OpenCascade BRep Format Description

eryar@163.com

摘要 Abstract：本文结合 OpenCascade 的 BRep 格式描述文档和源程序，对 BRep 格式进行分析，详细说明 BRep 的数据组织形式。结合源程序，可以对 OpenCascade 中 Modeling Data 模块中的模型数据结构进行理解。

关键字 Key Words：OpenCascade, BRep Format, ModelingData

一、引言 Introduction

OpenCascade 中的 BRep 格式主要用来存储 3D 模型，也可用来存储由下列元素组成的模型： vertices, edges, wires, faces, shells, solids, compsolids, compounds, edge triangulations, face triangulations, polylines on triangulations, space location and orientation.

本格式的目的就是为了便于理解，也使用了类似 BNF 的定义方式。以下章节都是按下面的格式组织的：

- 该部分的示例；
- 该部分的类 BNF 定义；
- 该部分的详细说明；
- 该部分的源程序片段。

二、通用结构 Format Common Structure

BRep 格式文件的读写采用了 ASCII 的编码方式，该格式的数据都是文本形式存储。BRep 格式使用了下面的 BNF 术语：

1) <\n>：换行；
2) <_\n>：
3) <_>：空格；
4) <flag>：标志位：0 和 1；
5) <int>：整数，范围-2^{31} 到 2^{31}-1；
6) <real>：实数，范围-1.7976931348623159X10^{308} 到 1.7976931348623158X10^{308}；
7) <2D point>：二维点，两个实数；
8) <3D point>：三维点，三个实数；
9) <2D direction>：二维方向矢量，两个实数，平方和为 1，即为单位方向矢量；
10) <3D direction>：三维方向矢量，三个实数，平方和为 1，即为单位方向矢量；
11) <+>

BRep 格式包含以下部分：

1) <content type>
2) <version>
3) <locations>

4) <geometry>

5) <shapes>

<content type>部分：

```
<content type> = “DBRep_DrawableShape” <_\n> <_\n>;
```

<content type>也可以有其它的值。

<version>部分：

```
<version> = (“CASCADE Topology V1, (c) Matra-Datavision” |
“CASCADE Topology V2, (c) Matra-Datavision”) <_\n>;
```

不同版本之间的区别将会在本文档中说明。

三、<locations>部分 Section <locations>

示例：

```
Locations 3
1
                0               0               1               0
                1               0               0               0
                0               1               0               0
1
                1               0               0               4
                0               1               0               5
                0               0               1               6
2  1 1 2 1 0
```

BNF 定义：

BNF-like Definition

<locations> = <location header> <_\n> <location records>;

<location header> = “Locations” <_> <location record count>;

<location record count> = <int>;

<location records> = <location record> ^ <location record count>;

<location record> = <location record 1> | <location record 2>;

<location record 1> = “1” <_\n> <location data 1>;

<location record 2> = “2” <_> <location data 2>;

<location data 1> = ((<_> <real>) ^ 4 <_\n>) ^ 3;

<location data 2> = (<int> <_> <int> <_>)* “0” <_\n>;

详细说明：

<location data 1>定义了 3X4 的矩阵 Q，描述了三维空间的线性变换，并满足如下约定：

$$Q=\begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{pmatrix}$$

$$1)Q_2=\begin{pmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{pmatrix}, d=|Q_2|, d\neq 0$$

$$2)Q_3=Q_2/d^{\frac{1}{3}}, Q_3^T=Q_3^{-1}$$

矩阵 Q 是线性变换矩阵，它可以通过矩阵乘法将一个点（x, y, z）变换成另外一点（u, v, w）：

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix}=Q\cdot\begin{pmatrix} x & y & z & 1 \end{pmatrix}^T=\begin{pmatrix} q_{11}\cdot x+q_{12}\cdot y+q_{13}\cdot z+q_{14} \\ q_{21}\cdot x+q_{22}\cdot y+q_{23}\cdot z+q_{24} \\ q_{31}\cdot x+q_{32}\cdot y+q_{33}\cdot z+q_{34} \end{pmatrix}$$

Q 也可能是以下基本变换矩阵的组合：

1) 平移变换矩阵：

$$\begin{pmatrix} 1 & 0 & 0 & q_{14} \\ 0 & 1 & 0 & q_{24} \\ 0 & 0 & 1 & q_{34} \end{pmatrix}$$

2) 绕任意轴旋转的变换矩阵，轴的方向为 D（Dx, Dy, Dz），旋转角度 ψ：

$$\begin{pmatrix} D_x^2\cdot(1-\cos\varphi)+\cos\varphi & D_x\cdot D_y\cdot(1-\cos\varphi)-D_z\cdot\sin\varphi & D_x\cdot D_z\cdot(1-\cos\varphi)+D_y\cdot\sin\varphi & 0 \\ D_x\cdot D_y\cdot(1-\cos\varphi)+D_z\cdot\sin\varphi & D_y^2\cdot(1-\cos\varphi)+\cos\varphi & D_y\cdot D_z\cdot(1-\cos\varphi)-D_x\cdot\sin\varphi & 0 \\ D_x\cdot D_z\cdot(1-\cos\varphi)-D_y\cdot\sin\varphi & D_y\cdot D_z\cdot(1-\cos\varphi)+D_x\cdot\sin\varphi & D_z^2\cdot(1-\cos\varphi)+\cos\varphi & 0 \end{pmatrix}$$

3) 缩放变换矩阵：

$$\begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \end{pmatrix}$$

4) 中心对称变换矩阵：

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

5) 轴对称变换矩阵：

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

6) 平面对称变换矩阵：

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

<location data 2>解释为组合变换的幂。<location data 2>是整数对 li, pi 的序列。这个序列将被解释为：

$$L_{l_1}^{p_1}, \cdots, L_{l_n}^{p_n}$$

L_{li}是<location record>部分的变换矩阵。

读取<locations>部分的类为 TopTools_LocationSet，程序代码如下所示：

```
//=======================================================================
//function : Read
//purpose  :
//=======================================================================
void  TopTools_LocationSet::Read(Standard_IStream& IS)
{
  myMap.Clear();

  char buffer[255];
  Standard_Integer ll,p;

  IS >> buffer;
  if (strcmp(buffer,"Locations")) {
    cout << "Not a location table "<<endl;
    return;
  }
```

```
  Standard_Integer i, nbLoc;
  IS >> nbLoc;

  TopLoc_Location L;
  gp_Trsf T;

  //OCC19559
  Message_ProgressSentry PS(GetProgress(), "Locations", 0, nbLoc, 1);
  for (i = 1; i <= nbLoc&& PS.More(); i++, PS.Next()) {
    if ( !GetProgress().IsNull() )
      GetProgress()->Show();

    Standard_Integer typLoc;
    IS >> typLoc;

    if (typLoc == 1) {
      ReadTrsf(T,IS);
      L = T;
    }

    else if (typLoc == 2) {
      L = TopLoc_Location();
      IS >> l1;
      while (l1 != 0) {
    IS >> p;
    TopLoc_Location L1 = myMap(l1);
    L = L1.Powered(p) *L;
    IS >> l1;
      }
    }

    if (!L.IsIdentity()) myMap.Add(L);
  }
}
```

虽然代码风格不好，缩进、括号什么的都不工整，看起来很吃力，但是结合源程序，对上面的详细说明的理解还是很有帮助的。

其中变量 nbLoc 是<location record count>的值，成员变量 myMap 是 TopLoc_Location 的一个 map。当是<location record 1>时把<location data 1>都放到 TopLoc_Location 的 map 中。当是<location record 2>时将 li 的变换矩阵 TopLoc_Location 乘 pi 次方。<flag>0 表示<location data 2>的结束。

四、<geometry>部分

<geometry>包含以下子部分：

1. <2D curves>
2. <3D curves>
3. <3D polygons>
4. <polygons on triangulations>
5. <surfaces>
6. <triangulations>

读取<geometry>部分的类为BRepTools_ShapeSet，程序代码如下所示：

```
//=======================================================================
//function : ReadGeometry
//purpose  :
//=======================================================================
void  BRepTools_ShapeSet::ReadGeometry(Standard_IStream& IS)
{
  //OCC19559
  myCurves2d.SetProgress(GetProgress());
  myCurves.SetProgress(GetProgress());
  mySurfaces.SetProgress(GetProgress());

  if ( !GetProgress().IsNull()) {
    if( GetProgress()->UserBreak() ) return;
    GetProgress()->NewScope ( 15, "2D Curves" );
  }
  myCurves2d.Read(IS);

  if ( !GetProgress().IsNull()) {
    if( GetProgress()->UserBreak() ) return;
    GetProgress()->EndScope();
    GetProgress()->Show();

    GetProgress()->NewScope ( 15, "3D Curves" );
  }
  myCurves.Read(IS);

  if ( !GetProgress().IsNull()) {
    if( GetProgress()->UserBreak() ) return;
    GetProgress()->EndScope();
    GetProgress()->Show();

    GetProgress()->NewScope ( 10, "3D Polygons" );
  }
  ReadPolygon3D(IS);
```

```
  if ( !GetProgress().IsNull() ) {
    if( GetProgress()->UserBreak() ) return;
    GetProgress()->EndScope();
    GetProgress()->Show();

    GetProgress()->NewScope ( 10, "Polygons On Triangulation" );
  }
  ReadPolygonOnTriangulation(IS);
  if ( !GetProgress().IsNull()) {
    if( GetProgress()->UserBreak() ) return;
    GetProgress()->EndScope();
    GetProgress()->Show();

    GetProgress()->NewScope ( 10, "Surfaces" );
  }
  mySurfaces.Read(IS);
  if ( !GetProgress().IsNull() ) {
    if( GetProgress()->UserBreak() ) return;
    GetProgress()->EndScope();
    GetProgress()->Show();

    GetProgress()->NewScope ( 15, "Triangulations" );
  }
  ReadTriangulation(IS);
  if ( !GetProgress().IsNull()) {
    if( GetProgress()->UserBreak() ) return;
    GetProgress()->EndScope();
    GetProgress()->Show();
  }
}
```

