FH University of Applied Sciences TECHNIKUM WIEN

Proposal

Degree Program BIF

Course Scientific Writing

Design and Implementation of a 3D Scene in OpenGL Using Ray Tracing and Rasterization: A Comparative Study

By: Paul Felgitsch
Student number: if22b151

Advisor: Wolfgang Berger

Vienna, 4.11.2024

Introduction/ Background

Computer graphics is a term that describes a computer which produces and manipulates images [1]. If these images are created tightly after one another, so that the human eye cannot distinguish between them, it forms an immersive and dynamic process. This is called real-time rendering. To produce those individual images, there are two dominant methods in the modern era.

The first one is rasterization. It has been the dominant technique in real-time rendering space for decades. Rasterization projects 3D data onto a 2D plane. It does that through determining which pixels are covered by a given triangle. This procedure is called coverage computation [2]. With this data, the individual pixels are provided with coloring, texture mapping, lighting and shading through mathematical calculations. This allows for a 2D image with 3D data.

On the other hand, ray tracing calculates the closest point of intersection of a ray on an object to its origin. For the computer, this is a time consuming operation because this has to be done for each individual ray [3].

Because rasterization is faster and more efficient than ray tracing, it is used in real-time applications like video games, whereas ray tracing is seen in static images and pre calculated scenes like movies or television shows. Nowadays, it is possible to use ray tracing in real-time applications. For example, with NVIDIA’s RTX technology it is feasible to offer improved visual fidelity while maintaining frame rates in games and dynamic environments. Nevertheless, real-time ray tracing has certain limitations that need to be studied to improve performance and make it more feasible for broader use. To use real-time ray tracing, specific hardware is still needed. On the lower end of computer hardware, it does not allow you for heavy computational operations like that. Moreover, over the years, scenes in video games have become increasingly complex. The amount of light source, shadows, reflections and objects with all different kinds of materials contribute to detailed scene architecture.

Research question

In this paper, I want to identify use-case scenarios where one technique is more advantageous than the other. To accomplish this, both techniques will be used to generate the same scene and analyzed by different parameters.

- Research question
  - How do ray tracing and rasterization differ in terms of rendering performance, visual fidelity, and computational resource requirements when implemented in the same 3D scene in OpenGL?
  - In what scenarios or applications is one rendering technique more suitable than the other based on the results of this comparative study?
- Hypotheses:
  - RQ1:
    - At the cost of higher computational resource consumption, ray tracing will produce higher visual fidelity than rasterization in the same 3D scene.
  - RQ2:
    - In video games and real-time rendering scenarios, where speed and efficiency are crucial, rasterization is more suitable. In contrast, ray tracing is better suited for applications requiring high-quality visual effects, such as realistic lighting and reflections.

Methods

Comparative Methods:

- Parameter Testing in different Scene complexities:
    - Performance:
        - Frames per Second
        - Latency (Time per Frame)
    - Memory Usage:
        - Consumption
    - Scalability with Hardware:
        - Different Hardware Configurations create different outcomes
        - Parallelism and multi-threading
- Image Quality:
    - Structural Similarity Index (Algorithm to measure how similar a picture is to a reference picture from 0 to 1)
    - Survey: Asking people to rate the scene from 1 to 10 on how visually pleasing it looks. (Qualitative metric)

I chose these methods because the effect of comparing two techniques that handle the same problems differently will be shown in detail with comparison. To do that, I will gather a lot of quantitative data and statistically analyze them. Moreover, the SSI (Structural Similarity Index) is a great tool for comparing static images to reference pictures. This is often used for films. Because Image Quality can be a subjective matter, I want to ask people what they think of both scenes. On a scale from 1 to 10, how visually pleasing is that scene to you? Which part of the scene immediately caught your attention? These kinds of questions are important to ask because they help us understand individual perceptions and preferences, providing insights into subjective experiences.

I expect high SSIM scores between rasterization and ray-traced images in scenarios with simple lighting conditions, as rasterization can effectively produce visually similar results under these circumstances. Conversely, we anticipate low SSIM scores in complex scenes, where ray tracing's strengths—realistic lighting, shadows, reflections, and transparency—create significant visual differences that rasterization cannot replicate accurately. Furthermore, rasterization will outperform ray tracing in real-time applications with regards to performance. In this instance, it is expected that the frame rate (FPS) will be higher, accompanied by reduced latency.

References

[1] S. Marschner and P. Shirley, *Fundamentals of Computer Graphics*, 5th ed. New York: A K Peters/CRC Press, 2021. doi: 10.1201/9781003050339.

[2] T. Davidovič, et al., "3D rasterization: a bridge between rasterization and ray casting,"

Proc. Graphics Interface 2012, 2012

[3] A. S. Glassner, *An Introduction to Ray Tracing*. Morgan Kaufmann, 1989.