UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA

Disciplina: [**GDSCO0051] Introdução à Computação Gráfica - Turma 01**.
Semestre: **2019.2**.
Professor: **Christian Azambuja Pagot (email: christian@ci.ufpb.br)**.
Data de entrega: **06/04/2020, às 23h55min**.

Prova II

Instruções:

- A prova é **individual** e com **consulta** (*e.g.* slides da aula, textos, livros, internet, etc.).
- A prova possui 4 questões numeradas de 1 à 4, e vale 10 pontos.
- As questões marcadas como **BÔNUS** são opcionais e não fazem parte da prova. Porém, os pontos obtidos nestas questões serão adicionados à pontuação final da prova.
- O **número máximo** de linhas de cada resposta (indicado em cada questão) deve ser **respeitado**.
- Se for o caso, as fontes bibliográficas utilizadas nas respostas devem ser **referenciadas** ao final de cada questão (referências bibliográficas) .
- O arquivo contendo as respostas deve ser **enviado pelo SIGAA** em **formato PDF**. O arquivo de respostas da prova deve conter o **nome** e **matrícula** do aluno, bem como identificar cada questão respondida. O tamanho da fonte utilizada nas respostas deve ser **12**.
- **Não serão aceitas respostas enviadas por outro meio que não o SIGAA.**
- **Não serão aceitas respostas enviadas após a data final de envio.**

Questão 1 (2.5 pontos):

O *z-buffer* é o algoritmo de remoção de superfícies ocultas mais popular e eficente em uso atualmente. Praticamente todas as aplicações que demandam renderização em tempo real utilizam o z-buffer. Apesar de muito eficiente e simples, entretanto, o z-buffer apresenta limitações no que se refere à renderização de primitivas transparentes.

Explique, com suas palavras, porque o z-buffer tradicional (aquele visto em aula) não consegue renderizar devidamente primitivas transparentes (**máximo 10 linhas, adicionar referências**).

Resposta

O *Algoritmo de Z-Buffer* rasteriza a primitiva e testa o valor de profundidade de cada fragmento contra a posição correspondente no *buffer* de profundidade. Se o valor de profundidade do fragmento for menor do que aquele armazenado no *buffer*, o valor de profundidade do *buffer* é atualizado e o pixel correspondente do *color buffer* é colorido. Caso contrário, o fragmento é descartado fazendo com que o *buffer* de profundidade e o *color buffer* não sejam atualizados.

Se uma primitiva transparente próxima a tela á rasterizada, ela atualizará o *buffer* de profundidade com valores pequenos. Isso impedirá que primitivas mais distantes, rasterizadas posteriormente, e que cubram esta mesma região do buffer de profundiade, contribuam para a atualização do *color buffer*, pois seus fragmentos serão descartados ainda durante o teste de profundidade.

>>> Questão 1.1 BÔNUS - (0.5 pontos): Pesquisa e descreva ao menos uma técnica existente que permita o rendering de primitivas transparentes com o z-buffer (**máximo 8 linhas, adicionar referências**).

Resposta

Alpha Blending: Neste algoritmo, primeiro as primitivas opacas são rasterizadas. Em seguida, as primitivas transparentes são rasterizadas, porém na ordem de trás-pra-frente. Esta ordem garante que cada nova primitiva transparente se sobreporá às primitivas transparentes rasterizadas anteriormente. Uma função de *blending* 'mistura' a cor da primitiva transparente sendo rasterizada atualmente com as cores presentes atualmente no *color buffer*. O resultado da função de *blending* é utilizado para atualizar os pixels correspondentes do *color buffer*.

Questão 2 (2.5 pontos):

A Figura a seguir ilustra os três modelos de sombreamento (ou *shading*) normalmente encontrados em implementações de pipelines gráficos: *Flat*, *Gouraud* e *Phong shading*. Explique suscintamente, e com suas próprias palavras, como cada um destes é calculado bem como suas vantagens e desvantagens. (**máximo de 8 linhas para cada modelo de *shading, adicionar referências***).

Por Davi.trip (wikimedia.org)

Resposta

No *flat shading* um único vetor normal é utilizado para calcular a cor final da primitiva inteira. O cálculo pode utilizar inclusive o modelo de reflectância de Phong. A cor é calculada em um ponto e replicada em todos os fragmentos da primitiva, atribuindo uma cor uniforme à toda a primitiva. Este método é muito eficiente porém gera discontinuidade das cores nas arestas das primitivas.

No Gouraud *shading* o modelo de reflectância é resolvido para cada vértice da primitiva que possui sua própria normal. Uma vez que estes cálculos estejam finalizados para os vértices, a cor dos demais fragmentos é determinada por meio de interpolação (*i.e.* interpolação hiperbólica). Este método é mais caro computacionalmente que o *flat shading*, porém gera resultados mais suaves (com discontinuidades de cor de maior ordem nas arestas da primitiva).

No Phong *shading* as propriedades dos vértices, incluindo seus vetores normais, são interpoladas sobre todos os fragmentos gerados pela rasterização da primitiva. Uma vez que cada fragmento agora tem seu próprio conjunto de propriedades, o modelo de reflectância pode ser resolvido independentemente para cada fragmento. Este método é o mais caro computacionalmente entre os três, porém é o que gera os resultados de iluminação mais suaves.

Questão 3 (2.5 pontos):

As duas figuras abaixo (Figuras A e B) ilustram quadriláteros sobre os quais foram mapeadas texturas idênticas. A diferença visual entre as duas imagens se deve às diferentes técnicas de filtragem utilizadas durante o mapeamento da textura. Na Figura A, por exemplo, foi utilizada a técnica de filtragem de textura conhecida como ***nearest neighbor***.

