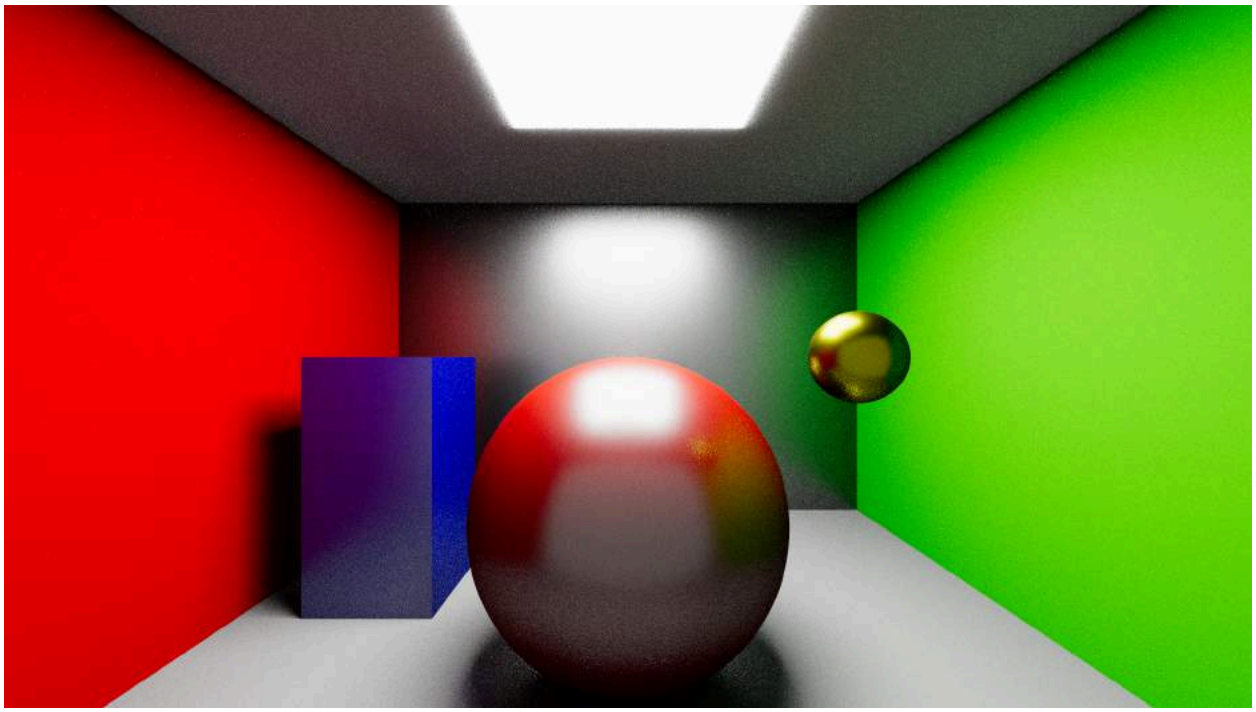
Projeto 2

Algoritmo de Traçado de Caminhos

INF2608 - Fundamentos da Computação Gráfica

Luís Fernando Garcia Jales



Descrição do aplicativo:

A atividade foi feita usando, como base, a tecnologia **WebGL2** para renderização com o auxílio da GPU. Para isso, o aplicativo usa o framework **Next.js**, que permite o uso de bibliotecas de *front-end*, sendo as principais o **React** e **Tailwind**. O framework facilita consideravelmente na criação de uma página **HTML** em vários aspectos como, por exemplo, realizar *data fetching* de arquivos de texto contendo os *shaders*, obter a posição do mouse, renderizar múltiplas vezes por segundo, aplicar estilo **CSS** e controlar o elemento **Canvas**, que é por onde é renderizado usando o contexto de **WebGL2**.

Dentro do contexto de **WebGL2**, o aplicativo faz uso de um *vertex shader*, dois *fragment shaders* e um *frame buffer*. A ideia principal é preencher a tela inteira com um retângulo para traçar caminhos usando um dos *fragment shaders*, acumular o resultado usando o *frame buffer* em duas texturas auxiliares, e finalizar renderizando o resultado acumulado aplicando *tone mapping*.

