

基于 GPU 的三维弧面渐变填充技术研究与实践

罗童心,仇建伟,王家润
(华北计算技术研究所,北京 100083)

摘要:在三维标图绘制应用系统中,线面标号的绘制经常使用渐变填充处理,以更直观地表达绘制对象所包含的趋势和程度信息。为解决标绘于三维地球上的光滑弧面对象的渐变填充绘制效率低下的问题,设计并实现了辐射渐变和线性渐变两种不同的渐变填充效果的高效算法;同时通过渐变方向、中心位置和颜色权重等参数来控制渐变效果,以适应实际使用中多样化的显示需求。考虑到固定管线渲染在大量点面数据处理上的劣势,算法使用 GPU 内部的可编程着色器,在不影响绘制效果的基础上大幅提升了绘制速度。实验结果表明,基于着色器对三维光滑弧面渐变填充,相比于固定管线渲染有着突出的性能优势,能够将运行时间缩短 5 倍以上,且在数据量较大的情况下也能保证较高的显示帧率。

关键词:OpenGL;GLSL;着色器;三维曲面

中图分类号:TP302

文献标识码:A

文章编号:1673-629X(2018)03-0165-04

doi:10.3969/j.issn.1673-629X.2018.03.035

Design and Implementation of 3D Line Symbol Gradient Fill Based on GPU

LUO Tong-xin, QIU Jian-wei, WANG Jia-run
(North China Institute of Computing Technology, Beijing 100083, China)

Abstract:In 3D plotting application system, the gradient fill of surface has been applied in many occasions to more intuitively expresses the trend and degree information in the drawing object. In order to solve the problem of low efficiency of gradient filling in smooth surface objects drawn on 3D earth, we design and realize an efficient algorithm of two different gradient filling effects (radiation gradient and linear gradient). At the same time, by setting the parameters like direction, center position and color weight it controls rendering effect to adapt the diversification of demand in actual use. Considering the low efficiency of fixed pipeline rendering in processing a large number of data, the algorithm uses a programmable shader inside the GPU to deliver a substantial performance increase while maintaining satisfactory rendering results. Experiments show that compared with traditional fixed rendering pipeline, shader has achieved outstanding performance advantages, reduction of the elapsed time by more than 80 percent, and ensuring high display frame rate in the case of a large amount of data.

Key words:OpenGL;GLSL;vertex shader;3D surface

0 引言

对于三维标图软件,线面标号绘制是其基本的功能。为使图形符号表达更多含义,尤其在涵盖颜色信息外还需要更直观形象地体现某种趋势和程度时,通常会对线面标号进行渐变填充处理,即利用两种不同颜色的混合效果,展现诸如污染程度、进攻方向和颜色权重等信息。这种绘制效果在轨迹绘制等场合也发挥着重要作用。此外,相比于单一色彩的填充,渐变填充通常有更好的视觉效果^[1-2]。

目前的三维标图应用广泛采用的是基于传统图形流水线的绘制方法,即使用固定功能的图形流水线。该方法需要对表面上的每个顶点绑定其对应的颜色属性等数据,再交由图形流水线完成余下渲染过程,其数据计算主要由 CPU 执行。如将该方法应用到三维场景的线面标号渐变填充中,则可能无法发挥较高的性能。这是由于为使线面标号的绘制贴合球面,通常需要填充的是细分程度较高的光滑弧面,这意味着有数量较多的顶点和三角面片数据,而在渐变填充中每个

收稿日期:2017-04-16

修回日期:2017-08-23

网络出版时间:2017-12-05

基金项目:总装“十三五”预研课题(31511070401)

作者简介:罗童心(1994-),女,硕士研究生,研究方向为三维图形处理和几何造型;仇建伟,硕士,研究员,研究方向为网络与分布式多媒体协同技术。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20171205.0904.040.html>

顶点的颜色不尽相同,使得 CPU 串行计算每个顶点的颜色值将耗费大量时间,此外为每个顶点存储其对应的颜色属性也需要占用大量空间^[3-4]。这些问题使得图形绘制效率较低,在复杂态势标绘应用中无法满足实时快速显示的要求。

为提高三维线面标号的绘制速度,文中在固定管线渲染的基础上使用能够替代原有功能模块的可编程模块,即顶点和片元着色器,将部分原本在 CPU 中串行计算的大量顶点操作转移到 GPU 中进行并行计算^[5]。当应用于三维弧面渐变填充的场合时,该方法只需传递少量参数,在顶点着色器内部利用顶点位置等信息计算顶点对应颜色值,同时省去颜色属性数组的构建和传输,在空间和时间性能上都有大幅度的提升。

1 算法基本概念与理论

1.1 球面的经纬映射

经纬映射是将球面上的三维笛卡尔坐标系下的点转换为经纬度坐标的映射方式,在三维地球的标绘中通常用该映射方式取代由三维坐标系到二维坐标系的正投影或透视投影变换。

对于球心位置为原点(0,0,0)的球,球面上坐标为(x,y,z)的三维空间点转换到经纬度坐标系的公式如下^[6]:

$$\tan\beta = \frac{y}{x} \quad (1)$$

$$\sin\alpha = \frac{z}{\sqrt{x^2 + y^2 + z^2}} \quad (2)$$

为使弧面的渐变填充效果均匀平滑地贴合球面弧度,算法需利用球面的经纬映射将在三维地球上弧面的各顶点从三维空间坐标系映射到二维坐标系下。

1.2 渐变模型

1.2.1 关键参数及含义

(1)颜色混合参数:对于弧面上每个点,都对应一个颜色混合参数值,表示该位置上参与渐变的两种颜色的混合比例,其取值范围为0.0~1.0。根据每个顶点的颜色混合参数 para,利用式(3)可以求得该位置应该被填充的颜色值。

$$\text{VertexColor} = (1 - \text{para}) * \text{MainColor} + \text{para} * \text{SubColor} \quad (3)$$

(2)渐变基准:可指定为弧面上任意位置,其对应的颜色混合参数为某个固定值,被作为弧面上其他点计算颜色混合参数的参照,通过弧面各个顶点到渐变基准的距离可以计算出其他位置对应的颜色混合参数。

(3)渐变半径:即渐变基准到弧面包围体边界的

距离,用于计算颜色混合参数。

$$\text{各顶点的颜色混合参数} = \frac{\text{顶点到渐变基准距离}}{\text{渐变半径}} \quad (4)$$

1.2.2 辐射渐变

如图1(a)所示,辐射渐变的渐变基准为一个点,该点的颜色混合参数为固定值0;辐射渐变的渐变半径即为指定圆心为渐变基准的包围圆半径;位于包围圆圆周上的点,其颜色混合参数均为1,其他位置的颜色混合参数则在其与渐变基准之间按距离远近插值。辐射渐变可以直观表达弧面上各点距离渐变基准的位置远近,通常应用于对区域内辐射强度等信息的表达。

1.2.3 线性渐变

如图1(b)所示,线性渐变的渐变基准是一条垂直于渐变方向的直线,该直线上所有位置的颜色混合参数均为固定值0.5;线性渐变的渐变基准到沿渐变方向的方向包围盒两端距离不一定相等,因此线性渐变的渐变半径有两个取值;在沿渐变方向包围盒边界,颜色混合参数为1,在与渐变方向相反的包围盒边界,颜色混合参数为0,其他位置的颜色混合参数则在二者与渐变基准之间按距离进行插值。线性渐变可以直观表达弧面的方向趋势信息以及两种颜色参与渐变的权重,权重较大的颜色,在渐变过程中衰减得较慢,即该颜色在整个面的填充效果上占的份额较多,通常应用于行进方向等信息的表达。

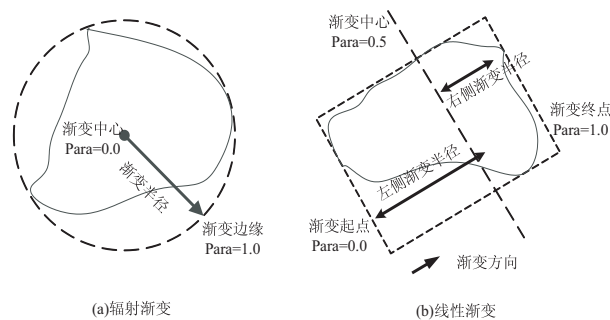


图1 两种渐变模型及参数示意

1.3 图形对象包围体

1.3.1 包围圆

对象的包围圆定义为包含该对象的最小的圆周。在辐射渐变的渐变半径计算中,需计算指定圆心位置的图形对象包围圆,即包围圆的圆心需为渐变基准点。在确定渐变基准位置后,将所有顶点遍历一次,计算并比较各顶点到中心位置的距离,其最长距离为指定圆心的包围圆半径,即辐射渐变的渐变半径。

