# Locations Section of OpenCascade BRep

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。本文主要通过对 BRep 文件中的 Locations 部分的读写代码进行分析，来完全理解 OpenCascade 中的 Location 部分。

关键字 Key Words：OpenCascade, BRep Format, Location, Location Set

## 一、引言 Introduction

为了跟踪拓朴体的位置（Shape Location），每个形状都有一个局部坐标系。局部坐标系可以通过如下两种方式来表示：

- 一个右手法则表示的三个互相垂直的向量，对应的类是 gp_Ax2；
- 一个相对于世界坐标系的变换（the transformation of coordinates between local and global references frames），对应的类是 gp_Trsf；

类 TopLoc_Location 表示了初等矩阵经过一系列变换后得到的坐标系，保存累积变换后的结果避免了矩阵变换的重新计算。

## 二、<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。

## 三、示例程序

通过分析 Location 数据输出和读取的程序，可以完全理解 Location 类的作用。Location 的输出与读取都是通过类 TopTools_LocationSet 来实现的。调试跟踪其代码，可以理解其具体实现了。

### 3.1 输出位置数据 Output Location data

将 Location 中的数据输出有两种方式，一种是在 Debug 模式下，可以输出到屏幕显示；一种是输出到文件。输出到文件还可以被读取。示例程序如下所示：

```
/*
*    Copyright (c) 2013 eryar All Rights Reserved.
*
*        File    : Main.cpp
*        Author  : eryar@163.com
*        Date    : 2013-11-16 20:08
*        Version : 1.0v
*
*    Description : Keeping track of shape location.
*                  The TopLoc_Location class represents a marker composed of
*                  references to elementary markers. The resulting cumulative
*                  transformation is stored in order to avoid recalculating the
*                  sum of the transformations for the whole list.
*/

#define WNT
#include <gp_Trsf.hxx>
#include <TopLoc_Location.hxx>
#include <TopTools_LocationSet.hxx>

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

int main(void)
{
    ofstream dumpFile("LocationTest.txt");

    TopTools_LocationSet locationSet;

    // 1. Null transformation, identity matrix.
    gp_Trsf trsfNull;
    TopLoc_Location locationNull(trsfNull);

    locationSet.Add(locationNull);
    locationSet.Add(locationNull.Powered(2));

    // 2. Translate transformation.
```

```
    gp_Trsf trsfTranslate;
    trsfTranslate.SetTranslation(gp_Vec(100, 200, 0));
    TopLoc_Location locationTranslate(trsfTranslate);

    locationSet.Add(locationTranslate);
    locationSet.Add(locationTranslate.Powered(3));

    // 3. Rotate transformation.
    gp_Trsf trsfRotate;
    trsfRotate.SetRotation(gp::OX(), M_PI_2);
    TopLoc_Location locationRotate(trsfRotate);

    locationSet.Add(locationRotate);
    locationSet.Add(locationRotate.Powered(6));

    // dump the location set and write to file.
    locationSet.Dump(std::cout);
    locationSet.Write(dumpFile);

    return 0;
}
```

Debug 模式下屏幕上输出结果为：

```
 -------
 Dump of 6 Locations
 -------

    1 :
Elementary location
(                   1                   0                   0                   0   )
(                   0                   1                   0                   0   )
(                   0                   0                   1                   0   )
    2 :
Complex : L1^2
(                   1                   0                   0                   0   )
(                   0                   1                   0                   0   )
(                   0                   0                   1                   0   )
    3 :
Elementary location
(                   1                   0                   0                 100   )
(                   0                   1                   0                 200   )
(                   0                   0                   1                   0   )
    4 :
Complex : L3^3
(                   1                   0                   0                 300   )
(                   0                   1                   0                 600   )
(                   0                   0                   1                   0   )
    5 :
Elementary location
(                   1                   0                   0                   0   )
(                   0      1.11022e-016                  -1                   0   )
(                   0                   1      1.11022e-016                   0   )
    6 :
Complex : L5^6
(                   1                   0                   0                   0   )
```

```
(                    0                    -1     -6.66134e-016                    0    )
(                    0      6.66134e-016                    -1                    0    )
Press any key to continue . . .
```

输出到文件中的内容为：

```
Locations 6
1
                    1                    0                    0                    0
                    0                    1                    0                    0
                    0                    0                    1                    0
2    1 2 0
1
                    1                    0                    0                  100
                    0                    1                    0                  200
                    0                    0                    1                    0
2    3 3 0
1
                    1                    0                    0                    0
                    0 1.11022302462516e-016                   -1                    0
                    0                    1 1.11022302462516e-016                    0
2    5 6 0
```

从输出结果可以看出，输出到文件中的内容与 BRep 文件中的内容一致。Location 有两种类型，当类型为 1 时，即是一个初等变换矩阵（Elementary location）；当类型为 2 时，是一个复合变换（Complex），即在初等变换矩阵的基础上做的一些变换操作。

### 3.2 读取位置数据 Input Location data

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);
  }
}
```

从读取 Location 部分的代码可以看出，分两情况来处理。一种是初等变换矩阵，类型值为 1，直接读取矩阵数据；一种是复合变换，类型值为 2，它是在初等变换矩阵的基础上通过 Power 来实现的复合变换。BRep 中记录复合变换的数据为初等变换矩阵的编号及其幂次。通过编号 Map 得出其对应的初等变换矩阵。结合读取 Location 的代码，对 BRep 中 Location 部分的数据有了清晰认识。

# 四、结论

通过对 OpenCascade 中 BRep 文件中的 Locations 部分的数据的输出与读取，理解其实现。即对 Location 分为两种类型：

1. 初等变换矩阵：存储数据为 3X4 变换矩阵；
2. 复合变换：存储数据为初等变换矩阵的编号及其幂次。