Implementation Model Editor of AVEVA in OpenSceneGraph

eryar@163.com

摘要 Abstract：本文主要对工厂和海工设计软件 AVEVA 的交互方式进行详细介绍，对 OpenSceneGraph 中的人机交互工具拖拽器进行说明，并在其中实现了模型直接交互操作。对交互建模感兴趣的读者可结合其源代码，对其实现细节进行分析。

关键字 Key Words：AVEVA, Model Editor, OpenSceneGraph, Dragger

一、引言 Introduction

在当代的三维辅助设计软件中，交互建模设计已经成为主流。友好、高效的对三维模型直接进行编辑或修改，不仅可以提高用户的工作效率，还会给用户留下美好印象，即软件良好的用户体验。交互建模的常见方法有：拖曳、约束、栅格捕捉、橡皮筋方法、引力场等，拖曳就是直接对选择的模型在三维空间中拖动来改变位置和方向；约束方法就是在拖曳的时候添加约束条件，如只能沿某个方向进行拖曳，软件中的应用有 AutoCAD 中的极轴捕捉功能；栅格捕捉也是一种带约束的拖曳，即拖曳的过程中只能沿正交网格中直线的交点拖动；橡皮筋方法主要用在绘制二维图形；引力场方法就像 AutoCAD 中的磁吸功能。如何设计高效、友好、方便的用户接口是当前各开发系统的厂家和专家所共同关心的问题，它的设计好坏可能直接影响用户是否接受其产品。

本文主要对工厂和海工设计软件 AVEVA 的交互方式进行详细介绍，对 OpenSceneGraph 中的人机交互工具拖拽器进行说明，并在其中实现了模型直接交互操作。通过程序实践，感觉使用 OpenSceneGraph 来进行编码还是很舒服的，因为其代码规范，设计很好。对交互建模感兴趣的读者可结合其源代码，对其实现细节进行分析。

二、模型编辑器 Model Editor of AVEVA

AVEVA（原 CADCENTRE）是国际著名的工厂工程信息技术企业，成立于 1967 年，总部设在英国剑桥；AVEVA 所提供的工厂工程一体化解决方案涵盖了陆地和海洋石油天然气、电力、石化、化工、核电、造船、环保、造纸、制药、冶金、矿山等多个行业，同时提供专业工厂工程技术咨询、技术服务和本地化可持续发展的应用开发。 AVEVA 是目前全球发展最快的工厂工程信息技术企业之一，1996 年在英国伦敦上市，2007 财年年产值超过 25 亿美元。AVEVA 在全球拥有超过 1600 名用户，每天有超过 26,000 名工程人员在使用 AVEVA 的解决方案。AVEVA 在世界 30 多个国家和地区设有超过 50 个常驻办事机构，在英国剑桥总部及其他研发中心拥有超过 300 名研究和开发人员，为世界上最大的工厂工程信息技术研究和开发团队。AVEVA 的快速发展与其方便易用，良好的交互建模方式分不开，本文主要对其交互建模部分进行介绍。

