Geometry Surface of OpenCascade BRep

eryar@163.com

摘要 Abstract：几何曲面是参数表示的曲面 ，在边界表示中其数据存在于 BRep_TFace 中，BRep_TFace 中不仅包括了几何曲线，还包含用于显示的离散几何信息，如三角剖分数据。本文主要对 OpenCascade 的 BRep 表示中几何曲面进行说明，将在后面分析 Topology 部分的读写程序时来说明包含几何数据的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 Surface, Topology

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：

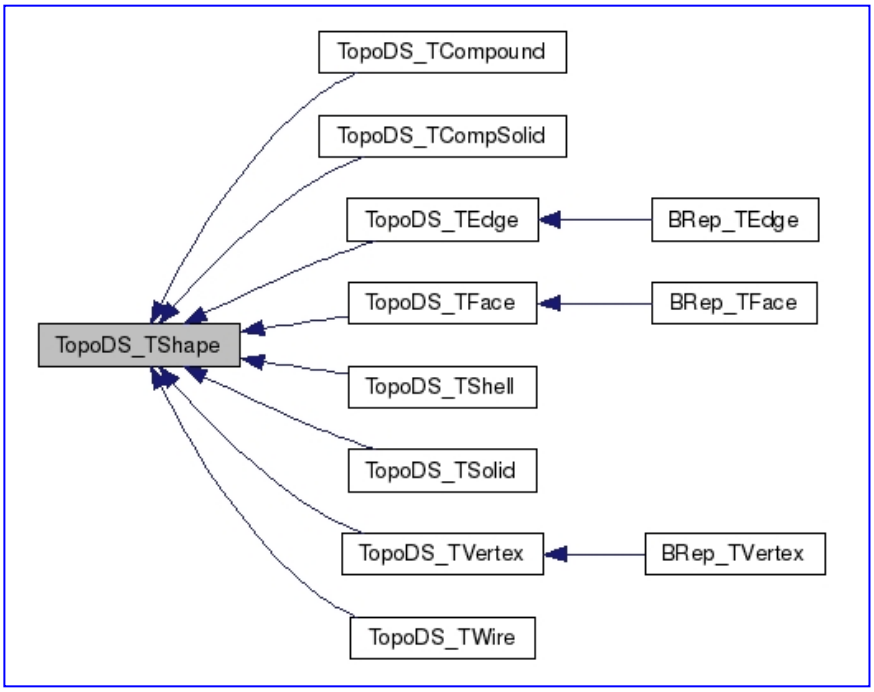


Figure 1.1 Topology data structure in OpenCascade

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 表示中几何曲面进行说明，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面在 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参数曲面，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 : 2013-12-01 12:18
*        Version : 1.0v
*
*    Description : Demonstrate the geometry surface section
*                  of the BRep file of OpenCascade.
*       KeyWords : OpenCascade, BRep File, Geometry Surface
*
*/

// OpenCascade library.
#define WNT
#include <Geom_Plane.hxx>
#include <Geom_CylindricalSurface.hxx>
#include <Geom_ConicalSurface.hxx>
#include <Geom_SphericalSurface.hxx>
#include <Geom_ToroidalSurface.hxx>
#include <Geom_SurfaceOfLinearExtrusion.hxx>
#include <Geom_SurfaceOfRevolution.hxx>
#include <Geom_BezierSurface.hxx>
#include <Geom_BSplineSurface.hxx>
#include <Geom_RectangularTrimmedSurface.hxx>
#include <Geom_OffsetSurface.hxx>

#include <TColgp_Array2OfPnt.hxx>
#include <TColStd_Array1OfReal.hxx>
#include <TColStd_Array2OfReal.hxx>
#include <TColStd_Array1OfInteger.hxx>

#include <GeomTools.hxx>
#include <Geom_Circle.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("geometrySurface.txt");
```

```
    // Surface record 1 - Plane.
    // Example: 1 0 0 3 0 0 1 1 0 -0 -0 1 0
    Handle_Geom_Plane thePlane = new Geom_Plane(gp_Pnt(0, 0, 3), gp_Dir(0, 0, 1));
    GeomTools::Write(thePlane, dumpFile);
    GeomTools::Dump(thePlane, dumpFile);
    GeomTools::Write(thePlane, std::cout);
    GeomTools::Dump(thePlane, std::cout);

    // Surface record 2 - Cylinder.
    // Example: 2 1 2 3 0 0 1 1 0 0 0 1 0 4
    Handle_Geom_CylindricalSurface theCylinder = new Geom_CylindricalSurface(axis,
4.0);
    GeomTools::Write(theCylinder, dumpFile);
    GeomTools::Dump(theCylinder, dumpFile);
    GeomTools::Write(theCylinder, std::cout);
    GeomTools::Dump(theCylinder, std::cout);

    // Surface record 3 - Cone.
    // Example: 3 1 2 3 0 0 1 1 0 0 0 1 0 4
    //          0.75
    Handle_Geom_ConicalSurface theCone = new Geom_ConicalSurface(axis, 0.75, 4.0);
    GeomTools::Write(theCone, dumpFile);
    GeomTools::Dump(theCone, dumpFile);
    GeomTools::Write(theCone, std::cout);
    GeomTools::Dump(theCone, std::cout);

    // Surface record 4 - Sphere.
    // Example: 4 1 2 3 0 0 1 1 0 -0 -0 1 0 4
    Handle_Geom_SphericalSurface theSphere = new Geom_SphericalSurface(axis, 4);
    GeomTools::Write(theSphere, dumpFile);
    GeomTools::Dump(theSphere, dumpFile);
    GeomTools::Write(theSphere, std::cout);
    GeomTools::Dump(theSphere, std::cout);

    // Surface record 5 - Torus.
    // Example: 5 1 2 3 0 0 1 1 0 -0 -0 1 0 8 4
    Handle_Geom_ToroidalSurface theTorus = new Geom_ToroidalSurface(axis, 8, 4);
    GeomTools::Write(theTorus, dumpFile);
    GeomTools::Dump(theTorus, dumpFile);
    GeomTools::Write(theTorus, std::cout);
    GeomTools::Dump(theTorus, std::cout);

    // Surface record 6 - Linear Extrusion.
    // Example: 6 0 0.6 0.8
    //          2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
    Handle_Geom_Circle baseCurve = new Geom_Circle(axis, 4.0);
    Handle_Geom_SurfaceOfLinearExtrusion       theExtrusion       =       new
Geom_SurfaceOfLinearExtrusion(baseCurve, gp_Dir(0, 0.6, 0.8));
    GeomTools::Write(theExtrusion, dumpFile);
    GeomTools::Dump(theExtrusion, dumpFile);
    GeomTools::Write(theExtrusion, std::cout);
```

```
    GeomTools::Dump(theExtrusion, std::cout);

    // Surface record 7 - Revolution Surface.
    // Example: 7 -4 0 3 0 1 0
    //          2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
    Handle_Geom_SurfaceOfRevolution theRevolution = new
Geom_SurfaceOfRevolution(baseCurve, gp::OY());
    theRevolution->SetLocation(gp_Pnt(-4, 0, 3));
    GeomTools::Write(theRevolution, dumpFile);
    GeomTools::Dump(theRevolution, dumpFile);
    GeomTools::Write(theRevolution, std::cout);
    GeomTools::Dump(theRevolution, std::cout);

    // Surface record 8 - Bezier Surface.
    // Example: 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
    TColgp_Array2OfPnt poles(1, 3, 1, 2);
    TColStd_Array2OfReal weights(1, 3, 1, 2);

    poles.SetValue(1, 1, gp_Pnt(0, 0, 1));      weights.SetValue(1, 1, 7.0);
    poles.SetValue(1, 2, gp_Pnt(1, 0, -4));     weights.SetValue(1, 2, 10.0);

    poles.SetValue(2, 1, gp_Pnt(0, 1, -2));     weights.SetValue(2, 1, 8.0);
    poles.SetValue(2, 2, gp_Pnt(1, 1, 5));      weights.SetValue(2, 2, 11.0);

    poles.SetValue(3, 1, gp_Pnt(0, 2, 3));      weights.SetValue(3, 1, 9.0);
    poles.SetValue(3, 2, gp_Pnt(1, 2, 6));      weights.SetValue(3, 2, 12.0);

