1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

dei departamento de engenharia informática 1995 – 2020

LEI - Computação Gráfica

prof. André Perrotta, prof. Evgheni Polisciuc

Teste 2:

Iluminação & textura

Nome: Tiago Jorge Coimbra da Silva

Número: 2022216215

Duração: 60min | 12 de Dezembro, 2023 | valor max: 20

Formulário

sejam os vetores $\vec{A}(a_1, a_2, a_3)$ e $\vec{B}(b_1, b_2, b_3)$

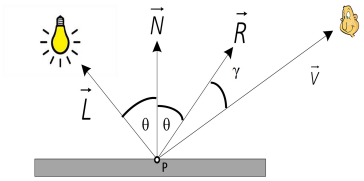
produto escalar:

$$\vec{A} \bullet \vec{B} = \sum_{i=1}^{3} a_i b_i = |\vec{A}||\vec{B}| \cos\theta$$

produto vetorial:

$$\vec{A} \times \vec{B} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2 b_3 - b_2 a_3)\hat{x} + (a_3 b_1 - b_3 a_1)\hat{y} + (a_1 b_2 - b_1 a_2)\hat{z}$$

Modelo de Phong para iluminação:



$$\vec{R} = 2(\vec{L} \bullet \vec{N})\vec{N} - \vec{L}$$

$$I_{vertice} = I_{luz_{amb}} K_{mat_{amb}} + I_{luz_{dif}} K_{mat_{dif}} \cos\theta + I_{luz_{spec}} K_{mat_{spec}} \cos\gamma^{ns}$$

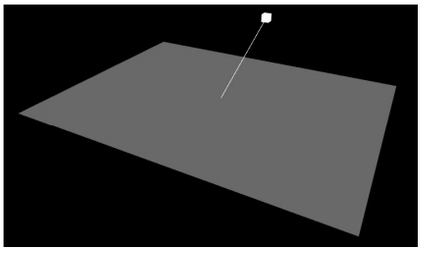
	0°	30°	45°	60°	90°
$\sin(\theta)$	0	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{\sqrt{3}}{2}$	1
$\cos(\theta)$	1	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	0
$\tan(\theta)$	0	$\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$	undefined

$I_{ver} = I_{luz_{amb}} K_{mat_{amb}} + I_{luz_{dif}} K_{mat_{dif}} \cos\theta + I_{luz_{spec}} K_{mat_{spec}} \cos\gamma$

Iluminação

Q1(5 valores):

A cena a seguir representa um frame de uma aplicação implementada em OpenFrameworks/OpenGL. A cena mostra uma quadrado unitário construido com malha de vértices, no plano $(z = 0)$, com centro na origem $(0, 0, 0)$ do espaço 3D, escala $(gw(), gh(), 1)$, vetor normal $\vec{N} = (0, 0, 1)$ em todos os vértices e que seu material tem coeficientes de reflexão $(1, 1, 1, 1)$ para todos os componentes e coeficiente de especularidade $ns = 1$. O pequeno cubo e linha de cor branca representa a posição da fonte de luz no frame.



cena - 01

Considere que a iluminação da cena é calculada com modelo de Phong clássico e que existe apenas uma fonte de luz com componentes de intensidade e cor definidos por:

$I_{amb} = (R_{amb}, G_{amb}, B_{amb})$

$I_{dif} = (R_{dif}, G_{dif}, B_{dif})$

$I_{spec} = (0, 0, 0)$

Luz diagonal

(a) (2 valores): Considere que a fonte de luz foi configurada com o seguinte comando:

```
glLightfv(GL_LIGHT0, GL_POSITION, {0, 1, 1, 0});
```

Determine um conjunto de (possíveis) valores (R, G, B) das suas componentes para que a cor final seja $(\frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4})$ em qualquer ponto da malha (justifique sua resposta).

resp:

$\frac{\vec{L} \cdot \vec{N}}{\|\vec{L}\| \|\vec{N}\|} = \cos\theta \Rightarrow \frac{1}{\sqrt{2}} = \cos\theta$

$I_{vtx} = I_{luz amb} k_{mat amb} + I_{luz dif} k_{mat dif} \cos\theta + I_{luz spec} k_{mat spec} \cos\gamma$

