{}

在大括号中添加代码：

```
//////////////////////给点_Point设这坐标值。x、y、z分别从对话框的Spinner中获得
_Point.SetCoord(_SpinnerX->GetCurrentValue(),
                _SpinnerY->GetCurrentValue(),
                _SpinnerZ->GetCurrentValue() );
CATMathPoint Point3D;
Point3D = _Point;          //将全局变量的值赋给局部变量

////////获得Editor、Document、Container、设置GSMFactory
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();
CATDocument *pDoc = pEditor->GetDocument();
CATIContainerOfDocument_var spConODocs = pDoc;
_pContainer = NULL;
HRESULT hr = spConODocs->GetSpecContainer(_pContainer);
spGSMFactory = NULL_var;
spGSMFactory = _pContainer;
//////////////////////用以上得到的Point3D画点
```

```
        CATIGSMPoint_var spPoint1 = spGSMFactory->CreatePoint(Point3D); //创建一个点
        spSpecPoint1 = spPoint1;                //Casts the point as a CATISpecObject
        CATIGSMProceduralView_var spSndPntObj = spSpecPoint1;
        spSndPntObj->InsertInProceduralView();
        ////////////////////////划线
        ///////用if语句判断是否为第一个点，如果不是第一个点就开始划线
        if(i==1)
        {
            ////////划线时spSpecPoint1和spSpecPoint2不能是同一个点。
            ////////如果是同一个点，那么两点之间的距离为，不能用CreatLine创建直线
            CATIGSMLinePtPt_var spLine1 = spGSMFactory->CreateLine(spSpecPoint1,
spSpecPoint2);
            CATISpecObject_var spSpecLine1 = spLine1;
            spSpecLine1->Update();
            CATIGSMProceduralView_var spCurObj = spLine1;
            spCurObj->InsertInProceduralView();
        }
        /////////将这次执行的spSpecPoint1赋值给spSpecPoint2，那么在下次点击APPLY按钮时可
以继续划线
        spSpecPoint2=spSpecPoint1;
        ///////修改用来判断是否为第一个点的变量值
        i=1;
```

现在运行程序会出现一系列的错误，这些错误大部分为声明的错误，所以要添加所用到的头文件和声明

打开 BoxOfPointDlg.h

添加头文件：

```
#include"CATMathPoint.h"
#include"CATISO.h"
#include"CATFrmEditor.h"
```

```
#include"CATDocument.h"
#include"CATIContainerOfDocument.h"
#include"CATIGSMFactory.h"
#include"iostream.h"
#include"CATIPrtFactory.h"
#include"CATICkeParmFactory.h"
#include"CATIGSMPoint.h"
#include"CATIGSMProceduralView.h"
#include"CATIGSMFactory.h"
#include"CATIDescendants.h"
#include"CATIGSMLinePtPt.h"
#include"CATIContainer.h"
#include"CATISpecObject.h"

#include"CATPanelState.h"
#include"CATDialogTransition.h"
#include"CATCustomizableTransition.h"
#include"CATApplicationFrame.h"
```

同样的这些头文件未必都是有用的，但是加上他们也没有错。

在类中声明全局变量：

```
public:
    CATMathPoint            _Point;
    CATIContainer          * _pContainer;
    CATISpecObject_var     spSpecPoint1;
    CATISpecObject_var     spSpecPoint2;
    CATIGSMFactory_var      spGSMFactory;
    int i;          //用来判断当前创建的点是否为第一个点的变量.i=1或i=0
```

