Geometry Curve of OpenCascade BRep

eryar@163.com

摘要 Abstract：几何曲线是参数表示的曲线 ，在边界表示中其数据存在于 BRep_TEdge 中，BRep_TEdge 中不仅包括了几何曲线，还包含其他类型的几何信息。本文主要对 OpenCascade 的 BRep 表示中几何曲线进行说明，将在后面分析 Topology 部分的读写程序时来说明这三种拓朴结构中分别包括哪些几何信息。

关键字 Key Words：OpenCascade BRep, Geometry Curve, Topology, Refactoring

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。

OpenCascade 中几何（Geometry）与拓朴（Topology）的关系也是按上述方式组织的。即几何信息在 BRep 中并不是单独存在的，而是依附于拓朴存在的。通过继承 TopoDS 包中的抽象的拓朴类实现了边界表示（BRep）模型。如下图所示：

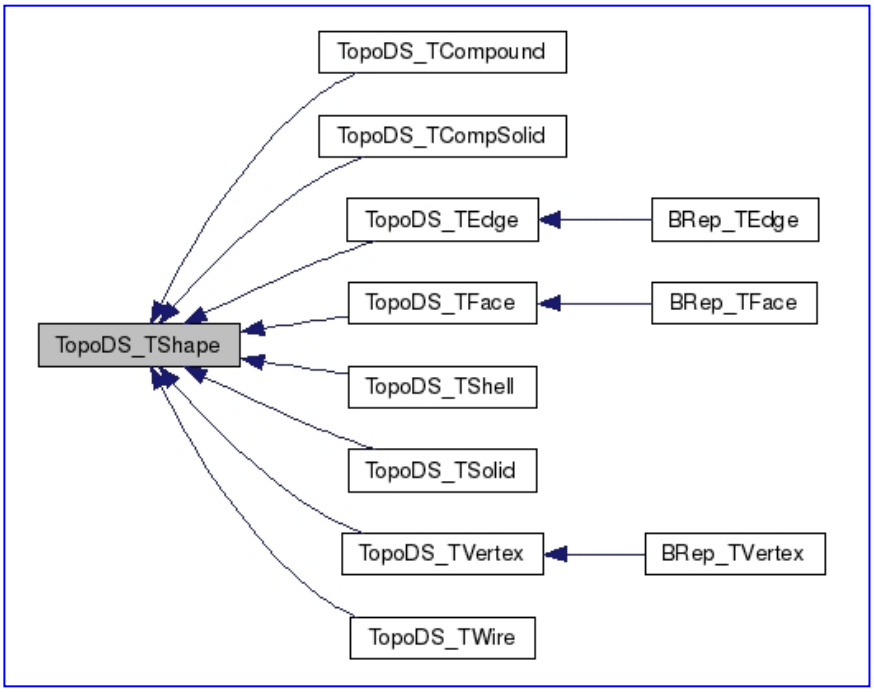


Figure 1.1 Topology data structure in OpenCascade

从上面的类图可以看出只有三种拓朴对象有几何数据：顶点（vertex）、边（edge）、面

（face），分别为 BRep_TVertex、BRep_TEdge、BRep_TFace。BRep_TVertex 中主要包含一个空间点（x, y, z）数据；几何曲线数据主要存在于 BRep_TEdge 中，BRep_TEdge 中不仅包括了几何曲线，还包含其他类型的几何信息；BRep_TFace 中主要包含几何曲面及其他的几何数据，如面的三角剖分等。本文主要对 OpenCascade 的 BRep 表示中几何曲线进行说明，将在后面分析 Topology 部分的读写程序时来说明这三种拓朴结构中分别包括哪些几何信息。

Draw Test Harness 是 OpenCascade 提供的一种灵活和简便的测试与演示 OCCT 造型库的工具。他不仅可以使用交互的方式来创建、显示和修改曲线、曲面和拓朴形状，还可以以脚本（script）的方式来使用，OpenCascade 就是用脚本的方式来对其造型内核进行自动化测试（Tests）。本文将示例程序的几何曲线在 Draw Test Harness 进行创建与显示，结合图形的直观显示便于对抽象概念的理解。

