编程作业 3 光栅化渲染着色 3D 场景

姓名 胡阳，冯朝航，黄嘉鑫，李俊熠 学号 2023141461174，2023141461176，

2023141461175，2023141461171

1 思路

本次实验基于 WebGL 框架实现光栅化渲染着色 3D 场景，核心围绕“几何体绘制 - 纹理映射 - 光照计算 - 阴影生成 - 交互控制”的流程展开，严格遵循实验要求完成功能开发：

1.环境初始化：基于老师提供的框架，初始化 WebGL 上下文、着色器程序（顶点着色器、片段着色器、深度着色器等），开启深度缓存以保证 3D 场景的深度排序正确性。

2.几何体与纹理实现：补全 configTexture.js 中的 configureTexture 函数（TODO1），创建 2D 纹理对象并加载图片，通过纹理参数设置（重复模式、过滤方式）实现物体表面细节表现；同时利用 Models.js 中的 colorCube 和 plane 函数生成立方体、平面等多个几何体，定义顶点坐标、法向量和纹理坐标数据。

3.光照效果实现：在/shader/box.frag 中补全 Phong 模型计算（TODO2），基于 Phong Shading 方法，在片段着色器中逐像素计算环境光（ambient）、漫反射（diffuse）和镜面反射（specular），结合材质参数和光源属性得到最终光照颜色。

4.阴影效果实现：在/shader/box.frag 中添加阴影计算（TODO3），采用 ShadowMap 技术，先通过深度着色器渲染光源视角下的深度纹理，再在片段着色器中采样深度纹理，对比当前片段的深度值与采样值，判断是否处于阴影中（返回 1 为阴影，0 为非阴影）。

5.交互与渲染循环：实现光源（点光源 / 平行光切换、位置旋转缩放）和观察者（相机旋转、平移、缩放）的交互控制，通过 render 函数构建渲染循环，完成场景的实时更新与绘制。

2 重点难点

2.1 重点内容

1.Phong 模型的精准计算：需严格遵循 Phong 光照公式，正确处理法向量、光照方向、视角方向的归一化，确保环境光、漫反射、镜面反射的权重分配合理，让物体光照过渡自然。

2.ShadowMap 的完整流程：包括深度纹理的生成、光源空间矩阵的计算、纹理采样时的坐标转换，这是实现阴影效果的核心，直接影响阴影的准确性。

3.纹理与着色器的绑定：确保纹理对象创建、参数设置、采样器绑定的流程正确，让纹理能准确映射到几何体表面，呈现细节纹理效果。

2.2 难点与解决方法

1.纹理加载与映射异常

难点：纹理图片加载异步导致纹理绑定失败，或纹理坐标与几何体顶点不匹配，出现纹理拉伸、错位。

解决：在纹理加载完成后再进行渲染初始化，通过 image.onload 回调确保纹理数据就绪；核对 texCoordsArray 中纹理坐标与顶点的对应关系，确保每个顶点的纹理坐标正确映射到纹理图片的对应区域。

2.Phong 光照计算失真

难点：法向量在模型变换中未正确更新，或光照方向、视角方向计算错误，导致漫反射过暗、镜面反射位置偏移。

解决：使用法向量矩阵（normalMatrix）对法向量进行变换，确保法向量方向正确；在片段着色器中重新计算顶点到光源、顶点到观察者的向量并归一化，严格按照公式计算漫反射系数（dot (法向量，光照方向)）和镜面反射系数（dot (反射方向，视角方向)）。

3.ShadowMap 阴影采样错误

难点：光源空间矩阵传递错误，或深度纹理采样时坐标转换不当，导致阴影错位、无阴影效果。

解决：正确计算 lightSpaceMatrix（投影矩阵 × 视图矩阵），并传递给片段着色器；将片段坐标从裁剪空间转换到纹理空间，处理纹理采样的精度问题，通过 clamp 函数限制采样范围，避免越界采样。

4.WebGL 着色器程序链接失败

难点：着色器语法错误（如版本声明缺失、变量类型不匹配）导致程序链接失败，无报错提示难以定位。

解决：在 initShaders.js 中添加详细的编译日志输出，检查着色器代码中#version 300 es 声明、变量声明与使用的一致性，确保 uniform 变量、attribute 变量的类型和名称与 JavaScript 代码中一致。

3 心得

通过本次实验，我深入理解了光栅化渲染的核心流程，掌握了 WebGL 编程的关键技术，收获颇丰：

1.技术层面：熟练掌握了 Phong 模型和 ShadowMap 的原理与实现，理解了纹理映射、光照计算、阴影生成在 3D 渲染中的作用机制；学会了 WebGL 中着色器程序的编写、uniform/attribute 变量的传递、纹理和缓冲对象的管理，以及交互控制的实现方法。

2.问题解决能力：在调试纹理映射、光照失真、阴影错位等问题的过程中，学会了通过日志输出、分步验证的方式定位问题，例如通过打印法向量、光照方向的数值验证计算正确性，通过单独渲染深度纹理检查 ShadowMap 生成是否正常。

3.工程思维：体会到 3D 渲染是一个多模块协同的过程，从几何体建模、纹理加载，到光照和阴影计算，再到交互控制，每个环节都需要严格遵循技术规范，任何一个细节的疏忽都会导致整体效果异常，培养了严谨的编程习惯。

4.知识拓展：深入理解了计算机图形学的核心概念，如 MVP 矩阵变换、法向量归一化、深度缓存、纹理采样等，为后续学习更复杂的渲染技术（如 PBR、全局光照）奠定了坚实基础。

本次实验不仅完成了既定的功能要求，更让我体会到理论与实践结合的重要性，只有将图形学理论转化为具体的代码实现，才能真正理解其底层逻辑，同时也提升了自己的编程能力和问题解决能力。