厦門大學

实验 6

实验名称 Ray Tracing 光线跟踪

姓　　名：雷昱

学　　号：22920202204666

学　　院：信息学院

专　　业：软件工程

年　　级：2020 级

二〇二二年 **5** 月 **18** 日

实验 6、Ray Tracing 光线跟踪

建议阅读资料：

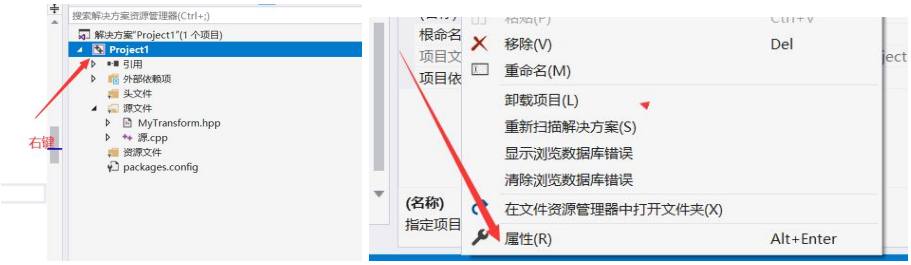
（1） 课件

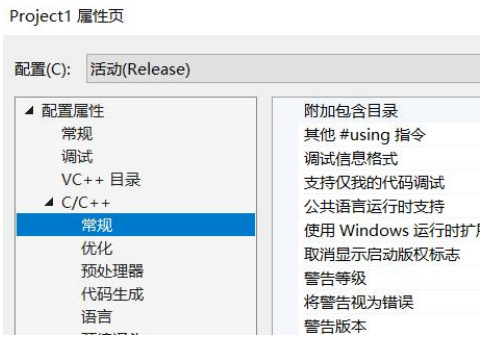
学习要求：

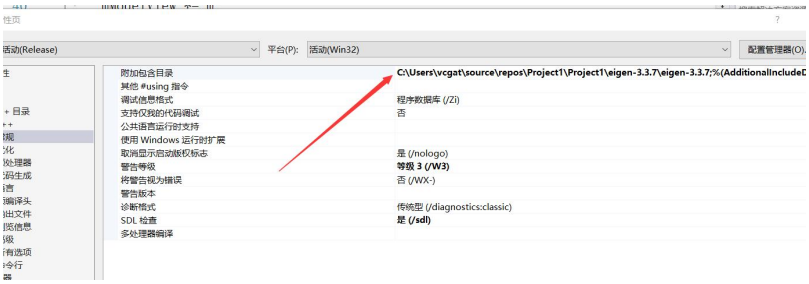
（1） 掌握 Whitted Style 光线跟踪算法

Task1. 自行实现 Whitted 光线跟踪程序，要求实现漫反射、镜面反射和折射功能。请严格按照以下要求实现：

1. **创建空工程，建议设置成 release 模式（本次实验不需要配置 freeglut 库）**
2. **配置 eigen 库（eigen 用于矩阵和向量的计算）。做法是：将 eigen 库解压，同时在刚才创建的工程中设置 eigen 库路径。右键点击工程->属性->C/C++ 常规 -> 将 eigen 所在目录填到“附加包含目录”这一栏。**