AVEVA 交互建模主要是使用模型编辑器 Model Editor，使用 Model Editor 可以只用鼠标就可以进行建模设计了。编辑选择的模型如下图所示：

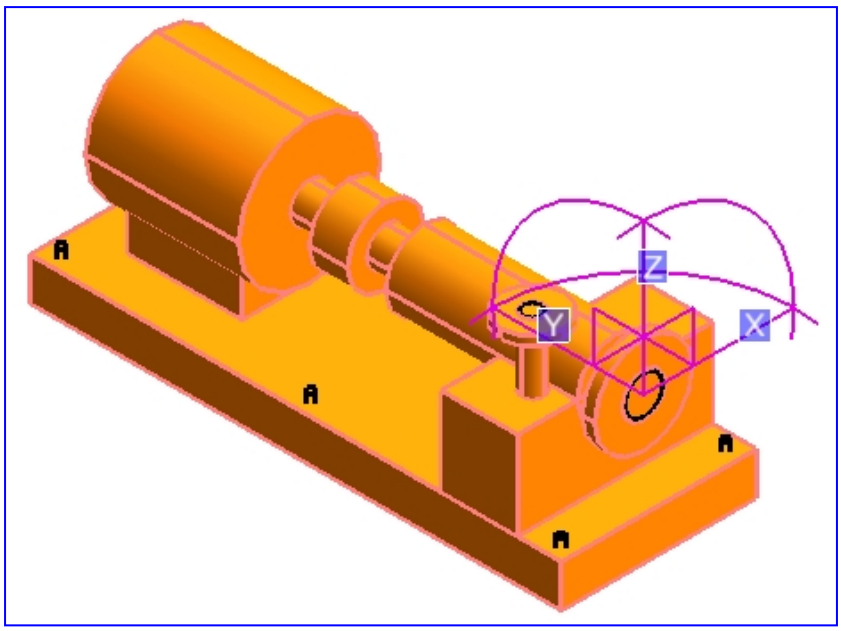

Figure 2.1 Model Editor on a Equipment

使用的 Handle 可以对模型多种方式的编辑，如轴向移动、平面移动、旋转等，如下图所示：

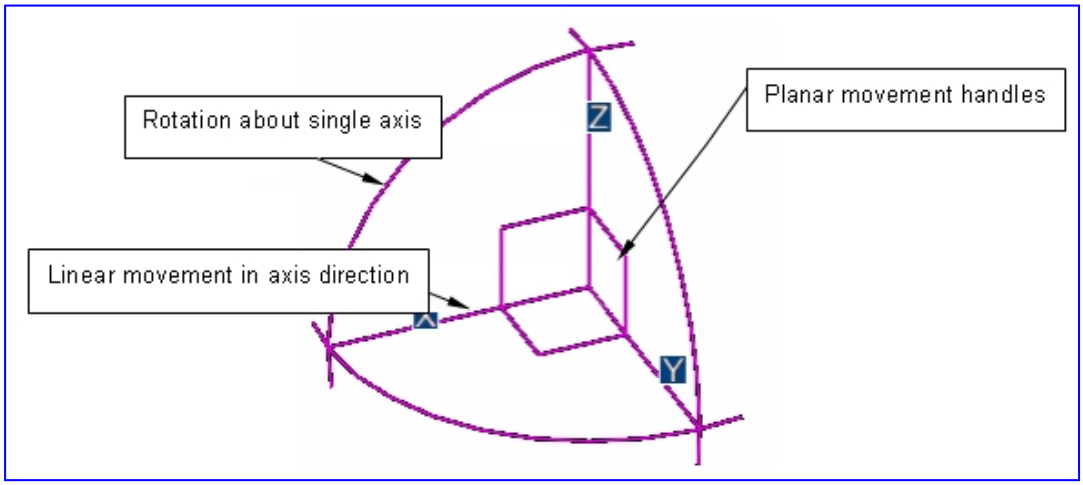

Figure 2.2 Locator Handle of Model Editor

2.1 移动 Movement

对选择的模型进行轴向移动或平面移动时，可以使用 Model Editor 的 Linear and Planar handles。沿轴向或锁定在某个平面上拖动模型，即可以移动模型。移动是按一定的长度递增的，可由移动增量（the Movement Increment）来设置，这样来确保拖动模型相对初始位置的精度。选中 handle 上某个轴，就可以沿这个轴的方向来移动模型，如下图所示：

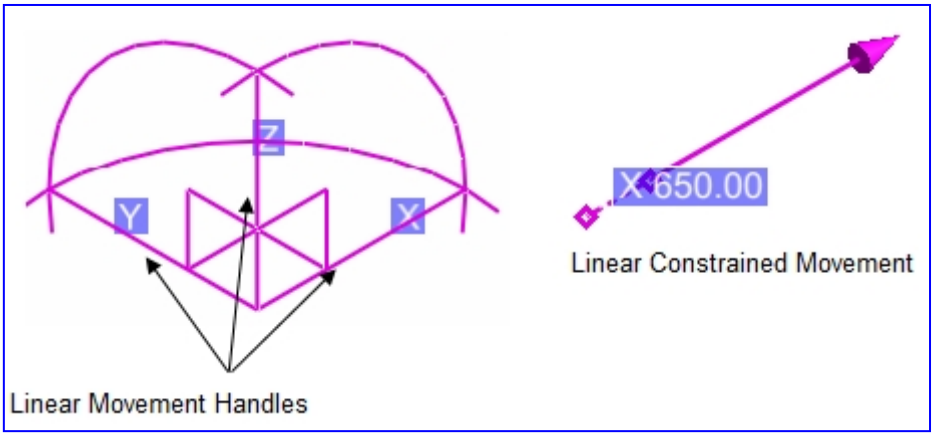

Figure 2.3 Linear Movement

选中并拖动 Model Editor 的平面移动 handle，就可以在锁定的平面上移动模型，如下图所示：

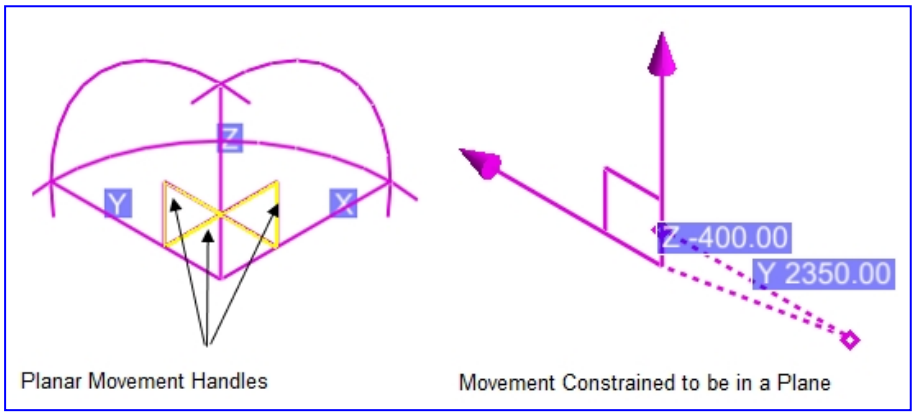

Figure 2.4 Planar Movement

2.2 旋转 Rotation

旋转是通过 Model Editor 的 Rotation handle 来完成的。旋转是按一定的角度来递增的，可由旋转增量（the Rotation Increment）来设置。这样就确保了旋转相对于初始位置的精度。选中并拖动 Rotation handle，就可以对模型进行旋转了，如下图所示：

