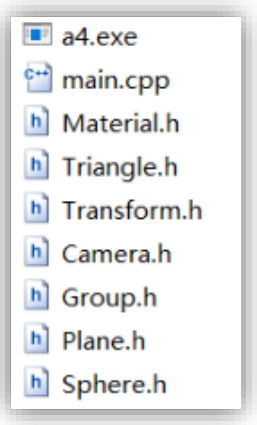
计算机图形学实习四

一、 提交文件结构



以上代码文件是做出修改的文件，其他的文件与所给代码文件保持不变不做修改。a4.exe 为 release 编译可执行文件。

二、 编译环境

Windows 10(64 位)+ visual studio 2010

三、 实现功能

基础要求全部实现。

1. 实现了 Sphere, plane, triangle, group, transform 的求交算法。
2. 实现 PerspectiveCamera 类。
3. 实现命令参数读入与处理。
4. 实现深度图像生成。
5. 实现 Phong 光照模型，包括环境光、高光、漫反射光。
6. 实现纹理映射。
7. 主循环中实现了所有逻辑的调用与生成要求的所有图片。

四、 可执行文件

编译生成可执行文件 a4，依次按照实习要求中给出的命令参数使用即可。下面给出所有的命令参数：

```
-input scene01_plane.txt -size 200 200 -output 1.bmp
-input scene02_cube.txt -size 200 200 -output 2.bmp
-input scene03_sphere.txt -size 200 200 -output 3.bmp
-input scene04_axes.txt -size 200 200 -output 4.bmp
-input scene05_bunny_200.txt -size 200 200 -output 5.bmp
-input scene06_bunny_1k.txt -size 200 200 -output 6.bmp
-input scene07_shine.txt -size 200 200 -output 7.bmp
-input scene08_c.txt -size 200 200 -output 8.bmp
-input scene09_s.txt -size 200 200 -output 9.bmp
```

五、 实现过程

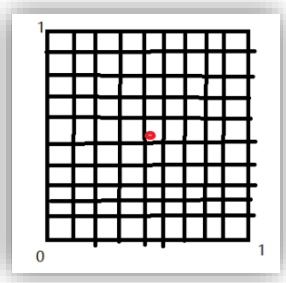
1. sphere 的求交算法。采用课上讲到的联立后的一元二次方程解出即可。这里需要注意的是，课上讲的是球心在坐标系原点的球的计算过程，这里需要任一位置的球心的计算方程，因此需要推导一下，球面方程是 (R-RO)*(R-RO)=0。替换代入即可求得。解方程产生的双根及处理办法课上已讲明。Hit 设置的法线需要归一化处理。

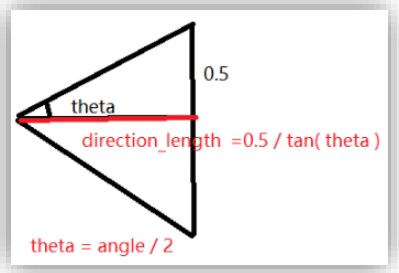
2. group 的求交算法。求交算法 intersect 的返回值是需要拿来使用的，如果交到了，返回 true，如果没有交到，则返回 false。这点需要在后面写其他的求交算法时注意。Group 的求交算法即遍历组内其他物体的求交算法即可。

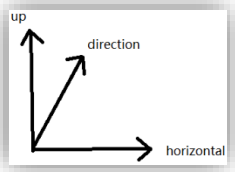
3. 实现 PerspectiveCamera 类。这里我的实现思路是，将视图看作边长为 1，则所有像素点在视图上可如下描述：中心点坐标 + horizontal_offset + up_offset。中心点坐标由 camera 的中心点 center 和方向 direction 得出，即 center + direction，不过这里需要注意的是，direction 不是构建 camera 时传进来的 direction，它不是归一化的，而是“缩短”了一定距离，是因为视角 angle 的关系，具体两者关系描述如下：direction_length= 0.5 / tan(angle / 2)*direction_base_length。这样的话，如果视图水平方向单位向量看作 horizontal，竖直方向看作 up，这两者均为单位向量，我们给出一个 point.x 属于[0,1]，point.y 属于[0,1]的二元组(x,y)，那么射向该位置的像素的光线(ray)可以求得：

光源 ray_origin=center

方向 ray_direction=(0.5 / tan(angle / 2)*direction +
(point.x() - 0.5)*horizontal + ((point.y() - 0.5)*up)).normalized()







4. 主循环实现场景装入。调用 sceneParse 即可。这里需要注意的是，在 shade 之前，给所有像素设置的初始值为 sceneparse->getBackground()的颜色值。

5. main 中实现读入场景，循环每一个像素，与场景中的 Group 求交即可。求得 hit 后，设置该像素的颜色值。Shade 的值后面实现。

6. 实现深度 t 的渲染。界定深度范围 tmax 和 tmin，由命令参数指定。当 tmin<t<tmax 时，深度比例为 k=(t-tmin)/(tmax-tmin)，深度颜色为 k*(1,1,1)。生成对应的深度图像即可。

