OpenCASCADE General Transformation

eryar@163.com

Abstract. OpenCASCADE provides a general transformation class: gp_GTrsf. It can be a transformation from gp, an affinity, or you can define your own transformation giving the matrix of transformation. The general transformation contains the vectorial part of the transformation and the translation part. A GTrsf transformation is only applicable to coordinates. Be careful if you apply such a transformation to all points of a geometric object, as this can change the nature of the object and thus render it incoherent. Typically a circle is transformed into an ellipse by an affinity transformation. To avoid modifying the nature of an object, use a gp_Trsf transformation instead, as objects of this class respect the nature of geometric objects.

Key Words. OpenCASCADE, Transformation, Affinity Transformation

1. Introduction

仿射变换（Affinity Transformation）是指线性变换后接着平移。因此，仿射变换的集合是线性变换的超集，任何线性变换都是仿射变换，但不是所有的仿射变换都是线性变换。

仿射变换的定义如下：在空间直角坐标系下，点（x,y,z)与点(x’, y’,z’)之间的变换

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = A\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \qquad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \qquad |A| \neq 0$$

称为仿射变换。如果采用特殊的齐次坐标来表达，仿射变换也可用下列形式：

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} & & & b1 \\ & A & & b2 \\ & & & b3 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

空间仿射变换是把平面变换到平面，直线变换到直线。两个平行平面的像也是平行的。共线三点的的简单比是不变量。平行六面体的体积是权为 1 的相对不变量。

OpenCASCADE 的 TKMath 库中提供了这上仿射变换类 gp_GTrsf，它能执行比 gp_Trsf 更通用的变换。对于 TopoDS_Shape，OpenCASCADE 分别提供了如下两个类进行变换：

- BRepBuilderAPI_GTransform
- BRepBuilderAPI_Transform

本文在 OpenCASCADE Draw Test Harness 中给出这两个类实现变换的结果。如果不想改变几何的特性，只想改变模型的位置或朝向，建议采用 BRepBuilderAPI_Transform。

2. BRepBuilderAPI_Transform

OpenCASCADE 中使用算法 BRepBuilderAPI_Transform 来实现：平移、旋转、缩放及镜像变换。在 Draw Test Harness 中实现的函数代码如下所示：

```
static Standard_Integer transform(Draw_Interpretor& di,Standard_Integer n,const
char** a)
{
  if (n <= 1) return 1;

  gp_Trsf T;
  Standard_Integer last = n;
  const char* aName = a[0];

  Standard_Boolean isBasic = Standard_False;

  if (!strcmp(aName,"reset")) {
  }
  else {
    isBasic = (aName[0] == 'b');
    aName++;

    if (!strcmp(aName,"move")) {
      if (n < 3) return 1;
      TopoDS_Shape SL = DBRep::Get(a[n-1]);
      if (SL.IsNull()) return 0;
      T = SL.Location().Transformation();
      last = n-1;
    }
    else if (!strcmp(aName,"translate")) {
      if (n < 5) return 1;

T.SetTranslation(gp_Vec(Draw::Atof(a[n-3]),Draw::Atof(a[n-2]),Draw::Atof(a[n-1]
)));
      last = n-3;
    }
    else if (!strcmp(aName,"rotate")) {
      if (n < 9) return 1;

T.SetRotation(gp_Ax1(gp_Pnt(Draw::Atof(a[n-7]),Draw::Atof(a[n-6]),Draw::Atof(a
[n-5])),

gp_Vec(Draw::Atof(a[n-4]),Draw::Atof(a[n-3]),Draw::Atof(a[n-2]))),
                    Draw::Atof(a[n-1])* (M_PI / 180.0));
      last = n-7;
    }
    else if (!strcmp(aName,"mirror")) {
      if (n < 8) return 1;

T.SetMirror(gp_Ax2(gp_Pnt(Draw::Atof(a[n-6]),Draw::Atof(a[n-5]),Draw::Atof(a[n
-4])),
gp_Vec(Draw::Atof(a[n-3]),Draw::Atof(a[n-2]),Draw::Atof(a[n-1]))));
      last = n-6;
    }
    else if (!strcmp(aName,"scale")) {
      if (n < 6) return 1;

T.SetScale(gp_Pnt(Draw::Atof(a[n-4]),Draw::Atof(a[n-3]),Draw::Atof(a[n-2])),Dr
aw::Atof(a[n-1]));
      last = n-4;
    }
  }

  if (T.Form() == gp_Identity || isBasic) {
    TopLoc_Location L(T);
    for (Standard_Integer i = 1; i < last; i++) {
      TopoDS_Shape S = DBRep::Get(a[i]);
      if (S.IsNull())
        di << a[i] << " is not a valid shape\n";
      else
        DBRep::Set(a[i],S.Located(L));
    }
  }
  else {
    BRepBuilderAPI_Transform trf(T);
    for (Standard_Integer i = 1; i < last; i++) {
      TopoDS_Shape S = DBRep::Get(a[i]);
      if (S.IsNull()) {
        di << a[i] << " is not a valid shape\n";
      }
      else {
        trf.Perform(S);
        if (!trf.IsDone())
          return 1;
        DBRep::Set(a[i],trf.Shape());
      }
    }
  }
  return 0;
}
```