    Handle_Geom_BezierSurface theBezierSurface = new Geom_BezierSurface(poles,
weights);
    GeomTools::Write(theBezierSurface, dumpFile);
    GeomTools::Dump(theBezierSurface, dumpFile);
    GeomTools::Write(theBezierSurface, std::cout);
    GeomTools::Dump(theBezierSurface, std::cout);

    // Surface record 9 - B-spline Surface.
    // Example: 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
```

```
    Standard_Integer uDegree = 1;
    Standard_Integer vDegree = 1;
    Standard_Boolean uPeriodic = Standard_False;
    Standard_Boolean vPeriodic = Standard_False;

    TColStd_Array1OfReal uKnots(1, 5);
    TColStd_Array1OfReal vKnots(1, 4);
    TColStd_Array1OfInteger uMults(1, 5);
    TColStd_Array1OfInteger vMults(1, 4);

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

    vKnots.SetValue(1, 0);
    vKnots.SetValue(2, 0.3);
    vKnots.SetValue(3, 0.7);
    vKnots.SetValue(4, 1.0);

    // Multiplicity of u and v are 1.
    uMults.Init(1);
    vMults.Init(1);

    Handle_Geom_BSplineSurface theBSplineSurface = new Geom_BSplineSurface(poles,
weights, uKnots, vKnots, uMults, vMults, uDegree, vDegree, uPeriodic, vPeriodic);
    GeomTools::Write(theBSplineSurface, dumpFile);
    GeomTools::Dump(theBSplineSurface, dumpFile);
    GeomTools::Write(theBSplineSurface, std::cout);
    GeomTools::Dump(theBSplineSurface, std::cout);

    // Surface record 10 - Rectangular Trim Surface.
    // Example: 10 -1 2 -3 4
    //          1 1 2 3 0 0 1 1 0 -0 -0 1 0
    Handle_Geom_Plane baseSurface = new Geom_Plane(axis);
    Handle_Geom_RectangularTrimmedSurface     theTrimmedSurface     =     new
Geom_RectangularTrimmedSurface(baseSurface, -1.0, 2.0, -3.0, 4.0);
    GeomTools::Write(theTrimmedSurface, dumpFile);
    GeomTools::Dump(theTrimmedSurface, dumpFile);
    GeomTools::Write(theTrimmedSurface, std::cout);
    GeomTools::Dump(theTrimmedSurface, std::cout);

    // Surface record 11 - Offset Surface.
    // Example: 11 -2
    //          1 1 2 3 0 0 1 1 0 -0 -0 1 0
    Handle_Geom_OffsetSurface          theOffsetSurface          =          new
Geom_OffsetSurface(baseSurface, -2.0);
    GeomTools::Write(theOffsetSurface, dumpFile);
    GeomTools::Dump(theOffsetSurface, dumpFile);
    GeomTools::Write(theOffsetSurface, std::cout);
```

```
    GeomTools::Dump(theOffsetSurface, std::cout);

    return 0;
    }
```

上述程序将《BRep Format Description White Paper》中的几何部分（Geometry Section）的参数曲面（Surfaces）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 geometrySurface.txt 中，输出数据如下所示：

```
1 0 0 3 0 0 1 1 0 -0 -0 1 0
Plane
   Origin :0, 0, 3
   Axis     :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0

2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
CylindricalSurface
   Origin :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 4
0.75
ConicalSurface
   Origin :1, 2, 3
   Axis     :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
   Radius :4

   Angle :0.75

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

5 1 2 3 0 0 1 1 0 -0 -0 1 0 8 4
ToroidalSurface
   Origin :1, 2, 3
   Axis     :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
   Radii    :8 4

6 0 0.6 0.8
2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
SurfaceOfLinearExtrusion
   Direction :0, 0.6, 0.8
```

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

7 -4 0 3 0 1 0
2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
SurfaceOfRevolution
   Origin        :-4, 0, 3
   Direction :0, 1, 0
   Basis curve :
Circle
   Center :1, 2, 3
   Axis      :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
   Radius :4

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

BezierSurface urational vrational
   Degrees :2 1
     1,    1 : 0, 0, 1    7
     1,    2 : 1, 0, -4    10

     2,    1 : 0, 1, -2    8
     2,    2 : 1, 1, 5    11

     3,    1 : 0, 2, 3    9
     3,    2 : 1, 2, 6    12



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

BSplineSurface urational vrational
   Degrees :1 1
   NbPoles :3 2
   NbKnots :5 4
 Poles :
```

```
    1,    1 : 0, 0, 1    7
    1,    2 : 1, 0, -4    10

    2,    1 : 0, 1, -2    8
    2,    2 : 1, 1, 5    11

    3,    1 : 0, 2, 3    9
    3,    2 : 1, 2, 6    12


 UKnots :

    1 : 0 1

    2 : 0.25 1

    3 : 0.5 1

    4 : 0.75 1

    5 : 1 1


 VKnots :

    1 : 0 1

    2 : 0.3 1

    3 : 0.7 1

    4 : 1 1


10 -1 2 -3 4
1 1 2 3 0 0 1 1 0 -0 -0 1 0
RectangularTrimmedSurface
Parameters : -1 2 -3 4
BasisSurface :
Plane
   Origin :1, 2, 3
   Axis      :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0

11 -2
1 1 2 3 0 0 1 1 0 -0 -0 1 0
OffsetSurface
Offset : -2
BasisSurface :
Plane
   Origin :1, 2, 3
   Axis      :0, 0, 1
   XAxis    :1, 0, -0
   YAxis    :-0, 1, 0
```

三、程序说明 Example Description

3.1 平面 Plane

示例：

```
// Surface record 1 - Plane.
// Example: 1 0 0 3 0 0 1 1 0 -0 -0 1 0
Handle_Geom_Plane thePlane = new Geom_Plane(gp_Pnt(0, 0, 3), gp_Dir(0, 0, 1));
GeomTools::Write(thePlane, dumpFile);
GeomTools::Dump(thePlane, dumpFile);
```

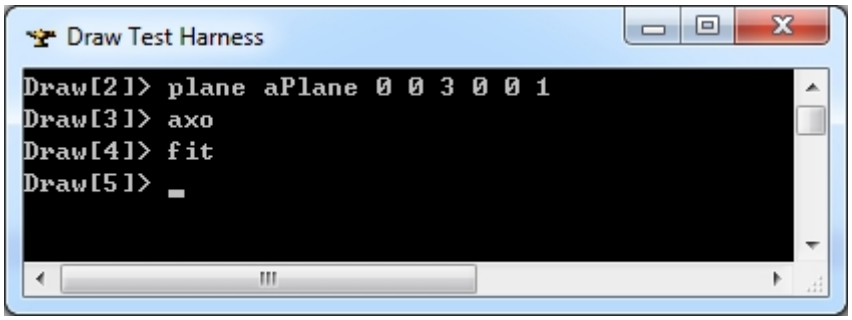
<surface record 1>定义了平面。平面数据包含三维点 P 和三维正交坐标系 N，Du，Dv。平面通过点 P，且其法向量为 N。其参数方程如下所示：

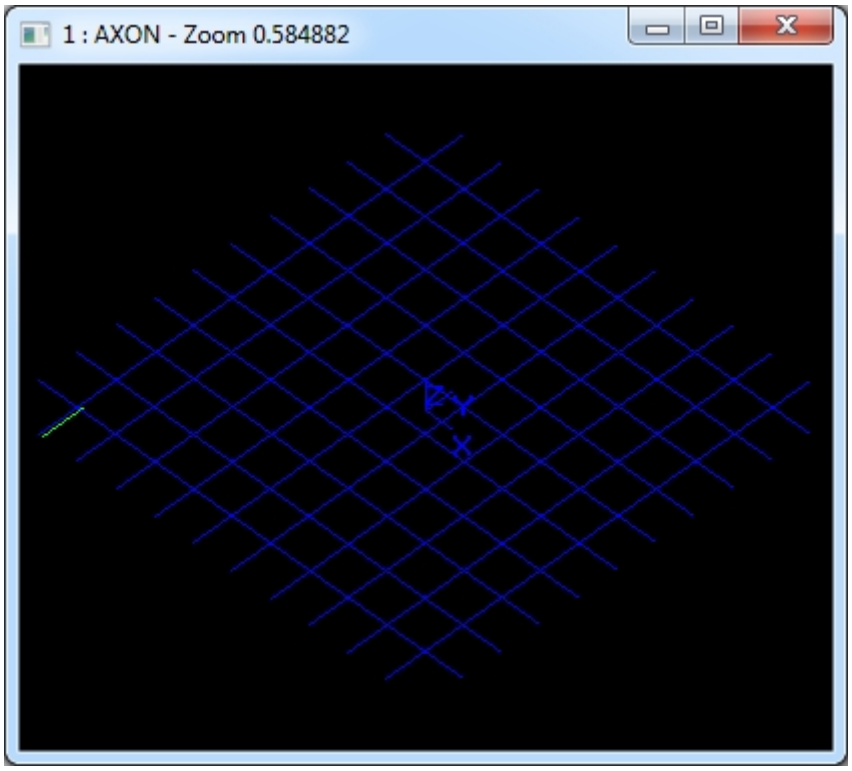
$$S(u,v)=P+u\cdot D_u+v\cdot D_v,\ (u,v)\in(-\infty,\infty)\times(-\infty,\infty).$$