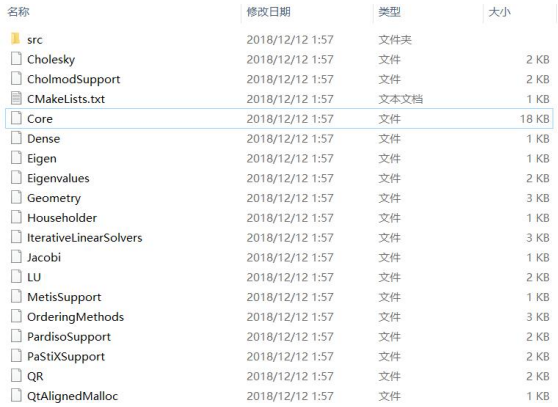






注意：上述路径是我的路径，请根据自己的解压位置填写路径。

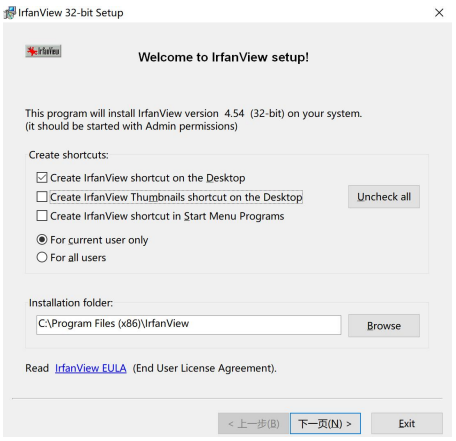
应该定位到这个目录的上一级：

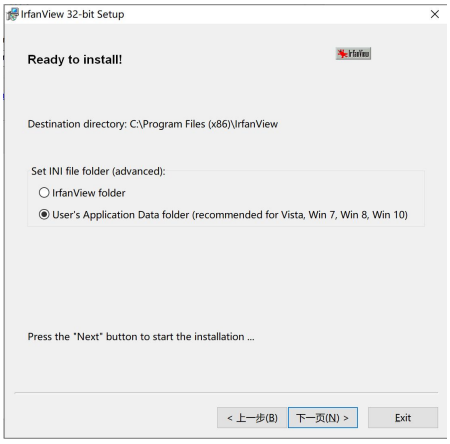


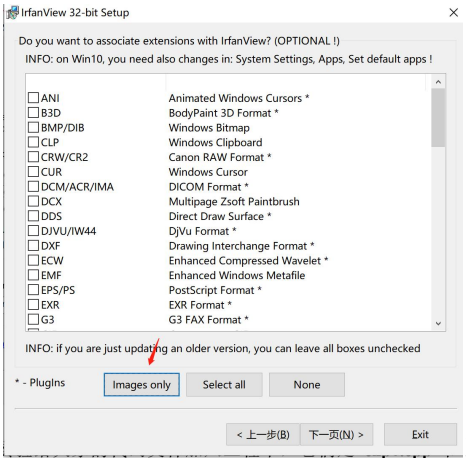
名称	修改日期	类型	大小
src	2018/12/12 1:57	文件夹	
Cholesky	2018/12/12 1:57	文件	2 KB
CholmodSupport	2018/12/12 1:57	文件	2 KB
CMakeLists.txt	2018/12/12 1:57	文本文档	1 KB
Core	2018/12/12 1:57	文件	18 KB
Dense	2018/12/12 1:57	文件	1 KB
Eigen	2018/12/12 1:57	文件	1 KB
Eigenvalues	2018/12/12 1:57	文件	2 KB
Geometry	2018/12/12 1:57	文件	3 KB
Householder	2018/12/12 1:57	文件	1 KB
IterativeLinearSolvers	2018/12/12 1:57	文件	3 KB
Jacobi	2018/12/12 1:57	文件	1 KB
LU	2018/12/12 1:57	文件	2 KB
MetisSupport	2018/12/12 1:57	文件	1 KB
OrderingMethods	2018/12/12 1:57	文件	3 KB
PardisoSupport	2018/12/12 1:57	文件	2 KB
PaStiXSupport	2018/12/12 1:57	文件	2 KB
QR	2018/12/12 1:57	文件	2 KB
QtAlignedMalloc	2018/12/12 1:57	文件	1 KB

3. **安装 IrFanViewer，用于查看生成的 ppm 图像文件。**

   安装过程若没有说明，则为默认选项。此处路径是默认安装位置，可根据自己需求选择路径。







不需要勾选其他辅助拓展。

4. **将本次实验给大家的代码文件加入工程中，它们是 exp6.cpp 和 camera.hpp，ppm.hpp， ray.hpp， raytracer.hpp， scene.hpp。在本次实验过程中，除非特别说明，不允许改动 exp6.cpp 的内容（也不必关心具体代码）。**

