Universidade do Minho
Escola de Engenharia

UC Visão e Iluminação II, Perfil de Computação Gráfica
Grupo Número 1
André R. A76387, Diogo N. A78957, Mariana C. A78824
Ano Letivo 2019/2020

Trabalho Prático 1 *Ray Tracing* Básico

Para tentar explicar toda a ideia por detrás deste primeiro trabalho prático, vão-se criar três tópicos centrais a este projeto – **Global**, **Grades** e **Vidros** - que definem as alterações feitas para ir de encontro com o pedido. Para isso, explica-se cada função criada, a forma como atuam em termos de código e qual a sua utilidade em termos de resultados visuais.

• Alterações Globais

Função	Execução da Função	Utilidade
`__closethit__radiance()`	Traçou-se um **shadow ray** com a **rayFlags** **OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT**, determinando-se o raio no primeiro hit. O ray foi lançado a partir da posição da câmara com a direção contrária do raio da luz.	Muda a cor do pixel dependendo da cor da textura. Caso não haja interceção, tenta-se tornar a cor mais escura de modo a simular a ideia de sombra.
`__miss__radiance()`	Uso de um **float3** de valores 1.0f de modo a conferir o branco ao fundo do modelo.	Confere a cor de fundo de todo o modelo.
`__closesthit__shadow()`	Muda o prd para a cor verde. No `__closethit__radiance()` verifica-se se tem a cor verde e, caso tenha, escurece o pixel.	A função é usada para quando a necessário acionar uma sombra.
`__miss__shadow()`	Muda o prd para a cor azul no caso de não existir nenhuma sombra.	Obriga que quando não é sombra, a cor vermelha não seja igual a 1, como sucede no `__closesthit__shadow()`.

- Alterações Grades

Função	Execução da Função	Utilidade
`__anyhit__phong_alphaTrans()`	Se o pixel das grades tiver a cor branca, então a cor do pixel é igual à cor da textura. Caso contrário, a interceção é ignorada.	Não executa qualquer desenho "entre as grades". Ou seja, fica uma transparência entre cada ferro que da grade faz parte.
`__anyhit__shadow_alphaTrans()`	Caso a cor da textura seja diferente de branco, a interceção é ignorada.	Cria todo o desenho para as sombras das grades.

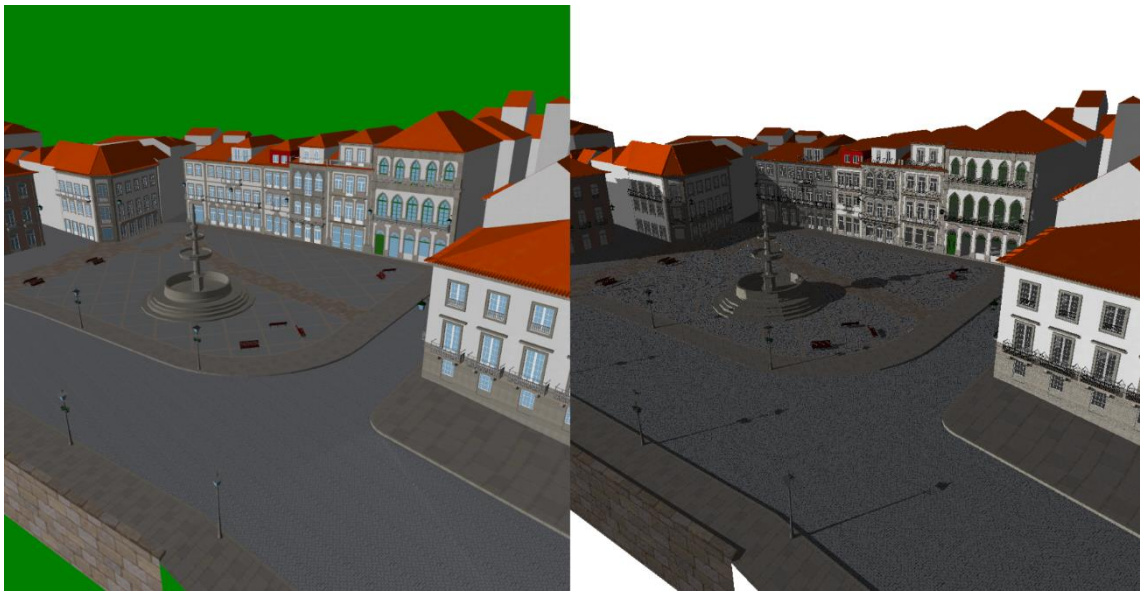
- Alterações Vidros

Função	Execução da Função	Utilidade
`__closesthit__phong_glass()`	Foi calculado a direção do raio de reflexão e de refração (é igual à direção de transmissão). Usando essas direções, foram traçados dois raios. Com as cores de cada raio, usou-se a fórmula de *Schlick* para as combinar e resultar na cor do pixel final.	Criação do efeito de reflexão e refração para os vidros.
`__closesthit__shadow_glass()`	Traça raio com a direção contrária da luz, resultando numa sombra interior ao vidro, conforme se explica melhor na utilidade.	Criação do efeito de sombra das grandes no interior do edifício do vidro em causa.

- Observação dos Resultados

Através da imagem abaixo anexada consegue-se comprovar que todo o código pensado para conseguir executar o *Ray Tracing* no Largo de Camões fez realmente sentido e que todas as alterações, tanto para as grades, como para os vidros, conferem um resultado muito mais real e preciso. Toda a transparência conferida ao vidro torna possível visualizar as sombras/reflexos que nele entram, tanto do próprio vidro como das grades que o cercam.

Na segunda imagem vê-se o cenário mais de longe e todas as sombras conferidas a todos os elementos nele presente.



Universidade do Minho
Escola de Engenharia

UC Visão e Iluminação II, Perfil de Computação Gráfica
Grupo Número 1
André R. A76387, Diogo N. A78957, Mariana C. A78824
Ano Letivo 2019/2020

Trabalho Prático 2 *Ambient Occlusion*

Neste segundo trabalho prático a ideia passou por aplicar *Ambient Occlusion* a duas bolas de materiais oposto – **Metal e Vidro**. Dessa forma e tendo em conta que o código pensado para o *Ambient Occlusion* dos dois materiais é **quase** o mesmo, explica-se o processo do *Ambient Occlusion* em si e aponta-se a diferença entre os dois.

• Execução do *Ambient Occlusion*

Para se implementar o *Ambient Occlusion* recorreu-se aos *slides* fornecidos pelo docente e aplicou-se a estratégia de *Sampling* - *Cosine Weighted Hemisphere Sampling*. Este processo de *Ambient Occlusion* é iterativo, ou seja, é executado/processado para cada raio:

1. Calcular os valores de $\boldsymbol{\beta, r, x, z}$ e $\boldsymbol{y}$

```
float r = sqrt(z1);                 // Cálculo do valor de r
float beta = 2 * M_PIf * z2;        // Cálculo do valor de Beta
p.x = r * sin(beta);                // Cálculo do valor de x
p.z = r * cos(beta);                // Cálculo do valor de z
p.y = sqrt(1-pow(r,2));             // Cálculo do valor de y
```

Para este cálculo tomou-se como referência dois números uniformes e aleatórios entre [0,1], de seu nome z1 e z2, de forma a se puder usar no cálculo dos valores anteriormente referidos.

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

2. Calcular a Tangente e a Bi Tangente

Para calcular a Tangente e Bi Tangente recorreu-se à normal. Caso c1 seja maior que c2, a Tangente vai ter as coordenadas (0, 0, 1). Se for menor, a Tangente vai ser igual a (0, 1, 0). A Bi Tangente acaba por ser o produto vetorial da tangente anterior com a normal.

```
float3 t;
float3 c1 = cross(nn, make_float3(0.0, 0.0, 1.0));
float3 c2 = cross(nn, make_float3(0.0, 1.0, 0.0));

if(length(c1) > length(c2)) t = c1;   // Caso seja maior
else                        t = c2;

normalize(t); float3 b = cross(t, nn);
```

3. Calcular o *Ray Direction*

Com todo o processo anteriormente calculado pode-se determinar a Direção do Raio e com ela aplicar finalmente o *Ambient Occlusion* por meio da função `optixTrace()`.

```
rayDir = p.x * b + p.y * nn + p.z * t;
```

É importante também referir que ao longo de todo o código, fez-se uma pequena alteração no que toca ao valor do prd:

