深 圳 大 学 实 验 报 告

课程名称：计算机图形学

实验项目名称：实验四　带纹理的 OBJ 文件读取和显示

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 11 月 19 日 -- 2024 年 12 月 02 日

实验报告提交时间：2024 年 11 月 22 日

教务部制

实验目的与要求：

1. 了解三维曲面和纹理映基本知识
2. 了解从图片文件载入纹理数据基本步骤
3. 掌握三维曲面绘制过程中纹理坐标和几何坐标的使用
4. 在程序中读取带纹理的 obj 文件，载入相应的纹理图片文件，将带纹理的模型显示在程序窗口中。

实验过程及内容：

1. 学习 obj 文件格式的知识

obj 文件的每一行都会以一个关键词或者字符开头，“#”开头的为注释内容，“mtllib”开头的关键字后面会跟着要使用的材质文件名字，“usemtl”开头的关键字后面会跟着材质文件中要使用的材质名字，然后下面就跟着顶点的各种数据，每类顶点数据的开头字符都不同：

“v”代表点的几何坐标。

“vt”代表点的贴图坐标。

“vn”代表点的法线。

“f”开头表示面的数据，记录的是顶点索引。

obj 文件一般会配套生成一个材质文件（.mtl 后缀），如果有纹理的话还有纹理图片。关于.mtl 材质文件，这个文件里面会记录该模型材质相关的参数，“#”开头的为注释内容，“newmtl”开头的关键字后面会跟着一个名称，作为材质的名字，比如这里就有一个材质名字叫“Material”，后面跟着的内容都是这个材质的信息。

“Ns”开头的是材质的高光系数，“Ka”是环境光系数，“Ks”是镜面光系数，“Kd”是漫反射系数，“map_Kd”后面跟着的是纹理图片的路径。除此之外可能还会有其他关键字，都是描述材质的参数。

2. 在 TriMesh.cpp 中补全 storeFacesPoints 函数：将数据存储到用于传递给 GPU 的容器

代码截图：

```
// @TODO Task1 根据每个三角面片的顶点下标存储要传入GPU的数据
for (int i = 0; i < faces.size(); i++)
{
    // 坐标
    points.push_back(vertex_positions[faces[i].x]);
    points.push_back(vertex_positions[faces[i].y]);
    points.push_back(vertex_positions[faces[i].z]);
    // 颜色
    if (color_index.size() > 0)
    {
        colors.push_back(vertex_colors[color_index[i].x]);
        colors.push_back(vertex_colors[color_index[i].y]);
        colors.push_back(vertex_colors[color_index[i].z]);
    }
    // 法向量
    if (normal_index.size() > 0)
    {
        normals.push_back(vertex_normals[normal_index[i].x]);
        normals.push_back(vertex_normals[normal_index[i].y]);
        normals.push_back(vertex_normals[normal_index[i].z]);
    }
    // 纹理
    if (texture_index.size() > 0)
    {
        textures.push_back(vertex_textures[texture_index[i].x]);
        textures.push_back(vertex_textures[texture_index[i].y]);
        textures.push_back(vertex_textures[texture_index[i].z]);
    }
}
```

图 1 补全 storeFacesPoints 函数

代码说明：

代码的作用是将三角面片的几何数据组织并准备好传输到 GPU，以支持后续的图形渲染。代码遍历所有的三角形面片，通过 faces 数组获取每个面片的三个顶点索引。对于每个顶点索引，从顶点坐标数组 vertex_positions 中提取对应的坐标，将它们依次存入 points，以形成最终的顶点位置数据。

在处理颜色时，代码会先检查是否存在颜色索引 color_index。如果存在，利用索引从 vertex_colors 中提取面片三个顶点的颜色数据，并将这些颜色按顺序存入 colors，为模型的每个顶点赋予颜色信息。同样地，对于法向量和纹理坐标，代码也会检查是否存在相关索引。如果有法向量索引 normal_index，则从 vertex_normals 中获取每个顶点的法向量，将其存入 normals，以支持光照计算。对于纹理坐标，则通过 texture_index 获取相应的顶点纹理坐标，从 vertex_textures 中提取这些数据并存储到 textures，以便后续纹理映射使用。