其中 exp6.cpp 包含主函数用于调用光线跟踪函数；

camera.hpp 用于生成光线，关于光线生成的函数在此；

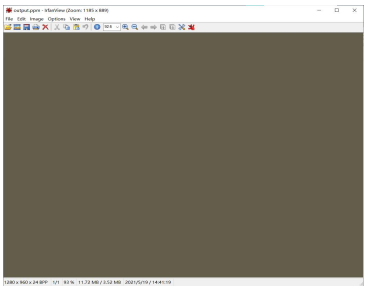
ray.hpp 是光线的类；

scene.hpp 组织场景，这里主要是球面；

raytracer.hpp 光线跟踪的主程序；

ppm.hpp 是用于 **ppm** 图像格式的存储。

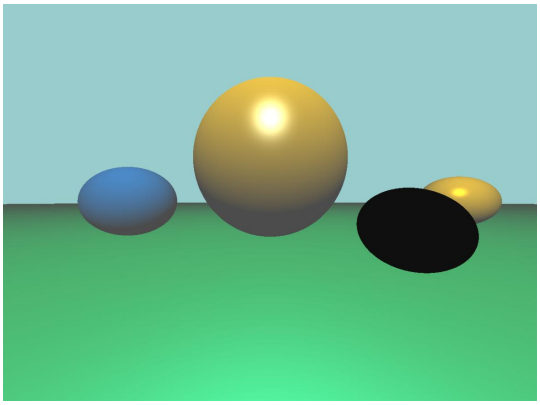
5. 不做任何更改，如果配置正确，运行代码后，你应该能在程序文件夹下找到 **output.ppm** 文件，用 **IrFanViewer** 打开应该是一张纯色的图像。



6. 修改 **camera.hpp** 中生成光线的函数

```
Ray GenerateRay(float u, float v)
```

如果改对的话，你应该能看到这样的图像，右侧的球呈现出黑色是因为其漫反射属性被设得很低。

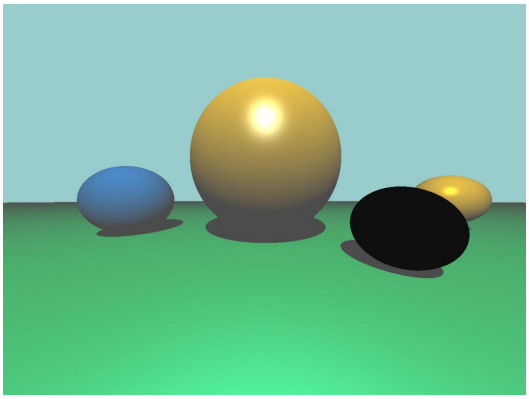


7. 在 **RayTracer.hpp** 的函数

```
Vector3f RayColor(const Ray& ray, Scene& scene, int depth=0, bool test=false)
```

加入 **shadow ray** 是否能看到光源的代码，以生成阴影。

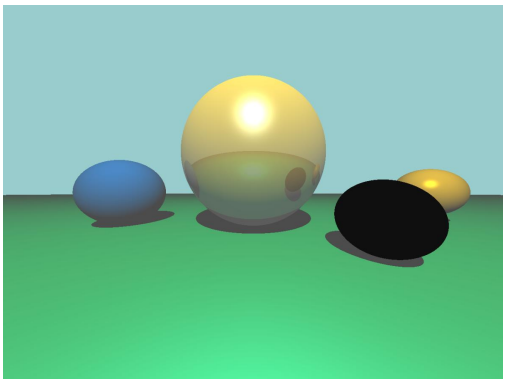
实现正确的话，你能看到阴影。



8. 在 **RayTracer.hpp** 的函数

```
Vector3f RayColor(const Ray& ray, Scene& scene, int depth=0, bool test=false)
```

