AIN SHAMS UNIVERSITY
FACULTY OF COMPUTER & INFORMATION SCIENCES
DEPARTMENT OF SCIENTIFIC COMPUTING

Performance Optimization of Image Rendering on Programmable Graphics Hardware

A Thesis Submitted to the Department of Scientific Computing, Faculty of Computer and Information Sciences, Ain Shams University, in Partial Fulfilment of the Requirements for the Master Degree of Computer and Information Sciences

By

Ahmed Osman Aiad

Supervised by

Prof. Dr. Mohammed Essam Khalifa
Dean of Faculty of Computer and Information Sciences,
Ain Shams University

Dr. Ahmed Hamed Kaboudan
Computer Science Department,
Higher Institute of Computer Science
And Information Technology

Dr. Safwat Helmy Hamad
Scientific Computing Department,
Faculty of Computer and Information Sciences,
Ain Shams University

Cairo, 2010

ACKNOWLEDGMENTS

I am deeply indebted to, Prof. Mohammed Essam Khalifa, for his endless help, and support during last year. His advice and insight have been invaluable throughout my entire time of research. Our meetings gave me academic guidance and provided the milestones that helped to develop my research skill.

I want to express my deep gratitude to Dr. Ahmed Kaboudan for his Significant and academic support. His continuous contact with me gave me a really strong push toward achieving my research goals.

I would like to extend my thanks to Dr. Safwat Hamad for his continuous encouragement, understanding and support. He stood by me throughout all the difficulties I faced during the last year. He pushed me to fill in that one last detail to elevate the level of my thinking and my work. I also admire and respect his integrity both as a researcher and as a person

I won't forget to also thank Prof. Dr. Ahmed Gabr and Dr. Ahmed alabbassy, who gave me a lot of time and effort. I cannot possibly thank them enough.

Finally I am really grateful to my wife for her deep caring and very strong encouragement that helped me in finishing this work. Furthermore, this thesis would have never been possible without the support of my family, I am deeply grateful for my father and my mother.

Abstract

Computer graphics have become commonplace in our daily lives. Nearly every modern high budget movie has some sort of digital special effects shot. Even small budget movies and local television advertisements tend to utilize some graphics technology. There are even several cartoon shows that are completely animated hand rendered using a computer. Yet aside from a small handful of high production cost movies, even non-experts can tell when computer rendered special effects are used.

Several factors contribute to effects shot, including modeling, animation, and rendering. If we focus just on rendering the answer is simple: time. When realistic imagery is required for example when adding computer graphics effects into a live action film the computational cost of generating the image increases. This cost increase prevents many films from achieving the visual fidelity necessary for seamless effects integration. It also prevents applications like games, which require fast rendering times, from having realistic imagery.

Ray tracing is an image synthesis technique which simulates the interaction of light with surfaces. Most high-quality, photorealistic renderings are generated by global illumination techniques built on top of ray tracing. Real-time ray tracing has been a goal of the graphics community for many years. Unfortunately, ray tracing is a very expensive operation. VLSI technology has just reached the point where the computational capability of a single chip is sufficient for real-time ray tracing. Supercomputers and clusters of PCs have only recently been able to demonstrate interactive ray tracing and global illumination applications. In this dissertation we show how a ray tracer can be written as a stream program. We present a stream processor abstraction for the modern programmable graphics processor (GPU) | allowing the GPU to execute stream programs. We describe an implementation of our streaming ray tracer on the GPU and provide an analysis of the bandwidth and computational requirements of our implementation. In addition, we use our ray tracer to evaluate simulated GPUs with enhanced program execution models. We also present an implementation and evaluation of global illumination with photon mapping on the GPU as an extension of our ray tracing system. Finally, we examine hardware trends that favor the streaming model of computation. Our results show that a GPU-based streaming ray tracer has the potential to outperform CPU-based algorithms without requiring fundamentally new hardware, helping to bridge the current gap between realistic and interactive rendering.

