

硕 士 研 究 生 读 书 报 告



题目 OpenGL光照绘制处理研究

作者姓名 张灏

作者学号 21651030

指导教师 李启雷

学科专业 软件工程

所在学院 软件学院

提交日期 二○一七 年 四 月

Research on OpenGLIllumination Processing

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software engineering

Advisor: Li Qilei

By

Zhang Hao

Zhejiang University, P.R. China

2017

摘要

光照处理是增强图形真实感最重要的组成部分,包括光照模型的选择和明暗处理。通过研究影响物体表面反射光的相关因素,提出了真实感图形绘制中如何进行模拟光照计算和明暗插值计算的问题。在分析和研究OpenGL图形库下实现光照效果的改善,并系统地阐述了利用OpenGL函数在VC + +环境下实现光照图形的基本过程和方法。通过光照模型的建立, 展现了OpenGL强大的图形功能, 并论述了OpenGL和Visual C+ + 的MFC 相结合的方法和各种优点。同时研究了如何在场景中添加OpenGL光照,介绍了添加光照的基本步骤以及实现方法,并对如何设置物体的材质属性作了简要介绍。

**关键词**：光照处理，真实感图，OpenGL，MFC

Abstract

Illumination processing is the most important part of enhancing graphic reality, including the selection of light models and shading. By studying the relevant factors influencing the reflected light on the surface of the object, the problem of how to simulate the calculation of light and the calculation of light and dark interpolation is put forward. In the analysis and study of OpenGLgraphics library to achieve the effect of lighting improvements, and systematically expounded the use of OpenGLfunctions in the VC + + environment to achieve the basic process and method of light graphics. Through the establishment of the light model, it shows the powerful graphics function of OpenGL, and discusses the combination of OpenGLand Visual C + + MFC methods and various advantages. At the same time, we study how to add OpenGLillumination to the scene, introduce the basic steps of adding light and how to implement it, and give a brief introduction to how to set the material properties of the object.

**Keywords：**Lighting processing, realism, OpenGL, MFC

# 引言

近年来,真实感图形技术发展很快,在产品设计、动画、场景漫游等都有广泛地应用。真实感图形技术中光照处理是增强图形真实感最重要的手段,表面加有光照的物体图形能给人以多层次的颜色渐变或明暗过渡的三维视觉效果。光照处理是在已知物体物理形态和光源性质的条件下,基于光学物理中的有关定律,通过计算确定场景物体可见面上任何一点投向观察者眼中的光强度大小(或RGB值) ,模拟真实光照效果。目前的一些光照处理方法对于某些特定类型的物体表面可以得到非常精确的模拟效果,如光线跟踪法更适用于高反射物体表面;辐射度法更适用于所有表面都是完全漫反射特性的场景。但逼真度提高了,算法复杂度也大大增加了,图形生成跟不上实时性的要求。绘制真实感景物的目标是:用尽可能接近于实时处理速度去生成尽可能真实的明暗效果。

OpenGL作为一种图形与硬件的接口,与其他图形程序开发工具相比较,它提供了众多图形函数,直观的编程环境简化了三维图形程序。OpenGL的光照处理也体现出其在真实感图形开发的技术优势。用OpenGL实现Phong光照模型,实践证明计算效率更高,而且在各种光照条件下和不同材质属性时,用它绘制得到的结果都比较理想地接近于物理真实。

虚拟现实是一门涉及众多学科的新的实用技术，它集先进的计算机技术、传感与测量技术、仿真技术、微电子技术于一体。在计算机技术中，它又特别依赖于计算机图形学、人工智能、网络技术、人机接口技术及计算机仿真技术。这些相关技术的发展带动了虚拟现实技术的进步。而在图形处理种，OpenGL无疑是最佳的三维图形标准之一。