二、示例程序 Example Code

在 OpenCascade 提供的文档《BRep Format Description White Paper》对其 BRep 文件数据进行了说明。BRep 文件的几何部分包含了三维曲线，根据文档中提供的数据，利用其提供的类来将示例数据进行输出，再调试其相关代码来分析其实现。示例程序如下所示：

```
/*
*    Copyright (c) 2013 eryar All Rights Reserved.
*
*        File    : Main.cpp
*        Author  : eryar@163.com
*        Date    : 2013-11-11 21:46
*        Version : 1.0v
*
*    Description : Demonstrate the geometry 3d curve section
*                  of the BRep file of OpenCascade.
*
*       KeyWords : OpenCascade, BRep File, Geometry Curve
*
*/

// OpenCascade library.
#define WNT
#include <Geom_Line.hxx>
#include <Geom_Circle.hxx>
#include <Geom_Ellipse.hxx>
#include <Geom_Parabola.hxx>
#include <Geom_Hyperbola.hxx>
#include <Geom_BezierCurve.hxx>
#include <Geom_BSplineCurve.hxx>
#include <Geom_TrimmedCurve.hxx>
#include <Geom_OffsetCurve.hxx>

#include <TColgp_Array1OfPnt.hxx>
#include <TColStd_Array1OfReal.hxx>
#include <TColStd_Array1OfInteger.hxx>

#include <GeomTools.hxx>
#include <GeomTools_CurveSet.hxx>

#pragma comment(lib, "TKernel.lib")
#pragma comment(lib, "TKMath.lib")
#pragma comment(lib, "TKG3d.lib")
#pragma comment(lib, "TKGeomBase.lib")

int main(void)
{
    gp_Ax2 axis(gp_Pnt(1, 2, 3), gp::DZ());
    std::ofstream dumpFile("geometryCurve.txt");

    // 3D curve record 1: Line.
    // Example: 1 1 0 3 0 1 0
```

```
    Handle_Geom_Line theLine = new Geom_Line(gp_Pnt(1, 0, 3), gp_Dir(0, 1, 0));
    GeomTools::Write(theLine, dumpFile);
    GeomTools::Dump(theLine, dumpFile);
    GeomTools::Dump(theLine, std::cout);

    // 3D curve record 2: Circle.
    // Example: 2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
    Handle_Geom_Circle theCircle = new Geom_Circle(axis, 4.0);
    GeomTools::Write(theCircle, dumpFile);
    GeomTools::Dump(theCircle, dumpFile);
    GeomTools::Dump(theCircle, std::cout);

    // 3D curve record 3: Ellipse.
    // Example: 3 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
    Handle_Geom_Ellipse theEllipse = new Geom_Ellipse(axis, 5.0, 4.0);
    GeomTools::Write(theEllipse, dumpFile);
    GeomTools::Dump(theEllipse, dumpFile);
    GeomTools::Dump(theEllipse, std::cout);

    // 3D curve record 4: Parabola.
    // Example: 4 1 2 3 0 0 1 1 0 -0 -0 1 0 16
    Handle_Geom_Parabola theParabola = new Geom_Parabola(axis, 16.0);
    GeomTools::Write(theParabola, dumpFile);
    GeomTools::Dump(theParabola, dumpFile);
    GeomTools::Dump(theParabola, std::cout);

    // 3D curve record 5: Hyperbola.
    // Example: 5 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
    Handle_Geom_Hyperbola theHyperbola = new Geom_Hyperbola(axis, 5.0, 4.0);
    GeomTools::Write(theHyperbola, dumpFile);
    GeomTools::Dump(theHyperbola, dumpFile);
    GeomTools::Dump(theHyperbola, std::cout);

    // 3D curve record 6: Bezier Curve.
    // Example: 6 1 2 0 1 0  4 1 -2 0  5 2 3 0  6
    TColgp_Array1OfPnt poles(1, 3);
    TColStd_Array1OfReal weights(1, 3);

    poles.SetValue(1, gp_Pnt(0, 1, 0));
    poles.SetValue(2, gp_Pnt(1, -2, 0));
    poles.SetValue(3, gp_Pnt(2, 3, 0));

    weights.SetValue(1, 4.0);
    weights.SetValue(2, 5.0);
    weights.SetValue(3, 6.0);

    Handle_Geom_BezierCurve theBezierCurve = new Geom_BezierCurve(poles,
weights);
    GeomTools::Write(theBezierCurve, dumpFile);
    GeomTools::Dump(theBezierCurve, dumpFile);
    GeomTools::Dump(theBezierCurve, std::cout);
```

```
    // 3D curve record 7: B-Spline Curve.
    // Example: 7 1 0  1 3 5  0 1 0  4 1 -2 0  5 2 3 0  6
    //          0 1 0.25 1 0.5 1 0.75 1 1 1
    Standard_Integer degree = 1;
    TColStd_Array1OfReal knots(1, 5);
    TColStd_Array1OfInteger multiplicities(1, 5);

    knots.SetValue(1, 0);
    knots.SetValue(2, 0.25);
    knots.SetValue(3, 0.5);
    knots.SetValue(4, 0.75);
    knots.SetValue(5, 1.0);

    // all knots multiplicity of the B-spline is 1.
    multiplicities.Init(1);

    Handle_Geom_BSplineCurve  theBSplineCurve  =  new  Geom_BSplineCurve(poles,
weights, knots, multiplicities, degree);
    GeomTools::Write(theBSplineCurve, dumpFile);
    GeomTools::Dump(theBSplineCurve, dumpFile);
    GeomTools::Dump(theBSplineCurve, std::cout);

    // 3D curve record 8: Trimmed Curve.
    // Example: 8 -4 5
    //          1 1 2 3 1 0 0
    Handle_Geom_Line theBaseCurve = new Geom_Line(gp_Pnt(1, 2, 3), gp_Dir(1, 0,
0));
    Handle_Geom_TrimmedCurve theTrimmedCurve = new Geom_TrimmedCurve(theBaseCurve,
-4, 5);
    GeomTools::Write(theTrimmedCurve, dumpFile);
    GeomTools::Dump(theTrimmedCurve, dumpFile);
    GeomTools::Dump(theTrimmedCurve, std::cout);

    // 3D curve record 9: Offset Curve.
    // Example: 9 2
    //          0 1 0
    //          1 1 2 3 1 0 0
    Handle_Geom_OffsetCurve theOffsetCurve = new Geom_OffsetCurve(theBaseCurve,
2.0, gp::DY());
    GeomTools::Write(theOffsetCurve, dumpFile);
    GeomTools::Dump(theOffsetCurve, dumpFile);
    GeomTools::Dump(theOffsetCurve, std::cout);

    return 0;
}
```

上述程序将《BRep Format Description White Paper》中的几何部分（Geometry Section）的三维曲线（3D Curves）示例数据分别使用类 GeomTools 的静态函数输出到屏幕和文件。

