# 基于 OSG 的 GLSL 着色器编辑环境①

陈继选，王毅刚

(杭州电子科技大学 图形图形所，杭州 310018)

**摘 要**：随着可编程图形硬件时代的到来，着色器替代了传统的固定渲染管线以实现更加逼真的图像效果。高级着色语言的兴起，给着色器编写带来了方便，然而编写着色器依然带来一些独有的挑战。通过实现一款基于 OSG（OpenSceneGraph）的 GLSL 着色器编辑环境，以大大简化着色器编写，帮助着色器实现者开发一个参数化的着色器，并借助 OSG 的插件机制大大扩展了应用范围。

**关键词**：OSG；GLSL；着色器；编辑环境

## GLSL Shader Editing Environment Based on OSG

CHEN Ji-Xuan, WANG Yi-Gang

(Institute of Graphics and Image, Hangzhou Dianzi University, Hangzhou 310018,China)

**Abstract:** With the era of programmable graphics hardware, shader replaces the traditional fixed rendering pipeline to achieve more realistic effects. The rise of high-level shading language gives the convenience to writing shader, but still brings some unique challenges. This paper implements a GLSL shader editing environment based on OSG, in order to greatly simplify the shader coding and help shader developers to develop a parameterized shader. We also extends the range of applications of the results by using OSG's plug-in mechanism.

**Keywords:** OSG; GLSL; shader; editing environment

随着可编程图形硬件的兴起，各种与之相关的着色语言也相继出现。与最初的汇编语言式的编程相比，高级着色器编程语言使着色器编程变得方便而灵活并可独立于底层硬件。随之，我们也了解到开发着色器不仅仅是开发着色器本身的代码，着色器是完全可以定制的，着色器源代码、纹理、几何对象以及一致变量都是优秀着色器的重要组成部分。为了帮助开发人员，有必要提供一个开发环境记录这些着色器的所有关键元素，并能轻易地对这些元素进行修改和维护。

如今显卡公司陆续推出了自己的着色器开发环境，如 RenderMonkey(AMD&ATI)，FX Composer (NVIDIA)；但是笔者没有发现国内在这方面的相关工作或相关产品问世。本文主要工作就是借助 OSG 对 OpenGL 良好的功能封装以方便、迅速地构建一款界面简洁、友好，所见即所得的 GLSL 着色器编辑环境；帮助开发者简化着色器编写，迅速构建一个参数化的着色器，并考虑实现结果的可用性、易用性和复用性。

## 1 GLSL与OSG

GLSL(OpenGL Shading Language，OpenGL 着色语言)是在可编程图形硬件逐渐兴起的趋势下应运而生的高级着色语言，它采用类似 C 语言的词法和语法结构，并且具有类似于 RenderMan 和其它着色语言的特性。该语言有一组丰富的类型，包括矢量和矩阵类型，它们可以使代码变得对典型的 3D 图形操作更为简洁。GLSL 包括了对循环、子例程调用和条件表达式的支持。大量内置的函数组提供了实现着色算法所需要的许多功能[1]。

OSG (OpenSceneGraph)是一个高性能的开源三维图形引擎，它以 OpenGL 为底层平台，使用 C++编写而成，可运行于 Windows、UNIX/Linux、Mac OS X、IRIX、Solaris 和 FreeBSD 等操作系统；发展至今，其

① 收稿时间:2010-07-17;收到修改稿时间:2010-08-27

功能特性涵盖了大规模场景的分页支持，多线程、多显示渲染，例子系统与阴影，各种文件格式的支持，以及对于 Java、Perl、Python 等语言的封装等[2]。

OSG 提供了完整的 GLSL 的功能封装，其主要实现者包括 osg 命名空间中的几个类(如表 1 所示)。

表 1　OSG 中 GLSL 的封装类

| osg::Program | osg::Shader | osg::Uniform |
|---|---|---|
| Program 类是 StateAttribute 的一个派生类，可以被设置到一个节点（Node）或者一个可绘制物体（Drawable）上，从而将着色器绑定到指定的场景对象。这相当于对 OpenGL 函数 glProgram（）的一个实现接口。 | Shader 类封装了顶点、几何和片元着色器的代码加载和编译功能，相当于 OpenGL 函数 glShaderSource（）和 glCompileShader（）的实现接口。 | Uniform 类是着色器一致变量（uniform）的接口类。对于 OpenGL 着色器而言，一致变量是用户应用程序与着色器的主要交互接口。 |

## 2　软件设计与实现

本文实现的 GLSL 着色器编辑环境主要由以下几个模块组成：着色器代码编辑器、变量编辑器、预览窗口、导入模块和导出模块(如图 1 所示)。


图 1　系统模块图

### 2.1 着色器代码编辑器

图形处理器中原有的顶点着色器和片元着色器只允许程序操作内存中已有的数据，这种开发模型不允许在图形处理器上生成新数据。最新的着色器模式 Shader Model4.0 引入的几何着色器允许程序在图形处理器中创建新数据。几何着色器被放在顶点着色器和光栅化阶段中间，它可以实现顶点的批量处理，几何着色器可以借助相邻的信息，让 GPU 不需 CPU 的帮助直接提供更为精细的模型细节。然而现有开发工具大都只支持顶点着色器和片元着色器的编写与实现，都未能加入对几何着色器的支持。

本文实现了对几何着色器的支持，并为顶点着色器、几何着色器和片元着色器分别实现了代码编辑窗口。而对于代码编辑器来说，无论是 Visual Studio 还是 Eclipse 等众多 IDE，映入我们眼帘最直观的就是各种语法上色。本文实现的着色器代码编辑器(如图 4①所示)同样支持 GLSL 的语法上色色。可以对用户输入的字符进行解析以识别出 GLSL 中的关键字、编译指令、内置函数以及内置的属性和状态，并为它们设置不同的字体颜色以区分它们，使代码更加的清晰明了，也为代码编写者带来了比较直观的视觉区分。

### 2.2 变量编辑器

在编写着色器的过程中我们需要对着色器有更多的控制与交互。对于 GLSL 而言，一致变量(uniform)为用户应用程序与着色器提供了主要交互接口。为了丰富物体的表达效果，纹理也是着色器编写过程中不可或缺的。变量编辑器(如图 4②所示)就是为一致变量的和纹理提供了一个 GUI 编辑与修改窗口，并能实时的反映到预览窗口中。

变量编辑器的设计是通过使用 MFC9.0 提供的最新的 CMFCPropertyGridCtrl 控件来实现为着色器代码中的一致变量和纹理提供一个可编辑的环境，从而控制着色器代码的实现效果。通过变量编辑器模块，用户可以方便的添加、修改各种数据类型(标量、矢量、矩阵、取样器)的一致变量和纹理贴图。而对于纹理贴图而言，我们往往需要设定纹理的滤波方式(Filter)、边界处的截取方式(Clamp)、明细层次(Mipmaps)以及纹理边界的颜色(BorderColor)的生成方式等以适应不同的纹理操作需求。本文对添加的纹理都相应的生成这些属性的编辑功能并能实时的反映到预览窗口中。

### 2.3 预览窗口

为了实时的显示可编辑对象的着色效果，一个预览窗口是必须的。本文借助 OSG 对视景器(viewer)、相机(camer)以及图形设备上下文(GraphicsContext)的封装实现一款自定义的预览窗口(如图 4③所示)。

通过设定继承自 GraphicsWindow 类的具体实现派生

类 GraphicasWindowsWin32 来创建 Windows 系统下的渲染窗口，同时它也继承了动作适配器 GUIActionAdapter 的成员对象，从而具备了接受用户交互请求并执行响应动作的能力，进而在 OSG 的事件处理器中处理这些原本与 Windows 平台密切相关的键盘和鼠标事件。

表 2 预览窗口交互操作

| 功能 | 键盘操作 | 描述 |
|---|---|---|
| 漫游器操作 | “1” | 选择轨迹球（Trackball）漫游器。 |
| | “2” | 选择飞行（Flight）漫游器。 |
| | “3” | 选择驾驶（Drive）漫游器。 |
| | “4” | 选择地形（Terrain）漫游器。 |
| | “5” | 选择 UFO 漫游器。 |
| 场景、渲染设置 | “l” | 允许/禁止光照。 |
| | “s” | 允许/禁止显示渲染统计信息。 |
| | “t” | 允许/禁止使用纹理。 |
| | “b” | 允许/禁止绘制背面。 |
| | “w” | 选择渲染模式（点、线框、面）。 |