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：

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

在 Draw Test Harness 中创建并显示平面如下所示：





3.2 圆柱面 Cylinder

示例：

```
// Surface record 2 - Cylinder.
// Example: 2 1 2 3 0 0 1 1 0 0 0 1 0 4
Handle_Geom_CylindricalSurface theCylinder = new Geom_CylindricalSurface(axis,
4.0);
GeomTools::Write(theCylinder, dumpFile);
GeomTools::Dump(theCylinder, dumpFile);
```

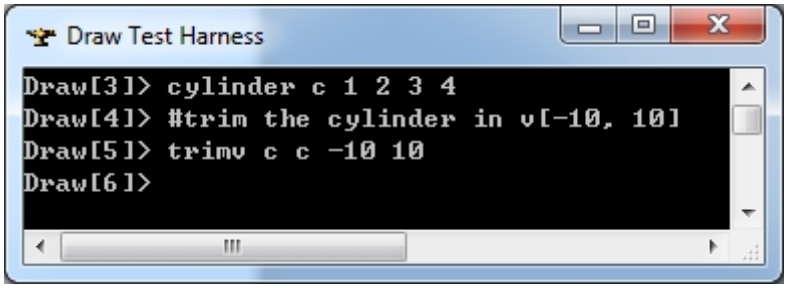
<surface record 2>定义了圆柱面。圆柱面的数据包含三维点 P，三维正交坐标系 Dv，Dx，Dy 和一个非负实数 r。圆柱面的轴通过点 P，方向为 Dv，圆柱面的半径为 r，其参数方程如下所示：

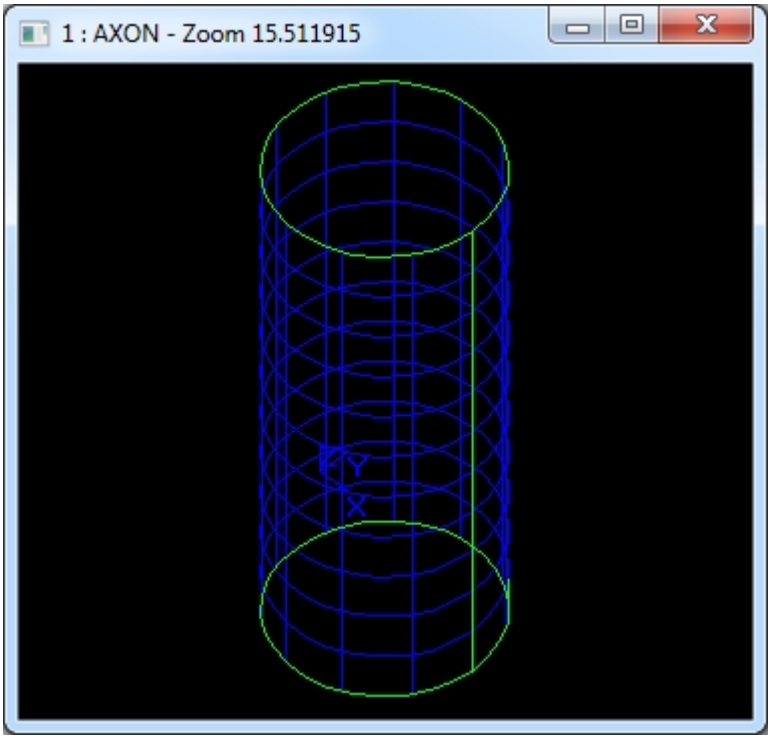
$$S(u,v)=P+r\cdot(\cos(u)\cdot D_X+\sin(u)\cdot D_Y)+v\cdot D_v,\ (u,v)\in[0,2\cdot\pi)\times(-\infty,\infty).$$

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：

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

在 Draw Test Harness 中创建并显示圆柱面如下所示：





3.3 圆锥面 Cone

示例：

```
// Surface record 3 - Cone.
// Example: 3 1 2 3 0 0 1 1 0 0 0 1 0 4
//          0.75
Handle_Geom_ConicalSurface theCone = new Geom_ConicalSurface(axis, 0.75, 4.0);
GeomTools::Write(theCone, dumpFile);
GeomTools::Dump(theCone, dumpFile);
```

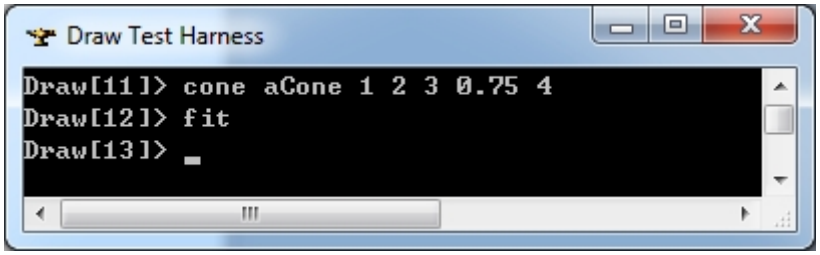
<surface record 3>定义了圆锥面。圆锥面的数据包含三维点 P，正交坐标系 Dz，Dx，Dy，非负实数 r 和实数 ψ（范围为（-π/2, π/2））。圆锥面通过点 P 且轴的方向为 Dz。过点 P 且与方向 Dx，Dy 平行的平面为圆锥面的参考平面（referenced plane）。参考平面截圆锥面为一个圆，其半径为 r。其参数方程如下所示：

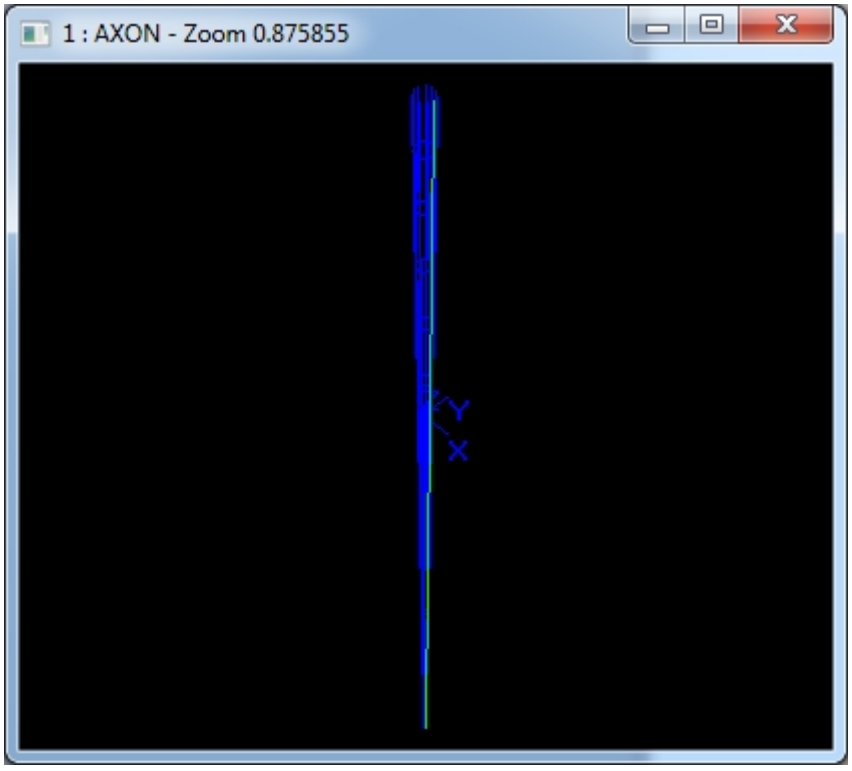
$$S(u,v)=P+(r+v\cdot\sin(\varphi))\cdot(\cos(u)\cdot D_X+\sin(u)\cdot D_Y)+v\cdot\cos(\varphi)\cdot D_Z,\ (u,v)\in[0,2\cdot\pi)\times(-\infty,\infty).$$

