

# CG Project Final

## 1. 总览

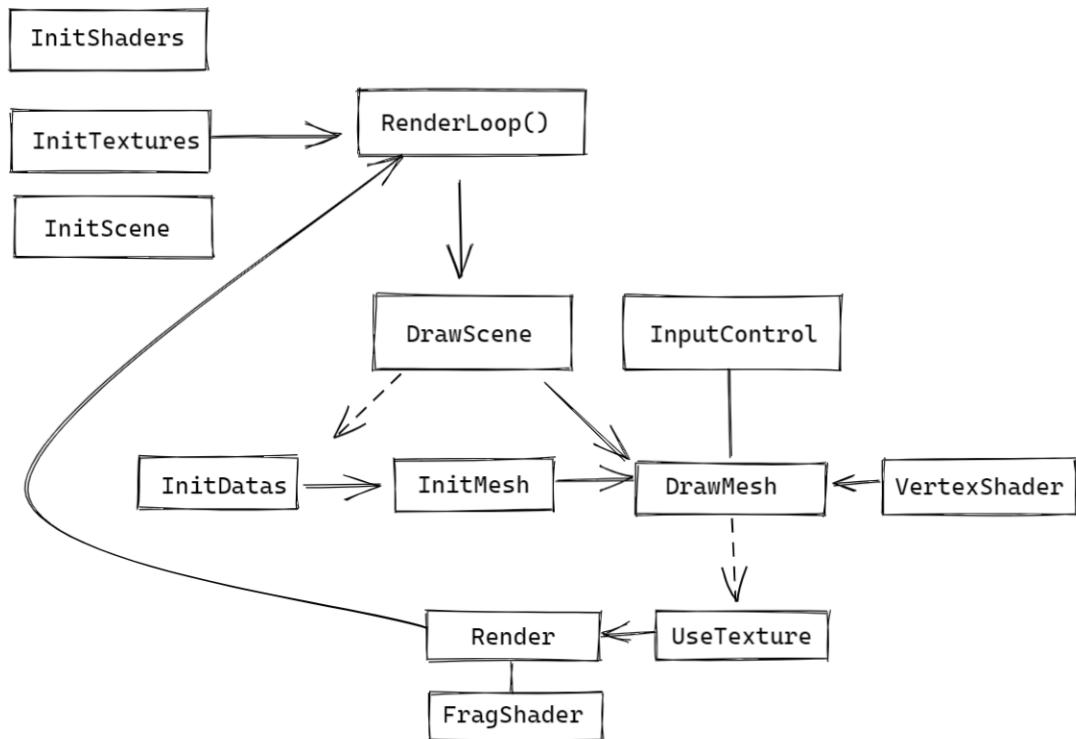


Diagram 1  
一个简要的项目流程

## 2. 功能实现

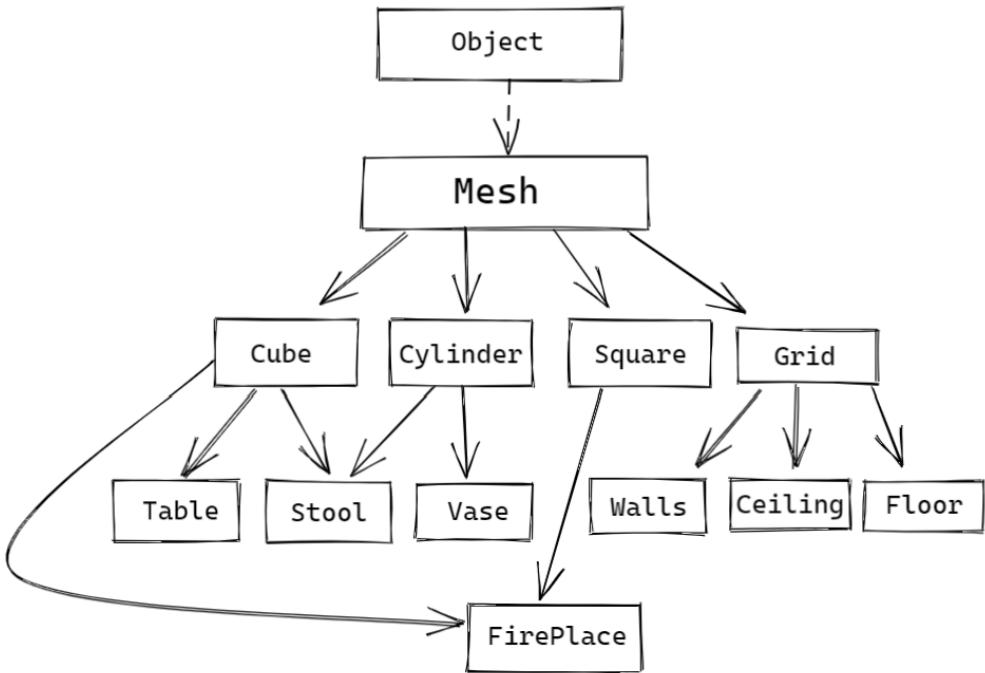
### 基本体素实现

Class `VertexData`为一个点所包含的所有数据，包括坐标，颜色，点法向量，平面法向量（对于一个点： $n = \text{flatn}$ ）和纹理坐标。Class `Object`为基类，包含绘制一个网格所需数据，包括顶点数组，材质三要素等，并提供

`virtual bool draw()`接口。Class `Mesh`继承自`Object`，为基本绘制单元，场景所有模型皆由`Mesh`完成真正的绘制。在此基础上，实现以下绘制单元

- Cube -> Stool / Table
- Cylinder -> Vase
- Grid -> Wall / Ceiling / floor
- Square -> FirePlace

他们之间的关系见下图，其中虚线代表继承关系，实线代表包含关系：



**Diagram 2**

由基础模型再到复杂模型的一个好处在于，可以重复利用模型。比如在绘制桌子和凳子时，可以使用同一批*Legs*。而凳子的凳面、花瓶、碗都由圆柱体变形而来。墙壁、天花板、地板都由格子*Grid*变形而来。因此我们可以在少量代码的前提下，创造出丰富多样的模型。

