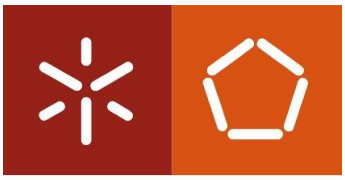

Universidade do Minho

Departamento de Informática

Visualização e Iluminação II
Elevado número de fontes de luz

José Ferreira (A83683), Jorge Mota (A85272)

June 28, 2021

Contents

1 Introdução **3**

2 Implementação **4**
2.1 Aumentar o número de luzes 5
2.2 Amostragem de luzes de forma uniforme 5
2.3 Amostragem de luzes de forma pesada 6

Chapter 1

Introdução

Iluminacao e algo fundamental para a criação de ambientes simulados em computador que sejam realistas e visualmente agradáveis. Para simular de forma realista o caminho que e necessário computar para cada ponte de cada superfície quais são as luzes que afectam a sua aparência. Desta forma, com o aumento das luzes numa dada cena a complexidade de a computar também tem tendência a aumentar.
Visto que tal e completamente impraticável existem soluces que tentam mitigar este problema.
Ao longo deste relatório iremos explorar e comparar diversas soluções desenvolvidas.

Chapter 2

Implementação

O nosso objetivo principal consiste em encontrar um equilíbrio na renderização das luzes num ponto, de forma a diminuir o custo de computação e diminuir o tempo de convergência.

Primeiro começamos por obter métricas de performance com base no código disponibilizado no Tutorial 4. Este tutorial aplica o *shading* de todas as luzes a todos os pontos, tendo uma solução O(n), sendo **n** o numero de luzes totais na cena.

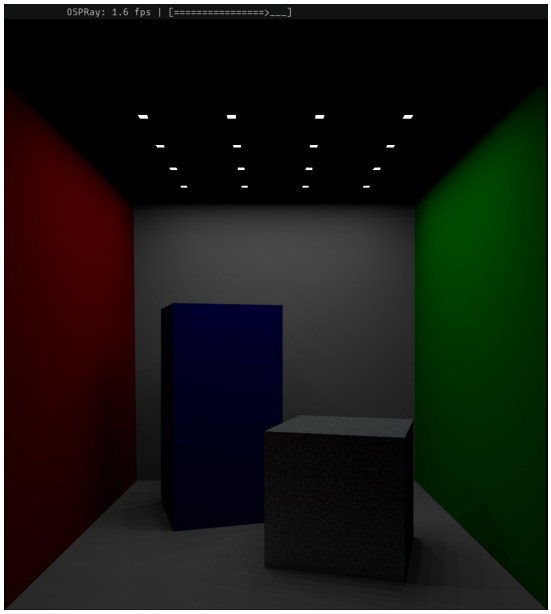

Figure 2.1: Amostragem de todas as luzes (∼`1.5 FPS`)

Esta foi a solução original que utilizamos como base para comparar os resultados das mudanças que fizemos.

2.1 Aumentar o número de luzes

Para que o cenário da Cornell Box seja aplicável para o nosso caso de estudo é necessário primeiro aumentar o número de luzes deste. Para isso inserimos novas luzes ao longo do teto, compondo o código de forma a podermos facilmente aumentar o número de luzes a qualquer momento.

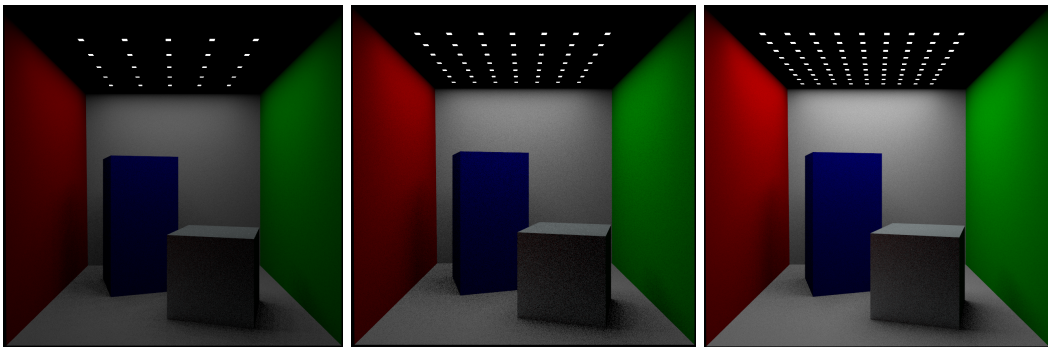

Figure 2.2: luzes (5x5, 7x7, 9x9)

2.2 Amostragem de luzes de forma uniforme

Uma possível solução para diminuir o custo de computação significativamente é selecionar de forma aleatória com uma distribuicao uniforme uma luz e dividir a radiância emitida pela luz (se não ocludida) pela probabilidade de cada luz ser selecionada.