当使用 GeomTools::Write()时输出的内容与 BRep 文件中一致，当使用 GeomTools::Dump()时输出更易读的信息。为了便于对比理解，将两种形式都输出到文件 geometryCurve.txt 中，输出数据如下所示：

```
1 1 0 3 0 1 0
Line
   Origin :1, 0, 3
   Axis      :0, 1, 0

2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
Circle
   Center :1, 2, 3
   Axis      :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
   Radius :4

3 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
Ellipse
   Center :1, 2, 3
   Axis      :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
   Radii    :5, 4

4 1 2 3 0 0 1 1 0 -0 -0 1 0 16
Parabola
   Center :1, 2, 3
   Axis      :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
   Focal    :16

5 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
Hyperbola
   Center :1, 2, 3
   Axis      :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
   Radii    :5, 4

6 1 2 0 1 0    4 1 -2 0    5 2 3 0    6
BezierCurve rational
   Degree :2
     1 : 0, 1, 0    4
```

```
    2 : 1, -2, 0    5
    3 : 2, 3, 0    6

7 1 0    1 3 5    0 1 0    4 1 -2 0    5 2 3 0    6
 0 1 0.25 1 0.5 1 0.75 1 1 1
BSplineCurve rational
   Degree 1, 3 Poles, 5    Knots
Poles :

    1 : 0, 1, 0    4
    2 : 1, -2, 0    5
    3 : 2, 3, 0    6
Knots :

    1 :    0 1
    2 :    0.25 1
    3 :    0.5 1
    4 :    0.75 1
    5 :    1 1

8 -4 5
1 1 2 3 1 0 0
Trimmed curve
Parameters : -4 5
Basis curve :
Line
   Origin :1, 2, 3
   Axis      :1, 0, 0

9 2
0 1 0
1 1 2 3 1 0 0
OffsetCurveOffset : 2
Direction : 0, 1, 0
Basis curve :
Line
   Origin :1, 2, 3
   Axis      :1, 0, 0
```

三、程序说明 Example Description

3.1 直线 Line

示例：

```
// 3D curve record 1: Line.
// Example: 1 1 0 3 0 1 0
Handle_Geom_Line theLine = new Geom_Line(gp_Pnt(1, 0, 3), gp_Dir(0, 1, 0));
GeomTools::Write(theLine, dumpFile);
```

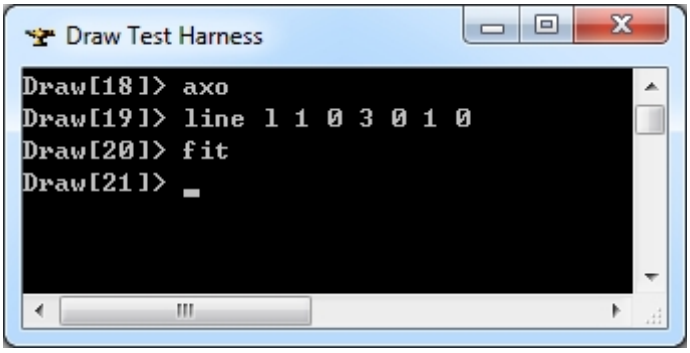
<3D curve record 1>描述的直线数据包含一个三维点 P 和三维方向 D，其参数方程为：

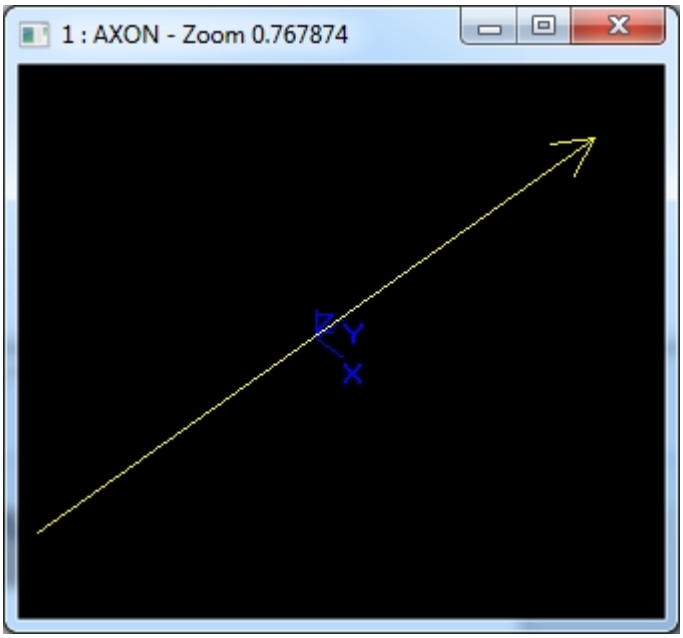
$$C(u) = P + u \cdot D\ ,\ u \in (-\infty, \infty).$$

示例数据表示的直线为经过点（1,0,3）且方向 D 为（0,1,0）的直线，其参数方程表示为：

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

在 Draw Test Harness 中创建并显示直线如下所示：





3.2 圆 Circle

示例：

```
// 3D curve record 2: Circle.
// Example: 2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
gp_Ax2 axis(gp_Pnt(1, 2, 3), gp::DZ());
Handle_Geom_Circle theCircle = new Geom_Circle(axis, 4.0);
GeomTools::Write(theCircle, dumpFile);
```

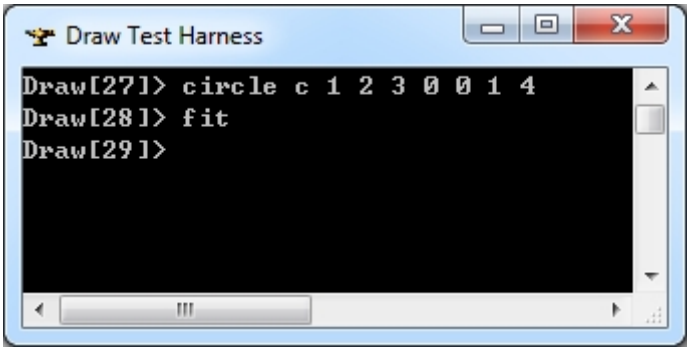
<3D curve record 2>描述的圆的数据包含表示圆心坐标的三维点 P，三个方向 N，Dx，Dy 表示的坐标系和半径 r。其参数方程如下所示：