具体实现上没有什么特别的花样，都是通过调整参数得来。稍微麻烦一点的是绘制花瓶和碗，考虑柱体的内半径、外半径和对称中心，需要比cube复杂一点的几何知识。

## 材质、纹理的显示和编辑

使用Class TextureManager管理所有纹理，初始化时将场景所需纹理全部依次载入，并将生成的id和纹理名称放入map textures中。在Class Scene调用draw函数时，通过TextureManager使用对应的纹理即可。因此若要编辑纹理也非常方便，只需要在初始化时载入更多的纹理，并通过键盘输入控制调用的纹理的名称即可。

材质方面，通过Class Scene的初始化列表，传入对应模型材质的vec3 matAmbient, vec3 matDiffuse, vec3 matSpecular，在绘制时将参数传入mesh，mesh绘制时将参数传入Class Shader，根据当前的光照模型进行计算，详细的介绍放在光照模型中。而要编辑材质也很方便，思路同纹理。

## 基本几何变换

利用内置函数实现的平移、旋转、缩放。

## 漫游功能

左右键控制rotateY，上下键盘控制rotateX，并设置不超过天花板(90°)不低于地板(15°)，使用内置Rotate()函数即可实现Orbit, I / O 控制perspective函数的fovy参数实现Zoom In / Out，并通过windows.size参数实现Zoom to Fit.