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：

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

在 Draw Test Harness 中创建并显示圆锥面如下所示：





3.4 球面 Sphere

示例：

```
// Surface record 4 - Sphere.
// Example: 4 1 2 3 0 0 1 1 0 -0 -0 1 0 4
Handle_Geom_SphericalSurface theSphere = new Geom_SphericalSurface(axis, 4);
GeomTools::Write(theSphere, dumpFile);
GeomTools::Dump(theSphere, dumpFile);
```

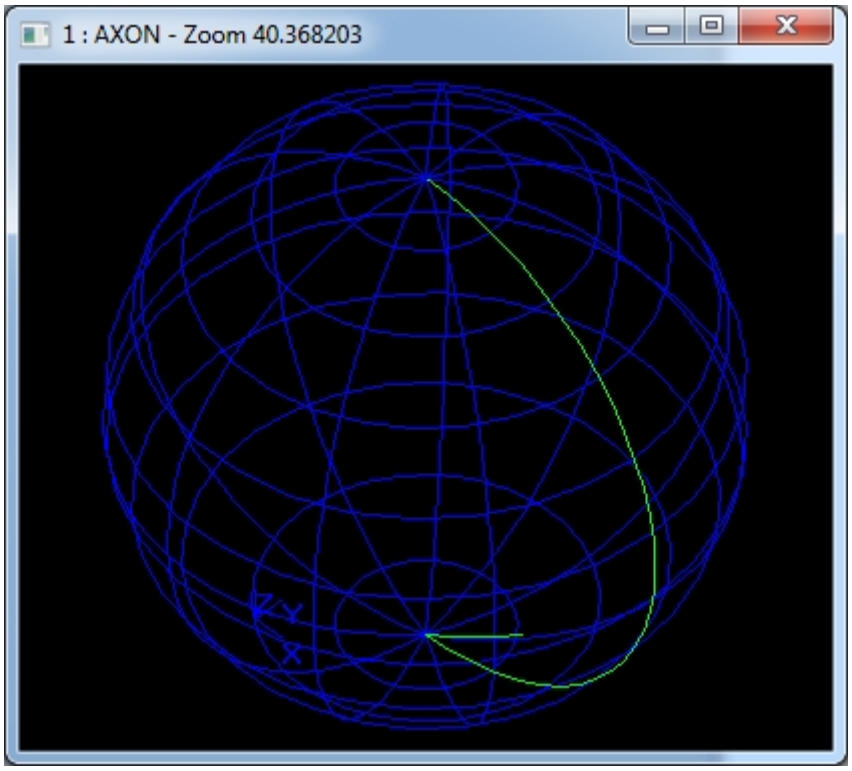
<surface record 4>定义了球面。球面的数据包含三维点 P，三维正交坐标系 Dz，Dx，Dy 和非负实数 r。即球面的球心为点 P，半径为 r，其参数方程如下所示：

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

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：

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

在 Draw Test Harness 中创建并显示球面如下所示：



3.5 圆环面 Torus

示例：

```
// Surface record 5 - Torus.
// Example: 5 1 2 3 0 0 1 1 0 -0 -0 1 0 8 4
Handle_Geom_ToroidalSurface theTorus = new Geom_ToroidalSurface(axis, 8, 4);
GeomTools::Write(theTorus, dumpFile);
GeomTools::Dump(theTorus, dumpFile);
```

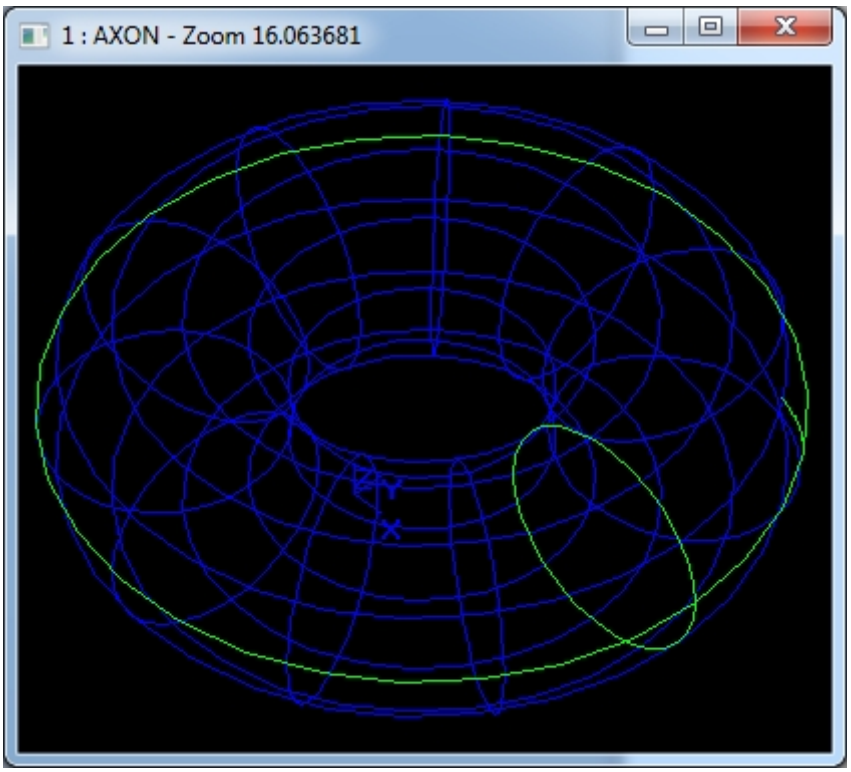
<surface record 5>定义了圆环面。圆环面的数据包含三维点 P，三维正交坐标系 Dz，Dx，Dy 和非负实数 r1，r2。圆环面的轴通过点 P，方向为 Dz，r1 是从圆环面的圆的中心到点 P 的距离，圆环面的圆的半径为 r2。圆环面的参数方程如下所示：

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

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：

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

在 Draw Test Harness 中创建并显示圆环面如下所示：



3.6 线性拉伸面 Linear Extrusion

示例：

```
// Surface record 6 - Linear Extrusion.
// Example: 6 0 0.6 0.8
//          2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
Handle_Geom_Circle baseCurve = new Geom_Circle(axis, 4.0);
Handle_Geom_SurfaceOfLinearExtrusion theExtrusion = new
Geom_SurfaceOfLinearExtrusion(baseCurve, gp_Dir(0, 0.6, 0.8));
GeomTools::Write(theExtrusion, dumpFile);
GeomTools::Dump(theExtrusion, dumpFile);
```

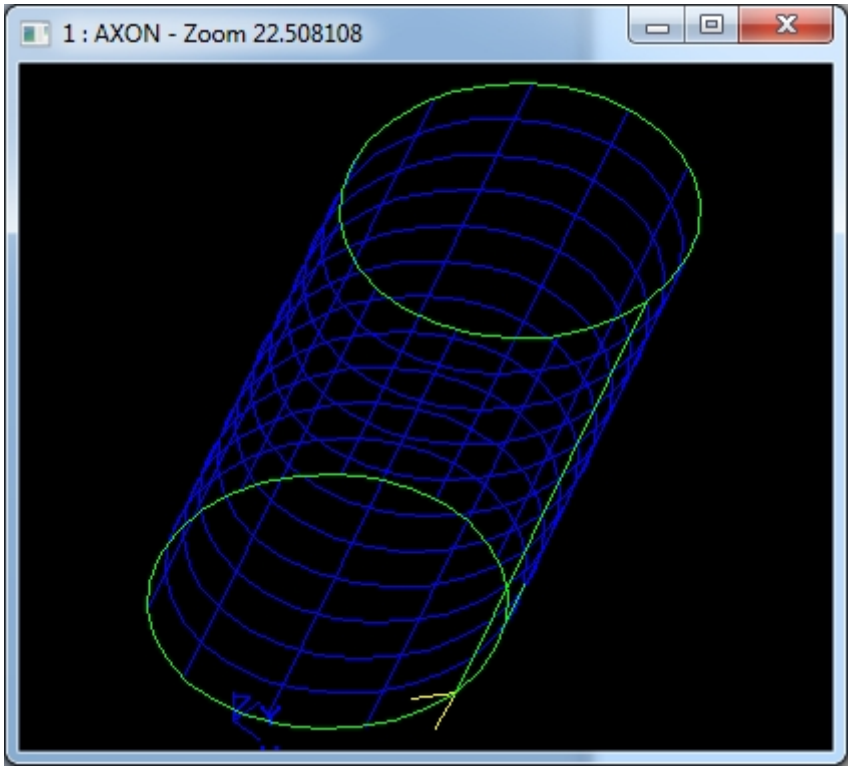
<surface record 6>定义了线性拉伸面。线性拉伸面的数据包含三维方向 Dv 和三维曲线<3D curve record>。其参数方程如下所示：