Tendo como referência os aspectos visuais das Figuras A e B, responda: qual foi a técnica de filtragem utilizada durante o mapeamento da textura da Figura B. Justifique a sua resposta (**máximo 6 linhas, adicionar referências**).

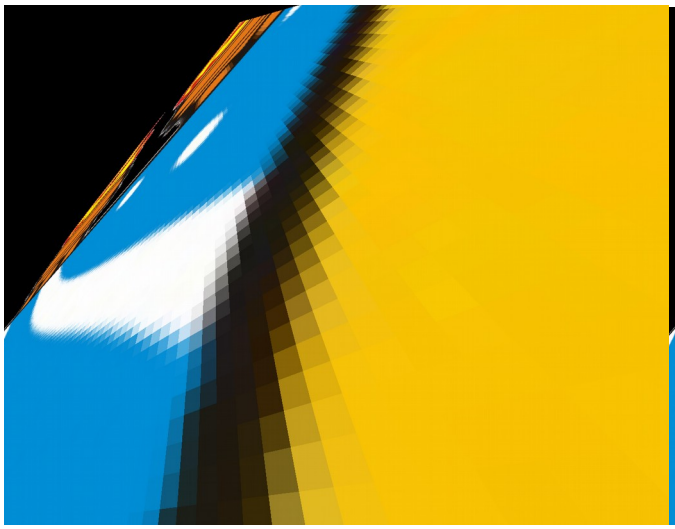

Figura: A

Figura: B

Resposta

Considerando que na primeira imagem utilizou-se a filtragem *nearest neighbor*, pode-se perceber, do ponto de vista visual, que está ocorrendo o fenômeno da magnificação de textura, ou seja, os texels estão cobrindo uma área maior que a do próprio um pixel. Neste caso, uma técnica de filtragem adequada seria uma que reduzisse os efeitos da magnificação (o serrilhado). Uma destas técnicas é a filtragem bilinear. Uma das desvantagens da filtragem bilinear, entretanto, é que ela pode gerar borramento da textura, o que compatível com o resultado da imagem B.

>>> Questão 3.1 BÔNUS - (0.5 pontos): Quais os objetivos comuns e as diferenças entre as técnicas de filtragem de textura chamadas de *MipMapping* e Filtragem Anisotrópica?(**máximo 8 linhas, adicionar referências**) Ilustre com imagens de exemplo (**máximo 4 imagens, adicionar referências**).

Resposta

As filtragens *MipMapping* e Anisotrópica procuram reduzir os efeitos de ruído gerados pela minificação de textura. Enquanto o *MipMapping* opera em cima de um conjunto pré determinado de texturas filtradas isotropicamente, a filtragem anisotrópica procura aproximar a região de filtragem da projeção do pixel sobre a cena. Por usar filtragem isotrópica das texturas, o *MipMapping* tende a gerar resultados mais borrados em regiões onde a textura apresenta variações de cor de alta frequência. A filtragem anisotrópica tende a gerar menor borramento, porém, a um custo computacional mais alto.

Ref: https://en.wikipedia.org/wiki/Anisotropic_filtering

Questão 4 (2.5 pontos):

A técnica de geração de imagem baseada em *ray tracing* (ou traçado de raios) é radicalmente diferente da técnica clássica baseada em rasterização. Tendo suas origens nos anos 60, nos trabalhos de Arthur Appel, o *ray tracing* evoluiu muito ao longo dos anos e hoje é uma das técnicas preferidas para os efeitos especiais e animações para o cinema. Nos últimos anos também muitos trabalhos de pesquisa têm investigado o uso do *ray tracing* em jogos e aplicações de tempo real.

Explique, com suas palavras, como a técnica de *ray tracing* funciona, e quais suas principais diferenças em relação ao método da rasterização (**máximo 12 linhas, adicionar referências**).

Resposta

A técnica de *ray tracing* gera suas imagens através de uma simulação do caminho percorrido pelos raios de luz utilizando o modelo de óptica geométrica. Porém, ao invés de simular os raios de luz partindo da fonte, e atingindo o sensor da câmera, o *ray tracing* segue o caminho inverso, emitindo raios de luz da câmera em direção à fonte de luz.

Assim que um raio parte da câmera, passando pelo centro de um pixel, o raio é testado contra todos os objetos a cena em busca de intersecções. A intersecção mais próxima da câmera é registrada e a partir deste ponto eventualmente novos raios são lançados de acordo com os materiais utilizados nos objetos interseccionados.

Enquanto a rasterização percorre todas as primitivas da cena e as projeta na tela tirando proveito de suas coerência espacial, o ray tracing percorre todos os pixels da tela testando a intersecção de raios contra todas as primitivas da cena, o que se mostra mais caro computacionalmente. A rasterização, por outro lado, por se basear em interpolações, limita o conjunto de primitivas com que pode operar.

>>> Questão 4.1 BÔNUS - (0.5 pontos): Enumere pelo menos três filmes que utilizaram *ray tracing* em seu rendering, ou no rendering de seus efeitos especiais (**máximo 9 linhas, adicionar referências**).

Resposta

- Carros 3
- Universidade Moonstros
- Viva, a Vida é uma Festa

>>> Questão 4.2 BÔNUS - (0.5 pontos): Enumere pelo menos dois jogos atuais que utilizam *ray tracing*, nem que em apenas alguns de seus efeitos visuais (**máximo 9 linhas, adicionar referências**).

Resposta

- Battlefield V
- Control