不同粒度的光线追踪

叶增渝, 李超,

摘 要:

（中文不少于300字，摘要需要精炼，一般摘要三四句话，首句通过背景引入我们的工作，第二句介绍我们的方法，第三句，介绍我们优势和创新点。如果第二句或者第三句太长，可以拆成两句；）

不同粒度的光线追踪

叶增渝，李超

**Abstract**: （不少于300words，英文摘要，内容相同，不会使用长句，可用简单句，但是篇幅不宜过长）

**Key word：Inline、光线追踪、粒度、mesh shader、DirectX 12**

# 简介与意义/Introduction

## 项目意义和依据/Significance

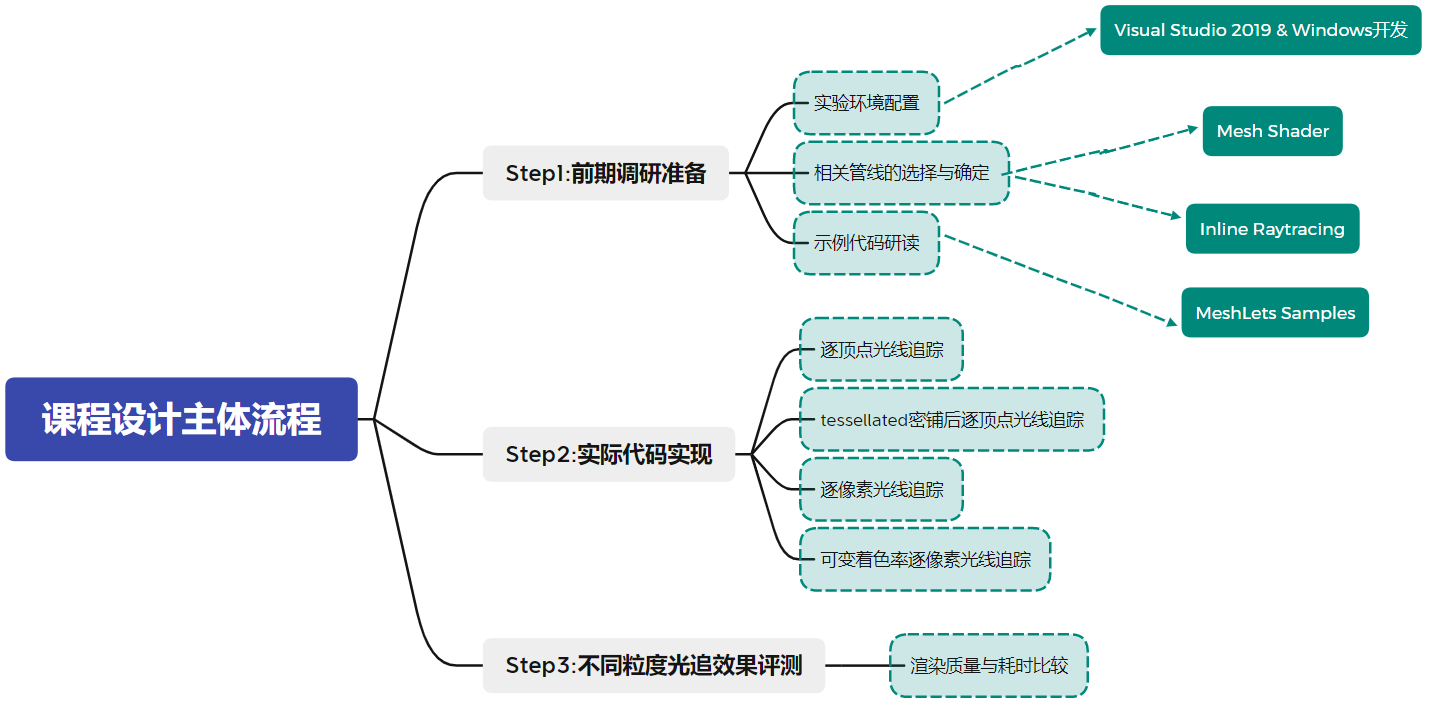
2018年,实时光线追踪技术元年。在这一年，微软宣布了DirectX Ray Tracing(DXR)的问世，Nvidia宣布了支持实时光线追踪技术的RTX系列显卡，同时诸如UE、Unity等引擎相继宣布兼容实时光线技术。此后，相关实时光线追踪技术得到了长足研究与发展。实现高效高质量的实时全局光线追踪能够使最终得到的渲染结果更加真实、立体，一直是科学界、工业界大力追求的一个研究方向。而由于光线追踪技术的耗时较长，为保持实时性，实现渲染耗时与渲染质量之间的平衡成为了一项重要的课题。而这项课题需要制定合理的实时渲染预算与方案、善用光照传输缓存技术，甚至探索全新的混合渲染管线等等，具有巨大的研究意义。

