INSTITUTO DE INFORMÁTICA UFRGS
INA
DEPARTAMENTO DE INFORMÁTICA APLICADA

Fundamentos de Computação Gráfica
Semestre 2009/1

Prova 2 – 29/junho/2009 – Turma A

NOME:

IDENTIFICAÇÃO UFRGS:

1. ***Marque V ou F, caso a afirmativa seja verdadeira ou falsa***. Acerca das **transformações** pelas quais passam as primitivas geométricas para a geração da imagem de objetos:
   ( ) A transformação de câmera transforma as coordenadas do sistema de referência do mundo para o sistema de referência da tela.
   ( ) A projeção é a parte final da transformação de câmera.
   ( ) Em OpenGL, as transformações geométricas e a transformação de câmera são compostas na matriz *ModelView*.
   ( ) As transformações armazenadas na matriz ModelView são realizadas no sistema de mundo.
   ( ) Em OpenGL, a transformação de projeção é armazenada na matriz *Projection*.

2. Em relação ao **modelo de iluminação de Phong**:
   (a) Os valores calculados utilizando o modelo de iluminação correspondem às _____ associadas aos vértices da geometria.
   (b) O modelo é baseado em três componentes de reflexão: _____, _____ e _____.
   (c) O valor associado à componente _____ é independente da orientação da superfície sendo tonalizada.
   (d) O valor associado à componente _____ é calculado levando em consideração a orientação da superfície e a direção de incidência da luz, mas é independente da posição do observador.
   (e) O valor associado à componente _____ é função da direção da normal à face, da direção de incidência da luz e da direção de observação.

3. **Iluminação**: Explique quais as diferenças entre os **modelos de sombreamento Gouraud e Phong**.

4. **Projeções:** Desenhe os dois volumes de visualização que determinam **projeção paralela ortográfica** e **perspectiva**, respectivamente. Identifique os principais elementos nesses volumes, envolvidos no processo de recorte e na projeção.

5. **Transformações geométricas:** Observe o código abaixo. Desenhe o **resultado da execução** do mesmo, identificando os objetos de acordo com os números em comentário no código. Lembrando: a função *glutWireCube* produz a imagem do cubo aramado centrado na origem.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt (4, 4, -10, 0, 0, 0, 0, 1, 0);
glColor3f(1.0,0.0,0.0);
glPushMatrix();
glutWireCube(4);  // cubo 1
glTranslatef(-1.5,-1.5,-1.5);
glScalef(0.25,0.25,0.25);
glColor3f(0.0,1.0,0.0);
glutWireCube(4);    // cubo 2
glTranslatef(12,6,12);
glColor3f(0.0,0.0,1.0);
glutWireCube(4);   // cubo 3
glPopMatrix();
glTranslatef(-6,0,0);
glScalef (0.5, 0.5, 0.5);
glColor3f(0.0,0.0,0.0);
glutWireCube(4);   // cubo 4
```

6. ***Mapeamento de textura:*** Considere um cubo com coordenadas (x, y, z) entre -1 e 1 (isto é, um cubo centrado na origem). Considere a imagem abaixo, a ser utilizada como textura a ser mapeada nas faces do cubo.

Mostre com desenho, com pseudo-código ou em OpenGL, como você faria o mapeamento das 6 primeiras regiões da imagem para obtê-las mapeadas nas 6 faces do cubo.

7. ***Marque V ou F, caso a afirmativa seja verdadeira ou falsa.***
   - ( ) O algoritmo de ray-tracing é uma alternativa ao pipeline convencional que usamos em OpenGL, e é baseado no traçado de raios dos objetos até o olho do observador.
   - ( ) Através do algoritmo de ray-tracing podemos obter imagens com reflexão especular mas não podemos modelar a refração.
   - ( ) Curvas Hermite são definidas por quatro pontos de controle.
   - ( ) Curvas Bézier podem ser definidas a partir de qualquer número de pontos de controle, mas as mais comuns são as com quatro pontos de controle.
   - ( ) Chamamos de funções mistura as funções do parâmetro **t** que justamente dão o peso de cada ponto de controle na composição da curva final.
   - ( ) A composição de curvas através da junção de segmentos deve ser feita levando em consideração apenas continuidade de primeira ordem, ou seja, de tangente.
   - ( ) Normalmente, utilizamos o modelo de iluminação de Phong para computar a iluminação em cada ponto de encontro dos raios (no ray-tracing) com os objetos.
   - ( ) Ray–tracing permite detectar regiões de sombra facilmente, traçando-se um raio da luz até o observador.

8. Suponha que você deve se deslocar num cenário 2D a partir de uma posição (x1, y1), usando comandos de *andar para frente 1 passo*, *girar à direita 1º* , *girar para a esquerda 1º* . Como você decomporia esses comandos para realizar esse cálculo? Como você calcula colisão com um obstáculo a cada passo?