ISEL – LERCM

Matemática para Computação Gráfica

Verão de 2013-2014

Guia Python para o projeto (*Revision* : 1.5)

Parte 5 - Ray Tracer

1 Objectivo

Desenvolver, em Python 3, o módulo `ray_tracer_xxxxx`, no ficheiro `ray_tracer_xxxxx.py`, com a classe `RayTracer`.

A classe `RayTracer` implementa um ray tracer. Permite obter renderizações de uma cena 3D, isto é, obter imagens da cena.

1.1 Observações

- `xxxxx` no nome do ficheiro deve ser substituído pelo número de aluno do autor do trabalho.

2 Testes

O conjunto de todos os exemplos fornecidos ao longo desta especificação constitui uma bateria de testes à implementação da classe `RayTracer`.

Esta bateria de testes deve ser executada sobre a implementação desenvolvida.

Caso ocorra **algum erro** durante a execução da bateria de testes, **a implementação desenvolvida não é considerada válida**.

3 Especificação da classe RayTracer

3.1 Nome

`RayTracer`

3.2 Dados

Atributos de dados – membros	Tipo
`lista_faces`	`list`
`lista_luzes`	`list`
`camara`	`Camara`
`cor_fundo`	`CorRGB`

Observação: para além destes atributos, pode acrescentar quaisquer atributos auxiliares que sejam úteis à sua implementação.

3.3 Operações

Atributos funções – métodos
`__init__`
`__str__`
`renderiza`

Observação: para além destes métodos, pode acrescentar quaisquer métodos auxiliares que sejam úteis à sua implementação.

3.4 Atributos de dados – membros

O atributo `lista_faces` é uma lista de objetos do tipo `FaceTriangular` que caracterizam o conteúdo da cena a renderizar.

O atributo `lista_luzes` é uma lista de objetos do tipo `Luz` que caracterizam as fontes de iluminação da cena.

O atributo `camara` é um objeto do tipo `Camara` que caracteriza o ponto de vista da renderização assim como as características da imagem resultante.

O atributo `cor_fundo` é um objeto do tipo `CorRGB` que caracteriza a cor dos pixels usada quando os raios de visão (com origem no foco da câmara e destino nos pontos de projeção da câmara) não intercetam qualquer das faces triangulares da cena.

3.5 Atributos funções – métodos

3.5.1 Construtor

Método `__init__`	Argumentos	Tipo
	`self`	`RayTracer`
	`lista_faces`	`list`
	`lista_luzes`	`list`
	`camara`	`Camara`
	`cor_fundo`	`CorRGB`

É da responsabilidade do utilizador desta classe fornecer os argumentos `lista_faces`, `lista_luzes`, `camara` e `cor_fundo` do tipo `list`, `list`, `Camara` e `CorRGB`, respetivamente. Para além disso, todos os elementos da lista `lista_faces` têm de ser do tipo `FaceTriangular` e todos os elementos da lista `lista_luzes` têm que ser do tipo `Luz`.

O construtor inicializa os atributos de dados com os argumentos correspondentes.

Exemplo

```
# teste ao construtor
# teste ao construtor - cor da face
verde  = CorRGB(0.0, 0.3, 0.0)
brilho = 100.0
cor    = CorPhong(verde, verde, verde, brilho)
# teste ao construtor - face
p1          = Ponto3D(0.0, 0.0, 0.0)
p2          = Ponto3D(1.0, 0.0, 0.0)
p3          = Ponto3D(0.0, 1.0, 0.0)
face        = FaceTriangular(p1, p2, p3, cor)
lista_faces = [face]
# teste ao construtor - luz
branco      = CorRGB(1.0, 1.0, 1.0)
luz_posicao = Ponto3D(1.0, 0.0, 2.0)
luz         =  Luz(luz_posicao, branco, branco, branco)
lista_luzes = [luz]
# teste ao construtor - camara
camara_posicao                = Ponto3D(0.0, 0.0, 2.0)
olhar_para                    = Ponto3D(0.0, 0.0, 0.0)
vertical                      = Vetor3D(0.0, 1.0, 0.0)
distancia_olho_plano_projecao = 1.5
largura_retangulo_projecao    = 2.0
altura_retangulo_projecao     = 2.0
resolucao_horizontal          = 50
resolucao_vertical            = 50
camara                        = Camara(camara_posicao,
                                       olhar_para,
                                       vertical,
                                       distancia_olho_plano_projecao,
                                       largura_retangulo_projecao,
                                       altura_retangulo_projecao,
                                       resolucao_horizontal,
                                       resolucao_vertical)
# teste ao construtor - cor de fundo
cor_fundo = CorRGB(0.0, 0.0, 0.2)
# teste ao construtor - ray tracer
ray_tracer = RayTracer(lista_faces, lista_luzes, camara, cor_fundo)
```

3.5.2 `__str__`

Método `__str__`	Argumentos	Tipo
	`self`	`RayTracer`
	Retorno	`string`

É a função usada pelo interpretador de Python para converter os objetos

desta classe em `string`'s. É usada quando se chama as funções built-in `str` e `print`, por exemplo.

