Ray tracing (gráfico), uma aproximação da equação de renderização

Antonio César de Andrade Júnior[1]

[1]Engenharia de Computação–Universidade Federal do Ceará (UFC)
Quixadá–CE–Brasil

ceand12@alu.ufc.br

Abstract. *Ray tracing is an approximation of James T. Kajiya's rendering equation, which simulates the effects of light on objects such as reflections, shadows and refractions. Applications of Ray tracing will be shown, the structure of the rendering equation, the operation of a Ray tracing algorithm, the results of this algorithm in practice and an idea of how to improve rendering and how to generate an animation using this algorithm.*

Resumo. *Ray tracing é uma aproximação da equação de renderização de James T. Kajiya, que simula os efeitos da luz em objetos, como reflxão, sombra e refração. Será mostrado aplicações do Ray tracing, a estrutura da equação de renderização, o funcionamento de uma algoritmo de Ray tracing, os resultados desse algoritmo na prática e uma ideia de como melhorar a renderização e como gerar uma animação utilizando esse algoritmo.*

1. Introdução

Uma das maiores ambições, pelo menos nas últimas 2 décadas, de produtoras de jogos, filmes e pesquisadores que trabalham diretamente com renderização gráfica, é a simulação da luz e seus efeitos em um ambiente virtual. Por exemplo, o reflexo em um espelho, raios de sol entrando por uma brecha, sombra e refração da luz em objetos.

A técnica mais utilizada para tornar esses efeitos possíveis é o *Ray tracing*, ou traçado de raio. Essa técnica não simula o raio de luz saindo da fonte, refletindo em algo e chegando ao observador, mas sim o caminho contrário, isso se deve a limitação de poder computacional [Wofgang Leister 1988].

O algoritmo de *Ray tracing* é uma aproximação da equação de renderização, citada pela primeira vez por James T. Kajiya, em 1986. Esse artigo irá mostrar o funcionamento de um algoritmo de *Ray tracing*.

2. Aplicações

Hoje, a maioria dos jogos usam imagens rasterizadas, ou seja, imagens formadas por pixels, essas imagens são renderizadas por *shaders*, que são conjuntos de instruções que definem os efeitos nas superfícies de objetos mostrados na tela. Esses *shaders* tentam imitar, não simular, os efeitos da luz na cena apresentada [Demartini 2011].

Os *shaders* estão sendo substituídos por *Ray tracing*, metódo usado para simular os efeitos da luz em uma imagem, e com menos carga de trabalho para o desenvolvedor, já

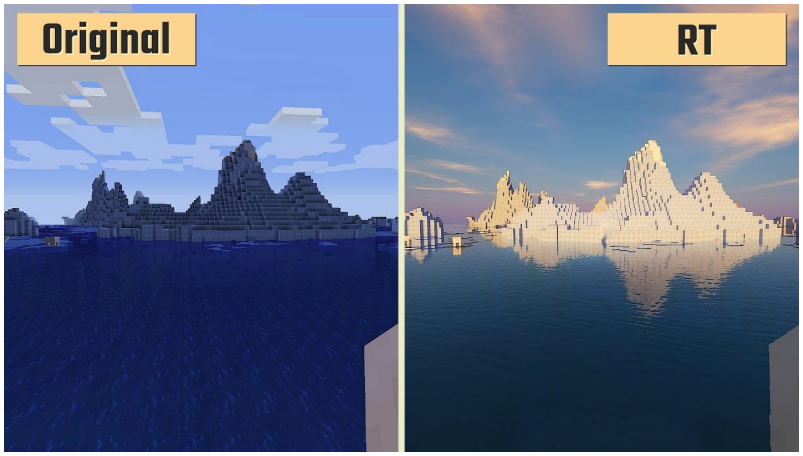

Figure 1. Comparativo entre shader e Ray tracing

que tudo é processado em tempo real. Porém, essa ainda é uma técnica de alto custo computacional, por isso ainda existe a opção de usar ou não o *Ray tracing*. Um comparativo entre *shader* e *Ray tracing* é mostrado na Figura 1.

