Universidade do Minho
Escola de Engenharia

# *Path Tracing* – Trabalho Prático Individual

## Visão e Iluminação II

Aplicação/Desenvolvimento da Reflexão Especular,
Refração e Roleta Russa para terminar os "*paths*"

Submetido por:

Diogo Nogueira A78957

# Conteúdo

# Contextualização

O propósito deste Trabalho Prático Individual passa por explorar a noção de *Path Tracing* por aplicação de um conjunto de funcionalidades que permitem assim testar este método de Computação Gráfico nos mais diversos cenários.

Com esse pensamento em mente, e de forma a simplificar o entendimento de todos os algoritmos pensados para a concretização estas funcionalidades, a ideia passa por abordar individualmente cada funcionalidade, com uma pequena discussão do algoritmo em si, seguindo-se de uma análise dos resultados para vários possíveis cenários.

Assim, o presente relatório assentará numa estrutura basilar e sucinta, formando os seguintes capítulos e sub capítulos:

- Funcionalidade de Materiais Especulares que apresentar-se-á nos materiais metálicos, usando-se para isso o exemplo da Esfera de Metal presente na Cornell Box inicialmente em uso

- Refração que acontece então nos meios translúcidos ou transparentes, usando-se para isso o exemplo da Esfera de Vidro e que depois será estendido para outros objetos/cenários

- Roleta Russa que permitirá criar uma abordagem probabilística, aumentando o número de FPS no *render* atual do *Composer* da NAU

# Desenvolvimento das Funcionalidades

## Materiais Especulares

### Algoritmo

Para criar o efeito de Materiais Especulares criou-se a função `__closesthit__radiance__metal()`. A mesma tem como base a versão iterativa do *path tracer* do OPTIX7, aplicando-se a produção da Reflexão e acrescentando-se a ideia de ***Glossiness*** e ***Glossy Rays***, de forma a lançar todos os *Glossy Rays* pedidos no *Composer* da NAU.

Para isso, assume-se a inserção das variáveis necessárias na *struct* GlobalParams.

- Aplicação Reflexão Raios

```
// 1. Normalizar o Vetor Normal
// O Vetor Normalizado é dado com a mesma direção
const float3 normalSurface = normalize(make_float3(n));

// 2. Calcular a Direção do Raio
const float3 rayDirection = optixGetWorldRayDirection();

// 3. Calcular a Posição
const float3 position = optixGetWorldRayOrigin() + rayDirec
tion * optixGetRayTmax();

// 4. Aplicar a Reflexão com a Direção do Raio e a Normal à
 Superfície
float3 reflection = reflect(rayDirection, normalSurface);
```

- Lançamento dos *Glossy Rays*

```
// Lançar o Número de Glossy Rays definidos no Composer
while(numberGlossyRays < glossyRays) {

  do{
     const float z1 = rnd(prd.seed);
     const float z2 = rnd(prd.seed);

     // Hemisphere Sampling consiste em disparar raios em d
ireção ao Hemisférico
     // O w_in no exemplo base passa a ser a Direção da Luz
 Incidente
     cosine_power_sample_hemisphere(z1, z2, directionLightI
ncident, glossiness);

     (…)

     // Para se poder criar a luz que é refratada da Bola
     prd.countEmitted = true;

} while (dot(directionLightIncident, normalSurface) < 0);

     // Dá o efeito Glossy ao Metal
     glossyMetal = glossyMetal + make_float3(0.5f);
     ++numberGlossyRays;
```

