# 实验五：Phong模型

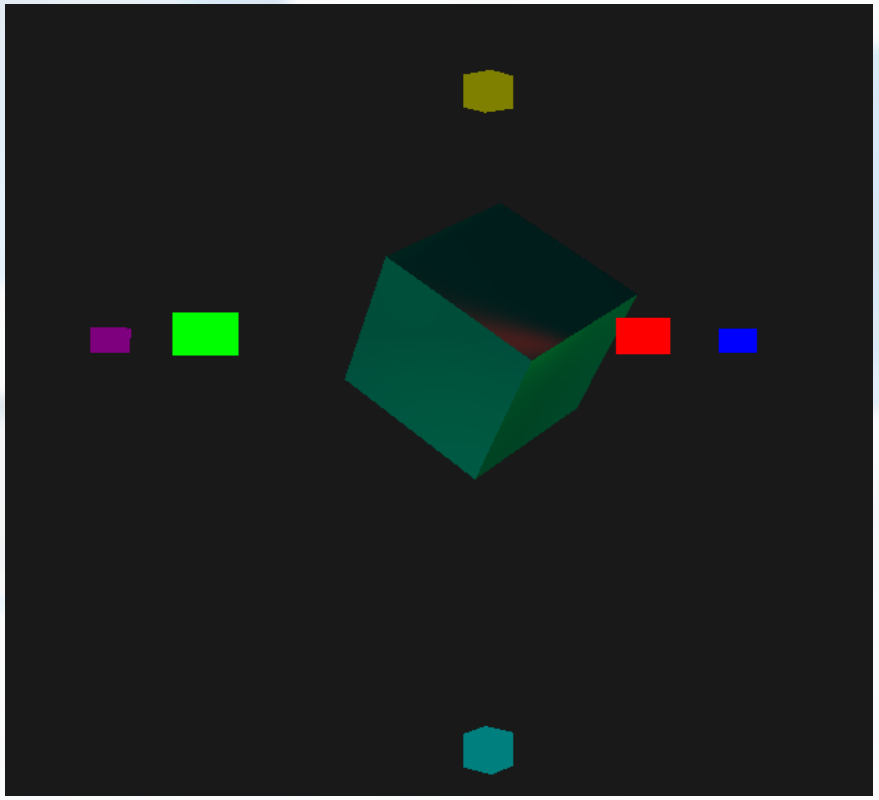
华中科技大学软件学院　万琳

# 提纲

1. 实验要求
2. 程序流程
3. 要点解析

## 1 实验要求

用Phong模型模拟光照效果：周围的六个小立方体是六个点光源，中间的立方体是被照射对象。这八个点光源包括：六个点光源、一个定向光、一个聚光光源。



# 2 程序流程


善后工作
初始化：初始化GLFW和GLAD
数据处理：生成和绑定VBO和VAO，设置属性指针
着色器：顶点和片段着色器
渲染

## 3 要点解析

➢问题分析

光源的设置

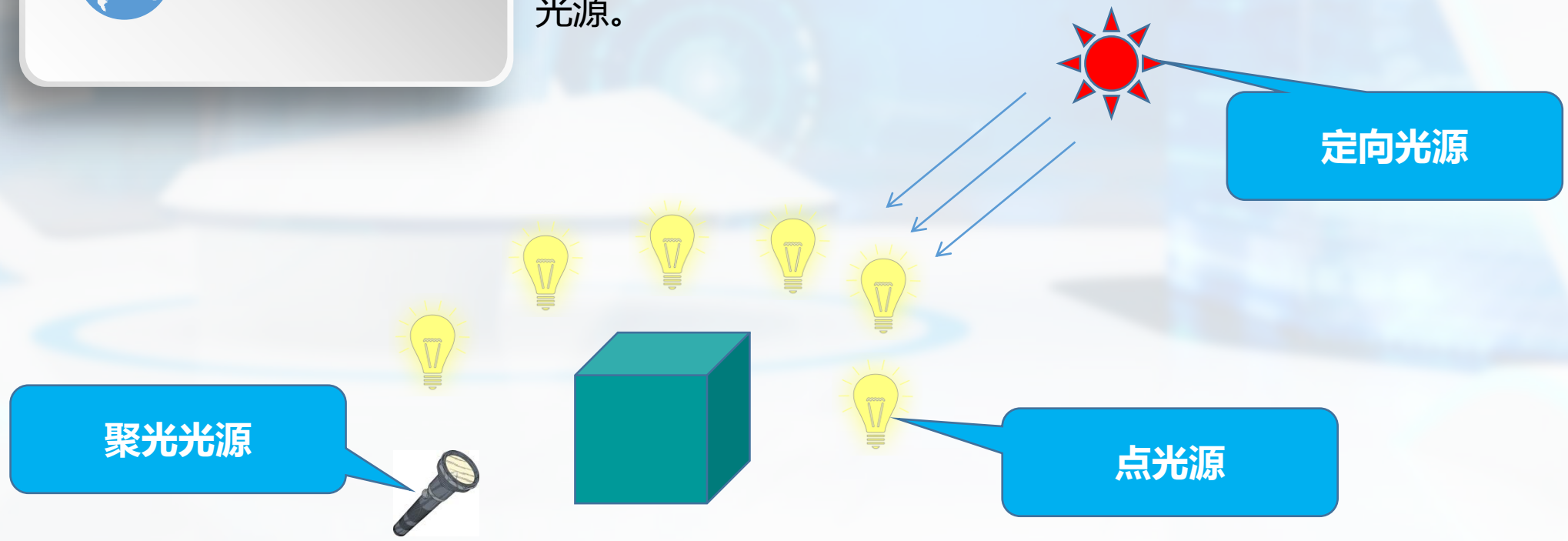
光照计算

# 3 要点解析

➢问题一：光源的设置

光源的设置

这八个点光源包括：六个点光源、一个定向光、一个聚光光源。



## 3 要点解析

➢问题一：光源的设置


```
DirectLight dirLight =
DirectLight(glm::vec3(-0.2f, -1.0f, -0.3f),
            glm::vec3(0.05f),
            glm::vec3(0.4f),
            glm::vec3(0.5f));
```

## 3 要点解析

➢问题一：光源的设置

点光源

```
// 点光源位置
glm::vec3 point_light_positions[] = {
    glm::vec3(1.0f, 0.0f, 0.0f),
    ......
};

// 光源颜色
glm::vec3 light_colors[] = {
    glm::vec3(1.0f, 0.0f, 0.0f),
    ......
};

PointLight pointLights    [6];

// 点光源
for (int i = 0; i < 6; i++)
    pointLights[i] = PointLight(point_light_positions[i],
                                0.05f * light_colors[i], 0.8f * light_colors[i],
                                light_colors[i], 1.0f, 0.09f, 0.032f);
```

## 3 要点解析

➢问题一：光源的设置

聚光光源

```
// 聚光
    SpotLight spotLight =
    SpotLight(camera.position, camera.forward,
              glm::vec3(0.0f), glm::vec3(1.0f), glm::vec3(1.0f),
              1.0f, 0.09f, 0.032f,
              cos(glm::radians(12.5f)),
              cos(glm::radians(15.0f)));
```

