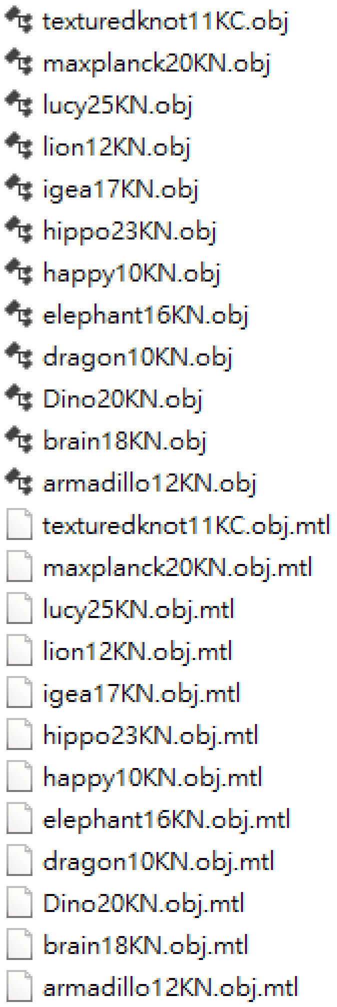
Notice

- Change color models to normal models.
- Traverse normal value instead of color value.
- The model is composed of different groups.
- A group is composed of a lot of triangles.
- Each group shares the same material.

How to traverse



texturedknot11KC.obj
maxplanck20KN.obj
lucy25KN.obj
lion12KN.obj
igea17KN.obj
hippo23KN.obj
happy10KN.obj
elephant16KN.obj
dragon10KN.obj
Dino20KN.obj
brain18KN.obj
armadillo12KN.obj
texturedknot11KC.obj.mtl
maxplanck20KN.obj.mtl
lucy25KN.obj.mtl
lion12KN.obj.mtl
igea17KN.obj.mtl
hippo23KN.obj.mtl
happy10KN.obj.mtl
elephant16KN.obj.mtl
dragon10KN.obj.mtl
Dino20KN.obj.mtl
brain18KN.obj.mtl
armadillo12KN.obj.mtl

How to traverse

```
GLMmodel* OBJ;
GLMgroup* group = OBJ->groups;
while (group)
{
        //Get material data here
        for (i = 0; i<(int)group->numtriangles; i++)
        {
                //Get OBJ data here
        }
        group = group->next;
}
```

Get material data

- OBJ->materials[group->material].ambient
- OBJ->materials[group->material].diffuse
- OBJ->materials[group->material].specular

Get triangle data

- ***Same method as Assigment # 01***

```
//Triangle index
int triangleID =group->triangles[i];
//the  index of each vertices
int indv1 = OBJ->triangles[triangleID].vindices[0];
int indv2 = OBJ->triangles[triangleID].vindices[1];
int indv3 =  OBJ->triangles[triangleID].vindices[2];
```

Use vertex normal for lighting

```
//the index of each normals
int indn1 = OBJ->triangles[triangleID].nindices[0];
int indn2 = OBJ->triangles[triangleID].nindices[1];
int indn3 = OBJ->triangles[triangleID].nindices[2];
// the vertex normal
OBJ->normals[indn1*3];
OBJ->normals[indn1*3+1];
OBJ->normals[indn1*3+2];
OBJ->normals[indn2*3];
OBJ->normals[indn2*3+1];
OBJ->normals[indn2*3+2];
OBJ->normals[indn3*3];
OBJ->normals[indn3*3+1];
OBJ->normals[indn3*3+2];
```

Shader code

- You need to add ambient ,diffuse , specular, and so on in your light equation.

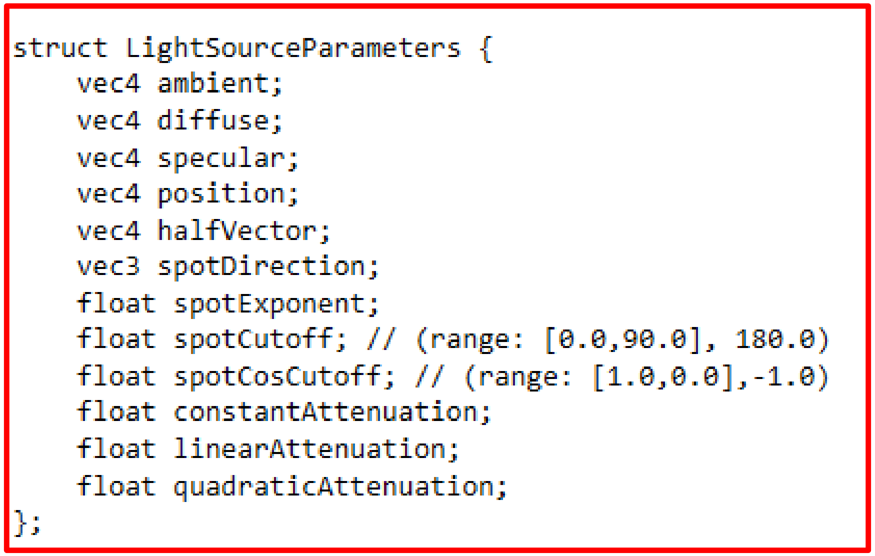
- In fact, there are so many lighting shader code in google. You can reference them, too.

