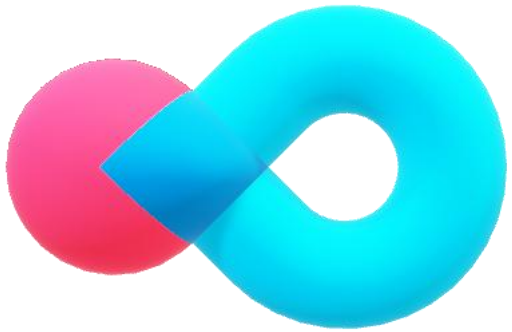


OpenCASCADE

开发培训

武汉欧凯德信息科技有限公司

功能概述	造型数据	造型算法	自动化测试	案例实践
• 功能介绍 • 用户案例 • 安装编译 • 编码规范 • 模块概览	• 几何曲线 • 几何曲面 • Topology	• Primitives • Mesh • Sweep • Fillet • HLR • BO • History	• Test Grid • Test Report	• occQt • PipeCAD • RvmTranslator

功能概述

Overview

What is Open CASCADE Technology?

Open CASCADE Technology (OCCT) is a powerful open-source C++ library, consisting of thousands of classes and providing solutions in the area of:

- Surface and solid modeling: to model any object.
- 3D and 2D visualization: to display and animate objects.
- Data exchange: to import and export standard CAD formats.
- Rapid application development: to manipulate custom application data.

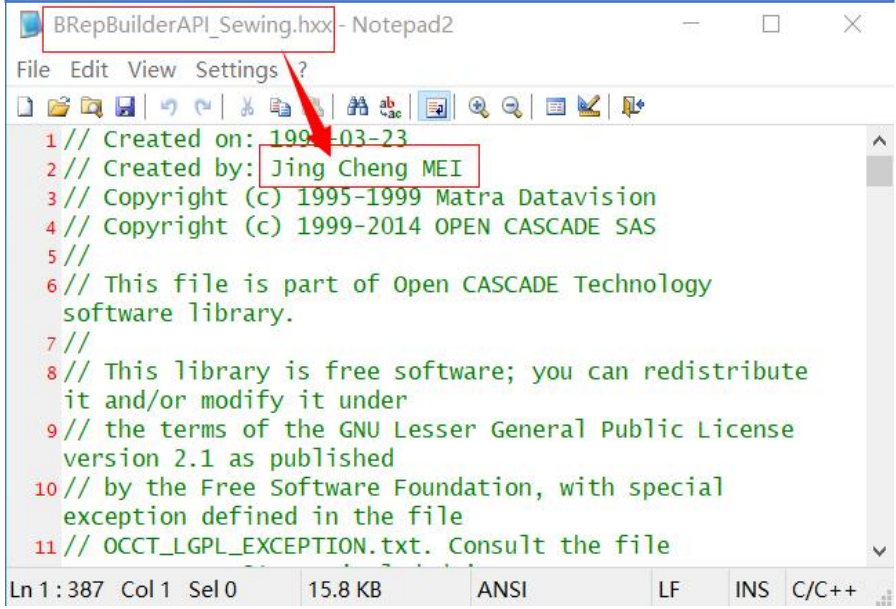
OCCT is also applicable in many other areas of CAD/CAM/CAE, including AEC, GIS, and PDM. OCCT is designed for the industrial development of 3D modeling and visualization applications that require good quality, reliability, and a robust set of tools.

OCCT libraries are distributed in open source for multiple platforms under GNU Lesser General Public License (LGPL) version 2.1 with an additional exception.

功能介绍

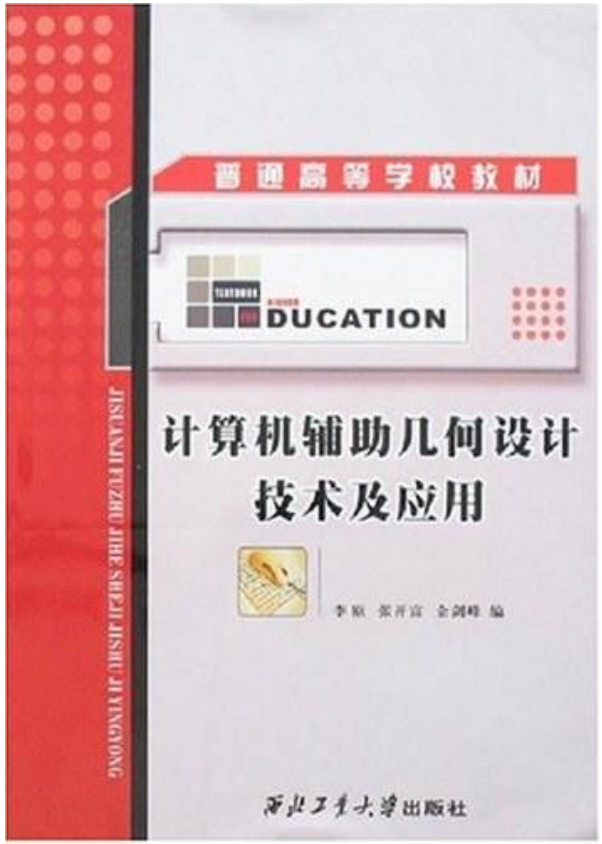
- Open CASCADE是一套开放源代码的CAD/CAM/CAE几何造型内核，源自于1980法国的Matra Datavision公司，这套SDK库原是著名的CAD/CAM软体EUCLID的开发平台，但在1998年， Matra Datavision改变了经营策略，从以销售软件为主改变为提供CAD/CAM/CAE软件服务为主的获利模式，并且在1999年公布Open CASCADE的程序源代码。

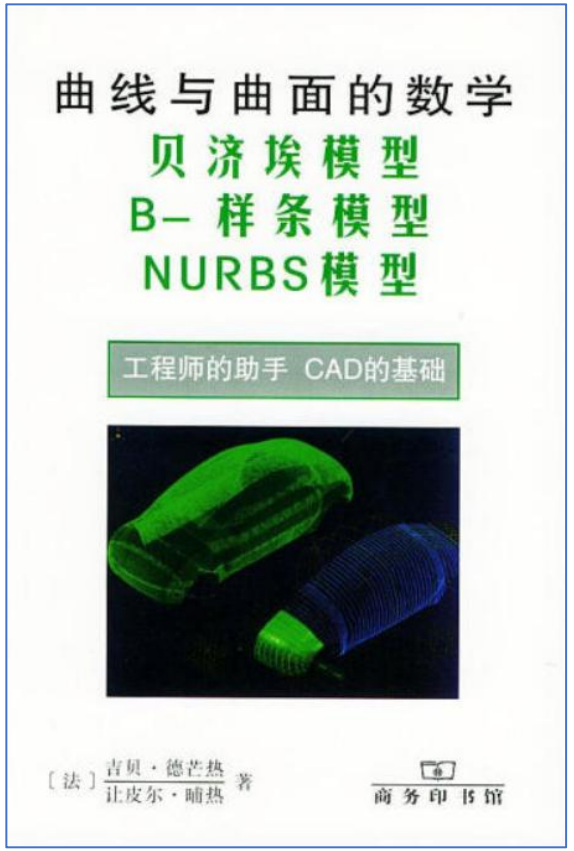
BRepBuilderAPI_FindPlane
BRepBuilderAPI_Sewing
BRepOffsetAPI_FindContigousEdges
DsgPrs_ShapeDirPresentation



功能介绍

- 法国Matra-Data Vision公司的Euclid集成系统是一个集机械与工厂设计于一身的企业级并行工程解决方案，其曲面功能在“ASD高级曲面设计”之中。曲面由NURBS和Bezier数学形式表达，通过强大的蒙皮、扭曲、放样、裁剪、联合等运算，系统能够形成复杂的外形。其实体造型功能可直接用于曲面，表现出突出的拓朴运算能力。例如，多曲面间的交、并、差运算；在多曲面间的空隙处填充成保持一致切矢、曲率的新曲面；构造相切于已知曲面的曲面等。Euclid动态自由造型功能，实现了以曲面曲率进行动态曲面跟踪、编辑、控制的设计修改过程，很好地体现了交互技术的应用。



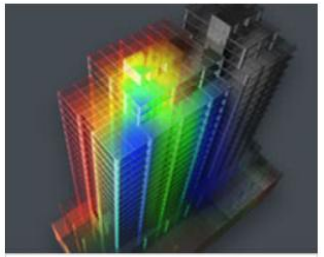


用户案例

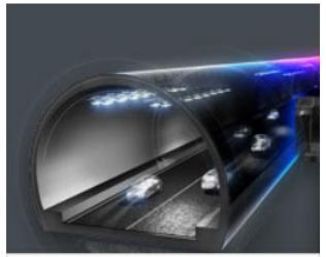
Midas(韩国)　　CFturbo(德国)
Tekla(芬兰)
CADMATIC(芬兰)
SIEMENS-FORAN(西班牙)

桥梁领域
Bridge Engineering

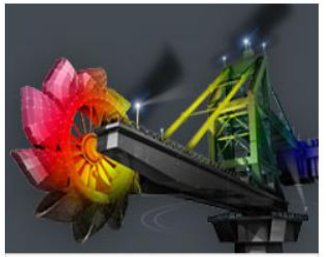
MIDAS



建筑领域
Building Engineering

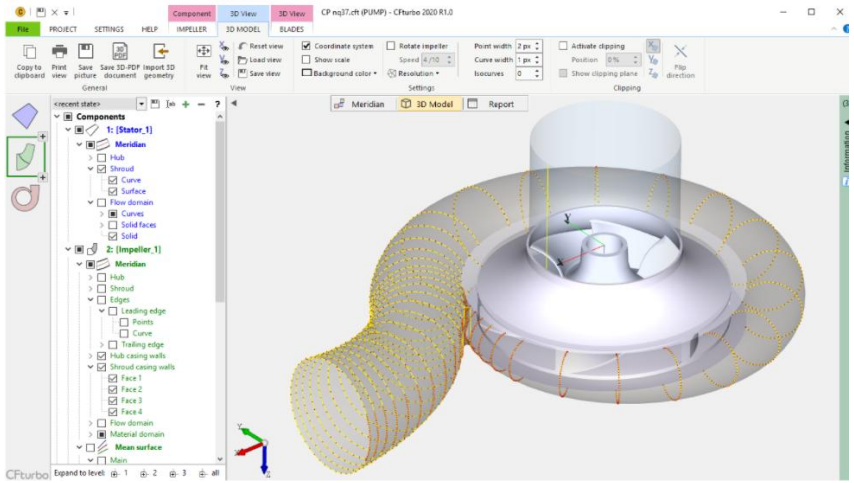


岩土领域
Geotechnical Engineering

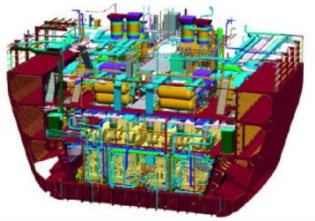


机械领域
Mechanical Engineering

Tekla
Structures



SIEMENS



FORAN - Industries & Applications

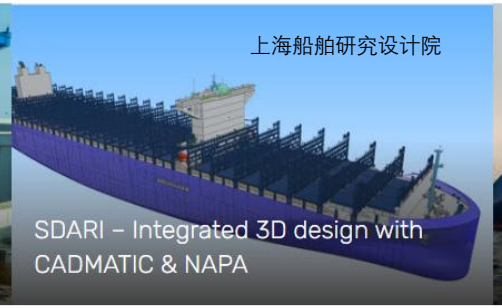
FORAN is a ship design CAD/CAM system that, because of high-level features like adaptability and customization, can be used to design and build any type of ship or marine structure. The wide-ranging functionality of this marine design software meets the needs of all the various industries in the marine sector.

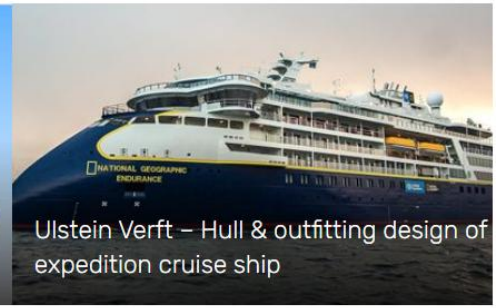
FORAN - Design Phases

FORAN is a CAD/CAM/CAE system for marine design. It is multidisciplinary and fully integrated and can be used at every phase of the ship design process, and in every discipline as it stores information in a single database.

CADMATIC

HOME　MARINE　PROCESS & INDUSTRY　CONSTRUCTION　RESOURCES　ABOUT US　CONTACT US





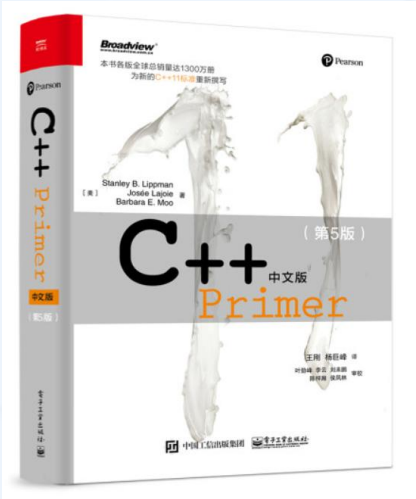
What you should know

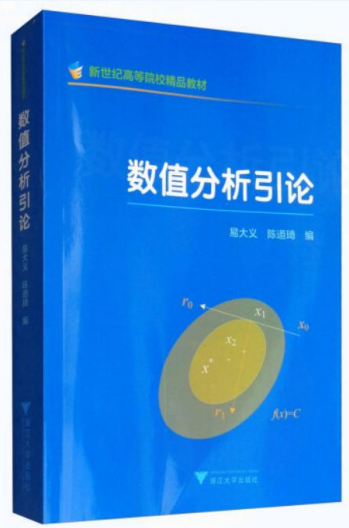
Mandatory knowledge:

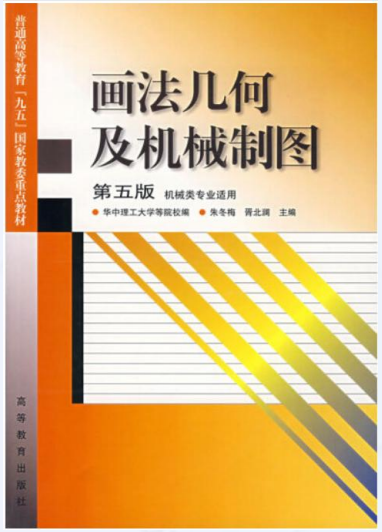
- C++ object-oriented language.
- Programming technologies.
- Common mathematics, algebra, geometry.

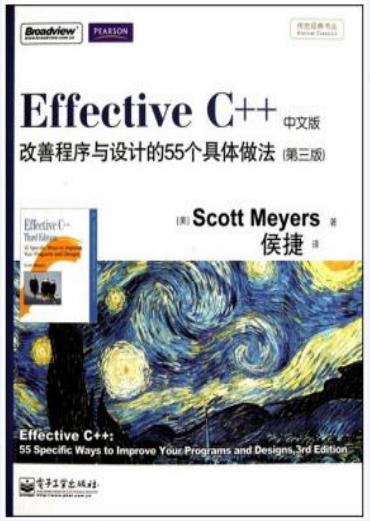
Optional knowledge :

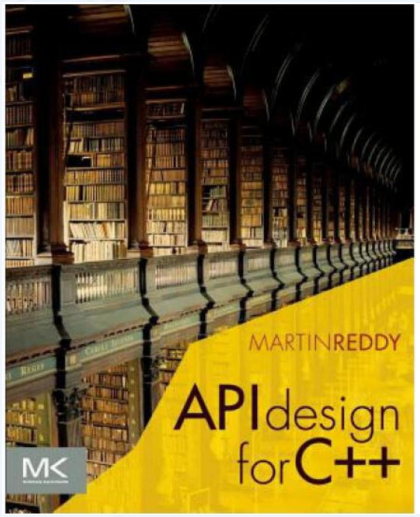
- Basics of computer-aided design.

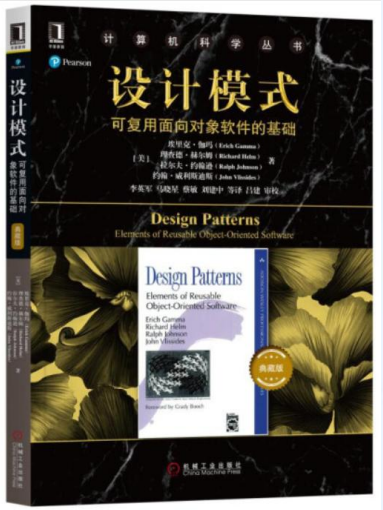












Literature

“Introduction to solid modeling” by Martti Mantyla

- Introduction-level book for a newcomer.
- Requirements to geometric modeling: geometric question concept.
- Various modeling techniques: wireframe, boundary representation, voxels. Their advantages and drawbacks.
- Difference between geometric modeling and other kinds of engineering software.

“Solid modelling and CAD systems” by Ian Stroud and Hildegarde Nagy

- Assembly structure.
- General pieces of advice about data model organization.

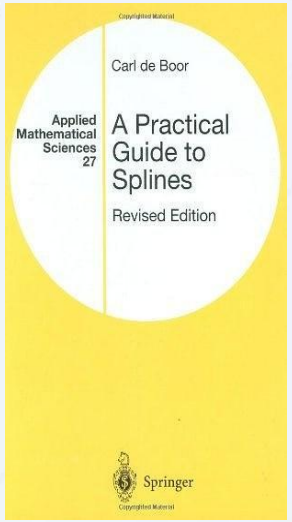
“Parametric and Feature Based CAD/Cam: Concepts, Techniques, and Applications” by Jami J. Shah and Martti Mantyla

- Feature concept.
- Parametric modeling concept.

Literature

"A Practical Guide to Splines" by Carl de Boor

- OCCT follows this book for underlying spline mathematics.
- Code samples are based on Fortran.
- Indexes start from 1.

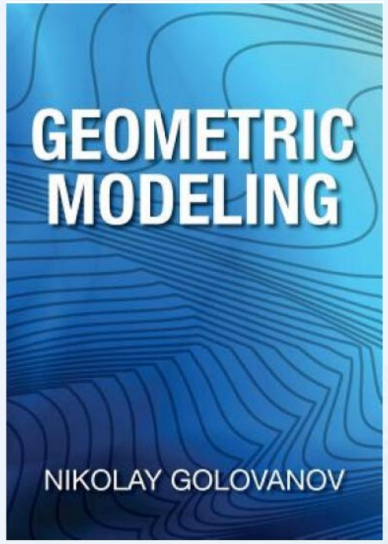


"The NURBS Book" by Les Piegl and Wayne Tiller

- SMLib kernel is based on this book.
- The best book about b-splines.
- A lot of code samples.
- Code samples are based on C/C++.
- Indexes start from 0.

"Geometric Modeling" by Nikolay Golovanov

- C3D Toolkit is built on this book.
- Covers a wide set of problems.
- Utilizes the recipe-based approach.



What is in distribution?

Documentation

- Automatically generated from Doxygen comments.
- Manually written user and developer guides.

Programming samples

Programming samples using different GUI:

- MFC.
- C#.
- Qt.
- Java.

Test harness application

- TCL-based command interpreter.
- A set of predefined commands.
- Most of OCCT API is available in Test Harness.
- Test Harness is a prototyping framework of OCCT.
- This application is used to test OCCT itself.

安装编译

- 跨平台编译CMake

Build Makefiles

Open console and go to the build folder. Type "mingw32-make" (Windows) or "make" (Ubuntu) to start build process.

```
mingw32-make
```

or

```
make
```

Parallel building can be started with using **"-jN"** argument of "mingw32-make/make", where N is the number of building threads.

```
mingw32-make -j4
```

or

```
make -j4
```

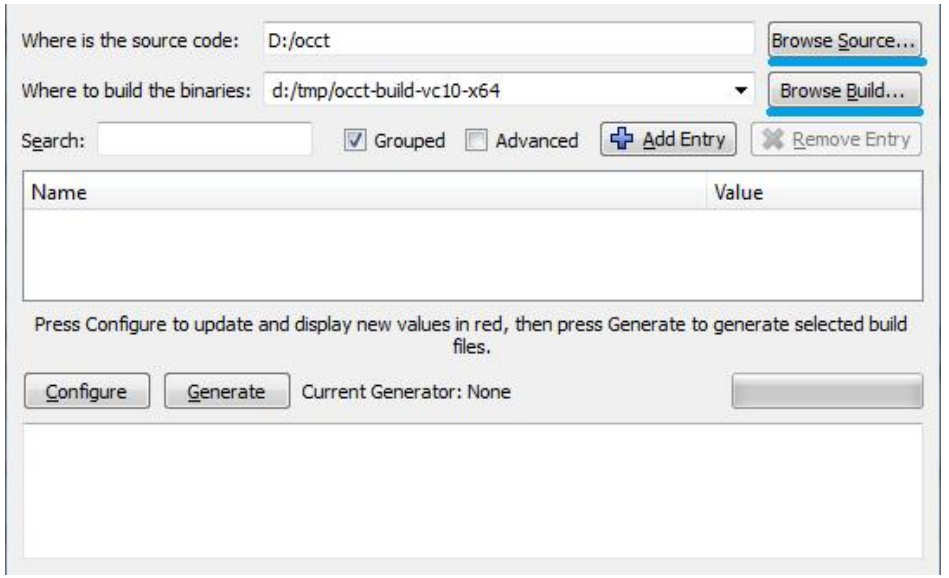
Install OCCT Libraries

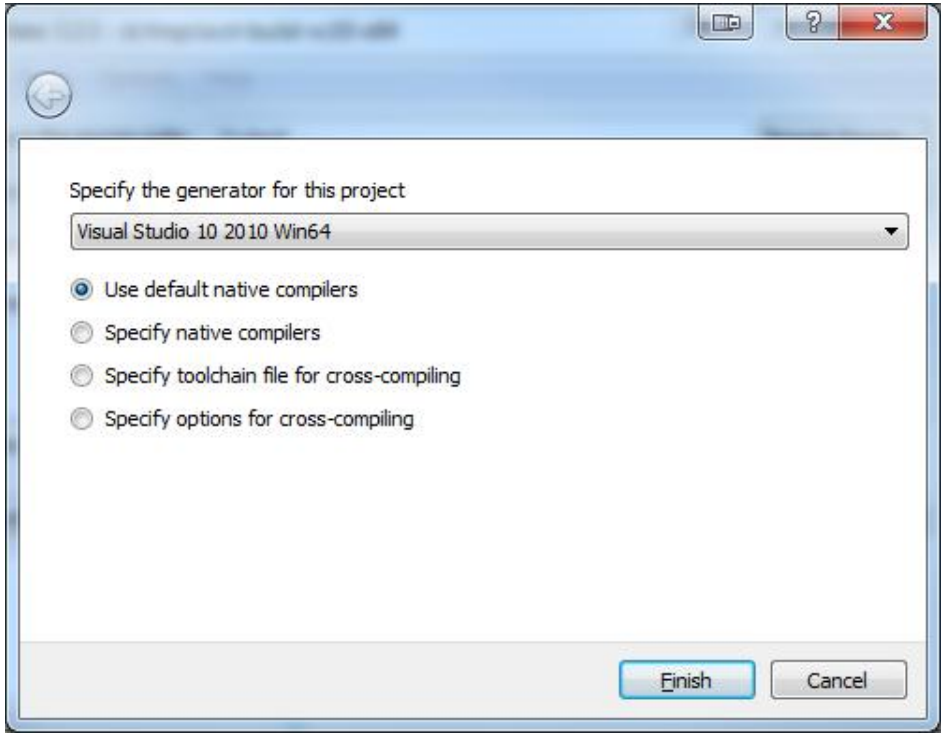
Type "mingw32-make/make" with argument "install" to place the libraries to the install folder

```
mingw32-make install
```

or

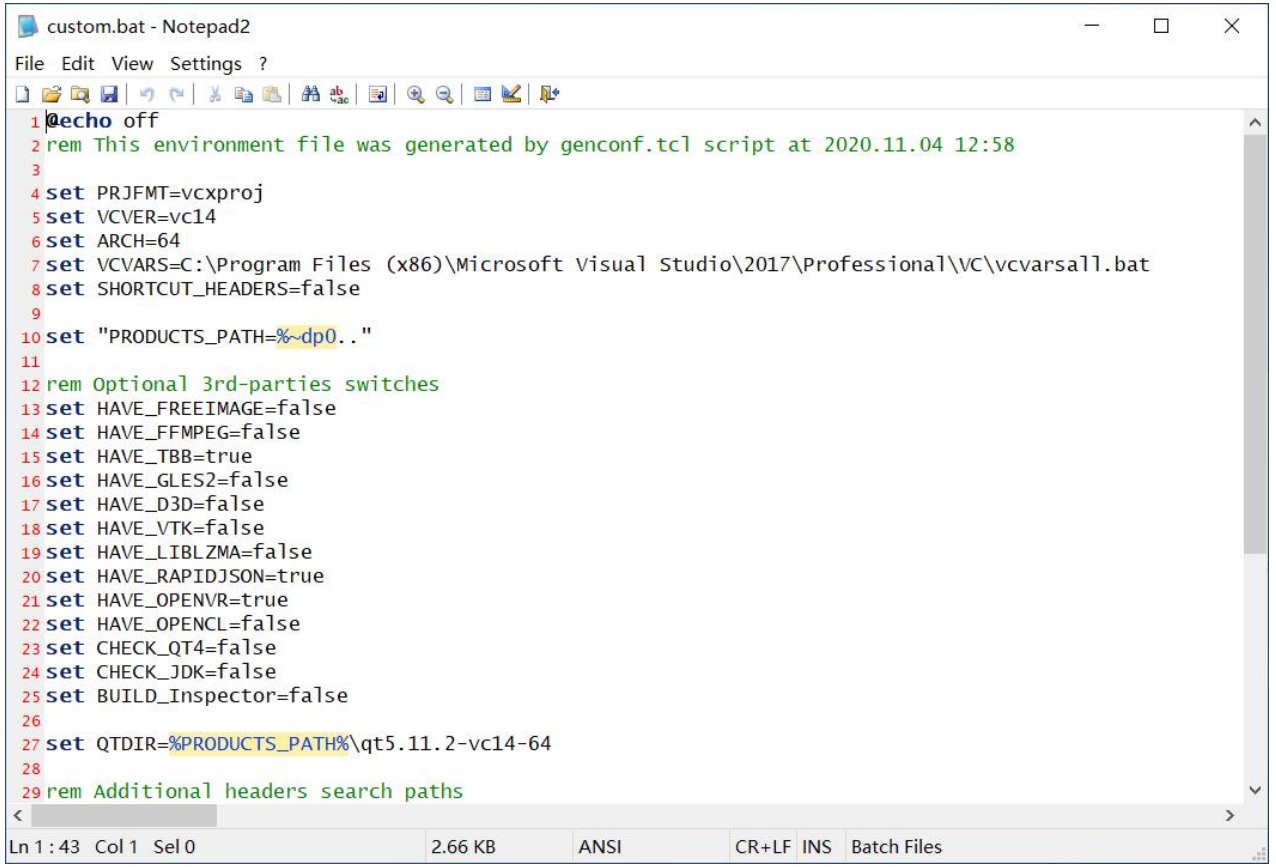
```
make install
```





安装编译

- Windows编译
  - 在custom.bat中配置好第三方库及其他选项，直接运行msvc.bat来启动Visual Studio进行编译。



安装编译

- Windows编译
- 配置选项说明

If you have Visual Studio projects already available (pre-installed or generated), you can edit file custom.bat manually to adjust the environment:

- VCVER – specification of format of project files, defining also version of Visual Studio to be used, and default name of the sub-folder for binaries:

VCVER	Visual Studio version	Windows Platform	Binaries folder name
vc10	2010 (10)	Desktop (Windows API)	vc10
vc11	2012 (11)	Desktop (Windows API)	vc11
vc12	2013 (12)	Desktop (Windows API)	vc12
vc14	2015 (14)	Desktop (Windows API)	vc14
vc14-uwp	2015 (14)	UWP (Universal Windows Platform)	vc14-uwp
vc141	2017 (15)	Desktop (Windows API)	vc14
vc141-uwp	2017 (15)	UWP (Universal Windows Platform)	vc14-uwp
vc142	2019 (16)	Desktop (Windows API)	vc14
vc142-uwp	2019 (16)	UWP (Universal Windows Platform)	vc14-uwp

- ARCH – architecture (32 or 64), affects only PATH variable for execution
- HAVE_* – flags to enable or disable use of optional third-party products
- CSF_OPT_* – paths to search for includes and binaries of all used third-party products
- SHORTCUT_HEADERS – defines method for population of folder inc by header files. Supported methods are:
  - Copy - headers will be copied from src;
  - ShortCut - short-cut header files will be created, redirecting to same-named header located in src;
  - "HardLink* - hard links to headers located in src will be created.

编码规范

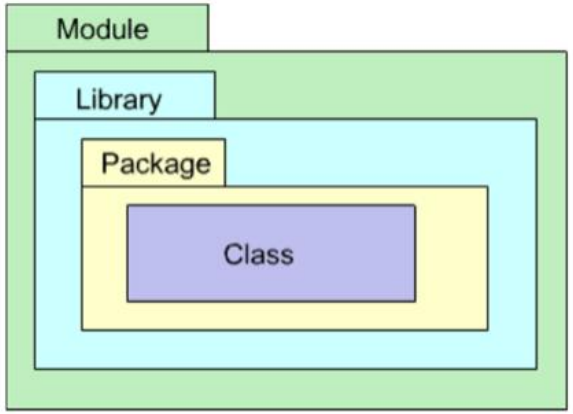
- 工具集前缀：TK
- 类名：<package-name>_<class-name>
- 见名知义：局部变量 aWidthOfBox; 函数参数 theWidth

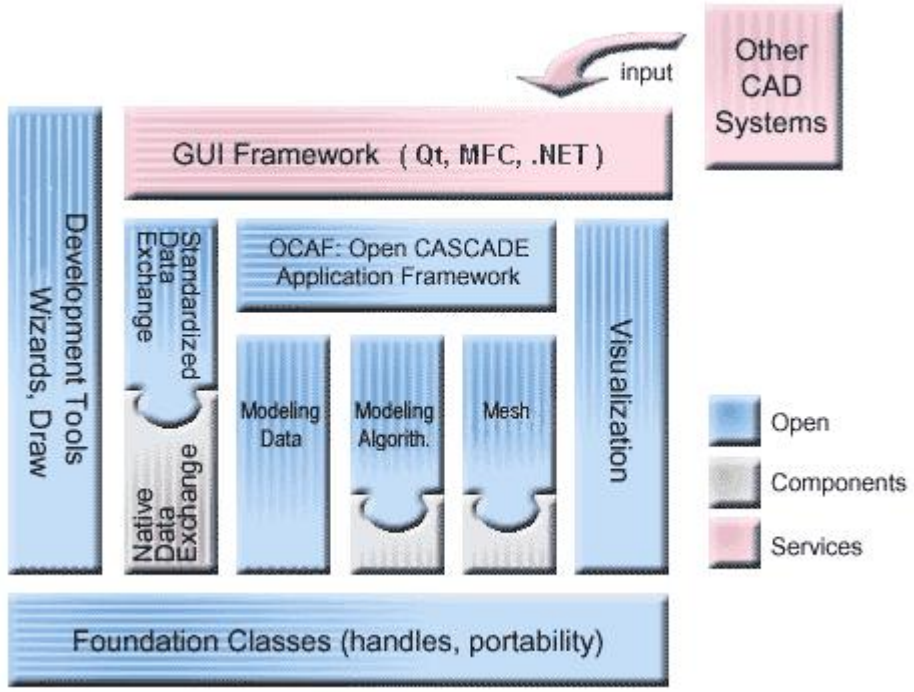
Sample documented class

```
class Package_Class
{

public: //! @name public methods

  //! Method computes the square value.
  //! @param theValue the input value
  //! @return squared value
  Standard_Export Standard_Real Square (const Standard_Real theValue);

private: //! \@name private methods

  //! Auxiliary method
  void increment();

private: //! \@name private fields

  Standard_Integer myCounter; //!< usage counter

};


#include <Package_Class.hxx>
// =======================================================
// function : Square
// purpose  : Method computes the square value
// =======================================================
Standard_Real Package_Class::Square (const Standard_Real theValue)
{
  increment();
  return theValue * theValue;
}

// =======================================================
// function : increment
// purpose  :
// =======================================================
void Package_Class::increment()
{
  ++myCounter;
}
```

模块概览

- OCCT library is designed to be truly modular and extensible, providing C++ classes for:
  - Basic data structures (geometric modeling, visualization, interactive selection and application specific services);
  - Modeling algorithms;
  - Working with mesh (faceted) data;
  - Data interoperability with neutral formats (IGES, STEP);
- The C++ classes and other types are grouped into packages. Packages are organized into toolkits (libraries), to which you can link your application. Finally, toolkits are grouped into seven modules.





Module FoundationClasses

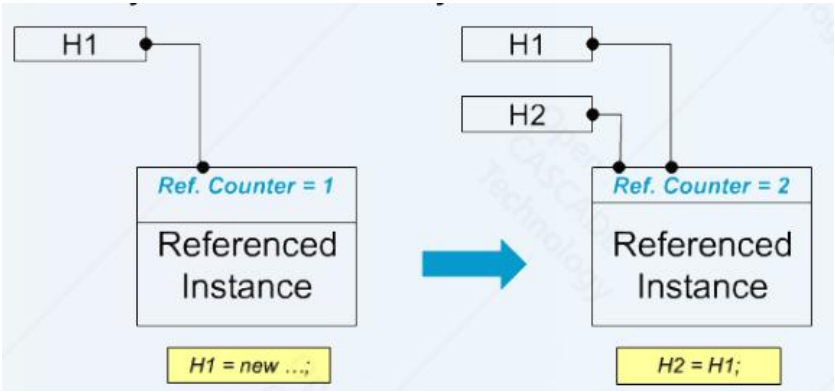
Foundation Classes模块为OCCT更高层的类提供一些数据结构和功能：

- 基本类型定义：Boolean，字符、整数和实数；
- 处理ASCII和Unicode字符串；
- 自定义的容器类：Array, List, Queue, Set, Hash Table(Data Map);
- 常用的数值算法和基本的线性代数计算；
- 基本的代数几何类的数据结构；
- 异常处理类；

FoundationClasses模块还提供一些通用功能：

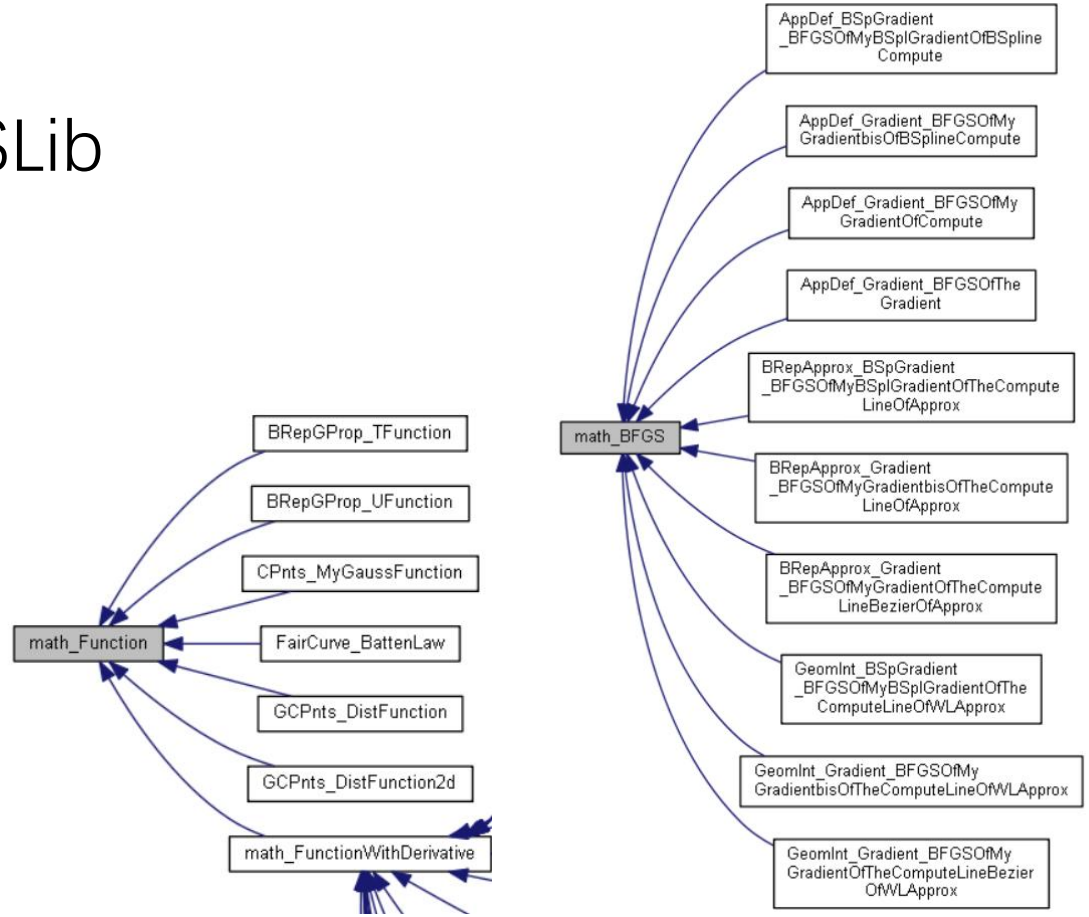
- 智能指针（动态创建对象的安全释放）
- 可配置的内存管理器（内存池）
- 基本的表达式解析器

OCCT class	STL equivalent	C++11 STL equivalent	comment
NCollection_Array	-	std::array	Lower and upper indexes are defined at the construction time
NCollection_Vector	std::vector	-	Indexation starts from 0
NCollection_List	std::list	std::forward_list	OCCT list is one-directional list
NCollection_IndexedMap	std::set	std::unordered_set	OCCT uses hash map instead of tree
NCollection_IndexedDataMap	std::map	std::unordered_map	OCCT uses hash map instead of tree



Module FoundationClasses

- 解析曲线曲面计算EICLib/EISLib
- B样条曲线曲面计算 BSplCLib/BSplSLib
- 幂次多项式计算 PLib
- 数值计算 math



Module ModelingData

- Modeling Data为边界表示（BRep）的3D模型提供数据结构。BRep模型是由拓朴（Topology）和几何（Geometry）来表示的。几何可以理解成模型的数学描述，如曲线和曲面；拓朴将几何对象关联起来。
- 边界表示（BoundaryRepresentation）也称为 BRep 表示，它是几何造型中最成熟、无二 义的表示法。实体的边界通常是由面的并集来表示，而每个面又由它所在的曲面的定义加上 其边界来表示，面的边界是边的并集，而边又是由点来表示的。 边界表示的一个重要特征是描述形体的信息包括几何信息（Geometry）和拓朴信息 （Topology）两个方面。拓朴信息描述形体上的顶点、边、面的连接关系，它形成物体边界 表示的“骨架”。形体的几何信息犹如附着在“骨架”上的肌肉。例如，形体的某个面位于 某一个曲面上，定义这一曲面方程的数据就是几何信息。此外，边的形状、顶点在三维空间 中的位置（点的坐标）等都是几何信息，一般来说，几何信息描述形体的大小、尺寸、位置 和形状等。 在边界表示法中，边界表示就按照体－面－环－边－点的层次，详细记录构成形体的所 有几何元素的几何信息及其相互连接的拓朴关系。这样，在进行各种运算和操作中，就可以 直接取得这些信息。 拓朴是指一个模型中的不同实体之间的关系，它描述了几何实体之间的连接方式。拓朴 定义了一个空间位置不固定的浮动模型。当拓朴实体与几何信息关联在一起时，它的空间位 置才确定。

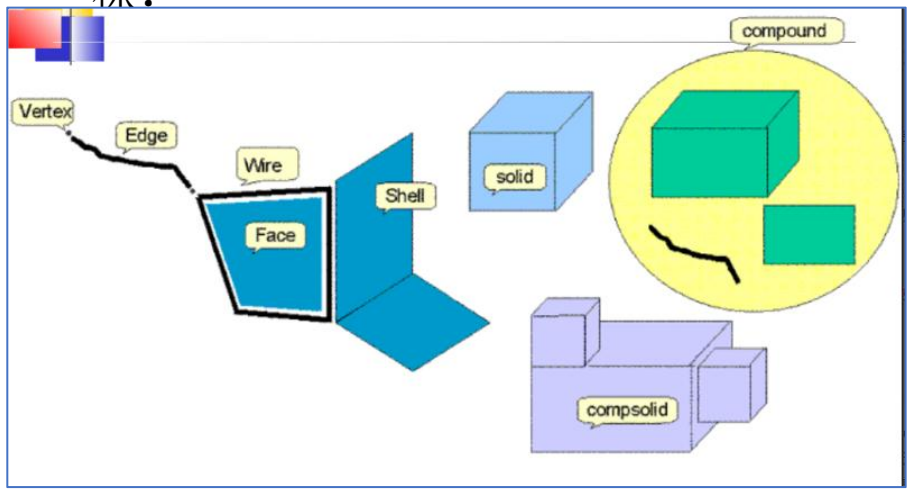
Module ModelingData

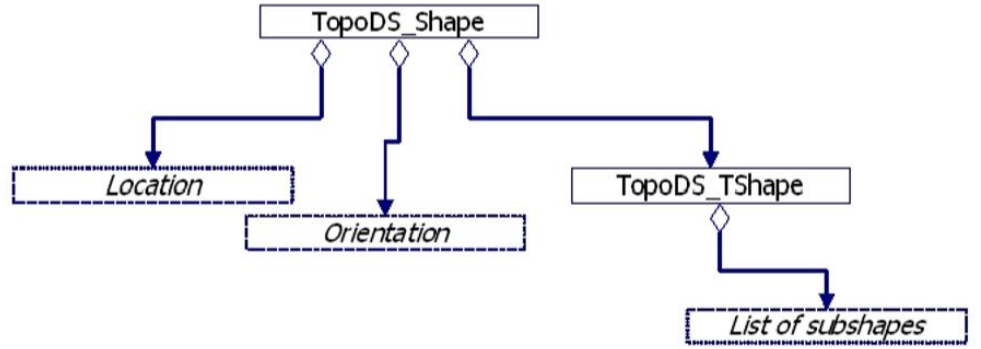
Geometry Versus Topology

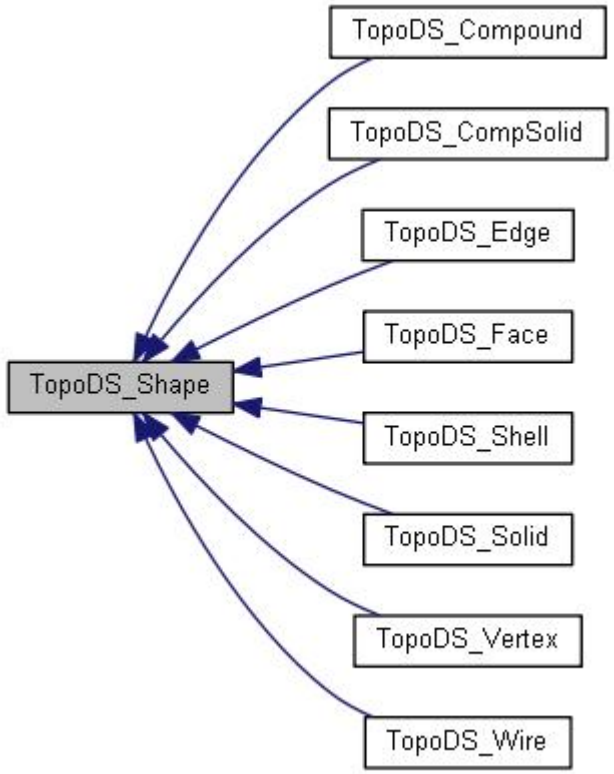
- It is important to understand the differences between geometry and topology when using Open CASCADE.
- **Geometry** is representation of simple shapes which have a mathematical description (Cylinder, Planes, Bezier and B-Splines surfaces...).
- **Topology** involves structuring geometry in order to :
  - Delimit a geometric region limited by boundaries
  - Group regions together to define complex shapes

