

Relatório do Projeto de Ray Tracing

Implementação de Primitivas, Transformações e Reflexão

February 22, 2026

Abstract

Este relatório descreve a formulação matemática e a implementação de recursos de *Ray Tracing* desenvolvidos no projeto. O escopo abrange desde o cálculo de interseção para primitivas geométricas básicas (Cubo e Cilindro), passando pela aplicação de transformações aos objetos, até à renderização de superfícies implícitas e simulação de reflexões (espelhos).

1 Interseção com Primitivas Geométricas

Para renderizar os objetos na cena, o algoritmo de *ray tracing* calcula o ponto de interseção do raio emitido pela câmara com a geometria do objeto. O raio paramétrico é dado por:

$$p(t) = o + td, \quad t \geq 0 \quad (1)$$

onde o é a origem do raio e d é a direção.

1.1 Cubo (Método das *Slabs*)

Considere um cubo alinhado aos eixos, centrado na origem, com faces limitadas por s :

$$x, y, z \in \left[-\frac{s}{2}, \frac{s}{2}\right] \quad (2)$$

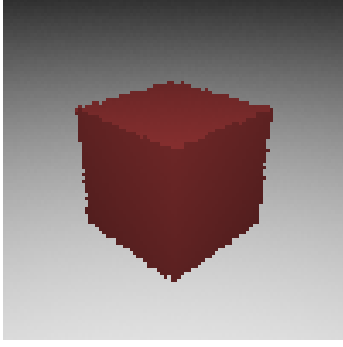
Para determinar a interseção, avaliamos o raio contra os três pares de planos paralelos (as *slabs*). Para cada eixo, montamos o seguinte sistema de desigualdades:

$$x_{min} \leq o_x + d_x t \leq x_{max} \quad (3)$$

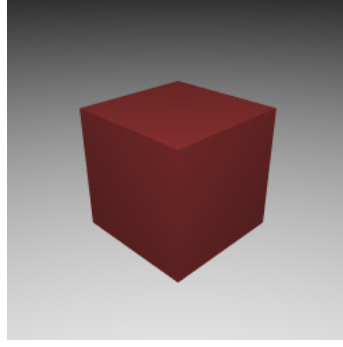
$$y_{min} \leq o_y + d_y t \leq y_{max} \quad (4)$$

$$z_{min} \leq o_z + d_z t \leq z_{max} \quad (5)$$

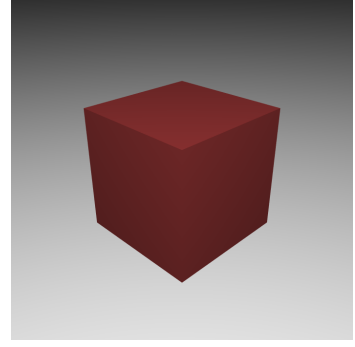
O menor valor de t positivo que satisfaz simultaneamente a interseção com todos os limites corresponde ao ponto de impacto visível. A evolução visual da renderização com o aumento de amostras (*samples*) pode ser observada abaixo:



(a) 100x100, 0 samples



(b) 250x250, 128 samples



(c) 700x700, 128 samples

Figure 1: Renderizações da primitiva Cubo com diferentes resoluções e *anti-aliasing*.

1.2 Cilindro

Para verificar a interseção do raio paramétrico com um cilindro alinhado ao eixo z , substituímos os componentes x e y do raio na equação da superfície lateral do cilindro:

$$(o_x + d_x t)^2 + (o_y + d_y t)^2 = r^2 \quad (6)$$

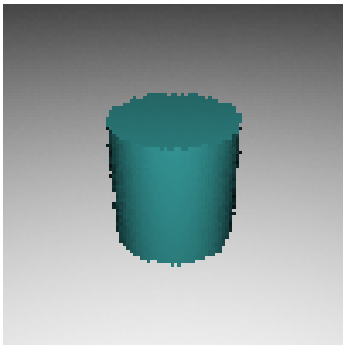
Expandindo os termos, obtemos uma equação de grau 2 na forma $at^2 + bt + c = 0$, onde:

$$a = d_x^2 + d_y^2 \quad (7)$$

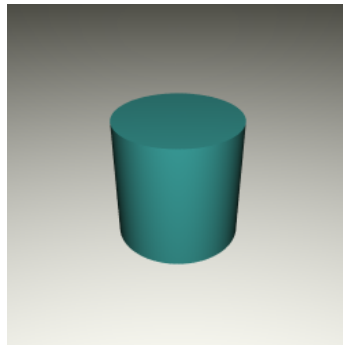
$$b = 2(o_x d_x + o_y d_y) \quad (8)$$

$$c = o_x^2 + o_y^2 - r^2 \quad (9)$$

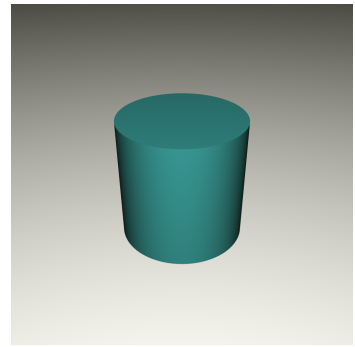
Pegamos no menor t positivo que resolve a equação e verificamos se a curva do raio avaliado nele satisfaz as restrições de altura. Caso contrário, verificamos a interseção com as tampas planas.



(a) 250x250, 64 samples



(b) 250x250, 128 samples



(c) 700x700, 128 samples

Figure 2: Evolução da renderização da primitiva Cilindro.

2 Transformações de Objetos

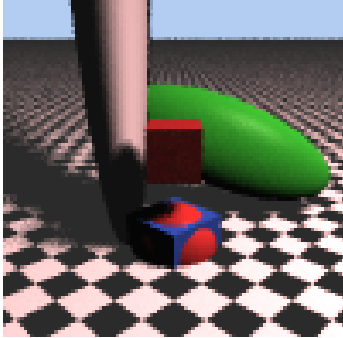
Para posicionar, rodar e escalonar objetos sem reescrever as suas equações de interseção, utilizamos a classe `ObjectTransform`. Seja M a matriz de transformação. Aplicamos

a transformação inversa M^{-1} ao raio incidente, levando-o para o espaço local do objeto:

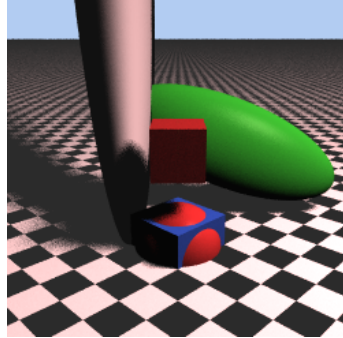
$$w = M^{-1}x \quad \Rightarrow \quad g(x) = f(M^{-1}x) \quad (10)$$

A normal $\nabla f(w)$ é mapeada de volta para o espaço do mundo multiplicando pela transposta da inversa:

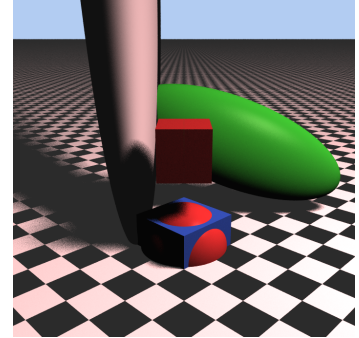
$$\nabla g(x) = (M^{-1})^T \nabla f(M^{-1}x) \quad (11)$$



(a) 100x100, 32 samples

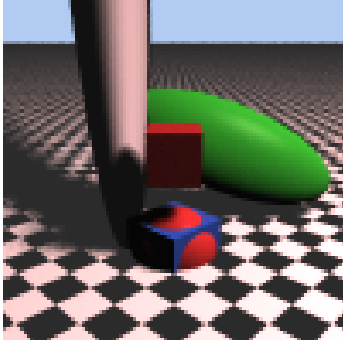


(b) 250x250, 32 samples

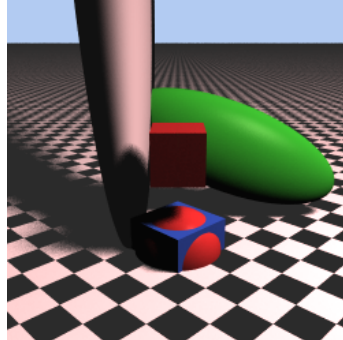


(c) 700x700, 32 samples

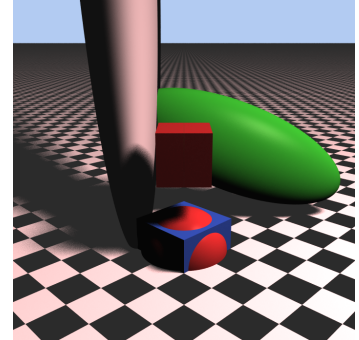
Figure 3: Aplicação de transformações matriciais a objetos na cena.



(a) 100x100, 700 samples



(b) 250x250, 700 samples

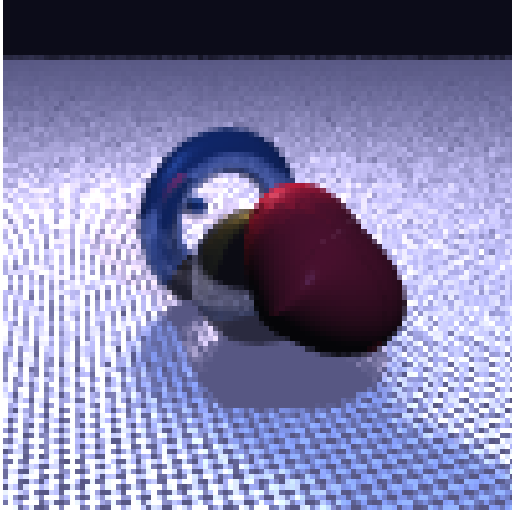


(c) 700x700, 700 samples

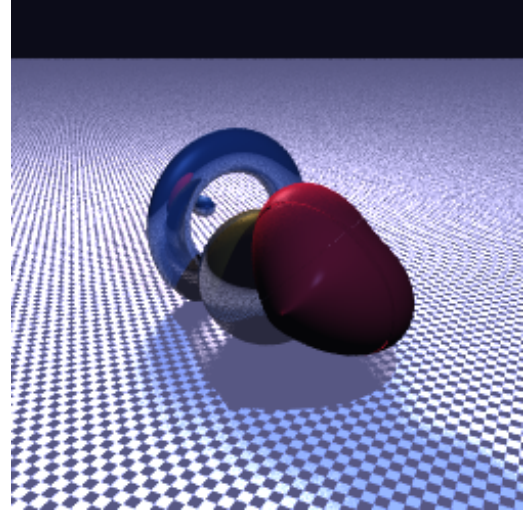
Figure 4: Aplicação de transformações matriciais a objetos na cena.

3 Superfícies Implícitas (Mitchel e Heart)

A interseção do raio com superfícies de *Mitchel* e coração (*Heart*) não possui solução analítica simples. Utilizamos algoritmos de busca numéricos (como verificação de sinais e bisseção) ao longo do raio para encontrar as raízes.