- Lighthouse3d
  - http://www.lighthouse3d.com/tutorials/glsl-tutorial/lighting/

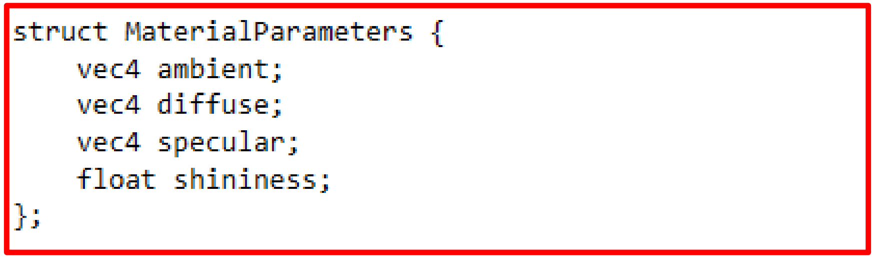
Hints

- 3 light source, each has their own parameter.
- Material parameter, Control variable
- MVP matrices
- A bunch of variable to pass…lol
- Well manage your variable before you go.
- Trace the framework to get more ideas

Hints



```
struct LightSourceParameters {
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    vec4 position;
    vec4 halfVector;
    vec3 spotDirection;
    float spotExponent;
    float spotCutoff; // (range: [0.0,90.0], 180.0)
    float spotCosCutoff; // (range: [1.0,0.0],-1.0)
    float constantAttenuation;
    float linearAttenuation;
    float quadraticAttenuation;
};
```



```
struct MaterialParameters {
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    float shininess;
};
```

What should we pass to shader?

- Transformation
  - Vertices position
  - mvp matrix
- Lighting
  - Normal vector
  - Model Transform matrix
  - Rotate matrix with transpose after inverse
  - Lighting parameters
  - Material parameters
  - Eye position (in world space)

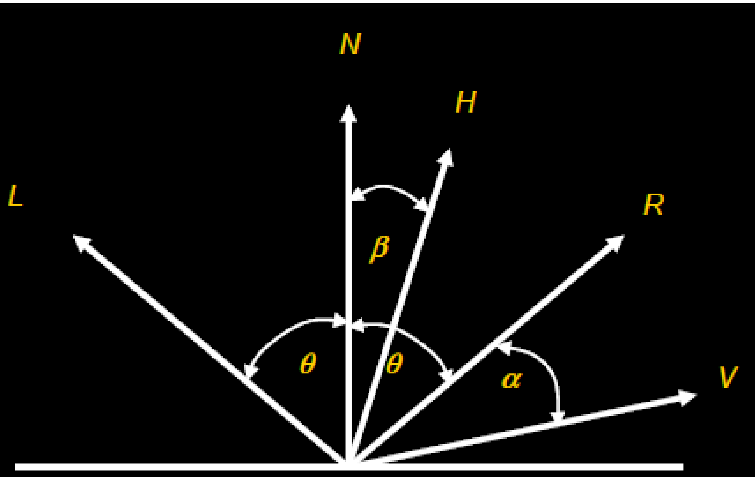
	Directional	Point	Spot
Ambient	O	O	O
Diffuse	O	O	O
Specular	O	O	O
Cut-off angle	X	X	O
Exp	X	X	O

$$I = I_a k_a + \sum_{p=1}^{m} f_p I_p \left(k_d (N \cdot L_p) + k_s (N \cdot H_p)^{n'}\right)$$

$$H = \frac{L+V}{|L+V|}$$

$$\theta + \beta = \theta - \beta + \alpha$$

$$\beta = \frac{1}{2}\alpha$$



- L : vector of vertex position to light source
- V : vector of vertex position to eye position
- vertex position : M * N * Vertices position
  (M = Model Transform matrix, N = Normalization matrix)
- N : R * Normal vector
  (R = Rotate matrix with transpose after inverse)