Module ModelingData

- Topology
  - Vertex：对应一个几何点；
  - Edge：由两个Vertex限定几何曲线得到的有界边；
  - Wire：由相连的Edge形成闭合的环；
  - Face：由Wire限定得到的有界面；
  - Shell：由多个相连面形成的壳；
  - Solid：由壳限定得到的一个闭合体；
  - Compound Solid：多个面相连的Solid体；



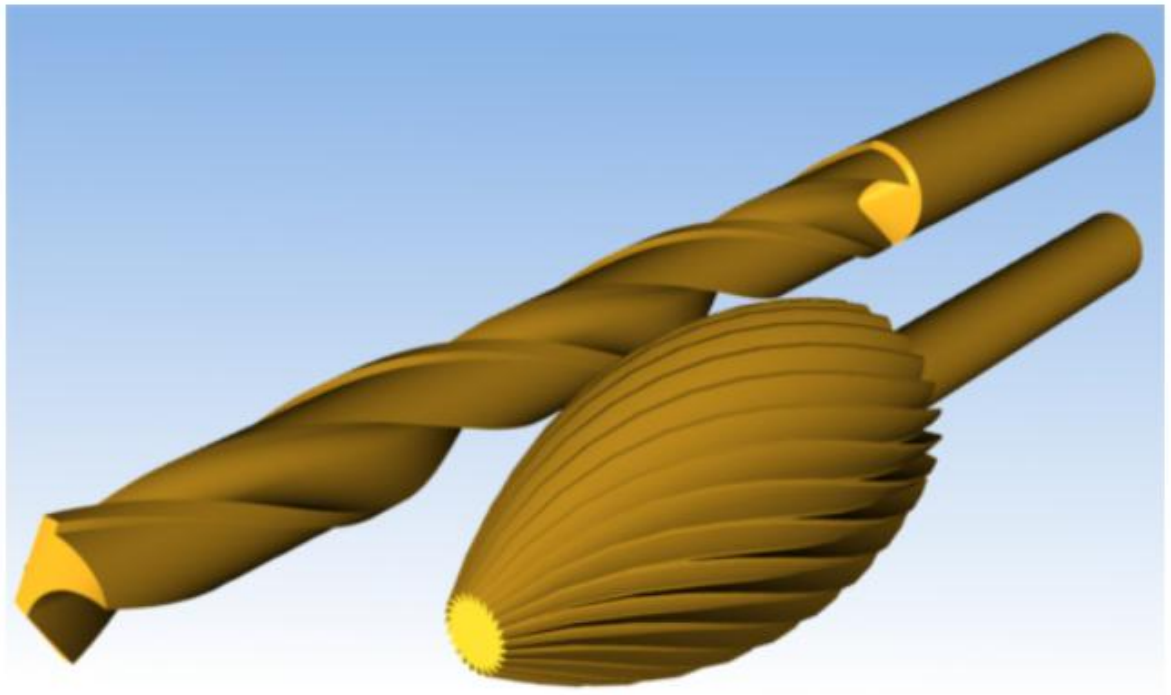




Module ModelingAlgorithms

Topology 拓朴算法

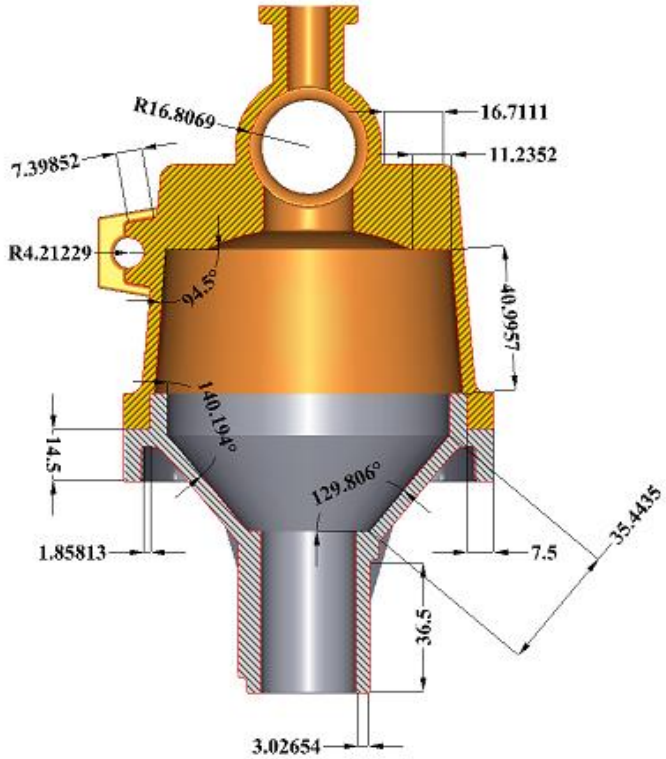
- 模型离散化；
- 计算模型属性：（长度、面积、体积等）；
- 刚性变换：（移动、旋转等）；
- 将模型的几何数据转换成NURBS形式；
- 创建基本体:
  - Box、Cylinder、Cone、Sphere、Torus；
- 扫略造型Sweep:
  - Prism-Linear Sweep;
  - Revolution-Rotational Sweep;
  - Pipes-General-form sweep;
- Boolean Opereations:
  - Common
  - Fuse
  - Cut



Shapes containing pipes with variable radius produced by sweeping

Module Visualization

- Visualization模块提供BRep模型及Mesh数据的可视化功能；
- Visualization模块支持快速交互选择机制；
- Camera驱动的投影处理，便于在正交投影、透视投影中设置投影模式；也便于实现视图的交互功能，如平移、缩放、旋转三维视图；
- 可以设置模型的颜色、材质及透明度；
- 提供三维模型中尺寸标注功能；



Module Data Exchange

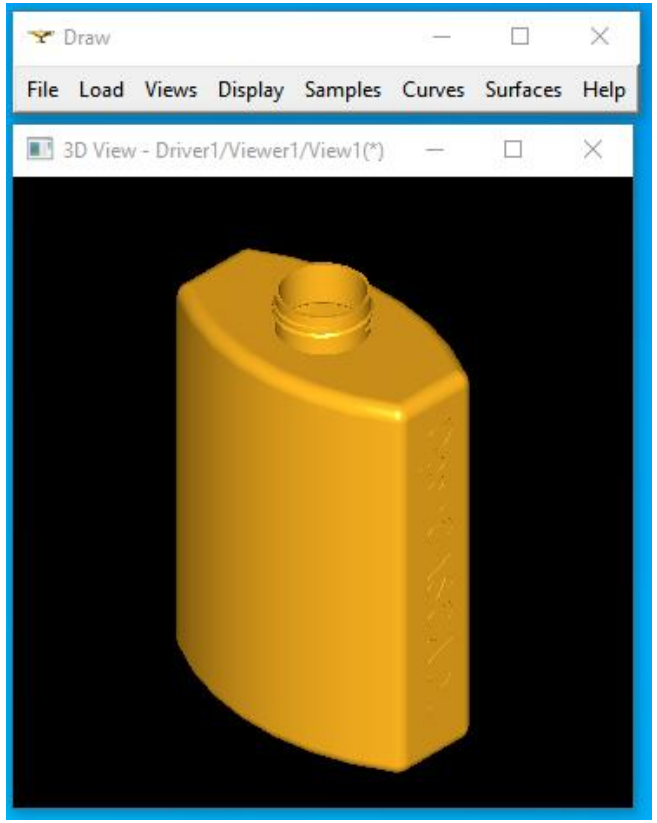
- 标准的DataExchange模块支持导入导出如下格式：
  - STEP(AP203 : Mechanical Design, this covers General 3D CAD; AP214: Automotive Design, AP242)
  - IGES（5.3)
  - glTF(2.0)
  - VRML
  - OBJ
  - STL

Module Application Framework

- Open CASCADE Application Framework(OCAF)基于程序/文档的模式，提供了一个快速开发的框架：
  - 灵活的数据框架；
  - 数据的存储和持久化（open/save）；
  - 支持多文档；
  - Undo-Redo功能；
  - Copy-Paste功能；

Module Draw Test Harness

- Draw Test Harness是一个便于测试OCCT库的功能的工具；
  - 基于脚本Tcl/Tk；
  - 2D和3D视图；
  - 自定义命令；
  - 创建模型，曲线曲面；
  - 造型算法命令；
- Draw Test Harness中实现了OCCT的各个模块的功能命令，可以创建和显示模型，导入导出模型数据，OCAF框架的使用等；



造型数据 ModelingData

Geometry

Non-parametric and parametric geometry

Non-parametric geometry

- These types are manipulated by value.
- These classes have no inheritance.

Additional information can be found in documentation:

- Foundation Classes User's Guide.
- Modeling Data User's Guide.
- Modeling Algorithms User's Guide.

Parametric geometry

- Entities from `Geom (Geom2d)` are manipulated by Handle (useful for data sharing), while controller classes are manipulated by value.
- Hierarchy of classes in general follows STEP (ISO 10303) standard.
- Provide methods to go back and forth from `Geom` to `gp`

Additional information can be found in documentation:

- Modeling Data User's Guide.
- Modeling Algorithms User's Guide.

Non-parametric geometry

Model classes (two-dimensional classes are available via adding “2d” suffix, `gp_Pnt2d`):

- `gp_Pnt` – Cartesian point.
- `gp_Vec` – Vector.
- `gp_Dir` – Direction (non-null vector with magnitude equal to 1.0).
- `gp_Trsf` – Euclidean transformation. It is possible to set translation, rotation, and scaling independently.
- `gp_Ax1` – Axis. Axis is point plus direction.
- `gp_Lin`, `gp_Circ`, `gp_Elips`, `gp_Hypr`, `gp_Parab`, `gp_Cylinder`, `gp_Sphere`, `gp_Torus` – primitives representing curves and surfaces.

Controller classes:

- Direct construction – `gce_MakeCircle`, `gce_MakeLin`
- Constrained construction (2d only) – `GccAna_Circ2d2TanRad`

Limitation of non-parametric geometry

Non-parametric geometry provides useful set classes, but there are some principle limitations with them:

- **Typical geometric questions cannot be answered:**
  - What is the value of curvature at this point?
  - What is the tangent vector to curve at this point?
  - What is the minimum Euclidean distance between the curve and the given point?
  - Do these objects intersect?
- **Some objects are infinite, and there is no way to make them finite:**
  - Line, Hyperbola, Parabola.
  - Plane, Cylinder.
- **It is not possible to represent free-form and non-trivial objects:**
  - How to represent aircraft fuselage? (Bezier and B-spline).
  - How to represent offset surface? (normal is required).
  - How to represent sweeping surfaces? (linear extrusion and revolution).

Parametric geometry

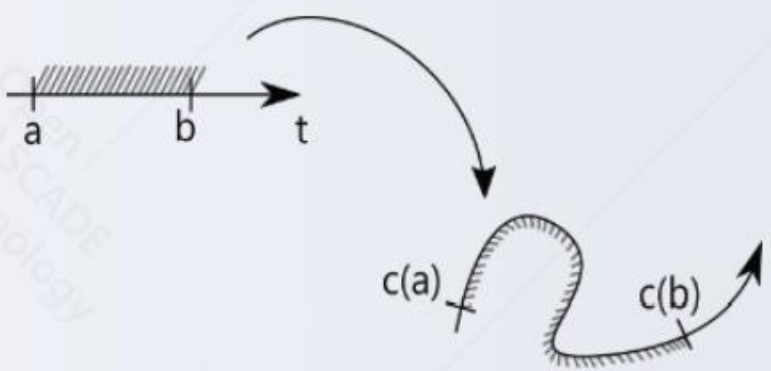
Model classes (two-dimensional classes are available in the `Geom2d` package):

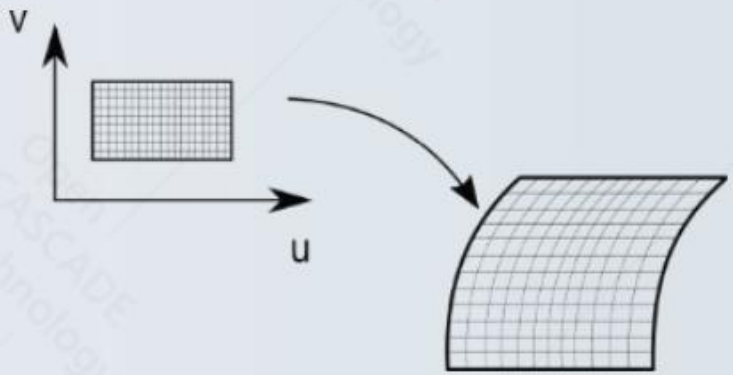
- **Curves – descendants of the** `Geom_Curve`:
  - `Geom_Line`
  - Conics: `Geom_Circle`, `Geom_Ellipse`, `Geom_Hyperbola`, `Geom_Parabola`
  - Free-form: `Geom_BSplineCurve`, `Geom_BezierCurve`
  - `Geom_OffsetCurve`
  - Trimming concept: `Geom_TrimmedCurve`
- **Surface – descendants of the** `Geom_Surface`:
  - Elementary surfaces: `Geom_Plane`, `Geom_CylindricalSurface`, `Geom_SphericalSurface`, `Geom_ToroidalSurface`, `Geom_ConicalSurface`
  - Free-form: `Geom_BSplineSurface`, `Geom_BezierSurface`
  - Sweeping surfaces: `Geom_SurfaceOfLinearExtrusion`, `Geom_SurfaceOfRevolution`
  - `Geom_OffsetSurface`
  - Trimming concept: `Geom_RectangularTrimmedSurface`

Parametric geometry

Controller classes (two-dimensional classes are available via adding “2d” suffix to package name, gce2d):

- Direct construction- `gce_MakeCircle, gce2d_MakeCircle`
- Constrained construction (2d only) – `Geom2dGcc_Circl2d3Tan`



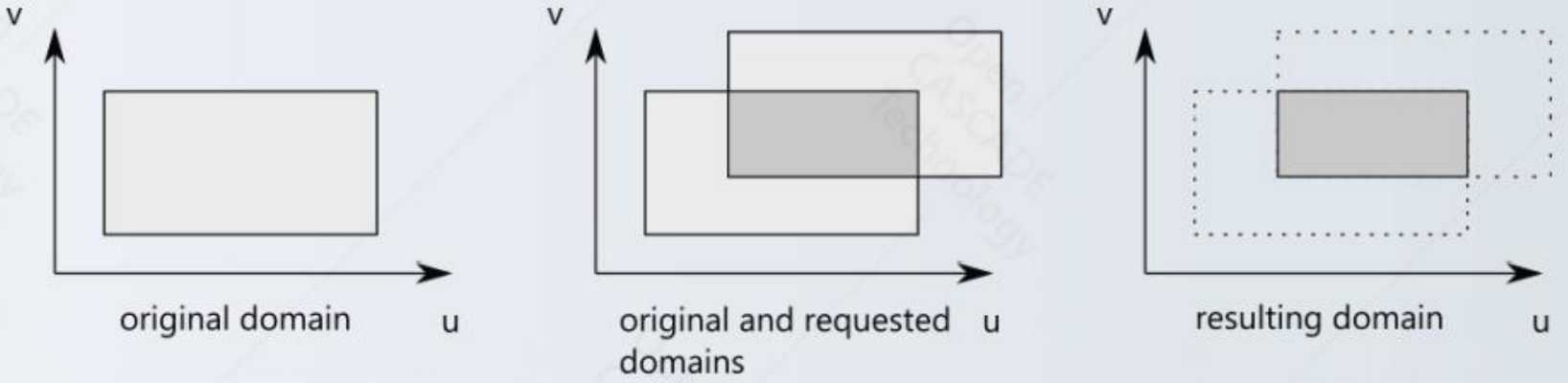


Trimming concept

Some parametric objects, like line (`Geom_Line`) or plane (`Geom_Plane`), are infinite as their non-parametric counterparts. How to bound them?

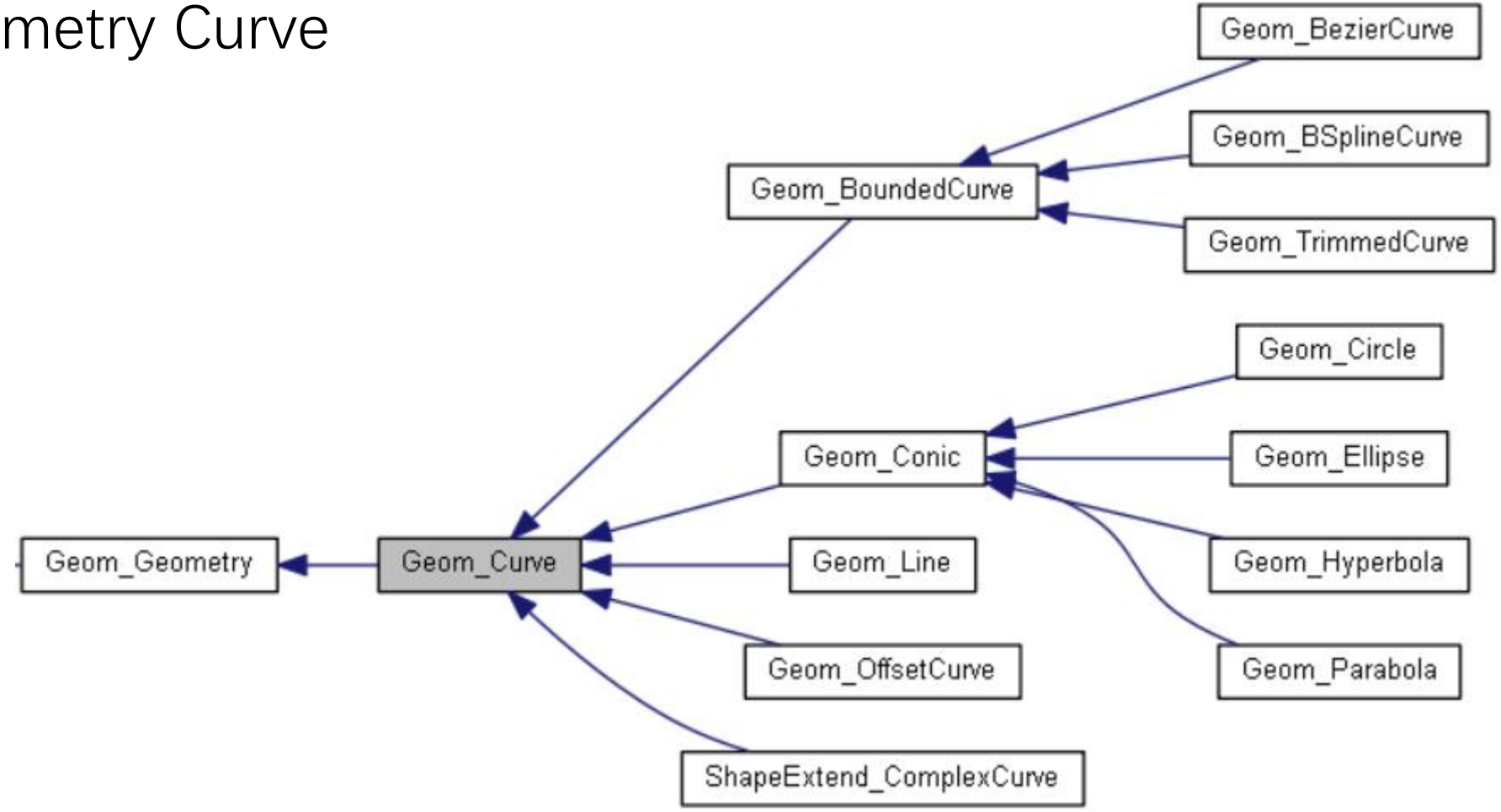
- Curve – bound using starting and finishing parameter – `Geom_TrimmedCurve`
- Surface – bound using rectangular domain – `Geom_RectangularTrimmedSurface`

The OCCT has no protection from surface evaluation outside boundaries. This functionality is used in high-level algorithms such as offset algorithm, but it is recommended to avoid evaluation outside the parametric domain.



ModelingData

- Geometry Curve



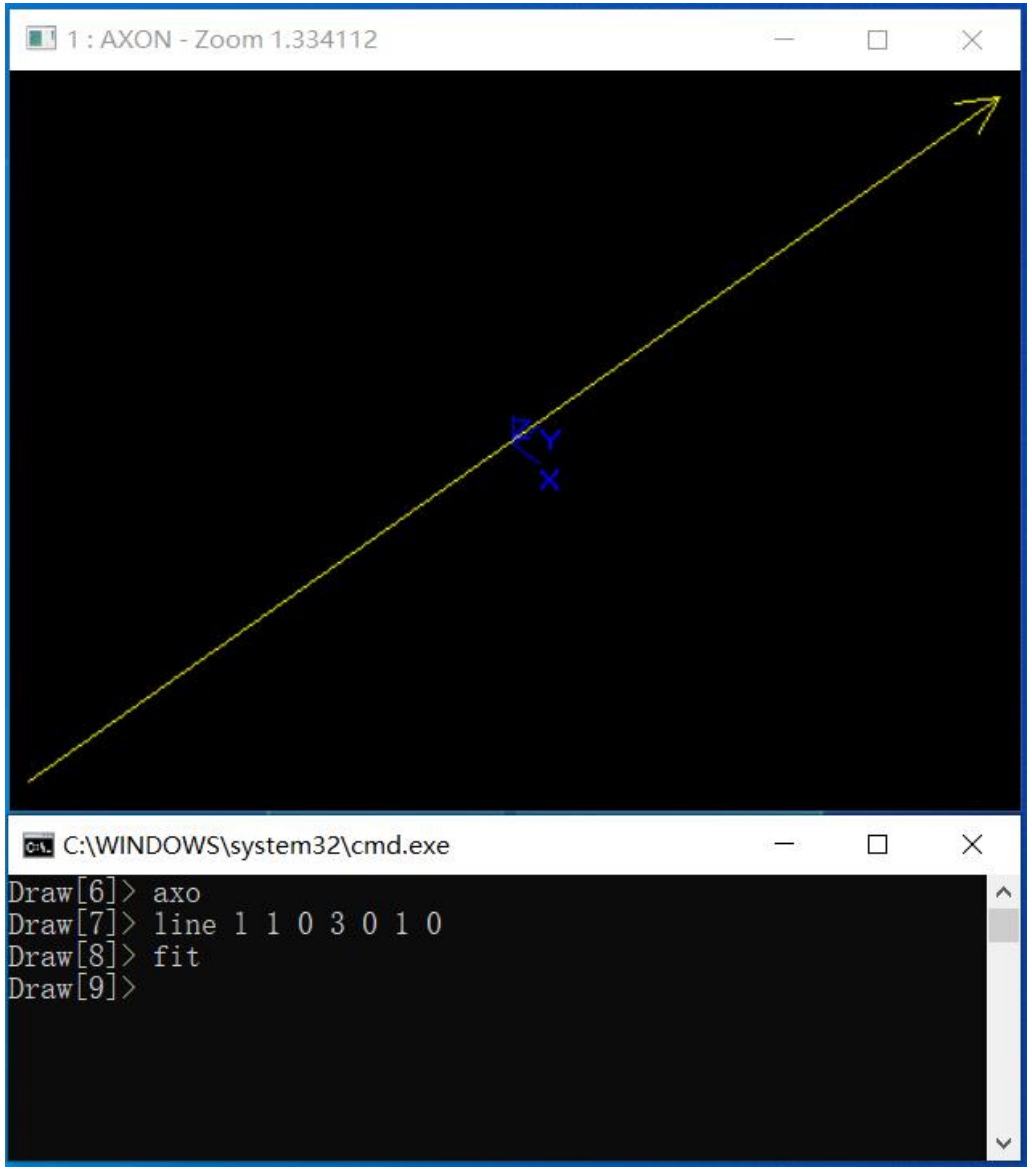
Geometry Curve – Line

直线数据包含一个三维点P和三维方向D，
其参数方程为：

$$C(u) = P + u \cdot D,\ u \in (-\infty, \infty).$$

如：经过点(1, 0, 3)，方向为(0, 1, 0)的
直线参数方程为：

$$C(u) = (1,0,3) + u \cdot (0,1,0)$$



Geometry Curve – Circle

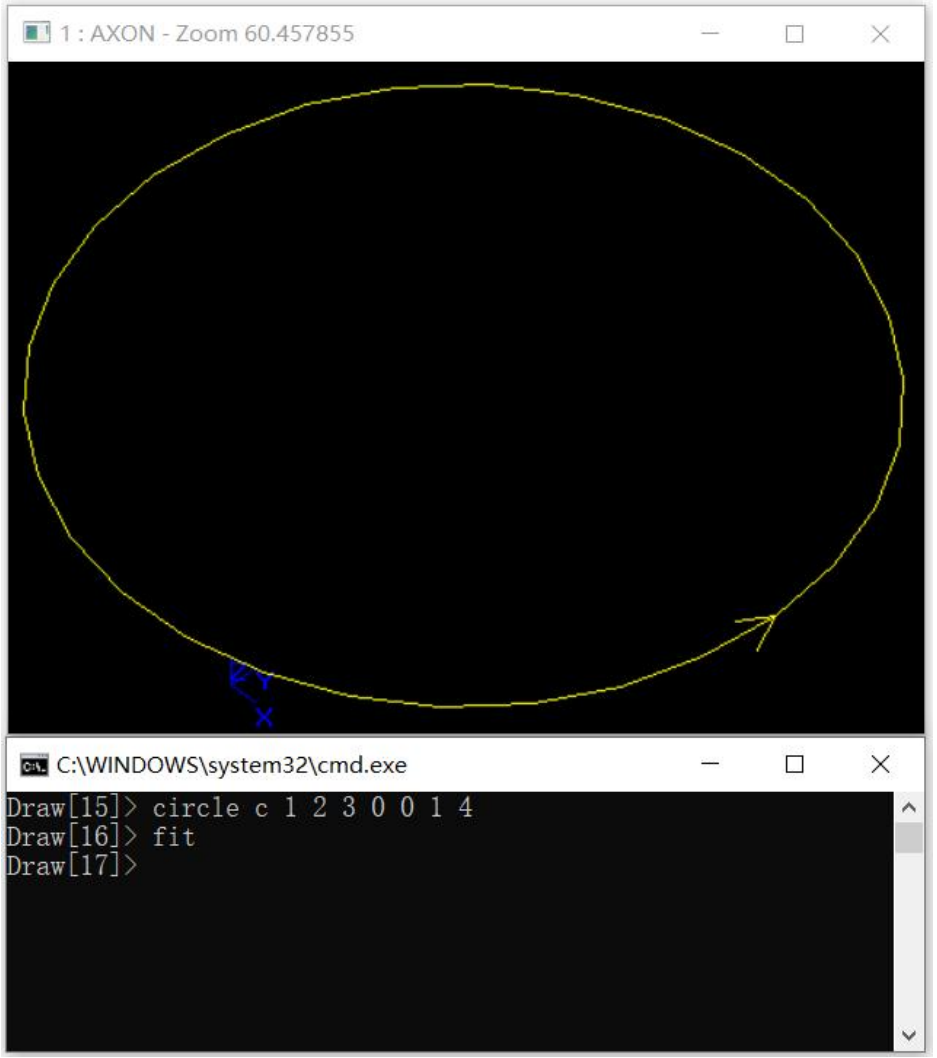
圆的数据包含表示圆心坐标的三维点P，三个方向N，Dx，Dy表示的坐标系和半径r。其参数方程为：

$$C(u) = P + r \cdot (\cos(u) \cdot D_X + \sin(u) \cdot D_Y),\ u \in [0, 2 \cdot \pi).$$

圆心坐标为（1,2,3），半径r为（4），圆所在平面的法向N为（0,0,1），圆的X方向（1,0,0）和Y方向为（0,1,0），其参数方程为：

$$C(u) = (1,2,3) + 4 \cdot (\cos(u) \cdot (1,0,-0) + \sin(u) \cdot (0,1,0))$$

$$X^2 + Y^2 = R^2$$



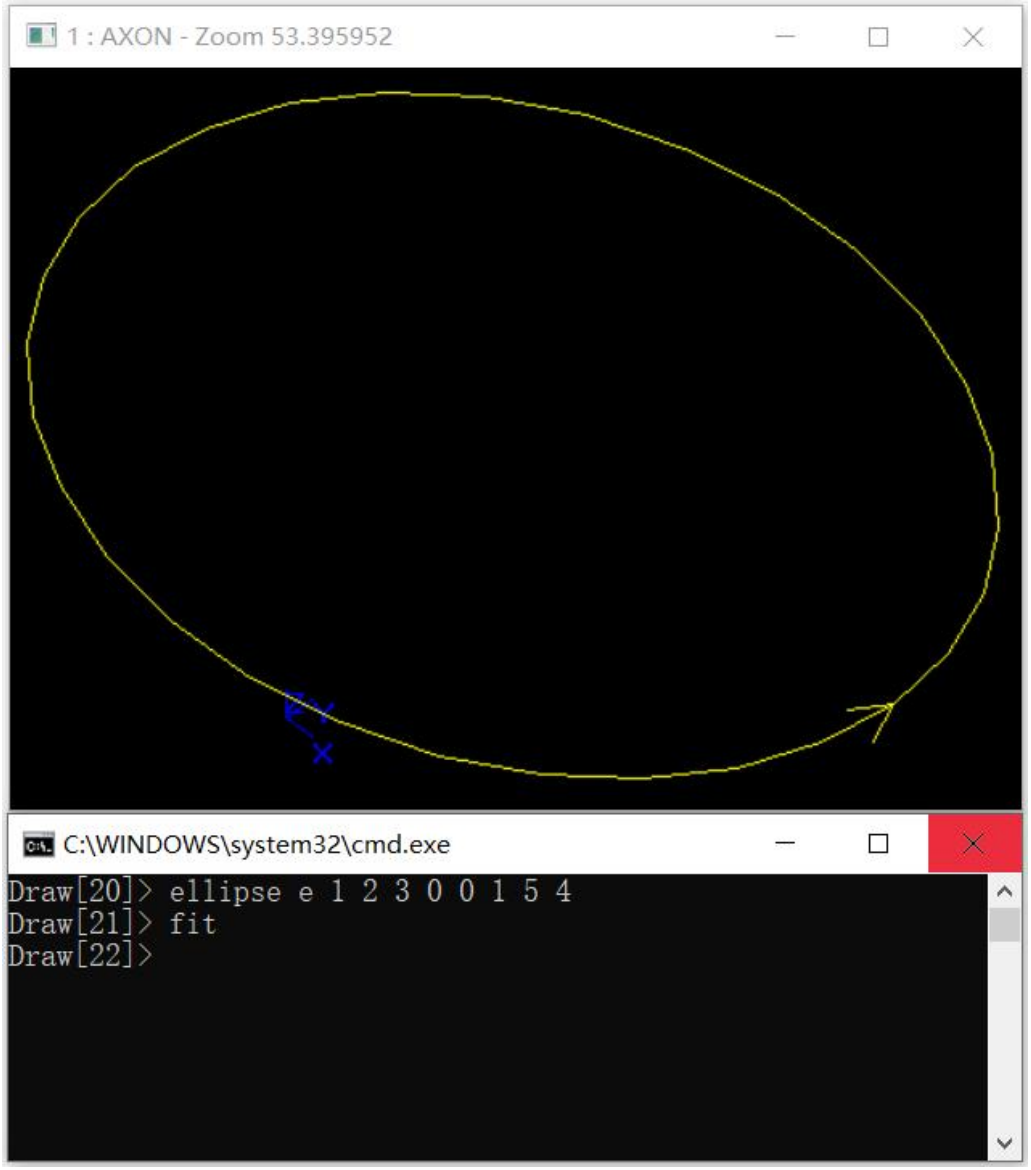
Geometry Curve – Ellipse

椭圆的数据包含三维点P，三维正交坐标系N、Dmaj、Dmin和两个非负实数rmaj和rmin，且rmin<=rmaj。椭圆位于中心点P，法向量为N的平面上，且长轴、短轴的方向分别为Dmaj, Dmin，长轴、短轴上的半径分别为rmaj, rmin。椭圆的参数方程为：

$$C(u) = P + r_{maj} \cdot \cos(u) \cdot D_{maj} + r_{min} \cdot \sin(u) \cdot D_{min},\ u \in [0, 2 \cdot \pi).$$

椭圆的中心点P=（1，2，3），平面的法向量N=（0，0，1），长轴方向Dmaj=（1，0，－0），短轴方向Dmin=（－0，1，0），长轴半径为5，短轴半径为4表示的参数方程为：

$$C(u) = (1,2,3) + 5 \cdot \cos(u) \cdot (1,0,-0) + 4 \cdot \sin(u) \cdot (0,1,0).$$



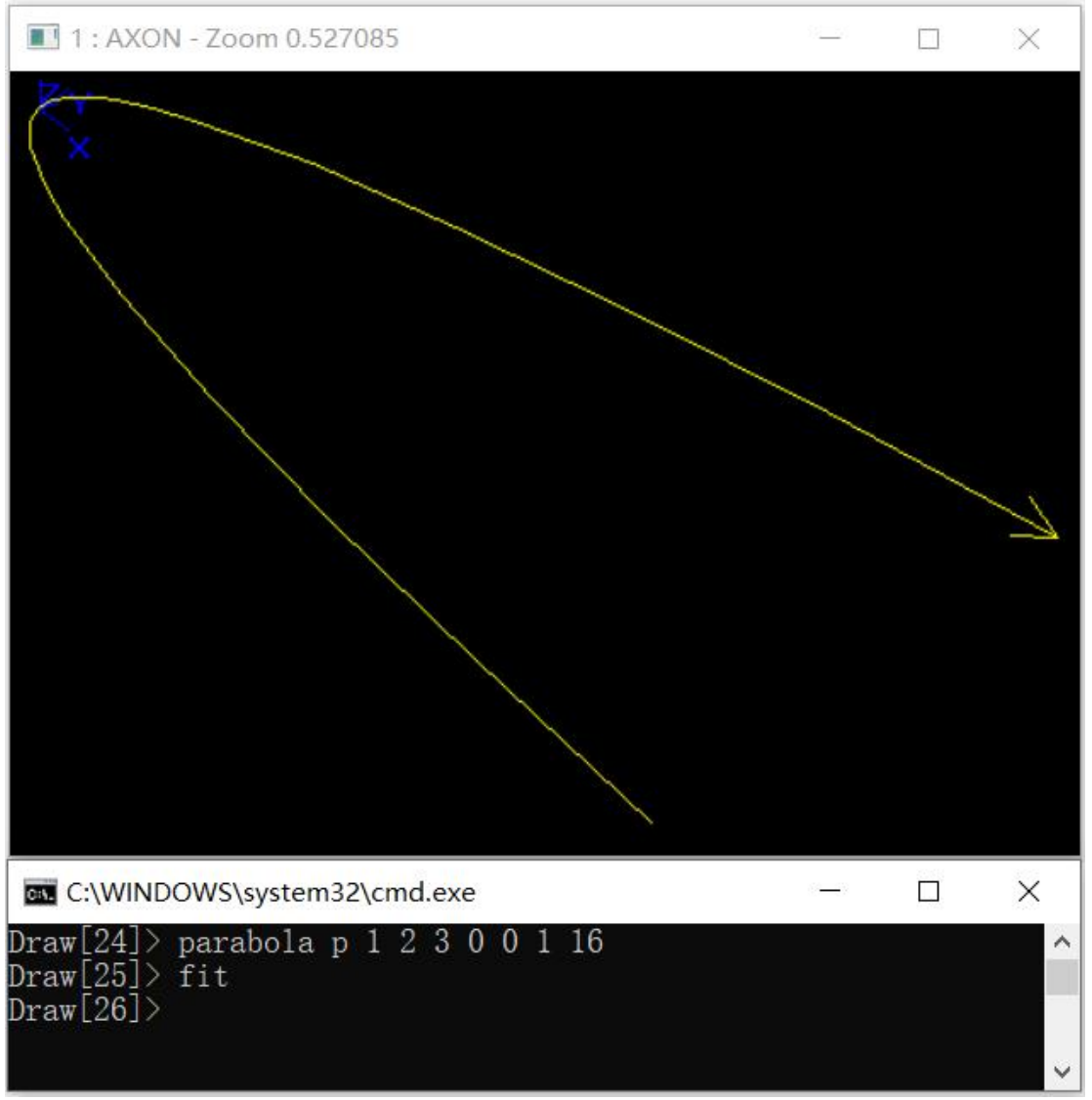
Geometry Curve – Parabola

抛物线数据包含三维点P，三维正交坐标系坐标轴方向N，Dx，Dy和一个非负的实数f。抛物线通过点P，且位于法向量为N的平面上，焦点长度为f，其参数方程为:

$$C(u)=P+\frac{u^2}{4\cdot f}\cdot D_X+u\cdot D_Y,\ u\in(-\infty,\infty)\Leftarrow f\neq 0;$$

抛物线过点P＝（1，2，3），位于平面的法向N＝（0，0，1），抛物线的另两个轴方向Dx＝（1，0，－0），Dy＝（－0，1，0），焦点长度f＝16。参数方程为:

$$C(u)=(1,2,3)+\frac{u^2}{64}\cdot(1,0,-0)+u\cdot(-0,1,0).$$



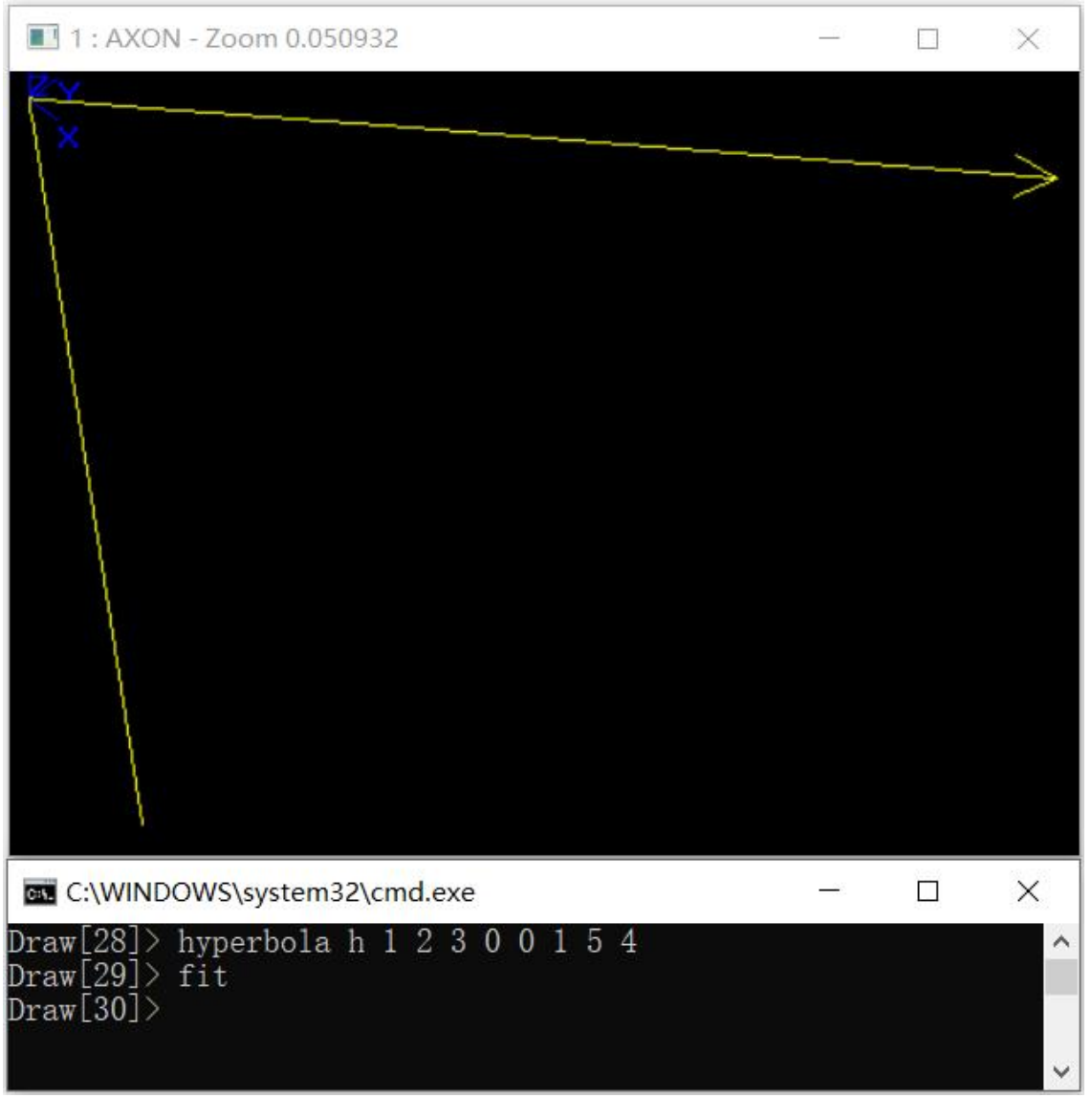
Geometry Curve – Hyperbola

双曲线定义数据有三维点P，三维正交坐标系坐标轴方向为N，Dx，Dy和两个非负实数Kx，Ky。双曲线过P点且法向量为N的平面上，其参数方程如下所示：

$$C(u) = P + k_X \cdot \cosh(u) \cdot D_X + k_Y \cdot \sinh(u) \cdot D_Y,\ u \in (-\infty, \infty).$$

过点P＝（1，2，3）且位于的平面的法向N＝（0，0，1），其它的数据Dx＝（1，0，－0），Dy＝（－0，1，0），Kx＝5和Ky＝4。其参数方程为：

$$C(u) = (1,2,3) + 5 \cdot \cosh(u) \cdot (1,0,0) + 4 \cdot \sinh(u) \cdot (0,1,0).$$