$= (R_{amb}, G_{amb}, B_{amb}) + (R_{dif}, G_{dif}, B_{dif}) \frac{\sqrt{2}}{2}$

$0,0,0 + \frac{1}{2}, \frac{1}{2}, \frac{1}{2}$

ou

$\frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4} + 0,0,0$

$I_{vis} = I_{luzamb} K_{matamb} + I_{luzdif} K_{matdif} \cos\theta + I_{luz\,spec} K_{mat\,spec} \cos\gamma$

$\frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$

(b) (3 valores): Considere agora que a configuração da luz é atualizada em todo frame através da seguinte lógica (pseudo-código):

```
x = 0;
y = gh() * 0.5;
z = A*gh()*0.5*(cos(theta*PI/180.)*0.5 + B);
light_pos = (x,y,z,0) - (0,0,0,0);
glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
theta++;
```

} Luz direcional

$(0,1,1)$

Sabendo-se que o valor da intensidade/cor final é a mesma em todos os pontos da malha e conhecendo os valores nos momentos de máximo $I_{max} = (1,1,1)$ e mínimo $I_{min} = (0.5, 0.5, 0.5)$, determine um (possível) conjunto de valores (R, G, B) para as componentes ambiente e difusa da fonte de luz, bem como para os parâmetros A e B utilizados no algoritmo (justifique sua resposta).

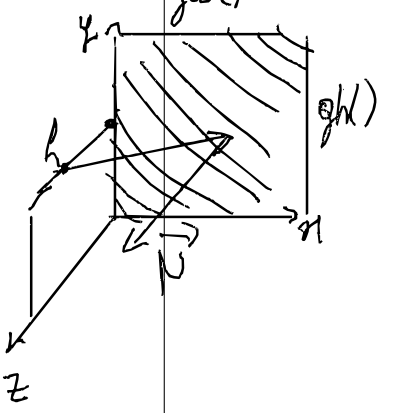
$\vec{N} = (0,0,1)$

resp:

Para $A=2, B=0$: $x=0, y=\frac{gh()}{2}, z=\frac{gh() \cdot \cos\left(\frac{\pi\theta}{180}\right)}{2}$

O valor máximo de intensidade é atingido quando $\theta = 0°$:

$(x,y,z) = (0, h/2, h) = \vec{pos}$



$(1,1,1) = I_{luzAmb} + I_{luz\,dif_{max}} \cos(45°)$

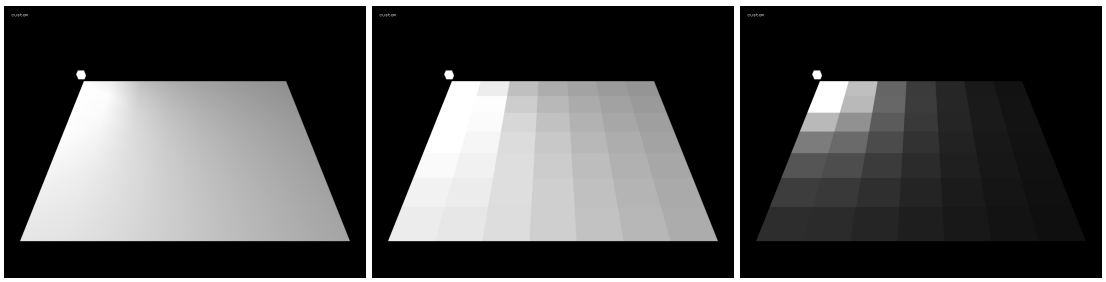
e mínimo quando $\theta = 90°$:

$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) = I_{luzAmb}$

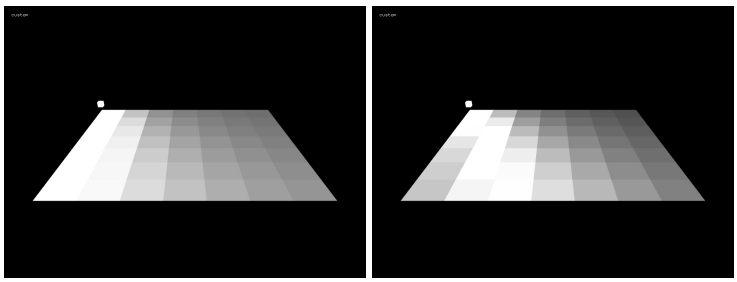
$\Rightarrow (1,1,1) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) + \left(\frac{\sqrt{2}}{2}R, \frac{\sqrt{2}}{2}G, \frac{\sqrt{2}}{2}B\right) \Leftrightarrow RGB = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

