



Universidade do Minho

Visualização e Iluminação

MEI - 1º ANO - 2º SEMESTRE
UNIVERSIDADE DO MINHO

TRABALHO PRÁTICO

<i>Trabalho realizado por:</i>	<i>Número</i>
Pedro Paulo Queirós Tavares	PG50698
Tiago Daniel Lopes Aroso da Costa	PG50777
Duarte Nuno Pereira Moreira	PG50349

Braga, 24 de junho de 2023

Índice

1	Introdução	2
2	Tone Mapping	3
3	Parallel Multithreading	5
4	Environment Lights	6
4.1	Implementação e resultados	6

1. Introdução

No âmbito da terceira fase do trabalho prático o grupo decidiu implementar três temas da lista proposta pelo docente, sendo estes, *tone mapping*, *parallel multithreading* e por fim, *environment lights*.

2. Tone Mapping

Para o tema *Tone Mapping* foram implementadas diversas versões de algoritmos, entre eles os seguintes:

- *Normalization*
- *Gamma compression*
- *Drago et al* (<https://resources.mpi-inf.mpg.de/tmo/logmap/logmap.pdf>)
- *Reinhard et al* (<https://www-old.cs.utah.edu/docs/techreports/2002/pdf/UUCS-02-001.pdf>)
- *ACES*
- *Mantiuk et al* (<https://www.cl.cam.ac.uk/~rkm38/pdfs/mantiuk08datm.pdf>)

No entanto, o que mostrou ser mais apelativo e versátil, destes algoritmos, foi o *tone mapping* apresentado no trabalho de *Mantiuk*. Este algoritmo permite ao utilizador fazer variar os parâmetros de **contraste**, **saturação** e **detalhe** que melhor se adequem á imagem em questão. Apresentam-se a seguir alguns dos resultados obtidos com este novo *tone mapper*.



Figura 2.1: Contraste = 0.4

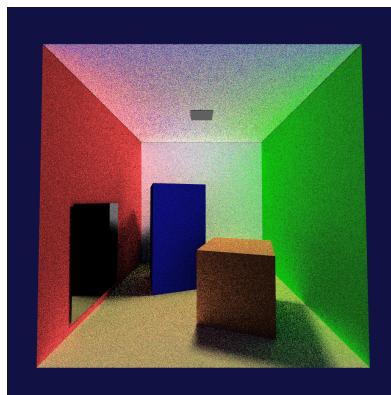


Figura 2.2: Contraste = 0.8

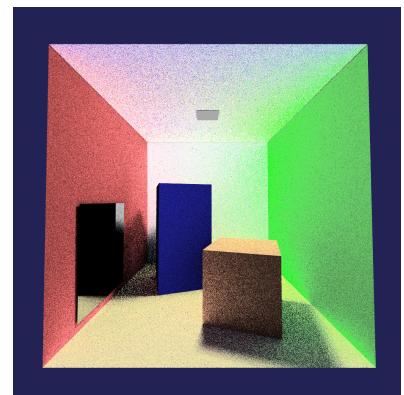


Figura 2.3: Contraste = 1.6



Figura 2.4: Saturação = 0.4



Figura 2.5: Saturação = 0.8

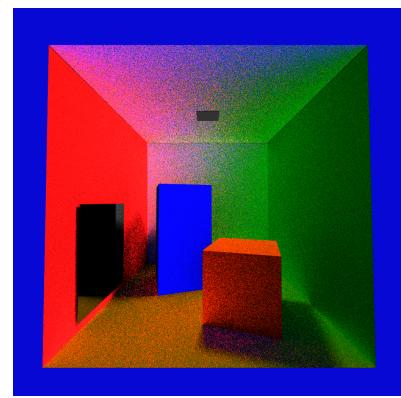


Figura 2.6: Saturação = 1.6

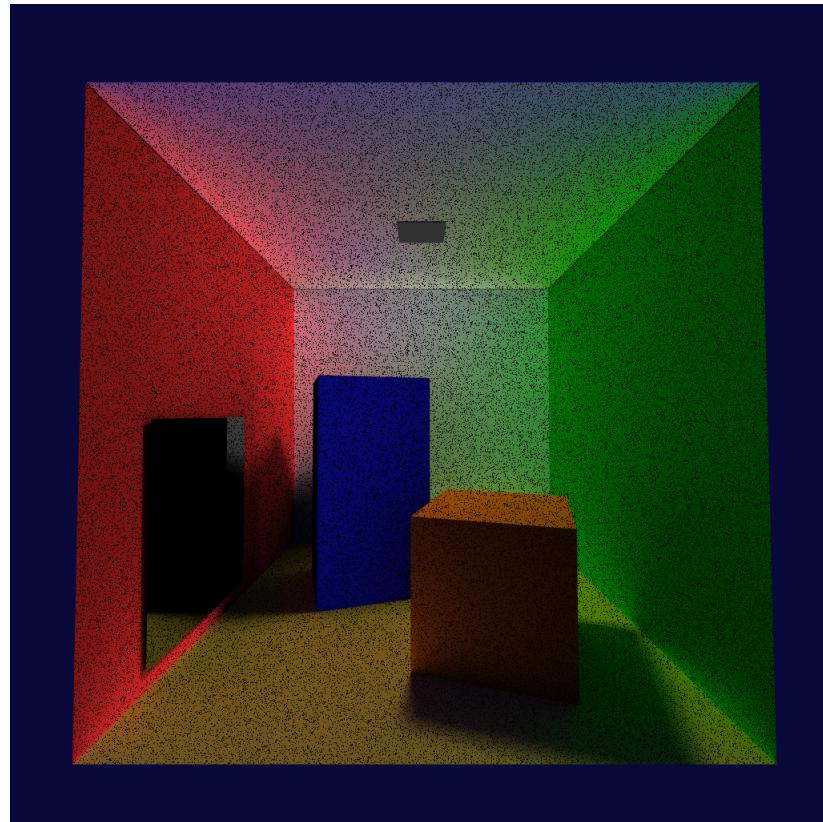


Figura 2.7: Resultado final com 4096spp.