本文还为预览窗口提供了丰富的辅助组件。这些辅助组件使预览窗口具备了丰富的交互特性(见表 2 所述)。对于开发者而言，这些特性在开发过程中十分有助于调试工作的进行。其中模型渲染信息的显示，包括当前帧速率、场景中的节点和顶点数目等。反复按下“s”键可以切换显示从简单到复杂的实时模型渲染信息(如图 2 所示)。

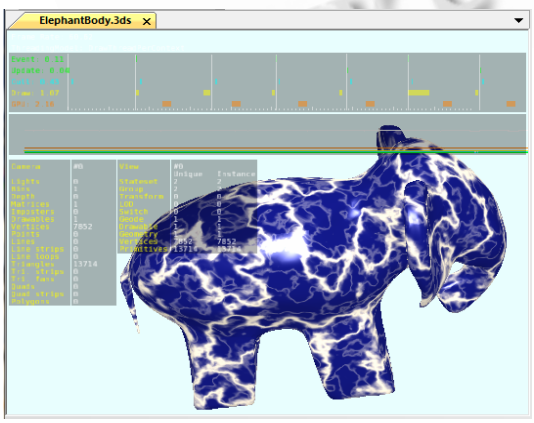



图 2 预览窗口模型渲染信息显示

### 2.4 导入模块与导出模块

随着各行业软件的飞速发展和爆炸式的增加，各种系统支持的文件格式也越来越多，不下数十种甚至数百种之多，光是三维开发者们耳熟能详的格式就包括 MAX、WRL、DWG、KML、FLT 等；而对于着色器的开发不可缺少的纹理图形格式，常见的又有 BMP、JPG、PNG、GIF、TGA、DDS 等。要对这些格式的读写操作进行支持，并不是一件容易的事，然而通过 OSG 提供的插件机制(如图 3[3]所示)我们很容易的在程序中对这些现有的 3D 模型、2D 图片格式进行读写操作。

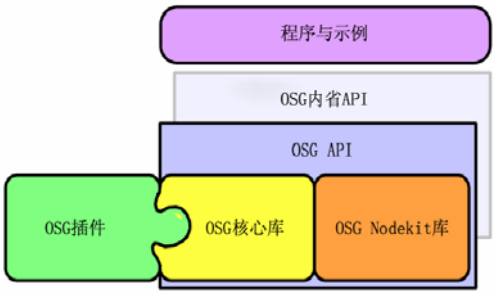



图 3 OSG 体系结构

本文实现了支持大约 70 种不同的文件格式读写操作，并且这一数量随着 OSG 插件的增加还在不断增加中。对不同文件格式的读写支持大大扩展了程序的使用范围，方便使用者对各种模型各种图形纹理的操作，而不需要进行过多的格式转换操作。着色器实现结果也可保存为用户应用程序需要的格式，同时可以在不同的环境下进行再编辑，二次开发，大大扩展了着色结果的可用性、复用性。

