Topology and Geometry in OpenCascade-Vertex

eryar@163.com

摘要 Abstract：本文简要介绍了几何造型中的边界表示法（BRep），并结合程序说明 OpenCascade 中的边界表示的具体实现，即拓朴与几何的联系。对具有几何信息的拓朴结构顶点（vertex）、边（edge）、面（face）进行了详细说明。本文只对顶点数据进行说明。

关键字 Key Words：OpenCascade、BRep、Topology、Geometry

一、引言 Introduction

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。

下图所示为由一条边连接的两个面组成的壳（shell）：

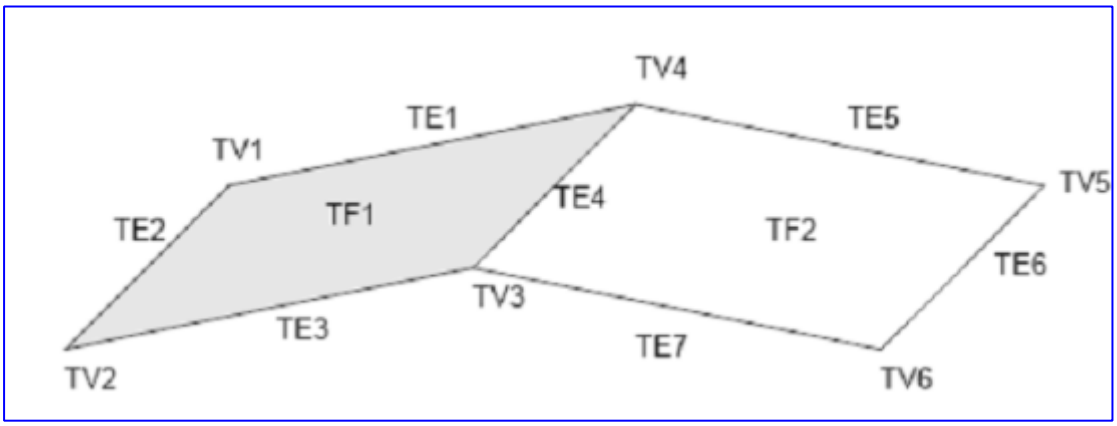

Figure 1.1 Structure of a shell formed from two faces

上图所示的形状表示为 TS，面 TF1 和 TF2，有七条边 TE1~TE7 和六个顶点 TV1~TV6。环 TW1 引用边 TE1~TE4；环 TW2 引用 TE4~TE7 。边引用的顶点如下：TE1（TV1，TV4），TE2（TV1，TV2），TE3（TV2，TV3），TE4（TV3，TV4），TE5（TV4，TV5），TE6（TV5，TV6），TE7（TV3，TV6）。

