Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AP2 - 1° semestre de 2009.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1) O Algoritmo de Ray-tracing realiza uma interseção de um raio com os objetos 3D de uma cena. Caso haja interseção, chama-se o algoritmo de iluminação para a superfície em questão. Descreva detalhadamente como é o cálculo de interseção do raio com uma esfera, lembrando que a equação da esfera pode ser dada por $(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$

Seja um raio **p(t) = e + td**, onde **e** é a origem do raio e **d** a sua direção, e uma superfície implícita f(p)=0. A interseção ocorre nos pontos do raio que satisfazem **f(p(t))=0**.

A esfera com centro em **c=(x_0,y_0,z_0)** pode ser representada através da equação implícita **$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2=0$**. Podemos escrever tal equação na forma vetorial como **<(p-c),(p-c)> - r^2=0**.

Qualquer ponto **p** que satisfaça esta equação está sobre a esfera. Se substituirmos os pontos do raio na equação teremos:

$$< (e + td - c), (e + td - c) > - r^2 = 0$$

Após a manipulação dos termos chegamos à:

$$< d,d > t^2 + 2 < d,(e-c) > t + < (e-c),(e-c) > -r^2$$

Na equação anterior todas as variáveis são conhecidas exceto **t**, o que nos dá uma equação quadrática clássica em **t**. A solução de tal equação é obtida conforme a expressão abaixo:

$$t = \frac{-2 < d,(e-c) > \pm\sqrt{4 < d,(e-c) >^2 -4 < d,d > (< (e-c),(e-c) >^2 -r^2)}}{2 < d,d >} \therefore$$

$$t = \frac{< d,(e-c) > \pm\sqrt{< d,(e-c) >^2 - < d,d > (< (e-c),(e-c) >^2 -r^2)}}{< d,d >}$$

Em uma implementação, primeiro checa-se o sinal do discriminante antes de se calcular os outros termos.

2) O que é o Environment Mapping? Qual a sua vantagem e desvantagem em relação ao Ray-tracing?

A técnica de mapeamento de ambiente (e*nviroment mapping*) é uma técnica usada para realizar a renderização de objetos que refletem um fundo texturizado. Em outras palavras, é o um mapeamento do ambiente sobre um objeto, no qual o próprio objeto está imerso, ou seja, reflete o cenário.

A primeira forma de obter este efeito é tomando um cubo, o qual requer seis imagens de textura, uma para cada direção contendo as informações dos objetos que compõem o ambiente. Para cada vértice do polígono, um vetor de reflexão é calculado, através do qual é indicada uma posição em uma das seis imagens. A segunda forma é gerar uma única imagem do objeto refletindo o ambiente. Cada ponto do objeto representará um ponto do ambiente.

O método foi proposto por Blinn & Newell em 1976 e permite simular efeitos de *Ray tracing* a baixo custo. A desvantagem desta técnica em relação ao Ray-tracing é a de que os objetos na cena envolvidos pelo mapa de ambiente não refletem a si mesmos, gerando efeitos que podem não condizer com o resultado esperado. O mesmo problema pode surgir em uma cena com um único objeto, caso o mesmo não seja convexo. Naturalmente, o environment mapping não produz os efeitos de inter-reflexão que um algoritmo de Ray-tracing real é capaz de realizar. Por outro lado, é uma técnica muito mais eficiente do que Ray-tracing e pode ser utilizada se são esperados efeitos modestos de refletividade.

3) Cite 2 problemas de aliasing que podem ocorrer com texturas.

O problema de *aliasing* pode surgir com intensidade quando a textura é minimizada, isto é quando vários texels cobrem a região correspondente a um pixel. Para resolver este problema de amostragem adequadamente é necessário

integrar o efeito dos vários texels influenciando o pixel, o que é muito difícil de fazer em tempo real. Na prática, a fins de processamento em tempo real, adota-se o uso de filtros vizinho mais próximo (*nearest neighbor*) e *bilinear*. O primeiro causa severos problemas de *aliasing*, porque escolhe somente um texel dentre os muitos que podem influenciar um pixel. O filtro baseado em interpolação *bilinear* combina quatro texels, mas ainda assim produz problemas de *aliasing*. Outro problema de aliasing grave é o denominado *aliasing temporal* que ocorre quando a textura se move com relação ao observador, tornando os efeitos de aliasing ainda mais evidentes.

Quando a textura é aplicada sobre uma área com um número de pixels maior do que o número de texels, podem surgir problemas de jagging (serrilhado) ao se usar o filtro do vizinho mais próximo, ou então um efeito de borramento, ao se utilizar o filtro bilinear. Isto, entretanto, não é considerado *aliasing* por ser mais um problema de reconstrução do que de amostragem.

4) Como é feita a projeção esférica de textura?

Na projeção esférica uma textura é inicialmente aplicada a uma esfera por meio de equações que transformam suas coordenadas em coordenadas esféricas. Em seguida o objeto é colocado na região central do interior da esfera, e para cada ponto da sua superfície, traça-se uma reta que passa pelo centro da esfera e por este ponto, até encontrar a textura aplicada na esfera. A cor associada ao ponto da superfície do objeto será dada pela cor da textura onde a reta encontrou a esfera.

5) Porque não se pode gerar a imagem diretamente no Front Buffer? Qual a estratégia adotada?

Não se deve gerar a imagem diretamente para o Front Buffer no caso de cenas dinâmicas, por exemplo, uma animação, uma vez que isso pode causar o efeito de *flickering* devido ao não sincronismo entre a cena, e o conteúdo do buffer que será exibido no dispositivo. Uma idéia é utilizar um buffer adicional denominado *Back-Buffer*. Durante a renderização de um quadro de uma cena, a imagem é construída no *Back-buffer*, que funciona como uma região de rascunho, até que seja completada e então, através de uma chamada a uma rotina de troca de buffers (glSwapBuffers(), por exemplo), o seu conteúdo é transferido para o *Front-Buffer* que, por sua vez, alimenta o dispositivo de exibição com suas informações. Desta forma a imagem da cena é construída de modo consistente no *Back-buffer*, antes que seja copiada para o *Front-Buffer*.