4.1 子部分<3D curves>

示例：

```
Curves 13
1 0 0 0 0 0 1
1 0 0 3 -0 1 0
1 0 2 0 0 0 1
1 0 0 0 -0 1 0
1 1 0 0 0 0 1
1 1 0 3 0 1 0
1 1 2 0 0 0 1
1 1 0 0 -0 1 0
1 0 0 0 1 0 -0
1 0 0 3 1 0 -0
1 0 2 0 1 0 -0
1 0 2 3 1 0 -0
1 1 0 0 1 0 0
```

BNF 定义：

BNF-like Definition

<3D curves> = <3D curve header> <_\n> <3D curve records>;

<3D curve header> = “Curves” <_> <3D curve count>;

<3D curve count> = <int>;

<3D curve records> = <3D curve record> ^ <3D curve count>;

<3D curve record> =
<3D curve record 1> |
<3D curve record 2> |
<3D curve record 3> |
<3D curve record 4> |
<3D curve record 5> |
<3D curve record 6> |
<3D curve record 7> |
<3D curve record 8> |
<3D curve record 9>;

详细说明：

由 Curves 开始，后面是曲线的数量，再下面是每条曲线的具体数据。

读取<curves>部分的类为 GeomTools_CurveSet，程序代码如下所示：

```
#define LINE      1
#define CIRCLE    2
#define ELLIPSE   3
#define PARABOLA  4
#define HYPERBOLA 5
#define BEZIER    6
#define BSPLINE   7
```

```
#define TRIMMED   8
#define OFFSET    9
//=======================================================================
//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;
}
```

因为重载了操作符>>，使不同的类调用了不同的处理函数。

因为读取点和方向用得很频繁，所以将读取点和方向的函数程序先列出如下所示：

```
//=======================================================================
//function : ReadPnt
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS, gp_Pnt& P)
{
  Standard_Real X=0.,Y=0.,Z=0.;
  IS >> X >> Y >> Z;
  P.SetCoord(X,Y,Z);
  return IS;
}

//=======================================================================
//function : ReadDir
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS, gp_Dir& D)
{
  Standard_Real X=0.,Y=0.,Z=0.;
  IS >> X >> Y >> Z;
  D.SetCoord(X,Y,Z);
  return IS;
}
```

4.1.1 <3D curve record 1>-Line

示例：

```
1 1 0 3 0 1 0
```

BNF 定义：

BNF-like Definition

<3D curve record 1> = “1” <_> <3D point> <_> <3D direction> <_\n>;

详细说明：

<3D curve record 1>定义了直线。直线数据由一个三维点 P 和一个三维方向矢量 D 组成。通过点 P 且方向为 D 的直线由下面的参数方程来定义：

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

示例数据表示的直线为通过点 P（1，0，3），方向 D（0，1，0），得到的参数方程为：

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

读取直线部分的程序代码如下所示：

```
//=========================================================================
//function : ReadCurve
//purpose  :
//=========================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_Line)& L)
{
  gp_Pnt P(0.,0.,0.);
  gp_Dir AX(1.,0.,0.);
  IS >> P >> AX;
  L = new Geom_Line(P,AX);
  return IS;
}
```

4.1.2 <3D curve record 2>-Circle

示例：

```
2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
```

BNF 定义：

BNF-like Definition

<3D curve record 2> = "2" <_> <3D circle center> <_> <3D circle N> <_> <3D circle Dx> <_> <3D circle Dy> <_> <3D circle radius> <_\n>;

<3D circle center> = <3D point>;

<3D circle N> = <3D direction>;

<3D circle Dx> = <3D direction>;

<3D circle Dy> = <3D direction>;

<3D circle radius> = <real>;

详细说明：

<3D curve record 2>定义了圆。圆的数据包含一个三维点 P，一个正交坐标系的三个轴的方向 N，Dx，Dy，还有一个非负的实数 r。其中点 P 为圆心坐标，圆位于平面的法向量为 N 的平面上，圆的半径为 r。圆的参数方程如下所示：

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

示例数据表示的圆为：圆心 P（1，2，3），位于平面的法向量 N（0，0，1），圆的方向 Dx＝（1，0，－0），Dy＝（－0，1，0），半径 r=4，其参数方向为：

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

读取圆部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_Circle)& C)
{
  gp_Pnt P(0.,0.,0.);
  gp_Dir A(1.,0.,0.),AX(1.,0.,0.),AY(1.,0.,0.);
  Standard_Real R=0.;
  IS >> P >> A >> AX >> AY >> R;
  C = new Geom_Circle(gp_Ax2(P,A,AX),R);
  return IS;
}
```

4.1.3 <3D curve record 3>-Ellipse

示例：

```
3 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
```

BNF 定义：

BNF-like Definition

<3D curve record 3> = "3" <_> <3D ellipse center> <_> <3D ellipse N> <_> <3D ellipse Dmaj> <_> <3D ellipse Dmin> <_> <3D ellipse Rmaj> <_> <3D ellipse Rmin> <_\n>;

<3D ellipse center> = <3D point>;

<3D ellipse N> = <3D direction>;

<3D ellipse Dmaj> = <3D direction>;

<3D ellipse Dmin> = <3D direction>;

<3D ellipse Rmaj> = <real>;

<3D ellipse Rmin> = <real>;

详细说明：

<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，

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

读取椭圆部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_Ellipse)& E)
{
  gp_Pnt P(0.,0.,0.);
  gp_Dir A(1.,0.,0.),AX(1.,0.,0.),AY(1.,0.,0.);
  Standard_Real R1=0.,R2=0.;
  IS >> P >> A >> AX >> AY >> R1 >> R2;
  E = new Geom_Ellipse(gp_Ax2(P,A,AX),R1,R2);
  return IS;
}
```

4.1.4 <3D curve record 4>-Parabola

示例：

```
4 1 2 3 0 0 1 1 0 -0 -0 1 0 16
```

BNF 定义：

BNF-like Definition

<3D curve record 4> = "4" <_> <3D parabola origin> <_> <3D parabola N> <_> <3D parabola Dx> <_> <3D parabola Dy> <_> <3D parabola focal length> <_\n>;

<3D parabola origin> = <3D point>;

<3D parabola N> = <3D direction>;

<3D parabola Dx> = <3D direction>;

<3D parabola Dy> = <3D direction>;

<3D parabola focal length> = <real>;

详细说明：

<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：

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

读取抛物线部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_Parabola)& C)
{
  gp_Pnt P(0.,0.,0.);
  gp_Dir A(1.,0.,0.),AX(1.,0.,0.),AY(1.,0.,0.);
  Standard_Real R1=0.;
  IS >> P >> A >> AX >> AY >> R1;
  C = new Geom_Parabola(gp_Ax2(P,A,AX),R1);
  return IS;
}
```

4.1.5 <3D curve record 5>-Hyperbola

示例：

```
5 1 2 3 0 0 1 1 0 -0 -0 1 0 5 4
```

BNF 定义：

BNF-like Definition

<3D curve record 5> = "5" <_> <3D hyperbola origin> <_> <3D hyperbola N> <_> <3D hyperbola Dx> <_> <3D hyperbola Dy> <_> <3D hyperbola Kx> <_> <3D hyperbola Ky> <_\n>;

<3D hyperbola origin> = <3D point>;

<3D hyperbola N> = <3D direction>;

<3D hyperbola Dx> = <3D direction>;

<3D hyperbola Dy> = <3D direction>;

<3D hyperbola Kx> = <real>;

<3D hyperbola Ky> = <real>;

详细说明：

<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：

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

读取双曲线部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_Hyperbola)& H)
{
  gp_Pnt P(0.,0.,0.);
  gp_Dir A(1.,0.,0.),AX(1.,0.,0.),AY(1.,0.,0.);
  Standard_Real R1=0.,R2=0.;
  IS >> P >> A >> AX >> AY >> R1 >> R2;
  H = new Geom_Hyperbola(gp_Ax2(P,A,AX),R1,R2);
  return IS;
}
```

4.1.6 <3D curve record 6>-Bezier Curve

示例：

```
6 1 2 0 1 0  4 1 -2 0  5 2 3 0  6
```

BNF 定义：

BNF-like Definition

<3D curve record 6> = "6" <_> <3D Bezier rational flag> <_> <3D Bezier degree> <3D Bezier weight poles> <_\n>;

<3D Bezier rational flag> = <flag>;

<3D Bezier degree> = <int>;

<3D Bezier weight poles> = (<_> <3D Bezier weight pole>) ^ (<3D Bezier degree> <+> "1");

<3D Bezier weight pole> = <3D point> [<_> <real>];

详细说明：

<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：

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

读取 Bezier 曲线部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_BezierCurve)& B)
{
  Standard_Boolean rational=Standard_False;
```

```
  IS >> rational;

  // poles and weights
  Standard_Integer i=0,degree=0;
  IS >> degree;

  TColgp_Array1OfPnt poles(1,degree+1);
  TColStd_Array1OfReal weights(1,degree+1);

  for (i = 1; i <= degree+1; i++) {
    IS >> poles(i);
    if (rational)
      IS >> weights(i);
  }

  if (rational)
    B = new Geom_BezierCurve(poles,weights);
  else
    B = new Geom_BezierCurve(poles);

  return IS;
}
```

4.1.7 <3D curve record 7>-B-Spline curve

示例：

```
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
```

BNF 定义：

BNF-like Definition

<3D curve record 7> = "7" <_> <3D B-spline rational flag> <_> "0" <_> <3D B-spline degree> <_> <3D B-spline pole count> <_> <3D B-spline multiplicity knot count> <3D B-spline weight poles> <_\n> <3D B-spline multiplicity knots> <_\n>;

<3D B-spline rational flag> = <flag>;

<3D B-spline degree> = <int>;

<3D B-spline pole count> = <int>;

<3D B-spline multiplicity knot count> = <int>;

<3D B-spline weight poles> = (<_> <3D B-spline weight pole>) ^ <3D B-spline pole count>;

<3D B-spline weight pole> = <3D point> [<_> <real>];

<3D B-spline multiplicity knots> =
(<_> <3D B-spline multiplicity knot>) ^ <3D B-spline multiplicity knot count>;

<3D B-spline multiplicity knot> = <real> <_> <int>;

详细说明：

<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，重节点数 k，带权控制点 wieght poles 和重节点 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，重节点数 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；重节点 u1＝0，q1＝1，u2＝0.25，q2＝1，u3＝0.5，q3＝1，u4＝0.75，q4＝1，u5＝1，q5＝1。B-Spline 曲线的参数方程如下所示：

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

读取 B-Spline 曲线部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_BSplineCurve)& B)
{

  Standard_Boolean rational=Standard_False,periodic=Standard_False;
  IS >> rational >> periodic;

  // poles and weights
  Standard_Integer i=0,degree=0,nbpoles=0,nbknots=0;
  IS >> degree >> nbpoles >> nbknots;

  TColgp_Array1OfPnt poles(1,nbpoles);
  TColStd_Array1OfReal weights(1,nbpoles);

  for (i = 1; i <= nbpoles; i++) {
    IS >> poles(i);
    if (rational)
      IS >> weights(i);
  }

  TColStd_Array1OfReal knots(1,nbknots);
  TColStd_Array1OfInteger mults(1,nbknots);
```

```
  for (i = 1; i <= nbknots; i++) {
    IS >> knots(i) >> mults(i);
  }

  if (rational)
    B = new Geom_BSplineCurve(poles,weights,knots,mults,degree,periodic);
  else
    B = new Geom_BSplineCurve(poles,knots,mults,degree,periodic);

  return IS;
}
```

