Open CASCADE Foundation Classes

Open CASCADE 基础类

一、简介

1. 基础类概述 ***Foundation Classes Overview***

本文将对 ***Open CASCADE*** 中的基础类进行介绍及如何使用基础类。更多信息可访问其官网：***www.opencascade.org/support/training/***。基础类库提供了一些通用功能，如自动动态内存管理、集合、异常处理、泛型编程和 ***plug-in*** 的创建。基础类库包含如下内容：

- 基类***(Root Classes)***：主要由包 ***Standard*** 和 ***MMgt*** 实现；
  - 基本数据类型，如：***Boolean***, ***Character***, ***Integer***, ***Real***；
  - 动态创建对象的安全手柄(***Safe Handle***)技术，确保未被引用对象自动销毁；
  - 可配置的最优化内存管理提升程序性能；
  - 可扩展的运行时类型信息(***RTTI***)机制为创建复杂程序提供便利；
  - 异常的管理(***Management of Exceptions***)；
  - 封装 ***C***++的流；
- 字符串***(Strings)***：
  - 字符串的类可以处理动态大小的字符数组，***ASCII***(通常 8 位的字符)和 ***UNICODE***(16 位字符)都支持；字符串是由包 ***TCollection*** 实现；
- 集合***(Collections)***：
  - 集合就是用来处理大小不固定的数据的类。集合中的类是泛型的，即通用的。当需要使用某种集合时，只需要实例化一种元素类型。
  - 集合包含一系列通用的类，如：长度变化的数组、链表、栈、队列、哈希表；
- 标准对象的集合***(Collections of Standard Objects)***：
  包 ***TColStd*** 提供经常使用的包 ***TCollection*** 和 ***Standard*** 中类的实例。
- 矢量和矩阵***(Vectors and Matrices)***：
  这些类提供矢量和矩阵常用的数学算法和基本计算，如：加、乘、转置、变换等；
- 基本几何元素***(Primitive Geometric Types)***：
  Open CASCADE 基本几何图元类型和代数实体是 ***STEP*** 格式兼容的；功能如下：
  - 初等几何形状的描述：
    1. 点 ***Points***；
    2. 矢量 ***Vectors***；
    3. 线 ***Lines***；
    4. 圆和二次曲线 ***Circles and Conics***;
    5. 平面和初等曲面 ***Planes and elementary surfaces***；
  - 通过轴 ***axis*** 或坐标系统在空间中或平面上放置这些形状；
  - 为这些形状定义和应用几何变换：
    1. 平移 ***Translations***;
    2. 旋转 ***Rotations***;
    3. 镜像 ***Symmetries***;
    4. 缩放变形 ***Scaling transformations***;
    5. 组合变形 ***Composed transformations***;

- 代数计算工具(坐标和矩阵)；

- 通用的数学算法***(Common Math Algorithms)***：

  Open CASCADE 通用数学算法库提供最常用的数学算法的 ***C***++实现，包含：

  - 线性代数方程组的求解；
  - 单变量或多变量的函数的极小值的求解；
  - 非线性方程（组）的求根；
  - 矩阵特征值的求解；

- 异常处理***(Exceptions)***：

  提供了常用异常类的层次，都继承自 ***Failure***。异常描述的是程序运行时出现的意外情况。通常情况下，程序遇到异常是会退出的。程序对异常情况的处理就叫做异常处理。

- 量度***(Quantities)***：

  OCCT 中有各式各样的类来支持日期和时间信息，及对大多数物理量的表示，如长度(***length***)、面积(***area***)、体积(***volume***)、密度(***density***)、质量(***mass***)、重量(***weight***)、温度(***temperature***)、压力(***pressure***)等。