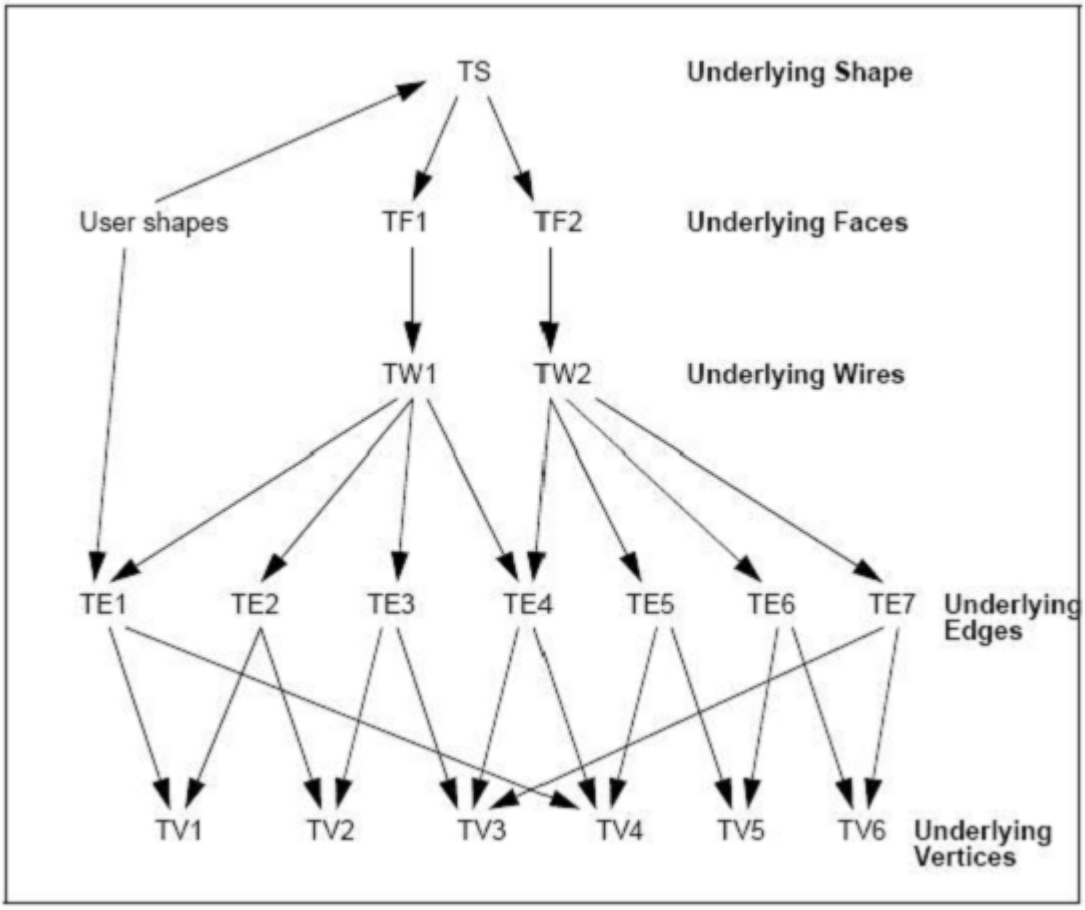

Figure 1.2 Data structure of the shell formed from two faces connected at an edge

注：OpenCascade 中的这个数据结构中不包含“回溯引用（back references）”，即所有的引用只从复杂形状到简单形状。（Note that this data structure does not contain any “back references”. All references go from more comples underlying shapes to less complex ones.）有点有向图的意思。

二、OpenCascade 中的边界表示 BRep in OpenCascade

2.1 拓朴结构 TopoDS_Shape data structure

OpenCascade 中的拓朴（topology）是根据 STEP 标准 ISO-10303-42 设计的。也许读一下这个标准中的有关概念还是很有帮助的。STEP ISO-10303-42 的相关资源：