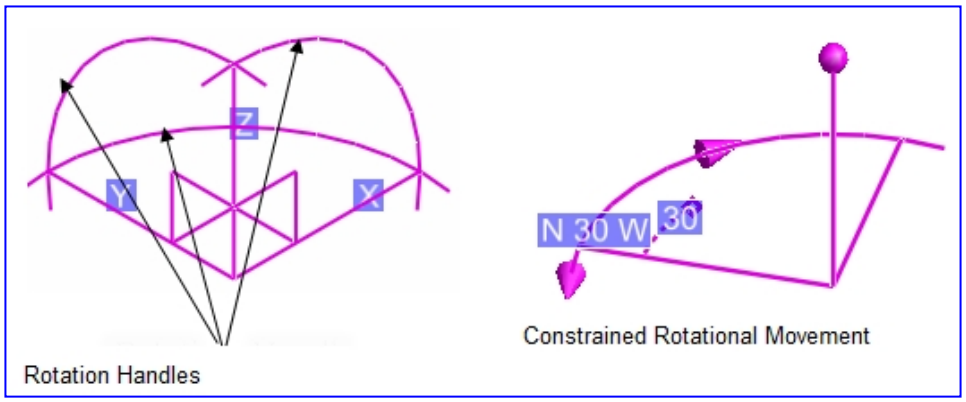

Figure 2.5 Rotation

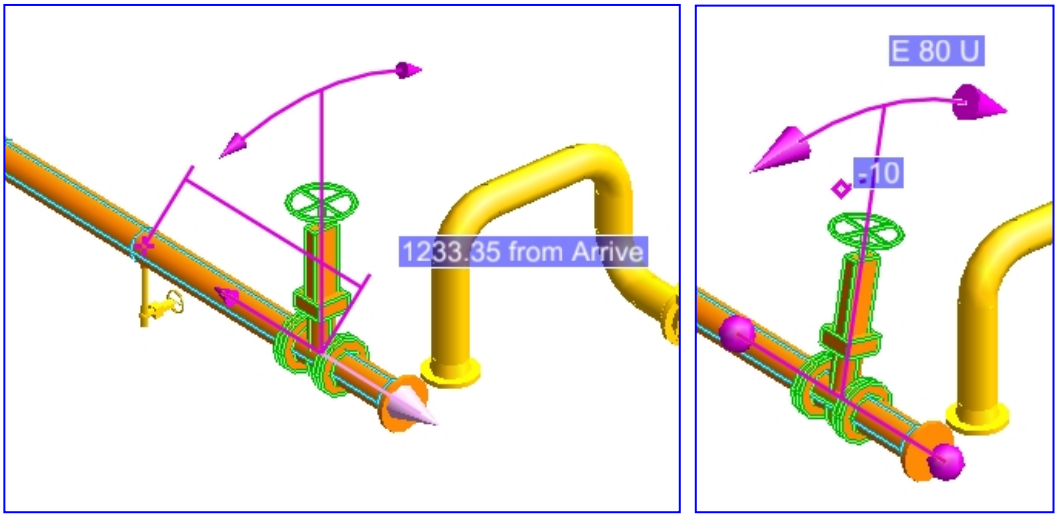

Figure 2.6 Use Model Editor to Modify a Valve

交互建模时使用 Model Editor 如上图 2.6 所示，由图可知，在 AVEVA 中对模型的移动和旋转非常方便。

2.3 对齐 Alignment

通过对齐功能，可以方便地将模型对齐到点、边或面，如下图所示：

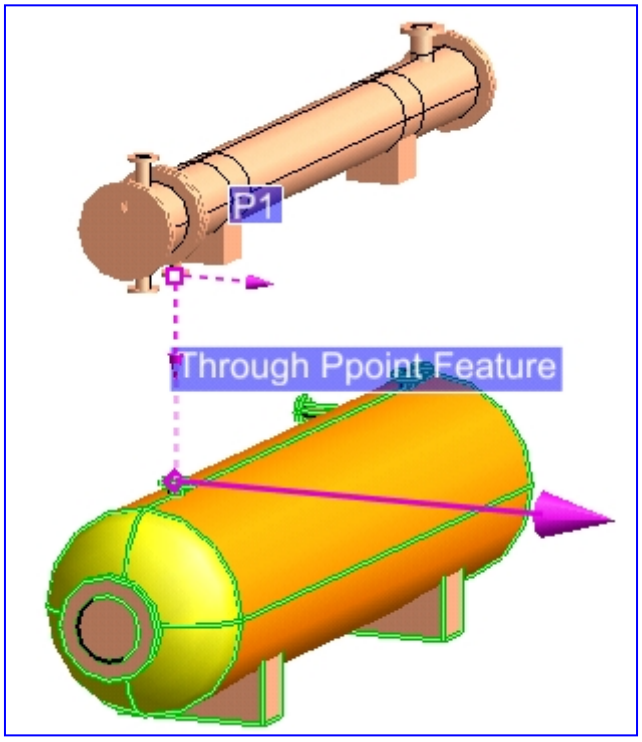

Figure 2.7 Alignment features

通过上文对 AVEVA 中的 Model Editor 的介绍可知，在 AVEVA 中三维交互建模很方便，且精度高。纵观国内目前类似产品，有些还停留在二维建模方式，有些借助于其他平台的交互方法，但是在易用性上感觉都稍有不足，或没有自主的知识产权。

三、拖拽器 Dragger of OpenSceneGraph

1997 年，一个名叫 Don Burns 的软件工程师受雇于当时的 Silicon Graphics(SGI)公司，负责针对滑翔机飞行的虚拟仿真工作进行研究。他使用当时的 OpenGL Performer 系统，设计了一套广受好评的滑翔仿真软件，并开始尝试在 Linux 中使用 Mesa3D 和 3dfx Voodoo 显卡设备继续完善自己的仿真软件。

1998 年，Don 在一个滑翔爱好者的邮件组中遇到了 Robert Osfield，也就是目前 OpenSceneGraph 项目的主要负责人。当时 Robert 在苏格兰的油气公司工作，但对计算机图形学和可视化技术有着浓厚的兴趣。志趣相投的两个人走到了一起，开始合作对 Don 的仿真软件进行改善。Robert 建议将 SG 作为独立的开源场景图形项目继续开发，并由自己担任项目主导，项目的名称改为 OpenSceneGraph，简称 OSG。

如今，相当一部分高性能的软件已经使用了 OSG 来完成复杂场景的渲染工作。大部分基于 OSG 的软件开发更适用于可视化设计和工业仿真，包括地理信息系统（GIS）、计算机辅助设计（CAD）、建模和数字媒体创作（DCC）及数据库开发、虚拟现实、动画、游戏和娱乐业等。

OpenSceneGraph 引擎由一系列图形学相关的功能模块组成，主要为图形图像应用程序的开发提供场景管理和图形渲染优化的功能。它使用可移植的 ANSI C++编写，并使用已成为工业标准的 OpenGL 底层渲染 API。OSG 具备跨平台的特性，可以运行在大多数类型的操作系统上，并使用抽象层的概念，使 OSG 的函数接口可以独立于用户的本地操作系统使用。OSG 遵循开源协议发布，其用户许可方式是一种修改过的 GNU 宽通用公共许可证（GNU Lesser General Public License, LGPL），称为 OSGPL。OSG 主要具备以下优势：快速开发，高品质，高性能，高质量代码，可扩展性，可移植性，低费用，没有知识产权问题，但是 OSG 目前也存在诸多不足，如参考文档较少、代码风格不统一、部分功能的实现过于臃肿，无法应用于实践等，这些都有待更多的开发者和贡献者去发现和完善。

三维用户交互是一种与三维环境本身特性相匹配的交互动作，可使用户在虚拟场景中获得身临其境的直观感受。三维世界的交互技术相当于一种“控制－显示”的映射，用户设备（例如鼠标、键盘、操纵杆等）向系统输入控制信息，然后系统向用户输出执行结果。三维交互涉及的任务繁多，包括三维场景对象的选择和操控、三维世界中的导航漫游、改变三维场景的状态，乃至时下流行的三维交互建模等。作为一款全面的实时渲染引擎，OSG 实现了三维场景的漫游及场景中三维对象的操控这两种主要的三维场景交互方式，更多的交互动作则需要我们自行研究和实现。

作为重要的三维空间的人机交互手段之一，场景漫游的特点是通过不断改变观察者（相机）的位置、姿态，使其相对世界的观察方位和角度有所变化，但是世界本身却不会发生任何改变。无论草木、建筑，还是街道上的车水马龙，构成它们的每一个顶点都没有发生任何偏移，如果观察者有朝一日回到原地的话，他眼中的一切都不会发生改变。

而对于三维物体的操控则是另一种概念，它没有改变观察者的视角和视点，而是根据用户传递的交互事件，对选中的对象进行平移、缩放和旋转操作，就像是玩弄橡皮泥一样。被修整过的对象将改变原有的形态，换句话说，只要不恢复到操控前的状态，那么无论从什么地方进行观察，这个对象都将维持它最终的模样。当然，即使操控物体的定义如此，直接修改物体的顶点坐标未免还是有些费力不讨好，最好的方式是为要操控的物体设置一个矩阵变换的父节点（MatrixTransform），通过改变这个父节点的变换矩阵的值，进行改变作为操控对象的子节点的表现形式—这就是 osgManipulator 库中拖拽器（Dragger）的实现方式。OSG 内置了几种拖拽器，其操作方式和效果说明如下：

- TabPlaneDragger 平面拖拽器：其边、顶点上都有拖拽点，可以进行某个 2D 平面上的缩放；

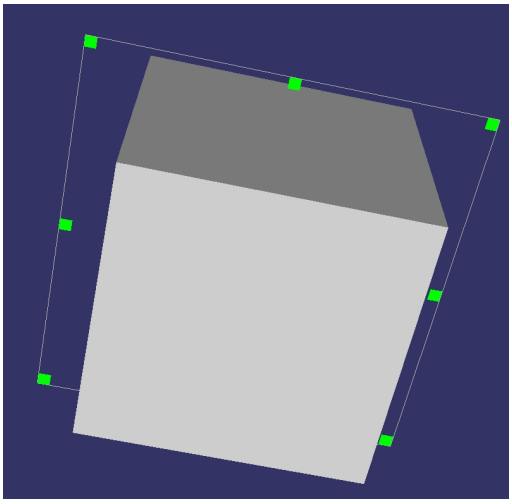

Figure 3.1 TabPlaneDragger in OpenSceneGraph

- TabPlaneTrackballDragger 平面轨迹球拖拽器：顾名思义除了平面拖拽器的功能外，还多了个轨迹球拖拽功能；

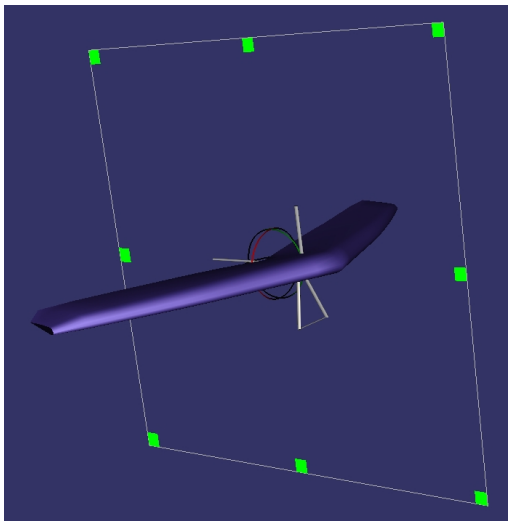

Figure 3.2 TabPlaneTrackballDragger in OpenSceneGraph

- TrackballDragger 轨迹球拖拽器：即旋转操纵器，没有缩放功能；

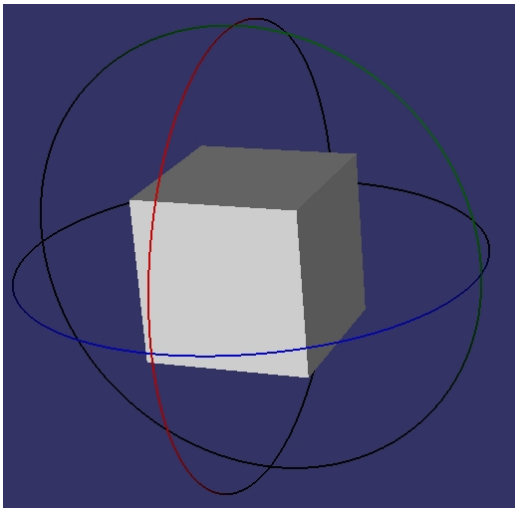

Figure 3.3 TrackballDragger in OpenSceneGraph

- Translate1DDragger 一维平移拖拽器：沿一个直线进行拖拽；

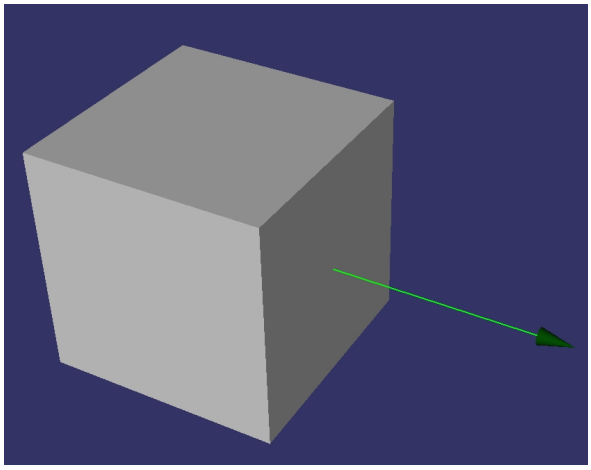

Figure 3.4 Translate1DDragger in OpenSceneGraph

- Translate2DDragger 二维平移拖拽器：在某个平面上对模型进行拖拽；

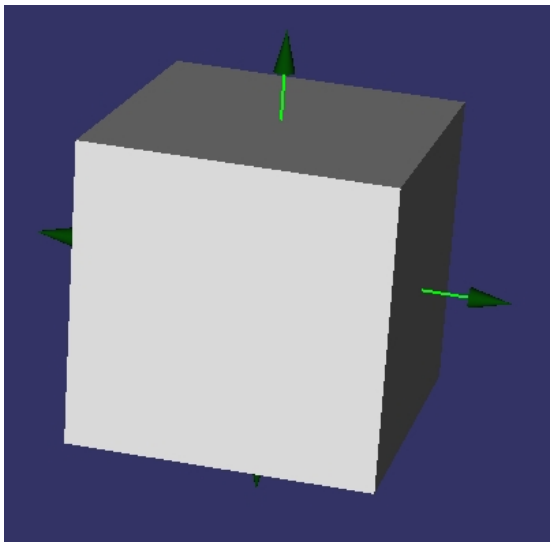

Figure 3.5 Translate2DDragger in OpenSceneGraph

- TranslateAxisDragger 三维平移拖拽器：可在三个方向上对模型进行拖拽；

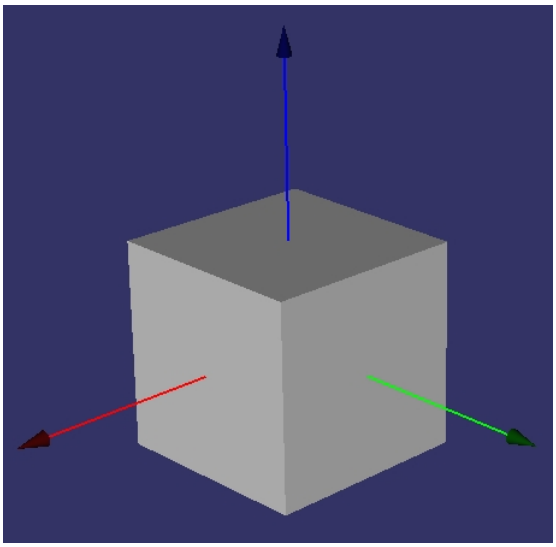

Figure 3.6 TranslateAxisDragger in OpenSceneGraph

- TabBoxDragger 盒式拖拽器：由六个平面拖拽器构成，可在各个面上进行缩放、平移；

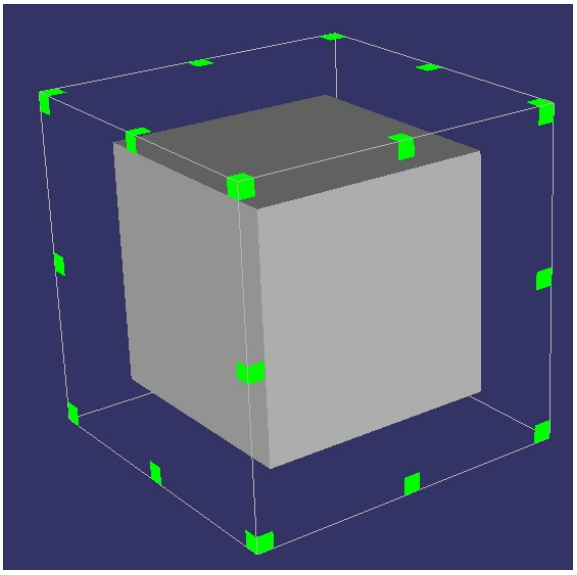

Figure 3.7 TabBoxDragger in OpenSceneGraph

还有其他的拖拽器可以参考其源代码：

- ScaleAxisDragger：三维缩放拖拽器；
- Scale2DDragger：二维缩放拖拽器；
- Scale1DDragger 一维缩放拖拽器；
- RotateSphereDragger：旋转球拖拽器；
- RotateCylinderDragger 旋转圆柱拖拽器；

