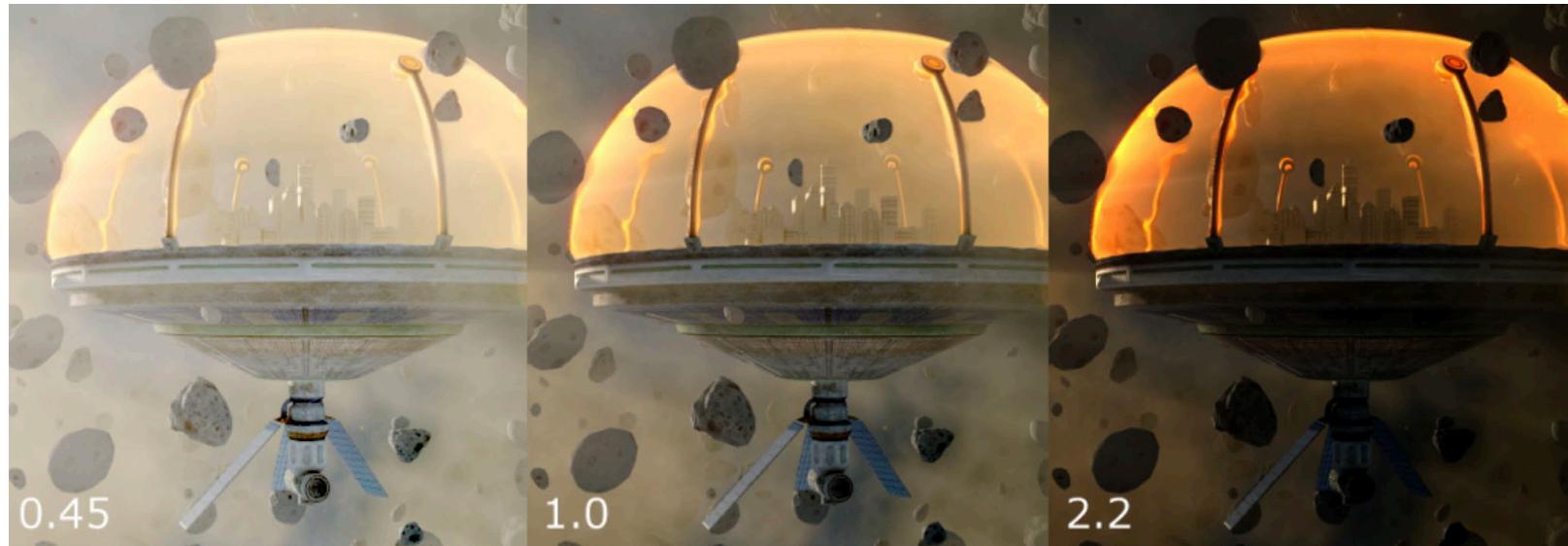


# Computação Gráfica

Raytracing 2

# Gamma space vs linear space

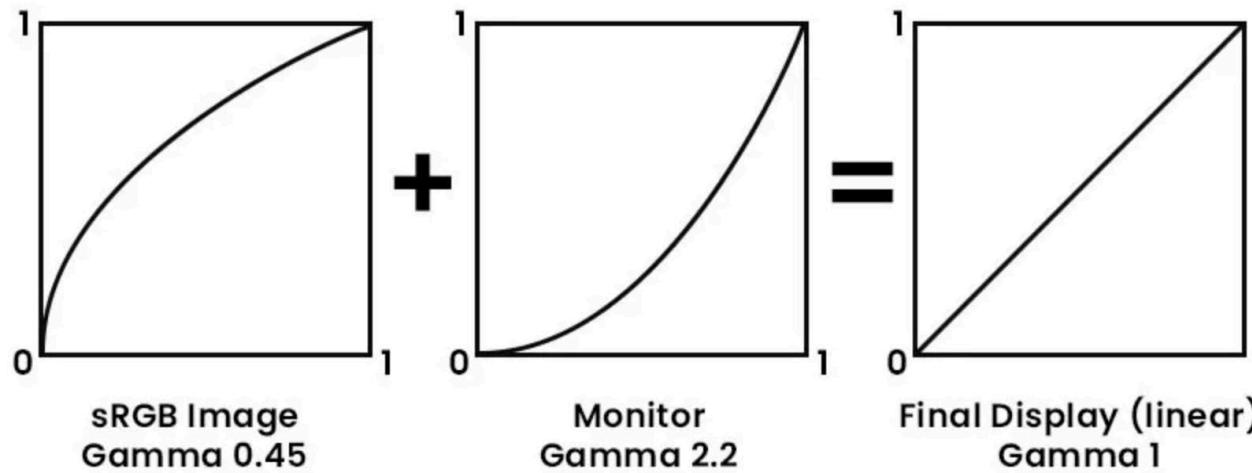
- **Armazenamento de iluminação em computadores:** Geralmente feito no formato sRGB, com correção gama.
- **Câmeras:** Capturam iluminação de forma linear, onde mais luz resulta em números proporcionalmente maiores.
- **Percepção humana:** Segue uma curva logarítmica, percebendo mais mudanças em áreas escuras e menos em áreas claras (vantagem evolutiva para detectar predadores no escuro).
- **Espaço de cores linear:** Armazena as informações de iluminação de forma mais realista, permitindo a aplicação de efeitos físicos realistas antes de converter para exibição final.



# Gamma space vs linear space

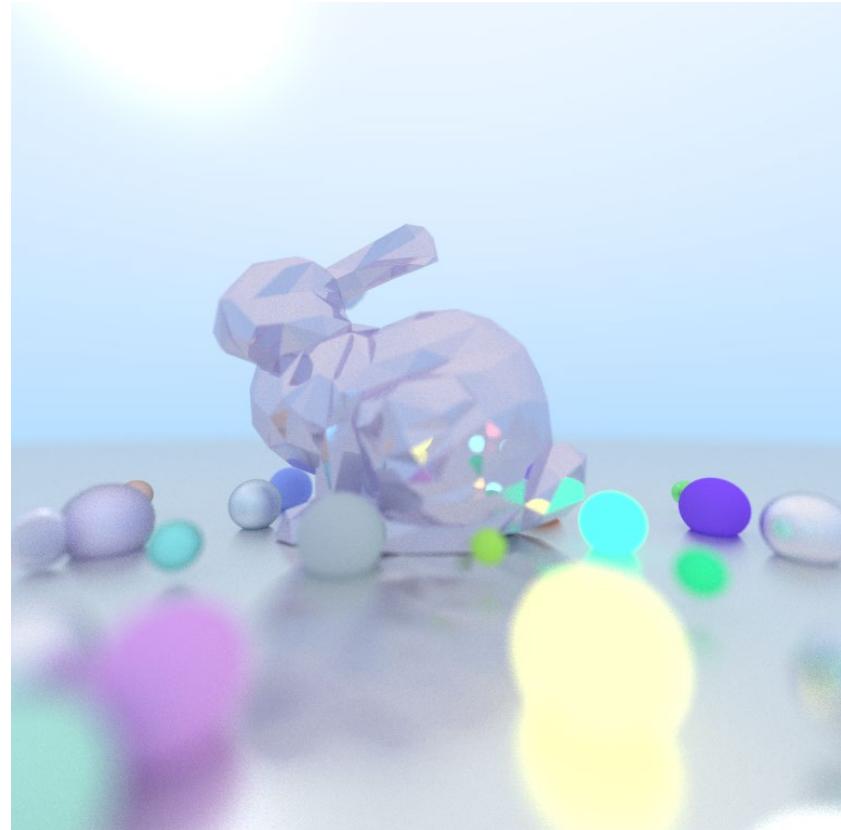
```
fn linear_to_gamma_channel(channel: f32) -> f32
{
    return pow(channel, 0.4545);
}

fn linear_to_gamma(color: vec3f) -> vec3f
{
    return vec3f(linear_to_gamma_channel(color.x),
        linear_to_gamma_channel(color.y),
        linear_to_gamma_channel(color.z));
}
```



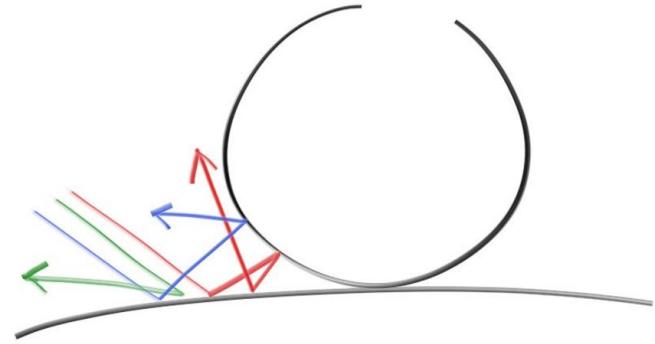
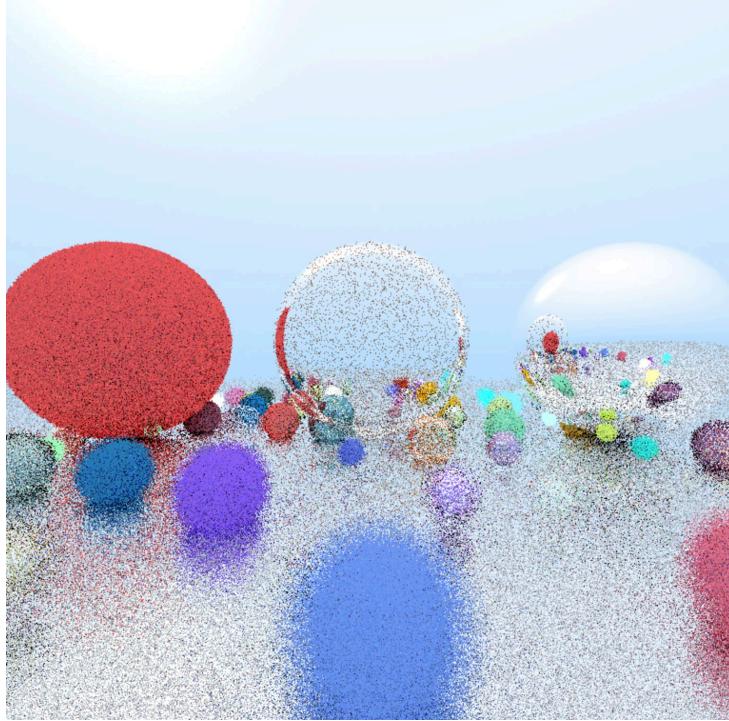
# Gamma space vs linear space

Resultado no projeto:



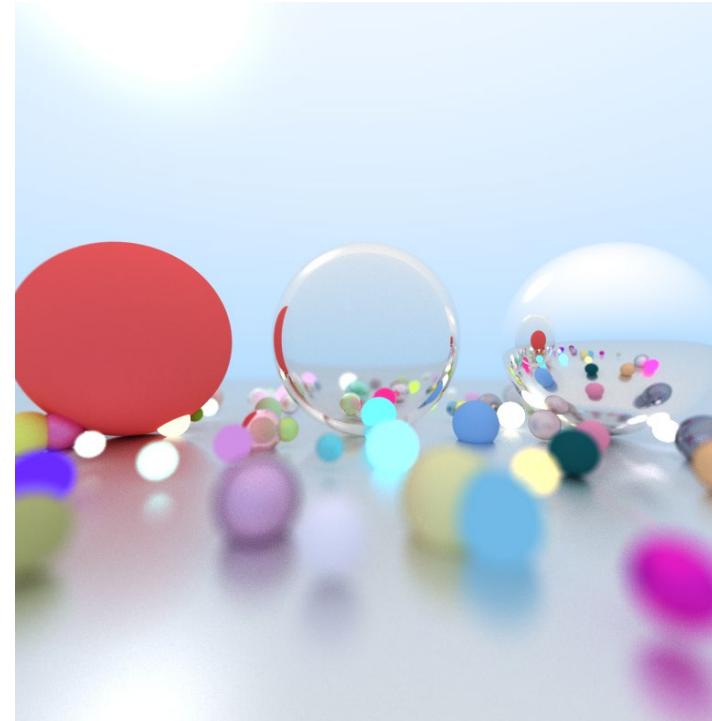
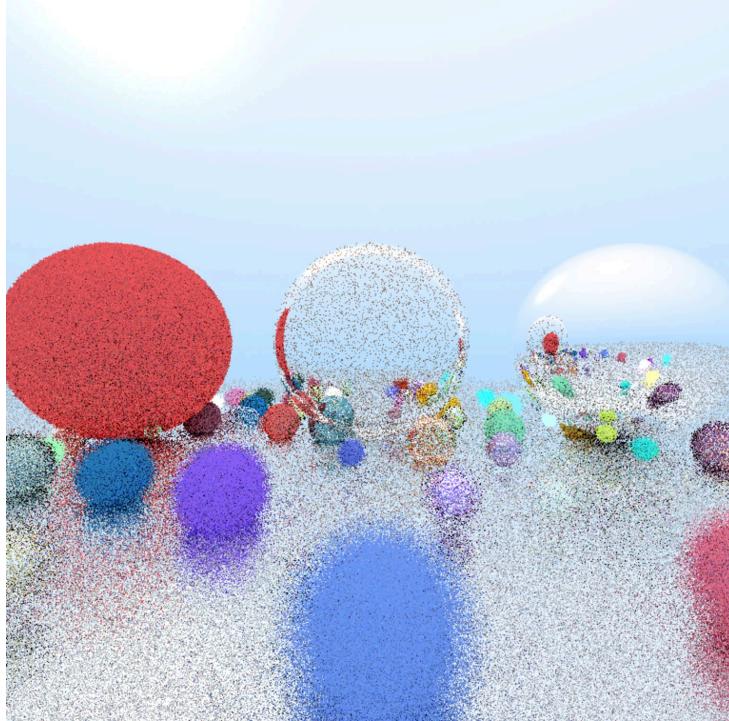
# Accumulate frames

Por que isso acontece?