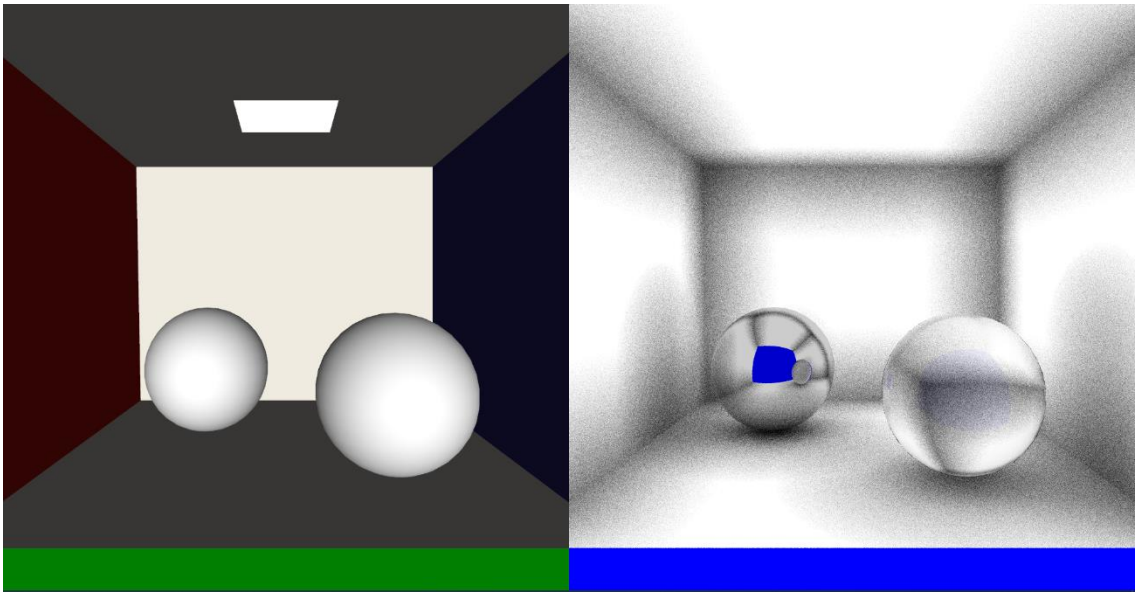
- Numa versão inicial e prematura, este valor consistia apenas num float3.
- Numa versão posterior, passou a consistir numa estrutura colorPRD que contém o valor da cor e do *seed* que deve ser tomado em conta ao longo de toda a execução do *Ambient Occlusion.*

```
colorPRD afterPRD;
    afterPRD.color = make_float3(1.0f);
    afterPRD.seed = prd.seed;
```

Outra coisa importante a referir é a diferença em termos do material Metal e Vidro. No Metal apenas se faz o uso da reflexão, tendo em conta a natureza do material e o efeito que se pretende criar em termos visuais. No Vidro usa-se a refração e a reflexão, tal como se analisou no primeiro Trabalho Prático.

- Observação dos Resultados

Através da imagem abaixo anexada consegue-se comprovar que todo o código pensado para conseguir executar o *Ambient Occlusion* nas duas bolas de diferentes materiais fez realmente diferença e permitiu distinguir na perfeição os diferentes materiais. A bola da esquerda apresenta um brilho totalmente característico do metal e isso nota-se também pelo reflexo que ele próprio acaba por transparecer.



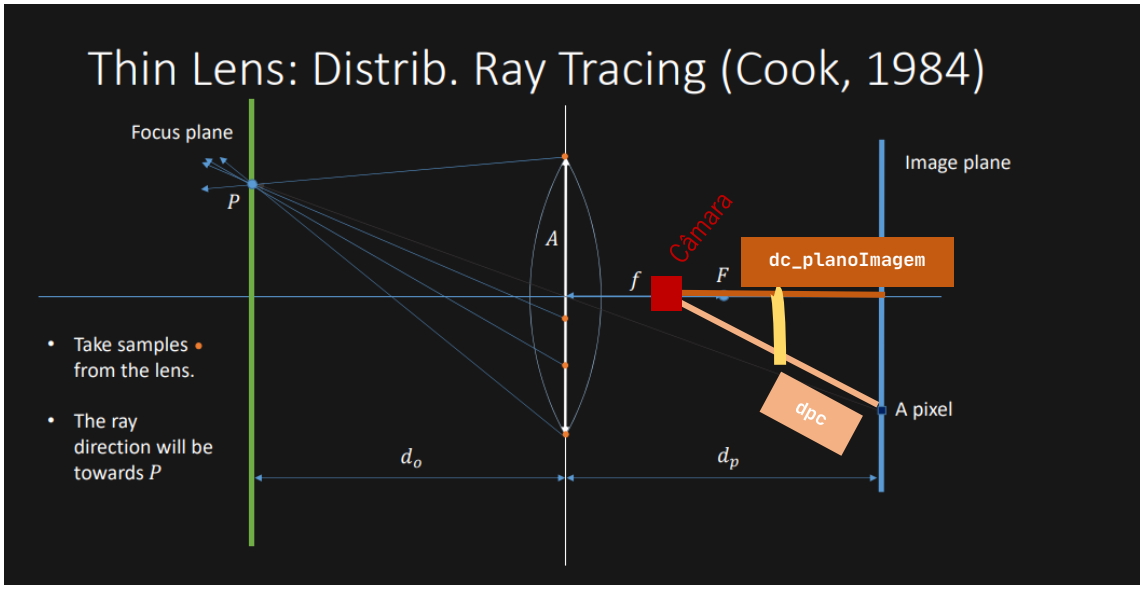
Universidade do Minho
Escola de Engenharia