4.1.8 <3D curve record 8>-Trimmed Curve

示例：

```
8  4  -5
1  1  2  3  1  0  0
```

BNF 定义：

BNF-like Definition

<3D curve record 8> = "8" <_> <3D trimmed curve u min> <_> <3D trimmed curve u max> <_\n>
<3D curve record>;

<3D trimmed curve u min> = <real>;

<3D trimmed curve u max> = <real>;

详细说明：

<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：

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

读取裁剪曲线部分的程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================

static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_TrimmedCurve)& C)
{
  Standard_Real p1=0.,p2=0.;
  IS >> p1 >> p2;
  Handle(Geom_Curve) BC;
  GeomTools_CurveSet::ReadCurve(IS,BC);
  C = new Geom_TrimmedCurve(BC,p1,p2);
  return IS;
}
```

4.1.9 <3D curve record 9>-Offset Curve

示例：

```
9 2
0 1 0
1 1 2 3 1 0 0
```

BNF 定义：

BNF-like Definition

<3D curve record 9> = "9" <_> <3D offset curve distance> <_\n>
<3D offset curve direction> <_\n>
<3D curve record>;

<3D offset curve distance> = <real>;

<3D offset curve direction> = <3D direction>;

详细说明：

<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：

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

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：

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

读取偏移曲线部分程序代码如下所示：

```
//=======================================================================
//function : ReadCurve
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_OffsetCurve)& C)
{
  Standard_Real p=0.;
  IS >> p;
  gp_Dir D(1.,0.,0.);
  IS >> D;
```

```
  Handle(Geom_Curve) BC;
  GeomTools_CurveSet::ReadCurve(IS,BC);
  C = new Geom_OffsetCurve(BC,p,D);
  return IS;
}
```

4.2 <surface>子部分

示例：

```
Surfaces 6
1 0 0 0 1 0 -0 0 0 1 0 -1 0
1 0 0 0 -0 1 0 0 0 1 1 0 -0
1 0 0 3 0 0 1 1 0 -0 -0 1 0
1 0 2 0 -0 1 0 0 0 1 1 0 -0
1 0 0 0 0 0 1 1 0 -0 -0 1 0
1 1 0 0 1 0 -0 0 0 1 0 -1 0
```

BNF 定义：

BNF-like Definition

<surfaces> = <surface header> <_\n> <surface records>;

<surface header> = “Surfaces” <_> <surface count>;

<surface records> = <surface record> ^ <surface count>;

<surface record> =
<surface record 1> |
<surface record 2> |
<surface record 3> |
<surface record 4> |
<surface record 5> |
<surface record 6> |
<surface record 7> |
<surface record 8> |
<surface record 9> |
<surface record 10> |
<surface record 11>;

读取<surface>部分的程序代码如下所示：

```
#define PLANE           1
#define CYLINDER        2
#define CONE            3
#define SPHERE          4
#define TORUS           5
#define LINEAREXTRUSION 6
#define REVOLUTION      7
#define BEZIER          8
#define BSPLINE         9
#define RECTANGULAR     10
```

```
#define OFFSET          11

//=======================================================================
//function : ReadSurface
//purpose  :
//=======================================================================
Standard_IStream& GeomTools_SurfaceSet::ReadSurface(Standard_IStream& IS,
                                                    Handle(Geom_Surface)& S)
{
  Standard_Integer stype;

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

    case PLANE :
      {
        Handle(Geom_Plane) SS;
        IS >> SS;
        S = SS;
      }
      break;

    case CYLINDER :
      {
        Handle(Geom_CylindricalSurface) SS;
        IS >> SS;
        S = SS;
      }
      break;

    case CONE :
      {
        Handle(Geom_ConicalSurface) SS;
        IS >> SS;
        S = SS;
      }
      break;

    case SPHERE :
      {
        Handle(Geom_SphericalSurface) SS;
        IS >> SS;
```

```
      S = SS;
    }
    break;

  case TORUS :
    {
      Handle(Geom_ToroidalSurface) SS;
      IS >> SS;
      S = SS;
    }
    break;

  case LINEAREXTRUSION :
    {
      Handle(Geom_SurfaceOfLinearExtrusion) SS;
      IS >> SS;
      S = SS;
    }
    break;

  case REVOLUTION :
    {
      Handle(Geom_SurfaceOfRevolution) SS;
      IS >> SS;
      S = SS;
    }
    break;

  case BEZIER :
    {
      Handle(Geom_BezierSurface) SS;
      IS >> SS;
      S = SS;
    }
    break;

  case BSPLINE :
    {
      Handle(Geom_BSplineSurface) SS;
      IS >> SS;
      S = SS;
    }
    break;
```

```
    case RECTANGULAR :
      {
        Handle(Geom_RectangularTrimmedSurface) SS;
        IS >> SS;
        S = SS;
      }
      break;

    case OFFSET :
      {
        Handle(Geom_OffsetSurface) SS;
        IS >> SS;
        S = SS;
      }
      break;

    default :
      {
        Handle(Geom_Surface) SS;
        GeomTools::GetUndefinedTypeHandler()->ReadSurface(stype,IS,SS);
        S = SS;
      }
      break;
    }
  }
  catch(Standard_Failure) {
#ifdef DEB
    Handle(Standard_Failure) anExc = Standard_Failure::Caught();
    cout <<"EXCEPTION in GeomTools_SurfaceSet::ReadSurface(..)!!!" << endl;
    cout << anExc << endl;
#endif
    S = NULL;
  }
  return IS;
}
```

4.2.1 <surface record 1>-Plane

示例：

```
1 0 0 3 0 0 1 1 0 -0 -0 1 0
```

BNF 定义：

BNF-like Definition

<surface record 1> = "1" <_> <3D point> (<_> <3D direction>) ^ 3 <_\n>;

详细说明：

<surface record 1>定义了平面。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：

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

示例数据表示的平面为通过点 P=（0，0，3），法向量 N=（0，0，1），其参数方程如下所示：

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

读取平面部分的程序代码如下所示：

```
//=======================================================================
//function : ReadAx3
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS, gp_Ax3& A3)
{
  gp_Pnt P(0.,0.,0.);
  gp_Dir A(1.,0.,0.),AX(1.,0.,0.),AY(1.,0.,0.);
  IS >> P >> A >> AX >> AY;
  gp_Ax3 ax3(P,A,AX);
  if (AY.DotCross(A,AX) < 0)
    ax3.YReverse();
  A3 = ax3;
  return IS;
}

static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_Plane)& S)
{
  gp_Ax3 A;
  IS >> A;
  S = new Geom_Plane(A);
  return IS;
}
```

4.2.2 <surface record 2>-Cylinder

示例：

```
2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
```

BNF 定义：

BNF-like Definition

<surface record 2> = "2" <_> <3D point> (<_> <3D direction>) ^ 3 <_> <real> <_\n>;

详细说明：

<surface record 2>定义了圆柱面。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：

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

示例数据表示的圆柱面为轴通过点 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，其参数方程如下所示：

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

读取圆柱面部分的程序代码如下所示：

```
//=======================================================================
//function : operator>>
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_CylindricalSurface)& S)
{
  gp_Ax3 A;
  Standard_Real R=0.;
  IS >> A  >> R;
  S = new Geom_CylindricalSurface(A,R);
  return IS;
}
```

4.2.3 <surface record 3>-Cone

示例：

```
3 1 2 3 0 0 1 1 0 -0 -0 1 0 4
0.75
```

BNF 定义：

BNF-like Definition

<surface record 3> = "3" <_> <3D point> (<_> <3D direction>) ^ 3 (<_> <real>) ^ 2 <_\n>;

详细说明：

<surface record 3>定义了圆锥面。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：

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

示例数据表示的圆锥面的轴通过点 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。其参数方程如下所示：

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

读取圆锥面部分的程序代码如下所示：

```
//=========================================================================
//function : operator>>
//purpose  :
//=========================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                                    Handle(Geom_ConicalSurface)& S)
{
  gp_Ax3 A;
  Standard_Real R=0.,Ang=0.;
  IS >> A >> R >> Ang;
  S = new Geom_ConicalSurface(A,Ang,R);
  return IS;
}
```

4.2.4 <surface record 4>-Sphere

示例：

```
4 1 2 3 0 0 1 1 0 -0 -0 1 0 4
```

BNF 定义：

BNF-like Definition

<surface record 4> = "4" <_> <3D point> (<_> <3D direction>) ^ 3 <_> <real> <_\n>;

详细说明：

<surface record 4>定义了球面。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：

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

示例数据表示的球面为球心过点 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。其参数方程如下所示：

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

读取球面部分的程序代码如下所示：

```
//=======================================================================
//function : operator>>
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_SphericalSurface)& S)
{
  gp_Ax3 A;
  Standard_Real R=0.;
  IS >> A >> R;
  S = new Geom_SphericalSurface(A,R);
  return IS;
}
```

4.2.5 <surface record 5>-Torus

示例：