由于使用了组合模式（CompositeDragger），可将上面的拖拽器组合成更复杂的拖拽器。如三维平称拖拽器（TranslateAxisDragger）就是包含了三个一维拖拽器（Translate1DDragger）的组合拖拽器。因为程序开源，所以可根据实际需要，将拖拽器进行自定义。即可以定义出和 AVEVA 的 Model Editor 完全一样的操纵器。

四、示例程序 Example Code

要对场景中的模型进行拖拽，首先需要将其选中，然后在选中的模型上打开拖拽器，对模型的位置和方向进行编辑。对象的拾取主要依靠场景图形的交运算来实现。拖拽器中用到的一个类 PointerInfo 表示一个信息的集合。例如，使用鼠标选中待操作对象上的某个点，以及当前相机的观察矩阵和投影矩阵等，都需要及时反映到这个输入参数中，以便拖拽器根据实际情况进行判断并生成命令。下面给出一个使用拖拽器 Dragger 来操作场景中模型的具体实例，程序代码如下所示：

1.首先定义了一个带拖拽器的节点 ModelShape：

```
#pragma once

#include <osg/Group>

#include <osgManipulator/Selection>
#include <osgManipulator/CommandManager>
#include <osgManipulator/TrackballDragger>
#include <osgManipulator/TranslateAxisDragger>


class ModelShape : public osg::Group
{
public:
	ModelShape(osg::Node* shape);
	~ModelShape(void);

	void EnableDragger(void);
	void DisableDragger(void);

private:
	osg::ref_ptr<osg::Node> mShape;
	osg::ref_ptr<osgManipulator::Dragger> mDragger;
	osg::ref_ptr<osgManipulator::Selection> mSelection;
};
```

类实现代码如下所示：

```
#include "ModelShape.h"

ModelShape::ModelShape(osg::Node* shape)
: mShape(shape)
, mDragger(new osgManipulator::TranslateAxisDragger())
, mSelection(new osgManipulator::Selection())
{
	float scale = shape->getBound().radius() * 1.6;
	mDragger->setMatrix(osg::Matrix::scale(scale, scale, scale) *
		osg::Matrix::translate(shape->getBound().center()));

	mDragger->setupDefaultGeometry();

	mSelection->addChild(shape);
	addChild(mSelection);
}


ModelShape::~ModelShape(void)
{
}
```

```
void ModelShape::EnableDragger()
{
	addChild(mDragger);

	mDragger->addTransformUpdating(mSelection);
	mDragger->setHandleEvents(true);
}


void ModelShape::DisableDragger()
{
	removeChild(mDragger);

	mDragger->removeTransformUpdating(mSelection);
	mDragger->setHandleEvents(false);
}
```

2.在处理选择事件时，打开拖拽器，实现类是 PickHandler：

```
#pragma once

#include <osgGA/GUIEventHandler>


class PickHandler : public osgGA::GUIEventHandler
{
public:
	PickHandler(void);
	~PickHandler(void);

	virtual bool handle(const osgGA::GUIEventAdapter& ea, osgGA::GUIActionAdapter& aa);

protected:
	void pick(const osgGA::GUIEventAdapter& ea, osgGA::GUIActionAdapter& aa);

private:
	float mX;
	float mY;
	bool mEnableDragger;
};
```

类实现代码如下所示：

```
#include "PickHandler.h"
#include "ModelShape.h"

#include <osgViewer/Viewer>


PickHandler::PickHandler(void)
: mX(0.0f)
, mY(0.0f)
, mEnableDragger(true)
{

}


PickHandler::~PickHandler(void)
{
```

```
}


bool PickHandler::handle(const osgGA::GUIEventAdapter &ea, osgGA::GUIActionAdapter &aa)
{
	osgViewer::View* view = dynamic_cast<osgViewer::View*> (&aa);
	if (NULL == view)
	{
		return false;
	}

	switch (ea.getEventType())
	{
	case osgGA::GUIEventAdapter::PUSH:
		{
			mX = ea.getX();
			mY = ea.getY();
		}
		break;

	case osgGA::GUIEventAdapter::RELEASE:
		{
			if (ea.getX() == mX && ea.getY() == mY)
			{
				pick(ea, aa);
			}
		}
		break;

	case osgGA::GUIEventAdapter::KEYDOWN:
		{
			if (ea.getKey() == 'd')
			{
				mEnableDragger = !mEnableDragger;
			}
		}
		break;

	default:
		break;
	}

	return false;
}


void PickHandler::pick(const osgGA::GUIEventAdapter &ea, osgGA::GUIActionAdapter &aa)
{
	osgViewer::View* view = dynamic_cast<osgViewer::View*> (&aa);

	osgUtil::LineSegmentIntersector::Intersections hits;
	if (view->computeIntersections(ea.getX(), ea.getY(), hits))
	{
		osgUtil::LineSegmentIntersector::Intersection intersection = *hits.begin();
		osg::NodePath& nodePath = intersection.nodePath;
		int nNodeSize = static_cast<int> (nodePath.size());
```

```
            if (nNodeSize > 0)
            {
                osg::Node* node = nodePath[nNodeSize - 1];
                osg::Node* grandParent = node->getParent(0)->getParent(0);

                // This method maybe not right?
                ModelShape* shape = dynamic_cast<ModelShape*> (grandParent);
                if (shape)
                {
                    mEnableDragger ? shape->EnableDragger() : shape->DisableDragger();
                }

            }
        }

}
```

