Exploring Ray-Space Hierarchy on the GPU for Ray-Tracing of Dynamic Scenes

Nuno Tiago Reis

Instituto Superior Técnico, Portugal

Abstract

Overview of the work i produced.

1. Introduction

Ray-tracing is a global illumination technique used for the synthesis of photo-realistic images that involves ray-casting. The first ray-casting algorithm was presented by Arthur Appel [App68], although it was only referred to as such until later on. This algorithm involved casting a set number of rays from the eye and intersecting those rays with the geometry composing the scene.

In 1979 Whitted [Whi80] presented a paper introducing a recursive ray-tracing algorithm, which is now known as Whitted Ray-Tracing. This algorithm casts rays, much like the original algorithm, casts rays from the eye but doesn't stop there. If the rays intersect geometry they can generate three different kinds of rays: shadow, reflection and refraction rays. These rays are also called secondary rays.

Shadow rays are cast towards the light sources present in the scene. These rays are cast to check if the intersected object is being illuminated directly or if there are other objects blocking the light source. Reflection rays are cast in a mirror-reflection direction. However, these rays are only cast if the intersected object has reflective properties. Refraction rays are cast through objects. This means that the intersected object must be translucent and that the refraction direction also depends on the material of both the rays current and next travel mediums refraction index.

These rays differentiate ray-tracing from rasterization since they allow realistic and complex effects like reflections, refractions and shadows. This doesn't come without a price however. Ray-tracing is very inefficient and up to this day there has been extensive research to try and optimize it. Most of this research involves creating hierarchies.

In Whitted Ray-Tracing [Whi80] each ray is tested with each polygon in the scene, which leads to N x M intersection tests per frame, assuming N rays and M polygons. This isn't feasible, especially with moderately complex scenes.

Object-space hierarchies help reduce the number of intersections by creating a structure that allows the scenes geometry to be tested for intersections without actually testing every single polygon. These techniques have problems handling dynamic scenes since the hierarchies must be rebuilt at each frame.

Ray-space hierarchies have the same goal, to reduce the number of intersections, but they approach the problem differently. One of the solutions is ray-bundling, which consists in grouping several coherent rays together and intersect the bundle of rays with the geometry. This saves many intersection tests since if a bundle doesn't intersect any geometry then the rays in said bundle will not be tested anymore in that particular frame. Another solution involve caching of rays, using 5D tuples to represent them. In these solutions the rays are represented as hypercubes in 5D space, which are then intersected with the scenes geometry. These intersection results are then cached and used to reduce future intersection tests for new rays.

Over the last few years many improvements have been made in both object-space and ray-space approaches. The development of General-purpose computing on Graphic Processing Units (GPGPU) over the past years has led to many attempts to implement Ray-tracing on the GPU. This is no simple task however. Modern GPUs are composed of Single instruction, Multiple data units (SIMD), which means that they are optimized to repeat the same operations over large

sets of input. Ray-tracing by nature isn't easily mapped onto GPUs since each ray can generate secondary rays, but it also might not do so, which is a problem since each ray-cast has the potential to have a different execution than the last one.

In this paper I will analyse two different approaches based on ray-space hierarchies, one using caching mechanisms and one using ray bundles. Finally I will propose an improvement to a ray-bundle approach, combining both ray-space and object-space hierarchies to achieve higher efficiency in the ray-object intersection tests. This ray-bundling approach will also be parallelized so that it is mapped more efficiently on the GPU, leading to better performance, with the goal of generating photorealistic images in real time.

For all figures please keep in mind that you **must not** use images with transparent background!

Figure 1: *Here is a sample figure.*

2. Related Work

[App68]

[AK87]

[GL10]

[RAH]

[SD]

[Whi80]

3. Original Algorithm

TODO

3.1. Overview

TODO

3.2. Improved Algorithm

TODO

3.3. Overview

TODO

4. Analysis

TODO

5. Results

TODO

References

[AK87] ARVO J., KIRK D.: Fast ray tracing by ray classification. *ACM SIGGRAPH Computer Graphics 21*, 4 (July 1987).

[App68] APPEL A.: Some techniques for shading machine renderings of solids, 1968.

[GL10] GARANZHA K., LOOP C.: Fast ray sorting and breadth-first packet traversal for gpu ray tracing. *Eurographics 29*, 2 (2010).

[RAH] ROGER D., ASSARSSON U., HOLZSCHUCH N.: Whitted ray-tracing for dynamic scenes using a ray-space hierarchy on the gpu.

[SD] SIMIAKIS G., DAY A. M.: Five-dimensional adaptive subdivision for ray tracing. *Computer Graphics forum 13*, 2, 133–140.

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Communications of the ACM 23* (1980).