```
 5  1  2  3  0  0  1  1  0  -0  -0  1  0  8  4
```

BNF 定义：

BNF-like Definition

<surface record 5> = "5" <_> <3D point> (<_> <3D direction>) ^ 3 (<_> <real>) ^ 2 <_\n>;

详细说明：

<surface record 5>定义了圆环面。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：

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

示例数据表示的圆环面的轴通过点 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：

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

读取圆环面部分的程序代码如下所示：

```
//=========================================================================
//function : operator>>
//purpose  :
//=========================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_ToroidalSurface)& S)
{
  gp_Ax3 A;
  Standard_Real R1=0.,R2=0.;
  IS >> A >> R1 >> R2;
  S = new Geom_ToroidalSurface(A,R1,R2);
  return IS;
}
```

4.2.6 <surface record 6>-Linear Extrusion

示例：

```
6 0 0.6 0.8
2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
```

BNF 定义：

BNF-like Definition

<surface record 6> = “6” <_> <3D direction> <_\n> <3D curve record>;

详细说明：

<surface record 6>定义了线性拉伸面。线性拉伸面的数据包含三维方向 Dv 和三维曲线<3D curve record>。其参数方程如下所示：

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

示例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。拉伸面的参数方程如下所示：

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

读取线性拉伸面部分的程序代码如下所示：

```
//=======================================================================
//function : operator>>
//purpose  :
//=======================================================================

static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_SurfaceOfLinearExtrusion)& S)
{
  gp_Dir D(1.,0.,0.);
  Handle(Geom_Curve) C;
  IS >> D;
  GeomTools_CurveSet::ReadCurve(IS,C);
  S = new Geom_SurfaceOfLinearExtrusion(C,D);
  return IS;
}
```

4.2.7 <surface record 7>-Revolution Surface

示例：

```
7 -4 0 3 0 1 0
2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
```

BNF 定义：

BNF-like Definition

<surface record 7> = "7" <_> <3D point> <_> <3D direction> <_\n> <3D curve record>;

详细说明：

<surface record 7>定义了旋转曲面。旋转曲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如下所示：

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

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

示例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。其参数方程如下所示：

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

读取旋转曲面部分的程序代码如下所示：

```
//=========================================================================
//function : operator>>
//purpose  :
//=========================================================================

static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_SurfaceOfRevolution)& S)
{
  gp_Pnt P(0.,0.,0.);
  gp_Dir D(1.,0.,0.);
  Handle(Geom_Curve) C;
  IS >> P >> D;
  GeomTools_CurveSet::ReadCurve(IS,C);
  S = new Geom_SurfaceOfRevolution(C,gp_Ax1(P,D));
  return IS;
}
```

4.2.8 <surface record 8>-Bezier Surface

示例：

```
8 1 1 2 1 0 0 1   7 1 0 -4   10
0 1 -2   8 1 1 5   11
0 2 3   9 1 2 6   12
```

BNF 定义：

BNF-like Definition

<surface record 8> = "8" <_> <Bezier surface u rational flag> <_> <Bezier surface v rational flag> <_> <Bezier surface u degree> <_> <Bezier surface v degree> <_> <Bezier surface weight poles>;

<Bezier surface u rational flag> = <flag>;

<Bezier surface v rational flag> = <flag>;

<Bezier surface u degree> = <int>;

<Bezier surface v degree> = <int>;

<Bezier surface weight poles> =
(<Bezier surface weight pole group> <_\n>) ^ (<Bezier surface u degree> <+> "1");

<Bezier surface weight pole group> = <Bezier surface weight pole>
(<_> <Bezier surface weight pole>) ^ <Bezier surface v degree>;

<Bezier surface weight pole> = <3D point> [<_> <real>];

详细说明：

<surface record 8>定义了 Bezier 曲面。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 曲面的参数方程如下所示：

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

示例数据表示的 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：

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

读取 Bezier 曲面部分的程序代码如下所示：

```
//=======================================================================
//function : operator>>
//purpose  :
//=======================================================================

static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_BezierSurface)& S)
{
  Standard_Boolean urational=Standard_False, vrational=Standard_False;
  IS >> urational >> vrational;
  Standard_Integer udegree=0, vdegree=0;
  IS >> udegree >> vdegree;
  TColgp_Array2OfPnt poles(1,udegree+1,1,vdegree+1);
  TColStd_Array2OfReal weights(1,udegree+1,1,vdegree+1);

  Standard_Integer i,j;
  for (i = 1; i <= udegree+1; i++) {
    for (j = 1; j <= vdegree+1; j++) {
      IS >> poles(i,j);
      if (urational || vrational)
    IS >> weights(i,j);
    }
  }

  if (urational || vrational)
    S = new Geom_BezierSurface(poles,weights);
  else
    S = new Geom_BezierSurface(poles);
  return IS;
}
```

4.2.9 <surface record 9>-B-Spline Surface

示例：

```
9 1 1 0 0 1 1 3 2 5 4 0 0 1  7 1 0 -4  10
0 1 -2  8 1 1 5  11
0 2 3  9 1 2 6  12

0 1
0.25 1
0.5 1
0.75 1
1 1

0 1
0.3 1
0.7 1
1 1
```

BNF 定义：

BNF-like Definition

<surface record 9> = "9" <_> <B-spline surface u rational flag> <_> <B-spline surface v rational flag> <_> "0" <_> "0" <_> <B-spline surface u degree> <_> <B-spline surface v degree> <_> <B-spline surface u pole count> <_> <B-spline surface v pole count> <_> <B-spline surface u multiplicity knot count> <_> <B-spline surface v multiplicity knot count> <_> <B-spline surface weight poles> <_\n> <B-spline surface u multiplicity knots> <_\n> <B-spline surface v multiplicity knots>;

<B-spline surface u rational flag> = <flag>;

<B-spline surface v rational flag> = <flag>;

<B-spline surface u degree> = <int>;

<B-spline surface v degree> = <int>;

<B-spline surface u pole count> = <int>;

<B-spline surface v pole count> = <int>;

<B-spline surface u multiplicity knot count> = <int>;

<B-spline surface v multiplicity knot count> = <int>;

<B-spline surface weight poles> =
(<B-spline surface weight pole group> <_\n>) ^ <B-spline surface u pole count>;

<B-spline surface weight pole group> =
(<B-spline surface weight pole> <_>) ^ <B-spline surface v pole count>;

<B-spline surface weight pole> = <3D point> [<_> <real>];

<B-spline surface u multiplicity knots> =
(<B-spline surface u multiplicity knot> <_\n>) ^ <B-spline surface u multiplicity knot count>;

<B-spline surface u multiplicity knot> = <real> <_> <int>;

<B-spline surface v multiplicity knots> =
(<B-spline surface v multiplicity knot> <_\n>) ^ <B-spline surface v multiplicity knot count>;

<B-spline surface v multiplicity knot> = <real> <_> <int>;

详细说明：

<surface record 9>定义了 B-Spline 曲面。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 k_u 对：$u_1,q_1,...,u_{ku},q_{ku}$。这里 u_i 是重数为 q_i>=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 k_v 对：$u_1,q_1,...,u_{kv},q_{kv}$。这里 v_i 是重数为 q_i>=1 的节点：

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

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

B-Spline 曲面的参数方程如下所示：

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

基函数 Ni,j 和 Mi,j 有如下的递归定义：

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

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

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

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

示例数据表示的 B-Spline 曲面为：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：

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

读取 B-Spline 曲面部分的程序代码如下所示：

```
//=======================================================================
//function : operator>>
//purpose  :
//=======================================================================
static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_BSplineSurface)& S)
{
  Standard_Boolean urational=Standard_False, vrational=Standard_False,
                   uperiodic=Standard_False, vperiodic=Standard_False;
  IS >> urational >> vrational;
  IS >> uperiodic >> vperiodic;
  Standard_Integer                                                udegree=0,
vdegree=0,nbupoles=0,nbvpoles=0,nbuknots=0,nbvknots=0;
  IS >> udegree >> vdegree;
  IS >> nbupoles >> nbvpoles;
  IS >> nbuknots >> nbvknots;

  TColgp_Array2OfPnt poles(1,nbupoles,1,nbvpoles);
  TColStd_Array2OfReal weights(1,nbupoles,1,nbvpoles);

  Standard_Integer i,j;
  for (i = 1; i <= nbupoles; i++) {
    for (j = 1; j <= nbvpoles; j++) {
      IS >> poles(i,j);
      if (urational || vrational)
    IS >> weights(i,j);
```

```
    }
  }

  TColStd_Array1OfReal uknots(1,nbuknots);
  TColStd_Array1OfInteger umults(1,nbuknots);
  for (i = 1; i <= nbuknots; i++) {
    IS >> uknots(i) >> umults(i);
  }

  TColStd_Array1OfReal vknots(1,nbvknots);
  TColStd_Array1OfInteger vmults(1,nbvknots);
  for (i = 1; i <= nbvknots; i++) {
    IS >> vknots(i) >> vmults(i);
  }

  if (urational || vrational)
    S = new Geom_BSplineSurface(poles,weights,uknots,vknots,umults,vmults,
                udegree,vdegree,uperiodic,vperiodic);
  else
    S = new Geom_BSplineSurface(poles,uknots,vknots,umults,vmults,
                udegree,vdegree,uperiodic,vperiodic);
  return IS;
}
```

4.2.10 <surface recored 10>-Rectangular Trim Surface

示例：

```
10 -1 2 -3 4
1 1 2 3 0 0 1 1 0 -0 -0 1 0
```

BNF 定义：

BNF-like Definition

<surface record 10> = "10" <_> <trim surface u min> <_> <trim surface u max> <_> <trim surface v min> <_> <trim surface v max> <_\n> <surface record>;

<trim surface u min> = <real>;

<trim surface u max> = <real>;

<trim surface v min> = <real>;

<trim surface v max> = <real>;

详细说明：

<surface record 10>定义了矩形裁剪曲面。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。曲面的参数方程如下所示：

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

示例数据表示的矩形裁剪曲面的矩形裁剪区域为[-1，2]x[－3，4]，被裁剪曲面 B（u，v）＝（1，2，3）+u（1，0，0）+v（0，1，0）。其参数方程如下所示：

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

读取矩形裁剪曲面部分的程序代码如下所示：

