UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2009.1

UM SISTEMA DE RAY TRACING EM TEMPO REAL

PROPOSTA DE TRABALHO DE GRADUAÇÃO

Aluno	Artur Lira dos Santos	{als3@cin.ufpe.br}
Orientadora	Veronica Teichrieb	{vt@cin.ufpe.br}

1 de Março de 2009

Índice

1. Introdução

Com o crescente poder computacional dos microprocessadores gráficos (GPUs – *Graphics Processing Units*) e a constante busca de cenas de renderização 3D mais realísticas, técnicas de renderização por iluminação global e de alto custo computacional como *ray tracing* (traçado de raios) [1] passaram a atrair os grupos de pesquisa da área de Computação Gráfica em tempo real. Além de pesquisadores, gigantes da indústria de microprocessadores como a Intel [4] e a NVIDIA [6] visualizam num futuro próximo a possibilidade do uso de *ray tracing* em aplicativos interativos como jogos e programas CAD. Ambas afirmam que muito em breve grandes mudanças na forma de se renderizar jogos e aplicativos interativos 3D ocorrerão.

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

As técnicas de rasterização [7] se baseiam principalmente no conceito de transformar todos os objetos da cena em polígonos e posteriormente serem ordenados espacialmente e projetados no plano projetivo, definindo a imagem final. Para se renderizar uma esfera, por exemplo, a mesma será transformada em um conjunto de polígonos, que posteriormente torna-se um grupo de triângulos que dependendo de sua projeção no plano projetivo, tais triângulos serão exibidos na tela.

Técnicas de iluminação global usufruem dos conceitos básicos da Física ótica Clássica [1] para renderizar cenas com um aspecto mais real do que as rasterizadas. Ao invés de se "desenhar" triângulos na tela, a técnica de *ray tracing*, por exemplo, utiliza a ideia de raios luminosos emitidos por fontes de luz da cena. Tais raios se propagam no ambiente, sendo refletidos/refratados por objetos na cena. Mesmo utilizando os conceitos mais básicos de Física ótica, o *ray tracing* oferece uma imagem final com uma riqueza de detalhes bem maior se comparado com cenas rasterizadas, como é demonstrado na **Figura 1**. Objetos transparentes ou refletivos são melhor representados em *ray tracing* do que utilizando as mais modernas técnicas de rasterização.

Ao contrário de rasterização, *ray tracing* não depende da transformação dos objetos de toda a cena em polígonos. Tais objetos podem ser representados de forma mais precisa. As esferas, por exemplo, podem ser consideradas como entidades algébricas que a partir de suas equações espaciais é possível obter as informações suficientes para a renderização. Esta diferença de representar os objetos é uma das grandes vantagens do *ray tracing*.

Apesar de pesquisadores de empresas como NVIDIA e Intel concordarem que o modelo de rasterização atual pode muito em breve se tornar ultrapassado, cada um visualiza essas mudanças sob perspectivas diferentes. Membros da Intel [3] afirmam que toda a técnica atual de renderização 3D (generalizada como rasterização) muito provavelmente será completamente substituída por *ray tracing* ou alguma técnica que ofereça um maior grau de realismo. Além disso, afirmam que há grandes chances do sistema de renderização 3D voltar para *software* (processamento em CPU), ao invés de se ter uma implementação definida em *hardware* (GPU), trazendo vantagens como escalabilidade e portabilidade, já que o aplicativo não ficaria dependente de um equipamento específico. Já a NVIDIA [5] visualiza essas mudanças com maior cautela, acreditando que ocorrerá progressivamente uma mistura entre rasterização e *ray tracing* a partir de um *hardware* otimizado, assim como é atualmente feito com as placas gráficas. Independentemente das opiniões divergentes, há um consenso de que grandes mudanças em renderização 3D interativa ocorrerão e que muito provavelmente *ray tracing* será a técnica de vanguarda desta mudança.

2. Objetivos

Este trabalho de graduação tem como objetivo desenvolver um sistema interativo de ambientes virtuais utilizando *ray tracing* como principal técnica de renderização. Tal sistema tem a proposta direta de substituir a rasterização de uma cena de baixa e média complexidade por poderosas técnicas de iluminação global, além de buscar manter a simulação em taxas interativas (acima de 24 quadros por segundo).

Para alcançar tais objetivos, será realizado um estudo do estado da arte em *ray tracing* e algoritmos relacionados, como estruturas de dados de divisão espacial, em especial a KD-Tree [2], um tipo de árvore por partição binária de extrema importância para um eficiente sistema de *ray tracing*. Como tais algoritmos demandam de alto poder computacional, também será feito um estudo da arquitetura de CUDA [6] (*Compute Unified Device Architecture*), que possibilita a programação de propósito geral em placas gráficas (GPGPU) da NVIDIA.