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

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：

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

在 Draw Test Harness 中创建并显示线性拉伸面如下所示：



3.7 旋转面 Revolution Surface

示例：

```
// Surface record 7 - Revolution Surface.
// Example: 7 -4 0 3 0 1 0
//          2 1 2 3 0 0 1 1 0 -0 -0 1 0 4
Handle_Geom_SurfaceOfRevolution          theRevolution          =          new
Geom_SurfaceOfRevolution(baseCurve, gp::OY());
theRevolution->SetLocation(gp_Pnt(-4, 0, 3));
GeomTools::Write(theRevolution, dumpFile);
GeomTools::Dump(theRevolution, dumpFile);
```

<surface record 7>定义了旋转面。旋转面的数据包含三维点 P，三维方向 D 和三维曲线。旋转曲面的轴通过点 P 且方向为 D，旋转曲线为 C 与旋转轴共面。旋转曲面的参数方程如下所示：

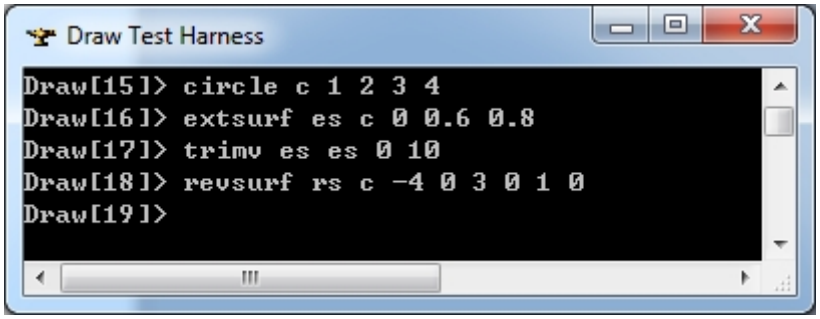
$$S(u,v)=P+V_D(v)+\cos(u)\cdot(V(v)-V_D(v))+\sin(u)\cdot[D,V(v)],\ (u,v)\in[0,2\cdot\pi)\times\mathrm{domain}(C)$$

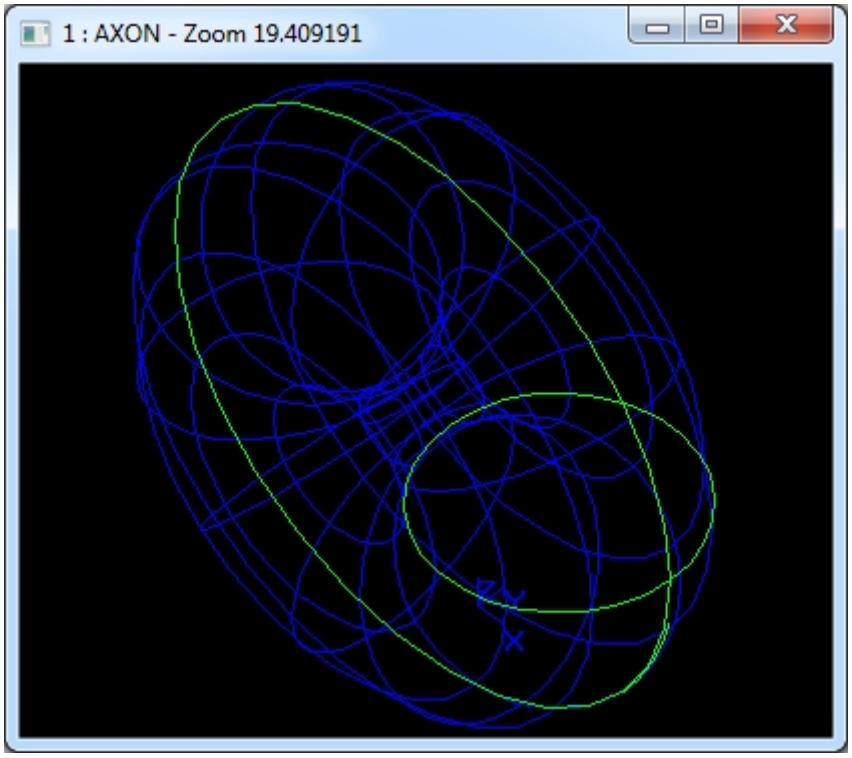
where $V(v)=C(v)-P$, $V_D(v)=(D,V(v))\cdot D$.

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
$$V(v)=(5,2,0)+4\cdot(\cos(v)\cdot(1,0,-0)+\sin(v)\cdot(-0,1,0)),\ V_D(v)=((0,1,0),V(v))\cdot(0,1,0).$$

在 Draw Test Harness 中创建并显示旋转面如下所示：




1 : AXON - Zoom 19.409191

3.8 Bezier 曲面 Bezier Surface

示例：

```
// Surface record 8 - Bezier Surface.
// Example: 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
TColgp_Array2OfPnt poles(1, 3, 1, 2);
TColStd_Array2OfReal weights(1, 3, 1, 2);

poles.SetValue(1, 1, gp_Pnt(0, 0, 1));      weights.SetValue(1, 1, 7.0);
poles.SetValue(1, 2, gp_Pnt(1, 0, -4));     weights.SetValue(1, 2, 10.0);

poles.SetValue(2, 1, gp_Pnt(0, 1, -2));     weights.SetValue(2, 1, 8.0);
poles.SetValue(2, 2, gp_Pnt(1, 1, 5));      weights.SetValue(2, 2, 11.0);

poles.SetValue(3, 1, gp_Pnt(0, 2, 3));      weights.SetValue(3, 1, 9.0);
poles.SetValue(3, 2, gp_Pnt(1, 2, 6));      weights.SetValue(3, 2, 12.0);

Handle_Geom_BezierSurface  theBezierSurface  =  new  Geom_BezierSurface(poles,
weights);
GeomTools::Write(theBezierSurface, dumpFile);
GeomTools::Dump(theBezierSurface, dumpFile);
```