- 程序服务***(Application Services)***：

  基础类库也包含用于创建自定义和用户友好的 ***Open CASCADE*** 程序的几个低级功能。如：

  - 单位转换工具：提供统一的机制来处理量及其相关的物理单位，检查单位的兼容性，在不同的单位制之间转换。具体可参考包 ***UnitsAPI***。
  - 表达式的基本解释器，为自定义脚本工具的创建提供条件，表达式的通用定义等。具体可参考包 ***ExprIntrp***。
  - 处理配置资源文件的工具，具体可参考包 ***Resource***；自定义消息文件，具体参考包 ***Message***；这些工具使程序对多种语言的支持变得容易。
  - 进程标示器和用户中断接口，使低级算法以统一和便利的方式与用户交互成为可能。

以上的具体内容将会在后面详细介绍，并根据功能编写简单程序以示其用法。关于基础类更具体的描述请参考：***Foundation Classes Reference Manual***。

2． 基本概念 *Fundamental Concepts*

使用面向对象编程语言组成的系统通常是以数据类型为中心而不是以作用于数据的操作为中心。在这种情况下，一个对象就是一种数据类型的实例，每种数据类型由一个或多个其他类来实现，即使用对象组合。在 ***Open CASCADE*** 中类通常由 ***CDL***(***CASCADE Definition Language***)来定义，这就确保了所有类的实现都很相似，关于 ***CDL*** 可参考《***CDL*** 用户指南》。本章介绍的基本概念大多是由 ***CDL*** 直接支持的，不仅在基础类中使用，而且将会贯穿整个 ***Open CASCADE*** 的库中。

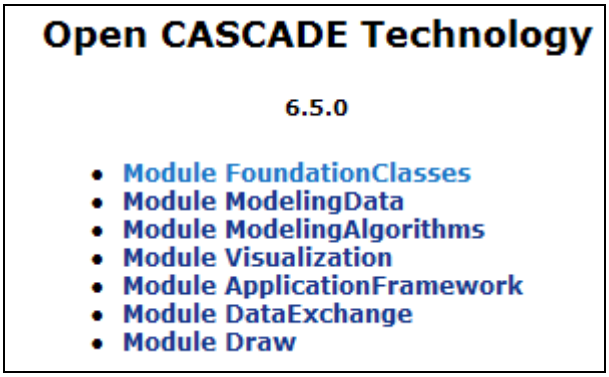
2．1 模块和工具箱 *Modules and Toolkits*

整个 ***Open CASCADE*** 库由一系列的模块(***Module***)组成。本文讲述的正是这些模块中的第一个：基础类(***Foundation Classes***)，这个模块提供的是一些基本功能，也被其他所有模块所用。每个模块包含一个或几个工具箱(***Toolkits***)。从物理上来讲，一个工具箱就是一个共享的库(***shared library***)。一个工具箱由一个或几个包(***package***)生成。

即在 ***Open CASCADE*** 中，模块由工具箱组成，工具箱由包组成，每个包就是由相应的类来实现相应的功能。理解它的组织结构，对编程还是很大帮助的，不会对其庞大的源文件

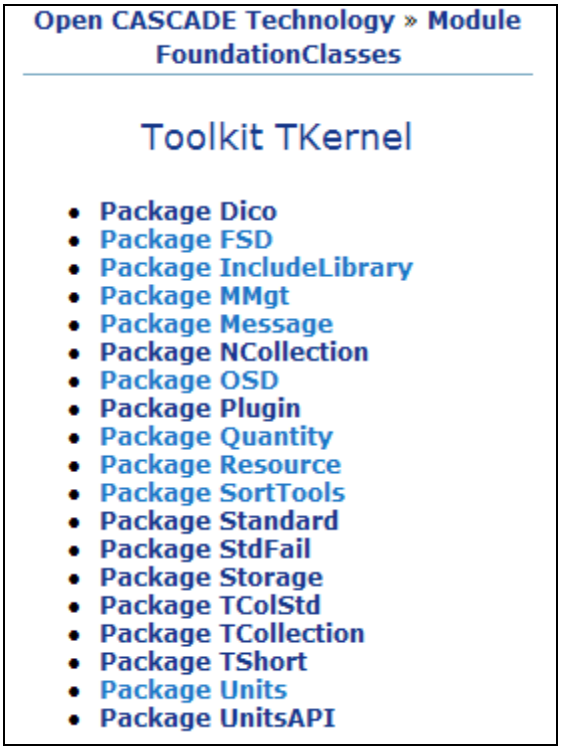
所吓到，可以有的放矢。

打开 ***Open CASCADE Reference Documentation***，可以看到 ***OCCT*** 包含以下几个模块：



打开基础库模块，其由以下工具箱组成，如图所示：

打开 ***Toolkit TKernel*** 工具箱，其由以下包组成，如图所示：



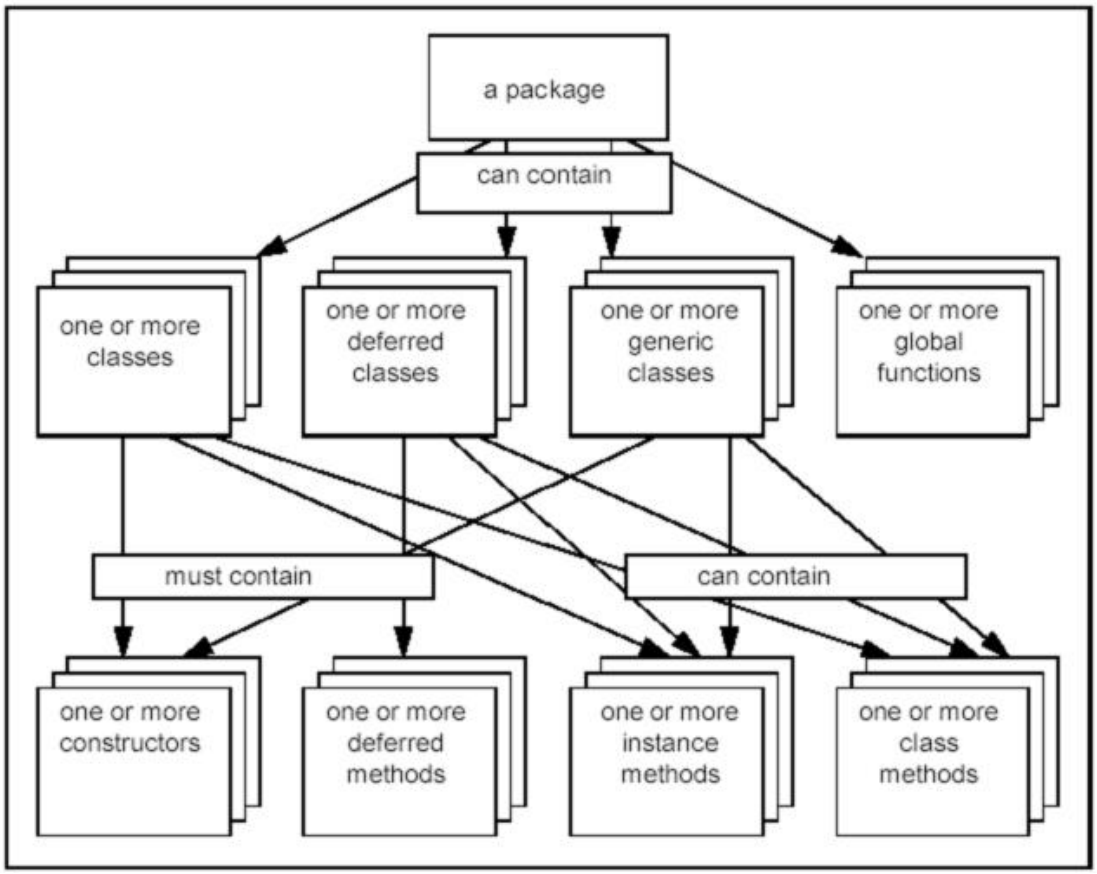
这种组织结构在其文档中表现得已经很清晰了。经常查看此帮助文档，一般的内容基本可以查到。

2．2 包 ***Packages***

包是由在语义上相近的一些类组成。如几何包(***Geometry package***)可能包含点、线、圆的类。包也可包含枚举类型、异常和包的方法函数。实际上，类名前会加上包的名字，如：***Geom_Circle***。包中描述的数据可能包含一些以下数据类型：

- ***Enumerations***;
- ***Object classes***;
- ***Exceptions***;
- ***Pointers to other object classes***;

包中两种数据类型不能是相同的名字。包的组成如下图所示：



方法(***Methods***)既可以是函数(***Functions***)，也可以是过程(***Procedures***)。函数返回一个对象，而过程只处理传递的参数。方法分以下三种情况：

- 对象构造函数(***Object Constructor***)：创建类的一个实例。类可能有一个或多个构造函数，即通过参数的不同来重载了构造函数。
- 实例方法(***Instance method***)：操作作用于它属于的实例；即类中普通函数；
- 类方法(***Class Method***)：可以直接使用类名来操作的方法，即类中的静态函数；

2．3 类 ***Classes***

面向对象的软件开发最基础的部件就是类。类是数据类型(***Data Type***)的一种实现。通过函数定义其行为(***Behavior***)，通过类的数据来定义其表现(***Representation***)。类分为以下三类：

- 普通类 ***Ordinary Classes***;
- 滞后类 ***Deferred Classes***;
- 泛型类 ***Generic Classes***;

滞后类(***Deferred Classes***)不能被实例化，即是含有纯虚函数的抽象类。使用这种类的目

的是使派生自这个类的所子类共享行为并由子类来实现具体的行为。***C***++中与 ***CDL*** 的滞后类(***Deferred Class***)等效的类就是抽象类(***Abstract Class***)。

泛型类(***Generic Class***)为操作其他数据类型提供一些基本函数。对泛型类的实例化需要将数据类型作为参数传递之。***C***++中与 ***CDL*** 中泛型类功能相同的就是模板(***Template***)。

2．4 泛型 *Genericity*

泛型的实现分两步。首先，声明泛型类创建模型；然后，通过指定泛型的类型来实例化类。

- 声明泛型类(***Declaring a Generic Class***)：***Open CASCADE*** 中的泛型类与 ***C***++中带显式实例化的模板的意图类似。泛型类在当要操作不固定类型的数据元素时在 ***CDL*** 中声明，声明其作为泛型类的参数。可以对这些泛型类作些约束，使其成为已经定义的类的子类。定义泛型类并没创建 ***C***++中新的类，只是为生成实际的类定义了一种模式。
- 实例化泛型类(***Instantiation of a Generic Class***)：当泛型类实例化时，参数类型会被真实的类型替换。实例化的结果是创建了一个名字任意的新的 ***C***++类。通常这样约定，实例化的类名由泛型类名和真实类型名组成。如：

Example:

```
class Array1OfReal instantiates Array1 from TCollection (Real);
```

声明位于包 ***TColStd*** 中一个 ***CDL*** 文件定义了一个新的 ***C***++类 ***TColStd_Array***1***OfReal*** 作为泛型类 ***TCollection_Array***1 将实型作为其实例参数的类。泛型类也不允许继承。泛型类也可作为滞后类。泛型类也可将滞后类作为其实例化的参数。

- 泛型类的嵌套(***Nested Generic Classes***)：

Example:

```
class MapOfReal instantiates Map from TCollection (Real,MapRealHasher);
```

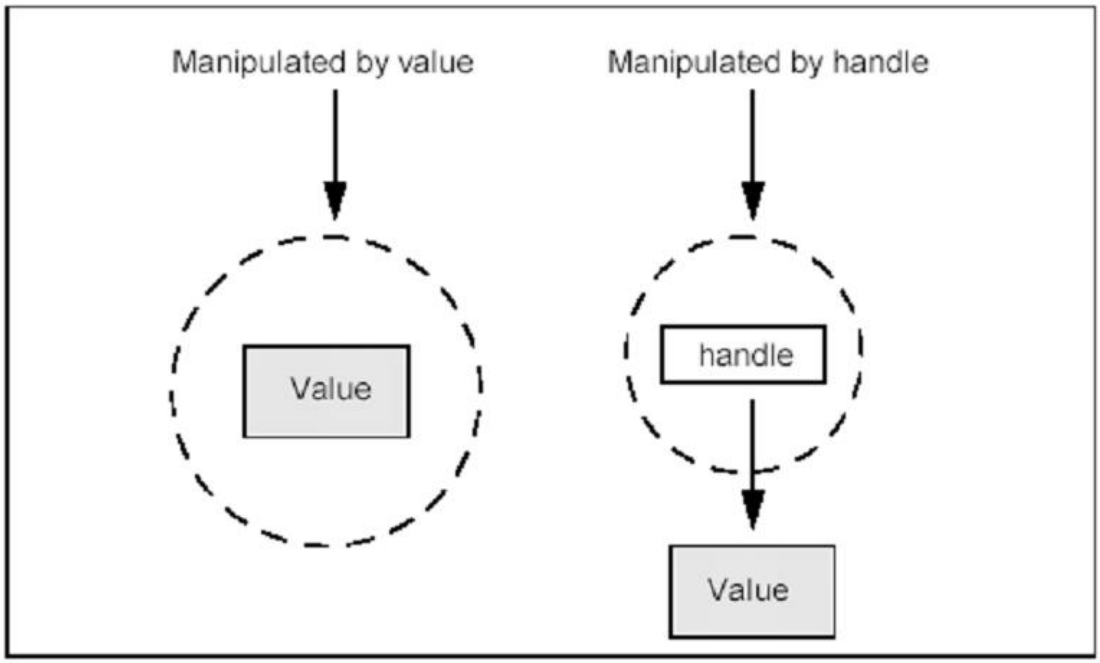
2．5 继承 *Inheritance*

继承的目的是为了减轻开发的劳动量。继承机制从已经存在的类创建新的类，并保留原来类的一些特性。如：已经开发了个 ***BankAccount***，可以通过继承很快得到新的类：***SavingsAccount***，***LongTermDepositAccount***，***MoneyMarketAccount***，***RevolvingCreditAccount***，等……

2．6 数据类型的分类 ***Categories of Data Types***

Open CASCADE 中数据分为两大类：

- 由手柄(***Handle***)控制的数据类型；***Data types manipulated by Handle***(***or Reference***);
- 由值(***Value***)控制的数据类型；***Data types manipulated by Value***;



数据类型是由类来实现的。类不仅定义其数据和实例可用的方法，还对其对象的实例化提供建议。由值控制的变量类型包含自己的实例，由手柄控制的变量包含实例的引用。由值控制的变量类型是预定义的原子类型，如：布尔型、字符型、整型、实型等……由手柄控制的类，当其未赋值时其值为 ***null***，通过构造函数来实例化。如：

Example

```
Handle(myClass) m = new myClass;
```

在 ***Open CASCADE*** 中，手柄是特定的类用来安全地释放动态内存，提供引用计数机制和自动销毁未被引用的对象。这也是智能指针。

2．7 异常 ***Exceptions***

任何对象的行为都是由方法来实现的。这些方法的定义不仅包含其声明，而且还有其作用域(***domain***)的验证。***Domain*** 由异常来表示。在上出错的情况下异常就出现了。这种机制确保了软件产品的质量。

2．8 持久性和数据模式 ***Persistence and Data Schema***

数据模式就是程序用来存储数据的结构。数据模式由持久性类构成。称为持久性的对象可以被永久保存。因此，这些对象可以重复利用。为了在 ***CDL*** 中使用持久性对象，它必须继承自类 ***Standard_Persistent***，或其父类继承自 ***Standard_Persistent***。注意继承自类 ***Standard_Persistent*** 的类是由引用控制的。继承自类 ***Standard_Storable*** 的实例化对象不能自

己保存，但是可作为由类 ***Standard_Persistent*** 的一部分保存。因为派生自类 ***Standard_Storable*** 的类是由值控制的。