LEI - Computação Gráfica

prof. André Perrotta, prof. Evgheni Polisciuc

Exame Recurso

Nome: Tiago Jorge Coimbra da Silva

Número: 2022216215

Duração: 90min | 31 de Janeiro, 2024 | 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:

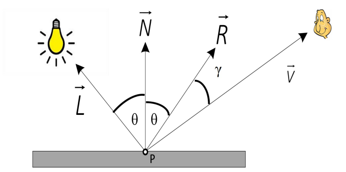
$$\vec{A} \times \vec{B} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2 b_3 - b_2 a_3)\hat{x} + (a_3 b_1 - b_3 a_1)\hat{y} + (a_1 b_2 - b_1 a_2)\hat{z}$$

transformações geométricas:

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Proj_{perspectiva_{openGl}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \quad Proj_{ortogonal} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Modelo de Phong para iluminação:



	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

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

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

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

Conceitos

Q1 (2 valores)

Em álgebra linear, a matriz de projeção ortográfica é definida como:

$$Proj_{ortogonal} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Porém, em OpenGL, a função *glOrtho(...)* utiliza a matriz:

$$Proj_{ortogonal_{openGl}} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & tx \\ 0 & \frac{2}{top-bottom} & 0 & ty \\ 0 & 0 & \frac{-2}{far-near} & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Desconsiderando-se os parâmetros de escala e translação (Sx, Sy, tx, ty, tz), Por que, no pipeline poligonal OpenGL, é necessário que o valor de Sz (terceira linha, terceira coluna) seja diferente de zero, apesar de se tratar de uma transformação que converte 3D em 2D?(resposta sucinta e objetiva)

O GPU precisa do valor da componente z para no passo das operações finais poder usar o depth testing.

Q2 (3 valores)

Na programação gráfica utilizando OpenGL, a ordem de especificação dos vértices em um desenho pode afetar significativamente a aparência visual do objeto. Sobre este tema, responda às seguintes perguntas de forma clara, sucinta e objetiva:

(a) (1 valor): O que é determinado pela ordem dos vértices?

Determina para onde a face "aponta" se está virada para fora ou para dentro. Em algumas primitivas a ordem afeta o resultado final.

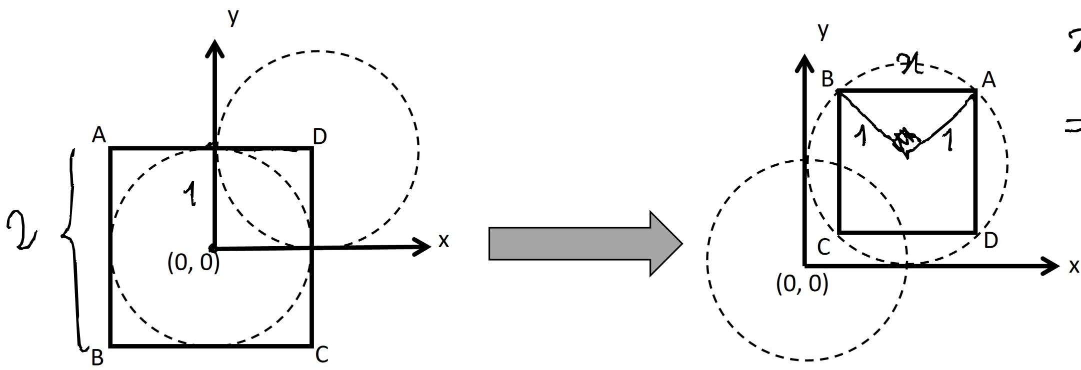
(b) (2 valores): Determine uma estratégia visual que permita saber se um objeto está definido com a ordem *correta* de vértices. (observação: *correta* aqui significa que o resultado visual corresponde ao esperado pelo desenvolvedor)

Uso da regra da mão direita, GL_CULL_FACE para ver se as faces estão viradas para o observador corretamente

Geometria e transformações

Q3 (3 valores)

Determine as transformações geométricas necessárias para transformar o quadrado $ABCD$ circunscrito ao círculo de raio 1 centrado na origem, no quadrado inscrito ao círculo de raio 1 que tangencia perfeitamente os eixos x e y, de forma a manter a ordem dos vértices tal qual representado na figura.



$x^2 = 2$

$\Rightarrow x = \sqrt{2}$

$\frac{\sqrt{2}}{2}$

quadrado ABCD circunscrito em r=1 → quadrado ABCD inscrito em r=1

(a) (1 valor): Indicação das transformações com valores e ordem correta.

(ex: Translação de 10 em x pode ser escrita como T(10, 0))

$R(-90) \rightarrow S\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 1\right) \longrightarrow T(1, 1)$

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & 0 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-90) & -\sin(-90) & 0 & 0 \\ \sin(-90) & \cos(-90) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) (2 valores): Representação da matriz final (M) resultante das transformações.

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & 0 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-90) & -\sin(-90) & 0 & 0 \\ \sin(-90) & \cos(-90) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Epa FDS calculem vocês!

Visualização, projeção e recorte

Q4 (3 valores)

Considere uma aplicação desenvolvida em OpenGl e configurada com uma janela quadrada (largura=altura). Mais, considere que no inicio do programa é configurado o recorte (glViewPort) para toda a tela e que seu volume de projeção é configurado através da função $glOrtho(left, right, bottom, top, near, far)$ com os seguintes valores:

$glOrtho(-1, 1, -1, 1, -2, 2)$.

(a) (1 valor): Considerando que a matriz Modelview está em suas condições iniciais (identidade). Quais valores de x, y, z podemos atribuir à um vértice de forma a garantir que o mesmo se encontra dentro do volume de projeção?

$-1 < x < 1 \quad -1 < y < 1 \quad -2 < z < 2$

Considere agora que, para além da projeção, é também definida uma vista da cena utilizando o algoritmo UVN implementado na função $lookat(\vec{pos}, \vec{target}, \vec{up})$ com os seguintes valores:

$lookat(0, 0, 1, 0, 0, 0, 0, 1, 0)$

E, alteram-se os valores de recorte, utilizando a função $glViewport(x0, y0, width, height)$ com os seguintes valores:

$glViewport(0, \frac{h}{2}, \frac{w}{4}, \frac{h}{2})$, onde w e h referem à largura e altura da janela da aplicação.

Após estas definições é então desenhada uma linha entre os vértices $A(-\frac{1}{2}, 0, -1)$ e $B(-\frac{1}{2}, 0, 1)$.

(b) (1 valor): Qual a distância entre os vértices A e B em coordenadas mundo? (justifique)

$$d_{AB} = \sqrt{0^2 + 0^2 + 4} = 2 \;//$$

(c) (1 valor): Qual a distância entre os vértices A e B em pixels na janela da aplicação? (justifique)

$$\text{Modelview} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad M_{proj} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$d_{A,B} = 0$ => os pontos estão em 2D e só diferem em Z logo na prespetiva otogonal estão no mesmo sítio

Iluminação e textura

Q5 (6 valores)

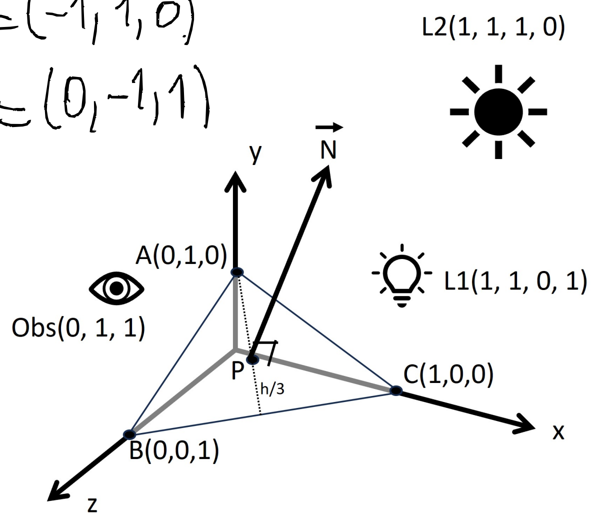
A imagem abaixo mostra uma cena realizada em OpenGl, utilizando o modelo de Phong teórico para o cálculo de iluminação. A cena é composta por um triângulo $\triangle ABC$, duas fontes de luz, L_1 e L_2, e um observador situado na posição $obs(0, 1, 1)$. Os vértices do triângulo têm materiais definidos com as cores: A(1, 0, 0), B(0, 1, 1), C(0, 1, 1); os coeficientes de reflexão ambiente k_A, difusa k_D e especular k_S dos materiais dos vértices são iguais a 1 ($k_A = k_D = k_S = 1$) e os coeficientes de especularidade ns também valem 1 ($ns = 1$). A normal $\vec{N}$ é a mesma em todos os vértices e é perpendicular à face do triângulo. As fontes de luz estão configuradas conforme especificado abaixo.

(1, 1, 1)

$\vec{CA} = (-1, 1, 0)$
$\vec{AB} = (0, -1, 1)$

$L_{1_{pos}} = (1, 1, 0, 1)$
$L_{1_{amb}}(R, G, B) = (0, 0, 0)$
$L_{1_{dif}}(R, G, B) = (1, 0, 0)$
$L_{1_{spec}}(R, G, B) = (1, 0, 0)$

$L_{2_{pos}} = (0, 1, 0, 0)$
$L_{2_{amb}}(R, G, B) = (0, 0, 0)$
$L_{2_{dif}}(R, G, B) = (0, 1, 1)$
$L_{2_{spec}}(R, G, B) = (0, 0, 0)$



(a) (2 valores): Qual é o vértice com maior intensidade de luz? (justifique)

Pontual direcional

i	R	G	B	L1	L2
A	1	0	0	S	N
B	0	1	1	N	S
C	0	1	1	N	S

$\vec{N} = \vec{AC} \times \vec{AB} = (1,1,1)$

$I_B = I_C = (0,1,1) \times (0,1,1) \times \cos\theta =$
$= (0,1,1)$

Como I_A será no máximo $(1,0,0)$ então B e C > intensidade

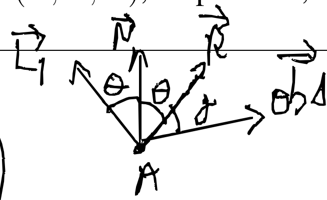
(b) (2 valores): Qual é o vértice com menor intensidade de luz? (justifique)

A

$\vec{ob1} = (0,1,1) - (0,1,0)$
$= (0,0,1)$

$P_A \rightarrow (0,1,0)$ ob1 $\left(0, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

(c) (2 valores): Considerando que a cena foi construída utilizando a opção $glShadeModel(GL\_SMOOTH)$, qual a cor e intensidade de luz, em valores (R, G, B), no ponto P, situado em $\frac{1}{3}$ da mediana h do triângulo (conforme a figura)? (justifique)



$\vec{N} = (1,1,1)$ Cálculo de I_A

$\vec{L_1} = L_1 - A = (1,1,0) - (0,1,0) = (1,0,0)$

$I_A = I_{dif} K_{dif} \cos\theta + I_{spec} K_{spec} \cos\gamma^{n1}$

$= (1,0,0)(1,0,0)\frac{\sqrt{3}}{3} + (1,0,0)(1,0,0) \cdot \frac{2}{3}$

$= (\sqrt{3}/3, 0, 0) + (2/3, 0, 0)$

$\cos\theta = \frac{\vec{L_1} \cdot \vec{N}}{\|\vec{L_1}\| \|\vec{N}\|} =$

$= \frac{1}{\sqrt{3} \times 1} = \frac{\sqrt{3}}{3}$

$\cos\gamma = \frac{\vec{R} \cdot \vec{ob1}}{\|\vec{R}\| \|\vec{ob1}\|}$

$= \frac{2}{3 \times 1}$

$= \frac{2}{3}$

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

$= 2(1)(1,1,1) - (1,0,0)$

$= (1,2,2)$

Interpolar com Gouraud deixo para os curiosos

Q6 (3 valores)

Complete o pseudo-código com as coordenadas de textura e configuração adequada para obter os resultados conforme as imagens.

observação 1: os parâmetros de configuração podem ser em pseudo-código, mas devem ser claros e coerentes com as configurações reais possíveis.

observação 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.

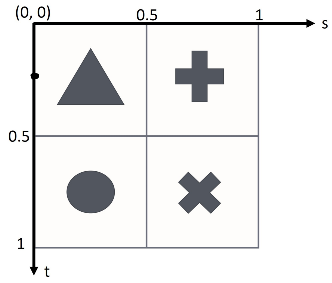


Figura 1: imagem original

```
texParameter(GL_TEXTURE_WRAP_S, GL_CLAM-TO_BORDER);
texParameter(GL_TEXTURE_WRAP_T, GL_CLAMP-TO-BORDER);
glBegin(GL_POLYGON);
texCoord(0, 1/4);   vertex_A(-0.5, 0.25, 0);
texCoord(0, 3/4);   vertex_B(-0.5, -0.25, 0);
texCoord(1/4, 1);   vertex_C(-0.25, -0.5, 0);
texCoord(3/4, 1);   vertex_D(0.25, -0.5, 0);
texCoord(1, 3/4);   vertex_E(0.5, -0.25, 0);
texCoord(1, 1/4);   vertex_F(0.5, 0.25, 0);
texCoord(3/4, 0);   vertex_G(0.25, 0.5, 0);
texCoord(1/4, 0);   vertex_H(-0.25, 0.5, 0);
glEnd();
```