EFFICIENT RAY-TRACING TECHNIQUES USING GPU

Guangfu Shi

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

June 2011

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Guangfu Shi**

Entitled: **Efficient Ray-Tracing Techniques Using GPU**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

______________________________ Chair

______________________________ Examiner

______________________________ Examiner

______________________________ Examiner

______________________________ Supervisor

______________________________ Co-supervisor

Approved ______________________________

Chair of Department or Graduate Program Director

__________ 20 ______ ______________________________

Rama Bhat, Ph.D.,ing., FEIC, FCSME, FASME, Interim Dean

Faculty of Engineering and Computer Science

Abstract

Efficient Ray-Tracing Techniques Using GPU

Guangfu Shi

Text of abstract.

Acknowledgments

Text of acknowledgments.

Contents

List of Figures vi

List of Tables vii

1 Introduction 1

List of Figures

List of Tables

Chapter 1

Introduction

Recently, with the dramatically boost of computational power of the current generati on CPU hardware, there has been a surge of new interetest in ray tracing as a core rendering algorithm for real-time applications such as video games since it is applicable to push the rendering task to interactive level on powerful computing devices which are easily accessible for a mainstream PC platform. Recently emerged powerful graphics processing units (GPUs) and flexible programming model provides researchers and developers with the ability to have the critical part of the algorithm parallelized and capture many secondary rendering effects such as shadows, refelections and refraction.

In the context of computer graphics, the term *rendering* refers to the process of generating a two-dimension image from a three-dimensional virtual scene. Various rendering techniques have been widely used in many fields where computers are needed to generate images. Based on the algorithm used for the rendering process, rendering approach can be classified to two major categories: rasterization-based rendering and ray tracing-based rendering.

At high level, rasterazation algorithm iterats projects the polygons, which forms the geometry objects in the scene, into screen space for rendering. With the application of Z-buffer, we could have a correct occlusion result, however, it may cause redundant raster and shading calculations which is also known *overdraw.* One of the biggest advantages to a rasterization renderer is that a simple, immediate-mode graphics pipeline can be interfaced and only a set of geometry is required to be submitted to the renderer per frame without any need to introduce any acceleration structure

(AS) of the scene. This pipeline interface could lead to a hardware implementation to this pipeline interface can be fast and able to handle the dynamic scene interactively. However, the weakness of rasterization rendering, which is its inablity to properly render secondary effects such as reflection, shadows and refraction, is vital especially in the fields where high quality rendering is required. It is required to adopt some multiple pass techniques to achieve such effects which are inefficient and produce visible artifacts.

The idea behind ray tracing is straight-forward, instead of projecting geomery into screen, ray tracing shoots rays from the virtual camera into scene space and finds intersection points with the geometry object, and then secondary rays can be generated from the intersection point and cast into the scene for secondary effects. Once the intersection points are found, the final colours are going to be calculated and produce the final image. The major advantage of ray tracing over rasterization is its inherent correctness in presenting the optical phenomenon in the final image, no other special techniques need to be adopted to achieve shadows, reflection and refraction such secondary effects. Due to the nature of ray tracing, massive demand of computation power is the shortcoming and becomes the major barrier in the way to bring ray tracing to real-time applications.

The essential ray tracing task is to find the intersection points between rays and geometry, it is safe to classify ray tracing as a spatial searching problem, therefore, it will be helpful to incorporate a kind of AS which subdivides the scene to achieve efficient searching of intersection points. The AS, however, leads to a more complex hardware implementation since the rendering is no longer immediate and how to effiently build and maintain the AS bring more challenges on boosting the performance ray tracer.

In addition to accelerate the ray tracing algorithmically, the revolutionary development of computing hardware driven by *Moore's law* opens up another opportunity that cannot be ignored to make the ray tracing faster. Current generation x86 Central Processing Unit (CPU) is designed to maximize the parallelism including *Instruction Level Parallelism* and *Thread Level Parallelism*, these new features interest researchers to present new ideas and algorithms to take advantage of, such as rays packet traversal algorithm. Another common computing device in a system, which is the Graphics Processin Unit (GPU), has become much more powerful especially in float-point computing, furthermore, more flexible programming model and massive parralelism architecture make it as an alternetive device to use for ray tracing. However, none of these optimization approaches is automatically ann cheap, data structures and algorithms have to be carefully re-designed to be friendly to a specific device, and some constrains also have to be considered or they will significantly hit the overall performance.

äÿŋæŰĞæţŃèŕŢ latex

Bibliography