5.3.4 运行程序

CAAV5 Workspace->Create/Update Runtime View->Ok

生成->mkmk->OK

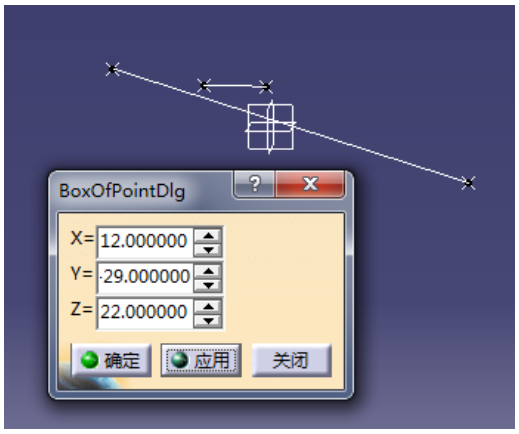
窗口->Open runtime Window-输入”Cnext”->回车

开始->机械设计->零件设计->装配a->连续划线2

输入一组 xyz ->点击应用 ->再输入一组不同于上次输入的 xyz->点击应用

（注意：在两种情况下程序会挂掉。1.运行环境不是 CATPart；2.两次输入的坐标值相同）

有图有真相*^_^*：



6. 实例：选点画线

6.1 创建命令（Command）

和以前学习的创建 Command 的方法一样：

单击：文件->Add CAAV5 Itrem->CATIA resource->Command

在弹出 Insert Command 对话框中输入和选择：

Command name：FourCmd（注意：此处名字必须和 BITAssemAddin.cpp 中 new BITAssemHeader() 中的倒数第二个参数相同。BITAssemAdding.cpp 是根据这个参数运行 FiveCmd 的）

Command style 选择 Basic Command

Command Mode 选择 Exclusive（默认值）

点击 OK 键后生成两个文件 FiveCmd. cpp 和 FiveCmd. h

6.2 创建对话框（Dialog）

6.2.1 生成文件

单击：文件->Add CAAV5 Itrem->CATIA resource->Dialog

在弹出 Insert Dialog 对话框中输入和选择：

Dialog object type：Dialog box（默认值）

Dialog Class name：SelectToPloyLineDlg

点击 OK 键后生成三个文件 SelectToPloyLineDlg.cpp 和 SelectToPloyLineDlg.h 和 SelectToPloyLineDlg.CATDlg

6.2.2 编辑对话框

在对话框中添加三个 PushButton 控件：

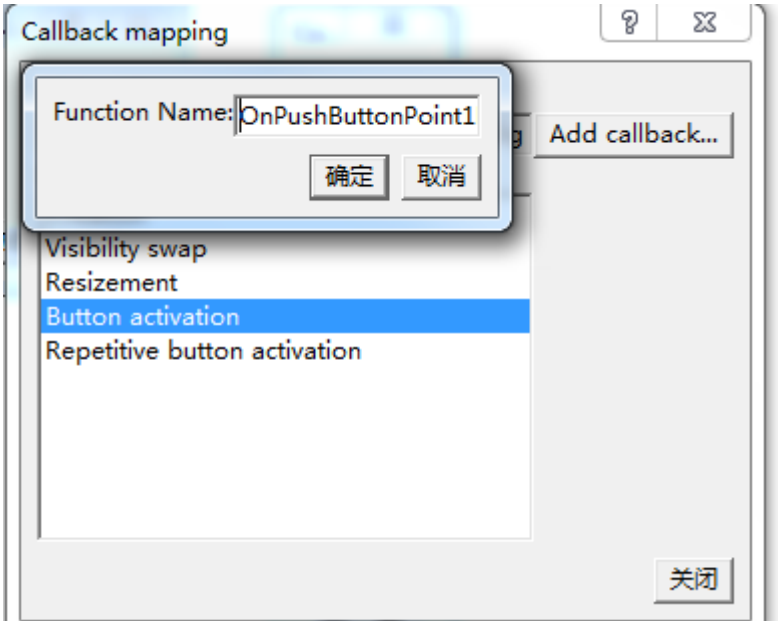
Title 分别为：“点 1”、“点 2”、“划线”