DX9时代低频球谐光照就可以逐顶点着色并保持一定质量，目前依旧应用于Unity等引擎的默认光照模式。1spp路径追踪配合降噪可以近似实现高效高质量实时全局光照，但其耗时依旧为传统屏幕空间反射、阴影的两三倍。在本课程的学习中本小组了解到，从到Gauroud Shading对每一个面片进行相同的颜色插值，到对一个面片进行线性插值赋予颜色，再到Phong Shading的像素层级进行的颜色赋值。我们不难知道，一个粗粒度的光线追踪所需要计算量相对较少，那么耗时相对更加细粒度的光线追踪更少，而从质量方面看，假设面片的数量较少，我们会比较容易看到一个个色块，获得的结果也会质量较低，但如果面片足够小，那么即使是粗粒度的光线追踪也能达到良好的效果，但具体如何寻找两者的平衡点，还是需要进行实际评估。因此，值得尝试更进一步稀疏化光线样本的粒度，减少光线追踪数量，在性能与质量之间找到一个更加良好的平衡点，由于实时光线追踪技术在图形学中尚处于初步探索发展阶段，这样的探索也并非易事，具有很强的探索意义，能为未来的游戏等渲染场景提供能力更加平衡的实时光线追踪方案。

因此，本小组探索一种稀疏采样方式，提高有效采样率尽可能保证质量并带来因光线追踪数减小的性能提升。首先本小组通过前期调研了解并决定使用的图形管线与光线追踪形式，并计划先研读微软官方文档与mesh shader sample的代码实例，了解对应管线从初始化到销毁的全部流程与hlsl文件的编写规范，最终确定

需要修改的shader stage，在不同要求下在不同位置加入光线追踪程序，最终实现不同粒度的光线追踪

## 本方法/系统框架/Article Structure



# 相关工作/Related Works

1.现代常用引擎默认使用的球谐光照的原理了解

2.1spp路径配合降噪实现的高效高质量全局实时光照实现分析

3.业界成熟的实时光线追踪方案了解,诸如经典的KDTre等[1][2]。

# 研究内容与方法(或算法)/Contents and Methods(or Algorithm)

工作环境：

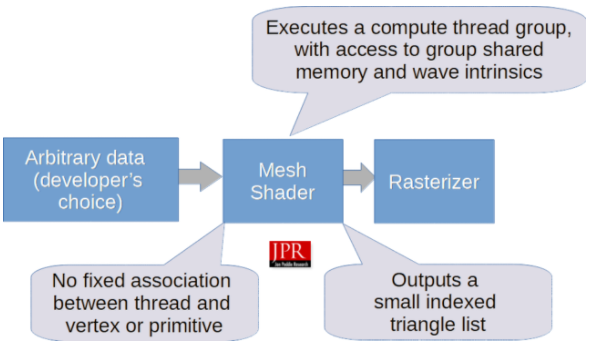
（1）DirectX 12库(使用dxdiag诊断，一般Windows电脑已含有其SDK，无需下载)

