《计算机图形学》实验报告

Add Texture to Solar System

报告日期：2023. 12. 11

1. 环境配置

操作系统：windows 10

开发工具： vscode

编译器：MinGW 6.3.0 (GCC-6.3.0-1)

OpenGL库：`glfw3,glew,glm,opengl32,glu32`

2. 实验思路

本次实验其实就是给上次做好的太阳系添加纹理，我们通过纹理贴图的方式来实现背景和球体的贴图。

2.1 纹理天空盒（bonus）

虽然是bonus，但是这个背景的实现网上其实有很多背景盒的实现参考，具体来说，我们只需要利用`stbi_load`读取六张背景图片，并根据设置好的各个纹理坐标实现我们的贴图。而对于背景的着色器设计也是相当简单的，我们甚至不用考虑任何光照信息，输入纹理坐标输入对应颜色即可。

可能最需要注意的就是我们在绘制纹理天空盒的时候需要关闭深度信息检测，这样才能起到背景的效果。

2.2 球体纹理贴图

我们对以下几个文件进行了修改：

2.2.1 `sphere.h` 和 `sphere.cpp`

在之前的作业中，我们并未考虑到后续的贴图，因此我们需要在原来的球体类中添加纹理贴图的坐标信息并将其绑定到Buffer中，代码如下所示：

```
for (int y = 0;y <= Y_SEGMENTS;y++){
    for (int x = 0;x <= X_SEGMENTS;x++){
        float xSegment = (float)x / (float)X_SEGMENTS;
        float ySegment = (float)y / (float)Y_SEGMENTS;
```

```
        // printf("%f %f\n",xSegment,ySegment);
        float xPos = std::cos(xSegment * 2.0f * PI) * std::sin(ySegment * PI) * radius;
        float yPos = std::cos(ySegment * PI) * radius;
        float zPos = std::sin(xSegment * 2.0f * PI) * std::sin(ySegment * PI) * radius;
        sphereVertices.push_back(xPos);
        sphereVertices.push_back(yPos);
        sphereVertices.push_back(zPos);
        // 计算纹理坐标
        textureCoords.push_back(1- xSegment); // 直接用xSegment会出现纹理颠倒的情况
        textureCoords.push_back(ySegment);
    }
}
```

同时，我们在进行最后绘制的同时，也需要将得到的纹理进行绑定，再进行最后的绘制。

```
void Sphere::draw(Shader &shader_ ,glm::mat4 & projection, glm::mat4 &view, glm::mat4
&model,unsigned int texture){
    shader_.use();

 shader_.set_3trans_matrix(glm::value_ptr(projection),glm::value_ptr(view),glm::value_ptr(mode
l));
    glBindVertexArray(VAO);
    // 添加绑定纹理的部分----------------------
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, texture);
    //---------------------------------------
    glDrawElements(GL_TRIANGLES, sphereIndices.size(), GL_UNSIGNED_INT, 0);
}
```

2.2.2 `lightingshader.fs` 和 `lightingshader.vs`

我们在之前绘制球体的着色器中没有考虑纹理的部分，现在加入了纹理信息，我们也要在着色器中补充这一部分：

```
/*lightingshader.fs*/
// Fetch the color from the texture
vec4 currentColor = texture(ourTexture, TexCoord);
```

```
/*lightingshader.vs*/
#version 330 core

layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
layout (location = 2) in vec2 aTexCoord;

out vec3 FragPos;
out vec3 Normal;
out vec3 objColor;  // For flat shading
out vec2 TexCoord;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main(){
```

```
    // Calculate transformed position in world space
    FragPos = vec3(model * vec4(aPos, 1.0));

    // Calculate transformed normal in world space
    Normal = mat3(transpose(inverse(model))) * aNormal;

    // Calculate screen-space position
    gl_Position = projection * view * vec4(FragPos, 1.0);

    // Pass object color and texture coordinates to the fragment shader
    TexCoord = aTexCoord;
}
```

2.2.3 添加 `load_texture`

对于纹理贴图的加载，我们直接参考 `learningopengl` 教程中的方式，通过 `stb_image.h` 标准库来实现纹理贴图的加载。对 `stb_image.h` 的调用方式如下：

```
//stb_image.cpp
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
```

对于纹理加载函数，如下所示：

```
unsigned int load_texture(char const* path){
    unsigned int textureID;
    glGenTextures(1, &textureID);

    int width, height, nrComponents;
    unsigned char* data = stbi_load(path, &width, &height, &nrComponents, 0);
    if (data){
        GLenum format;
        if (nrComponents == 1)
            format = GL_RED;
        else if (nrComponents == 3)
            format = GL_RGB;
        else if (nrComponents == 4)
            format = GL_RGBA;

        // 绑定纹理并设置参数
        glBindTexture(GL_TEXTURE_2D, textureID);
        glTexImage2D(GL_TEXTURE_2D, 0, format, width, height, 0, format, GL_UNSIGNED_BYTE,
data);
        glGenerateMipmap(GL_TEXTURE_2D);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        // 释放图像数据内存
        stbi_image_free(data);
    }
    else{
        std::cout << "Texture failed to load at path: " << path << std::endl;
        stbi_image_free(data);
```

```
    }
    return textureID;
}
```

3 效果展示