下面给出应用 Tcl 脚本来实现这些变换的例子：

```
# make rotated copies of a sphere in between two cylinders
# create a file source toto.tcl
# toto.tcl code:
pload ALL

#create a sphere
psphere s 3
ttranslate s 25 0 12.

for {set i 0} {$i < 360} {incr i 20} {
    copy s s$i
    trotate s$i 0 0 0 0 0 1 $i

    vdisplay s$i
}
# create two cylinders
pcylinder c1 30 5
copy c1 c2
ttranslate c2 0 0 20
vdisplay c1 c2 s
```

脚本运行效果如下图所示：

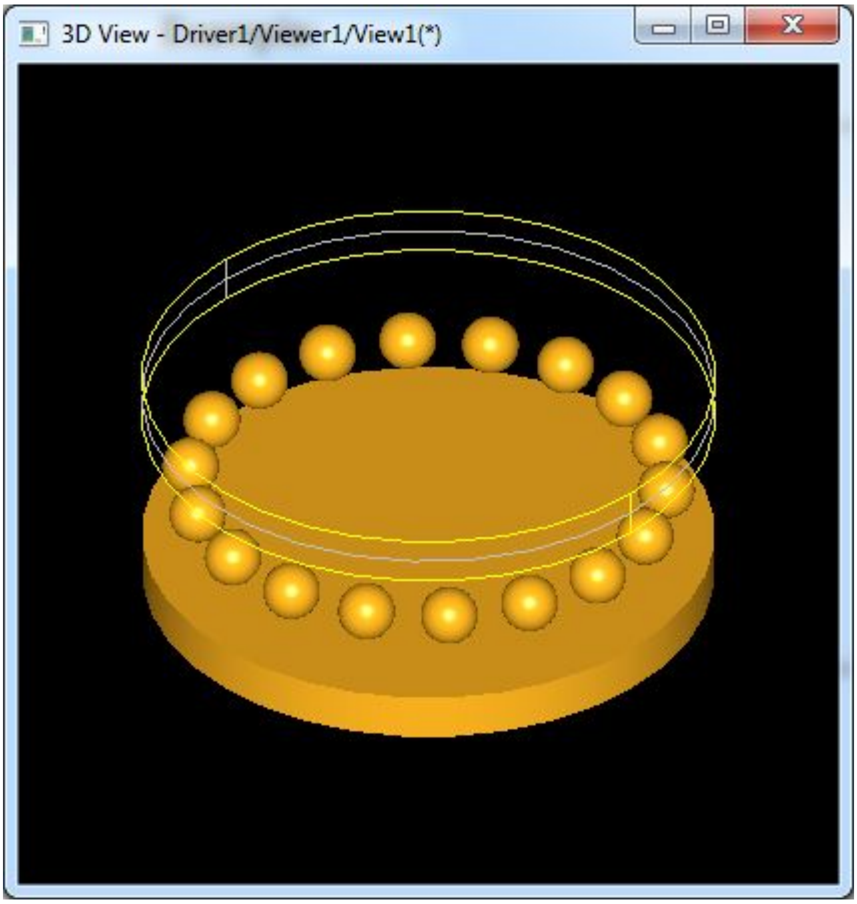


Figure 2.1 Transform Tcl demo

从 Draw 中实现的函数来看，移动、旋转及缩放变换都是使用类 BRepBuilderAPI_Transformation 来实现。Tcl 脚本中先创建出一个球体，再平移后，复制 13 份，最后又创建出两个圆柱体。如果要对 TopoDS_Shape 进行变换且不改变其中的几何性质，建议都使用这个类来完成。