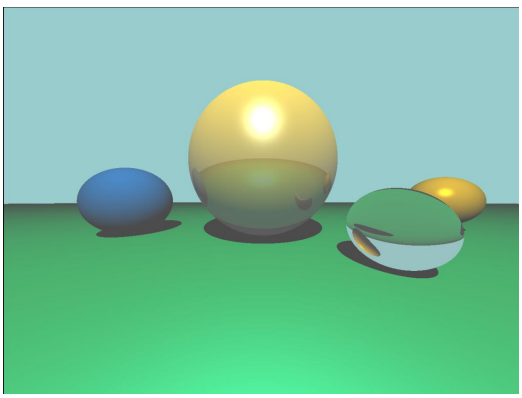
加入完成镜面反射的递归代码。如果加入正确的话，你能看到中间黄球有镜面反射：



9. 在 **RayTracer.hpp** 的函数

```
Vector3f RayColor(const Ray& ray, Scene& scene, int depth=0, bool test=false)
```

加入完成折射的递归代码。如果加入正确的话，你能看到右侧小球有折射效果：



10. 附加选项：（根据实现情况，期末最终的实验成绩可以*1.05~1.1）

在此代码基础上，生成一段小球从天而降的动画，可以考虑加入运动模糊、软阴影效果，小球弹跳符合物理规律。

作业提交说明：

本次实验共有 1 个任务。

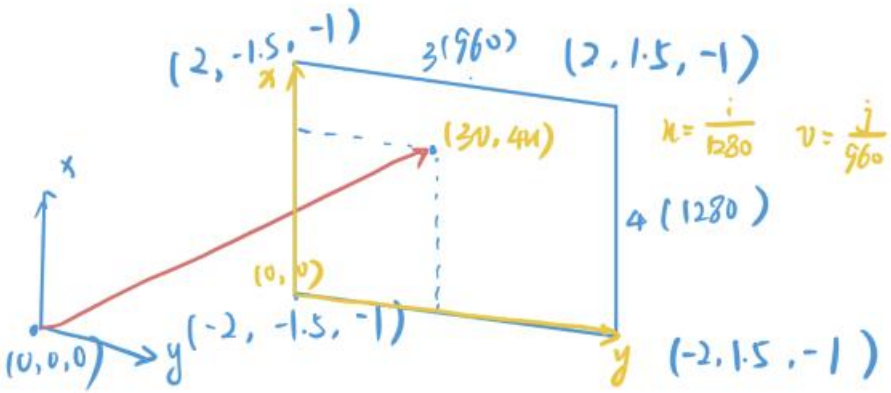
提交方式为：*将代码、实验报告、对应帧的 ppm 文件、附加题的 gif(可选)*放到一个文件夹中，命名为“您的学号_姓名”，打包上传到 ftp 服务器中相应目录下。每次实验作业的提交截止日期为下一次实验课前一天晚上。

特别说明：本次实验 1 个任务都需要提交。您需要提供一个完整的文档（不限格式），逐条说明您完成每一条任务的具体情况。

实验步骤如下：

1.修改 `Ray GenerateRay(float u, float v)`

①光线生成计算



②代码展示

```
Ray GenerateRay(float u, float v)
{
    //请改掉此函数，使得生成正确的光线
    return Ray(_origin,
        _horizontal * u + _vetical * v + _lower_left_corner - _origin);
}
```

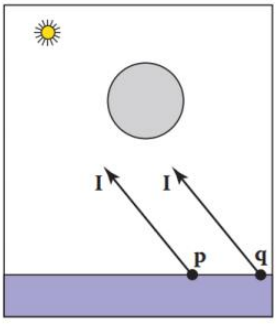
2.在 `Vector3f RayColor(const Ray& ray, Scene& scene, int depth=0, bool test=false)`中加入 shadow ray 是否能看到光源的代码，以生成阴影。

