OpenGL 光照模型

姓名：雷翔
学号：2053932
日期：2023.12.25

1 开发环境

- **代码编写**: 使用 macOS 系统上的 Visual Studio Code 编辑器。
- **编译环境**: 在 Windows 10 虚拟机中使用 Visual Studio 进行编译。

win 系统和 mac 系统，尤其注意路径 `\` 和 `/` 区别！！！

2 `model.cpp`

在渲染循环中，将光源的位置和颜色传递给着色器。

```
// ************ 修改 ************
// 定义光源位置
glm::vec3 lightPos = glm::vec3(1.2f, 1.0f, 2.0f);
// 定义光源颜色
glm::vec3 lightColor = glm::vec3(1.0f, 1.0f, 1.0f); // 白色光源
// 定义第二个光源位置
glm::vec3 lightPos2 = glm::vec3(-1.2f, 1.0f, 2.0f);
// 定义第二个光源颜色
glm::vec3 lightColor2 = glm::vec3(1.0f, 0.0f, 0.0f); // 红色光源
// ************ 修改 ************

// xxx

// render loop
while (!glfwWindowShouldClose(window))
{
    /// xxx

    // ************ 修改 ************
    // 将光源位置传递到着色器
    ourShader.setVec3("lightPos", lightPos);
    // 将光源颜色传递到着色器

```

```
    ourShader.setVec3("lightColor", lightColor);
    // 第二个光源
    ourShader.setVec3("lightPos2", lightPos2);
    ourShader.setVec3("lightColor2", lightColor2);
    // ************ 修改 ************

    // xxx
}
```

3 model.h

macOS 和 Windows 在处理文件路径时的区别。

```
// ************ 修改 ************
directory = path.substr(0, path.find_last_of('\\'));  // for mac
// ************ 修改 ************

// ************ 修改 ************
filename = directory + '\\' + filename;  //  for mac
// ************ 修改 ************
```

4 modelVS.vs 顶点着色器

计算片段的世界空间位置和法线，传递纹理坐标。

```
#version 330 core
// ************ 修改 ************
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
layout (location = 2) in vec2 aTexCoords;

out vec3 FragPos;   // 片段的世界空间位置
out vec3 Normal;    // 片段的法线
out vec2 TexCoords; // 纹理坐标（将纹理坐标传递给片段着色器）

uniform mat4 model;       // 模型矩阵
uniform mat4 view;        // 视图矩阵
uniform mat4 projection;  // 投影矩阵

void main()
{
    FragPos = vec3(model * vec4(aPos, 1.0)); // 将顶点位置转换到世界空间
```

```
    Normal = mat3(transpose(inverse(model))) * aNormal; // 转换法线到世界空间
    TexCoords = aTexCoords;  // 传递纹理坐标
    gl_Position = projection * view * vec4(FragPos, 1.0);  // 计算顶点的裁剪空间位置
}
// ************ 修改 ************
```

5 modelFS.fs 片面着色器

实现 Phong Lighting Model，包括环境光、漫反射和镜面反射。

```
#version 330 core
// ************ 修改 ************
out vec4 FragColor;

in vec3 FragPos;  // 从顶点着色器传来的片段位置
in vec3 Normal;   // 从顶点着色器传来的法线
in vec2 TexCoords; // 从顶点着色器传来的纹理坐标

uniform sampler2D texture_diffuse1; // 纹理
uniform vec3 viewPos;               // 观察者位置
uniform vec3 lightPos;              // 光源位置
uniform vec3 lightColor;            // 光源颜色
uniform vec3 lightPos2;             // 第二个光源位置
uniform vec3 lightColor2;           // 第二个光源颜色

void main()
{
    // Phong Lighting Model
    // 环境(Ambient)、漫反射(Diffuse)和镜面(Specular)光照

    // 环境光
    float ambientStrength = 0.4f;
    vec3 ambient = ambientStrength * lightColor;

    // 漫反射光照
    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPos - FragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    // 镜面反射光照
    float specularStrength = 0.5f;
```

```
    vec3 viewDir = normalize(viewPos - FragPos);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
    vec3 specular = specularStrength * spec * lightColor;

    // 漫反射光照 - 第二个光源
    vec3 lightDir2 = normalize(lightPos2 - FragPos);
    float diff2 = max(dot(norm, lightDir2), 0.0);
    vec3 diffuse2 = diff2 * lightColor2;

    // 镜面反射光照 - 第二个光源
    vec3 reflectDir2 = reflect(-lightDir2, norm);
    float spec2 = pow(max(dot(viewDir, reflectDir2), 0.0), 32);
    vec3 specular2 = specularStrength * spec2 * lightColor2;

    // 纹理颜色
    vec3 textureColor = texture(texture_diffuse1, TexCoords).rgb;

    // 结合两个光源的光照效果和纹理颜色
    vec3 result = (ambient + diffuse + specular + diffuse2 + specular2) * textureColor;
    FragColor = vec4(result, 1.0);
}
// ************ 修改 ************
```

如有任何问题或建议，请通过邮箱2053932@tongji.edu.cn与我联系。