Estúdios de filmes usam outra aproximação da equação de renderização para renderizar, por exemplo, animações, chamada de *Path tracing*. Funciona como um *Ray tracing* aprimorado, porém ao invés de traçar um raio por vez, ela traça vários [Disney 2019]. O problema dessa técnica é seu custo computacional, um dos motivos de ser utilizada apenas em imagens estáticas e filmes. Abaixo é apresentada uma imagem que exemplifica essa técnica (Figura 2).

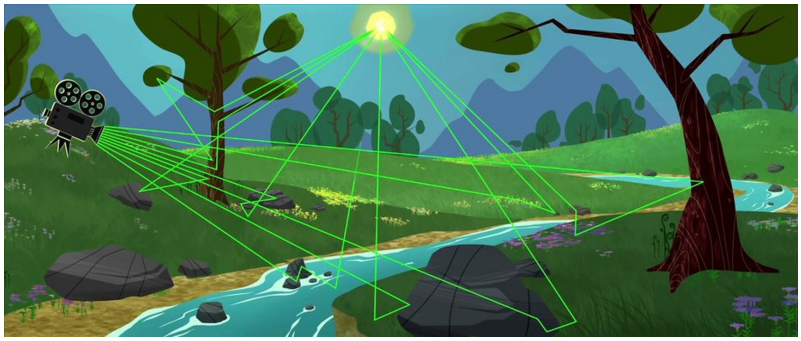

Figure 2. Traçado de raios no path tracing

3. Equação de renderização

A equação de renderização é uma equação integral de Fredholm do segundo tipo, que define a quantidade de luz emitida de um ponto x até um determinado observador. É apresentada na seguinte forma [Kajiya 1986]:

$$L_0(x, \omega_0, \lambda, t) = L_e(x, \omega_0, \lambda, t) + \int_\Omega f_r(x, \omega_i, \omega_0, \lambda, t) L_i(x, \omega_i, \lambda, t)(\omega_i \cdot n) d\omega_i \quad (1)$$

- $L_0(x, \omega_0, \lambda, t)$ é a radiância espectral total do comprimento de λ dirigido para fora ao longo da direção ω_0 no tempo t, de uma posição particular x;

- x é um ponto no espaço;
- ω_0 é a direção da luz que sai;
- λ é o comprimento da onda de luz;
- t é tempo;
- $L_e(x, \omega_0, \lambda, t)$ é a radiação espectral emitida;

$$L_{e,\Omega,\lambda} = \frac{\partial L_{e,\Omega}}{\partial \lambda} \tag{2}$$

$$L_{e,\Omega} = \frac{\partial^2 \Phi_e}{\partial \Omega \partial A cos\theta} \tag{3}$$

- $\int_\Omega ...d\omega_i$ é uma integral sobre Ω;
- Ω é a unidade do hemisfério centrado em torno de n, contendo todos os valores possíveis para ω_i;
- $f_r(x, \omega_i, \omega_0, \lambda, t)$ é a função de distribuição de refletância bidirecional (BRDF). A proporção de luz refletida de ω_i para ω_0 na posição x, tempo t, e comprimento de onda λ;

$$f_r(\omega_i, \omega_r) = \frac{dL_r(\omega_r)}{L_i(\omega_i) cos\theta_i d\omega_i} \tag{4}$$

- ω_i é a direção que a luz entra;
- $L_i(x, \omega_i, \lambda, t)$ é a radiância espectral do comprimento de onda λ vindo para dentro em direção x da direção ω_i no tempo t;
- n é a superfície normal em x;
- $\omega_i \cdot n$ é o fator de enfraquecimento da irradiância externa devido ao ângulo de incidência.Pode ser escrito como $cos\theta_i$;

3.1. Solução

A equação de renderização não possui uma solução analítica exata, pelo fato de que a equação terá que ser calculada em infinitos pontos recursivamente, ou seja, terão que ser resolvidas infinitas integrais. Mas, exitem aproximações que limitam o número de recursões [Křivánek 2015]. O *Ray tracing* é uma dessas aproximações.