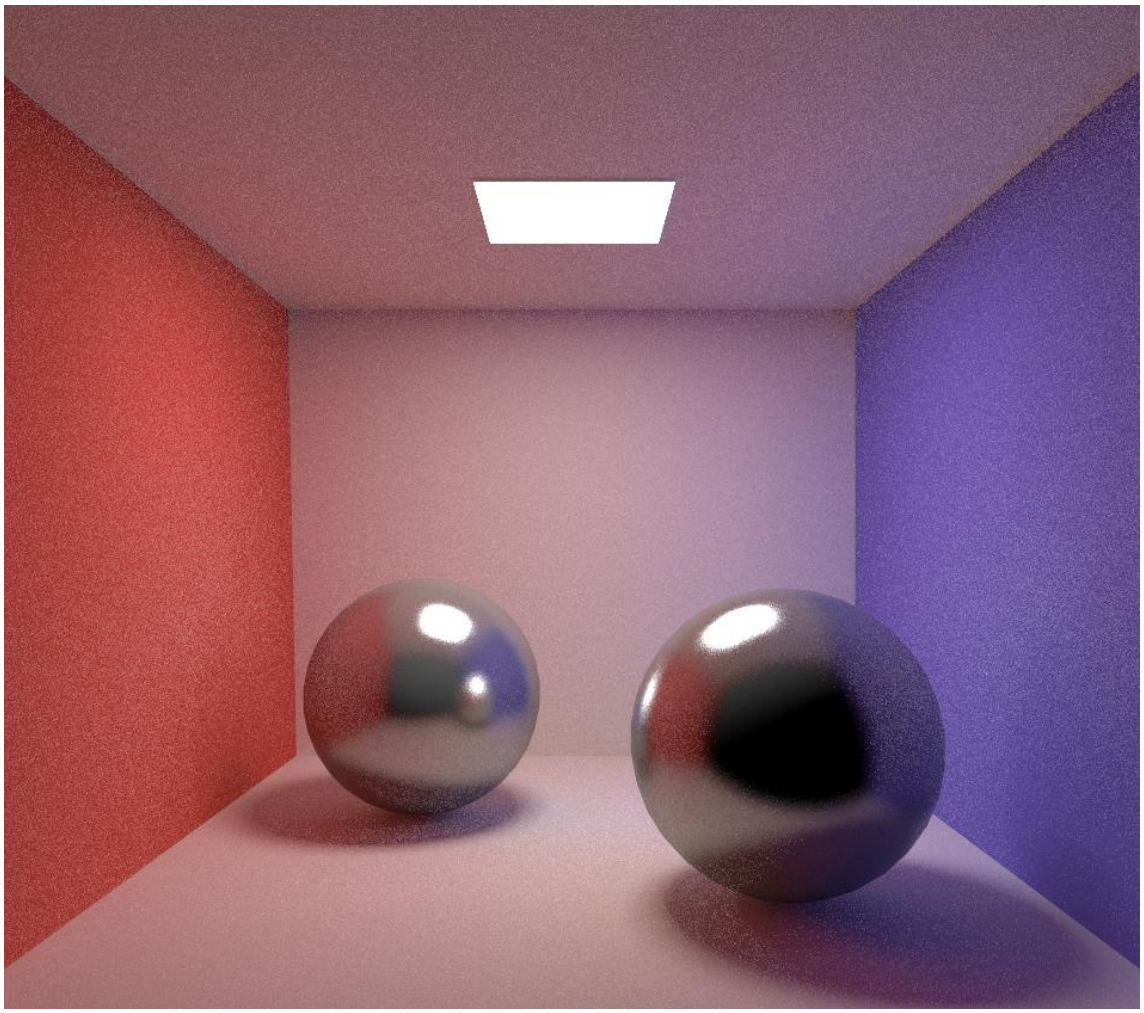
Estando todo o restante do código executado, calcula-se o novo valor da luz emitida, transmitida ou recebida pela superfície em si – *radiance*. Note-se que para este cálculo se mantém o que já exista multiplicando-se à divisão entre o valor da variável `glossyMetal` e da variável `glossyRays`.

Quanto maior o valor da variável **`glossyMetal`** maior efeito *glossy* que incide sobre a superfície.

```
prd.radiance += make_float3(5.0f, 5.0f, 5.0f) * weight
             * optixLaunchParams.global->lightScale
             * (glossyMetal/glossyRays);
```

## Visualização dos Resultados

Cenário *Cornell Box Sphere2*



# Refração

## Algoritmo

Para criar o efeito da Refração, típico dos materiais translúcidos ou transparentes criou-se a função `__closesthit__radiance__glass()`. Relembre-se que o fenómeno da Refração vem sempre acompanhado do fenómeno da Reflexão, na medida em que parte da luz que incide sobre a superfície acaba por ser refletida e outra parte refratada.

Dessa forma, esta função trata dos dois fenómenos, que têm e devem ser controlados/balanceados no algoritmo em si.