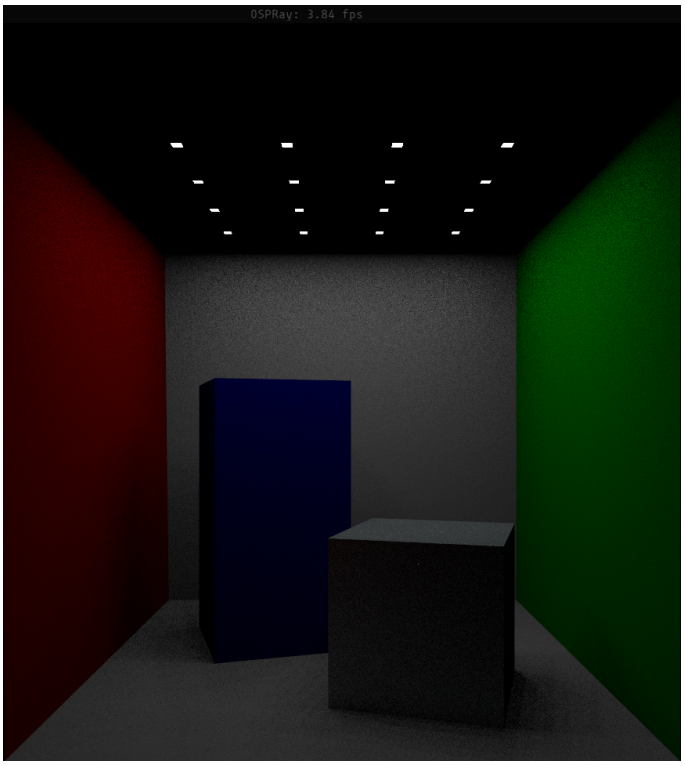

Figure 2.3: Amostragem uniforme (∼`3.8 FPS`)

A performance do path tracer melhorou significativamente de ∼1.5FPS para ∼3.8FPS. No

entanto, podemos notar algum aumento do ruído presente na imagem, nomeadamente na parede e no chão. Tal deve-se ao facto de como escolhemos de forma aleatoriamente uniforme a luz que vai dar *shade* a um certo ponto tem igual probabilidade de escolher qualquer luz, o que significa que não há qualquer noção de quais as luzes influenciam mais aquele ponto. Tal também leva a um aumento do tempo de convergência.

2.3 Amostragem de luzes de forma pesada

Finalmente chegamos à solução final que produz resultados relevantes para manter uma quantidade de ruído reduzida, e reduzir o tempo de computação.

Esta solução, reutiliza o modo de amostragem uniforme, mas as probabilidades de cada luz são reflectidas pelo grau de contribuição que essa luz produz para um certo ponto.

Este método é implementado a partir de dois vetores com tamanho `numLights` (`lights_pdf`, `lights_cdf`) que têm a probabilidade de cada luz e a probabilidade acumulada. É amostrado um número de 0 a 1 e utiliza-se a probabilidade acumulada para saber qual das foi seleccionada, de seguida calcula-se a radiância emitida e divide-se pela respectiva probabilidade para aquela luz.

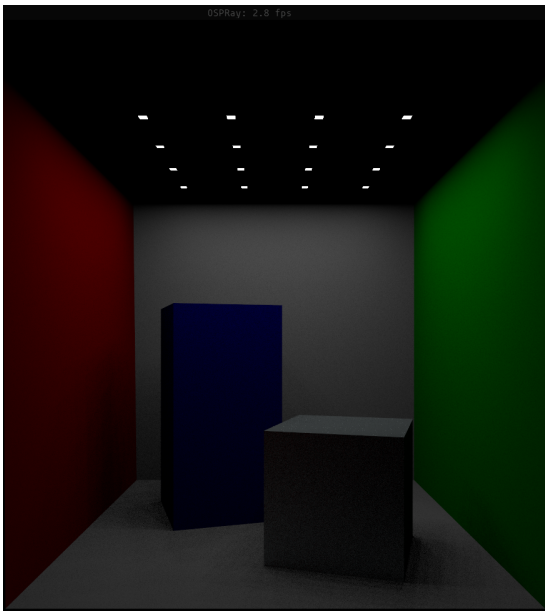

Figure 2.4: Amostragem pesada de uma luz (∼`3.0 FPS`)

Como podemos observar os resultados mostram que há muito menos ruído, enquanto que se mantém a performance melhor que a solução do Tutorial 4.