# 基于OCC的M3d系统设计

## 2022年2月28日星期一：M3d系统模型Modelica代码生成

1. 参数表，可以允许子装配体的参数作为映射参数

子装配体的参数 是 其 参数表 中的参数名！！！

1. 多体常用Parts的参数设置问题

* World

inner Modelica.Mechanics.MultiBody.World world ( gravityType=Modelica.Mechanics.MultiBody.Types.GravityTypes.UniformGravity,n={0,0,-1});默认g的方向-Z；可以通过设置和读取ParameterValue表改变设置；world一般加属性inner限制；另外加上几个：

animateWorld=false,

animateGravity=false,

enableAnimation=false

* Revolute

Modelica.Mechanics.MultiBody.Joints.Revolute rev (n={0,1,0},useAxisFlange=true, animation=false, phi(fixed=true),w(fixed=true));n为旋转轴矢量，根据frame\_a到frame\_b自动确定；如果使用flange如增加阻尼，则设置useAxisFlange=true；（一般设为true没关系）

Phi和w的(fixed=true)设置去掉似乎没影响

可以在关注“变量”（不是参数）var时，固定一个名字 fixed，则，则增加 var(fixed=true)

* Body

Modelica.Mechanics.MultiBody.Parts.Body body(m=1.0, r\_CM={0.5,0,-0.1}, animation=false, useQuaternions = false)；只考虑平动时，否则用BodyShape吧。用角位移即变换矩阵作为多体的位姿表示，不采用四元数法

注：如果采用四元数法，直接用gp\_Trsf的SetTransformation (const gp\_Quaternion& R, const gp\_Vec& T);应该也可行。Frames.Quaternions.Orientation Q: "Orientation type defining rotation from a frame 1 into a frame 2 with quaternions {p1,p2,p3,p0}"

* BodyShape同样，不给I惯量积数据，可以看作平移运动的质点

Modelica.Mechanics.MultiBody.Parts.BodyShape b3(

r={0,0.414,-0.155},//frame\_a到frame\_b的矢量，自动计算

r\_CM={0.064,-0.034,0},//frame\_a到质心的矢量,自动计算

m=26.4,//质量，自动计算

I\_11=0.279,//惯量积矩阵I，自动计算,坐标原点在质心，方向与Frame\_a平行的系统惯量积

I\_22=0.245,

I\_33=0.413,

I\_21=-0.070,

shapeType="modelica://Modelica/Resources/Data/Shapes/RobotR3/b3.dxf",

r\_shape={0,0,0},

lengthDirection={1,0,0},

widthDirection={0,1,0},

length=0.15,

width=0.15,

height=0.15,

animation=animation,//false

useQuaternions = false,//注意！不使用四元素表达状态

animateSphere=false,

color={255,0,0})

1. 零件的CyberInfo界面增加功能：

对于设置“关注”特别时关注名的**参数**，选择后，自动填充参数值. 简单一点，增加一个“自动计算”来自动填充设置关注参数值。只需要调用函数 CyberInfo的成员函数UpdateFocusParameterValues()，更新CyberInfo的界面。

特定的关注参数名字：（除了一些特别设置外）

animation：直接设置false

nX: (1,0,0); n\_X: (-1, 0, 0); ……

ra\_b: frame\_a到frame\_b的矢量；rb\_a：frame\_b到frame\_a的矢量

r\_CM：frame\_a到质心的矢量

//r: frame\_a到frame\_b的矢量

I\_11~I\_33: 自动计算的惯量积矩阵元素

m: 质量

1. 物性参数的计算

ra\_b的计算，直接由frame\_b – frame\_a

r\_CM的计算：由CM质心的坐标 cm – ra

质量计算：体积\*密度

I矩阵计算：先要将坐标系（原点在CM，轴与Frame\_a对齐）对齐全局坐标系，再计算。方法GProp\_GProps:: MatrixOfInertia()，惯性积是以CM为坐标，轴与全局坐标系XYZ对齐的！

## 2022年2月22日星期二

* 零部件图示？
* 零部件改名
* 仿真结果CSV读取与动画实现：两种方式(QTimer与QThread)均可

## 2022年2月18日星期五

M3d参数化设计思考：基于体素和三维约束的参数化零件设计，装配设计；基于程序库的机械零件库参数化设计（典型的零件：轴、齿轮、螺纹件、蜗轮蜗杆、连杆、凸轮，等等）

（先不考虑自由曲面）

* 零件设计

多个体素进行布尔运算（四种）：球、长方体、圆柱体、拉伸体、旋转体等…，不同的体素带有不同的参数表；拉伸或旋转体素的截面轮廓生成，不采用通用的交互建模，可以采用脚本描述，先考虑机械标准件所需的特征，来考虑轮廓线的描述；

体素之间建立几何约束；

可以导入外部几何作为体素，与其他体素保持约束关系，但不带有参数表；

显示并修改尺寸（参数表），根据体素参数和体素间约束，及其之间定义的布尔操作，调用求解器获得最终的几何体；

体素（包括参数表）链表和约束关系存于文档；

* 装配设计

零件尺寸驱动，装配体参数化改动；

具有代数环的装配，如四连杆机构；

装配体的调姿参数化；

* 零（部）件库

建立零件模板，包括零件几何和完备的参数约束表，可以实现系列尺寸（关键尺寸）的驱动；

建立零件的数据表，附着在模板上，形成零建族。用户可以选择其中的任意一组数据驱动生成实例零件；

部件（如蜗轮蜗杆）由零件参数化和装配约束（接口）和调姿约束共同作用即可完成。

## 2022年2月11日星期五

后续计划：五一之前，推出第一个运行的实例，单摆；五月份15日：六自由度机械手

梁清清：论文、Modelica代码生成，前面的代码完善；

邓尔文：论文，前面的代码完善；

唐思豪：调姿、连接完成；零件、装配和调姿的参数化研究

余鸿儒：仿真求解与后处理；机械柔性多体建模方法研究（Flexible Multibody Systems in Modelica，Modelling and simulation of rigid and flexible multibody systems in Modelica）

## 2022年2月8日星期二

工作总结：

* 子装配体的装配与调姿调试、测试
* Bug的修复：装配与调姿混用，Modelica信息提取，……
* 一些小功能添加：零部件图示，隐藏/显示便于选择，……
* 模型编译与仿真实现思路：

