

# Interseção Raio – Cubo (Slabs)

*tcv<sub>r</sub>aytracer<sub>2</sub>026*

February 13, 2026

# Objetivo

- Explicar o raciocínio matemático por trás do método das "slabs" para interseção raio-cubo.
- Mostrar as desigualdades fundamentais e como combiná-las para obter o intervalo de parâmetros  $t$  do raio.
- Incluir um trecho de código de referência e comandos para teste em baixa resolução.

## Equações iniciais

Considere um cubo alinhado aos eixos, centrado na origem, com faces em  $x \in [-\frac{s}{2}, \frac{s}{2}]$ ,  $y \in [-\frac{s}{2}, \frac{s}{2}]$ ,  $z \in [-\frac{s}{2}, \frac{s}{2}]$ . O raio paramétrico é

$$\mathbf{p}(t) = \mathbf{o} + t \mathbf{d}, \quad t \geq 0$$

Para cada eixo temos as desigualdades (conforme a imagem fornecida):

$$x_{\min} \leq o_x + d_x t \leq x_{\max}$$

$$y_{\min} \leq o_y + d_y t \leq y_{\max}$$

$$z_{\min} \leq o_z + d_z t \leq z_{\max}$$

Rearranjando cada desigualdade isolamos  $t$  (assumindo  $d_i \neq 0$ ):

$$\frac{x_{\min} - o_x}{d_x} \leq t \leq \frac{x_{\max} - o_x}{d_x}$$

$$\frac{y_{\min} - o_y}{d_y} \leq t \leq \frac{y_{\max} - o_y}{d_y}$$

## Combinação dos intervalos (slabs)

Para cada eixo calculemos os dois valores  $t_{i1} = \frac{\min_i - o_i}{d_i}$  e  $t_{i2} = \frac{\max_i - o_i}{d_i}$  e definimos:

$$t_{i\_min} = \min(t_{i1}, t_{i2}), \quad t_{i\_max} = \max(t_{i1}, t_{i2}).$$

O intervalo de interseção global é a interseção dos três intervalos:

$$t_{enter} = \max\{t_{x\_min}, t_{y\_min}, t_{z\_min}\},$$

$$t_{exit} = \min\{t_{x\_max}, t_{y\_max}, t_{z\_max}\}.$$

Condição de interseção:  $t_{enter} \leq t_{exit}$  e existe um  $t$  válido maior que 'CastEpsilon'.

# Detalhes de implementação

- Tratar componentes de direção próximas de zero: se  $d_i \approx 0$ , o raio é paralelo ao par de planos naquela direção; o teste verifica se  $o_i$  está dentro do intervalo (slab).
- Seleção do t final: se  $t_{enter} > \varepsilon$  use  $t = t_{enter}$ ; senão, se  $t_{exit} > \varepsilon$  use  $t = t_{exit}$  (o raio começa dentro).
- Determinar a normal: depois de obter o ponto de interseção, comparar qual coordenada do ponto está mais próxima de uma face (usar tolerância).

## Trecho de código (referência)

```
def hit(self, ray):
    half = self.size / 2.0
    eps_dir = 1e-12

    # X slab
    if abs(ray.direction.x) < eps_dir:
        tx_min, tx_max = float('-inf'), float('inf')
    else:
        t1 = (-half - ray.origin.x) / ray.direction.x
        t2 = ( half - ray.origin.x) / ray.direction.x
        tx_min, tx_max = min(t1,t2), max(t1,t2)

    # Y slab (similar)
    if abs(ray.direction.y) < eps_dir:
        ty_min, ty_max = float('-inf'), float('inf')
    else:
        t1 = (-half - ray.origin.y) / ray.direction.y
        t2 = ( half - ray.origin.y) / ray.direction.y
```