O *vertex shader* tem, como único objetivo, preencher um retângulo na tela inteira e passar coordenadas de textura para o *fragment shader*.
O *fragment shader* de traçado de caminhos cria a cena inteira, traça caminhos usando várias técnicas, e acumula o resultado no *frame buffer*.
O *fragment shader* de exibição exibe a textura do *frame buffer* e aplica um *tone mapping* no espaço **ACEScg** usando uma curva de autoria minha baseada em EDOs de segunda ordem lineares homogêneas.

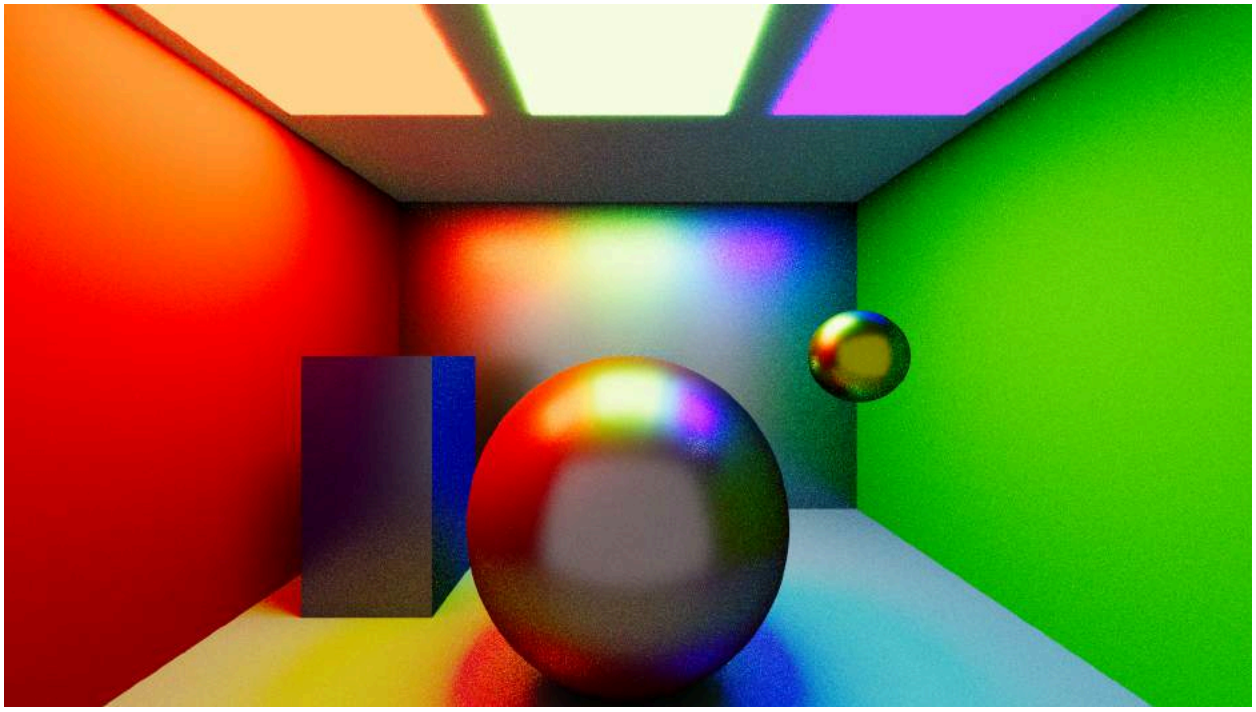
Técnicas adotadas:

A estrutura do código do algoritmo de traçado de raios feito em **C++** foi adaptado em **GLSL**. Como o **GLSL** é bastante semelhante à linguagem **C**, boa parte das estruturas, dos cálculo de interseção, das funções de BRDF e de outros códigos puderam ser aproveitadas com pouca alteração, portanto, boa parte do que foi adotado também se manteve.
Começando pela aplicação básica:

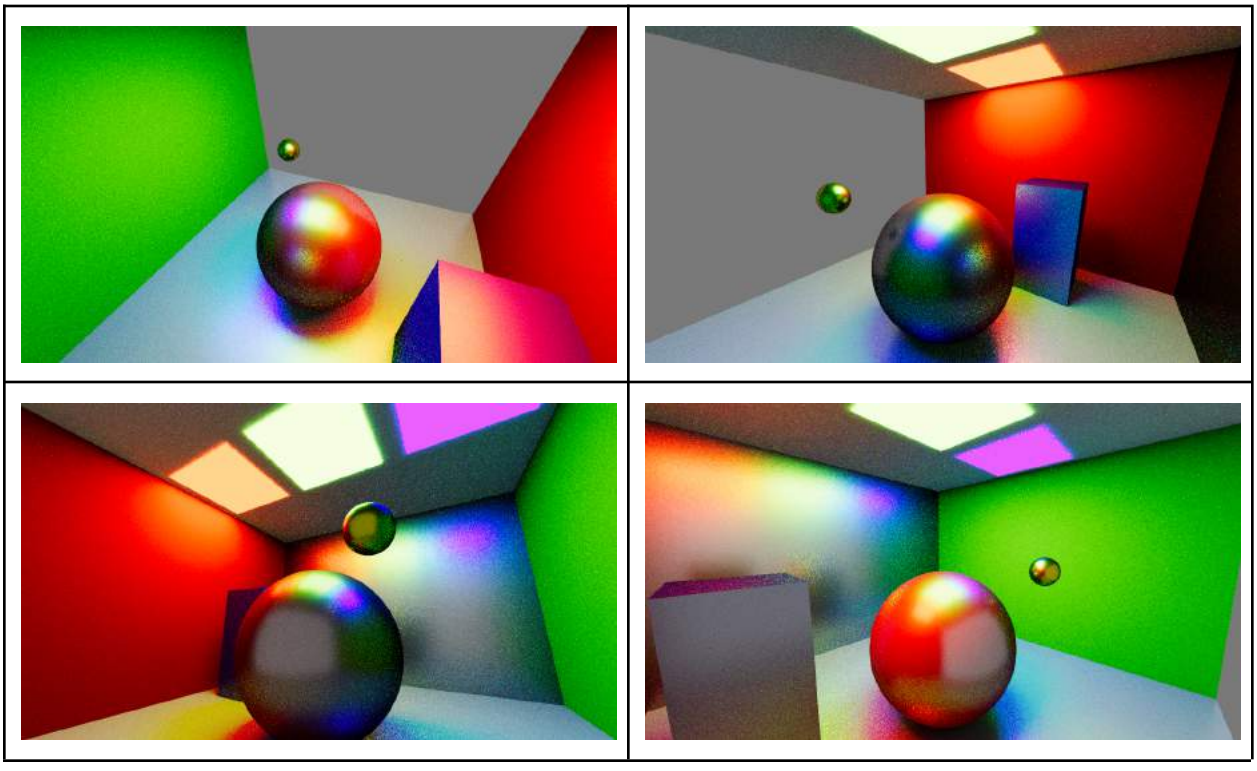
- A cena contém esferas, caixas e planos.
- A cena é iluminada por três fontes de luz retangulares, uma vermelha (esquerda), uma verde(centro) e uma azul(direita).
- A cena contém objetos com materiais cujo BRDF usa o modelo Cook-Torrance, onde as paredes da esquerda (vermelha), da direita (verde), e o teto são perfeitamente difusos, ou seja, BRDF = $\frac{cor}{\pi}$. O chão, a caixa azul (esquerda), a bola vermelha (centro) são dielétricos, ou seja, possuem uma componente difusa colorida e outra especular branca. A parede do fundo e a bola amarela (direita) são metais, ou seja, possuem apenas a componente especular colorida.
- A cena acumula caminhos a cada quadro de renderização, portanto pode-se dizer que temos múltiplos caminhos por pixel.
- Os caminhos possuem profundidade 4 + Roleta Russa de profundidade esperada 4.

