学号	姓名	论文规范性（10）	问题分析与调研（30）	方案创新性（20）	实验结果分析与讨论（40）	结课论文总成绩（100）
21301169	魏义卓	6	22	16	33	77

北京交通大学

计算机图形学期末课程论文

设计（论文）题目

高级光照和阴影效果在计算机图形学中的应用与实践

—— 以 Room 项目为例

设计（论文）英文题目

Application and practice of advanced lighting and shadow effects in computer graphics

学　院：软件学院

专　业：软件工程

学生姓名：魏义卓

学　号：21301169

指导教师：吴雨婷

北京交通大学

2024 年 6 月

中文摘要

摘要：

当你正在玩一款非常逼真的 3A 大作，阳光透过树叶洒下斑驳的光影，角色的影子随着动作变化而移动。这样的视觉效果是不是让人惊叹不已？你可能会想，这样的阴影是如何实现的呢？这篇论文，我将开始探索阴影映射这一神奇的技术并应用于 Room 项目当中。

整体来说，本课程论文旨在探讨高级光照和阴影效果在计算机图形学中的应用，并通过对现有研究现状的调研，选择适当的技术在 Room 项目中进行实现。具体而言，我将深入分析实时阴影映射技术的实现方法并在 Room 项目中应用该技术。通过实验结果的分析与讨论，评估该技术的应用效果，并总结其在实际项目中的应用价值！

ABSTRACT

ABSTRACT:
When you are playing a very realistic 3A masterpiece, the sun casts mottled light and shadow through the leaves, and the character's shadow moves with the movement. Isn’t this visual effect amazing? You may be thinking, how is such a shadow achieved? In this paper, I will begin to explore the magical technology of shadow mapping and apply it to the Room project. Overall, the purpose of this course paper is to explore the application of advanced lighting and shadow effects in computer graphics, and to select appropriate technologies for implementation in the Room project through a survey of existing research status. Specifically, I will deeply analyze the implementation method of real-time shadow mapping technology and apply this technology in the Room project. Through the analysis and discussion of experimental results, evaluate the application effect of this technology, and summarize its application value in actual projects!

目　录

目录

1 引言

1.1 研究背景

计算机图形学自其诞生以来，经历了迅速的发展和广泛的应用，从早期简单的二维图形绘制到如今复杂的三维图形渲染，已经成为现代计算机科学中的重要领域之一。在这个领域中，光照和阴影效果的研究和实现尤为重要，因为它们在提升图像的视觉真实感方面起着至关重要的作用。光照效果能够模拟现实世界中光源与物体之间的相互作用而阴影效果则能够为场景增添深度和真实感。特别是在实时渲染领域，高质量的光照和阴影效果不仅能够提升用户体验，还能够为 **3A** 游戏、虚拟现实和电影等应用提供更具沉浸感的视觉效果。

随着硬件性能的提升和算法的不断优化，越来越多的高级光照和阴影技术得以实现，如基于物理的渲染（Physically Based Rendering, PBR）、实时阴影映射（Real-Time Shadow Mapping）等。这些技术在提升图像真实感和渲染效率方面展现出了巨大的潜力。然而，这些技术在实际项目中的应用依然面临诸多挑战，需要综合考虑渲染性能、视觉效果和实现难度等因素。因此，研究和实践这些技术在实际项目中的应用具有重要的理论和实际意义。

1.2 研究目的及意义

本研究的目的在于通过对高级光照和阴影效果的深入研究，选择适当的技术并在 Room 项目中进行实现和验证，从而提升项目的视觉效果和用户体验。具体而言，本文将重点

探讨实时阴影映射技术的实现方法，并在 Room 项目中应用该技术，分析其在实际场景中的表现和效果。通过实验结果的分析与讨论，我们将评估阴影映射技术的应用效果总结其在实际项目中的应用价值，并为相关领域的研究提供参考。

2 本研究具有以下几个方面的意义：

3 理论意义：通过系统地研究和分析高级光照和阴影效果的实现方法，丰富计算机图形学领域的理论研究，推动相关技术的发展。

4 实践意义：在 Room 项目中实践和验证阴影映射技术，提升项目的视觉效果和用户体验，为类似项目的开发提供技术参考和实现方案。

5

6 实验设计与实现

6.1 工具及技术原理

- **操作系统：Windows10 64 位**
- **开发工具：Visual Studio 2022**
- **图形库: GLUT、 GLU**
- **编程语言：C++**

本节实验采用的阴影映射技术及相关的光照模型。

2.1.1 光照模型

光照模型在计算机图形学中用于模拟现实世界中的光照效果，以提升视觉真实感。本文主要讨论与阴影映射技术相关的光照模型：Phong 光照模型和基于物理的渲染（Physically Based Rendering, PBR）。