（2）GPU： NVIDIA GeForce RTX 2060

（3）代码环境：Viusal Studio 2019（通用Windows平台开发）

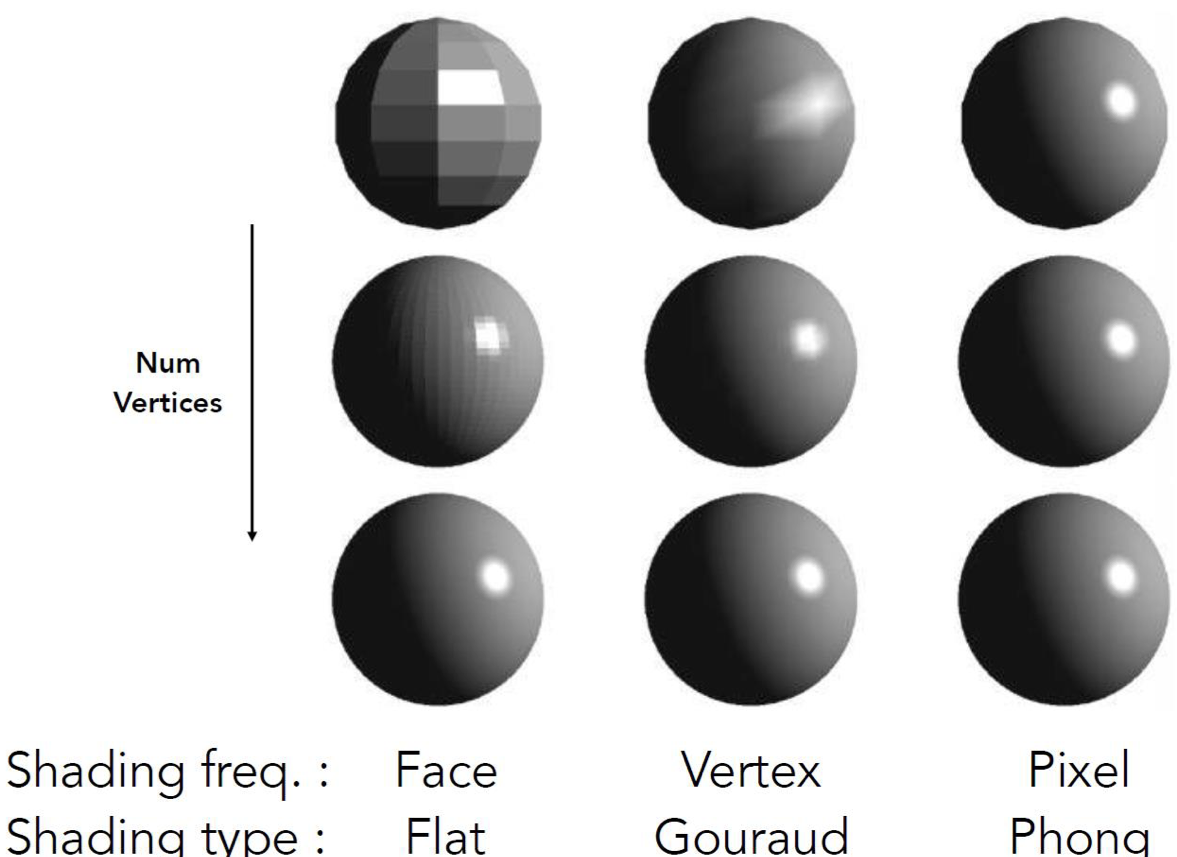
前期调研:

2019年，在DirectX中支持的图形新管线mesh shader，它将原先的图形管线中pixel shader前的部分简化为amplification shader与mesh shader，并于2020年提出了一个通用compute shader—meshlet，本小组决定采用这种新型图形管线进行相关工作以提高效率；其次，传统地想要实现光线追踪的功能，往往需要新建光线追踪加速结构并构建一个相应的光追管线，这对于本小组改变光线追踪的粒度进行测试带来了一定的程序迁移困难，幸运的是实时光线追踪技术提出了一种不需要额外独立管线的光线追踪实现形式—inline raytracing，通过简单地定义加速结构与RayQuery，我们可以在任意shader stage通过添加inline raytracing相关函数实现不同阶段、粒度的光线追踪，大大减少了重复代码量。



Mesh Shader

根据本课程了解到的图形学课程知识可知粗细粒度的划分起源于shading时采用的方式与选择的对象（如Phong Shading与Gauroud Shading之间的差别），那么我们可以根据shader stage的不同将本项目中的粒度粗略划分为逐顶点、 tessellated密铺后逐顶点、逐像素、可变着色率逐像素四种光线追踪粒度。



不同shading结果示例

综上，本小组决定采用DirectX 12下的mesh shader与inline raytracing进行工程实现上述四种不同粒度的光线追踪并进行比较。

前期知识准备：

（1）DirectX 12入门：

参考DircetX 12龙书及Microsoft公布的DirectX相关示例源码中的HelloWorld与RayTracing相关示例代码进行学习，对内部采用的各类函数通过Microsoft相关开发文档进行了解学习，掌握一个程序的基本组件极其依赖关系

（2）mesh shader入门 :

参考Microsoft公布的DirectX相关示例源码MeshShaders与Microsoft相关开发文档进行了解学习

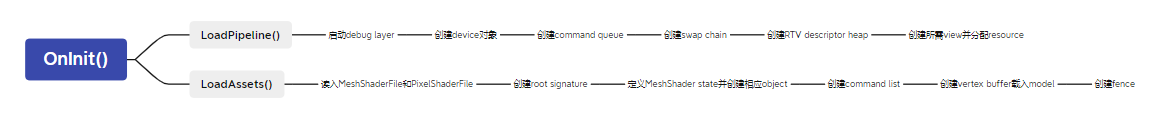
（3）inline raytracing入门 :

参考DXR 1.1的相关文献与档案进行了解学习

（4）可变着色率VRS(variable rate shading)入门：

参考Microsoft相关开发文档相关开发文档进行了解学习

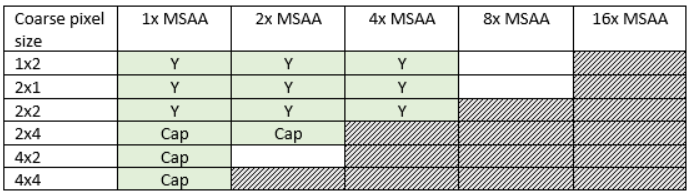
解决方案：



通过研读mesh shader的示例代码，不难发现在OnInit()部分，我们读入了两个File，分别对应mesh shader与pixel shader，然后根据自身的model与两个hlsl文件的内容进行渲染，我们想要在其中实现inline raytracing，那么就应当在两个hlsl文件中进行shader进行修改。

在两个代表两个shader的hlsl文件中找到对应部分的位置，添加inline raytracing内容并判断光线可能遇到的各种情况，实现光线追踪，而不同粒度的实现应当是在shader的合适阶段添加raytracing从而产生不同的效果。

其中比较特殊的是Variable-rate Shading，它需要对不同大小的Coarse pixel size采用不同的采样方式，需要根据size确定shading rate，还需要更进一步的研究。



Variable-rate Shading shading rate table

由于整个项目并未完全完成，故到各粒度的光线追踪的具体实现方案还未最终确定，可能会有小幅改动，但基本思路如上所述。

# 实验结果与分析/Experiment Results and Analysis

# 特色与创新/ Distinctive or Innovation Points

# 补充说明：

6.1、术语：请对本领域的技术词语进行解释说明，如果有英文要给出中文注释或解释

|  |
| --- |
| Inline RayTracing:种光线追踪技术，使得不再需要新建一个独立的光线追踪管线，易于在各个shader stage进行不同粒度的光线追踪  Mesh Shader:于2019年实现部署的图形新管线，将pixel shader前的步骤简化为amplification shader与mesh shader  Variable-rate Shading:基于自身的定义对图像的不同区域采用不同的shading rate，以期解决性能无法维持全局相同shading的限制 |

References:

1. Andreas Polychronakis, George Alex Koulieris, and Katerina Mania. 2021. Emulating Foveated Path Tracing. In Motion, Interaction and Games (MIG '21). Association for Computing Machinery, New York, NY, USA, Article 10, 1–9.
2. Daniel Reiter Horn, Jeremy Sugerman, Mike Houston, and Pat Hanrahan. 2007. Interactive k-d tree GPU raytracing. In Proceedings of the 2007 symposium on Interactive 3D graphics and games (I3D '07). Association for Computing Machinery, New York, NY, USA, 167–174.

Docs Webistes:

1. DirectX 12示例代码:https://github.com/microsoft/DirectX-Graphics-Samples/tree/master/Samples/Desktop
2. DirectX 12开发文档:https://docs.microsoft.com/en-us/windows/win32/direct3d12/direct3d-12-graphics
3. VRS文档: https://docs.microsoft.com/en-us/windows/win32/direct3d12/vrs
4. Inline Raytracing文档: https://github.com/microsoft/DirectX-Specs/blob/master/d3d/Raytracing.md#inline-raytracing

时间安排与分工统计表

|  |  |  |  |
| --- | --- | --- | --- |
| **组员信息**（含组长） | | | |
| 学生姓名 | 叶增渝 | 学 号 | 519030910168 |
| 项目分工 | DX12项目生成，样例注释，功能实现，报告编写 | | |
| 学生姓名 | 李超 | 学 号 | 519030910198 |
| 项目分工 | DX12项目代码功能实现，报告编写 | | |
| 学生姓名 |  | 学 号 |  |
| 项目分工 |  | | |
| **时间安排/**  **Schedule** | 2021.10.29选题  2021.10.30-2021.11.14前期调研  2021.11.15-2021.11.28前期知识储备与学习  2021.11.29-2021.12.19研制开发，代码调试  2021.12.04-2021.12.05撰写中期报告  2021.12.20-2021.12.25总结实验结果，撰写总结报告，准备presentation | | |