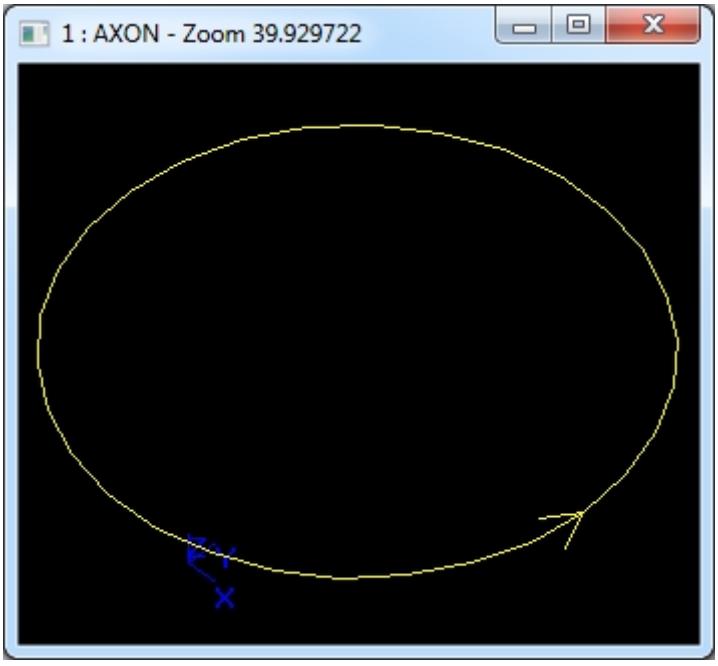
$$C(u) = P + r \cdot (\cos(u) \cdot D_X + \sin(u) \cdot D_Y),\ u \in [0, 2 \cdot \pi).$$

示例数据表示的圆是圆心坐标为（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：

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

在 Draw Test Harness 中创建并显示圆如下所示：





3.3 椭圆 Ellipse

示例：

```
// 3D curve record 3: Ellipse.
// Example: 3 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
Handle_Geom_Ellipse theEllipse = new Geom_Ellipse(axis, 5.0, 4.0);
GeomTools::Write(theEllipse, dumpFile);
```

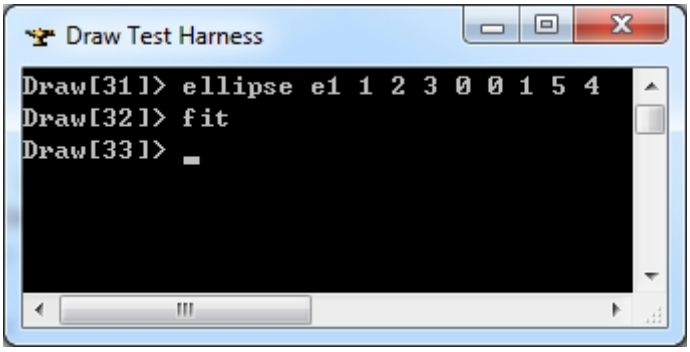
<3D curve record 3>定义了椭圆。椭圆的数据包含三维点 P，三维正交坐标系 N、D_{maj}、D_{min} 和两个非负实数 r_{maj} 和 r_{min}，且 r_{min}<=r_{maj}。椭圆位于中心点 P，法向量为 N 的平面上，且长轴、短轴的方向分别为 D_{maj}, D_{min}，长轴、短轴上的半径分别为 r_{maj}, r_{min}。椭圆的参数方程定义如下所示：

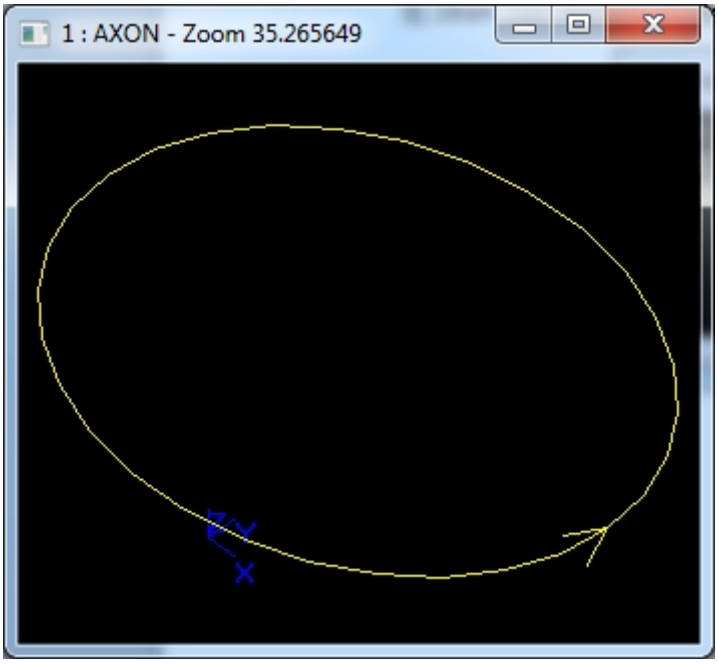
$$C(u) = P + r_{maj} \cdot \cos(u) \cdot D_{maj} + r_{min} \cdot \sin(u) \cdot D_{min},\ u \in [0, 2 \cdot \pi).$$

示例数据表示的椭圆的中心点 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，

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

在 Draw Test Harness 中创建并显示椭圆如下所示：





3.4 抛物线 Parabola

示例：

```
// 3D curve record 4: Parabola.
// Example: 4 1 2 3 0 0 1 1 0 -0 -0 1 0 16
Handle_Geom_Parabola theParabola = new Geom_Parabola(axis, 16.0);
GeomTools::Write(theParabola, dumpFile);
```

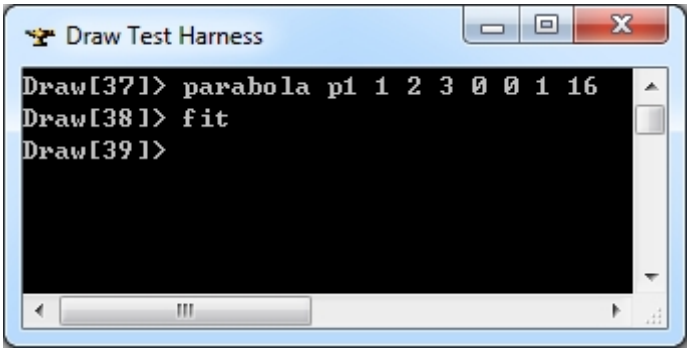
<3D curve record 4>定义了抛物线。抛物线数据包含三维点 P，三维正交坐标系坐标轴方向 N，Dx，Dy 和一个非负的实数 f。抛物线通过点 P，且位于法向量为 N 的平面上，焦点长度为 f，其参数方程如下所示：