http://www.steptools.com/support/stdev_docs/express/step_irs/index.html

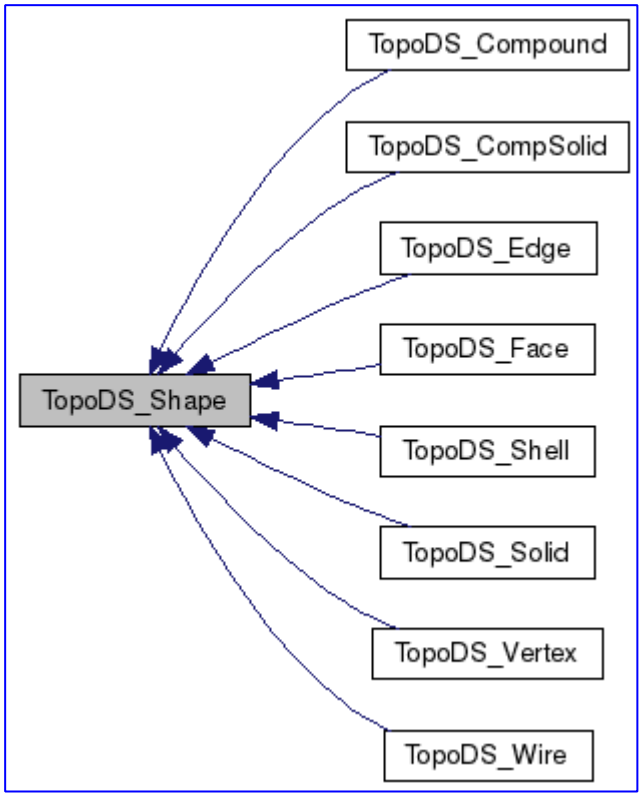

Figure 2.1 Topology data structure in OpenCascade

TopoDS_Shape 由值控制，包含三个成员变量：myLocation、myOrient、myTShape。

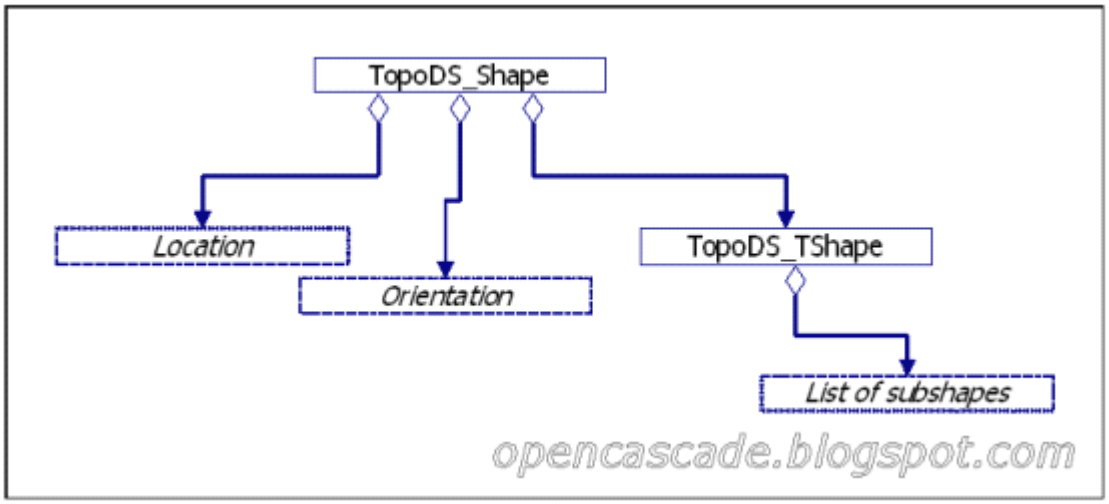

Figure 2.2 TopoDS_Shape member fields

2.2 拓朴与几何的联系 Connection with Geometry

现在我们来考虑一下拓朴结构与几何的关系。通过继承 TopoDS 包中的抽象的拓朴类实现了边界表示模型。如下图所示：

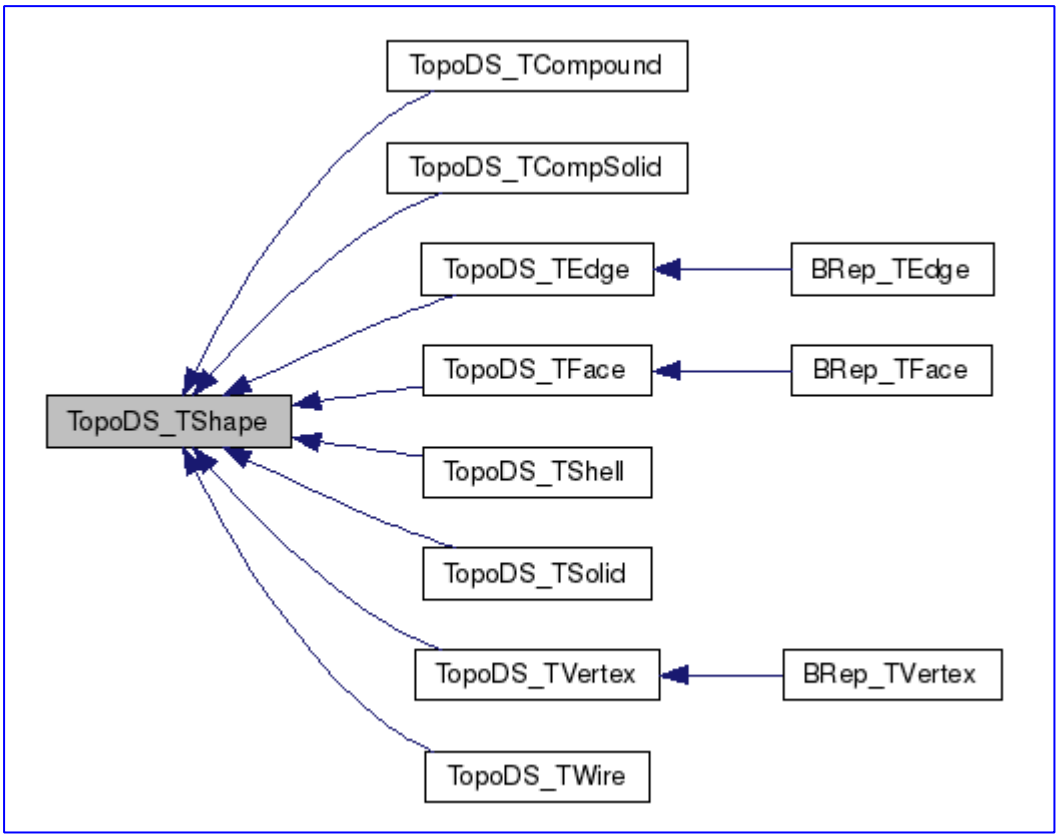

