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

Frequência (mini-teste) :

vetores & álgebra linear
transformações geométricas
projeção e visualização

Nome: _____

Número: _____

Duração: 60min | 30 de Outubro, 2024 | valor max: 20 | TIPO: A

Instruções

- Assinale apenas uma alternativa em cada questão.
- Não há descontos para questões erradas.

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:

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

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

	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

Implementação das funções $perspective(...)$ e $lookat(...)$, $rect()$ e $axis()$ (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]);
}
```

```
void rect(){
  glBegin(GL_QUADS);
  glVertex3f(-0.5, 0.5, 0.);
  glVertex3f(-0.5, -0.5, 0.);
  glVertex3f(0.5, -0.5, 0.);
  glVertex3f(0.5, 0.5, 0.);
  glEnd();
}
```

```
void axis() {
  glBegin(GL_LINES);
  glVertex3f(0, 0, 0);
  glVertex3f(1, 0, 0);
  glVertex3f(0, 0, 0);
  glVertex3f(0, 1, 0);
  glVertex3f(0, 0, 0);
  glVertex3f(0, 0, 1);
  glEnd();
}
```

Vetores e álgebra linear

Q1(2 valores):

Considere um ponto A, definido em coordenadas cartezianas como $A = (x, y, z)$. Assinale a alternativa que NÃO representa o ponto A em coordendas homogênas.

(a) $A = (xw, yw, zw, w)$

(b) $A = (x, y, z, 1)$

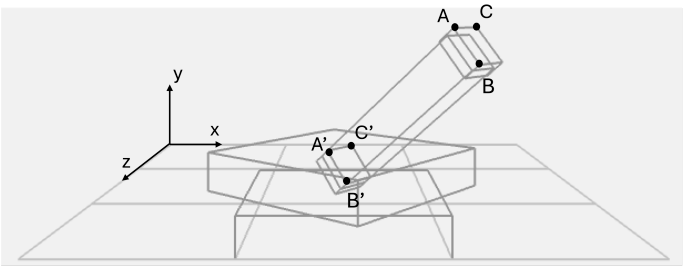
(c) $A = x\hat{i} + y\hat{j} + z\hat{k} + \hat{w}$

(d) $A = [x\, y\, z\, 0]$

(e) $A = \begin{bmatrix} 4x \\ 4y \\ 4z \\ 4 \end{bmatrix}$

Q2(2 valores):

O tanque desenhado na figura abaixo foi implementado com Openframeworks/OpenGL. Conhecendo-se as coordendas (x, y, z) dos vértices A, B, C, A', B', C', queremos calcular a direção e sentido do vetor velocidade de um tiro que é disparado pelo canhão. Assinale a alternativa que contém uma forma possível de calcular o vetor velocidade.



hfill

(a) $\vec{vel} = \{(A - B) + (A - C) + (A' - B') + (A' - C')\}\, \frac{1}{2}$

(b) $\vec{vel} = ||(A - B')||$

(c) $\vec{vel} = (C - A) \times (C' - A')$

(d) $\vec{vel} = \frac{(B - B')}{\sqrt{(B_x - B'_x)^2 + (B_y - B'_y)^2 + (B_z - B'_z)^2}}$

(e) $\vec{vel} = (C - A) \bullet (B - A)$

Transformações

Q3(2 valores):

A matriz ao lado representa uma combinação de transformações euclideanas (transformações de corpo rígido, que preservam os ângulos e distâncias entre vértices e arestas) em coordenadas homogêneas. Assinale a alternativa que descreve uma relação verdadeira entre os diversos elementos da matriz e que valha para qualquer combinação de transformações euclideanas.

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

(a) $a = b, -d = -e$

(b) $g = h = i = 1$

(c) $g = h = 0$

(d) $a = e = i$

(e) $a = -e$

Q4(2 valores):

Uma aplicação desenvolvida em OF/OpenGl utiliza a função *rect()* (ver pág 2) e uma série de transformações para desenhar um retângulo. Conhecesendo-se o estado da matriz *Modelview* (M) no momento em que a função *rect()* é chamada, qual é a posição final da coordenada da origem do retângulo em coordenadas do mundo (x, y, z)?

$$M = \begin{bmatrix} 135.7 & -135.7 & 0.86 & -384 \\ 271.5 & 271.5 & 0 & 384 \\ -235.1 & 235.1 & 0.5 & -665.107 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) $(x, y, z) = (0, 0, 0)$

(b) $(x, y, z) = (135.7, -135.7, 0.86)$

(c) $(x, y, z) = (135.7, 271.5, 0.5)$

(d) $(x, y, z) = (384, -384, 665.107)$

(e) $(x, y, z) = (-384, 384, -665.107)$

Q5(2 valores):

Uma aplicação desenvolvida em OF/OpenGl utiliza a função *rect()* (ver pág 2) e uma série de transformações para desenhar um retângulo. Conhecesendo-se o estado da matriz *Modelview* (M) no momento em que a função *rect()* é chamada, qual é a posição final do primeiro vértice do retângulo em coordenadas do mundo (x, y, z)?

$$M = \begin{bmatrix} 2 & 0 & 0 & 10 \\ 0 & 2 & 0 & 10 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) $(x, y, z) = (0, 0, 0)$

(b) $(x, y, z) = (10, 10, -1)$

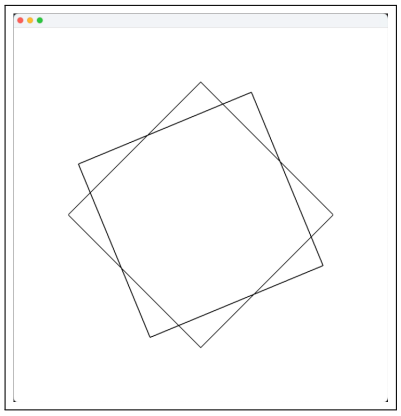
(c) $(x, y, z) = (20, 20, -1)$

(d) $(x, y, z) = (9, 11, -1)$

(e) $(x, y, z) = (18, 22, -1)$

Q6(2 valores):

A figura à seguir foi desenhada com uma aplicação implementada em OF/OpenGL utilizando as configuração de defeito do OF e tamanho de janela *W x W*. Assinale a alternativa que completa corretamente os parâmetros (A, B, C, D) das transformações geométricas utilizadas na rotina de desenho implementada no método ofApp::draw().



```
void ofApp::draw(){
  glPushMatrix();
  glTranslatef(A, B, 0.);
  glRotatef(C, 0, 0, 1);
  glScalef(gw()*0.5, gh()*0.5, 1.);
  rect();
  glRotatef(D, 0, 0, 1);
  rect();
  glPopMatrix();
}
```

$(A, B, C, D) =$

(a) $(W, W, 90, 45)$

(b) $(W/2, W/2, -45, 22.5)$

(c) $(0, 0, -45, 45)$

(d) $(W/2, -W/2, 0, 45)$

(e) $(-W/2, -W/2, 30, 60)$

Câmera, projeção e visualização

Q7(2 valores):

Considere uma aplicação desenvolvida em OF/OpenGL, configurada para uma janela de tamanho *W x W*, em que a matriz de projeção está configurada com a função $perspective(60, 10, 1000)$, e a matriz Modelview está configurada utilizando a função $lookat(0, 0, \frac{W}{tan(\frac{\pi}{6})}, 0, 0, 0, 0, 1, 0)$. Após estas configurações, queremos utilizar a função $rect()$ para desenhar um retângulo no centro da tela à ocupar exatamente toda a janela da aplicação. Para isto ser possível devemos escalar e transladar o retângulo. Assinale a alternativa que indica os valores corretos das operações de escala e translação.

(a) $S(2, 2, 1),\ T(0, 0, 0)$

(b) $S(W/2, W/2, 1),\ T(W/2, W/2, 0)$

(c) $S(2W, 2W, 1),\ T(0, 0, 0)$

(d) $S(W/4, W/4, 1),\ T(W/2, W/2, 0)$

(e) $S(W/4, W/4, 1),\ T(0, 0, 0)$

Q8(2 valores):

O trecho de código à seguir é utillizado para desenhar uma vista ortográfica de um cubo em OF/OpenGl. Qual face do cubo será desenhada na tela?

```
void ofApp::draw(){
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1, 1, -1, 1, -2, 2);
    lookat(1, 0, 0, 0, 0, 0, 0, 1, 0);
    cube_unit();
}
```

(a) face esquerda

(b) face frontal

(c) face de cima

(d) face direita

(e) n.d.a

Q9(2 valores):

Analise o trecho de código OF/OpenGl e assinale a alternativa correta.

```
void ofApp::draw(){
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-1, 1, -1, 1, -2, 2);
  lookat(0, 0, 1, 0, 0, 0, 0, 1, 0);
  glTranslatef(0, 0, -10);
  cube_unit();
}
```

(a) o cubo será desenhado no centro da tela.

(b) o cubo será desenhado com seu centro no canto esquerdo superior da tela.

(c) o cubo não será desenhado, pois a câmera definida com lookat está apontada na direção oposta.

(d) o cubo não será desenhado, pois está transladado para fora do volume de projeção.

(e) n.d.a

Q10(2 valores):

O código à seguir é parte de uma aplicação OF/OpenGL configurada para uma janela de tamanho 500x500 (parâmetro definido em ofMain.h). Analise o código e assinale a alternativa correta.

```
void ofApp::draw(){
  glViewport(200, 200, 100, 100);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-1, 1, -1, 1, -2, 2);
  lookat(0, 0, 1, 0, 0, 0, 0, 1, 0);
  glColor3f(1, 0, 0);
  glBegin(GL_LINES);
  glVertex3f(-1, 1, 0);
  glVertex3f(1, -1, 0);
  glEnd();
}
```

(a) é desenhada uma linha de cor vermelha, do canto direito superior ao canto esquerdo inferior da janela.

(b) a distância entre os vértices na janela da aplicação será de $100\sqrt{2}$ pixels.

(c) a distância entre os vértices em coordenadas do mundo é 2.

(d) é desenhada uma linha de cor azul, à partir do meio da tela até o canto direito inferior.

(e) n.d.a