Universidade Federal de Mato Grosso do Sul
Faculdade de Computação

Computação Gráfica 2019

Trabalho – Parte 4

Paulo A. Pagliosa

O objetivo principal da **Parte 4** é completar a implementação de um traçador de raios simples, conforme explicado em sala.

1 Passo 1: Código-fonte e geração do executável

O código-fonte específico para este exercício está no diretório `cg/p4` do repositório `https://git.facom.ufms.br/pagliosa/cg.git`. O código comum a todas as partes está no diretório `cg/common` do repositório.

Após atualizar o código da parte comum e baixar o código da **Parte 4** em seu diretório de trabalho, `cg` conterá os diretórios `common`, `p0`, `p1`, `p2`, `p3` e `p4`. O conteúdo do diretório `p4` será:

```
assets
  meshes
    bunny.obj
    f-16.obj
  shaders
    p4.fs
    p4.vs
build
  vs2019
    imgui.ini
    p4.sln
    p4.vcxproj
    p4.vcxproj.filters
Assets.cpp
Assets.h
Camera.cpp
Camera.h
Component.h
GLRenderer.cpp
GLRenderer.h
Intersection.h
Light.h
Main.cpp
Material.h
P4.cpp
P4.h
Primitive.cpp
```

Primitive.h
RayTracer.cpp
RayTracer.h
Renderer.cpp
Renderer.h
Scene.h
SceneEditor.cpp
SceneEditor.h
SceneNode.h
SceneObject.cpp
SceneObject.h
Transform.cpp
Transform.h

O diretório `assets/meshes` foi copiado da **Parte 3** e vem com exemplos de arquivos de malhas de triângulos no formato OBJ Wavefront. Como no exercício anterior, o diretório `assets/shaders` contém arquivos com o código GLSL dos *shaders* de vértice e fragmento e `build/vs2019` o projeto do Visual Studio 2019. Os arquivos listados em verde foram copiados de `p3` e não precisam ser alterados. Desses, o único modificado foi `Material.h` (agora, um material tem uma propriedade adicional: a cor de reflexão especular para a iluminação indireta). Os arquivos listados em azul também foram copiados de `p3`, mas deverão ser substituídos com sua implementação da **Parte 3**. Desses, o único modificado foi `Primitive.h`. A definição e implementação da classe de janela gráfica da aplicação são codificadas, respectivamente, em `P4.h` e `P4.cpp`. Além desses, você deverá completar, em `Primitive.cpp`, o método de interseção com um raio, bem como a implementação do traçador de raios em `RayTracer.h` e `RayTracer.cpp`, conforme detalhado em sala.

No Visual Studio 2019, abra o arquivo `build/vs2019/p4.sln` e construa o arquivo executável. Você notará, entre outros gerados pelo Visual Studio, o arquivo `p4.exe` no diretório `p4`. A execução do programa exibe em sua tela uma janela gráfica cuja interface com o usuário é a mesma do exercício anterior, como mostrado na Figura 1.

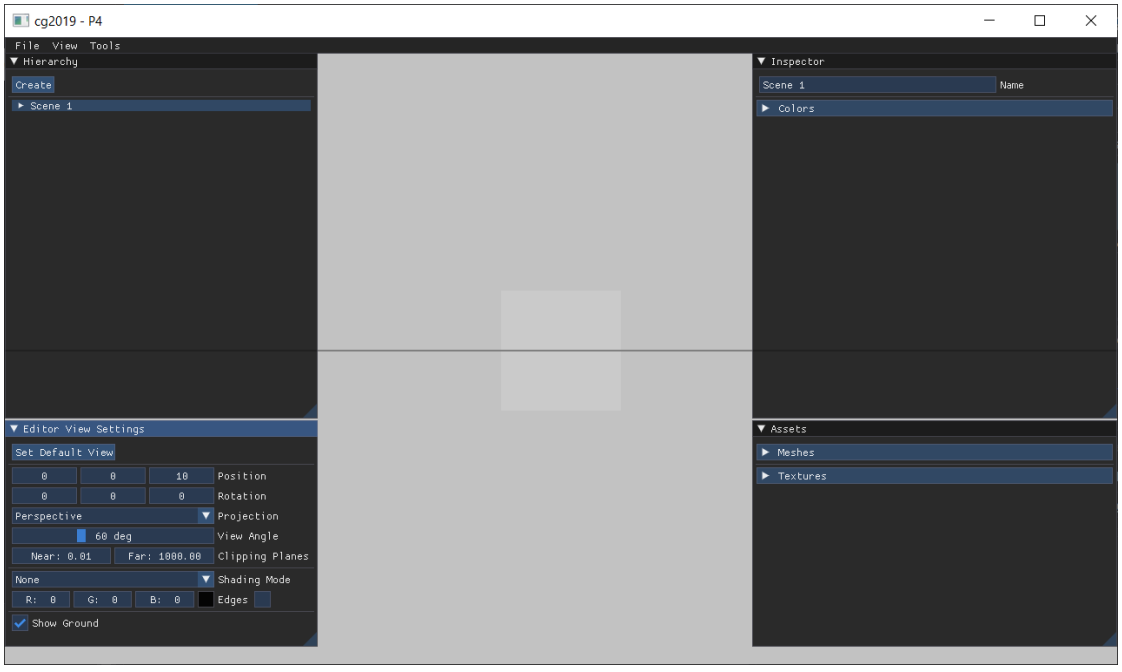

Figura 1: Programa `p4.exe` (no Windows).