Geometry Curve – Bezier

Bezier曲线数据包含有理标志r，曲线的次数m（degree m <= 25）和带权的控制点（weight poles）。当有理标志位r = 0时，weight poles就是m+1个三维点：B0，B1...Bn；当有理标志位r = 1时，weight poles就是带权的控制点B0 h0... Bm hm。Bi是三维点，hi是[0，m]正实数，即权因子。当有理标志位r = 0时，即不是有理Bezier曲线时，hi = 1。Bezier曲线参数方程如下所示：

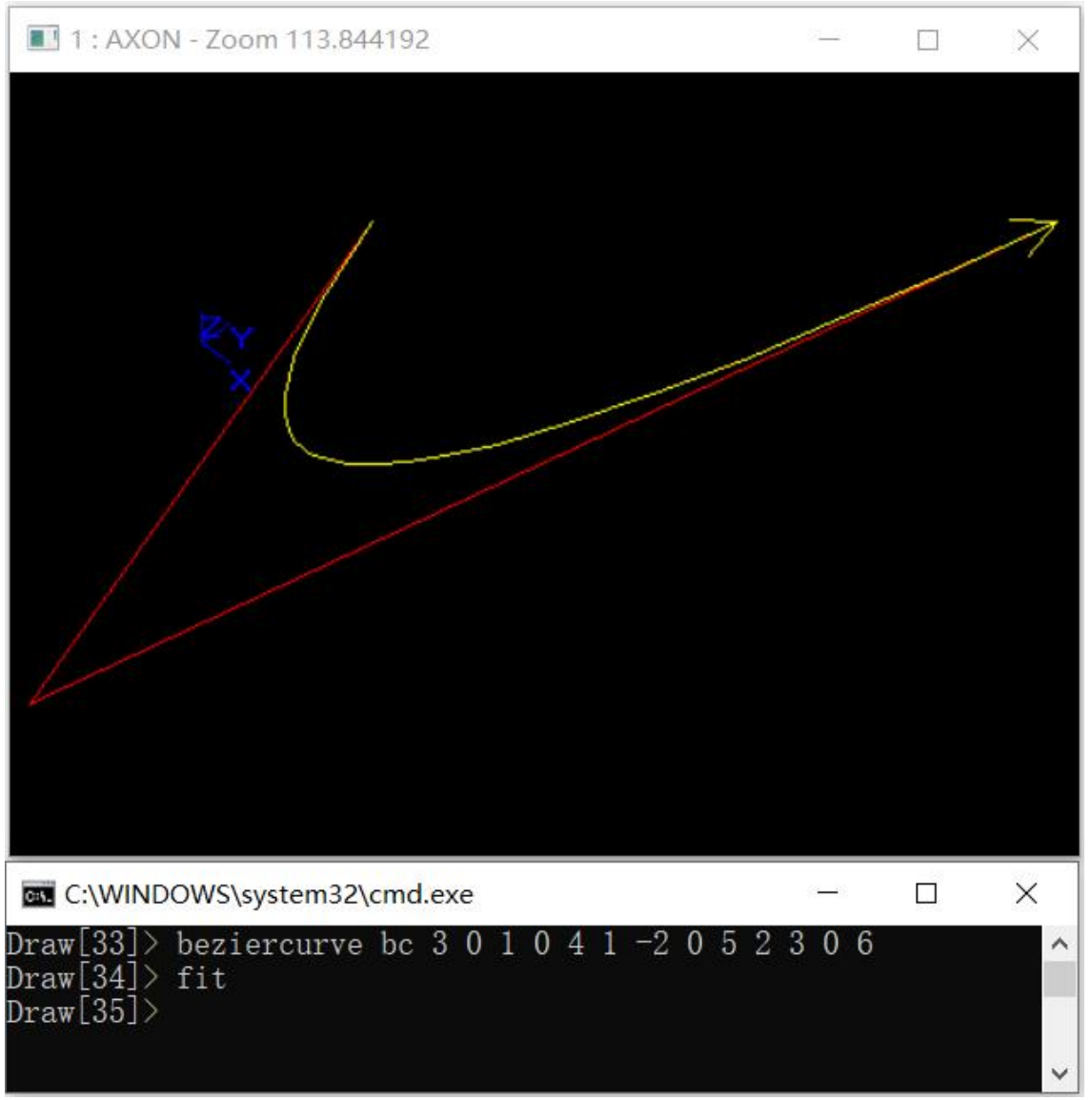
$$C(u)=\frac{\sum_{i=0}^{m} B_i \cdot h_i \cdot C_m^i \cdot u^i \cdot (1-u)^{m-i}}{\sum_{i=0}^{m} h_i \cdot C_m^i \cdot u^i \cdot (1-u)^{m-i}},\ u \in [0,1]$$

示例数据表示的Bezier曲线是有理Bezier曲线，因其有理标志位r = 1，次数m = 2，带权控制点及权因子分别为：B0 = （0，1，0），h0=4，B1 = （1，－2，0），h1 = 5，B2 = （2，3，0），h2 = 6。Bezier曲线的参数方程如下所示：

$$C(u)=\frac{(0,1,0)\cdot 4\cdot(1-u)^2+(1,-2,0)\cdot 5\cdot 2\cdot u\cdot(1-u)+(2,3,0)\cdot 6\cdot u^2}{4\cdot(1-u)^2+5\cdot 2\cdot u\cdot(1-u)+6\cdot u^2}.$$

Geometry Curve – Bezier

- 在Draw Test Harness中绘制Bezier曲线



Geometry Curve – BSpline

B-Spline曲线包含了有理标志位r，曲线次数m<=25，控制点数n>=2，节点数k，带权控制点wieght poles（x, y, z, w）和节点向量multiplicity knots。

[0 0 0 0.25 0.25 0.5 0.5 0.75 0.75 1 1 1] -> ***Knot Sequence***

[0 0.25 0.5 0.75 1] -> ***Knots***

[3 2 2 2 3] -> ***Mults***

$$u_i < u_{i+1}\ (1 \le i \le k-1),$$

$$q_1 \le m+1,\ q_k \le m+1,\ q_i \le m\ (2 \le i \le k-1),\ \sum_{i=1}^{k} q_i = m+n+1.$$

Geometry Curve – BSpline

- B-Spline曲线的参数方程

$$C(u)=\frac{\sum_{i=1}^{n} B_i \cdot h_i \cdot N_{i,m+1}(u)}{\sum_{i=1}^{n} h_i \cdot N_{i,m+1}(u)},\ u \in [u_1, u_k]$$

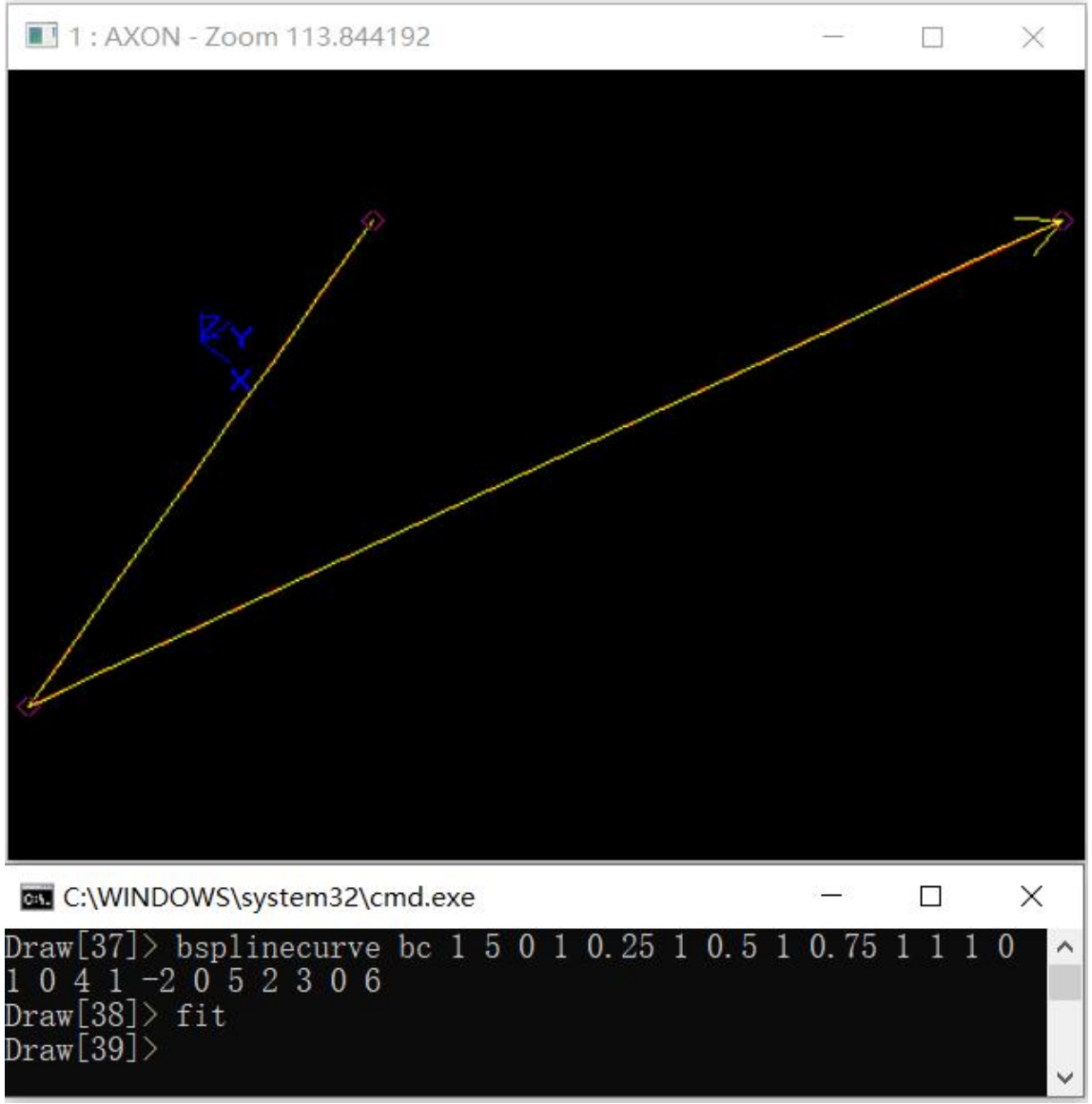
其中基函数Ni,j有如下的递归定义:

$$N_{i,1}(u)=\begin{cases}1 \Leftarrow \bar{u}_i \le u < \bar{u}_{i+1} \\ 0 \Leftarrow u < \bar{u}_i \vee \bar{u}_{i+1} \le u\end{cases},\ N_{i,j}(u)=\frac{(u-\bar{u}_i)\cdot N_{i,j-1}(u)}{\bar{u}_{i+j-1}-\bar{u}_i}+\frac{(\bar{u}_{i+j}-u)\cdot N_{i+1,j-1}(u)}{\bar{u}_{i+j}-\bar{u}_{i+1}}\ (2 \le j \le m+1)$$

Geometry Curve – BSpline

B样条曲线为：有理标志位r = 1，次数m = 1，控制点数n = 3，节点数k = 5，带权控制点：B1 = （0，1，0），h1 = 4，B2 = （1，－2，0），h2 = 5，B3 = （2，3，0），h3 = 6；节点及其重数u1 = 0，q1 = 1，u2 = 0.25，q2 = 1，u3 = 0.5，q3 = 1，u4 = 0.75，q4 = 1，u5 = 1，q5 = 1。B-Spline曲线的参数方程如下所示：

$$C(u)=\frac{(0,1,0)\cdot 4\cdot N_{1,2}(u)+(1,-2,0)\cdot 5\cdot N_{2,2}(u)+(2,3,0)\cdot 6\cdot N_{3,2}(u)}{4\cdot N_{1,2}(u)+5\cdot N_{2,2}(u)+6\cdot N_{3,2}(u)}.$$



Geometry Curve – Trimmed & Offset

- Trimmed Curve: 截剪曲线是基线加上参数范围限制

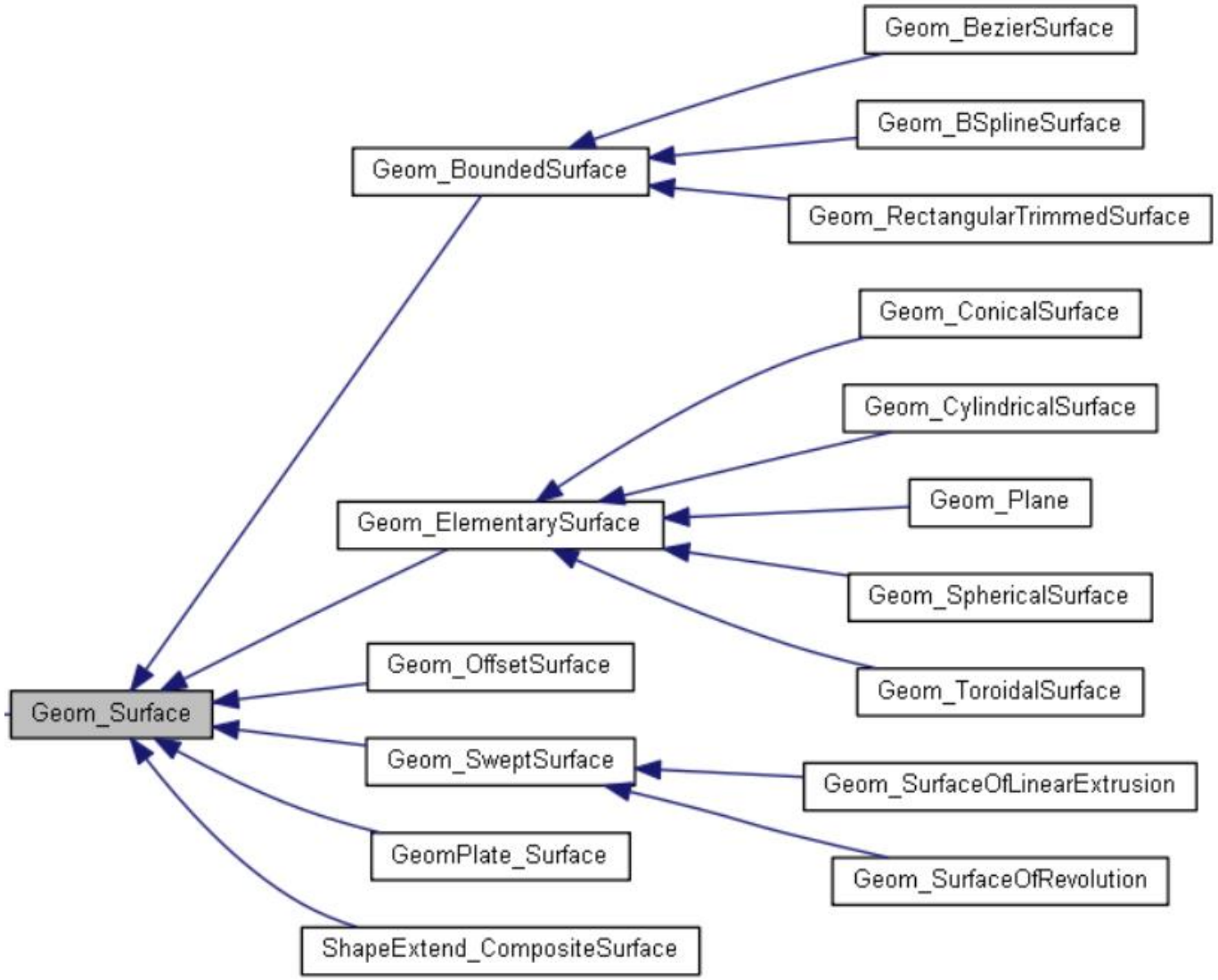
$$C(u) = B(u),\ u \in [u_{\min}, u_{\max}].$$

- Offset Curve： 偏移曲线是基线加上偏移方向和距离

$$C(u) = B(u) + d \cdot \frac{[B'(u), D]}{\|[B'(u), D]\|},\ u \in \text{domain}(B).$$

ModelingData

- Geometry Surface



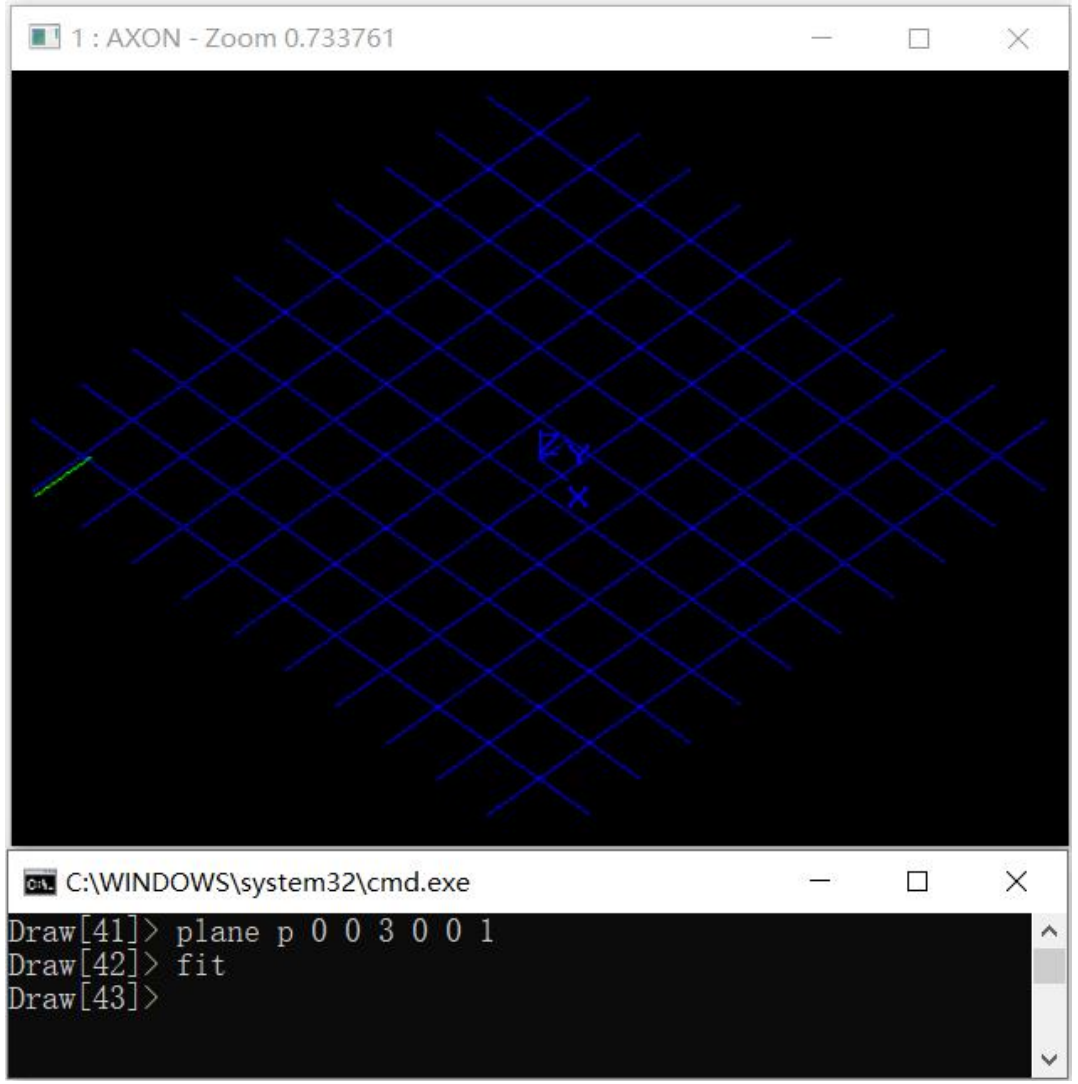
Geometry Surface – Plane

平面数据包含三维点P和三维正交坐标系N，Du，Dv。平面通过点P，且其法向量为N。其参数方程如下所示：

$$S(u,v)=P+u\cdot D_u+v\cdot D_v,\ (u,v)\in(-\infty,\infty)\times(-\infty,\infty).$$

通过点P＝（0，0，3），法向量N＝（0，0，1）的平面参数方程如下所示：

$$S(u,v)=(0,0,3)+u\cdot(1,0,0)+v\cdot(0,1,0).$$



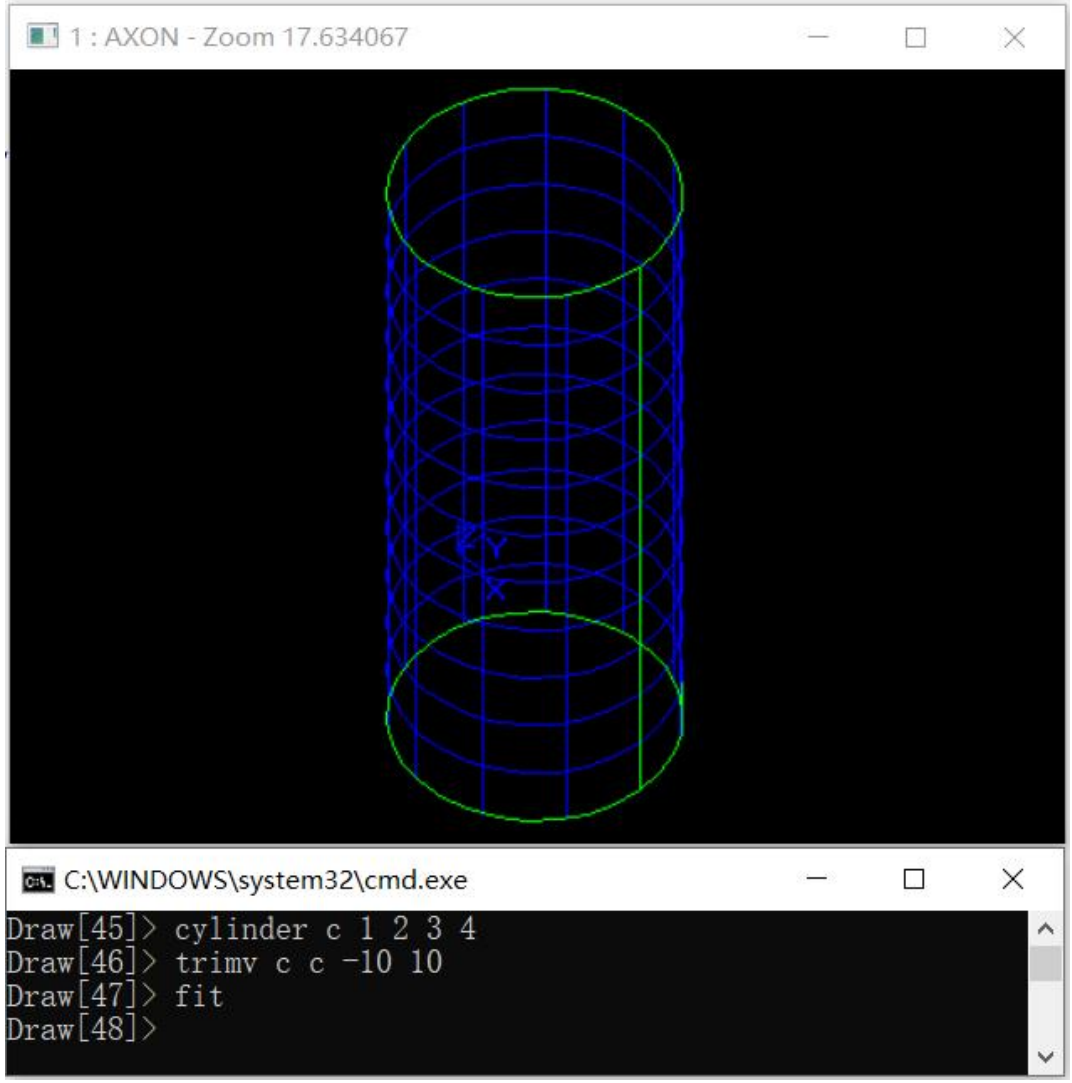
Geometry Surface – Cylinder

圆柱面的数据包含三维点P，三维正交坐标系Dv，Dx，Dy和一个非负实数r。圆柱面的轴通过点P，方向为Dv，圆柱面的半径为r，其参数方程如下所示：

$$S(u,v)=P+r\cdot(\cos(u)\cdot D_X+\sin(u)\cdot D_Y)+v\cdot D_v,\ (u,v)\in[0,2\cdot\pi)\times(-\infty,\infty).$$

轴通过点P＝（1，2，3），轴的方向Dv＝（0，0，1），方向Dx＝（1，0，－0），Dy＝（－0，1，0），半径r＝4，圆柱的参数方程如下所示：

$$S(u,v)=(1,2,3)+4\cdot(\cos(u)\cdot D_X+\sin(u)\cdot D_Y)+v\cdot D_v.$$



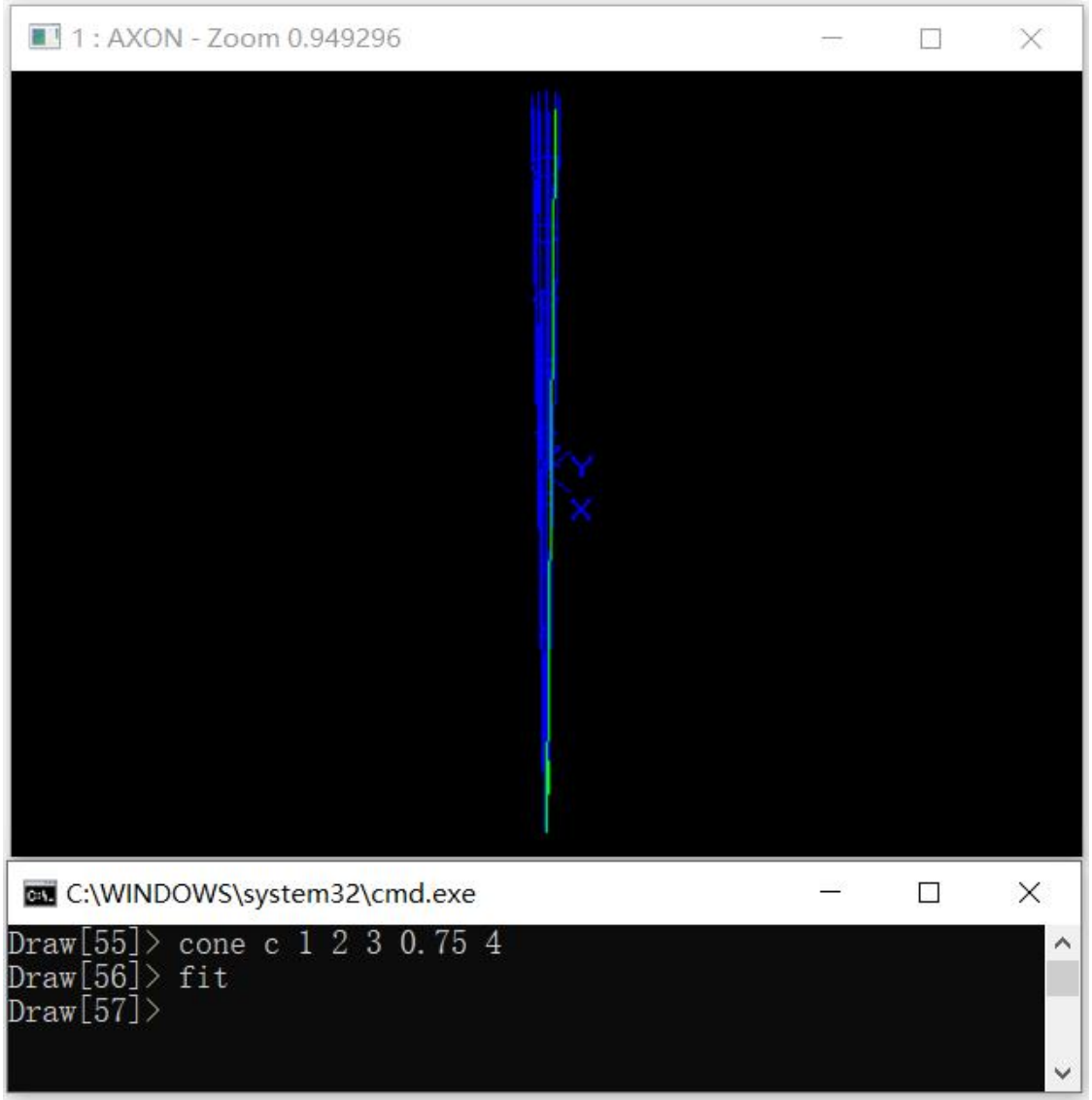
Geometry Surface – Cone

圆锥面的数据包含三维点P，正交坐标系Dz，Dx，Dy，非负实数r和实数ψ（范围为（-π/2, π/2））。圆锥面通过点P且轴的方向为Dz。过点P且与方向Dx，Dy平行的平面为圆锥面的参考平面（referenced plane）。参考平面截圆锥面为一个圆，其半径为r。其参数方程如下所示：

$$S(u,v)=P+(r+v\cdot\sin(\varphi))\cdot(\cos(u)\cdot D_X+\sin(u)\cdot D_Y)+v\cdot\cos(\varphi)\cdot D_Z,\ (u,v)\in[0,2\cdot\pi)\times(-\infty,\infty).$$

轴通过点P =（1，2，3），方向Dz =（0，0，1）。圆锥面的其他数据是Dx =（1，0，－0），Dy =（－0，1，0），半径r = 4，角度ψ = 0.75的圆锥参数方程如下所示：

$$S(u,v)=(1,2,3)+(4+v\cdot\sin(0.75))\cdot(\cos(u)\cdot(1,0,-0)+\sin(u)\cdot(-0,1,0))+v\cdot\cos(0.75)\cdot(0,0,1).$$



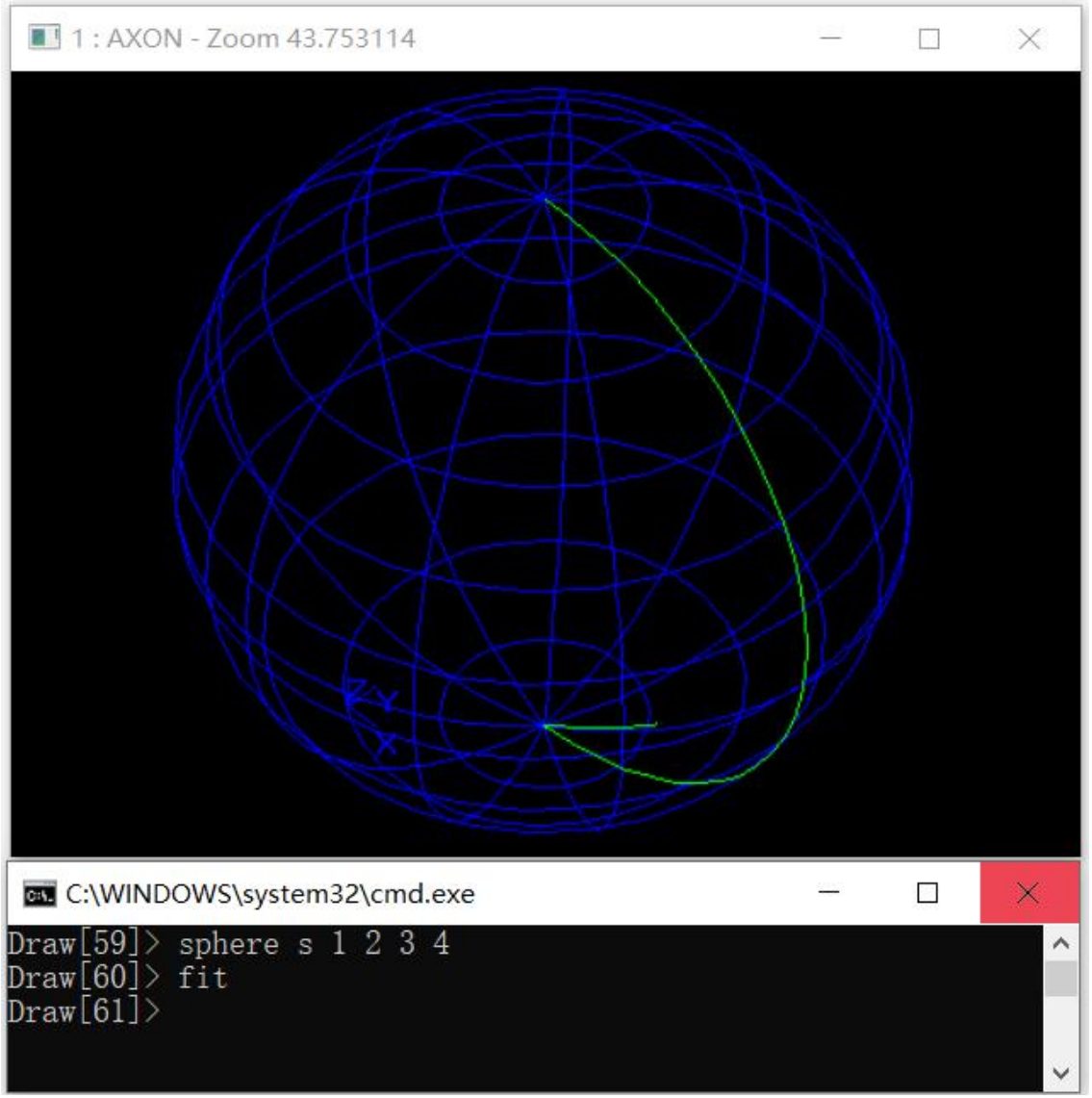
Geometry Surface – Sphere

球面的数据包含三维点P，三维正交坐标系Dz，Dx，Dy和非负实数r。即球面的球心为点P，半径为r，其参数方程如下所示：

$$S(u,v)=P+r\cdot\cos(v)\cdot(\cos(u)\cdot D_X+\sin(u)\cdot D_Y)+r\cdot\sin(v)\cdot D_Z,\ (u,v)\in[0,2\cdot\pi)\times[-\pi/2,\pi/2].$$

球心过点P＝（1，2，3），方向分别为Dz＝（0，0，1），Dx＝（1，0，－0），Dy＝（－0，1，0），半径r＝4的球面参数方程如下所示：

$$S(u,v)=(1,2,3)+4\cdot\cos(v)\cdot(\cos(u)\cdot(1,0,-0)+\sin(u)\cdot(-0,1,0))+4\cdot\sin(v)\cdot(0,0,1).$$



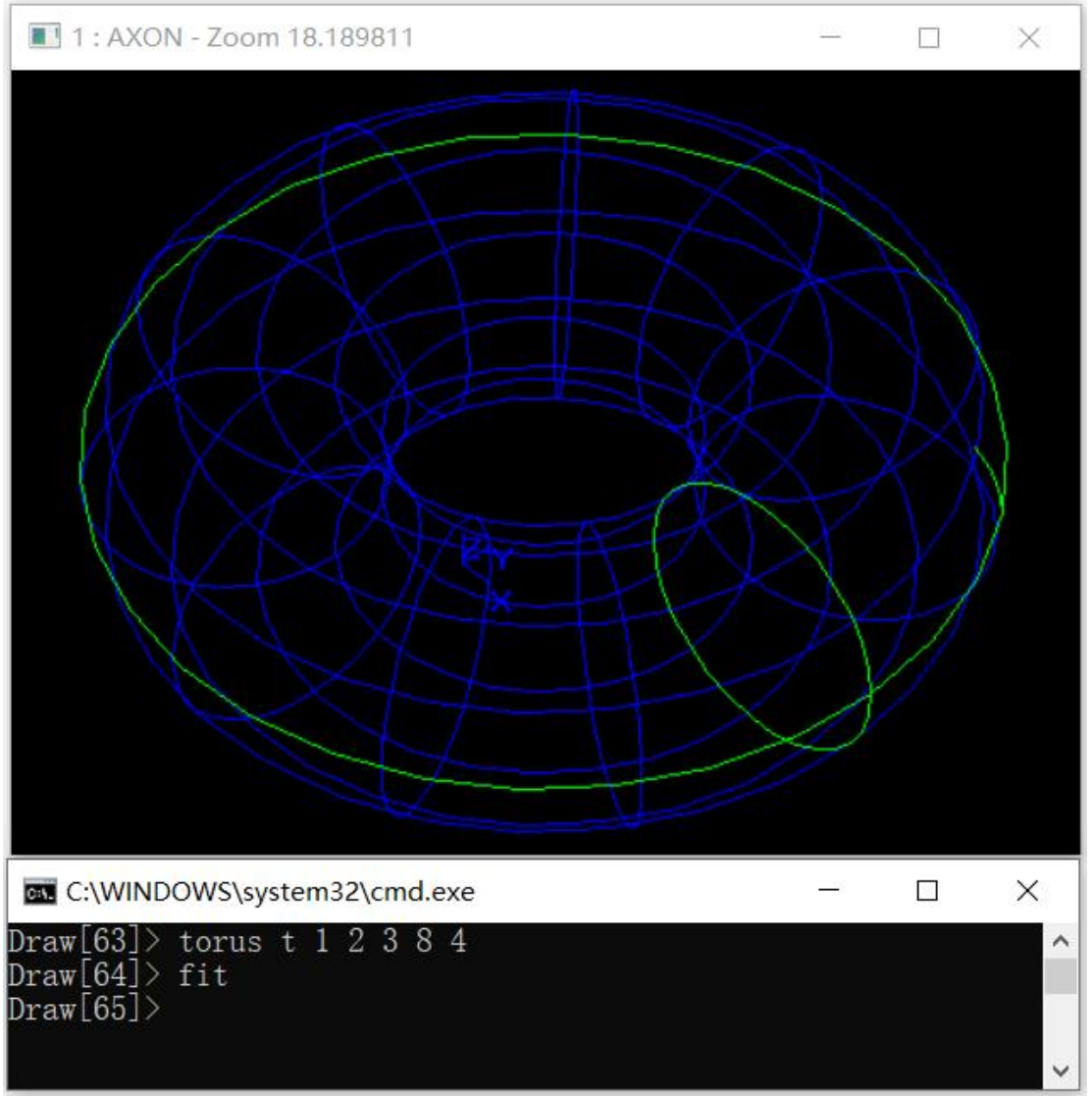
Geometry Surface – Torus

圆环面的数据包含三维点P，三维正交坐标系Dz，Dx，Dy和非负实数r1，r2。圆环面的轴通过点P，方向为Dz，r1是从圆环面的圆的中心到点P的距离，圆环面的圆的半径为r2。圆环面的参数方程如下所示：

$$S(u,v)=P+(r_1+r_2\cdot\cos(v))\cdot(\cos(u)\cdot D_X+\sin(u)\cdot D_Y)+r_2\cdot\sin(v)\cdot D_Z,\ (u,v)\in[0,2\cdot\pi)\times[0,2\cdot\pi).$$

轴通过点P＝（1，2，3），轴的方向为Dz＝（0，0，1）。其它数据为Dx＝（1，0，－0），Dy＝（0，1，0），r1＝8，r2＝4，其参数方程如下所示：

$$S(u,v)=(1,2,3)+(8+4\cdot\cos(v))\cdot(\cos(u)\cdot(1,0,-0)+\sin(u)\cdot(-0,1,0))+4\cdot\sin(v)\cdot(0,0,1).$$



Geometry Surface-Linear Extrusion

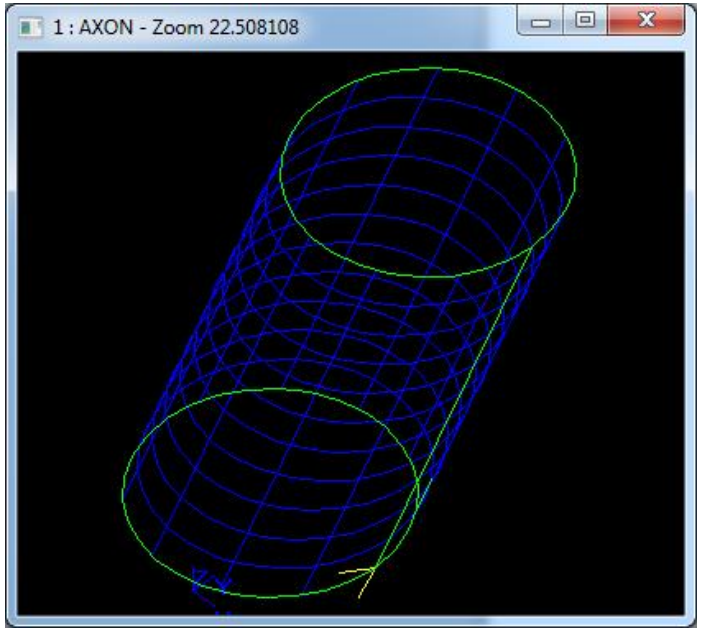
- 线性拉伸面的数据包含三维方向Dv和三维曲线<3D curve record>。其参数方程为：

$$S(u,v)=C(u)+v\cdot D_v,\ (u,v)\in \mathrm{domain}(C)\times(-\infty,\infty).$$

- 数据表示的线性拉伸面的拉伸方向Dv = （0，0.6，0.8），拉伸曲线为圆。拉伸面的参数方程为：

$$S(u,v)=(1,2,3)+4\cdot(\cos(u)\cdot(1,0,-0)+\sin(u)\cdot(-0,1,0))+v\cdot(0,0.6,0.8),\ (u,v)\in[0,2\cdot\pi)\times(-\infty,\infty).$$

- 在Draw Test Harness中创建并显示线性拉伸面为：



Geometry Surface-Revolution

- 旋转面的数据包含三维点P，三维方向D和三维曲线。旋转曲面的轴通过点P且方向为D，旋转曲线为C与旋转轴共面。旋转曲面的参数方程为：

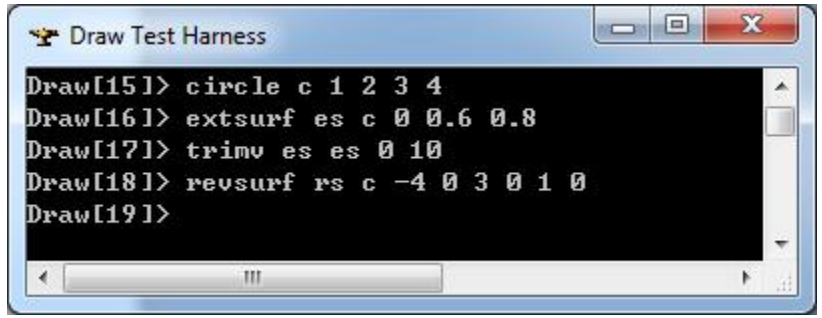
$$S(u,v)=P+V_D(v)+\cos(u)\cdot(V(v)-V_D(v))+\sin(u)\cdot[D,V(v)],\ (u,v)\in[0,2\cdot\pi)\times\mathrm{domain}(C)$$

