计算机图形学报告

余婉莹 SA21168286

December 15, 2021

1 依赖包

- glad github 当前稳定版
- glfw github 当前稳定版
- opengl version 3.3
- glm version 0.9.8

[1]

2 实现功能

- 从给定的 obj 文件读入网格模型的顶点位置和颜色数据，显示在屏幕上。
- 上下左右方向键或 WSAD 键移动模型；
- 鼠标左键采用虚拟跟踪球方法旋转模型（不动点可选为质心）
- z 或 Z 键缩小或放大模型（把模型拉远或拉近照相机）
- ESC 键退出程序
- 鼠标滚动缩小或放大模型（把模型拉远或拉近照相机）

[1] 由于自身基础为零，对 cmake 没有深入学习以及自身 C++11 也没有进一步的了解其特性，在 codeblock 实现功能过程中出现配置问题，但始终无法修复，同时自身比较习惯使用 clion 进行编程，glew 由于采用需在 cmd 中调试，clion 中无法直接运行故配置过程出现问题，在其他同学的指导下选择采用 glad+glfw 进行功能实现。

3 代码解释

3.1 obj 加载器结构体

```
struct vertex{
    glm::vec3 position;
    glm::vec3 color;
};

    std::vector<vertex> V;
    std::vector<unsigned int> S;
```

3.2 obj 加载实现主要部分

```
try{
        objFile.open(filename);
        std::stringstream objStream;
        while(!objFile.eof()&&getline(objFile,line)){
            if(line.length()<=0)
                continue;
            std::istringstream in(line);
            if(line[0]=='v'){
                if(line[1]=='c'){
                    in>>head>>V[i].color.x>>V[i].color.y>>V[i].color.z;
                    i++;
                }else{
                    p = new vertex();
                    in>>head>>p->position.x>>p->position.y>>p->
                        position.z;
                    V.push_back(*p);
                }
            }else if(line[0]=='f'){
                j+=3;
                in>>head>>a>>b>>c;
                S.push_back(a-1);
                S.push_back(b-1);
                S.push_back(c-1);
```

```
        }

    }
```

3.3 shader 类主要功能

- 读取着色器文件
- 对不同数据类型 uniform 变量设置

```
class shader {
public:
    ~shader()= default;
    unsigned int ID;
    shader(const GLchar* vertexPath,const GLchar* fragmentPath);
    void use();
    void setBool(const std::string &name , bool value) const;
    void setInt(const std::string &name , int value) const;
    void setFloat(const std::string &name , float value) const;
    void setVec2(const std::string &name, const glm::vec2 &value) const
        ;
    void setVec2(const std::string &name, float x, float y) const;
    void setVec3(const std::string &name, const glm::vec3 &value) const
        ;
    void setVec3(const std::string &name, float x, float y, float z)
        const;
    void setVec4(const std::string &name, const glm::vec4 &value) const
        ;
    void setVec4(const std::string &name, float x, float y, float z,
        float w);
    void setMat2(const std::string &name, const glm::mat2 &mat) const;
    void setMat3(const std::string &name, const glm::mat3 &mat) const;
    void setMat4(const std::string &name, const glm::mat4 &mat) const;


private:
    void checkCompileErrors(unsigned int shader,std::string type);
```

};

3.4 摄像机类

- 初始化 view 矩阵数据
- 根据各类操作修改 view 矩阵数据并规范化

```
camera(glm::vec3 position=glm::vec3(0.0f,0.0f,0.0f),glm::vec3 up=
    glm::vec3(0.0f,1.0f,0.0f), float  yaw = YAW,float pitch = PITCH
    );
camera(float posX, float posY, float posZ, float upX, float upY,
    float upZ, float yaw, float pitch);
glm::mat4 GetViewMatrix();
void ProcessKeyboard(Camera_Movement direction, float deltaTime)
    ;
void ProcessMouseMovement(float xoffset, float yoffset, GLboolean
    constrainPitch = true);
void ProcessMouseScroll(float yoffset);
```

3.5 键盘按键功能实现

```
void processInput(GLFWwindow *window)
{
    if (glfwGetKey(window, GLFW_KEY_ESCAPE) ==
        GLFW_PRESS)
        glfwSetWindowShouldClose(window, true);
    if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS||
        glfwGetKey(window, GLFW_KEY_UP) == GLFW_PRESS)
        c.ProcessKeyboard(FORWARD, deltaTime);
    if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS||
        glfwGetKey(window, GLFW_KEY_DOWN) ==
        GLFW_PRESS)
        c.ProcessKeyboard(BACKWARD, deltaTime);
    if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS||
        glfwGetKey(window, GLFW_KEY_RIGHT) ==
        GLFW_PRESS)
```

```
        c.ProcessKeyboard(LEFT, deltaTime);
    if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS||
        glfwGetKey(window, GLFW_KEY_LEFT) == GLFW_PRESS)
        c.ProcessKeyboard(RIGHT, deltaTime);
    if(glfwGetKey(window,GLFW_KEY_Z)==GLFW_PRESS){
        if(glfwGetKey(window,GLFW_KEY_LEFT_SHIFT)==
            GLFW_PRESS||glfwGetKey(window,
            GLFW_KEY_RIGHT_SHIFT)==GLFW_PRESS){
            c.ProcessMouseScroll(-0.005f);
        }else{
            c.ProcessMouseScroll(0.005f);
        }
    }
}
```