# Accumulate frames

Como podemos melhorar?



# Accumulate frames

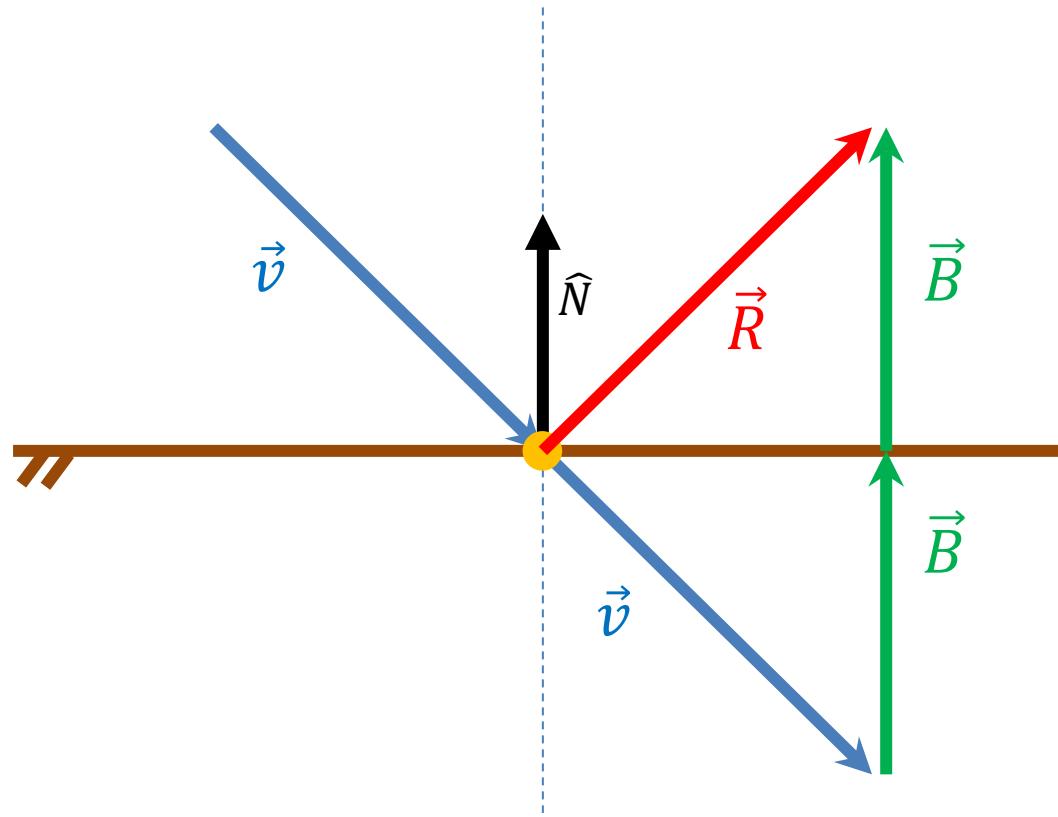
O que acontece se não tivermos "pesos" nos frames?



```
// Solucao:  
// pra que serve?  
var should_accumulate = uniforms[3];  
  
var accumulated_color = cor_anterior *  
    should_accumulate + color_saida;  
  
rtb[map_fb] = accumulated_color;  
frameb[map_fb] = accumulated_color / peso;
```

# Reflexão em Materiais Espelhados

Como calcular o vetor de reflexão?



# Reflexão em Materiais Espelhados

Para calcular o vetor de reflexão (vermelho) somaremos o vetor de entrada ( $\mathbf{v}$ ) com duas vezes  $\mathbf{B}$ .

$\mathbf{B}$  tem a direção e sentido de  $\mathbf{N}$  com um comprimento de  $|\mathbf{v} \cdot \mathbf{N}|$ .

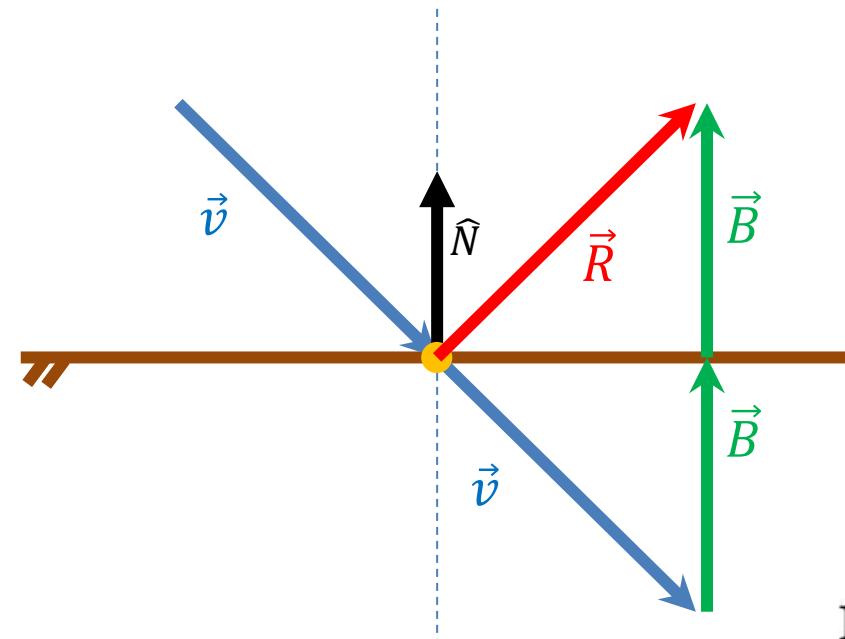
$$\mathbf{R} = \mathbf{v} - 2 \cdot (\mathbf{v} \cdot \mathbf{n}) \cdot \mathbf{n}$$

Em WGLS: `reflect()`

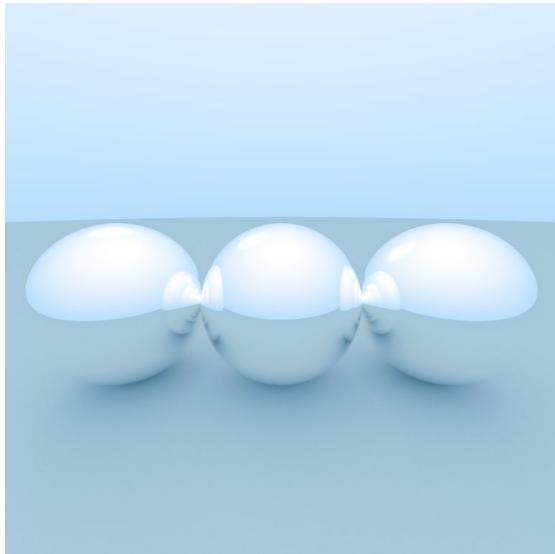
A variável `smoothness` controla a componente metálica no projeto.

0 - Lambertiano

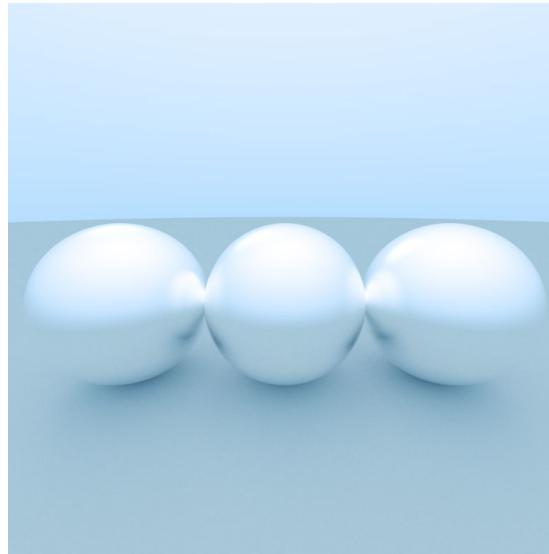
1 - Metálico



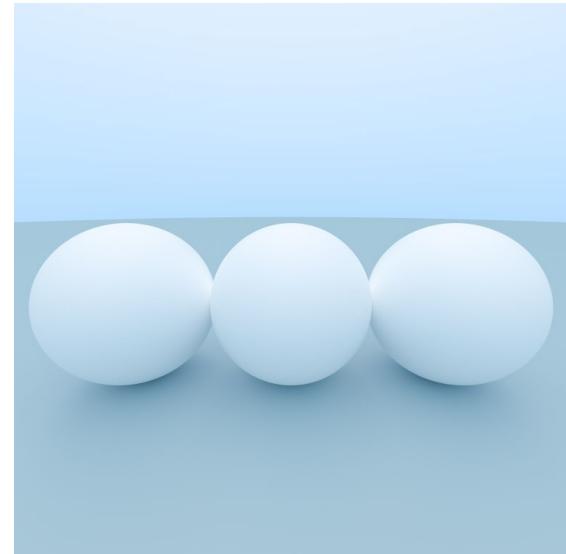
# Reflexão em Materiais Espelhados - Resultados