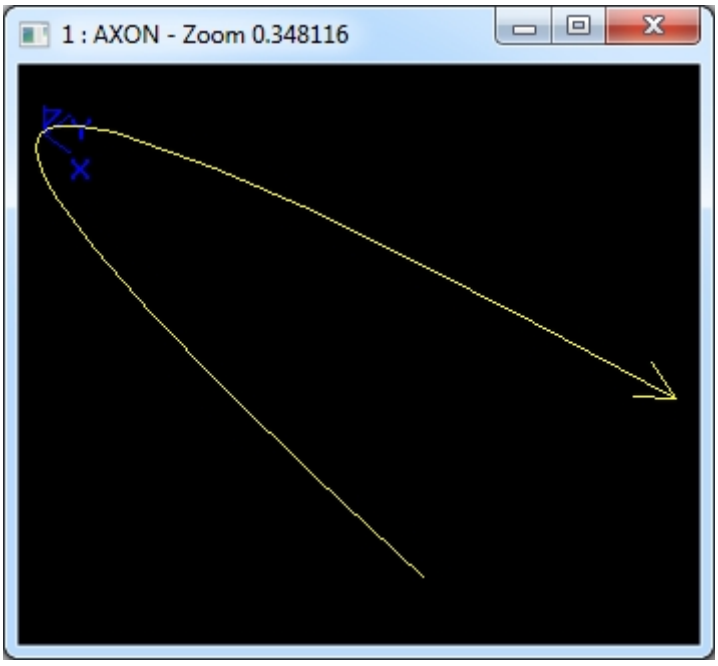
$$C(u)=P+\frac{u^2}{4\cdot f}\cdot D_X+u\cdot D_Y,\ u\in(-\infty,\infty)\Leftarrow f\neq 0;$$

示例数据表示的抛物线过点 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：

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

在 Draw Test Harness 中创建并显示抛物线如下所示：





3.5 双曲线 Hyperbola

示例：

```
// 3D curve record 5: Hyperbola.
// Example: 5 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
Handle_Geom_Hyperbola theHyperbola = new Geom_Hyperbola(axis, 5.0, 4.0);
GeomTools::Write(theHyperbola, dumpFile);
```

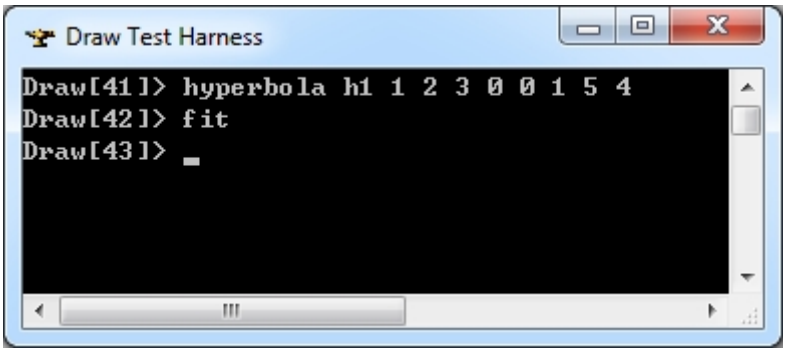
<3D curve record 5>定义了双曲线。双曲线定义数据有三维点 P，三维正交坐标系坐标轴方向为 N，Dx，Dy 和两个非负实数 Kx，Ky。双曲线过 P 点且法向量为 N 的平面上，其参数方程如下所示：

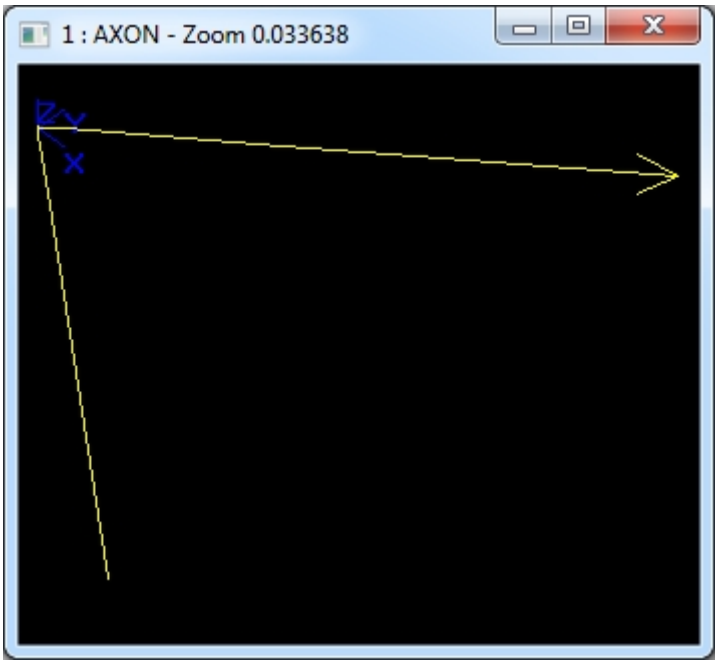
$$C(u)=P+k_X\cdot\cosh(u)\cdot D_X+k_Y\cdot\sinh(u)\cdot D_Y,\ u\in(-\infty,\infty).$$

