**\*\*\*\*RayTracerProMaxUltra\*\*\*\***

****1组员学号姓名及其承担的工作内容****

**李宣廷 23280224 实现了光线追踪算法，进行了光线求交、递归光线追踪**

**谭宇浩23280225 实现了Phong光照模型与阴影，完成实时渲染的功能**

**李岩 23280220 实现了场景加载与初始化、交互功能开发、相机控制**

****2系统概述****

本系统是一个基于 OpenGL 和 GLUT 的图形渲染程序，主要用于实现 3D 场景的渲染和交互。系统支持光线追踪算法，能够生成高质量的图像，并提供了键盘和鼠标交互功能，用户可以动态调整视角和场景内容。系统还支持加载 .obj 模型文件和纹理图像，能够渲染复杂的 3D 场景。

系统功能：

场景加载与渲染：支持加载 obj 格式的 3D 模型文件和纹理图像，使用 OpenGL 进行场景渲染，支持正交投影和透视投影。

光线追踪：实现基于光线追踪的渲染算法，支持漫反射、镜面反射和光照模型，通过递归计算光线路径生成高质量图像，支持多光源和阴影效果。

交互功能：支持键盘控制视角切换（前后左右移动、旋转等），支持鼠标点击事件动态调整场景，提供实时渲染和光线追踪两种模式，用户可通过按键切换。

图像输出：支持将渲染结果输出为 PPM 格式的图像文件，提供屏幕显示功能，用户可在窗口中查看渲染结果。

****3设计思路****

****3.1 场景漫游****

**移步换景渲染多个位置和视角和六个模型**

**实现了一个可交互的场景漫游功能，允许用户手动操作或者自动运行给定的路径进行漫游。用户可以通过键盘操作进行平移、旋转操作，调整相机的视角位置。具体实现过程中，W键向上旋转场景视图。按下W键场景视图将向前移动，可以更好地观察前方的模型。S键切换到后视图，移动场景视图。按下S键，场景视图将向后移动，可以更好地观察后方的模型。A键切换到右视图，向左移动场景视图(相机右移)。按下A键，场景视图将向左移动，可以更好地观察左侧的模型。D键切换到右视图，向右移动场景视图。按下D键，场景视图将向右移动，可以更好地观察右侧的模型，**L键导出场景图像光线渲染ppm文件。鼠标左键点击重置场景视图。按下鼠标左键，相机位置重置到原点**，所有变换通过实时更新相机位置和视向量完成。**

****漫游中的光照****

**在场景漫游过程中，为了提供真实感和可交互性，我们添加了对多光源光照和阴影的支持。每当用户改变视角时，系统会实时计算并更新不同光源位置和场景物体之间的光照效果，确保场景中物体表面光照的变化能够实时反映出当前视角的变化。此外，场景中还同时实现了阴影映射技术，光源产生的阴影能够动态显示，并随着相机的运动而调整其形态。**

****3.2 离线渲染****

****光照与阴影****

离线渲染部分，采用光线跟踪技术实现高质量的图像渲染。为每个像素发射光线，从相机视点出发，计算光线与场景物体的交点，并根据材质属性、光源位置、视角方向等因素，应用 Blinn-Phong 光照模型进行光照计算。该模型综合考虑了环境光、漫反射和镜面反射部分，确保了场景中物体的细节得到充分体现。在渲染过程中，除了光照和反射效果外，还实现了阴影映射。通过生成光源的深度图，判断物体表面是否在阴影中，从而正确渲染物体的阴影效果。阴影与光照结合，进一步增强了图像的真实感。

**纹理映射**

场景中的物体表面可以根据物体的纹理坐标从纹理图像中采样，得到纹理颜色并与计算得到的光照结果混合，渲染出更真实的视觉效果。

**消隐**

本系统中消隐的实现依赖于光线追踪算法。光线追踪通过从相机视点出发向场景发射光线，与所有场景物体表面进行交点计算，并通过对交点深度（即交点到相机的距离）进行比较，实现了深度消隐功能。