Abaixo, encontram-se imagens de vários ângulos do cenário descrito acima:

Imagem principal:



Ângulos diferentes:

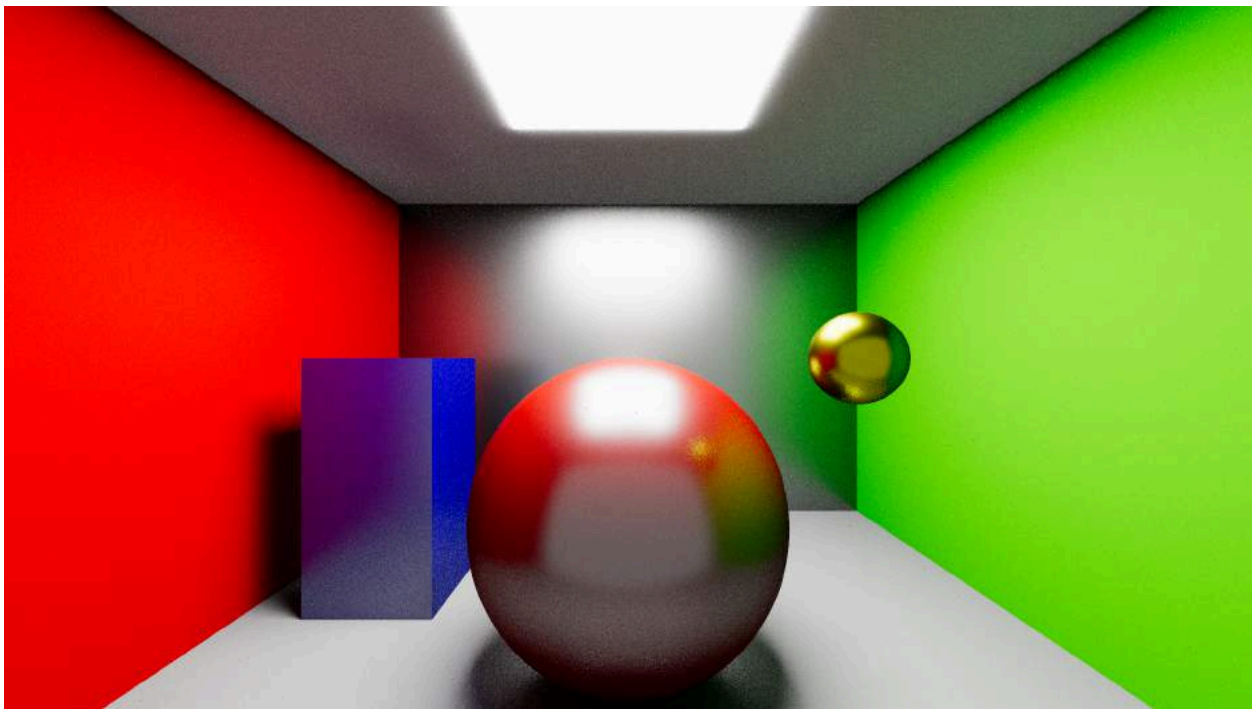


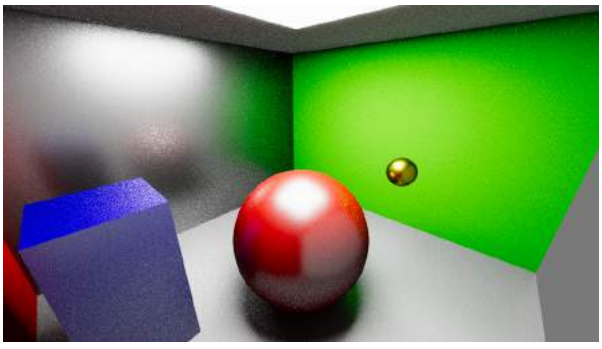
Na versão estendida da aplicação, temos:

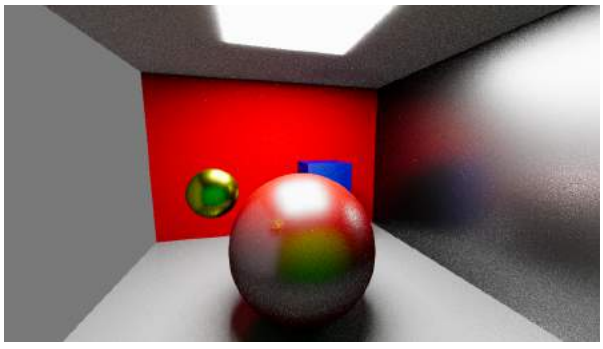
- Multiple Importance Sampling aplicado em todos os trechos do caminho em uma versão e no último trecho do caminho em outra versão. Os pesos usam a heurística da potência de Veach com $\beta = 2$.
- Roleta Russa com profundidade esperada de mais 4 trechos, ou seja, $q = \frac{1}{4}$.
- Luz ambiente (luz vindo do infinito de todas as direções).
- Objetos metálicos com rugosidade baixa com $\sqrt{\alpha} = r^2 = 0.3$, ou seja, reflexivos, além de objetos dielétricos com rugosidade baixa (mesmo α).

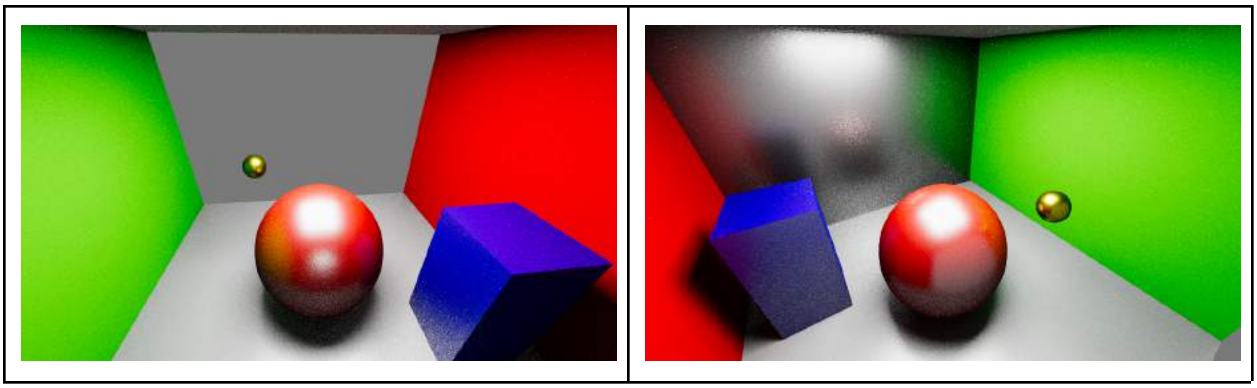
Abaixo, encontram-se imagens que comprovam a implementação dos métodos acima.

Multiple Importance Sampling em todos os trechos + Roleta Russa:









Roleta Russa implementada:

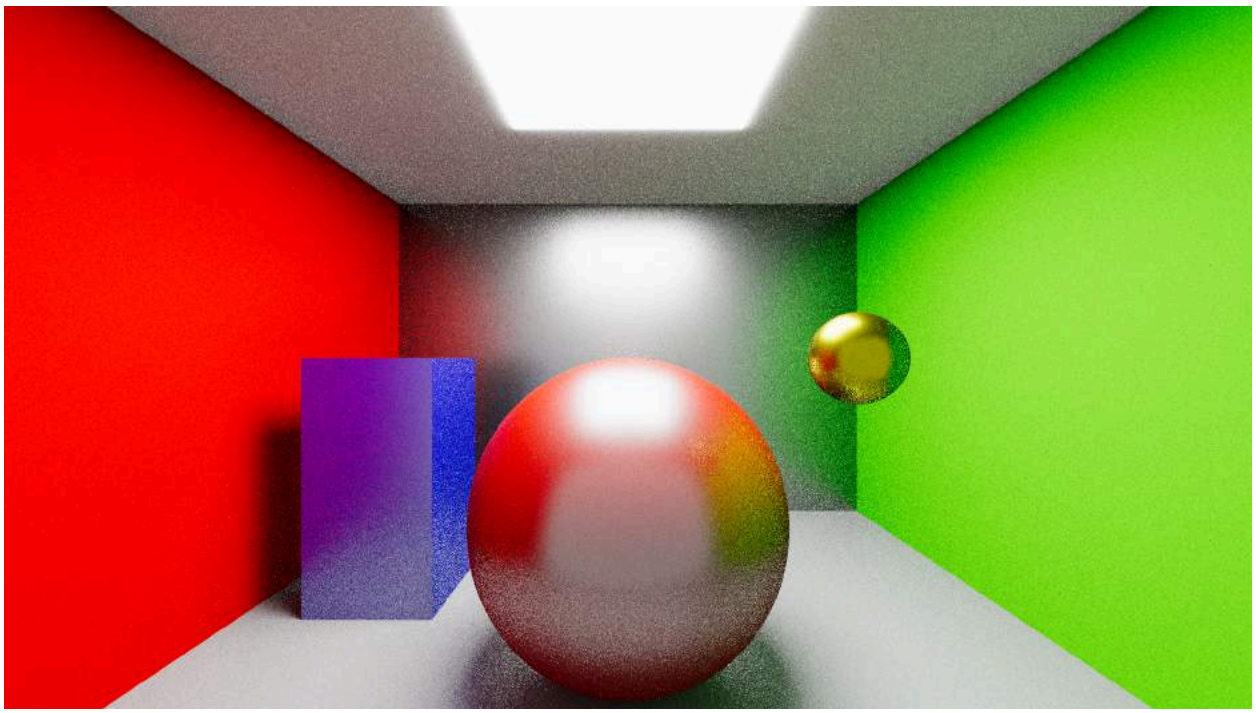
```
// Rendering
#define ERROR 0.0001
#define MINIMUM_DEPTH 4
#define GEOMETRIC_MEAN 4.0
```

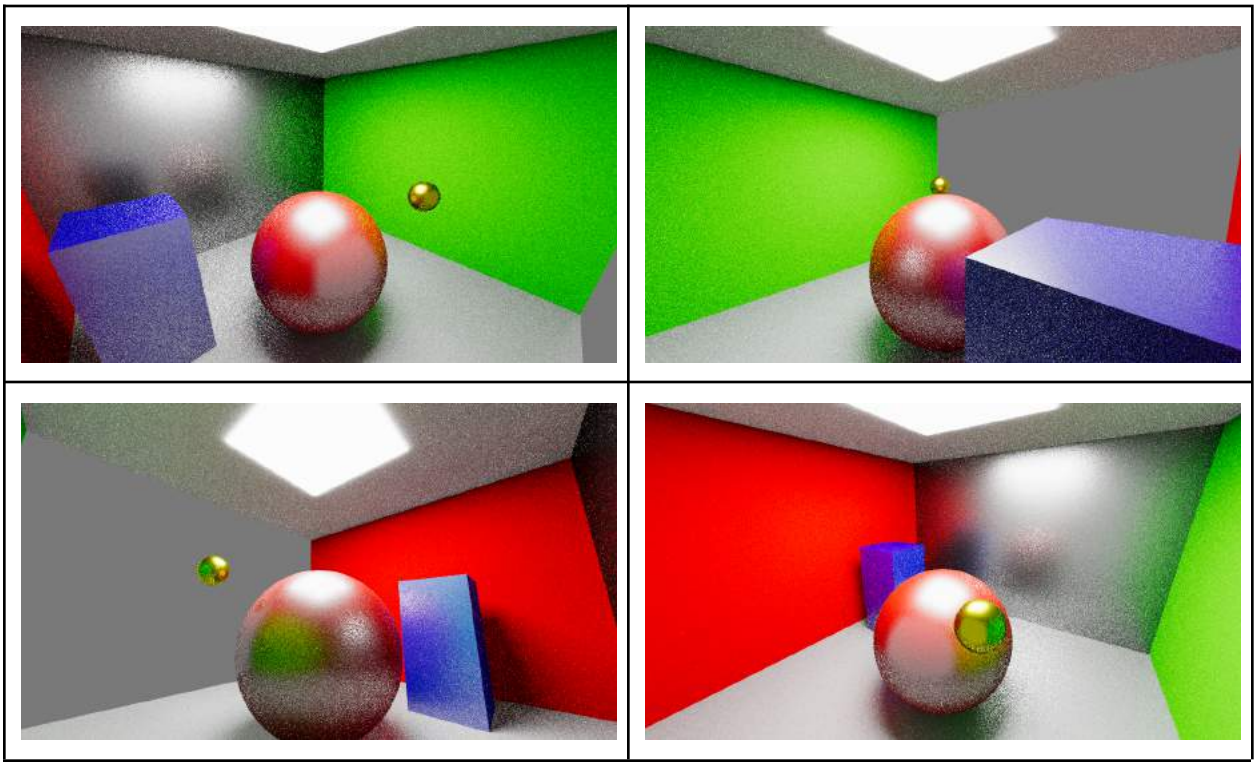
```
// Constructing the path
int i = 0;
while (true) {
```

