

Ray Tracing é uma técnica da computação gráfica que tenta replicar o funcionamento da luz do mundo real, onde a fonte emissora emite fótons que rebatem em superfícies e vão se degradando até serem completamente absorvidos. Em cada rebate, o fóton perde energia, pois cada material absorve algumas cores e reflete outras. O raio de luz original tem suas cores subtraídas e, consequentemente, alteradas em cada contato, até chegar no observador, onde a cor final depende do último contato feito com algum material. Por exemplo, uma luz branca rebate em uma parede branca e chega ao observador. A cor final será branca, pois a parede branca reflete todas as cores. Se fosse vermelha, a cor final seria vermelha, pois a parede absorveu todas as cores menos a vermelha. Se após ter rebatido na parede vermelha, o raio rebateu em um objeto azul, o observador verá uma região preta, pois o objeto azul absorveu o vermelho que era a única cor que sobrava no raio.

O ângulo de incidência do raio, também influencia a cor observada, tornando-a mais clara ou escura. Uma fonte de luz produz diversos raios que se espalham em quase todas as direções (depende do formato da fonte de luz). Para o computador, calcular todos esses raios (na computação são objetos matemáticos que possuem uma origem e uma direção, assim como um vetor) com ângulos, reflexões e refrações é muito custoso, tornando o desempenho muito lento, além de ser pouco otimizado, pois vários desses raios não chegam na câmera, desperdiçando processamento em coisas que serão invisíveis para o observador.

Para otimizar o modelo do mundo real, os desenvolvedores do Ray Tracing fizeram os raios de luz partirem da câmera, como se ela fosse a fonte de luz. Fazendo o processo inverso da luz, o raio sai da câmera, rebate no ambiente (mesmo processo explicado anteriormente, com a luz) e vai em direção à fonte de luz. Também é definido um limite de reflexão máximo para os raios. Ou seja, um raio pode rebater até 'x' vezes, antes de ir ao encontro da fonte de luz.