****4主要函数及其简要说明****

****initScene 函数**：初始化场景，加载模型文件并设置初始变换。**

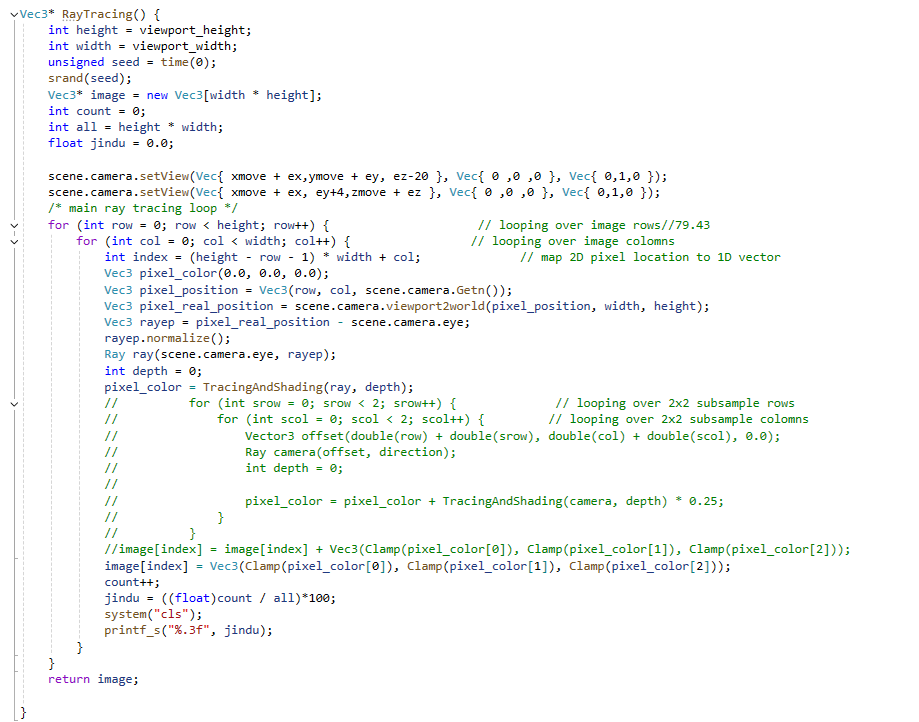
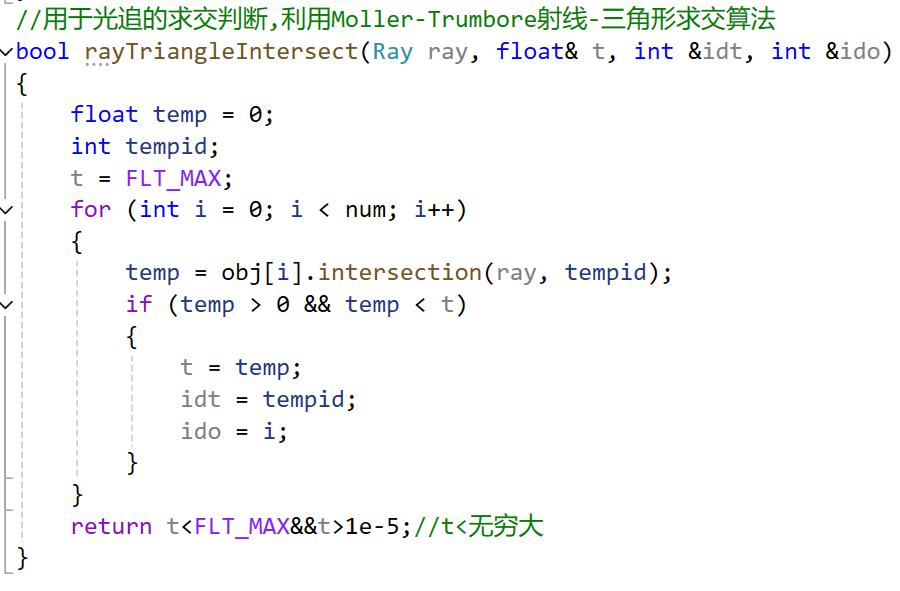
****display 函数：**绘制场景内容，支持实时渲染和离线渲染两种模式。state == 0：实时渲染模式，使用 OpenGL 绘制场景。state == 1：离线渲染模式，调用 RayTracing 函数进行光线追踪渲染。**

