深 圳 大 学 实 验 报 告

课程名称：计算机图形学

实验项目名称：实验二 三维模型读取与控制

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：周虹

报告人：吴嘉楷 学号：2022150168 班级：国际班

实验时间：2024 年 09 月 24 日 -- 2024 年 11 月 04 日

实验报告提交时间：2024 年 10 月 21 日

教务部制

实验目的与要求：

1. 熟悉 OpenGL 三维模型的读取与处理；理解三维模型的基本变换操作；掌握鼠标键盘交互控制逻辑；掌握着色器中 uniform 关键字的使用以及数据传输方法。
2. OFF 格式三维模型文件的读取：完成对 OFF 格式三维模型文件的读取与显示，可改变物体的显示颜色。
3. 三维模型的旋转动画：结合模型进行旋转变换的过程，为模型添加自动的旋转动画。
4. 键盘鼠标的交互：通过键盘设定选择绕 x、y、z 轴进行旋转，鼠标左右键控制动画的开始与暂停。

实验过程及内容：

1. Copy 实验 2.3 的项目代码作为基础代码，在此基础上进行改动

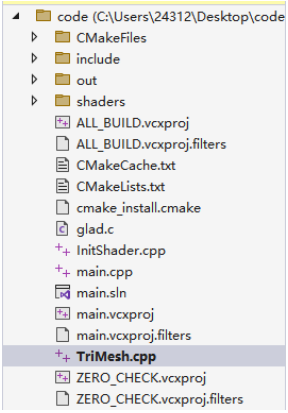

图 1 基础代码的目录结构

2. 使用实验二的 Model 文件夹提供的 off 文件，以绘制牛的图像

（1）将 Models 文件拷贝到 TriMess.cpp 的同级目录下

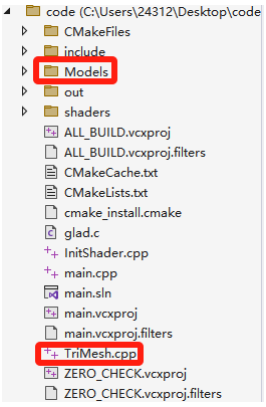

图 2.1 拷贝 Models 文件夹

（2）更改 init 函数以读入 cow.off 文件

```
void init()
{
    std::string vshader, fshader;
    // 读取着色器并使用
    vshader = "shaders/vshader.glsl";
    fshader = "shaders/fshader.glsl";

    //cube->generateCube(); 修改为下面的代码
    cube->readOff("Models/cow.off"); // 读取牛的模型文件

    bindObjectAndData(cube, cube_object, vshader, fshader);
    // 黑色背景
    glClearColor(0.0, 0.0, 0.0, 1.0);
}
```

图 2.2 读入 cow.off 模型文件

代码说明：

TriMesh 类中封装了两个关键方法：generateCube 和 readOff。其中，generateCube 方法专门用于生成立方体的几何表示，而 readOff 方法则负责读取 OFF 文件格式的模型数据。鉴于本次实验聚焦于绘制牛的图形，因此，在初始化函数 init 中，我需要使用 readOff 方法来加载 cow.off 模型文件，以便准确获取并呈现牛的三维模型数据。

3. 修改牛的颜色

（1）在 TriMesh.cpp 中定义一个牛的基准颜色 cow_colors

```
// 定义牛的基准颜色
const glm::vec3 cow_colors = glm::vec3(1.0, 0.5, 0.0);
```

图 3.1 定义 cow_colors

（2）修改 vertex_colors 数组的值

在 readOff 函数中，将读取到的颜色 + 基准颜色所得到的新颜色 push 进 vertex_colors 中，从而让颜色在 cow_colors 颜色的基础上产生渐变。

```
fin >> nVertices >> nFaces >> nEdges;
// 根据顶点数，循环读取每个顶点坐标
for (int i = 0; i < nVertices; i++)
{
    glm::vec3 tmp_node;
    fin >> tmp_node.x >> tmp_node.y >> tmp_node.z;
    vertex_positions.push_back(tmp_node);
    vertex_colors.push_back(tmp_node + cow_colors); // 顶点颜色为偏移色加上基准颜色
}
// 根据面片数，循环读取每个面片信息，并用构建的vec3i结构体保存
for (int i = 0; i < nFaces; i++)
{
    int num, a, b, c;
```

图 3.2 生成渐变颜色

4. 修改完的新颜色

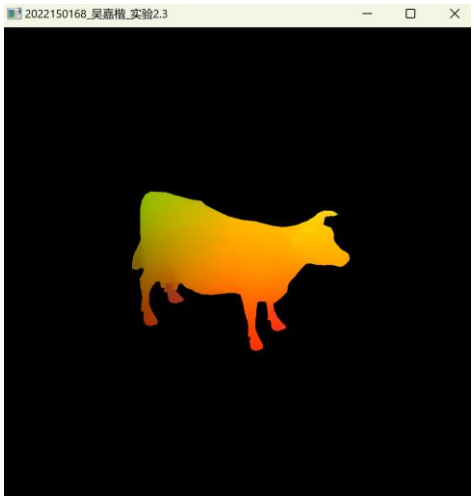

图 4 新颜色效果

5. 修改旋转参数，确定可以合适的初始旋转速度

```
15  const double DELTA_DELTA = 0.001;       // Delta的变化率
16  const double DEFAULT_DELTA = 0.001; // 默认的Delta值
```