（Mworks二次开发包为32位，而且开发工具版本不匹配；暂时用OpenModelica）

方案一：OpenModelica或其他商用平台的Script，生成\*.mos，直接命令行调用，生成结果文件\*.csv，读取csv文件获得变量的时间序列。

方案二：采用OpenModelica的PyModelica，希望可以解决返回值问题，比如获得组件的信息。

方案三：采用MWorks或Dymola商业软件的脚本或Python外调方式来求解并获得仿真结果。

组件的动画及后处理：首先，改变显示模式：拆开现有的部件选择与显示模式，全部爆炸成为各个零件（关闭仿真和后处理模式时恢复原来的显示）；接着，隐藏现在的左边停靠窗口，显示后处理停靠窗口（关注的参数和变量）；动画采用线程驱动每个组件的位姿

## Modelica.Mechanics.MultiBody.Parts.Body

每个非固定的组件在t时刻的位姿：SetLocation(gp\_Trsf);

由该组件的frame\_a的位置和姿态来确定，即由原来ti-1的位姿到新的时刻ti的位姿的变换矩阵来确定

* The **position vector** frame\_a.r\_0 from the origin of the world frame to the origin of frame\_a of the body, resolved in the world frame and the **absolute velocity** v\_0 of the origin of frame\_a, resolved in the world frame (= der(frame\_a.r\_0)).
* If parameter **useQuaternions** in the "Advanced" menu is **true** (this is the default), then **4 quaternions** are potential states. Additionally, the coordinates of the absolute angular velocity vector of the body are 3 potential states.  
  If **useQuaternions** in the "Advanced" menu is **false**, then **3 angles** and the derivatives of these angles are potential states. The orientation of frame\_a is computed by rotating the world frame along the axes defined in parameter vector "sequence\_angleStates" (default = {1,2,3}, i.e., the Cardan angle sequence) around the angles used as potential states. For example, the default is to rotate the x-axis of the world frame around angles[1], the new y-axis around angles[2] and the new z-axis around angles[3], arriving at frame\_a.

接口frame\_a的位置为 frame\_a.r\_0【3】；负责平移

Frame\_a的姿态为：frame\_a.R ---- Orientation，一个变换矩阵T[3,3]只旋转

record Orientation "Orientation object defining rotation from a frame 1 into a frame 2"

extends Modelica.Icons.Record;

Real T[3, 3] "Transformation matrix from world frame to local frame";

SI.AngularVelocity w[3];

…

end Orientation;

## 2022年1月21日星期五

Modelica模型库的结构：模型树读取效率？（后台加载与前台结合？）

模型信息（参数，部件，变量和连接）的解析？参考modelicaSpecification3.2语法定义

如何查找模型全名和模型文件Mo的路径？高效而全面？

并行加载模型库？

库的序列化与反序列化

QThread : run 系统启动后台 (先反序列化，如果没有反序列化文件，则先序列化产生)

{

//获得设置的加载路径

list<string> molibs = GetAllLibPaths();

For\_each aMoFile in molibs

ParallelLoad(aMoFile, ModelicaModel\* & theModel, hash\_map<string, ModelicaModel\*>& theModelMap);

//加入list<ModelicaModel\*> lstAllModels

lstAllModels.pushback(theModel);

}

其中 ModelicaModel的结构大约：

Class ModelicaModel

{

String sName; //模型简名

String sType; //class, model, connector, block, …

vector<ModelicaModel\*> lstExtends; //基类

vector<ModelicaModel\*> vImports; //导入类

Vector<myParameter\*> vParameters; //参数

Vector<myOutputVar\*> vOutVars;//简短类变量，一般变量（Real, int,…），除了组件

map<string, ModelicaModel\*> mapParts; //组件（类型为class, model, block, connector的变量）；

Vector<connector\*> vConnectors; //接口类组件:只存储类型和名字

ModelicaModel\* pParent; //父模型：可以递归向上，获得模型的全名

Vector<ModelicaModel\*> vEnbeddedModels; //嵌套类

}；

一个重要函数：通过全名在加载的模型库中查找模型

ModelicaModel\* LookupModel(list<ModelicaModel\*> lstAllModels, string sMdlFullName);

