A5 Project Proposal

Title: Real-time ClusterLOD Renderer

Name: Yang Chen
Student ID: 20816397
User ID: y2588che

1 Purpose

To tie together three totally unrelated rendering issues.

2 Statement

For Ray Tracers: Paragraph describing interesting scene to be rendered and what features are needed to achieve this scene.

Paragraph: What it's about.

Paragraph: What to do.

Paragraph: Why it is interesting and challenging.

Paragraph: What I will learn

3 Technical Outline

This project aims to enhance mesh rendering through efficient Level of Detail (LOD) management, starting with partitioning the mesh into spatial clusters using METIS and simplifying them with Quadric Error Metrics (QEM) to form a multi-level LOD hierarchy. A Bounding Volume Hierarchy (BVH) will then organize these LODs, supporting fast, compute shader-based selection and streaming of visible clusters. A visibility buffer will store per-pixel geometry data, enabling deferred shading, while additional effects like Screen Space Reflection (SSR), Ambient Occlusion (SSAO), and Multi-Sample Anti-Aliasing (MSAA) will improve realism without excessive computational cost. Post-processing effects (bloom, tone mapping, and color grading) and the Disney Principled BSDF shader for realistic material rendering will further enhance visual quality, all while utilizing an indexed mesh structure to reduce memory usage.

4 Bibliography

Articles and/or books with important information on the topics of the project are referenced below.

- Visibility Buffer techniques are explored in [1].
- Nanite technology for massive model visualization is detailed in [2].
- Deferred Shading methods are discussed in [3].
- Screen Space Ambient Occlusion (SSAO) is explained in [4].

Objectives:

___ 1: [PRECOMPUTE-GEO] Mesh Partition
Use the METIS library to partition the triangle mesh into multiple clusters, each representing a group of spatially adjacent triangles. This partitioning process defines the 0-LOD (highest detail) mesh. To optimize for multi-level rendering, select **N** adjacent clusters and combine them into larger entities known as Cluster Groups, which can later be used to construct higher levels of detail.

___ 2: [PRECOMPUTE-GEO] QEM Simplification
Apply the Quadric Error Metric (QEM) simplification algorithm to the previously created Cluster Groups. This simplification process reduces the number of triangles in each cluster while preserving the visual fidelity as much as possible. Track the simplification error for each cluster and Cluster Group to guide dynamic LOD selection later. This error metric will be critical for runtime decisions on which clusters to render at different distances.

___ 3: [PRECOMPUTE-GEO] LOD Error Tree Management
Construct a Level of Detail (LOD) hierarchy by recursively applying the mesh partitioning and QEM simplification steps (Objectives 1 and 2) to generate progressively lower levels of detail. During this process, ensure that the simplification error grows monotonically with each LOD level. This hierarchical structure will allow for smooth transitions between LODs at runtime.

___ 4: [PRECOMPUTE-GEO] LOD-BVH Construction
Create a Bounding Volume Hierarchy (BVH) for each LOD level using a Surface Area Heuristic (SAH) approach. In this BVH, each node represents a Cluster Group, which allows for efficient spatial queries. This BVH structure will support fast LOD selection and culling during rendering by organizing Cluster Groups spatially and hierarchically.

___ 5: [RUNTIME-GEO] LOD Selection & Streaming
At runtime, implement LOD selection and streaming using a Compute Shader. A job scheduler system traverses the LOD-BVH, identifying which Cluster Groups are within the view frustum and within a desired error threshold. Worker threads then go through clusters in each selected Cluster Group to further refine visibility based on error metrics. This hierarchical approach enables selective loading of clusters based on viewer distance and screen resolution.

___ 6: [RUNTIME-GEO] Visibility Buffer
Implement a visibility buffer to record per-pixel information about the scene. For each pixel, store the barycentric coordinates of the triangle it falls within, along with the depth information. This buffer will be essential for deferred shading, as it enables deferred rendering without storing full geometry data per frame.

___ 7: [RUNTIME-PIPELINE] Deferred Shading
Utilize the visibility buffer to perform deferred shading, allowing for efficient lighting and shading calculations. Instead of recalculating light interactions per triangle, deferred shading operates directly on the per-pixel visibility data, reducing the computational load and enhancing performance in scenes with multiple light sources.

___ 8: [RUNTIME-SHADING] Screen Space Reflection (SSR) and Ambient Occlusion (SSAO)
Implement SSR and SSAO effects using the visibility buffer. Screen Space Reflection adds realistic reflections by tracing rays in screen space, while Ambient Occlusion provides subtle shading in occluded areas. These effects add realism to the scene without requiring ray tracing, balancing visual fidelity with performance.

___ 9: [RUNTIME-SHADING] Post-Processing Effects
Apply a range of post-processing effects, including bloom, tone mapping, and color grading, to the final image. These effects use the visibility buffer to add cinematic visual quality and enhance color and brightness dynamics in the rendered scene.

___ 10: [PRECOMPUTE-OPTIM] Index Buffer Storage Utilization
Modify the mesh structure to use indexed vertices, which reduces memory usage by reusing shared vertices across triangles. By switching to an indexed format, the overall data footprint for vertex storage is minimized, especially in large meshes with numerous shared edges.

___ 11: [ADDI RUNTIME-OPTIM] Multi-Sample Anti-Aliasing (MSAA)
Implement MSAA to improve visual quality by reducing jagged edges in rendered images. Enable MSAA before creating the visibility buffer to ensure that anti-aliasing is applied during shading. This approach smooths out edges and contributes to the overall visual appeal of the rendering.

___ 12: [ADDI RUNTIME-SHADING] Disney Principled BSDF Shader
Integrate the Disney Principled BSDF shader model to render materials more realistically. This shader model, developed by Disney, offers a unified approach to rendering different material types, balancing realism with performance, and simplifying shader complexity. It allows for smoother transitions between material properties and supports a wide variety of looks.

References

[1] Pascal Hillaire. *Visibility Buffer Rendering with Material Graphs.* `http://filmicworlds.com/blog/visibility-buffer-rendering-with-material-graphs/`. Accessed: 2024-11-05.

[2] Brian Karis. “Nanite: Towards Real-Time Massive Model Visualization”. In: *SIGGRAPH Advances.* `https://advances.realtimerendering.com/s2021/Karis_Nanite_SIGGRAPH_Advances_2021_final.pdf`. 2021.

[3] Joey de Vries. *Deferred Shading.* `https://learnopengl.com/Advanced-Lighting/Deferred-Shading`. Accessed: 2024-11-05.

[4] Joey de Vries. *SSAO.* `https://learnopengl.com/Advanced-Lighting/SSAO`. Accessed: 2024-11-05.