UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

IMPLEMENTAÇÃO E OTIMIZAÇÃO DE UMA BOUNDING INTERVAL HIERARCHY PARA UM RAYTRACER DE TEMPO REAL USANDO CUDA

Proposta de Trabalho de Graduação

Aluno(a): Rafael de Melo Aroxa (rma5@cin.ufpe.br)
Orientador(a): Veronica Teichrieb (vt@cin.ufpe.br)

31 de agosto de 2010

1. INTRODUÇÃO

Cada vez mais a busca por resultados mais realísticos quando da renderização de cenas virtuais tem aumentado consideravelmente. Técnicas como o *ray tracing* (traçado de raios) utilizam sistemas baseados em iluminação global para renderizar cenas com mais realismo. Para alcançar este resultado, nuances como reflexão e refração devem ser consideradas e com isso, o custo computacional também é aumentado consideravelmente.

Motivando esta demanda por renderizações mais realistas de cenas 3D virtuais, tem-se a evolução da capacidade de processamento das GPUs (*Graphics Processing Units*). Com processadores gráficos que possuem até 480 núcleos [7], a utilização de técnicas de renderização mais realísticas se torna muito mais viável. Este crescimento dos dispositivos gráficos pode levar a uma mudança radical na forma como as cenas virtuais são renderizadas atualmente. Gigantes da indústria, como a NVIDIA e a Intel, vislumbram em um futuro próximo a possibilidade da utilização de *ray tracing* em aplicações interativas como jogos, em face à utilização das mais evoluídas técnicas de rasterização de hoje.

Os algoritmos de rasterização se baseiam principalmente na idéia de planificar um mundo virtual 3D em um plano projetivo a ser exibido pela câmera virtual. Os objetos da cena são ordenados pela distância à câmera e processados um a um, do mais distante, para o mais próximo, sendo subdivididos em triângulos e "jogados" sobre o plano projetivo para a obtenção da imagem final. Esta técnica traz resultados aceitáveis a um custo computacional relativamente baixo, diferentemente das técnicas que utilizam modelos de iluminação global, sendo capazes de entregar resultados com uma fidelidade bem maior.

Figura 1. Comparativo entre rasterização e *ray tracing*.

As técnicas de iluminação global se baseiam na idéia de raios luminosos que são emitidos pelas fontes de luz e que se propagam pelo ambiente, sendo refletidos e refratados conforme o material dos objetos em que os raios colidem. O cálculo da travessia do raio no ambiente é regido pelos princípios

básicos da física ótica clássica [2] e mesmo utilizando os conceitos mais básicos, a técnica de *ray tracing* consegue entregar imagens com uma qualidade bem superior às das cenas rasterizadas.

O alto custo computacional que envolve o algoritmo de *ray tracing* reside na necessidade do tratamento de colisão do raio traçado com os muitos objetos da cena. Por exemplo, uma cena com 100 objetos renderizada a uma resolução de 800x600 *pixels* realizará 48.000.000 (100 testes de colisão para cada *pixel*) testes de colisões na renderização de um único *frame*. Os testes de colisão são totalmente independentes para cada *pixel*, o que torna o algoritmo facilmente paralelizável. Mesmo com placas gráficas mais antigas e menos potentes, a utilização de CUDA para implementação de *ray tracing* leva a taxas de fps (*frames* por segundo) bastante interativas, devido a esta natureza altamente paralelizável deste algoritmo.

Em conjunto com as técnicas padrões do *ray tracing*, está a utilização de estruturas de dados que aceleram o processo de travessia dos raios pelo ambiente. Estas estruturas particionam a cena em sub-partes e fazem com que os raios traçados só realizem teste de colisão para os objetos contidos na(s) sub-área(s) em que colidir o raio.

Existem diversas estruturas que se propõem a acelerar este processo de travessia [5]. Entre as mais utilizadas e estabelecidas se encontram a KD-Tree [3][4] e a BVH [6]. Ambas se configuram como arvores binárias que conseguem realizar a travessia do raio de modo direcionado, evitando testes de colisão desnecessários, porém com algoritmos de construção complexos. A estrutura proposta por [1] possui o mesmo princípio de divisão binária do espaço, porém com o algoritmo de construção relativamente mais simplificado. Esta abordagem traz vantagens quando da utilização da BIH (*Bounding Interval Hierarchy*) para cenas dinâmicas, já que se faz necessária a reconstrução de toda a estrutura da BIH para cada *frame* a taxas interativas de fps.

2. OBJETIVOS

2.1. OBJETIVO GERAL

Este trabalho de graduação tem por objetivo implementar uma *Bounding Interval Hierarchy* para acelerar a travessia de raios em um *Ray Tracer* de tempo real. O escopo deste trabalho também contempla diversas otimizações que serão realizadas na implementação proposta por [1]. Como parte da pesquisa proposta por este trabalho de graduação, está a análise detalhada do algoritmo de construção e travessia da *Bounding Interval Hierarchy*. Com base nesta análise, diversas propostas de otimização surgirão e, para cada uma delas, será feito um estudo de viabilidade de implementação e de retorno em termos de performance, utilização de memória, entre outros.

Pretende-se analisar e estudar os recursos das mais novas placas gráficas para abrir ainda mais possibilidades de otimizações e economia de recursos na implementação da estrutura investigada por este trabalho. Abordagens técnicas como o uso de texturas, memória compartilhada, memória global e operações especiais propostas por CUDA devem ser ponderadas de modo a maximizar os ganhos. Se faz necessário também um estudo aprofundado que contemple cada uma destas técnicas e como elas poderiam ajudar no desempenho final da aplicação caso fossem utilizadas.

O algoritmo de construção da *Bounding Interval Hierarchy* proposto por [1] merece uma atenção especial devido à sua relativa simplicidade, pois quanto mais simples o fluxo e controle de um algoritmo, mais complexo é torná-lo mais veloz e fazer com que a quantidade de recursos consumida seja reduzida. A arquitetura dos dispositivos que dão suporte a CUDA não suporta recursão, o que torna impossível a utilização da *Bounding Interval Hierarchy* para cenas dinâmicas. Isto se dá pelo fato do algoritmo de construção ser de natureza recursiva e requerer um cuidado maior caso se queira transformá-lo em um algoritmo iterativo. Este trabalho também contemplará este estudo de viabilidade da transformação do algoritmo de construção proposto por [1].

2.2. OBJETIVOS ESPECÍFICOS

1) Estudo aprofundado dos algoritmos que envolvem a implementação de um sistema de *ray tracing*.
2) Estudo dos conceitos e detalhes sobre a estrutura de dados a ser investigada e melhorada por este trabalho de graduação.
3) Pesquisa do estado-da-arte de outras estruturas de dados propostas para a utilização com o *Ray Tracer*, bem como suas otimizações.
4) Implementação e otimização da estrutura investigada por este trabalho de graduação.

5) Validação das vantagens na utilização desta estrutura de dados [1] para o *Ray Tracer*.
6) Escrita do relatório do Trabalho de Graduação.

3. CRONOGRAMA

Etapa/Fase	Mês				
	Agosto	**Setembro**	**Outubro**	**Novembro**	**Dezembro**
Pesquisa Bibliográfica					
Leitura e seleção do material					
Estudo dos algoritmos envolvidos no *Ray Tracer*					
Implementação e Validação					
Estudo e implementação da estrutura de dados proposta					
Otimizações					
Validação das vantagens de utilização da estrutura de dados proposta					
Escrita do Trabalho					

REFERÊNCIAS BIBLIOGRÁFICAS

[1] WÄCHTER C., KELLER A.: Instant ray tracing: The bounding interval hierarchy. In Rendering Techniques 2006 (Proc. 17th Eurographics Symposium on Rendering), pp. 139–149. 2006.

[2] GLASSNER A.S. An Introduction to Ray Tracing. 1989. Academic Press. San Diego, Estados Unidos.

[3] SHEVTSOV M., SOUPIKOV A., KAPUSTIN A.: Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. In Computer Graphics Forum (Proc. Eurographics 2007), pp. 395–404. 2007.

[4] POPOV S., GUNTHER J., SEIDEL H.P. AND SLUSALLEK P. 2007. Stackless kd-tree traversal for high performance GPU ray tracing. Em Eurographics'07, 415–424

[5] GOLDSMITH J., SALMON J.: Automatic Creation of Object Hierarchies for Ray Tracing. IEEE Computer Graphics and Applications 7, 5 (May 1987), 14–20.

[6] WALD I., BOULOS S., SHIRLEY P.: Ray Tracing De- formable Scenes using Dynamic Bounding Volume Hierarchies. ACM Transactions on Graphics (2006). To appear.

[7] NVIDIA Home. http://www.nvidia.com. Acessado em 31 de Agosto de 2010.

DATAS E ASSINATURAS

Recife, 31 de agosto de 2010.

_____

Veronica Teichrieb (Orientadora)

_____

Rafael de Melo Arôxa (Aluno)