Para exemplificar essa recursão, a equação de renderização, pode ser escrita na forma de operador linear:

$$L = L_e + T \circ L \tag{5}$$

Na próxima interação, ela ficaria como:

$$L = L_e + T \circ (L_e + T \circ L) \tag{6}$$

$$L = L_e + TL_e + T^2 \circ L \tag{7}$$

Isso produz uma série de Neumann na forma:

$$L = \sum_{i=0}^{n} T^i L_e + T^{n+1} \circ L \tag{8}$$

4. Algoritmo Ray tracing

Suponha um espaço de 3 dimensões, com uma única fonte de luz. Agora suponha uma câmera nesse espaço, como uma tela feita de pixels na frente da câmera e uma esfera depois da tela (Figura 3).

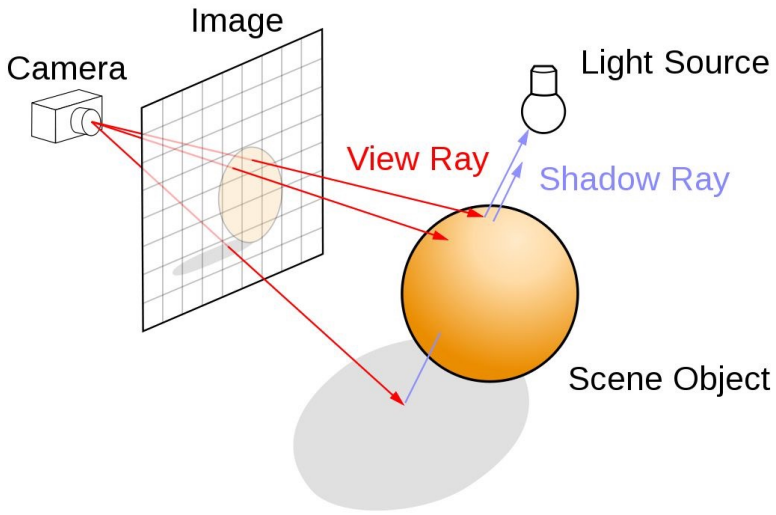

Figure 3. Modelo

O algoritmo funciona de uma maneira contrária ao mundo real, ou seja, o raio sai do observador, passa pela tela e se o raio refletir em algum objeto, entre a tela e a fonte de luz, o pixel por onde o raio passou ganhará a cor associada ao objeto [Wofgang Leister 1988]. Abaixo é apresentado um pseudo-código.

```
for cada pixel p(x,y,z) do
    p recebe a cor preta;
    if raio cruzar um objeto then
        calcula a intersecção para o objeto mais próximo;
    else
        calcula a cor do ponto;
        p recebe cor do ponto;
    end
end
```

Algorithm 1: Ray tracing

5. Resultados

Foram feitas 2 simulações, uma com a câmera fixada nas coordenadas (0;0;1) e outra com a câmera em (2;5;1). Mas, nos dois casos, as duas esferas usadas, tiveram seus centros fixados nas coordenadas (0,1;-0,3;0) e (-0,3;0;0), e a fonte de luz em (5;5;5).

No primeiro caso, a renderização saiu como esperado, com as formas esféricas perfeitas, sombra, refração e reflexão (Figura 4). Já no segundo, as esferas ficaram distorcidas (Figura 5).