OpenGL的前身是由SGI公司为其图形工作站开发的IRIS GL是一个工业标准的三维计算机图形软件接口。它有GL的功能，而且是开放的，适用于多种硬件平台及操作系统，可以灵活方便的实现二维和三维的图形技术，在性能上表现得异常优越。用户可以方便的利用这个图形库，创建出接近光线跟的高质量静止或动画的三维彩色图像，而且是要比光线跟踪算法快一个数量级。以OpenGL为基础开发的应用程序可以十分方便的在各种平台间移植。现笔者仅对利用OpenGL进行光照处理做一点探讨。

对物体进行透视投影，然后在可见面上产生自然光照效果，可以实现场景的真实感显示。一个光照明模型亦称光照模型，或明暗模型，主要用于物体表面某点处的光强度的计算。面绘制算法是通过光照模型中的光强度计算，来确定场景中物体表面的所有投影像素点的光强度。在实际工作时通常由两种做法：其一是将光照模型应用于每个可见面的每个点，另一种方法则是，经过少量的光照模型计算而在面片上对亮度进行插值。要绘制逼真的三维物体，必须做光照处理。OpenGL可以控制光照与物体的关系，产生多种不同的视觉效果。利用OpenGL进行光照处理时，在屏幕上最终显示的像素颜色，受到glColor命令指定的颜色影响，同时也要反映出在场景中使用光照的特性，以及物体反射和吸收光的属性，OpenGL的光是由红、绿、蓝颜色的数量决定，材料的属性是由在不同方向反射、入射的红、绿、蓝的百分数决定的，OpenGL的光照方程只是一个近似的，但是计算量较小，也比较精确。OpenGL的光可来自多个光源，每光源可以单独控制开关，有的光来自某个特定的方向、位置，也有的光源分散在整个场景，如墙壁的泛光；经过来自光源的光线多次反射，无法确定其光线的方向，要绘制真实的三维物体，仅有光源是不够的，只有物体表面吸收、反射光线时，光源才起作用，而材料本身可能发光，也可能漫反射光线，或在特定方向反射光线，光照只对有上述属性的材料起作用。

光线由四个成分构成：发射光、环境光、漫反射、镜面光。这四个成分单独计算，然后加起来。发射光最为简单，它来自物体且不受任何其他光源的影响。环境光从光源发出并经过多次反射形成，它均匀地从周围环境入射至物体表面并朝各个方向等量反射。漫反射光来自一个方向，直接照射到表面的要比仅仅扫过表面亮，一旦照射到表面上，无论在何处观察，亮度相同。镜面光来自一个特定方向，以一个特定方向离开，可以把这个成分看成材料的光洁度。材料颜色取决于反射的红、绿、蓝光的百分数，与光线的特性相似，材料也有泛光、扩散、反射颜色。材料的泛光与每个入射光源的泛光组份相结合，漫反射与光源的漫反射组合，镜面反射与镜面反射组合相结合。泛光和漫反射定义材料的颜色，两者通常是相似的，镜面反射通常是白或灰。

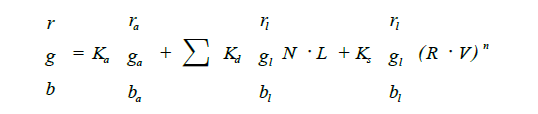
# 光照模拟

光照模型：

根据光学、物理学的知识,自然界中的物体表面显现的颜色是由其表面向视线方向反射的光能及其所含不同波长光的比例决定的。但在计算机上生成的真实感图形要进行光谱计算和光谱至颜色的转换会耗费很多的计算时量。一般采用适当的数学模型公式来近似和模拟物体表面对光的反射状况。研究表明,从物体表面反射出来的光强决定于光源的位置、光源强度和物体材质、物体表面位置、朝向及视点的位置等。物体表面上任一点,射向视点的光强是环境反射光、漫反射光和镜面反射光的共同作用。对于多个光源,可将它们的效果线性叠加,如Phong光照模型：



N、L分别为表面法线和沿光源入射方向的单位向量; R、V分别为反射方向和视线方向的单位向量。Ka、Kd、Ks 分别是材质对环境光、漫射光、镜面光的反射系数; n反映了反射光的会聚程度。在给定了光源和物体材质属性后, 反射光强仅与N、L、R、V有关。实际应用时将其转换成光栅图形显示器的RGB 三基色模式：



光照组成

1、发射光：发射光是最简单的，它由物体发出，并且不受任何光源的影响。

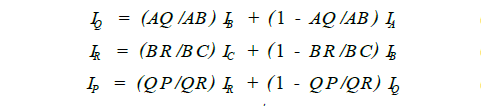
2、环境光：环境光是从某个光源发出并由环境经多次散射得到的，难以确定其最初的方向，看起来好像来自于四面八方。环境光照射到一个表面上时，将均匀地在各个方向都产生散射。

3、散射光：散射光来自于一个固定方向，所以当光线垂直照射到物体表面上时要比斜照时亮一些。然而，一旦光线照到表面上就均匀地在各个方向都发生散射，因此不论观察点处于哪个位置，其亮度都是一样的。来自于特定位置与方向的任何光线都可能带有散射光成分。

4、镜面反射光：镜面反射光也来自于特定方向，但它几乎全部反射到对称的方向上。光滑的镜面、金属都可以产生较高的镜面反射成分，而柏油马路或地毯等材质几乎不发生镜面反射。镜面反射反映了物体的光洁度。OpenGL独立计算四种成分光线，然后再将其结果叠加到一起产生最终的光照效果。

明暗插值

用前面提出的光照模型计算图形表面每一点RGB值,计算量非常大。一般把物体几何表面分解成若干多边形面片,充分利用多边形的高效绘制的特性使计算量大大减轻。当光源和观察者距离多边形较远, 或者多边形面积相对这个距离较小时, 多边形面片中各点的三个向量N、L、V可近似为常数, 每个多边形的色调只需计算一次, 其所有点被赋予相等的明暗色调。但是这种平面明暗处理方法在光源和观察者离多边形较近时,N 为常数而L、V均不相同,结果会导致物体曲表面呈现多面体状,物体曲表面光强度或颜色是不连续跃变的,失去原有的光滑性。本文采用Gouraud明暗插值处理,在用扫描线算法绘制多边形时,首先进行基于顶点的光照计算,然后用顶点的光强度或颜色值通过双线性插值计算出面片内每一点的光强度或颜色。



增量法求出相邻点光强　I′P = IP + ( IQ - IR ) /QR

经线性插值可得到光滑的图形效果(图4) 。一般来说,由于顶点的个数远小于像素点的个数,在硬件支持下线性插值计算的速度非常快, 适于实时处理。但在大平面上插值计算会使多边形内的镜面高光丢失。Phong法向插值可解决这个问题,但计算量太大, 帧频率会下降, 影响实时应用。其实在多边形个数较多、面积很小的精确建模时, 两者方法产生的光照效果差别并不大。

材质颜色

材质是指物体表面对光的反射特性。一种材质由环境色、散射色、镜面高亮色等组成，分别说明了它对环境光、漫反射光和镜面反射光反射的多少，即反射率。材质影响物体的颜色、反光度、透明度等。一个只反射蓝光的物体，在一束白光的照射下就呈蓝光。

材质和光的相互作用

一个光源决定了光的颜色、反射特性、强度和入射光的方向。物体的几何性质及材质决定发射光的颜色、反射特性、强度。光和物体间的相互作用远比我们能够实时模拟的情况复杂。比如，在不同光照条件下模拟人的皮肤的颜色是一件十分困难的事。OpenGL在逼真显示和实时绘制之间通过以下方法实现均衡：只在几何顶点进行光照计算，对在所绘制的每个像素上执行光照运算的复杂处理进行简化。光和材质的颜色都采用红、绿、蓝三原色合成，如果光源的参数为（RL，GL，BL），材料的参数为（RM，GM，BM），若不考虑其他影响因素，则观察到的光线的参数为（RL\*RM，GL\*GM，BL\*BM）。另外对于多个光源来说，它们的各个成分满足叠加原理，但要注意叠加和超过3 的颜色分量将被置为3

# 程序实现

增加光照

1、为了能够确定物体相对于光源的方位，首先需要定义物体上每个顶点的法向量。OpenGL中通过法向量来确定指定顶点从每个光源得到的受光量，并用于像素点的颜色计算中。

2、创建、选择一个或者多个光源，并为光源定位。

3、创建并选择光照模型，它定义了全局环境光的等级和观察点的有效位置（用于光照计算）。

4、为场景中的物体定义材质属性。

主要函数

1、

void glLight {I，F}（Glenum light，Glenum pname，Glenum param）；函数功能：创建光源（由参数light指定）。

light： 指定创建的光源。光源数目由GL\_MAX\_LIGHTS设定（最少支持8个光源），各个光源以GL\_LIGHTi的形式标识。

pname：为light指定光源参数。GL\_AMBIENT指定光源的环境光亮度；GL\_DIFFUSE指定指定光源散射光亮度；GL\_SPECULAR 指定光源的镜面反射光亮度；GL\_POSITION指定指定光源的位置GL\_SPOT\_DIRECTION指定聚光方向；GL\_SPOT\_EXPONENT指定聚光指数；GL\_SPOT\_CUTOFF指定聚光终止角度；GL\_CONTANT\_ATTENUATION指定恒定衰减因子；GL\_LINEAR\_ATTENUATION指定线性衰减因子；GL\_QUADRATIC\_ATTENUATION指定二次衰减因子。

Param：指定light光源的参数pname的值，在向量形式下，param是一个指向数组的指针，非向量形式下是一个数值。

2、

void glLightModel {I，F}（Glenum pname，TYPE param）；函数功能：设定场景中的光照模型。

Pname：设置光照模型的特征。GL\_\_MODEL\_AMBIENT定义场景全局环境光的强度；GL\_LIGHT\_MODEL\_LOCAL\_VIEWER确定视点位置；GL\_LIGHT\_MODEL\_TWO\_SIDE确定是单面受光还是双面受光。

param：指定待要设定的参数值。

3、

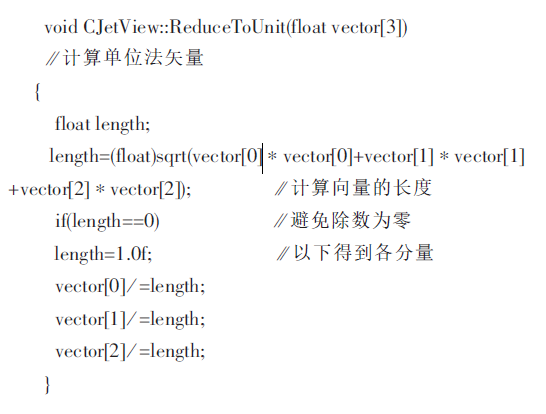
void glMaterial {I，F}（Glenum face，Glenum pname，TYPE param）；函数功能：定义当前的材质属性。

face：指定待设定材料属性的物体表面。取值为GL\_FRONT、GL\_BACK或GL\_FRONT\_AND\_BACK。

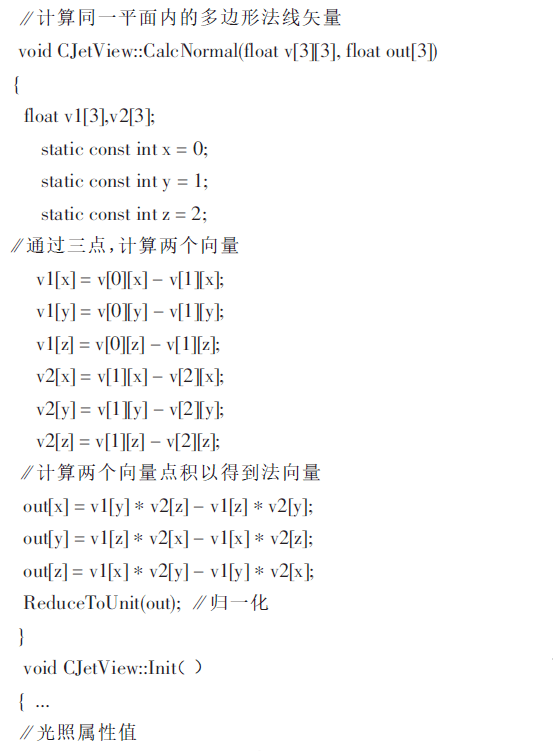
pname：指定材质参数、GL\_AMBIENT材质的环境颜色；GL\_DIFFUSE材质的散射颜色；GL\_AMBIENT\_AND\_DIFFUSE材质的环境颜色和散射颜色；GL\_SPECULAR材质的镜面反射颜色；GL\_SHINESS镜面反射参数；GL\_EMISSION材质的反射光颜色。

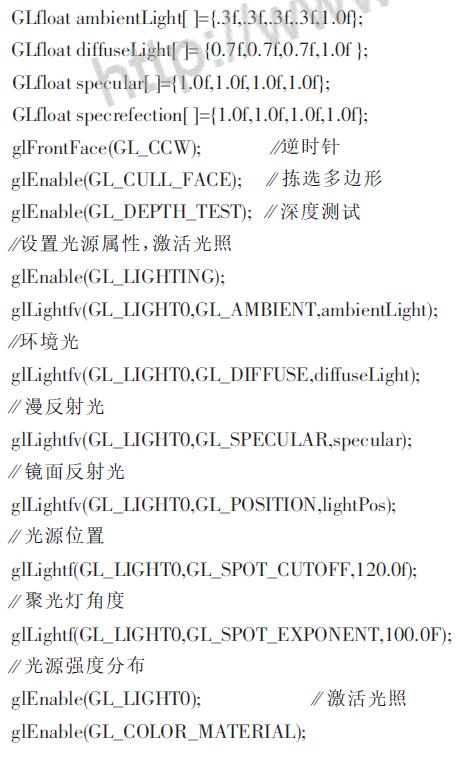
主要源代码

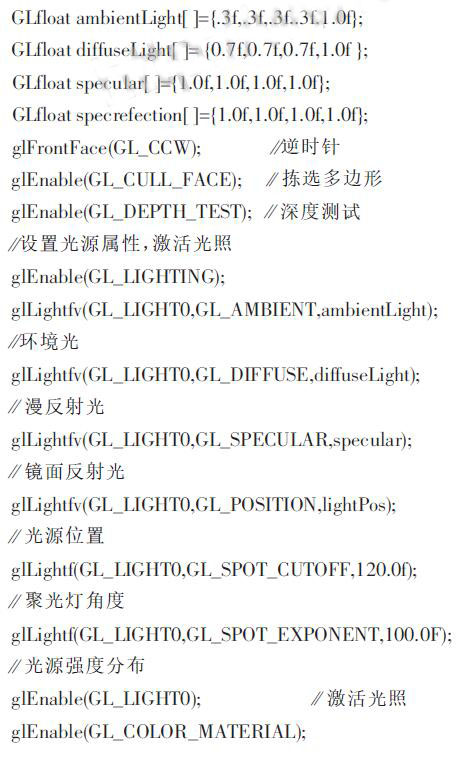
对于光照计算来说，可以让OpenGL自动将法线转化成单位法线（归一化）glEnable（GL\_NORMALIZE）但是，这种方法是带有惩罚性的一种操作，提前把法线归一化比依靠OpenGL替你做这个工作要好得多。

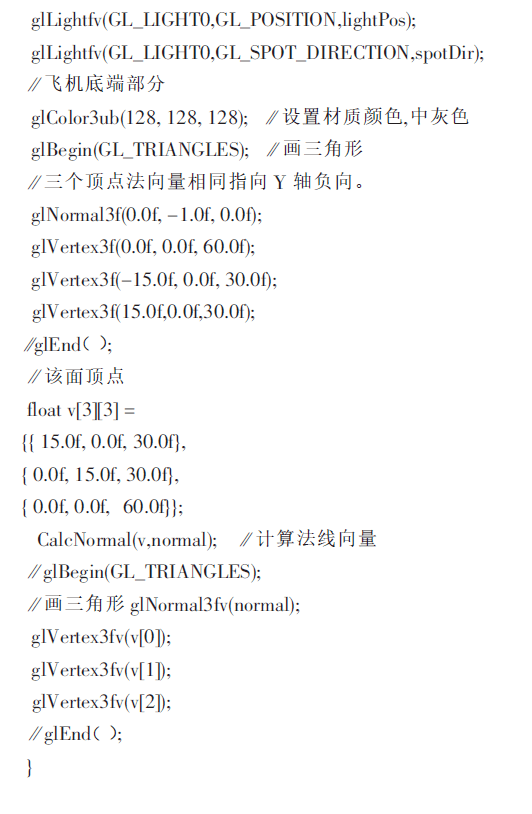


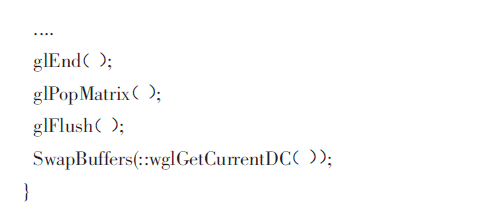
显然，为每个顶点或者多边形指定一条法线很麻烦，尤其是几乎没有几个表面在坐标轴的主要平面上，故以下提供一个函数可以反复地调用它来计算法线。考虑到三角形具有共面性这一特点，在绘制模型时，都采用小三角形片逼近。











#include <gl/glut.h>

#define WIDTH 400

#define HEIGHT 400

static GLfloat angle = 0.0f;

void myDisplay(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); //清空颜色和深度缓冲

// 创建透视效果视图

glMatrixMode(GL\_PROJECTION);//对投影矩阵操作

glLoadIdentity();//将坐标原点移到中心

gluPerspective(90.0f, 1.0f, 1.0f, 20.0f);//设置透视投影矩阵

glMatrixMode(GL\_MODELVIEW);//对模型视景矩阵操作

glLoadIdentity();

gluLookAt(0.0, 5.0, -10.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);//视点转换

// 定义太阳光源，它是一种白色的光源

{

GLfloat sun\_light\_position[] = {0.0f, 0.0f, 0.0f, 1.0f};

GLfloat sun\_light\_ambient[] = {0.0f, 0.0f, 0.0f, 1.0f};

GLfloat sun\_light\_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};

GLfloat sun\_light\_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};

glLightfv(GL\_LIGHT0, GL\_POSITION, sun\_light\_position); //指定第0号光源的位置

glLightfv(GL\_LIGHT0, GL\_AMBIENT, sun\_light\_ambient); //GL\_AMBIENT表示各种光线照射到该材质上，

//经过很多次反射后最终遗留在环境中的光线强度（颜色）

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, sun\_light\_diffuse); //漫反射后~~

glLightfv(GL\_LIGHT0, GL\_SPECULAR, sun\_light\_specular);//镜面反射后~~~

glEnable(GL\_LIGHT0); //使用第0号光照

glEnable(GL\_LIGHTING); //在后面的渲染中使用光照

glEnable(GL\_DEPTH\_TEST); //这句是启用深度测试，这样，在后面的物体会被挡着，例如房子后面有棵树，如果不启用深度测试，

//你先画了房子再画树，树会覆盖房子的；但启用深度测试后无论你怎么画，树一定在房子后面（被房子挡着）

}

// 定义太阳的材质并绘制太阳

{

GLfloat sun\_mat\_ambient[] = {0.0f, 0.0f, 0.0f, 1.0f};

GLfloat sun\_mat\_diffuse[] = {0.0f, 0.0f, 0.0f, 1.0f};

GLfloat sun\_mat\_specular[] = {0.0f, 0.0f, 0.0f, 1.0f};

GLfloat sun\_mat\_emission[] = {0.5f, 0.0f, 0.0f, 1.0f};

GLfloat sun\_mat\_shininess = 0.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, sun\_mat\_ambient); //定义材料的前面采用 "多次反射"

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, sun\_mat\_diffuse); //材料的前面为 漫反射

glMaterialfv(GL\_FRONT, GL\_SPECULAR, sun\_mat\_specular); //定义材料的前面为 镜面反射

glMaterialfv(GL\_FRONT, GL\_EMISSION, sun\_mat\_emission); //定义材料的前面为 镜面指数

glMaterialf (GL\_FRONT, GL\_SHININESS, sun\_mat\_shininess); //材料的前面 采用 的颜色

glutSolidSphere(2.0, 40, 32);

}

// 定义地球的材质并绘制地球

{

GLfloat earth\_mat\_ambient[] = {0.0f, 0.0f, 0.5f, 1.0f};

GLfloat earth\_mat\_diffuse[] = {0.0f, 0.0f, 0.5f, 1.0f};

GLfloat earth\_mat\_specular[] = {0.0f, 0.0f, 1.0f, 1.0f};

GLfloat earth\_mat\_emission[] = {0.0f, 0.0f, 0.0f, 1.0f};

GLfloat earth\_mat\_shininess = 30.0f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, earth\_mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, earth\_mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, earth\_mat\_specular);

glMaterialfv(GL\_FRONT, GL\_EMISSION, earth\_mat\_emission);

glMaterialf (GL\_FRONT, GL\_SHININESS, earth\_mat\_shininess);

glRotatef(angle, 0.0f, -1.0f, 0.0f);

glTranslatef(5.0f, 0.0f, 0.0f);

//如果使用glutSolidSphere函数来绘制球体，则该函数会自动的指定这些法线向量，

//不必再手工指出。如果是自己指定若干的顶点来绘制一个球体，则需要自己指定法线向量。

glutSolidSphere(2.0, 40, 32);

}

glutSwapBuffers(); //交换缓冲区

}

void myIdle(void)

{

angle += 1.0f;

if( angle >= 360.0f )

angle = 0.0f;

myDisplay();

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE);

glutInitWindowPosition(200, 200);

glutInitWindowSize(WIDTH, HEIGHT);

glutCreateWindow("OpenGL光照演示");

glutDisplayFunc(&myDisplay);

glutIdleFunc(&myIdle);

glutMainLoop();

return 0;

}

# 总结

OpenGL把光照分解为光源、材质、光照模式三个部分，根据这三个部分的各种信息，以及物体表面的法线向量，可以计算得到最终的光照效果。

光源、材质和光照模式都有各自的属性，尽管属性种类繁多，但这些属性都只用很少的几个函数来设置。使用glLight函数可设置光源的属性，使用glMaterial函数可设置材质的属性，使用glLightModel函数可设置光照模式。

GL\_AMBIENT、GL\_DIFFUSE、GL\_SPECULAR这三种属性是光源和材质所共有的，如果某光源发出的光线照射到某材质的表面，则最终的漫反射强度由两个GL\_DIFFUSE属性共同决定，最终的镜面反射强度由两个GL\_SPECULAR属性共同决定。

可以使用多个光源来实现各种逼真的效果，然而，光源数量的增加将造成程序运行速度的明显下降。

在使用OpenGL光照过程中，属性的种类和数量都非常繁多，通常，需要很多的经验才可以熟练的设置各种属性，从而形成逼真的光照效果。。然而，设置这些属性的艺术性远远超过了技术性，往往是一些美术制作人员设置好各种属性（并保存为文件），然后由程序员编写的程序去执行绘制工作。因此，即使目前无法熟练运用各种属性，也不必过于担心。

参考文献

[1] Teaching OpenGL and Computer Graphics with

Programmable Shaders. Hongwei Li[M]. Proceeding SA '14 SIGGRAPH Asia 2014 Courses.2014.3.6．

[2]OpenGL中的光照技术研究.杨键 张敏[J].软件导刊. 2011,10(4):84-86．

[3] 基于OpenGL的光照处理技术及实现.李治国 郭立[J].计算机仿真. 2008,25(3):221-223．

[4] 基于OpenGL的光照处理技术绘制真实感图形.王玉华 杨克俭 王玲[J].现代计算机.2002(9):72-75．