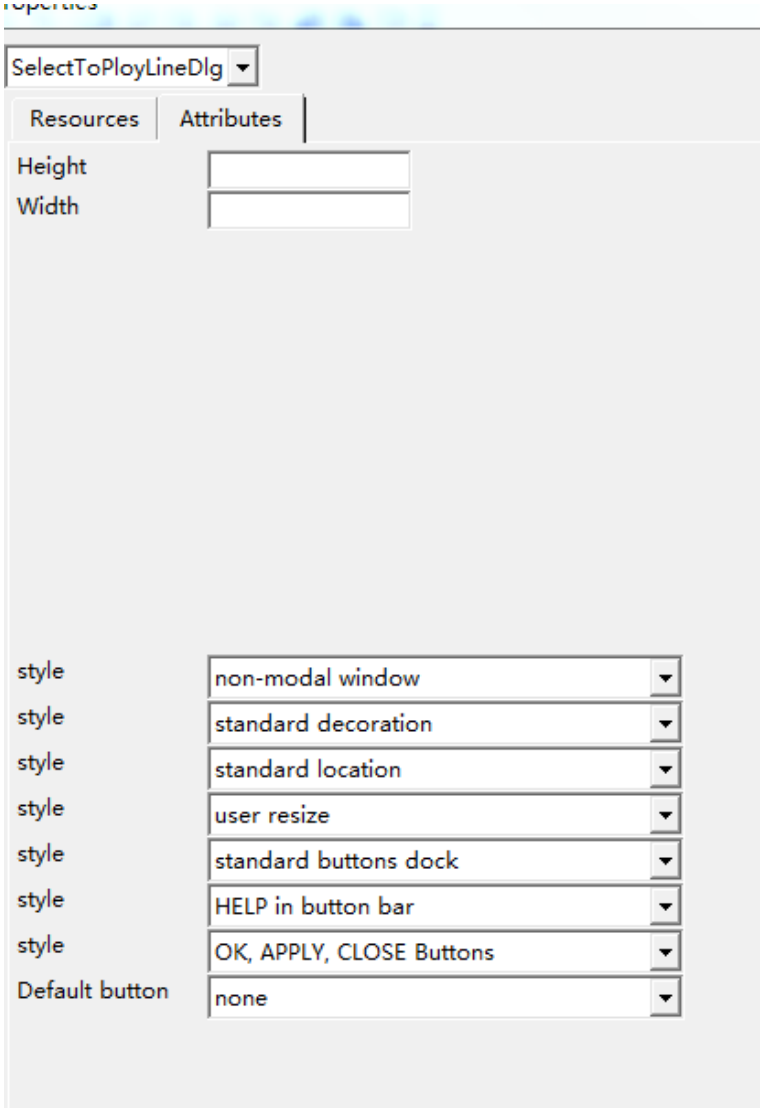
Name分别为： PushButtonPoint1、PushButtonPoint2、PushButtonPloyLine

在对话框的空白处击鼠标右键，选择 property，进入修改对话框属性界面，attributes 选项卡中：

在第 7 行 Style 中选择 OK， APPLY ， CLOSE Buttons

给每个 PushBUtton 添加回调函数。方法为：右击 PushButton 控件，进入 Callback Mapping 对话框：选择 Button activation，再点击 Add CallBack，再点击确定。这样就添加了 Callback 函数。也就是说在 SelectToPloyLineDlg.h 中添加了对应函数的声明，在 SelectToPloyLineDlg.cpp 中添加了对应函数的实现代码框架。


Callback mapping
Function Name:
OnPushButtonPoint1
确定
取消
Add callback...
Visibility swap
Resizement
Button activation
Repetitive button activation
关闭


SelectToPloyLineDlg
Resources
Attributes
Height
Width
style
non-modal window
style
standard decoration
style
standard location
style
user resize
style
standard buttons dock
style
HELP in button bar
style
OK, APPLY, CLOSE Buttons
Default button
none

6.2.3 再次创建 Command（SelectToPloyLineCmd）

单击：文件->Add CAAV5 Itrem->CATIA resource->Command

在弹出 Insert Command 对话框中输入和选择：

Command name：SelectToPloyLineCmd；Command style 选择 StateChat Command

Command Mode 选择 Exclusive（默认值）

点击 OK 键后生成两个文件 SelectToPloyLineCmd. cpp 和 SelectToPloyLineCmd. h

6.3 添加代码

6.3.1 将 FiveCmd 和 BITAssemAddin 关联

打开 BITAssemAddin.cpp 文件找到相关的那一行

方法如同上边所述，不在重复，一定记得

```
new BITAssemHeader("BITAssemFiveHdr", "BITAssemCmds", "FiveCmd", (void*)NULL);
```

中的第三个参数，务必要和Command关联。

（详细见一些注意事项中的源代码打开方式，仔细看源代码去。）

6.3.2 将 FiveCmd 和 SelectToPloyLineDlg 关联

在 FiveCmd.cpp 中找到

```
CATStatusChangeRC FiveCmd::Activate( CATCommand * iFromClient, CATNotification * iEvtDat){}
```

在大括号中添加：

```
SelectToPloyLineDlg * pDlg = new SelectToPloyLineDlg();
pDlg->Build();
pDlg->SetVisibility(CATDlgShow);
RequestDelayedDestruction();
```

上述程序在前边已经使用过多次。

添加头文件：

```
#include"SelectToPloyLineDlg.h"
```

6.3.3 修改 SelectToPloyLineDlg

打开SelectToPloyLineDlg.cpp

找到函数：

```
void SelectToPloyLineDlg::OnPushButtonPoint1PushBActivateNotification(CATCommand* cmd, CATNotification* evt, CATCommandClientData data){}
```

在大括号中添加代码：

```
m_Index = 1;//单击第一个按钮时设置变量值为，便于后面判断
this->SetVisibility(CATDlgHide);
SelectToPloyLineCmd * pCmd = new SelectToPloyLineCmd(this);
```

再找到函数：

```
void SelectToPloyLineDlg::OnPushButtonPoint2PushBActivateNotification(CATCommand* cmd, CATNotification* evt, CATCommandClientData data){}
```

在大括号中添加代码：

```
m_Index = 2;//单击第二个按钮时设置变量值为，便于后面判断
this->SetVisibility(CATDlgHide);
SelectToPloyLineCmd* pCmd = new SelectToPloyLineCmd(this);
```

再找到函数：

```
void SelectToPloyLineDlg::OnPushButtonPloyLinePushBActivateNotification(CATCommand* cmd, CATNotification* evt, CATCommandClientData data)
{}
```

在大括号中添加：

```
/////////设置代码工厂
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();
CATDocument *pDoc = pEditor->GetDocument();
CATIContainerOfDocument_var spConODocs = pDoc;
CATIContainer  * _pContainer = NULL;
HRESULT hr = spConODocs->GetSpecContainer(_pContainer);
```

```
    CATIGSMFactory_var spGSMFactory = NULL_var;
    spGSMFactory = _pContainer;
    //////判断是否为同一个点
    if(m_spObjFst==m_spObjSnd)
    {
        CATDlgNotify * _OpenNotify = new CATDlgNotify(this, "", CATDlgNfyOK);
        _OpenNotify->SetText("两个点相同不能划线！\n重新选点。");
        _OpenNotify->SetVisibility(CATDlgShow);
    }else
    {
        ////////////如果不是同一点，划线并将线显示出来
        CATIGSMLinePtPt_var spLine1 = spGSMFactory->CreateLine(m_spObjFst,m_spObjSnd);
        CATISpecObject_var spSpecLine1 = spLine1;
        spSpecLine1->Update();
        CATIGSMProceduralView_var spCurObj = spLine1;
        spCurObj->InsertInProceduralView();
    }
```

在后边在添加一个用来传递选中的内容的函数：SetSelectObj（）

代码如下：

```
void SelectToPloyLineDlg::SetSelectObj(CATISpecObject_var ispObj)
{
    ////这个函数在SelectToPloyLineCmd中调用，用来传递ispObj
    ////这个函数有点Class的味道
    if ( 1 == m_Index )
    {
        m_spObjFst = ispObj;
    }
        //当前选择的为第二个对象
```

```
    elseif ( 2 == m_Index )
    {
        m_spObjSnd = ispObj;
    }
}
```

同样，光有代码没有声明和头文件是不能运行的

在 SelectToPloyLineDlg. cpp 中在添加头文件：

```
#include"SelectToPloyLineCmd.h"
```

不要忘记在添加关闭对话框的命令：

在SelectToPloyLineDlg.cpp中找到带有 “CLOSE”字样的函数，在大括号中添加：

```
    this->SetVisibility(CATDlgHide);
    this->RequestDelayedDestruction();
```