****keyboard 函数：**处理键盘输入事件，支持视角切换、光源调整和渲染模式切换。W/A/S/D：控制相机前后左右移动或者旋转视角。L：切换到离线渲染模式，调用 RayTracing 函数生成图像。G：切换回实时渲染模式。**

****reshape 函数：**处理窗口大小变化事件，调整视口和投影矩阵。**

****RayTracing 函数：**实现光线追踪算法，生成高质量图像。遍历画布中的每个像素，计算其颜色值。调用 TracingAndShading 函数递归计算光线路径和颜色。将生成的图像保存到 Canvas 类中，并输出为 PPM 文件。**

****TracingAndShading 函数：**计算光线与场景物体的交点，并根据光照模型生成颜色。使用 **rayTriangleIntersect 函数**计算光线与三角形的交点。根据交点位置和法向量计算漫反射、镜面反射和光照效果。递归调用自身实现反射和折射效果。**



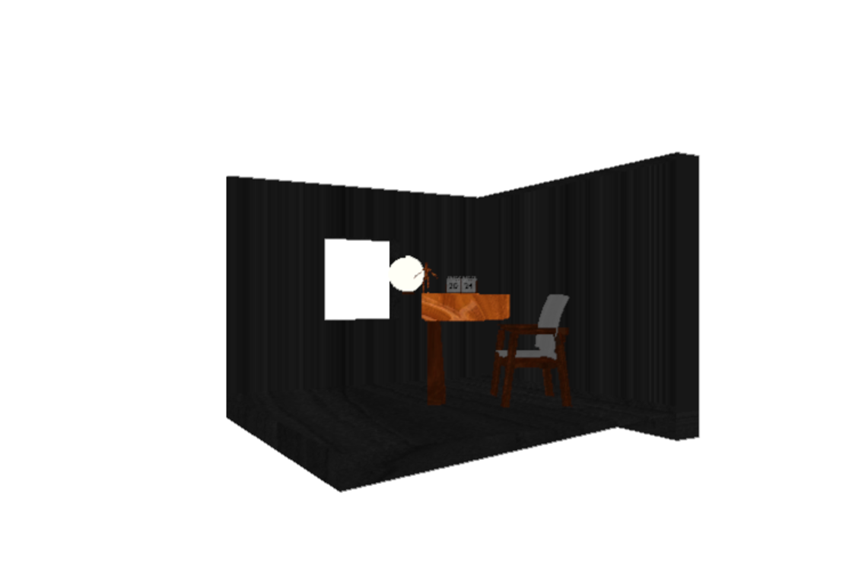
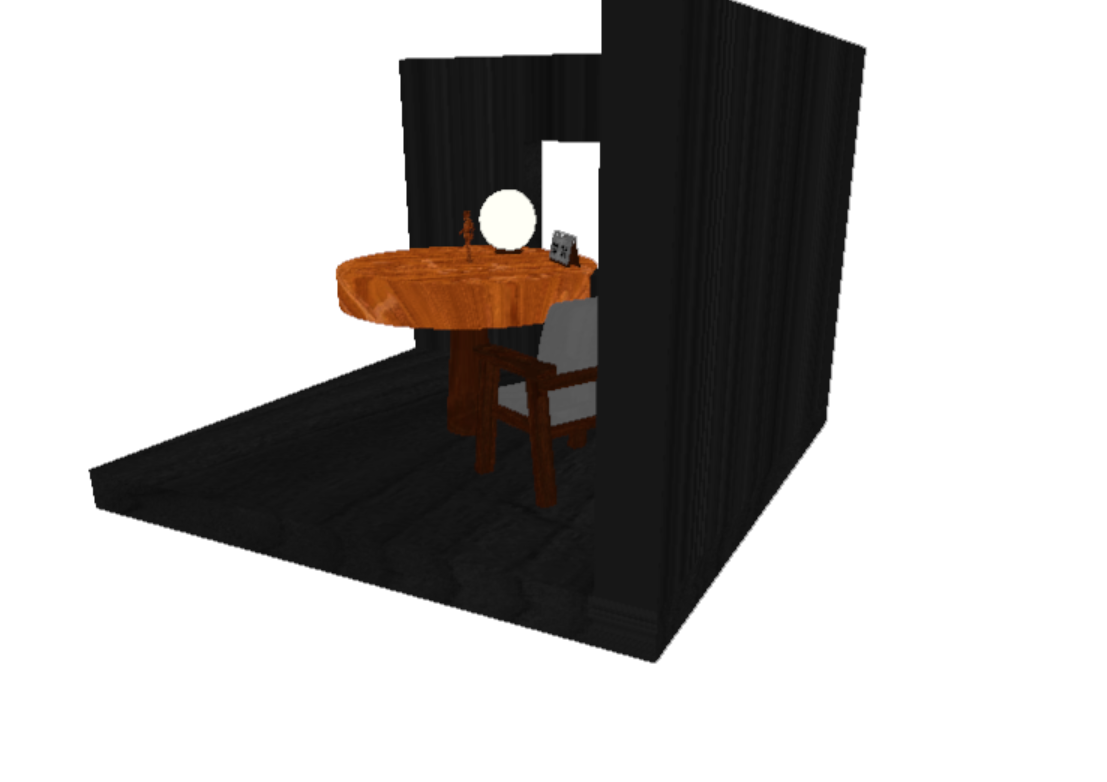
**5. 开发过程中遇到的主要问题及解决思路**

