同济大学
TONGJI UNIVERSITY

计算机图形学大作业
——“三维弹球”项目报告

学　　院：电子与信息工程学院

专　　业：计算机科学与技术

指导教师：赵君峤

组　　员：**1652195 肖潇灏　1652212 徐凯**

1652208 孟庆勇　1652205 齐济

1651629 黄中昱　1651792 杨龙雨

2018 年　12 月　15 日

一、项目概述

1.1 Motivation

实时渲染视频级别的计算机三维图形是计算图形领域的终极目标，与现在普遍使用的光栅化渲染技术相比，光线追踪普遍被视为技术的未来方向。光想追踪技术可带来真实的真正电影级别图形和光影物理效果。

研究光线追踪技术对于学习计算机图形学来说意义重大，所以我们小组想通过制作一个三维弹球光线追踪器来学习并掌握简单的光线追踪技术。

1.2 Goal of the project

先做出一个带材质纹理的三维弹球模型，然后再加上周围环境背景，最后以光线的反向追踪为基础，实现一个支持场景描述、光线反射、软阴影、材质贴图的简易光线追踪器。

1.3 Scope of the project

建立一个动态的三维小球模型，并为小球添加上材质纹理、阴影处理，加上天空盒作为周边环境。

在静态小球上实现光线追踪算法，使之能够映射周围环境。

1.4 Achievement

我们小组最终做出的程序有两个：第一个是动态的三维弹球模型，即一个小球在地板上上下反弹，并在动态的小球模型上实现了材质纹理映射和阴影处理，程序的主体使用了 OpenGL 完成；第二个是程序是在天空盒中对静态的小球模型进行光线追踪，项目主体使用了 OpenGL Shading Language(GLSL)完成。

二、弹球建模及动态反弹

2.1 OpenGL 中的模型视图变换

模型变换，只涉及仿射变换（互相平行的两条直线，经过变换后仍然能保持平行），这里介绍其中的 3 个变换：位移、旋转和缩放。三维空间上的，位移用矩阵 $\boldsymbol{T}$ 表示，则$(x,y,z,1)\cdot\boldsymbol{T}=(x+a,y+b,z+c,1)$，相当于把齐次坐标(x,y,z,1)移动(a,b,c)的偏移量。

$$\boldsymbol{T}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\a&b&c&1\end{pmatrix}$$

在三维空间上，绕着 x 轴、y 轴、z 轴的旋转 4×4 的矩阵 $\boldsymbol{R}_x(\theta),\boldsymbol{R}_y(\theta),\boldsymbol{R}_z(\theta)$ 表示，有：

$$\boldsymbol{R}_x(\theta)=\begin{pmatrix}1&0&0&0\\0&\cos\theta&\sin\theta&0\\0&-\sin\theta&\cos\theta&0\\0&0&0&1\end{pmatrix},\quad \boldsymbol{R}_y(\theta)=\begin{pmatrix}\cos\theta&0&-\sin\theta&0\\0&1&0&0\\\sin\theta&0&\cos\theta&0\\0&0&0&1\end{pmatrix},$$

$$\boldsymbol{R}_z(\theta)=\begin{pmatrix}\cos\theta&\sin\theta&0&0\\-\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

其中，沿着轴相反的方向观察，θ 表示逆时针旋转的角度。

在三维空间上，对模型的缩放变换的矩阵用 ***S*** 表示，分别将坐标 x 轴、y 轴、z 轴上的分量分别缩放至原先的 a,b,c 倍。

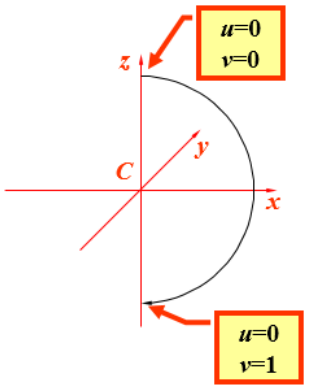
$$\boldsymbol{S}=\begin{pmatrix}a&0&0&0\\0&b&0&0\\0&0&c&0\\0&0&0&1\end{pmatrix}$$

这 3 个变换对应 OpenGL 中的三个接口为：glTranslate*(), glRotate*(), glScale*().

2.2 OpenGL 中小球的绘制

2.2.1 球面参数方程

球面的参数曲线可以用球坐标表示，引入参数 u,v，其中 v 是球面点与原点的连线与 z 轴正向的夹角，u 表示连线在 xy 平面的投影与 x 轴正向的夹角，如下图所示：



则球面参数方程可以表示为：

$$\begin{cases} x = cx + r \times \sin(\pi v) \times \cos(2\pi u) \\ y = cy + r \times \sin(\pi v) \times \sin(2\pi u) \\ z = cz + r \times \cos(\pi v) \end{cases}$$

2.2.2 球面法向量

已知球面的参数方程以后，很容易求得给定点的法向量，分别对 u 和 v 方向求偏导数，然后对两个所得向量进行叉积即可：

$$\text{法向量为} \begin{pmatrix} x_u \\ y_u \\ z_u \end{pmatrix} \times \begin{pmatrix} x_v \\ y_v \\ z_v \end{pmatrix} = \begin{pmatrix} y_u z_v - y_v z_u \\ x_v z_u - x_u z_v \\ x_u y_v - y_u x_v \end{pmatrix} = \begin{pmatrix} y_u z_v \\ -x_u z_v \\ x_u y_v - y_u x_v \end{pmatrix}$$

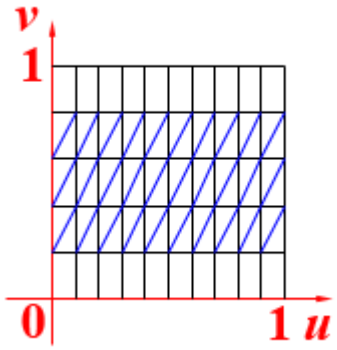
$$= A \begin{pmatrix} \sin(\pi v) \times \cos(2\pi u) \\ \sin(\pi v) \times \sin(2\pi u) \\ \cos(\pi v) \end{pmatrix}$$

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{vmatrix}$$

其中，$A=-2\pi^2 r^2 \times \sin(\pi v)$

2.2.3 实现细节

已知参数方程以后，需要进行离散，分别设定 u 和 v 的步长：ustep、vstep。然后通过不同的 u 和 v，求得坐标系中点的实际坐标(x,y,z)，在实现中有一点需要注意的是：



u=0 与 u=v 这两条线上点的是球体的两个上下极点，所以进行渲染时需要区分，其中如图中间段的离散点可以按照四边形进行渲染，而上下两段则需要按照三角形进行渲染。

在 glut 中提供了相应的函数去做这些事，便直接利用了。

2.3 小球的运动

先根据小球的坐标对小球进行模型变换，变换到相应的坐标，对小球进行绘制，然后计算出小球的下一个坐标，实反弹还有运动。

三、材质纹理及周边环境

3.1 总体思路

小球的材质使用光照+glMaterial*完成，作为试验，为小球准备了五种材质；地板的材质具有半透明效果，其 alpha 值为 0.5，同时为了让它的颜色与小球倒影的颜色混合，在绘制地板之前需要开启 GL_BLEND；光源点用一个白色的小球表示，因为是光源本身，所以其颜色不受光线的影响，其 emission 是{1.0, 1.0, 1.0}。

在 display 中，先绘制出小球倒影和光源倒影，再绘制小球和光源，最后绘制地板；在这里地板最后绘制，因为若地板在倒影和实物之间绘制，则实光源不能照亮，影响场景的真实性。

天空盒的添加使用了 2D 纹理映射的方法，解析六张位图生成六张绑定到 GL_TEXTURE_CUBE_MAP 目标的 2D 纹理。开启 2D 纹理使用，在周围绘制六个正方形，大小足够可以包围模型即可。

3.2 数据结构与函数

（1）为每一个材质准备 m_amb，m_diff，m_pec 三个数组，分别表示对环境光、散射光和镜面光各三个通道的反射程度

```
static float m_amb[][4] = {
        {0.250000, 0.250000, 0.250000, 1.0},//铬
        {0.329412, 0.223529, 0.027451, 1.0},//黄铜
        {0.212500, 0.127500, 0.054000, 1.0},//青铜
        {0.053750, 0.050000, 0.066250, 1.0},//黑曜石
        {0.0, 1.0, 1.0, 1.0}};
static float m_diff[][4] = {
        {0.400000, 0.400000, 0.400000, 1.0},
        {0.780392, 0.568627, 0.113725, 1.0},
        {0.714000, 0.428400, 0.181440, 1.0},
        {0.182759, 0.170000, 0.225250, 1.0},
        {1.0, 0.0, 1.0, 0.6}};
static float m_spec[][4] = {
        {0.774597, 0.774597, 0.774597, 1.0},
        {0.992157, 0.941176, 0.807843, 1.0},
        {0.393548, 0.271906, 0.166721, 1.0},
```

```
{0.332741, 0.328634, 0.346435, 1.0},
{1.0, 1.0, 0.0, 1.0}};
```

（2）为每一个材质准备一个 m_shinn 一维数组，其中唯一元素的大小决定了高光的程度：

```
static float m_shinn[][1] = { {76.800003}, {27.8974}, {55.6}, {38.4}, {100.0} };
```

用一个全局变量标识选择的材质：`static int mat = 0;`

用一个全局变量表示光源的位置：`static float tht = 0;`

（3）绘制小球和光源，在其中使用固定的坐标：`void drawScene()`

绘制地板：`void drawGround()`

（4）读取键盘输入，若按空格，可以暂停/开始小球的跳动，按数字键可以选择材质：`void KeyboardFunc(unsigned char key, int x, int y)`

读取特殊按键输入，用左右方向键可以使光源在地板附近做圆周运动：`void SpecialKeyFunc(int key, int x, int y)`

（5）绘制天空盒的类：

```
class SkyBox
{
public:
    //构造函数
    SkyBox();
    //绘制函数
    void draw();
    //初始化函数
    void init();
protected:
    //将bmp图像文件转化为2D纹理
    GLuint createTexture2DFromBMP(const char *bmpPath);
    //载入bmp图像文件内容
    unsigned char * loadFileContent(const char *path, int &filesize);
    //解码bmp文件内容
    unsigned char *decodeBMP(unsigned char*bmpFileData, int&width, int&height);
    //具体生成纹理
    GLuint createTexture2D(unsigned char *piexlData, int width, int height, GLenum
type);
private:
    GLuint m_textures[6]; //存储六个面的纹理：顺序为，前后左右上下
};
```

四、光线追踪框架设计

4.1. 总体设计

使用 OpenGL Shading Language（GLSL）结合 Qt 5 的前端实现在一个天空盒中对一个球体进行光线追踪。GLSL 是 OpenGL 中着色编程的语言，它代替了渲染管线中的一部分，使渲染管线中不同层次具有可编程性。GLSL 语言的代码由两部分组成，Vertex Shader（顶点着色器）和 Fragment Shader（片元着色器），顶点着色器的工作是对每一个送入流水线的顶点进行处理，并负责生成所有后续阶段需要的信息。最低要求是生成裁剪空间坐标值。片元着色器的工作是为当前光栅化的像素提供颜色值，每个片元都将调用一次片元着色器来获取颜色值。

在 Qt 5 中，需要实现 initializeGL、paintGL 和 resizeGL 三个函数：在 initializeGL 函数中，加载 GLSL 的顶点着色器和片元着色器，并对球体的参数进行初始化。同时读取天空盒图片，加载天空盒；paintGL 函数会被反复调用，用来实时绘制场景；resizeGL 函数的作用是当窗口大小改变时，调整界面坐标显示高度和宽度。此外，在 Qt 5 的前端设计中，还加入了一些鼠标、键盘事件函数，与用户实现实时交互。

4.2. 数据结构与函数设计

4.2.1 片元着色器（glsl.frag 文件）相关数据结构与函数、功能

```
struct material {
    float phong_factor; // between 0 and 1
    vec3 ambiant; //环境光反射系数
    vec3 diffuse; //漫反射系数

    // 若 eta == 0  无折射
    float eta; // η=n2/n1，折射率 n2 材质相对折射率 n1 材质的相对折射率
};
```

```
struct sphere {
    vec3 center; //球心
    float radius; //半径
    material mat; //球体材质
};
struct ray {
    float power; //光线强度
    vec3 origin; //光源坐标
    vec3 direction; //光线的方向
};
```

```
struct plane {
    vec3 point; //面左下方顶点坐标
    vec3 normal; //法向量
    vec3 width; //宽度向量
    vec3 height; //高度向量
    //width,height,normal 满足右手系
    material mat; //面材质
};
```

函数	功能
bool line_sphere_intersection(in vec3 origin, in vec3 direction, in sphere s, out float dist)	求解光线与球体相交。若相交则返回 true，并通过 dist 返回光源到交点的距离。
bool line_plane_intersection(in vec3 origin, in vec3 direction, in plane p, out float dist)	求解光线与方形有界平面相交。若相交则返回 true，并通过 dist 返回光源到交点的距离。
bool next_intersection(inout vec3 origin, in vec3 direction, out vec3 normal, out material mat)	遍历所有球体和面，求光线与物体的下一个交点。若与物体相交则返回 true，并通过 normal 和 mat 返回交点处的法向量和材质。
bool light_intersection(in vec3 origin, in vec3 direction, out vec3 normal, out material mat)	调用 next_intersection()计算某点到光源之间是否有物体。若有则产生阴影，若无则利用 phong 反射模型计算该点的颜色。
float refraction(in vec3 v, in vec3 n, in float cosi, out vec3 t, in float eta)	根据入射光线、法线、入射角和相对折射率，计算反射率和折射光线。

4.2.2 Qt 前端 RayWindow 类中的函数及功能

函数	功能
void RayWindow::***initializeGL***()	加载顶点着色器和片元着色器，设定物体的参数，加载天空盒
void RayWindow::***paintGL***()	反复调用，用来实时绘制场景
void RayWindow::***resizeGL***(int w, int h)	当窗口大小改变时，调整界面坐标显示高度和宽度
void RayWindow ::***mousePressEvent***(QMouseEvent* ev)	点击鼠标事件函数。点击左键切换内外窗口操作，点击右键切换全屏
void RayWindow::***mouseMoveEvent***(QMouseEvent* ev)	鼠标移动事件函数。根据鼠标的移动，在不改变远近的情况下，切换观察点的位置。
void RayWindow::***wheelEvent***(QWheelEvent* ev)	鼠标滚轴事件函数，滚动鼠标滚轴确定观察点离物体的远近。

void RayWindow::***resize Event***(QResizeEvent* ev)	继承 GLWindow::*resizeEvent*(ev)函数，处理窗口大小的改变
void RayWindow::***move Event***(QMoveEvent* ev)	继承 GLWindow::*moveEvent*(ev)函数，处理窗口的移动
void RayWindow::***hide Event***(QHideEvent* ev)	继承 GLWindow::*hideEvent*(ev)函数，处理窗口的隐藏
void RayWindow::***keypress Event***(QKeyEvent* ev)	控制键盘按键按下事件的函数
void RayWindow::***keyRelease Event***(QKeyEvent* ev)	控制键盘按键完毕事件的函数
void RayWindow::***timer Event***(QTimerEvent *ev)	计时器，控制键盘操作的速度。
void RayWindow::**set Track**(bool track)	控制窗口切换。点击鼠标左键来确定鼠标操作窗口内还是窗口外，并完成每次窗口内操作的初始化动作。

五、光线追踪理论模型与代码分析

由光源发出的光到达景物表面后，产生反射和折射，简单光照明模型和简单光透射模型模拟了这两种现象。在简单光照明模型中，反射光被分为理想漫反射光和镜面反射光；在简单光透射模型中，透射光被分为理想漫透射光和规则透射光。由光源发出的光称为直接光，景物对直接光的反射或折射称为直接反射和直接折射。相对地，将景物表面间反射和折射引起的光称为间接光，由此产生间接反射、间接折射。这些是光线在景物之间的传播方式，也是光线跟踪算法的基础。

最基本的光线跟踪算法是跟踪镜面反射和折射。从光源发出的光遇到景物的表面，发生反射和折射，光就改变方向，沿着反射方向和折射方向继续前进，直到遇到新的景物。但是光源发出光线，经反射与折射，只有很少的部分可以进入人的眼睛。因此，实际的光线跟踪算法的跟踪方向与光传播的方向是相反的，而是所谓的“视线跟踪”：由视点向像素(x,y)发出一根射线，与第一个景物相交后，在其反射与折射方向上进行跟踪。[5]

5.1. 光线追踪基本模型[5]

如图五-1 所示，设场景中有一点光源 L，两个透明的球体 O_1 与 O_2，一个不透明的景物 O_3。首先，从视点 V 出发经过屏幕一个像素点的视线 E 传播到达球体 O_1，其交点为 P_1。

（1）从 P_1 向光源 L 作一条光源线 S_1，发现其间没有遮挡的景物，用局部光照明模型计算光源对 P_1 在其视线 E 的方向上的光强，作为该点的局部光强。

（2）P_1 点处的反射光线 R_1，它对 P_1 点的光强有贡献，对其跟踪：

在反射光线 R_1 方向上，没有再与其他景物相交，因此设该方向的光强为零，并结束此光线方向的跟踪。

（3）P_1 点处的折射光线 T_1，与球体 O_1 相交于点 P_2，它对 P_1 点的光强也有贡献，对其跟踪：

① 折射光线 T_1 作为球体 O_1 的光线在球体 O_1 内部传播，由于该点在球体内部，假设它的局部光强为零。

② 折射光线 T_1 产生了反射光线 R_2。在反射光线 R_2 方向，可以继续递归跟踪下去计算它的光强，此处省略。

③ 折射光线 T_1 产生了折射光线 T_2。继续对折射光线 T_2 进行跟踪：

T_2 与景物 O_3 交于点 P_3，作 P_3 与光源 L 的光源线 S_3，没有景物遮挡，计算该处的局部光强；由于景物 O_3 是非透明的，可以继续跟踪 T_2 在景物 O_3 的反射光线 R_3 方向的光强，结合局部光强，得到 P_3 处的光强（反射光线 R_3 的跟踪与前面的过程类似，算法可以递归地进行下去）。

重复上面的过程，就可以得到屏幕上一个像素点的光强，最后得到它相应的颜色值。

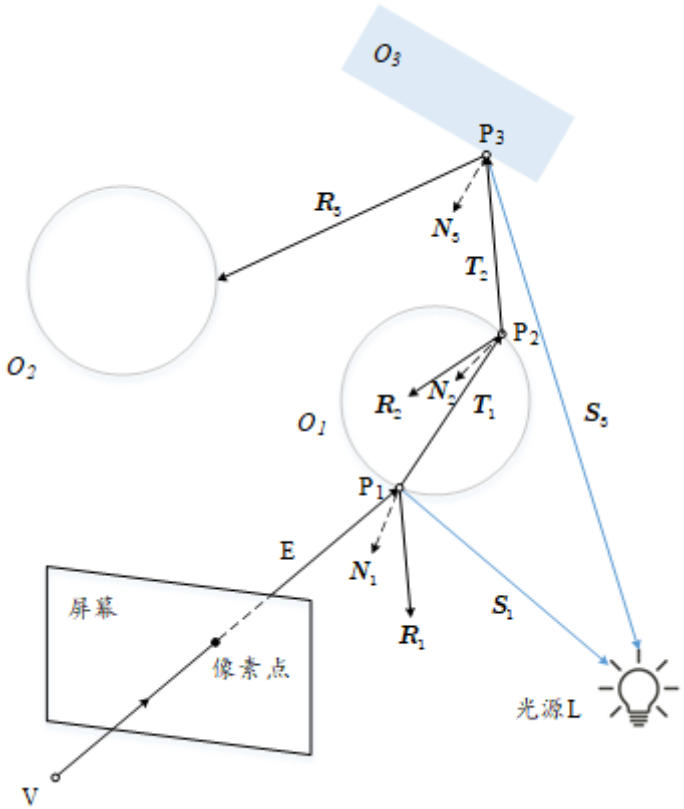


图 五-1. 光线追踪基本模型

5.2. 递归光线追踪算法伪码

注：最终代码以光线队列数据结构取代递归，详见第 5.7.节

```
选择投影中心以及视图平面上的窗口;
for(图像的每条扫描线) {
    for (扫描线上的每个像素pixel) {
        确定从投影中心穿过pixel的光线ray;
        pixel = RayTracing(ray,1);
    }
}

/* ---------- ---------- ---------- ----------
    计算与景物的交点并计算最近交点的光照值
---------- ---------- ---------- ---------- */
RayColor RayTracing(RayRay ray, int depth)
{
    确定光线与某个景物object的最近交点intersection;
    if(命中景物) {
        计算交点处景物表面的法向normal;
        return RayShading(object, ray, intersection,
        normal , depth);    // depth是光栅树的当前深度
    }
    else
        return BACKGROUND_VALUE;
}// RayTracing

/* ---------- ---------- ---------- ---------- ----------
    计算景物上一点的光照值，跟踪光源线、反射光和折射光线
---------- ---------- ---------- ---------- ---------- */
RayColor RayShading(
    RayObject objct,    // 相交的景物
    RayRay ray,         // 入射光线
    RayPoint point,     // 交点
    RayNormal normal,   // 交点处的法向
    int depth)          // 所处光线树的深度
```

```
{
    RayColor color;      // 光线的颜色
    RayRay rRay,tRay,sRay;  // 反射光线、折射光线、光源线
    RayColor rColor, tColor;     // 反射光线、折射光线颜色

    color = 环境光项;
    for(每个光源) {
        sRay=从交点到光源的光线;
        if (法向到光源方向的点积为正) {
            计算被不透明景物和透明景物遮挡的光的量,
            并用该值乘漫射项和镜面反射项,
            然后加到color;
        }
    }   //每个光源

    if(depth<maxDepth) {     // 如果深度太深则返回
        if (景物是反射体) {
            rRay = 从交点向反射方向发射的光线;
            rColor = RayTracing(rRay ,depth+1);
            color = color + Kr*rColor;
                // Kr为反射系数
        }   //反射体
        if (景物是透明体) {
            tRay = 从交点向折射方向发射的光线;
            if(没有发生全反射) {
                tColor = RayTracing(tRay, depth+1);
                color = color + Kt*rColor;
                // Kt为折射系数
            }
        }   //透明体
    }

    return color;   // 返回光线的颜色值RayShading

}
```

5.3. 光线与球体相交

如图五-2 所示，设光源坐标为$\boldsymbol{O}$，光线方向向量为$\boldsymbol{D}$，球心坐标为$\boldsymbol{C}$，半径为R，因此光线上每个点可表示为$\boldsymbol{O}+t\boldsymbol{D}$，其中$t$为参数，表示光源至交点的距离。对于光线与球体的交点，一定满足：

$$|\boldsymbol{O}+t\boldsymbol{D}-\boldsymbol{C}|=R$$

将上式两边同时平方，展开、移项后可得关于参数t的一元二次方程：

$$\boldsymbol{D}^2t^2+2\boldsymbol{D}(\boldsymbol{O}-\boldsymbol{C})t+|\boldsymbol{O}-\boldsymbol{C}|^2-R^2=0$$

上述一元二次方程的判别式$\Delta=b^2-4ac$，其中：

$$a=\boldsymbol{D}^2,\quad b=2\boldsymbol{D}(\boldsymbol{O}-\boldsymbol{C}),\quad c=|\boldsymbol{O}-\boldsymbol{C}|^2-R^2$$

通过计算判别式Δ，判断光线与球体的相交情况：若$\Delta<0$，则光线与球体不相交；若$\Delta=0$，则光线与球体恰有一个交点，即光线与球体相切；若$\Delta>0$，表示光线与球体有两个交点，此时保留与光源$\boldsymbol{O}$更近的交点（即需同时满足$t>0$且t取较小值），再分以下两类情况进行讨论：

在$\Delta>0$的前提下，若$c<0$，则光源在球内，交点分布于光源两侧，为满足

条件$t>0$，则取$t=\dfrac{-b+\sqrt{\Delta}}{2a}$；若$c>0$，则光源在球外，为满足条件$t$取较小值，则取$t=\dfrac{-b-\sqrt{\Delta}}{2a}$。

具体代码实现见 glsl.frag 文件中 bool line_sphere_intersection(in vec3 origin, in vec3 direction, in sphere s, out float dist) 函数。

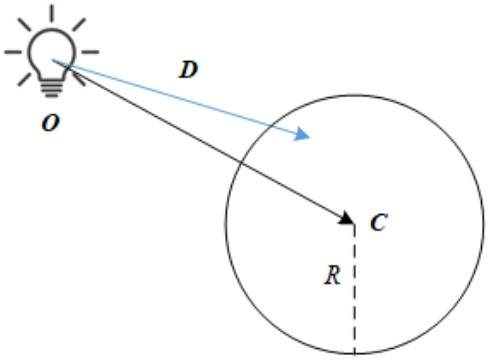


图 五-2. 光线与球体相交模型

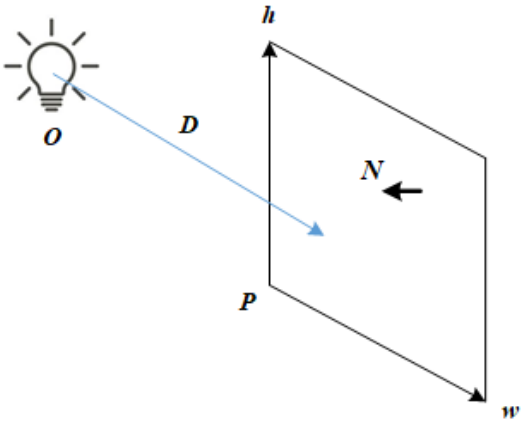


图 五-3. 光线与有界平面相交模型

5.4. 光线与有界平面相交

如图五-3 所示，令对象面为面σ，对象面所在平面为平面Π，设光源坐标为$\boldsymbol{O}$，面σ左下方顶点坐标为$\boldsymbol{P}$，光线方向向量为$\boldsymbol{D}$，平面法向量为$\boldsymbol{N}$，面σ高度向量为$\boldsymbol{h}$（模为高度），面σ宽度向量为$\boldsymbol{w}$（模为宽度），且$\boldsymbol{N}=\boldsymbol{w}\times\boldsymbol{h}$。

光线上每个点可表示为$\boldsymbol{O}+t\boldsymbol{D}$，其中$t$为参数，表示光源至交点的距离。令$\boldsymbol{L}(t)=\boldsymbol{O}+t\boldsymbol{D}$表示以$\boldsymbol{O}$点为光源、与方向向量$\boldsymbol{D}$平行的光线（$t$变化时可表示光线上的任意一点），以平面方程$\boldsymbol{N}\cdot\boldsymbol{L}(t)+d=0$表示以$\boldsymbol{N}$为法向量、包含点$\boldsymbol{L}(t)$的平面$\Pi$，其中$d$表示平面到原点的符号距离为$d$.

由对象平面条件：平面左下方顶点坐标为$\boldsymbol{P}$，平面法向量为$\boldsymbol{N}$，建立平面Π的平面方程：

$$\boldsymbol{N}\cdot\boldsymbol{P}+d=0$$

••• ①

显然，若$(\boldsymbol{w}\times\boldsymbol{h})\cdot\boldsymbol{D}=0$，意味着平面$\Pi$法向量$\boldsymbol{N}$与光线方向向量$\boldsymbol{D}$垂直，

即光线与平面Π平行而没有交点；若$(\boldsymbol{w}\times\boldsymbol{h})\cdot\boldsymbol{D}\neq 0$，则光线与面$\sigma$的所在平面$\Pi$相交，但不一定与面有交点。对于光线与平面$\Pi$（非面$\sigma$）的交点，一定满足：

$$\boldsymbol{N}\cdot\boldsymbol{L}(t)+d=0$$

亦即，$\boldsymbol{N}\cdot(\boldsymbol{O}+t\boldsymbol{D})+d=0$

$\cdots$ ②

联立①②两式得：

$$\begin{cases}\boldsymbol{N}\cdot\boldsymbol{P}+d=0\\ \boldsymbol{N}\cdot(\boldsymbol{O}+t\boldsymbol{D})+d=0\end{cases}$$

两式相减、移项得：

$$\boldsymbol{N}\cdot(\boldsymbol{P}-\boldsymbol{O})=\boldsymbol{N}\cdot t\boldsymbol{D}$$

即：

$$t=\frac{\boldsymbol{N}\cdot(\boldsymbol{P}-\boldsymbol{O})}{\boldsymbol{N}\cdot\boldsymbol{D}}=\frac{(\boldsymbol{w}\times\boldsymbol{h})\cdot(\boldsymbol{P}-\boldsymbol{O})}{(\boldsymbol{w}\times\boldsymbol{h})\cdot\boldsymbol{D}}$$

至此，已求得：若光线与面σ的所在平面Π相交，光源至与平面Π交点的距离$t=\frac{(\boldsymbol{w}\times\boldsymbol{h})\cdot(\boldsymbol{P}-\boldsymbol{O})}{(\boldsymbol{w}\times\boldsymbol{h})\cdot\boldsymbol{D}}$.

若光线与面σ相交，还应满足以下条件：

$$\begin{cases}\boldsymbol{P}\le\boldsymbol{O}+t\boldsymbol{D}\le\boldsymbol{P}+\boldsymbol{w}\ \text{（宽度约束）}\\ \boldsymbol{P}\le\boldsymbol{O}+t\boldsymbol{D}\le\boldsymbol{P}+\boldsymbol{h}\ \text{（高度约束）}\end{cases}$$

两边同时减去$\boldsymbol{P}$得：

$$\begin{cases}\boldsymbol{0}\le(\boldsymbol{O}-\boldsymbol{P})+t\boldsymbol{D}\le\boldsymbol{w}\ \text{（宽度约束）}\\ \boldsymbol{0}\le(\boldsymbol{O}-\boldsymbol{P})+t\boldsymbol{D}\le\boldsymbol{h}\ \text{（高度约束）}\end{cases}$$

将$t=\frac{(\boldsymbol{w}\times\boldsymbol{h})\cdot(\boldsymbol{P}-\boldsymbol{O})}{(\boldsymbol{w}\times\boldsymbol{h})\cdot\boldsymbol{D}}$代入，两式两边分别同时点乘$\frac{\boldsymbol{w}}{|\boldsymbol{w}|^2}$、$\frac{\boldsymbol{h}}{|\boldsymbol{h}|^2}$得：

$$\begin{cases}0\le\frac{(\boldsymbol{P}-\boldsymbol{O})\cdot(\boldsymbol{D}\times\boldsymbol{h})}{(\boldsymbol{w}\times\boldsymbol{h})\cdot\boldsymbol{D}}\le 1\ \text{（宽度约束）}\\ 0\le\frac{(\boldsymbol{P}-\boldsymbol{O})\cdot(\boldsymbol{D}\times\boldsymbol{w})}{(\boldsymbol{w}\times\boldsymbol{h})\cdot\boldsymbol{D}}\le 1\ \text{（高度约束）}\end{cases}$$

注：代码中为减少中间变量使得运算方便，利用以下公式对点乘与叉乘的顺序进

行调整：

$$(\boldsymbol{a}\times\boldsymbol{b})\cdot\boldsymbol{c}=(\boldsymbol{b}\times\boldsymbol{c})\cdot\boldsymbol{a}=(\boldsymbol{c}\times\boldsymbol{a})\cdot\boldsymbol{b}=\boldsymbol{a}\cdot(\boldsymbol{b}\times\boldsymbol{c})=\boldsymbol{b}\cdot(\boldsymbol{c}\times\boldsymbol{a})=\boldsymbol{c}\cdot(\boldsymbol{a}\times\boldsymbol{b})$$

$$(\boldsymbol{a}\times\boldsymbol{c})\cdot\boldsymbol{b}=(\boldsymbol{b}\times\boldsymbol{a})\cdot\boldsymbol{c}=(\boldsymbol{c}\times\boldsymbol{b})\cdot\boldsymbol{a}=\boldsymbol{a}\cdot(\boldsymbol{c}\times\boldsymbol{b})=\boldsymbol{b}\cdot(\boldsymbol{a}\times\boldsymbol{c})=\boldsymbol{c}\cdot(\boldsymbol{b}\times\boldsymbol{a})$$

上式各值与下式各值互为相反数关系.

具体代码实现见 glsl.frag 文件中 bool line_plane_intersection(in vec3 origin, in vec3 direction, in plane p, out float dist) 函数。

5.5. 光的反射、折射

根据光线折射的 Snell 定律，折射光 T 与入射光 I 位于平面法线 N 的两侧，入射角 θ_I 与折射角 θ_T 的关系符合：

$$\frac{\sin\theta_I}{\sin\theta_T}=\frac{n_I}{n_T}=\eta$$

••• ①

其中，n_I 为第一介质的折射率，n_T 为第二介质的折射率，η 为第二介质相对于第一介质的相对折射率。

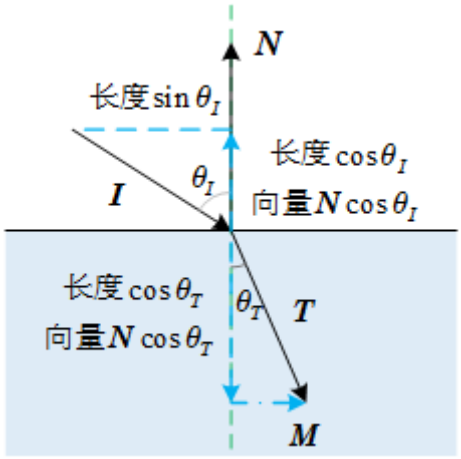


图 五-4. 折射光向量计算模型

如图五-4 所示，假设单位入射光向量为 $\boldsymbol{I}$，单位折射光向量为 $\boldsymbol{T}$，界面的单位法向量为 $\boldsymbol{N}$，由向量点积性质，经三角函数关系变换，得：

$$\cos\theta_I=-\boldsymbol{N}\cdot\boldsymbol{I}$$

$$\cos\theta_T=\sqrt{1-\frac{1-\cos^2\theta_I}{\eta^2}}$$

由图示向量几何关系，根据式①Snell 定律得：

$$\boldsymbol{M}=\frac{\boldsymbol{I}+\boldsymbol{N}\cos\theta_I}{\eta}$$

••• ②

因此，折射向量$\boldsymbol{T}$满足：

$$\boldsymbol{T}=-\boldsymbol{N}\cos\theta_T+\boldsymbol{M}$$

••• ③

式②代入式③，合并同类项得：

$$\boldsymbol{T}=\frac{1}{\eta}\boldsymbol{I}+\left(\frac{\cos\theta_I}{\eta}-\cos\theta_T\right)\boldsymbol{N}$$

••• 输出项 out vec3 t

由菲涅尔（Fresnel）公式可得：

$$r_s=\frac{\cos\theta_I-\eta\cos\theta_T}{\cos\theta_I+\eta\cos\theta_T}$$

$$r_p=\frac{\eta\cos\theta_I-\cos\theta_T}{\eta\cos\theta_I+\cos\theta_T}$$

因此，反射率$k_r=\frac{r_s^2+r_p^2}{2}$，折射率$k_t=1-k_r$。

具体代码实现见 glsl.frag 文件中 float refraction(in vec3 v, in vec3 n, in float cosi, out vec3 t, in float eta)函数。

5.6. Phong 模型效果对照

Phong 光照模型是真实图形学中提出的第一个有影响的光照明模型。该模型只考虑物体对直接光照的反射作用，将环境光视作常量，不考虑物体之间相互的反射光，物体间的反射光只用环境光表示。镜面反射的颜色也只是光源的颜色，与物体的材料无关。

本处采用多光源的漫反射和镜面反射模型，在多个点光源的情况下，任意表面点上产生的光照效果计算公式为：

$$I=k_aI_a+\sum_{l=1}^{n}I_l[k_d(\boldsymbol{N}\cdot\boldsymbol{L})+k_s(\boldsymbol{N}\cdot\boldsymbol{H})^{n_s}]$$

其中k_a为环境光反射系数，k_d为漫反射系数，k_s为镜面反射系数，I_a为环境光强度，I_l为某一光源强度，n_s为镜面反射参数，$\boldsymbol{N}$为表面单位法向量，$\boldsymbol{L}$为指向点光源的单位向量，$\boldsymbol{H}$为半角向量。

加入 Phong 模型实际上只是为了与光线追踪效果产生对比。由于目的在于对比，因此对上式作了进一步简化处理。最终计算当前像素的颜色时，套用上述公式求得光强，再乘以漫射项（入射光辐射能 power）和镜面反射项（材质相关 phong_factor），然后加到 color 变量（此步骤请见 5.2.节伪码描述）。具体代码实现见 glsl.frag 文件中 main 函数 phong 模型部分。

5.7. 光线追踪实现算法

建立一个最大大小为 qmax 的光线队列（用循环队列实现），用 color 来记录当前像素的颜色。首先将第一条光线加入到队列中。

取出队首的光线，调用 next_intersection()函数判断其是否与物体相交。如果未与任何物体相交，则将（该光线的强度乘以天空盒在该光线方向上的颜色）加到 color 变量。

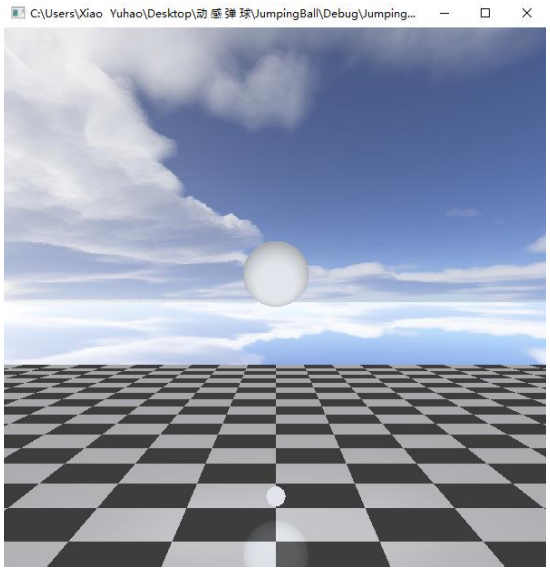
否则，由于 phong_factor 与镜面反射系数 k_s 类似，阈值限定在 0 与 1 之间。若 phong_factor>0，则寻找交点到光源的路径上是否存在物体，如果存在物体则该交点为阴影，如果不存在物体则利用 phong 反射模型计算该交点的颜色，然后加到 color 变量；若还满足 phong_factor<1，则还需计算折射，并在光线队列中加入当前反射光线与折射光线。

对于反射与折射相关计算，先假设物体表面仅产生反射效果，由入射光线方向与法向量求得反射光线方向。在调用 refraction()函数计算反射率 k_r 和折射光线方向后，若 $k_r<1$，说明同时存在折射效果，则新建一条折射光线，加入到光线队列中。否则认为只存在反射效果。无论 k_r 取何值，都需要新建一条反射光线，并加入到光线队列中。

重复循环执行上述过程，直至光线队列队首指针与队尾指针重合，即光线队列为空追踪完毕时，算法结束。具体代码实现详见 glsl.frag 文件。

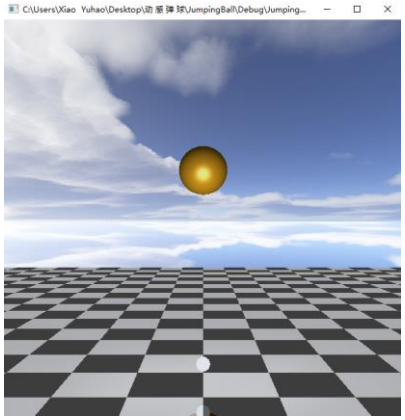
六、程序结果展示

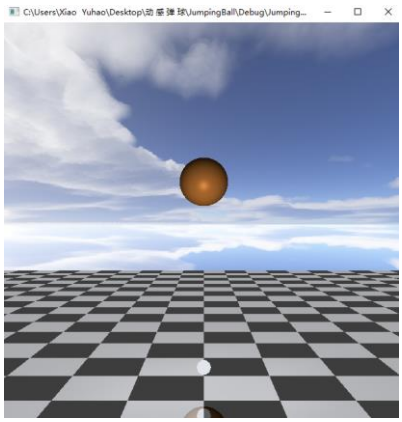
6.1. 小球动态反弹

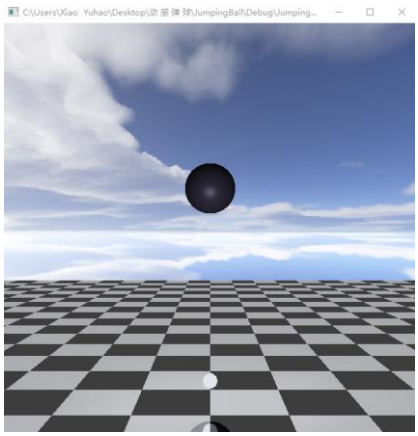


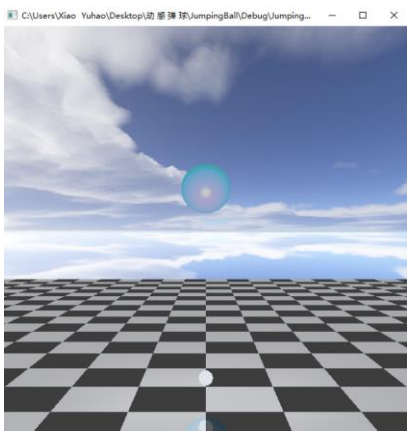
小球可以在地板上进行上下反弹

6.2. 小球材质纹理



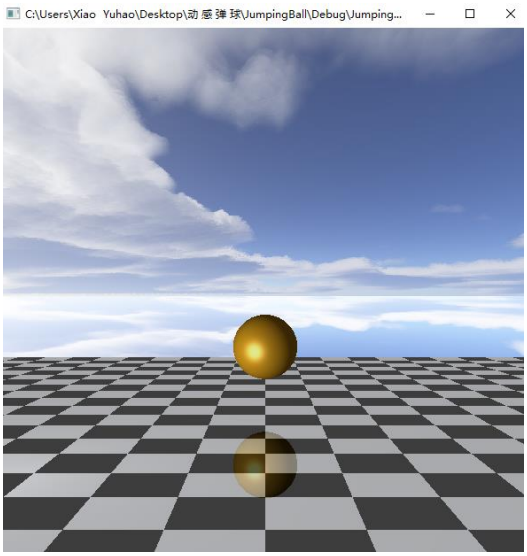






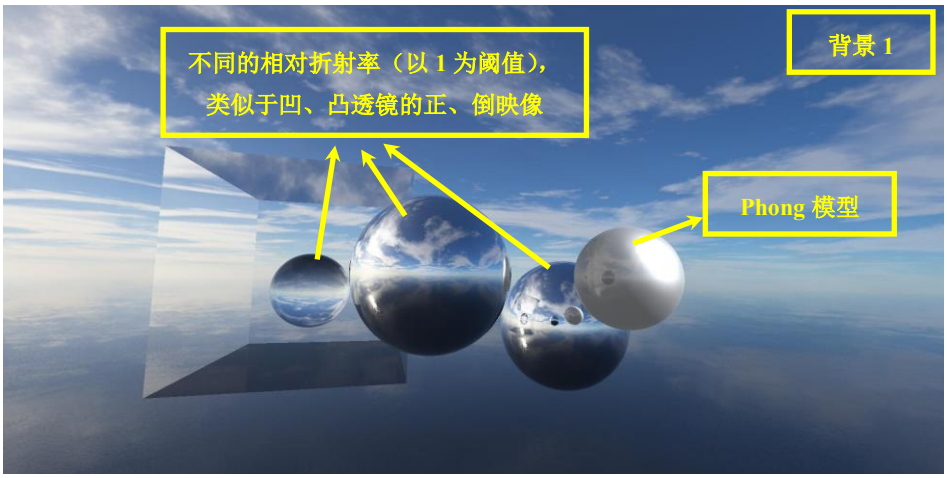
如图所示，我们可以通过键盘按键的方式改变小球的材质纹理，使之反射不同的光线。

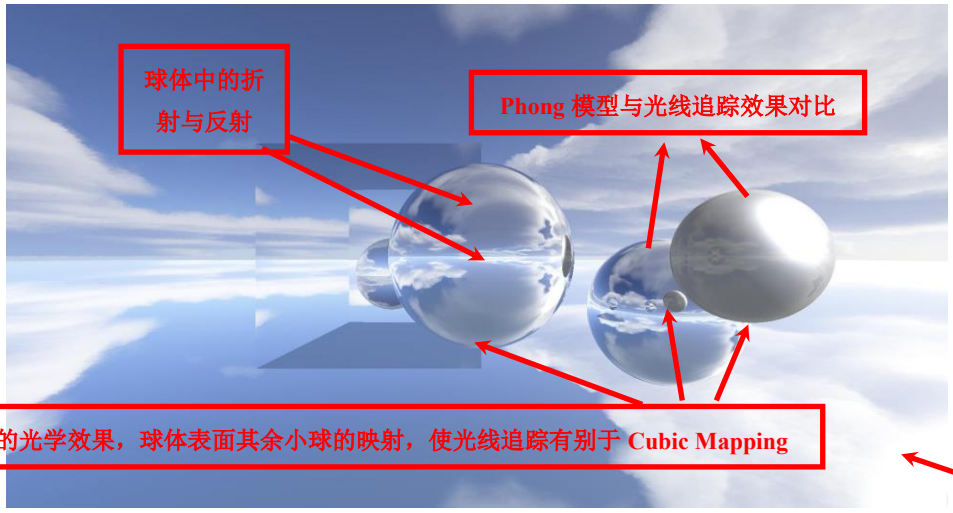
6.3. 阴影处理

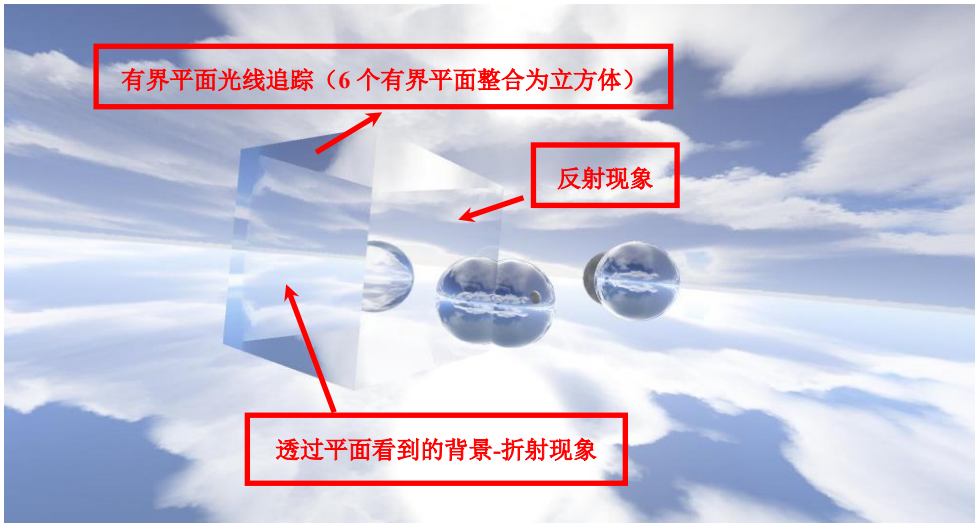


如图所示，小球在反弹的过程中，地板上的阴影也会随之变化。

6.4. 光线追踪







在静态的小球上实现光线追踪算法（同时考虑反射与折射）。为了使光线追踪更加形象，我们添加了几个不同材质的小球，以便观察光线在它们之间反射时产生的效果；除了研究球面的光线追踪之外，我们还建立了一个基于有界平面的正方体模型来研究平面光线追踪算法。

七、心得体会

本次“三维弹球”的大作业设计中，我们小组探索了 OpenGL 光照-材质系统；学习了制作天空盒的基本原理，了解了位图的基本格式以及如何将其转化成 2D 纹理用于 OpenGL 绘制；通过 GLSL 语言结合 Qt 5 的前端，实现了在天空盒中对多个透明球体进行光线追踪；将空间解析几何、物理光学定律与图形学知识完美结合，在学科交叉中深刻体会到了图形学的艺术内涵。

整个项目进展过程中，我们将计算机图形学理论运用到实践中，巩固掌握了计算机图形学的基本知识，拓展延伸了逻辑创造思维，锻炼了团队合作意识与合作精神。短短几周之内我们收获了许多，在图形学的课程经历中留下了难忘的回忆。

八、参考文献

[1] 赫恩（Hearn, D.），巴克（Baker, M. P.），卡里瑟斯（Carithers, W. R.）著，蔡士杰，杨若瑜译. 计算机图形学[M]. 4 版. 北京：电子工业出版社，2014.

[2] 施莱尔（Shreiner, D.）等著，王锐等译. OpenGL 编程指南[M]. 8 版. 北京：机械工业出版社，2014.

[3] 赖特（Wright, R. S.），利普恰克（Lipchak, B.），黑内尔（Haemel, N.）著，张琪，付飞译. OpenGL 超级宝典[M]. 4 版. 北京：人民邮电出版社，2010.
[4] 伦吉尔（Lengyel, E.）著，詹海生译. 3D 游戏与计算机图形学中的数学方法[M]. 3 版. 北京：清华大学出版社，2016.
[5] 何援军. 计算机图形学[M]. 3 版. 北京：机械工业出版社，2016.
[6] CSDN. Monte-Carlo Ray Tracing System (一)原理以及架构设计[EB/OL]. https://blog.csdn.net/admintan/article/details/71598413，2017.
[7] Scratchapixel. A Minimal Ray-Tracer: Rendering Simple Shapes (Sphere, Cube, Disk, Plane, etc.) [EB/OL]. http://www.scratchapixel.com/lessons/3d-basic-rendering/minimal-ray-tracer-rendering-simple-shapes/ray-sphere-intersection.
[8] Wikipedia. Fresnel equations[EB/OL]. https://www.cnblogs.com/night-ride-depart/p/7429618.html, 2018.
[9] Wikipedia. Phong reflection model[EB/OL]. https://en.wikipedia.org/wiki/Phong_reflection_model, 2018.

其中，对于[M]类型的参考文献：文献[1]主要参考第 17 章“光照模型与面绘制算法”、第 22 章“可编程着色器”理论部分，页码范围：p385-399、p506-524；文献[2]主要用于 OpenGL 编程细节、语法规则等查看，页码范围全书；文献[3]主要参考第 9.4.4 节“立方体贴图”原理说明与程序源码、第 15 章“可编程管线：这已不是旧式的 OpenGL”中 GLSL 语言编程，页码范围：p234-237、p336-356，同时也移植了配套光盘中相关部分的一些代码；文献[4]主要基于第 5.2 节“三维空间中的平面”、第 6 章“光线追踪”二者中的数学方法构建属于本项目的数学模型，页码范围：p60-62、p81-95；文献[5]主要参考、引用第 10 章“光照计算”中的 Phong 光照模型、光线追踪基本原理、光线追踪基本模型等，页码范围：p172-198；