README

作业4 3-2

Ray Casting

编写虚类Object3D

实现很简单，只需要按照要求添加一个纯虚函数

virtual bool intersect(const Ray &r, Hit &h, float tmin) = 0;

代表着光线与物体求交的算法。

添加保护成员变量Material代表材质，只需要添加不需要实现，具体实现由子类来进行。

编写类Sphere

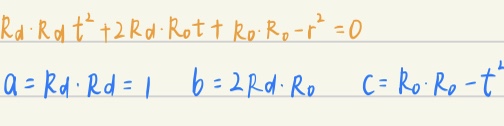
该类继承于虚类Object3D，在实现纯虚函数的同时还添加新的私有成员变量m\_center（球心），m\_radius（半径），并且需要对传入的变量进行赋值。

该类代表球类对象，所以函数intersect实现的是光线与球类求交的算法。

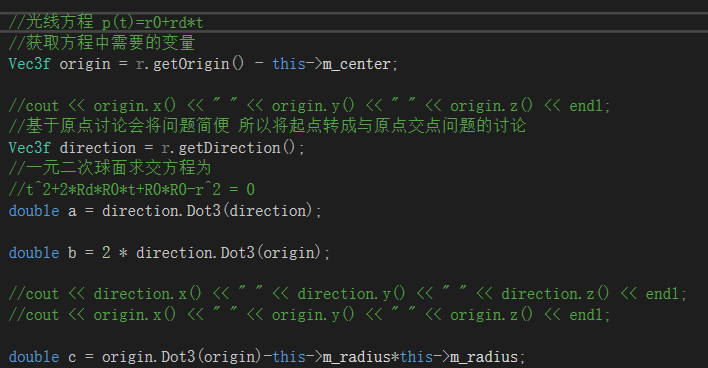
* **Sphere intersect的实现**

传入参数有光线类，交点类，和参数t的最小值。

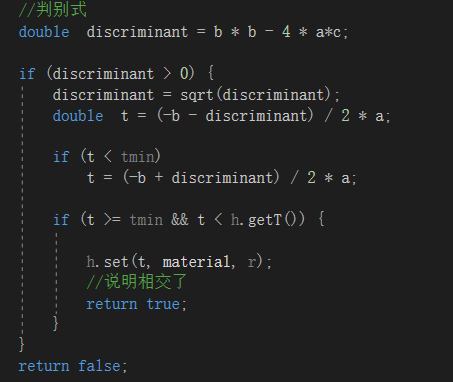
根据光线与球面求交公式可知



需要使用到光线的direction，origin成员变量，由于网课和ppt中计算光线与球面交点是基于球心进行计算，所以在最开始将变量origin转换到世界坐标的原点处，然后将变量带入以上公式即可，代码如下：



之后就是求解一元二次方程，可以由判别式进行有无解进行判断，如果有解则带入解一元二次方程的式子中，这里取辨别式小于等于0时返回false，原因是网课中提到有一种情况，就是视线位于切线处，虽然位于面上，但是可能由于误差的关系，判断为不在面上，简单的处理方法就是不取该点。并且在判定判别式有解之后，还需要对t的最小值，和当前值进行大小的判断，t必须大于t最小值，并且小于t当前值才能更新交点h的值，目的是保证是最近的点，并且保证物体是位于视线前面而不是后面，具体代码如下：

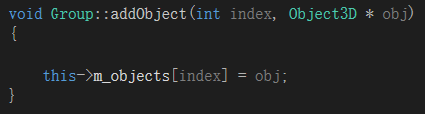


这里默认取负值，因为理论上如果是两个正解，负值会更小，为了防止出现一正一负，又加上了一层条件，就是如果取赋值小于t最小值，就取正值。

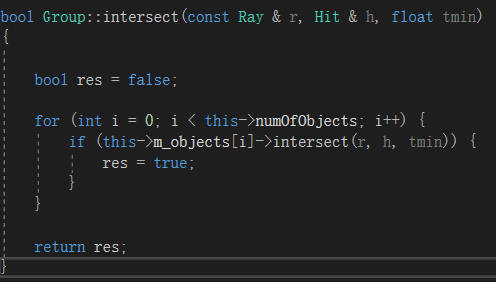
编写类Group

该类的作用仅仅是作为一个数组类，保存读入的物体数据，并且遍历整个数组进行求交算法。

实现函数void addObject(int index, Object3D \*obj)，目的是更新成员变量数组对应下标的对象，实现代码为：



实现函数intersect，目的是遍历数组成员，分别进行求交算法，并且判断光线是否与该物体有交点，实现代码为：



编写纯虚Camera 类

在该类中添加纯虚函数

virtual Ray generateRay(Vec2f point) = 0; 目的是根据模拟屏幕坐标生成光线起点变量。

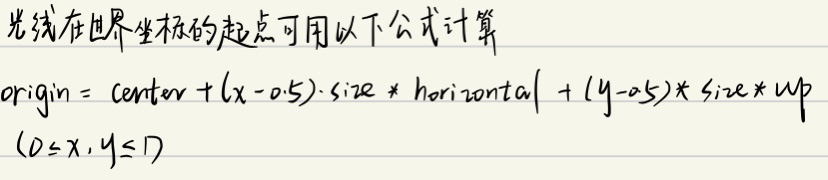
virtual float getTMin() const = 0; 返回最小t值

编写类orthographic

为了模拟相机需要添加五个变量，分别是m\_center相机中心，m\_up相机竖直方向，m\_horizontal相机水平方向，m\_direction相机朝向，m\_size相机可视范围。

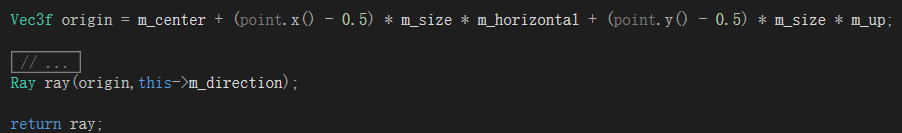
1. 实现generateRay函数

求解屏幕中任意一条光线起点也有相应的公式计算，公式如下：



将全部参数带入方程即可，并将求出的origin向量和成员变量m\_direction初始化类Ray。

代码实现为：

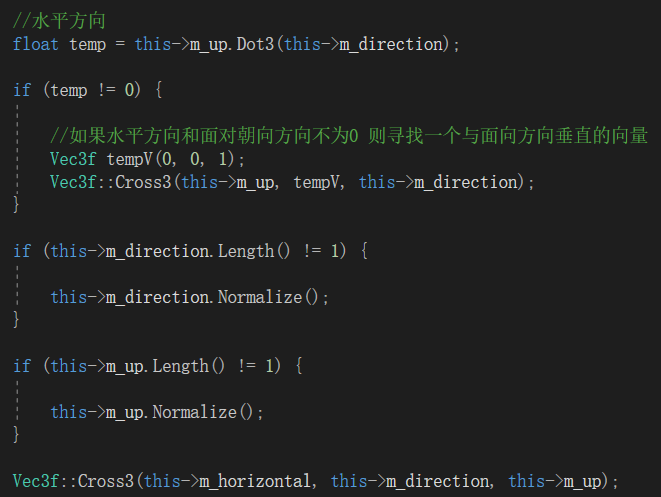


1. 实现getTMin函数

返回自设定的t最小值即可，这里设置最小值为-100000.0。

1. 注意事项

因为方程中的三个正交变量都需要单位化，并且需要互相垂直，所以在进行初始化时要进行判断传入的竖直向量是否垂直于朝向变量，这两个变量是否为单位向量，如果两个条件都满足，才可以继续求解水平向量，代码如下：



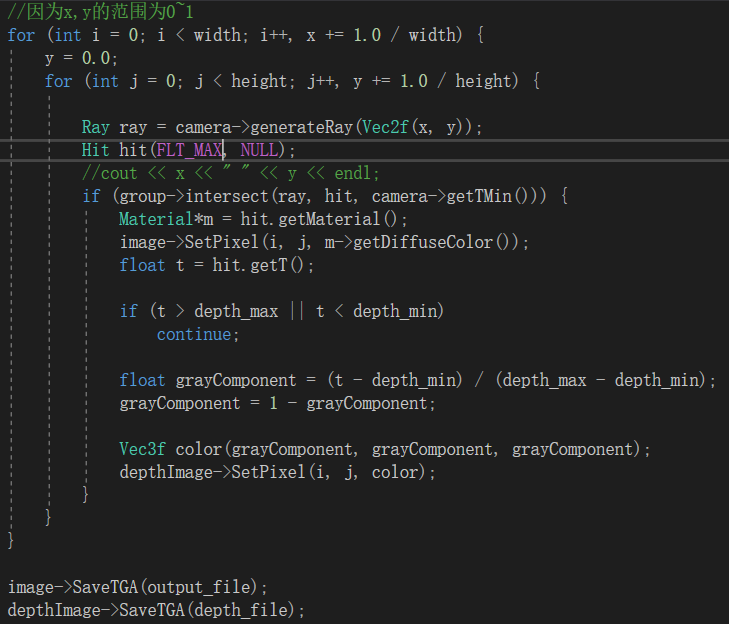
编写主程序main

网课中介绍光线投射的整个过程为：外层循环遍历模拟屏幕中的所有点，内层循环遍历模拟屏幕上的所有物体，求模拟屏幕上的光线与物体表面的交点，求出最近点，之后求出像素的颜色，然后保存到图片中。

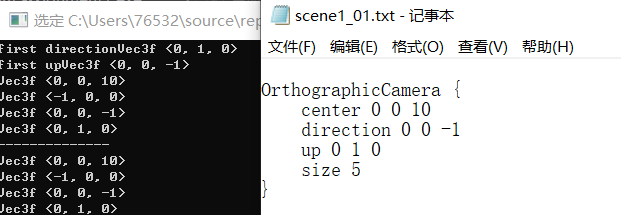
遍历整个屏幕的的点，遍历次数由传入的width和height值决定，并且传入坐标值x，y的范围都为（0<= and >=1)所以每次的循环x增长1/width，y增长1/height，这样就能保证x，y能处在需要的范围之中。

第二层循环就是要计算最近交点并且设置像素点数据。

使用group中的intersect函数进行成员数组中的物体求交，如果存在交点，则向图片类写入像素数据，代码如下：



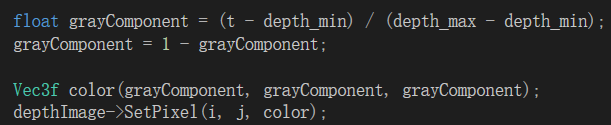
实习中遇到的问题以及解决方法

1. 在进行求交点解一元二次方程时，发现求出的朝向向量和屏幕线条向量垂直，导致生成的图片没有任何东西。后来检查了传入的变量值，发现相机朝向向量和竖直向量反了。

后来调整过来之后，生成的图片才有图形显示。

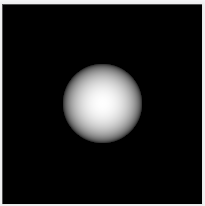
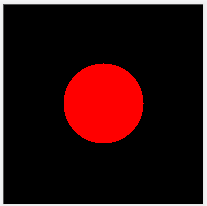
1. 但是显示图形之后，发现计算最近点的算法不起作用，后面的球显示在了最前面。后来向交点类hit传入最大值FLT\_MAX之后，才保存正确的图片。后来分析了原因，可能是因为之前宏定义的最大值不够大，又或者是没有对类hit进行初始化。
2. 计算深度值生成深度图片因为不知道计算方法，上网搜索之后，使用了网上代码，能够生成相应的深度图片。

计算方法如下：

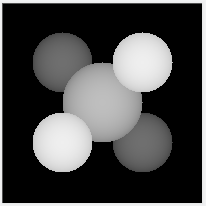
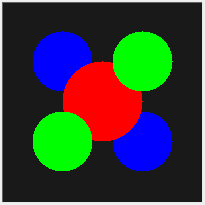


实验结果展示：

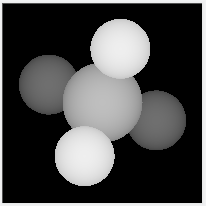
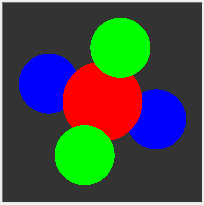
assighment1 -input scene1\_01.txt -size 200 200 -output output1\_01.tga -depth 9 10 depth1\_01.tga



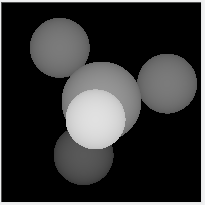
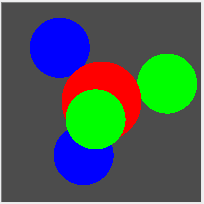
assighment1 -input scene1\_02.txt -size 200 200 -output output1\_02.tga -depth 8 12 depth1\_02.tga



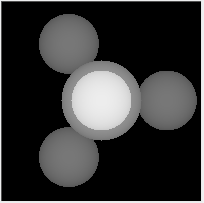
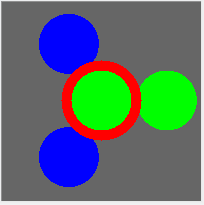
assighment1 -input scene1\_03.txt -size 200 200 -output output1\_03.tga -depth 8 12 depth1\_03.tga



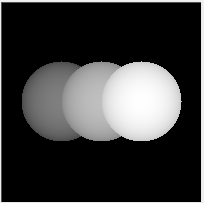
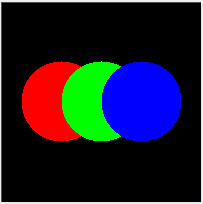
assighment1 -input scene1\_04.txt -size 200 200 -output output1\_04.tga -depth 12 17 depth1\_04.tga



assighment1 -input scene1\_05.txt -size 200 200 -output output1\_05.tga -depth 14.5 19.5 depth1\_05.tga



assighment1 -input scene1\_06.txt -size 200 200 -output output1\_06.tga -depth 3 7 depth1\_06.tga



assighment1 -input scene1\_07.txt -size 200 200 -output output1\_07.tga -depth -2 2 depth1\_07.tga

