Open Cascade Data Exchange --- STL

eryar@163.com

摘要 Abstract：介绍了三维数据交换格式 STL 的组成，以及 Open Cascade 中对 STL 的读写。并将 Open Cascade 读进来的 STL 的三角面片在 OpenSceneGraph 中显示。

关键字 Key Words：STL, Open Cascade, OpenSceneGraph, Data Exchange

STL（the Stereo Lithograpy）是快速原型系统所应用的标准文件类型。它的目的是将几何数据发送到可以读取和解释这些数据的机器，这种机器可将模型转换成塑料的物理模型。STL 是用三角网格来表示三维模型的。STL 文件格式简单，只能描述三维物体的几何信息，不支持颜色、材质等信息，是三维打印机支持的最常见的文件格式。由于 STL 文件的网格表示方法只能表示封闭的形状，所以要转换的形状必须是实体，或封闭的面和体。STL 文件有两种：一种是明码（ASCII）格式，一种是二进制（Binary）格式。

一、STL 的明码（ASCII）格式

ASCII 格式的 STL 文件逐行给出三角面片的几何信息，每行以 1 个或 2 个关键字开头。STL 文件中的三角面片的信息单元 facet 是一个带法向方向的三角面片，STL 三维模型就是由这一系列的三角面片构成。整个 STL 文件的首行给出了文件路径及文件名。在一个 STL 文件中，每个 facet 由 7 行数据组成：facet normal 是三角面片指向实体外部的单位法矢量；outer loop 说明随后的 3 行数据分别是三角面片的 3 个顶点坐标，3 顶点沿指向实体外部的法矢量方向逆时针排列。

ASCII 格式的 STL 文件结构如下：

```
solid name
facet normal ni nj nk
  outer loop
    vertex v1x v1y v1z
    vertex v2x v2y v2z
    vertex v3x v3y v3z
  endloop
endfacet
endsolid name
```

说明如下：

```
1  solid filename stl      //文件路径及文件名
2  facet normal x y z      //三角面片法向量的3个分量值
3      outer loop
4          vertex x y z    //三角面片第一个顶点坐标
5          vertex x y z    //三角面片第二个顶点坐标
6          vertex x y z    //三角面片第三个顶点坐标
7      endloop
8  endfacet                //完成一个三角面片定义
9  ***
10 endsolid filename stl   //整个STL文件定义结束
```

