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

Lucas Araújo Pena
13/0056162

***Resumo*—Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec id nisl eros. Phasellus eleifend nunc mi, nec rutrum purus lacinia sit amet. Vivamus eu tellus dui. 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**

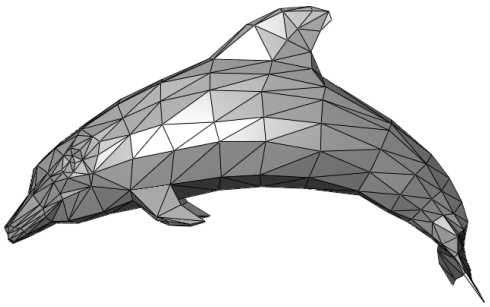

Figura 1. Uma malha de polígonos.

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.

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 pixeis 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]

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

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

- $L_o(p, \omega_o)$: 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 destaques 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 Hamburguer"(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 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 do materias 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 e 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 video 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 video 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 NVIDA 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. Entretando, ainda não ha 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.

B. AMD

As placas da NVidia foram as primeiras a possuirem 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 da 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.

C. Intel

D. Outras

IV. Definição Técnica

V. Analisando o Ray Tracing in one Weekend

VI. Resultados de estudo

VII. 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=frLwRLS_ZR0

[4] "Introduction to real-time ray tracing," 2018, nVIDIA. [Online]. Available: https://www.youtube.com/watch?v=AmMUqpjQfo0

[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. Fascione, 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/