3. 读取带纹理的 obj 文件：补全 TriMesh 类中的 readObj 函数

vertex_textures——存储 UV 坐标数据

vertex_positions——存储顶点坐标数据

vertex_normals——存储顶点法线数据

vertex_colors——存储顶点颜色数据

faces 存储三角面片的顶点索引数据

normal_index——存储三角面片的顶点的法向量数据的索引下标

texture_index——存储三角面片的顶点的纹理坐标数据的索引下标。

代码截图：

```
    // @TODO: Task2 读取obj文件，记录里面的这些数据，可以参考readOff的写法
    sin >> type;

    // vertex_positions
    if (type == "v")
    {
        sin >> _x >> _y >> _z;  // 解析OBJ文件中的顶点信息
        // 存储顶点坐标
        vertex_positions.push_back(glm::vec3(_x, _y, _z));
    }
    // vertex_normals
    if (type == "vn")
    {
        sin >> _x >> _y >> _z;  // 解析OBJ文件中的法向量信息
        // 存储法向量
        vertex_normals.push_back(glm::vec3(_x, _y, _z));
    }
    // vertex_textures
    if (type == "vt")
    {
        sin >> _x >> _y >> _z;  // 解析OBJ文件中的纹理坐标信息
        // 存储纹理坐标
        vertex_textures.push_back(glm::vec2(_x, _y));
    }
    // faces
    if (type == "f")
    {
        // 解析OBJ文件中的面信息
        sin >> a0 >> slash >> b0 >> slash >> c0;
        sin >> a1 >> slash >> b1 >> slash >> c1;
        sin >> a2 >> slash >> b2 >> slash >> c2;
        // 存储面的顶点、纹理坐标和法向量的索引信息
        faces.push_back(vec3i(a0 - 1, a1 - 1, a2 - 1));
        texture_index.push_back(vec3i(b0 - 1, b1 - 1, b2 - 1));
        normal_index.push_back(vec3i(c0 - 1, c1 - 1, c2 - 1));
    }
}
// 其中vertex_color和color_index可以用法向量的数值赋值
vertex_colors = vertex_normals;
color_index = normal_index;
storeFacesPoints();
```

图 2 读取带纹理的 obj 文件

代码说明：

代码的功能是解析 OBJ 文件中描述三维模型的几何数据，并将这些数据存储到相应的容器中，便于后续的渲染操作。通过创建一个字符串流解析当前行的内容，根据行首的标识符决定处理的类型。

若标识符是 "v"，表示这一行包含顶点坐标信息，程序从流中读取三个浮点数 _x、_y 和 _z，分别代表顶点的三维坐标，并将它们构造成 glm::vec3 存入 vertex_positions。

当标识符是 "vn" 时，说明这一行是法向量数据，同样读取三个浮点数后，将其作为法向量存入 vertex_normals 容器。

对于标识符为 "vt" 的行，解析的是纹理坐标信息，读取两个浮点数 _x 和 _y，构造成 glm::vec2 并存储到 vertex_textures。

如果标识符为 "f"，则表示当前行描述了一个面片的信息。面片由多个顶点组成，每个顶点由顶点坐标索引、纹理坐标索引和法向量索引组成，索引以斜杠分隔的形式写出。

4. 模型和纹理显示：补全 main.cpp 中的 init 函数

代码截图：

```
// @TODO: Task2 读取桌子模型
TriMesh* table = new TriMesh();
table->setNormalize(true);
table->readObj("./assets/table.obj");
// 设置物体的旋转位移
table->setTranslation(glm::vec3(-0.8, -0.3, 0.0));
table->setRotation(glm::vec3(-90.0, 0.0, 0.0));
table->setScale(glm::vec3(2, 2, 2));
// 加到painter中
painter->addMesh(table, "mesh_a", "./assets/table.png", vshader, fshader);  // 指定纹理与着色器
// 加入一个容器内，为了程序结束时将这些数据释放
meshList.push_back(table);

// @TODO: Task2 读取娃娃模型
TriMesh* wawa = new TriMesh();
wawa->setNormalize(true);   // 归一化
wawa->readObj("./assets/wawa.obj");
// 设置物体的旋转位移
wawa->setTranslation(glm::vec3(0.8, 0.0, 0.0));
wawa->setRotation(glm::vec3(-90.0, 0.0, 0.0));
wawa->setScale(glm::vec3(2.0, 2.0, 2.0));
// 加到painter中
painter->addMesh(wawa, "mesh_b", "./assets/wawa.png", vshader, fshader);    // 指定纹理与着色器
// 加入一个容器内，为了程序结束时将这些数据释放
meshList.push_back(wawa);
```