```
//=======================================================================
//function : operator>>
//purpose  :
//=======================================================================

static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_RectangularTrimmedSurface)& S)
{
  Standard_Real U1=0.,U2=0.,V1=0.,V2=0.;
  IS >> U1 >> U2 >> V1 >> V2;
  Handle(Geom_Surface) BS;
  GeomTools_SurfaceSet::ReadSurface(IS,BS);
  S = new Geom_RectangularTrimmedSurface(BS,U1,U2,V1,V2);
  return IS;
}
```

4.2.11 <surface record 11>-Offset Surface

示例：

```
11 -2
1 1 2 3 0 0 1 1 0 -0 -0 1 0
```

BNF 定义：

BNF-like Definition

<surface record 11> = “11” <_> <surface record distance> <_\n> <surface record>;

<surface record distance> = <real>;

详细说明：

<surface record 11>定义了偏移曲面。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如下所示：

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

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

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

示例数据表示的偏移曲面的偏移距离 d＝－2，基准曲面 B（u，v）＝（1，2，3）+u（1，0，0）+v（0，1，0）。其参数方程如下所示：

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

读取偏移曲面部分的程序代码如下所示：

```
//=======================================================================
//function : operator>>
//purpose  :
//=======================================================================

static Standard_IStream& operator>>(Standard_IStream& IS,
                    Handle(Geom_OffsetSurface)& S)
{
  Standard_Real O=0.;
  IS >> O;
  Handle(Geom_Surface) BS;
  GeomTools_SurfaceSet::ReadSurface(IS,BS);
  S = new Geom_OffsetSurface(BS,O);
  return IS;
}
```

4.3 <2D curves>子部分

因与<3D curves>部分基本相同，略去此部分，请参考原文。

4.4 <3D polygons>子部分

示例：

```
Polygon3D 1
2 1
0.1
1 0 0 2 0 0
0 1
```

BNF 定义：

BNF-like Definition

<3D polygons> = <3D polygon header> <_\n> <3D polygon records>;

<3D polygon header> = “Polygon3D” <_> <3D polygon record count>;

<3D polygon records> = <3D polygon record> ^ <3D polygon record count>;

<3D polygon record> =
<3D polygon node count> <_> <3D polygon flag of parameter presence> <_\n>
<3D polygon deflection> <_\n>
<3D polygon nodes> <_\n>
[<3D polygon parameters> <_\n>];

<3D polygon node count> = <int>;

<3D polygon flag of parameter presence> = <flag>;

<3D polygon deflection> = <real>;

<3D polygon nodes> = (<3D polygon node> <_>) ^ <3D polygon node count>;

<3D polygon node> = <3D point>;

<3D polygon u parameters> = (<3D polygon u parameter> <_>) ^ <3D polygon node count>;

<3D polygon u parameter> = <real>;

详细说明：

<3D polygon record>定义了空间多段线（3D polyline）L，用来逼近空间曲线 C。多段线的数据包含节点数 m>=2，参数显示标志位 p，逼近偏差（deflection）d>=0，节点 Ni（1=<i<=m），参数 ui（1=<i<=m）。当参数显示标志位 p＝1 时，参数 u 才会显示。多段线 L 通过这些节点，多段线 L 逼近曲线 C 的逼近偏差定义如下所示：

$$\max_{P\in C}\min_{Q\in L}|Q-P|\le d\,.$$

参数 ui（1=<i<=m）是曲线 C 上通过节点 Ni 的参数值：

$$C(u_i)=N_i\,.$$

示例数据表示的多段线为：m＝2，参数显示标志位 p＝1，逼近偏差 d＝0.1，节点 N1＝（1，0，0），N2＝（2，0，0），参数 u1＝0，u2＝1。

读取 polygons 部分的程序代码如下所示：

```
//=======================================================================
//function : ReadPolygon3D
//purpose  :
//=======================================================================
void BRepTools_ShapeSet::ReadPolygon3D(Standard_IStream& IS)
{
  char buffer[255];
  //  Standard_Integer i, j, p, val, nbpol, nbnodes, hasparameters;
  Standard_Integer i, j, p, nbpol=0, nbnodes =0, hasparameters = Standard_False;
  Standard_Real d, x, y, z;

  IS >> buffer;
  if (strstr(buffer,"Polygon3D") == NULL) return;
  Handle(Poly_Polygon3D) P;
  IS >> nbpol;
  //OCC19559
  Handle(Message_ProgressIndicator) progress = GetProgress();
  Message_ProgressSentry PS(progress, "3D Polygons", 0, nbpol, 1);
  for (i=1; i<=nbpol && PS.More(); i++, PS.Next()) {
    if ( !progress.IsNull() )
      progress->Show();

    IS >> nbnodes;
    IS >> hasparameters;
    TColgp_Array1OfPnt Nodes(1, nbnodes);
    IS >> d;
    for (j = 1; j <= nbnodes; j++) {
      IS >> x >> y >> z;
      Nodes(j).SetCoord(x,y,z);
    }
    if (hasparameters) {
      TColStd_Array1OfReal Param(1,nbnodes);
      for (p = 1; p <= nbnodes; p++) {
        IS >> Param(p);
      }
      P = new Poly_Polygon3D(Nodes, Param);
    }
    else P = new Poly_Polygon3D(Nodes);
    P->Deflection(d);
    myPolygons3D.Add(P);
  }
}
```

4.5 <triangulations>子部分

示例：

```
Triangulations 6
4 2 1 0
0 0 0 0 0 3 0 2 3 0 2 0 0 0 3 0 3 -2 0 -2 2 4 3 2 1 4
4 2 1 0
0 0 0 1 0 0 1 0 3 0 0 3 0 0 0 1 3 1 3 0 3 2 1 3 1 4
4 2 1 0
0 0 3 0 2 3 1 2 3 1 0 3 0 0 0 2 1 2 1 0 3 2 1 3 1 4
4 2 1 0
0 2 0 1 2 0 1 2 3 0 2 3 0 0 0 1 3 1 3 0 3 2 1 3 1 4
4 2 1 0
0 0 0 0 2 0 1 2 0 1 0 0 0 0 0 2 1 2 1 0 3 2 1 3 1 4
4 2 1 0
1 0 0 1 0 3 1 2 3 1 2 0 0 0 3 0 3 -2 0 -2 2 4 3 2 1 4
```

BNF 定义：

BNF-like Definition

<triangulations> = <triangulation header> <_\n> <triangulation records>;

<triangulation header> = "Triangulations" <_> <triangulation count>;

<triangulation records> = <triangulation record> ^ <triangulation count>;

<triangulation record> = <triangulation node count> <_> <triangulation triangle count> <_> <triangulation parameter presence flag> <_> <triangulation deflection> <_\n> <triangulation nodes> [<_> <triangulation u v parameters>] <_> <triangulation triangles> <_\n>;

<triangulation node count> = <int>;

<triangulation triangle count> = <int>;

<triangulation parameter presence flag> = <flag>;

<triangulation deflection> = <real>;

<triangulation nodes> = (<triangulation node> <_>) ^ <triangulation node count>;

<triangulation node> = <3D point>;

<triangulation u v parameters> =
(<triangulation u v parameter pair> <_>) ^ <triangulation node count>;

<triangulation u v parameter pair> = <real> <_> <real>;

<triangulation triangles> = (<triangulation triangle> <_>) ^ <triangulation triangle count>;

<triangulation triangle> = <int> <_> <int> <_> <int>.

详细说明：

<triangulation record>定义了逼近曲面 S 的三角剖分 T（triangulation）。三角剖分的数据包含节点数 m>=3，三角形数 k>=1，参数显示标志位 p，逼近偏差 d>=0，节点 Ni（1<=i<=m），参数对 ui,vi（1<=i<=m），三角形 nj,1，nj,2，nj,3。参数只有当参数显示标志位 p＝1 时才显示。三角剖分逼近曲面的偏差 d 定义如下所示：

$$\max_{P\in S}\min_{Q\in T}|Q-P|\le d\,.$$

参数对 ui，vi 描述了曲面 S 上过节点 Ni 的参数：

$$S(u_i, v_i) = N_i .$$

三角形 nj,1， nj,2， nj,3 用来取得三角形的三个顶点值 Nnj,1，Nnj,2，Nnj,3，节点遍历的顺序就是 Nnj,1，Nnj,2，Nnj,3。从三角剖分 T 的任意一侧遍历，所有三角形都有相同的方向：顺时针或逆时针。

三角剖分中的三角形数据：

```
4 2 1 0
0 0 0 0 0 3 0 2 3 0 2 0 0 0 3 0 3 -2 0 -2 2 4 3 2 1 4
```

表示的三角剖分为：m＝4 个节点，k＝2 个三角形，参数显示标志位 p＝1，逼近偏差 d＝0，节点 N1（0，0，0），N2（0，0，3），N3（0，2，3），N4（0，2，0），参数值（u1,v1）＝（0，0），（u2,v2）＝（3，0），（u3,v3）＝（3，－2），（u4,v4）＝（0，－2）。从点（1，0，0）((－1，0，0))，三角形是顺时针（逆时针）的。

读取三角剖分部分的程序代码如下所示：

```
//=======================================================================
//function : ReadTriangulation
//purpose  :
//=======================================================================
void BRepTools_ShapeSet::ReadTriangulation(Standard_IStream& IS)
{
  char buffer[255];
  //  Standard_Integer i, j, val, nbtri;
  Standard_Integer i, j, nbtri =0;
  Standard_Real d, x, y, z;
  Standard_Integer nbNodes =0, nbTriangles=0;
  Standard_Boolean hasUV= Standard_False;

  Handle(Poly_Triangulation) T;

  IS >> buffer;
  if (strstr(buffer,"Triangulations") == NULL) return;

  IS >> nbtri;
  //OCC19559
  Handle(Message_ProgressIndicator) progress = GetProgress();
  Message_ProgressSentry PS(progress, "Triangulations", 0, nbtri, 1);
  for (i=1; i<=nbtri && PS.More();i++, PS.Next()) {
    if ( !progress.IsNull() )
      progress->Show();