- Aplicação Reflexão Raios

```
// 1. Normalizar o Vetor Normal
// O Vetor Normalizado é dado com a mesma direção
const float3 normalSurface = normalize(make_float3(n));

// 2. Calcular a Direção do Raio
const float3 rayDirection = optixGetWorldRayDirection();

// 3. Calcular a Posição
const float3 position = optixGetWorldRayOrigin() + rayDirec
tion * optixGetRayTmax();

// 4. Aplicar a Reflexão com a Direção do Raio e a Normal à
 Superfície
float3 reflection = reflect(rayDirection, normalSurface);
```

- Aplicação Refração Raios

```
float3 refraction;

// Produto Escalar do Vetor da Direção do Raio e do Vetor d
a Normal à Superfície < 0
if(dot(rayDirection, normalSurface) < 0)
{
   refraction = refract(rayDirection, normalSurface, 0.50);
}
// Produto Escalar do Vetor da Direção do Raio e do Vetor d
a Normal > 0
else
{
   refraction = refract(rayDirection, -normalSurface, 1.5);
}

prd.attenuation *= sbtData.diffuse;
prd.countEmitted = true;
prd.origin = position;
prd.done = false;
```

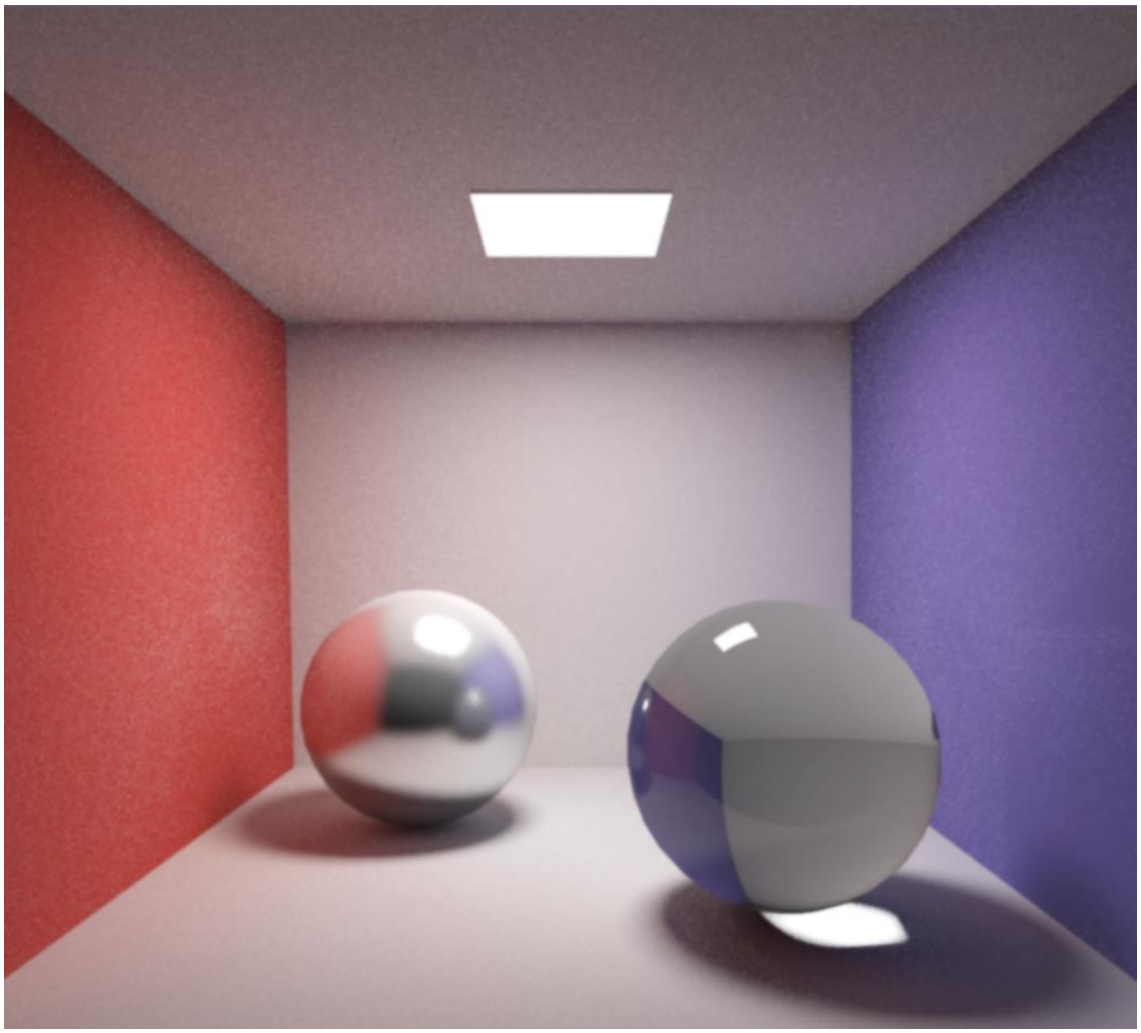
Para se finalizar, cria-se uma espécie de aleatoriedade através da função `rnd` e com isso estabelece-se a ideia de definir a direção da luz como Refração ou Reflexão.

