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

deis departamento de engenharia informática 1995 – 2020

LEI - Computação Gráfica

prof. André Perrotta, prof. Hugo Amaro

Exame Normal

Nome: _____

Número: _____

Duração: 90min | 14 de Janeiro, 2025 | 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:

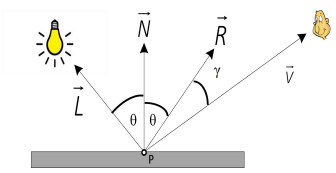
$$\vec{A} \times \vec{B} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2 b_3 - b_2 a_3)\hat{x} + (a_3 b_1 - b_3 a_1)\hat{y} + (a_1 b_2 - b_1 a_2)\hat{z}$$

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(θ)	0	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{\sqrt{3}}{2}$	1
cos(θ)	1	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	0
tan(θ)	0	$\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$	undefined

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

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

Implementação das funções $perspective(...)$ e $lookat(...)$.(implementação utilizada nas aulas PL)

```
void perspective(
GLfloat theta,
GLfloat alpha,
GLfloat beta,
bool invertX = false,
bool invertY = false
){
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  GLfloat tan = tanf(theta*0.5 * PI / 180.0);
  GLfloat d = (gh() / 2.0) / tan;
  GLfloat nearClip = d / alpha;
  GLfloat farClip = d * beta;
  GLfloat ymax = nearClip * tan;
  GLfloat xmax = (gw() / gh()) * ymax;
  if (invertX) {
        xmax = -xmax;
  }
  if (invertY) {
        ymax = -ymax;
  }
  glFrustum(-xmax, xmax,-ymax, ymax, nearClip, farClip);
}
```

```
void lookat(
GLfloat camX,
GLfloat camY,
GLfloat camZ,
GLfloat targetX,
GLfloat targetY,
GLfloat targetZ,
GLfloat upX,
GLfloat upY,
GLfloat upZ
){
ofVec3f cam = ofVec3f(camX, camY, camZ);
ofVec3f target = ofVec3f(targetX, targetY, targetZ);
ofVec3f up = ofVec3f(upX, upY, upZ);


ofVec3f N = cam - target;
N = N.normalized();
ofVec3f U = cross(up, N);
U = U.normalized();
ofVec3f V = cross(N, U);
V = V.normalized();


GLfloat camTransformMatrix[4][4] = {
        {U.x, V.x, N.x, 0},
        {U.y, V.y, N.y, 0},
        {U.z, V.z, N.z, 0},
        {-U.dot(cam), -V.dot(cam), -N.dot(cam), 1}
};


glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(&camTransformMatrix[0][0]);
}
```

Conceitos

Q1 (2 valores)

Na conceptualização e implementação de uma cena 3D utilizando o pipeline poligonal do OpenGL, como é possível ao utilizador definir a origem de um modelo (ou forma geométrica) antes das operações de transformação? Explique também como essa escolha afeta as futuras transformações aplicadas ao modelo. Esperam-se respostas sucintas e precisas.

Q2 (2 valores)

As coordenadas homogêneas são utilizadas em diversas etapas do pipeline de renderização poligonal do OpenGL, porém, há uma etapa onde esta utilização se mostra imprescindível. Qual é esta etapa e por que as coordenadas homogêneas são fundamentais em suas operações?

Geometria e transformações

Q3 (1 valores)

Determine uma rotina de desenho OpenGL (vértices e respetiva primitiva de desenho) que permita desenhar a figura representada na imagem. Utilize código ou pseudo-código OpengL. (Atenção ao desenho da figura com preenchimento (GL_FILL, cinza) e sem preenchimento (GL_LINE, preto), que deve poder ser obtido sem alteração da primitiva de desenho escolhida).

resp:

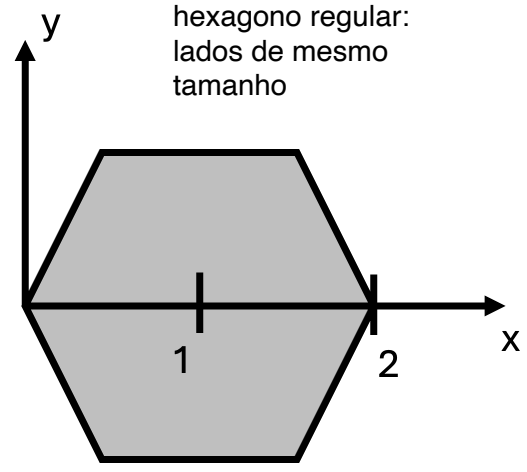
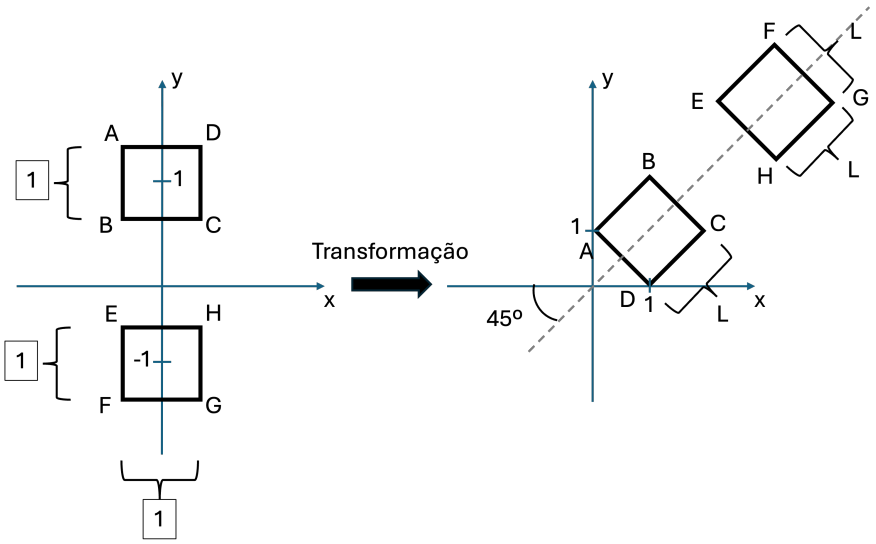


figura geométrica: em preto seu contorno, em cinza seu preenchimento

Q4 (3 valores)

Determine as transformações geométricas necessárias para transformar os quadrados unitários $ABCD$ e $EFGH$, centrados em $(0, 1, 0)$ e $(0, -1, 0)$ respectivamente, conforme mostra a figura.
obs: não é possível aplicar transformações independentes para cada quadrado.
Indique a sequência de transformações das seguintes formas:



$\triangle BCE \rightarrow \triangle CDE$

(a) (1 valor): Indicação das transformações com valores e ordem correta.
(ex: Translação de 10 em x pode ser escrita como T(10, 0))

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

Visualização, projeção e recorte

Q5 (4 valores)

Considere o jogo *Flower Shooter*, onde os jogadores podem arremessar flores uns nos outros, desenvolvido em Openframeworks/OpenGL e configurado para uma janela de aplicação quadrada de $W = H$ pixels. No jogo, as personagens são desenhadas utilizando quadrados unitários com textura, e a configuração inicial em coordenadas mundo é determinada com transformações geométricas. O resultado final (rotação, escala, transformação) pode ser observado na figura 1.

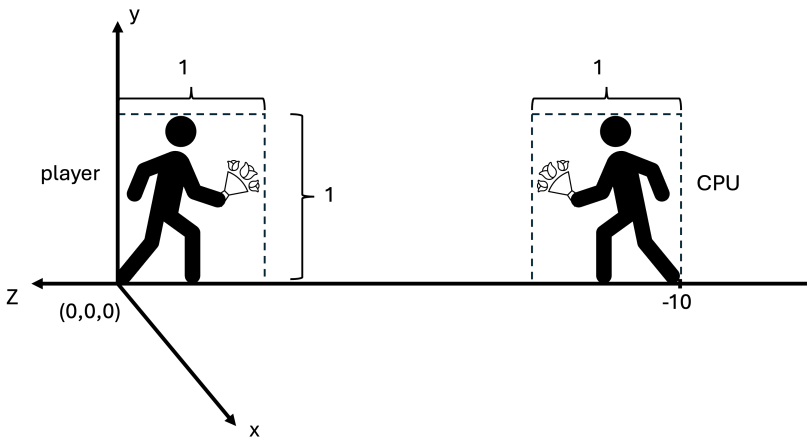


Figura 1: Flower Shooter, personagens em coordenadas mundo

Queremos visualizar o jogo em vista ortográfica, de maneira que o limite inferior da janela da aplicação funcione como o chão para as personagens, e que elas fiquem posicionadas exatamente nos cantos inferiores (ou seja, a distância entre os vértices mais distantes do quadrado de cada personagem, ocupa exatamente a largura da janela), conforme mostra a figura 2.

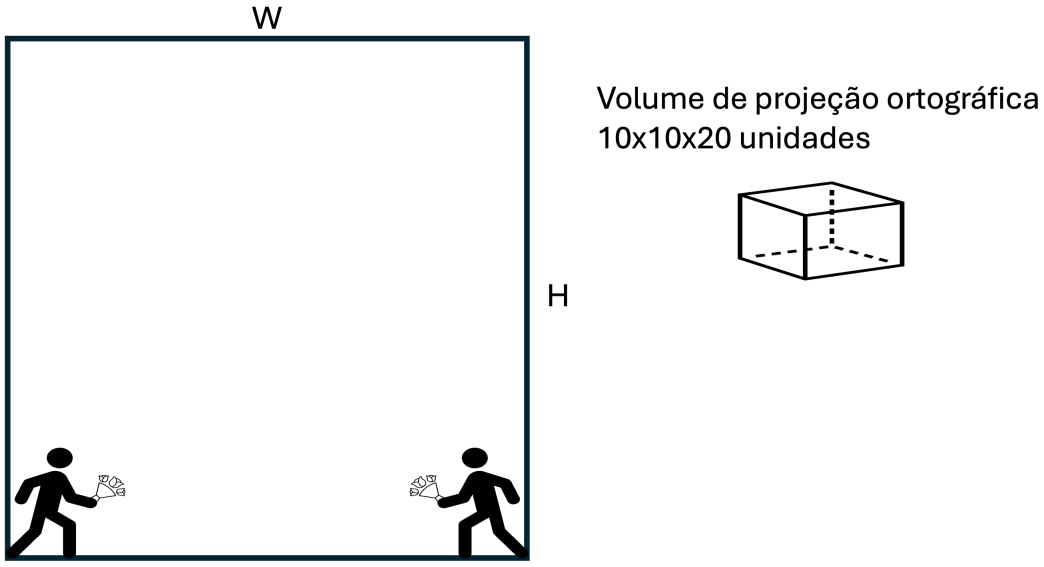


Figura 2: Flower Shooter, vista ortográfica do jogo

a)(2 valores) Determine uma possível configuração das funções $glOrtho(left, right, bottom, top, near, far)$ e $lookat(\vec{pos}, \vec{target}, \vec{up})$, para que essa visualização seja obtida? Considere um volume de projeção ortográfica de 10x10x20 unidades. Justifique sua resposta.

b) (2 valores) Numa segunda fase do jogo, as personagens desenhadas por quadrados serão substituídas por cubos unitários com textura, e define-se uma projeção perspectiva utilizando a função $perspective(theta, 100, 1000)$. Para que seja mantida extamente a mesma visualização da vista ortográfica, mas agora com perspectiva, como devemos configurar a função $lookat(\vec{pos}, \vec{target}, \vec{up})$? A resposta pode ser dada com valores em função de $W, H, theta$. Justifique sua resposta.

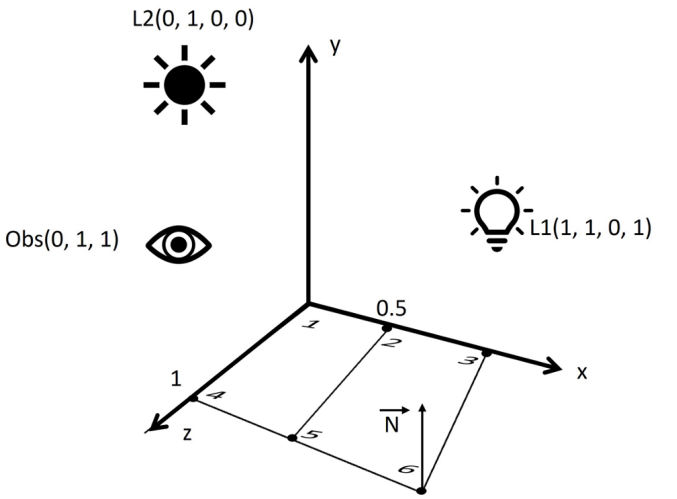
Iluminação

Q6 (4 valores)

A imagem abaixo mostra uma cena realizada em OpenGl formada por um malha de retângulos de resolução (2, 1), duas fontes de luz, L_1 e L_2, e um observador situado na posição $obs(0, 1, 1)$. A malha foi construida de forma a atribuir o mesmo material para todos os vértices, com valor $RGB = (1, 0, 1)$. Os coeficientes de reflexão ambiente k_A, difusa k_D e especular k_S do material são iguais a 1 ($k_A = k_D = k_S = 1$) e o coeficiente de especularidade ns também vale 1 ($ns = 1$). A normal $\vec{N}(0, 1, 0)$ é a mesma em todos os vértices. As fontes de luz estão configuradas conforme especificado abaixo.

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

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



a)(2 valores) Qual é a cor final RGB do vértice 1? (justifique sua resposta)

b)(2 valores) Qual é a cor final RGB do vértice com maior intensidade de luz (maior abs(RGB))? (justifique sua resposta)

Textura

Q3(4 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.

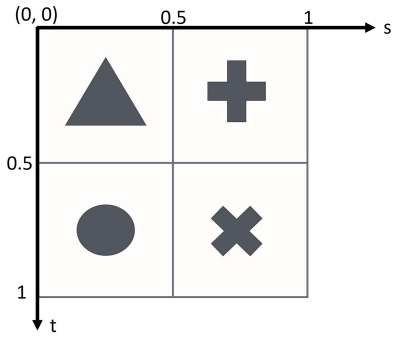
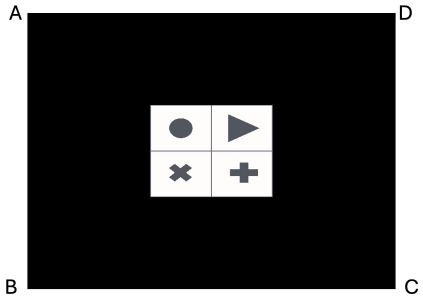


Imagem original

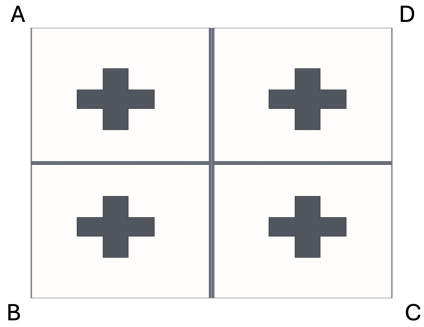
(a)(1 valor):

```
texParameter(GL_TEXTURE_WRAP_S, ________________);
texParameter(GL_TEXTURE_WRAP_T, ________________);
glBegin(GL_QUADS);
texCoord(___,___); vertex_A(-0.5, 0.5, 0);
texCoord(___,___); vertex_B(-0.5, -0.5, 0);
texCoord(___,___); vertex_C(0.5, -0.5, 0);
texCoord(___,___); vertex_D(0.5, 0.5, 0);
glEnd();
```



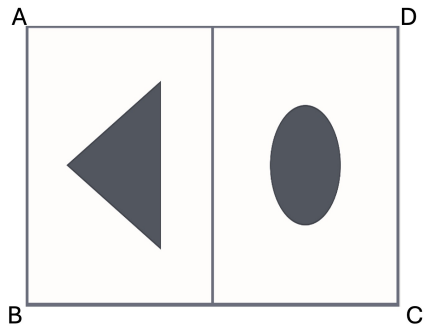
(b)(1 valor):

```
texParameter(GL_TEXTURE_WRAP_S, ________________);
texParameter(GL_TEXTURE_WRAP_T, ________________);
glBegin(GL_QUADS);
texCoord(___,___); vertex_A(-0.5, 0.5, 0);
texCoord(___,___); vertex_B(-0.5, -0.5, 0);
texCoord(___,___); vertex_C(0.5, -0.5, 0);
texCoord(___,___); vertex_D(0.5, 0.5, 0);
glEnd();
```



(c)(1 valor):

```
texParameter(GL_TEXTURE_WRAP_S, ________________);
texParameter(GL_TEXTURE_WRAP_T, ________________);
glBegin(GL_QUADS);
texCoord(___,___); vertex_A(-0.5, 0.5, 0);
texCoord(___,___); vertex_B(-0.5, -0.5, 0);
texCoord(___,___); vertex_C(0.5, -0.5, 0);
texCoord(___,___); vertex_D(0.5, 0.5, 0);
glEnd();
```



(d)(1 valor):

```
texParameter(GL_TEXTURE_WRAP_S, ___________________);
texParameter(GL_TEXTURE_WRAP_T, ___________________);
glBegin(GL_QUADS);
texCoord(___,___); vertex_A(-0.5, 0.5, 0);
texCoord(___,___); vertex_B(-0.5, -0.5, 0);
texCoord(___,___); vertex_C(0.5, -0.5, 0);
texCoord(___,___); vertex_D(0.5, 0.5, 0);
glEnd();
```