Q2(3 valores):

As 5 imagens a seguir representam um frame de uma aplicação implementada em OpenFrameworks/OpenGL. A cena mostra um quadrado unitário construido com malha de vértices de resolução 7x7, no plano $(z = 0)$, com centro na origem $(0, 0, 0)$ do espaço 3D, escala $(gw(), gh(), 1)$, vetor normal $\vec{N} = (0, 0, 1)$ em todos os vértices e que seu material tem coeficientes de reflexão $(1, 1, 1, 1)$ para todos os componentes e coeficiente de especularidade $ns = 1$. A cena possui apenas uma fonte de luz pontual. O pequeno cubo de cor branca representa a posição da fonte de luz no frame. A luz é inicialmente configurada com componentes difusa e especular brancas e componente ambiente preta.



frame A frame B frame C



frame D frame E

Com base nas imagens, responda às perguntas a seguir de forma clara, sucinta e justificada. Utilize pseudo-código OpenFrameworks/OpenGL se achar necessário.

(a) (1 valor): Qual a diferença de configuração da cena entre os frames A e B?
resp:

Gouraud x Flat shading

(b) (1 valor): Qual a diferença de configuração da luz da cena entre os frames B e C?
resp:

Atenuação da componente especular
vs
Aumento da componente especular

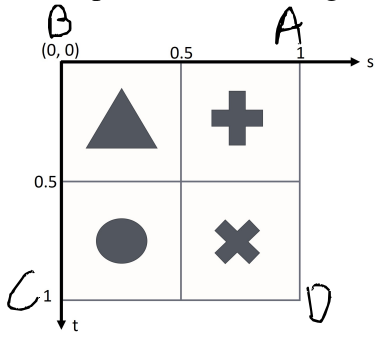
(c) (1 valor): Qual a diferença de configuração da iluminação da cena entre os frames D e E?
resp:

FAR

Textura

Q3(10 valores):

Complete o pseudo-código com as coordenadas de textura e configuração adequada para obter os resultados conforme as imagens.
obs 1: os parâmetros de configuração podem ser em pseudo-código, mas devem ser claros e coerentes com as configurações reais possíveis.
obs 2: A imagem está em espaço de coordenadas de textura com dimensão normalizada, eixo t orientado para baixo, eixo s orientado para a direita e origem no topo-esquerdo da imagem.



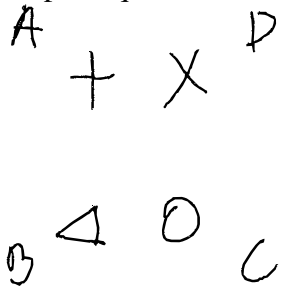
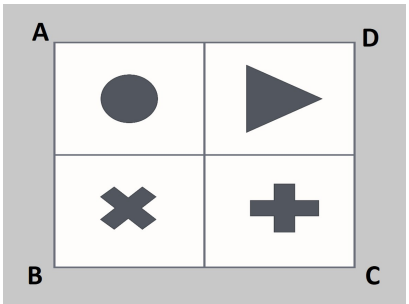


Imagem original

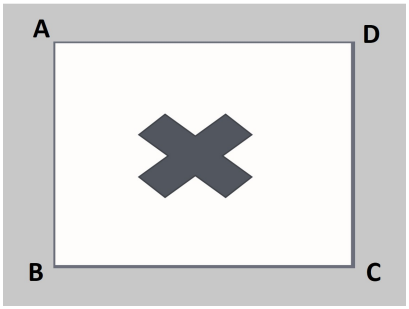
(a)(2 valores):

```
texParameter(GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
texParameter(GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glBegin(GL_QUADS);
texCoord(0, 1); vertex_A(−0.5, 0.5, 0);
texCoord(1, 1); vertex_B(−0.5, −0.5, 0);
texCoord(1, 0); vertex_C(0.5, −0.5, 0);
texCoord(0, 0); vertex_D(0.5, 0.5, 0);
glEnd();
```



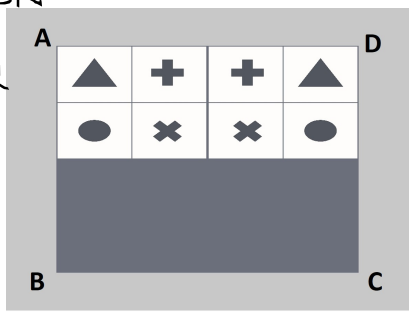
(b)(2 valores):

```
texParameter(GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
texParameter(GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glBegin(GL_QUADS);
texCoord(0.5, 0.5); vertex_A(−0.5, 0.5, 0);
texCoord(0.5, 1); vertex_B(−0.5, −0.5, 0);
texCoord(1, 1); vertex_C(0.5, −0.5, 0);
texCoord(1, 0.5); vertex_D(0.5, 0.5, 0);
glEnd();
```



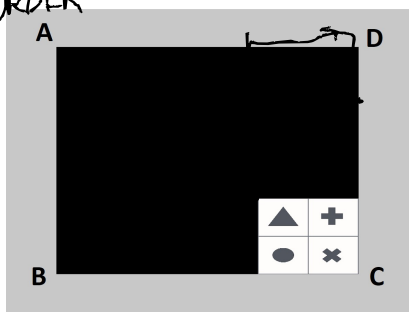
(c)(2 valores):

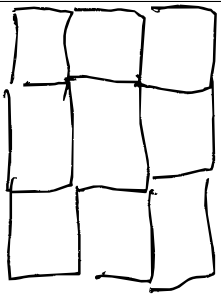
```
texParameter(GL_TEXTURE_WRAP_S, GL_REPEAT / MIRROR);
texParameter(GL_TEXTURE_WRAP_T, GL_CLAMP_TO_BORDER);
glBegin(GL_QUADS);
texCoord(0, 0); vertex_A(−0.5, 0.5, 0);
texCoord(0, 2); vertex_B(−0.5, −0.5, 0);
texCoord(2, 2); vertex_C(0.5, −0.5, 0);
texCoord(2, 0); vertex_D(0.5, 0.5, 0);
glEnd();
```



(d)(2 valores):

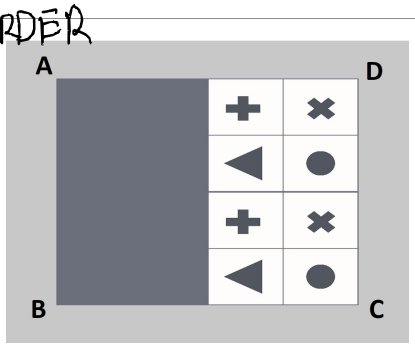
```
texParameter(GL_TEXTURE_WRAP_S, GL_CLAMP_TO_BORDER);
texParameter(GL_TEXTURE_WRAP_T, CLAMP2BORDER);
glBegin(GL_QUADS);
texCoord(-2, -2); vertex_A(−0.5, 0.5, 0);
texCoord(-2, 1); vertex_B(−0.5, −0.5, 0);
texCoord(1, 1); vertex_C(0.5, −0.5, 0);
texCoord(1, -2); vertex_D(0.5, 0.5, 0);
glEnd();
```





(e)(2 valores):

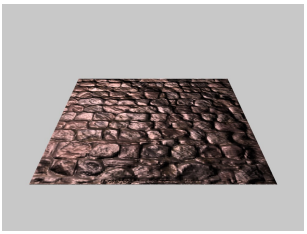
```
texParameter(GL_TEXTURE_WRAP_S, GL_CLAMP_TO_BORDER);
texParameter(GL_TEXTURE_WRAP_T, GL_REPEAT);
glBegin(GL_QUADS);
texCoord(1, 0); vertex_A(−0.5, 0.5, 0);
texCoord(0, −1); vertex_B(−0.5, −0.5, 0);
texCoord(−1, 0); vertex_C(0.5, −0.5, 0);
texCoord(1, 1); vertex_D(0.5, 0.5, 0);
glEnd();
```



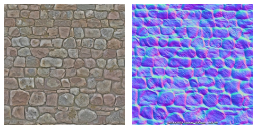
Efeitos de iluminação e textura

Q4(2 valores):

A cena ao lado foi gerada utilizando a técnica de *bump mapping* implementada em OpenFrameworks com pipeline poligonal de OpenGL. Descreva de forma clara, objetiva e resumida, utilizando pseudo-código onde achar pertinente, um possível algoritmo para implementação da cena utilizando as imagens albedo(cor) e normalMap como ponto de partida. Tenha atenção aos detalhes da cena e etapas fundamentais do algoritmo.



cena com *bump mapping*



albedo normalMap

resp: