



**Paulo Ivson Netto Santos**

## **Ray Tracing Dynamic Scenes on the GPU**

### **DISSERTAÇÃO DE MESTRADO**

Dissertation presented to the Postgraduate Program in Informatics of the Departamento de Informática PUC–Rio as partial fulfillment of the requirements for the degree of Mestre em Informática

Advisor: Prof. Waldemar Celes Filho

Rio de Janeiro  
March 2009



**Paulo Ivson Netto Santos**

## **Ray Tracing Dynamic Scenes on the GPU**

Thesis presented to the Post-graduate Program in Computer Science of Departamento de Informática PUC-Rio as partial fulfillment of the requirements for the degree of Master in Computer Science. Approved by the following commission:

**Prof. Waldemar Celes Filho**

Adviser

Departamento de Informática — PUC-Rio

**Prof. Marcelo Gattass**

Departamento de Informática — PUC-Rio

**Prof. João Comba**

Instituto de Informática — UFRGS

**Prof. Luiz Henrique de Figueiredo**

Visgraf — IMPA

**Prof. José Eugenio Leal**

Head of Centro Técnico Científico — PUC-Rio

Rio de Janeiro — March 23, 2009

All rights reserved.

### Paulo Ivson Netto Santos

Paulo Ivson graduated in Computer Engineering from PUC-Rio, in 2006. Since 2005 he works at the university's Computer Graphics research laboratory (Tecgraf), specializing in realtime 3D rendering and virtual reality applications.

#### Bibliographic data

Santos, Paulo Ivson Netto

Ray Tracing Dynamic Scenes on the GPU / Paulo Ivson Netto Santos; advisor: Waldemar Celes Filho. – 2009.

61 f. : il. ; 29,7 cm

1. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009. Inclui bibliografia

1. Informática – Teses. 2. Traçado de raios interativo. 3. Grades uniformes. 4. Cenas dinâmicas. 5. Reconstrução de grades na GPU. 6. Programação em GPU. I. Celes Filho, Waldemar. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my father Adilson, my mother Ana Maria and my brother Marcelo Ivson.  
Without your love I would never have gotten this far.

## Acknowledgments

To my love Natália, for always being there by my side with a beautiful smile. You make everything worthwhile.

To my adviser Waldemar Celes for the unending dedication, incentive and patience even in the most dire situations. Thank you for all the great ideas.

To professor Marcelo Gattass, whose invitation I took without knowing it would change my life for the better.

To professors Alberto Raposo and Thadeu Leite, for all the support having me in the Virtual Reality group.

To Tecgraf and PUC-Rio, for the financial support, without which this work would not have been realized.

To Pedro Luchini for helping with model conversion and debugging, while also enduring my daily outcries of chaos.

To Rafael Delerue for helping with testing configuration, while actually agreeing with my daily outcries of chaos.

To Walter, for helping me face the rage of all our printers when trying to make the physical copies of this work.

To Aurélio for giving numerous rides from work whenever I needed.

To Leonardo Duarte, for all the help during our initial research.

To these people for teaching me the “art” of C++ programming: Gustavo, Thiago, Google.

To Sandra, Claudinei, Herivelto, Lucindo, the support team and many others without whom the lab would come to a halt.

To all my friends and colleagues in Tecgraf, which include but are not limited to: Bruno, Cadu, César, Negão, Fabíola, Fera, Filip, Flávia, Lucas, Luciana, Luciano, Manuel, Marcela, Márcio, Malf, Pablo, Peter, Renato, Vinícius and others.

To my true friend Alessandra, for putting up with me after 14 long years. Bee happy!

To Baloo for 14 years of unquestioning love and friendship.

To every person who has given me moments of joy throughout my life.

To Domino’s Pizza, for the Tuesday 2 for 1 promotion. And for arriving after 30 mins a number of times.

## Abstract

Ivson, Paulo; Celes Filho, Waldemar. **Ray Tracing Dynamic Scenes on the GPU**. Rio de Janeiro, 2009. 61p. MSc Thesis — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

We present a technique for ray tracing dynamic scenes using the GPU. In order to achieve interactive rendering rates, it is necessary to use a spatial structure to reduce the number of ray-triangle intersections performed. Whenever there is movement in the scene, this structure is entirely rebuilt. This way, it is possible to handle general unstructured motion. For this purpose, we have developed an algorithm for reconstructing Uniform Grids entirely inside the graphics hardware. In addition, we present ray-traversal and shading algorithms fully implemented on the GPU, including textures, shadows and reflections. Combining these techniques, we demonstrate interactive ray tracing performance for dynamic scenes, even with unstructured motion and advanced shading effects.

## Keywords

Interactive ray tracing. Uniform Grids. Dynamic scenes. Grid rebuild on the GPU. GPU programming.

## Resumo

Ivson, Paulo; Celes Filho, Waldemar. **Traçado de Raios de Cenas Dinâmicas na GPU**. Rio de Janeiro, 2009. 61p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O objetivo deste trabalho é desenvolver uma solução completa para o traçado de raios de cenas dinâmicas utilizando a GPU. Para que este algoritmo atinja desempenho interativo, é necessário utilizar uma estrutura espacial para reduzir os testes de interseção entre raios e triângulos da cena. Observa-se que, quando há movimento na cena, é necessário atualizar esta estrutura de aceleração, seja alterando-a parcialmente ou reconstruindo-a inteiramente. Adotamos a segunda estratégia por ser capaz de tratar o caso geral de movimento não-estruturado. Como a construção da estrutura deve ser feita da forma mais eficiente possível, escolhemos utilizar uma Grade Uniforme como foco de nossa pesquisa. Suas vantagens incluem um algoritmo de construção simples e um percurso de raios eficiente.

Para explorar o poder de processamento em paralelo de uma GPU, é necessário manter os dados da cena e da estrutura de aceleração dentro da placa gráfica, evitando transferências custosas de memória entre a GPU e a CPU. Propomos neste trabalho uma técnica para construir uma grade uniforme inteiramente na GPU. Usando nosso método, é possível reconstruir toda a estrutura em poucos milissegundos, enquanto mantém-se a alta qualidade da grade obtida.

Além disso, propomos uma implementação do algoritmo de traçado de raios de forma a aproveitar o processamento em paralelo da GPU. Nosso procedimento é implementado inteiramente dentro da placa gráfica, onde há acesso direto para os dados dos triângulos da cena, bem como as informações da grade uniforme construída. Utilizando a solução proposta, somos capazes de obter taxas de visualização interativas mesmo para cenas com movimentos não-estruturados, incluindo texturas, sombras e até mesmo reflexões.

## Palavras-chave

Traçado de raios interativo. Grades uniformes. Cenas dinâmicas. Reconstrução de grades na GPU. Programação em GPU.

# Contents

1	Introduction	<b>13</b>
2	Evolution of Ray Tracing	<b>20</b>
2.1	Dynamic Scenes on the CPU	20
2.2	Research on the GPU	21
3	Solution Overview	<b>24</b>
3.1	Uniform Grid Structure	24
3.2	Accelerating Ray Traversal	26
3.3	Programming the Graphics Hardware	28
3.4	Ray-Tracing Architecture	31
4	Parallel Grid Construction	<b>33</b>
4.1	Conceptual Algorithm	34
4.2	Proposed Implementation	36
4.3	Summary	40
5	Ray Tracing on the GPU	<b>42</b>
5.1	Data Layout	42
5.2	Common Routines	43
5.3	Optimized Implementation	45
5.4	Enabling Reflections	47
6	Performance Results	<b>48</b>
6.1	Grid Construction	48
6.2	Static Scenes	49
6.3	Dynamic Scenes	53
6.4	Comparison with Related Work	54
7	Conclusion	<b>57</b>
7.1	Future Work	57
	Bibliography	<b>59</b>



## List of Figures

1.1	A triangle projected on the image plane (left) and its corresponding pixels after rasterization (right).	13
1.2	The environment map is obtained by rendering the scene from the object's point of view an additional six times [Nvidia 2004].	14
1.3	The ray tracing algorithm. In the image, shadow rays determine incoming radiance from the light source.	15
1.4	The rendering equation describes the total amount of light emitted from a point $x$ along a particular viewing direction, given a function for incoming light and a BRDF.	16
1.5	Real-time global illumination effects obtained with ray tracing [Wald et al. 2003].	17
3.1	Examples of Uniform Grids.	24
3.2	In blue, cells overlapped only by the triangle's AABB. In green, cells overlapped by the actual primitive.	25
3.3	The ray traversal procedure uses the current cell ID to obtain its corresponding list of primitives, to be tested for intersection next. Different colors identify separate lists.	26
3.4	Examples of ray traversal using a Uniform Grid.	27
3.5	The distance between consecutive cell boundaries on each axis is the same.	27
3.6	Main traversal loop for the 2D example. Notice that $t_{\Delta X}$ and $t_{\Delta Y}$ are constants.	28
3.7	This Nvidia GPU is made of 128 general-purpose streaming processors (green boxes) arranged in clusters to share memory and texture functions. Vertex, geometry and pixel processing are load-balanced dynamically during rendering [Nvidia 2006].	29
3.8	In CUDA, a kernel computation is distributed amongst several threads organized in blocks. There are different kinds of memory spaces available, which differ in access latency, available bandwidth, total size and read/write access [Nvidia 2008].	30
3.9	General view of the proposed ray-tracing architecture. Highlighted in grey are the two main steps: grid construction and ray tracing.	31
4.1	Cell-primitive pairs on the left are sorted by primitive ID. The desired result, on the right, with pairs sorted by cell ID. Colors highlight elements which belong to the same list.	35
4.2	Data flow of the proposed grid rebuild algorithm. Red and green arrows indicate write and read operations, respectively. Grey boxes highlight the main output of the procedure.	36
4.3	Example of our grid construction algorithm using 4 primitives and 9 cells. Colors indicate elements that belong to the same list.	40
5.1	Conceptual ray-tracing algorithm. Texture read and write operations are highlighted in green and red, respectively.	44

6.1	Ray tracing CAD models at interactive rates, including shadows and reflections.	50
6.2	Deformable meshes used in static and dynamic scene tests.	51
6.3	Scenes with unstructured movement used for static and dynamic test cases.	53

## List of Tables

6.1	Time in milliseconds to rebuild the entire grid structure.	48
6.2	Performance in frames per second (fps) for different static scenes.	49
6.3	Performance in frames per second (fps) for a single key-frame of the benchmark scenes.	52
6.4	Performance in frames per second (fps) for the entire animation of each benchmark scene.	54
6.5	Frames-per-second comparison between our method and two state of the art research, using complex shading and shadow rays.	55
6.6	Analysis of times in milliseconds from our proposed implementation, compared to related work.	55

*Though this be madness, yet there is method in't.*

**William Shakespeare**, *Hamlet Act 2, scene 2, 193-206.*