2.1.1.1 Phong 光照模型

Phong 光照模型由 Bui Tuong Phong 在 1975 年提出，用于计算表面的光照效果。它主要包括三个分量：

1. **环境光（Ambient Light）**：模拟环境中均匀存在的光线，赋予物体基本的可见性。

2. **散射光（Diffuse Light）**：模拟光线在粗糙表面上扩散后的效果，使表面在光源方向上的亮度随着入射角度的变化而变化。
3. **镜面反射光（Specular Light）**：模拟光线在光滑表面上产生的高光，使物体在光源方向上出现亮点。

Phong 光照模型的计算公式如下：

$I=IaKa+Id(Kd(L\cdot N))+Is(Ks(R\cdot V)n)$*I=Ia**Ka+Id(Kd(***L**·**N**))+Is(Ks(**R**·**V**)n)*

其中，$Ia,Id,IsIa,Id,Is$ 分别表示环境光、散射光和镜面反射光的强度；$Ka,Kd,KsKa,Kd,Ks$ 为材质的反射系数；L**L** 为光源方向向量，N**N** 为表面法线向量，R**R** 为反射光方向向量，V**V** 为视线方向向量，nn 为高光指数。

2.1.1.2 基于物理的渲染（PBR）

基于物理的渲染（PBR）是一种旨在更真实地模拟光与物体之间相互作用的光照模型 。PBR 使用能量守恒和微表面理论等概念，能够生成更逼真的视觉效果。

PBR 模型包括两个主要部分：

1. **反射率（Albedo）**：表示物体表面在各个方向上的反射率分布，反映物体的颜色和材质特性。
2. **金属度（Metalness）和粗糙度（Roughness）**：金属度决定物体的金属性质，粗糙度决定表面的光滑程度。

PBR 模型通常基于 BRDF（双向反射分布函数）进行计算，描述光线在表面反射的概率分布。PBR 具有高度一致性和逼真的光照效果，已成为现代图形学渲染的重要技术之一。

2.1.2 阴影效果

阴影效果是提升场景真实感的关键技术之一。常见的阴影实现方法包括阴影映射（Shadow Mapping）和阴影体积（Shadow Volume）。本实验主要采用阴影映射技术。

2.1.2.1 阴影映射

阴影映射是一种基于深度缓冲区的阴影生成技术，主要步骤如下：

1. **从光源视角渲染场景，生成深度图**：初始化阴影映射资源，包括深度缓冲区和深度纹理，用于存储光源视角下的深度信息。
2. void initShadowMap() {
3. glGenFramebuffers(1, &depthMapFBO);
4. glGenTextures(1, &depthMap);
5. glBindTexture(GL_TEXTURE_2D, depthMap);
6. glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, SHADOW_WIDTH, SHADOW_HEIGHT, 0, GL_DEPTH_COMPONENT, GL_FLOAT, NULL);

```
7.  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
    GL_NEAREST);
8.  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
    GL_NEAREST);
9.  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
    GL_CLAMP_TO_BORDER);
10. float borderColor[] = {1.0, 1.0, 1.0, 1.0};
11. glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR,
    borderColor);
12. glBindFramebuffer(GL_FRAMEBUFFER, depthMapFBO);
13. glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT,
    GL_TEXTURE_2D, depthMap, 0);
14. glDrawBuffer(GL_NONE);
15. glReadBuffer(GL_NONE);
16. glBindFramebuffer(GL_FRAMEBUFFER, 0);
17.}
18.
```

19. **渲染实际场景时，使用生成的深度图判断像素是否在阴影中**：在渲染过程中，通过比较光空间坐标的深度值与深度图中的深度值判断像素是否在阴影中。

```
20.void renderSceneWithShadows() {
21.  glViewport(0, 0, SCR_WIDTH, SCR_HEIGHT);
```

```
22.    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
23.
24.    shader.use();
25.    glm::mat4 lightSpaceMatrix = lightProjection * lightView;
26.    shader.setMat4("lightSpaceMatrix", lightSpaceMatrix);
27.
28.    glActiveTexture(GL_TEXTURE1);
29.    glBindTexture(GL_TEXTURE_2D, depthMap);
30.    shader.setInt("shadowMap", 1);
31.
32.    renderScene(shader);
33.}
34.
```

阴影映射实现简单且计算效率高，适用于实时渲染。然而，在处理阴影边缘时可能出现锯齿现象，需要使用 PCF（Percentage-Closer Filtering）等技术进行抗锯齿处理。

2.2 阴影效果实现

针对本项目，拟采用阴影映射方向进行实验：

主要包括以下几个步骤：生成深度图、渲染深度图、应用深度图。

初始化阴影映射

首先，需要初始化阴影映射所需的资源，包括深度缓冲区和深度纹理。深度缓冲区用于存储从光源视角下生成的深度信息，而深度纹理则用于在渲染阶段采样深度信息。

按照原理 initShadowMap 一个帧缓冲对象和一个纹理对象。然后将纹理对象绑定到帧缓冲对象上，以存储深度信息。通过设置纹理参数和禁用颜色缓冲区的写入，确保帧缓冲对象只存储深度信息。

渲染深度图

在每一帧渲染过程中，首先需要从光源视角渲染场景，以生成深度图。该过程包括设置光源的投影矩阵和视图矩阵，并将渲染结果存储到深度缓冲区中。

```
void renderSceneToDepthMap() {
    // 设置视口大小为深度图的尺寸
    glViewport(0, 0, SHADOW_WIDTH, SHADOW_HEIGHT);

    // 绑定帧缓冲对象
    glBindFramebuffer(GL_FRAMEBUFFER, depthMapFBO);

    // 清除深度缓冲区
    glClear(GL_DEPTH_BUFFER_BIT);
```

```
    // 设置光源视角的投影矩阵和视图矩阵
    glm::mat4 lightProjection, lightView;
    glm::mat4 lightSpaceMatrix;
    float near_plane = 1.0f, far_plane = 7.5f;
    lightProjection = glm::ortho(-10.0f, 10.0f, -10.0f, 10.0f, near_plane, far_plane);
    lightView = glm::lookAt(lightPos, glm::vec3(0.0f), glm::vec3(0.0f, 1.0f, 0.0f));
    lightSpaceMatrix = lightProjection * lightView;

    // 将光空间矩阵传递给着色器
    // 设置相应的 uniform 变量
    glUniformMatrix4fv(glGetUniformLocation(depthShader.ID,
"lightSpaceMatrix"), 1, GL_FALSE, glm::value_ptr(lightSpaceMatrix));

    // 渲染场景
    renderScene(depthShader);

    // 解除绑定帧缓冲对象
    glBindFramebuffer(GL_FRAMEBUFFER, 0);
}
```

首先设置视口大小为深度图的尺寸，然后绑定帧缓冲对象并清除深度缓冲区。接下来，设置光源的投影矩阵和视图矩阵，并将光空间矩阵传递给深度图着色器。最后，渲染场景并生成深度图。

应用阴影映射

在最终的渲染阶段，使用生成的深度图来判断像素是否在阴影中。具体方法是将每个像素转换到光空间坐标系，并与深度图中的深度值进行比较。

```
void renderSceneWithShadows() {}
```

在最终渲染过程中，首先清除颜色缓冲区和深度缓冲区，然后使用常规的着色器进行渲染。将光空间矩阵和深度图传递给着色器，并在着色器中计算每个像素是否在阴影中

实现阴影映射的着色器

为了实现阴影映射，需要在顶点着色器和片段着色器。在顶点着色器中，计算光空间坐标，并传递给片段着色器。

```
// 顶点着色器
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
```

```
layout (location = 2) in vec2 aTexCoords;

out vec2 TexCoords;
out vec4 FragPosLightSpace;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
uniform mat4 lightSpaceMatrix;

void main() {
    TexCoords = aTexCoords;

    vec4 fragPos = model * vec4(aPos, 1.0);
    FragPosLightSpace = lightSpaceMatrix * fragPos;

    gl_Position = projection * view * fragPos;
}
```