图 3 补全 init 函数

代码说明：

这里通过加载桌子和娃娃的模型文件，将其设置为适当的变换属性后添加到渲染系统中，并维护一个对象列表以便后续管理和释放资源。

首先，为桌子模型创建了一个新的 TriMesh 对象，并通过调用 setNormalize(true) 方法启用了归一化功能，以确保模型大小适配渲染需求。随后调用 readObj 方法加载 ./assets/table.obj 文件中描述的模型几何数据。

在加载桌子模型之后，设置了该对象的位移、旋转和缩放属性，其中位移向量将模型移动到三维空间中的特定位置，旋转角度以欧拉角形式定义了模型的方向，而缩放因子控制了模型的尺寸调整。设置完成后，通过 painter->addMesh 方法将桌子模型添加到渲染器中，同时指定了桌子模型的纹理路径 ./assets/table.png 和所使用的顶点着色器与片元着色器。最后，将桌子对象存入 meshList 容器中，便于在程序结束时统一释放资源。

对于娃娃模型，过程与桌子模型类似。新建一个 TriMesh 对象后，归一化设置为开启状态，并通过 readObj 方法加载 ./assets/wawa.obj 文件中描述的几何数据。紧接着，对模型应用平移、旋转和缩放的变换属性，使其在场景中位于特定位置并具有正确的方向和尺寸。随后，通过调用 addMesh 方法，将其与纹理 ./assets/wawa.png 以及对应的着色器绑定并添加到渲染器中。最后，娃娃模型也被存入 meshList，与其他对象一起统一管理和释放。

5. 优化程序执行速度：在编译的时候选择 Release 模式

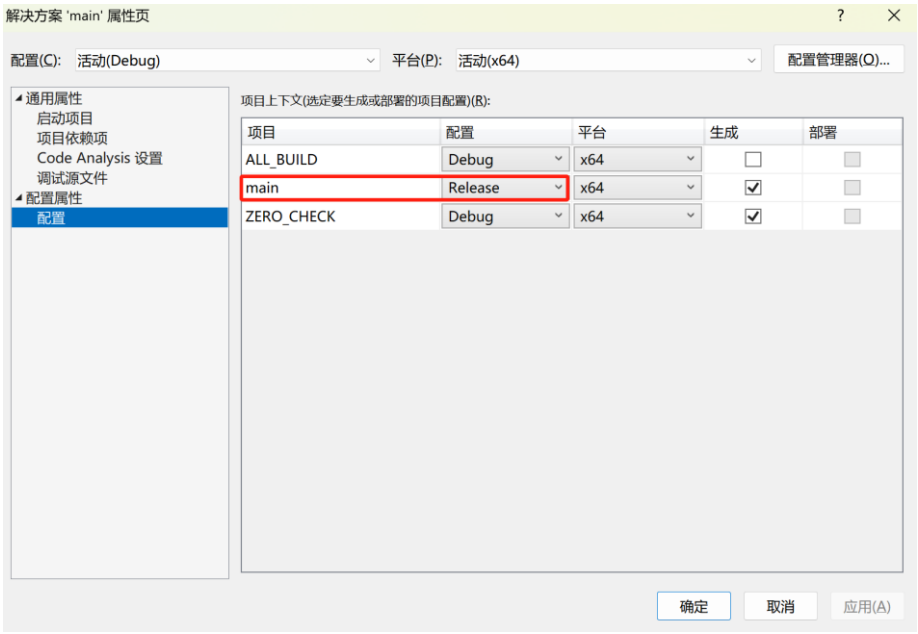

图 4 选择 Release 模式

原理说明：

与 Debug 模式不同，Release 模式会去除调试信息，减少程序的体积，并优化代码结构。编译器会通过内联函数、死代码消除、循环优化等方式，减少不必要的计算和函数调用，提高程序的运行效率。

此外，Release 模式下的编译器还会对内存访问进行优化，通过合理布局数据和减少内存访问次数来提升性能。编译器会根据目标平台的硬件架构生成更加高效的汇编指令，并启用更多的优化选项，如提升并行计算能力和减少运行时检查，从而最大化程序的执行速度。