1.3.2 方向包围盒

方向包围盒(OBB)是较为常用的包围体类型。它是包含指定绘制对象且相对于坐标轴方向任意的最小的长方体^[7]。对于线性渐变的渐变半径计算,则只

需求出该图形沿指定方向(线性渐变方向)的方向包围盒。遍历所有顶点,利用向量投影计算并比较各顶点到渐变基准的距离,同时运用向量的点积判断顶点相对于渐变基准的位置,在沿两个相反方向上的最大距离则分别为方向包围盒的上下边界到渐变基准的距离^[8],即为线性渐变的渐变半径。

2 着色器

2.1 着色程序与 GLSL 语言

实验使用 GLSL 语言编辑着色程序,实现着色器绘制,以提高渲染过程中 GPU 的利用率。顶点着色程序是被 GPU 内可编程顶点处理器执行的程序,应用程序设定的图元信息,如顶点坐标、颜色值、法向量、纹理坐标等顶点属性,传入到顶点着色器中进行并行处理。存在片元着色程序时,顶点着色器的输出会作为片元着色器的输入,再由片元着色器的输出决定屏幕上像素显示的颜色,在单一使用顶点着色时,则只对输入的顶点进行操作,顶点之间的部分则按照硬件默认的方式进行自动插值^[9-11]。

GLSL 是基于 OpenGL 的着色语言,在各显卡上均有较好的支持,相较于仅在 NVIDIA 显卡上得到支持的 Cg 等着色语言,具有良好的可移植性^[12]。GLSL 语言提供了少量的内置变量和部分内置函数。

实验中用内置变量 `gl_Vertex` 获取顶点的三维坐标,使用 `transform()` 内置函数对顶点进行 MVP 矩阵变换,以及使用 `dot()`、`length()` 函数计算向量点积和长度等。

2.2 Uniform 参数

考虑到图形流水线的结构和着色器的并行计算能力,实验选择将计算每个顶点各自颜色的过程放到顶点着色器中进行,无需再额外构建和绑定顶点颜色数组。由于着色器中每个顶点的计算都是独立在各个 GPU 单元间进行的,无法在独立单元之间进行数据交换,因此涉及其他顶点的计算需要在外部程序中预先求出,再作为常量参数 Uniform 或顶点的属性参数 Attribute 传给着色器。

其中 Uniform 参数是绑定到整个着色器、用于所有顶点的参数,而 Attribute 参数则是绑定到每个独立顶点的参数,通常为与顶点数组等大的数组^[13-15]。为保证着色器的效率,应尽量使用 Uniform 传参。

根据该算法的基本思想,对于中心渐变填充,选择将渐变半径和渐变基准点坐标作为 Uniform 常量参数传递给着色器;对于线性渐变填充,则将渐变半径、中心点坐标,以及渐变方向向量作为常量参数传递给着色器,其中线性渐变半径使用二维向量 `vec2` 类型以承载渐变基准向量的半径值。

3 算法设计与实现

3.1 绘制流程

图2给出了基于可编程图形管线的渐变填充绘制过程及每个计算阶段的输入输出内容,大致可以分为在 CPU 中进行的预处理计算过程和在 GPU 中进行的计算和渲染过程。其中顶点着色器和片元着色器替代了固定管线渲染中原有的固定功能操作模块,通过将原本在 CPU 中串行计算的部分转移到着色器中进行并行计算,以提升整个计算和绘制过程的效率。

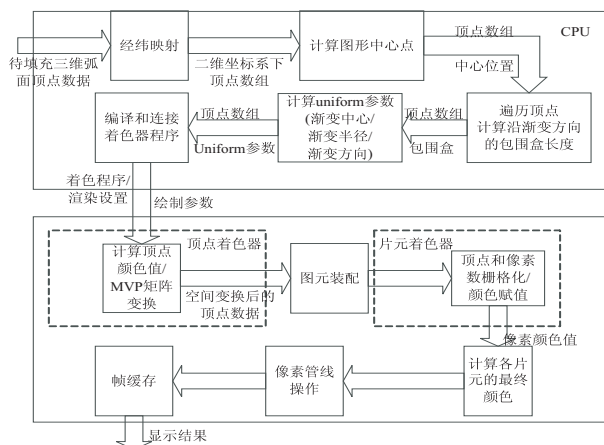


图2 计算和绘制过程

3.2 着色程序实现

以下给出着色程序的核心代码,用于并行计算弧面上各点显示的颜色,分别被顶点着色器和片元着色器执行:

(1) 顶点着色器。

```
uniform vec4 mainColor;//渐变主色
uniform vec4 subColor;//渐变辅色
uniform vec2 centerNode;//渐变基准位置
uniform vec2 radius;//两侧渐变半径值
uniform vec2 angle;//渐变方向角度,为减少三角函数计算使用方向向量代替
```

```
varying vec4 vertexColor;//顶点颜色
float para;//颜色混合参数
```

```
vec2 vector = gl_Vertex.xy - centerNode.xy;//由渐变基准指向顶点的向量
```

```
float distance = dot(vector, angle) / length(angle);//顶点向量沿渐变方向向量的投影距离,大于0表示向量夹角小于90°,反之大于90°
```

```
if(distance < 0.0) para = distance / radius.x;//计算颜色混合参数,向量夹角大于90°时应用左侧的渐变半径计算,反之应用右侧渐变半径计算
```

```
else para = distance / radius.y;
```

```
vertexColor = subColor * para + mainColor * (1 - para);//计算顶点颜色
```

```
transform();//顶点 MVP 矩阵变换
```

(2) 片元着色器。

```
varying vec4 vertexColor;
```


gl_FragColor=vertexColor;//渲染像素颜色

4 实验结果及分析

图 3 给出了基于可编程渲染管线绘制对地球上某弧面分别进行辐射渐变和线性渐变填充的效果。AB 为渐变基准位置不同的辐射渐变填充,CD 为渐变方向和颜色权重不同的线性渐变填充。其中 C 表示渐变角度为 0°,两种颜色权重均为 0.5 的线性渐变填充效果;D 表示渐变角度为 135°,颜色权重分别为 0.3 和 0.7 的效果。AC 的填充中未使用 α 通道。

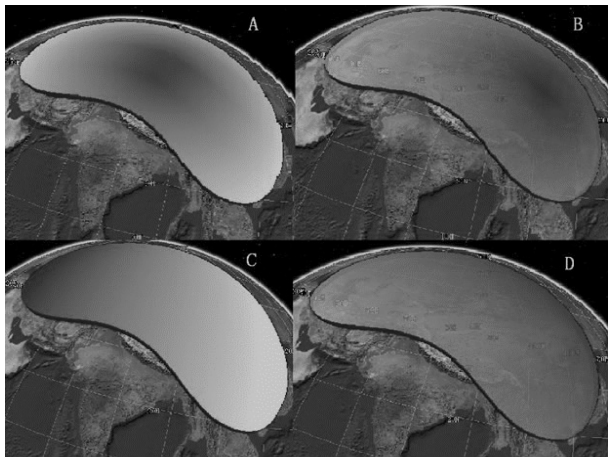


图 3 渐变填充效果

图 4 给出了使用着色器渲染的实验组与使用固定管线绘制的对照组在性能上的比对结果,分别从运行时间和显示帧率进行对照,展示了在单一绘制对象内,随着顶点数目和三角面片数目的增加,不同的绘制方案呈现的性能趋势。其中实线代表实验组的测试结果,虚线代表对照组的测试结果。测试结果统计均基于算法相对复杂的线性渐变填充。

由图 4 可以看出,在三角面数量较少、网格较为简单时,二者并未呈现较大的性能差异;当三角网格数目达到十万级时,对照组的显示帧率已经开始大幅降低,达到百万级时,帧率已经下降到 20 帧以下,几乎不足以支持流畅显示;而即使三角网格数目达到 100 万时,实验组仍能保持流畅的显示,帧率维持在 30 帧以上,直到数量接近 200 万时才不足 20 帧。对照组的运行时间始终呈线性增长趋势,实验组运行时间则仅在数量级跨度较大时有少量的增加,在三角面数量接近 200 万时,二者的运行时间差异有 5 倍之多。