Figure 2.3 Topology data structure in OpenCascade

从上面的类图可以看出只有三种拓朴对象有几何表示数据：顶点（vertex）、边（edge）、面（face），分别为 BRep_TVertex、BRep_TEdge、BRep_TFace。

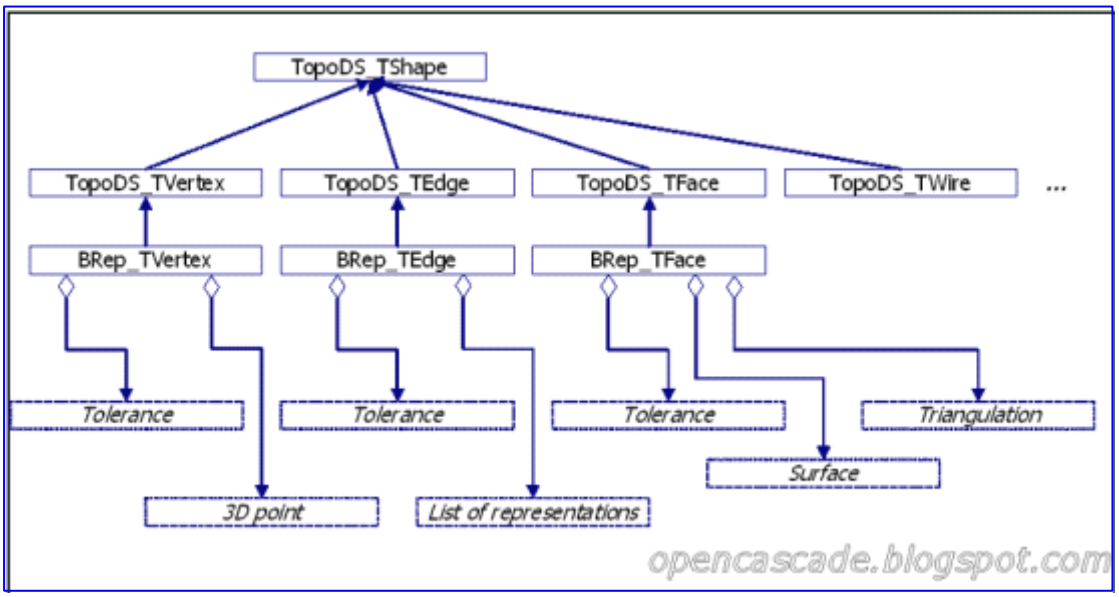

Figure 2.4 TopoDS_TShape class diagram

三、顶点 Vertex

顶点（vertex）的位置用几何点（gp_Pnt）来表示。点是几何造型中的最基本元素，自由曲线、曲面或其他形体均可用有序的点集表示。用计算机存储、管理、输出形体的实质就是对点集及其连接关系的处理。在正则形体定义中，不允许孤立点存在。

顶点的另一个重要属性是容差（Tolerance），用来表示位置精度。顶点容差 T 的几何意义为以顶点为圆心半径为 T 的球。这个球必须包含所有与这个顶点相连的边的曲线的端点。