## 2022年1月14日星期五

考虑Modelica信息读取

1. 读取mo的模型树信息；
2. 读取选中模型的信息：模型类型；参数信息；变量信息；接口信息

## 2022年1月12日星期三

1. 体素建模
2. 组件属性 : 存储在（1-1-2）
3. 接口的坐标，装配变换以后可能出错
4. 接口需要增加一个判断：是什么运动副？

## 2022年元月3日

装配环境下的可选择实体（ColoredShape）: 第一层组件（不含接口）、第一层组件的接口

加入到 myDisplayList

接口类型零件的主接口：参数表保存重复增加

接口坐标系的显示问题

接口的新位姿 Ax3’ = Ax3\*组件的变换矩阵 M

在Connector对象增加 ShowCoordinate(context, gp\_Trsf)；方法，以显示接口坐标系

## 2021年12月31日星期五

寒假前的计划（明年开学前）：完成全部建模工作

我的计划：完成数据结构重整；体素建模；考虑或实现Modelica模型库的结构读取以及信息读取；接口类型零件的主接口：参数表保存重复增加

邓尔文：零件库、装配树的图标显示；装配参数表设置；装配树功能完善；

梁清清：Modelica代码生成与测试；组件属性设置

唐思豪：装配体的调姿、自由体的调姿；连接线增加

## 2021年12月29日星期三

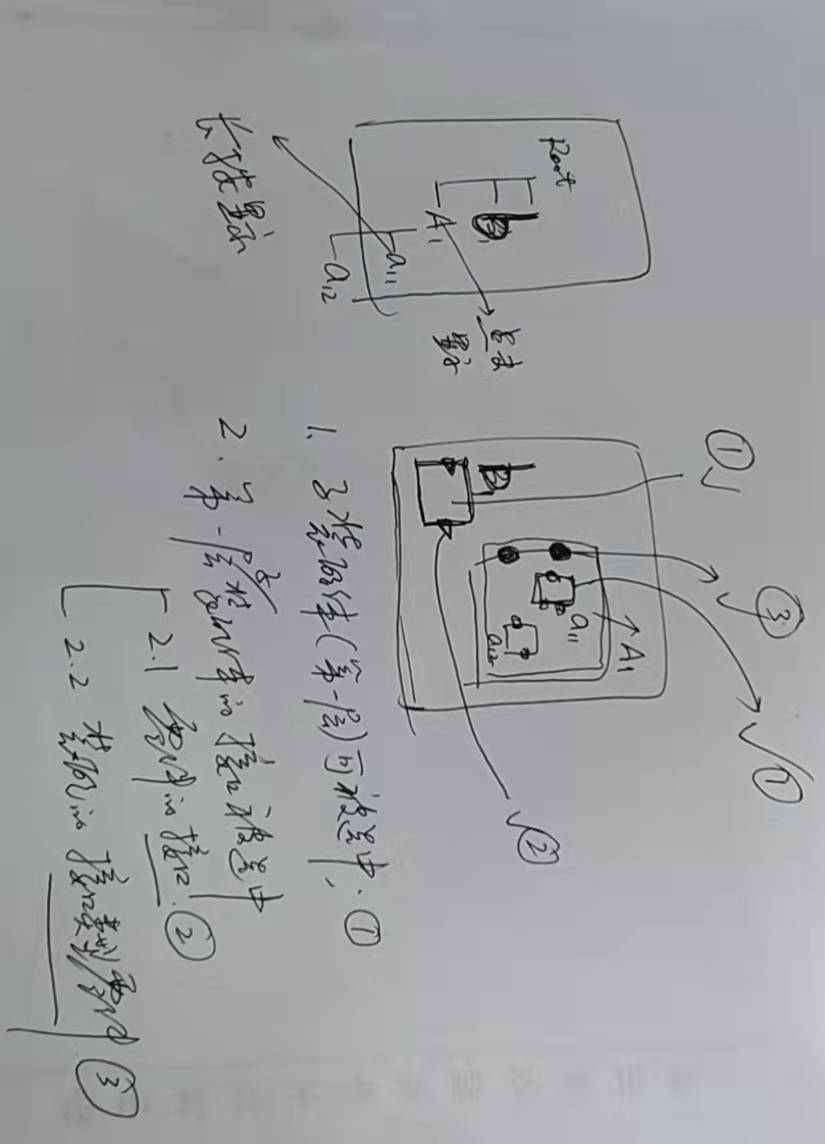
数据结构调整：

删去原来的M3d\_DS体系，只保留Document和Component

Document负责文件-文档相关

Component负责组件（组合体）相关逻辑操作，一个Component可能是多个Document组合

零部件及其接口的选择方式：



组件位姿的存储与改变：绝对位姿（导入和装配）V12、相对位姿（调姿）V15

相对位姿在原来的组件下增加一个子节点，Tag序号为1

组件的属性存储：int Array

存储在原来组件下，子节点Tag序号为 2

## 2021年12月24日星期五

调姿的问题：调姿的方式先改变，暂放Manipulator对象

在DocumentModel中增加调姿信息：

组件的变化量：map(string 组件名，相对位姿 transfDelta)，另存一个节点

组件的真正位置是：原来的m\_transf \* transfDelta;

可以归位操作，即将当前的 m\_transf \* transfDelta 保存，代替原来的 m\_transf，这样相对位姿又重新开始，即为单位阵

M3dMdl\_DS里面获得DocumentModel？

## 2021年12月21日星期二

增加了材料选择

内存泄漏问题：各个子控件只有new , 没有delete?

体素建模：参数化，引用的考虑

输入的合法性检查？[QRegExp解析 - lpx15312 - 博客园 (cnblogs.com)](https://www.cnblogs.com/lpxblog/p/6046740.html)

[(][,][,][)]（4.55, 5.4, 0）

匹配浮点数

^(-?\d+)(\.\d+)?$

点：^\((-?\d+)(\.\d+)?[,](-?\d+)(\.\d+)?[,](-?\d+)(\.\d+)?\)$

参数和变量的数据类型：增加

Del键删除？？？

数据结构的改进？

Mo结构与参数、变量和接口读取？

## 2021年12月16日星期四

1. 接口的选择问题解决
2. 完善了删除选中实体，
3. 约束对象需要增加一个序号！！

## 2021年12月13日星期一

问题：部件下面的子部件或零件的选择问题？

直接跟零部件的存储相关，需要考虑零件-装配的数据统一性问题

## 021年12月10日星期五

本周工作：

1. 调试bug：建议大家养成一个好习惯，一个功能要认真思考：方便易用、无bug，逻辑性；成败在于细节，花时间琢磨

装配树显示：文件还没有读，就显示了；更改：增加UpdateAssemblyTree();

装配下接口有的选不上？？？

基准点：面中心？？？

接口定义：名字和类型固定了？？？应该根据CyberInfo自适应啊！！！

更新了，接口添加只检查了个数，没有检查类型？？

装配树和零件库树没有焦点？需要增加点击选择部件，我已经增加了接口，参考现在的deleteComponent()函数

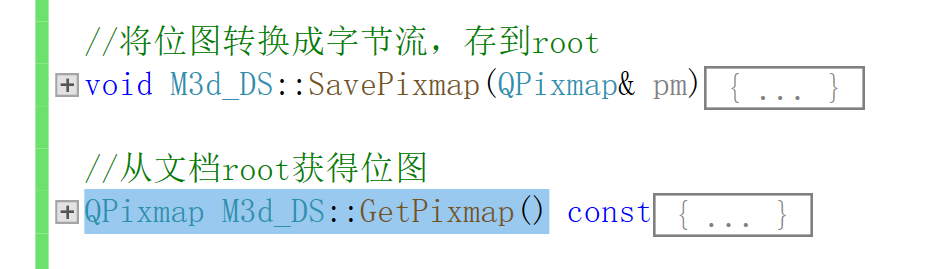
1. Del删除功能：只在编辑菜单下（不在工具条下），增加del键盘；删除选中的实体（参考点、参考面和接口），并做提示，拓扑部分实体不能删除

依赖实体的删除，留一个bug：

参考点---参考点—参考线---接口

零件---连接---装配接口约束

1. M3dMdl\_DS接口函数完成
2. 连接线增加与删除
3. 位图的存取

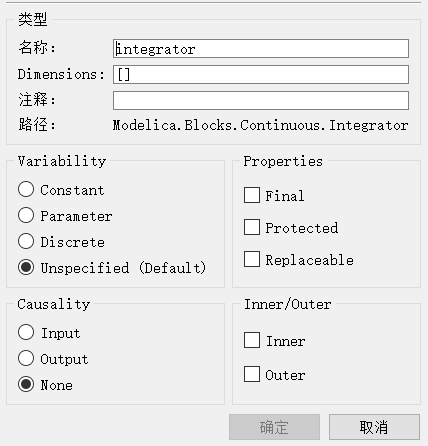


1. 装配树上部件的选择显示
2. 实体选择的亮显模式：采用动态AIS\_InteractiveObject显示和删除显示，鼠标按下显示，释放则删除
3. 简单体素造型与参数化支持？
4. 模型信息的读取？
5. 接口约束要不要增加一个形状显示？？

## 2021年12月9日星期四

装配模型的主要信息：

1. 组件、属性及其参数设置
   1. 组件信息：DocumentModel中，CompInfo
   2. 组件属性(限定词)：可以增加界面进行设置，参考MWorks或OpenModelica；目前这个还没支持，只在Component加了数据



* 1. 参数设置：一是有些从几何自动获取（初始位姿、质心、惯量等）；二是通过界面增加：（参数设置存储在m\_lstParamValues）

|  |  |  |
| --- | --- | --- |
| 映射组件参数名 | 参数名 | 设置值 |
| A.p1 | p1（可为空） | 100 |

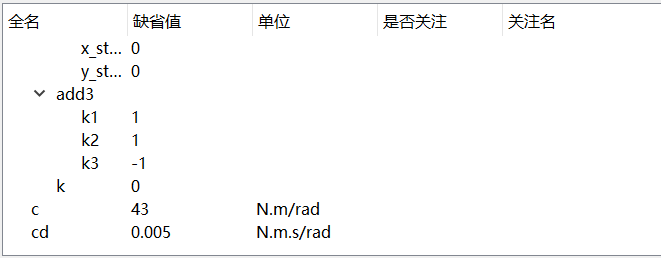
这样，生成Modelica代码时，在参数声明部分产生一句：

parameter 数据类型（查找A.p1的） p1 = 100;

而在A的描述中增加参数设置：

A a1(p1 = p1);

问题：参数的界面重新改进？



1. 需不需要树形？保持
2. 维度：缺省0维，即标量
3. 需要增加参数数据类型？
4. 连接（包括装配约束和连接线），连接约束将直接产生相关modelica代码

connect(A.c, B.c);//无Annotation

* 1. 装配约束，在m\_lstConnectConstraints中保存，
  2. 连接线约束，在m\_lstConnectCurves

## 2021年11月29日星期一

* 接口名字问题，需要改进：添加接口时都要设置好
* 接口的形状，Compound时如何拾取？
* 接口添加准则：
  + 对于零件就是一个接口，则添加接口是其子接口，多个；再增加一个复合接口
  + 否则，添加的接口就是复合接口，只有一个
* 零部件的位图存储与图标显示：在零件库的树视图显示

## 2021年11月26日星期五

进展：

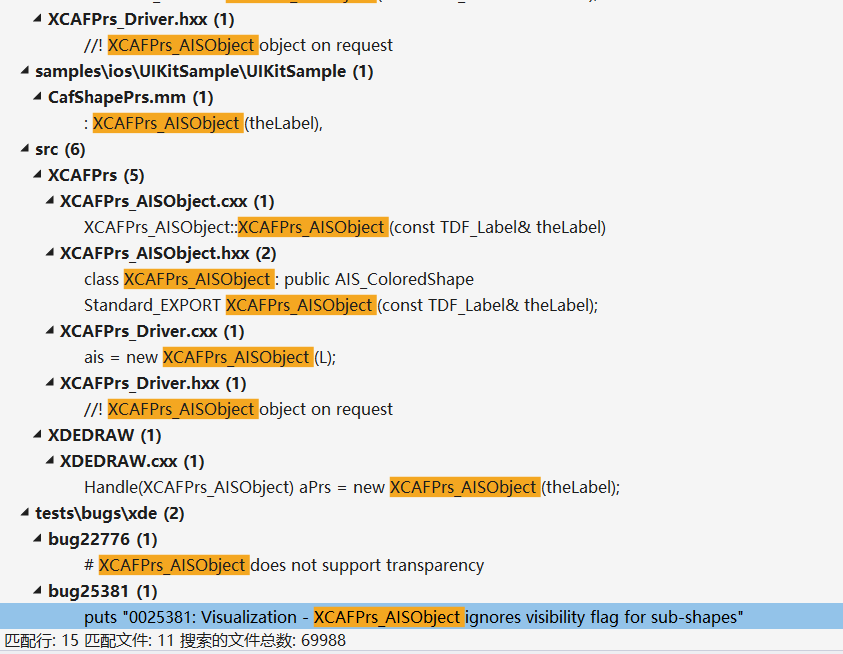
1. 鼠标移动事件，显示实体名；零件名，接口名
2. 关于接口零件的支持：
   1. 在CyberInfo输入，允许编辑全模型名，带Connector
   2. 添加接口，增加一个无名接口，就是本身，缺省位置在质心处
   3. 鼠标移动显示：零件下显示零件名；装配下显示组件名（后面的.不需要）
3. 点击名字，显示选中实体：一种方案如何由TopoDS\_Shape得到AIS\_Shape直接显示后选择上？；第二种，设计一个AIS表，记录，在小窗口处于焦点时，显示，否则不显示，动态
4. 对话框中选择实体的实现思路：
5. 点击某个编辑框（或列表框），激活 SelectCommand
6. 鼠标选择实体，在SelectCommand的LbuttonDown中
7. 将选中的实体名字，返回给被调用的QWidget，设置窗口名字
8. Command---SelectCommand(增加一个QWidget\*)调用的窗口
9. 装配测试：

现在子装配体的装配没法测试

需要增加单零件是接口的功能，因为接口，只能在零部件的第一层进行，即只能带一个点，如XXX.y

1. 关于带颜色的实体存储

继续原来的方案。装配体的颜色、透明度和材料等不好管理



## 2021年11月23日星期二

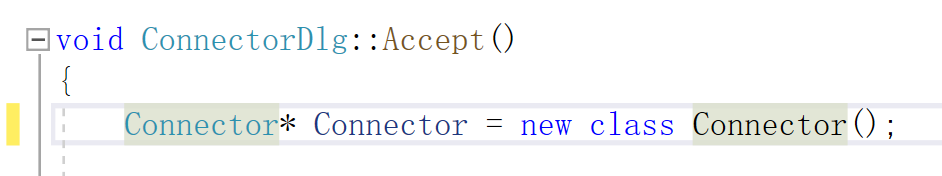
Label的属性更改问题？这是导致许多错误的原因

## 2021年11月22日星期一

### #include的使用？？如FeatureButton.h

一般不要随便#include

### 变量名定义



改为：Connector\* Connector1 = new Connector();

### 测试问题

1. 约束类型没有保存；打开再编辑不能保存；约束添加的逻辑？
2. 中文文件名存储失败：已经解决
3. CyberInfo的时候，标题名字：接口类型 | 接口名字，而不是 1，2；存储也有问题
4. 基准点、基准线添加的时候，左键可以 设置选择实体类型；
5. 连接：没有存储shape，坐标系？信息不全？；

## 2021年11月12日星期五

* Modelica信息

组件（model）里面的参数、变量、方程等信息怎么处理？

还有继承extends，输入import, 。。。等语句

还有inner，outer，output, within等 关键字 的处理？？？？

可以增加界面来输入和初步检查

* 装配体的显示问题

两种方式：改变TopoDS\_Shape的位姿；改变AIS\_Object的位姿

决定采用第二种？

* 零件颜色等属性的考虑：XCAF的测试？
* 文档节点如何删除？通过忘记属性来标记了！！！

//组件的节点序号为 1

TDF\_Label compLabel = m\_lblRoot.FindChild(1);

