Visão e Iluminação

(1º ano de Mestrado em Computação Gráfica)

Trabalho de Grupo

Relatório

Alexandre Flores	Pedro Alves	Rui Armada
(PG50165)	(A93272)	(PG50737)

7 de julho de 2023

Resumo

O presente relatório serve para documentar o processo de desenvolvimento de um *raytracer*, implementado em C++, no âmbito da cadeira de Visão e Iluminação. Aqui, serão apresentados todos os *renders* feitos com a aplicação desenvolvida de forma a comprovar o foto-realismo alcançado pelo mesmo.

Conteúdo

Capítulo 1

Introdução

Este relatório apresenta uma análise detalhada do projeto de Visão e Iluminação realizado, que consistiu na implementação de um *raytracer* em C++. O objetivo principal deste projeto foi explorar os princípios fundamentais da renderização de imagens por meio de raios, visando compreender e aplicar conceitos relacionados à física da luz, geometria e interações entre os objetos presentes numa cena tridimensional.

O desenvolvimento de um *raytracer* envolve a criação de um *software* capaz de simular a trajetória dos raios de luz a partir de uma fonte luminosa, passando pelos objetos presentes na cena e acabado por atingir uma câmara virtual. Esta técnica é amplamente utilizada em áreas como animação, jogos e computação gráfica, pois permite gerar imagens de alta qualidade com efeitos de iluminação realistas.

Durante o processo de implementação, foram explorados conceitos-chave, como lançamento de raios primários, interseção com objetos geométricos, cálculo de sombras e reflexões. Além disso, foram aplicados modelos de iluminação, como o modelo de microfacetas, para determinar a aparência dos objetos com base nas propriedades dos materiais, incluindo reflexão especular e difusa.

Neste relatório, serão apresentados especificamente os aspetos da última fase do trabalho, nomeadamente a exportação em vários formatos diferentes e um modelo de *physically-based shading*.

Capítulo 2

Implementação

2.1 Estratégia Geral

O código foi baseado no que foi desenvolvido nas aulas, com algumas modificações. Nomeadamente, os *shaders* desenvolvidos nas aulas foram removidos, sendo que eram, em geral, incompatíveis com o novo modelo utilizado, e inutilizados.

Para além disso, o sistema de iluminação foi abstraído para um sistema de classes baseado na classe `Light`, que implementa todos os métodos necessários para implementar uma luz.

2.2 Descrição de cenas

De modo a poder especificar características não presentes no formato `obj` (posição da câmara, configurações do *renderer*, luzes, etc.), foi criado um formato de descrição de cena baseado em JSON.

Este formato foi combinado com um *script* para o Blender que permite exportar qualquer cena para um formato compreensível pelo nosso renderizador.

Um exemplo desta descrição de cena para a imagem de referência pode ser visto aqui:

```
{
  "title": "Reference",
  "model": "models/reference.obj",
  "samplesPerPixel": 16,
  "width": 960,
  "height": 540,
  "frames": [
    {
      "frame": 0,
      "camera": {
        "position": {
          "x": 7.358891487121582,
          "y": 4.958309173583984,
          "z": 6.925790786743164
```

```
        },
        "up": {
          "x": 0,
          "y": -1,
          "z": 0
        },
        "lookingAt": {
          "x": 6.6686553955078125,
          "y": 4.641676425933838,
          "z": 6.2751617431640625
        },
        "fov": {
          "x": 0.6911112070083618,
          "y": 0.4710899591445923
        }
      }
    }
  ],
  "lights": [
    {
      "type": "point",
      "pos": {
        "x": 4.076245307922363,
        "y": 5.903861999511719,
        "z": -1.0054539442062378
      },
      "color": {
        "r": 1.0,
        "g": 1.0,
        "b": 1.0
      },
      "power": 1000.0
    },
    {
      "type": "ambient",
      "color": {
        "r": 1,
        "g": 1,
        "b": 1
      }
    }
  ],
  "output": "image.png"
}
```

Como podemos ver, esta especificação define a localização do modelo `obj` a utilizar, os *samples* por pixel, a resolução da imagem, os parâmetros da câmara, a configuração das luzes, e a imagem final. O formato especifica a posição da câmara num *array*, para uma possível implementação futura de animação simples.