所有程序都是这样

前提是在编辑对话框的时候要添加回调函数，否则不会有CLOSE字样的函数出现。

打开SelectToPloyLineDlg.h

添加头文件：

```
#include"CATMathPoint.h"
#include"CATISO.h"
#include"CATFrmEditor.h"
#include"CATDocument.h"
#include"CATIContainerOfDocument.h"
#include"CATIGSMFactory.h"
#include"iostream.h"
#include"CATIPrtFactory.h"
#include"CATICkeParmFactory.h"
#include"CATIGSMPoint.h"
#include"CATIGSMProceduralView.h"
#include"CATIGSMFactory.h"
```

```
#include"CATIDescendants.h"
#include"CATIGSMLinePtPt.h"
#include"CATPathElementAgent.h"
#include"CATPoint.h"
#include"CATISpecObject.h"
#include"SelectToPloyLineDlg.h"
#include"CATDlgNotify.h"
#include"FiveCmd.h"
```

添加声明：

```
public:
    int  m_Index;
    CATISpecObject_var m_spObjFst;
    CATISpecObject_var m_spObjSnd;
    CATISpecObject_var     SpecPoint1;
    CATISpecObject_var     SpecPoint2;
    void SelectToPloyLineDlg::SetSelectObj(CATISpecObject_var );
```

6.3.4 修改 SelectToPloyLineCmd

首先修改 SelectToPloyLineCmd.h 文件

添加头文件：

```
#include"CATMathPoint.h"
#include"CATISO.h"
#include"CATFrmEditor.h"
#include"CATDocument.h"
#include"CATIContainerOfDocument.h"
#include"CATIGSMFactory.h"
#include"iostream.h"
#include"CATIPrtFactory.h"
```

```
#include"CATICkeParmFactory.h"
#include"CATIGSMPoint.h"
#include"CATIGSMProceduralView.h"
#include"CATIGSMFactory.h"
#include"CATIDescendants.h"
#include"CATIGSMLinePtPt.h"
#include"CATPathElementAgent.h"
#include"CATPoint.h"
#include"CATISpecObject.h"
#include"SelectToPloyLineDlg.h"
#include"CATDlgNotify.h"
```

添加声明：在构造函数SelectToPloyLineCmd();之后添加代码：

```
  SelectToPloyLineCmd(SelectToPloyLineDlg* pFather); //将构造函数重构pFather是用来传递表明当前所对应的对话框的指针
    CATPathElementAgent * m_pPathAgent;
    CATIGSMPoint_var    spPoint1;
    CATIGSMPoint_var    spPoint2;
     CATDlgNotify   * _OpenNotify;
    SelectToPloyLineDlg * m_pFather;  //操作对话框
```

打开 SelectToPloyLineCmd.cpp 文件

在声明中重构了构造函数，那么就一定要有重构的构造函数的实现：

在原始的构造函数 SelectToPloyLineCmd()结束后添加

```
SelectToPloyLineCmd::SelectToPloyLineCmd(SelectToPloyLineDlg* pFather) :
CATStateCommand ("SelectToPloyLineCmd", CATDlgEngOneShot, CATCommandModeExclusive)
//  Valid states are CATDlgEngOneShot and CATDlgEngRepeat
,m_pPathAgent(NULL)
,m_pFather(pFather)                    //对话框将指针传到这儿，使m_pFather赋值
{
}
```

找到

```
void SelectToPloyLineCmd::BuildGraph()
{.....}
```

将大括号中的内容替换为：

```
    m_pPathAgent = new CATPathElementAgent("SelectionAgent", NULL, NULL);
    m_pPathAgent->AddElementType(IID_CATISpecObject );//添加可以选择的对象类型，特征树上可以找到的都行。头文件#include "CATISpecObject.h"添加
    m_pPathAgent->SetBehavior(CATDlgEngRepeat|CATDlgEngWithCSO|
        CATDlgEngWithPrevaluation | CATDlgEngWithUndo);//对象的行为
    CATDialogState *stState = GetInitialState("stFirstPointId");
//定义对话框状态，初始状态。
    stState->AddDialogAgent(m_pPathAgent);
    CATDialogTransition *pTrasition = AddTransition(stState, NULL,
    //从初始状态到NULL
        IsOutputSetCondition (m_pPathAgent),//判断agent是否赋值
        Action((ActionMethod)&SelectToPloyLineCmd::ActionOne));
//执行action，在action内处理过程。
```