//组件的个数！！！注意：可能包含是删除的，也可能是Undo无效的

//判断一个节点是否有效，用第一个属性是否获得成功来判断

int nCount = compLabel.NbChildren();

* 接口组件（单独）的考虑：包括总线

在Component对象增加：IsConnector()，getConnectorPoint()等函数支持。

#### 任务分工：

吴义忠：提供数据结构支持，文档存储，显示

邓尔文：目录树视图，显示\*.m3dcom和m3dmdl，空目录不显示，两个弹出命令：打开和插入；装配树视图，显示各个Component\*，点击选中显示，命令：删除、打开组件、调姿、参数表；参数表设计；自由体的连接线

参数表： 组件名、组件参数、设置值、参数名（可为空）

梁清清：基于Component对象，考虑生成Modelica代码，递归生成；几个关键字？还有除了组件外，还有变量和方程，如何处理？

唐思豪：装配和调姿，基于Command的命令类系实现

## 2021年11月9日星期二

* 力元、速度等的考虑

参考modelica库，作为一个组件（零件），映射到库

* 组件放置考虑：主要数据 P, N, Vx；可能需要FromSystem，ToSystem两个坐标系

gp\_Ax3 theToSystem (const gp\_Pnt& P, const gp\_Dir& N, const gp\_Dir& Vx);

gp\_Trsf aCrankArmTrsf;

aCrankArmTrsf.SetTransformation (const gp\_Ax3& theToSystem);

myAISContext->SetLocation(myAisCrankArm,aCrankArmTrsf);

* 鼠标交互操作考虑：装配，连接，调姿（包括运动副约束体和自由体）

建立Command类系

## 2021年11月3日星期三

数据结构存储折中方案：

1. 所有操作均进行逻辑内存model\*访问与操作，同时更改文档内存；
2. 只有在Undo/Redo时，对文档操作，读出文档内容，更新逻辑内存Component\*；所以，Undo/Reo可能会耗时

* 主界面：

模型库树视图

支持右键弹出菜单

->打开…

->插入

装配树视图

可以设置零部件参数

->几何参数(暂不支持)

->物理参数（关注）

插入 | 装配 | 连接 | 调姿 | 检查 | 仿真

* 主要操作：

1. 插入组件到缺省位置；删除（del）
2. 安装机械部件到接口缺省位置
3. 调整机械部件姿态（记录参数）
4. 调整自由构件的位置和姿态，如果带连接呢
5. 建立组件接口之间的连接；删除(del)
6. 参数设置(在树视图直接输入，回车确认)
7. 模型检查（接口是否连接、匹配等基本检查；生成Mo，编译检查）
8. 仿真（调用求解器执行，打开后处理曲线界面；获得仿真结果）

关注变量树

曲线显示

三维动画显示区

动画控制条：启动暂停/继续，终止

关闭

转换到后处理菜单模式

## 2021年10月26日星期二

系统mdl的主要数据：组件，接口约束，连接，参数设置

几个重要函数过程：

Document::Redisplay()

{//根据文档m3dMdl\_DS的内容

//递归装载全部组件: Component\* parent = new Component();

Parent->SetDoc(m\_m3dmdl\_ds); //给parent赋值第一层值

Recurse\_Load(parent); //递归装载，填写parent的子组件内容

//刷新几何显示：组件和连接

Transform I0;

ShowComponent(parent, I0); 递归显示几何内容

//刷新装配树

updateAssemTree(parent);

//刷新菜单使能

updateMdlMenus();

}

//递归显示组件

Document:: ShowComponent(Component\* parent, Transform & trans)

{

//如果parent是零件，则显示

If(!parent.IsAssembly())

{

TNCollectionShape colShapes = parent->GetShapes(); //颜色透明属性？

ShowShapes(colShapes, trans);

}

Else

{

For\_each(com in parent.GetComponents())

{

ShowComponent(com, trans\*com->GetTransform());

}

}

}

Document::insert(String & sFileName) //导入文件，插入零部件

{

//parent在Document新建mdl时，new

//得到几何，缺省放置：transform单位阵

Recurse\_insert(sFileName, parent);

//记录到文档

M3dMdl\_DS->AddComponent(sFileName);

//刷新显示

Redisplay ();

}

//递归插入文件：读出sFile零（部）件的内容，创建Component对象，加到parent下

Void Recurse\_insert(string sFile，Component\* &parent)

{

如果sFile是零件，\*.m3dcom，则：

{

新建：OcafDocument

New m3dCom对象

得到零件的全部几何

如果pTrans为空，缺省放置：当前文档视窗OCC与导入的几何坐标系重合

否色，进行pTrans变换

//后面通过“装配/布置”命令来完成安装固定：对自由体（无机械接口）自由调整位姿，避免干涉；对机械零件，则通过接口匹配（选择）来安装

//记录ocafDocument\*，方便对参数化几何和约束值进行存储！！！

}

如果sFile是部件，\*.m3dmdl，则：

{

新建: Ocafdoc

New m3dMdl对象

得到全部的子装配体，lstComponents

Foreach sfile\_i in lstComponents

得到其 transform\_i

Recurse\_insert( sfile\_i，transform\_i);

}

}

Document::LoadFile(string sFile) //打开文档

{

新建: Ocafdoc

New m3dMdl对象

得到全部的子装配体，lstComponents

Foreach sfile\_i in lstComponents

得到其 transform\_i

Recurse\_insert( sfile\_i，transform\_i);

}

## 2021年10月22日星期五

### M3dMdl\_DS设计

Class M3dMdl\_DS //模型数据结构

{

TDF\_Label m\_LblRoot;

Component\* m\_ComRoot;

list<Component> components; //组件

list<ConnConstraint> lstConnectConstraints; //接口约束

list<ConnectCurve> lstConnectCurves; //连接线

list<ParameterValue> lstParamValues; //参数设置

//主要接口函数

Bool InsertComponent(string sFilename); //插入组件到文档

Component\* GetComponent(TopoDS\_Shape & shape); //得到组件

Connector \* GetComponentConnector(TopoDS\_Shape & shape); //得到接口，**只搜索组件的第一层接口**，其子孙的接口不搜索；

Void TransformComponent(Component\* pCom, Transform& trns); //对组件进行变换

Void DeleteComponent(Component\* pCom); //删除组件

Void AddConnConstraint(ConnConstraint\*); //增加接口约束

//Void DeleteConnConstraint(ConnConstraint\*); //删除接口约束

Void GetAllConnConstraints(list<ConnConstraint\*> & lstConnCons); //得到全部ConnCons

Void SetConnConstraints(); //设置全部接口约束，在接口约束界面“更新”时调用

Void AddConnectCurve(ConnectCurve\* pCurve); //增加一条连接线

ConnectCurve\* GetConnectCurve(TopoDS\_Shape & shape);//拾取一条连接线

Void DeleteConnectCurve(ConnectCurve\*); //删除一条连接线

Void AddParameterValue(ParameterValue\*);