2.3 Formato de Imagem

A primeira parte deste trabalho envolveu permitir a exportação em vários formatos de imagem. Préviamente, o nosso programa apenas suportava exportação de PNGs através da biblioteca SPNG.

Para esta fase, como pretendiamos exportar formatos *floating-point* como EXR, alterámos a representação interna de uma imagem para um *array* de *floats*. Com isto, utilizámos o OpenImageIO, uma biblioteca *open-source* utilizada profissionalmente por vários estúdios de efeitos visuais, para suportar vários formatos de imagem.

O formato é detetado automaticamente através da extensão do ficheiro especificado na descrição da cena, e o OpenImageIO trata de toda a conversão, permitindo-nos suportar vários formatos sem grande esforço da nossa parte.

2.4 Physically-Based Rendering

A parte principal deste trabalho envolveu refazer o modelo de *shading* do nosso renderizador para ser *physically-based*. Para isto, o nosso principal objetivo foi obter resultados semelhantes ao Blender, portanto baseámo-nos no modelo por ele utilizado, o Principled BSDF.

2.4.1 Shader Principled BSDF

Este *shader* é baseado no Principled BSDF desenvolvido pela Disney [1] para uso nos seus filmes de animação. O modelo de iluminação é baseado em modelos *physically-based*, nomeadamente modelos de microfacetas, mas focando-se mais em permitir controlo artístico do que em ser 100% realista.

Nomeadamente, este *shader* implementa várias funcionalidades como camadas de *clearcoat* e transmissão que nós não implementámos, tendo-nos focado apenas em implementar a reflexão difusa e especular.

Ambos estes tipos de reflexão são baseados no modelo de microfacetas, controlado por três parâmetros: *roughness*, *metalness* e *base color*.

A *roughness*, que varia entre 0 e 1, afeta o quão *glossy* uma superfície será. Uma *roughness* de 0 será uma superfície completamente plana, com reflexões puramente especulares (sendo, efetivamente, um espelho), enquanto que uma *roughness* de 1 terá reflexões puramente difusas.

O componente *metalness*, que também varia entre 0 e 1, serve para fazer um *blend* entre dois modelos de reflexão diferentes. O modelo metálico não tem componente difusa e tem uma componente especular tingida com a *base color*.

A *base color* é utilizada para definir a cor principal do material. É a única cor utilizada, sendo que tanto a iluminação difusa como especular se baseiam nela.

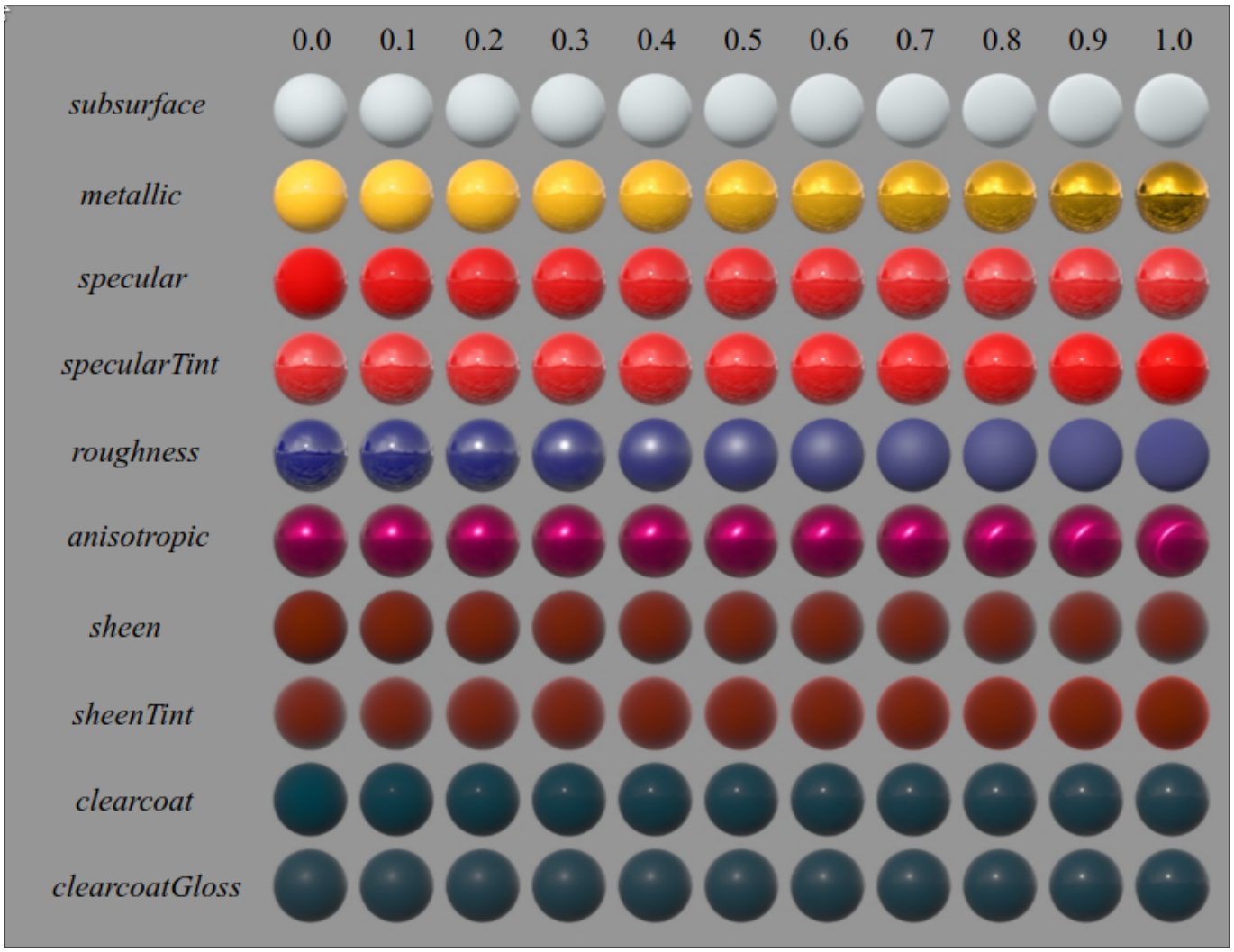

Figura 2.1: Todos os parâmetros suportados pelo shader Principled BSDF. Apenas utilizamos *roughness* e *metallic*.

2.4.2 Modelo de microfacetas

O nosso *shader* baseia-se num modelo de microfacetas, com uma distribuição GGX.

Neste modelo, uma superfície é modelada como sendo composta por várias micro-superfícies (microfacetas) perfeitamente reflexivas. Estas microfacetas fazem com que, visto de longe, um material apresente uma reflexão difusa ou com alguma *roughness*.

No nosso modelo, a direção de cada raio refletido é calculada utilizando a distribuição GGX, utilizada no shader Principled BSDF.

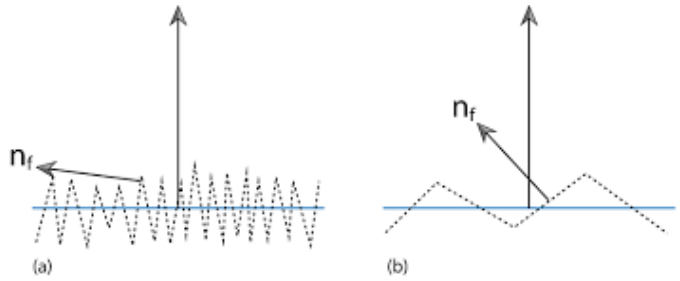

Figura 2.2: (a) *roughness* elevada; (b) *roughness* baixa

Este modelo permite-nos, então, calcular tanto a cor estimada como a direção para onde o raio deve seguir. Implementanto estes algoritmos num integrador simples permite-nos, então, obter uma imagem fotorealista.