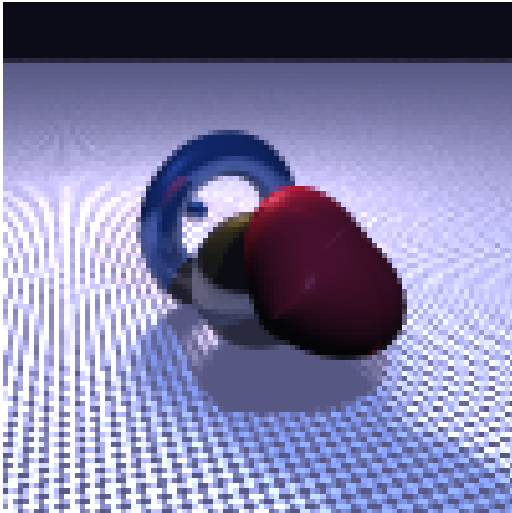


(a) Heart/Mitchel 100x100, 32 samples

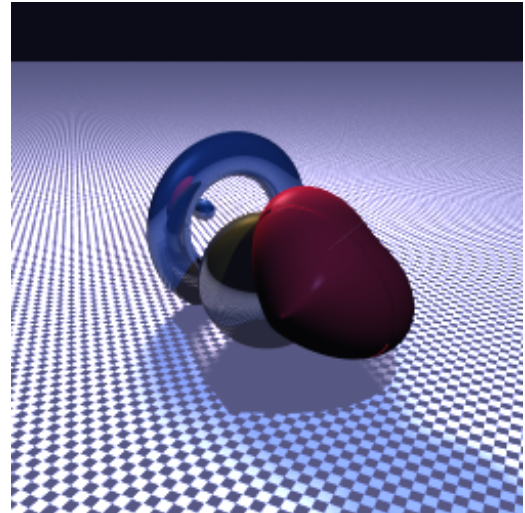


(b) Heart/Mitchel 700x700, 32 samples

Figure 5: Renderização de superfícies implícitas com aproximação numérica.



(a) Heart/Mitchel 100x100, 700 samples



(b) Heart/Mitchel 700x700, 700 samples

Figure 6: Renderização de superfícies implícitas com aproximação numérica.

4 Reflexão e Espelhos

O *ray tracer* foi estendido para suportar reflexões. A direção da reflexão \vec{R} é dada por:

$$\vec{R} = \vec{d} - 2(\vec{d} \cdot \vec{n})\vec{n} \quad (12)$$

O raio refletido continua a propagar-se pela cena num processo recursivo, limitado pelo parâmetro `max_depth` para evitar sobrecarga em reflexões infinitas.

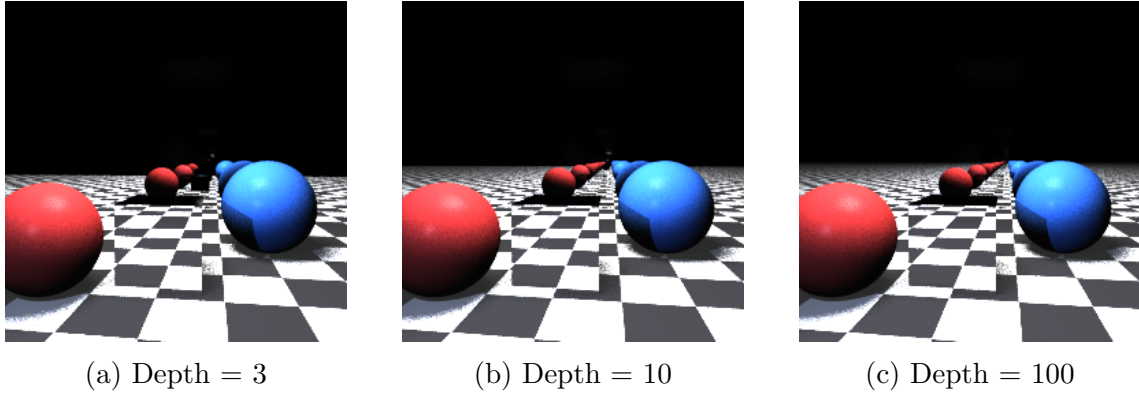


Figure 7: Simulação de espelhos infinitos variando o limite de recursão (max_depth).

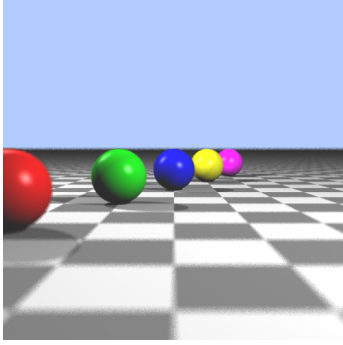
5 Profundidade de Campo

Para simular o efeito fotográfico de profundidade de campo, substituímos a clássica câmara do tipo *pinhole* “ideal” por um modelo de câmara com abertura de lente maior que zero (abertura > 0). No modelo *pinhole* ideal, a imagem inteira está sempre perfeitamente em foco pois todos os raios convergem exatamente num único ponto matemático.

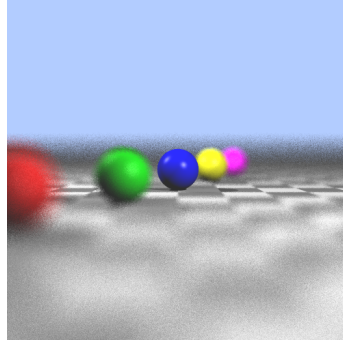
Ao utilizar uma abertura, estabelece-se um **plano focal** na cena. O ponto de interseção ótimo projetado nesse plano é chamado de p_f . Durante o processamento, a origem do raio da câmara não parte mais apenas do centro geométrico, mas sim de pontos amostrados uniformemente ao longo da área de uma lente (um disco). O novo raio toma a direção apontando da origem amostrada na lente diretamente para o ponto focal p_f .

Com essa técnica, a luz advinda de objetos que intercebam o plano focal continua alinhada com p_f , e sua imagem fica nítida na projeção final. Por outro lado, a luz de objetos fora desta distância focal dispersa os raios, provocando o característico desfoque nas regiões próximas ou longínquas.

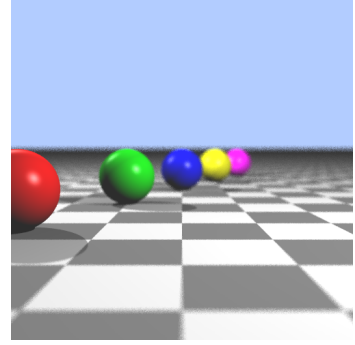
Abaixo, podemos observar as renderizações demonstrando este fenómeno, variando a posição do plano focal:



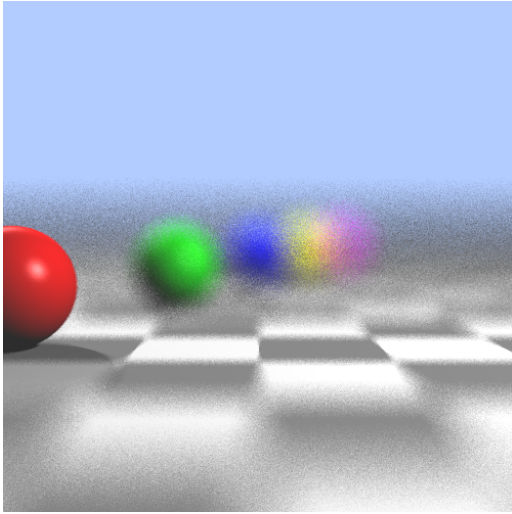
(a) Raio = 0.1, Distância Focal = 20



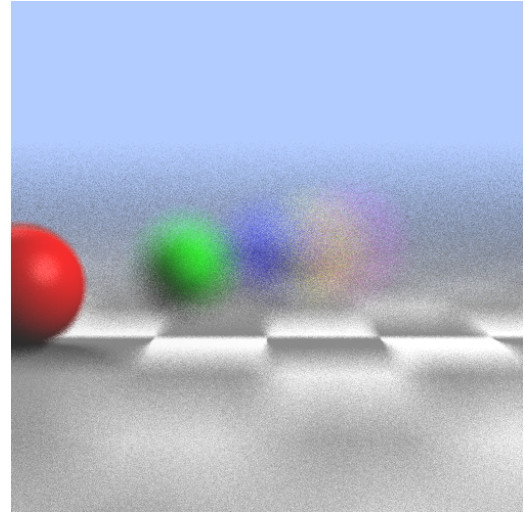
(b) Raio = 0.8, Distância Focal = 20



(c) Raio = 0.1, Distância Focal = 10



(d) Raio = 0.8, Distância Focal = 10



(e) Raio = 1.5, Distância Focal = 10.77

Figure 8: Renderizações ilustrando o efeito de Profundidade de Campo com variados parâmetros focais.

6 Conclusões

A base matemática e algorítmica do rasterizador foi corretamente implementada, como demonstram as imagens. O uso de equações quadráticas para geometrias primitivas, combinado com a aproximação numérica para superfícies implícitas e reflexões limitadas por profundidade, resultou num motor robusto capaz de gerar as cenas pretendidas com alta qualidade visual e *anti-aliasing*.