Void DeleteParameterValue(string sParamFullName);

};

Class Component //组件

{

Bool IsAssembly; //是否装配体 不存储

list<Component\*> m\_lstComponents; //组件，不存储

int m\_nIndex; //组件名序号，根据零件名加，如”cam\_1”，从1开始编号

string m\_sFileName; //组件或零件的文件名，\*.m3dcom或\*.m3dmdl

point m\_position; //原点位置，在当前模型中的坐标

orientation m\_orient; //姿态，**可以与位置合并，transform**

list<ConnectConstraint> lstConnectConstraints; //接口约束，（物理）运动副约束

list<ConnectCurve> lstConnectCurves; //连接线

list<ParameterValue> lstParamValues; //参数设置

list<myParameter\*> lstParameters; //关注的参数列表，不存储

几何参数怎么办？可以通过增加 尺寸约束的名字来标注。由于零件的尺寸参数化导致装配体的位姿参数化，求解过程复杂，放在后面考虑吧。。。。。。。

List<myOutputVar\*> lstVars; //关注的输出变量表，不存储

M3dCom\_DS\* m\_pDS? 还是基类 M3d\_DS\* ?

形状？

}；

Class ConnectConstraint

{

Kinematic\_pair\_type kpType； //运动副约束类型

String component1; //第一个物体

String component2; //第二个物体

String connector1; //第一个物体上的接口名字

String connector2; //第二个物体上接口名字

}；

Class ConnectCurve

{

ConnectCurveType ccType; //连接线类型：单线，控制线，总控线，。。。

String component1; //第一个物体

String component2; //第二个物体

String connector1; //第一个物体上的接口名字

String connector2; //第二个物体上接口名字

List<Point> lstPoints； 中间控制点，用B样条曲线

}；

Class ParameterValue

{

String sParamFullName; //参数全名，带点

String sParamValue；//用字符串，因为可能是数组，可能是bool，甚至字符串

};

M3dmdl\_DS的文档结构

Root

Components (1) //组件

Component1(1-1)：TDataStd\_ExStringlist: 名字 + 文件路径；RealArray: 12（3X4）; 位姿FromSystem 和ToSystem

deltaTrans(1-1-1)只有一个：相对位姿（1-1-1）：Ax3(pt, Z, X); + x,y,z; rx, ry, rz = V15 RealArray: 15

***组件属性（1-1-2）：int Array***

接口约束(1-1-1)：对于com零件，为空

连接线(1-1-2) ：对于com零件，为空

参数设置(1-1-3)：组件1的参数设置

Component2(1-2): //组件2

。。。。。。

ConnectConstraints(2) //接口约束，装配约束

* ConnectConstraint1(2-1)：运动副类型(int)+组件名和接口名（StringArray[4]）
* ConnectConstraint2(2-2): 同上

。。。。。。

ConnectCurves(3) //连接线(3)

* ConnectCurve1（3-1）：几何TNameShape; + 连接线类型(int)+组件及接口名（stringArray）+points（realArray, 第一个可以作为点的个数，如：[3，0.0,0,0,0.0, 1,0,1.0,1.0, 2.0,2.0,2.0]）
* ConnectCurve2（3-2）：同上

。。。。。。

ParameterValues(4) //参数设置(4)

* ParameterValue1(4-1) ：参数名+映射组件参数名+参数值，stringArray[3]
* ParameterValue2(4-2)

。。。。。。

//载入组件，插入…

Document::load

新建一个DocumentOCAF，openFile

## 2021年10月19日星期二

### M3dMdl功能分析：

1. 文件功能：\*.m3dmdl，新建、打开、保存、另存，最近文档，退出
2. 编辑：Undo/Redo，选择类型设置，拷贝、粘贴
3. 建模：插入…(包括模型m3dmdl和组件m3dcom)，包括运动副、world（作为先建立的M3dCom组件插入）

装配——在插入零部件时，如果部件中带运动副接口，则弹出装配界面，选择进行装配；如果插入的是自由体，则弹出操纵界面，进行移动和旋转

连接——建立连线（粗细表示总线的级别，细线表示单个接口信号）

1. 仿真：求解…（显示求解工具条），求解设置，仿真结果
2. 设置：系统设置
3. 帮助

应用场景1：6自由度机械手**机械结构**部分

1. 新建mdl， 左边停靠窗口：上面是模型库（m3dcom和m3dmdl），由所在的当前目录加载，支持右键加载、打开功能；下面是当前模型树，各个组件及其参数，包括几何参数和物理参数，显示缺省值，可以设置新值
2. 插入World，带Frame接口（在图形中心0，0）
3. 插入b0，与World的Frame接口坐标系重合，可以绕X,Y或Z轴旋转调整，注：如果只有一个同类型接口，则直接连接，坐标系匹配；如果有多个接口，则需要从待插入模型上选择，和从当前装配模型中选择相应的接口
4. 插入运动副r1（旋转副），连接b0的Frame接口，同上可以调整
5. 插入b1，与r1相连接，可以调整
6. …依次，插入r2,b2,r3,b3,r4,b4,r5,b5, load
7. 插入fang接口6个，建立连接

保存为mechStruct.m3dmdl

应用场景2：机械手整体

1. 新建mdl
2. 插入 mechStruct.m3dmdl，机械结构
3. 插入轴控制器axiscontrol1，连接flang接口到r1（装配），可以调整，只能绕Z轴旋转；
4. 插入其他轴控制器，连接到相应的flang接口，安装调整到合适方位；
5. 插入控制总线零件（已经建好的m3dcom），作为自由体，只是调整其位姿，可以施加临时约束来定位，提示接触碰撞，但不记录；
6. 建立连接：连接控制总线（分线）与各个轴控制器相应的接口，以曲线红色、蓝色显示；
7. 插入路径规划零件，同上，作为自由体；
8. 建立连接：连接路径规划的总线与控制总线直接相连，连接点在控制总线的中心
9. 保存模型fullRobot.m3dmdl
10. 仿真，先设置求解时间和算法；仿真求解，提示完成，弹出动画控制条，点击动画，可以暂停继续或停止
11. 打开仿真结果显示窗口，右边是关注的各个组件的变量，不关注的不显示，点击可显示时域曲线
12. 修改物理参数，再运行仿真
13. 修改几何参数？

### M3dmdl的主要数据

1. 组件（部件和零件）：自动编名，文件路径全名，形状+初始位姿（每个组件有一个坐标系，相对于当前模型的坐标系的值）
2. 接口约束（用于参数化驱动）
3. 参数设置
4. 连接线

### 重点思路

1. 装配过程，基于历史，不采用无历史，简化问题；
2. 参数化导致的装配形状改变，依赖三维约束求解，欠约束可以装配
3. 关于多自由度装配，根据前面的欠约束
4. 装配局部坐标系，支持移动旋转

## 2021年10月16日星期六

实体，引用均使用 名字，如Face\_6

## 2021年10月14日星期四

（1）数据结构改进

对于零件，只允许一个主Solid，有且仅有一个

导入时，如果没有Solid，则提示；多于一个也提示。导入时，提示清除当前所有。

缺省的材料用钢。

Root

Shape(1) – Solid (TNameShape)----------------------只需存储一个Solid!!!!!!!!!!!!!!