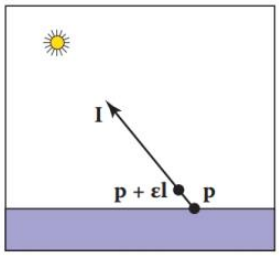
①将 shadow_ray 与小球求交，判断是否被遮挡。

当被遮挡时无镜面、漫反射。

②如图所示 shadow ray

Shadow ray





③代码实现

```
for (int i = 0; i < scene.ObjectCount(); ++i)
{
    HitInfo ht;
    bool bhit = scene.GetObjectPtr(i)->Hit(shadow_ray, ht);
    if (bhit)
    {
        if (ht.t > 0 && ht.t < closest_hp.t)
        {
                isShadow = true;
                break;
        }
    }
}
```

3. 在 `Vector3f RayColor(const Ray& ray, Scene& scene, int depth=0, bool test=false)` 加入完成镜面反射的递归代码。如果加入正确的话，你能看到中间黄球有镜面反射。

①反射计算公式

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}.$$

$$\text{color } c = c + k_m \text{raycolor}(\mathbf{p} + s\mathbf{r}, \epsilon, \infty)$$

②分析

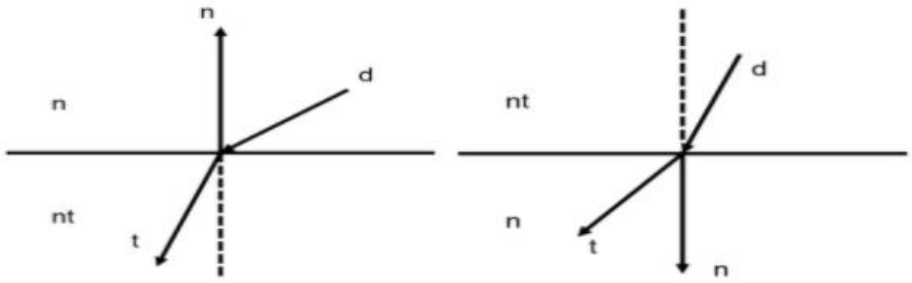
这样的递归定义来决定像素颜色，这个实质上就是不断加上反射光带来的颜色，其中 km 就是这次反射的表面颜色，p+sr 就是入射光线的方向（r 是入射光线，p 就是交点）

③代码展示

```
if (mtl._reflective)
{
    //镜面光，调用递归

    //请以下部分，填入正确的递归调用

    reflectColor = RayColor(reflectRay, scene, ++depth);

    //请以上部分，填入正确的递归调用
}
```

4. 在 `Vector3f RayColor(const Ray& ray, Scene& scene, int depth=0, bool test=false)` 加入完成折射的递归代码。如果加入正确的话，你能看到右侧小球有折射效果。

①折射



折射的两种情况

左图这种情况时，即 d 和 n 成钝角用下列的折射公式

$$t = \frac{n(\vec{d} - \vec{n}(\vec{d}\cdot\vec{n}))}{nt} - \vec{n}\sqrt{1 - \frac{n^2(1-(\vec{d}\cdot\vec{n})^2)}{nt^2}}$$

右图这种情况时，即 d 和 n 成锐角用下列的折射公式

$$t = \frac{nt(\vec{d} - \vec{n}(\vec{d}\cdot\vec{n}))}{n} + \vec{n}\sqrt{1 - \frac{nt^2(1-(\vec{d}\cdot\vec{n})^2)}{n^2}}$$

②代码展示

```
if (refract)
{
    Ray refractRay(closest_hp.position, refractDir);
    refractRay._origin = closest_hp.position + 1e-2*refractDir.normalized();

    //请把下面代码改掉，填入正确的递归调用

    refractionColor = RayColor(refractRay, scene, ++depth);

    //请把上面代码改掉，填入正确的递归调用
}
else
{
    refractionColor = Vector3f(0, 0, 0);
}
```