**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： 胡瑞珍**

**报告人： 学号： 班级：**

**实验时间： 2024 年 11 月 12 日 -- 2024 年 12 月 10 日**

**实验报告提交时间： 2024 年 11 月 23 日**

**教务部制**

|  |
| --- |
| 实验目的与要求：   1. 掌握OpenGL三维场景的读取与绘制方法，熟悉obj文件的读取与处理，了解obj文件的基本格式。 2. 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. **读取带纹理的obj文件**   obj 文件格式是一种常用的 3D 模型文件格式，通常用于存储 3D 网格数据，包含顶点、面、法向量、纹理坐标等信息。它是文本格式，易于解析和编辑。以下是 obj 文件格式的主要内容：   * 顶点 (v)：   + 用于定义 3D 空间中的点。   + 格式：v x y z   + x, y, z：顶点的坐标。   + 示例：v 1.0 2.0 3.0 * 法向量 (vn)：   + 定义每个顶点的法向量，用于光照计算和表面着色。   + 格式：vn x y z   + x, y, z：法向量的方向。   + 示例：vn 0.0 0.0 1.0 * 纹理坐标 (vt)：   + 定义每个顶点的纹理坐标，通常用于映射纹理图像到模型表面。   + 格式：vt u v   + u, v：纹理坐标。   + 示例：vt 0.5 1.0 * 面 (f)：   + 定义多边形的面。每个面由若干个顶点组成，通常为三角形或四边形，支持顶点索引、纹理索引和法向量索引。   + 格式：f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3   + v：顶点索引（从1开始）。   + vt：纹理坐标索引（可选）。   + vn：法向量索引（可选）。   + 示例：f 1/1/1 2/2/2 3/3/3 或 f 1//1 2//2 3//3（没有纹理坐标和法向量）   根据 obj 文件的格式，我们可以实现以下读取 obj 文件数据的代码。  代码实现：  （1）解析每一行数据   * 使用 std::getline() 逐行读取文件内容。 * 对每一行使用 std::istringstream 进行解析。通过读取每行的第一个单词（如 v、vn、vt、f）来确定该行的数据类型。 * 为了处理顶点、法向量、纹理坐标和面数据，程序根据不同的行类型采取不同的处理方式。     （2）处理顶点坐标行v、法向量行vn和纹理坐标行vt   * 如果行的类型为 v，则表示这一行包含一个顶点的三维坐标。程序从这一行中提取 x、y、z 坐标并将其存储到 vertex\_positions 向量中。 * 如果行的类型为 vn，表示这一行包含一个法向量的三维分量。程序读取法向量的 x、y、z 分量，将其存储到 vertex\_normals 向量中。同时，还将法向量的分量当作颜色信息存储到 vertex\_colors 向量中。 * 如果行的类型为 vt，表示这一行包含一个纹理坐标。程序读取纹理坐标 x 和 y，并将其存储到 vertex\_textures 向量中。     （3）处理面数据行f   * 如果行的类型为 f，表示这一行定义了一个面。面数据可以包括多个顶点，通常是三角形或四边形。每个面由顶点的索引以及可能存在的纹理坐标的索引、法向量的索引组成。 * 使用 sscanf 提取面数据中的每个顶点的索引、纹理坐标的索引和法向量的索引。每个面可能包含 3 或 4 个顶点，三角形面有 3 个顶点，四边形面有 4 个顶点。 * 对于三角形面，直接将顶点索引、纹理索引、法向量索引存储到对应的向量中。 * 对于四边形面，先将四边形拆分成两个三角形，然后分别存储两个三角形的数据。拆分规则是：第一个三角形由顶点 0、1、2 组成，第二个三角形由顶点 2、3、0 组成。 * 对于每个三角形或拆分后的三角形，面数据（顶点索引、纹理索引、法向量索引）被分别存储到 faces、texture\_index、normal\_index 和 color\_index 向量中。 * 对于每个面，obj 文件中顶点索引从 1 开始，而数组的索引是从 0 开始的，因此需要将索引减 1。        1. **完善数据的读取**   读取 obj 文件数据后，还需要将其数据传入 GPU 中，下面将用并行化处理顶点坐标、颜色、法向量和纹理坐标的存储操作来加速数据传入。  代码实现：  （1）内存预分配  为存储所有顶点、颜色、法向量和纹理的坐标预留足够的内存空间。    （2）创建数据处理函数  通过 lambda 表达式定义了四个函数，这些函数分别负责处理顶点、颜色、法向量和纹理数据，并将处理结果存储到相应的容器中。每个函数会遍历所有面（faces），并根据顶点索引将数据添加到对应的容器中。   * processPoint： 负责处理顶点坐标数据。它遍历 faces，将每个面（即三角形）的三个顶点的坐标添加到 points 容器中。      * processColor： 负责处理颜色数据。如果 color\_index 非空（说明有颜色信息），它会将每个三角形的顶点颜色添加到 colors 容器中。      * processNormal： 负责处理法向量数据。如果 normal\_index 非空（说明有法向量信息），它会将每个三角形的顶点法向量添加到 normals 容器中。      * processTexture： 负责处理纹理坐标数据。如果 texture\_index 非空（说明有纹理信息），它会将每个三角形的顶点纹理坐标添加到 textures 容器中。     （3）启动并行线程  std::async 用于启动异步任务。每个 std::async 调用都启动一个新的线程来执行一个数据处理函数。std::launch::async 确保任务在独立的线程中执行。    （4）等待所有线程完成  future.get() 方法会阻塞调用线程，直到任务完成。通过调用 future1.get()、future2.get()、future3.get() 和 future4.get()，主线程等待所有四个并行任务完成，确保所有数据都已经处理完毕，准备好传递给GPU。     1. **模型和纹理显示**   导入并绘制桌子（table）和娃娃（wawa）模型。  代码实现：  首先创建了 TriMesh 对象来表示桌子（table）和娃娃（wawa）模型，并通过调用 readObj 方法分别从 obj 文件加载它们的顶点数据和面片信息。接着，通过 setNormalize 方法确保加载的模型会进行大小归一化。然后，分别使用 setTranslation、setRotation 和 setScale 设置物体的平移、旋转和缩放变换，以调整它们在三维空间中的位置和形态。最后，通过 painter->addMesh 方法将这两个物体添加到渲染器中，指定它们的纹理文件路径以及对应的顶点着色器和片段着色器。     1. **最终成果**   可以看到，桌子（table）和娃娃（wawa）模型成功绘制，且带有纹理。 |

深圳大学学生实验报告用纸

|  |
| --- |
| 实验结论：  通过本次实验，我成功掌握了OpenGL三维场景的读取与绘制方法，熟悉了obj文件的读取与处理，并了解了obj文件的基本格式。在实验过程中，我学会了如何读取带纹理的obj文件，并将相应的纹理图片文件载入程序中，最终成功地将带纹理的模型显示在程序窗口中。通过对顶点、法向量、纹理坐标和面数据的解析和处理，我加深了对3D模型数据结构的理解。此外，通过并行化处理顶点坐标、颜色、法向量和纹理坐标的存储操作，我体会到了并行计算在提高数据处理效率方面的优势。实验结果表明，所编写的程序能够正确读取和显示带纹理的3D模型，验证了代码的正确性和有效性。 |
| 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。