7. Material 类实现了 shade 函数求解漫反射。

```
Vector3f Shade( const Ray& ray, const Hit& hit,
                const Vector3f& dirToLight, const Vector3f& lightColor )
```

shade 中 dirToLight 为朝向光源的方向，lightColor 为光颜色，hit 为求交的 hit。根据课上学习的公式和指导书中的公式：（记得正值化）

$$d=\begin{cases} L\bullet N & if L\bullet N>0 \\ 0 & otherwise \end{cases}$$

L 为 dirToLight，C_light 为 lightColor，N 为 hit.getNormal，Kd 为 diffuseColor。代入如下公式即可求出颜色值：（main 循环中记得添加环境光）

$$c_{pixel}=c_{ambient}*k_a+\sum[clamp(L_i\bullet N)*c_{light}*k_d]$$

Shade 返回该颜色值。在 main 函数中累加该颜色值即可。

8. shade 函数添加高光反射。

$$c_s=\begin{cases} (L\bullet R)^s 若 L\bullet R>0 \\ 0 其它情形 \end{cases}$$

L 为 dirToLight，R 可以由如下公式求解：

$$\mathbf{r}+\mathbf{l}=2\ \cos\theta\,\mathbf{n}$$
$$\mathbf{r}=2(\mathbf{n}\cdot\mathbf{l})\mathbf{n}-\mathbf{l}$$

其中 N 为法向量。R 可以求得代入上述公式，s 给出为 shininess，记得正值化。代入如下公式可求出累加高光后所有的 shade：

$$c_{pixel}=c_{ambient}*k_a+\sum_i[clamp(L_i\bullet N)*c_{light}*k_d+clamp(L_i\bullet R)^s*c_{light}*k_s]$$

其中 Ks 给出为 specularColor，C_light 给出为 lightColor。C_pixel 为最终颜色，在 main 函数中累加即可。

9. 实现 plane 的求交算法。Plane 方程为 H(P) = n • P - D = 0，光线方程 P(t) = Ro + t * Rd，可以求得 t = -(-D + n • Ro) / n • Rd。求解即可。

10. 实现 triangle 的求交算法。使用质心表示的三角形方程：

P(a, b, g) = aa + bb + gc

其中 a + b + g =1

且 0 < a < 1 & 0 < b < 1 & 0 < g < 1

与光线求交方程联立得：

Ro + t * Rd = a + b(b-a) + g(c-a)

即矩阵形式：

$$\begin{bmatrix} a_x-b_x & a_x-c_x & R_{dx} \\ a_y-b_y & a_y-c_y & R_{dy} \\ a_z-b_z & a_z-c_z & R_{dz} \end{bmatrix}\begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix}=\begin{bmatrix} a_x-R_{ox} \\ a_y-R_{oy} \\ a_z-R_{oz} \end{bmatrix}$$

使用克莱姆法则分别求解：

$$\beta = \frac{\begin{vmatrix} a_x - R_{ox} & a_x - c_x & R_{dx} \\ a_y - R_{oy} & a_y - c_y & R_{dy} \\ a_z - R_{oz} & a_z - c_z & R_{dz} \end{vmatrix}}{|A|} \quad \gamma = \frac{\begin{vmatrix} a_x - b_x & a_x - R_{ox} & R_{dx} \\ a_y - b_y & a_y - R_{oy} & R_{dy} \\ a_z - b_z & a_z - R_{oz} & R_{dz} \end{vmatrix}}{|A|} \quad t = \frac{\begin{vmatrix} a_x - b_x & a_x - c_x & a_x - R_{ox} \\ a_y - b_y & a_y - c_y & a_y - R_{oy} \\ a_z - b_z & a_z - c_z & a_z - R_{oz} \end{vmatrix}}{|A|}$$

Alpha = 1 - beta - gama

求交成功的条件需要注意,不仅 t>tmin,t<hit.getT(),而且要 alpha>=0,beta>=0,gama>=0,

求交 hit 设置法向量为:（需要归一化）

$$\text{Normal} = \lambda_0 n_0 + \lambda_1 n_1 + \lambda_2 n_2$$

11. 实现纹理映射。在三角形的求交函数中插值纹理坐标，利用上述求得的 alpha，beta，gama 对纹理插值：

hit.setTexCoord(alpha * texCoords[0] + beta * texCoords[1] + gama * texCoords[2])

Material 的 shade 函数中使用纹理坐标代替漫反射光：

```
if (t.valid()) {
    Vector3f texture = t(hit.texCoord[0], hit.texCoord[1]);
    diffuse_color= diffuse_temp * lightColor*texture;
}else{diffuse_color= diffuse_temp * lightColor*diffuseColor;}
```

六、 问题与解决

1. 相机生成光线 ray 的过程。这个对于相机这个模型需要研究透彻，需要搞清楚相机的三个向量怎样表示视图上单个点。

2. 三角形求交的过程，需要注意不仅仅 t>=tmin，t<hit.getT()，而且要注意 alpha>=0，Beta>=0，Gama>=0。这个很影响到求解。

作者：AnDJ