    IS >> nbNodes >> nbTriangles >> hasUV;
    IS >> d;
```

```
    TColgp_Array1OfPnt Nodes(1, nbNodes);
    TColgp_Array1OfPnt2d UVNodes(1, nbNodes);

    for (j = 1; j <= nbNodes; j++) {
      IS >> x >> y >> z;
      Nodes(j).SetCoord(x,y,z);
    }

    if (hasUV) {
      for (j = 1; j <= nbNodes; j++) {
        IS >> x >> y;
        UVNodes(j).SetCoord(x,y);
      }
    }

    // read the triangles
    Standard_Integer n1,n2,n3;
    Poly_Array1OfTriangle Triangles(1, nbTriangles);
    for (j = 1; j <= nbTriangles; j++) {
      IS >> n1 >> n2 >> n3;
      Triangles(j).Set(n1,n2,n3);
    }

    if (hasUV) T =  new Poly_Triangulation(Nodes,UVNodes,Triangles);
    else T = new Poly_Triangulation(Nodes,Triangles);

    T->Deflection(d);

    myTriangulations.Add(T);
  }
}
```

4.6 <polygons on triangulations>子部分示例：

```
PolygonOnTriangulations 24
2 1 2
p 0.1 1 0 3
2 1 4
p 0.1 1 0 3
2 2 3
p 0.1 1 0 2
2 1 2
p 0.1 1 0 2
2 4 3
p 0.1 1 0 3
2 1 4
p 0.1 1 0 3
2 1 4
p 0.1 1 0 2
2 1 2
p 0.1 1 0 2
2 1 2
p 0.1 1 0 3
2 2 3
p 0.1 1 0 3
2 2 3
p 0.1 1 0 2
2 4 3
p 0.1 1 0 2
2 4 3
p 0.1 1 0 3
2 2 3
p 0.1 1 0 3
2 1 4
p 0.1 1 0 2
2 4 3
p 0.1 1 0 2
2 1 2
p 0.1 1 0 1
2 1 4
p 0.1 1 0 1
2 4 3
p 0.1 1 0 1
2 1 4
p 0.1 1 0 1
2 1 2
p 0.1 1 0 1
2 2 3
p 0.1 1 0 1
2 4 3
p 0.1 1 0 1
2 2 3
p 0.1 1 0 1
```

BNF 定义：

BNF-like Definition

<polygons on triangulations> = <polygons on triangulations header> <_\n>
<polygons on triangulations records>;

<polygons on triangulations header> =
"PolygonOnTriangulations" <_> <polygons on triangulations record count>;

<polygons on triangulations record count> = <int>;

<polygons on triangulations records> =
<polygons on triangulations record> ^ <polygons on triangulations record count>;

<polygons on triangulations record> =
<polygons on triangulations node count> <_> <polygons on triangulations node numbers> <_\n>
"p" <_> <polygons on triangulations deflection> <_>
<polygons on triangulations parameter presence flag>
[<_> <polygons on triangulations u parameters>] <_\n>;

<polygons on triangulations node count> = <int>;

<polygons on triangulations node numbers> =
<polygons on triangulations node number> ^ <polygons on triangulations node count>;

<polygons on triangulations node number> = <int>;

<polygons on triangulations deflection> = <real>;

<polygons on triangulations parameter presence flag> = <flag>;

<polygons on triangulations u parameters> =
(<polygons on triangulations u parameter> <_>) ^ <polygons on triangulations node count>;

<polygons on triangulations u parameter> = <real>;

详细说明：

<polygons on triangulations>定义三角剖分上逼近曲线 C 的多段线 L。多段线 L 的数据包含节点数 m>=2，节点号 ni>=1，逼近偏差 d>=0，参数显示标志位 p 和参数 ui（1=<i<=m）。参数只有在参数显示标志位 p＝1 时才显示。多段线 L 逼近曲线 C 的偏差 d 的定义为：

$$\max_{P\in C}\min_{Q\in L}|Q-P|\le d\ .$$

参数 ui（1<=i<=m）是曲线 C 上的第 ni 个节点的参数。

读取多段线部分的程序代码如下所示：

```
//=======================================================================
//function : ReadPolygonOnTriangulation
//purpose  :
//=======================================================================
void BRepTools_ShapeSet::ReadPolygonOnTriangulation(Standard_IStream& IS)
{
  char buffer[255];
  IS >> buffer;
  if (strstr(buffer,"PolygonOnTriangulations") == NULL) return;
  Standard_Integer i, j, val, nbpol = 0, nbnodes =0;
```

```
  Standard_Integer hasparameters;
  Standard_Real par;
  Handle(TColStd_HArray1OfReal) Param;
  Handle(Poly_PolygonOnTriangulation) Poly;
  IS >> nbpol;
  //OCC19559
  Handle(Message_ProgressIndicator) progress = GetProgress();
  Message_ProgressSentry PS(progress, "Polygons On Triangulation", 0, nbpol, 1);
  for (i=1; i<=nbpol&& PS.More(); i++, PS.Next()) {
    if ( !progress.IsNull() )
      progress->Show();

    IS >> nbnodes;
    TColStd_Array1OfInteger Nodes(1, nbnodes);
    for (j = 1; j <= nbnodes; j++) {
      IS >> val;
      Nodes(j) = val;
    }
    IS >> buffer;
//      if (!strcasecmp(buffer, "p")) {
      Standard_Real def;
      IS >> def;
      IS >> hasparameters;
      if (hasparameters) {
        TColStd_Array1OfReal Param1(1, nbnodes);
        for (j = 1; j <= nbnodes; j++) {
          IS >> par;
          Param1(j) = par;
        }
        Poly = new Poly_PolygonOnTriangulation(Nodes, Param1);
      }
      else Poly = new Poly_PolygonOnTriangulation(Nodes);
      Poly->Deflection(def);
//      }
//      else {
//      IS.seekg(ppp);
//      Poly = new Poly_PolygonOnTriangulation(Nodes);
//      }
    myNodes.Add(Poly);
  }
//  }
//  else IS.seekg(pos);
}
```

几何意义上的曲线 C 是由参数方程中的参数往 u 增加的方向确定的。

五、<shapes>部分

示例：

一个包含<shapes>部分的完整*.brep 文件在附录中给出了。

BNF 定义：

BNF-like Definition

<shapes> = <shape header> <_\n> <shape records> <_\n> <shape final record>;

<shape header> = "TShapes" <_> <shape count>;

<shape count> = <int>;

<shape records> = <shape record> ^ <shape count>;

<shape record> = <shape subrecord> <_\n> <shape flag word> <_\n> <shape subshapes> <_\n>;

<shape flag word> = <flag> ^ 7;

<shape subshapes> = (<shape subshape> <_>)* "*";

<shape subshape> =
<shape subshape orientation> <shape subshape number> <_> <shape location number>;

<shape subshape orientation> = "+" | "-" | "i" | "e";

<shape subshape number> = <int>;

<shape location number> = <int>;

<shape final record> = <shape subshape>;

<shape subrecord> =
("Ve" <_\n> <vertex data> <_\n>) |
("Ed" <_\n> <edge data> <_\n>) |
("Wi" <_\n> <_\n>) |
("Fa" <_\n> <face data>) |
("Sh" <_\n> <_\n>) |
("So" <_\n> <_\n>) |
("CS" <_\n> <_\n>) |
("Co" <_\n> <_\n>);

详细说明：

<shape flag word>f1f2f3f4f5f6f7 中的每一位都有意义，如下所示：

1) f1 : free
2) f2 : modified
3) f3: IGNORED(version 1)/checked (version 2)
4) f4: orientable
5) f5: closed
6) f6: infinite
7) f7: convex

The flags are used in a special way[1].

<shape subshape orientation> 表示意义如下所示：

1) + : forward
2) - : reversed
3) i: internal
4) e: external

<shape subshape orientation> is used in a special way[1].

<shape subshape number>

<shape recored>

<shape subrecord>的类型有如下几种：

1) “Ve”: vertex
2) “Ed”: edge
3) “Wi”: wire
4) “Fa”: face
5) “Sh”: shell
6) “So”: solid
7) “CS”: compsolid
8) “Co”: compound

<shape final record>

读取<shapes>部分的程序代码如下所示：

```
//=======================================================================
//function : ReadGeometry
//purpose  :
//=======================================================================
void  BRepTools_ShapeSet::ReadGeometry(const TopAbs_ShapeEnum T,
                                       Standard_IStream&      IS,
                                       TopoDS_Shape&          S)
{
  // Read the geometry
  Standard_Integer val,c,pc,pc2,s,s2,l,l2,t, pt, pt2;
  Standard_Real tol,X,Y,Z,first,last,p1,p2;
  Standard_Real PfX,PfY,PlX,PlY;
  gp_Pnt2d aPf, aPl;
  Standard_Boolean closed;
#ifndef DEB
  GeomAbs_Shape reg = GeomAbs_C0;
#else
  GeomAbs_Shape reg;
#endif
  switch (T) {
    //---------
    // vertex
    //---------
```

```
  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;

  //---------
  // edge
  //---------
  case TopAbs_EDGE :

    // Create an edge
```

```
{
  TopoDS_Edge& E = TopoDS::Edge(S);
  myBuilder.MakeEdge(E);

  // Read the curve geometry
  IS >> tol;
  IS >> val;
  myBuilder.SameParameter(E, (val == 1));
  IS >> val;
  myBuilder.SameRange(E, (val == 1));
  IS >> val;
  myBuilder.Degenerated(E, (val == 1));

  do {
    IS >> val;
    switch (val) {

    case 1 :                                  // -1- Curve 3D
      IS >> c >> l;
  if (!myCurves.Curve(c).IsNull()) {
    myBuilder.UpdateEdge(E,myCurves.Curve(c),
             Locations().Location(l),tol);
  }
      IS >> first >> last;
  if (!myCurves.Curve(c).IsNull()) {
    Standard_Boolean Only3d = Standard_True;
    myBuilder.Range(E,first,last,Only3d);
  }
      break;

    case 2 :                                  // -2- Curve on surf
    case 3 :                                  // -3- Curve on closed surf
      closed = (val == 3);
      IS >> pc;
      if (closed) {
        IS >> pc2;
        reg = ReadRegularity(IS);
      }

      // surface, location
      IS >> s >> l;

      // range
      IS >> first >> last;
```

```
          // read UV Points // for XML Persistence higher performance
          if (FormatNb() == 2)
          {
            IS >> PfX >> PfY >> PlX >> PlY;
            aPf = gp_Pnt2d(PfX,PfY);
            aPl = gp_Pnt2d(PlX,PlY);
          }

//  Modified by Sergey KHROMOV - Wed Apr 24 12:11:16 2002 Begin
        if (myCurves2d.Curve2d(pc).IsNull() ||
        (closed && myCurves2d.Curve2d(pc2).IsNull()) ||
        mySurfaces.Surface(s).IsNull())
          break;
//  Modified by Sergey KHROMOV - Wed Apr 24 12:11:17 2002 End

          if (closed) {
            if (FormatNb() == 2)
              myBuilder.UpdateEdge(E,myCurves2d.Curve2d(pc),
                                   myCurves2d.Curve2d(pc2),
                                   mySurfaces.Surface(s),
                                   Locations().Location(l),tol,
                                   aPf, aPl);
            else
              myBuilder.UpdateEdge(E,myCurves2d.Curve2d(pc),
                                   myCurves2d.Curve2d(pc2),
                                   mySurfaces.Surface(s),
                                   Locations().Location(l),tol);

            myBuilder.Continuity(E,
                                 mySurfaces.Surface(s),
                                 mySurfaces.Surface(s),
                                 Locations().Location(l),
                                 Locations().Location(l),
                                 reg);
          }
          else
          {
            if (FormatNb() == 2)
              myBuilder.UpdateEdge(E,myCurves2d.Curve2d(pc),
                                   mySurfaces.Surface(s),
                                   Locations().Location(l),tol,
                                   aPf, aPl);
            else
```

```
                myBuilder.UpdateEdge(E,myCurves2d.Curve2d(pc),
                                     mySurfaces.Surface(s),
                                     Locations().Location(l),tol);
              }
              myBuilder.Range(E,
                              mySurfaces.Surface(s),
                              Locations().Location(l),
                              first,last);
              break;

            case 4 :                                  // -4- Regularity
              reg = ReadRegularity(IS);
              IS >> s >> l >> s2 >> l2;
//  Modified by Sergey KHROMOV - Wed Apr 24 12:39:13 2002 Begin
          if (mySurfaces.Surface(s).IsNull() ||
          mySurfaces.Surface(s2).IsNull())
            break;
//  Modified by Sergey KHROMOV - Wed Apr 24 12:39:14 2002 End
              myBuilder.Continuity(E,
                                   mySurfaces.Surface(s),
                                   mySurfaces.Surface(s2),
                                   Locations().Location(l),
                                   Locations().Location(l2),
                                   reg);
              break;

            case 5 :   // -5- Polygon3D
              IS >> c >> l;
//szy-02.05.2004
myBuilder.UpdateEdge(E,Handle(Poly_Polygon3D)::DownCast(myPolygons3D(c)));
              if (c > 0 && c <= myPolygons3D.Extent())

myBuilder.UpdateEdge(E,Handle(Poly_Polygon3D)::DownCast(myPolygons3D(c)),
Locations().Location(l));
              break;

            case 6 :
            case 7 :
              closed = (val == 7);
              IS >> pt;
              if (closed) {
                IS >> pt2;
              }
              IS >> t >> l;
```

```
            if (closed) {
              if (t > 0 && t <= myTriangulations.Extent() &&
                  pt > 0 && pt <= myNodes.Extent() &&
                  pt2 > 0 && pt2 <= myNodes.Extent())
                myBuilder.UpdateEdge
                  (E,
Handle(Poly_PolygonOnTriangulation)::DownCast(myNodes(pt)),
                   Handle(Poly_PolygonOnTriangulation)::DownCast(myNodes(pt2)),
                   Handle(Poly_Triangulation)::DownCast(myTriangulations(t)),
                   Locations().Location(l));
            }
            else {
              if (t > 0 && t <= myTriangulations.Extent() &&
                  pt > 0 && pt <= myNodes.Extent())
                myBuilder.UpdateEdge

(E,Handle(Poly_PolygonOnTriangulation)::DownCast(myNodes(pt)),
                   Handle(Poly_Triangulation)::DownCast(myTriangulations(t)),
                   Locations().Location(l));
            }
            // range

            break;

          }
        } while (val > 0);
      }
    break;

    //---------
    // wire
    //---------
  case TopAbs_WIRE :
    myBuilder.MakeWire(TopoDS::Wire(S));
    break;

    //---------
    // face
    //---------
  case TopAbs_FACE :
    {
    // create a face :
    TopoDS_Face& F = TopoDS::Face(S);
//    streampos pos;
```

```
    myBuilder.MakeFace(F);

    IS >> val; // natural restriction
    if (val == 0 || val == 1) {
      IS >> tol >> s >> l;
//  Modified by Sergey KHROMOV - Wed Apr 24 12:39:13 2002 Begin
      if (!mySurfaces.Surface(s).IsNull()) {
//  Modified by Sergey KHROMOV - Wed Apr 24 12:39:14 2002 End
    myBuilder.UpdateFace(TopoDS::Face(S),
                mySurfaces.Surface(s),
                Locations().Location(l),tol);
    myBuilder.NaturalRestriction(TopoDS::Face(S),(val == 1));
      }
//      pos = IS.tellg();
//      IS >> val;
    }
    else if(val == 2) {
      //only triangulation
      IS >> s;
      myBuilder.UpdateFace(TopoDS::Face(S),

Handle(Poly_Triangulation)::DownCast(myTriangulations(s)));
    }
//    else pos = IS.tellg();

    // BUC60769
    if(val == 2) break;

    char string[260];
    IS.getline ( string, 256, '\n' );
    IS.getline ( string, 256, '\n' );

    if (string[0] == '2') {
      // cas triangulation
      s = atoi ( &string[2] );
      if (s > 0 && s <= myTriangulations.Extent())
        myBuilder.UpdateFace(TopoDS::Face(S),

Handle(Poly_Triangulation)::DownCast(myTriangulations(s)));
    }
//    else IS.seekg(pos);
    }
    break;
```

```
    //---------
    // shell
    //---------
  case TopAbs_SHELL :
    myBuilder.MakeShell(TopoDS::Shell(S));
    break;

    //---------
    // solid
    //---------
  case TopAbs_SOLID :
    myBuilder.MakeSolid(TopoDS::Solid(S));
    break;

    //---------
    // compsolid
    //---------
  case TopAbs_COMPSOLID :
    myBuilder.MakeCompSolid(TopoDS::CompSolid(S));
    break;

    //---------
    // compound
    //---------
  case TopAbs_COMPOUND :
    myBuilder.MakeCompound(TopoDS::Compound(S));
    break;

  default:
    break;
  }
}
```

5.1 通用术语 Common Terms

下面的数据定义在<vertex data>、<edge data>、<face data>中都有用到：

BNF 定义：

BNF-like Definition

<location number> = <int>;

<3D curve number> = <int>;

<surface number> = <int>;

<2D curve number> = <int>;

<3D polygon number> = <int>;

<triangulation number> = <int>;

<polygon on triangulation number> = <int>;

<curve parameter minimal and maximal values> = <real> <_> <real>;

<curve values for parameter minimal and maximal values> =
real> <_> <real> <_> <real> <_> <real>;

详细说明：

<location number>是<locations>部分的<location record>编号。<location record>从 1 开始编号，编号 0 表示是单位矩阵变换。

<3D curve number>是<geometry>部分的<3D curves>子部分中<3D curve record>编号。<3D curve record>编号从 1 开始。

<surface number>是<geometry>部分的<surfaces>子部分中<surface record>编号。<surface record>编号从 1 开始。

<2D curve number>是<geometry>部分的<2D curves>子部分中<2D curve record>编号。<2D curve record>编号从 1 开始。

<3D polygon number>是<geometry>部分的<3D polygons>子部分中<3D polygon record>编号。<3D polygon record>编号从 1 开始。

<triangulation number>是<geometry>部分的<triangulations>子部分中<triangulation record>编号。<triangulation record>编号从 1 开始。

<polygon on triangulation number>是<geometry>部分的<polygons on triangulations>子部分中<polygons on triangulations>编号。<polygons on triangulations>编号从 1 开始。

<curve parameter minimal and maximal values>umin 和 umax 都是参数曲线的取值边界：umin<=u<=umax。

<curve values for parameter minimal and maximal values>根据参数 umin 和 umax 求得的曲线 C 上的值 xmin，ymin，xmax，ymax，即（xmin, ymin）＝C（umin)，（xmax, ymax）＝C（umax）。

5.2 <vertex data>

BNF 定义：

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>;

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：

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

5.3 <edge data>

BNF 定义：

BNF-like Definition

<edge data> = <_> <edge data tolerance> <_> <edge data same parameter flag> <_> edge data same range flag> <_> <edge data degenerated flag> <_\n> <edge data representations>;

<edge data tolerance> = <real>;

<edge data same parameter flag> = <flag>;

<edge data same range flag> = <flag>;

<edge data degenerated flag> = <flag>;

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

<edge data representation> =
"1" <_> <edge data representation data 1>
"2" <_> <edge data representation data 2>
"3" <_> <edge data representation data 3>
"4" <_> <edge data representation data 4>
"5" <_> <edge data representation data 5>
"6" <_> <edge data representation data 6>
"7" <_> <edge data representation data 7>;

<edge data representation data 1> = <3D curve number> <_> <location number> <_>
<curve parameter minimal and maximal values>;
<edge data representation data 2> = <2D curve number> <_> <surface number> <_>
<location number> <_> <curve parameter minimal and maximal values>
[<_\n> <curve values for parameter minimal and maximal values>];

<edge data representation data 3> = (<2D curve number> <_>) ^ 2 <continuity order> <_>
<surface number> <_> <location number> <_> <curve parameter minimal and maximal values>
<\n> <curve values for parameter minimal and maximal values>];

<continuity order> = "C0" | "C1" | "C2" | "C3" | "CN" | "G1" | "G2".

<edge data representation data 4> =
<continuity order> (<_> <surface number> <_> <location number>) ^ 2;

<edge data representation data 5> = <3D polygon number> <_> <location number>;

<edge data representation data 6> =
<polygon on triangulation number> <_> <triangulation number> <_> <location number>;

<edge data representation data 7> = (<polygon on triangulation number> <_>) ^ 2
<triangulation number> <_> <location number>;

详细说明：

标志位<edge data same parameter flag>，<edge data same range flag>，<edge data degenerated flag>有特别的用途。

<edge data representation data 1>表示一个三维曲线；

<edge data representation data 2>表示曲面上的一个二维曲线；

<curve values for parameter minimal and maximal values>只在 2 版本中使用；

<edge data representation data 3>表示闭合曲面上的一个二维曲线；

<curve values for parameter minimal and maximal values>只在 2 版本中使用；

<edge data representation data 5>表示一个三维的多段线（3D polyline）；

<edge data representation data 6>表示三角剖分上一条多段线；

<edge data tolerance> t 的定义如下所示：

$$\max_{C \in R} \max_{P \in E} \min_{Q \in C} |Q - P| \leq t .$$

5.4 <face data>

BNF 定义：

BNF-like Definition

<face data> = <face data natural restriction flag> <_> <face data tolerance> <_> <surface number> <_> <location number> <\n> ["2" <_> <triangulation number>];

<face data natural restriction flag> = <flag>;

<face data tolerance> = <real>;

详细说明：

<face data>描述了面 F 的曲面 S 和三角剖分 T。曲面 S 可能为空：<surface number>＝0.

<face data tolerance> t 的定义如下所示：

$$\max_{P\in F}\min_{Q\in S}|Q-P|\le t\,.$$

标志位<face data natural restriction flag>有特别的用途。

六、**实例分析**

OpenCascade 的 data 目录中的 face1.brep 文件：

```
DBRep_DrawableShape

CASCADE Topology V1, (c) Matra-Datavision
Locations 5
1
                1               0               0            -0.4
                0               1               0               0
                0               0               1               0
1
                1               0               0            -0.4
                0               1               0               0
                0               0               1               0
1
                1               0               0             0.8
                0               1               0               0
                0               0               1               0
1
                1               0               0             0.8
                0               1               0               0
                0               0               1               0
2    1 1 2 1 3 1 4 1 0
Curve2ds 8
1 0 0.8 1 0
2 0 0 1 0 0 1 0.8
1 1.82347658193698 0 0 1
1 1 -1.77459666924148 1 0
1 4.45970872524261 0 0 1
1 1 -0.225403330758516 1 0
1 0 0 1 0
2 0 0 1 0 0 1 0.8
Curves 4
2 0 0 0.8 0 0 1 1 0 0 0 1 0 0.8
1 -0.2 0.774596669241484 0 0 0 1
1 -0.2 -0.774596669241484 0 0 0 1
2 0 0 0 0 0 1 1 0 0 0 1 0 0.8
Polygon3D 0
PolygonOnTriangulations 0
Surfaces 4
2 0 0 0 0 0 1 1 0 0 0 1 0 0.8
1 0 0 0.8 0 0 1 1 0 0 0 1 0
1 -1 -1 -1 1 0 0 0 0 1 0 -1 0
```

```
1 0 0 0 0 0 1 1 0 0 0 1 0
Triangulations 0

TShapes 10
Ve
2.0000002e-007
-0.2 0.774596669241483 0.8
0 0

0101101
*
Ve
2.0000002e-007
-0.2 -0.774596669241484 0.8
0 0

0101101
*
Ed
  1e-007 1 1 0
1    1 0 1.82347658193698 4.45970872524261
2    1 1 0 1.82347658193698 4.45970872524261
2    2 2 0 1.82347658193698 4.45970872524261
0

0101000
+10 0 -9 0 *
Ve
2.0000002e-007
-0.2 0.774596669241483 0
0 0

0101101
*
Ed
  1e-008 1 1 0
1    2 0 0 0.8
2    3 1 0 0 0.8
2    4 3 5 0 0.8
0

0101000
+7 0 -10 0 *
Ve
```

```
2.0000002e-007
-0.2 -0.774596669241484 0
0 0

0101101
*
Ed
  1e-008 1 1 0
1    3 0 0 0.8
2    5 1 0 0 0.8
2    6 3 5 0 0.8
0

0101000
+5 0 -9 0 *
Ed
  1e-007 1 1 0
1    4 0 1.82347658193698 4.45970872524261
2    7 1 0 1.82347658193698 4.45970872524261
2    8 4 0 1.82347658193698 4.45970872524261
0

0101000
+7 0 -5 0 *
Wi

0101100
-8 0 -6 0 +4 0 +3 0 *
Fa
0    1e-007 1 0

0101000
+2 0 *

+1 0
0
```

显示结果如下图所示：

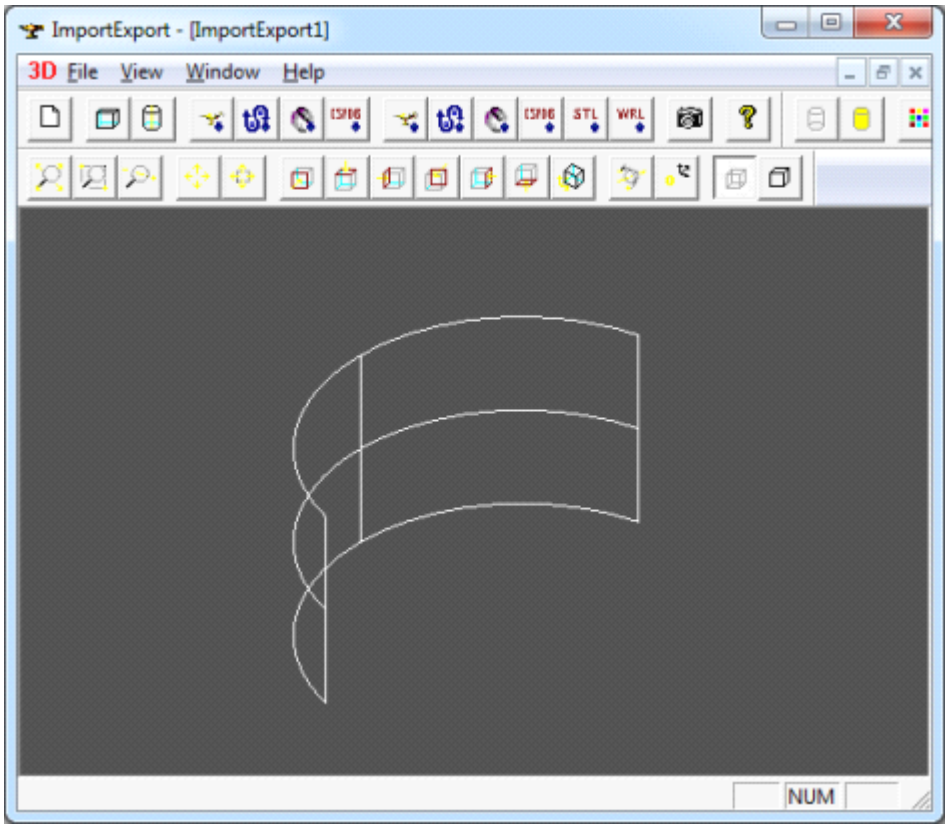

Figure 1. Wireframe mode

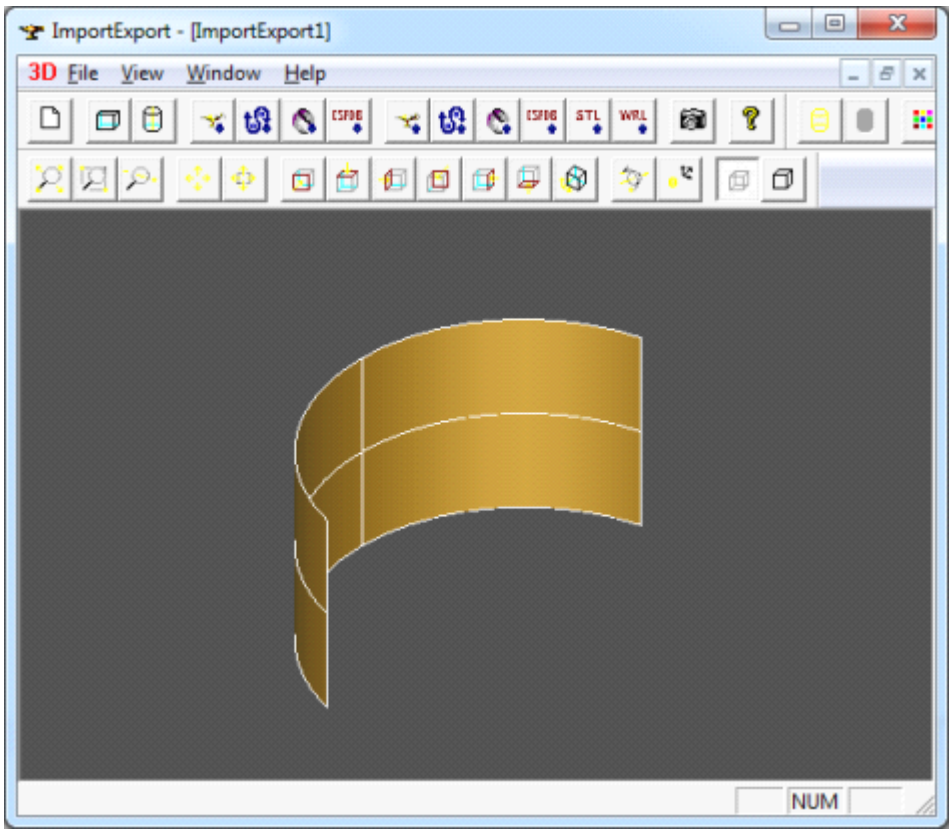

Figure 2. Shaded mode

七、结论

因为 OpenCascade 的 Brep 格式是自己的格式，只用到了 ModelingData 模块，不使用 DataExchange 模块，可以作为数据交换的一种格式。

八、参考资料

1. BNF 范式：http://hi.baidu.com/xuyingming2012/item/9060029141427bd81b49df9e
2. BRep Format Description
3. OpenCascade source code