

Introdução ao Ray Tracing e análise do "Ray Tracing in One Weekend" de Peter Shirley

Lucas Araújo Pena

13/0056162

Resumo—*Resumão ipsum dolor sit amet, consectetur adipiscing elit. Donec id nisl eros. Phasellus eleifend nunc mi, nec rutrum purus lacinia sit amet. Vivamus eu tellus du. Quisque fringilla nibh nec leo pretium eleifend. Nullam et mi in tortor laoreet viverra sit amet id ligula. Etiam vel mi a quam fringilla aliquam. Nulla facilisis pellentesque odio, sit amet iaculis arcu porta eget. Nunc ut tempor enim. In sit amet rhoncus metus. Sed euismod vulputate nulla non rutrum. Proin nisl est, blandit accumsan malesuada vel, pharetra a quam. Nam mollis nunc nisl, in ornare mi hendrerit id. Nullam posuere est ipsum, ac commodo neque ultrices at.*

Index Terms—Computação Gráfica, Ray Tracing, Traçado de Raios, Path Tracing

I. INTRODUÇÃO

O Ray Tracing, uma técnica de renderização baseada na simulação do comportamento da luz, tem desempenhado um papel fundamental no avanço dos gráficos por computador. Ao permitir a criação de imagens fotorrealistas, o Ray Tracing tem sido aplicado em várias áreas, como filmes de animação, design de produtos e jogos digitais. Neste artigo, examinaremos a história do Ray Tracing, seus princípios teóricos e sua aplicação contemporânea, além de realizar um estudo de desempenho em um software de Ray Tracing.

O objetivo deste artigo é fornecer uma visão abrangente dessa técnica essencial no campo da computação gráfica, discutir seu uso na atualidade, aprender como funciona e como criar um Ray Tracer, e realizar uma análise de performance nesta implementação, para que se possa ter uma ideia das complexidades envolvidas em seu algoritmo.

<FALTA> Achados da pesquisa

II. BACKGROUND

Na computação gráfica, para que sejam gerados os mundos virtuais que todos conhecem, é necessário que haja primeiro sua sintetização. Os métodos mais utilizados na atualidade para renderizar uma cena virtual são a rasterização, o Ray Tracing e a mistura dos dois, gerando uma renderização híbrida.

A rasterização é o método mais utilizado em aplicações que necessitam ser em tempo real, como jogos eletrônicos e alguns tipos de simulações que precisam que haja uma interação em tempo real, como um simulador de vôo para pilotos. A rasterização é a conversão de um objeto em um ambiente virtual em pixéis para que seja mostrado na tela. Este método foi otimizado para que gere imagens a partir de triângulos na tela, por isso todo modelo 3D atualmente é uma malha de triângulos. [1]

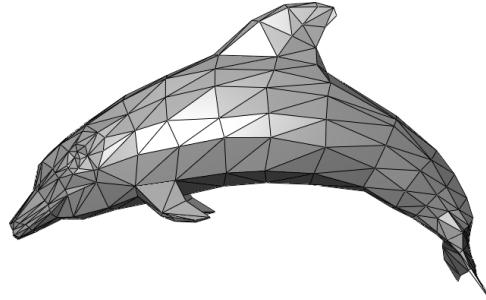


Figura 1. Uma malha de polígonos.

A rasterização é capaz de gerar resultados impressionantes, porém não alcança resultados fotorrealistas. Para se alcançar este tipo de resultado, o ray tracing começou a ser utilizado.

O Ray Tracing, é uma técnica de renderização baseada na simulação do comportamento da luz no mundo real. Os métodos mais utilizadas para a renderização atualmente é a rasterização, onde a imagem é gerada a partir da varredura dos pixeis na tela e calculando qual será o modelo a ser exibido e o Ray Tracing. Ao contrário da abordagem utilizada pela rasterização, o Ray Tracing traça raios a partir da câmera virtual da cena e simula a interação da luz nos objetos. Com este método, é possível obter imagens foto realistas, com reflexos precisos, sombras suaves, refrações e efeitos de iluminação complexos.

A Rasterização processa as primitivas da cena de maneira diferente. Ela utiliza um método onde as primitivas são divididas em triângulos, e a partir desta divisão,

Para diferenciar de maneira mais eficiente o Ray Tracing e a Rasterização, pode-se resumidamente descrever o algoritmo dos dois. A Rasterização irá para cada primitiva, isto é, objeto em nosso mundo 3D, definir os pixeis na tela, enquanto o Ray Tracing irá para cada pixel, definir a primitiva mais próxima. Considera-se que estes dois métodos realizam uma operação inversa para a renderização. [2]

É possível catalogar os principais métodos utilizando Ray Tracing em duas categorias: a *Online* e a *Offline*. O método *Online* equivale dizer que é maneira de renderização em tempo real. Este é o modelo mais desafiador, pois para que uma simulação seja considerada tempo real, o Ray Tracing necessita do melhor desempenho possível. Esta é uma das áreas mais estudadas do Ray Tracing, pois o Ray Tracing completo em tempo real é o tesouro que a indústria procura. Já a metodologia do Ray Tracing *Offline* é o modelo que utiliza a renderização sem ser em tempo real, como animações e efeitos especiais para filmes. Nesta abordagem, é possível a

utilização do Ray Tracing completo e gerar a melhor imagem possível que a técnica pode prover. Para que o Ray Tracing *Online*, chegue na definição e complexidade do *Offline*, será necessário um grande avanço na tecnologia dos dispositivos de processamento. [2]

Na abordagem *Online*, como é impraticável o uso do Ray Tracing completo, costuma-se gerar uma parte da cena com o método de rasterização que é mais eficiente, e utiliza-se o Ray Tracing para renderizar efeitos relacionados à iluminação e reflexos. Exemplos de artefatos que o Ray Tracing pode gerar em conjunto com a Rasterização são: reflexos, sombras, oclusão de ambiente e iluminação global. Oclusão de ambiente considera-se a sombra que cada objeto faz ao contato com outros objetos na cena, e a Iluminação Global refere-se à iluminação gerada pelo Sol ou pelo céu, que ilumina a cena por completo com uma fonte de luz que gera feixes de luz paralelos. Na era em que este artigo está sendo escrito, não existem soluções satisfatórias para Iluminação Global utilizando somente a Rasterização. É possível criar, porém muitas vezes não atinge os padrões esperados para uma aplicação com uso do Ray Tracing.

Além desta classificação, o Ray Tracing pode ser classificado de acordo com o tipo de algoritmo usado. Ray Tracing nada mais quer dizer que traçado de raios, ou seja, traçar vetores dentro de uma cena 3D. Utilizando-se do método do Ray Tracing, também temos o Path Tracing, que quer dizer Traçado de Caminho em português. A principal diferença para o Ray Tracing convencional, é que o Path Tracing utiliza um modelo de distribuição aleatória baseado nas características de Monte Carlo, e que ele gera os raios a partir da câmera virtual até que cheguem em alguma fonte de luz ou a partir da própria fonte de luz. Esta é a principal abordagem utilizada pela indústria, pois gera a imagem ligeiramente mais rápido e é possível utilizar outras técnicas em conjunto do Path Tracing, que serão discutidas em outra seção. [3]

O Ray Tracing também é utilizado para outros fins dentro de uma simulação além da renderização. É possível utilizá-lo para simular áudio em VR, física, detecção de colisão e para auxiliar a inteligência artificial. Estes métodos baseiam-se em calcular o tamanho do traço que foi traçado para realizar os cálculos e as modificações necessárias. [4]

A. Sua história

É complicado definir um criador e uma data para este método. Porem, há vários pesquisadores e artigos sobre o Ray Tracing que contribuíram significamente para seu desenvolvimento e evolução. Um dos marcos para sua história foi o trabalho de Arthur Appel em 1968 no artigo "*Some Techniques for Shading Machine Renderings of Solids*". Neste trabalho, foi descrito os fundamentos do Ray Tracing e foi apresentado técnicas para simular iluminação em objetos tridimensionais. Outros nomes que aparecem em sua história são os de Turner Whitted, James Kajiya e David Kirk. Uma destas contribuições que foi de bastante importância é a Equação de Renderização, proposta por James em 1986. É uma equação que descreve a iteração da luz em objetos tridimensionais, fornecendo um modelo matemático para calcular a aparência de uma superfí-

cie e gerar imagens fotorrealistas. A equação de renderização é definida desta maneira:

$$L_o(\mathbf{p}, \omega_o) = L_e(\mathbf{p}, \omega_o) + \int_{\Omega} f(\mathbf{p}, \omega_i, \omega_o) L_i(\mathbf{p}, \omega_i)(\omega_i \cdot \mathbf{n}) d\omega_i$$

- $L_o(p, \omega_o)$: Radiância de saída do ponto p na direção ω_o .
- $L_e(p, \omega_o)$: Radiância emitida pelo ponto p na direção ω_o .
- $f(p, \omega_i, \omega_o)$: Função de reflexão que descreve a interação da luz entre as direções ω_i e ω_o no ponto p .
- $L_i(p, \omega_i)$: Radiância incidente no ponto p vinda da direção ω_i .
- ω_i, ω_o : Direções incidente e de observação, respectivamente.
- n : Vetor normal à superfície no ponto p .
- Ω : Hemisfério sólido que contém todas as direções de incidência.

Sua solução exata é bastante custosa, porém utilizando o Path Tracing, é possível aproximá-la, graças ao modelo Monte Carlo de amostragem.

B. Seu funcionamento

Primeiro, são traçados os chamados Raios Primários da câmera virtual, passando por cada pixel da imagem. São realizadas checagens para verificar se estes raios interceptaram algum objeto na cena. Caso afirmativo, são gerados os Raios Secundários a partir do local de intersecção e o ângulo que o raio atingiu o objeto. Quando isso ocorre, pode-se dizer que houve um bounce, ou quique, em português. Através desta interação de raios com os objetos, é possível simular reflexões, refrações e sombras. Repete-se este procedimento até que o critério de parada seja atingido. Normalmente, os critérios de parada são um número máximo de bounces, ou se o raio não atingir nenhum objeto na cena. Neste caso, retorna-se a cor de fundo do ambiente, onde na maioria dos casos tem-se uma imagem do horizonte, como uma Skybox, ou uma cor para representar o céu e o horizonte.

Esta técnica do Ray Tracing é interessante, pois nota-se que este algoritmo simula a física de um feixe de luz no mundo real, porém na direção inversa. Por que não simular como na vida real? Infelizmente, tentar simular todos os raios de luz saindo de uma fonte de luz é extremamente custoso e não vale a pena em na maioria dos casos. Com este algoritmo do Ray Tracing, pode-se simular somente os feixes de luz que atingem a câmera virtual, poupando o processamento de feixes que não irão alterar nada na cena.

No Path Tracing, para que seja gerada a imagem, é necessário que haja a interseção do raio com algum artefato na cena. Quando há a interseção, é necessário realizar o seu cálculo. Como mencionado anteriormente, utiliza-se o cálculo de interseção com o triângulo para gerar a imagem. Neste artigo e na maioria dos tutoriais de como construir um Ray Tracer, será usada a esfera como objeto da cena, pois seu cálculo de interseção com a reta é simples.

C. Métodos de Iluminação e Modelos de Materiais

Para que cada objeto possa reagir com a iluminação de maneira correta, cada objeto possui uma propriedade que irá definir como será calculado sua textura. Estas propriedades irão definir como irão interagir com o raio de luz, que são definidas para reflexão, transparência e texturas. Para que seja definida a cor que o objeto terá, também se determina qual será o modelo de reflexão usado. O mais utilizado na computação gráfica é o modelo Phong.

O modelo Phong é utilizado para representar a iluminação em objetos, e é composto por três componentes principais: ambiental, difusa e especular. O componente ambiental representa a luz ambiente que é refletida uniformemente em todas as direções pelo objeto. É a cor base que ele terá independente das outras fontes de iluminação. O componente difuso descreve a reflexão difusa da luz incidente em uma superfície rugosa, que não possui um reflexo com boa definição, por exemplo. A intensidade deste componente depende do ângulo entre a direção da incidência da luz com a normal da superfície, ou seja, o vetor que sai da superfície em 90 graus. Em relação ao componente especular, este se trata da reflexão especular, que ocorre em superfícies polidas, como metais e espelhos. A intensidade deste componente depende do ângulo entre a direção da luz refletida e a direção da câmera. Utiliza-se este componente para representar destaque brilhosos em objetos.

D. Seu uso na mídia de entretenimento

Atualmente, o Ray Tracing é bastante utilizado na mídia do entretenimento. Muito se fala do Ray Tracing em jogos no momento, mas ele também está presente no cinema, em efeitos especiais ou filmes feitos inteiramente em animação. A princípio, utilizava-se a rasterização para a criação de efeitos e modelos, porém, com o tempo, houve uma transição para o uso do Ray Tracing. Esta adaptação ocorreu em duas etapas. Na primeira, utilizou-se de modelos híbridos com rasterização de micro-polígonos e iluminação colocada manualmente para representar quiques de luz refletidas na maioria dos modelos e Ray Tracing padrão para reflexos e algumas sombras. Este estilo foi utilizado no filme "Vida de Insetos"(1998). A segunda etapa foi a transição para o Path Tracing completo, como em "Tá chovendo Hambúrguer"(2009). [5] [6]

O filme "Carros"(2006) utilizou a tecnologia do Ray Tracing em sua produção. Os diretores decidiram usá-lo para poderem alcançar resultados realistas com reflexos, sombras e Oclusão de Ambiente. Na indústria do cinema, o Ray Tracing possui algumas dificuldades, como uma cena ser grande demais para caber em memória, ter milhares de texturas que também não irão caber todas em memória e milhares de pontos de iluminação por exemplo. Por isso foi utilizada uma abordagem híbrida com Ray Tracing e a metodologia REYES, que foi criada pela Pixar. [7]

E. Trabalhos Correlatos

A seguir, serão apresentados trabalhos correlatos à este artigo. Estes trabalhos são recomendações para uma leitura futura, onde o leitor deseja buscar mais conhecimento sobre

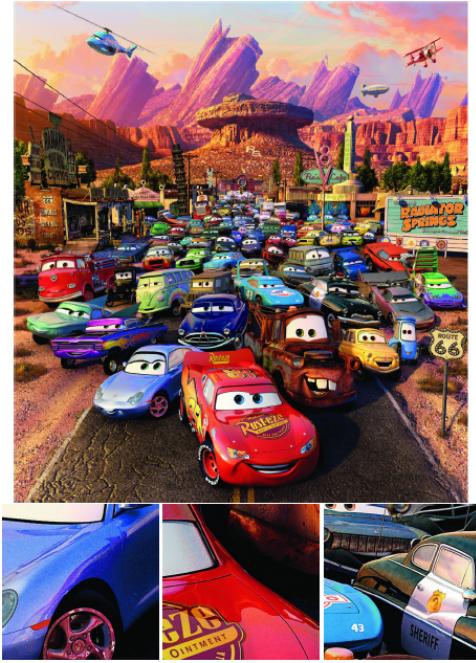


Figura 2. Imagem com todo o elenco principal de Carros. É possível perceber reflexos, sombras e oclusão de ambiente. [7]

este tópico. Estes trabalhos também foram de suma importância para este artigo, sendo que um deles será usado como base para ensinar o funcionamento de um Ray Tracer simples e uma breve análise de performance.

O artigo "*Understandable RayTracing in 256 lines of bare C++*"[8] postado no GitHub, ensina o básico de um Ray Tracer utilizando esferas como formas básicas de materiais na cena, e demonstra o funcionamento de sombras, reflexos e refrações. Além de imagens demonstrativas e descrições sucintas, o código está completo no GitHub junto ao artigo.

O artigo "*smallpt: Global Illumination in 99 lines of C++*"[9] demonstra um Path Tracer que realiza Iluminação Global em apenas 99 linhas de código. Como o objetivo era criar um código enxuto, ele acaba sendo um pouco difícil de acompanhar, porém há várias contribuições da comunidade listada no fim do site, e uma delas é uma apresentação explicando como funciona o código e o que cada linha faz. Este artigo mencionado busca criar um Path Tracer com o menor código possível, omitindo algumas coisas e tornando o código menos legível e mais difícil de compreender. Neste artigo busca-se demonstrar o funcionamento de um Ray Tracer da maneira mais educativa possível, tentando ao máximo explicar tudo que foi realizado.

Já o material mais recomendado pela comunidade para o início da aprendizagem de Ray Tracing é o "*Ray Tracing in One Weekend*" [10] de Peter Shirley (NVIDIA). Peter é um cientista da computação e pesquisador de computação gráfica que é referência na área por suas inúmeras contribuições[11], além desta trilogia. Este livro apresenta uma abordagem passo a passo para ensinar os fundamentos do Ray Tracing por meio da implementação de um renderizador simples. Neste artigo, serão abordados os principais tópicos cobertos pelo primeiro livro.

III. AVANÇOS TECNOLÓGICOS PARA O RAY TRACING

Com a evolução das placas de vídeo e processadores, foi ficando cada vez mais eficaz a renderização de simulações em 3D. Com o surgimento das placas de vídeo dedicadas, a renderização que antes era feita pelo processador, agora poderia ser feita pela placa de vídeo.

Para que o Ray Tracing pudesse ser implementado em simulações em tempo real, como jogos eletrônicos, por exemplo, as fabricantes de placa de vídeo começaram a desenvolver técnicas e chips especiais focados em otimizar os cálculos do Ray Tracing. As técnicas mais utilizadas para otimizar o desempenho e tornar possível o uso de Path Tracing em tempo real incluem algoritmos de *Upscaling*, que utilizam inteligência artificial para aumentar a resolução da imagem, onde a placa de vídeo renderiza o jogo em uma resolução menor e utiliza este algoritmo para aumentar a sua resolução. Outra técnica utilizada é o *Denoising*, onde utiliza-se o Path Tracing com poucos raios para gerar uma imagem incompleta e através da inteligência artificial, gerar uma imagem completa.

A. NVIDIA

A NVIDIA foi a primeira empresa a criar uma placa de vídeo com suporte nativo ao Ray Tracing. A GeForce RTX 20 series, foram lançadas em setembro de 2018. Essa série de placas de vídeo apresentou uma tecnologia de ray tracing em tempo real chamada de NVIDIA RTX. A RTX 2080 Ti, que era a placa de vídeo mais potente desta série, possui 68 RT Cores, que são os núcleos dedicados ao Ray Tracing. Além de um hardware dedicado, a NVIDIA também desenvolveu o DLSS, que significa *Deep Learning Super Sampling*, que utiliza inteligência artificial para implementar o *Upscaling*. Entretanto, como este era a primeira geração de placas de vídeo com componentes dedicados ao Ray Tracing, a maioria dos jogos não conseguiam ter um bom desempenho com esta tecnologia funcionando, principalmente sem o uso do DLSS. Em 2020, a NVIDIA lançou A GeForce 30 series, onde sua placa mais potente é a RTX 3090 Ti. Esta apresentou a segunda geração de sua RT Cores, possuindo desta vez 84. Também foi apresentado o DLSS 2.0, que teve um grande ganho em relação ao primeiro. Nesta etapa, já era possível utilizar Ray Tracing em jogos utilizando o DLSS 2.0 e adquirir um resultado minimamente satisfatório. Em 2022, foram lançadas as NVIDIA GeForce RTX 40 series. Até o momento de desenvolvimento deste artigo, tem-se a RTX 4090 como a placa mais poderosa da NVIDIA. Esta placa possui 128 RT Cores, suporte ao novo DLSS 3.0 e à nova tecnologia desenvolvida pela NVIDIA para melhorar o desempenho, que é o *Frame Generation*. Esta tecnologia cria frames novos entre dois frames, através da inteligência artificial previamente treinada.

Com a geração atual das placas da série 40 e todos estes outros recursos auxiliando a renderização, como o DLSS 3.0 e o *Frame Generation* [12], já é possível ter uma experiência com Ray Tracing bastante satisfatória. Entretanto, ainda não há tecnologia o suficiente para que se possa ter um jogo ou simulação em tempo real utilizando-se apenas o Ray Tracing. Os jogos que possuem esta funcionalidade estão na categoria de renderização híbrida, utilizando a rasterização

para a maioria das cenas, porém deixando iluminação, sombras e/ou reflexos para serem calculados pelo Ray Tracing.



Figura 3. A placa de vídeo RTX 4090 da NVIDIA. Modelo de referência.

B. AMD

As placas da NVidia foram as primeiras a possuírem o suporte ao Ray Tracing. A AMD ficou uma geração atrás de sua concorrente por este motivo. Porém, a AMD se prontificou a adicionar o suporte ao Ray Tracing em sua próxima geração de placas de vídeo.

Para concorrer com a tecnologia RTX de sua concorrente, a AMD adicionou o suporte em sua arquitetura já existente RDNA, porém em sua segunda versão, o RDNA 2, em 2020. Esta arquitetura de chips está presente nas placas da série RX 6000, a partir da placa RX 6600. Nesta série, a placa mais potente da AMD é a RX 6950 XT, que possui 80 Ray Accelerators dedicado a cálculos relacionados ao Ray Tracing. Em 2022, a AMD anunciou a arquitetura RDNA 3 para suas novas placas de vídeo da série RX 7000. Durante o desenvolvimento deste artigo, tem-se que a placa mais potente da AMD é a RX 7900 XTX, possuindo 96 Ray Accelerators.

A AMD também possui tecnologias que auxiliam na renderização, tanto na rasterização comum, quanto no uso do Ray Tracing, o *FidelityFX Super Resolution*, ou como é mais conhecido, FSR. Esta tecnologia funciona de maneira semelhante ao DLSS da NVidia, renderizando o jogo em uma resolução e utilizando técnicas de inteligência artificial para criar uma imagem de resolução maior, através do *Upscaling*. Estas duas tecnologias podem ser configuradas para que seja obtido um resultado de acordo com a necessidade do usuário. É possível selecionar se estes algoritmos irão priorizar a performance ou a qualidade de imagem, onde o usuário escolhe onde ele quer. O FSR possui as configurações: *Ultra Quality*, *Quality*, *Balanced*, *Performance* e *Ultra Performance*. Este tipo de otimização é primordial para que um jogo com Ray Tracing ativado possa ter um desempenho aceitável.

C. NVIDIA x AMD

A NVidia, no momento, possui uma vantagem sobre a AMD tratando-se de desempenho de jogos ou aplicações que utilizem Ray Tracing, principalmente se esta aplicação possuir suporte às técnicas de performance da NVidia como o DLSS e o mais recente *Frame Generation*. Entretanto, as placas de vídeo da AMD vêm apresentando um ótimo desempenho na



Figura 4. A placa de vídeo Radeon RX 7900 XTX da AMD. Modelo de referência.

rasterização, que é a técnica padrão usada pelos jogos para renderizar uma cena. Outros fatores que ajudam a AMD, são que os preços das placas de vídeo da AMD costumam ser mais acessíveis ao público, que se tornou mais evidente nesta última geração de placas de vídeo.

Recentemente, a Intel entrou no mercado de placas de vídeo dedicadas, onde mesmo com uma entrada um pouco desastrada, agora no período em que este artigo está sendo escrito, está sendo uma alternativa viável para o consumidor que deseja obter um computador de ótima performance, gastando menos do que iria gastar com a concorrência.

D. Intel

A Intel surgiu recentemente no mercado de placas de vídeo dedicadas, porém esta tecnologia não é novidade para ela. Em sua primeira geração de processadores da linha Core, estes processadores vinham com uma placa de vídeo integrada, porém é capaz de somente atividades básicas como vídeos e jogos muito leves.[13] Já na segunda geração de sua linha Core, estas placas de vídeo integradas foram atualizadas para a Intel HD Graphics 3000. [14] Ainda bastante longe de um desempenho de uma placa dedicada de sua mesma geração. Atualmente, a Intel está na décima terceira geração de seus processadores Core, e possui versões com e sem a placa de vídeo integrada. Os processadores que não possuem a placa de integrada têm a letra "F" no final de sua descrição, como por exemplo, o Intel Core i9-13900F. A Intel possui processadores tanto para computadores de mesa, quanto um modelo para notebooks. O processador para computador de mesa possuem a placa integrada *Gráficos UHD Intel 770*, enquanto os processadores de notebook possuem a *Intel Iris Xe*, que são mais potentes.

Indo agora para as placas dedicadas, a Intel chegou a este mercado em 2022, começando pelo seu modelo de entrada, a Intel Arc A350M. Em se tratando de placas de vídeo, quando vê-se o pós-fixo "M" em um modelo, sabe-se que este é um modelo para notebooks. A Intel começou lançando notebooks com a sua placa de vídeo dedicada e depois começou a comercializar suas placas voltadas ao computador de mesa. Primeira mente com a Intel ARC A380. Um modelo de entrada que não agradou bastante, principalmente pelos grandes problemas iniciais relacionados aos drivers, onde os jogos possuíam um desempenho bem abaixo do que a placa conseguia e muita instabilidade. Esta placa possui 8 Ray Tracing Units, que infelizmente, devido ao fato da placa já

possuir um desempenho de entrada, não pode ser aproveitado para o Ray Tracing. Depois de alguns meses, a Intel finalmente mostrou suas placas que seriam as placas que iriam competir com a NVIDIA e a AMD no mercado de placas de vídeo dedicadas intermediárias, as Intel ARC A750 e Intel ARC A770. Estas placas possuem 32 Ray Tracing Units, que devido ao fato destas placas possuírem um desempenho satisfatório, é possível utilizar esta placa para jogos e simulações com Ray Tracing, desde que não seja em uma resolução maior que 1080p e que preferivelmente a aplicação possua suporte ao XeSS, a tecnologia da Intel de *Upscaling* de resolução.

Após quase um ano de lançamento, os drivers já estão em um estado satisfatório, onde vários problemas foram resolvidos, como o péssimo suporte a versões anteriores do DirectX 12, uma API utilizada para a criação de jogos atuais. Seu preço também foi diminuído, o que a torna ainda mais competitiva no mercado.



Figura 5. A placa de vídeo Intel ARC A770. Modelo de referência.

E. Consoles <TERMINAR>

A nova geração de consoles também apresenta hardware dedicado ao processamento de Ray Tracing. Tanto o Xbox Series S e X da Microsoft, quanto o PlayStation 5 da Sony, utilizam uma placa de vídeo dedicada moderna com a capacidade de renderizar com a aceleração de hardware. Os três consoles previamente citados possuem um processador gráfico feito pela AMD. O Xbox Serie X, que é o mais potente dos Xbox, e o PlayStation 5 possuem uma placa de vídeo bastante similar.

F. Outras <TERMINAR>

Com o rápido avanço na tecnologia, já existem chips gráficos para celulares que possuem suporte ao Ray Tracing.

IV. ANÁLISE DO Ray Tracing in one Weekend

Nessa seção, será discutido o livro "*Ray Tracing in One Weekend*" de Peter Shirley. [10] Será realizada um breve resumo do que cada capítulo do primeiro livro da trilogia relata. Ao final, será realizado um teste para mensurar a performance do programa. Com isto, pode-se ter uma noção do que cada seção do livro trata. A análise será feita no código completo que está disponível no GitHub [15], porém, como o código pode ser alterado a qualquer momento, foi criado um repositório no GitHub [16] com um fork do original, onde a versão utilizada neste artigo estará intacta, e as alterações

realizadas para realizar os testes e os programas utilizados para criar os gráficos também estão presentes, para que o leitor possa reproduzir estes testes de maneira mais simples possível. Os comandos utilizados e como rodar o projeto também se encontra no README do projeto.

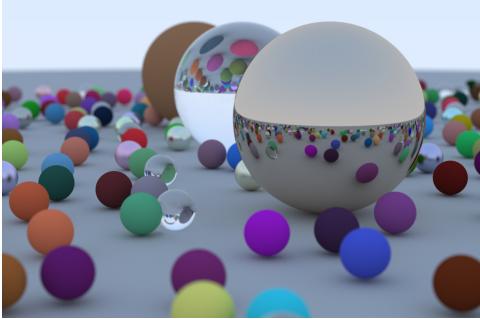


Figura 6. Imagem final gerada do Ray Tracing in One Weekend.

Capítulo 1: Overview

Neste capítulo, o autor fornece o background do livro e o que o motivou a fazê-lo. Também dá uma ideia geral do que será feito e recomendações para o leitor. Uma nota interessante que o autor faz que cabe neste artigo: "Ray Tracing pode ser qualquer coisa, o que irei ensinar é tecnicamente um Path Tracer". É citado o conhecimento prévio que o autor espera que o leitor tenha, porém é possível acompanhar todo o processo mesmo que o conhecimento sobre estes tópicos sejam básicos. O livro é lecionado totalmente em C++.

Capítulo 2: Output an Image

O foco aqui é demonstrar ao leitor como será gerada a saída do programa. A imagem gerada pelo Ray Tracer será salva em um formato PPM. O Windows, por exemplo, não possui suporte nativo a este tipo de formato, o que pode ser um pouco inconveniente. Entretanto, é possível visualizar este formato de imagem direto da internet através de um site, como o *PPM Viewer* [17]. Mas vale ressaltar que o formato PPM possui um formato simples, que ajuda na criação de imagens. O autor também sugere o uso da biblioteca *stb_image.h* [18] caso o leitor queira produzir outros tipos de formatos de imagem, porém fica a escolha do leitor, pois não é demonstrado como utilizar esta biblioteca. Neste capítulo, é demonstrado como inserir uma barra de progresso no seu programa.

Capítulo 3: The vec3 Class

Operações com vetores estão presente em praticamente qualquer aplicação que envolva gráficos em 3D, principalmente produto escalar e produto vetorial. Neste capítulo, é fornecido o arquivo de cabeçalho contendo todas as funções de utilidade em vetores que são utilizadas. Também é demonstrado como será escrita a cor obtida do arquivo de saída.

Capítulo 4: Rays, a Simple Camera, and Background

Como o título sugere, esta parte é dedicada à demonstração de como a classe Raio será definida e como funciona. Logo após, o autor adiciona esta classe ao projeto e define uma câmera simples. Com estas duas estruturas definidas, já é possível traçar os raios na cena. Define-se a cor que será retornada caso o raio traçado não atinja nenhuma entidade na cena, que neste caso representa o céu com um gradiente em azul, e é testado o resultado. Deve-se obter uma imagem contendo o gradiente definido anteriormente, devido ao fato da cena ainda não conter nenhum objeto.

Capítulo 5: Adding a Sphere

Ao final deste capítulo, será possível adicionar o primeiro objeto à cena: uma esfera. A escolha da esfera tem uma razão. A equação de interseção da reta com a esfera é bem direta, o que reduz a complexidade do programa. Após o autor demonstrar como a matemática funciona, fornece ao leitos o trecho de código equivalente às funções demonstradas. Após a definição da função de interseção e da adição de uma esfera na cena, obtém-se uma imagem contendo uma esfera no meio. Nota-se que a esfera não tem aparência de um objeto 3D. Isto se deve ao fato que ainda não foi implementado a iluminação, que é o principal fator que nos dá a impressão da cena em que estamos vendo é uma representação de três dimensões.

Capítulo 6: Surface Normals and Multiple Objects

Nesta seção, o autor aborda a importância das normais de superfície na renderização realista. Primeiro é explicado o conceito de normal, que são vetores perpendiculares às superfícies do objeto. As normais desempenham um papel crucial na determinação das interações da luz com os objetos, pois são elas que determinam a cor que o objeto terá, definindo seu material. Demonstra-se uma função para retornar a normal do objeto atingido pelo raio traçado. Este capítulo também aborda a criação de uma lista de objetos para serem adicionados na cena, através de uma estrutura de dados. Uma função para determinar qual foi o hit mais próximo também implementada, pois será de ajuda quando há mais de um objeto na trajetória do raio. Ao final, após adicionar outra esfera para representar o chão, traça-se os raios e gera-se uma imagem final. Com uma função definida previamente neste capítulo que determina a cor do objeto de acordo com sua normal, tem-se agora uma bola colorida que já é possível perceber que trata-se de um objeto em três dimensões.

Capítulo 7: Antialiasing

Observa-se que a imagem gerada não apresenta uma esfera perfeitamente redonda, é possível notar um serrilhado em seu contorno. Para resolver este artefato, utiliza-se alguma das técnicas de Antialiasing. Este problema ocorre pois neste caso tem-se uma amostragem discreta, onde há somente uma amostragem por pixel para definir sua cor. Neste Path Tracer, o problema será solucionado traçando mais de um raio por pixel para que a cor definida por este pixel seja mais precisa, fazendo uma média das cores obtidas pelos raios. O autor

também demonstra uma função para gerar números aleatórios que serão úteis para o programa. Após as alterações no programa e na câmera, tem-se a imagem final da esfera com suas bordas suavizadas.

Capítulo 8: Diffuse Materials

Este capítulo discute a diferença entre a geometria dos objetos e seus materiais. O autor questiona se é mais adequado ter uma separação entre estes objetos ou se seria melhor mantê-los relacionados. A opção escolhida é mantê-los separados, como é feito na maioria dos renderizadores, porém ressalta que há limitações com esta escolha. Ele apresenta o conceito de um material difuso, que são objetos que não emitem luz, mas adquirem a cor do ambiente ao seu redor. Este objeto reflete os raios de luz de maneira aleatória. Em seguida, é definido uma profundidade máxima para os raios, pois caso eles não demorem muito para pararem de refletir, é possível que a pilha da memória seja estourada. Logo após, o autor introduz ao conceito da correção gama para garantir a intensidade da cor precisa antes que a imagem seja armazenada. O autor também demonstra uma maneira de resolver um problema e melhorar a sombra gerada pela esfera. Ao final, Peter apresenta o método da dispersão hemisférica, que é um método alternativo de representar objetos difusos.

Capítulo 9: Metal

Este capítulo introduz o conceito de materiais metálicos, que possuem uma taxa alta de reflexão. O fato destes materiais terem uma propriedade de reflexão especular difere dos outros. O material metálico irá refletir a luz de maneira ordenada e precisa, criando reflexões nítidas e brilhantes. O autor demonstra o funcionamento deste tipo de material e o adiciona ao código principal.

Capítulo 10: Dielectrics

Materiais dielétricos são materiais como a água, vidro e diamantes. Este tipo de material é um dos materiais que o Ray Tracing é capaz de renderizar de maneira mais realista, que o difere de outros métodos de renderização. A refração pode ser adquirida através da lei de Snell, que descreve o comportamento de um raio sendo refratado. Outro desafio é retratar o raio que está sendo refletido dentro do corpo dielétrico, que é chamada de refração interna total. O autor trata deste problema e demonstra formas de resolvê-lo. Para o vidro, tem-se a equação de aproximação de Schlick, que modela a variação da reflectividade do vidro de acordo com o ângulo de incidência. Ao final, uma esfera com refração é adicionada à cena, junto de outra metálica.

Capítulo 11: Positionable Camera

Neste capítulo, o autor foca em dar mais funcionalidade para a câmera, onde é possível alterar sua posição, sua orientação e seu ângulo de visão. Com isso é possível gerar cenas a partir de pontos de visão diferentes e com aproximações diferentes, através da alteração do ângulo de visão.

Capítulo 12: Defocus Blur

Um dos efeitos utilizados para simular a visão humana ou de câmeras focais é a profundidade de campo, onde um objeto em foco fica nítido e o fundo fica fora de foco. Isso ocorre devido ao fato das câmeras possuírem um orifício para a entrada da luz, que acaba gerando o desfoque. Para corrigir isso, usa-se lentes angulares. Para gerar o efeito final que será adicionado ao Path Tracer, o autor implementa uma aproximação do que seria uma lente fina, que é a mais utilizada em computação gráfica para este efeito. Ajustando a distância do foco e a abertura da lente é possível ajustar o efeito.

Capítulo 13: Where Next?

Esta seção são as considerações finais do autor. Ele demonstra como reproduzir a imagem da capa do livro, que está representada pela figura 6, e sugere trabalhos futuros como implementar luz, adicionar a equação de interseção de triângulos, já que quase todos os modelos 3D hoje em dia são baseados em triângulos, adicionar texturas, adicionar modelos com volume e implementar este programa utilizando paralelismo.

V. ANALISANDO O RAY TRACING IN ONE WEEKEND

Ao chegar ao final da implementação deste Path Tracer, pode-se ter o questionamento de como a performance deste programa se comporta, pois a cada capítulo, nota-se que a imagem final leva cada vez mais tempo para que seja finalizada. Ao alterar os parâmetros do renderizador para testar a função de cada uma e para gerar uma imagem que seja agradável ao leitor, também percebe-se que a imagem final demora cada vez mais. A seguir, serão apresentados testes realizados para que fosse possível ter uma ideia de como cada parâmetro que é alterado no Path Tracer irá influenciar em sua performance geral.

A. Configurações de estudo

Este teste baseia-se em executar o código fonte do Path Tracer várias vezes, onde foi alterado somente o parâmetro que está sendo estudado. O código fonte do livro encontra-se no GitHub [15], porém, o código está aberto para a comunidade e vários usuários já contribuíram para o código final. Para que fosse mantida a reproduzibilidade deste estudo, foi criado um fork do código fonte no momento do desenvolvimento deste artigo [16].

Os testes foram feitos alterando o código fonte original em C++, criando um laço *for* para que o programa fosse executado várias vezes com valores incrementais para o parâmetro escolhido, enquanto os outros permaneceram inalterados. No final, o programa gera um arquivo texto contendo o tempo da execução e os parâmetros utilizados naquela execução. Um outro programa em Python3 lê este arquivo e cria um gráfico.

Estes testes foram realizados em dois computadores diferentes, o primeiro possui a seguinte configuração, sendo um computador de mesa:

- Processador: AMD Ryzen 5 3600X
- Memória RAM: 16 GB, 2400 MHz DDR4

- Armazenamento: SSD NVMe
- Sistema Operacional: Linux Mint (Ubuntu)

O outro computador usado para os testes é um notebook com configurações inferiores, representadas a seguir:

- Processador: Intel Core i5 Segunda Geração
- Memória RAM: 8 GB, 2400 MHz DDR4
- Armazenamento: SSD
- Sistema Operacional: Linux Mint (Ubuntu)

B. Resultados de Pesquisa

Discussões

Limitações

VI. CONCLUSÃO

REFERÊNCIAS

- [1] C. Yuksel, “Interactive graphics 03 - rendering algorithms,” 2022, university of Utah, YouTube. [Online]. Available: <https://www.youtube.com/watch?v=owx-R-Ary9I>
- [2] ——, “Interactive graphics 26 - gpu ray tracing,” 2022, university of Utah, YouTube. [Online]. Available: <https://www.youtube.com/watch?v=qW6rJ0s2Xv0>
- [3] “Disney’s practical guide to path tracing,” 2016, disney. [Online]. Available: https://www.youtube.com/watch?v=ftrLwRLS_ZR0
- [4] “Introduction to real-time ray tracing,” 2018, nVIDIA. [Online]. Available: <https://www.youtube.com/watch?v=AmMUqqjQfo0>
- [5] A. Keller, T. Viitanen, C. Barré-Brisebois, C. Schied, and M. McGuire, “Are we done with ray tracing?” *ACM SIGGRAPH 2019 Courses (SIGGRAPH ’19)*, p. 2, 2019. [Online]. Available: <https://doi.org/10.1145/3305366.3328096>
- [6] A. Keller, L. Foscione, M. Fajardo, I. Georgiev, P. Christensen, J. Hanika, C. Eisenacher, and G. Nichols, “The path tracing revolution in the movie industry,” *ACM SIGGRAPH 2015 Courses (SIGGRAPH ’15)*, p. 24, 2015.
- [7] P. H. Christensen, J. Fong, D. M. Laur, and D. Batali, “Ray tracing for the movie ‘cars’,” *Pixar Graphics*, 2006, pixar Animation Studio. [Online]. Available: <https://graphics.pixar.com/library/RayTracingCars/paper.pdf>
- [8] D. V. Sokolov, “Understandable raytracing in 256 lines of bare c++,” 2019, university of Lorraine, GitHub. [Online]. Available: <https://github.com/ssloy/tinyraytracer/wiki/Part-1:-understandable-raytracing>
- [9] K. Beason, “smallpt: Global illumination in 99 lines of c++,” 2014, university of Utah, YouTube. [Online]. Available: <https://www.youtube.com/watch?v=owx-R-Ary9I>
- [10] P. Shirley, “Ray tracing in one weekend,” December 2020. [Online]. Available: <https://raytracing.github.io/books/RayTracingInOneWeekend.html>
- [11] ——, “Publications of peter shirley,” Online, 2023, <https://www.researchgate.net/profile/Peter-Shirley-2>.
- [12] B. Burke, “Nvidia introduces dlss 3 with breakthrough ai-powered frame generation for up to 4x performance,” 2022, nVIDIA Corp. [Online]. Available: https://nvidianews.nvidia.com/_gallery/download_pdf/6329dab9ed6ae51abfa606ed/
- [13] Intel, “Processador intel® core™ i7-660ue,” 2023, intel Corporation. [Online]. Available: <https://ark.intel.com/content/www;br/pt/ark/products/50022/intel-core-i7660ue-processor-4m-cache-1-33-ghz.html>
- [14] ——, “Processador intel® core™ i3-2310e,” 2023, intel Corporation. [Online]. Available: <https://ark.intel.com/content/www;br/pt/ark/products/54643/intel-core-i32310e-processor-3m-cache-2-10-ghz.html>
- [15] S. Hollasch, “Ray tracing in one weekend book series,” 2020, intel Corporation. [Online]. Available: <https://github.com/RayTracing/raytracing.github.io>
- [16] L. Pena, “Repositório para os testes.” 2023, universidade de Brasília. [Online]. Available: <https://github.com/lucpena/raytracing.github.io-Artigo>
- [17] “Ppm viewer,” 2023. [Online]. Available: https://www.cs.rhodes.edu/welshc/COMP141_F16/ppmReader.html
- [18] S. Hollasch, “Ray tracing in one weekend book series,” 2020, intel Corporation. [Online]. Available: <https://github.com/RayTracing/raytracing.github.io>