<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。

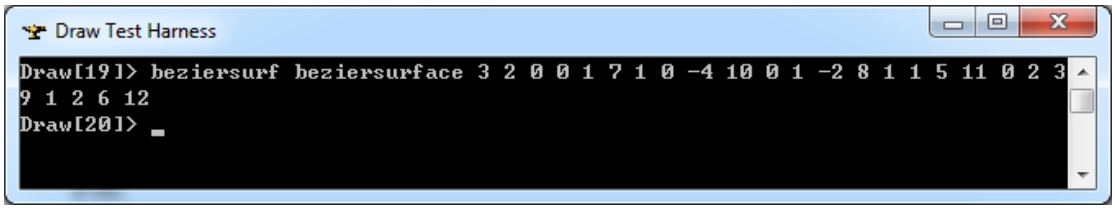
Bezier 曲面的参数方程如下所示：

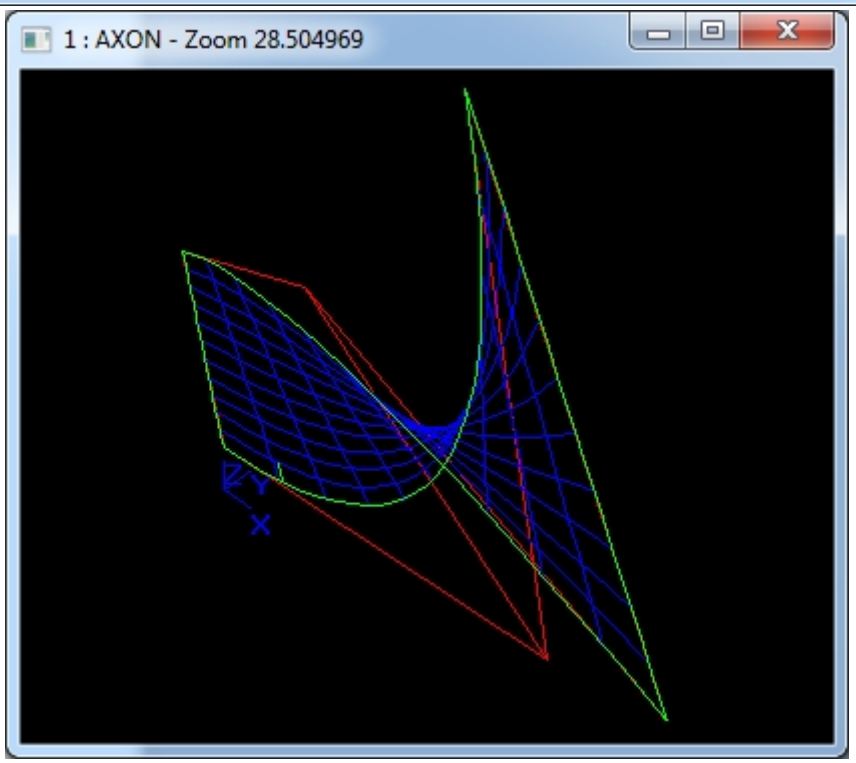
$$S(u,v)=\frac{\sum_{i=0}^{m_u}\sum_{j=0}^{m_v}B_{i,j}\cdot h_{i,j}\cdot C_{m_u}^{i}\cdot u^{i}\cdot(1-u)^{m_u-i}\cdot C_{m_v}^{j}\cdot v^{j}\cdot(1-v)^{m_v-j}}{\sum_{i=0}^{m_u}\sum_{j=0}^{m_v}h_{i,j}\cdot C_{m_u}^{i}\cdot u^{i}\cdot(1-u)^{m_u-i}\cdot C_{m_v}^{j}\cdot v^{j}\cdot(1-v)^{m_v-j}},\ (u,v)\in[0,1]\times[0,1]$$

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：

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

在 Draw Test Harness 中创建并显示 Bezier 曲面如下所示：





3.9 B 样条曲面 B-spline Surface

示例：

```
// Surface record 9 - B-spline Surface.
// Example: 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
Standard_Integer uDegree = 1;
Standard_Integer vDegree = 1;
Standard_Boolean uPeriodic = Standard_False;
Standard_Boolean vPeriodic = Standard_False;

TColStd_Array1OfReal uKnots(1, 5);
TColStd_Array1OfReal vKnots(1, 4);
TColStd_Array1OfInteger uMults(1, 5);
TColStd_Array1OfInteger vMults(1, 4);

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

vKnots.SetValue(1, 0);
vKnots.SetValue(2, 0.3);
vKnots.SetValue(3, 0.7);
vKnots.SetValue(4, 1.0);

// Multiplicity of u and v are 1.
uMults.Init(1);
vMults.Init(1);

Handle_Geom_BSplineSurface  theBSplineSurface  =  new  Geom_BSplineSurface(poles,
weights, uKnots, vKnots, uMults, vMults, uDegree, vDegree, uPeriodic, vPeriodic);
GeomTools::Write(theBSplineSurface, dumpFile);
GeomTools::Dump(theBSplineSurface, dumpFile);
```

<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：

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

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

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

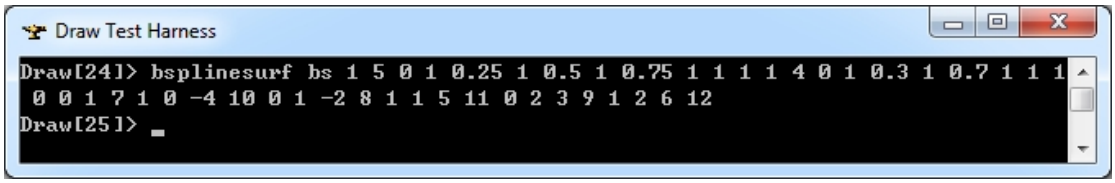
$$\bar{v}_i = v_j\ \left(1 \le j \le k_v,\ \sum_{l=1}^{j-1} t_l + 1 \le i \le \sum_{l=1}^{j} t_l\right).$$

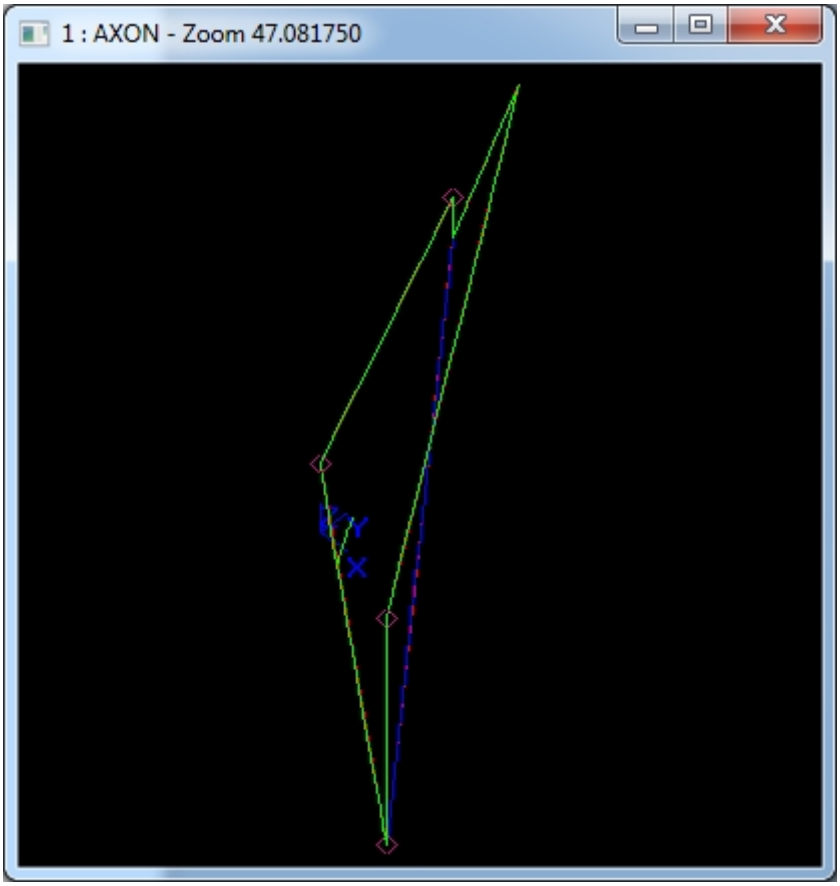
示例数据表示的 B-Spline 曲面为：u 有理标志位 ru＝1，v 有理标志位 rv＝1，u 次数 mu＝1，v 次数 mv＝1，u 控制点数 nu＝3，v 控制点数 nv＝2，u 有重复度的节点数 ku＝5，v 有重复度节点数 kv＝4，带权控制点 B1,1＝（0，0，1），h1,1＝7，B1,2＝（1，0，－4），h1,2＝10，B2,1＝（0，1，－2），h2,1＝8，B2,2＝（1，1，5），h2,2＝11，B3,1＝（0，2，3），h3,1＝9，B3,2＝（1，2，6），h3,2＝12，u 有重复度节点 u1＝0，q1＝1，u2＝0.25，q2＝1，

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：

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

在 Draw Test Harness 中创建并显示 B 样条曲面如下所示：





3.10 矩形裁剪曲面 Rectangular Trim Surface

示例：

```
// Surface record 10 - Rectangular Trim Surface.
// Example: 10 -1 2 -3 4
//          1 1 2 3 0 0 1 1 0 -0 -0 1 0
Handle_Geom_Plane baseSurface = new Geom_Plane(axis);
Handle_Geom_RectangularTrimmedSurface       theTrimmedSurface       =       new
Geom_RectangularTrimmedSurface(baseSurface, -1.0, 2.0, -3.0, 4.0);
GeomTools::Write(theTrimmedSurface, dumpFile);
GeomTools::Dump(theTrimmedSurface, dumpFile);
```

