

**Eduardo Telles Carlos**

## **Frustum Culling Híbrido Utilizando CPU e GPU**

### **Dissertação de Mestrado**

Dissertação apresentada ao Programa de Pós-graduação em Informática da PUC-Rio como requisito parcial para obtenção do título de Mestre em Informática.

Orientador : Prof. Alberto Barbosa Raposo  
Co-Orientador: Prof. Marcelo Gattass

Rio de Janeiro  
Abril de 2009



**Eduardo Telles Carlos**

## **Frustum Culling Híbrido Utilizando CPU e GPU**

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

**Prof. Alberto Barbosa Raposo**

Orientador

Departamento de Informática — PUC-Rio

**Prof. Marcelo Gattass**

Co-Orientador

Departamento de Informática — PUC-Rio

**Prof. Waldemar Celes Filho**

Departamento de Informática — PUC-Rio

**D.Sc. Luciano Soares**

Departamento de Informática — PUC-Rio

**Ph.D. Rodrigo de Toledo**

Cenpes— Petrobras

**Prof. José Eugênio Leal**

Coordenador Setorial do Centro Técnico Científico — PUC-Rio

Rio de Janeiro, 02 de Abril de 2009

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

### **Eduardo Telles Carlos**

Graduou-se em Engenharia da computação na PUC-Rio em 2006. Desde 2002 trabalha no Grupo de Tecnologia em Computação Gráfica da universidade (TecGraf) desenvolvendo sistemas de realidade virtual e visualização científica.

#### Ficha Catalográfica

Carlos, Eduardo Telles

Frustum Culling Híbrido Utilizando CPU e GPU / Eduardo Telles Carlos; orientador: Alberto Barbosa Raposo; co-orientador: Marcelo Gattass. — 2009.

99 f: il. (color.); 30 cm

Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.

Inclui bibliografia

1. Informática – Teses. 2. Algoritmos de visualização. 3. Descarte de volumes envolventes. 4. GPGPU. 5. Primitivas GPU. I. Raposo, Alberto Barbosa. II. Gattass, Marcelo. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

À minha mãe Nilce Telles Carlos, ao meu pai William Braga Carlos e à  
minha irmã Érica Telles Carlos. Sem o carinho, amor e dedicação deles, este  
trabalho não seria possível.

## Agradecimentos

Ao meu orientador Alberto Raposo, pela dedicação, confiança e conselhos dados ao longo de toda a execução deste trabalho.

Ao meu co-orientador Marcelo Gattass pelas sugestões e confiança no meu trabalho.

Aos professores Marcelo Gattass e Waldemar Celes Filho pelas excelentes aulas na graduação e no mestrado.

A meus pais e a minha irmã por tudo que sou hoje.

A minha avó, tios, tias, primos, primas e a minha madrinha pela ajuda, conselhos e amor.

A Pontifícia Universidade Católica do Rio de Janeiro pela excelente formação a mim dada.

Ao Tecgraf pela oportunidade de aprendizado e auxílios concedidos.

A todos os integrantes das equipes do SiVIEP, Environ e demais dentro do Tecgraf pela compreensão, aprendizado e momentos de descontração.

Aos grandes amigos André Luis Pinto Ferreira e Rafael Brito de Farias pelas amizades verdadeiras e apoio nos momentos difíceis.

A Rodrigo Toledo por toda a ajuda dada antes mesmo da definição da área de pesquisa a ser seguida e ajuda no framework das GPU primitives.

A Paulo Ivson Netto pela ajuda nos trabalhos ao longo da graduação, do mestrado e no desenvolvimento deste trabalho.

A equipe de suporte do Tecgraf por terem me ajudado com os problemas de hardware.

A paciência dos integrantes do futebol de domingo pela queda no meu rendimento.