找到：

```
CATBoolean SelectToPloyLineCmd::ActionOne( void *data ){}
```

在大括号内添加：

```
//-----------------------------------------------------------------------------
CATBoolean SelectToPloyLineCmd::ActionOne( void *data )
{

    if (NULL == m_pPathAgent  )    //使用之前先判断是否为空，要不然程序会挂掉；
    {
        return FALSE;
    }
////CATPathElement存放路径，CATPathElementAgent中获得路径的方法GetValue()。
```

```
        ////获得当前选择对象的路径Product1/Part.1/零件几何体/点
        CATPathElement* pPath = m_pPathAgent->GetValue();

        if ( NULL == pPath )
        {
            return FALSE;
        }
        CATISpecObject_var spObj = NULL_var;//CATISpecObject操作特征模型的接口
        pPath->Search(IID_CATISpecObject, (void**)&spObj );//从路径中获得当前对象非常关键
        if ( NULL_var != spObj )//测试，对象不空时传给对话框
        {
            //将该对象返回给对话框
            m_pFather->SetVisibility(CATDlgShow);      //显示对话框
            ////SelectToPloyLineDlg中自定义的函数，这个函数做的很巧妙。
            ////把选定的对象存放到对应的m_spObjFst和m_spObjSnd中，将值返回给
SelectToPloyLineDlg
            m_pFather->SetSelectObj(spObj );
        }
        m_pPathAgent->InitializeAcquisition();
        return TRUE;
```

