Abstract

This presentation explores the practical side of ray tracing for games. First some myths regarding ray tracing are rectified. After that, an explanation of the practical techniques that exist is given. Then, the hybrid approach of combining rasterized graphics and ray tracing is explored in more detail. After an intermezzo explaining how ray tracing works on a basic level, details and applications of ray tracing are discussed. The presentation ends with a series of examples of applications of the ray tracing and rasterization hybrid approach. These examples include shadows, reflections and transparency.

1 Summary of Talk

1.1 Introduction

Traditionally, games have been using rasterization as the method to project the scene onto the screen. Because ray tracing has a history of being relatively heavy on computations that need to be performed in real-time, developers of games and game engine have used rasterization for the majority of games that have been developed so far. At least, this is the general view. In this presentation the presenters give some nuance to these generally accepted views on the differences between rasterization and ray tracing.

1.2 Myths Debunked

Myth: Ray tracing is only for photorealistic / physically accurate rendering. Explanation: Ray tracing is a tool, and like any tool it can be used for a range of purposes. That means that ray tracing can be used in real-time graphics processing.

Myth: Ray tracing is incompatible with rasterized graphics. Explanation: In order to achieve certain effects, ray tracing can be combined with rasterized graphics in games.

Myth: Ray tracing is a less efficient way to render a given number of pixels. Explanation: It can actually be computationally cheaper to use ray tracing for some effects.

1.3 How Does Ray Tracing work?

The first step in ray tracing is to submit the geometry to the ray tracer. This means that all the triangles in the scene are stored in a database in the ray tracer. The second step is to describe what the scene should look like when ray traced. In other words, for each triangle a shader needs to be defined to define what happens when a ray intersects the scene. The final step is to shoot some primary rays for each pixel so that the ray tracing process can get going.

1.4 Possibilities of Ray Tracing

Ray tracing can be used to create a very wide range of realistic effects in your scene. Including shadows, reflections and global illumination.
There are a number of approaches you can take to add raytracing to your game. One of these possibilities is to simply remove all rasterization code from your game: Simply ray trace everything! This would give the most realistic / beautiful scene, but comes at the cost of having to perform unrealistically heavy computations.

A less computationally expensive way of adding ray tracing to your game is to use a hybrid model and combine rasterization and ray tracing. In this model the first step is to render the scene into a so-called G-buffer. This step is done as in a normal rasterization pipeline. This G-buffer contains all the information necessary to ray trace. Based on the G-buffer, it must be decided which pixels to use for ray tracing, and which effects should be created my means of ray tracing.

2 Overview and Relevance

2.1 Relevance

The topic of ray tracing in video games in extremely relevant today. Due to an increase in computational resources, it becoming more and more likely that we will see ray traced games in the near future. Friedrich et al. have already made such claims in 2006, and although nowadays there are not yet many real-time ray traced games available, there is reason to believe that we will see such games in the near future.

2.2 Overview

As said before, already since at least 2006 there has been talk of real-time ray traced games. The main problem that is being faced right now is hardware. GPUs today are specifically designed for rasterization. On GPUs vector calculations are heavily optimized, while ray tracing cannot be done. In order to have ray tracing in a game the CPU needs to be used, which is often slower than the GPU. In order for real-time ray traced games to become viable, either new hardware needs to be designed for ray tracing, a process which is very expensive, or commodity CPUs need to become powerful enough to be handle the complex calculations involved in real-time ray tracing.

3 References and Further Sources