3.在主函数中建立场景：

```
/*
*       Copyright (c) 2013 eryar All Rights Reserved.
*
*                File : Main.cpp
*              Author : eryar@163.com
*                Date : 2013-12-28 17:00
*             Version : 1.0v
*
*       Description : Use dragger to manipulate shape objects.
*                           press key 'd' for enable or disable the dragger.
*
*         Key Words : OpenSceneGraph, Dragger
*
*/

#include <osgDB/ReadFile>
#include <osgGA/StateSetManipulator>

#include <osgViewer/Viewer>
#include <osgViewer/ViewerEventHandlers>


#include "ModelShape.h"
#include "PickHandler.h"


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


int main(void)
{
    osgViewer::Viewer viewer;

    osg::ref_ptr<osg::Group> root = new osg::Group();

    // build the scene with gliders.
```

```
    for (int i = 1; i < 10; ++i)
    {
        for (int j = 1; j < 10; ++j)
        {
            osg::ref_ptr<osg::MatrixTransform> box = new osg::MatrixTransform();
            osg::ref_ptr<osg::MatrixTransform> glider = new osg::MatrixTransform();

            box->setMatrix(osg::Matrix::translate(i * 6.0, j * 6.0, 0.0));
            glider->setMatrix(osg::Matrix::translate(i * 2.5, j * 2.5, 6.0));

            box->addChild(new ModelShape(osgDB::readNodeFile("box.stl")));
            glider->addChild(new ModelShape(osgDB::readNodeFile("glider.osg")));

            root->addChild(box);
            root->addChild(glider);
        }
    }

    viewer.setSceneData(root.get());

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

    // add pick event handler to add dragger on the shape.
    viewer.addEventHandler(new PickHandler());

    return viewer.run();
}
```

程序使用方法为选择要拖拽的模型，选中后为模型打开拖拽器，使用拖拽器对模型进行拖拽就可以修改模型的位置了。在键盘上按下‘d’可以打开/关闭拖拽器，默认为打开。当设置为关闭时，再选中带有拖拽器的模型后，将会关闭拖拽器。程序运行效果如下图所示：

