专业：计算机科学与技术
姓名：方彦祺
学号：3220102829
日期：2024.11.19
地点：

浙江大学实验报告

课程名称：计算机图形学　指导老师：童若锋　成绩：
实验名称：OpenGL 摄像机　实验类型：基础实验　同组学生姓名：无

装 订 线

一、实验目的和要求

掌握四元数表示旋转的原理，编程完成正交投影和透视投影的四元数相机，并能使用键盘移动观察相机，鼠标改变观察视角，以实现 FPS 风格的相机控制。

二、实验内容和原理

四元数是一种扩展了复数的代数结构，常用于表示旋转。通过旋转轴和旋转角度，可以使用四元数表示 3D 空间中的旋转。

三、主要仪器设备

Visual Studi 2022

四、操作方法和实验步骤

```
if (_input.keyboard.keyStates[GLFW_KEY_W] != GLFW_RELEASE) {
    std::cout << "W" << std::endl;
    // TODO: move the camera in its front direction
    // write your code here
    // ----------------------------------------------
    // camera->transform.position = ...;
    // ----------------------------------------------
    camera->transform.position += cameraMoveSpeed * camera->transform.getFront();
}
```

通过检查 GLFW_KEY_W 键的状态来判断 W 键是否被按下。如果 W 键被按下，执行以下操作：
输出 "W" 到控制台。
相机的位置沿着相机前方的方向（camera->transform.getFront()）前进。cameraMoveSpeed 控制移动的速度。

```
if (_input.keyboard.keyStates[GLFW_KEY_A] != GLFW_RELEASE) {
    std::cout << "A" << std::endl;
    // TODO: move the camera in its left direction
    // write your code here
    // ------------------------------------------------
    // camera->transform.position = ...;
    // ------------------------------------------------
    camera->transform.position -= cameraMoveSpeed * camera->transform.getRight();
}
```

通过检查 GLFW_KEY_A 键的状态来判断 A 键是否被按下。如果 A 键被按下，执行以下操作：
输出 "A" 到控制台。
相机的位置沿着相机的右方向（camera->transform.getRight()）向左移动。

```
if (_input.keyboard.keyStates[GLFW_KEY_S] != GLFW_RELEASE) {
    std::cout << "S" << std::endl;
    // TODO: move the camera in its back direction
    // write your code here
    // ------------------------------------------------
    // camera->transform.position = ...;
    // ------------------------------------------------
    camera->transform.position -= cameraMoveSpeed * camera->transform.getFront();
}
```

通过检查 GLFW_KEY_S 键的状态来判断 S 键是否被按下。如果 S 键被按下，执行以下操作：
输出 "S" 到控制台。
相机的位置沿着相机前方的反方向（camera->transform.getFront()）后退。

```
if (_input.keyboard.keyStates[GLFW_KEY_D] != GLFW_RELEASE) {
    std::cout << "D" << std::endl;
    // TODO: move the camera in its right direction
    // write your code here
    // ------------------------------------------------
    // camera->transform.position = ...;
    // ------------------------------------------------
    camera->transform.position += cameraMoveSpeed * camera->transform.getRight();
}
```

通过检查 GLFW_KEY_D 键的状态来判断 D 键是否被按下。如果 D 键被按下，执行以下操作：
输出 "D" 到控制台。
相机的位置沿着相机的右方向（camera->transform.getRight()）向右移动。

```
if (_input.mouse.move.xNow != _input.mouse.move.xOld) {
    std::cout << "mouse move in x direction" << std::endl;
    // TODO: rotate the camera around world up: glm::vec3(0.0f, 1.0f, 0.0f)
    // hint1: you should know how do quaternion work to represent rotation
    // hint2: mouse_movement_in_x_direction = _input.mouse.move.xNow - _input.mouse.move.xOld
    // write your code here
    // -------------------------------------------------------------------------
    // camera->transform.rotation = ...
    // -------------------------------------------------------------------------
    float mouseMovementX = _input.mouse.move.xNow - _input.mouse.move.xOld;
    glm::quat rotation = glm::angleAxis(-cameraRotateSpeed * mouseMovementX, glm::vec3(0.0f, 1.0f, 0.0f));
    camera->transform.rotation = glm::normalize(rotation * camera->transform.rotation);
}
```

如果鼠标在 x 轴方向上有移动（_input.mouse.move.xNow 和_input.mouse.move.xOld 不相等），则执行以下操作：

输出 "mouse move in x direction" 到控制台。

计算鼠标在 x 轴上的移动量。

使用 glm::angleAxis 创建一个绕世界坐标系的 Y 轴（即 glm::vec3(0.0f, 1.0f, 0.0f)）旋转的四元数。旋转的角度与鼠标的 x 轴移动量和 cameraRotateSpeed 有关。

更新相机的旋转，使得当前旋转应用到原有旋转上。通过四元数乘法实现，glm::normalize 确保旋转四元数的规范化，避免旋转的累积误差。

```
if (_input.mouse.move.yNow != _input.mouse.move.yOld) {
    std::cout << "mouse move in y direction" << std::endl;
    // TODO: rotate the camera around its local right
    // hint1: you should know how do quaternion work to represent rotation
    // hint2: mouse_movement_in_y_direction = _input.mouse.move.yNow - _input.mouse.move.yOld
    // write your code here
    // -------------------------------------------------------------------------
    // camera->transform.rotation = ...
    // -------------------------------------------------------------------------
    float mouseMovementY = _input.mouse.move.yNow - _input.mouse.move.yOld;
    glm::quat rotation = glm::angleAxis(-cameraRotateSpeed * mouseMovementY, camera->transform.getRight());
    camera->transform.rotation = glm::normalize(camera->transform.rotation * rotation);
}
```

如果鼠标在 y 轴方向上有移动（_input.mouse.move.yNow 和_input.mouse.move.yOld 不相等），则执行以下操作：

输出 "mouse move in y direction" 到控制台。

计算鼠标在 y 轴上的移动量。

使用 glm::angleAxis 创建一个绕相机右方向（camera->transform.getRight()）的旋转四元数。旋转的角度与鼠标的 y 轴移动量和 cameraRotateSpeed 有关。

更新相机的旋转，使得当前旋转应用到原有旋转上。同样，使用四元数乘法，并规范化旋转。

五、实验结果与分析

见可执行程序。

六、讨论、心得

本次实验内容较为简单，只要理解了四元数及 camera 在 OpenGL 中的原理即可根据提示完成实验代码。

七、参考链接

[1] 摄像机 - LearnOpenGL CN (learnopengl-cn.github.io)

[2] 四元数(Quaternions) - 知乎 (zhihu.com)