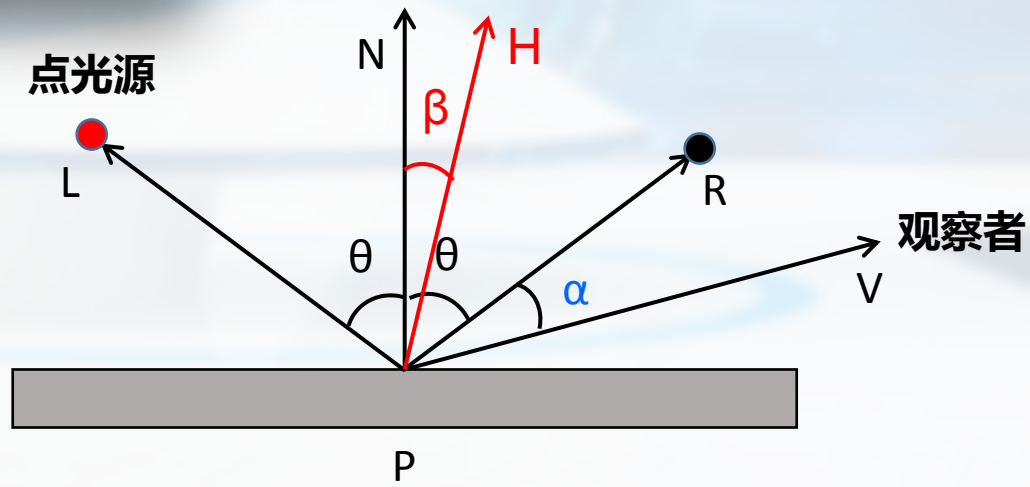
# 3 要点解析

➢问题二：光照的计算

光照计算

Blinn-Phong模型

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(H\cdot N)^n$$



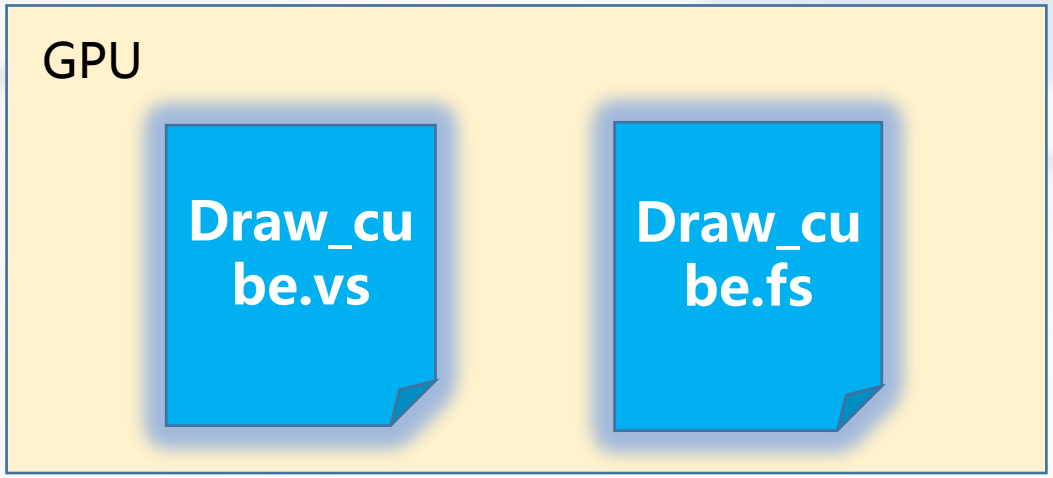
## 3 要点解析

➢问题二：光照的计算

光照计算

Blinn-Phong模型

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(H\cdot N)^n$$



## 3 要点解析

➢问题二：

光照计算

定向光源

```
// 计算定向光
vec3 CalcDirLight(DirLight light, vec3 normal, vec3 viewDir)
{
    if(!light.on) {
        return vec3(0.0);
    }
    // 漫反射
    vec3 lightDir = normalize(-light.direction);
    float diff = max(dot(normal, lightDir), 0.0);
    // 镜面反射
    vec3 reflectDir = reflect(-lightDir, normal);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
    // 在漫反射光下物体颜色
    vec3 diffuseColor = vec3(material.diffuse);
    // 计算环境光，漫反射光和镜面光
    vec3 ambient = light.ambient * diffuseColor;
    vec3 diffuse = light.diffuse * diff * diffuseColor;
    vec3 specular = light.specular * spec * vec3(material.specular);
    return ambient + diffuse + specular;
}
```

## 3 要点解析

➢问题二：光照计算

点光源

```
// 计算点光源
vec3 CalcPointLight(PointLight light, vec3 normal, vec3 fragPos, vec3 viewDir)
{
    ......
// 距离和衰减
    float d = length(light.position - fragPos);
    float attenuation = 1.0 / (light.c + light.l * d + light.q * d * d);
// 计算环境光，漫反射光和镜面光
    vec3 ambient = light.ambient * diffuseColor;
    vec3 diffuse = light.diffuse * diff * diffuseColor;
    vec3 specular = light.specular * spec * vec3(material.specular);
    return (ambient + diffuse + specular) * attenuation;
}
```

## 3 要点解析

➢问题二：

光照计算

聚光光源

```
// 计算聚光
vec3 CalcSpotLight(SpotLight light, vec3 normal, vec3 fragPos, vec3 viewDir)
{
// 聚光强度
    float theta = dot(lightDir, normalize(-light.direction));
    float epsilon = light.cutOff - light.outerCutOff;
    float intensity = clamp((theta - light.outerCutOff) / epsilon, 0.0, 1.0);
// 计算环境光，漫反射光和镜面光
    vec3 ambient = light.ambient * diffuseColor;
    vec3 diffuse = diff * light.diffuse * diffuseColor;
    vec3 specular = vec3(material.specular) * spec * light.specular;
    // 乘聚光强度为了避免光照过强，平滑聚光边缘
    return (ambient + (diffuse + specular) * intensity) * attenuation;
}
```


谢谢
软件学院　万琳