Tais placas gráficas oferecem hoje um altíssimo poder computacional, com alguns dispositivos chegando a ultrapassar 1.5 Tflops (1.5 trilhões de soma-multiplicação em ponto flutuante / segundo), além de funcionarem bem com algoritmos altamente paralelizáveis, como é o caso do núcleo/kernel do *ray tracing*. Também será tirado proveito dos processadores *desktop* da Intel com as suas devidas extensões SSE2→SSE4, que apesar de terem um poder computacional relativamente baixo (+-50 Gflops) se comparado com as GPUs, têm como compensação um sofisticado sistema de *cache* e hierarquia de memória, sendo crucial para a implementação de algumas etapas não paralelizáveis do sistema.

Sendo assim, o sistema de *ray tracing* proposto será implementado de forma híbrida, tirando proveito dos melhores atributos que cada tipo de microprocessador disponível oferece. Como o *ray tracing* em si é altamente paralelizável, o seu núcleo será implementado em GPU, arquitetura mais adequada para algoritmos massivamente paralelos. Entretanto, algoritmos essenciais e seqüenciais como criação e manipulação de estrutura de dados necessários para o *ray tracing*, como uma KD-Tree e controle de câmera, serão implementados em CPU.

Este *Ray Tracer* propõe se estabelecer como uma biblioteca de referência em *ray tracing* em tempo real, com uma API bem planejada e documentada. Para isto, a arquitetura do sistema será estruturada como demonstra a **Figura 2.**

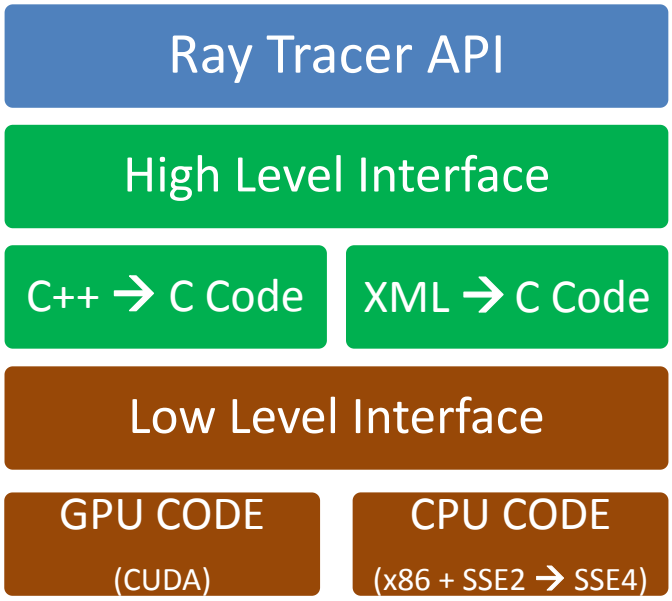

Figura 2. Arquitetura do *Ray Tracer*.

Apesar de depender diretamente de um nível de programação em baixo nível para obter um bom resultado, o sistema oferecerá uma camada de abstração em alto nível para renderização de cenas 3D, com suporte a descritores de cena em formato similar ao X3D/VRML [8], significando que o cliente ou usuário final não precisará ter conhecimentos de bibliotecas de baixo nível como CUDA para implementar seu aplicativo.

3. Cronograma

O cronograma abaixo demonstra algumas datas para as atividades principais do processo de desenvolvimento do trabalho de graduação.

Atividades/mês	**Março**	**Abril**	**Maio**	**Junho**
Levantamento do material bibliográfico				
Estudo aprofundado sobre *ray tracing*				
Estudo sobre CUDA e x86 SSE2→SSE4				
Estudo aprofundado sobre KD-Trees				
Modelagem e implementação do sistema				
Escrita do TG				
Finalização do trabalho de graduação				

4. Referências

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

[2] HAVRAN, V. **Heuristic Ray Shooting Algorithms.** 2000. Dissertação. Praga, República Tcheca.

[3] HOWARD, J. **Real Time Ray-Tracing: The End of Rasterization?** Disponível em: http://blogs.intel.com/research/2007/10/real_time_raytracing_the_end_o.php. Acessado em 1 de Março de 2009.

[4] Intel. **Intel Corporation homepage**. Disponível em: http://www.intel.com. Acessado em 1 de Março de 2009.

[5] LUEBKE, D. & PARKER, S. **Interactive Ray Tracing with CUDA.** Disponível em: http://developer.download.nvidia.com/presentations/2008/NVISION/NVISION08-IRT.pdf. Acessado em 1 de Março de 2009.

[6] NVIDIA. **CUDA homepage**. Disponível em: http://developer.nvidia.com/cuda. Acessado em 1 de Março de 2009.

[7] OpenGL. **OpenGL Architecture Review Board**. The OpenGL Programming Guide. Addison-Wesley Professional, ed. 5, 2005.

[8] VRML. **Web3D Consortium homepage**. Disponível em: http://www.web3d.org/. Acessado em 1 de Março de 2009.

5. Possíveis Avaliadores

Sílvio de Barros Melo.

6. Assinaturas

Artur Lira dos Santos
Aluno

Veronica Teichrieb
Orientadora