## 动画

火焰和喷泉基于粒子效果实现，关键是基于时间的位置函数，其他重要参数包括

- lifetime，粒子显示时间，以火焰为例，当粒子上升到一定高度后，我们将其设置一个透明度，以达到火焰逐渐熄灭的效果

```

float age = time - vertexStartTime;
transparency = 1.0 - age / particleLifetime;

```

同样地，若要调整火焰大小（即部分粒子最大高度），我们可以通过间接改变age实现

- Size: 粒子的尺寸，这里并不是单个粒子的大小，而是每一帧画面中，使用多少个粒子绘制。对于单个粒子，我们使用内置的Point绘制，好处在于可以自动生成纹理坐标，直接贴图

时间函数以喷泉为例，我们并不需要严格按照斜抛运动的方程来绘制，而是计算以一定初始速度（矢量）抛出，粒子能够达到的运动范围（XYZ）是什么，在此基础上，通过对不同粒子设置随机的初始速度，就可以达到喷泉的效果。

```

// get the direction of the initial velocity
theta = mix(0.0f, PI / 6.0f, float(rand()) / RAND_MAX);
phi = mix(0.0f, 2.0f * PI, float(rand()) / RAND_MAX);
vel.x = sin(theta) * cos(phi);
vel.y = cos(theta);
vel.z = sin(theta) * sin(phi);

```

我将时间函数放在了顶点着色器中，好处是可以供火焰、喷泉方便地使用，弊端在于粒子无法带有材质效果，也就是喷泉中的“水滴”和“火焰”粒子仅存在贴图上的不同。

## 光照

光照模型的关键在于着色器，以Phong为例，简单介绍一下原理。

光照效果取决于如何计算光源与材质表面的Diffuse, Ambient, Specular三个分量。我们设最终光照向量

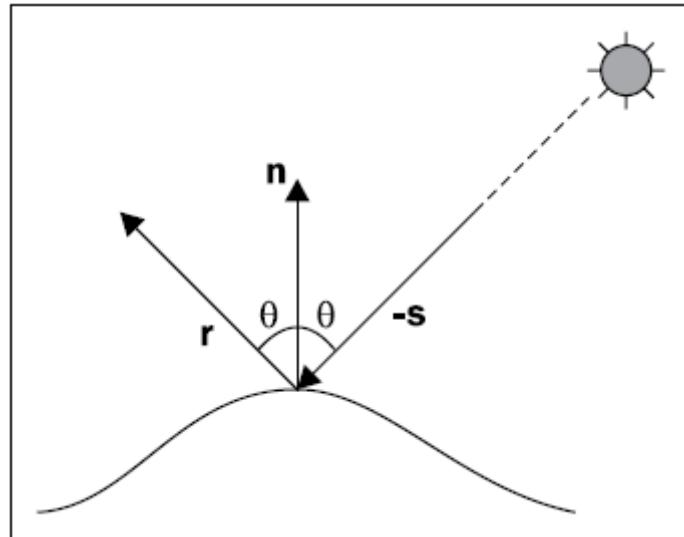
$$I = I_a + I_d + I_s$$

其中环境光 $I_a$ 最简单，等于材质表面\*光源，即 $I_a = L_a K_a$

而漫反射模拟了一个粗糙的表面，其上反射所有光，因此我们需要考虑光源的方向和表面的法向方向，即

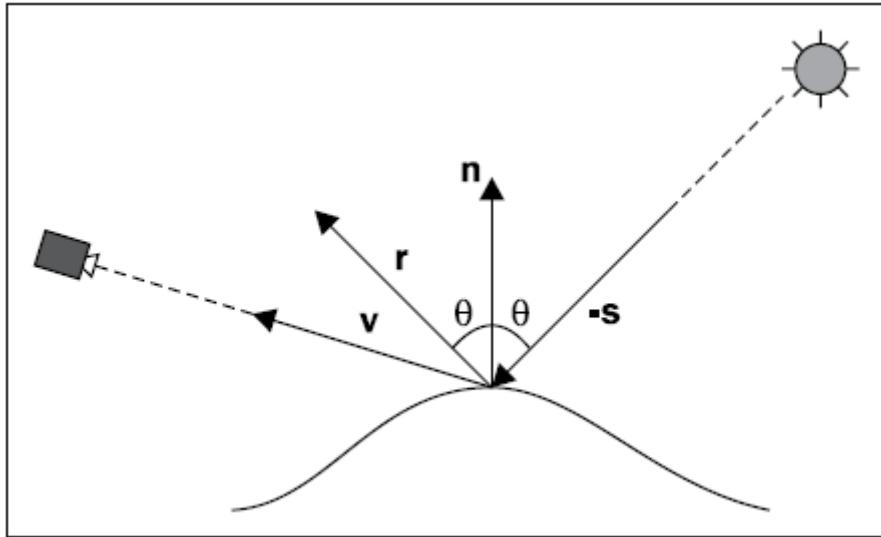
$$I_d = L_d K_d (S \cdot N)$$

最后是镜面反射，我们需要先严格地计算出光源方向到表面的镜面反射向量



其中 $r = -s + 2(s \cdot n)n$

在此基础上，我们考虑镜头的方向



得到

$$I_s K_s (r \cdot v)^f$$

参数 $f$ 是高光，可以看出， $f$ 越大，代表正对着光源的地方越亮，而偏离光源则会迅速衰减，也就是表面越光滑

具体我们使用GLSL语言实现，利用矩阵和向量，加上reflect函数。

而flat shading的区别在于如何渲染表面，我们只需要利用flag标识符标记着色器的输入输出，就可以在Phong的代码上实现。

Gouraud将计算过程放在顶点着色器，比起Phong来效率更高，不过对于当顶点数较少的表面，效果会差

Wavy使用一个时间相关函数，改变顶点位置，和动画中的粒子类似。

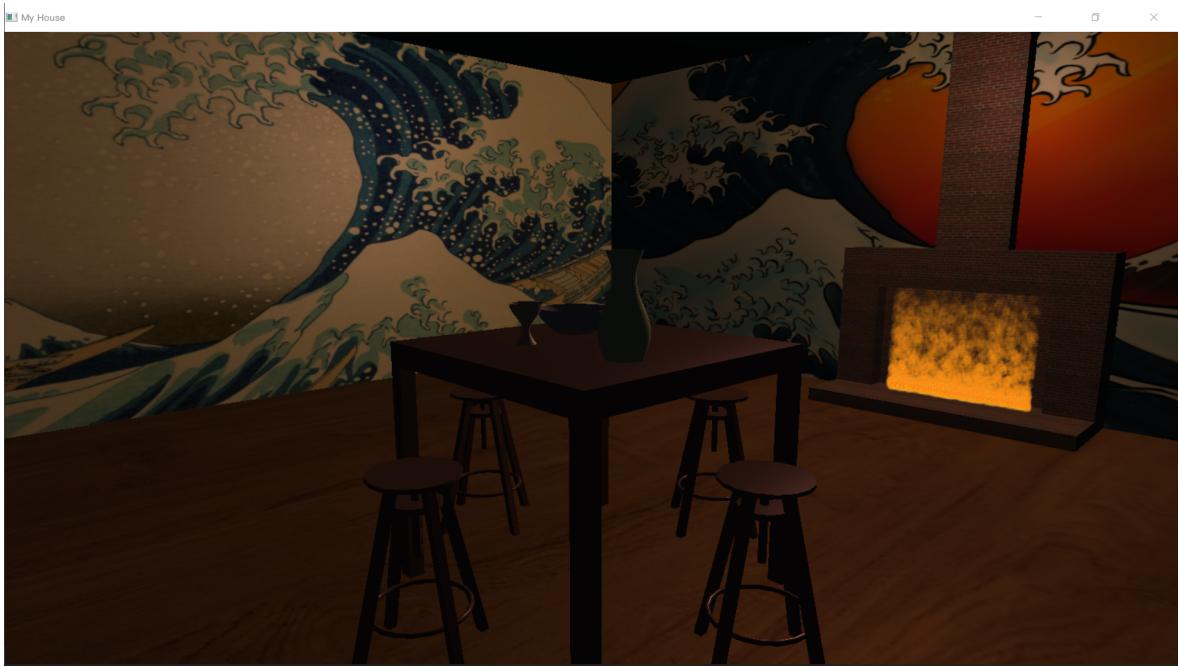
在此基础上，我们可以通过计算得到的 $I_a, I_d, I_s$ ，以及中间计算结果EyeNorm和LightReflect实现多种光照效果。

## 导入Obj

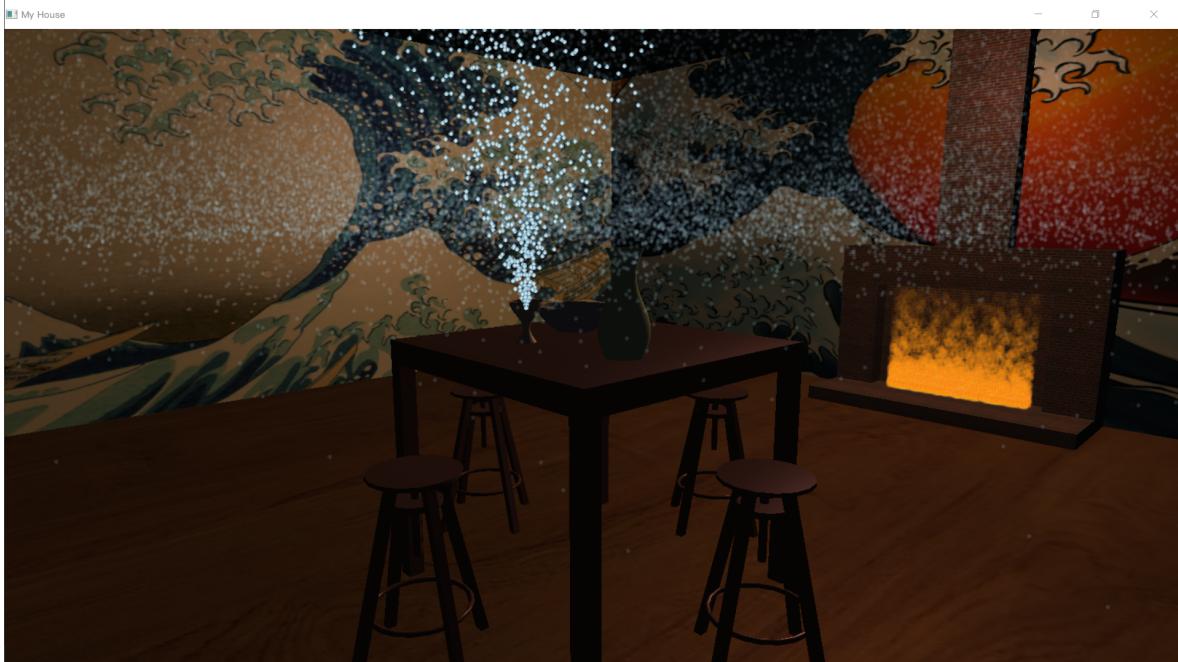
Obj文件格式比较简单，v表示点的坐标值，vt表示纹理贴图坐标，vn表示点的法线，f是面，也是我们绘制obj格式模型的最小单元，我们这里以三角面为例，表示点索引/纹理索引/法线索引

若要绘制不含材质和纹理的模型，我们只需要将文件中点的坐标值放入数组中，对f之后的空格和/进行正则替换，得到三个顶点的索引，在绘制每个面时，利用索引取得点的坐标，并计算面的法线，调用内置函数即可。

## 3. 效果



**Picture 1**



**Picture 2**



Picture 3



Picture 4(未整合)

更多功能见readme->options

#### 4. 参考

- David Wolff - *OpenGL 4.0 Shading Language Cookbook*
- *Learning Modern 3D Graphics Programming*
- *Learning OpenGL*

#### 5. 分工

All by YANG Rui.