下面给出由 Open Cascade 中导出的一个长方体的 STL 文件：
长方体的尺寸为长 200，宽 150，高 100，原点在一个角点上。

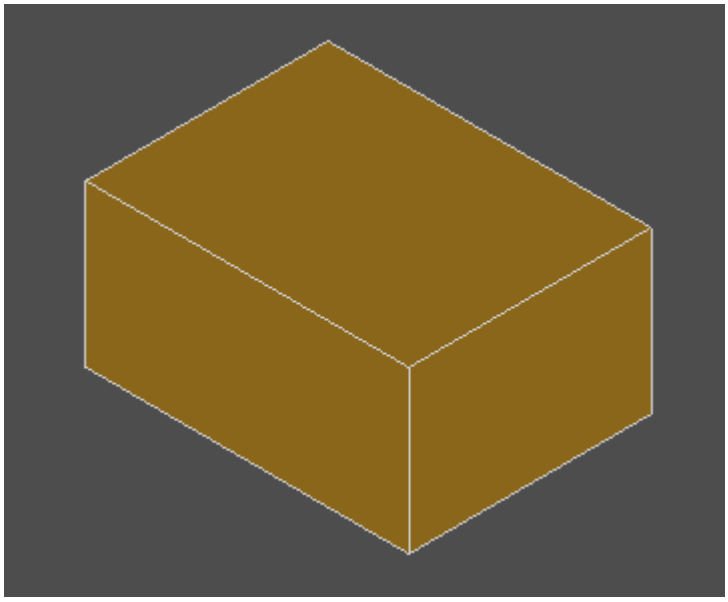

Figure 1.1 Box in Open Cascade

```
solid
  facet normal -1.000000e+000 -0.000000e+000 -0.000000e+000
    outer loop
      vertex    0.000000e+000    1.500000e+002    1.000000e+002
      vertex    0.000000e+000    1.500000e+002    0.000000e+000
      vertex    0.000000e+000    0.000000e+000    1.000000e+002
    endloop
  endfacet
  facet normal -1.000000e+000    0.000000e+000    0.000000e+000
    outer loop
      vertex    0.000000e+000    1.500000e+002    0.000000e+000
      vertex    0.000000e+000    0.000000e+000    0.000000e+000
      vertex    0.000000e+000    0.000000e+000    1.000000e+002
    endloop
  endfacet
  facet normal    1.000000e+000 -0.000000e+000    0.000000e+000
    outer loop
```

```
      vertex    2.000000e+002    0.000000e+000    1.000000e+002
      vertex    2.000000e+002    1.500000e+002    0.000000e+000
      vertex    2.000000e+002    1.500000e+002    1.000000e+002
   endloop
endfacet
facet normal    1.000000e+000 -0.000000e+000    0.000000e+000
   outer loop
      vertex    2.000000e+002    0.000000e+000    1.000000e+002
      vertex    2.000000e+002    0.000000e+000    0.000000e+000
      vertex    2.000000e+002    1.500000e+002    0.000000e+000
   endloop
endfacet
facet normal    0.000000e+000 -1.000000e+000    0.000000e+000
   outer loop
      vertex    0.000000e+000    0.000000e+000    0.000000e+000
      vertex    2.000000e+002    0.000000e+000    0.000000e+000
      vertex    2.000000e+002    0.000000e+000    1.000000e+002
   endloop
endfacet
facet normal    0.000000e+000 -1.000000e+000    0.000000e+000
   outer loop
      vertex    0.000000e+000    0.000000e+000    1.000000e+002
      vertex    0.000000e+000    0.000000e+000    0.000000e+000
      vertex    2.000000e+002    0.000000e+000    1.000000e+002
   endloop
endfacet
facet normal    0.000000e+000    1.000000e+000    0.000000e+000
   outer loop
      vertex    2.000000e+002    1.500000e+002    1.000000e+002
      vertex    2.000000e+002    1.500000e+002    0.000000e+000
      vertex    0.000000e+000    1.500000e+002    0.000000e+000
   endloop
endfacet
facet normal    0.000000e+000    1.000000e+000 -0.000000e+000
   outer loop
      vertex    2.000000e+002    1.500000e+002    1.000000e+002
      vertex    0.000000e+000    1.500000e+002    0.000000e+000
      vertex    0.000000e+000    1.500000e+002    1.000000e+002
   endloop
endfacet
facet normal    0.000000e+000    0.000000e+000 -1.000000e+000
   outer loop
      vertex    0.000000e+000    0.000000e+000    0.000000e+000
      vertex    0.000000e+000    1.500000e+002    0.000000e+000
```

```
      vertex    2.000000e+002    1.500000e+002    0.000000e+000
    endloop
  endfacet
  facet normal    0.000000e+000    0.000000e+000 -1.000000e+000
    outer loop
      vertex    2.000000e+002    0.000000e+000    0.000000e+000
      vertex    0.000000e+000    0.000000e+000    0.000000e+000
      vertex    2.000000e+002    1.500000e+002    0.000000e+000
    endloop
  endfacet
  facet normal    0.000000e+000    0.000000e+000    1.000000e+000
    outer loop
      vertex    2.000000e+002    1.500000e+002    1.000000e+002
      vertex    0.000000e+000    1.500000e+002    1.000000e+002
      vertex    0.000000e+000    0.000000e+000    1.000000e+002
    endloop
  endfacet
  facet normal -0.000000e+000    0.000000e+000    1.000000e+000
    outer loop
      vertex    2.000000e+002    1.500000e+002    1.000000e+002
      vertex    0.000000e+000    0.000000e+000    1.000000e+002
      vertex    2.000000e+002    0.000000e+000    1.000000e+002
    endloop
  endfacet
endsolid
```

由上面的 STL 明码文件可知，上述数据将一个长方体的 6 个面用 12 个三角形来表示。在 OpenSceneGraph 中显示效果如下图所示，分别为此长方体的实体渲染模式和线框渲染模式：

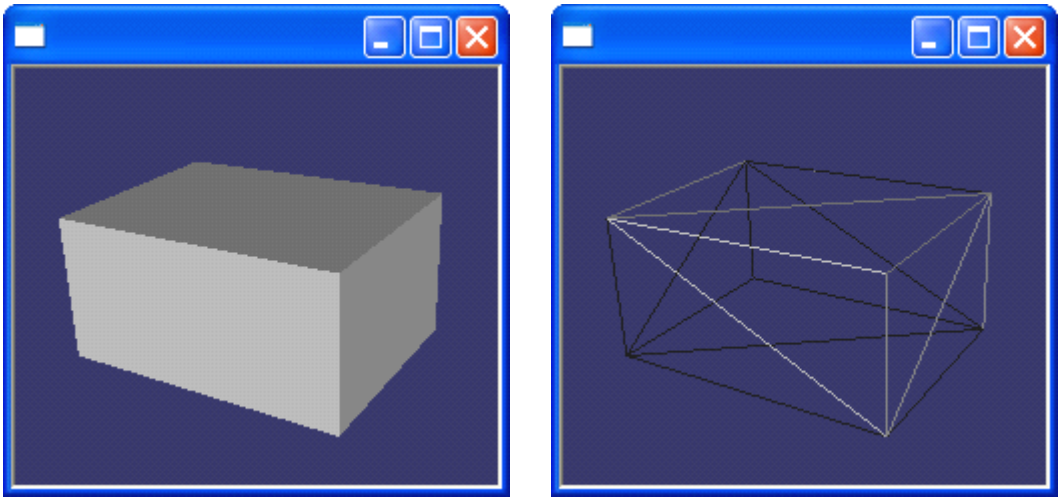

Figure 1.2 Shaded and Wireframe box in OpenSceneGraph

二、STL 的二进制（Binary）格式

二进制的 STL 文件用固定的字节数来给出三角面片的几何信息。文件起始 80 个字节是文件头，用于存贮零件名；紧接着 4 个字节的整数来描述模型的三角面片个数；后面逐个给出每个三角面片的几何信息。每个三角面片用固定的 50 个字节，依次是表示三角面片的法矢量的 3 个 4 字节浮点数；表示三角面片三个顶点的 3x3 个 4 字节浮点数；最后 2 个字节用来描述三角面片的属性信息。

```
UINT8[80] ---------- Header
UINT32 ------------ Number of triangles
foreach triangle
  REAL32[3] -------- Normal vector
  REAL32[3] -------- Vertex 1
  REAL32[3] -------- Vertex 2
  REAL32[3] -------- Vertex 3
  UINT16 ----------- Attribute byte count
end
```

三、OCC 中 STL 文件的读写 Read/Write STL in Open Cascade

在 Open Cascade 中 STL 文件的读写分别使用类：StlAPI_Reader/StlAPI_Writer 来实现。查看源程序可知，写 STL 文件的步骤如下：

- 遍历一个 TopoDS_Shape 所有的面 Face；
- 使用工具 BRep_Tool::Triangulation 将每个面 Face 三角面片化；
- 计算每个三角面片的法矢量；
- 将结果写入文件。

类 RWStl 对 STL 的读定也是有两种格式，即 ASCII 格式和 Binary 格式：

- RWStl::WriteBinary
- RWStl::WriteAscii
- RWStl::ReadBinary
- RWStl::ReadAscii

程序的具体实现可以查看 Open Cascade 源代码，将读写部分主要代码 RWStl.cxx 列出如下：

```
// File: RWStl.cxx
// Created:   Thu Oct 13 13:41:29 1994
// Author:    Marc LEGAY
//       <mle@bourdon>

// Copyright:   Matra Datavision

#include <RWStl.ixx>
#include <OSD_Protection.hxx>
#include <OSD_File.hxx>
#include <TCollection_AsciiString.hxx>
#include <Standard_NoMoreObject.hxx>
```

```
#include <Standard_TypeMismatch.hxx>
#include <Precision.hxx>
#include <StlMesh_MeshExplorer.hxx>
#include <OSD.hxx>
#include <OSD_Host.hxx>
#include <gp_XYZ.hxx>
#include <gp.hxx>
#include <stdio.h>
#include <gp_Vec.hxx>


// constants
static const int HEADER_SIZE         =  84;
static const int SIZEOF_STL_FACET     =  50;
static const int STL_MIN_FILE_SIZE    = 284;
static const int ASCII_LINES_PER_FACET =   7;

//============== ============================================================
//function : WriteInteger
//purpose  : writing a Little Endian 32 bits integer
//============================================================================

inline static void WriteInteger(OSD_File& ofile,const Standard_Integer value)
{
  union {
    Standard_Integer i;// don't be afraid, this is just an unsigned int
    char c[4];
  } bidargum;

  bidargum.i = value;

  Standard_Integer entier;

  entier  =  bidargum.c[0] & 0xFF;
  entier |= (bidargum.c[1] & 0xFF) << 0x08;
  entier |= (bidargum.c[2] & 0xFF) << 0x10;
  entier |= (bidargum.c[3] & 0xFF) << 0x18;

  ofile.Write((char *)&entier,sizeof(bidargum.c));
}

//============================================================================
//function : WriteDouble2Float
//purpose  : writing a Little Endian 32 bits float
```

```
//=============================================================================

inline static void WriteDouble2Float(OSD_File& ofile,Standard_Real value)
{
  union {
    Standard_ShortReal f;
    char c[4];
  } bidargum;

  bidargum.f = (Standard_ShortReal)value;

  Standard_Integer entier;

  entier  =  bidargum.c[0] & 0xFF;
  entier |= (bidargum.c[1] & 0xFF) << 0x08;
  entier |= (bidargum.c[2] & 0xFF) << 0x10;
  entier |= (bidargum.c[3] & 0xFF) << 0x18;

  ofile.Write((char *)&entier,sizeof(bidargum.c));
}



//=============================================================================
//function : readFloat2Double
//purpose  : reading a Little Endian 32 bits float
//=============================================================================

inline static Standard_Real ReadFloat2Double(OSD_File &aFile)
{
  union {
    Standard_Boolean i; // don't be afraid, this is just an unsigned int
    Standard_ShortReal f;
  }bidargum;

  char c[4];
  Standard_Address adr;
  adr = (Standard_Address)c;
  Standard_Integer lread;
  aFile.Read(adr,4,lread);
  bidargum.i  =  c[0] & 0xFF;
  bidargum.i |=  (c[1] & 0xFF) << 0x08;
  bidargum.i |=  (c[2] & 0xFF) << 0x10;
  bidargum.i |=  (c[3] & 0xFF) << 0x18;
```

```
  return (Standard_Real)(bidargum.f);
}



//=============================================================================
//function : WriteBinary
//purpose  : write a binary STL file in Little Endian format
//=============================================================================

Standard_Boolean RWStl::WriteBinary(const Handle(StlMesh_Mesh)& aMesh, const OSD_Path& aPath)
{

  OSD_File theFile = OSD_File (aPath);
  theFile.Build(OSD_WriteOnly,OSD_Protection());

  Standard_Real x1, y1, z1;
  Standard_Real x2, y2, z2;
  Standard_Real x3, y3, z3;

  //pgo  Standard_Real x,y,z;

  char sval[80];
  Standard_Integer NBTRIANGLES=0;
  unsigned int NBT;
  NBTRIANGLES = aMesh->NbTriangles();
  NBT = NBTRIANGLES ;
  theFile.Write ((Standard_Address)sval,80);
  WriteInteger(theFile,NBT);
//  theFile.Write ((Standard_Address)&NBT,4);
  int dum=0;
  StlMesh_MeshExplorer aMexp (aMesh);


  for (Standard_Integer nbd=1;nbd<=aMesh->NbDomains();nbd++) {
    for (aMexp.InitTriangle (nbd); aMexp.MoreTriangle (); aMexp.NextTriangle ()) {
      aMexp.TriangleVertices (x1,y1,z1,x2,y2,z2,x3,y3,z3);
      //pgo      aMexp.TriangleOrientation (x,y,z);
      gp_XYZ Vect12 ((x2-x1), (y2-y1), (z2-z1));
      gp_XYZ Vect13 ((x3-x1), (y3-y1), (z3-z1));
      gp_XYZ Vnorm = Vect12 ^ Vect13;
      Standard_Real Vmodul = Vnorm.Modulus ();
      if (Vmodul > gp::Resolution()) {
        Vnorm.Divide(Vmodul);
```

```
    }
    else {
      // si Vnorm est quasi-nul, on le charge a 0 explicitement
      Vnorm.SetCoord (0., 0., 0.);
    }

    WriteDouble2Float (theFile,Vnorm.X());
    WriteDouble2Float (theFile,Vnorm.Y());
    WriteDouble2Float (theFile,Vnorm.Z());

    WriteDouble2Float (theFile,x1);
    WriteDouble2Float (theFile,y1);
    WriteDouble2Float (theFile,z1);

    WriteDouble2Float (theFile,x2);
    WriteDouble2Float (theFile,y2);
    WriteDouble2Float (theFile,z2);

    WriteDouble2Float (theFile,x3);
    WriteDouble2Float (theFile,y3);
    WriteDouble2Float (theFile,z3);

    theFile.Write (&dum,2);

    }
  }

  theFile.Close ();
  return Standard_True;
}
//=======================================================================
//function : WriteAscii
//purpose  : write an ASCII STL file
//=======================================================================

Standard_Boolean RWStl::WriteAscii(const Handle(StlMesh_Mesh)& aMesh, const OSD_Path& aPath)
{
  OSD_File theFile = OSD_File (aPath);
  theFile.Build(OSD_WriteOnly,OSD_Protection());
  TCollection_AsciiString buf = TCollection_AsciiString ("solid\n");
  theFile.Write (buf,buf.Length());buf.Clear();

  Standard_Real x1, y1, z1;
  Standard_Real x2, y2, z2;
```

```
  Standard_Real x3, y3, z3;

  //pgo  Standard_Real x,y,z;

  char sval[16];

  StlMesh_MeshExplorer aMexp (aMesh);

  for (Standard_Integer nbd=1;nbd<=aMesh->NbDomains();nbd++) {
    for (aMexp.InitTriangle (nbd); aMexp.MoreTriangle (); aMexp.NextTriangle ()) {
      aMexp.TriangleVertices (x1,y1,z1,x2,y2,z2,x3,y3,z3);

//      Standard_Real x, y, z;
//      aMexp.TriangleOrientation (x,y,z);

      gp_XYZ Vect12 ((x2-x1), (y2-y1), (z2-z1));
      gp_XYZ Vect23 ((x3-x2), (y3-y2), (z3-z2));
      gp_XYZ Vnorm = Vect12 ^ Vect23;
      Standard_Real Vmodul = Vnorm.Modulus ();
      if (Vmodul > gp::Resolution()){
      Vnorm.Divide (Vmodul);
      }
      else {
      // si Vnorm est quasi-nul, on le charge a 0 explicitement
      Vnorm.SetCoord (0., 0., 0.);
      }
      buf += " facet normal ";
      sprintf (sval,"% 12e",Vnorm.X());
      buf += sval;
      buf += " ";
      sprintf (sval,"% 12e",Vnorm.Y());
      buf += sval;
      buf += " ";
      sprintf (sval,"% 12e",Vnorm.Z());
      buf += sval;
      buf += '\n';
      theFile.Write (buf,buf.Length());buf.Clear();
      buf += "   outer loop\n";
      theFile.Write (buf,buf.Length());buf.Clear();

      buf += "    vertex ";
      sprintf (sval,"% 12e",x1);
      buf += sval;
      buf += " ";
```

```
      sprintf (sval,"% 12e",y1);
      buf += sval;
      buf += " ";
      sprintf (sval,"% 12e",z1);
      buf += sval;
      buf += '\n';
      theFile.Write (buf,buf.Length());buf.Clear();

      buf += "    vertex ";
      sprintf (sval,"% 12e",x2);
      buf += sval;
      buf += " ";
      sprintf (sval,"% 12e",y2);
      buf += sval;
      buf += " ";
      sprintf (sval,"% 12e",z2);
      buf += sval;
      buf += '\n';
      theFile.Write (buf,buf.Length());buf.Clear();

      buf += "    vertex ";
      sprintf (sval,"% 12e",x3);
      buf += sval;
      buf += " ";
      sprintf (sval,"% 12e",y3);
      buf += sval;
      buf += " ";
      sprintf (sval,"% 12e",z3);
      buf += sval;
      buf += '\n';
      theFile.Write (buf,buf.Length());buf.Clear();
      buf += "   endloop\n";
      theFile.Write (buf,buf.Length());buf.Clear();
      buf += " endfacet\n";
      theFile.Write (buf,buf.Length());buf.Clear();
    }
  }

  buf += "endsolid\n";
  theFile.Write (buf,buf.Length());buf.Clear();

  theFile.Close ();
  return Standard_True;
}
```

```
//=============================================================================
//function : ReadFile
//Design   :
//Warning  :
//=============================================================================

Handle_StlMesh_Mesh RWStl::ReadFile(const OSD_Path& aPath)
{
  OSD_File file = OSD_File (aPath);
  file.Open(OSD_ReadOnly,OSD_Protection(OSD_RWD,OSD_RWD,OSD_RWD,OSD_RWD));
  Standard_Boolean IsAscii;
  unsigned char str[128];
  Standard_Integer lread,i;
  Standard_Address ach;
  ach = (Standard_Address)str;

  // we skip the header which is in Ascii for both modes
  file.Read(ach,HEADER_SIZE,lread);

  // we read 128 characters to detect if we have a non-ascii char
  file.Read(ach,sizeof(str),lread);

  IsAscii = Standard_True;
  for (i = 0; i< lread && IsAscii; ++i) {
    if (str[i] > '~') {
      IsAscii = Standard_False;
    }
  }

  printf("%s\n",(IsAscii?"ascii":"binary"));

  file.Close();

  if (IsAscii) {
    return RWStl::ReadAscii (aPath);
  } else {
    return RWStl::ReadBinary (aPath);
  }
}

//=============================================================================
//function : ReadBinary
//Design   :
//Warning  :
```

```
//=============================================================================

Handle_StlMesh_Mesh RWStl::ReadBinary(const OSD_Path& aPath)
{
  Standard_Integer NBFACET;
  Standard_Integer ifacet;
  Standard_Real fx,fy,fz,fx1,fy1,fz1,fx2,fy2,fz2,fx3,fy3,fz3;
  Standard_Integer i1,i2,i3,lread;
  char buftest[5];
  Standard_Address adr;
  adr = (Standard_Address)buftest;

  // Open the file
  OSD_File theFile = OSD_File(aPath);
  theFile.Open(OSD_ReadOnly,OSD_Protection(OSD_RWD,OSD_RWD,OSD_RWD,OSD_RWD));

  // the size of the file (minus the header size)
  // must be a multiple of SIZEOF_STL_FACET

  // compute file size
  Standard_Integer filesize = theFile.Size();

  if ( (filesize - HEADER_SIZE) % SIZEOF_STL_FACET !=0
     || (filesize < STL_MIN_FILE_SIZE)) {
    Standard_NoMoreObject::Raise("RWStl::ReadBinary (wrong file size)");
  }

  // don't trust the number of triangles which is coded in the file
  // sometimes it is wrong, and with this technique we don't need to swap endians for integer
  NBFACET = ((filesize - HEADER_SIZE) / SIZEOF_STL_FACET);

  // skip the header
  theFile.Seek(HEADER_SIZE,OSD_FromBeginning);

  // create the StlMesh_Mesh object
  Handle(StlMesh_Mesh) ReadMesh = new StlMesh_Mesh ();
  ReadMesh->AddDomain ();

  for (ifacet=1; ifacet<=NBFACET; ++ifacet) {
    // read normal coordinates
    fx = ReadFloat2Double(theFile);
    fy = ReadFloat2Double(theFile);
    fz = ReadFloat2Double(theFile);
```

```
    // read vertex 1
    fx1 = ReadFloat2Double(theFile);
    fy1 = ReadFloat2Double(theFile);
    fz1 = ReadFloat2Double(theFile);

    // read vertex 2
    fx2 = ReadFloat2Double(theFile);
    fy2 = ReadFloat2Double(theFile);
    fz2 = ReadFloat2Double(theFile);

    // read vertex 3
    fx3 = ReadFloat2Double(theFile);
    fy3 = ReadFloat2Double(theFile);
    fz3 = ReadFloat2Double(theFile);

    i1 = ReadMesh->AddOnlyNewVertex (fx1,fy1,fz1);
    i2 = ReadMesh->AddOnlyNewVertex (fx2,fy2,fz2);
    i3 = ReadMesh->AddOnlyNewVertex (fx3,fy3,fz3);
    ReadMesh->AddTriangle (i1,i2,i3,fx,fy,fz);

    // skip extra bytes
    theFile.Read(adr,2,lread);
  }

  theFile.Close ();
  return ReadMesh;

}
//=======================================================================
//function : ReadAscii
//Design   :
//Warning  :
//=======================================================================

Handle_StlMesh_Mesh RWStl::ReadAscii(const OSD_Path& aPath)
{
  TCollection_AsciiString filename;
  long ipos;
  Standard_Integer nbLines = 0;
  Standard_Integer nbTris = 0;
  Standard_Integer iTri;
  Standard_ShortReal x[4],y[4],z[4];
  Standard_Integer i1,i2,i3;
  Handle(StlMesh_Mesh) ReadMesh;
```

```
  aPath.SystemName( filename);

  // Open the file
  FILE* file = fopen(filename.ToCString(),"r");

  fseek(file,0L,SEEK_END);

  long filesize = ftell(file);

  fclose(file);
  file = fopen(filename.ToCString(),"r");



  // count the number of lines
  for (ipos = 0; ipos < filesize; ++ipos) {
       if (getc(file) == '\n')
        nbLines++;
  }

  // compute number of triangles
  nbTris = (nbLines / ASCII_LINES_PER_FACET);

  // go back to the beginning of the file
//  fclose(file);
//  file = fopen(filename.ToCString(),"r");
  rewind(file);

  // skip header
  while (getc(file) != '\n');

  cout<< "start mesh\n";
  ReadMesh = new StlMesh_Mesh();
  ReadMesh->AddDomain();

  // main reading
  for (iTri = 0; iTri < nbTris; ++iTri) {
    // reading the facet normal
    fscanf(file,"%*s %*s %f %f %f\n",&x[0],&y[0],&z[0]);

    // skip the keywords "outer loop"
    fscanf(file,"%*s %*s");
```

```
    // reading vertex
    fscanf(file,"%*s %f %f %f\n",&x[1],&y[1],&z[1]);
    fscanf(file,"%*s %f %f %f\n",&x[2],&y[2],&z[2]);
    fscanf(file,"%*s %f %f %f\n",&x[3],&y[3],&z[3]);

    // here the facet must be built and put in the mesh datastructure

    i1 = ReadMesh->AddOnlyNewVertex
((Standard_Real)x[1],(Standard_Real)y[1],(Standard_Real)z[1]);
    i2 = ReadMesh->AddOnlyNewVertex
((Standard_Real)x[2],(Standard_Real)y[2],(Standard_Real)z[2]);
    i3 = ReadMesh->AddOnlyNewVertex
((Standard_Real)x[3],(Standard_Real)y[3],(Standard_Real)z[3]);
    ReadMesh->AddTriangle
(i1,i2,i3,(Standard_Real)x[0],(Standard_Real)y[0],(Standard_Real)z[0]);

    // skip the keywords "endloop"
    fscanf(file,"%*s");

    // skip the keywords "endfacet"
    fscanf(file,"%*s");

  }

  cout<< "end mesh\n"<<endl;
  fclose(file);
  return ReadMesh;

}
```

程序开始定义了一些常量：

```
// constants
static const int HEADER_SIZE                =   84;
static const int SIZEOF_STL_FACET           =   50;
static const int STL_MIN_FILE_SIZE          = 284;
static const int ASCII_LINES_PER_FACET =     7;
```

分别对应二进制文件中相关信息，即文件头 84 个字节，每个三角面片 50 个字节，STL 文件最小为 284 字节。ASCII 的 STL 中每个三角面有 7 行。

在数据的读写过程中，对数据进行了小端转换。将 double 数据转换成小端表示的代码如下所示：

```
//=============================================================================
//function : WriteDouble2Float
```

```
//purpose    : writing a Little Endian 32 bits float
//=========================================================================

inline static void WriteDouble2Float(OSD_File& ofile,Standard_Real value)
{
   union {
      Standard_ShortReal f;
      char c[4];
   } bidargum;

   bidargum.f = (Standard_ShortReal)value;

   Standard_Integer entier;

   entier    =    bidargum.c[0] & 0xFF;
   entier |= (bidargum.c[1] & 0xFF) << 0x08;
   entier |= (bidargum.c[2] & 0xFF) << 0x10;
   entier |= (bidargum.c[3] & 0xFF) << 0x18;

   ofile.Write((char *)&entier,sizeof(bidargum.c));
}
```

使用联合体（union）来处理显得很优雅。关于大端、小端的相关信息请参考：
http://www.cppblog.com/tx7do/archive/2009/01/06/71276.html

四、在 OpenSceneGraph 中显示 STL

结合 OpenCascade 中对 STL 文件读写的功能和 OpenSceneGraph 的显示功能，将 STL 读取所得数据进行显示。源程序如下所示：

```
// Open Cascade
#include <gp_Vec.hxx>
#include <OSD_Path.hxx>
#include <RWStl.hxx>
#include <StlMesh_Mesh.hxx>
#include <StlMesh_MeshExplorer.hxx>

#pragma comment(lib, "TKernel.lib")
#pragma comment(lib, "TKMath.lib")
#pragma comment(lib, "TKSTL.lib")

// OpenSceneGraph
#include <osgDB/ReadFile>
#include <osgViewer/Viewer>
#include <osgViewer/ViewerEventHandlers>
#include <osgGA/StateSetManipulator>

#pragma comment(lib, "osgd.lib")
#pragma comment(lib, "osgDbd.lib")
#pragma comment(lib, "osgGAd.lib")
#pragma comment(lib, "osgViewerd.lib")

osg::Node* readSTLFile(const std::string& fileName)
{
    osg::Group* root = new osg::Group();

    OSD_Path stlFile(fileName.c_str());
    Handle_StlMesh_Mesh stlMesh = RWStl::ReadFile(stlFile);
    Standard_Integer nDomains = stlMesh->NbDomains();
    StlMesh_MeshExplorer meshExplorer(stlMesh);

    Standard_Real x[3] = {0};
    Standard_Real y[3] = {0};
    Standard_Real z[3] = {0};
    Standard_Real n[3] = {0};

    gp_XYZ p1;
    gp_XYZ p2;
    gp_XYZ p3;
    gp_XYZ normal;
```

```
    gp_Vec vecNormal;

    for (int i = 1; i <= nDomains; i++)
    {
       for (meshExplorer.InitTriangle(i); meshExplorer.MoreTriangle();
meshExplorer.NextTriangle())
       {
          meshExplorer.TriangleVertices(x[0], y[0], z[0], x[1], y[1], z[1], x[2], y[2], z[2]);
          meshExplorer.TriangleOrientation(n[0], n[1], n[2]);

          p1.SetCoord(x[0], y[0], z[0]);
          p2.SetCoord(x[1], y[1], z[1]);
          p3.SetCoord(x[2], y[2], z[2]);
          normal.SetCoord(n[0], n[1], n[2]);

          //gp_Vec vec12((x[1] - x[0]), (y[1] - y[0]), (z[1] - z[0]));
          //gp_Vec vec23((x[2] - x[1]), (y[2] - y[1]), (z[2] - z[1]));
          //vecNormal = vec12.Crossed(vec23).Normalized();

          osg::ref_ptr<osg::Geode> geode = new osg::Geode();
          osg::ref_ptr<osg::Geometry> triGeom = new osg::Geometry();
          osg::ref_ptr<osg::Vec3Array> vertices = new osg::Vec3Array();
          osg::ref_ptr<osg::Vec3Array> normals = new osg::Vec3Array();

          vertices->push_back(osg::Vec3(x[0], y[0], z[0]));
          vertices->push_back(osg::Vec3(x[1], y[1], z[1]));
          vertices->push_back(osg::Vec3(x[2], y[2], z[2]));

          normals->push_back(osg::Vec3(n[0], n[1], n[2]));

          triGeom->setVertexArray(vertices.get());
          triGeom->addPrimitiveSet(new osg::DrawArrays(osg::PrimitiveSet::TRIANGLES, 0,
vertices->size()));

          triGeom->setNormalArray(normals);
          triGeom->setNormalBinding(osg::Geometry::BIND_PER_PRIMITIVE);

          geode->addDrawable(triGeom);

          root->addChild(geode);
       }
    }

    return root;
```

```
}

int main(int argc, char* argv[])
{
    osgViewer::Viewer myViewer;
    osg::ref_ptr<osg::Group> root = new osg::Group();

    //root->addChild(readSTLFile("D:\\OpenCASCADE6.5.0\\data\\stl\\propeller.stl"));
    root->addChild(readSTLFile("D:\\OpenCASCADE6.5.0\\data\\stl\\sh1.stl"));
    //root->addChild(readSTLFile("D:\\OpenCASCADE6.5.0\\data\\stl\\motor.stl"));
    //root->addChild(readSTLFile("D:\\box.stl"));

    myViewer.setSceneData(root);

    myViewer.addEventHandler(new
osgGA::StateSetManipulator(myViewer.getCamera()->getOrCreateStateSet()));
    myViewer.addEventHandler(new osgViewer::StatsHandler);
    myViewer.addEventHandler(new osgViewer::WindowSizeHandler);

    return myViewer.run();
     }
```

以下所示为 OpenCascade 提供的几个 STL 文件在 OpenSceneGraph 中显示的效果：

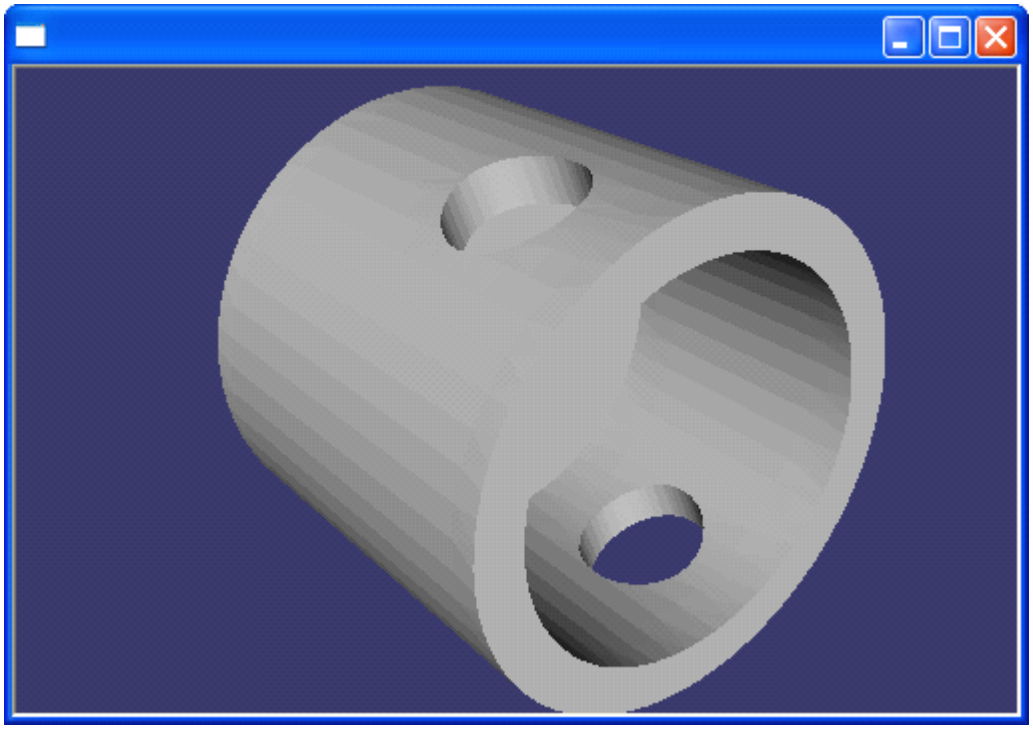

Figure 4.1 Shaded Piston

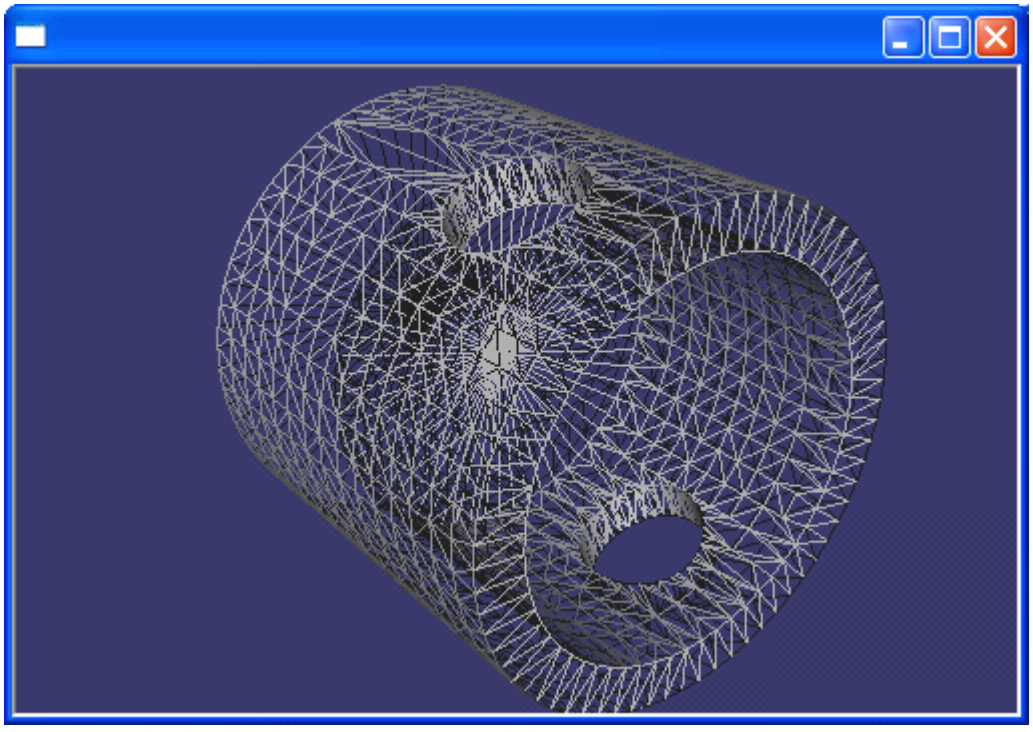

Figure 4.2 Wireframe Piston

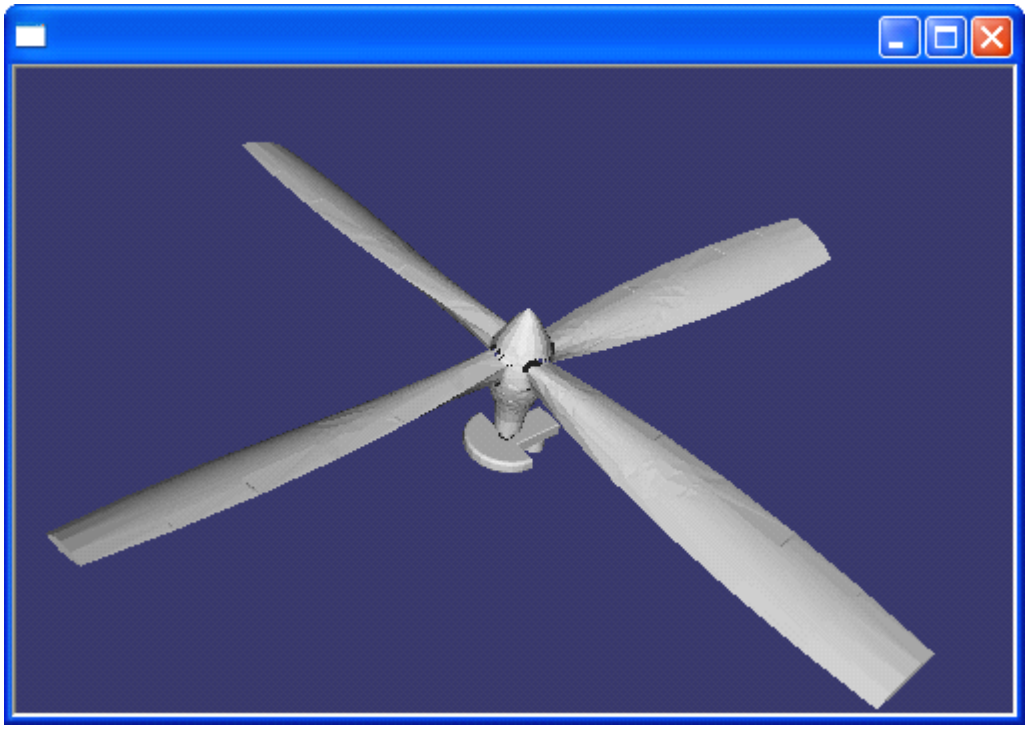

Figure 4.3 Shaded Propeller

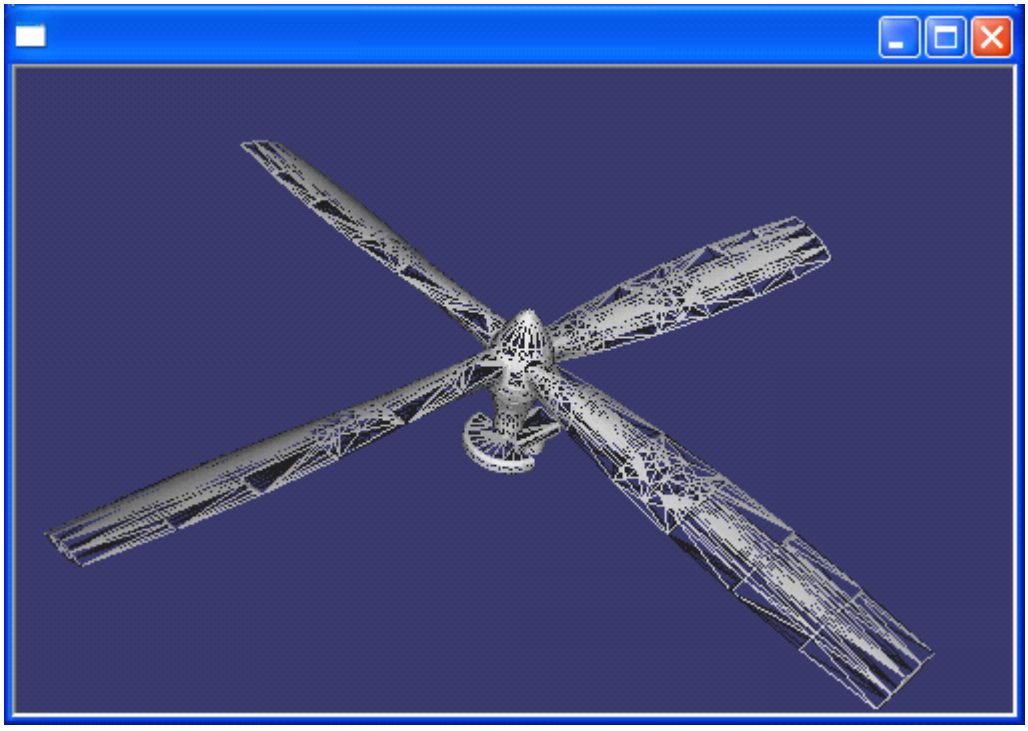

Figure 4.4 Wireframe Propeller

五、结论

通过使用 OpenCascade 的类 RWStl 来读取 STL 格式的文件，理解了 STL 文件格式；通过将读取的三角面面片数据在 OpenSceneGraph 中显示，对三维物体在计算机中的表示有了感性的认识。

六、参考资料

1. OpenCascade 中类 RWStl.cxx
2. OpenCascade 中 STL 模型数据
3. 字节序、大端、小端：http://www.cppblog.com/tx7do/archive/2009/01/06/71276.html