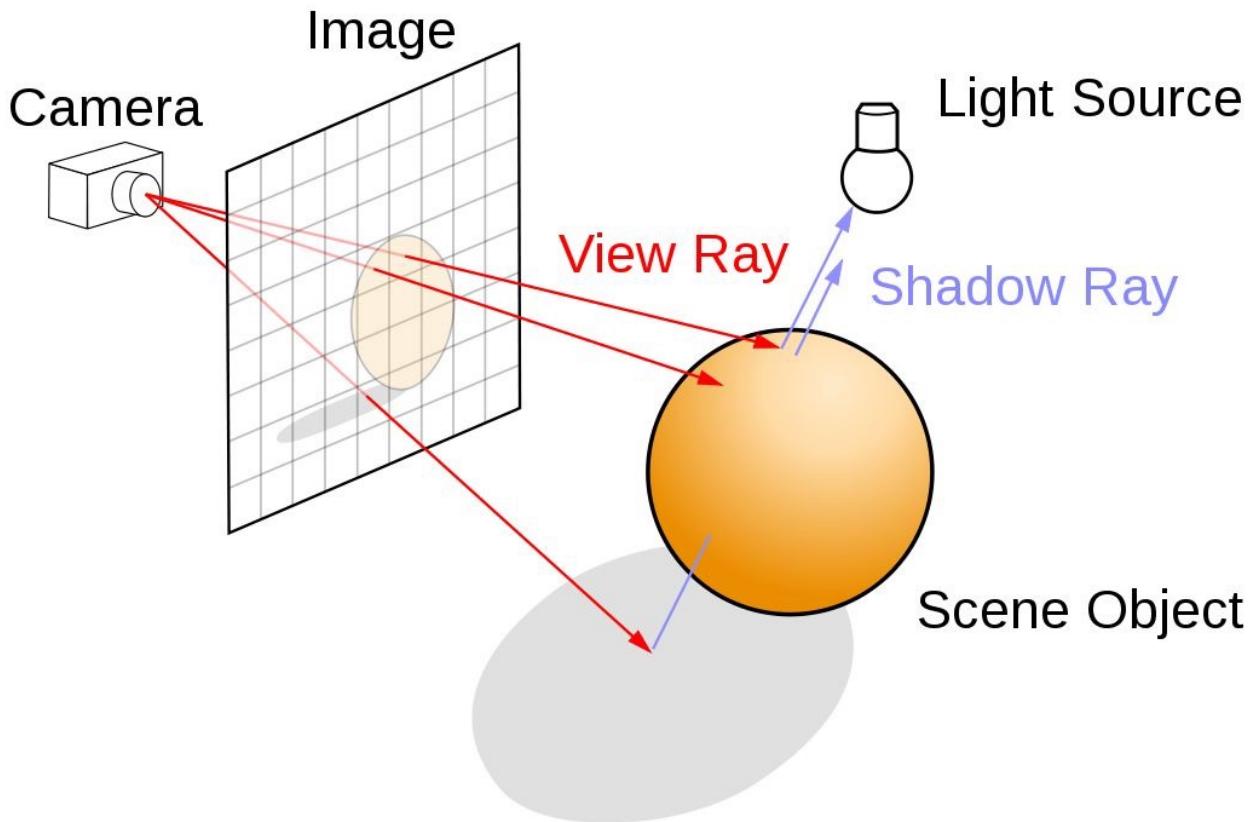


Figura 1: Básico do Ray Tracing.

Fonte: <https://developer.nvidia.com/discover/ray-tracing>

A figura 1 mostra o funcionamento básico do Ray Tracing. Os raios são criados com o ponto de origem na câmera, passam por um plano onde será gerada a imagem e encontram o objeto ou um

ponto qualquer. A partir disso a cor do pixel é calculada, combinando a quantidade estimada de luz nesse ponto de encontro com a informação do material da superfície.

A computação trabalha com números. Para calcular o valor do pixel obtém-se o valor da luz em valores RGB e multiplica-se pelo valor da cor da superfície, combinado com o ângulo de incidência do raio, que controla o brilho da cor. O vetor normal também é usado em conjunto com o ângulo de incidência para controlar o brilho. Quando o vetor normal aponta na mesma direção do ângulo de onde vem a fonte de luz, o valor será 1, tornando a cor, muito brilhante. Se esses vetores forem perpendiculares, o valor será 0, tornando a cor, escura.

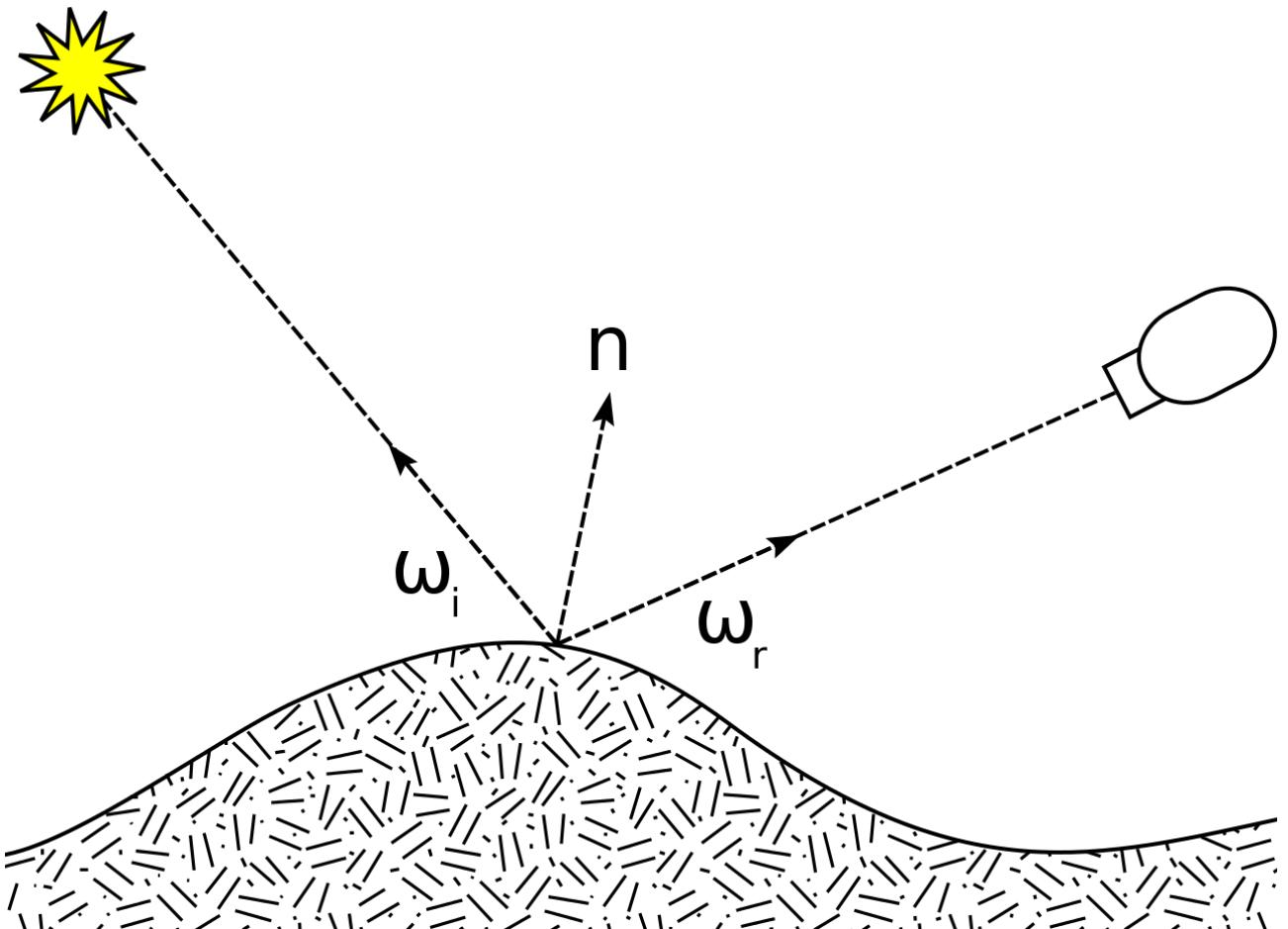


Figura 2: Ângulo de reflexão da luz e o vetor normal.

Fonte: https://en.wikipedia.org/wiki/Bidirectional_reflectance_distribution_function

O vetor normal é necessário pois, com o Ray Tracing emitindo os raios da câmera em direção ao ambiente pode fazer com que todos os raios apontem para a fonte de luz, mesmo mudando a inclinação do objeto que está sendo apontado pela câmera. Isso causa imprecisão, com a cor do pixel sendo brilhante, quando, na verdade, deveria ser escuro, pois no modelo real, os raios se espalhariam da fonte de luz até a câmera, “errando” ela, fazendo a imagem ficar mais escura.

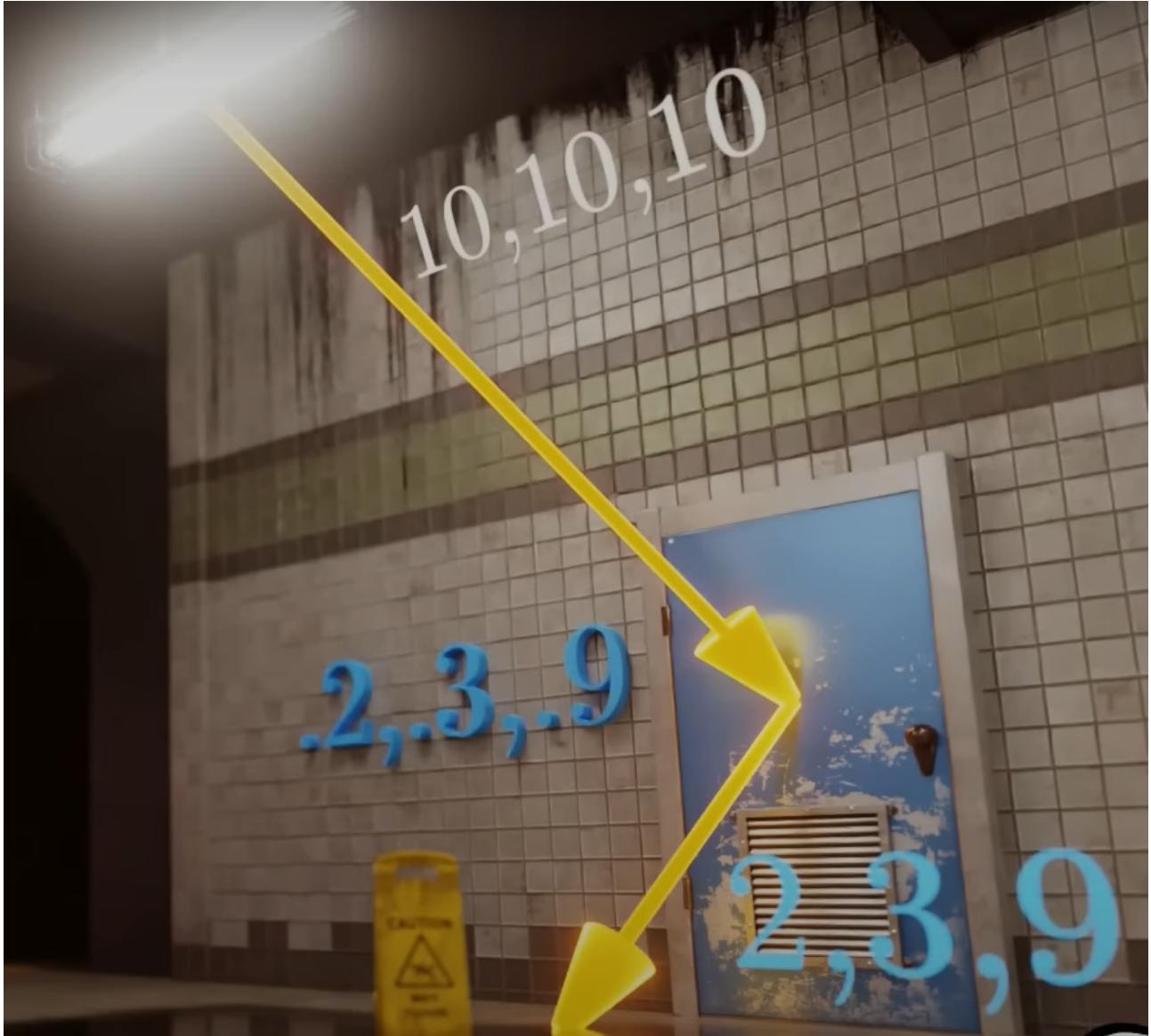


Figura 3: Cálculo feito para definir a cor do pixel. Quando o raio de luz com o valor (10, 10, 10) passa pela porta com valor (0.2, 0.3, 0.9) o resultado é a multiplicação desses 2 valores (2, 3, 9).

Fonte: <https://youtu.be/gsZiJeaMO48>

O Ray Tracing possui diferentes técnicas que conseguem acelerar ou tornar mais preciso o resultado do final do pixel. Uma delas, criado por Kajiya, usa vários raios que são disparados através de um único pixel em diferentes direções. Esses raios são combinados e a média dos valores deles é calculada. Essa técnica usa o método de Monte Carlo, que usa amostragem aleatória para conseguir bons resultados.

O desempenho do Monte Carlo pode ser acelerado definindo quais amostras possuem mais chances de serem escolhidas por contribuírem mais com o resultado final. Regiões que possuem fontes de luzes mais próximas podem ser selecionadas mais frequentemente que outras mais distantes.