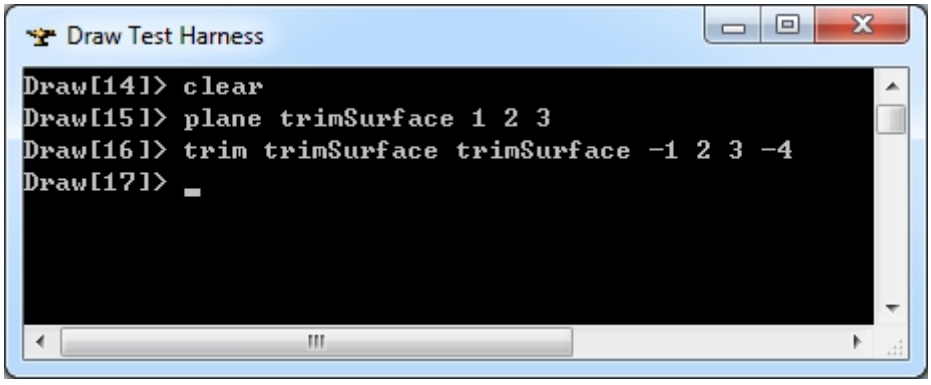
<surface record 10>定义了矩形裁剪曲面。矩形裁剪曲面的数据包含实数 umin，umax，vmin，vmax 和一个曲面。矩形裁剪曲面是将曲面限制在矩形区域[umin，umax]x[vmin，vmax]内得到的曲面。曲面的参数方程如下所示：

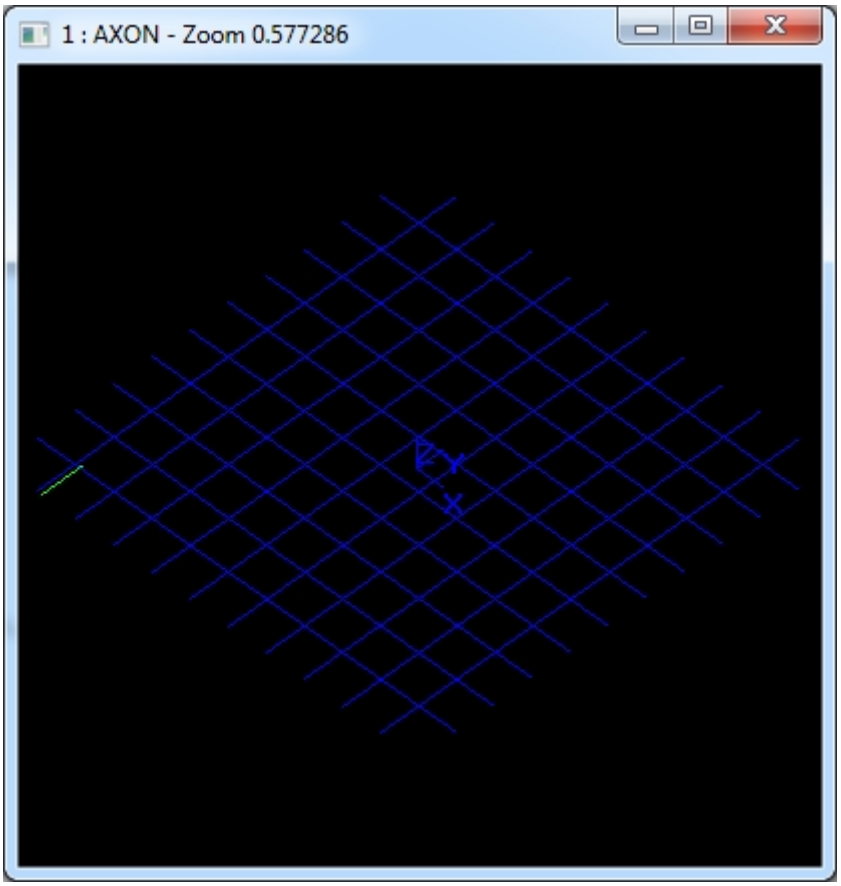
$$S(u,v)=B(u,v),\ (u,v)\in[u_{\min},u_{\max}]\times[v_{\min},v_{\max}].$$

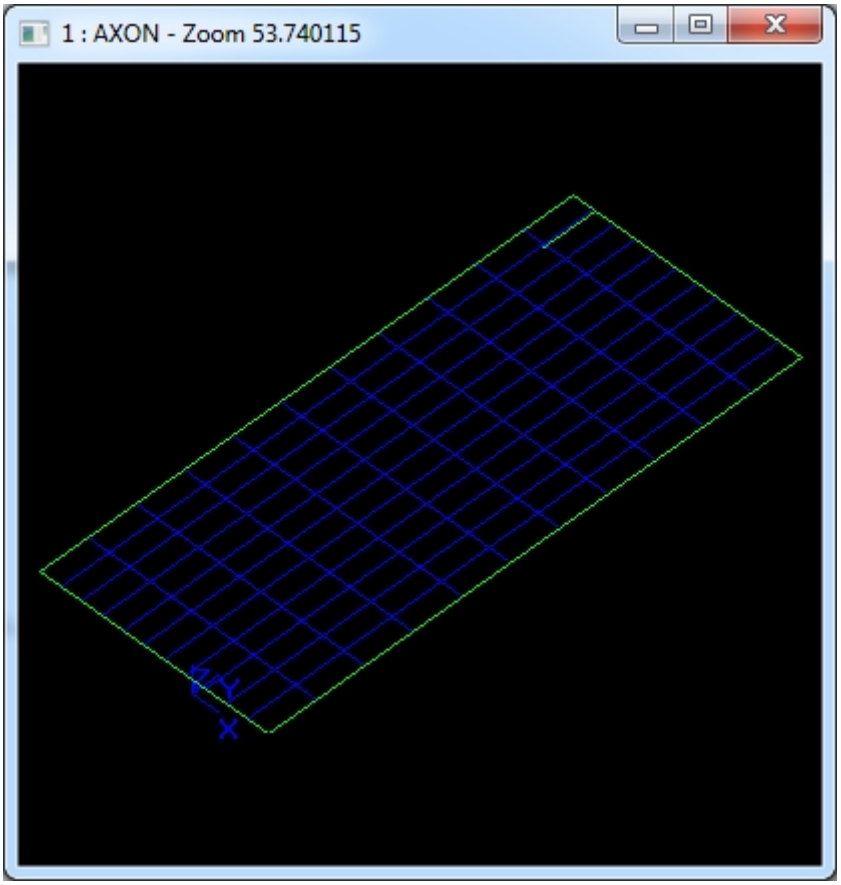
示例数据表示的矩形裁剪曲面的矩形裁剪区域为[-1，2]x[－3，4]，被裁剪曲面 B（u，v）＝（1，2，3）+u（1，0，0）+v（0，1，0）。其参数方程如下所示：

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

在 Draw Test Harness 中创建并显示矩形裁剪曲面如下所示：




1 : AXON - Zoom 0.577286
Z
Y
X


1 : AXON - Zoom 53.740115
Z
Y
X

3.11 偏移曲面 Offset Surface

示例：

```
// Surface record 11 - Offset Surface.
// Example: 11 -2
//          1 1 2 3 0 0 1 1 0 -0 -0 1 0
Handle_Geom_OffsetSurface theOffsetSurface = new Geom_OffsetSurface(baseSurface,
-2.0);
GeomTools::Write(theOffsetSurface, dumpFile);
GeomTools::Dump(theOffsetSurface, dumpFile);
```