2.5 World Lighting

Para obter resultados semelhantes ao Blender, reaproveitou-se as antigas luzes de ambiente como iluminação de mundo. Esta é, efetivamente, *environment lighting* com uma cor e intensidade fixa, modelada no Blender como a cor de fundo do mundo.

Sendo integrada no nosso modelo de classes de luzes, a sua implementação não afetou o resto do código, mas permitiu-nos obter resultados mais semelhantes ao Blender.

Capítulo 3

Resultados

De forma a desmonstrar os nossos resultados, foi utilizada a seguinte imagem como referência:

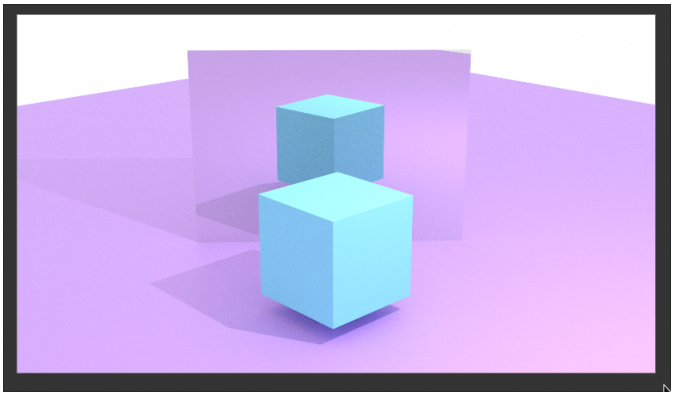

Figura 3.1: *Render* de referência feito no *blender*.

Assim sendo, foram obtidos os seguintes resultados com recurso ao nosso *raytracer*:

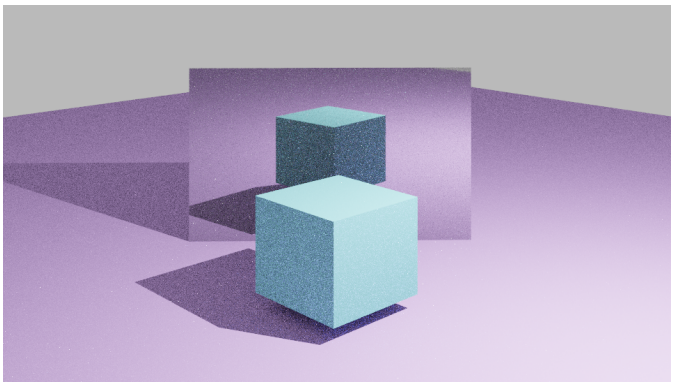

Figura 3.2: *Render* final do projeto.

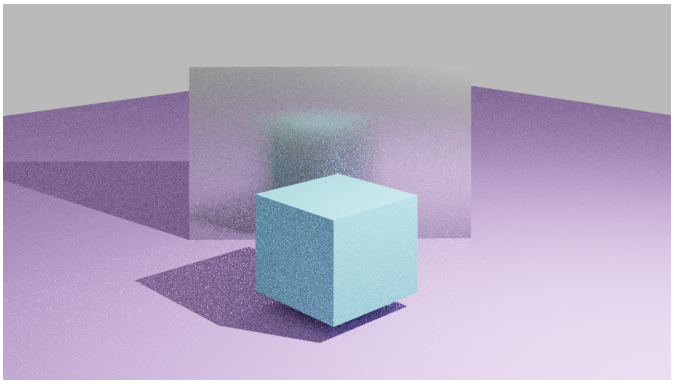

Figura 3.3: *Main scene* com *roughnesses* = 0.15.