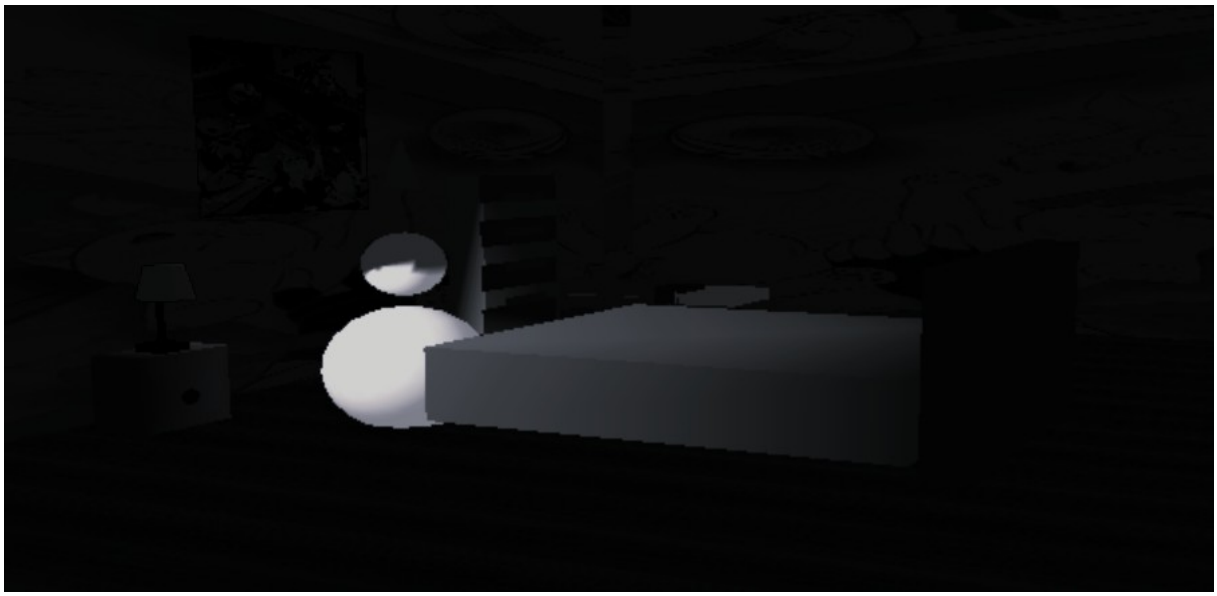
在片段着色器中，通过比较光空间坐标的深度值和深度图中的深度值来判断像素是否在阴影中。

```
// 片段着色器
#version 330 core
out vec4 FragColor;

in vec2 TexCoords;
in vec4 FragPosLightSpace;

uniform sampler2D texture_diffuse1;
uniform sampler2D shadowMap;

float ShadowCalculation(vec4 fragPosLightSpace) {
    // 进行透视除法，将坐标从齐次坐标系转换到标准化设备坐标系
    vec3 projCoords = fragPosLightSpace.xyz / fragPosLightSpace.w;
    // 转换到[0, 1]范围内
    projCoords = projCoords * 0.5 + 0.5;

    // 从深度图中获取当前片段的深度值
    float closestDepth = texture(shadowMap, projCoords.xy).r;
    // 获取当前片段的深度值
    float currentDepth = projCoords.z;
```

```
    // 计算阴影：当前深度值大于从深度图中获取的深度值时，表示在阴影中
    float shadow = currentDepth > closestDepth  ? 1.0 : 0.0;

    return shadow;
}

void main() {
    vec4 color = texture(texture_diffuse1, TexCoords);
    float shadow = ShadowCalculation(FragPosLightSpace);
    vec3 lighting = (1.0 - shadow) * vec3(1.0); // 简单地将阴影效果应用于光照结果

    FragColor = vec4(lighting * color.rgb, color.a);
}
```

7 实验结果分析

7.1 结果展示



可以看出，雪人自身发出的淡白色光芒在周围物体尤其是后面的书架侧面投下无锯齿柔和阴影

3.2 结果分析

对结果分析，可以看出：

1. **阴影精度**：
    - 由于采用了深度图技术，阴影的边界较为清晰且没有明显的锯齿现象。通过调整深度图的分辨率（如**SHADOW_WIDTH 和 SHADOW_HEIGHT**），可以进一步提高阴影的精度和质量。
2. **实时性能**：
    - 实时生成和应用阴影的性能表现良好。在较新的硬件上，阴影映射过程不会显著降低帧率，保持了较流畅的用户体验。即使在复杂场景下，帧缓冲对象和深度纹理的优化设置也确保了较高的渲染效率。

8 结论

1. 在 Room 项目中成功实现了阴影映射技术，并通过实验展示了其渲染效果。对比实验结果表明，应用阴影映射技术后，场景中的阴影效果显著提升，视觉真实感明显增强。渲染性能方面也表现出较好的实时性，满足实际应用需求。
2. 从结果可以看出，阴影映射技术在提升计算机图形学中场景的视觉真实感方面具有显著作用。通过生成光源视角下的深度图，可以有效地模拟物体之间的遮挡关系，使得场景中的光影变化更加逼真!

4.致谢

首先，我要感谢吴老师，在我的学习生活中给予了极大的理解和帮助。老师的专业知识和悉心指导，让我对图形学视角下的计算机科学初萌兴趣！其次，感谢 bilibili 上的相关课程和 CSDN 上的相关文章，这些文章为我提供了许多宝贵的参考资料和代码示例。最后还要特别感谢几本经典的计算机图形学教材 *Computer Graphics: Principles and Practice*. Addison-Wesley.不仅提供了扎实的理论知识，还包含了丰富的实践案例，使我在学习和应用阴影映射技术时受益匪浅！