在开发过程中，遇到了多个关键问题，以下是主要问题及对应的解决思路：

* **光线跟踪算法效率问题**：光线跟踪渲染计算量极大，导致渲染速度缓慢，严重影响体验。为解决此问题，首先对场景进行了空间划分，采用了八叉树数据结构。通过这种方式，在光线与场景物体求交计算时，能够快速排除大量无关物体，减少不必要的计算量。同时，实现了光线的早期终止策略，当光线的贡献低于一定阈值时，提前结束光线追踪过程，进一步提升了渲染效率。
* **材质模型实现复杂**：精确模拟各种真实材质的光学特性非常复杂，不同材质的反射、折射、漫反射等属性差异很大。通过深入研究物理光学原理和相关文献，建立了一套基于物理的材质模型参数体系。我们收集了大量实际测量数据来确定其参数，如粗糙度、折射率等，并利用预计算的查找表来加速材质光照计算，在保证一定精度的前提下提高了计算速度。
* **交互操作响应延迟**：在处理鼠标和键盘的交互操作时，由于场景渲染和视图更新的计算量较大，出现了明显的响应延迟。采用了多线程技术，将渲染线程和交互线程分离。交互操作在主线程中立即响应，更新相机参数等相关信息，而渲染线程在后台根据新的参数进行场景渲染。同时，优化了视图变换算法，减少了不必要的矩阵运算，提高了视图更新的速度，使得交互操作更加流畅。

**6. 开发系统的运行结果**

* **运行效果图：**

**** 

* **效果描述**：整体效果在光影表现上较为出色，基于物理的光照模型准确地模拟了现实世界中的光线传播、阴影生成和材质光照效果。例如，在模拟金属材质物体时，高光部分的反射效果非常逼真，能够清晰地展现出金属的光泽和质感。场景中的间接光照也得到了较好的处理，使得阴影过渡自然，物体之间的光照相互影响表现合理，增强了场景的真实感和立体感。然而，在复杂场景下，仍然存在一些问题。部分区域的光线噪点较为明显，尤其是在光线较暗的角落或物体细节丰富的地方。这主要是由于在光线采样过程中，为了平衡渲染速度和效果，采样数量设置不够高，导致光线分布不均匀，从而产生噪点。

**7. 系统存在的不足**

* **性能瓶颈**：尽管采取了一系列优化措施，但在处理时，系统性能仍有待提高。随着场景物体数量和光线反射次数的增加，渲染时间会显著增长。
* **交互功能单一**：目前仅支持基本的鼠标和键盘操作来调整视图，缺乏更丰富的交互功能。这使得在操作和探索场景时的灵活性和便捷性受到限制，无法满足更复杂交互方式的需求。

****8.相应的参考资料****

[1] Pharr M, Jakob W, Humphreys G. Physically Based Rendering: From Theory to Implementation [J]. Morgan Kaufmann, 2016.   
[2] Kajiya J T. The Rendering Equation [J]. ACM SIGGRAPH Computer Graphics, 1986, 20 (4): 143-150.。  
[3] Veach E, Guibas L J. Metropolis Light Transport [J]. ACM SIGGRAPH Computer Graphics, 1997, 31 (2): 65-76.   
[4] Walter B, Marschner S R, Li H, et al. Microfacet Models for Refraction through Rough Surfaces [J]. ACM Transactions on Graphics (TOG), 2007, 26 (3): 1-10.   
[5] Cook R L, Torrance K E. A Reflectance Model for Computer Graphics [J]. ACM Transactions on Graphics (TOG), 1982, 1 (1): 7-24.   
[6] Hanrahan P, Salzman D. A Rapid Hierarchical Radiosity Algorithm [J]. ACM SIGGRAPH Computer Graphics, 1994, 28 (4): 197-206.   
[7] Irawan P, Chen M, Zhang E. Progressive Photon Mapping for Real-Time Global Illumination [J]. IEEE Transactions on Visualization and Computer Graphics, 2016, 22 (4): 1411-1424.   
[8] Stam J. Multiple Scattering as a Diffusion Process [J]. ACM SIGGRAPH Computer Graphics, 1995, 29 (4): 581-588.。  
[9] Sillion F X, Puech C. A General Two-Pass Method Integrating Specular and Diffuse Reflection [J]. ACM SIGGRAPH Computer Graphics, 1989, 23 (3): 335-344.。  
[10] Dachsbacher C, Stamminger M. Reflective Shadow Maps [J]. ACM SIGGRAPH Computer Graphics, 2005, 39 (4): 1003-1010.。  
[11] Lensch H P A, Kautz J, Goesele M, et al. Image-Based Re-lighting [J]. ACM Transactions on Graphics (TOG), 2003, 22 (3): 1-10.。  
[12] Bala K, Dorsey J, Rushmeier H. Rendering of Fur, Grass, and Other Natural Phenomena [J]. ACM SIGGRAPH Computer Graphics, 2003, 37 (4): 1024-1033.   
[13] McGuire M, Bavoil L, Singh M. Real-Time GPU-Based Ray Tracing [J]. ACM SIGGRAPH Computer Graphics, 2008, 42 (8): 1-8.   
[14] Sloan P P J, Kautz J, Snyder J. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments [J]. ACM SIGGRAPH Computer Graphics, 2002, 36 (3): 527-536.   
[15] Pharr M, Humphreys G. Graphics Hardware Ray Tracing [J]. ACM SIGGRAPH Computer Graphics, 2005, 39 (4): 703-712.  
[16] Ritschel T, Grosch T, Seidel H P. Approximate Reflectance Filters [J]. ACM Transactions on Graphics (TOG), 2009, 28 (5): 1-10.   
[17] d'Eon E, Luebke D. Advanced Techniques for Real-Time Global Illumination on GPUs [J]. IEEE Computer Graphics and Applications, 2011, 31 (4): 10-19.   
[18] Dachsbacher C, Vogelgsang C. Interactive Indirect Illumination Using Voxel Cone Tracing [J]. ACM SIGGRAPH Computer Graphics, 2010, 44 (8): 1-8.   
[19] Aila T, Laine S. Temporal Reprojection Anti-Aliasing in Call of Duty: Advanced Warfare [J]. ACM SIGGRAPH Talks, 2014, 1-2.   
[20] Mitchell D P. Hierarchical Photon Maps [J]. ACM SIGGRAPH Computer Graphics, 1996, 30 (4): 217-226.