Material(1)： Int 直接放上去

Tranparency(2): double 放上面属性

//Color(2): RealArray

//physicalProperty(4)：RealArray体积（质量），质心[3]，惯性矩[6],…

Colors(2)

Child1(1) : 图元名称(String) + Color（RealArray）

。。。。。。

Datums(3)

3-1 参考点

Child1(1): Shape +数据结构：

3-2 参考线

3-3 参考面

。。。。。。

Connectors(4)

Child1(1): Shape + 数据结构：根据Connector的数据结构，定义其存储

。。。。。。

Constraints(5)

…参考前面的设计

CyberInfo(6)

参考前面设计

（2）材料的问题：

Material : 名称（Graphic3d\_NameOfMaterial），对应的颜色，密度，透明度

比如：Graphic3d\_NameOfMaterial\_Steel，Gray,7.8e-6***kg/mm3***，0

## 2021年10月13日星期三

Shape存储的改变？

Root

Shape（1）

Solid（1）（只有一个）两个属性：形状+颜色+透明度

Property（1）: 材料+physicalProperty

Faces（2~N）…（直接两个属性：形状+颜色）

Edges(1~N)…（直接两个属性：形状+颜色）

Vertexes(1~N)…（直接两个属性：形状+颜色）

Connectors（2）

Constraints（3）

CyberInfo（4）

将shape改为树型存储，便于找到其父节点，如一个顶点属于哪条边，哪个面

## 2021年10月10日星期日

关于Vertex/Edge的存储：

考虑参数化需求，将Vertex（Edge?）的存储稍作改变，即将相同坐标Gn\_Point的Vertex存储在同一个TDF\_Label下，这样改变某个Vertex时，同时其父节点下面的全部子节点（即该节点的全部兄弟节点），这些节点在逻辑上（理论上）是同一个点。

……

TDF\_Label ---“Vertexes”

TDF\_Label\_1 ---- “V1” 逻辑上的一个顶点**IsSame()**---gp\_pnt（TDataStd\_RealArray）

TDF\_Label\_1\_1 ---TNameShape --某个边e1上的顶点，在V1

TDF\_Label\_1\_2 ---TNameShape --某个边e2上的顶点，在V1

……

对于一个长方体的顶点，其重合度为6.

对于Edge，也有重合的两条边，即公共边的两个有向边，但可以先不考虑，因为我们采用基于顶点的驱动。只要对应的顶点移动，公共边的两个有向边都会跟随发生变动。

## 2021年10月7日星期四

数据结构存储的两种模式：

1. 考虑Undo/Redo的模式，每次操作即对文档的TDF\_Header进行查询操作（编辑、增加或删除）；
2. 不考虑Undo/Redo操作，每次操作只对内存数据进行更改，只在读写操作时，更新互换内存与文档数据。
3. 文档树



## 2021年9月27日星期一

### M3dCom的对象设计再考虑

* **CLASS m3dcom:**
* Shapes – 全部形状，包括导入的几何和接口几何形状
* NamedShapes – 命名的形状: list<Vertex>; list<Edge>; list<Face>；Solid
* Colors – 颜色 list<EntColor> 颜色存储的三种方法：直接属性，引用，编码
  + 图元：TDF\_Reference->TDF\_Label
  + 颜色：index – Standard\_Integer：用 RealArray存储Quantity\_color
* Constraints – 约束数据；list<Constraint>
  + 类型：Standard\_Integer
  + 图元1：GUID？用二元组编码表示【类别，序号】，或“面3”（显示）
  + 图元2：GUID
  + 约束值：Stardard\_Real
* Material – 材料
* Transparency – 透明度
* ValidationProperty – 物性参数值？
* CyberInfo – 信息数据
  + 接口
* **接口放置的界面操作思路**

多体Frame接口：选择一个顶点（参考点），一个平面（或者XOY标准面），只在上面平移(delX, delY)

转动副接口（法兰Flange接口）：选择一个圆柱面，依次圆柱面中心为参考，给定移动量(delZ)

平移副接口：选定一个顶点、一条边（或X,Y,Z）作为移动方向，移动平面？

电、控制的接口只是一个示意点，参考Frame接口定义

* **物性参数的需求？**

质心、质量、转动惯量，惯性矩…需要分析Modelica的多体建模

* **界面实体（顶点、边、面）选择的基本操作思路：**

**对一个编辑框（或者列表框的一个表格）：**

双击事件：将选中的实体（类型判断合理）编码提取，填入，如[2,3]

单击事件：如果为空，没有相应；否则，亮显该实体；

右键（或del键）：取消选择，编辑框恢复为空。

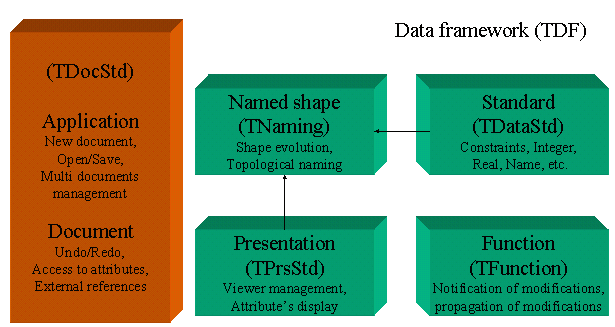
选中整行（列表框），则显示全部的实体。

## 2021年9月24日星期五

### M3dCom的功能实现进展

1. 文件操作：打开，保存/另存，新建，导入
2. 选择设置
3. Undo/Redo
4. 打开的选择问题解决！！！
5. 保存遇到：\*.cbf的暂时方案

### M3dCom的数据存取思路



1. **M3dCom\_DS**的数据存取：树状，将复杂数据类型转换成树型数据结构，叶子节点均为标准的类型：TNaming\_Named\_shape，TDataStd…，TPrsStd, TFunction等，存取过程参考示例OCAF的Make box。
2. 图元（面、边、顶点）引用的两种思路：1）在TopoDS\_Shape中增加GUID；2）使用一个序号来标识（需要测试，每次存取或操作，该序号保持不变。可以用两维整型数组表示，第一个是0，1，2分别表示面、边还是顶点，第二个是序号，在当前Shape（一个零件一个主Shape）的第几个）；

## 2021年9月18日星期六

### M3dMdl功能：

1. OCCT基本功能
2. 文件：new, open, Save/SaveAs，最近文件列表，退出
3. 编辑：Undo, Redo, Select类型；旋转，平移，放缩；Add…，delete
4. 连接：建立连接；连接器（运动副、控制总线BUS）
5. 仿真：生成代码并检查；仿真求解（显示仿真后处理界面，添加关注输出变量的曲线显示）；动画驱动（初始位置的考虑）
6. 设置：系统设置
7. 帮助



## 2021年9月7日星期二

### M3dCom功能：

1. OCCT基本功能
2. 文件：New, Open, Import，Export，Save/SaveAs，最近文件列表，退出
3. 编辑：Undo, Redo, Select类型设置->
4. 约束：显示约束表（拓扑约束不可修改），形位约束，尺寸约束
5. 信息：关联mo（显示参数表）；接口定义（机械，电气电子，液压，控制，……）
6. 窗口：支持多文档
7. 系统设置：约束显示方式；接口显示方式
8. 帮助

* OCCT基本功能

以***toolbar***提供

基本操作：

视窗变换：



光线：

* 文件

***工具条FileToolbar***: new, open，save, import, export

1. New: 新建文档和窗口，不关闭现有窗口文档，支持多文档
2. Open…，打开文件对话框，显示扩展名：\*.m3dcom，即m3d的组件文件类型
3. Import…输入，打开文件对话框，输入类型：\*.brep, \*.step, \*.iges
4. Export…输出，打开文件对话框，输出类型：\*brep, \*step,\*.iges, vrml, stl，xml等
5. Save/SaveAs，保存文件，类型为\*.m3dcom
6. 最近文件列表，选择即打开
7. 退出系统，检查文档修改，提示保存

* 编辑

***工具条EditToolbar***: Undo, redo, 选择类型列表

1. Undo/Redo: 设置无穷次Undo/Redo，采用OCAF
2. 选择类型：顶点、边、面、实体、特征、接口、任意等

* 约束

***工具条ConsToolbar:*** 显示/关闭约束表；添加常用约束…；重构刷新

1. 显示约束表，可以在此表中统一管理约束



元素，用label来记录

注：对于自动识别生成的拓扑约束，灰色不可修改，除了是否起作用

1. 添加…形位约束：垂直，平行，相切，重合，…（参考DCM），先选择两个实体，再添加约束，对于矛盾的约束给出提示，不添加
2. 添加…尺寸约束: 长度、角度（参考DCM），同上，添加的约束缺省的都是起作用约束
3. 重构：调用求解器，求解刷新显示

注：尺寸约束将用于系统建模

* 信息

***CyberToolbar***：关联mo；显示/选择标记参数列表；添加接口不同类别

1. 关联mo：打开文件选择对话框，选择\*.mo类型，返回模型全名，如Modelica.Blocks.Integrator, User.Mylib.DCmotor等，记录文件path；读取参数列表并记录；读取输出变量列表；读取接口类型信息并记录。
2. 显示/选择标记参数列表



删去

先允许编辑

不需要设置

1. 接口

***ConnToolbar***：常用的接口，机电液控混在一起即可。

接口添加只是在关联了mo模型以后才能添加，获取mo模型顶层接口信息，才能添加对应的接口几何，选择的几何元素也要进行过滤，另外，接口的方位应该考虑与选择的几何元素关联，以适应后面的参数化，可采用OCC的引用key功能。

根据modelica，参考各专业知识，对接口进行分类：

* 机械：低副接口：旋转副接口、平移副接口，…；高副接口：齿轮副，…；多体法兰副接口，…
* 电气电子：pin，…
* 液压：
* 控制：

不同的接口添加，显示不同的几何示意。

根据选中的几何元素，点击添加允许的接口，直接显示即可，不需要弹出选择对话框。

接口放置的位姿可以绝对或相对。相对即根据选择的几何元素来确定，需要分析不同的接口，不同的相对确定规则。

接口几何可以不存储，打开文件时，扫描接口加以显示。

接口矢量：同一类型接口的个数

复合数据接口与BUS总线的考虑：

复合数据接口：记录各接口类型，显示接口的个数，每个接口与一种数据类型对应，将来联结时要对应。

BUS总线与复合数据接口类似。BUS接线盒在系统建模中实现。

### M3dCom数据结构

M3dComDS – 主对象

* TopoDS\_Shape Shape : 形状外形，导入的几何体
* List<Constraint> ConsList : 约束链表
  + <Node>: Constraint: enum 类型；GUID 元素索引1,2；real 值；bool 是否起作用
* CyberInfo : 信息数据结构
  + String moPath； 文件全路径
  + String mdlName; 模型全名
  + List<parameter> lstParameters; 参数表
    - Parameter : string sName; Real rDefVal; bool bFocused; string sMarkName; real rSetNewValue;
  + List<output> lstOutputs; 输出变量表
    - Output : string sName; bool bFocused; string sMarkName;
  + List<connector> lstConnectors; 接口表（接口的个数与mo模型一致）
    - Connector : enum type; string sName接口全名；Int 接口（矢量）个数；ConDisp 接口位姿；
      * ConDisp: pos3d 位置，dir 方向；bool 相对/绝对; GUID 引用1，GUID 引用2；还要仔细分析！暂时就在此处界面定义矢量及个数。

## 2021年6月15日星期二 周计划

1. SVN项目重整，只留下：自研代码；写下相对目录结构文档说明；
2. 更改现有的解决方案m3d和两个项目名称m3dcom和m3dmdl；
3. 设计两种主界面（粗）：组件（com）和模型(mdl)的下拉菜单和窗口及布局
4. 新建两种文档，\*.m3dcom和\*.m3dmdl，对应激活两种主界面

## 2021年6月21日 星期一 周计划 组件模块实现1

1. 组件界面详细设计；
2. 组件与Modelica组件的映射关系定义；
3. 接口对象体系实现；先考虑机械、电子、控制
4. 接口几何显式+操作交互；
5. 组件模型的序列化与反序列化；

# 基于FreeCAD的CPS组件/系统建模软件设计

## 主界面：采用QT

模型扩展名: \*.m3d/c3d；分别对应模型和组件

New/Open：两种文档，组件模型（零件）、系统模型（装配）

系统模型能否映射Modelica模型？

Import: freeCAD零件（参数化），也可以Step, IGES, STL等格式，但不参数化

从哪儿来？

### 组件界面

不提供建模功能，只导入形体，并支持参数化

### 模型界面