Table of Contents

List of Figures

List of Tables

1

Introduction

CHAPTER 1
Introduction

1.1 Motivation and problem Statement

Computer graphics have become commonplace in our daily lives. Nearly every modern high budget movie has some sort of digital special effects shot. Even small budget movies and local television advertisements tend to utilize some graphics technology. There are even several cartoon shows that are completely animated, and rendered using a computer. Yet aside from a small handful of high production cost movies, even non-experts can tell when computer rendered special effects are used. Why are computer rendered images easy to pick out? Several factors contribute to an effects shot, including modelling, animation, and rendering. If we focus just on rendering the answer is simple: time. When realistic imagery is required for example when adding computer graphics effects into a live action film the computational cost of generating the image are increased. This cost increase prevents many films from achieving the visual fidelity necessary for seamless effects integration. It also prevents applications like games, that require fast rendering times, from having realistic imagery. Most of the high-quality, photorealistic renderings made today rely on a rendering technique called ray tracing. Ray tracing simulates the interaction of light with surfaces, a process which is very computationally expensive. When the rendering does not have to happen interactively, such as a special effects shot for a movie, it is simply a matter of compute time and resources to perform the simulations necessary to generate realistic images. When rendering speed matters, image quality is often traded off for increased speed. This trade off point varies from production to production and plays a large role in determining its achievable level of realism. We are interested in bridging the gap between realistic and interactive graphics. State of the art ray tracers today can render scenes at several frames per second on a supercomputer or on a cluster of high-end PCs. Unfortunately games, CAD, and other single-user applications, cannot rely on every person having a supercomputer or cluster to run them on. Instead, they exploit commodity graphics processors (GPUs) and render images with as high of quality as possible. Graphics processors have improved significantly in the past several years both in terms of speed and in terms of the type of shading they support. Most recently, they have exposed a fairly general programming environment which allows developers the freedom to write very realistic looking shader. In this thesis we explore the reformulation of high quality rendering algorithms for interactive rendering. We show that ray tracing can be expressed as a streaming computation, and that high performance processors like GPUs are well suited for supporting streaming computations. We can abstract the GPU as a general stream processor. The combination of this reformulation and abstraction allows us to implement high quality rendering algorithms at interactive rates on commodity programmable graphics hardware. Furthermore, we will argue that ray tracing is most naturally and efficiently expressed in the stream programming model.

1.2 Research Objectives

This thesis makes several contributions to the areas of computer graphics and graphics hardware design:_ We present a streaming formulation for ray tracing. Streaming is a natural way to express ray tracing. Modern high performance computing hardware is well suited for the stream programming model. The stream programming model helps to organize the ray tracing computation optimally to take advantage of modern hardware trends. We have developed a stream processor abstraction for the programmable fragment engine found in modern graphics processors. This abstraction helps us to focus on our algorithm rather than on the details of the underlying graphics hardware. More broadly, this abstraction can support a wide variety of computations not previously thought of as being mappable to a graphics processor. We have also developed an abstraction for the GPU memory system. We show that dependent texturing allows texture memory to be used as a read only general purpose memory. We can navigate complex data structures stored in texture memory via dependent texture lookups. As with our stream processor abstraction, our memory abstraction has proved useful to a wide variety of algorithms. We have implemented and evaluated a prototype of our streaming ray tracer on current programmable graphics hardware. Our implementation runs at comparable speeds to a fast CPU-based implementation even though current GPUs lack a few key stream processor capabilities. The GPU-based implementation validates our ray tracing decomposition and GPU abstractions. We have extended our ray tracer to support photon mapping. Our implementation solves the sorting and searching problems common to many global illumination algorithms in the streaming framework. This implementation is an initial step in demonstrating that the streaming model is valid for advanced global illumination computations as well as simple ray tracing.

We analyze how other ray tracing systems are optimized to take advantage of fundamental hardware trends. We show that these optimizations take advantage of the same trends that the streaming approach does, only in a much more cumbersome way than is exposed by the stream programming model. We conclude that high performance ray tracing is most naturally expressed in the stream programming model. In this thesis, we are primarily concerned with studying static scenes { scenes in which the light sources or eye position can move but scene geometry is fixed. Our analysis does not account for any scene pre-processing times. Efficient ray tracing of dynamic scenes is an active research area in the ray tracing community, and is beyond the scope of this thesis.

1.3 Thesis Overview

This thesis is organized in seven chapters including this one. Their contents are described briefly in the following lines:

- ***Chapter*** ***2*** *Background Overview*: in this chapter we give a brief review of some of the subjects that are related to the field of our research.
- ***Chapter 3***- *Literature Survey*: this chapter provides an overview on rendering techniques; both real time and non real time techniques, while focusing on the ray tracer considered in this research.
- ***Chapter 4-*** *Programmable Graphics Processor Abstractions:*: this chapter provides an overview on abstract the programmable fragment engine in current GPUs as a stream processor

- **Chapter 5**- *Stream Ray Tracer Design And Acceleration:* this chapter presents algorithms for decompose ray tracer on graphics hardware and how to accelerate this algorithms
- **Chapter 6**- *Photon mapping on GPU:* this chapter presents algorithms for decompose photon mapping on graphics hardware and how to map this algorithms
- **Chapter 7-** *Performance Analysis & Evaluation*: in this chapter, the performance of the proposed algorithms was tested and compared over a number of case studies representing a combination of implement ray tracer and photon mapping on CPU and GPU .