Codigo…

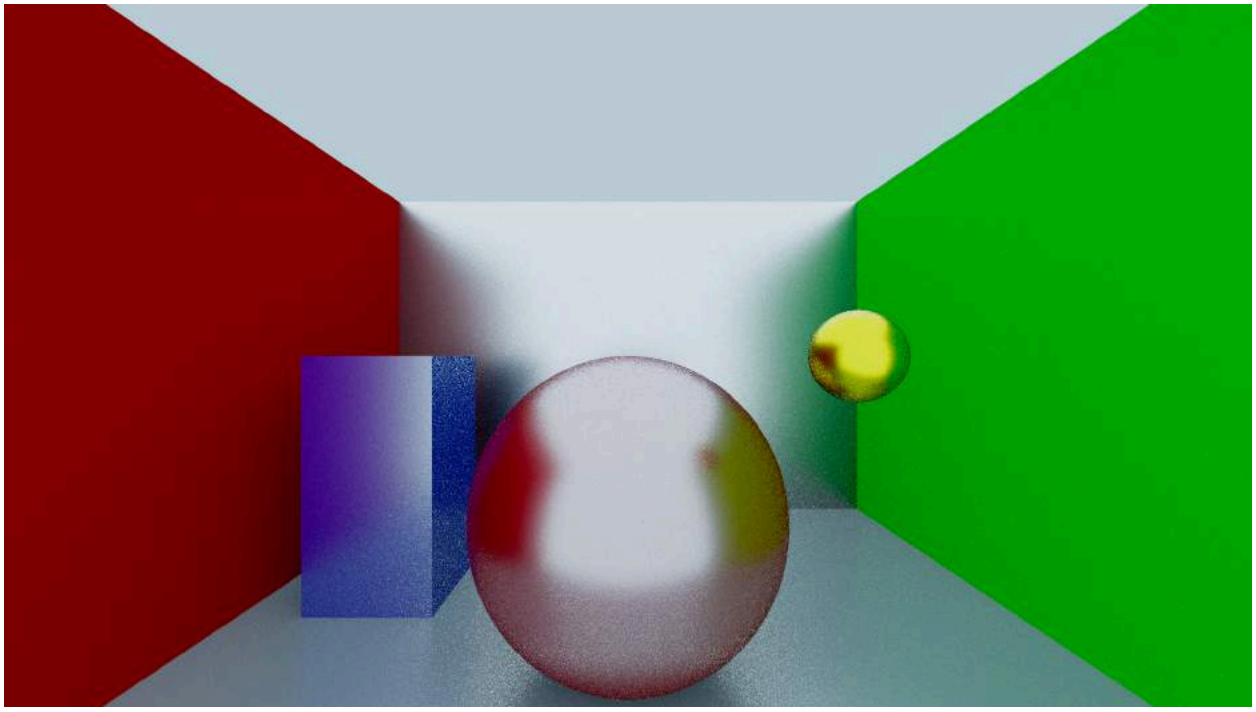
```
// Another Specular
if (i < MINIMUM_DEPTH)
    i++;
else {
    float xi = xi1();
    float q = 1.0/GEOMETRIC_MEAN;
    if (q <= xi)
        transmittance /= 1.0 - q;
    else
        break;
}
```