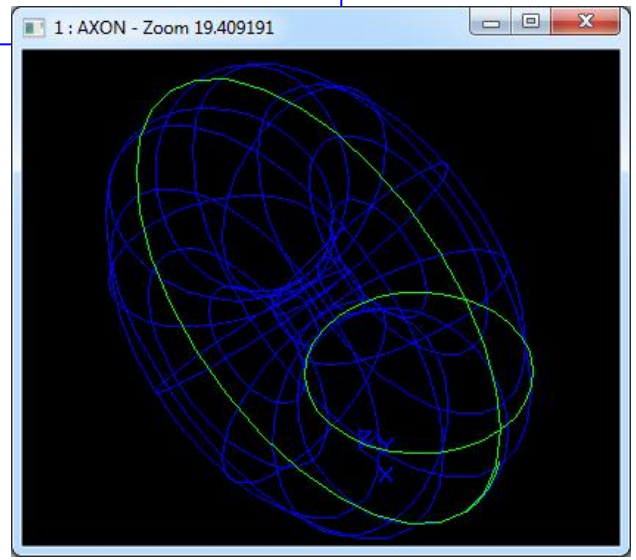
where $V(v)=C(v)-P,\ V_D(v)=(D,V(v))\cdot D.$

- 数据表示的旋转曲面的旋转轴通过点P＝（－4，0，3），方向D＝（0，1，0），旋转曲线是一个圆。其参数方程为：

$$S(u,v)=(-4,0,3)+V_D(v)+\cos(u)\cdot(V(v)-V_D(v))+\sin(u)\cdot[(0,1,0),V(v)],\quad (u,v)\in[0,2\cdot\pi)\times[0,2\cdot\pi)\quad \text{where}$$
$$V(v)=(5,2,0)+4\cdot(\cos(v)\cdot(1,0,-0)+\sin(v)\cdot(-0,1,0)),\ V_D(v)=((0,1,0),V(v))\cdot(0,1,0).$$

- 在Draw Test Harness中创建并显示旋转面：





Geometry Surface-Bezier

- Bezier曲面的数据包含u有理标志位ru，v有理标志位rv，曲面次数mu, mv，和weight poles。u,v的次数都不能大于25。
- 当ru+rv＝0时，weight poles是（mu+1）（mv+1）个三维点Bi,j（（i，j）∈{0，…，mu}x{0,…,mv}），hi,j＝1（（i，j）∈{0，…，mu}x{0,…,mv}）；
- 当ru+rv≠0时，weight poles是（mu+1）（mv+1）个带权控制点对Bi,j，hi,j。Bi,j是三维点，hi,j是权因子，正实数。
- Bezier曲面的参数方程为：

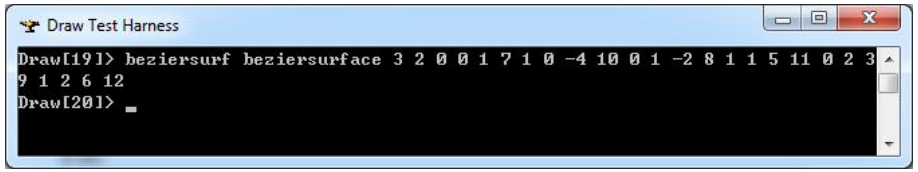
$$S(u,v)=\frac{\sum_{i=0}^{m_u}\sum_{j=0}^{m_v}B_{i,j}\cdot h_{i,j}\cdot C_{m_u}^{i}\cdot u^{i}\cdot(1-u)^{m_u-i}\cdot C_{m_v}^{j}\cdot v^{j}\cdot(1-v)^{m_v-j}}{\sum_{i=0}^{m_u}\sum_{j=0}^{m_v}h_{i,j}\cdot C_{m_u}^{i}\cdot u^{i}\cdot(1-u)^{m_u-i}\cdot C_{m_v}^{j}\cdot v^{j}\cdot(1-v)^{m_v-j}},\ (u,v)\in[0,1]\times[0,1]$$

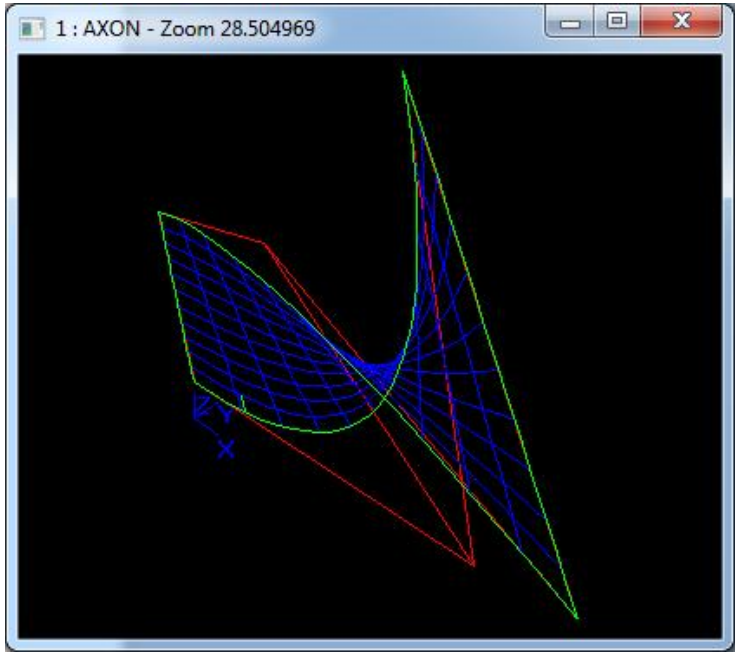
- 数据表示的Bezier曲面为：u有理标志位ru＝1，v有理标志位rv＝1，次数mu＝2，mv＝1，weight poles为：B0,0＝（0，0，1），h0,0＝7，B0,1＝（1，0，－4），h0,1＝10，B1,0＝（0，1，－2），h1,0＝8，B1,1＝（1，1，5），h1,1＝11，B2,0＝（0，2，3），h2,0＝9，B2,1＝（1，2，6），h2,1＝12。曲面的参数方程为：

Geometry Surface-Bezier

$$
\begin{aligned}
S(u,v) = \quad & \Big[(0,0,1)\cdot 7\cdot(1-u)^2\cdot(1-v)+(1,0,-4)\cdot 10\cdot(1-u)^2\cdot v+ \\
& (0,1,-2)\cdot 8\cdot 2\cdot u\cdot(1-u)\cdot(1-v)+(1,1,5)\cdot 11\cdot 2\cdot u\cdot(1-u)\cdot v+ \\
& (0,2,3)\cdot 9\cdot u^2\cdot(1-v)+(1,2,6)\cdot 12\cdot u^2\cdot v\Big]\div \\
& \Big[7\cdot(1-u)^2\cdot(1-v)+10\cdot(1-u)^2\cdot v+ \\
& 8\cdot 2\cdot u\cdot(1-u)\cdot(1-v)+11\cdot 2\cdot u\cdot(1-u)\cdot v+ \\
& 9\cdot u^2\cdot(1-v)+12\cdot u^2\cdot v\Big]
\end{aligned}
$$

- 在Draw Test Harness中创建并显示Bezier曲面：





Geometry Surface-BSpline

- B样条曲面数据包含u有理标志位ru，v有理标志位rv，u次数mu<=25；v次数mv<=25，u控制点数nu>=2，v控制点数nv>=2，u重节点数ku，v重节点数kn，weight poles，u重节点，v重节点。
- 当ru+rv＝0时，weight poles是（mu+1）（mv+1）个三维点Bi,j（（i，j）∈{0，…，mu}x{0,…,mv}），hi,j＝1（（i，j）∈{0，…，mu}x{0,…,mv}）；
- 当ru+rv≠0时，weight poles是（mu+1）（mv+1）个带权控制点对Bi,j，hi,j。Bi,j是三维点，hi,j是权因子，正实数。
- u重节点及其重数有ku对：u1,q1,…,uku,qku。这里ui是重数为qi>=1的节点：

$$u_i < u_{i+1}\ (1 \le i \le k_u - 1),$$

$$q_1 \le m_u + 1,\ q_{k_u} \le m_u + 1,\ q_i \le m_u\ (2 \le i \le k_u - 1),\ \sum_{i=1}^{k_u} q_i = m_u + n_u + 1.$$

- v重节点及其重数有kv对：u1,q1,…,ukv,qkv。这里vi是重数为qi>=1的节点：

$$v_j < v_{j+1}\ (1 \le j \le k_v - 1),$$

$$t_1 \le m_v + 1,\ t_{k_v} \le m_v + 1,\ t_j \le m_v\ (2 \le j \le k_v - 1),\ \sum_{j=1}^{k_v} t_j = m_v + n_v + 1.$$

Geometry Surface-BSpline

- B-Spline曲面的参数方程为：

$$S(u,v)=\frac{\sum_{i=1}^{n_u}\sum_{j=1}^{n_v}B_{i,j}\cdot h_{i,j}\cdot N_{i,m_u+1}(u)\cdot M_{j,m_v+1}(v)}{\sum_{i=1}^{n_u}\sum_{j=1}^{n_v}h_{i,j}\cdot N_{i,m_u+1}(u)\cdot M_{j,m_v+1}(v)},\ (u,v)\in\lfloor u_1,u_{k_u}\rfloor\times\lfloor v_1,v_{k_v}\rfloor$$

- 基函数Ni,j和Mi,j有如下的递归定义：

$$N_{i,1}(u)=\begin{cases}1 \Leftarrow \bar{u}_i\le u<\bar{u}_{i+1}\\ 0 \Leftarrow u<\bar{u}_i\vee\bar{u}_{i+1}\le u\end{cases},\ N_{i,j}(u)=\frac{(u-\bar{u}_i)\cdot N_{i,j-1}(u)}{\bar{u}_{i+j-1}-\bar{u}_i}+\frac{(\bar{u}_{i+j}-u)\cdot N_{i+1,j-1}(u)}{\bar{u}_{i+j}-\bar{u}_{i+1}}\ (2\le j\le m_u+1);$$

$$M_{i,1}(u)=\begin{cases}1 \Leftarrow \bar{v}_i\le v<\bar{v}_{i+1}\\ 0 \Leftarrow v<\bar{v}_i\vee\bar{v}_{i+1}\le v\end{cases},\ M_{i,j}(v)=\frac{(v-\bar{v}_i)\cdot M_{i,j-1}(v)}{\bar{v}_{i+j-1}-v_i}+\frac{(\bar{v}_{i+j}-v)\cdot M_{i+1,j-1}(v)}{\bar{v}_{i+j}-\bar{v}_{i+1}}\ (2\le j\le m_v+1);$$

$$\bar{u}_i=u_j\ \left(1\le j\le k_u,\sum_{l=1}^{j-1}q_l+1\le i\le\sum_{l=1}^{j}q_l\right),$$

$$\bar{v}_i=v_j\ \left(1\le j\le k_v,\sum_{l=1}^{j-1}t_l+1\le i\le\sum_{l=1}^{j}t_l\right).$$

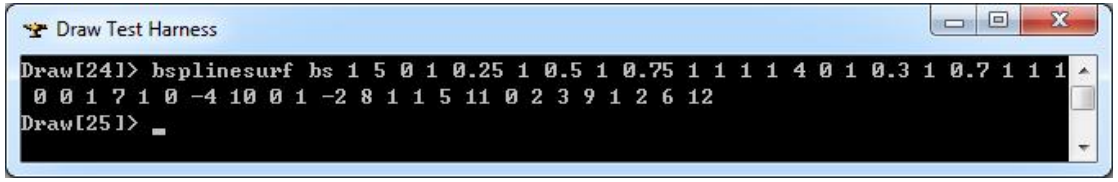
Geometry Surface-BSpline

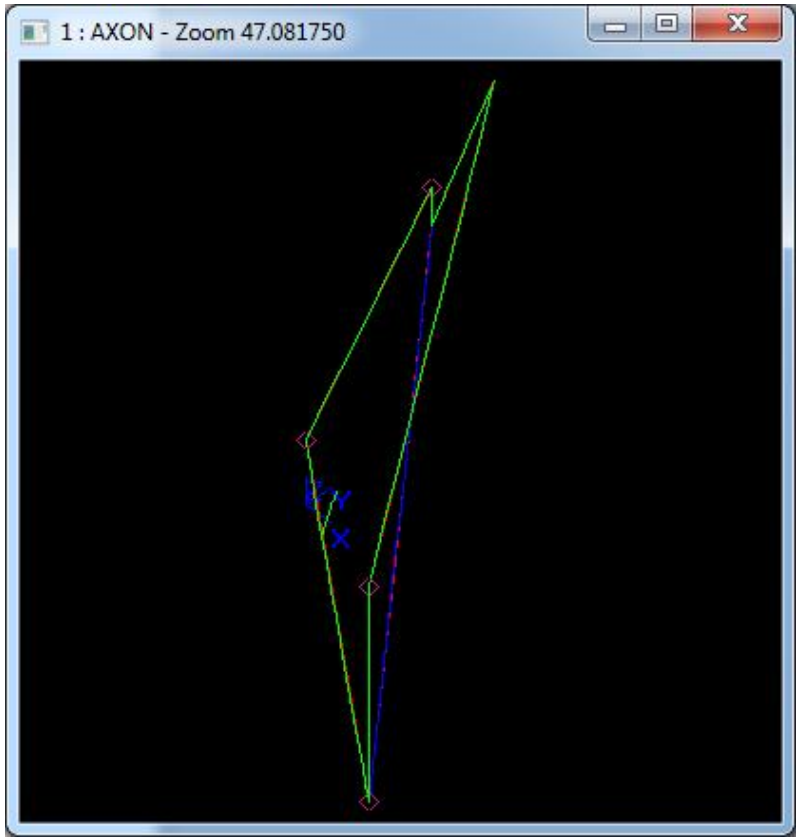
- B-Spline曲面示例数据为：u有理标志位ru＝1，v有理标志位rv＝1，u次数mu＝1，v次数mv＝1，u控制点数nu＝3，v控制点数nv＝2，u有重复度的节点数ku＝5，v有重复度节点数kv＝4，带权控制点B1,1＝（0，0，1），h1,1＝7，B1,2＝（1，0，－4），h1,2＝10，B2,1＝（0，1，－2），h2,1＝8，B2,2＝（1，1，5），h2,2＝11，B3,1＝（0，2，3），h3,1＝9，B3,2＝（1，2，6），h3,2＝12，u有重复度节点u1＝0，q1＝1，u2＝0.25，q2＝1，u3＝0.5，q3＝1，u4＝0.75，q4＝1，u5＝1，q5＝1，v有重度度节点v1＝0，r1＝1，v2＝0.3，r2＝1，v3＝0.7，r3＝1，v4＝1，r4＝1。B-Spline曲面的参数方程如下所示：

$$
\begin{aligned}
S(u,v)= \quad & \big[(0,0,1)\cdot 7\cdot N_{1,2}(u)\cdot M_{1,2}(v)+(1,0,-4)\cdot 10\cdot N_{1,2}(u)\cdot M_{2,2}(v)+ \\
& (0,1,-2)\cdot 8\cdot N_{2,2}(u)\cdot M_{1,2}(v)+(1,1,5)\cdot 11\cdot N_{2,2}(u)\cdot M_{2,2}(v)+ \\
& (0,2,3)\cdot 9\cdot N_{3,2}(u)\cdot M_{1,2}(v)+(1,2,6)\cdot 12\cdot N_{3,2}(u)\cdot M_{2,2}(v)\big]\div \\
& \big[7\cdot N_{1,2}(u)\cdot M_{1,2}(v)+10\cdot N_{1,2}(u)\cdot M_{2,2}(v)+ \\
& 8\cdot N_{2,2}(u)\cdot M_{1,2}(v)+11\cdot N_{2,2}(u)\cdot M_{2,2}(v)+ \\
& 9\cdot N_{3,2}(u)\cdot M_{1,2}(v)+12\cdot N_{3,2}(u)\cdot M_{2,2}(v)\big]
\end{aligned}
$$

Geometry Surface-BSpline

- 在Draw Test Harness中创建并显示B样条曲面





Geometry Surface-Rectangular Trimmed

- 矩形裁剪曲面的数据包含实数umin，umax，vmin，vmax和一个曲面。矩形裁剪曲面是将曲面限制在矩形区域[umin，umax]x[vmin，vmax]内得到的曲面。曲面的参数方程为：

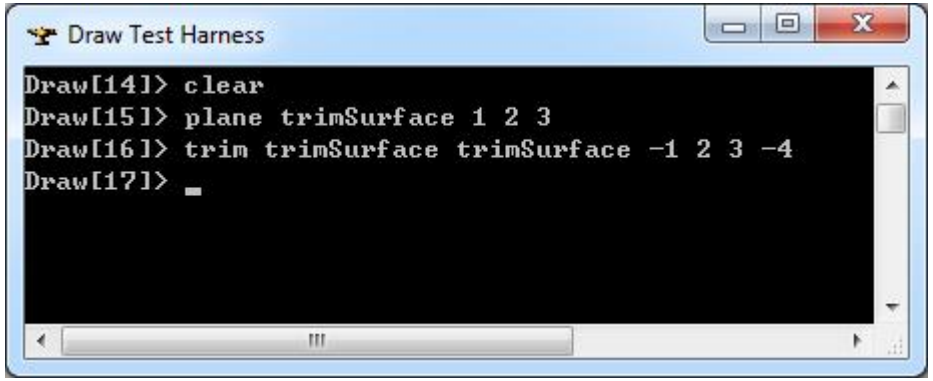
$$S(u,v)=B(u,v),\ (u,v)\in[u_{\min},u_{\max}]\times[v_{\min},v_{\max}].$$

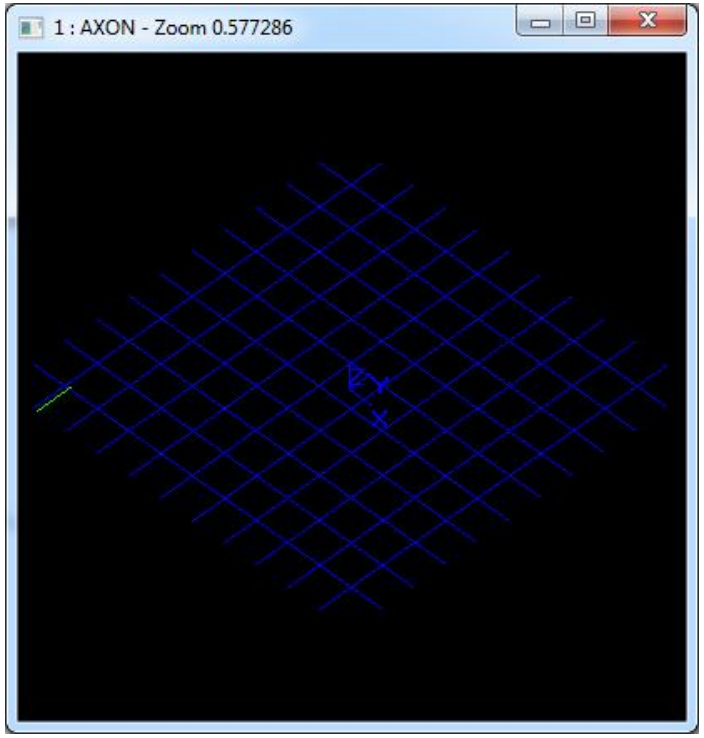
- 矩形裁剪曲面的示例矩形裁剪区域为[-1，2]x[－3，4]，被裁剪曲面B（u，v）＝（1，2，3）+u（1，0，0）+v（0，1，0）。其参数方程为：

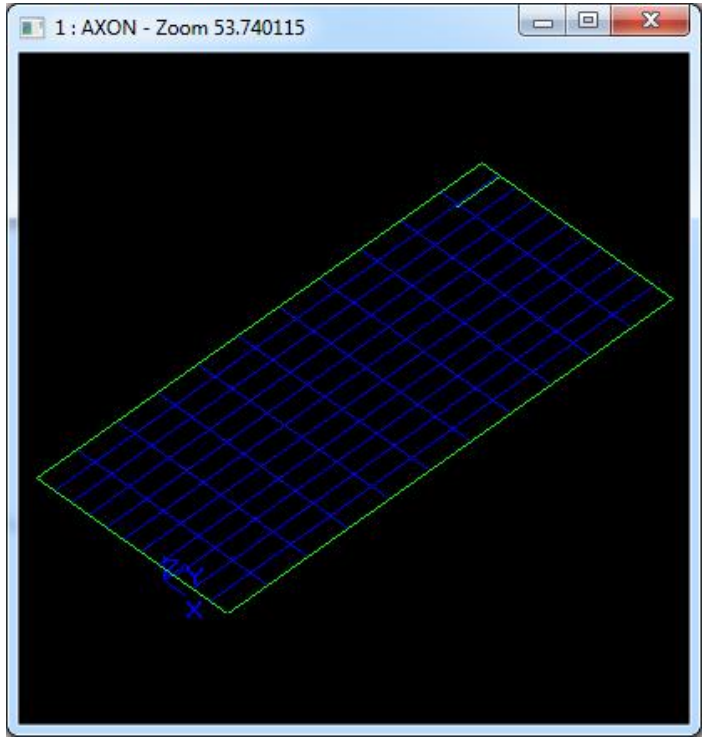
$$B(u,v)=(1,2,3)+u\cdot(1,0,0)+v\cdot(0,1,0),\ (u,v)\in[-1,2]\times[-3,4].$$

Geometry Surface-Rectangular Trimmed

- 在Draw Test Harness中创建并显示矩形裁剪曲面







Geometry Surface-Offset

- 偏移曲面的数据包含偏移距离d和曲面。偏移曲面的就将基准曲面B没曲面的法向N上偏移距离d得到的曲面。偏移曲面的参数方程为：

$$S(u,v)=B(u,v)+d\cdot N(u,v),\ (u,v)\in \text{domain}(B).$$

$$N(u,v)=[S'_u(u,v),S'_v(u,v)]$$

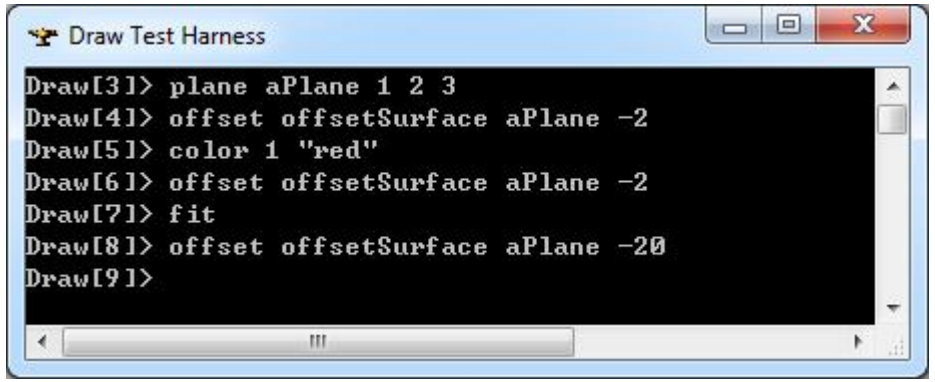
if $[S'_u(u,v),S'_v(u,v)]\neq \vec{0}$.

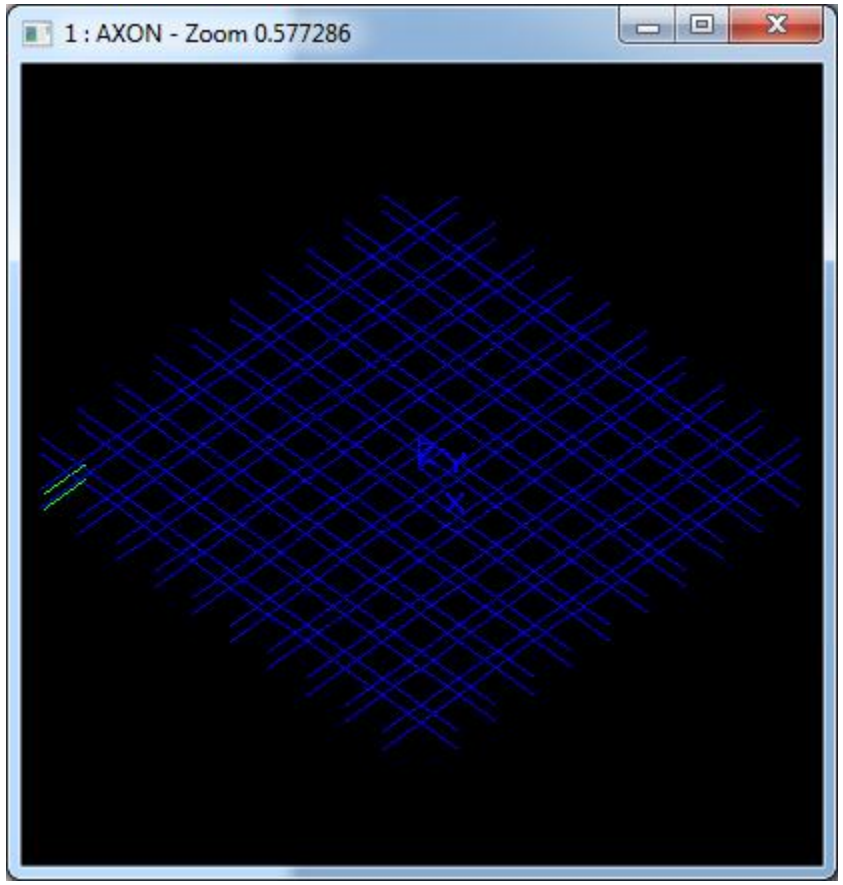
- 偏移曲面的示例：偏移距离d＝－2，基准曲面B（u，v）＝（1，2，3）+u（1，0，0）+v（0，1，0）。其参数方程为：

$$S(u,v)=(1,2,3)+u\cdot(1,0,0)+v\cdot(0,1,0)-2\cdot(0,0,1).$$

Geometry Surface-Offset

- 在Draw Test Harness中创建并显示偏移曲面





当偏移－2时，效果不明显示，

所以偏移了－20，这样看上去比较明显。

Interpolation vs approximation

Interpolation

Interpolation is finding a polynomial which passes through all points.

The typical problems with them are:

- High order polynomial in result (depends on interpolation algorithm).
- Complex result.
- Oscillations are possible.

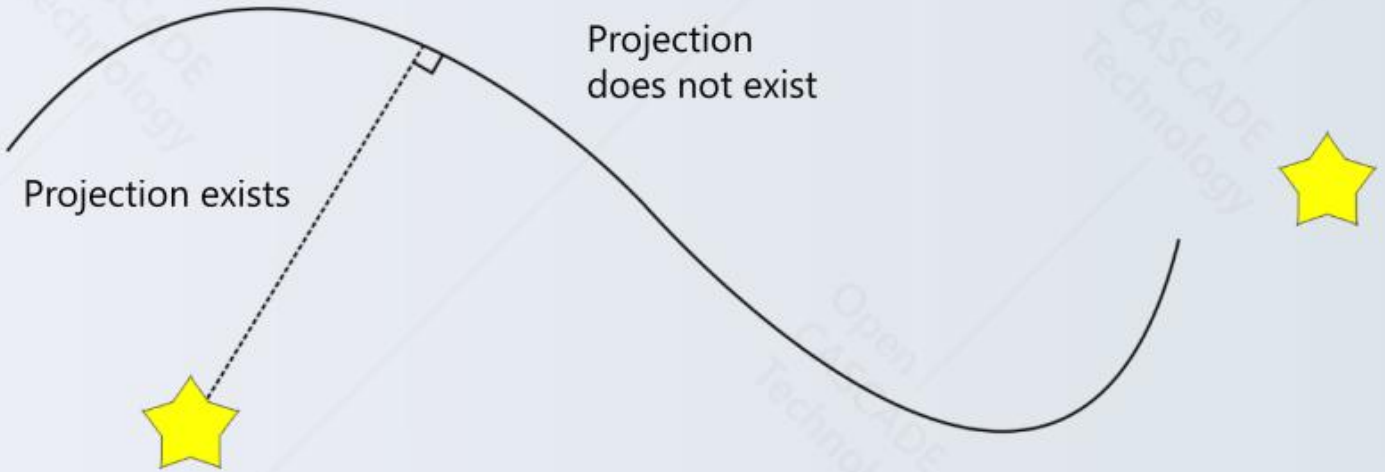
Approximation

Approximation is process of polynomial construction which is close to a given set of points but not exactly passes through them. The approximation algorithm has the following drawbacks:

- It is much more computationally intensive comparing to interpolation.
- Requires more efforts to tune parameters comparing to interpolation.

Point projection on curve

OCCT contains various projection algorithms such as projection of point on curve or projection curve on surface. Unlike interpolation algorithm, sometimes projection does not exist:

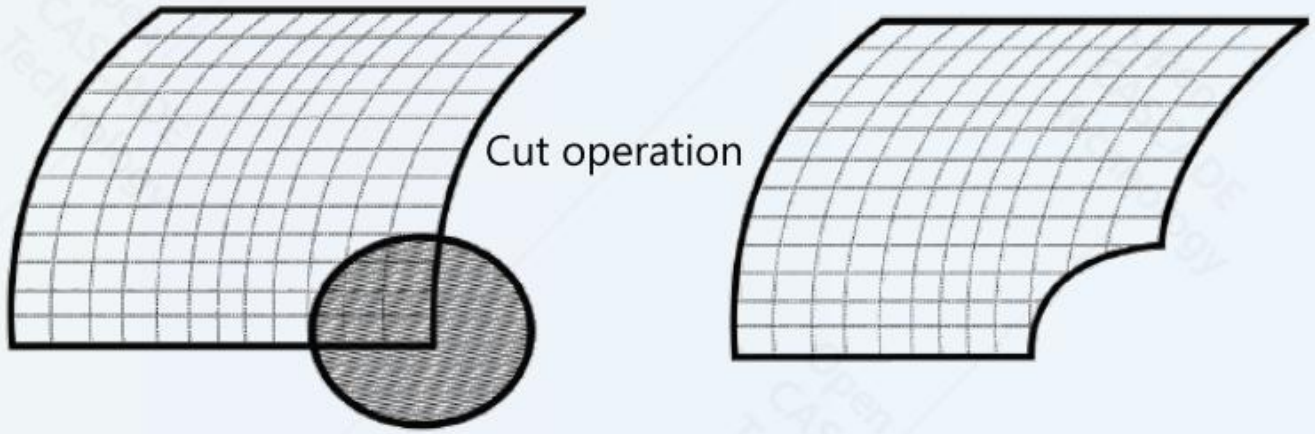


造型数据 ModelingData

Topology

Geometry limitations

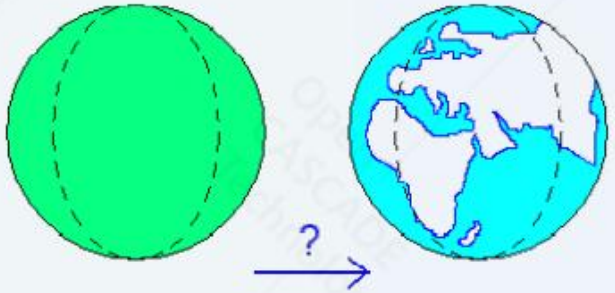
OCCT surfaces support rectangular trimming. A non-rectangular domain may arise after the Boolean operation.



How to store the result of the cut operation?

Purpose of topology

In general, Topology is a means to describe the limitation of an object.



Open CASCADE Topology is used to describe:

- Boundaries of objects.
- Connectivity between objects (via common boundaries).

Open CASCADE topological entities are called shapes.

Definition of topology

Topological shapes are defined following these two concepts:

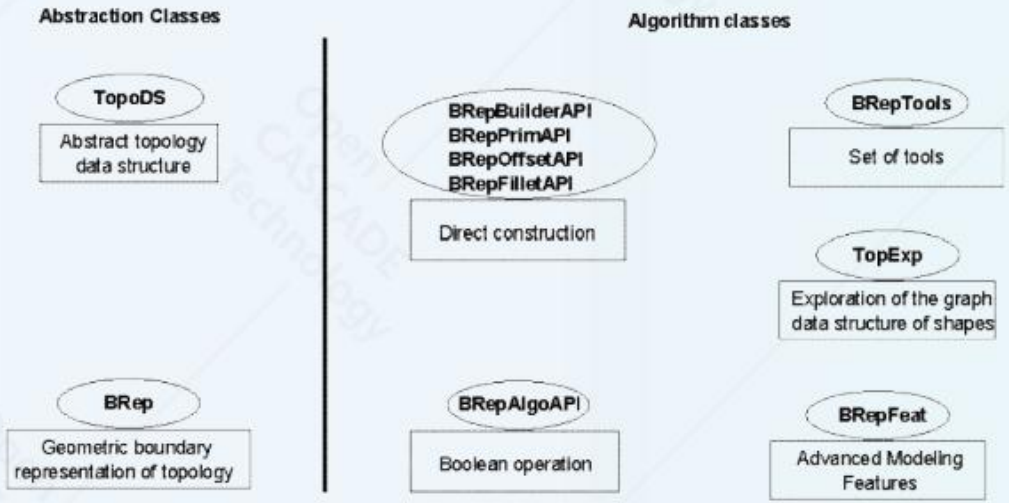
- Abstract Topology (TopoDS): defines the data structure by describing the relation between bounded and bounding objects.

Example: an edge is described by its boundaries which are vertices.

- Boundary representation (B-Rep): completes the definition of an object by associating topological and geometric information.

Example: an edge lies on a curve and is bounded by points.

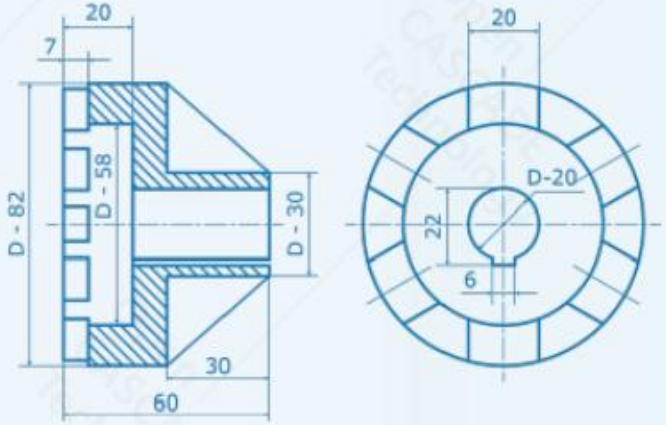
Abstract, boundary representation and algorithm classes are grouped in different packages.

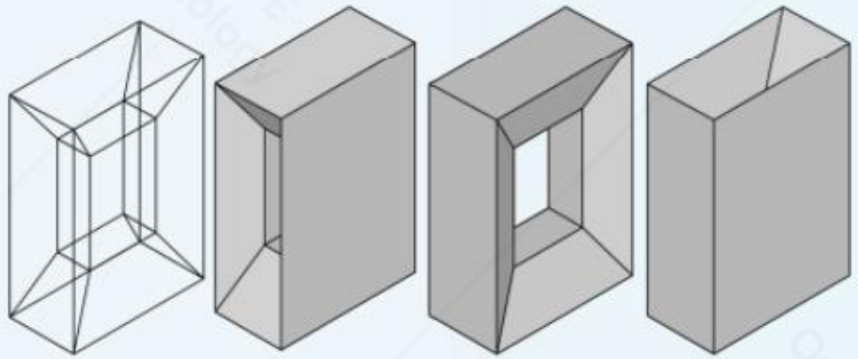


Why boundary representation?

There are several alternatives to B-Rep:

- Constructive solid geometry (CSG). CSG does not allow to model arbitrary figures.
- Drawing. Drawings are not suitable for downstream engineering operations.
- Wireframe. Several solids may correspond to a single wireframe model.
- Mesh. Meshes do not support curved geometry.





B-Rep has some drawbacks. It is glassy, verbose, and complex. B-Rep model uses vertices, edges, and faces as geometry carriers but also has special topological types. Model complexity causes fragility.

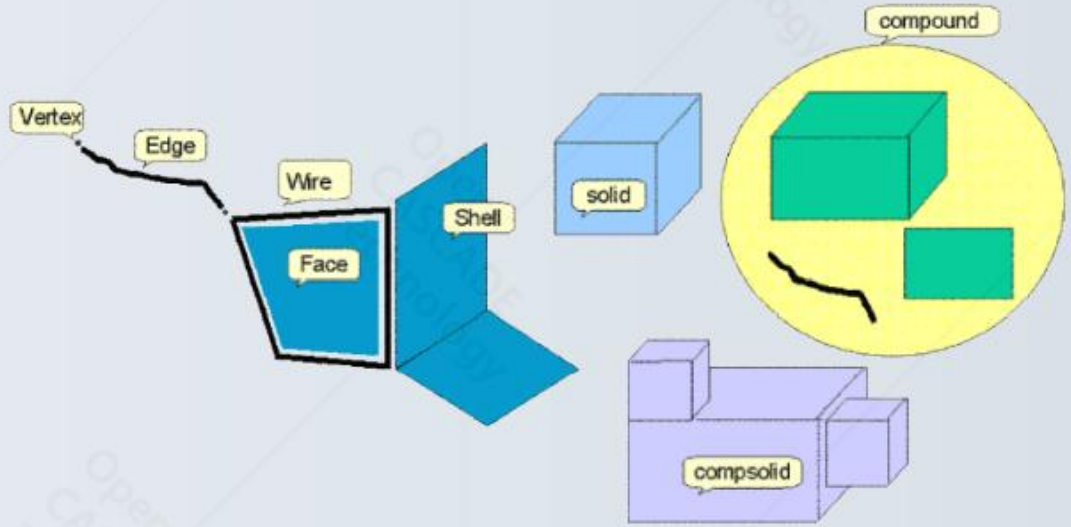
ModelingData BRep

- 边界表示（Boundary Representation）也称为BRep表示，它是几何造型中最成熟、无二义的表示法。实体的边界通常是由面的并集来表示，而每个面又由它所在的曲面的定义加上其边界来表示，面的边界是边的并集，而边又是由点来表示的。
- 边界表示的一个重要特征是描述形体的信息包括几何信息（Geometry）和拓朴信息（Topology）两个方面。拓朴信息描述形体上的顶点、边、面的连接关系，它形成物体边界表示的“骨架”。形体的几何信息犹如附着在“骨架”上的肌肉。例如，形体的某个面位于某一个曲面上，定义这一曲面方程的数据就是几何信息。此外，边的形状、顶点在三维空间中的位置（点的坐标）等都是几何信息，一般来说，几何信息描述形体的大小、尺寸、位置和形状等。
- 在边界表示法中，边界表示就按照体－面－环－边－点的层次，详细记录构成形体的所有几何元素的几何信息及其相互连接的拓朴关系。这样，在进行各种运算和操作中，就可以直接取得这些信息。

Topological shapes

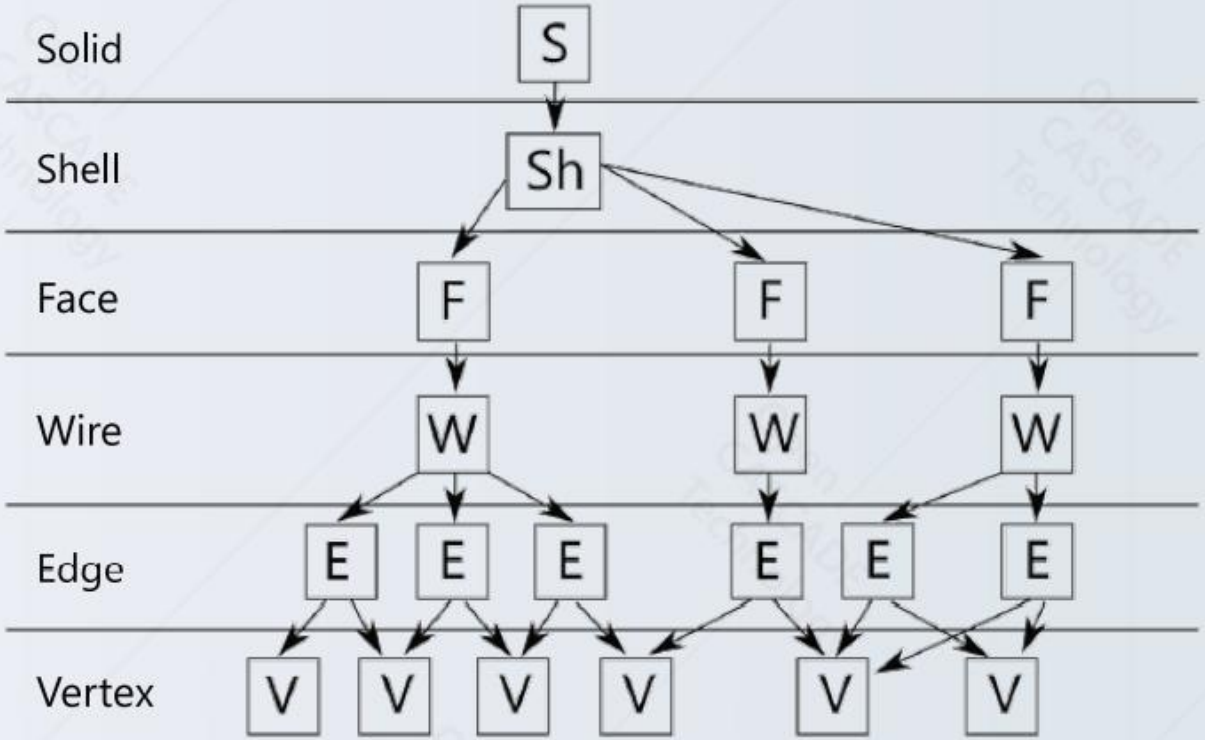
Open CASCADE Technology defines the following types of topological shapes:

- Vertex: a point.
- Edge: a part of a curve limited by vertices.
- Wire: a set of edges (connected by their vertices).
- Face: a part of a surface limited by wires.
- Shell: a set of faces (connected by their edges).
- Solid: a part of space limited by shells.
- Compsolid: a set of solids connected by their faces.
- Compound: a group of any topological shapes.



Graph structure

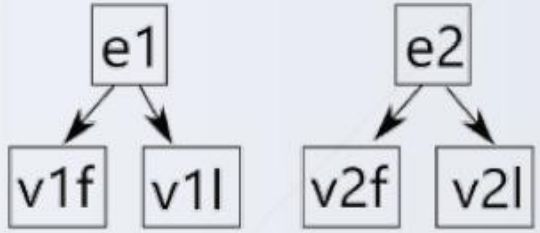
The following graph shows an example of relations between sub-shapes of a complex shape (a solid in this case):

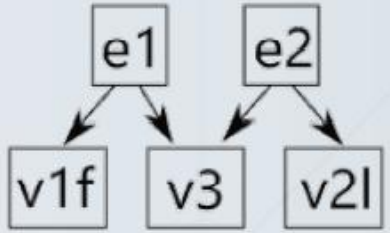


Connectivity of shapes

Two shapes are connected if they share some bounding sub-shape(s).