3. Parallel Multithreading

Para a realização da tarefa de paralelismo é utilizada a classe *Thread* do *standard namespace* de C++. A paralelização acontece aquando da renderização da cor RGB de cada *pixel*, ou seja, na classe *StandardRenderer*. A estratégia utilizada foi a seguinte: de forma a tirar proveito da localidade espacial - acesso a dados que estejam armazenados de forma contígua em memória - e de forma a realizar uma eficiente distribuição da carga de trabalho pelas *threads*, "constrói-se" uma matriz de tamanho *Width*/16 por *Height*/16 sobre a imagem e atribui-se de forma alternada, horizontalmente, cada quadradinho de 16x16 pixels à respetiva *thread*. Isto irá impedir que uma *thread* perca demasiado tempo a processar uma região mais complexa da imagem, como por exemplo, espelhos. Esta abordagem permite que grande parte das *threads* contribuam para o processamento de uma região pesada computacionalmente desde que esta não apresente uma dimensão inferior a 16x16 pixels.

Assuma-se uma imagem de tamanho 128x128 a ser processada por 3 *threads*. A *thread* 1 dá a cor vermelha ao *pixel*, a *thread* 2 a cor verde e a *thread* 3 a cor azul. A distribuição da carga de trabalho pelas *threads* pode ser vista na figura seguinte.

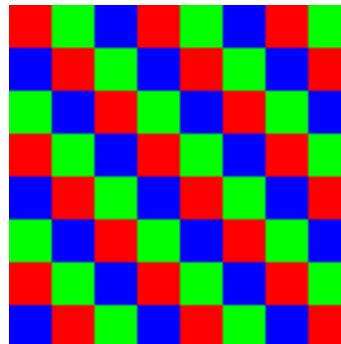


Figura 3.1: Exemplo da distribuição de um *frame* por 3 *threads*.

4. Environment Lights

Os tipos de luzes já implementadas não dão um certo sentido de realismo às cenas porque são luzes "artificiais", e, então, surge o conceito de *environment lights*. Este tipo de luzes pode ser considerado como uma Area Light infinitamente distante que envolve toda a cena. A maneira como estas luzes podem ser vistas é uma esfera que emite raios de luz para a cena de todas as direções. Estas permitem iluminar o objeto de uma forma mais realística como se o mesmo estivesse no ambiente.

A implementação das *environment lights* começou por procurar como seriam mapeadas estas luzes para as cenas. A forma mais comum de se realizar este passo é utilizar uma imagem de alta resolução com o formato *.exr* ou *.hdr* que se chama *Environment Map*. De modo a integrar estas imagens no nosso projeto necessitamos de implementar uma estrutura denominada *HDRImageBuffer* que contém as dimensões da imagem e um vetor da mesma dimensão que guarda os RGB da imagem. A leitura das imagens de alta resolução foi feita com recurso à biblioteca *tinyexr*.

Como já referido, este tipo de luzes pode ser visto como esfera que engloba a cena sendo então necessário criar uma *bounding sphere* à volta da cena para obtermos o seu raio e o seu centro. Existem vários métodos que implementam esta *bounding sphere*, mas na nossa implementação optamos por um método proposto por *Jack Ritter* https://en.wikipedia.org/wiki/Bounding_sphere. Com este método obtemos o raio da esfera e o seu centro que serão mais à frente utilizados para a *pdf*.

4.1 Implementação e resultados

Toda a implementação destas luzes pode ser encontrada no ficheiro *InfiniteAreaLight.hpp* que contém o construtor destas luzes. Este construtor recebe então o environment map já criado, o raio e o centro da esfera. Por forma a manter o *path tracer* já implementado, era necessário definir o método *Sample_L* que recebe dois números aleatórios e gera um ponto com uma certa pdf. A implementação deste método foi utilizando o Global Illumination Compendium que gera um ponto com uma probabilidade uniforme dentro da esfera criada. Com este ponto da esfera podemos então calcular qual é a contribuição da fonte de luz no mesmo. Isto é feito convertendo o ponto cartesiano para coordenadas esféricas, e posteriormente para coordenadas de textura que vão ser interpoladas bilinearmente no environment Map. De notar que preferimos utilizar uma distribuição uniforme pela simplicidade de implementação, porém uma técnica mais avançada seria realizar uma distribuição conforme o environment map.

A única alteração que foi realizada no *Path Tracer* foi no caso em que o raio primário não interseca nenhuma primitiva, sendo que neste caso não é retornada uma cor fixa, mas sim é feita uma procura na textura para retornar esse valor.

Para efeitos de teste, foram utilizados os paralelepípedos da *Cornell Box* sendo que os seus materiais foram alterados para espelho de modo a confirmar os resultados.



Figura 4.1: Imagem obtida pelo renderer com um environment map de um campo de trigo