<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：

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

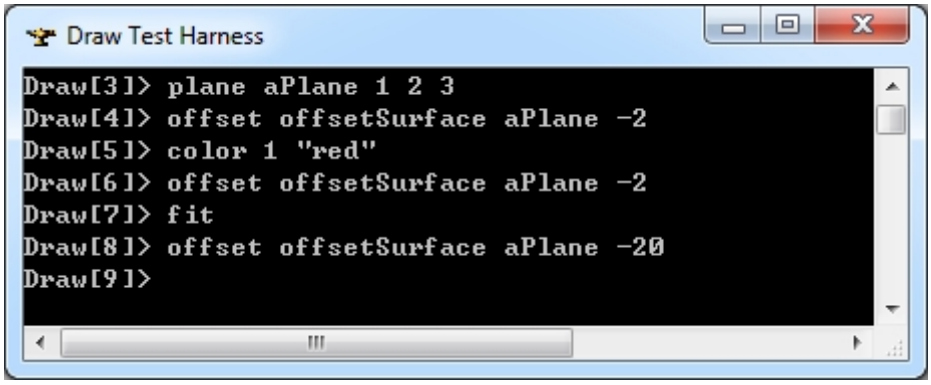
$$N(u,v) = [S'_u(u,v), S'_v(u,v)]$$

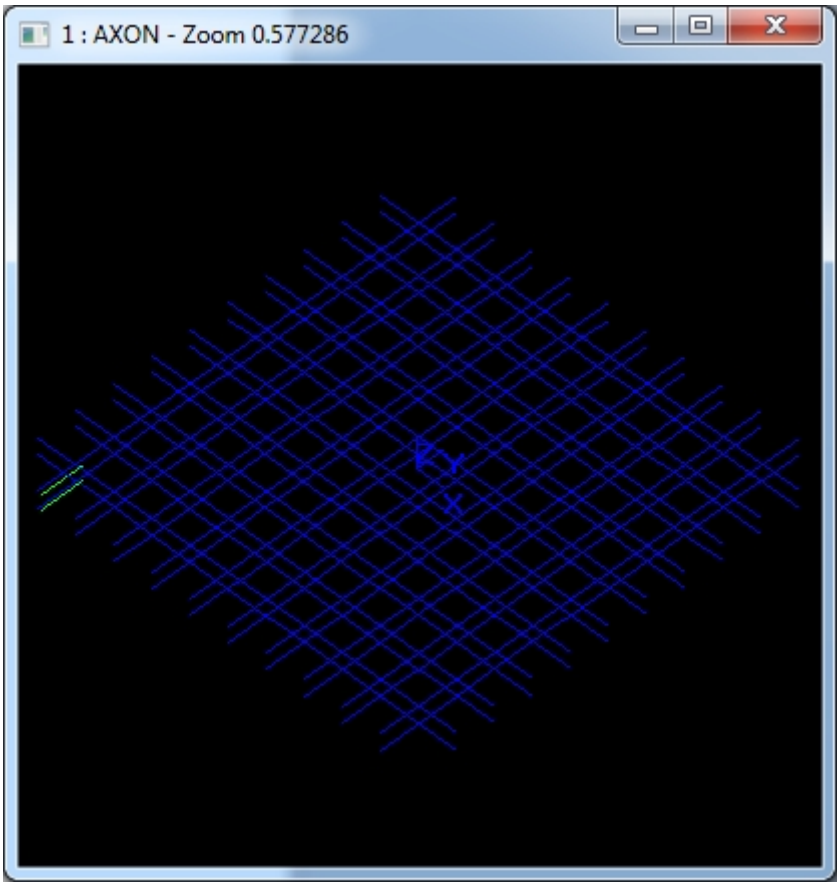
if $[S'_u(u,v), S'_v(u,v)] \neq \vec{0}$.

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：

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

在 Draw Test Harness 中创建并显示偏移曲面如下所示：





注：当偏移－2 时，效果不明显示，所以偏移了－20，这样看上去比较明显。

四、程序分析 Refactoring the Code

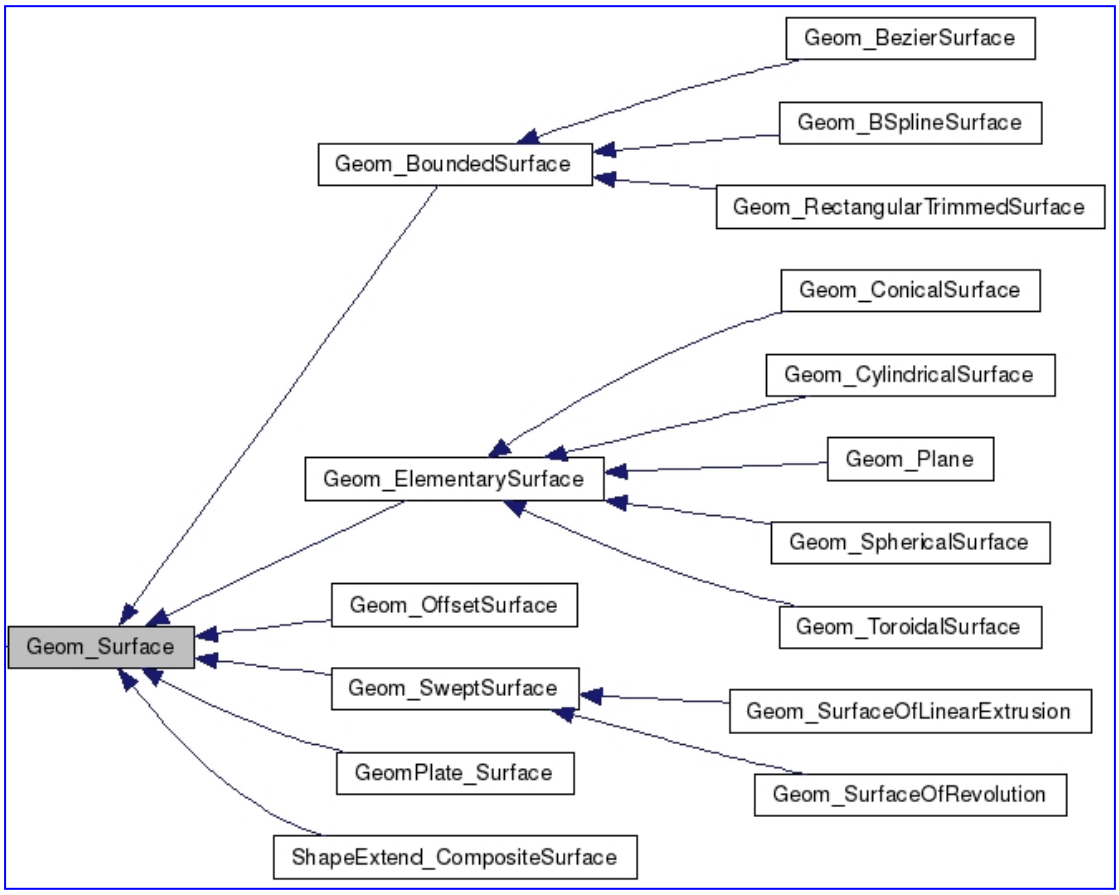


Figure 4.1 Class diagram of Geom_Surface

根据几何曲面的类图可知，几何曲面有个共同的基类 Geom_Surface。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 : PrintSurface
//purpose  :
//=======================================================================
void GeomTools_SurfaceSet::PrintSurface(const Handle(Geom_Surface)& S,
                                        Standard_OStream& OS,
                                        const Standard_Boolean compact)
{
  Handle(Standard_Type) TheType = S->DynamicType();

  if ( TheType ==  STANDARD_TYPE(Geom_Plane)) {
    Print(Handle(Geom_Plane)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_CylindricalSurface)) {
    Print(Handle(Geom_CylindricalSurface)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_ConicalSurface)) {
    Print(Handle(Geom_ConicalSurface)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_SphericalSurface)) {
    Print(Handle(Geom_SphericalSurface)::DownCast(S),OS,compact);
```

```
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_ToroidalSurface)) {
    Print(Handle(Geom_ToroidalSurface)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_SurfaceOfLinearExtrusion)) {
    Print(Handle(Geom_SurfaceOfLinearExtrusion)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_SurfaceOfRevolution)) {
    Print(Handle(Geom_SurfaceOfRevolution)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_BezierSurface)) {
    Print(Handle(Geom_BezierSurface)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_BSplineSurface)) {
    Print(Handle(Geom_BSplineSurface)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_RectangularTrimmedSurface)) {
    Print(Handle(Geom_RectangularTrimmedSurface)::DownCast(S),OS,compact);
  }
  else if ( TheType ==  STANDARD_TYPE(Geom_OffsetSurface)) {
    Print(Handle(Geom_OffsetSurface)::DownCast(S),OS,compact);
  }
  else {
    GeomTools::GetUndefinedTypeHandler()->PrintSurface(S,OS,compact);
    //if (!compact)
    //  OS << "***** Unknown Surface ********\n";
    //else
    //  cout << "***** Unknown Surface ********"<<endl;
  }
}
```

读入部分的程序代码如下所示：

```
//=======================================================================
//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;
}
```

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 的几何曲面已经有一个基类 Geom_Surface 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，参数表示的几何曲面并不会孤立存在，他总是依附于拓朴面中。在 OpenCascade 的 BRep 格式的文件中三维几何曲面共有十一种，通过将这十一种几何曲面输出，理解参数表示的几何曲面的数据结构。

通过查看其读写几何曲面的源程序，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