由测试结果可以得出,使用着色器绘制的弧面渐变填充算法有着明显的性能优势,尤其针对绘制对象数量较少而三角面数较多的复杂线面标号,能表现出远优于传统固定管线绘制的性能。在渲染管线内使用着色器编程,更充分应用了 GPU 的计算能力,使得每个独立的顶点计算能够并行进行,极大地缩短了计算时间,优化了显示效果。

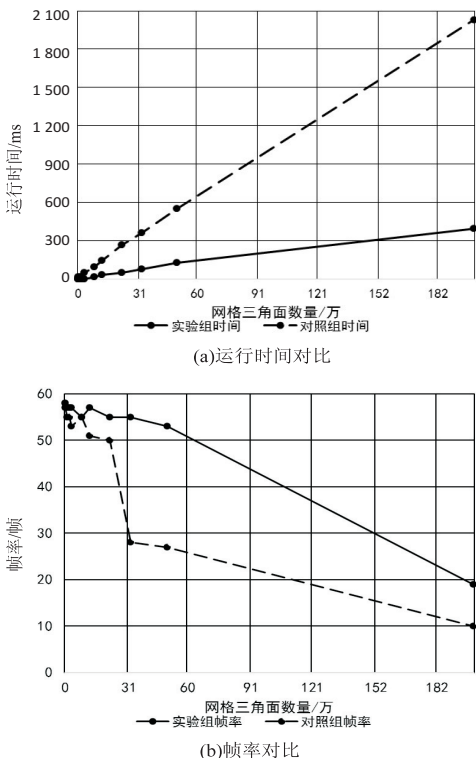


图 4 两种绘制策略在单一绘制对象中性能随三角面数的变化

5 结束语

针对传统固定管线渲染方法在沿地球弧面的三维线面标号渐变填充性能上表现出的不足,提出了基于 GPU 并行计算的性能优化方法。实验结果表明,通过在渲染管线过程中使用着色器编程,提高了算法对 GPU 的利用率,使得对于顶点的部分计算过程能够并行进行,极大提升了图形的绘制效率,同时也减少了顶点部分属性数组在空间上的占用,实现了较高性能应用于光滑弧面的两种渐变填充效果;同时通过在线面标号渐变填充这一典型实例上的应用和大量数据测试,验证了着色器在大量点线面数据处理上的高效率。

参考文献:

[1] 韩李涛,范克楠. 三维地形颜色渐变渲染的光滑过渡方法研究[J]. 地球信息科学学报,2015,17(1):31-36.
[2] 张旭,戴宁,廖文和,等. 基于三维渐变的牙齿磨耗可视化仿真[J]. 机械工程学报,2013,49(3):95-100.
[3] 简洪登,范湘涛. 基于 GLSL 的多重视频纹理映射与融合[J]. 计算机工程与设计,2014,35(11):3873-3878.
[4] MILOSAVLJEVIĆ A,DIMITRIJEVIĆ A,RANČIĆ D. GIS-augmented video surveillance[J]. International Journal of Geographical Information Science,2010,24(9):1415-1433.
[5] 裘初,费广正,石民勇. 可编程图形硬件综述[J]. 北京广播学院学报:自然科学版,2004,11(3):13-19.

数据不能被及时处理,起到一个缓冲的作用。Kafka有两个接收端,除了上面用来计算浏览器指纹的Storm,还有下面的 Storm,在计算浏览器指纹的同时,它根据用户的信息,去 Hbase 中查找是否存在该用户的浏览器指纹,如果存在,则将信息同步到 Redis 中。

Storm 是分布式实时计算,具有较强的实时性,在这里用来计算浏览器指纹,具有较高的效率,能够保证系统的实时性,因此使用上面的技术架构可以达到非常好的应用效果。

3 结束语

随着网络技术的迅速发展,手机、平板电脑等各种智能设备的加入,网络访问和交易也变得越来越频繁,各种安全问题也逐渐凸显。传统的基于用户名密码的验证手段已经不能满足安全的需要,因此需要综合用户的多种信息为其建立一个唯一标识,以此来提高认证的准确性,保护用户的信息安全。

文中介绍了浏览器指纹的生成算法及其应用系统架构的设计。在指纹识别的过程中,综合使用了完全匹配和部分匹配的方式,可以有效地识别有效指纹的变种,降低错误增加指纹的数目,达到最佳的指纹识别效果。在浏览器指纹的架构设计中,综合考虑了高访问量和实时性的问题,在原系统的基础上,加上指纹识别系统,可以提高认证用户的安全性。随着电子商务遇到越来越多的安全问题,浏览器指纹在机器人防范、用户访问特征、客户端数据等方面必将有着更广泛的应用。

参考文献:

[1] 杨青龙.网购中的欺诈行为研究[J].东西南北·教育观察,2011(9):19.
[2] MIYAZAKI A D,FERNANDEZ A. Consumer perceptions of

privacy and security risks for online shopping[J]. Journal of Consumer Affairs,2001,35(1):27-44.
[3] 黎志成,刘枚莲.电子商务环境下的消费者行为研究[J].中国管理科学,2002,10(6):88-91.
[4] 吉 绚,胡 曼,刘广宇.网上购物安全性现状分析[J].中国水运:理论版,2006,4(12):116-117.
[5] 吴凌娇.网上购物安全问题探讨[J].江苏技术师范学院学报,2006,12(4):68-73.
[6] BODA K,FÖLDES Á M,GULYÁS G G,et al. User tracking on the web via cross-browser fingerprinting[M]//Information security technology for applications. Berlin:Springer,2011:31-46.
[7] 张梦媛.浏览器的安全访问及指纹识别技术[D].南京:南京邮电大学,2012.
[8] YEN T F,HUANG X,MONROSE F,et al. Browser fingerprinting from coarse traffic summaries:techniques and implications[C]//International conference on detection of intrusions and malware,and vulnerability assessment. [s. l.]:[s. n.],2009:157-175.
[9] 徐 晏,张代远.基于浏览器的用户身份识别系统[J].计算机技术与发展,2013,23(8):79-82.
[10] 王 维,王伟平,王建新.一种基于升级浏览器指纹的用户识别方法[C]//信息安全漏洞分析与风险评估大会.出版地不详:中国信息安全测评中心,2014.
[11] MUÑOZ - GARCÍA Ó, MONTERRUBIO - MARTÍN J, GARCÍA-AUBERT D. Detecting browser fingerprint evolution for identifying unique users[J]. International Journal of Electronic Business,2012,10(2):120-141.
[12] 姚全珠,蒋鹏飞,颜丽菁,等.基于浏览器指纹技术的预防黄牛党挂号系统[J].计算机应用,2016,36:276-279.
[13] 李周辉,黄燕群,唐 屹.浏览器识别研究[J].信息安全,2016(3):34-39.
[14] 陆 婷,房 俊,乔彦克.基于HBase的交通流数据实时存储系统[J].计算机应用,2015,35(1):103-107.

(上接第168页)

[6] 王俊杰,徐小刚,胡运发,等.鱼眼投影在虚拟实景中的应用研究[J].小型微型计算机系统,2004,25(2):287-290.
[7] 芦鸿雁.基于层次包围盒的碰撞检测算法研究[J].计算机与数字工程,2008,36(2):23-25.
[8] 李运锋,刘修国.基于方向包围盒投影转换的轮廓线拼接算法[J].计算机应用,2011,31(12):3353-3356.
[9] CIRNE M V M,PEDRINI H. Marching cubes technique for volumetric visualization accelerated with graphics processing units[J]. Journal of the Brazilian Computer Society,2013,19(3):223-233.
[10] WATSON B,LUEBKE D. The ultimate display:where will all the pixels come from[J]. Computer,2005,38(8):54-61.

[11] 韩俊刚,蒋 林,杜慧敏,等.一种图形加速器和着色器的体系结构[J].计算机辅助设计与图形学学报,2010,22(3):363-372.
[12] 黄 龙.基于GLSL三维渲染效果的研究与应用[D].西安:西安科技大学,2014.
[13] WOO J H,SOHN J H,KIM H,et al. A 195 mW,9.1M vertices/s fully programmable 3-D graphics processor for low power mobile devices[J]. IEEE Journal of Solid-State Circuits,2008,43(11):2370-2380.
[14] 陈继选,王毅刚.基于OSG的GLSL着色器编辑环境[J].计算机系统应用,2011,20(3):153-156.
[15] WOLF D. OpenGL4.0 shading language cookbook[M]. [s. l.]:PACKT Publishing,2011.