6.4 运行

CAAV5 Workspace->Create/Update Runtime View->Ok

生成->mkmk->OK

窗口->Open runtime Window-输入”Cnext”->回车

打开一个有两个点的文件比如 TwoPoint.CATPart，或者现场画两个点，

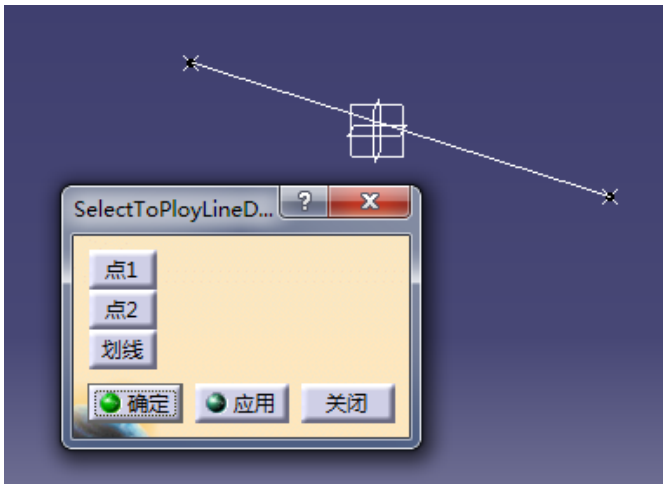
装配 a->选点划线

弹出的对话框中：

点 1->选择一个点->点 2->选择另一个点->划线

（注意：运行环境不是 CATPart；时程序会挂掉）

有图有真相：



7.程序实例

以下为核心代码说明。

7.1 BuildGraph 函数定义

```
void CAADegCreateCircleCmd::BuildGraph()
{
// 1- Creates dialog agents 创建个选择代理（其实可以用个选择代理，每次用完之后重置即可）
//  1-1 选中第一个点
  _daPathEltPoint1 = new CATPathElementAgent("SelPoint1");//选择代理的名称ID
  _daPathEltPoint1->AddElementType(IID_CATIGSMPoint);//只能选择CATIGSMPoint
// 设置行为，这里是选中高亮，不能重复等
```

```
  _daPathEltPoint1->SetBehavior(CATDlgEngWithPSOHSO |
                                CATDlgEngWithPrevaluation |
                                CATDlgEngWithUndo );
  AddCSOClient(_daPathEltPoint1);
//  1-2  选中第个点
  _daPathEltPoint2 = new CATPathElementAgent("SelPoint2");
  _daPathEltPoint2->AddElementType(IID_CATIGSMPoint);//只能选择CATIGSMPoint
  _daPathEltPoint2->SetBehavior(CATDlgEngWithPSOHSO |
                                CATDlgEngWithPrevaluation |
                                CATDlgEngWithUndo );
  AddCSOClient(_daPathEltPoint2);
//  1-3 选中第个点

  _daPathEltPoint3 = new CATPathElementAgent("SelPoint3");
  _daPathEltPoint3->AddElementType(IID_CATIGSMPoint);//只能选择CATIGSMPoint
  _daPathEltPoint3->SetBehavior(CATDlgEngWithPSOHSO |
                                CATDlgEngWithPrevaluation |
                                CATDlgEngWithUndo );
  AddCSOClient(_daPathEltPoint3);
// 2-  创建状态 状态和选择代理关联
  CATDialogState *stGetpoint1 = GetInitialState("stGetPlane1");//初始状态用
GetInitialState，当然要设置状态名称ID
  stGetpoint1->AddDialogAgent(_daPathEltPoint1);// 状态和选择代理关联

  CATDialogState *stGetpoint2 = AddDialogState("stGetPlane2");//后续状态用
AddDialogState
  stGetpoint2->AddDialogAgent(_daPathEltPoint2);// 状态和选择代理关联（其实多个状
态可以关联同一个代理）

  CATDialogState *stGetpoint3 = AddDialogState("stGetPlane3");
  stGetpoint3->AddDialogAgent(_daPathEltPoint3);// 状态和选择代理关联
// 3- Defines transitions 定义状态转移
  CATDialogTransition *pFirstTransition = AddTransition
  (
     stGetpoint1,//状态（或者称之为初始状态）
     stGetpoint2,//状态
     IsOutputSetCondition(_daPathEltPoint1),  //转移条件
     Action((ActionMethod) & CAADegCreateCircleCmd::GetSelect1)//转移条件达成时要
执行的函数
  );//选择点

  CATDialogTransition *pSecondTransition = AddTransition
```

```
  (
     stGetpoint2,
     stGetpoint3,
     IsOutputSetCondition(_daPathEltPoint2),
     Action((ActionMethod) & CAADegCreateCircleCmd::GetSelect2)
  );//选择点


  CATDialogTransition *pThirdTransition = AddTransition
  (
     stGetpoint3,
     NULL,//终止状态就是NULL（这儿若是stGetpoint3 那么就是一个循环）
     IsOutputSetCondition(_daPathEltPoint3),
     Action((ActionMethod) & CAADegCreateCircleCmd::CreatePlane)
  );//选择点同时创建平面
}
```

7.2 GetSelect1 和 GetSelect2 函数定义

```
CATBoolean CAADegCreateCircleCmd::GetSelect1(void *iDummy)
{
  CATBoolean rc = FALSE ;

  CATPathElement * pPathModel = _daPathEltPoint1->GetValue();//_daPathEltPoint1中
存放了选择的对象路径（对象表）
CATBaseUnknown * obj = NULL ;
if ( (NULL != pPathModel) && (0 != pPathModel->GetSize()) )
   {
// Retrieves the leaf of the path这儿的obj中存放的就是一个CATIGSMPlane
     obj = (*pPathModel)[pPathModel->GetSize()-1];//最后一个对象就是我们要找的点
  }
if ( NULL != obj )
  {
     spSpecPoint1 = NULL_var;
    HRESULT hr = obj->QueryInterface(IID_CATISpecObject, (void**)&spSpecPoint1);//
获取了spSpecPoint1
   }
return TRUE;
}
```

GetSelect2和GetSelect1大致一样，不在赘述。

7.3 CreatePlane 函数定义

```
CATBoolean CAADegCreateCircleCmd::CreatePlane(void *iDummy)
{
  CATBoolean rc = FALSE ;

  CATPathElement * pPathModel = _daPathEltPoint3->GetValue();
CATBaseUnknown * obj = NULL ;
if ( (NULL != pPathModel) && (0 != pPathModel->GetSize()) )
  {
// Retrieves the leaf of the path这儿的obj中存放的就是一个CATIGSMPlane
     obj = (*pPathModel)[pPathModel->GetSize()-1];
  }
if ( NULL != obj )
  {
     spSpecPoint3 = NULL_var;
    HRESULT hr = obj->QueryInterface(IID_CATISpecObject, (void**)&spSpecPoint3);//
获取了spSpecPoint3
    }
//三点画圆 首先要将CATIGSMFactory工厂赋值
    CATIGSMFactory_var spGSMFactory = NULL_var;
    spGSMFactory = _pContainer;
 CATIGSMPlane3Points_var  _3PointPlane =
spGSMFactory->CreatePlane(spSpecPoint1,spSpecPoint2,spSpecPoint3);
    CATISpecObject_var spSpec = _3PointPlane;
//将面显示在视图中并出现在结构树中
    spSpec->Update();
    CATIGSMProceduralView_var spCurObj = _3PointPlane;
    spCurObj->InsertInProceduralView();
return TRUE;
}
```