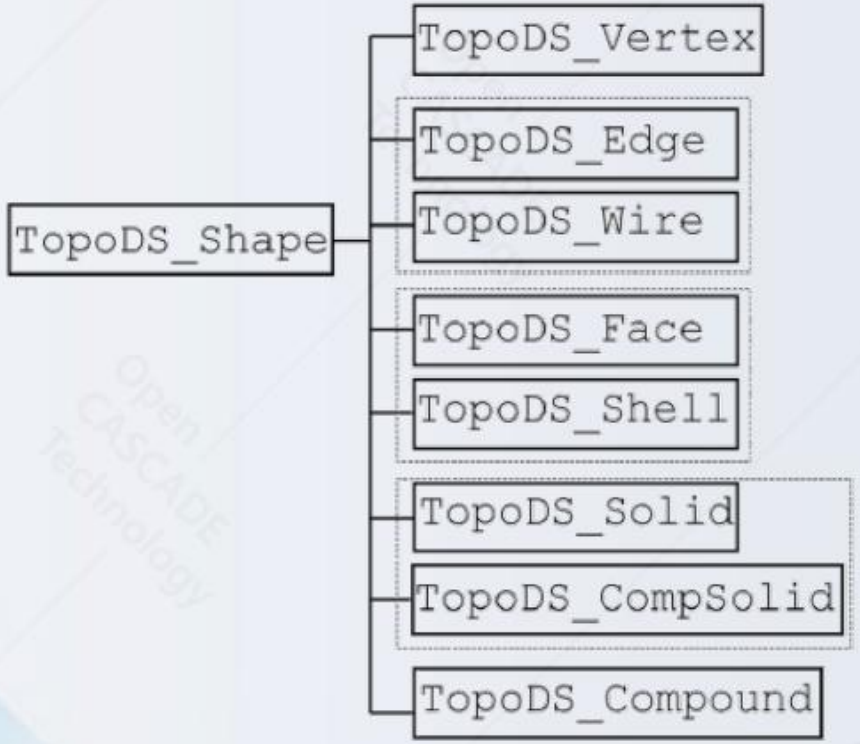
Example: Let's consider two edges – `e1` and `e2`. Each of them is limited by its boundaries, which are vertices (`v1f` and `v1l` for `e1` and `v2f` and `v2l` for `e2`). When these two edges share a common vertex `v3`, they are connected.





Hierarchy of shapes

TopoDS_Shape is the root class for all classes of topological shapes.



TopoDS_Vertex keeps information about point (zero-dimensional object).

TopoDS_Edge keeps information about curves (one-dimensional object). TopoDS_Wire is a collection of edges.

TopoDS_Face keeps information about surfaces (two-dimensional object). TopoDS_Shell is a collection of faces.

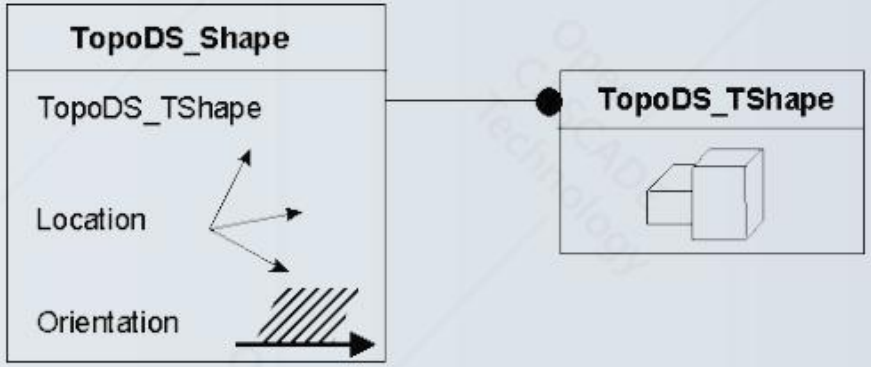
TopoDS_Solid, TopoDS_Compsolid keeps information about solids.

TopoDS_Compound represents a shape which is a collection of shapes.

Structure of shape

The TopoDS_Shape class defines a shape by:

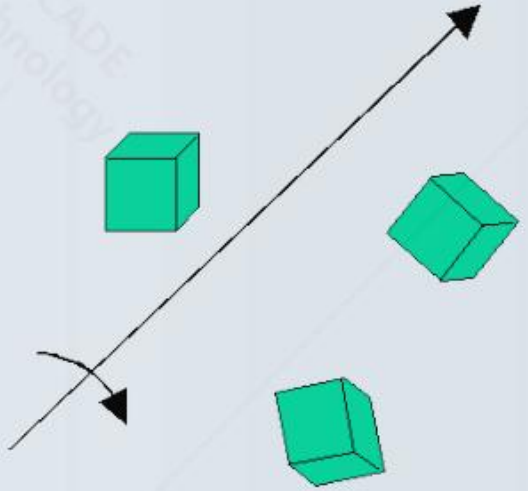
- A TopoDS_TShape handle (TopoDS package).
- A local coordinate system (TopLoc package).
- An orientation (TopAbs package).



TopoDS_TShape: a handle class which describes the object in its default coordinate system. This class is never used directly, TopoDS_Shape is used.

TopLoc_Location: defines a local coordinate system which places a shape at a different position from that of its definition.

Example: all these boxes share the same TShape but have different locations.



Structure of shape

`TopAbs_Orientation`: describes how a shape delimits a geometry in terms of material (or inner and outer regions).

The orientation and location parameters of the shape are also assumed to affect its sub-shapes when considered in the context of that shape. When a shape is explored to sub-shapes, orientation, and location of the sub-shape is combined with that of the main shape. This ensures consistent interpretation of parameters of the sub-shape in the context of each shape that refers to it. For example, the edge shared by two connected faces will have opposite orientations when explored in the context of those faces.

Location modification

Data sharing concept, available in OCCT, allows re-use topological information and instance model several times by means of the location. Location within a shape is a set of consecutive transformations which are manipulated as a single transformation.

Note: empty location and existing identical locations are considered as different. As a result, `IsSame()` check will return `false`.

Location methods are as follows:

- `Location`
- `Move`
- `Moved`

```
// Construct location.
gp_Ax1 axis(gp_Pnt(aX, aY, aZ),
            gp_Vec(aDX, aDY, aDZ));

gp_Trsf T;
T.SetRotation(axis, aR);

// Update existing location.
aShape.Move(T);
```


Orientation: vertex within an edge

OCCT orientation concept aims at the completion of the boundary representation by information about inner and outer regions. This information is a set of rules affecting shape correctness.

The orientation itself has no meaning for a vertex. Vertex orientation makes sense only when vertex bounds some edge. Edge is constructed using a pair of vertices (at least one vertex using twice in case of the periodic curve); the first vertex has `TopAbs_FORWARD` orientation, and the second one has `TopAbs_REVERSED` orientation by a convention.

Note: `BRepBuilderAPI_MakeVertex` constructs a vertex with `TopAbs_FORWARD` orientation.

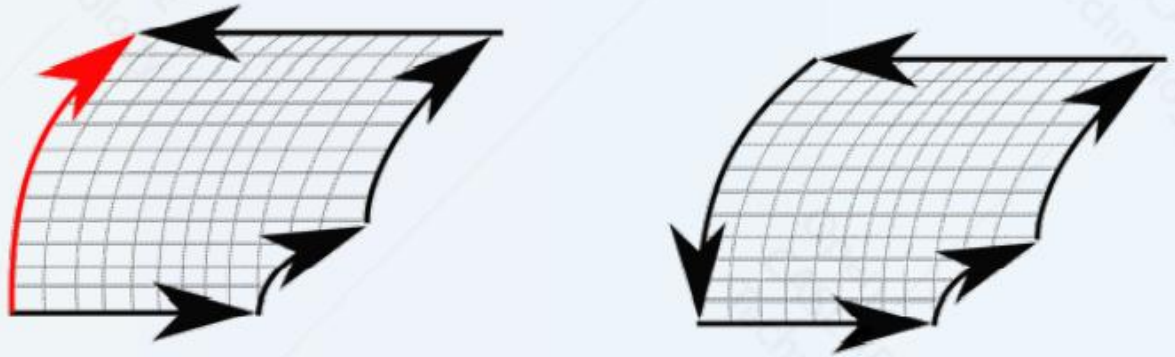
```
gp_Pnt aP1(10.0, 0.0, 0.0), aP2(20.0, 0.0, 0.0);
TopoDS_Edge anEdge = BRepBuilderAPI_MakeEdge(aP1, aP2);
TopExp_Explorer anExpEV(anEdge, TopAbs_VERTEX);
for(; anExpEV.More(); anExpEV.Next())
{
  const TopoDS_Vertex& aV =
TopoDS::Vertex(anExpEV.Current());
  const gp_Pnt& aPnt = BRep_Tool::Pnt(aV);
  if (aV.Orientation() == TopAbs_FORWARD)
  {
    std::cout << "Forward vertex is: "
              << aPnt.X() << " "
              << aPnt.Y() << " "
              << aPnt.Z() << std::endl;
  }
  else if (aV.Orientation() == TopAbs_REVERSED)
  {
    // There is inner orientation. That is why
    // else-if expression is needed.
    std::cout << "Reversed vertex is: "
              << aPnt.X() << " "
              << aPnt.Y() << " "
              << aPnt.Z() << std::endl;
  }
}
```


Orientation: edge within a wire

The face is a part surface bounded by edges. Edges are organized into wires to be able to track each loop (outer or inner) individually.

There are no limitations when a wire is free; edges can be added to wire without constraints.

The right-hand rule comes in action when wire belongs to a face; each edge in a wire should be orientated to have material on a left side according to parameter increasing direction on a curve. What to do with the wrongly oriented curve?



It is possible to rebuild a curve, but it is preferable to revert the underlying curve virtually. The second option was chosen in the OCCT.

Orientation: face orientation in a solid

Solid is a part of modeling space bounded by faces organized into a shell. Normals in the solid should point outside the material by a convention. Differential geometry states that normal to surface in point can be evaluated using the following formula ("x" stands for cross product):

$$N(u_0, v_0) = s'_u(u_0, v_0) \times s'_v(u_0, v_0)$$

Normal is calculated up to a sign, so an alternative formula exists where partial derivatives are swapped (the OCCT uses the formula presented above). How to ensure the correct normal orientations in a solid?

The face orientation determines the sign before normal:

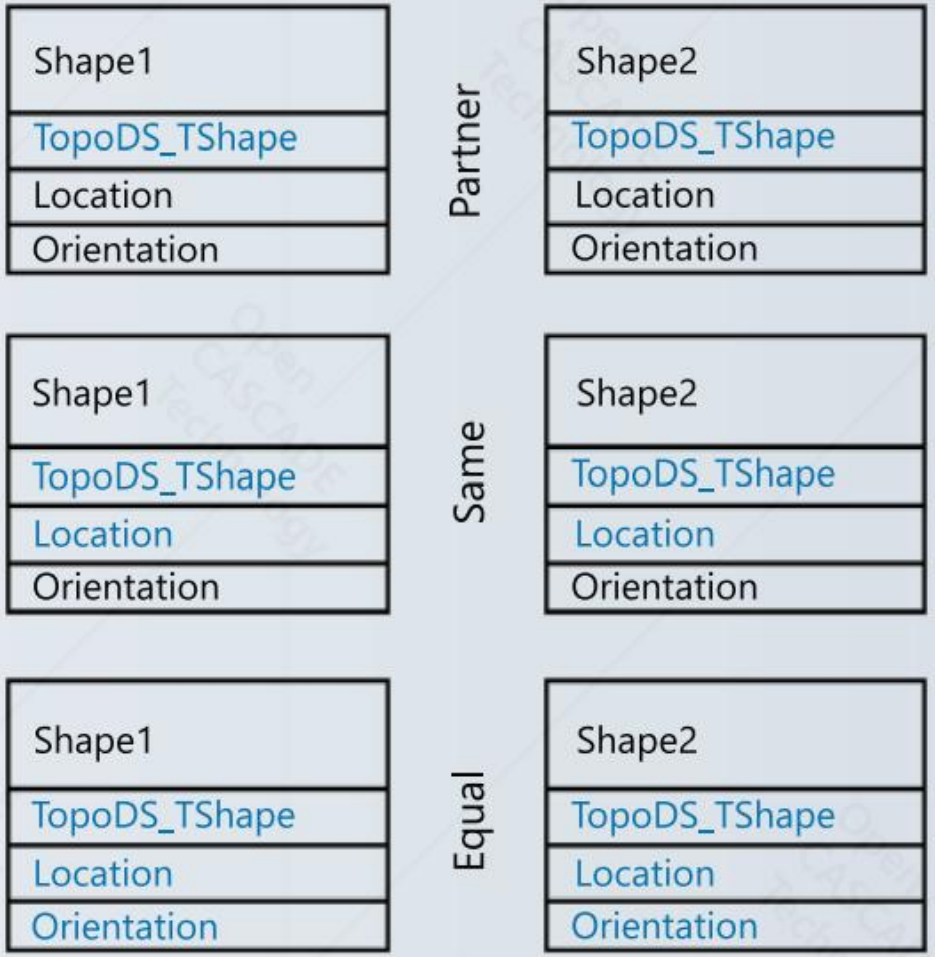
- `TopAbs_FORWARD` – normal is evaluated according to the formula above.
- `TopAbs_REVERSED` – normal is multiplied by -1.0.

Note: solid may represent the whole modeling space except a part of space bounded by solid boundaries. Normals will point inside the bounded part of space in that case.

Shape manipulations

The TopoDS_Shape class and it descendants provide various useful methods such as:

- Access to TShape
  - IsNull() – checks whether TShape is null or not.
  - Nullify() – nullifies TShape smart pointer.
- Access to location
  - Location() – returns existing location.
  - Move() – applies transformation to actual shape.
  - Moved() – returns new shape with applied transformation.
- ShapeType() – returns the type of the TopoDS_Shape
- Shapes comparison:
  - IsPartner() – the same TShape
  - IsSame() – the same TShape and location.
  - IsEqual() – the same TShape, location and orientation.



Shape downcasting

`TopoDS_Shape` objects are manipulated by value. That is why special methods are implemented to supply downcast functionality:

- `TopoDS::Vertex()` Returns a `TopoDS_Vertex`
- `TopoDS::Edge()` Returns a `TopoDS_Edge`
- `TopoDS::Wire()` Returns a `TopoDS_Wire`
- `TopoDS::Face()` Returns a `TopoDS_Face`
- `TopoDS::Shell()` Returns a `TopoDS_Shell`
- `TopoDS::Solid()` Returns a `TopoDS_Solid`
- `TopoDS::CompSolid()` Returns a `TopoDS_CompSolid`
- `TopoDS::Compound()` Returns a `TopoDS_Compound`

Note: exception is raised when an inappropriate conversion is done.

Example: the first block is correct, but the second is rejected by the compiler.

```
// Correct.
if (aShape.ShapeType() == TopAbs_VERTEX)
{
  const TopoDS_Vertex& aV1 =
                    TopoDS::Vertex(aShape);
}
```

```
// Rejected by compiler.
if (aShape.ShapeType() == TopAbs_VERTEX)
{
  TopoDS_Vertex aV2 = aShape;
}
```


Collections of shapes

The `TopTools` package provides:

- Classes to compute hash code for a shape with or without orientation.
- Instantiation of collections for shapes.

```
BRep_Builder BB;
TopTools_MapOfShape anEmap = anItl.Value();
TopTools_ListIteratorOfListOfShape anItl(anEdges);
for (;anItl.More();anItl.Next())
  BB.Remove(aWire, anItl.Value());

for (anItl.Initialize(edges); anItl.More();anItl.Next())
{
  TopoDS_Shape anEdge = anItl.Value();
  if (anEmap.Contains(anEdge))
    anEdge.Reverse();
  BB.Add(aWire, anEdge);
}
```


Exploration tools

Exploring a topological shape means finding its sub-shapes, possibly matching specific criterion.

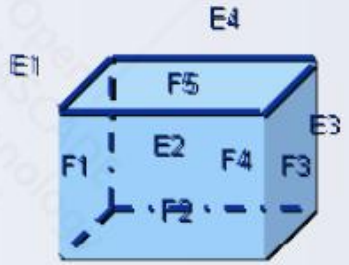
- `TopoDS_Iterator` class explores the first level sub-shapes of the given shape (from the list in its `TShape`).
- `TopExp_Explorer` class explores all sub-shapes in the given shape, with a possibility to select the kind of entities (for example, faces only).

```
TopExp_Explorer anExp(aShape, TopAbs_EDGE);
for (; anExp.More(); anExp.Next())
{
  TopoDS_Edge anEdge = TopoDS::Edge(anExp.Current());
}
```

- `TopExp::MapShapes()` method explores sub-shapes and puts them in a map (thus detecting the same elements).

Exploration tools

- `TopExp::MapShapesAndAncestors()` method returns all the entities that reference another one.



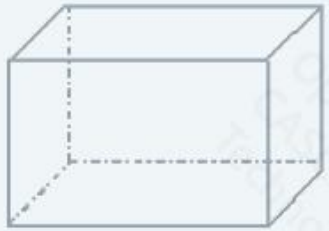
E1		E2		E3		E4	
F1	F5	F2	F5	F3	F5	F4	F5

In Open CASCADE Technology, there are no back pointers from a sub-shape to its ancestor shapes. Instead, `TopExp::MapShapesAndAncestors()` may be used to restore this information. For example, if you want to find all faces that contain a given vertex or an edge, you may use this method.

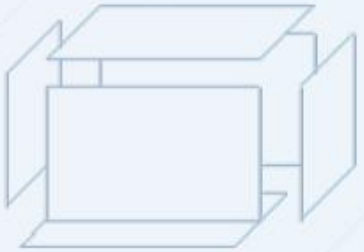
Boundary representation

The Boundary Representation (B-Rep) describes the model objects in three dimensions.

In B-Rep modeling, entities are represented by their boundaries.



Model in 3D



Faces bound model

Edges bound faces

B-Rep immerses Geometry into Topology:

- Geometry: a face lies on a surface, an edge lies on a curve, and a vertex lies on a point.
- Topology: connectivity of shapes.

Thus the description of a model object becomes complete.

B-Rep description is based on:

- `TopoDS` package - to describe the topological structure of objects.
- `Geom` and `Geom2d` packages - to describe the geometry of these objects.

B-Rep entities

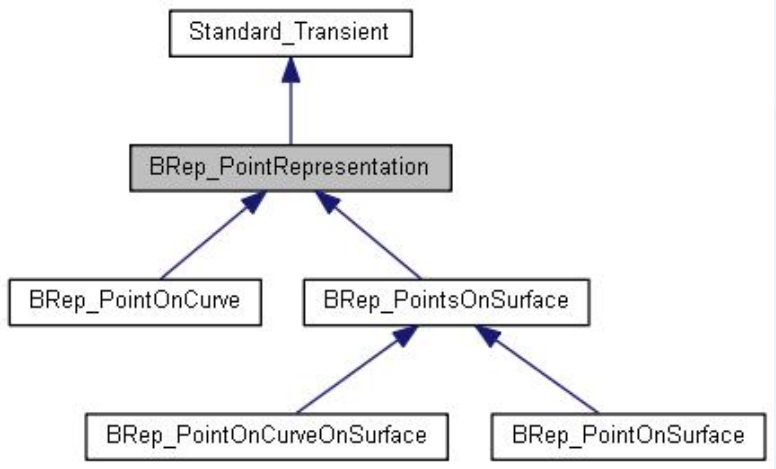
`BRep_TVertex`, `BRep_TEdge`, and `BRep_TFace` are defined to add geometric information to a topological model.

`BRep_TVertex`, `BRep_TEdge`, and `BRep_TFace` inherit `TopoDS_TShape`.

The geometric information is stored in a different way according to the topological entity.
Entities that store geometric information allows to describe:

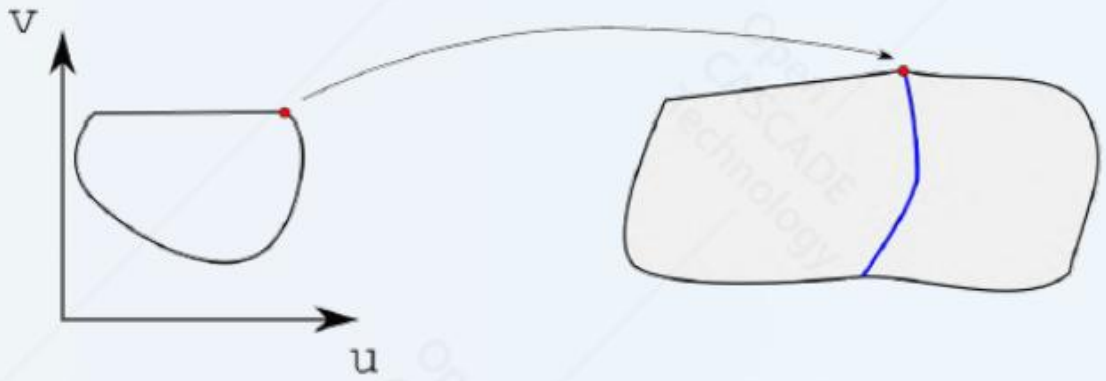
- An edge: a curve limited by vertices.
- A face: a surface limited by edges.
- A solid: space limited by faces.

Geometry in BRep_TVertex

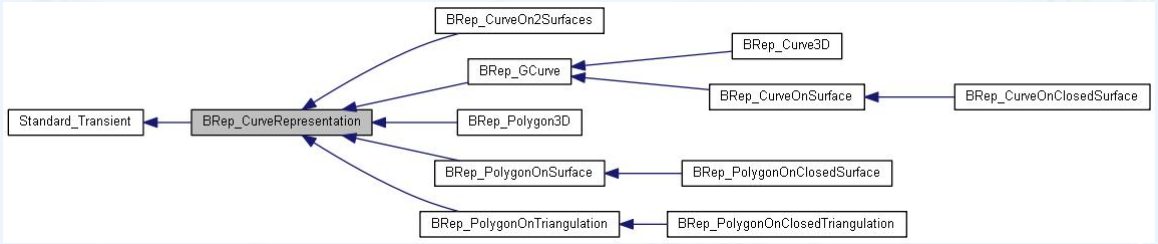


`BRep_TVertex` geometry is stored as:

- A 3D point (`gp_Pnt`) - for all vertices
- A list of point representations that can be:
  - A point on a curve (`Geom_Curve`, parameter) - if a vertex bounds an edge.
  - A point on a curve on a surface (`Geom_Surface`, `Geom2d_Curve`, parameter) - if a vertex bounds an edge lying on a surface.
  - A point on a surface (`Geom_Surface`, Uparameter, Vparameter) - if a vertex bounds a face.



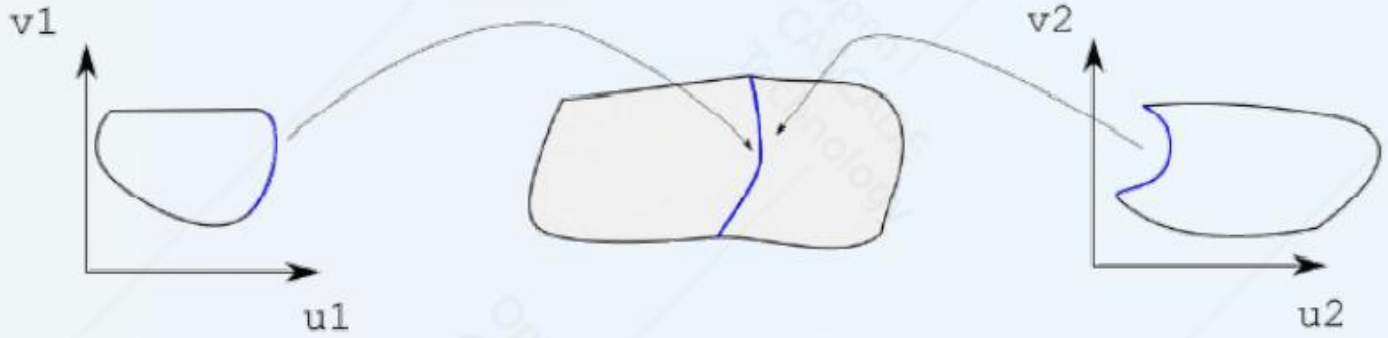
Geometry in BRep_TEdge



BRep_TEdge geometry is stored as a list of curve representations that can be :

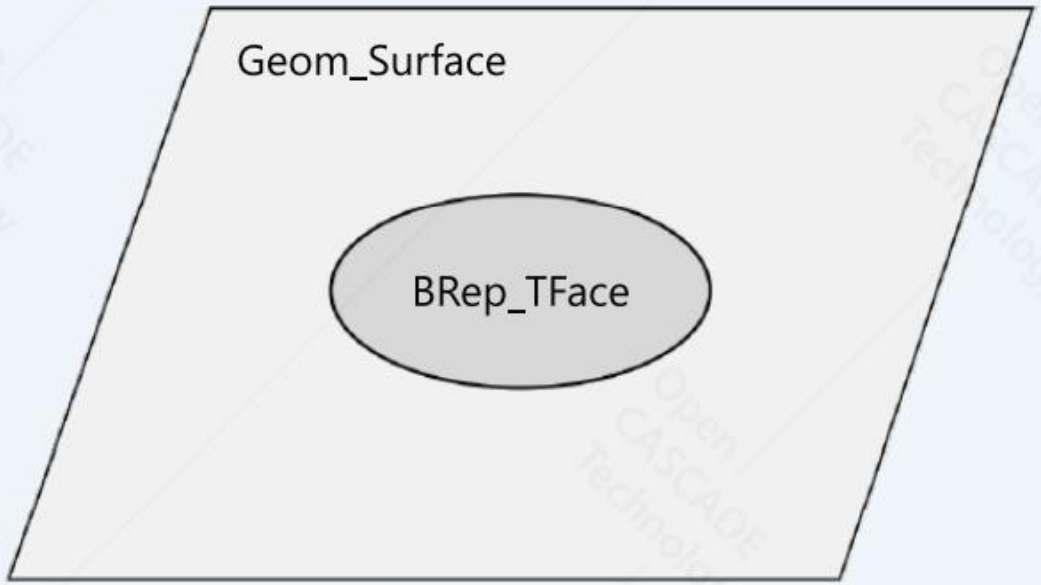
- A 3D curve and two parameters on a curve (Geom_Curve, FirstParameter, LastParameter).
- A curve on a surface, two parameters on a curve and two pairs of parameters on a surface (Geom2d_Curve, FirstParameter, LastParameter, Geom_Surface, FirstUVCoord, LastUVCoord).

The SameParameter property indicates whether different representations of the edge are parametrized synchronously i.e., points with the same parameter value on the 3D curve and each of 2D curves coincide (within the edge's tolerance)



Geometry in BRep_TFace

`BRep_TFace` geometry is stored as a `Geom_Surface`



Precision in B-Rep

Several geometric representations may be attached to a topological (B-Rep) object. For example, a vertex can be represented by:

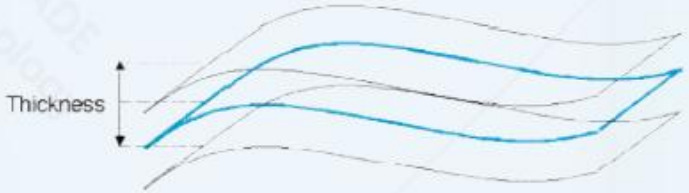
- 3D point;
- parameter on a curve;
- pair of parameters on a surface.

These representations are similar but rarely identical. For modeling algorithms, it is necessary to know exactly the precision associated with this approximation. The numeric value of this precision is associated with each B-Rep shape and is called `tolerance`. It defines the zone in which all geometrical representations of the object are located.

In BRep_TVertex precision defines the `radius of a sphere` around a 3D point:

In BRep_TEdge precision defines the `radius of a pipe` around a 3D curve

In BRep_TFace precision defines the `thickness` above and below a surface

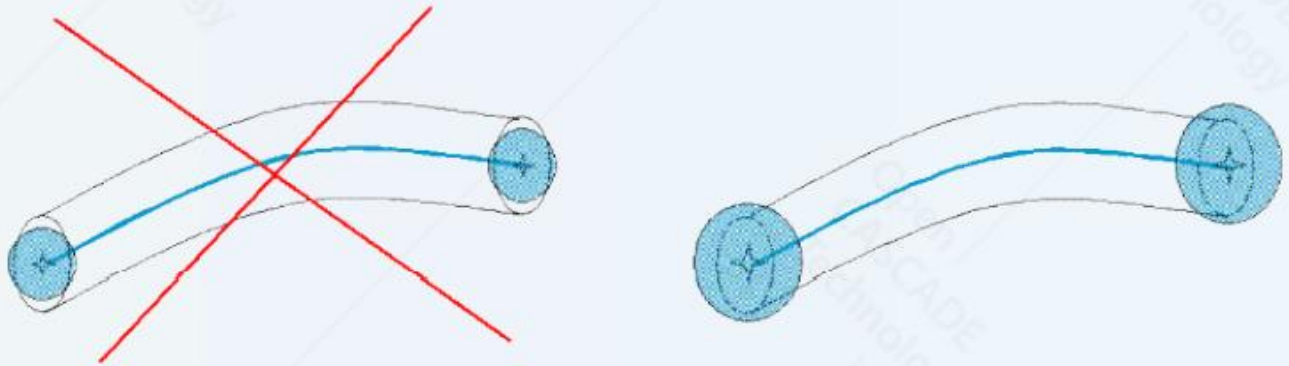


Precision in B-Rep

Since tolerance is associated with geometry carriers, it is defined by the algorithms creating or modifying the geometry in B-Rep.

Open CASCADE Technology requires that:

Tolerance(Vertex) >= Tolerance(Edge) >= Tolerance(Face)



Package: BRepAdaptor

OCCT enables topological entities usage in geometric algorithms via the adapter pattern. Adapters for boundary representation work as trimmed curves and surfaces, thus eliminating the necessity of manual trimming. The following adapters are available:

- `BRepAdapter_Curve` – curve adapter accepting edge.
- `BRepAdapter_Curve2d` – curve adapter accepting edge and face.
- `BRepAdapter_CompCurve` – curve adapter accepting wire.
- `BRepAdaptor_Surface` – surface adapter accepting face.

Note: Handle version of adapters are available. For instance, `BRepAdaptor_HSurface` class is handled-version of the surface adapter.

ModelingData BRep

- 访问**Geometry**
  - BRep_Tool

BRep_Tool class provides methods to access the geometric information in a model :

- **Tolerance(...)** Returns a Standard_Real
- **Surface(...)** Returns a Geom_Surface
- **Curve(...)** Returns a Geom_Curve
- **CurveOnSurface(...)** Returns a Geom2d_Curve
- **Pnt(...)** Returns a gp_Pnt

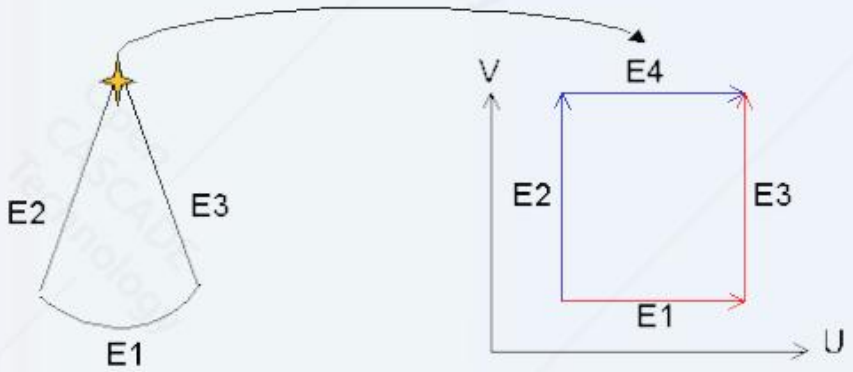
B-Rep particularities

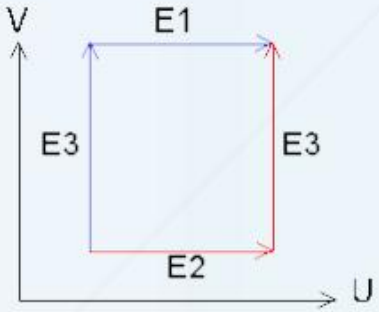
Representation of the same face in 3D space (3D topology) and in parametric space (UV topology) are usually topologically similar.

Sometimes 3D topology and UV topology are different. It is the case of seam edges and degenerated edges.

B-Rep particularities

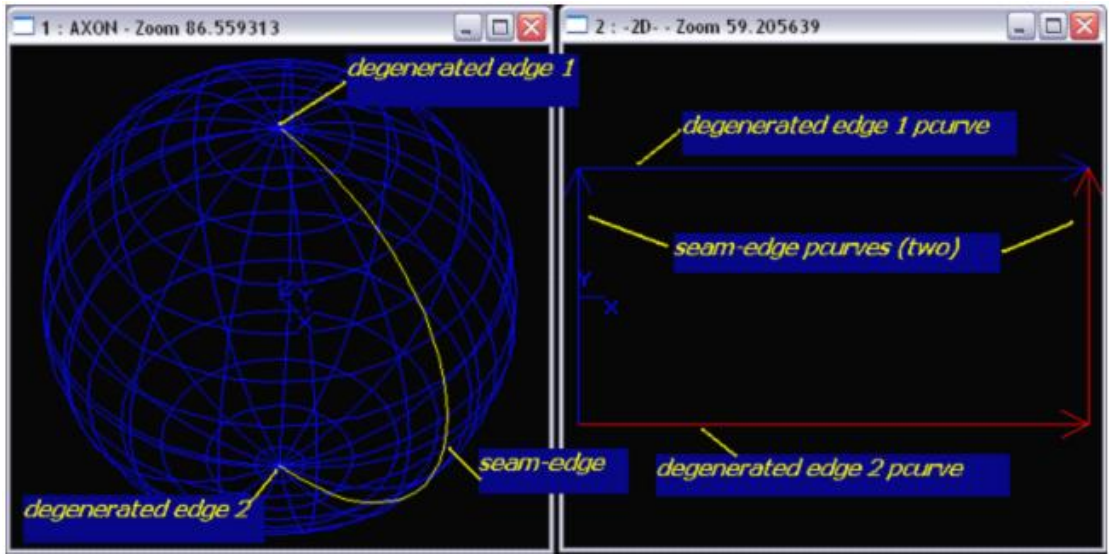
- A seam edge is an edge, which defines a seam on a closed face (usually built on a periodic surface). In this case, one 3D topology corresponds to several UV topologies.
- An edge is said to be degenerated when one or several UV 2d curves correspond to a single 3D vertex. Such edge does not have a 3d curve representation and includes the same vertex twice (with opposite orientations).





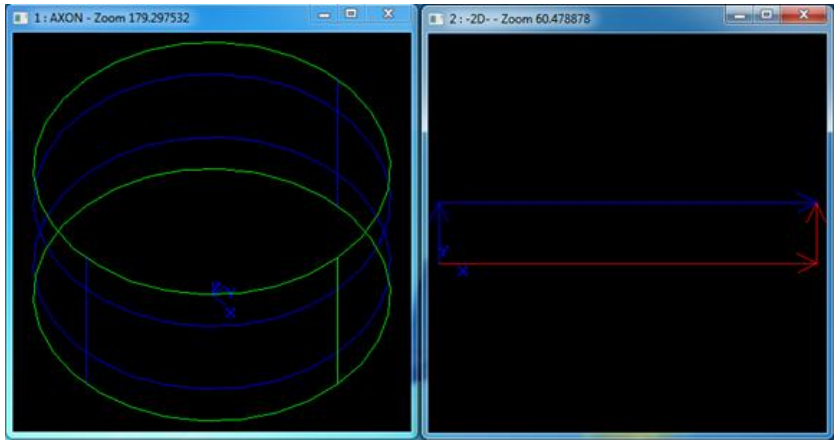
ModelingData BRep- Edge

- 特殊类型的边 Special edge types
  - 缝合边（seam edge）：即在同一个面上出现两次的边
  - 退化边（degenerated edge）：这种边位于曲面的奇异点处，在三维空间中退化为一个点；

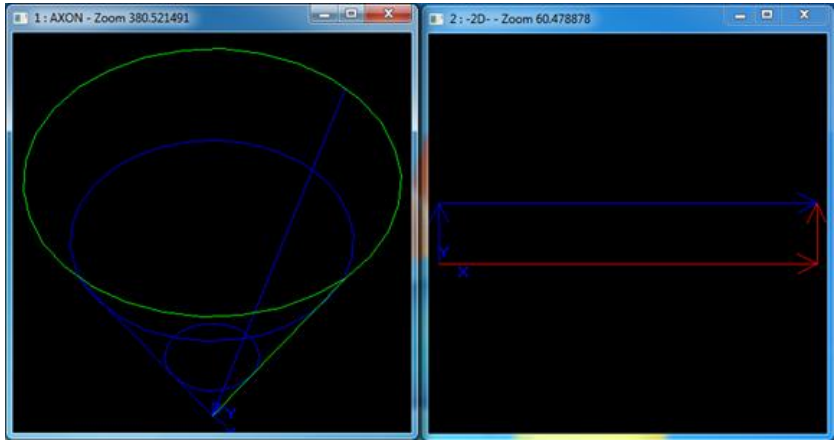


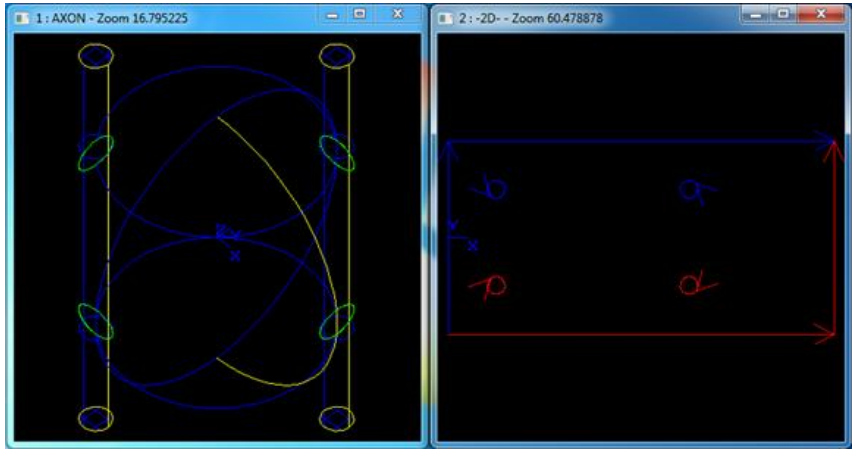
ModelingData BRep- PCurve

- PCurve - Curve On Surface
- 曲面上的曲线对应到曲面的参数空间中的曲线。




```
# 4. view the pcurves of a sphere face
sphere s 1

# make the topo face
mkface s s

# extract pcurves
pcurve s

# display pcurves in 2d viewer
av2d
2dfit
fit

# display sphere face in 3d viewer
vdisplay s
```





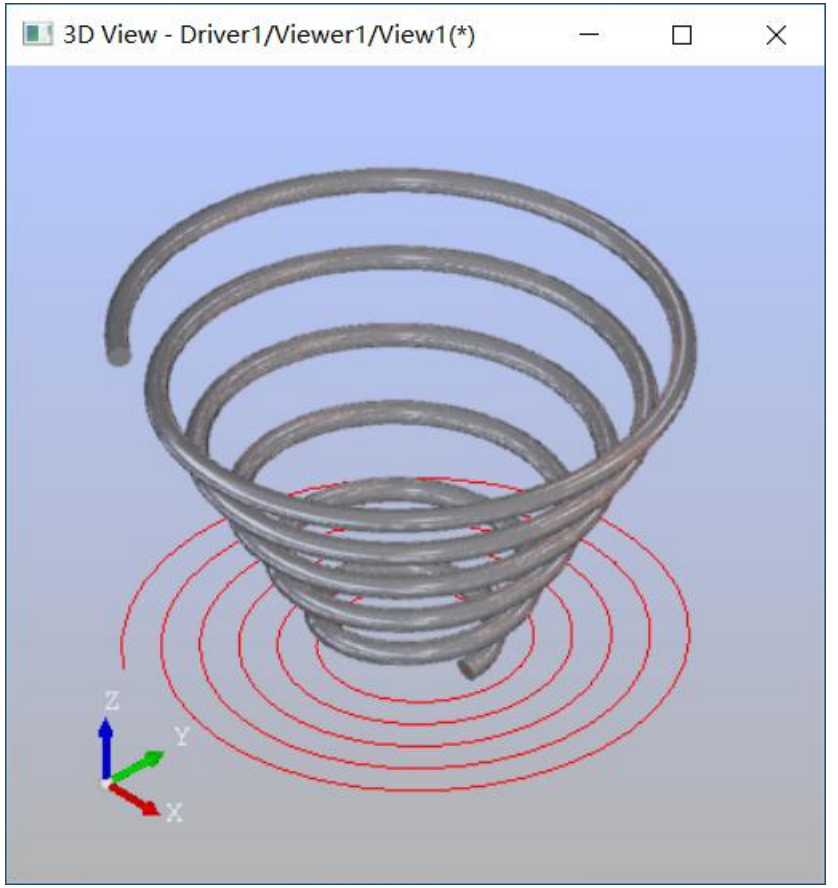
PCurve-Parametric Curve

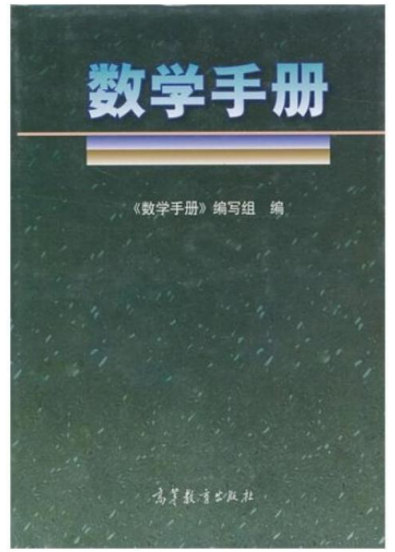
- 平面螺旋线-圆锥对数螺线

$$\begin{cases} x = \rho \sin\alpha \cos\theta \\ y = \rho \sin\alpha \sin\theta \\ z = \rho \cos\alpha \end{cases}$$

$$\rho = \rho_0 \exp\left(\frac{\sin\alpha}{tg\beta}\theta\right)$$

α为圆锥顶角的一半，β为螺旋角，ρ_0为常数





ModelingData Question

- Geometry Surface 为什么有求导数的函数?