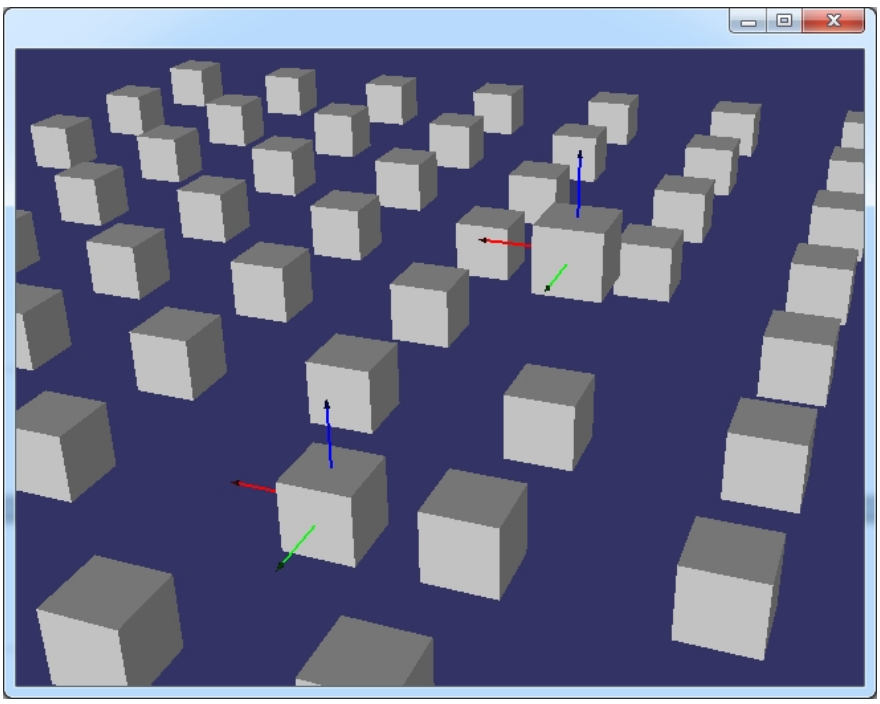

Figure 3.8　Use TranslateAxisDragger to dragger box

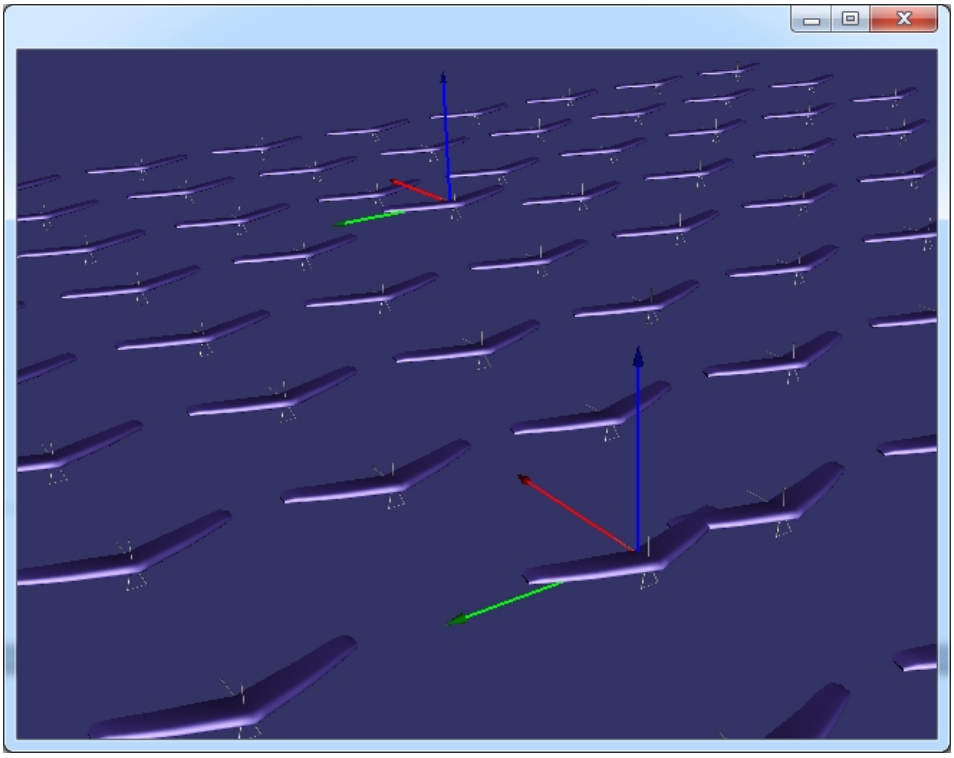

Figure 3.9 Use TranslateAxisDragger to dragger glider

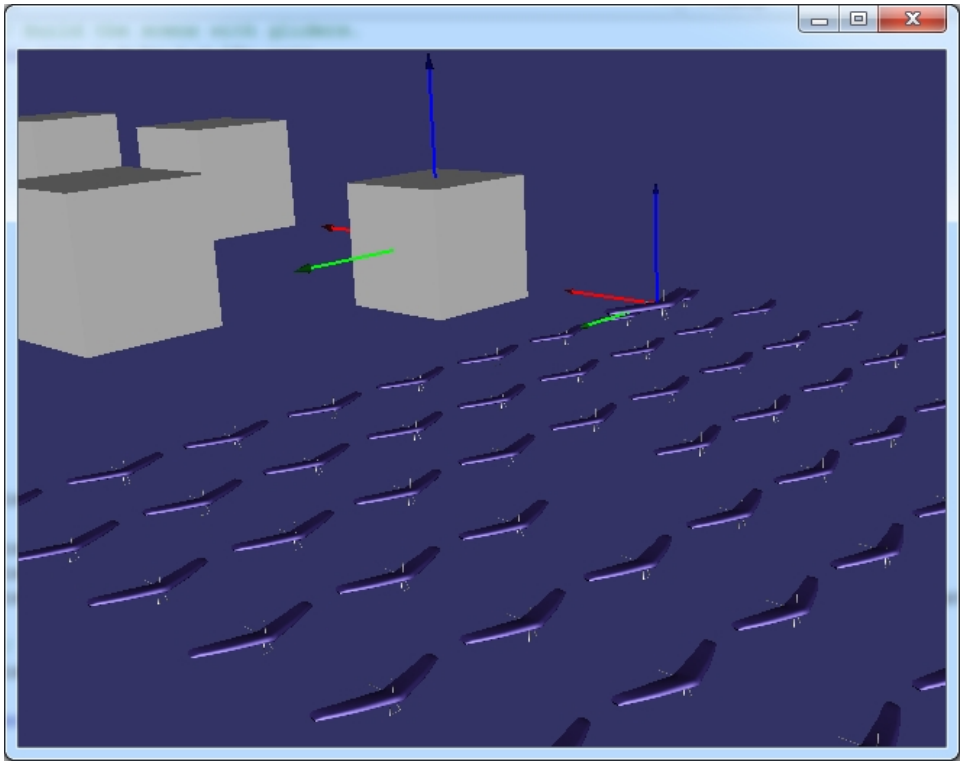

Figure 3.10 Use TranslateAxisDragger in OpenSceneGraph

五、结论 Conclusion

通过程序实践表明，OpenSceneGraph 中的人机交互的方式还是很方便的，并提供了几种拖拽器来操纵模型。并且拖拽器采用了组合模式，便于扩展，即根据用户实际需要组合出更复杂或更具个性的拖拽器。

由此可见，使用 OpenSceneGraph 来对模型进行显示与操作很方便，且是开源程序，方便程序调试，还不存在知识产权的问题。因为 OpenSceneGraph 主要是用于虚拟仿真，还提供了很多仿真效果，如烟雾、火焰、粒子效果（雨、雪、爆炸）、动画等，如果在建模设计的过程中适量添加部分效果，是不是很 cool？

六、参考资料 References

1. AVEVA，Graphical Model Manipulation Guide
2. Donald Hearn，M. Pauline Baker，Computer Graphics with OpenGL，电子工业出版社
3. 何援军，计算机图形学，机械工业出版社
4. 王锐，钱学雷，OpenSceneGraph 三维渲染引擎设计与实践，清华大学出版社
5. 肖鹏，刘更代，徐明亮，OpenSceneGraph 三维渲染引擎编程指南，清华大学出版社