Retorna uma `string` constituída por `RayTracer(lista1, lista2, camara, fundo)`, onde `lista1` é a string que resulta da conversão para string de cada um dos elementos do atributo `lista_faces`, `lista2` é a string que resulta da conversão para string de cada um dos elementos do atributo `lista_luzes`, `camara` é a string que resulta da conversão para string do atributo `camara` e `fundo` é a string que resulta da conversão para string do atributo `cor_fundo`. Por razões de legibilidade, acrescenta-se um fim de linha `"\n"` a cada uma destas strings.

Exemplo

```
# teste a __str__
print(ray_tracer)
```

Output

```
RayTracer(FaceTriangular(Plano(Ponto3D(0.0,0.0,0.0), Ponto3D(1.0,0.0,0.0), Ponto3D
     (0.0,1.0,0.0), Vetor3D(0.0, -0.0, 1.0)), CorPhong(0 76 0, 0 76 0, 0 76 0, 100.0))
,
Luz(Ponto3D(1.0,0.0,2.0), 255 255 255, 255 255 255, 255 255 255)
,
Camara(Ponto3D(0.0,0.0,2.0),
Ponto3D(0.0,0.0,0.0),
Vetor3D(0.0, 1.0, 0.0),
1.5,
2.0,
2.0,
50,
50,
Vetor3D(1.0, -0.0, 0.0),
Vetor3D(0.0, 1.0, 0.0),
Vetor3D(0.0, 0.0, -1.0),
0.04,
0.04,
-0.98,
0.98,
1.5,
Matriz(4, 4)
1.0 0.0 0.0 0.0
-0.0 1.0 0.0 0.0
0.0 0.0 -1.0 2.0
0.0 0.0 0.0 1.0
),
0 0 51
)
```

3.5.3 `renderiza`

Método `renderiza`	Argumentos	Tipo
	`self`	`RayTracer`
	Retorno	`Imagem`

Retorna um objeto da classe `Imagem` com a renderização obtida.
O processo de renderização pode ser decomposto nas seguintes etapas:

Corpo Principal

1. Criar uma imagem com o número de linhas e de colunas iguais à resolução vertical e horizontal da câmara, respetivamente. Esta imagem armazenará a cor dos pixels, obtida pelo processo de ray tracing, e será retornada como resultado da renderização.
2. Para cada ponto do plano de projeção da câmara (duplo ciclo `for` nas linhas e colunas do plano de projeção):

   (Sugere-se o `print` do número da linha que está a ser processada bem como do número total de linhas.)

   (a) Obter a posição da câmara.
   (b) Obter a posição do ponto do plano de projeção, no sistema de coordenadas global.
   (c) Criar uma reta com origem na posição da câmara (o foco) e com destino no ponto do plano de projeção.
   (d) Obter a cor vista pelo raio de visão definido pela reta criada no ponto anterior.
   (e) Armazenar a cor na imagem criada inicialmente
3. Depois de processar todos os pontos do plano de projeção, retornar a imagem criada inicialmente.

Cor vista por raio

1. Obter a face intercetada mais próxima (assim como o ponto de interceção).
2. Caso não exista retornar a cor de fundo.
3. Caso exista, obter a direção do olho, a cor da face no ponto de interceção e retornar a cor obtida.

Face intrecetada mais próxima

Para cada face da lista de faces (ciclo `for` na lista de faces) obter a interceção com o raio de visão e armazenar a interceção a que corresponde o menor parâmetro t da interceção (a face mais próxima).

Cor da face

Para cada luz da lista de luzes (ciclo `for` na lista de luzes) obter a cor da face gerada pela luz em causa.

Para obter a cor da face gerada por uma luz construir uma reta com origem no ponto da face e destino na posição da luz para averiguar se essa reta interceta alguma face da cena. Se intreceta obter a cor Phong em sombra, senão inteceta obter a cor Phong com todas as componentes.

Retornar a soma das contribuições de todas as luzes.

Para implementação deste processo sugere-se o desenvolvimento dos métodos seguintes.

Método `get_face_intercetada_mais_proxima`	Argumentos	Tipo
	`self`	`RayTracer`
	`raio`	`Reta`
	Retorno	`list`

Caso a reta `raio` intercete alguma das faces da cena, da origem para a frente, retorna a lista:

```
[True, ponto_intercecao, t, face]
```

Caso não intercete nenhuma das faces da cena, da origem para a frente, retorna a lista:

```
[False, None, None, None]
```

Método `get_cor_face`	Argumentos	Tipo
	`self`	`RayTracer`
	`face`	`FaceTriangular`
	`ponto_intercecao`	`Ponto3D`
	`direcao_olho`	`Vetor3D`
	Retorno	`CorRGB`

Retorna a cor RGB num ponto de uma face triangular resultante da soma das contribuições de todas as luzes em cena mesmo quando o ponto está em sombra (de uma ou mais luzes).