## Resumo

Carlos, Eduardo Telles; Raposo, Alberto Barbosa; Gattass, Marcelo. **Frustum Culling Híbrido Utilizando CPU e GPU**. Rio de Janeiro, 2009. 99p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Um dos problemas mais antigos da computação gráfica tem sido a determinação de visibilidade. Vários algoritmos têm sido desenvolvidos para viabilizar modelos cada vez maiores e detalhados. Dentre estes algoritmos, destaca-se o *frustum culling*, cujo papel é remover objetos que não sejam visíveis ao observador. Esse algoritmo, muito comum em várias aplicações, vem sofrendo melhorias ao longo dos anos, a fim de acelerar ainda mais a sua execução. Apesar de ser tratado como um problema bem resolvido na computação gráfica, alguns pontos ainda podem ser aperfeiçoados, e novas formas de descarte desenvolvidas. No que se refere aos modelos massivos, necessita-se de algoritmos de alta performance, pois a quantidade de cálculos aumenta significativamente. Este trabalho objetiva avaliar o algoritmo de *frustum culling* e suas otimizações, com o propósito de obter o melhor algoritmo possível implementado em CPU, além de analisar a influência de cada uma de suas partes em modelos massivos. Com base nessa análise, novas técnicas de *frustum culling* serão desenvolvidas, utilizando o poder computacional da GPU (Graphics Processing Unit), e comparadas com o resultado obtido apenas pela CPU. Como resultado, será proposta uma forma de *frustum culling* híbrido, que tentará aproveitar o melhor da CPU e da GPU.

## Palavras-chave

Algoritmos de visualização. Descarte de volumes envolventes. GPGPU. Primitivas GPU.

## Abstract

Carlos, Eduardo Telles; Raposo, Alberto Barbosa; Gattass, Marcelo. **Hybrid Frustum Culling Using CPU and GPU**. Rio de Janeiro, 2009. 99p. MSc Dissertation — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The definition of visibility is a classical problem in Computer Graphics. Several algorithms have been developed to enable the visualization of huge and complex models. Among these algorithms, the frustum culling, which plays an important role in this area, is used to remove invisible objects by the observer. Besides being very usual in applications, this algorithm has been improved in order to accelerate its execution. Although being treated as a well-solved problem in Computer Graphics, some points can be enhanced yet, and new forms of culling may be disclosed as well. In massive models, for example, algorithms of high performance are required, since the calculus arises considerably. This work analyses the frustum culling algorithm and its optimizations, aiming to obtain the state-of-the-art algorithm implemented in CPU, as well as explains the influence of each of its steps in massive models. Based on this analysis, new GPU (Graphics Processing Unit) based frustum culling techniques will be developed and compared with the ones using only CPU. As a result, a hybrid frustum culling will be proposed, in order to achieve the best of CPU and GPU processing.

## Keywords

Visualization algorithms. Frustum culling. GPGPU. GPU Primitives.

## Sumário

1	Introdução	<b>13</b>
1.1	Motivação	13
1.2	Objetivo da dissertação	15
1.3	Trabalhos relacionados	15
1.4	Organização da dissertação	16
2	Algoritmos de Visibilidade	<b>17</b>
2.1	Determinação de visibilidade	17
2.2	Classificação dos algoritmos de visibilidade	20
2.3	Pipeline dos algoritmos de visualização	22
3	Frustum culling	<b>26</b>
3.1	Implementação clássica	27
3.2	Estado da arte	28
3.3	Memória utilizada em CPU	42
3.4	Ambiente de benchmark	43
4	Implementação em CPU	<b>48</b>
4.1	Renderização	48
4.2	Radar X Planos	48
4.3	Hierarquia	51
4.4	Percurso	53
4.5	Otimizações	56
4.6	SIMD	58
4.7	Multiprocessamento	60
4.8	Pipeline final em CPU	64
5	Frustum culling em GPU	<b>65</b>
5.1	Timeline gpu	65
5.2	Gpu primitives	67
5.3	Algoritmos de frustum culling em GPU	71
5.4	Memória utilizada em GPU	74
5.5	Percurso sem pilha em GPU	75
5.6	Implementação	76
6	Frustum culling híbrido	<b>79</b>
6.1	Heurísticas	79
6.2	Implementação	85
7	Conclusão e Trabalhos futuros	<b>89</b>
7.1	Conclusão	89
7.2	Trabalhos futuros	90
	Referências Bibliográficas	<b>92</b>



## Lista de figuras

1.1	Modelos Massivos.	13
2.1	Problema com o algoritmo do pintor.	18
2.2	Técnicas de culling.	19
2.3	Pipeline do OpenGL.	22
2.4	Frustum culling em CPU.	23
2.5	Frustum culling em GPU.	24
3.1	Tipos de Projeção.	26
3.2	Pseudo-código de descarte de pontos.	28
3.3	Volumes envolventes, imagem divulgada por Ericson <i>et al.</i> [23]	29
3.4	Teste de descarte de AABB.	30
3.5	Classificação de um ponto P contra o frustum.	31
3.6	Teste de pontos contra radar.	31
3.7	Teste de AABB contra radar.	32
3.8	Hierarquia de volumes envolventes.	33
3.9	Classificação dos vértices n e p.	35
3.10	Volume envolvente do frustum.	36
3.11	Teste de descarte com apenas quatro planos.	37
3.12	Escape index.	38
3.13	SISD X SIMD.	40
3.14	Dados utilizados no nó da hierarquia.	43
3.15	Aplicação desenvolvida para os testes.	44
3.16	Modelos com informações paramétricas.	45
3.17	Caminhos de câmera pelas plataformas.	46
3.18	Caminho de câmera pelo Boeing (73 segundos).	47
4.1	Caminhos de câmera sem hierarquia nas plataformas.	50
4.2	Caminhos de câmera sem hierarquia no Boeing.	51
4.3	Caminhos de câmera com hierarquias onde média foi melhor.	53
4.4	Caminho de câmera com hierarquias onde mediana foi melhor.	54
4.5	Caminho de câmera sem pilha nas plataformas.	55
4.6	Caminho de câmera sem pilha no Boeing.	55
4.7	Otimizações sugeridas por Assarsoon nas plataformas.	57
4.8	Otimizações sugeridas por Assarsoon no Boeing.	57
4.9	Caminhos de câmera pelas plataformas com otimizações.	58
4.10	Caminhos de câmera pelo Boeing com otimizações.	59
4.11	Resultados obtidos por Assarsson em duas cenas.	61
4.12	Percurso com <i>multithread</i> .	61
4.13	Multiprocessamento sem troca de tarefas entre threads.	62
4.14	Multiprocessamento sem troca de tarefas entre threads no Boeing.	62
4.15	Multiprocessamento com troca de tarefas entre threads nas plataformas.	63
4.16	Multiprocessamento com troca de tarefas entre threads no Boeing.	64
4.17	Pipeline final em CPU.	64

5.1	Evolução das placas gráficas até a quarta geração.	66
5.2	Evolução das placas gráficas a partir da quarta geração.	66
5.3	Pipeline das placas gráficas modernas.	67
5.4	Variáveis do vertex shader das gpu primitives (extraída de [70]).	68
5.5	Variáveis do pixel shader das gpu primitives (extraída de [70]).	68
5.6	Cilindro Gpu primitives.	69
5.7	Cone GPU primitives.	70
5.8	Torus slice GPU primitives.	70
5.9	Frustum culling junto com GPU primitives.	71
5.10	Frustum culling em shader separado.	72
5.11	Frustum culling em modelos genéricos.	74
5.12	Frustum culling em modelos genéricos.	74
5.13	Memória utilizada no percurso em GPU.	76
5.14	Frustum culling nas GPU primitives.	77
5.15	Frustum culling em GPU para modelos genéricos.	77
6.1	Melhores caminhos de câmera em CPU e GPU nas plataformas.	80
6.2	Melhores caminhos de câmera em CPU e GPU no Boeing.	81
6.3	Esquema do algoritmo de frustum culling híbrido.	83
6.4	Possíveis estados do frustum culling híbrido.	84
6.5	Resultados do frustum culling híbrido nas plataformas.	86
6.6	Resultados do frustum culling híbrido no Boeing.	87
6.7	Resultados do frustum culling híbrido com render ligado.	88

## Lista de tabelas

2.1	Tabela comparativa de alguns algoritmos de visibilidade.	25
3.1	Modelos paramétricos.	45
3.2	Modelos com malhas triangulares.	46
4.1	Radar X Planos.	49
4.2	Radar sem interseção X Radar com interseção para o Boeing.	50
4.3	Hierarquias utilizando mediana.	52
4.4	Hierarquias utilizando média.	52
4.5	Detalhes dos caminhos com hierarquias de média e mediana.	54
4.6	Detalhes dos caminhos com hierarquias de média e mediana.	54
4.7	Resultados obtidos no trabalho de Assarsson.	56
4.8	Melhor combinação de otimizações utilizadas em cada modelo.	59
4.9	SIMD nos cálculos de descarte da P-43.	60
5.1	Memória necessária para os volumes envolventes.	75
6.1	Comparação dos resultados entre melhor algoritmo em CPU e Híbrido.	87

*"A verdadeira felicidade está na própria casa,  
entre as alegrias da família."*

**Leon Tolstoi,** *Leon Tolstoi.*