8.总结

学习过程中必须注意以下几点：

A.在 CAA 开发过程中很大一部分代码是自动生成的。一般不要轻易修改自动生成的代码。

B. 文件->add CAA V5 Item-> CATIA resource->dialog方式添加的对话框。.h文件中已经声明

了所有控件的指针，不需要自己声明。自己只需要声明：CATDlgNotify

C. 所有用到的函数都可以在 CAA V5 Help Viewer 中查到。打开方式为：社区->Help CAA V5。

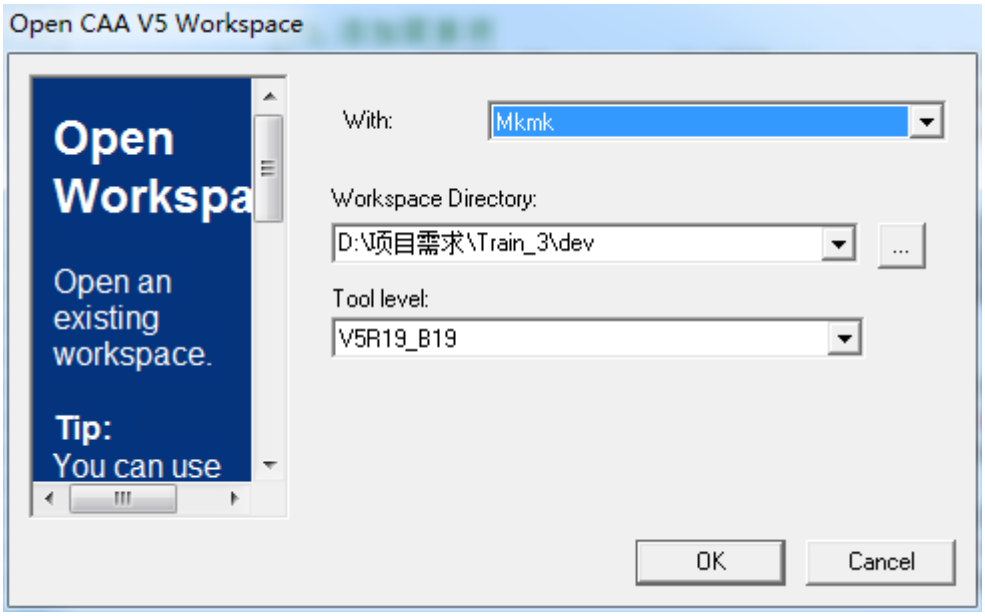
比如要查找与类 CATDlgEditor 相关的函数就在 Index 选项卡中输入 CATDlgEditor。

D. 与对话框中控件的操作相关的文档在百科全书里边查找：

User Interface ->WInTop Dialogs

E. 刚开始编程程序出错可能的原因有：头文件是否添加完全、变量是否声明、当界面更改时是否更新

F.所有的 Train 附带的 Case 都能使用，方法为：打开 Visual Studio->文件->Open CAAV5 WorkSpace 弹出对话框：



选择 WorkSpace Directory：xxxx\Train_x\xx

其他选项按照上图选择。点击 OK 打开即可查看源代码。

若要运行，则首先需要 Locate prerequisite

G. 很多东西需要不断的尝试，探索，所以遇到难题不要灰心。

H. 有些头文件放在.h 文件和.cpp 文件里是完全不一样的情况，所以当把头文件的位置改变，可能会导致出错。

I. .CATDlg 文件的编辑器很脆弱，一不小心就会挂掉，所以在编辑对话框的时候要仔细、小心。否则只能重启电脑了（系统内部有个服务会崩溃掉，重启之后又会打开）。

J. 百科全书是一系列的脱机网页文件：目录为：

安装目录\B19\CAAV5HomePage