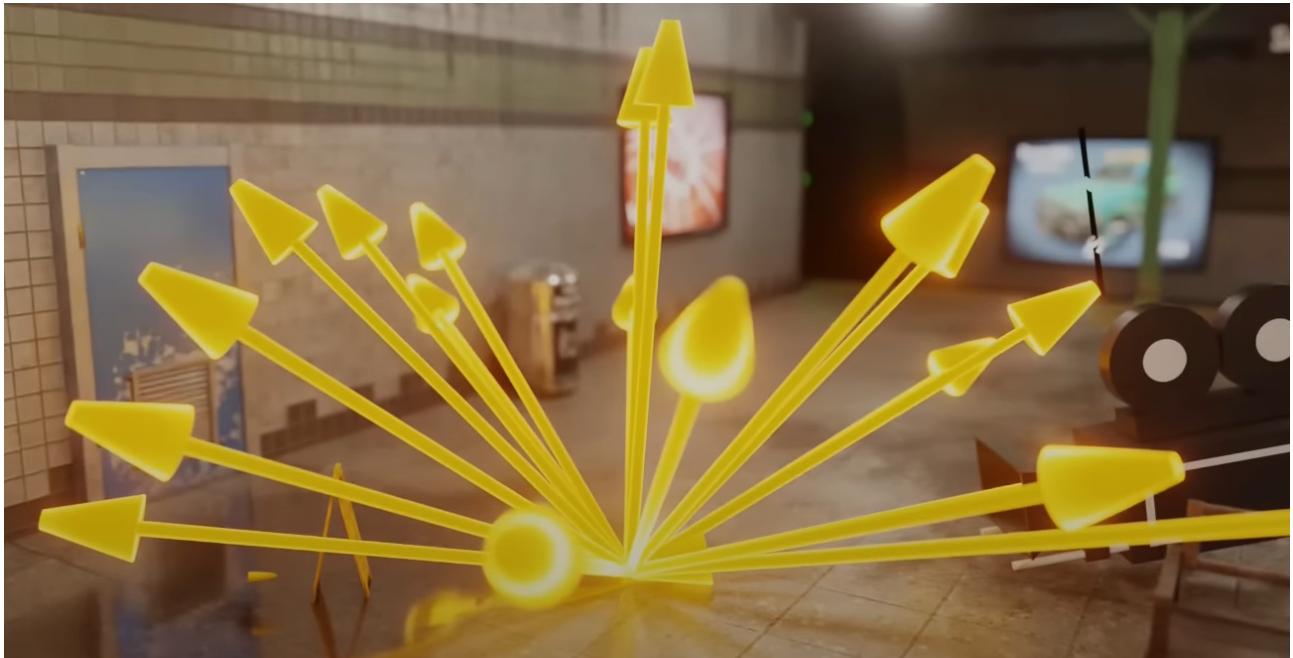


Figura 4: Representação do método criado por Kajiya. Vários raios são disparados a partir de um único pixel.

Fonte: <https://youtu.be/gsZiJeaMO48>

Outra técnica é chamada de Ray Tracing estocástico, que é semelhante ao Monte Carlo, possuindo vários raios por pixel, mas eles são direcionados à fonte de luz, com uma pequena variação. O resultado gerado são sombras mais suaves.

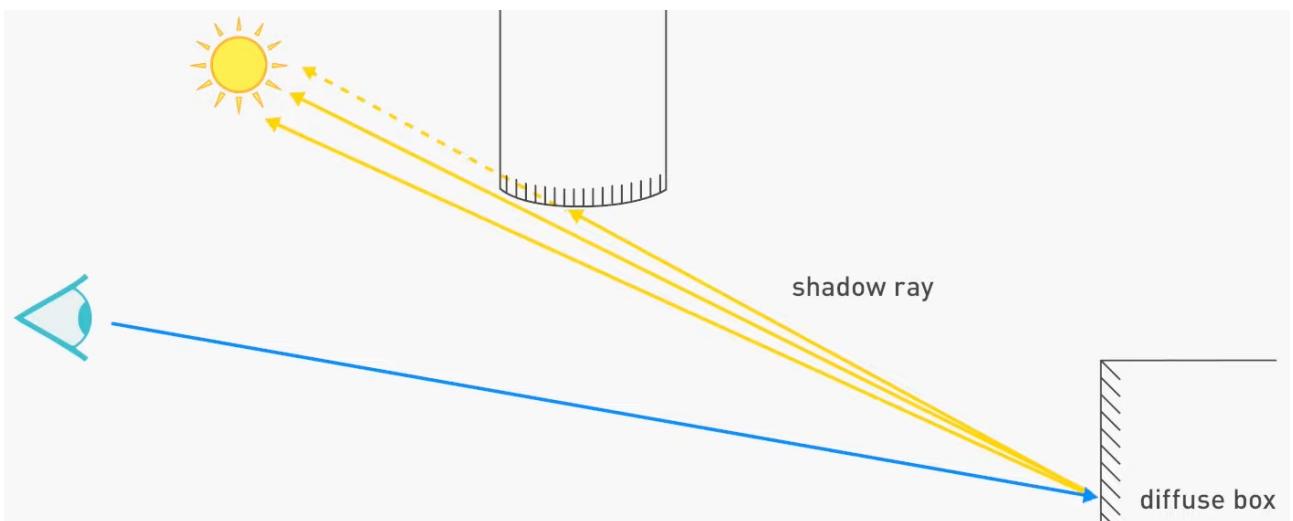


Figura 5: Ray Tracing estocástico. São emitidos 3 raios e 2 deles atingem a fonte de luz. Nesse caso, o pixel está 2/3 iluminado. Quanto mais raios são emitidos, mais preciso é o resultado da iluminação do pixel, mas mais custoso e mais lento de processar.

Fonte: <https://youtu.be/gBPNO6ruevk>

Existe uma técnica chamada de inter reflexão difusa. Vários raios são disparados em direções aleatórias que geram novos raios secundários. Os pixels que atingem a fonte de luz, são os mais importantes e terão um peso maior no cálculo do pixel. Cada pixel é armazenado como uma amostra, pois todos os resultados são válidos para o valor final do pixel. A característica difusa de cada objeto, causa um problema para se calcular o comportamento de cada raio de luz.

A quantidade de raios computados, definem a qualidade da iluminação da imagem final. O desenvolvedor Kajiya desenvolveu o algoritmo de path tracing com uma equação de renderização para resolver o problema de complexidade de vários raios. Através de técnicas estatísticas com integração Monte Carlo, foi possível resolver a equação de renderização através do caminho de cada raio.

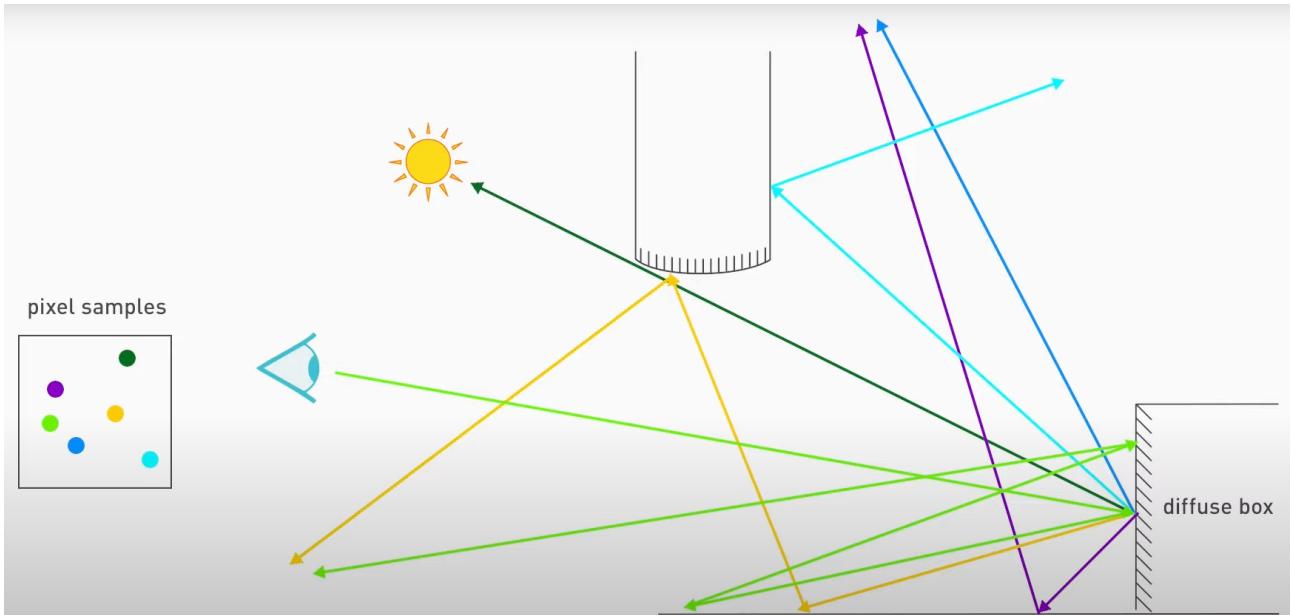


Figura 6: Esquema da inter reflexão difusa, por Kajiya.

Fonte: <https://youtu.be/gBPNO6ruevk>

O aumento do desempenho das tecnologias de hardware, levaram o Ray Tracing a ser usado em filmes, pinturas digitais e, mais recentemente, em jogos.

Alguns filmes que usam RT: City of Lost Children (1995), Vida de Inseto (1998), A Casa Monstro (2006), Carros (2006), Poseidon (2006), Speed Racer (2008), Universidade Monstros (2013), Toy Story 4 (2019).

Como filmes não precisam trabalhar com renderização em tempo real, o processo pode ser focado em precisão e aparência. Ou seja, pode levar diversas horas para renderizar um único frame. Para não levar anos para finalizar um filme, são usadas diversas máquinas que trabalham frames diferentes, em paralelo.



Figura 7: Carros 3 (2017). Detalhe das luzes refletidas no Lightning McQueen.

Fonte: <https://www.vfxvoice.com/cars-3-pixar-embraces-new-renderer-with-stunning-detail/>

Alguns jogos que usam RT: Battlefield V (2018), Shadow of the Tomb Raider (2018), Control (2019), Metro Exodus (2019), Cyberpunk 2077 (2020), Minecraft RTX (2020), Battlefield 2042 (2021), Justice (2022).



Figura 8: Battlefield V (2018). Comparaçao entre o Ray Tracing desativado à esquerda e ativado à direita.

Fonte: <https://adrenaline.com.br/artigos/v/57227/diferenca-visual-e-de-performance-do-ray-tracing-rodando-nas-rtx-2080-ti-e-2080>

Os jogos precisam de hardware poderoso para executar a renderização em tempo real de cada frame, a 60 ou mais vezes por segundo. Com o avanço das tecnologias de placas de vídeo, foi possível aplicar Ray Tracing nos jogos, com um bom desempenho. O avanço tecnológico nos hardwares levou à implementação de núcleos de processamento dedicados ao cálculo de Ray Tracing, otimizando a linha de montagem de cada frame.

As séries placas de vídeo que possuem hardware dedicado para cálculo de Ray Tracing, são os seguintes:

NVIDIA GeForce séries 20, 30 e 40.
AMD Radeon RX séries 6000 e 7000.

Referências

<https://youtu.be/gsZiJeaMO48>

<https://youtu.be/oCsgTrGLDiI>

<https://youtu.be/0FMlPUEAZfs>

<https://youtu.be/oCsgTrGLDiI>

<https://youtu.be/gBPNO6ruevk>

<https://developer.nvidia.com/discover/ray-tracing>

<https://blog.nvidia.com.br/2022/05/10/o-que-e-path-tracing/>

<https://graphics.pixar.com/library/PathTracedMovies/paper.pdf>