**2 图形渲染综合实验**

**李俊辉 2201 U202217182**

**2.1 实验目的**

1. **熟悉OpenGL中的渲染流程**
2. **掌握一些OpenGL API的用法**
3. **了解一些渲染算法的具体实现**

**2.2 实验要求**

1. **参考所给的14个实验项目，在一个场景中，自行选择一些渲染算法来实现，确保这些算法能够协同运行；并回答以下问题：**
2. **简述你对OpenGL中，VAO,VBO,EBO的理解。**

1. VBO（Vertex Buffer Object，顶点缓冲对象）用于在GPU上存储大量顶点数据（如位置、法线、颜色、纹理坐标等），提高渲染效率。本质：是一个缓冲区对象，存储的是一大块原始的浮点数据。

2. EBO（Element Buffer Object，索引缓冲对象）用于存储顶点索引数据，通过索引复用顶点，减少重复数据，提高效率。本质：也是缓冲对象，只不过是专门用来存储绘图索引。

3. VAO（Vertex Array Object，顶点数组对象）记录和管理一组顶点属性配置（如：位置、颜色、纹理坐标的属性指针绑定），相当于一个“状态容器”。优势：切换 VAO 即可快速切换渲染对象，而无需重复设置顶点属性指针。

1. **说明你实现了哪些渲染算法，并简述其思想。**

我实现了颜色混合算法中的Alpha混合算法，加法混合算法。加权平均混合算法。

1.Alpha混合（传统透明）​​

公式：Final = SrcColor \* SrcAlpha + DestColor \* (1 - SrcAlpha)

原理：基于透明度通道的线性插值

优点：物理正确的透明效果，适合玻璃、水等真实材质

缺点：必须从远到近严格排序（CPU开销大）；可能出现边缘暗边（未预乘Alpha）

性能：（最差）

OpenGL设置：

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

2.加法混合（发光效果）​​

公式：Final = SrcColor \* 1 + DestColor \* 1

原理：直接叠加颜色值（忽略Alpha）

优点：无需排序，性能极高；适合发光体（火焰、魔法特效）

缺点：颜色会过曝（可能超出[0,1]范围）；无法表现真实透明

性能：（最佳）

OpenGL设置​：glBlendFunc(GL\_ONE, GL\_ONE);

3. 加权平均混合（优化版透明）​​

公式：Final = SrcColor \* 1 + DestColor \* (1 - SrcAlpha)

原理：预乘Alpha后的混合（类似Alpha混合的优化版）

优点：比Alpha混合更平滑（无暗边）；性能优于传统Alpha混合

缺点：仍需部分排序；需预乘Alpha纹理

性能：（中等）

OpenGL设置​：glBlendFunc(GL\_ONE, GL\_ONE\_MINUS\_SRC\_ALPHA);

1. **在将多个图形算法结合时，需要注意什么问题？遇到了什么难点？**

**需要注意的问题**​

渲染顺序与深度测试​

​不透明物体​：应先渲染（开启深度写入 GL\_DEPTH\_WRITE），避免被后续半透明物体错误遮挡。

​半透明物体​：必须从远到近排序（依赖CPU或GPU排序），否则混合结果不正确（如Alpha混合）。

​发光体（加法混合）​​：通常最后渲染（关闭深度写入），避免深度测试干扰发光效果。

混合模式冲突​

不同算法需动态切换混合函数（如 glBlendFunc），若未正确切换会导致颜色计算错误（例如发光体误用Alpha混合公式）。

​解决方案​：在渲染不同物体前显式设置混合模式，并保存/恢复OpenGL状态。

性能优化​

​状态切换开销​：频繁切换VAO、混合模式、Shader会增加GPU驱动负担。应尽量合并相同状态的渲染调用。

​排序代价​：Alpha混合需CPU排序半透明物体，可能成为性能瓶颈（可尝试分桶排序或近似排序）。

颜色范围与精度​

加法混合可能导致颜色值超出 [0,1]（过曝），需后续色调映射（如HDR或 glClampColor）。

加权平均混合要求纹理预乘Alpha，否则会出现颜色失真（如边缘变暗）。

深度缓冲与透明​

半透明物体若开启深度写入会阻止后续透明物体渲染，需通过 glDepthMask(GL\_FALSE) 临时禁用。

**实际遇到的难点**​

半透明物体排序问题​

​现象​：未排序时，透明玻璃窗与背景混合顺序错误，导致视觉异常。

​解决​：在CPU端计算物体到摄像机的距离并排序，但对动态物体（如移动的粒子）需每帧更新，性能损耗较大。

混合模式切换遗漏​

​现象​：UI提示框（加权平均）误用加法混合，导致颜色过亮。

​解决​：封装渲染逻辑，为每类物体预设混合状态（如使用状态机模式）。

深度测试与发光体冲突​

​现象​：火焰粒子（加法混合）被不透明物体深度测试剔除。

​解决​：渲染发光体时关闭深度写入（glDepthMask(GL\_FALSE)），但需确保其Shader不依赖深度值。

预乘Alpha的纹理处理​

​现象​：未预乘的UI纹理在加权平均混合中出现边缘黑边。

​解决​：在纹理导入时预处理（如Photoshop中勾选“预乘Alpha”），或在Shader中手动预乘。

**2.3 实验小结**

我学会了OpenGL渲染流程与核心对象，深入理解了 VAO、VBO、EBO 的作用与关系，掌握了顶点着色器和 片段着色器的基本编写方法，理解GPU渲染管线的工作流程。熟悉混合（Blending）算法与透明渲染，实现了三种混合模式：Alpha混合，加法混合，加权平均混合，理解了 预乘Alpha（Premultiplied Alpha）的重要性，避免边缘黑边问题。我学会了如何减少OpenGL状态切换（如VAO、混合模式、Shader切换）以提高性能，认识到渲染顺序对透明物体的影响，并尝试优化排序策略（如按深度排序），掌握了深度测试（Depth Test）与 混合（Blending） 的配合方式，避免渲染错误。