光线投射实验报告

代码逻辑

此部分按顺序介绍代码的实现逻辑

Sphere

Sphere()

默认构造函数会构造一个球心在原点的单位球

intersect()

总体逻辑为判断r是否与球有交，同时与hit取最小

- 计算球心到光源的向量l,并将l与光线的单位方向向量点乘得到t_p
- 若t_p小于零切l的长度大于半径，则无交点返回false

- 计算球心到光线的距离d = $l^2 - t_p^2$
- 若d大于等于球的半径，则没有交点返回false
- 计算 $t_p i = r^2 - d^2$
- 当l长度大于半径，即光源在球外时,$t = t_p - t_{pi}$，当l的长度小于半径，即光源在球内时,$t = t_p + t_{pi}$
- 如果新的t比hit中记录的t大或等,返回false
- 反之，计算法向量n,并用t,改球的材质,以及t值更新hit,返回true

Plane

intersect()

- 根据光线和平面的方程

$$P(t) = R_o + tR_d$$

$$nP(t) + D = 0$$

直接计算出$t = -(D + nR_o)/(nR_d)$

- 当计算出的t大于hit中的t时，返回false
- 反之更新n,t,material并返回true

Triangle

构造函数

- 主要是normal的计算,方法为normalized(cross(a-b,a-c))

intersect()

- 计算$e_1 = a - b, e_2 = a - c, s = a - r.origin$
- 构建矩阵m1,m2,m3,d

```
Matrix3f m1 = Matrix3f(s, e1, e2, true);
Matrix3f m2 = Matrix3f(unit_direction, s, e2, true);
Matrix3f m3 = Matrix3f(unit_direction, e1, s, true);
Matrix3f d = Matrix3f(unit_direction, e1, e2, true);
```

- 计算出t,beta,gama

```
float div = d.determinant();
float t = m1.determinant() / div;
float beta = m2.determinant() / div;
float gama = m3.determinant() / div;
```

- 接下来进行判断，若不满足$t > 0, 0 \leq \beta, \gamma \leq 1, \beta + \gamma \leq 1$则返回false
- 如果t的值大于hit中t的值，则返回false
- 反之更新n,t和material

Group

构造函数

将所有的Object3D存储在vector中，且进行如下初始化操作

```
this->all_object.resize(num_objects);
this->num_objects = num_objects;
```

intersect()

对于光线r，循环all_object中的所有Object3D，并调用其求交算法，若有交则返回True，同时在遍历的同时，更新hit使其t为与所有物体求交得到的最小的t

PerspectiveCamera

该相机的大致思路为，将距离视线origin一个单位长度的相机空间的坐标映射到真实的世界空间坐标

- 计算光线在相机空间的方向

$$d_{R_c} = normalized(\frac{u - c_x}{f_x}, \frac{c_y - v}{f_y}, 1)$$

其中f_x, f_y分别代表真实世界空间的坐标单位长度在相机空间的坐标中的长度,故计算公式为$f_x = \frac{2*tan(\frac{\theta}{2})}{width}, f_y = \frac{2*tan(\frac{\theta}{2})}{height}$

- 然后将这个方向映射到真实的世界坐标

$$d_{R_w} = Rd_{R_c}$$

- 其中R为正交单位方向向量组成的矩阵 $R = [horizontal, -up, direction]$

Shade 方法

使用不含环境光项的冯模型公式计算即可

代码参考

没有参考代码，算法思路参考课上讲的内容和PPT

遇到的问题

一个有价值的问题是关于Transformer类的，由于其实现逻辑为坐标变化，故不能再transformer后进行光线方向的normalized，否则则会出现光线位置偏移的错误.