Método `get_cor_vista_por_raio`	Argumentos	Tipo
	`self`	`RayTracer`
	`raio`	`Reta`
	Retorno	`CorRGB`

Retorna a cor da face intercetada pelo `raio` mais próxima da origem do raio (no sentido do destino do raio).

Exemplo

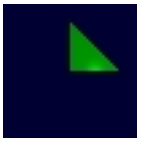
```
# teste a renderiza
imagem = ray_tracer.renderiza()
imagem.guardar_como_ppm("teste1.ppm")
```

Output

```
linha = 1 de 50
linha = 2 de 50
linha = 3 de 50
linha = 4 de 50
linha = 5 de 50
linha = 6 de 50
linha = 7 de 50
linha = 8 de 50
linha = 9 de 50
linha = 10 de 50
linha = 11 de 50
linha = 12 de 50
linha = 13 de 50
linha = 14 de 50
linha = 15 de 50
linha = 16 de 50
linha = 17 de 50
linha = 18 de 50
linha = 19 de 50
linha = 20 de 50
linha = 21 de 50
linha = 22 de 50
linha = 23 de 50
linha = 24 de 50
linha = 25 de 50
linha = 26 de 50
linha = 27 de 50
linha = 28 de 50
linha = 29 de 50
linha = 30 de 50
linha = 31 de 50
linha = 32 de 50
linha = 33 de 50
linha = 34 de 50
linha = 35 de 50
linha = 36 de 50
linha = 37 de 50
linha = 38 de 50
linha = 39 de 50
linha = 40 de 50
linha = 41 de 50
linha = 42 de 50
```

```
linha = 43 de 50
linha = 44 de 50
linha = 45 de 50
linha = 46 de 50
linha = 47 de 50
linha = 48 de 50
linha = 49 de 50
linha = 50 de 50
```

Figura 1: A imagem no ficheiro `teste1.ppm`.



4 Teste de referência

O teste descrito nesta secção é o teste principal à implementação do ray tracer. Segue-se a descrição da cena que permite efectuar este teste.

4.1 Câmara

Posição $(0.0, 0.0, 2.5)$

Olhar para $(0.0, 0.0, 0.0)$

Vertical $[0.0, 1.0, 0.0]^\top$

Distância olho plano de projeção 0.5

Largura retângulo projeção 8.0

Altura retângulo projeção 4.0

Resolução horizontal 600

Resolução vertical 300

4.2 Triângulos cinzentos

Coeficiente de reflexão ambiente $(0.5, 0.5, 0.5)$

Coeficiente de reflexão difusa $(0.75, 0.75, 0.75)$

Coeficiente de reflexão especular $(0.0, 0.0, 0.0)$

Concentração do brilho 1.0

Vértices do triângulo 1 (de baixo)

$(16.0, -5.1, 0.0), (150.0, -5.1, -20.0), (-16.0, -5.1, 0.0)$

Vértices do triângulo 2 (de cima)

$(-16.0, -5.1, 0.0), (150.0, -5.1, -20.0), (-16.0, 5.1, 0.0)$

4.3 Triângulos vermelhos

Coeficiente de reflexão ambiente $(1.0, 0.0, 0.0)$

Coeficiente de reflexão difusa $(1.0, 0.0, 0.0)$

Coeficiente de reflexão especular $(1.0, 0.0, 0.0)$

Concentração do brilho 1000.0

Vértices dos triângulos da letra M

$(-16.0, -5.0, 0.0), (-12.0, -5.0, 0.0), (-16.0, 5.0, 0.0)$
$(-16.0, 5.0, 0.0), (-11.0, 0.0, 0.0), (-6.0, 5.0, 0.0)$
$(-10.0, -5.0, 0.0), (-6.0, -5.0, 0.0), (-6.0, 5.0, 0.0)$

Vértices dos triângulos da letra C

$(-5.0, -2.0, 0.0), (5.0, 5.0, 0.0), (-5.0, 3.0, 0.0)$
$(-5.0, -3.0, 0.0), (5.0, -5.0, 0.0), (-5.0, 2.0, 0.0)$

Vértices dos triângulos da letra G

$(6.0, -2.0, 0.0), (16.0, 5.0, 0.0), (6.0, 5.0, 0.0)$
$(6.0, -5.0, 0.0), (16.0, -5.0, 0.0), (6.0, 2.0, 0.0)$
$(16.0, -5.0, 0.0), (16.0, 0.0, 0.0), (12.0, 0.0, 0.0)$