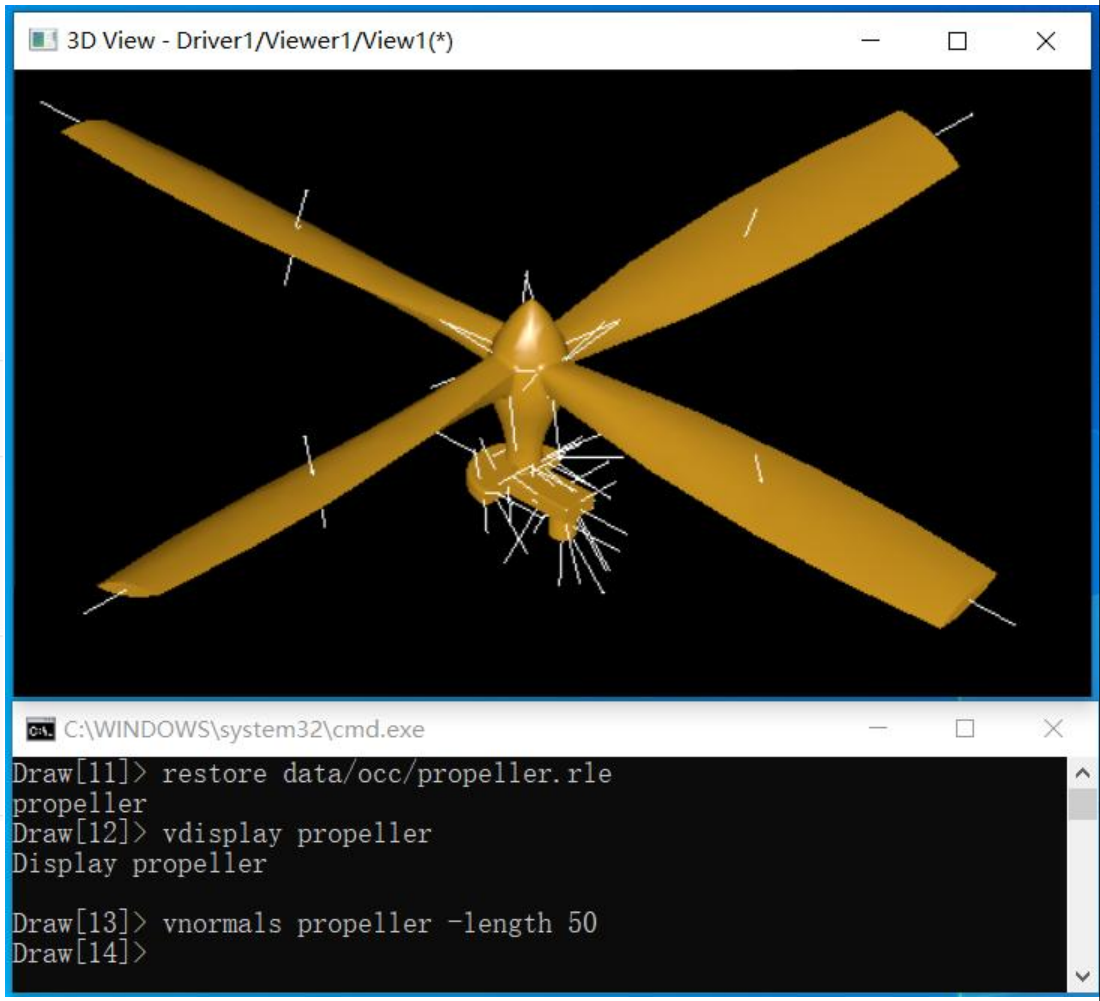
答：因为数值算法的要求。如求曲面上某点处的法向等;

$$Dn = Du \times Dv$$

virtual void **D0** (const **Standard_Real** U, const **Standard_Real** V, **gp_Pnt** &P) const =0
Computes the point of parameter U,V on the surface. More...

virtual void **D1** (const **Standard_Real** U, const **Standard_Real** V, **gp_Pnt** &P, **gp_Vec** &D1U, **gp_Vec** &D1V) const =0
Computes the point P and the first derivatives in the directions U and V at this point. Raised if the continuity of the surface is not C1. More...

virtual void **D2** (const **Standard_Real** U, const **Standard_Real** V, **gp_Pnt** &P, **gp_Vec** &D1U, **gp_Vec** &D1V, **gp_Vec** &D2U, **gp_Vec** &D2V, **gp_Vec** &D2UV) const =0
Computes the point P, the first and the second derivatives in the directions U and V at this point. Raised if the continuity of the surface is not C2. More...



ModelingData Question

- Geometry Curve 为什么有求导数的函数?

答：因为数值算法的要求。如求曲线长度、极值计算，如点到曲线距离等。

$$L = \int_{u1}^{u2} |C'(u)| du$$

virtual void **D0** (const **Standard_Real** U, **gp_Pnt** &P) const =0
Returns in P the point of parameter U. If the curve is periodic then the returned point is P(U) with U = Ustart + (U - Uend) where Ustart and Uend are the parametric bounds of the curve. More...

virtual void **D1** (const **Standard_Real** U, **gp_Pnt** &P, **gp_Vec** &V1) const =0
Returns the point P of parameter U and the first derivative V1. Raised if the continuity of the curve is not C1. More...

virtual void **D2** (const **Standard_Real** U, **gp_Pnt** &P, **gp_Vec** &V1, **gp_Vec** &V2) const =0
Returns the point P of parameter U, the first and second derivatives V1 and V2. Raised if the continuity of the curve is not C2. More...

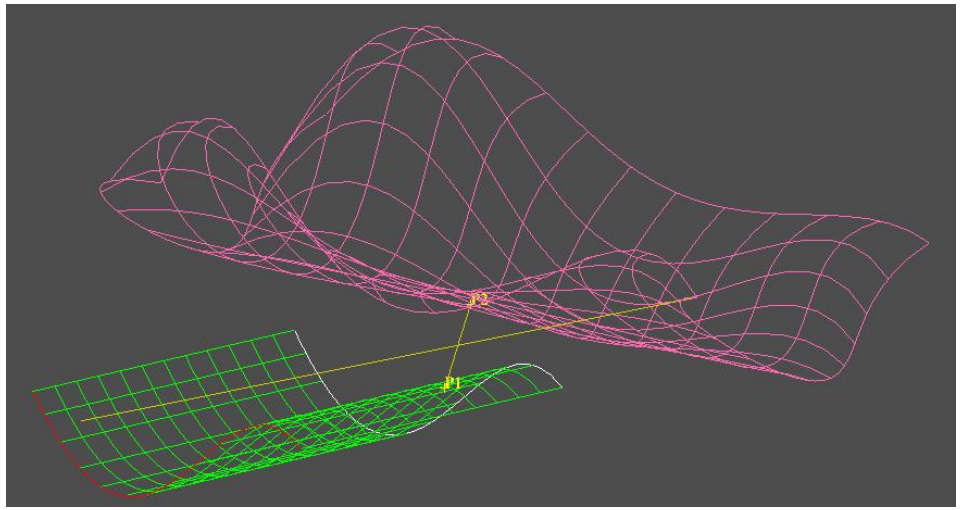
```
151 Standard_Real CPnts_AbscissaPoint::Length(const Adaptor3d_Curve& C,
152             const Standard_Real U1,
153             const Standard_Real U2)
154 {
155   CPnts_MyGaussFunction FG;
156 //POP pout WNT
157   CPnts_RealFunction rf = f3d;
158   FG.Init(rf,(Standard_Address)&C);
159 //  FG.Init(f3d,(Standard_Address)&C);
160   math_GaussSingleIntegration TheLength(FG, U1, U2, order(C));
161   if (!TheLength.IsDone()) {
162     throw Standard_ConstructionError();
163   }
164   return Abs(TheLength.Value());
165 }
```

Extrema

- Extrema_FuncExtCS
- Extrema_GenExtCS

```
/*-----------------------------------------------------------------------------
 Fonction permettant de rechercher une distance extremale entre une courbe C
et une surface S.
 Cette classe herite de math_FunctionWithDerivative et est utilisee par
les algorithmes math_FunctionRoot et math_FunctionRoots.
{ F1(t,u,v) = (C(t)-S(u,v)).Dtc(t) }
{ F2(t,u,v) = (C(t)-S(u,v)).Dus(u,v) }
{ F3(t,u,v) = (C(t)-S(u,v)).Dvs(u,v) }
{ Dtf1(t,u,v) = Dtc(t).Dtc(t)+(C(t)-S(u,v)).Dttc(t)
              = ||Dtc(t)||**2+(C(t)-S(u,v)).Dttc(t) }
{ Duf1(t,u,v) = -Dus(u,v).Dtc(t) }
{ Dvf1(t,u,v) = -Dvs(u,v).Dtc(t) }
{ Dtf2(t,u,v) = Dtc(t).Dus(u,v) }
{ Duf2(t,u,v) = -Dus(u,v).Dus(u,v)+(C(t)-S(u,v)).Duus(u,v)
              = -||Dus(u,v)||**2+(C(t)-S(u,v)).Duus(u,v) }
{ Dvf2(t,u,v) = -Dvs(u,v).Dus(u,v)+(C(t)-S(u,v)).Duvs(u,v) }
{ Dtf3(t,u,v) = Dtc(t).Dvs(u,v) }
{ Duf3(t,u,v) = -Dus(u,v).Dvs(u,v)+(C(t)-S(u,v)).Duvs(u,v) }
{ Dvf3(t,u,v) = -Dvs(u,v).Dvs(u,v)+(C(t)-S(u,v)).Dvvs(u,v) }
-----------------------------------------------------------------------------*/
```

$$\begin{cases} F_1(t,u,v) = (C(t) - S(u,v)).Dtc(t) \\ F_2(t,u,v) = (C(t) - S(u,v)).Dus(t) \\ F_3(t,u,v) = (C(t) - S(u,v)).Dvs(t) \end{cases}$$

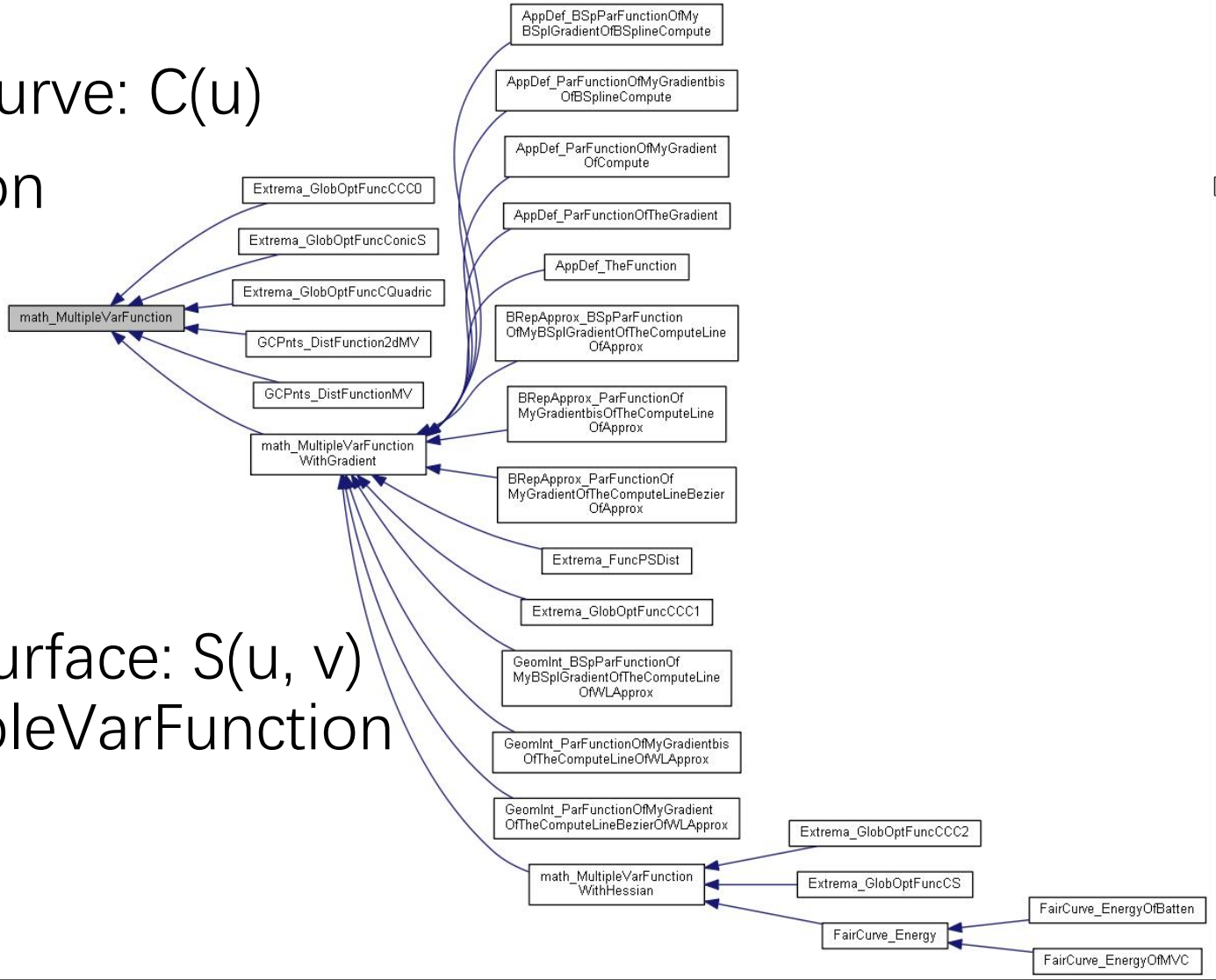



```
Standard_Boolean Extrema_FuncExtCS::Value(const math_Vector& UV,
            math_Vector& F)
{
  if (!myCinit || !mySinit) throw Standard_TypeMismatch();

  myt = UV(1);
  myU = UV(2);
  myV = UV(3);

//  gp_Vec Dtc, Dttc;
  gp_Vec Dtc;
///  gp_Vec Dus, Dvs, Duvs, Duus, Dvvs;
  gp_Vec Dus, Dvs;
  myC->D1(myt, myP1, Dtc);
  myS->D1(myU,myV,myP2,Dus,Dvs);

  gp_Vec P1P2 (myP2,myP1);

  F(1) = P1P2.Dot(Dtc);
  F(2) = P1P2.Dot(Dus);
  F(3) = P1P2.Dot(Dvs);

  return  Standard_True;
}
```

ModelingData Aha!

- Geometry Curve: C(u)
  math_Function

- Geometry Surface: S(u, v)
  math_MultipleVarFunction



造型算法

ModelingAlgorithms

ModelingAlgorithms

- BRepBuilderAPI
- BRepBuilderAPI_MakeFace

```
void  BRepLib_MakeFace::Init(const Handle(Geom_Surface)& SS,
                             const Standard_Real Um,
                             const Standard_Real UM,
                             const Standard_Real Vm,
                             const Standard_Real VM,
                             const Standard_Real TolDegen)
{
  myError = BRepLib_FaceDone;

  Standard_Real UMin = Um;
  Standard_Real UMax = UM;
  Standard_Real VMin = Vm;
  Standard_Real VMax = VM;

  Standard_Real umin,umax,vmin,vmax,T;

  Handle(Geom_Surface) S = SS, BS = SS;
  Handle(Geom_RectangularTrimmedSurface) RS =
    Handle(Geom_RectangularTrimmedSurface)::DownCast(S);
  if (!RS.IsNull())
    BS = RS->BasisSurface();

  Standard_Boolean OffsetSurface =
    (BS->DynamicType() == STANDARD_TYPE(Geom_OffsetSurface));

  // adjust periodical surface or reordonate
  // check if the values are in the natural range
  Standard_Real epsilon = Precision::PConfusion();

  BS->Bounds(umin,umax,vmin,vmax);
```

ModelingAlgorithms

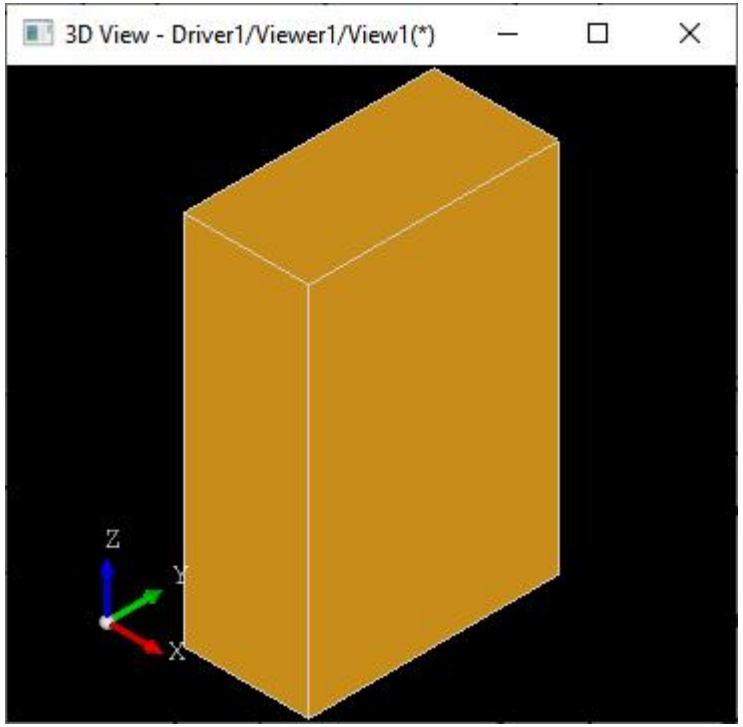
- Primitives

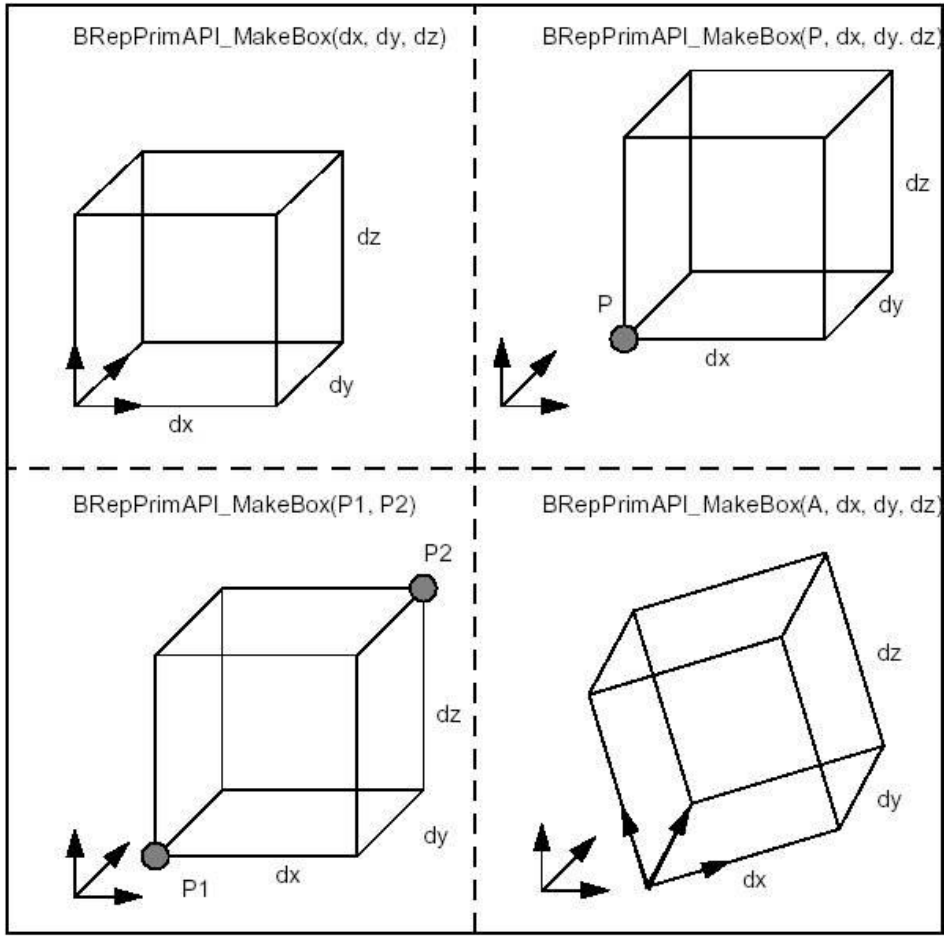
基本体构造算法主要是生成常见的基本体的BREP数据，这个的基础是理解BRep表示法的数据结构。

ModelingAlgorithms

- Primitive - Box

```
TopoDS_Shape aBox =
BRepPrimAPI_MakeBox(1,2,3);
```

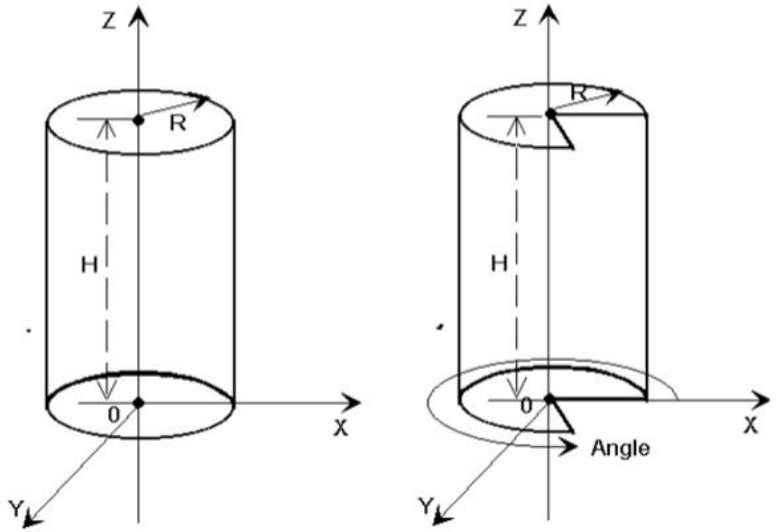


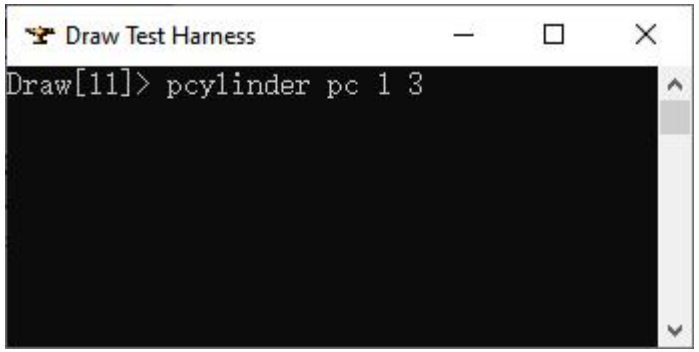


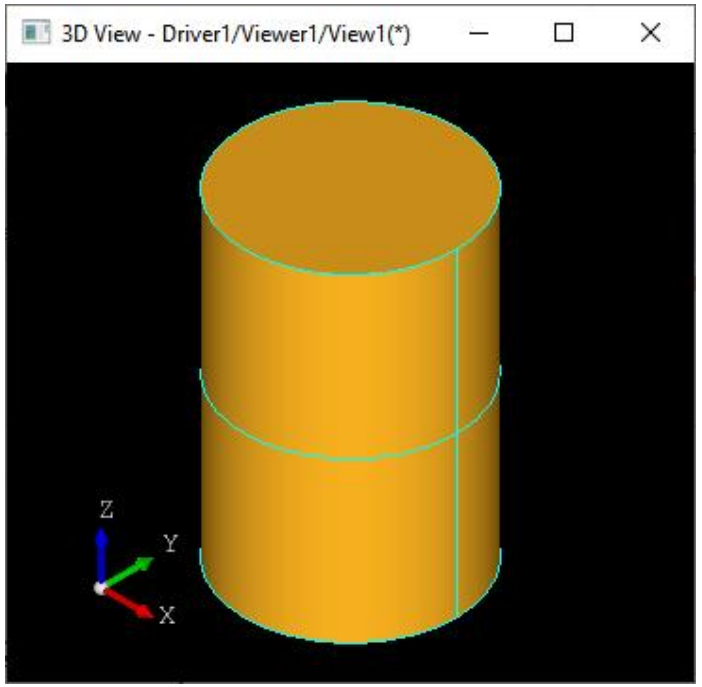
ModelingAlgorithms

- Primitive - Cylinder

```
TopoDS_Shape aCylinder =
BRepPrimAPI_MakeCylinder(1,3);
```



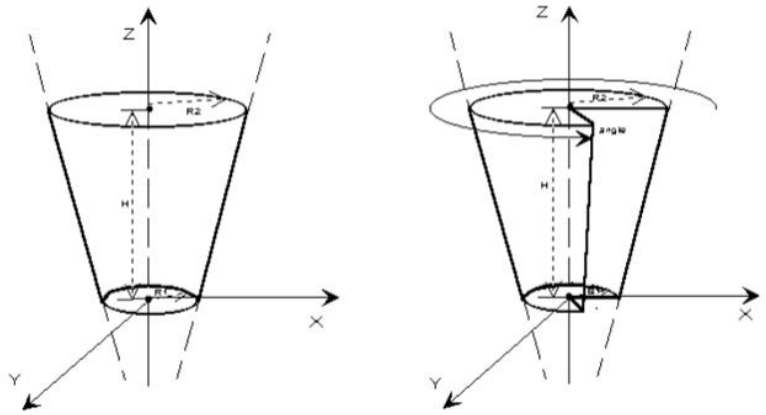


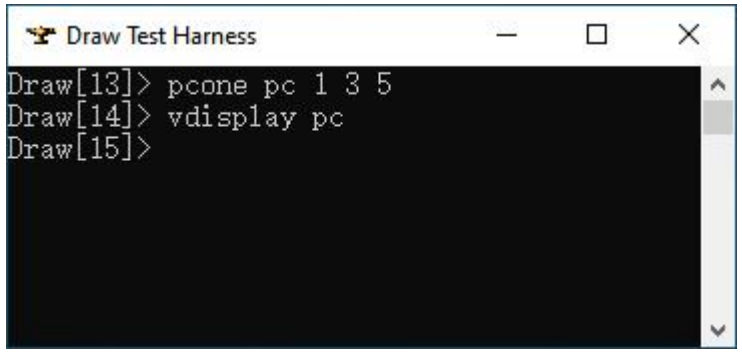


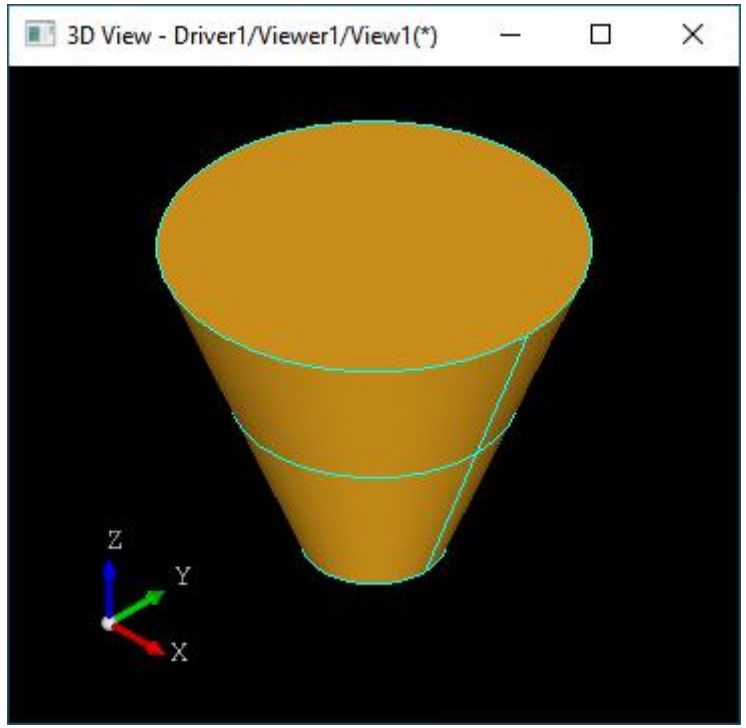
ModelingAlgorithms

- Primitive - Cone

```
TopoDS_Shape aCone =
BRepPrimAPI_MakeCone(1,3, 5);
```



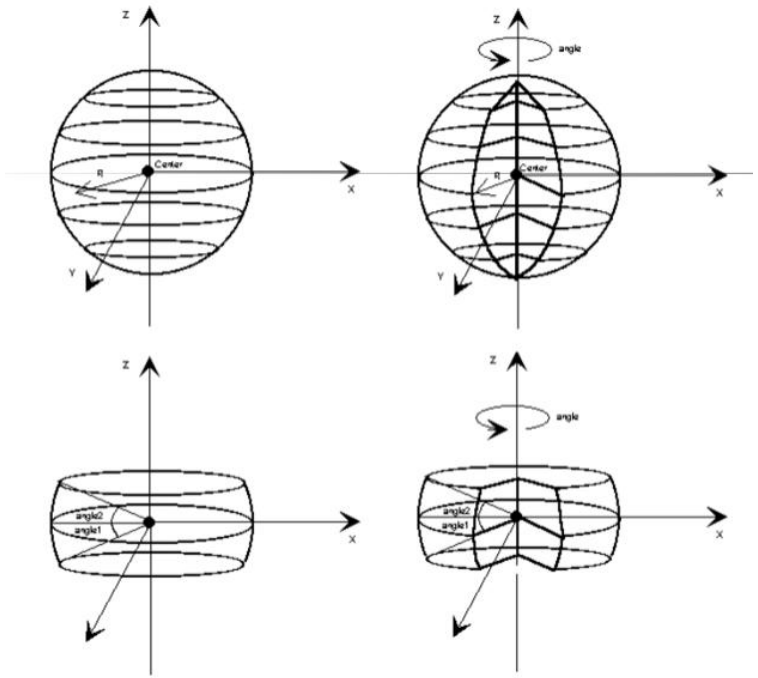


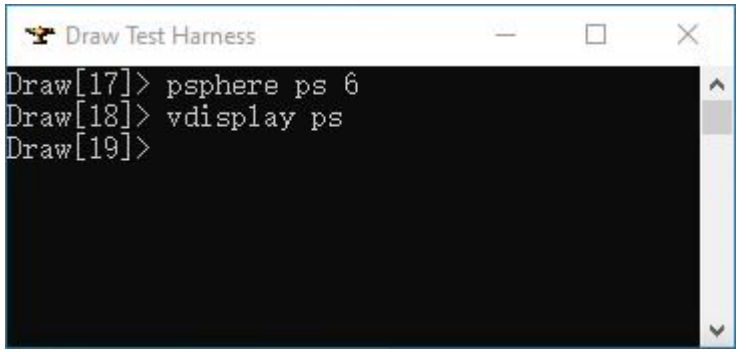


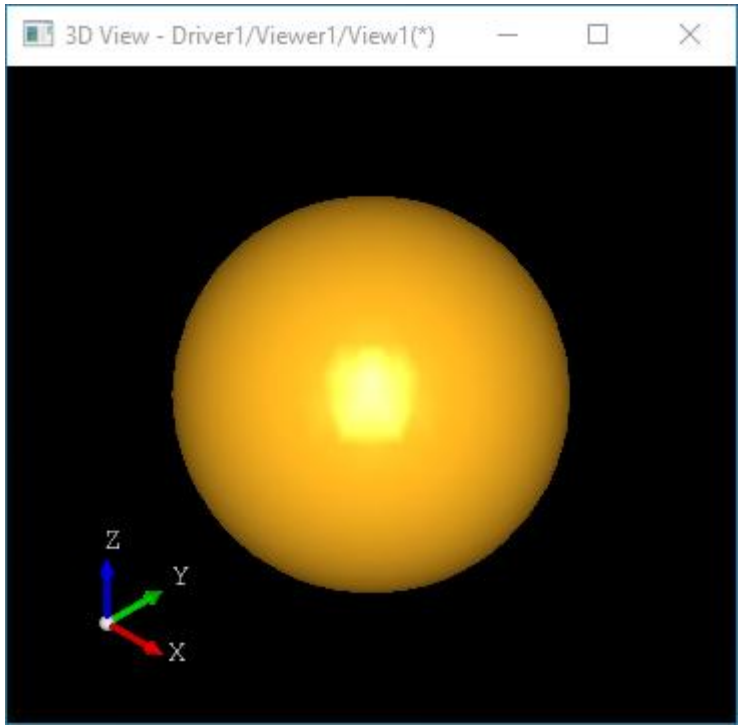
ModelingAlgorithms

- Primitive - Sphere

```
TopoDS_Shape aSphere =
BRepPrimAPI_MakeSphere(6);
```



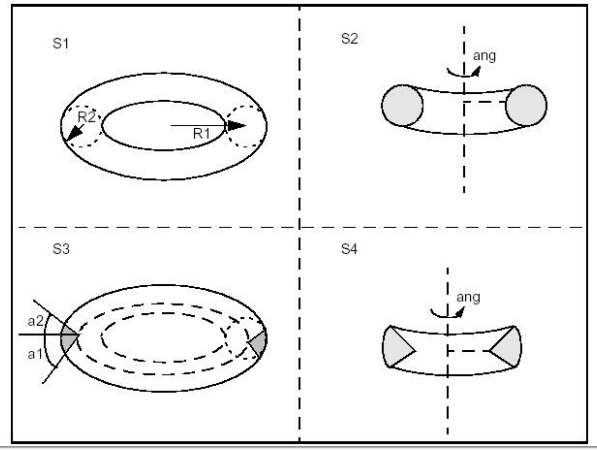


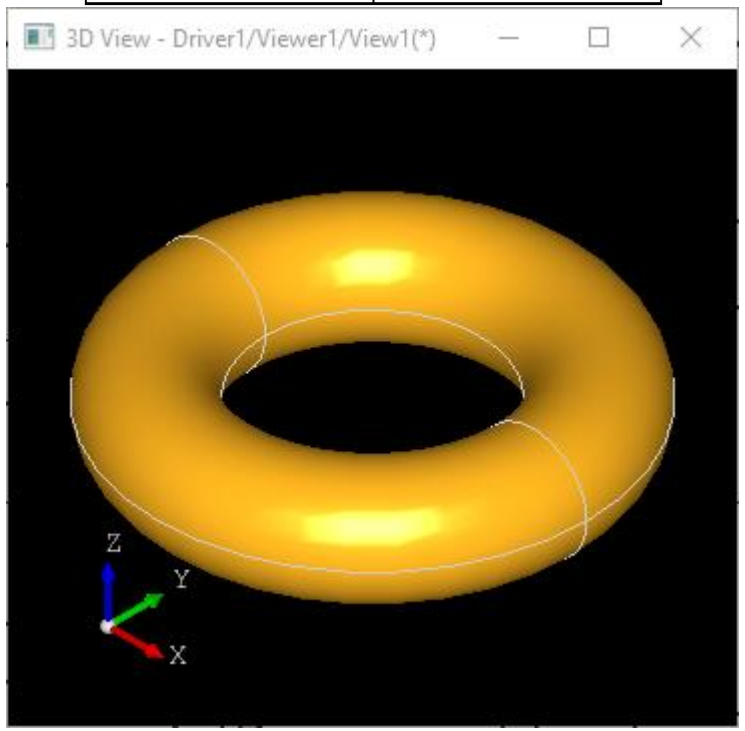


ModelingAlgorithms

- Primitive - Torus

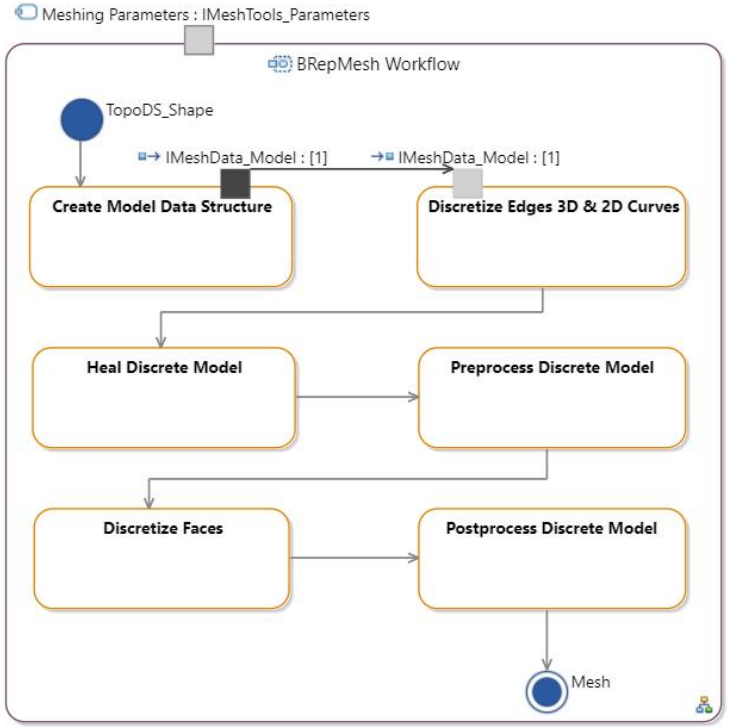
```
TopoDS_Shape aTorus = BRepPrimAPI_MakeTorus(30, 10);
```





BRepMesh Architecture

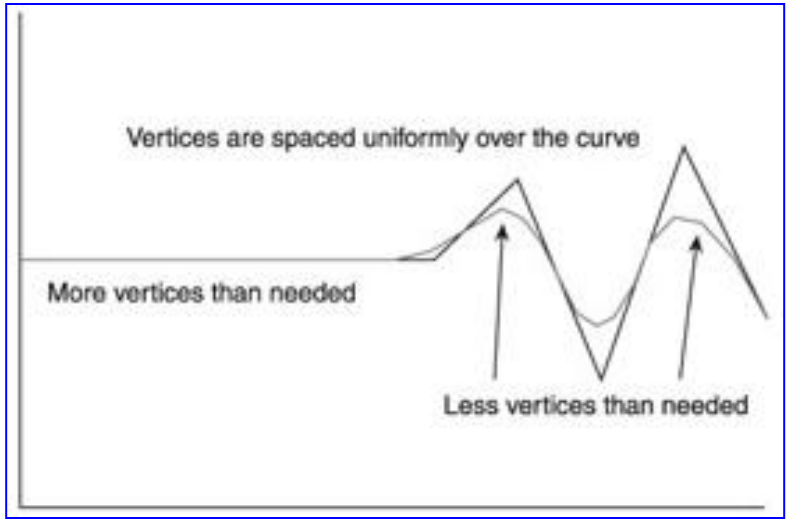
- Creation of model data structure: source TopoDS_Shape passed to algorithm is analyzed and exploded into faces and edges. The reflection corresponding to each topological entity is created in the data model. Note that underlying algorithms use the data model as input and access it via a common interface which allows creating a custom data model with necessary dependencies between particular entities (see the paragraph "Data model interface");
- Discretize edges 3D & 2D curves: 3D curve as well as an associated set of 2D curves of each model edge is discretized in order to create a coherent skeleton used as a base in face meshing process. If an edge of the source shape already contains polygonal data which suits the specified parameters, it is extracted from the shape and stored in the model as is. Each edge is processed separately, the adjacency is not taken into account;
- Heal discrete model: the source TopoDS_Shape can contain problems, such as open wires or self-intersections, introduced during design, exchange or modification of model. In addition, some problems like self-intersections can be introduced by roughly discretized edges. This stage is responsible for analysis of a discrete model in order to detect and repair problems or to refuse further processing of a model part in case if a problem cannot be solved;
- Preprocess discrete model: defines actions specific to the implemented approach to be performed before meshing of faces. By default, this operation iterates over model faces, checks the consistency of existing triangulations and cleans topological faces and adjacent edges from polygonal data in case of inconsistency or marks a face of the discrete model as not required for the computation;
- Discretize faces: represents the core part performing mesh generation for a particular face based on 2D discrete data. This operation caches polygonal data associated with face edges in the data model for further processing and stores the generated mesh to TopoDS_Face;
- Postprocess discrete model: defines actions specific for the implemented approach to be performed after meshing of faces. By default, this operation stores polygonal data obtained at the previous stage to TopoDS_Edge objects of the source model.

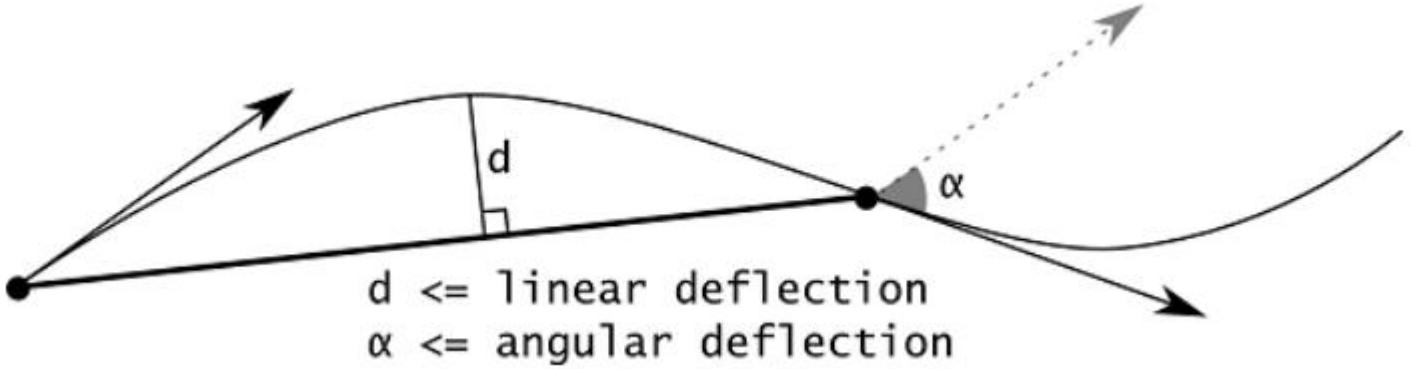


General workflow of BRepMesh component

Discretize edges 3D & 2D curves

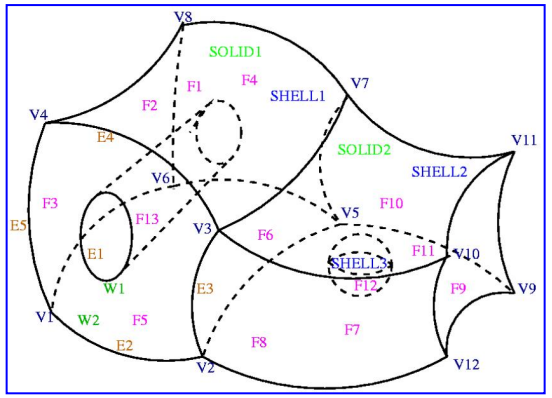
- **离散边Edge**
  - ***GCPnts_TangentialDeflection***
  - GCPnts_UniformAbscissa
  - GCPnts_UniformDeflection
  - GCPnts_QuasiUniformDeflection
  - GCPnts_QuasiUniformAbscissa

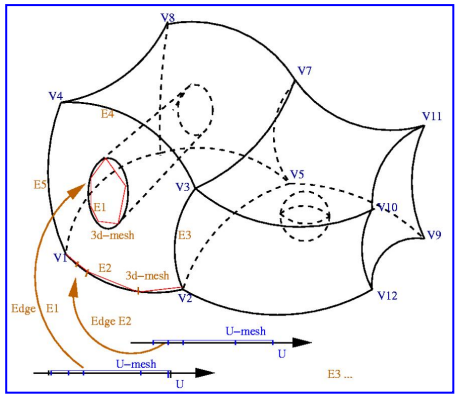


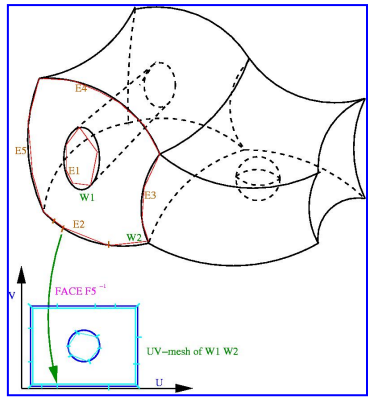


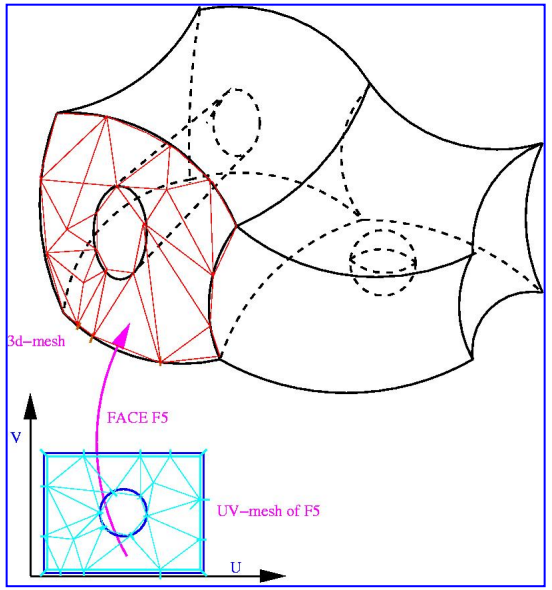
Discretize faces

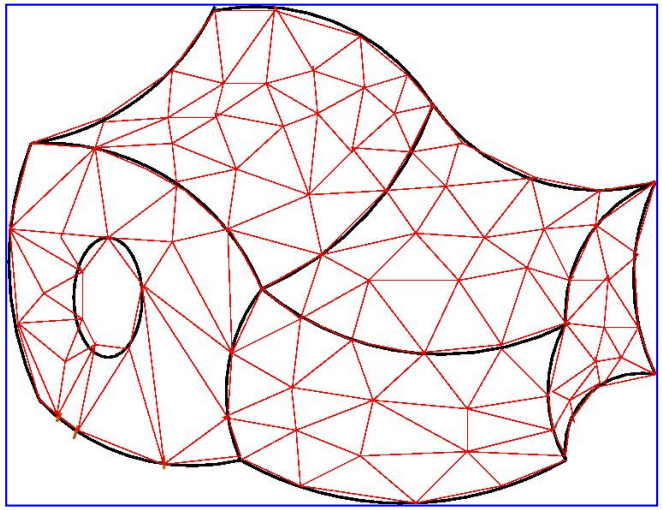
- 离散面Face







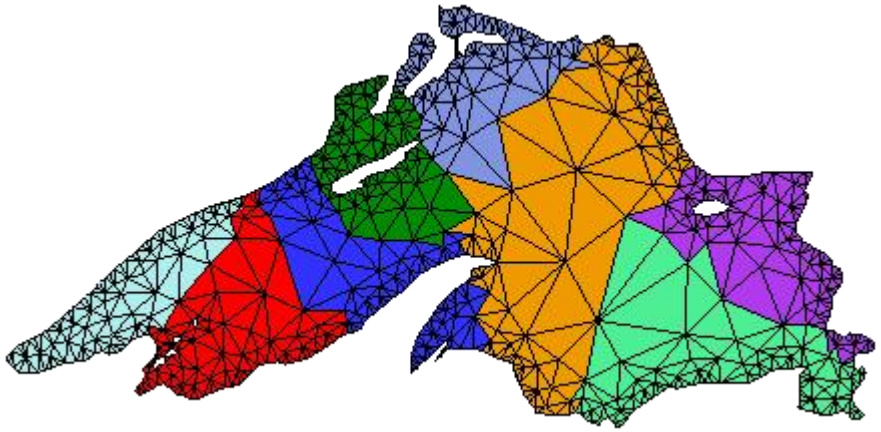




Delaunay Triangulation

- 三角剖分库
  - Triangle http://www.cs.cmu.edu/~quake/triangle.html
  - DELABELLA - Delaunay triangulation library

https://github.com/msokalski/delabella



Triangle

A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator.

