Trabalho Prático 3
Ray Tracing

Arthur Souto Lima

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

arthursl@ufmg.br

1. Instruções

1.1. Dependências

O programa foi implementado em Python 3.8. Dado esforço computacional das operações de raytracing, escolheu-se utilizar o interpretador PyPy3[1] ao invés do usual CPython, o que significou uma melhora de cerca de cinco vezes nos tempos de renderização. Essa alternativa é facilmente instalada no Linux, via *apt-get install* no Ubuntu, por exemplo. Além disso, para a geração de imagens animadas, no caso GIFs, foi necessária uma biblioteca externa, ImageIO [2] e algumas configurações desta. Os comandos pip para instalar essa biblioteca e suas configurações externas estão fornecidas juntamente com o projeto, no arquivo *config.sh*, podendo ser executado pelo *bash*.

1.2. Entradas

O programa recebe alguns argumentos em sua entrada (o argumento *-h* traz um detalhamento destes também)

```
pypy3 main.py [entrada] [saida]

pypy3 main.py [entrada] [saida] [-t comprimento altura] [-a samplesAA]
```

Descrição:

- **entrada**: nome (ou caminho) do arquivo de entrada
- **saida**: nome (ou caminho) do arquivo de saída
- **-t comprimento altura**: opcional, comprimento e altura da imagem de saída. Padrão: 800x600
- **-a samplesAA**: opcional, quantidade de samples que serão feitos por cada pixel. Padrão: 6 samples

Exemplos:

```
pypy3 main.py CG_examples/test1.in out.ppm

pypy3 main.py examples/gif1.in outs/out.ppm -t 1280 720

pypy3 main.py CG_examples/test5.in CG_examples/out5.ppm -t 1980 1080 -a 15
```

[1] https://www.pypy.org/
[2] https://imageio.github.io/

1.3. Breve demonstração

Uma breve demonstração de algumas imagens geradas pelo projeto podem ser conferidas no álbum disponível no Imgur:

`https://imgur.com/a/tyS1ZK2`

Na legenda de cada imagem está o arquivo de entrada (ou o caminho relativo dele) que a produziu, bem como algum comentário acerca da funcionalidade a ser destacada na referida imagem. Convém ressaltar que, embora a saída do programa seja em ppm, foi necessária a conversão para PNG ou para JPEG a fim de possibilitar o upload para o Imgur. Tal operação foi feita com auxílio da biblioteca ImageIO, no código do arquivo *converter.py*.

2. Implementação

2.1. Concepção

O projeto objetiva implementar um programa que gere imagens utilizando a técnica de raytracing a partir de uma cena descrita num arquivo de entrada, seguindo o modelo corrente na especificação. Para a computação da imagem, baseou-se na série de vídeos de [Ravindran 2020], no qual criou-se um raytracer usando Python. Contudo, fez-se necessária a adição de algumas funcionalidades como, por exemplo, a refração, as quais foram obtidas, principalmente, da adaptação do explicado em [Shirley 2020] e em [Bikker 2004]. Além disso, essas duas fontes também contribuíram com ideias para a implementação da reflexão difusa e do anti-aliasing.

2.2. Implementação

A ideia do algoritmo básica foi vista em aula: para cada pixel da imagem, gera-se um raio da câmera passando por esse pixel, encontre o primeiro objeto que o raio acerta, determine a cor no ponto de interseção (que pode depender de sombreamento, reflexão e até mesmo refração) e desenhe esta cor.

2.3. Funcionalidades Adicionais

Como indicado pela especificação, dever-se-ia implementar funcionalidades além das básicas. Aqui estão listadas as funcionalidades escolhidas no projeto:

- **Anti-Aliasing**: com apenas um raio por cada pixel, há uma grande quantidade de serrilhados. A fim de mitigá-los, faz-se um super-sampling, com múltiplos raios para cada pixel, randomicamente em torno do raio principal e posteriormente tiramos a média entre as cores obtidas por cada um para uma imagem mais bem definida. Seguiu-se a proposta de [Shirley 2020].
- **Reflexão Difusa**: seguindo o modelo implementado por [Bikker 2004], foi feita uma implementação da reflexão difusa, numa ideia semelhante ao super-sampling do anti-aliasing, para dar ideia de borramento na reflexão.
- **Imagens em movimento**: uma forma simples de gerar uma simples imagem em movimento é renderizar cada frame e concatená-los numa GIF através da biblioteca ImageIO. O grande problema foi adaptar a entrada para permitir essa funcionalidade sem afetar o funcionamento normal do programa.

- **Lei de Beer-Lambert**: essa lei diz que a luz refratada é levemente afetada pela cor do objeto que está refratando. A lei é mais complexa e possui mais detalhes, mas para o nosso contexto, bastou isso juntamente com uma pequena computação durante o cálculo do raio refratado e sua cor, como visto em [Bikker 2004].

2.3.1. Reflexão Difusa

Seguimos o proposto em [Bikker 2004], de gerar vários raios para o raio refletido e tirar a média para dar uma ideia de borramento da reflexão, indicando uma irregularidade do acabamento da superfície. Para propiciar a entrada pelo arquivo dessa propriedade, deve-se adicionar um oitavo parâmetro entre 0 e 1 na descrição de cada acabamento. Por padrão esse parâmetro é zero, para não afetar os exemplos fornecidos.

Foi utilizada uma estratégia vista também em [Bikker 2004] para dar um pouco de celeridade a esse processo, já que um raio refletido, no nosso caso, geraria dez outros raios que poderiam refletir em outra superfície e gerar outros dez raios, causando uma cascata imensa recursivamente. A ideia era que a cada nível que se descia na recursão, gera-se menos raios para a reflexão, até o limite de um. Mesmo porque o borramento de um raio refletido, por exemplo, na quinta camada de recursão não influencia tanto o da primeira camada, precisamos realmente é da sua cor, se tanto, e não precisamos tanto do seu borramento.

2.3.2. Imagens em Movimento

A solução encontrada para permitir a entrada de uma descrição de cena em movimento, num único arquivo, que seguisse os moldes previstos, foi fazer uma pequena mudança no início, onde se descreve a câmera. Ao invés de descrever as três coordenadas da posição da câmera, podemos descrever a quantidade de estados desta, seguido da quantidade de segundos da animação. Isso facilita diferenciar a leitura dos dados da câmera entre se teremos uma imagem estática ou dinâmica. Após essa descrição, há uma listagem dos tempos (em segundos) de cada estado. Finalmente, é enumerado cada estado da câmera nos moldes da câmera estática. Para fazer a animação, é feita uma interpolação linear entre cada um dos estados proporcionalmente à quantidade de frames em cada transição. Decidiu-se por 5 FPS para evitar um tempo muito longo de renderização, já que, por exemplo, uma animação de 6 segundos teria 30 frames, ou seja, 30 imagens para serem geradas.

References

Bikker, J. (2004). Raytracing topics techniques. [3]. Acesso em: 30 out 2020.

Ravindran, A. (2020). Building a ray tracer in python - tutorial series. [4]. Acesso em: 25 out 2020.

Shirley, P. (2020). Ray tracing in one weekend. [5]. Acesso em: 30 out 2020.

[3] https://www.flipcode.com/archives/Raytracing_Topics_Techniques-Part_1_Introduction.shtml

[4] https://www.youtube.com/playlist?list=PL8ENypDVcs3H-TxOXOzwDyCm5f2fGXlIS

[5] https://raytracing.github.io/books/RayTracingInOneWeekend.html