**Smoothness 1.0**

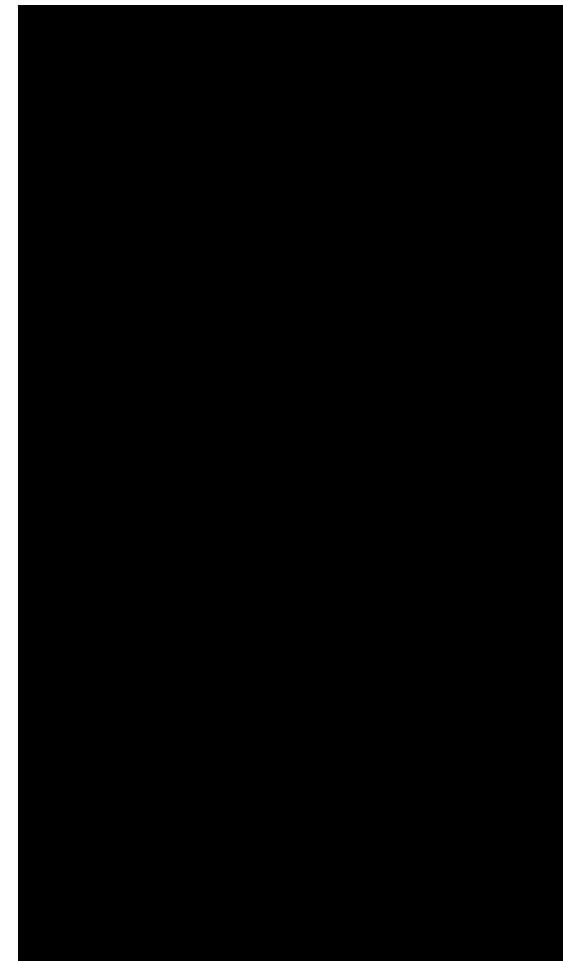
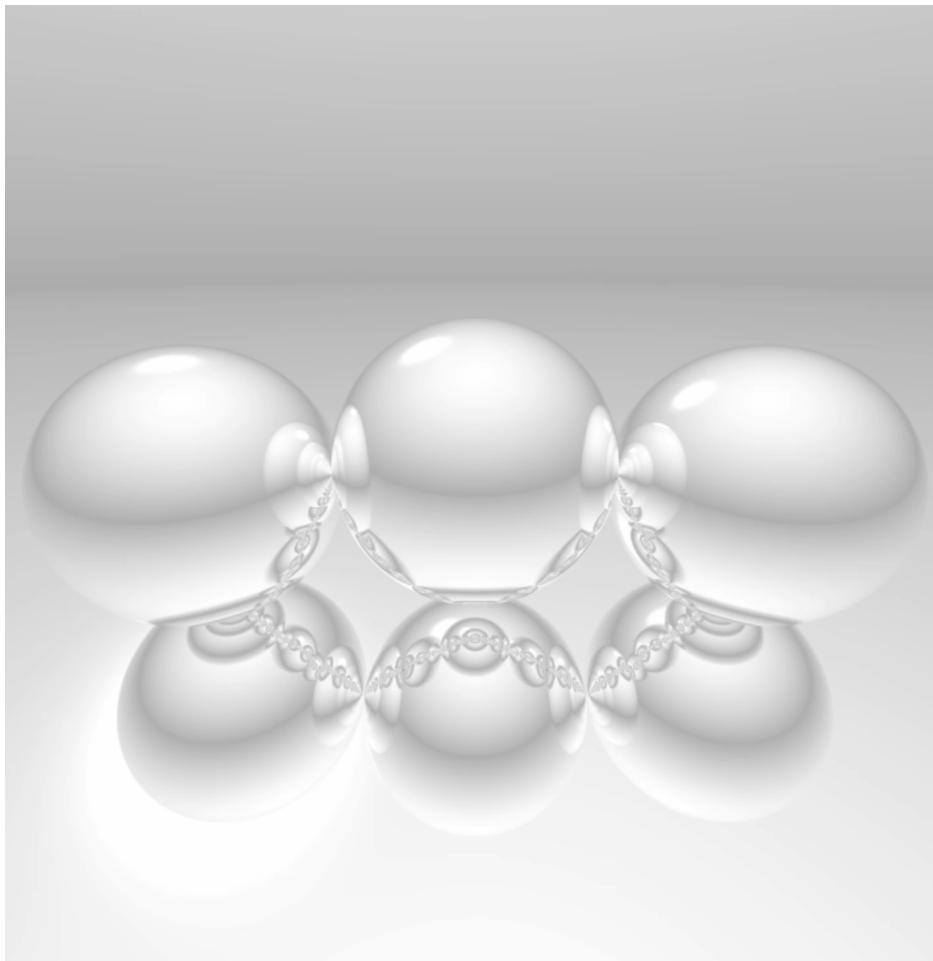


**Smoothness 0.8**



**Smoothness 0.2**

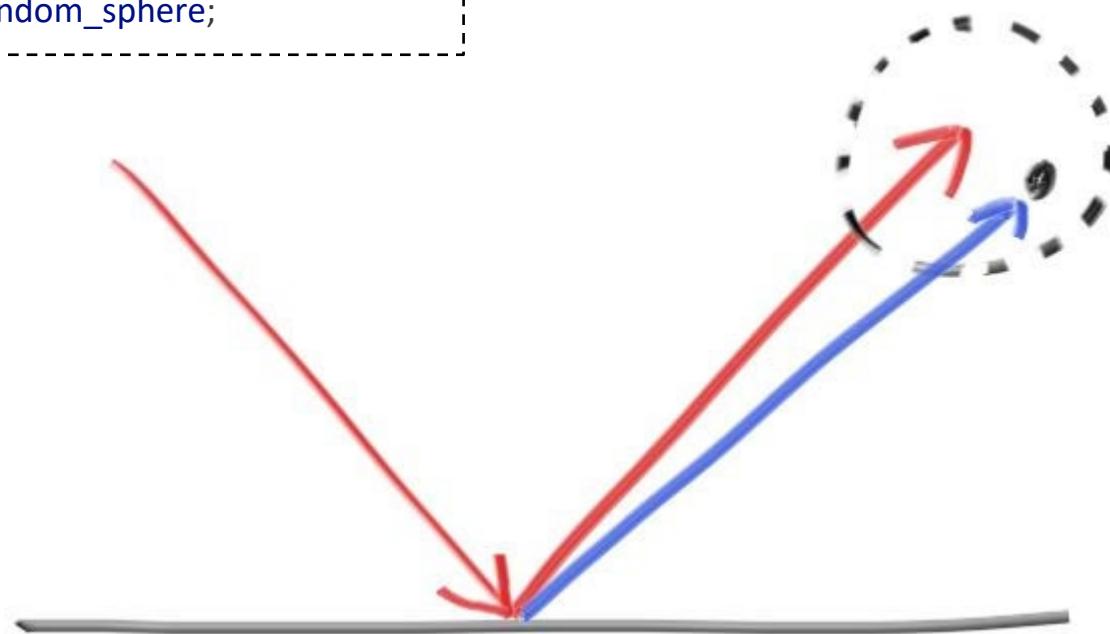
# Reflexão em Materiais Espelhados – Mundo real



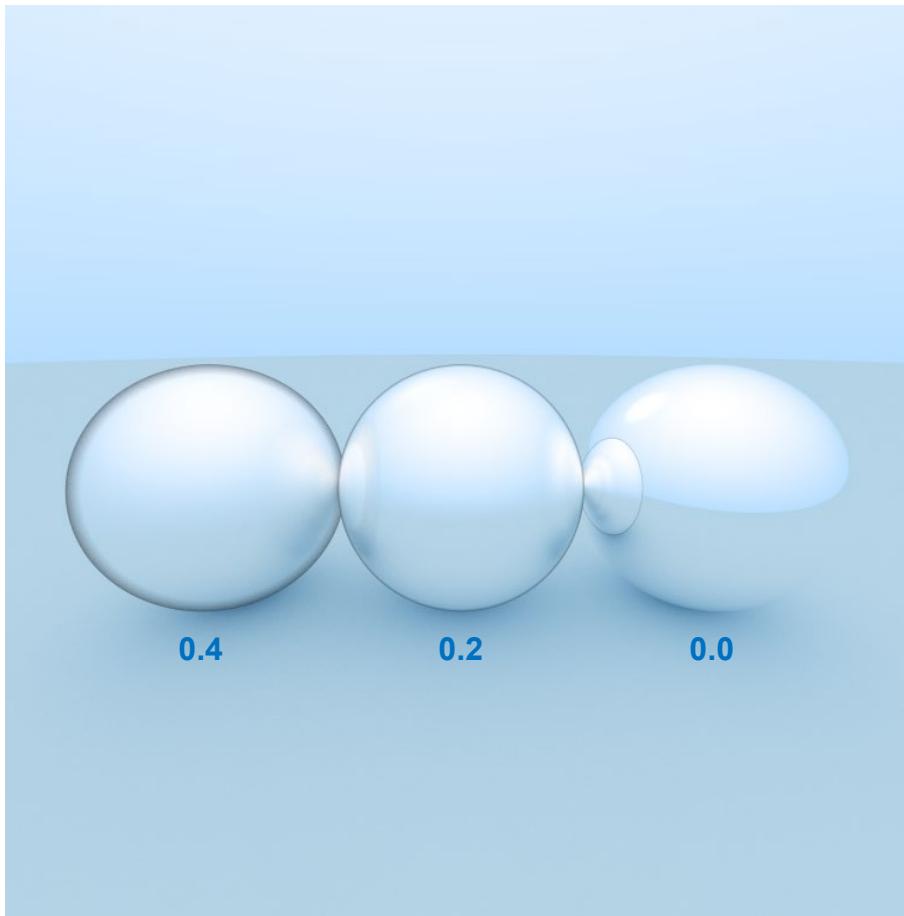
# Reflexão Fuzzy

Podemos colocar alguma aleatoriedade na reflexão, deslocando o destino do raio pelo deslocamento unitário radial.

```
reflect + fuzz * random_sphere;
```



# Reflexão Fuzzy - Resultados



# Tomate



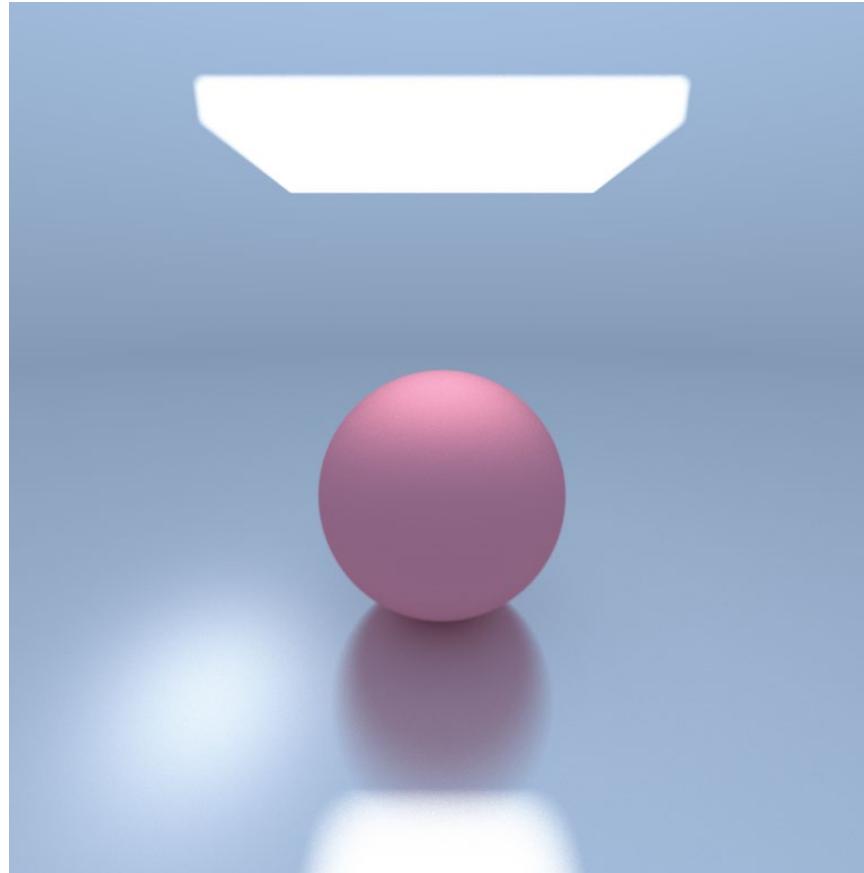
Como o tomate reflete a luz?