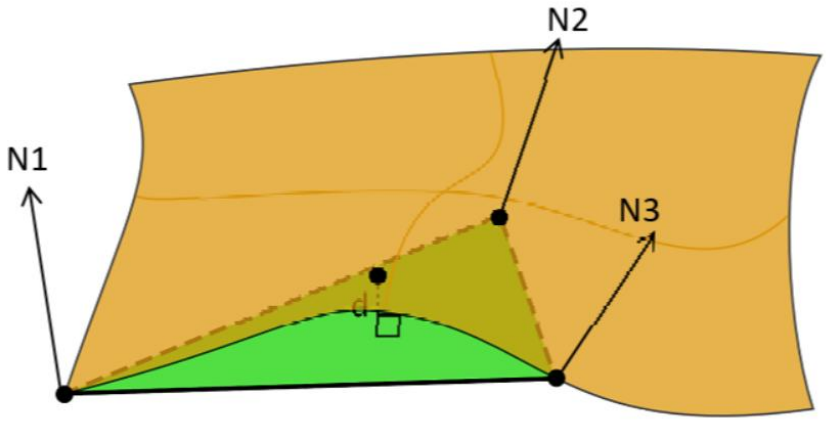
Jonathan Richard Shewchuk
Computer Science Division
University of California at Berkeley

Discretize faces

- Deflection Control 精度控制

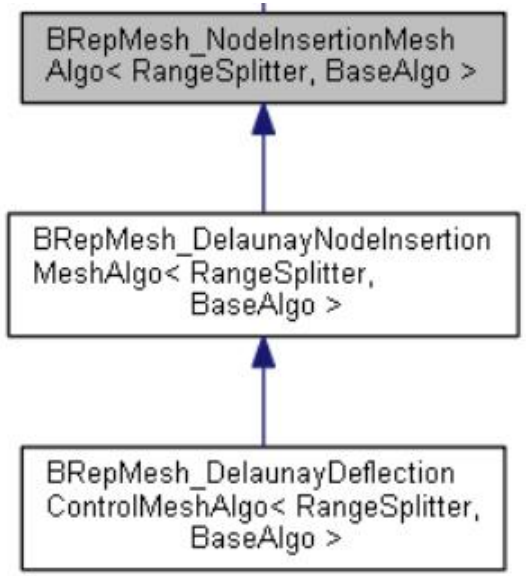
通过插入点的方式来控制精度。

BRepMesh_DelaunayDeflectionControlMeshAlgo



d ≤ linear deflection;
N1.Angle(N2) ≤ AngleInterior;
N2.Angle(N3) ≤ AngleInterior;

Linear and angular interior deflections



BRepMesh_DelaunayDeflectionControlMeshAlgo.hxx - Notepad2

File Edit View Settings ?

```
210
211  // Check geometry of the given triangle. If triangle does not suit specified deflection, inserts new point.
212  void splitTriangleGeometry(const BRepMesh_Triangle& theTriangle)
213  {
214    if (theTriangle.Movability() != BRepMesh_Deleted)
215    {
216      Standard_Integer aNodexIndices[3];
217      this->getStructure()->ElementNodes(theTriangle, aNodexIndices);
218
219      TriangleNodeInfo aNodesInfo[3];
220      getTriangleInfo(theTriangle, aNodexIndices, aNodesInfo);
221
222      gp_Vec aNormal;
223      gp_Vec aLinkVec[3];
224      if (computeTriangleGeometry(aNodesInfo, aLinkVec, aNormal))
225      {
226        myIsAllDegenerated = Standard_False;
227
228        const gp_XY aCenter2d = (aNodesInfo[0].Point2d +
229                                 aNodesInfo[1].Point2d +
230                                 aNodesInfo[2].Point2d) / 3.;
231
232        usePoint(aCenter2d, NormalDeviation(aNodesInfo[0].Point, aNormal));
233        splitLinks(aNodesInfo, aNodexIndices);
234      }
235    }
236  }
```

Ln 1 : 474 Col 1 Sel 0 | 14.7 KB | ANSI | LF | INS | C/C++ Source Code

Range splitter

- Range splitter tools provide functionality to generate internal surface nodes defined within the range computed using discrete model data. The base functionality is provided by BRepMesh_DefaultRangeSplitter which can be used without modifications in case of planar surface. The default splitter does not generate any internal node.

- BRepMesh_ConeRangeSplitter, BRepMesh_CylinderRangeSplitter and BRepMesh_SphereRangeSplitter are specializations of the default splitter intended for quick generation of internal nodes for the corresponding type of analytical surface.

- BRepMesh_UVParamRangeSplitter implements base functionality taking discretization points of face border into account for node generation. Its successors BRepMesh_TorusRangeSplitter and BRepMesh_NURBSRangeSplitter extend the base functionality for toroidal and NURBS surfaces correspondingly.

BRepMesh_DelabellaMeshAlgoFactory.cxx - Notepad2

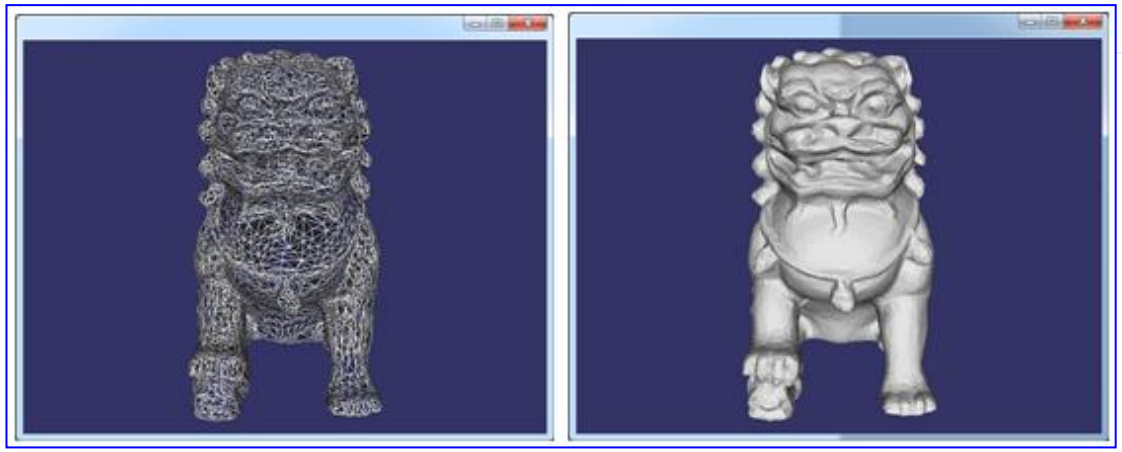
File Edit View Settings ?

```
79 //=============================================================================
80 // Function: GetAlgo
81 // Purpose :
82 //=============================================================================
83 Handle(IMeshTools_MeshAlgo) BRepMesh_DelabellaMeshAlgoFactory::GetAlgo(
84   const GeomAbs_SurfaceType    theSurfaceType,
85   const IMeshTools_Parameters& theParameters) const
86 {
87   switch (theSurfaceType)
88   {
89   case GeomAbs_Plane:
90     return theParameters.InternalVerticesMode ?
91       new NodeInsertionMeshAlgo<BRepMesh_DefaultRangeSplitter>::Type :
92       new BaseMeshAlgo::Type;
93     break;
94
95   case GeomAbs_Sphere:
96     {
97       NodeInsertionMeshAlgo<BRepMesh_SphereRangeSplitter>::Type* aMeshAlgo =
98         new NodeInsertionMeshAlgo<BRepMesh_SphereRangeSplitter>::Type;
99       aMeshAlgo->SetPreProcessSurfaceNodes (Standard_True);
100      return aMeshAlgo;
101    }
102    break;
103
104  case GeomAbs_Cylinder:
105    return theParameters.InternalVerticesMode ?
106      new DefaultNodeInsertionMeshAlgo<BRepMesh_CylinderRangeSplitter>::Type :
107      new DefaultBaseMeshAlgo::Type;
108    break;
109
```

Ln 1 : 146 Col 1 Sel 0 | 4.79 KB | ANSI | LF | INS | C/C++ Source Code

Mesh Data Structure

- 离散边Edge得到多段线：
  - Poly_Polygon2D
  - Poly_Polygon3D
  - Poly_PolygonOnTraingulation
- 离散面Face得到三角网格：
  - Poly_Triangulation



BRep_Tool Class Reference

Provides class methods to access to the geometry of BRep shapes. More...

`#include <BRep_Tool.hxx>`

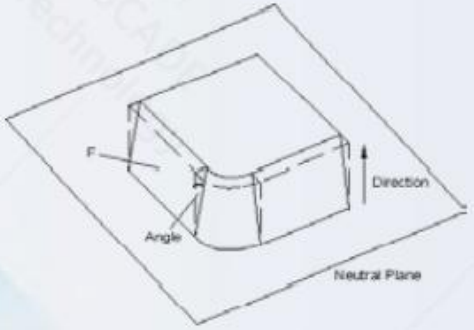
Static Public Member Functions

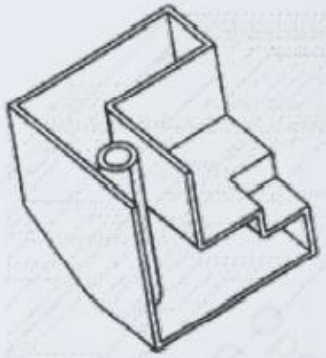
static **Standard_Boolean**	**IsClosed** (const **TopoDS_Shape** &S) If S is Shell, returns True if it has no free boundaries (edges). If S is Wire, returns True if it has no free ends (vertices). (Internal and External sub-shepes are ignored in these checks) If S is Edge, returns True if its vertices are the same. For other shape types returns S.Closed(). More...
static const **Handle**< **Geom_Surface** > &	**Surface** (const **TopoDS_Face** &F, **TopLoc_Location** &L) Returns the geometric surface of the face. Returns in <L> the location for the surface. More...
static **Handle**< **Geom_Surface** >	**Surface** (const **TopoDS_Face** &F) Returns the geometric surface of the face. It can be a copy if there is a Location. More...
static const **Handle**< **Poly_Triangulation** > &	**Triangulation** (const **TopoDS_Face** &F, **TopLoc_Location** &L) Returns the Triangulation of the face. It is a null handle if there is no triangulation. More...
static **Standard_Real**	**Tolerance** (const **TopoDS_Face** &F) Returns the tolerance of the face. More...

Modeling API: BRepOffsetAPI

This package provides additional tools for construction, such as:

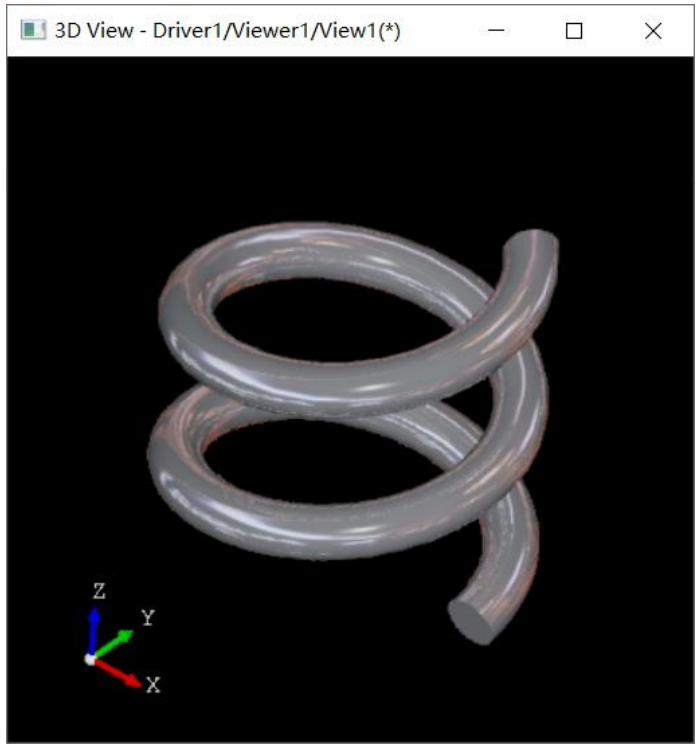
- `BRepOffsetAPI_ThruSections` – builds a shell or a solid from a sequence of wire profiles.
- `BRepOffsetAPI_DraftAngle` – tapers a set of faces of a shape with a given angle.
- `BRepOffsetAPI_MakeOffsetShape` – builds offset shape on the given shape.
- `BRepOffsetAPI_MakeThickSolid` – builds a hollowed solid from a given solid and a set of faces to be removed.
- `BRepOffsetAPI_MakePipe` – builds a pipe shape by sweeping a base shape (profile) along a wire (spine).
- `BRepOffsetAPI_MakeEvolved` – builds an evolved shape from a planar face or wire (spine) and a wire (profile).



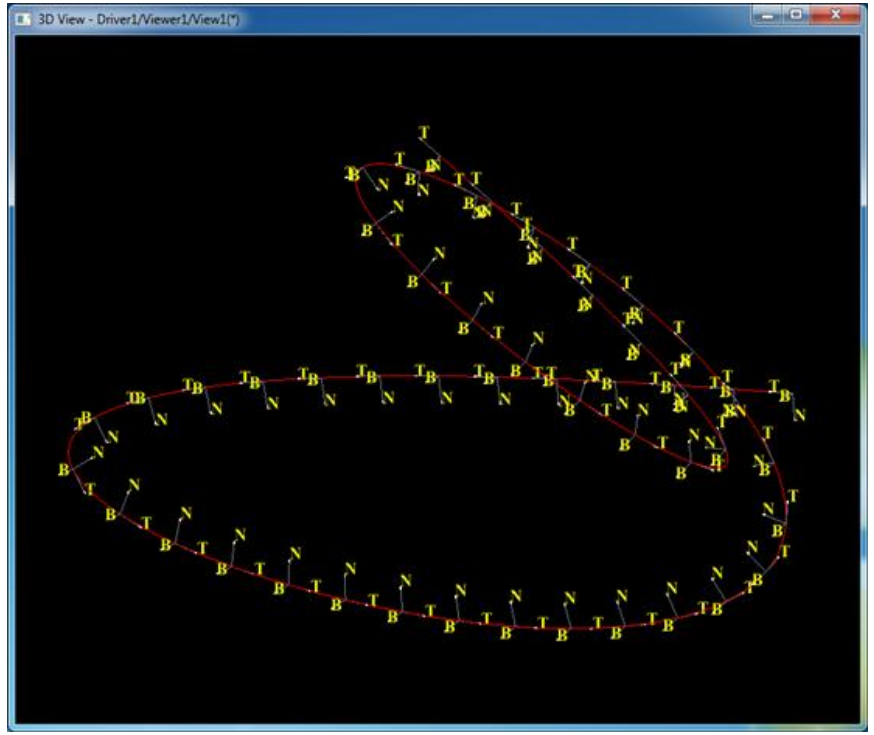


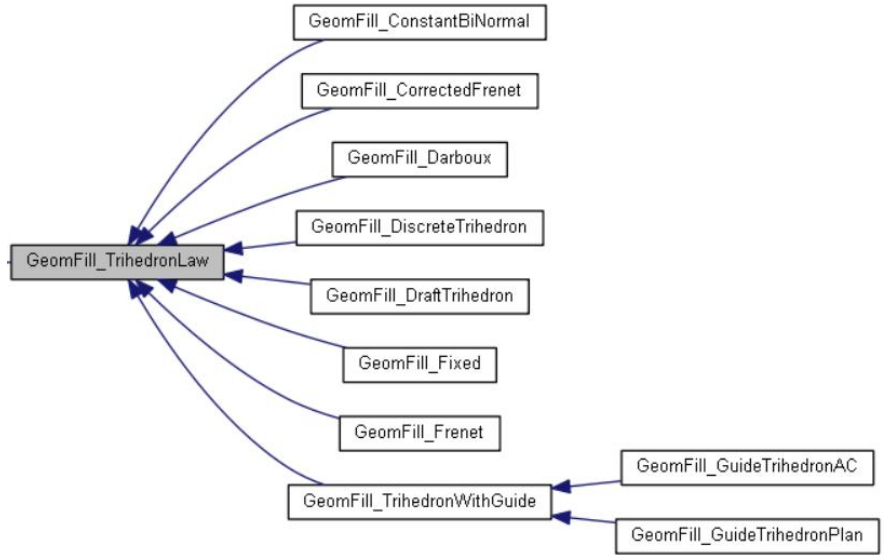
Sweep

扫掠曲面是一条截面曲线沿任意轨道曲线扫掠生成的曲面。



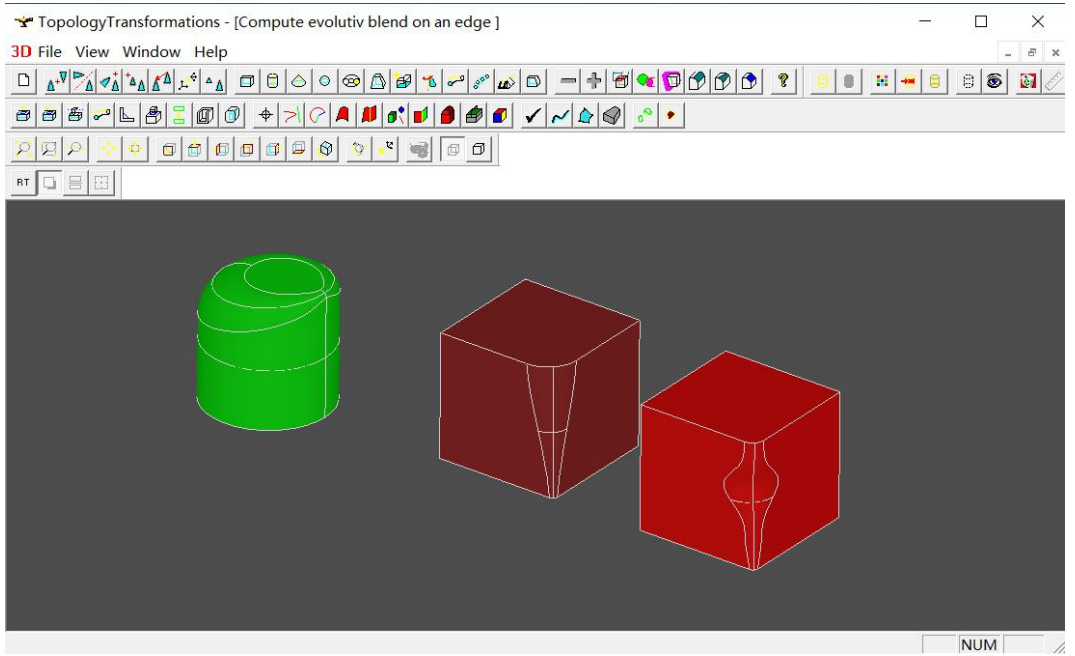

```
cylinder aCylinder 6

line aLine2d 0 0 1 1
trim aSegment aLine2d 0 2*pi

mkedge aHelixEdge aSegment
aCylinder 0 6*pi

# there is no curve 3d in the
pcurve edge.
mkedgecurve aHelixEdge 0.001

wire aHelixWire aHelixEdge

circle profile 6 0 0 0 4 1 1
mkedge profile profile
wire profile profile
mkplane profile profile

pipe aSpring aHelixWire profile
```

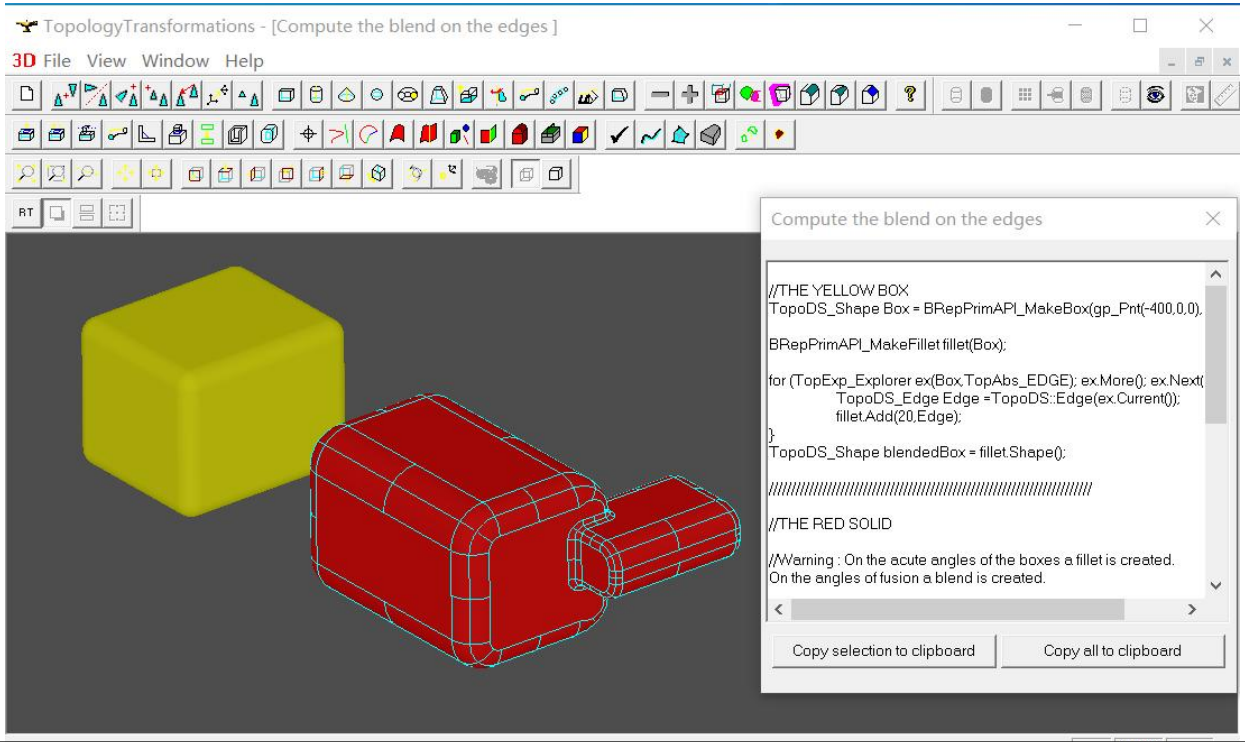


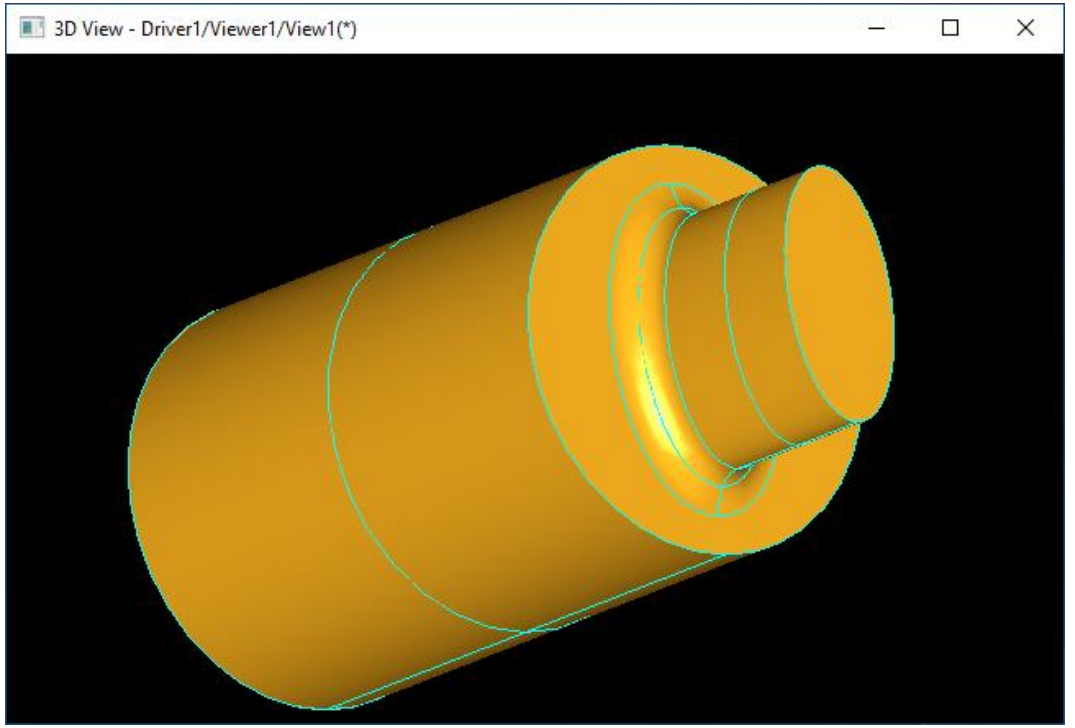


Fillet

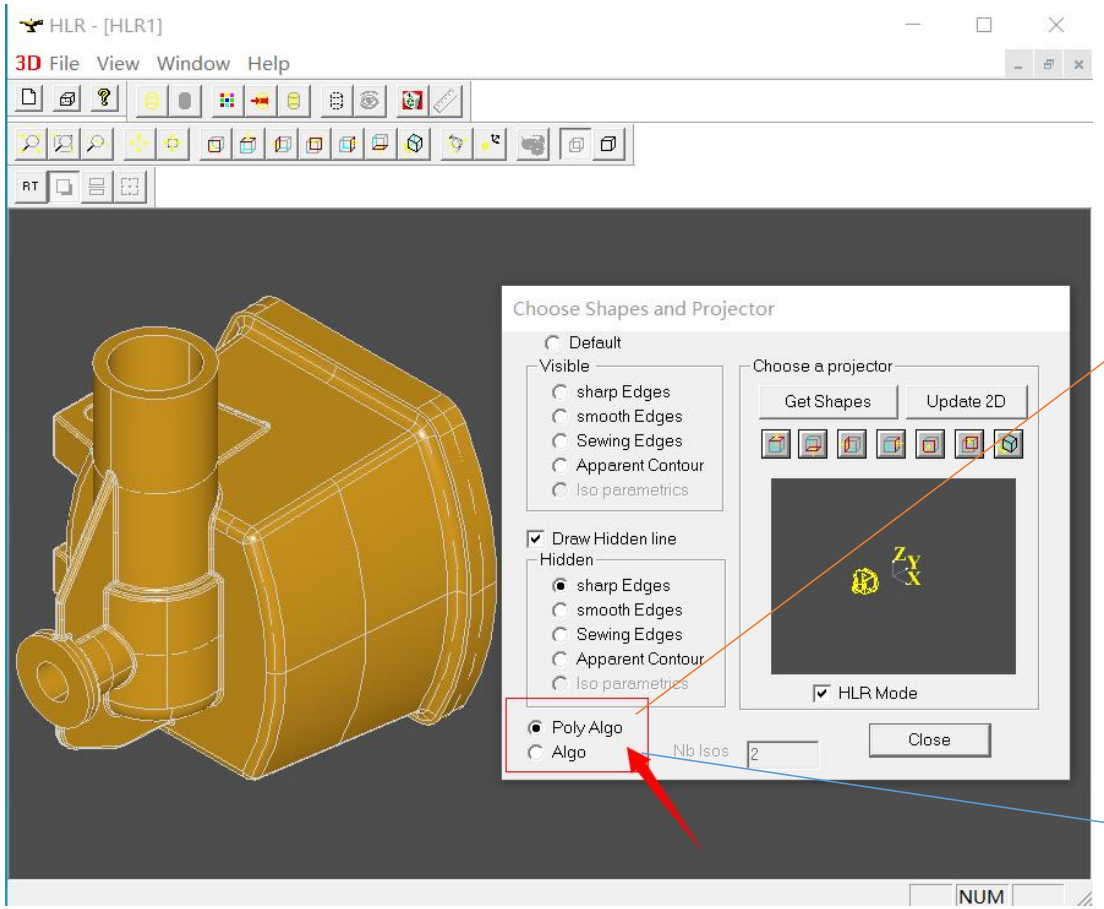
- Fillet
- Chamfer
- Rolling Ball

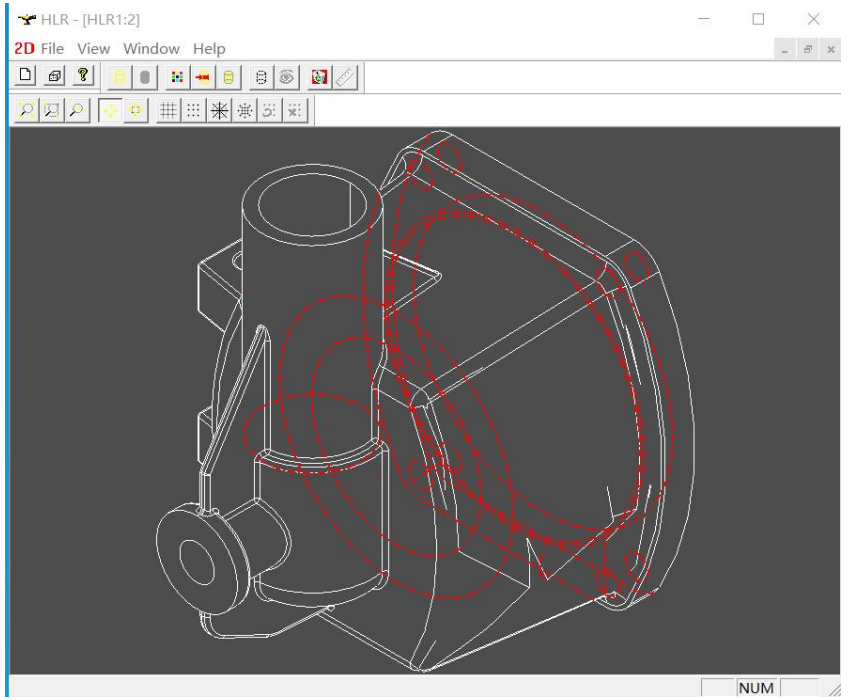


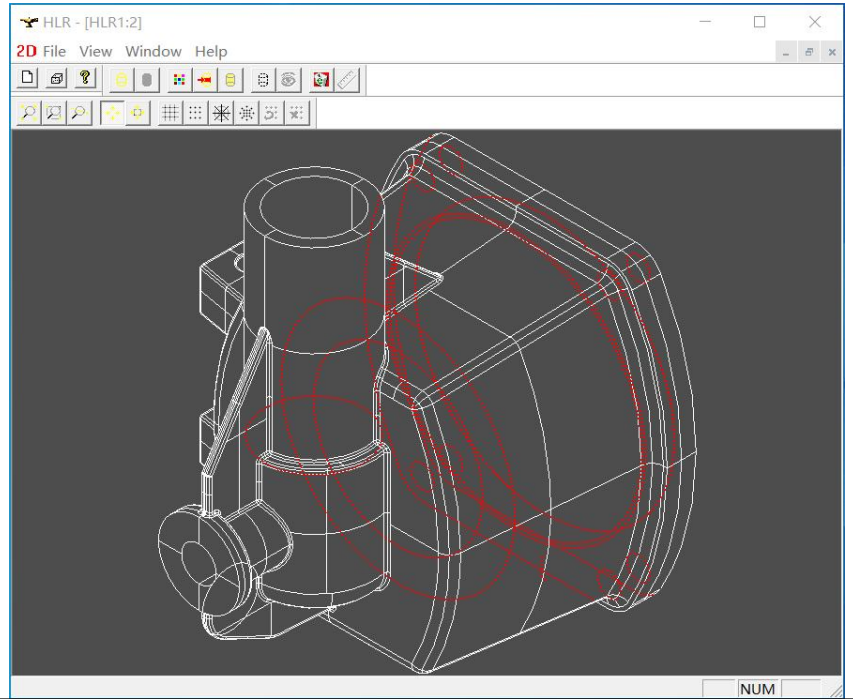




HLR


HLR - [HLR1]
3D File View Window Help
RT
Choose Shapes and Projector
Default
Visible
sharp Edges
smooth Edges
Sewing Edges
Apparent Contour
Iso parametrics
Draw Hidden line
Hidden
sharp Edges
smooth Edges
Sewing Edges
Apparent Contour
Iso parametrics
Poly Algo
Algo
Nb Isos
2
Choose a projector
Get Shapes
Update 2D
HLR Mode
Close
NUM

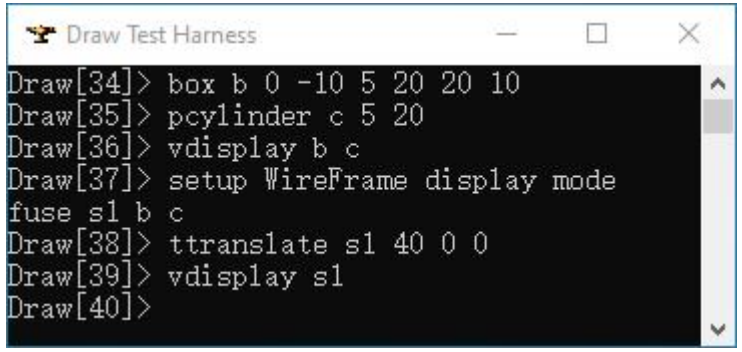

HLR - [HLR1:2]
2D File View Window Help
NUM

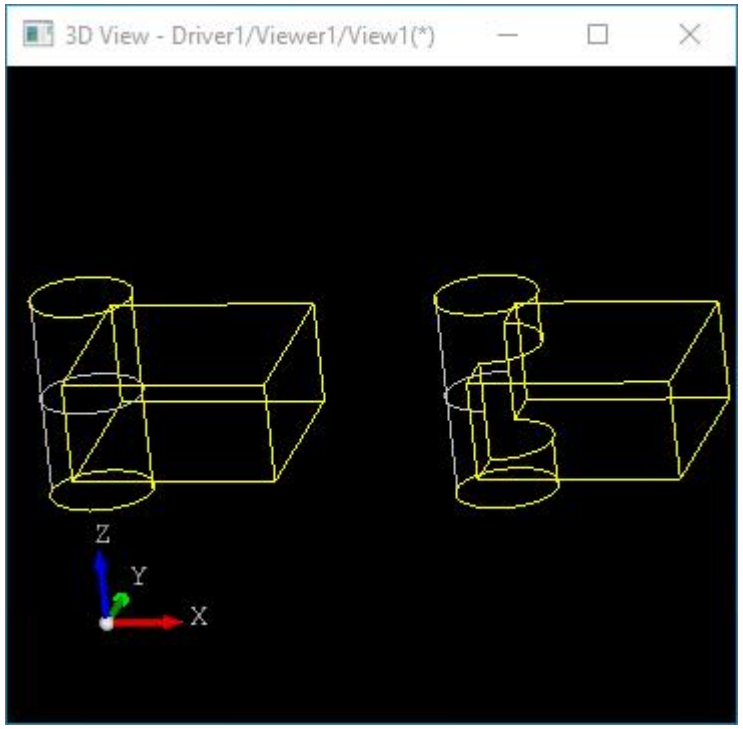

HLR - [HLR1:2]
2D File View Window Help
NUM

ModelingAlgorithms

- Boolean Operation - Fuse

```
TopoDS_Shape aShape =
BRepAlgoAPI_Fuse(S1, S2);
```

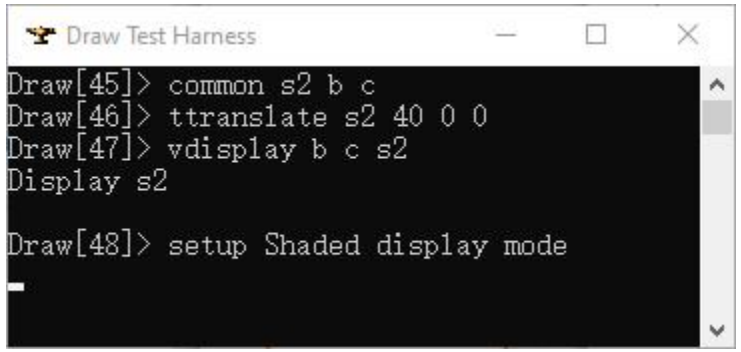


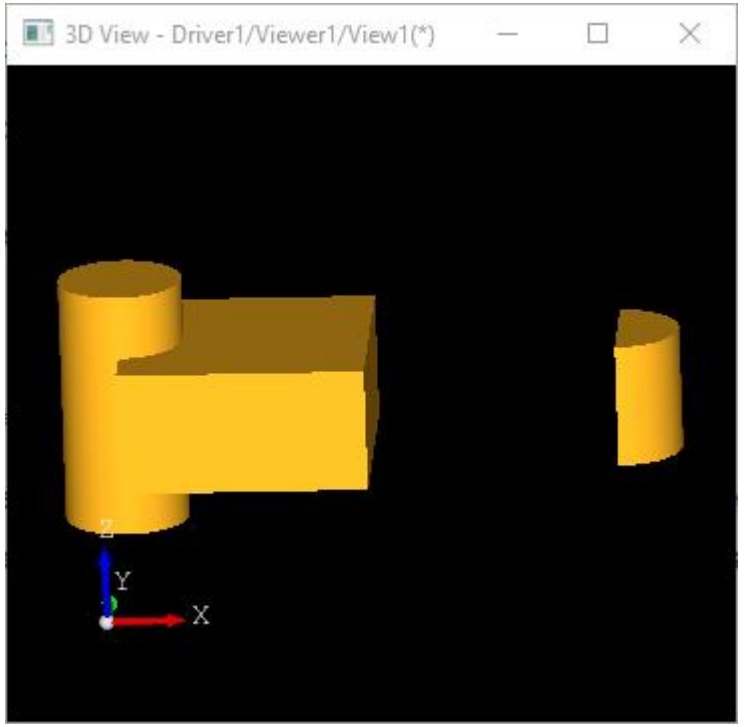


ModelingAlgorithms

- Boolean Operation - Common

```
TopoDS_Shape aShape =
BRepAlgoAPI_Common(S1, S2);
```

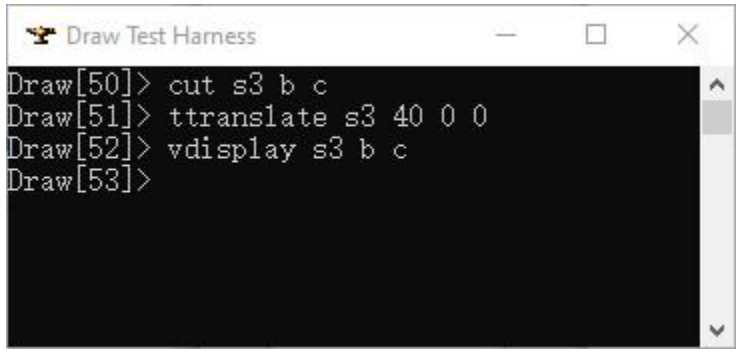


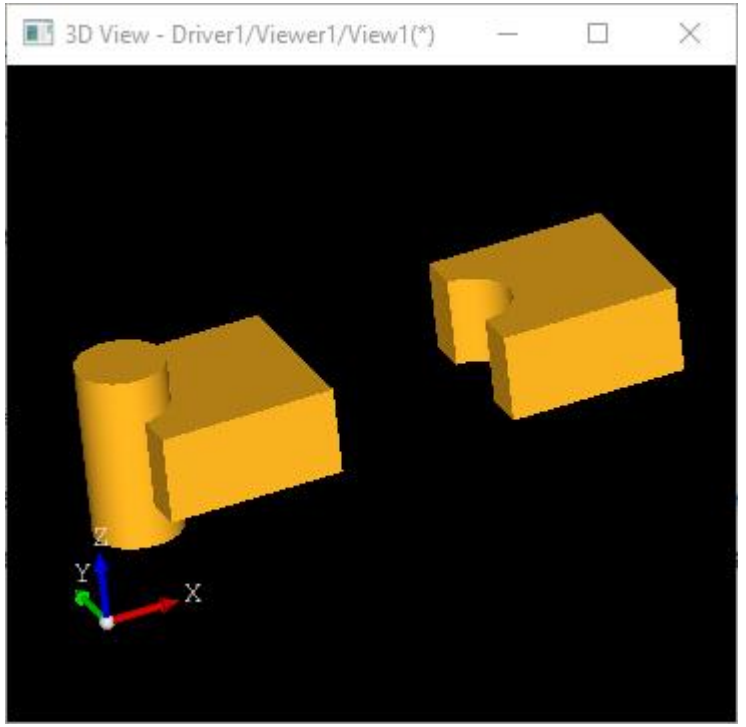


ModelingAlgorithms

- Boolean Operation - Cut

```
TopoDS_Shape aShape =
BRepAlgoAPI_Cut(S1, S2);
```