6. 修改运行窗口的标题和尺寸大小

```
// 配置窗口属性
GLFWwindow* window = glfwCreateWindow(700, 700, "2022150168_吴嘉楷_实验四", NULL, NULL);
```

图 5 修改窗口配置

但是，运行程序后，发现中文出现乱码。因为文件本身的编码格式是“utf-8”。于是，我修改了编译器执行时的解码格式为“utf-8”，从而使编解码格式一致，避免了中文乱码问题：

```
#pragma execution_character_set("utf-8");   // 设置执行字符集为utf-8
// 配置窗口属性
GLFWwindow* window = glfwCreateWindow(700, 700, "2022150168_吴嘉楷_实验四", NULL, NULL);
```

图 6 防止中文乱码

7. 程序运行结果

默认效果：

图 7 初始效果

调整 rotate 角度后：

图 8 调整 rotate 角度后的效果

调整 up 角度后：

图 9 调整 up 角度后的效果

综合调整 rotate 和 up 角度后：

图 10 同时调整 rotate 和 up 角度的效果

实验结论：

在本次实验中，我成功实现了带纹理的 OBJ 文件的读取和显示。通过学习 OBJ 文件格式和材质文件（.mtl）的结构，我掌握了如何从这些文件中提取顶点坐标、纹理坐标、法线等关键数据。

在 TriMesh 类的 readObj 函数中，我实现了对 OBJ 文件的解析，并将解析出的 UV 坐标、顶点坐标、顶点法线和顶点颜色数据存储到相应的数据结构中。

此外，我还完善了 TriMesh.cpp 中的 storeFacesPoints 函数，确保了数据能够被有效地组织并传递给 GPU，为后续的图形渲染打下了基础。

在 main.cpp 中，我通过 init 函数加载了带纹理的模型，并成功地将模型和纹理显示在程序窗口中。通过设置模型的变换属性，我能够控制模型在三维空间中的位置、旋转和缩放，实现了模型的逼真渲染。

最后，我还学习了如何在编译时选择 Release 模式来优化程序的执行速度，并通过调整编译器的解码格式解决了中文乱码问题。

通过本次实验，我不仅加深了对三维图形处理和纹理映射的理解，还提升了我的编程技能和问题解决能力。实验结果表明，我能够成功地将理论知识应用于实际编程任务中，实现了一个功能完整的三维模型显示程序。

实验难点：

1. **OBJ 文件解析**：OBJ 文件格式相对复杂，包含了多种类型的数据和多个部分。正确解析这些数据并将其存储到适当的数据结构中是一个挑战，需要对文件格式有深入的理解。

2. **纹理坐标和几何坐标的对应**：在处理纹理映射时，确保每个顶点的纹理坐标与其几何坐标正确对应是一个技术难点。这需要精确地解析 OBJ 文件中的面数据，并正确地将顶点索引映射到纹理坐标。

3. **性能优化**：在实验过程中，我遇到了程序执行速度较慢的问题。通过实验指导和实践，我学会了使用 Release 模式编译程序，并尝试了代码优化，以提高程序的运行效率。

4. **中文乱码问题**：在程序运行时，窗口标题和部分输出出现了中文乱码。这需要我了解字符编码的知识，并调整编译器的设置以匹配文件的编码格式，以解决乱码问题。

实验问题：

在我们使用键盘“o”键想要控制相机与物体的距离时，发现一个现象：**无论怎么改变 radius，物体呈现出来的大小都没有发生变化。**

经过一番 debug 后，我发现键盘交互是生效的，因为 radius 参数确实在发生改变。于是我猜测，物体的模型计算使用了**正交投影而不是透视投影**，导致物体与相机的距离改变没有引起视图大小的变化。

在 MeshPainter.cpp 中，我们定位到了第 156 行的代码：

```
155     camera->viewMatrix = camera->getViewMatrix();
156     camera->projMatrix = camera->getProjectionMatrix(true); // 正交投影
157
158 #ifdef __APPLE__    // for MacOS
```

图 11 使用正交投影

根据 getProjectionMatrix 函数的定义，我们可知，参数为 true 时使用正交投影，参数为 false 时采用透视投影。于是，如果我们希望在按下键盘“o”键时改变物体视图的大小，则可以把参数 true 修改为 false 即可。

但是，由于使用透视投影会使玩偶的视觉朝向向外偏移，导致观感不佳，于是，我们没有进行修改。

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。