2 Passo 2: Tarefas

Antes de começar as atividades específicas da **Parte 4** do trabalho, você deve primeiro integrar o código que você desenvolveu em todos os exercícios anteriores com o código fornecido para este exercício, ou seja, nesta última parte, seu programa deve estar **completo** de acordo com as especificações das demais partes do trabalho.

Em seguida, complete o código em `RayTracer.cpp` a fim de que o traçador de raios funcione. Para tal, é necessário implementar o método de interseção de um raio com a cena e os métodos de tonalização de um ponto, de acordo com o modelo de iluminação global do algoritmo (como explicado em sala, o único efeito de iluminação indireta a ser tratado é a reflexão especular).

As atividades dessa parte do trabalho consistem em:

A1 Completar a implementação do método `RayTracer::intersect()` responsável pela interseção de um raio com a cena. Este deve visitar cada primitivo da cena e determinar a interseção do raio com o primitivo. O primitivo interceptado pelo raio, se houver, é aquele cujo ponto de interseção for o mais próximo à origem do raio. A determinação da interseção de um raio com um primitivo é objeto de **A2** e **A3**.

A2 Completar, na classe `Primitive`, a implementação do método de interseção de raio com malha de triângulos. Na versão disponível no repositório, o método inicialmente transforma o raio para o espaço local do primitivo e, então, calcula a interseção do raio no espaço local com a caixa envolvente da malha de triângulos do primitivo. Se o raio não interceptar a caixa, obviamente não intercepta o primitivo nela contido. Caso contrário, deve-se computar a interseção do raio com a malha propriamente dita, o que envolve a implementação do algoritmo de interseção de raio com triângulo visto em sala. O triângulo interceptado pelo raio, se houver, é aquele cujo ponto de interseção for o mais próximo à origem do raio (no sistema local). Por fim, o método transforma a distância entre a origem do raio e o ponto de interseção, se houver, de volta para o sistema global. Você pode usar força-bruta, ou

A3 Implementar uma BVH para acelerar a interseção raio/malha, conforme explicado em sala. Nesse caso, a BVH vai substituir, na implementação de **A2**, o teste simples de interseção do raio com a caixa envolvente da malha e, caso houver interseção com a caixa envolvente, o cálculo de interseção do raio com todos os triângulos da malha. Deverá haver somente **uma** BVH para cada malha usada na cena, construída no espaço local da malha.

A4 Completar a implementação do método `RayTracer::shade()` responsável pela tonalização de um ponto de interseção com um raio. A tonalização leva em conta a iluminação direta de todas as fontes de luz da cena que iluminam o ponto, isto é, cujos não são obstruídas por nenhum primitivo em seu caminho até o ponto. Para a iluminação direta, considere o modelo de iluminação local de Phong. A tonalização também leva em conta a reflexão especular indireta, computada pelo traçado recursivo de raios secundários de reflexão, conforme explicado em sala.

A5 Criar duas ou mais cenas de teste para eu traçador de raios, as quais devem explorar todas as capacidades de seu traçador de raios. Use arquivos OBJ para a geometria dos primitivos. Defina materiais, adicione luzes e posicione a câmera de maneira planejada e criativa. Crie um novo item na barra de menu principal chamado `Examples` para possibilitar o acesso as cenas que você criou.

3 Passo 3: README

Como no exercício anterior, o arquivo `README` deve conter o nome(s) do(s) autor(es) e uma descrição de como gerar (caso o Visual Studio 2019 não tenha sido utilizado) e executar o programa. Em seguida, você deve descrever quais as atividades você conseguiu ou não implementar, parcial ou totalmente, além de um breve manual do usuário (por exemplo, se é preciso usar alguma tecla para alguma funcionalidade para a qual não há ajuda textual na interface gráfica com o usuário). Descreva também quaisquer outras extensões que você implementou por iniciativa própria, as quais podem valer pontuação extra em sua nota do exercício.

4 Passo 4: Entrega do programa

O exercício deve ser entregue via AVA em arquivo único compactado (somente um arquivo por grupo), chamado `p4.zip` (nome(s) do(s) autor(es) vão no arquivo `README`), contendo:

- o diretório `p4 com` o código-fonte completo e com o arquivo executável `p4.exe`, mas **sem** quaisquer arquivos intermediários de backup ou resultantes da compilação, tanto em `p4` como em seus subdiretórios;
- o arquivo `README`; e
- os arquivos OBJ usados em suas cenas de teste, bem como as respectivas imagens geradas pelo traçador de raios, na resolução 1280×720 pixeis.

Não serão considerados e terão nota zero programas plagiados de qualquer que seja a fonte, mesmo que parcialmente, exceção feita ao código fornecido pelo professor.

5 Lista de objetivos

A verificação dos objetivos da **Parte 4** levará em conta se:

☐ O arquivo `p4.zip` foi submetido com os arquivos corretos, e o arquivo `README` contém as informações solicitadas.

☐ O programa final está completo, isto é, as implementações da **Parte 1**, **Parte 2** e **Parte 3** estão integradas e funcionando no código constante em `p4.zip`.

☐ A interseção de um raio com a cena (e, consequentemente, com seus primitivos) foi implementada, e a implementação está correta, com ou sem BVH. A implementação da BVH valerá 20% da nota do exercício.

☐ A iluminação direta de um ponto foi implementada, e a implementação está correta.

☐ O traçado recursivo de raios secundários de reflexão foi implementado, e a implementação está correta.

☐ Pelo menos duas cenas de teste foram criadas, e o programa é capaz de gerar corretamente as respectivas imagens com o traçador de raios.