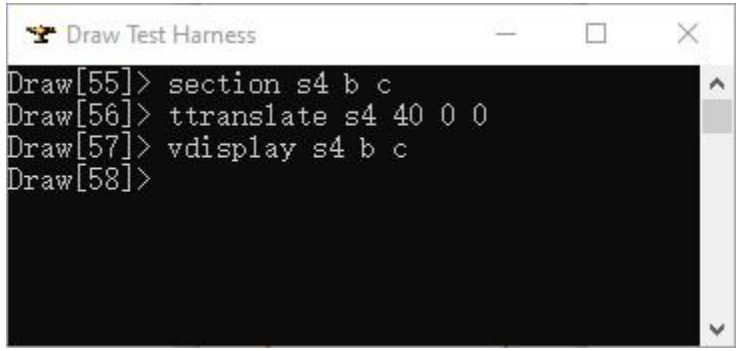


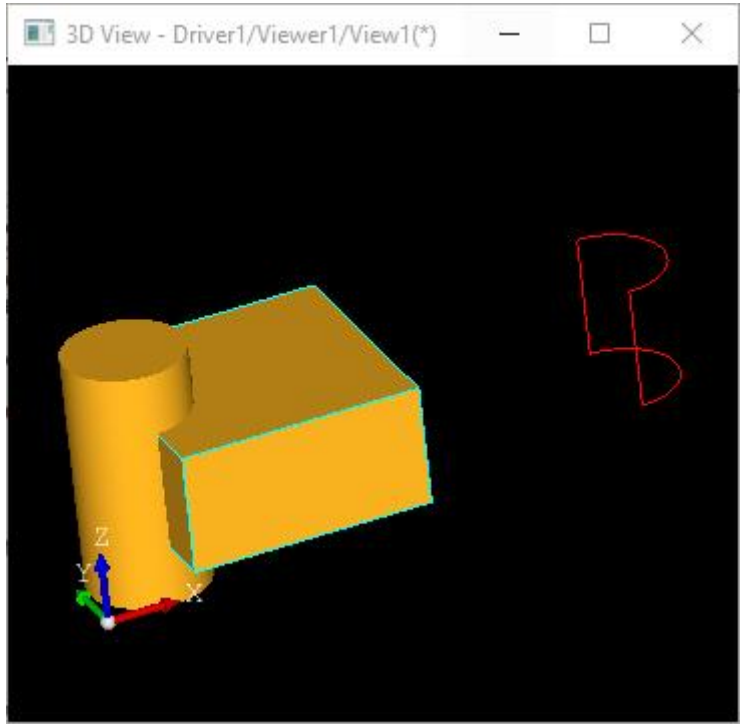


ModelingAlgorithms

- Boolean Operation - Section

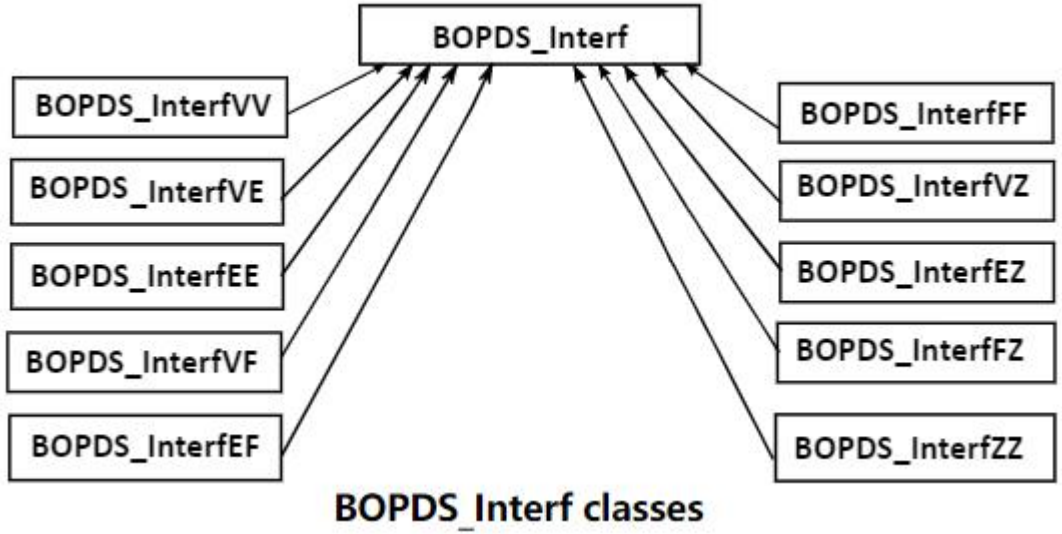
```
TopoDS_Shape aShape =
BRepAlgoAPI_Section(S1, S2);
```



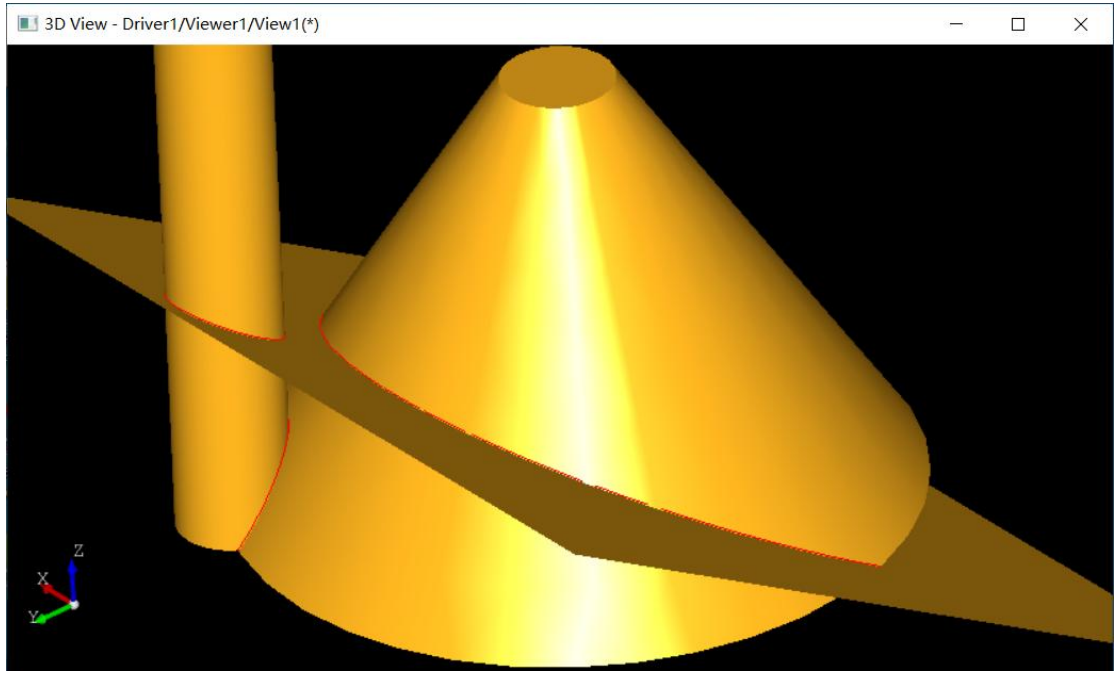


Boolean Operations algorithm

- Interferences



BOPDS_Interf classes



Boolean Operations algorithm

- Analytic Geometric Intersection

对于有隐式方程的几何曲线、曲面，通过将参数方程C(u)代入隐式方程F(x,y,z)的方式，得到相交方程，相交问题变成方程的求解。对应的包有IntAna2d, IntAna。

3. Intersection between Circle and Conic

当对二维圆和圆锥曲线进行求交时，先得到圆的半径和圆锥曲线的一般式方程。将圆用参数方程表示并代入圆锥曲线的一般式方程中得到:

$$\begin{cases} x = R \cdot \cos\alpha \\ y = R \cdot \sin\alpha \end{cases}$$

$$Ax^2 + By^2 + 2Cxy + 2Dx + 2Ey + F = 0$$

$$A(R \cdot \cos\alpha)^2 + B(R \cdot \sin\alpha)^2 + 2C(R \cdot \cos\alpha)(R \cdot \sin\alpha) + 2DR \cdot \cos\alpha + 2ER \cdot \sin\alpha + F = 0$$

$$A(R \cdot \cos\alpha)^2 + BR^2(1 - \cos^2\alpha) + 2C(R \cdot \cos\alpha)(R \cdot \sin\alpha) + 2DR \cdot \cos\alpha + 2ER \cdot \sin\alpha + F = 0$$

$$(AR^2 - BR^2)\cos^2\alpha + 2CR^2 \cos\alpha \cdot \sin\alpha + 2DR \cdot \cos\alpha + 2ER \cdot \sin\alpha + F + BR^2 = 0$$

Boolean Operations algorithm

- Analytic Geometric Intersection

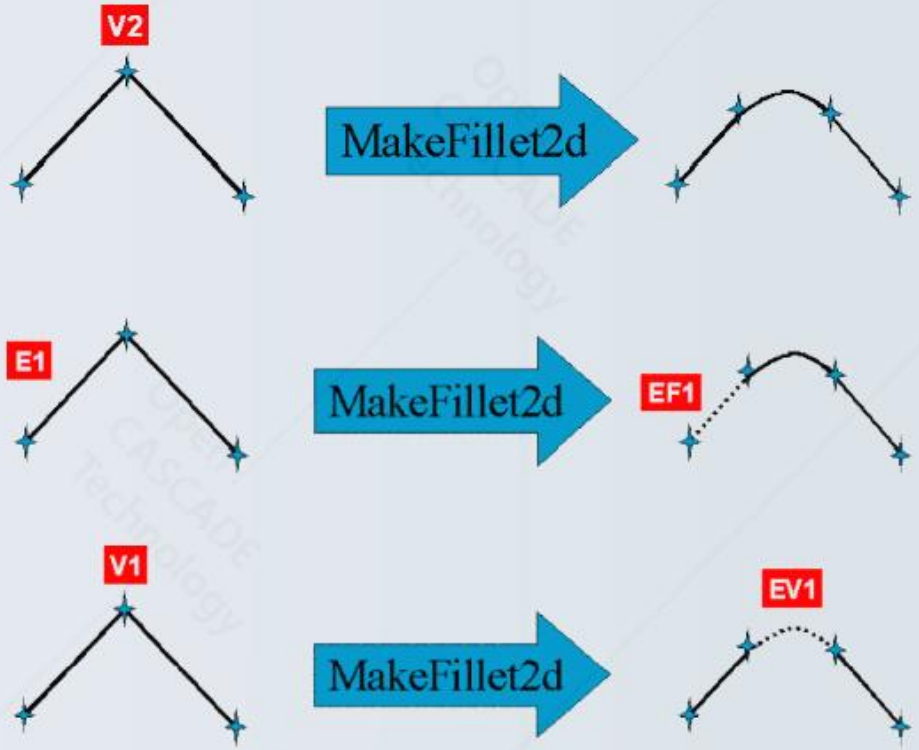
二次曲面与二次曲面的求交计算，也可以把二次曲面的参数形式代入二次曲面的隐式方程。对应的类是IntAna_QuadQuad，其中对圆柱与二次曲面的求交计算源码注释如下图所示：

```
//-------------------------------------------------------------------
//-- Coefficients de la quadrique :
//--------  2       2       2
//-- Qxx x + Qyy y + Qzz z + 2 ( Qxy x y + Qxz x z + Qyz y z )
//---+ 2 ( Qx x + Qy y + Qz z ) + Q1
//-------------------------------------------------------------------
Quad.Coefficients(Qxx,Qyy,Qzz,Qxy,Qxz,Qyz,Qx,Qy,Qz,Q1);

//-------------------------------------------------------------------
//--- Ecriture des Coefficients de la Quadrique dans le repere liee
//--- au Cylindre
//-------------------  Cyl.Position() -> Ax2
//-------------------------------------------------------------------
Quad.NewCoefficients(Qxx,Qyy,Qzz,Qxy,Qxz,Qyz,Qx,Qy,Qz,Q1,Cyl.Position());

//-------------------------------------------------------------------
//-- Parametrage du Cylindre Cyl :
//--------  X = Rcyl * Cos(Theta)
//--------  Y = Rcyl * Sin(Theta)
//--------  Z = Z
//-------------------------------------------------------------------
//-- Donne une Equation de la forme :
//-----  F(Sin(Theta),Cos(Theta),ZCyl) = 0
//-----  (Equation du second degre en ZCyl)
//-------  ZCyl**2  CoeffZ2(Theta) + ZCyl CoeffZ1(Theta) + CoeffZ0(Theta)
//-------------------------------------------------------------------
//-- CoeffZ0 = Q1 + 2*Qx*RCyl*Cos[Theta] + Qxx*RCyl^2*Cos[Theta]^2
//-------------  2*RCyl*Sin[Theta]* ( Qy + Qxy*RCyl*Cos[Theta]);
```

Modeling API: history of modifications

Sometimes, built-in operations are unable to solve a modeling problem. A custom modeling pipeline is used in that case. The OCCT does not have persistent indexing; topological items may change unpredictably after an operation. History support allows overcoming this issue. Modeling algorithms have three methods that allow to know the modification status of an initial shape 'S' after a topological operation:

- `IsDeleted(S)` tells if the shape 'S' has been deleted by the algorithm.
- `Modified(S)` lists shapes that represent the modified state of the shape 'S' (such shapes have the same underlying geometry with 'S').
- `Generated(S)` lists shapes generated by the algorithm from the shape 'S' (such shapes have new underlying geometry not existing in 'S')

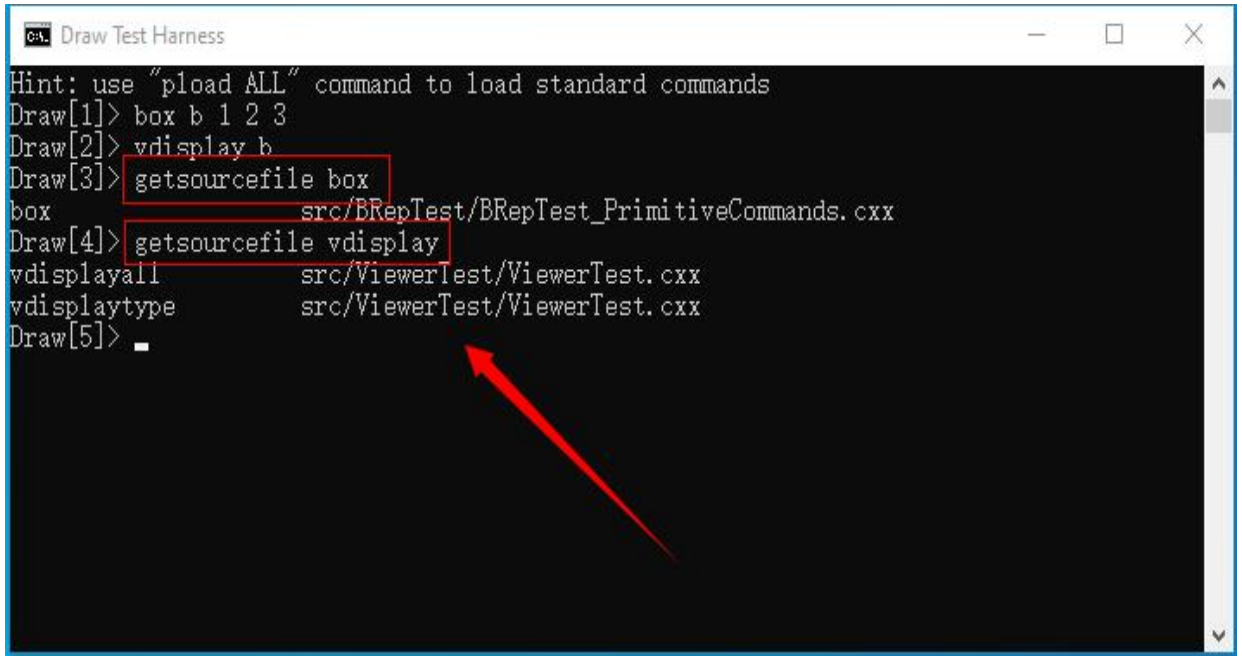


自动化测试

Draw Test Harness

Draw Test Harness

- 自定义Tcl命令
- 根据命令查找源码实现




```c
#include <tcl.h>
#include <stdlib.h>
#include <string.h>

#pragma comment(lib, "tcl85.lib")

int EqualCmd(ClientData clientData, Tcl_Interp* interp, int objc, Tcl_Obj *CONST objv[])
{
    if (objc != 3)
    {
        Tcl_WrongNumArgs(interp, 1, objv, "string1 string2");
        return TCL_ERROR;
    }
    Tcl_Obj* result = NULL;
    char* arg1 = Tcl_GetString(objv[1]);
    char* arg2 = Tcl_GetString(objv[2]);
    if (strcmp(arg1, arg2) == 0)
    {
        result = Tcl_NewBooleanObj(1);
    }
    else
    {
        result = Tcl_NewBooleanObj(0);
    }
    Tcl_SetObjResult(interp, result);
    return TCL_OK;
}

int Tcl_AppInit(Tcl_Interp* interp)
{
    // Initialize packages Tcl_Init sets up the Tcl library facility.
    if (Tcl_Init(interp) == TCL_ERROR)
    {
        return TCL_ERROR;
    }
    // Register application-specific commands.
    Tcl_CreateObjCommand(interp, "equalcmd", EqualCmd, NULL, NULL);
    return TCL_OK;
}

int main(int argc, char* argv[])
{
    Tcl_Main(argc, argv, Tcl_AppInit);
    return 0;
}

```

Automated Testing System

- OCCT基于Draw Test Harness提供了一套自动化测试系统。当OCCT代码有修改时，需要进行自动化测试，来保证软件质量。

Download Open CASCADE Technology source package, tgz archive:

- **opencascade-7.6.0.tgz** (230 101 903 bytes)

Download Open CASCADE Technology testing dataset:

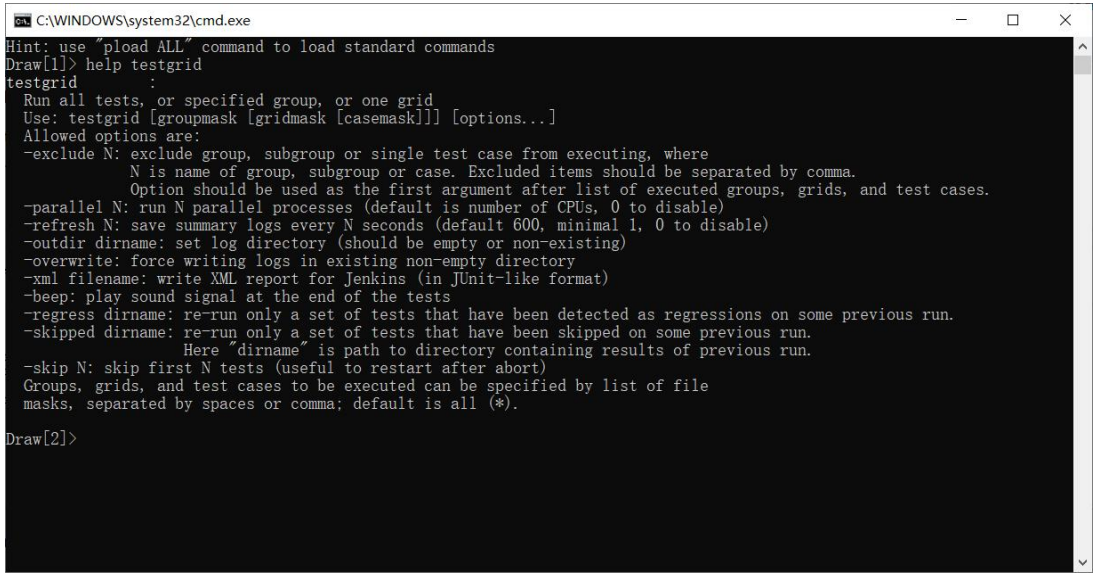
- **opencascade-dataset-7.6.0** (67 976 419 bytes)

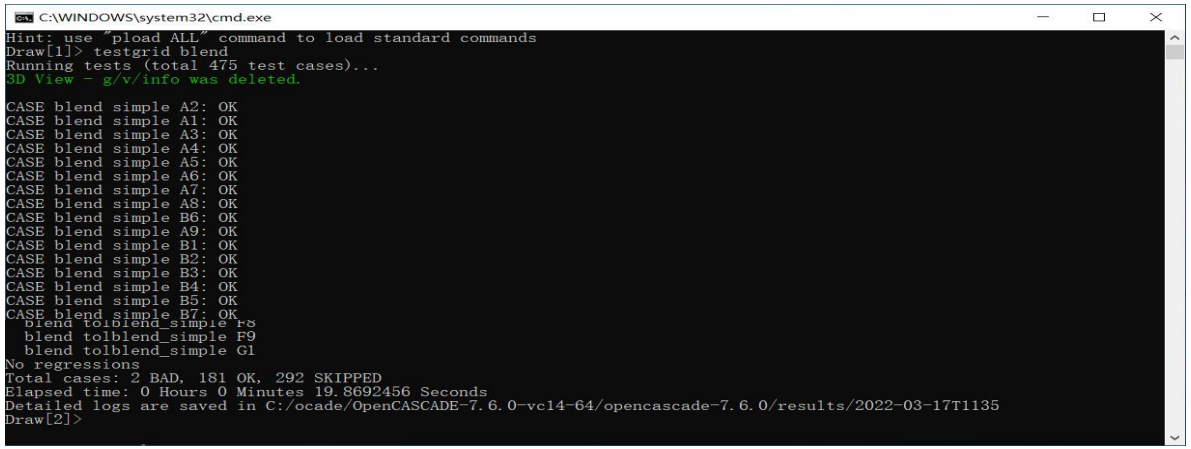
Automated Testing System

- 设置测试数据的环境变量

set env(CSF_TestDataPath) $env(CSF_TestDataPath)\;d:/occt/test-data

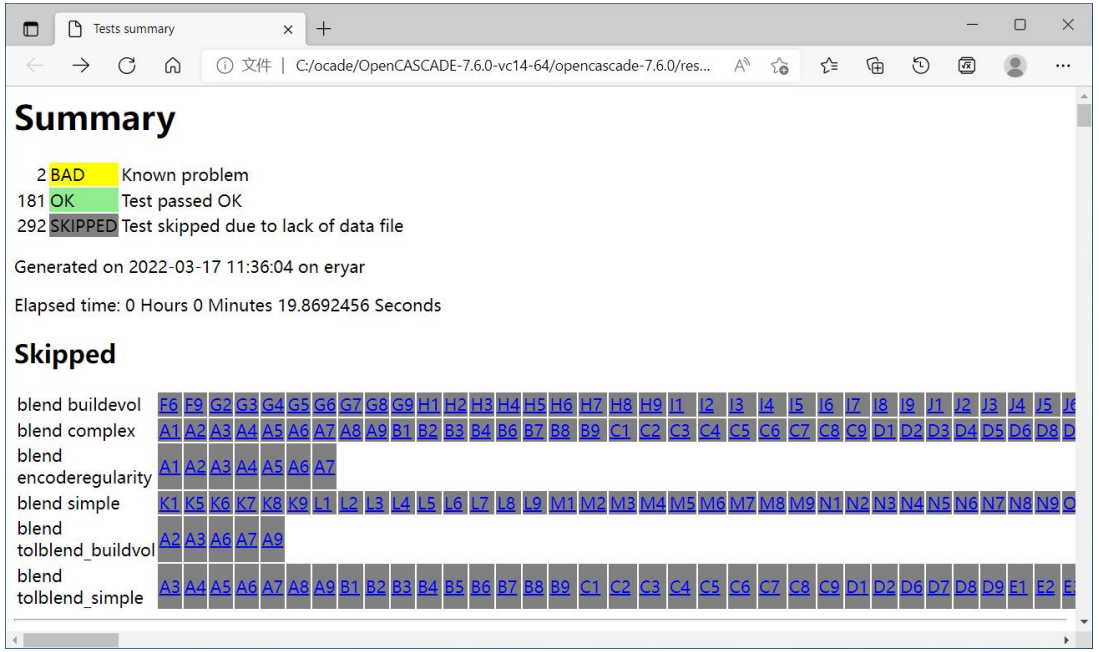
- 运行所有测试: testgrid
- 运行指定测试用例： testgrid bugs caf moddata* xde

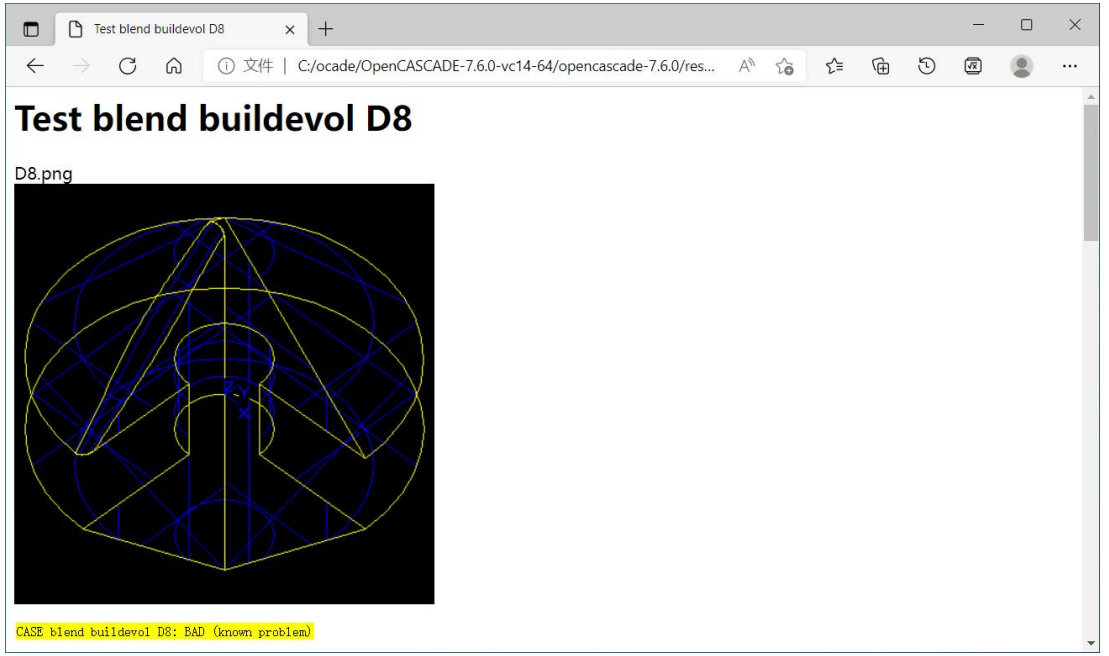




Automated Testing System

- 测试报告





案例实践

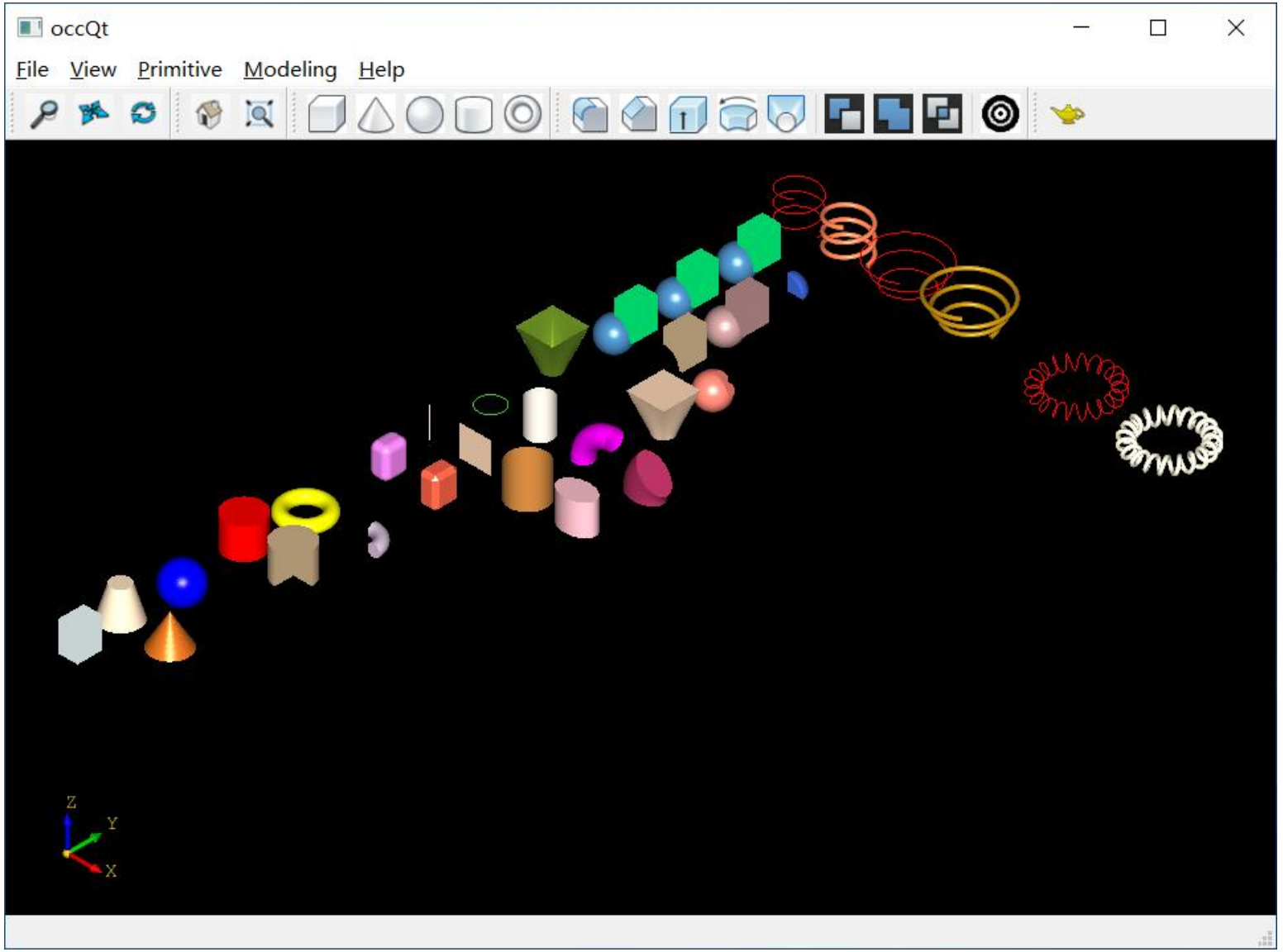
Case Practice

案例实践

- occQt

一个最简的occt在Qt中应用，下载地址：
https://github.com/eryar/occQt/

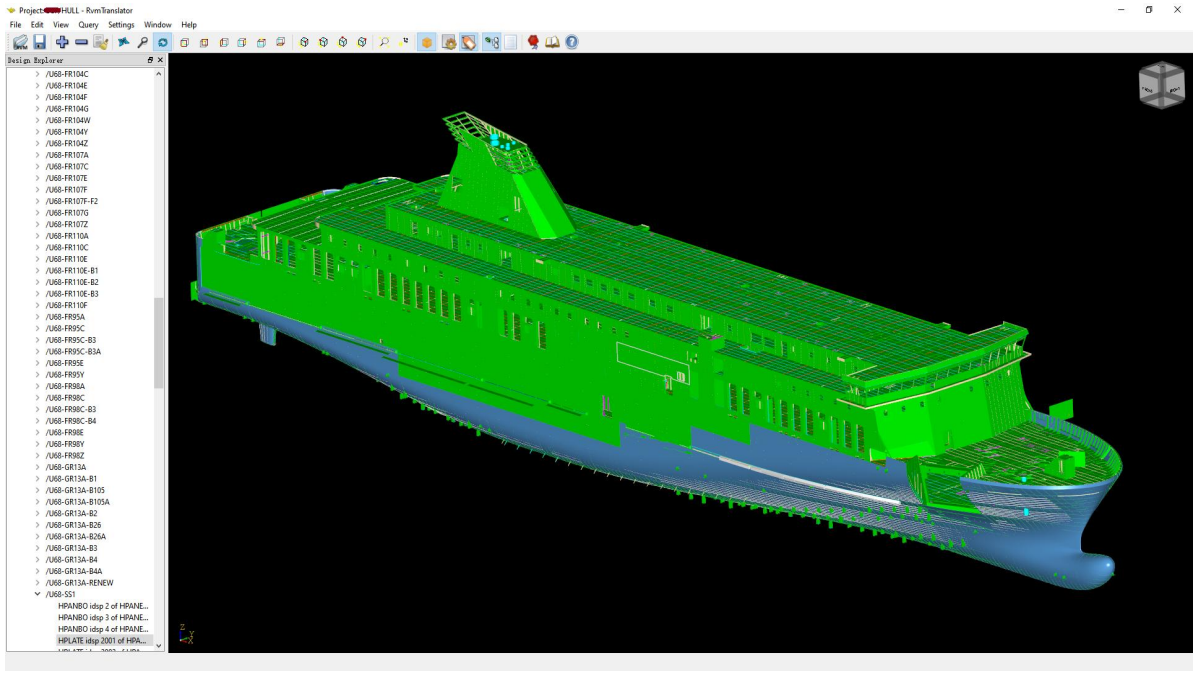
还可以参考 KGV 的：
https://github.com/gkv311/occt-samples-qopenglwidget

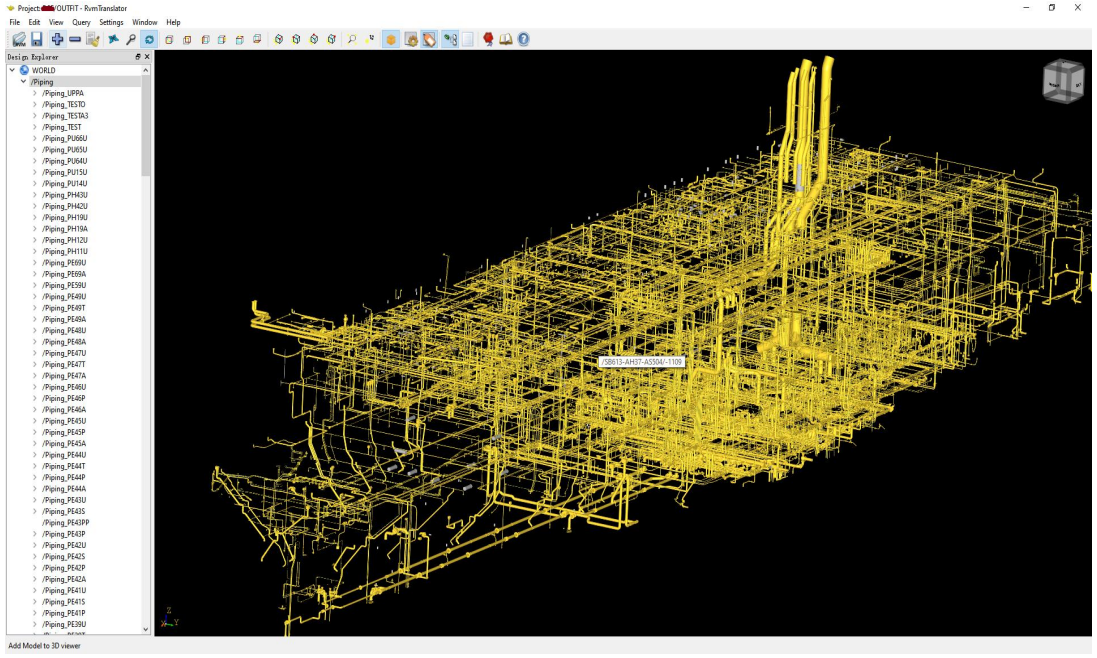


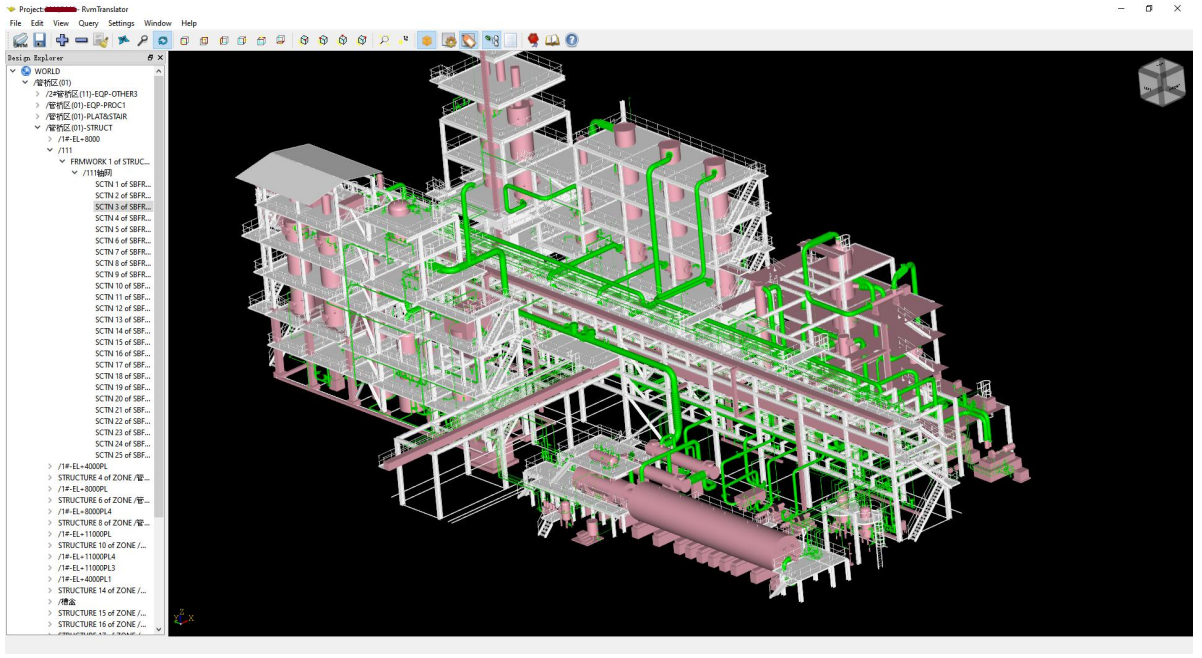
案例实践

- RvmTranslator

使用数据交换模块将船舶、管道、工厂模型进行数字化交付。

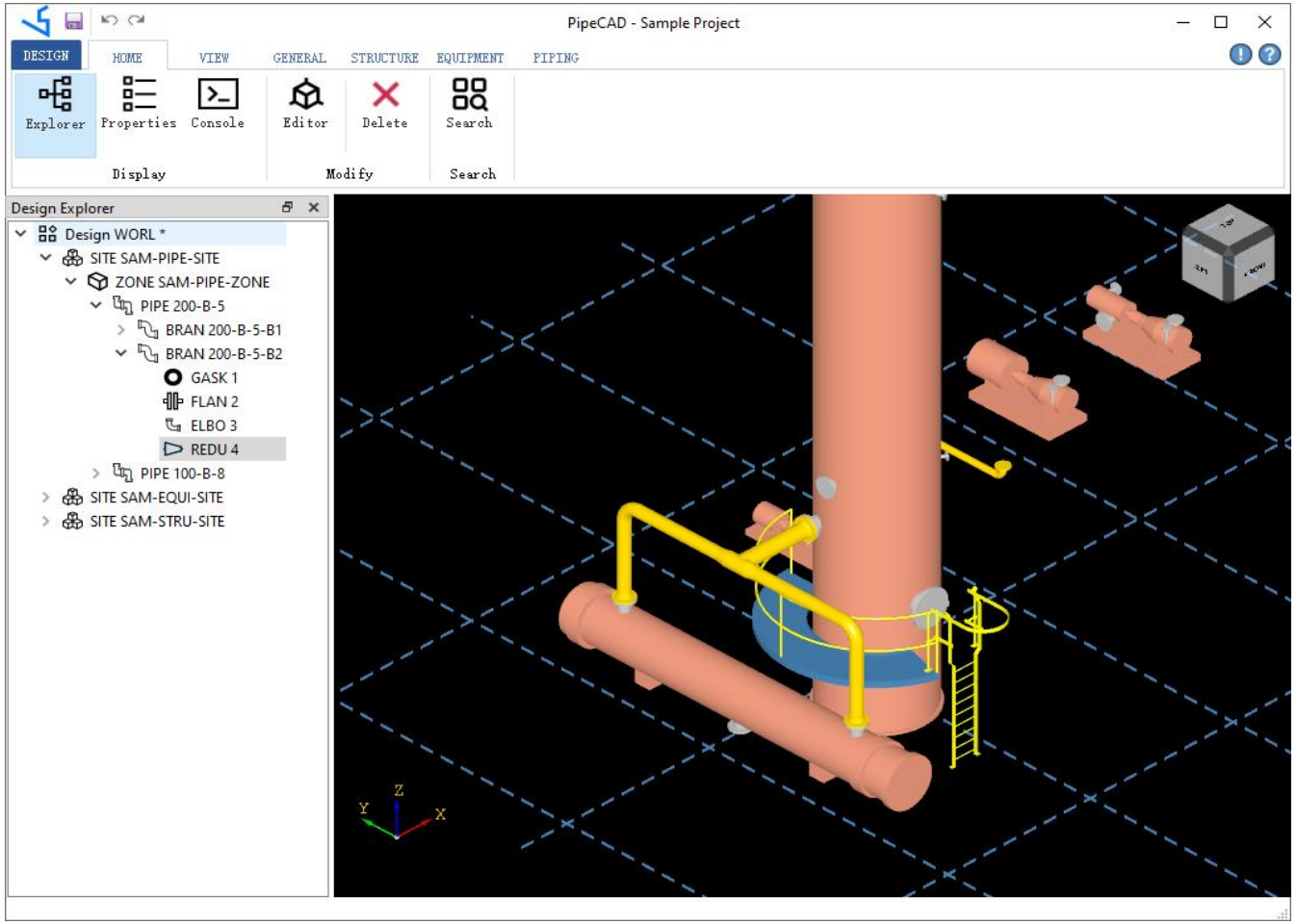






案例实践

- PipeCAD三维管道设计软件。
  - 轴网建模
  - 结构建模
  - 设备建模
  - 管道建模
  - 管道ISO图
  - 管道平面布置图



The End

Thank You