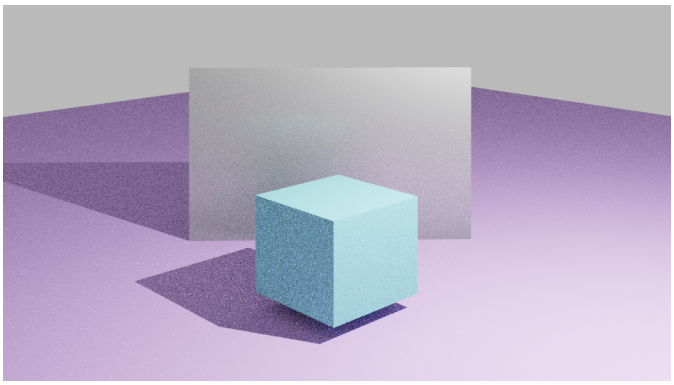

Figura 3.4: *Main scene* com *roughnesses* = 0.4.

De modo a analisar diferentes cenários de teste, criou-se uma *scene* nova que pode repesentar um ambiete durante a noite e com uma fonte de luz, como uma lampada, por exemplo.

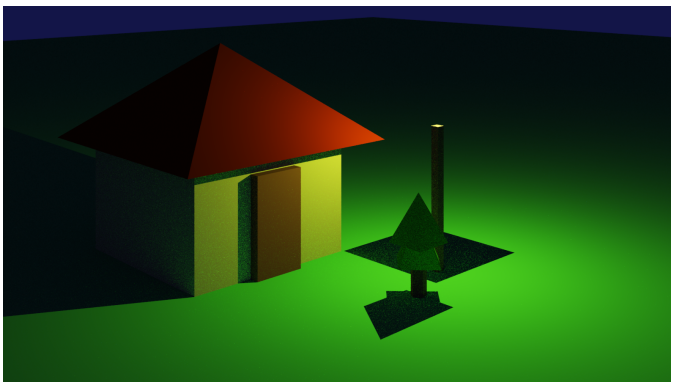

Figura 3.5: *Scene* secundária de uma casa durante a noite.

Capítulo 4

Conclusões e análise

O projeto de Visão e Iluminação, que envolveu a implementação de um *raytracer* em C++, permitiu a exploração dos princípios fundamentais da renderização de imagens por meio de raios. Ao longo do desenvolvimento, foram aplicados conceitos de física da luz, geometria e interações entre os objetos, resultando em imagens com efeitos de iluminação realistas.

A implementação do *raytracer* exigiu a compreensão e aplicação de algoritmos complexos, como o lançamento de raios primários, o cálculo de interseção com objetos e a determinação de sombras, reflexões e refrações. Além disso, foram utilizados modelos de iluminação para simular a aparência dos objetos com base nas propriedades dos materiais.

Os resultados obtidos foram analisados com base em diferentes configurações de cena e parâmetros de renderização. Foi possível observar a influência direta da iluminação nos objetos, bem como a capacidade do *raytracer* em simular reflexões e sombras de forma precisa.

Apesar das conquistas alcançadas, é importante destacar que a implementação de um *raytracer* é um campo de estudo vasto e em constante evolução. Existem inúmeras técnicas avançadas que podem ser exploradas para melhorar a qualidade das imagens geradas, como o uso de técnicas de *anti-aliasing*, *soft shadows*, mapeamento de texturas e efeitos de pós-processamento.

Em suma, este projeto foi uma oportunidade valiosa para adquirir conhecimentos sólidos sobre os princípios da renderização por raios e a sua aplicação prática. O *raytracer* implementado em C++ demonstrou a capacidade de simular de forma realista a interação entre a luz e os objetos, permitindo a criação de imagens visualmente atraentes. Com base nessa experiência, é possível afirmar que o estudo e a prática contínua nessa área abrirão portas para um maior domínio das técnicas de renderização e uma maior compreensão dos desafios enfrentados na criação de cenas e efeitos visuais de alta qualidade.

Bibliografia

[1] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *Acm Siggraph*, volume 2012, pages 1–7. vol. 2012, 2012.