3.6 鼠标左键点击

```
void mouse_button_callback(GLFWwindow* window, int button, int
    action, int mods)
{
    if (action == GLFW_PRESS) switch(button)
        {
            case GLFW_MOUSE_BUTTON_LEFT:
            {
                Click= true;
                firstMouse= true;
            }
                break;
            case GLFW_MOUSE_BUTTON_MIDDLE:
                break;
            case GLFW_MOUSE_BUTTON_RIGHT:
                break;
            default:
                return;
        }else{
        Click = false;
```

```
    }
}
```

3.7 鼠标移动捕捉

- 点击初始化位置坐标
- 只有点击状态才能移动图像

```
void mouse_callback(GLFWwindow* window, double xpos, double ypos)
{
    if (firstMouse)
    {
        lastX = xpos;
        lastY = ypos;
        firstMouse = false;
    }
    if (!Click){
        return;
    }
    float xoffset = xpos - lastX;
    float yoffset = lastY - ypos; // reversed since y-coordinates go
        from bottom to top
    lastX = xpos;
    lastY = ypos;

    c.ProcessMouseMovement(xoffset,yoffset);
}
```

3.8 鼠标滚动功能

- 滚动放大缩小（摄像头远近）

```
void scroll_callback(GLFWwindow* window, double xoffset, double
    yoffset)
{
    c.ProcessMouseScroll(yoffset);
}
```

4 结果展示

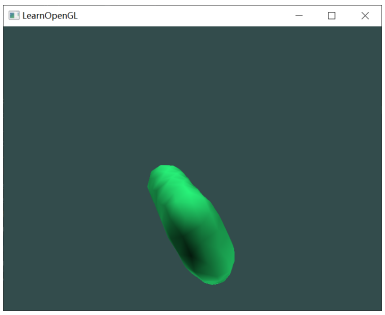

图 1: V 模型

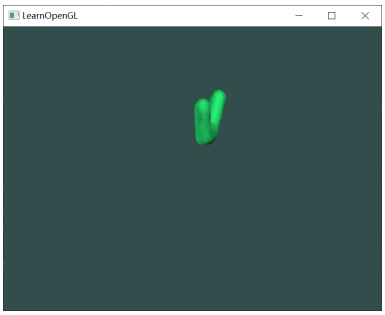

图 2: V 模型

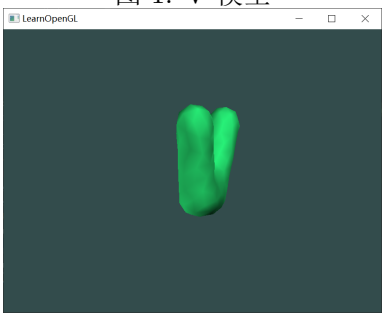

图 3: V 模型

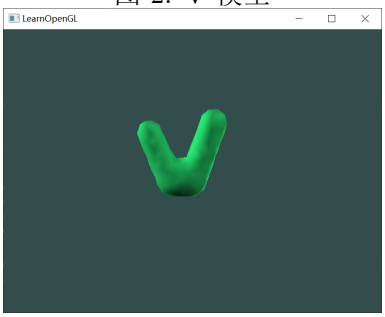

图 4: V 模型

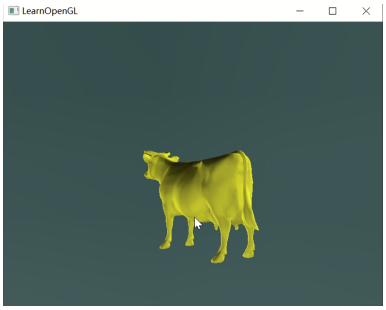

图 5: 模型牛

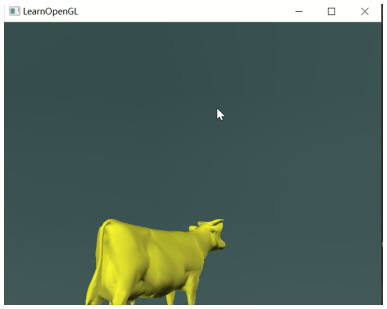

图 6: 模型牛

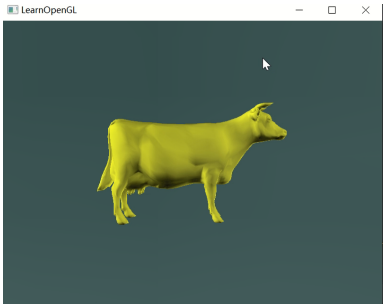

图 7: 模型牛

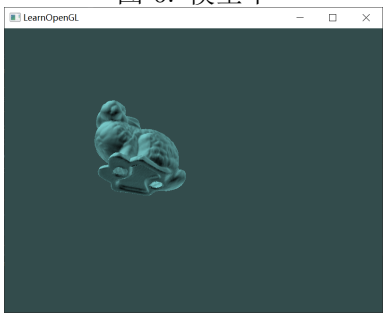

图 8: 模型兔

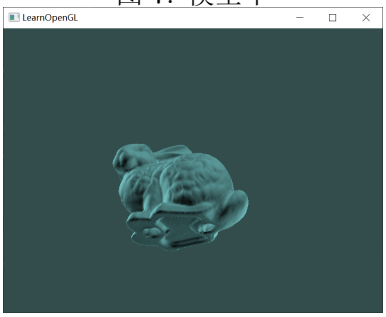

图 9: 模型兔

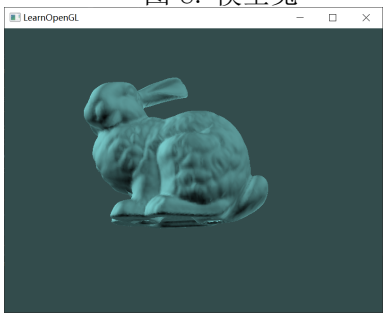

图 10: 模型兔