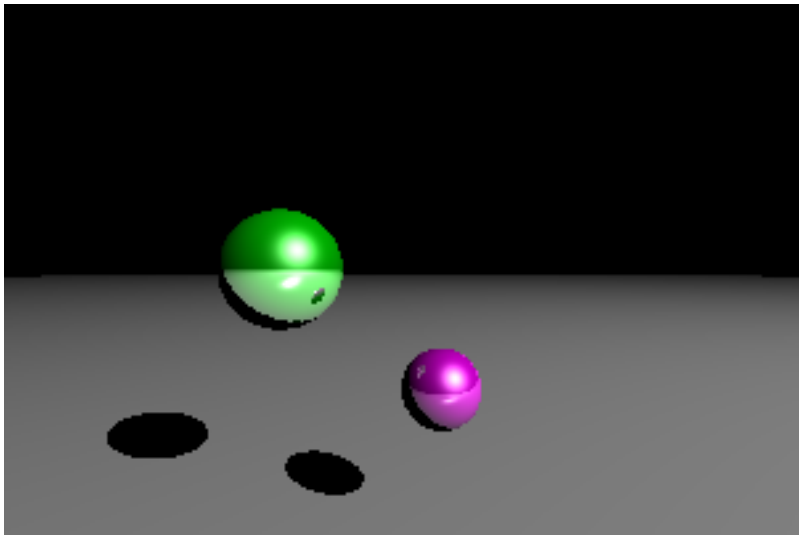

Figure 4. Caso 1

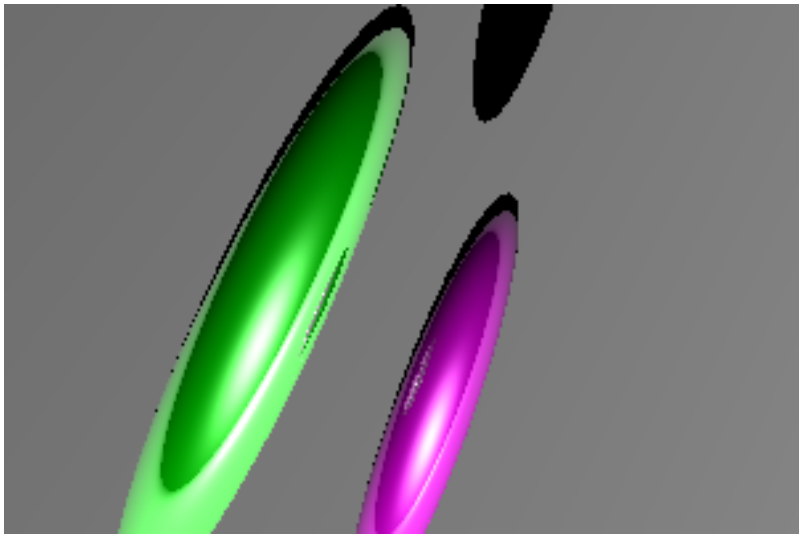

Figure 5. Caso 2

6. Conclusão

Na primeira simulação foi obtido o resultado esperado, já que a câmera foi posicionada levando em consideração a posição das esferas. No segundo caso, as esferas ficaram distorcidas, pois a câmera foi deslocada para uma posição que considera pixels com coordenadas diferentes.

Levando esse fato em consideração, pode ser feita uma renderização com várias coordenadas diferentes para a câmera, gerando, desse modo, uma animação. Também poderiam ser colocadas outras fontes de luz e ser implementado um algoritmo de *Path tracing*, para gerar imagens mais realistas.

References

Demartini, F. (2011). Shaders: o que são e para que servem? https://www.tecmundo.com.br/voxel/especiais/

182970-shaders-o-que-sao-e-para-que-servem-.htm.

Disney (2019). Disney's hyperion renderer. https://www.disneyanimation.com/technology/hyperion/.

Kajiya, J. T. (1986). The rendering equation. *Dallas*, 20(4).

Křivánek, J. (2015). Computer graphics iii – rendering equation and its solution. *MFF UK*.

Wofgang Leister, Heinrich Muller, A. S. (1988). Ray tracing algorithms – theory and practice. *Theoretical Foundations of Computer Graphics and CAD - Springer Verlag*.