示例数据表示的双曲线过点 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：

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

在 Draw Test Harness 中创建并显示双曲线如下所示：





3.6 Bezier 曲线 Bezier Curve

示例：

```
// 3D curve record 6: Bezier Curve.
// Example: 6 1 2 0 1 0  4 1 -2 0  5 2 3 0  6
TColgp_Array1OfPnt poles(1, 3);
TColStd_Array1OfReal weights(1, 3);

poles.SetValue(1, gp_Pnt(0, 1, 0));
poles.SetValue(2, gp_Pnt(1, -2, 0));
poles.SetValue(3, gp_Pnt(2, 3, 0));

weights.SetValue(1, 4.0);
weights.SetValue(2, 5.0);
weights.SetValue(3, 6.0);

Handle_Geom_BezierCurve theBezierCurve = new Geom_BezierCurve(poles, weights);
GeomTools::Write(theBezierCurve, dumpFile);
```

<3D curve record 6>定义了 Bezier 曲线。Bezier 曲线数据包含有理标志 r，曲线的次数 m（degree m <= 25 查看源代码可知 OpenCascade 可处理的 B 样条次数不超过 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：

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

在 Draw Test Harness 中创建并显示 Bezier 曲线如下所示：


1 : AXON - Zoom 106.537422
Y
X

3.7 B 样条曲线 B-Spline Curve

示例：

```
// 3D curve record 7: B-Spline Curve.
// Example: 7 1 0  1 3 5  0 1 0  4 1 -2 0  5 2 3 0  6
//          0 1 0.25 1 0.5 1 0.75 1 1 1
Standard_Integer degree = 1;
TColStd_Array1OfReal knots(1, 5);
TColStd_Array1OfInteger multiplicities(1, 5);

knots.SetValue(1, 0);
knots.SetValue(2, 0.25);
knots.SetValue(3, 0.5);
knots.SetValue(4, 0.75);
knots.SetValue(5, 1.0);

// all knots multiplicity of the B-spline is 1.
multiplicities.Init(1);

Handle_Geom_BSplineCurve theBSplineCurve = new Geom_BSplineCurve(poles, weights,
knots, multiplicities, degree);
GeomTools::Write(theBSplineCurve, dumpFile);
```

<3D curve record 7>定义了 B-Spline 曲线。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 和节点重数 multiplicity knots。

当有理标志位 r＝0 时，是非有理 B 样条曲线，weight poles 有 n 个三维点 B1，...，Bn；当有理标志位 r＝1 时，是有理 B 样条曲线，weight poles 是 n 个带权控制点对：B1, h1, ... Bn, hn。这里 Bi 表示一个三维点，hi 表示一个[0，1]正实数。当有理标志位 r＝0 时，hi＝1。

重节点有 k 对 u1, q1, ... uk, qk。这里 ui 是重复度为 qi>=1 的节点。

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

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

B-Spline 曲线的参数方程如下所示：

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

其中基函数 Ni,j 有如下的递归定义：

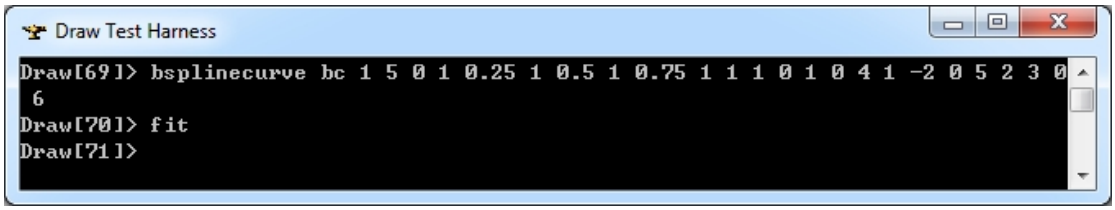
$$N_{i,1}(u)=\begin{cases}1 \Leftarrow \bar{u}_i \le u < \bar{u}_{i+1} \\ 0 \Leftarrow u < \bar{u}_i \vee \bar{u}_{i+1} \le u\end{cases},\quad N_{i,j}(u)=\frac{(u-\bar{u}_i)\cdot N_{i,j-1}(u)}{\bar{u}_{i+j-1}-\bar{u}_i}+\frac{(\bar{u}_{i+j}-u)\cdot N_{i+1,j-1}(u)}{\bar{u}_{i+j}-\bar{u}_{i+1}}\quad (2\le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

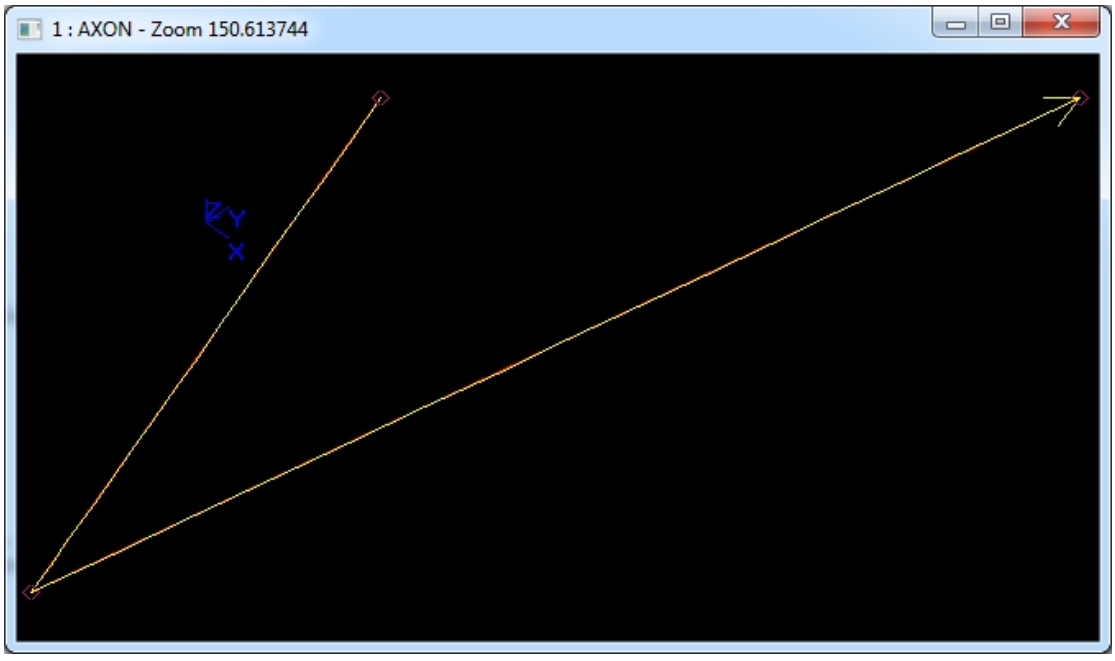
$$\bar{u}_i = u_j \ (1\le j\le k,\ \sum_{l=1}^{j-1} q_l + 1 \le i \le \sum_{l=1}^{j} q_l).$$

示例数据表示的 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：

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

在 Draw Test Harness 中创建并显示 B-Spline 曲线如下所示：





3.8 裁剪曲线 Trimmed Curve

示例：

```
// 3D curve record 8: Trimmed Curve.
// Example: 8 -4 5
//          1 1 2 3 1 0 0
Handle_Geom_Line theBaseCurve = new Geom_Line(gp_Pnt(1, 2, 3), gp_Dir(1, 0, 0));
Handle_Geom_TrimmedCurve theTrimmedCurve = new Geom_TrimmedCurve(theBaseCurve, -4,
5);
GeomTools::Write(theTrimmedCurve, dumpFile);
```

<3D curve record 8>定义了裁剪曲线（trimmed curve）。裁剪曲线数据包含：两个实数 u_{min}，u_{max} 和<3D curve record>，且 $u_{min}<u_{max}$。裁剪曲线是将<3D curve record>描述的曲线 B 限制在[u_{min}，u_{max}]。裁剪曲线的参数方程如下所示：

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

示例数据表示的裁剪曲线为：umin＝－4，umax＝5，曲线 B(u)＝（1，2，3）+u（1，0，0）。裁剪曲线的参数方程如下所示：

$$C(u)=(1,2,3)+u\cdot(1,0,0),\ u\in[-4,5].$$

3.9 偏移曲线 Offset Curve

示例：

```
// 3D curve record 9: Offset Curve.
// Example: 9 2
//          0 1 0
//          1 1 2 3 1 0 0
Handle_Geom_OffsetCurve theOffsetCurve = new Geom_OffsetCurve(theBaseCurve, 2.0,
gp::DY());
GeomTools::Write(theOffsetCurve, dumpFile);
```

<3D curve record 9>定义了偏移曲线（offset curve）。偏移曲线的数据包含偏移距离 d，偏移方向 D 和曲线数据<3D curve record>。偏移曲线是将<3D curve record>描述的曲线沿矢量$[B'(u),D]\neq\vec{0}$偏移距离 d 后的结果。偏移曲线的参数方程如下所示：