# Tomate

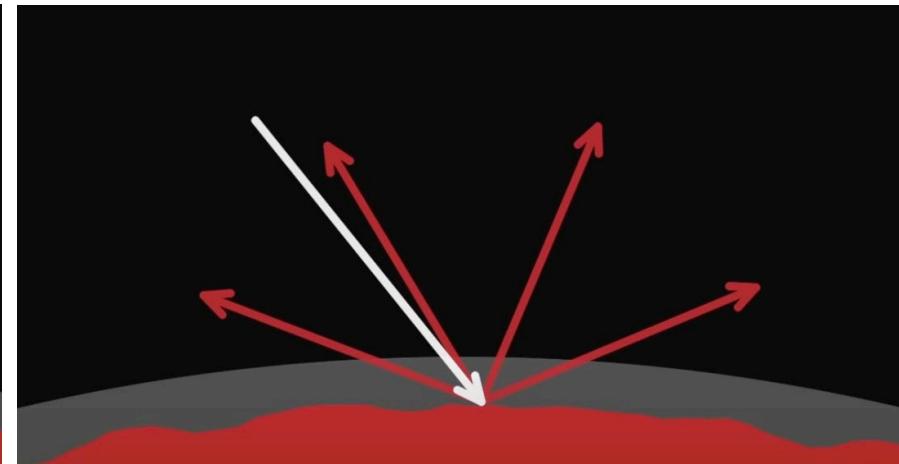
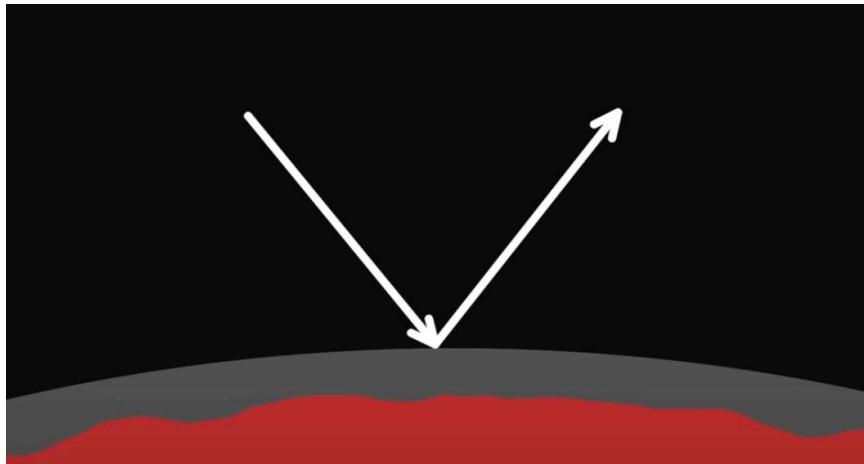


Smoothness mais baixo?



# Reflexão especular

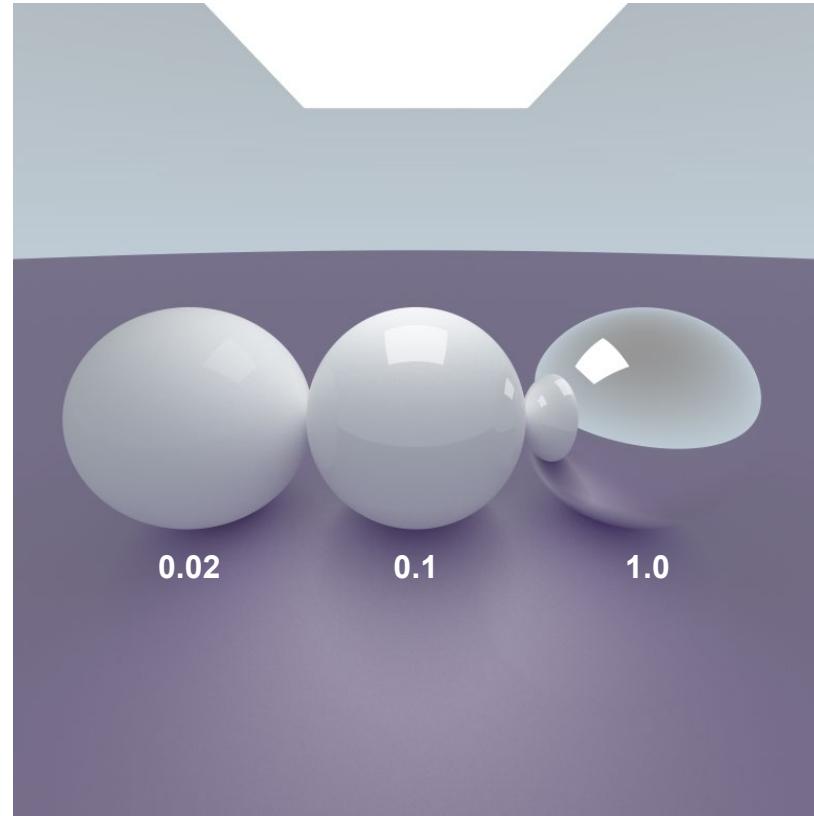
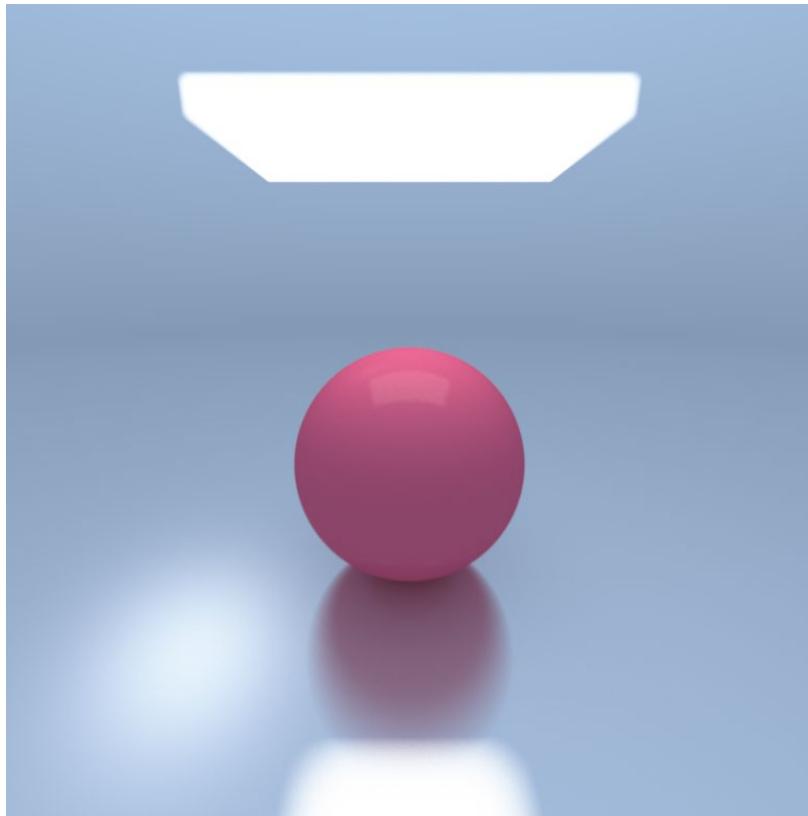
Alguns raios são refletidos e outros se espalham igual um material lambertiano



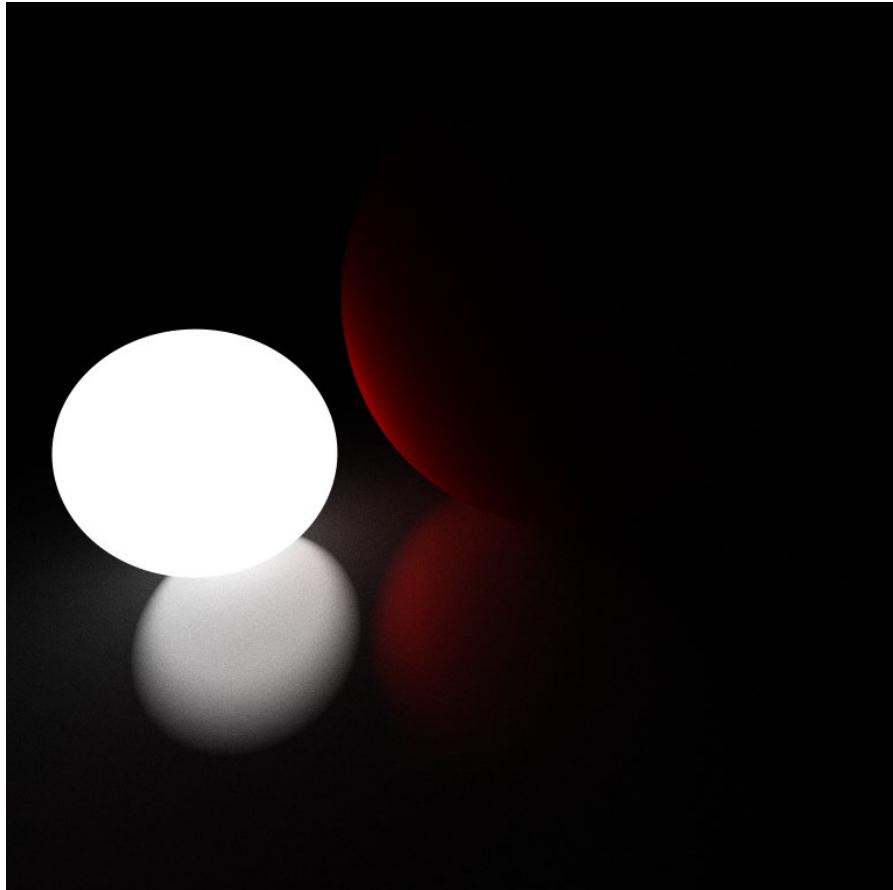
Como fazer esse efeito?

Specular probability > random()

# Reflexão especular - Resultados



# Materiais emissivos



**Cor multiplicada pela intensidade**

# Materiais Dielétricos

Materiais transparentes como água, vidro e diamantes são dielétricos.

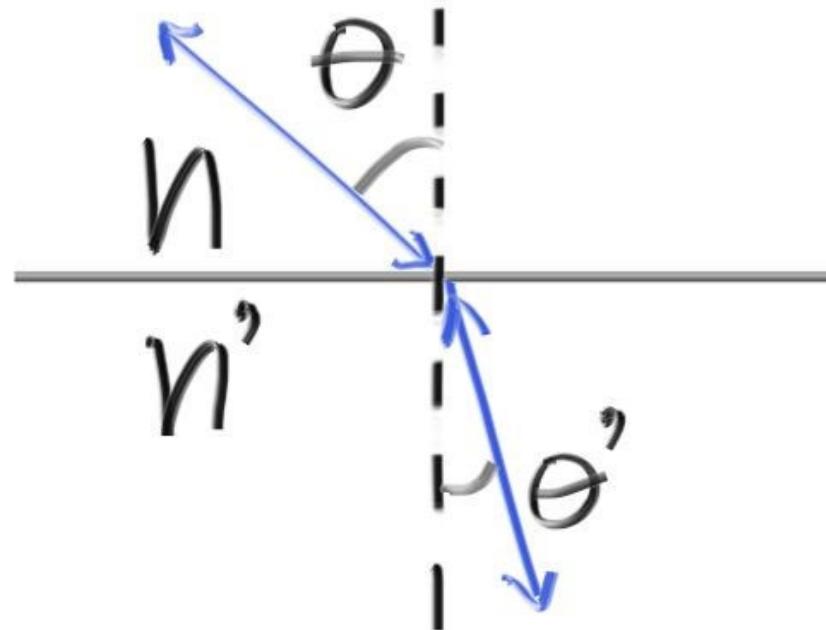
Quando um raio de luz atinge esses materiais, ele se divide em um raio refletido e um refratado (transmitido).

Vamos lidar com isso escolhendo aleatoriamente entre reflexão ou refração, e apenas gerando um raio de espalhamento por interação.

# Lei de Snell-Descartes

A refração é descrita por:

$$\eta \cdot \sin \theta = \eta' \cdot \sin \theta'$$



# Determinando o raio refratado

O raio de entrada ( $\mathbf{R}$ ) vai ser refratado em um raio ( $\mathbf{R}'$ ).

Para efetuar os cálculos vamos dividir o raio refratado nos componentes perpendicular e paralelos a normal.

$$\mathbf{R}' = \mathbf{R}'_{\perp} + \mathbf{R}'_{\parallel}$$

A solução para o  $\mathbf{R}$  perpendicular e paralelo é:

$$\mathbf{R}'_{\perp} = \frac{\eta}{\eta'} (\mathbf{R} + \cos \theta \mathbf{n})$$

$$\mathbf{R}'_{\parallel} = -\sqrt{1 - |\mathbf{R}'_{\perp}|^2} \cdot \mathbf{n}$$

# Determinando o raio refratado

Para calcular o valor do cosseno de  $\theta$  podemos usar o produto escalar dos vetores. Ou seja:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| |\mathbf{b}| \cos \theta$$

Restringindo **a** e **b** a vetores unitários:

$$\mathbf{a} \cdot \mathbf{b} = \cos \theta$$

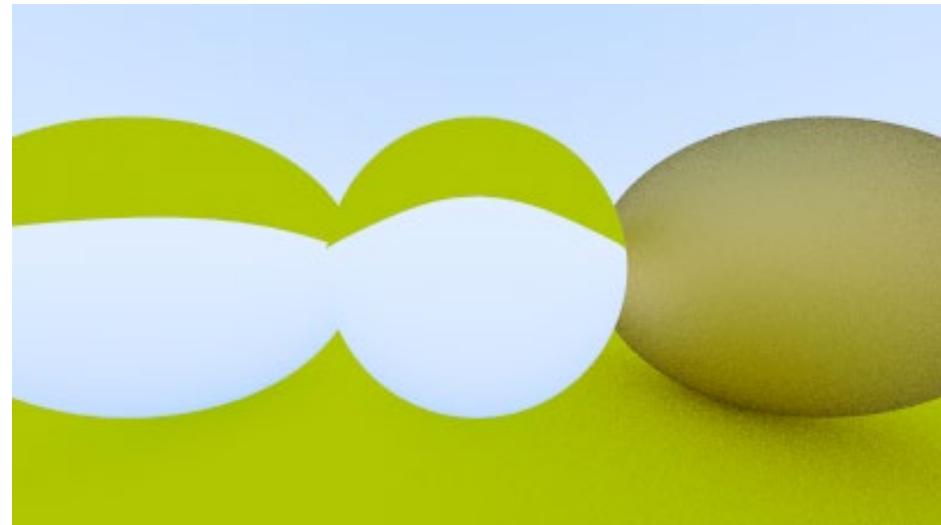
Dessa forma a componente perpendicular do raio refratado fica:

$$\mathbf{R}'_{\perp} = \frac{\eta}{\eta'} (\mathbf{R} + (-\mathbf{R} \cdot \mathbf{n})\mathbf{n})$$

# Resultado das Esferas de Vidro



Uma esfera de vidro tem dois efeitos principais sobre a cena. Primeiro, ela atua como uma lente olho de peixe e tenta mostrar uma visão de 180 graus. Segundo, ela inverte a imagem de cabeça para baixo.

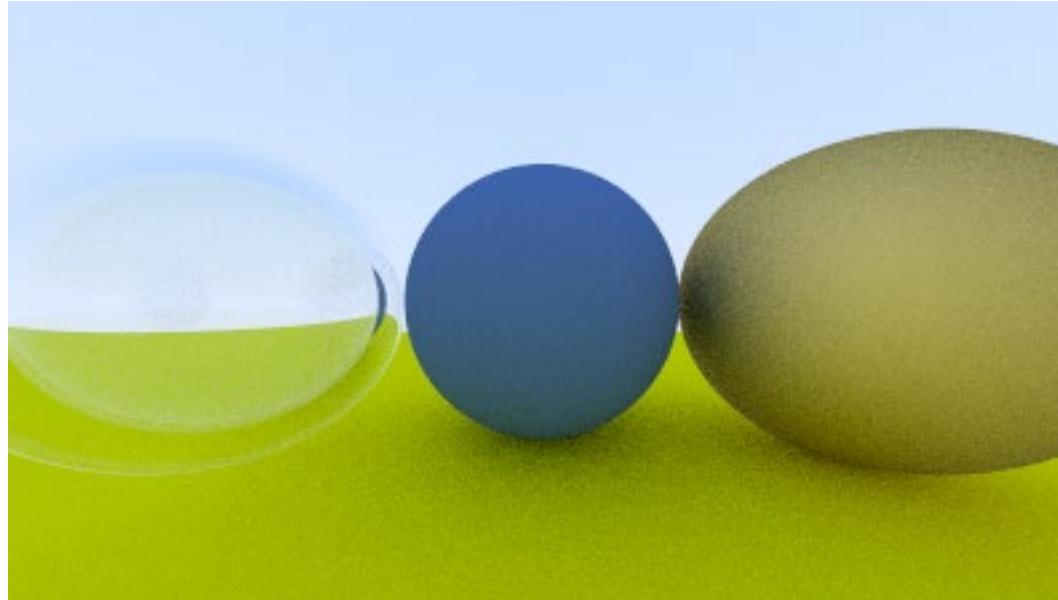


Testando o algoritmo no Ray Tracer:

Referências: <https://www.hackingphotography.com/crystal-ball-photography-7-tips-get-started/>  
<https://raytracing.github.io/books/RayTracingInOneWeekend.html>

# Resultado com novos materiais

Um truque interessante pode ser criar uma esfera oca (com ar dentro). Isso cria uma borda que fica deformada e depois a imagem central menos deformada. Para isso sempre é preciso analisar a mudança do índice de refração.



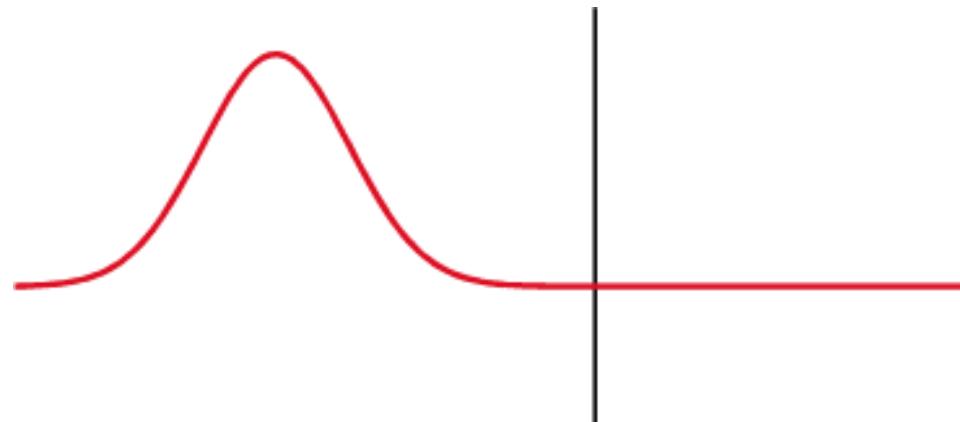
# Aproximação de Schlick

Permite calcular o coeficiente de reflexão:

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos \theta)^5$$

onde:

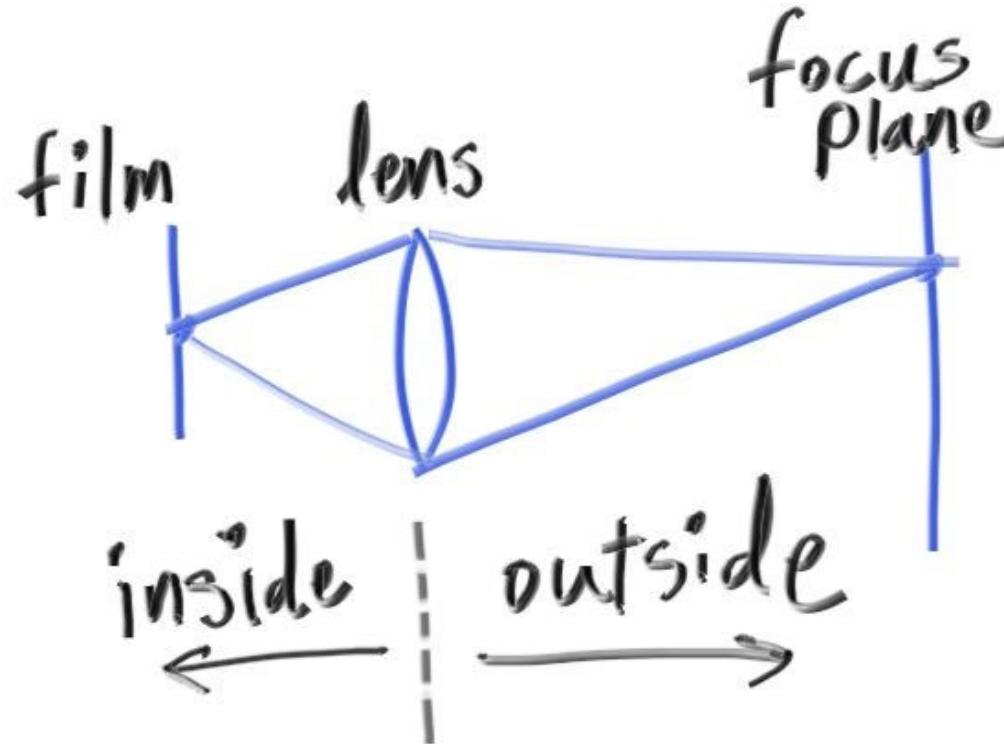
$$R_0 = \left( \frac{n_1 - n_2}{n_1 + n_2} \right)^2$$



# Desfocando - Blur

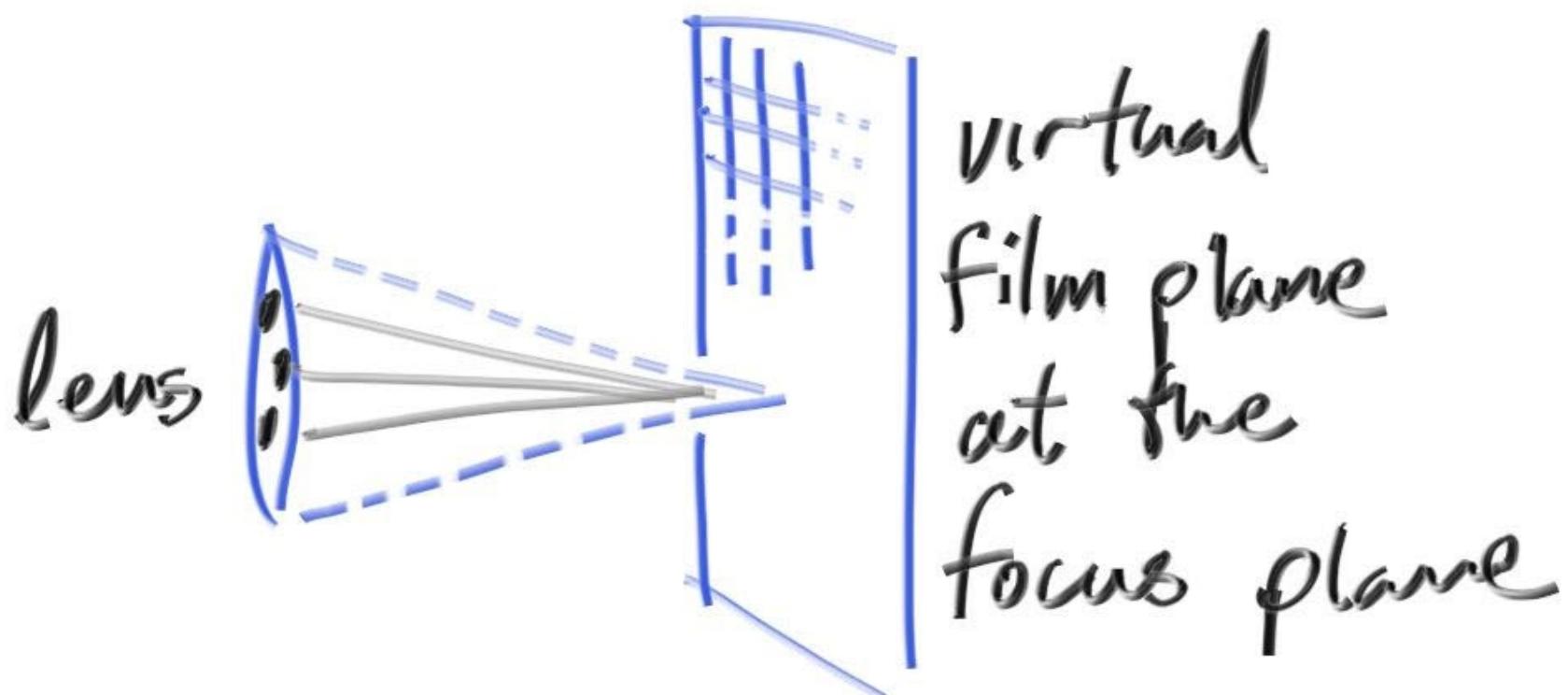
Simula um efeito de profundidade de campo (depth of field)

Podemos criar o efeito por um conjunto sensor, lente e abertura.



# Desfocando - Blur

O truque a ser usado é lançar os raios como se fosse da lente.



# Projeto Raytracer

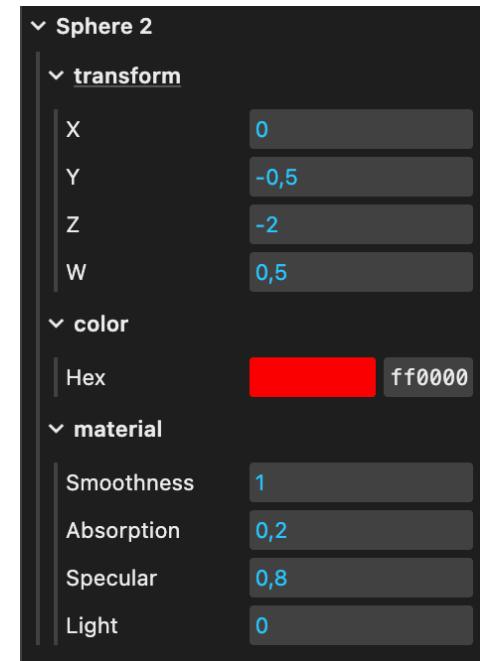
## Rubrica/Dicas/Código

<https://github.com/Gustavobb/raytracing-wgsl-template> - Fazer um fork

## Orientações:

O projeto já define esferas com:

- Posição [x, y, z e w]
- Cor [RGB em Hexadecimal]
- Material
  - Smoothness
  - Absorption
  - Specular
  - Light



# Projeto Raytracer

## **Smoothness (que define a aparência metálica):**

Índice que define o quanto se mistura de características difusas (0) ou metálicas (1), ou senão puramente dielétricas (<0). Use algum recurso de interpolação linear para o novo raio que será criado. Chamaremos esse raio de ***metal\_reflect***.

## **Absorption (que define o quão escuro e borrado):**

Índice que define proporção de raios absorvidos pelo objeto (1) e que continuam de alguma forma: refletem, refratam, espalham (0), em materiais metálicos também define o fuzz (fosco) do objeto.

```
reflect + fuzz * random_sphere;
```

# Projeto Raytracer

## **Specular (combinação de tipos de reflexão):**

Índice que define a probabilidade de um raio específico refletir de forma especular (1) ou difusa (0), ou seja, se 0.5 (50%), a probabilidade é de metade dos raios refletirem de forma especular (*metal\_reflect*) e a outra metade refletir de forma bem espalhada numa distribuição lambertiana.

## **Light (define quanto emite de luz o objeto):**

Valor que define o quanto um material emite luz própria, com sua tonalidade definida pela cor do material. Esse valor tem um faixa de valores infinita, podendo ser mais que um.

# Projeto Raytracer

## Importante:

**Objetos podem ter várias propriedades, você vai precisar combinar elas.**

### Smoothness:

- Se  $> 0$ , componente com fator metálico (metallic)
- Se  $= 0$ , componente é puramente difuso (diffuse)
- Se  $< 0$ , o material é dielétrico (dielectric)

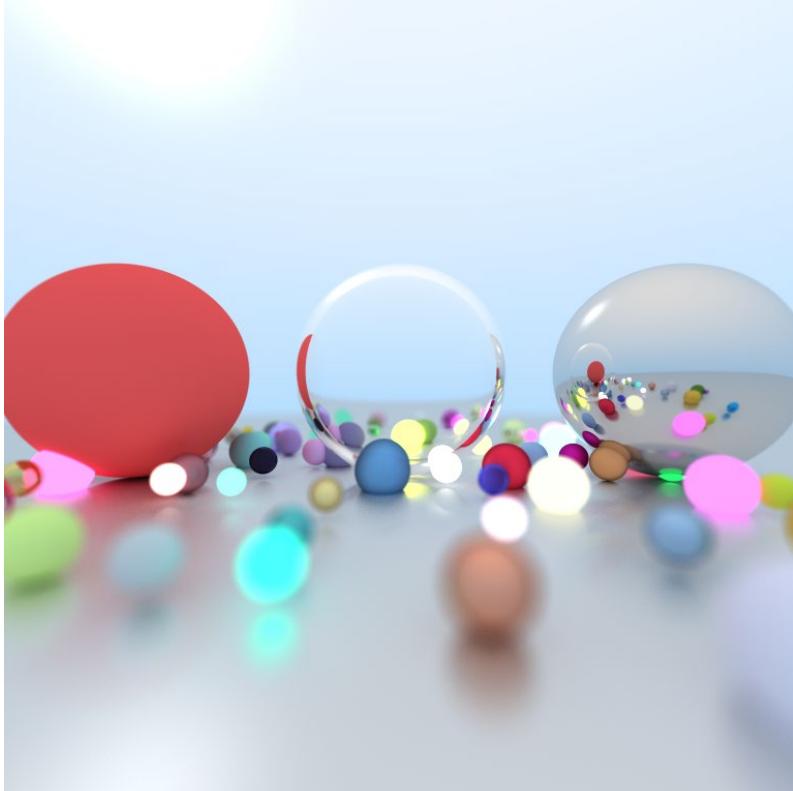
### Reflexão Metálica Fuzzy

- Se  $\text{Absorption} > 0$ , materiais metálico ficam foscos

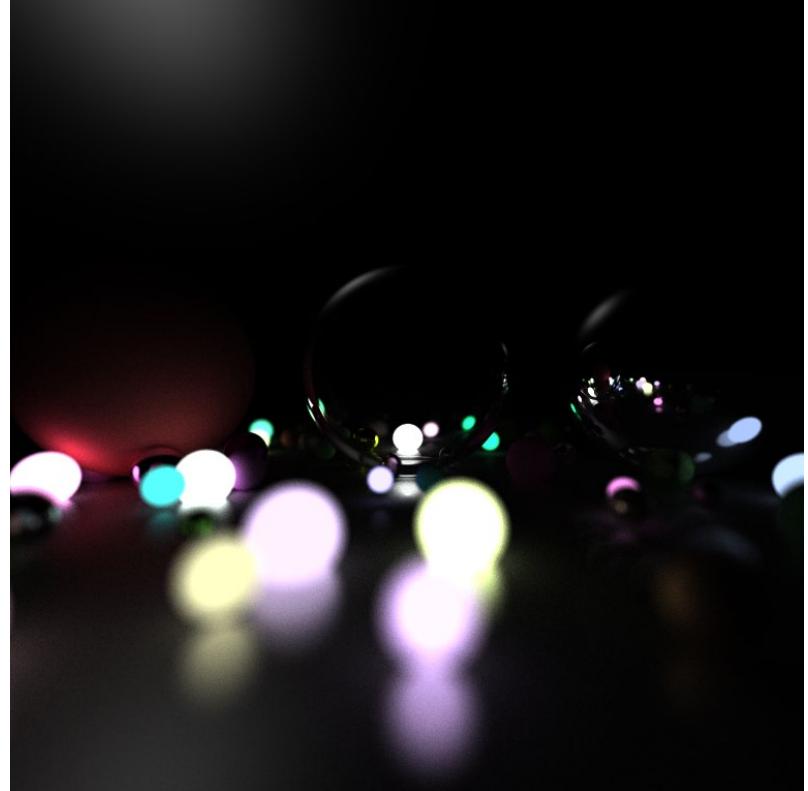
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Spheres



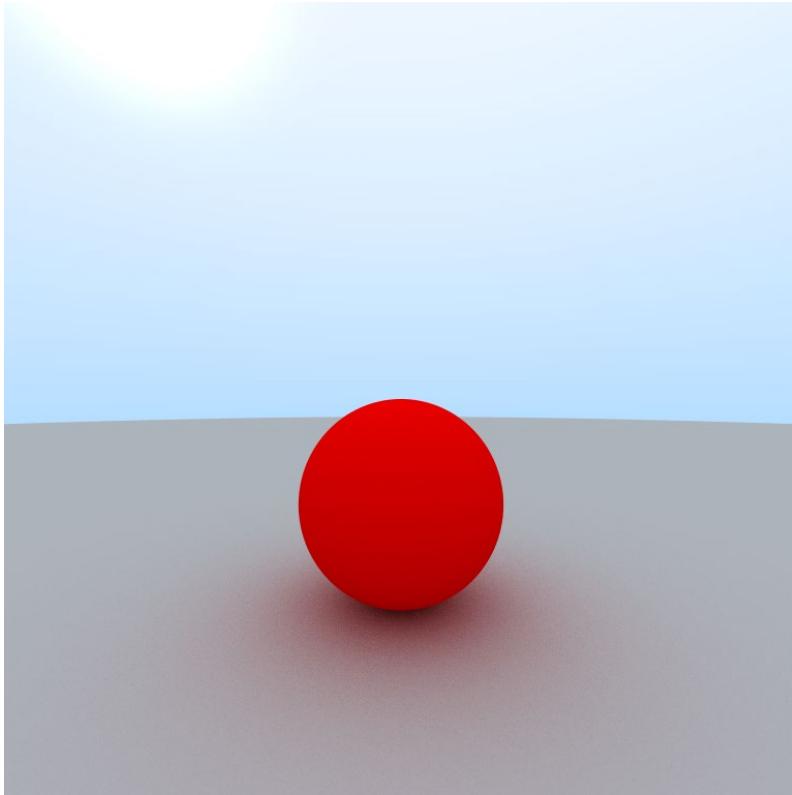
Night



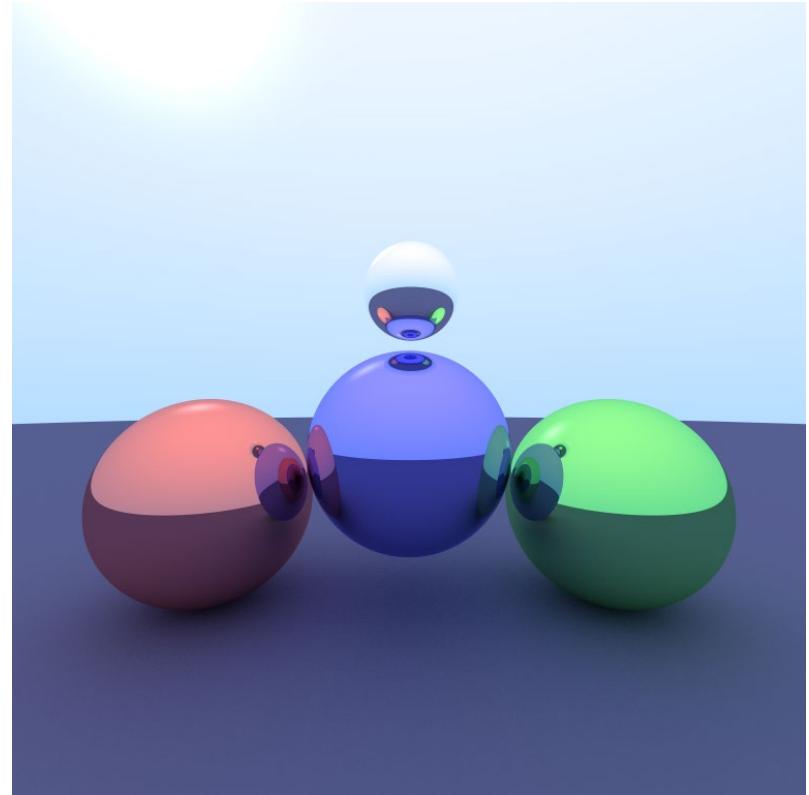
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Basic



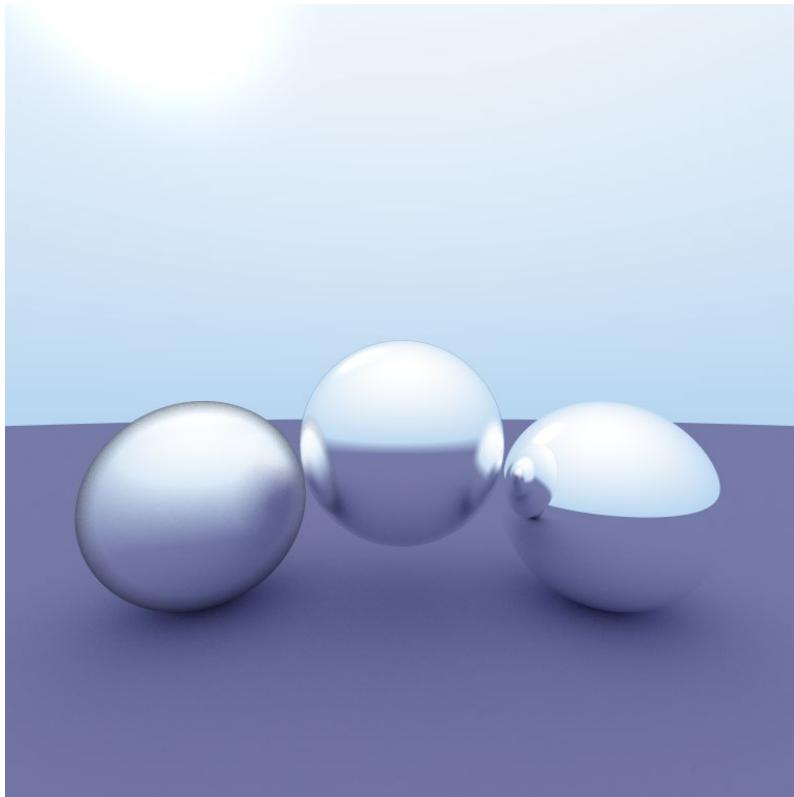
Metal



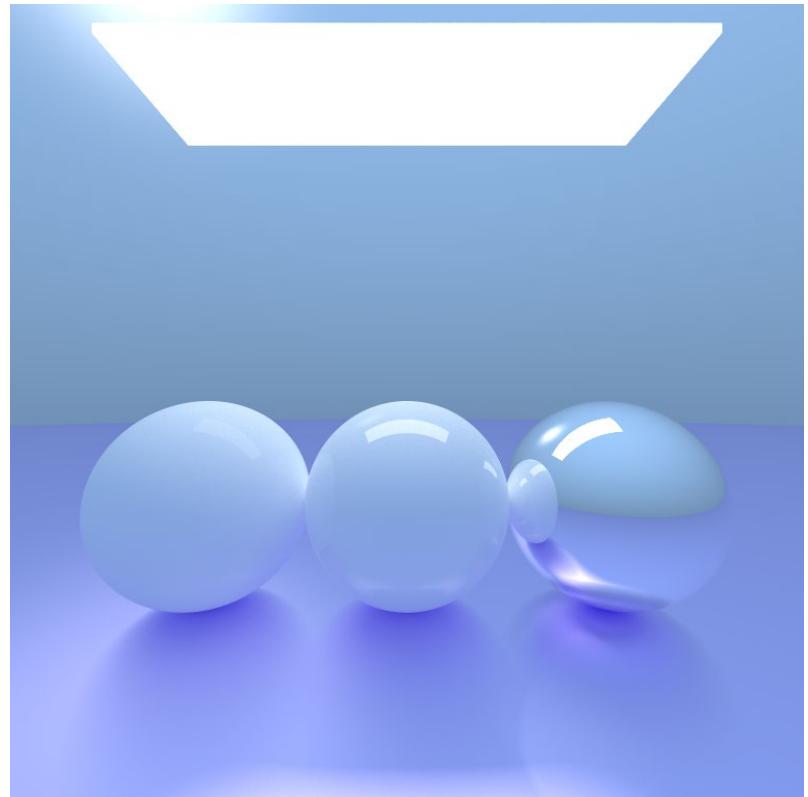
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Fuzz



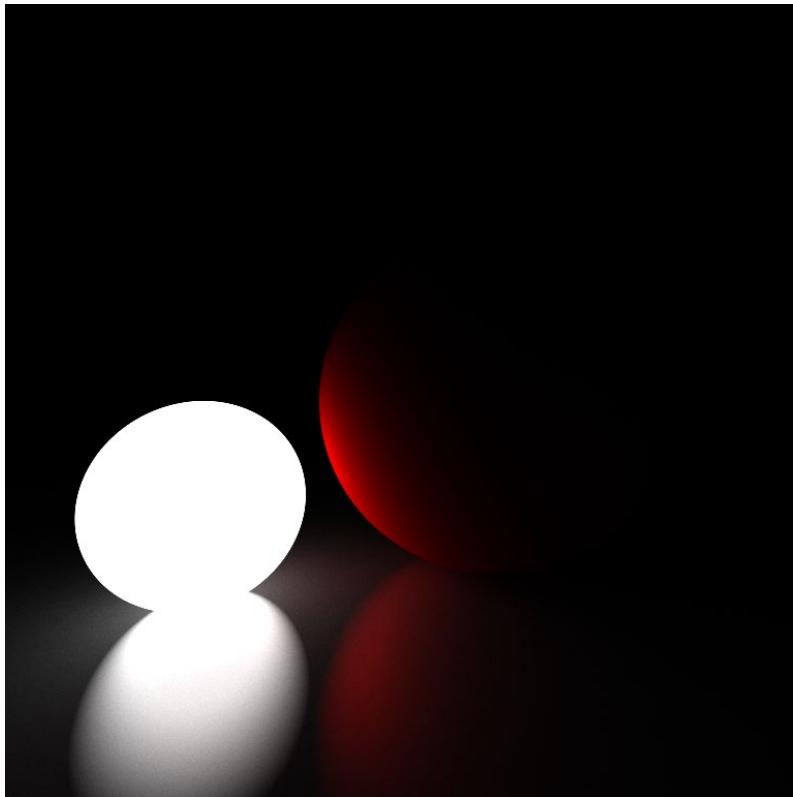
Specular



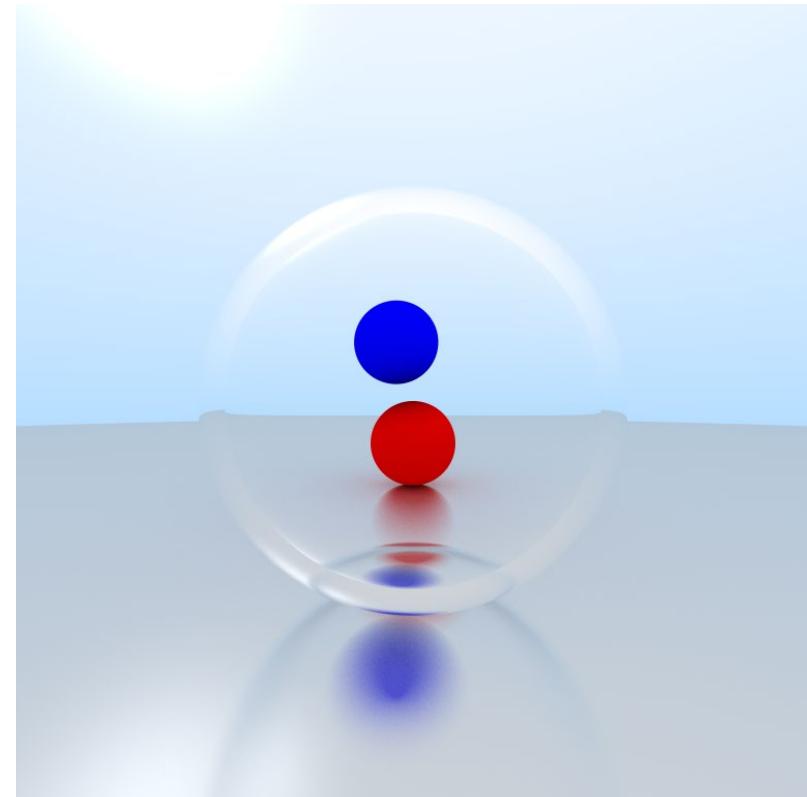
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Emissive



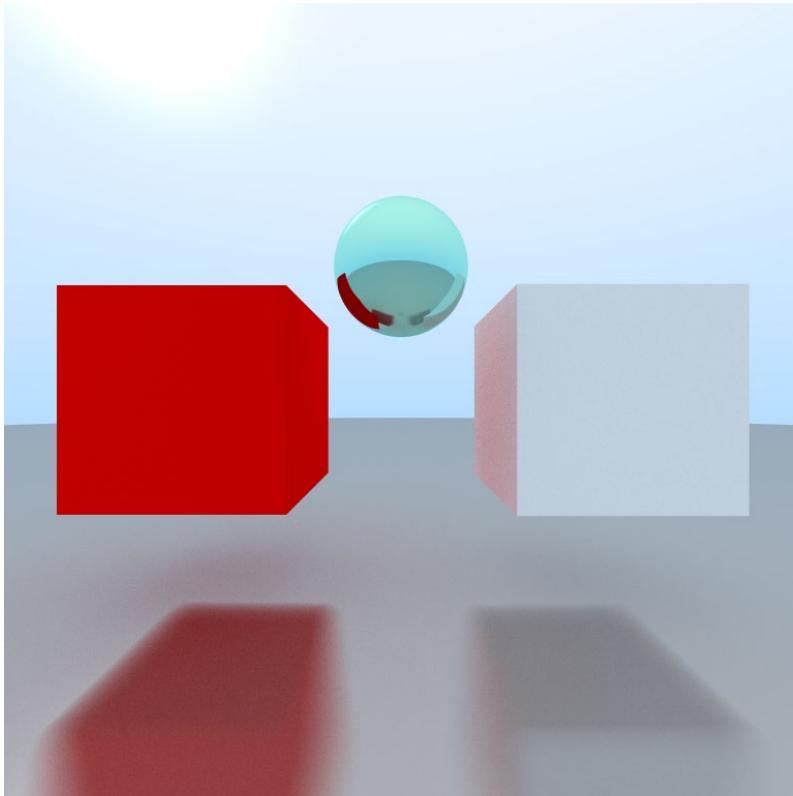
Dielectric



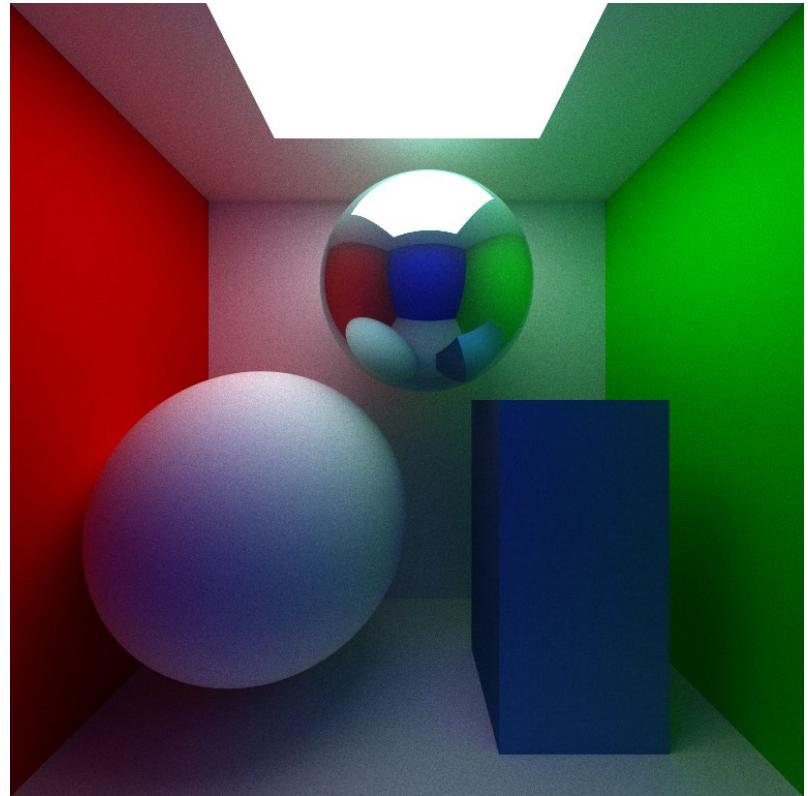
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Cubes



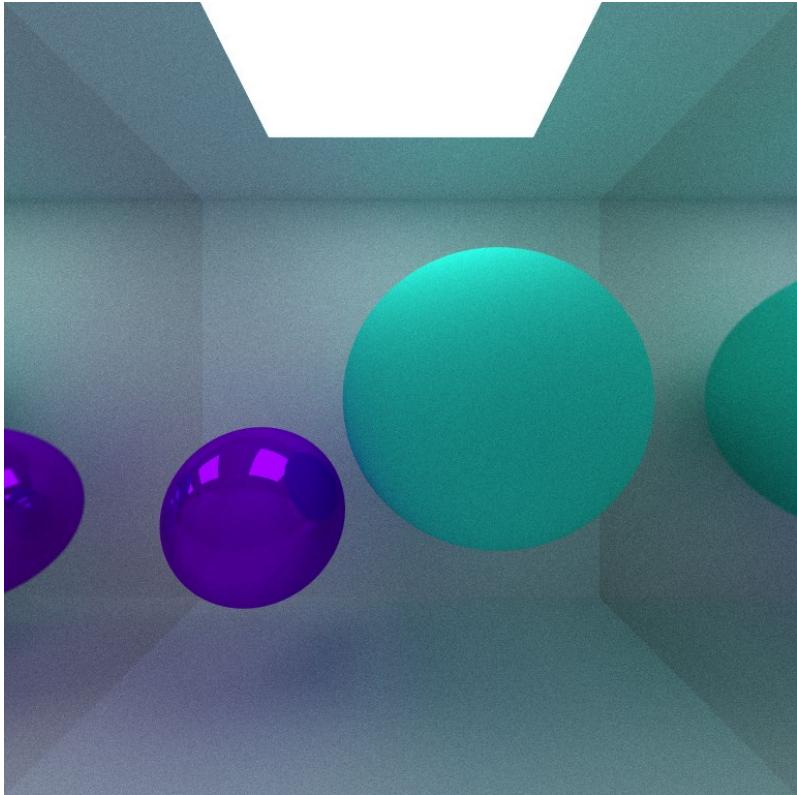
Cornell



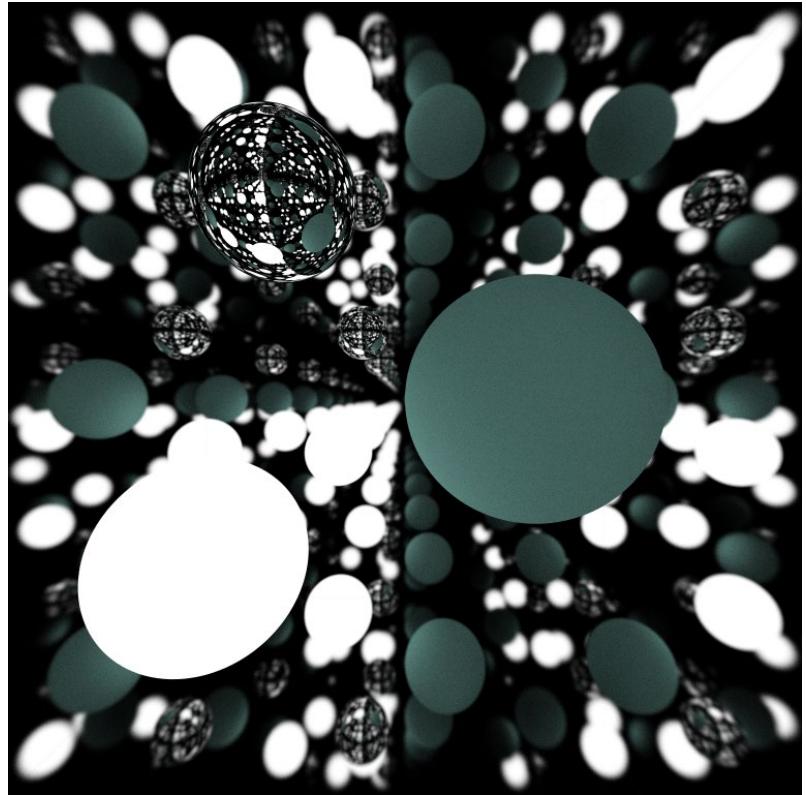
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Mirror



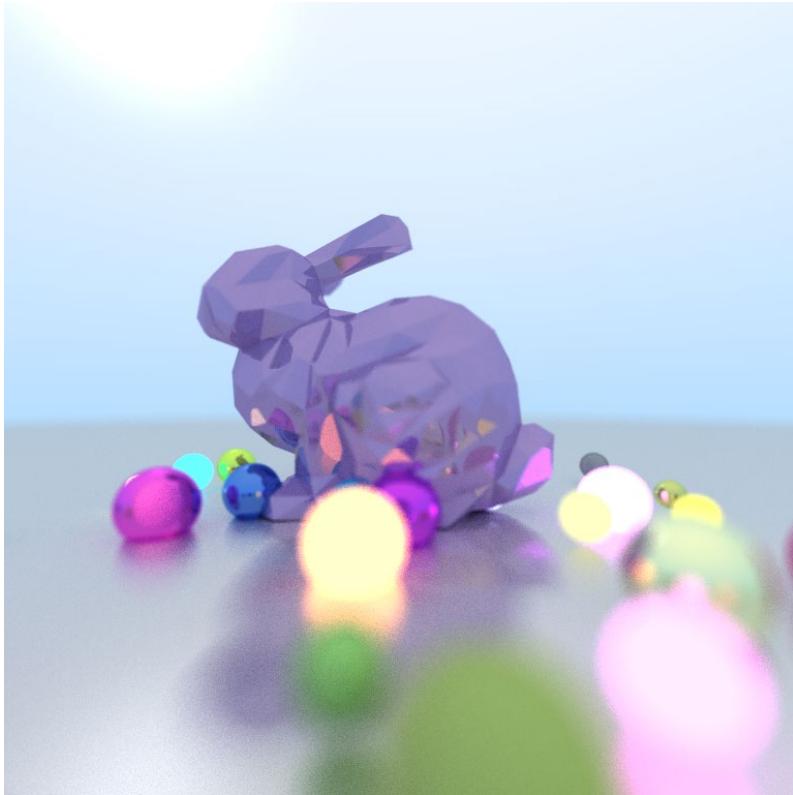
Infinite



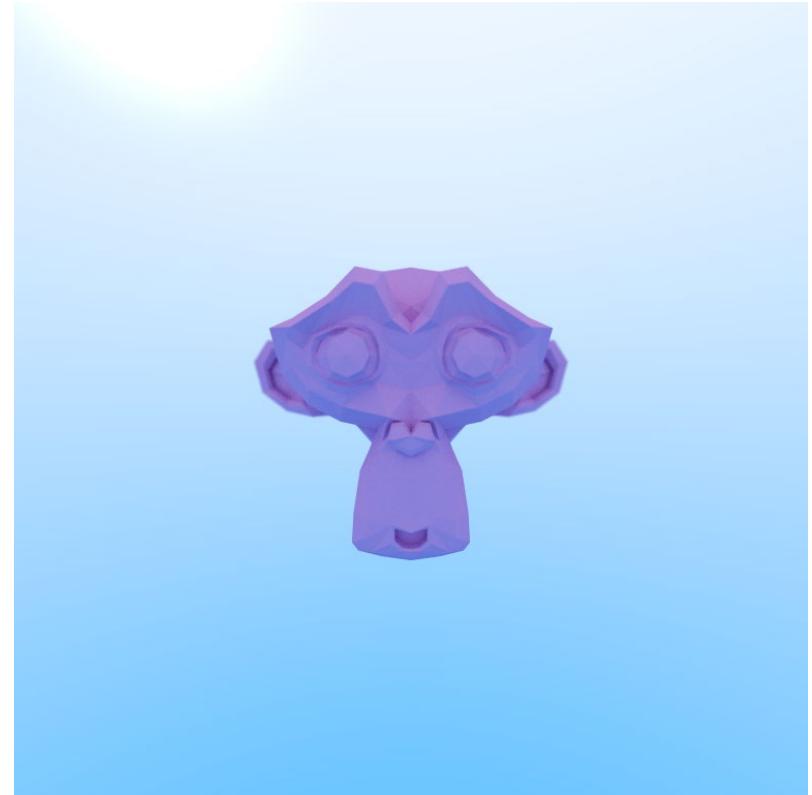
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Bunny



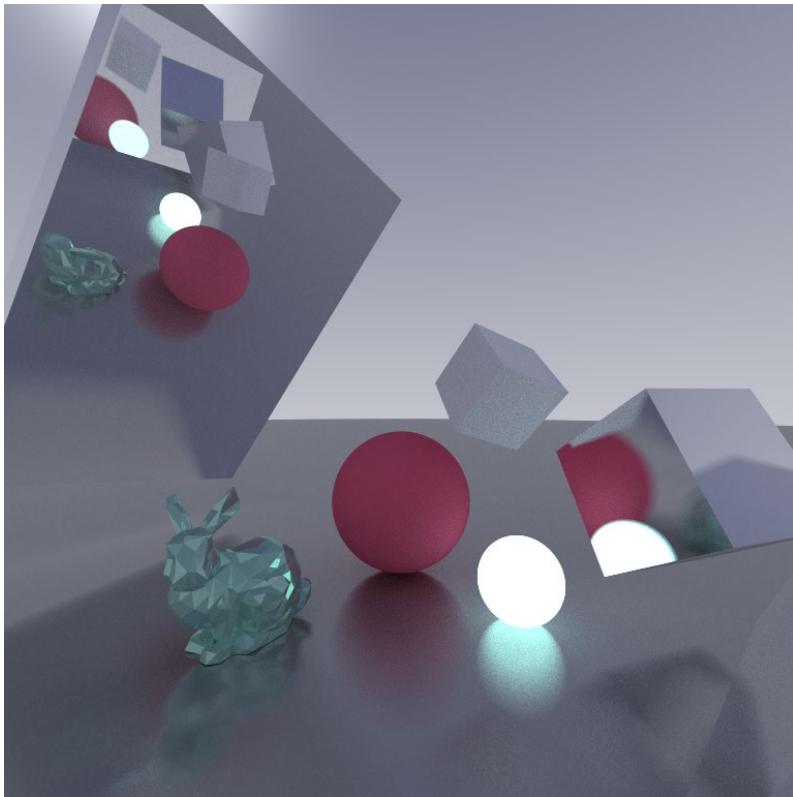
Suzanne



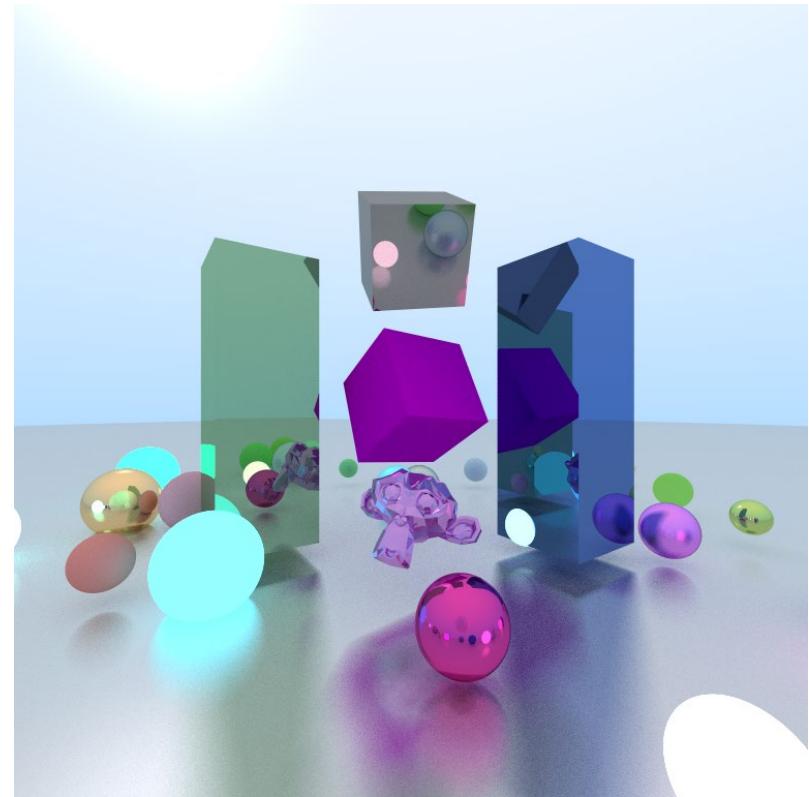
# Gabarito do Projeto

<https://gubebra.itch.io/raytracing>

Rotation



Everything



# Computação Gráfica

Luciano Soares  
[`<lpsoares@insper.edu.br>`](mailto:<lpsoares@insper.edu.br>)

Fabio Orfali  
[`<fabioo1@insper.edu.br>`](mailto:<fabioo1@insper.edu.br>)

Gustavo Braga  
[`<gustavobb1@insper.edu.br>`](mailto:<gustavobb1@insper.edu.br>)