图 5 设置初始旋转速度

6. 定义 currentAxios 变量用于记录当前的旋转轴

```
int currentTransform = TRANSFORM_ROTATE;    // 设置当前变换
int currentAxios = X_AXIS;                  // 设置当前轴
int mainWindow;
```

图 6 记录当前的旋转轴

7. 修改 key_callback 键盘交互回调函数

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
    switch (key)
    {
        // 退出。
        case GLFW_KEY_ESCAPE:
            if(action==GLFW_PRESS) glfwSetWindowShouldClose(window, GL_TRUE);
            break;
        // 绕X轴旋转
        case GLFW_KEY_X:
            if (action == GLFW_PRESS || action == GLFW_REPEAT)
                currentAxios = X_AXIS;
            break;
        // 绕Y轴旋转
        case GLFW_KEY_Y:
            if (action == GLFW_PRESS || action == GLFW_REPEAT)
                currentAxios = Y_AXIS;
            break;
        // 绕Z轴旋转
        case GLFW_KEY_Z:
            if (action == GLFW_PRESS || action == GLFW_REPEAT)
                currentAxios = Z_AXIS;
            break;
        // R: 增加变化量。
        case GLFW_KEY_R:
            if(action==GLFW_PRESS) updateDelta(1);
            break;
        // F: 减少变化量。
        case GLFW_KEY_F:
            if(action==GLFW_PRESS) updateDelta(-1);
            break;
        // T: 所有值重置。
        case GLFW_KEY_T:
            if(action==GLFW_PRESS) resetTheta();
            break;
    }
}
```

图 7 key_callback 回调函数

函数说明：

参数中，key 表示用户按下的键，action 指示按键的行为（按下、释放或重复按下）。

首先，函数通过 switch 语句检查按键。当用户按下 ESC 键时，action 被检测为 GLFW_PRESS，此时调用 glfwSetWindowShouldClose 将窗口的关闭标志设置为 GL_TRUE，让程序退出。对于 X、Y 和 Z 键，如果 action 是 GLFW_PRESS 或 GLFW_REPEAT，程序会将 currentAxios 设置为相应的 X_AXIS、Y_AXIS 或 Z_AXIS，用于控制模型围绕这些轴旋转。

此外，R 键用于增加变化量，通过调用 updateDelta(1) 实现，而 F 键用于减少变化量，调用 updateDelta(-1) 来实现调节。最后，按下 T 键时会调用 resetTheta()，将所有旋转参数重置为初始状态。

8. 修改 resetTheta 函数，需要 reset 旋转轴为 x 轴（默认轴）

```
void resetTheta()
{
    scaleTheta = glm::vec3(1.0, 1.0, 1.0);
    rotateTheta = glm::vec3(0.0, 0.0, 0.0);
    translateTheta = glm::vec3(0.0, 0.0, 0.0);
    scaleDelta = DEFAULT_DELTA;
    rotateDelta = DEFAULT_DELTA;
    translateDelta = DEFAULT_DELTA;
    currentAxios = X_AXIS; // 默认X轴
}
```

图 8 修改 resetTheta 函数

9. 定义鼠标点击回调事件

```
bool isRotate = false; // 是否开启旋转动画
// 鼠标点击回调函数
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS)
    {
        isRotate = true;
    }
    else if (button == GLFW_MOUSE_BUTTON_RIGHT && action == GLFW_PRESS)
    {
        isRotate = false;
    }
}
```

图 9 mouse_button_callback 函数

函数说明：

mouse_button_callback 函数通过修改一个布尔变量 isRotate 来控制是否开启旋转动画。IsRotate 将在 main 函数中使用，用于判断当前是否为旋转状态。

首先，检查按下的鼠标按钮 button 是否是左键（GLFW_MOUSE_BUTTON_LEFT），并且检测到的动作 action 是否是按下（GLFW_PRESS）。如果条件满足，程序将 isRotate 设置为 true，表示开启旋转动画。

接下来，函数又检查是否按下的是右键（GLFW_MOUSE_BUTTON_RIGHT），如果满足条件，同样检测到的动作为按下，此时将 isRotate 设置为 false，表示关闭旋转动画。

10. 在 main 函数中绑定鼠标回调事件

```
glfwMakeContextCurrent(window);
glfwSetKeyCallback(window, key_callback); // 设置键盘回调函数
glfwSetMouseButtonCallback(window, mouse_button_callback); // 设置鼠标回调函数
glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
```

图 10 绑定鼠标回调函数

11. 修改提示语输出函数

```
void printHelp() {
    printf("%s\n\n", "3D Transfomations");
    printf("Mouse options:\n");
    printf("Left Button: Start Animation\n");
    printf("Right Button: Stop Animation\n");
    printf("\n");
    printf("Keyboard options:\n");
    printf("X: Rotate around the X axis\n");
    printf("Y: Rotate around the Y axis\n");
    printf("Z: Rotate around the Z axis\n");
    printf("R: Increase delta of currently selected transform\n");
    printf("F: Decrease delta of currently selected transform\n");
    printf("T: Reset all transformations and deltas\n");
    printf("ESC: Exit\n");
}
```

图 11 printHelp 函数

12. 在 main 函数中添加旋转动画

```
// 输出帮助信息
printHelp();
// 启用深度测试
glEnable(GL_DEPTH_TEST);
while (!glfwWindowShouldClose(window))
{
    // 旋转动画
    if (isRotate) {
        updateTheta(currentAxios, 1);
    }
    display();

    // 交换颜色缓冲 以及 检查有没有触发什么事件（比如键盘输入、鼠标移动等）
    glfwSwapBuffers(window);
    glfwPollEvents();
}
cleanData();

return 0;
}
```

图 12 编写旋转动画逻辑

代码说明：

在循环体内，首先检查 isRotate 变量。如果 isRotate 为 true，表示启用了旋转动画，则调用 updateTheta(currentAxios, 1) 函数更新旋转角度。currentAxios 表示当前旋转的轴，第二个参数 1 表示旋转的增量值。

接下来调用 display() 函数，该函数负责渲染当前的图形或场景。此处的 display() 函数通常会包含 OpenGL 的绘制命令，用于将更新后的场景绘制到屏幕上。

13. 修改运行窗口的标题、尺寸等属性参数

```
// 配置窗口属性
GLFWwindow* window = glfwCreateWindow(1000, 1000, "2022150168_吴嘉楣_实验2", NULL, NULL);
```

图 13 修改窗口属性

14. 运行效果：

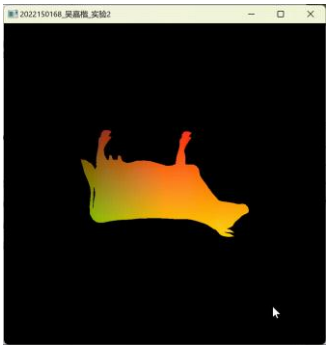

图 14.1 绕 X 轴旋转（默认）

图 14.2 绕 Y 轴旋转

图 14.3 绕 Z 轴旋转

深圳大学学生实验报告用纸

实验结论：

通过本次计算机图形学实验，我对 OpenGL 相关知识有了更为深刻的理解与应用能力的提升。

我不仅熟练掌握了 OpenGL 在三维模型读取与处理方面的技巧，还深入理解了三维模型的基本变换操作原理。实验过程中，我成功实现了对 OFF 格式三维模型文件的读取与显示，并能够灵活调整物体的显示颜色，从而增强了模型的视觉效果。

在三维模型的旋转动画方面，我结合模型的旋转变换过程，巧妙地在 main 函数中添加了自动旋转动画，使模型呈现出生动、动态的效果。同时，我通过精心设计的键盘与鼠标交互逻辑，实现了对旋转轴选择的精确控制，以及动画开始与暂停的便捷操作。

在着色器编程方面，我深入掌握了 uniform 关键字的使用以及数据传输方法。通过巧妙地运用这些知识，我成功地将 OFF 格式的三维模型文件整合到项目中，并实现了对模型颜色的渐变调整，使模型在视觉上更加丰富多彩。

综上所述，通过本次实验，我不仅巩固了 OpenGL 三维模型读取与处理、基本变换操作以及鼠标键盘交互控制等基础知识，还进一步提升了自己的实践能力和问题解决能力。这些宝贵的经验和知识，将对我未来的学习和工作产生深远的影响。

实验难点：

1. **OFF 格式文件解析**：OFF 格式的三维模型文件包含顶点和面的信息，需要正确解析这些数据并将其转换为 OpenGL 可以理解的格式，这是实验中的一个挑战。

2. **颜色渐变实现**：为了使模型颜色更加丰富，我需要在读取模型颜色的基础上添加渐变效果。这涉及到对颜色数据的理解和操作，以及如何将这些颜色应用到模型的顶点上。
解决方法：在从.off 文件中读取的颜色数据的基础上，加上一个 rgb 偏移值（基准颜色），使得新的颜色数据在基准颜色附近范围内变化）。

3. **旋转动画的控制逻辑**：实现模型的旋转动画需要对 OpenGL 的变换操作有深入的理解，同时还需要编写键盘交互逻辑来控制旋转轴，这在编程上较为复杂。
解决方法：在 main 函数的循环渲染逻辑中，添加绕轴旋转逻辑。

4. **键盘鼠标交互的实现**：为了实现用户通过键盘和鼠标与模型的交互，我需要编写相应的回调函数，并在这些函数中处理用户的输入，这要求我对 OpenGL 的事件处理机制有较好的掌握。
解决方法：使用 glfwSetKeyCallback、glfwSetMouseButtonCallback 的 OpenGL 内置方法绑定键盘、鼠标回调函数，在回调函数中设置不同的反馈逻辑。

指导教师批阅意见：   成绩评定：   指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。