3. BRepBuilderAPI_GTransform

在 OpenCASCADE 也可使用仿射变换 BRepBuilderAPI_GTransform 来对形状实现上述变换操作，还可提供变形的变换，因此仿射变换是更一般的变换方法。在 Draw 中实现的函数代码如下所示：

```
///=======================================================================
// gtransform
//=======================================================================
static  Standard_Integer  deform(Draw_Interpretor&  di,Standard_Integer  n,const
char** a)
{
  if (n <= 1) return 1;

  Standard_Integer last = n;

  gp_Trsf T;
  gp_GTrsf GT(T);

//                                                                          gp_Mat
rot(Draw::Atof(a[last-3]),0,0,0,Draw::Atof(a[last-2]),0,0,0,Draw::Atof(a[last-1]
));
  gp_Mat rot(Draw::Atof(a[3]),0,0,0,Draw::Atof(a[4]),0,0,0,Draw::Atof(a[5]));
  GT.SetVectorialPart(rot);
  last -= 3;
  BRepBuilderAPI_GTransform gtrf(GT);
  BRepBuilderAPI_NurbsConvert nbscv;
  //  for (Standard_Integer i = 1; i < last; i++) {
  //    TopoDS_Shape S = DBRep::Get(a[i]);
  TopoDS_Shape S = DBRep::Get(a[2]);
  if (S.IsNull()) {
    //cout << a[2] << " is not a valid shape" << endl;
    di << a[2] << " is not a valid shape" << "\n";
  }
  else {
    gtrf.Perform(S);
    if (gtrf.IsDone()){
      DBRep::Set(a[1],gtrf.Shape());
    }
    else {
      return 1;
    }
  }

  return 0;
}
```

根据仿射变换的定义，给定一个球面的数学表达式：

$$\boxed{x^2 + y^2 + z^2 = 1}$$

应用如下的仿射变换，将会得到一个椭球面：

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 & b1 \\ 0 & b & 0 & b2 \\ 0 & 0 & c & b3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad abc \neq 0$$

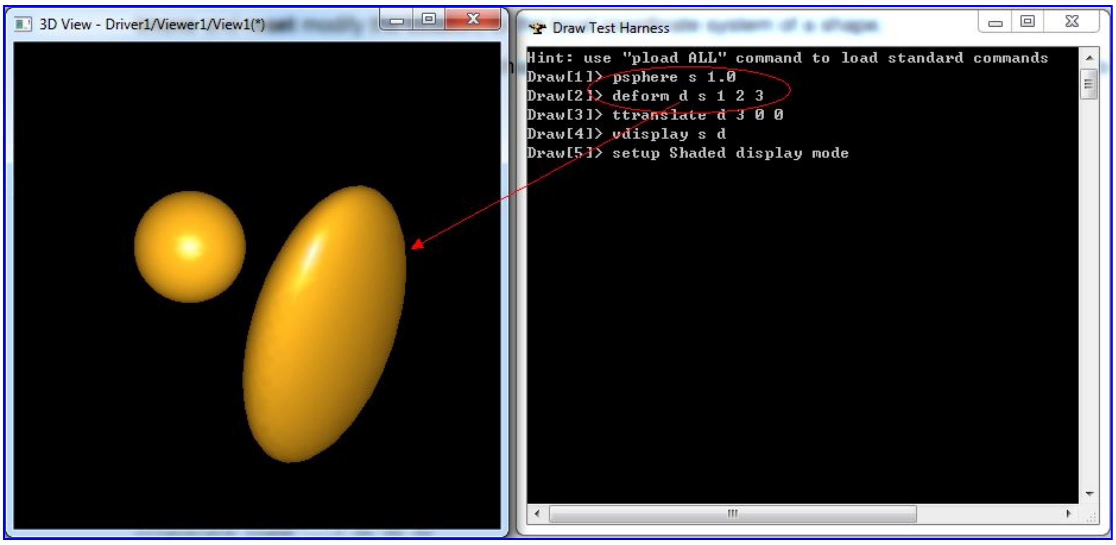
由变换公式解得：

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 1/a & 0 & 0 & 0 \\ 0 & 1/b & 0 & 0 \\ 0 & 0 & 1/c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

将它代入球面方程得到：

$$\frac{x'^2}{a^2} + \frac{y'^2}{b^2} + \frac{z'^2}{c^2} = 1$$

在 Draw 中使用 BRepBuilderAPI_GTransform 变换得到如下图所示：



```
Hint: use "pload ALL" command to load standard commands
Draw[1]> psphere s 1.0
Draw[2]> deform d s 1 2 3
Draw[3]> ttranslate d 3 0 0
Draw[4]> vdisplay s d
Draw[5]> setup Shaded display mode
```

Figure 3.1 Shape Deformation

关于仿射变换有个重要定理：一般仿射变换是正交变换、沿着三个互相正交方向的压缩或放大和平移这三者的乘积。上述命令的实现代码就是设置了仿射矩阵中的 a,b 和 c 值，从而达到对模型变形的效果。

4. Conclusion

在三维建模软件中经常需要对模型的位置和其朝向进行变换，如果不想改变模型中的几何特性，在 OpenCASCADE 中建议使用类 BRepBuilderAPI_Transform 来实现。如果需要对模型进行更通用的变换即仿射变换，可以使用类 BRepBuilderAPI_GTransform 来实现。使用此类后，会改变模型中的几何特性，必须谨慎使用。

5. References

1. OpenCASCADE, OpenCASCADE Draw Test Harness User Guide, 2014
2. 苏步表, 华宣积. 应用几何教程. 复旦大学出版社. 2012
3. Fletcher Dunn. 3D Math Primer for Graphics and Game Development. Wordware. 2002