4.4 Luzes

Intensidade ambiente $(0.1, 0.1, 0.1)$

Intensidade difusa $(0.75, 0.75, 0.75)$

Intensidade especular $(0.75, 0.75, 0.75)$

Posição luz 1 $(16.0, 2.0, 2.0)$

Posição luz 2 $(-16.0, -2.0, 2.0)$

4.5 Cor de fundo

Preto $(0.0, 0.0, 0.0)$

4.6 Resultado

Exemplo

```
# teste de referência

# falta aqui a definição da câmara

# falta aqui a definição da lista de faces

# falta aqui a definição da cor de fundo

# falta aqui a definição da lista de luzes

# ray tracer
ray_tracer = RayTracer(lista_faces, lista_luzes, camara, cor_fundo)

# renderização
imagem = ray_tracer.renderiza()

# ficheiro com a renderização
imagem.guardar_como_ppm("teste2.ppm")
```

Output

```
linha = 1 de 300
linha = 2 de 300
linha = 3 de 300
linha = 4 de 300
linha = 5 de 300
linha = 6 de 300
linha = 7 de 300
linha = 8 de 300
linha = 9 de 300
linha = 10 de 300
.
.
.
```

```
(linhas 11 a 289 removidas, apenas neste pdf,
porque a listagem fica demasiado longa)
.
.
.
linha = 290 de 300
linha = 291 de 300
linha = 292 de 300
linha = 293 de 300
linha = 294 de 300
linha = 295 de 300
linha = 296 de 300
linha = 297 de 300
linha = 298 de 300
linha = 299 de 300
linha = 300 de 300
```

Figura 2: A imagem no ficheiro `teste2.ppm`.

5 Código

ray_tracer_xxxxx.py

```
from ponto_xxxxx     import Ponto3D
from cor_rgb_xxxxx   import CorRGB
from cor_phong_xxxxx import CorPhong
from face_xxxxx      import FaceTriangular
from luz_xxxxx       import Luz
from vetor_xxxxx     import Vetor3D
from camara_xxxxx    import Camara
from reta_xxxxx      import Reta
from imagem_xxxxx    import Imagem

class RayTracer:

    # falta aqui o construtor

    # falta aqui o método __str__

    # falta aqui o método renderiza

# testes
if __name__ == "__main__":

    # teste ao construtor
    # teste ao construtor - cor da face
    verde  = CorRGB(0.0, 0.3, 0.0)
    brilho = 100.0
    cor    = CorPhong(verde, verde, verde, brilho)
    # teste ao construtor - face
    p1          = Ponto3D(0.0, 0.0, 0.0)
    p2          = Ponto3D(1.0, 0.0, 0.0)
    p3          = Ponto3D(0.0, 1.0, 0.0)
    face        = FaceTriangular(p1, p2, p3, cor)
    lista_faces = [face]
    # teste ao construtor - luz
    branco      = CorRGB(1.0, 1.0, 1.0)
    luz_posicao = Ponto3D(1.0, 0.0, 2.0)
    luz         =  Luz(luz_posicao, branco, branco, branco)
    lista_luzes = [luz]
    # teste ao construtor - camara
    camara_posicao                = Ponto3D(0.0, 0.0, 2.0)
    olhar_para                    = Ponto3D(0.0, 0.0, 0.0)
    vertical                      = Vetor3D(0.0, 1.0, 0.0)
    distancia_olho_plano_projecao = 1.5
    largura_retangulo_projecao    = 2.0
    altura_retangulo_projecao     = 2.0
    resolucao_horizontal          = 50
    resolucao_vertical            = 50
    camara                        = Camara(camara_posicao,
                                           olhar_para,
                                           vertical,
                                           distancia_olho_plano_projecao,
                                           largura_retangulo_projecao,
                                           altura_retangulo_projecao,
                                           resolucao_horizontal,
                                           resolucao_vertical)
    # teste ao construtor - cor de fundo
    cor_fundo = CorRGB(0.0, 0.0, 0.2)
```

```
# teste ao construtor - ray tracer
ray_tracer = RayTracer(lista_faces, lista_luzes, camara, cor_fundo)

# teste a __str__
print(ray_tracer)

# teste a renderiza
imagem = ray_tracer.renderiza()
imagem.guardar_como_ppm("teste1.ppm")

# teste de referência

# falta aqui a definição da câmara

# falta aqui a definição da lista de faces

# falta aqui a definição da cor de fundo

# falta aqui a definição da lista de luzes

# ray tracer
ray_tracer = RayTracer(lista_faces, lista_luzes, camara, cor_fundo)

# renderização
imagem = ray_tracer.renderiza()

# ficheiro com a renderização
imagem.guardar_como_ppm("teste2.ppm")
```