## 3 实现结果

本文借助 Visual C++ 2008 Feature Pack 最新的 MFC 类库构建了丰富的用户界面(如图 4 所示)：

1) 支持 Office 2007、Office 2003 和 Office XP 外观的界面；

2) 完全可定制的工具栏和菜单；

3) Visual Studio 风格的对接工具栏及可停靠窗格

4) 更加丰富的高级 GUI 控件等；

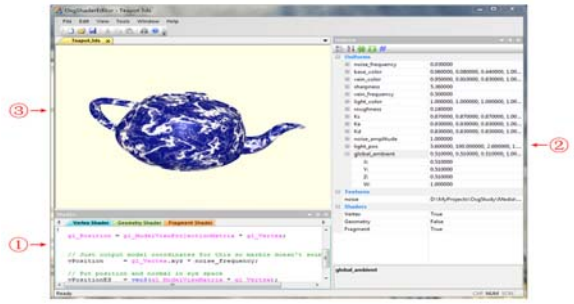



图 4　用户界面

### 3.1 着色器实现示例

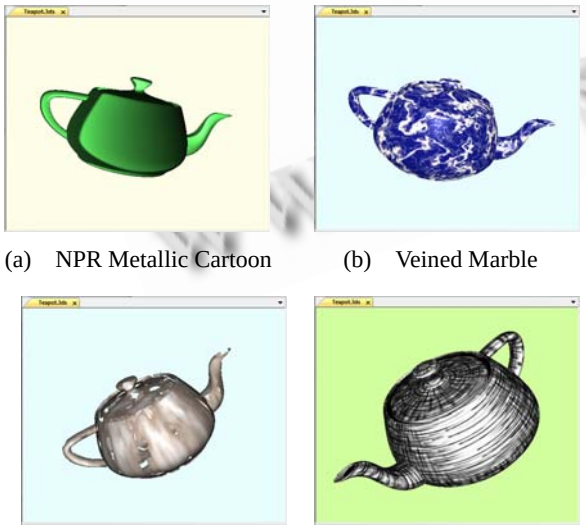
首先，导入我们需要着色的 3D 模型，由于 OSG 插件的支持我们可以导入现有的各种 3D 模型。并在预览窗口中实时的显示，并可对导入模型进行模型信息的显示，以及对模型的各种交互操作。

其次，为需要着色的模型编写着色器代码。

并为着色器代码添加一致变量、纹理，并设置初始值。将着色器代码、一致变量和纹理等着色器关键元素与导入的 3D 模型绑定，并在预览窗口显示着色效果。

最后，修改一致变量的值，设置纹理的滤波方式、边界处的截取方式、明细层次和边界颜色，以达到我们需要的着色效果。最终将着色效果导出为我们需要的文件格式。

以下是使用本文实现的 GLSL 着色器编辑环境实现 RenderMonkey(AMD&ATI)中的几款不同着色效果(如图 5 所示)。



(a) NPR Metallic Cartoon　(b) Veined Marble

(c) Erosion　(d) NPR Hatch

图 5　着色器实现示例

与 RenderMonkey 相比，本文实现的 GLSL 着色器编辑环境更易于初学者使用，着色结果的扩展性也更好：

1) 支持最新的几何着色器的代码编辑功能；

2) 预览窗口交互操作更加自然、方便，对模型渲染调试信息的更多支持；

3) 对第三方 3D 模型、2D 图形纹理格式的支持更加丰富，扩展了应用范围和着色效果的可用性、复用性，方便的与应用程序更好的结合；

## 4 结论和未来工作

本文借助于最新的 MFC 类库以及 0SG 对 OpenGL 功能的丰富封装，构建了一个 OpenGL 着色语言的编辑环境。整个编辑环境简洁、直观；实现了语法上色、关键字大小写识别的着色器代码编辑器；支持多达几十种的 3D 模型、2D 图形格式的导入导出操作；支持对一致变量、纹理的 GUI 编辑操作；同时提供了一个预览窗口方便的进行实时观察着色效果和调试工作。

虽然本文提供了一个用于 OpenGL 着色器开发的工具，但是和其它现有工具一样没能很好的解决调试和剖析着色器的工作；也没能实现一致变量的每帧变化，进而达到各种动画特效；还有一些设计和实现上的不足需要在未来的工作中不断改进而完善。

## 参考文献

1 Rost RJ.天宏工作室译.OpenGL 着色语言.北京:人民邮电出版社,2006.23－25.

2 王锐,钱学雷.OpenSceneGraph 三维渲染引擎设计与实践.北京:清华大学出版社,2009.

3 Martz P.王锐,钱学雷译.OpenSceneGraph 快速入门指导.http://www.osgbooks.com/books/osg_qs.html

4 Shreiner D, Woo M, Neider J, Davis T.徐波译.OpenGL 编程指南(第五版).北京:机械工业出版社,2006.

5 Gamma E, Helm R, Johnson R, Vlissides J.李英军,马晓星,蔡敏,刘建中,等译.设计模式——可复用的面向对象软件的基础.北京:机械工业出版社,2009.