Quanto maior o valor da variável estimado em `rnd(prd.seed) * M_PIf`, maior efeito de refração na superfície, como seria espectado.

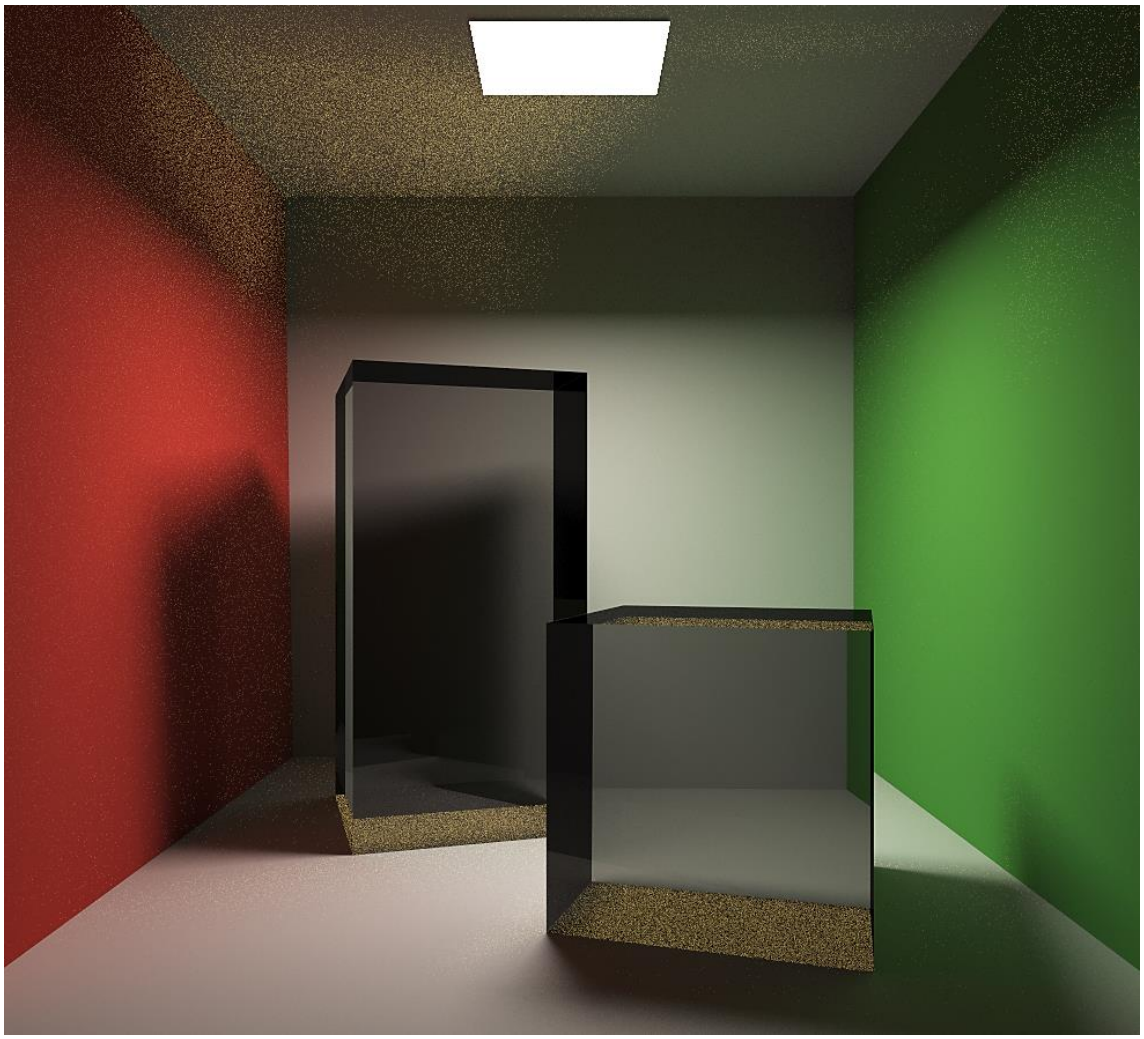
```
// Seed é a semente que se usa para gerar o número através
da função random(rnd)
if(rnd(prd.seed) < rnd(prd.seed) * M_PIf)
{
    prd.direction = refraction;
}
else
{
    prd.direction = reflection;
}
```

## Visualização dos Resultados

Cenário *Cornell Box Sphere2*



Cenário *Cornell Box Original*

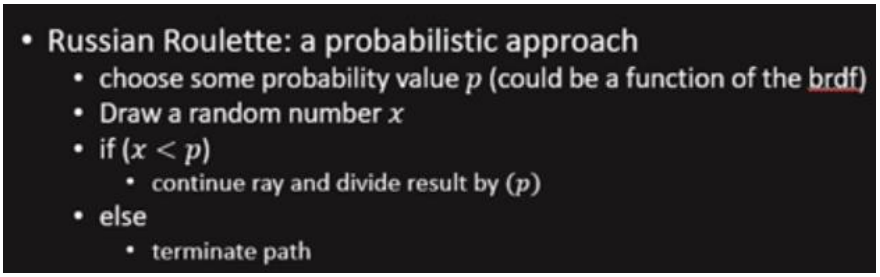


Cenário *Sponza*

# Russian Roulette

## Algoritmo

Relativamente à implementação da Roleta Russa, tomou-se como base o pseudo algoritmo fornecido pelo docente numa das aulas não presenciais. A imagem abaixo anexada refere os passos que criam a ideia de como construir a abordagem probabilística da Roleta Russa.

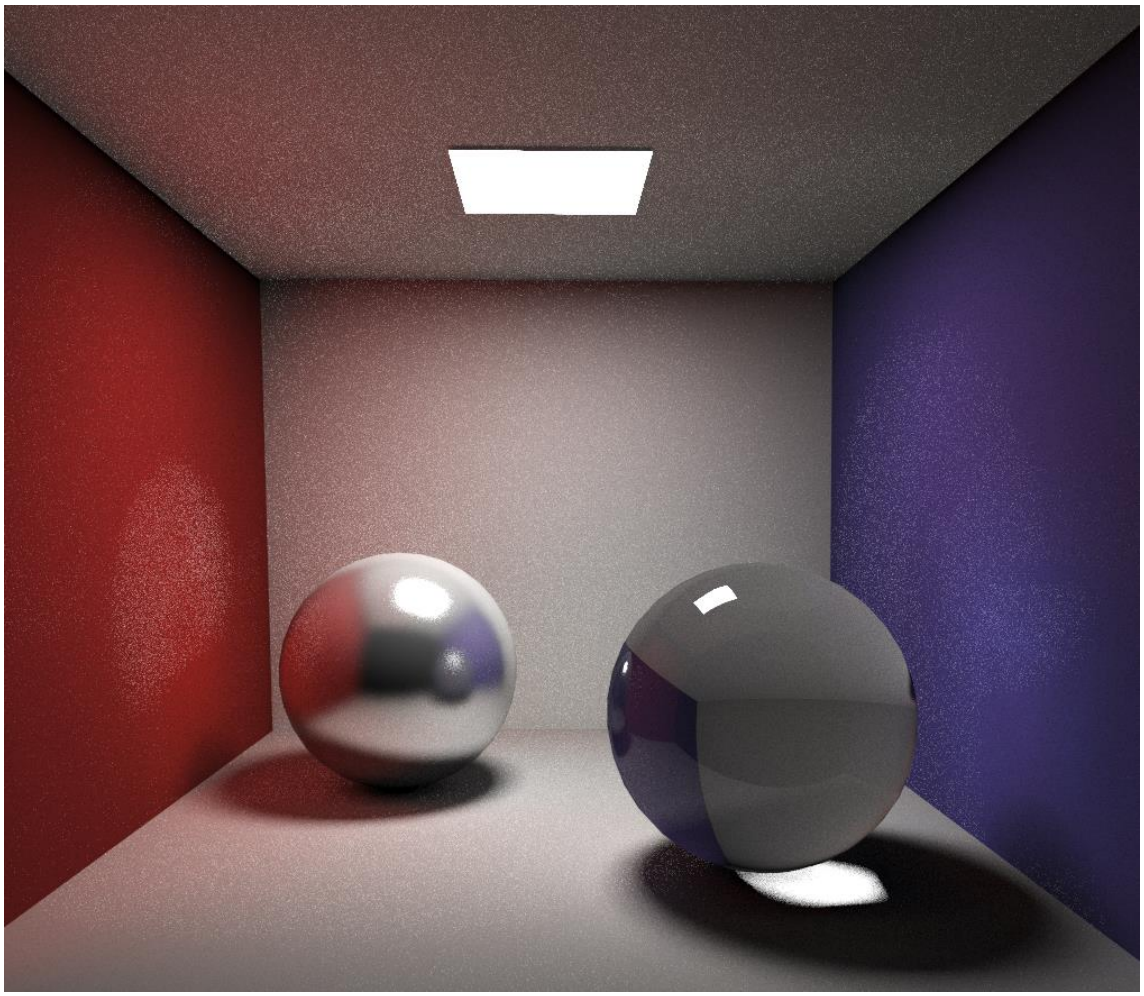



```
float randomValue = rnd(prd.seed);
float probability;

if(randomValue < probability)
{
   (…)
    prd.radiance
      += make_float3(5.0f, 5.0f, 5.0f)
      * weight
      * optixLaunchParams.global->lightScale/probability;
}

else
{
     prd.done= true;
}
```

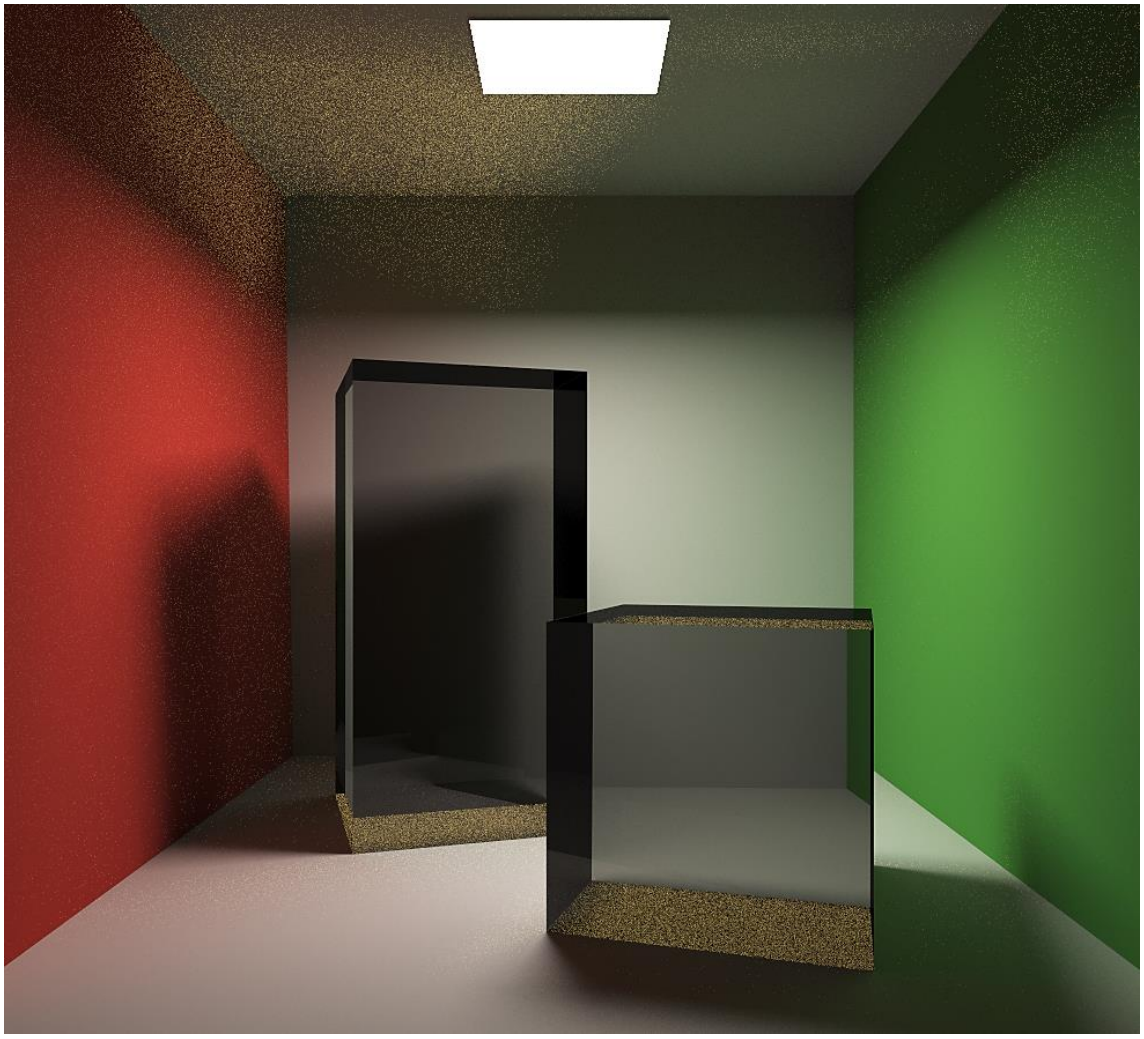
Torna-se então a usar a função `rnd` para gerar o tal *random number x*. Depois disso, apenas se aplica a verificação e dentro da mesma continua-se a parte do raio que já existia na base do código, terminando-se com a divisão do resultado pelo valor da probabilidade calculado.

## Visualização dos Resultados

Cenário *Cornell Box Sphere2*



Cenário *Cornell Box Original*



Cenário *Sponza*

# Análise do Resultados e Observações Finais

Estando toda a parte da visualização dos resultados totalmente apresentada, pode-se fazer uma análise dos resultados e através disso criar algumas observações finais. Esta análise tem como base os mesmos valores de *Max Depth*, *Gamma* e etc.

1. Com Roleta Russa

- Cenário *Cornell Box Sphere2* e *Original*

    - O cenário das Esferas foi o que reproduziu melhores resultados
    - O cenário das Caixas apresenta muito mais ruído visual
    - O cenário das Caixas apenas suporta o material de Vidro, sendo que o *Composer* deixa de responder quando se tenta introduzir o material de Metal
    - Quando se tenta alterar os valores da Refração e Reflexão, o *Composer* deixa também de responder e quando não o faz (dependendo dos valores atribuídos) cria um material de Vidro com uma qualidade muito inferior.

Quanto mais se mexe no valor da Refração, pior o resultado.

```
if(rnd(prd.seed) < rnd(prd.seed) * M_PIf)
{
    prd.direction = refraction;
}
else
{
    prd.direction = reflection;
}
```

- Cenário *Sponza*

    - Contrariamente ao que acontece nos dois cenários anteriores, quando se tenta alterar os valores da Refração e Reflexão, o resultado visual melhora, sendo que o *Composer*, comporta-se de forma normal
    - Este cenário apenas suporta o material de Vidro, tal como o cenário da *Cornell Box Original*

## 2. Com Roleta Russa

- A Roleta Russa funciona no sentido de aumentar os FPS, na tentativa de melhor a qualidade de imagem produzida e terminar os "*paths*"
- No cenário *Cornell Box Sphere2* e *Original* a mesma funciona, levando para o dobro dos FPS iniciais
- No cenário *Sponza* não existiu qualquer diferença com o uso da Roleta Russa