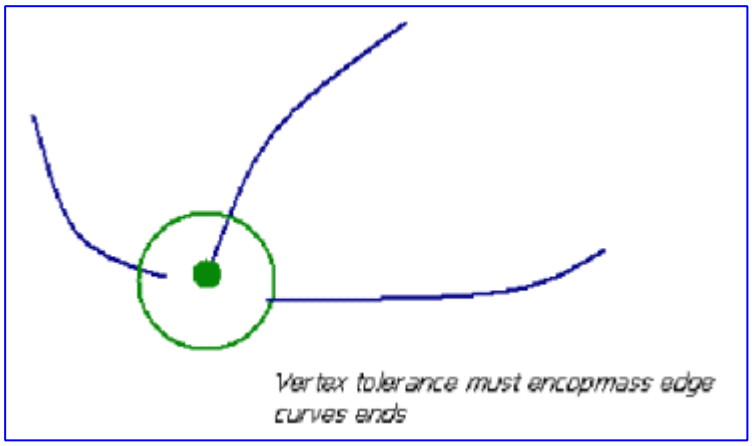

Figure 3.1 Vertex Tolerance

与其他几何库有全局精度（global precision）不同，OpenCascade 把容差作为局部属性（local properties）。由图 2.4 可知，容差是顶点、边、面的属性。这种方法有助于用更一般的方式来描述高精度的模型。如下图所示：

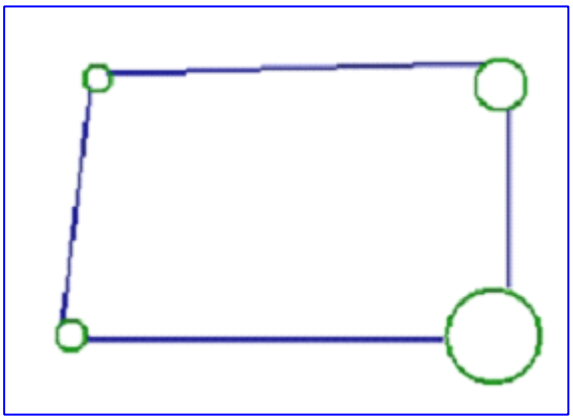

Figure 3.2 Vertex with different tolerance

如果从底层来创建形状，最好的方法就是指定最小的允许误差。默认值 Precision::Confusion()为 1e-07。

下面讨论顶点的朝向（orientation）属性。它没有直接的几何意义，但是根据约定，若顶点的朝向属性值为 TopAbs_FORWARD，它就必须与表示边的曲线的参数值小的端部匹配。相应地，TopAbs_REVERSED 的顶点与参数值大的端部匹配。例如，有条边位于圆弧上，圆弧半径为 1 且在 Z＝0 的平面上，起点为（1，0，0），向－Z 轴向，曲线为逆时针方向。所以顶点（1，0，0）的朝向为 TopAbs_FORWARD，顶点（0，1，0）的朝向为 TopAbs_REVERSED。如下图所示：

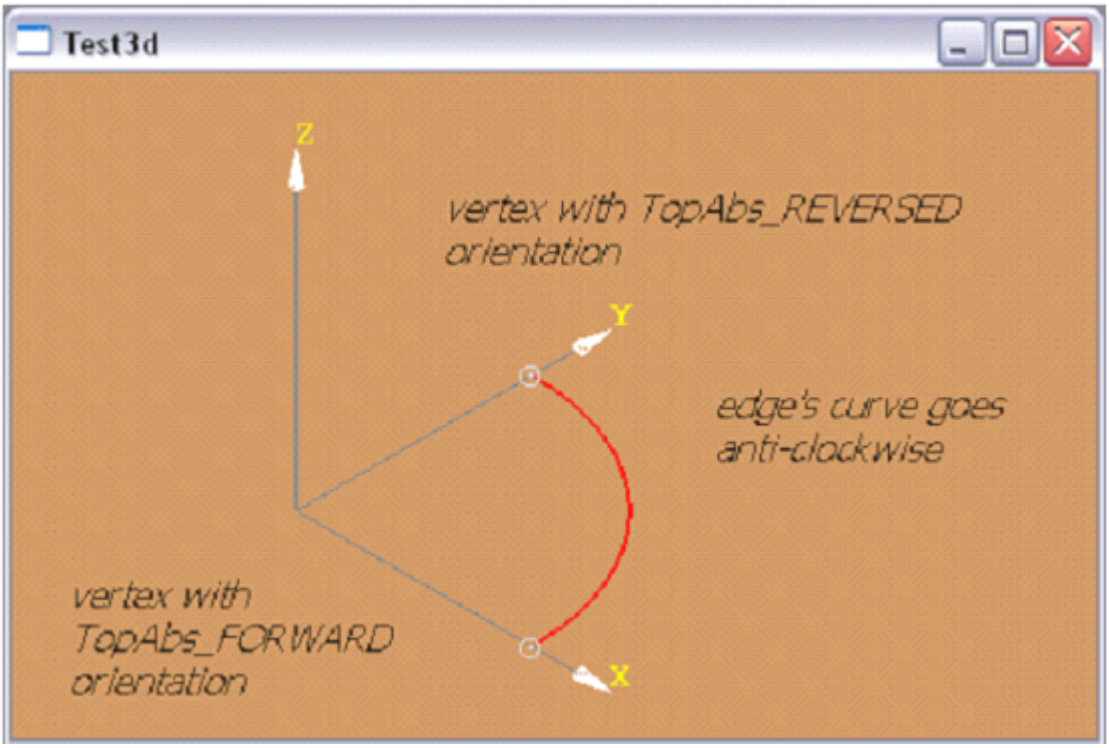

Figure 3.3 Vertex Orientation attribute

实现上图所示的程序代码如下所示：

```
/*
*    Copyright (c) 2013 eryar All Rights Reserved.
*
*        File    : Main.cpp
*        Author  : eryar@163.com
*        Date    : 2013-08-17 21:46
*        Version : 1.0v
*
*         Description  :  Demonstrate  how  to  build  a  edge  bottom-up  by
BRepBuilderAPI_MakeEdge,
*                 and how to access vertex infomation by BRep_Tool.
*
*/

// OpenCascade library.
#define WNT
#include <gp_Circ.hxx>
#include <GC_MakeArcOfCircle.hxx>

#include <TopoDS_Edge.hxx>
#include <BRep_Tool.hxx>
#include <BRepBuilderAPI_MakeEdge.hxx>
#include <BRepBuilderAPI_MakeVertex.hxx>

#pragma comment(lib, "TKernel.lib")
#pragma comment(lib, "TKMath.lib")
#pragma comment(lib, "TKBRep.lib")
#pragma comment(lib, "TKGeomBase.lib")
```

```
#pragma comment(lib, "TKTopAlgo.lib")

/**
* @breif Dump orientation types.
*        Orientation definitaion:
*        enum TopAbs_Orientation {
*            TopAbs_FORWARD,
*            TopAbs_REVERSED,
*            TopAbs_INTERNAL,
*            TopAbs_EXTERNAL
*        };
*/
std::string DumpOrientation(const TopAbs_Orientation& orient)
{
    std::string strType;

    switch (orient)
    {
    case TopAbs_FORWARD:
        strType = "TopAbs_FORWARD";
        break;

    case TopAbs_REVERSED:
        strType = "TopAbs_REVERSED";
        break;

    case TopAbs_INTERNAL:
        strType = "TopAbs_INTERNAL";
        break;

    case TopAbs_EXTERNAL:
        strType = "TopAbs_EXTERNAL";
        break;
    }

    return strType;
}

/**
* @breif Dump attributes of the vertex.
*/
void DumpVertex(const TopoDS_Vertex& v)
{
    gp_Pnt p = BRep_Tool::Pnt(v);
    Standard_Real dTolerance = BRep_Tool::Tolerance(v);

    std::cout<<"Vertex        position:        ("<<p.X()<<",        "<<p.Y()<<",
"<<p.Z()<<")"<<std::endl;
    std::cout<<"Vertex Tolerance: "<<dTolerance<<std::endl;
    std::cout<<"Vertex                                                orientation:
"<<DumpOrientation(v.Orientation())<<std::endl;
```

```
    std::cout<<std::endl;
}

int main(int argc, char* argv[])
{
    gp_Circ circle;
    TopoDS_Edge edge;
    TopoDS_Vertex vertex1;
    TopoDS_Vertex vertex2;
    BRepBuilderAPI_MakeEdge edgeBuilder;

    circle.SetRadius(1.0);
    circle.SetAxis(gp::OZ());

    edgeBuilder.Init(GC_MakeArcOfCircle(circle, 0.0, M_PI/2.0, Standard_True));

    // Test single vertex.
    /*vertex1 = BRepBuilderAPI_MakeVertex(gp_Pnt(100.0, 200.0, 300.0));
    vertex2 = BRepBuilderAPI_MakeVertex(gp_Pnt(500.0, 600.0, 700.0));

    std::cout<<"Single vetex test: "<<std::endl;

    std::cout<<"Vertex 1 attributes: "<<std::endl;
    DumpVertex(vertex1);

    std::cout<<"Vertex 2 attributes: "<<std::endl;
    DumpVertex(vertex2);*/

    edge = edgeBuilder.Edge();
    vertex1 = edgeBuilder.Vertex1();
    vertex2 = edgeBuilder.Vertex2();

    std::cout<<"Test vertex belong to edge:"<<std::endl;

    std::cout<<"Vertex 1 attributes: "<<std::endl;
    DumpVertex(vertex1);

    std::cout<<"Vertex 2 attributes: "<<std::endl;
    DumpVertex(vertex2);

    return 0;
}
```

程序运行结果如下图所示：

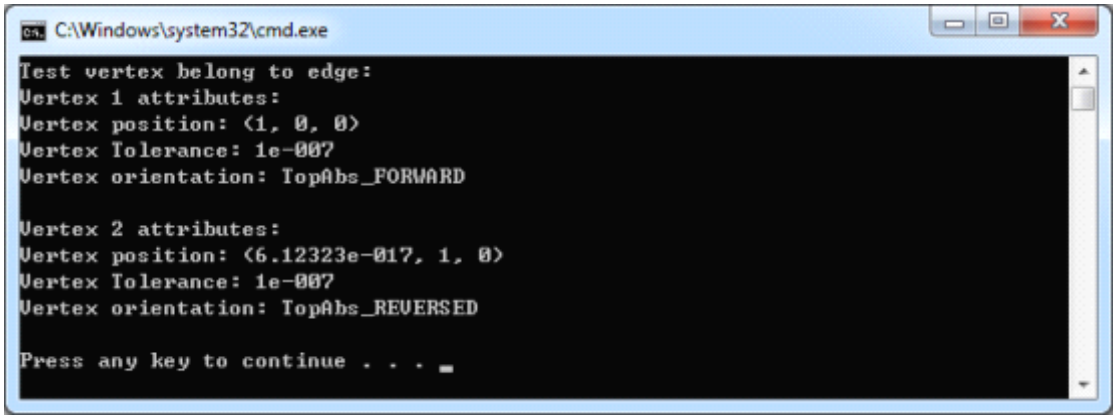

Figure 3.4 Code example result

BRep_Builder 是从底层创建拓朴结构的类。如下代码所示为从底层创建顶点的示例：

```
gp_Pnt aPoint(100.0, 200.0, 300.0)
BRep_Builder aBuilder;
TopoDS_Vertex aVertex;

aBuilder.MakeVertex(aVertex, aPoint, Precision::Confusion());
aVertex.Orientation(TopAbs_REVERSED);
```

有一个方便的类也可用来创建顶点 BRepBuilderAPI_MakeVertex，其内部也是使用了类 BRep_Builder。所以，若想从底层创建拓朴结构，必须要熟悉 BRep_Builder。

BRep_Tool 是用来访问拓朴结构中几何信息的工具，他的大部分的函数是静态的。如下代码所示为获取顶点的容差和几何点的方法：

```
Standard_Real aTolerance = BRep_Tool::Tolerance(aVertex);
gp_Pnt aPoint = BRep_Tool::Pnt(aVertex);
```

四、BRep 文件中 Vertex 的数据

结合《BRep Format Description White Paper》中对<vertex data>的描述，及程序代码中对顶点数据的读取，分析 OpenCascade 的 BRep 表示中的顶点。

BNF-like Definition

<vertex data> = <vertex data tolerance> <_\n> <vertex data 3D representation> <_\n> <vertex data representations>;

<vertex data tolerance> = <real>;

<vertex data 3D representation> = <3D point>;

<vertex data representations> = (<vertex data representation> <_\n>)* "0 0";

<vertex data representation> = <vertex data representation u parameter> <_>
<vertex data representation data> <_> <location number>;

<vertex data representation u parameter> = <real>;

<vertex data representation data> =
("1" <_> <vertex data representation data 1>) |
("2" <_> <vertex data representation data 2>) |
("3" <_> <vertex data representation data 3>);

<vertex data representation data 1> = <3D curve number>;

<vertex data representation data 2> = <2D curve number> <_> <surface number>;

<vertex data representation data 3> =
<vertex data representation v parameter> <_> <surface number>;
<vertex data representation v parameter> = <real>;

Figure 4.1 NBF-like definition of Vertex

详细说明：

<vertex data representation u parameter>u 的使用方法说明如下：

<vertex data representation data 1> 和参数 u 定义了三维曲线 C 上的点 V 的位置。参数 u 是曲线 C 上点 V 对应的参数：C（u）＝V。

<vertex data representation data 2>和参数 u 定义了曲面上的二维曲线 C 上点 V 的位置。参数 u 是曲线 C 上点 V 对应的参数：C（u）＝V。

<vertex data representation data 3>和参数 u 及<vertex data representation v parameter>v 定义了曲面 S 上的点 V：S（u，v）＝V。

<vertex data tolerance>t 定义如下所示：

$$\max_{P \in R} |P - V| \le t .$$

读取 Vertex 部分的程序代码摘抄如下：

```
case TopAbs_VERTEX :
  {
    TopoDS_Vertex& V = TopoDS::Vertex(S);

    // Read the point geometry
    IS >> tol;
    IS >> X >> Y >> Z;
```

```
      myBuilder.MakeVertex(V,gp_Pnt(X,Y,Z),tol);
      Handle(BRep_TVertex) TV = Handle(BRep_TVertex)::DownCast(V.TShape());

      BRep_ListOfPointRepresentation& lpr = TV->ChangePoints();
      TopLoc_Location L;

      do {
        IS >> p1 >> val;

        Handle(BRep_PointRepresentation) PR;
        switch (val) {

        case 1 :
          {
            IS >> c;

//  Modified by Sergey KHROMOV - Wed Apr 24 13:59:09 2002 Begin
        if (myCurves.Curve(c).IsNull())
          break;
//  Modified by Sergey KHROMOV - Wed Apr 24 13:59:13 2002 End

            Handle(BRep_PointOnCurve) POC =
              new BRep_PointOnCurve(p1,
                                    myCurves.Curve(c),
                                    L);
            PR = POC;
          }
          break;

        case 2 :
          {
            IS >> pc >> s;

//  Modified by Sergey KHROMOV - Wed Apr 24 13:59:09 2002 Begin
        if (myCurves2d.Curve2d(pc).IsNull() ||
        mySurfaces.Surface(s).IsNull())
          break;
//  Modified by Sergey KHROMOV - Wed Apr 24 13:59:13 2002 End

            Handle(BRep_PointOnCurveOnSurface) POC =
              new BRep_PointOnCurveOnSurface(p1,
                                             myCurves2d.Curve2d(pc),
                                             mySurfaces.Surface(s),
                                             L);
            PR = POC;
          }
          break;

        case 3 :
          {
            IS >> p2 >> s;
```

```
//  Modified by Sergey KHROMOV - Wed Apr 24 13:59:09 2002 Begin
        if (mySurfaces.Surface(s).IsNull())
          break;
//  Modified by Sergey KHROMOV - Wed Apr 24 13:59:13 2002 End

          Handle(BRep_PointOnSurface) POC =
            new BRep_PointOnSurface(p1,p2,
                                    mySurfaces.Surface(s),
                                    L);
          PR = POC;
        }
        break;
      }

      if (val > 0) {
        IS >> l;
    if (!PR.IsNull()) {
      PR->Location(Locations().Location(l));
      lpr.Append(PR);
    }
      }
    } while (val > 0);
  }
  break;
```

从 BRep 文件中可知，大部分的 Vertex 只有如下数据：

```
0101101
*
Ve
1e-007
2 3 0
0 0
```

即只使用了 BRep_Builder.MakeVertex()创建创建顶点（vertex），还可记录顶点的类型：

1. 若顶点在三维空间中的曲线上 Geom_Curve，则记录三维曲线的索引号及参数 u；
2. 若顶点在二维空间中的曲线上 Geom2d_Curve，则记录二维曲线的索引号及参数 u；
3. 若顶点在曲面上 Geom_Surface，则记录曲面的索引号及参数（u，v）；

五、结论 Conclusion

结合博客“OpenCascade notes”及《BRep format Description white paper》对 OpenCascade 的拓朴结构中的顶点（vertex）的属性数据进行说明。结合程序说明了顶点的容差及朝向的意义及从底层创建顶点的方法。通过 BRep_Tool 的静态函数可以获取顶点的几何数据及其他属性。

发现在 BRep 文件中还对顶点进行了分类：三维曲线上的点、二维曲线上的点和曲面上的点。

六、参考资料

1. OpenCascade notes: opencascade.blogspot.com
2. 孙家广等. 计算机图形学. 清华大学出版社