UC Visão e Iluminação II, Perfil de Computação Gráfica
Grupo Número 1
André R. A76387, Diogo N. A78957, Mariana C. A78824
Ano Letivo 2019/2020

Trabalho Prático 3 *Depth of Field*

O terceiro e último trabalho prático foca-se em obter uma implementação do efeito *Depth of Field*, usando-se como meio de ilustração a cena de *Sponza*.

Recorrendo aos *slides* fornecidos pelo docente, toma-se maior atenção ao esquema de Cook de seu nome ***Distrib. Ray Tracing***, totalmente adaptado/modificado consoante a interpretação feita para proceder aos respetivos cálculos.



Para melhor se entender o esquema, considera-se o exemplo com a camara numa posição fixa. Claro que esta vai variar sempre consoante a posição da câmara, cuja variável tem de nome `camera.position`.

O objetivo basilar e que fica implícito pelo esquema acima modificado passa por determinar um conjunto de pontos que em conjunto permitem executar o efeito de *Depth of Field*.

1. Descobrir as coordenadas referentes ao ponto P (referenciado na imagem abaixo), ou seja, o *Focus Plane Point*

- Calculou-se as coordenadas do A pixel e consequentemente a direção desse pixel à câmara (variáveis `pixel` e `dir_pixel_camera`, correspondentemente);
- O ângulo entre o vetor desse pixel ao centro da câmara e do vetor do centro da câmara ao centro do *Image Plane*;
  Este ângulo encontra-se representado pela cor amarela (variável **`angle_dcp_dcpi`**);
- Cálculo da distância entre o pixel ao centro da câmara (variável `dpc`);
- Cálculo da distância entre o centro da câmara e o *Image Plane* (variável `dc_planoImagem`);
- Com a distância do centro da câmara ao *Image Plane* e o d_p (distância da lente ao *Image Plane*) já definida no código fornecido pelo docente, conseguiu-se achar as coordenadas corretas da lente (variável `centroLente`);
- Calculou-se a distância do pixel ao centro da lente (variável `dpl`);
- Achou-se o fator de proporção `m` através da divisão da distância d_o (distancia da lente ao *Focus Plane*) e d_p (distância da lente ao *Image Plane*);
- Calculou-se a distância do centro da lente ao ponto `P` (variável `dlP`).

Com as coordenadas do pixel, as coordenadas do centro da lente, a distância do pixel ao centro e a distância do centro da lente ao ponto P torna-se perfeitamente possível calcular/descobrir as coordenadas do ponto P:

```
float3 P = pixel + normalize(centroLente - pixel) * (dpl + dlP);
```

2. Cálculo da *Ray Direction* lançado ao ponto P

- Calculou-se o ponto de origem para se lançar o raio, o ponto ou pontos *sampling*, através da seguinte fórmula:

```
float3 ponto_sampling = centroLente + raio * (perpendicular
* cos (angulo) + outra_perp * sin(angulo));
```

- Com o ponto *sampling* (variável **`ponto_sampling`**) e o ponto P, consegue-se calcular a direção do raio da seguinte forma:

```
float3 direcao = normalize(P - ponto_sampling);
```

3. Lançar o raio

Com todos os cálculos feitos anteriormente, passou-se ao lançamento dos raios através da função já muito usada anteriormente – **`optixTrace()`** - com o ponto de origem (**`ponto_sampling`**) e a direção (**`direcao`**), calculado no ponto/passo anterior.

• Observação dos Resultados

As duas imagens abaixo anexadas permitem observar o fenómeno do *Depth of Field* na perfeição para dois objetos/cenários diferentes. Na primeira imagem foca-se o fundo e desfoca-se a jarra, enquanto que na segunda segue-se o processo oposto, focando-se a jarra e fazendo uma espécie de modo retrato para o fundo.


settings
4
Squared rays per pixel
0.43
Aperture
1.800
Lens distance
24.045
Distance to focal plane