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

示例数据表示的偏移曲线为偏移距离 d＝2，方向 D＝（0，1，0），基曲线 B(u)＝（1，2，3）+u（1，0，0），其参数方程如下所示：

$$C(u)=(1,2,3)+u\cdot(1,0,0)+2\cdot(0,0,1).$$

四、程序分析 Refactoring the Code

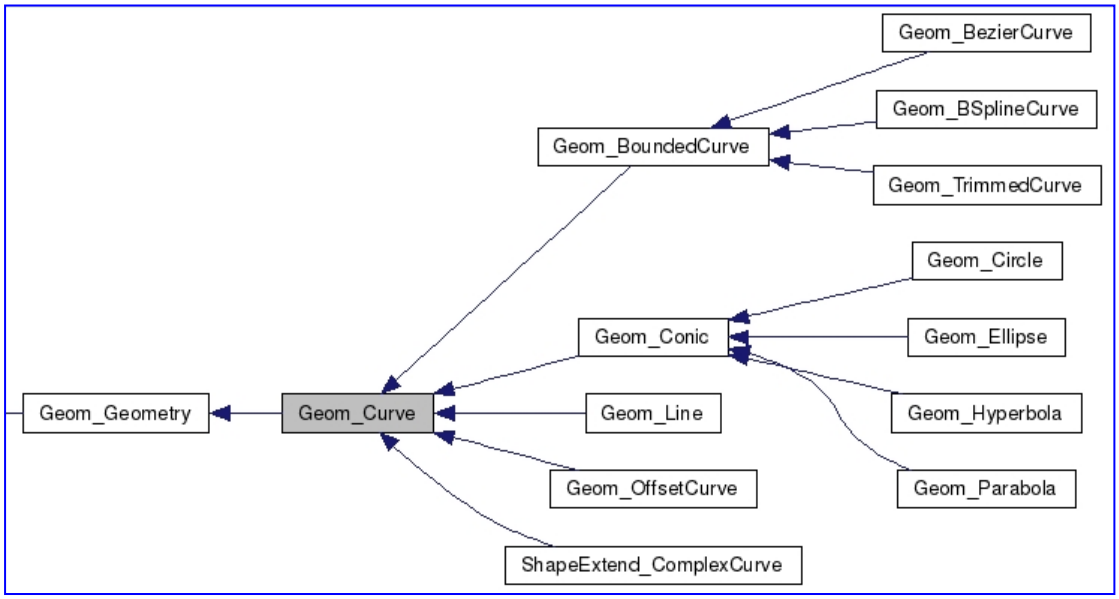


Figure 4.1 Class diagram of Geom_Curve

根据几何曲线的类图可知，几何曲线有个共同的基类 Geom_Curve。而在对几何数据进行输出与读入时，用了很多条件判断。输出部分程序代码如下所示：

```
//=======================================================================
//function : Print
//purpose  :
//=======================================================================
void GeomTools_CurveSet::PrintCurve(const Handle(Geom_Curve)& C,
                    Standard_OStream& OS,
                    const Standard_Boolean compact)
{
  Handle(Standard_Type) TheType = C->DynamicType();

  if ( TheType ==STANDARD_TYPE(Geom_Line)) {
    Print(Handle(Geom_Line)::DownCast(C),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_Circle)) {
    Print(Handle(Geom_Circle)::DownCast(C),OS,compact);
  }
  else if ( TheType == STANDARD_TYPE(Geom_Ellipse)) {
    Print(Handle(Geom_Ellipse)::DownCast(C),OS,compact);
  }
  else if ( TheType == STANDARD_TYPE(Geom_Parabola)) {
    Print(Handle(Geom_Parabola)::DownCast(C),OS,compact);
  }
  else if ( TheType == STANDARD_TYPE(Geom_Hyperbola)) {
    Print(Handle(Geom_Hyperbola)::DownCast(C),OS,compact);
  }
  else if ( TheType == STANDARD_TYPE(Geom_BezierCurve)) {
    Print(Handle(Geom_BezierCurve)::DownCast(C),OS,compact);
  }
  else if ( TheType == STANDARD_TYPE(Geom_BSplineCurve)) {
```

```
    Print(Handle(Geom_BSplineCurve)::DownCast(C),OS,compact);
  }
  else if ( TheType == STANDARD_TYPE(Geom_TrimmedCurve)) {
    Print(Handle(Geom_TrimmedCurve)::DownCast(C),OS,compact);
  }
  else if ( TheType == STANDARD_TYPE(Geom_OffsetCurve)) {
    Print(Handle(Geom_OffsetCurve)::DownCast(C),OS,compact);
  }
  else {
    GeomTools::GetUndefinedTypeHandler()->PrintCurve(C,OS,compact);
    //if (!compact)
    //  OS << "****** UNKNOWN CURVE TYPE ******\n";
    //else
    //  cout << "****** UNKNOWN CURVE TYPE ******" << endl;
  }
}
```

读入部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
Standard_IStream& GeomTools_CurveSet::ReadCurve(Standard_IStream& IS,
                        Handle(Geom_Curve)& C)
{
  Standard_Integer ctype;

  try {
    OCC_CATCH_SIGNALS
    IS >> ctype;
    switch (ctype) {

    case LINE :
      {
        Handle(Geom_Line) CC;
        IS >> CC;
        C = CC;
      }
      break;

    case CIRCLE :
      {
        Handle(Geom_Circle) CC;
        IS >> CC;
        C = CC;
      }
      break;

    case ELLIPSE :
      {
        Handle(Geom_Ellipse) CC;
        IS >> CC;
```

```
      C = CC;
    }
    break;

  case PARABOLA :
    {
      Handle(Geom_Parabola) CC;
      IS >> CC;
      C = CC;
    }
    break;

  case HYPERBOLA :
    {
      Handle(Geom_Hyperbola) CC;
      IS >> CC;
      C = CC;
    }
    break;

  case BEZIER :
    {
      Handle(Geom_BezierCurve) CC;
      IS >> CC;
      C = CC;
    }
    break;

  case BSPLINE :
    {
      Handle(Geom_BSplineCurve) CC;
      IS >> CC;
      C = CC;
    }
    break;

  case TRIMMED :
    {
      Handle(Geom_TrimmedCurve) CC;
      IS >> CC;
      C = CC;
    }
    break;

  case OFFSET :
    {
      Handle(Geom_OffsetCurve) CC;
      IS >> CC;
      C = CC;
    }
    break;
```

```
    default:
      {
        Handle(Geom_Curve) CC;
        GeomTools::GetUndefinedTypeHandler()->ReadCurve(ctype,IS,CC);
        C = CC;
      }
    }
  }
  catch(Standard_Failure) {
#ifdef DEB
    Handle(Standard_Failure) anExc = Standard_Failure::Caught();
    cout <<"EXCEPTION in GeomTools_CurveSet::ReadCurve(..)!!!" << endl;
    cout << anExc << endl;
#endif
    C = NULL;
  }
  return IS;
}
```

正如《Refactoring-Improving the Design of Existing Code》书中以多态取代条件表达式（Replace Conditional with Polymorphism）所说，在面向对象术语中，听上去最高贵的词非“多态”莫属。多态最根本的好处就是如果你需要根据对象的不同类型而采取不同的行为，多态使你不必编写明显的条件表达式。正因为有了多态，所以你会发现“类型码的 switch 语句”以及“基于类型名称的 if-then-else 语句”在面向对象程序中很少出现。

多态能够带给你很多好处。如果同一组条件表达式在程序许多地方出现，那么使用多态的收益是最大的。使用条件表达式时，如果你想添加一种新类型，就必须查找并更新所有条件表达式。但如果改用多态，只需要一个新的子类，并在其中提供适当的函数就行了。类的用户不需要了解这个子类，这就大降低了系统各部分之间的依赖，使系统升级更容易。

OpenCascade 的几何曲线已经有一个基类 Geom_Curve 了，可将输出做为虚函数，就不需要做判断了。在读入（创建）时引入工厂模式，对于 UndefinedTypeHandler()可以引入 Null 对象。经过这样重构之后的程序可读性应该会更好吧！

五、结论 Conclusion

在边界表示 BRep 的形状中，参数表示的几何曲线并不会孤立存在，他总是依附于拓朴边中。在 OpenCascade 的 BRep 格式的文件中三维几何曲线共有九种，通过将这九种几何曲线输出，理解参数表示的几何曲线的数据结构。

通过查看其读写几何曲线的源程序，提出重构的方法。当在面向对象的程序中出现很条件表达式时，那么程序就有“坏味道”了，需要进行重构改进。

六、参考资料 References

1. OpenCascade. BRep Format Description White Paper
2. Martin Fowler. Refactoring:Improving the Design of Existing Code. Addison-Wesley
3. Les Piegl, Wayne Tiller. The NURBS Book. Springer-Verlag