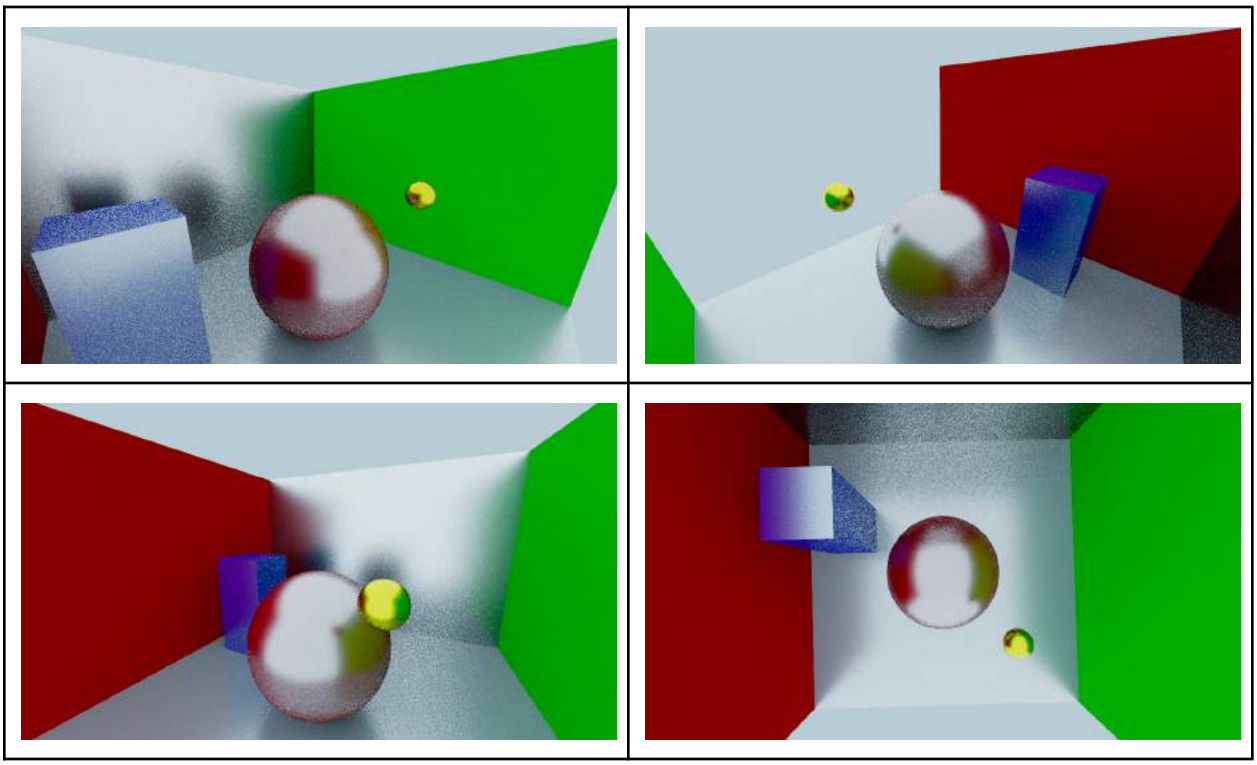
Multiple Importance Sampling apenas no último trecho
4º trecho sem Roleta Russa:





Luz ambiente, vindo do infinito de todas as direções:





Apenas objetos metálicos: