

Geral

Formatos de Ficheiros

2016-2017 (Recurso)

[2.5] Os formatos **BMP** e **SVG (Scalable Vector Graphics)** de representação de gráficos constituem exemplos de

- i. Representações matriciais
- ii. Representações vectoriais
- iii.** Uma representação matricial e uma vectorial, respectivamente
- iv. Uma representação vectorial e uma matricial, respectivamente

WIMP (Windows, Icons, Menus, Pointer)

2018-2019 (Recurso)

[3.3] Da comparação de uma interface WIMP (*Windows, Icons, Menus, Pointer*) com uma de linha de comandos resulta normalmente que

- i.** A segunda está mais vocacionada do que a primeira para ser usada por utilizadores experientes
- ii. O esforço de memorização e a carga cognitiva impostos aos utilizadores são menores na segunda do que na primeira
- iii. A probabilidade de ocorrência de erros de interacção é menor na segunda do que na primeira
- iv. Todas as anteriores

2018-2019 (Normal)

[3.3] O *design* de uma interface WIMP (*Windows, Icons, Menus, Pointer*)

- i. Não deve contemplar o uso de teclas aceleradoras, pois o mesmo implica um esforço de memorização considerável por parte do utilizador
- ii. Deve ser centrado no sistema e não no utilizador, pois o comportamento deste último é imprevisível
- iii. Não deve basear-se em analogias com o mundo real, pois pode suscitar confusão na mente do utilizador
- iv.** Nenhuma das anteriores

2017-2018 (Normal)

[2.5] Da comparação de uma interface WIMP (*Windows, Icons, Menus and Pointers*) com uma de linha de comandos resulta normalmente que

- i. A primeira está mais vocacionada do que a segunda para ser usada por utilizadores experientes
- ii. O esforço de memorização e a carga cognitiva impostos aos utilizadores são menores na primeira do que na segunda
- iii. A probabilidade de ocorrência de erros de interacção é maior na primeira do que na segunda
- iv. Nenhuma das anteriores

Frame Buffer RGBA

2018-2019 (Recurso)

[3.3] Um sistema gráfico dotado de um *frame buffer* RGBA de 1024 x 1024 x 32 bits

- i. Permite a reprodução de imagens compostas por mais do que 1 milhão de píxeis
- ii. Permite a reprodução de imagens com 1024 níveis de vermelho, 1024 níveis de verde e 32 níveis de azul
- iii. Permite a reprodução de imagens com 2^{32} milhões de cores
- iv. Todas as anteriores

2017-2018 (Recurso)

[2.5] Qual a dimensão em bytes de um *frame buffer* RGBA de 1024 x 1024 x 16 bits?

- i. 0.5 Megabyte
- ii. 1 Megabyte
- iii. 2 Megabyte
- iv. Nenhuma das anteriores

1 megabyte = 1.000.000 bytes

1024 x 1024 = apox. 1.000.000 e como tem 16 bits em vez de 8, temos q multiplicar por 2

2017-2018 (Normal)

[2.5] Num sistema gráfico dotado de um *frame buffer* RGBA de 32 bits/píxel

- i. Cada píxel é descrito por 8 bits para a componente vermelha, 8 bits para a verde, 8 bits para a azul e 8 bits para o canal alfa
- ii. É possível reproduzir imagens com $2^8 = 256$ níveis de vermelho, 256 níveis de verde e 256 níveis de azul
- iii. É possível reproduzir imagens de objectos transparentes
- iv. Todas as anteriores

2016-2017 (Recurso)

[2.5] Qual a dimensão em bytes de um *frame buffer* RGB de $1024 \times 512 \times 8$ bits?

- i. 512 byte
- ii. 512 Kilobyte
- iii. 512 Megabyte
- iv. Nenhuma das anteriores

Matrizes

2017-2018 (Recurso)

[2.5] Qual das seguintes matrizes representa em coordenadas homogéneas o ponto $(2, -3, 4)$?

- i. $[2.0, -3.0, 4.0, 0.0]^T$
- ii. $[4.0, -6.0, 8.0, 1.0]^T$
- iii. $[8.0, -12.0, 16.0, 4.0]^T$
- iv. Nenhuma das anteriores

Porque $2^4 = 8$, $-3^4 = -12$ e $4^4 = 16$

2016-2017 (Recurso)

[2.5] Qual das seguintes matrizes representa o ponto de coordenadas $(3, -1, 8)$?

- i. $[3.0, -1.0, 8.0, 0.0]^T$
- ii. $[3.0, -1.0, 8.0, 1.0]^T$
- iii. $[3.0, -1.0, 8.0, 2.0]^T$
- iv. Nenhuma das anteriores

2016-2017 (Normal)

[2.5] Se se efectuar uma operação de divisão perspectiva do ponto de coordenadas homogéneas $[1.0, 2.0, 3.0, 4.0]^T$ ir-se-á obter

- i. $[4.0, 8.0, 12.0, 1.0]^T$
- ii.** $[0.25, 0.5, 0.75, 1.0]^T$
- iii. $[-3.0, -2.0, -1.0, 0.0]^T$
- iv. Nenhuma das anteriores

Divisão perspectiva é por o w a zero?

Geometria / Transformações Geométricas

Transformação Linear

- Se um conjunto de pontos pertence a uma recta, depois de transformados eles também pertencerão a uma recta
- Se um ponto P guarda uma relação de distância com dois outros pontos Q e R, então essa relação de distância é mantida pela transformação

Transformação	Faz corresponder a origem à origem?
Transformação Linear	Sim
Transformação Linear Afim	Não (translações são permitidas)

Tipos de Transformações

- Transformação identidade
- Transformação rígida
 - Não modificam a forma (dimensões/ângulos) do objeto
 - São compostas por uma rotação e uma translação
- Transformação homotética (ou ortogonal)
 - Preserva os ângulos

Coordenadas Homogéneas

- Para **vectores**: coordenada extra $w = 0$;
- Para **pontos**: coordenada extra $w = 1$; (memorize: ponto tem um P que parece um 1)

Perguntas

2018-2019 (Recurso)

[3.3] As transformações perspectivas

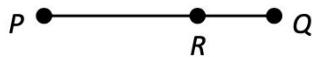
- i. Obrigam a que o centro de projecção esteja sempre posicionado na origem
- ii. São incompatíveis com o uso de coordenadas homogéneas
- iii. Preservam as combinações afins
- iv. Nenhuma das anteriores

[3.3] A transformação que resulta da composição de uma translação e de uma rotação

- i. Não depende da ordem pela qual a translação e a rotação são efectuadas
- ii. É rígida
- iii. Não preserva as dimensões e os ângulos dos objectos transformados
- iv. Nenhuma das anteriores

2018-2019 (Normal)

[3.3] Dados dois pontos distintos P e Q e a combinação linear afim $R = (1 - \alpha)P + \alpha Q$, qual o valor de α para o qual o ponto R fica duas vezes mais afastado de P do que de Q ?



- i. $\alpha = -0.33$
- ii. $\alpha = 0.33$
- iii. $\alpha = 1 - 0.33$
- iv. Nenhuma das anteriores

0 ----- 1

[3.3] As transformações perspectivas

- i. Obrigam a que o centro de projecção esteja sempre posicionado na origem
- ii. São incompatíveis com o uso de coordenadas homogéneas
- iii. Preservam as combinações afins
- iv. Nenhuma das anteriores

2017-2018 (Recurso)

[2.5] Qual das seguintes transformações compostas é rígida?

- i. `glTranslated(1.0, 2.0, 3.0); glScaled(-1.0, -2.0, -3.0);`
- ii. `glRotated(0.0, 0.0, 0.0, 1.0); glScaled(0.0, 0.0, -1.0);`
- iii. `glScaled(1.0, 2.0, 4.0); glScaled(1.0, 0.5, 0.25);`
- iv. Nenhuma das anteriores

Porque o segundo scale anula o primeiro!

[2.5] Considere o objecto delimitado pela superfície descrita pela equação $x^2 + y^2 + z^2 - 1 = 0$. O ponto de coordenadas $(0.4, 0.6, 0.8)$ encontra-se

- i. No interior do objecto
- ii. Na fronteira do objecto
- iii. No exterior do objecto
- iv. Nenhuma das anteriores

Equação de uma esfera

Substituir o $x^2 + y^2 + z^2 - 1$ pelos números das coordenadas (visto que já está igualada a zero!!!!)

Se o valor for MAIOR que 0 -> Está fora

Se o valor for IGUAL a 0 -> Está na fronteira do objecto

Se o valor for MENOR que 0 -> Está dentro

[2.5] Qual a representação associada ao objecto referido na alínea anterior?

- i. Paramétrica
- ii. Implícita
- iii. CSG
- iv. Nenhuma das anteriores

Paramétrico ->

CSG -> Quando é por transformações sucessivas (árvore)

Implícita -> Quando temos uma equação

2017-2018 (Normal)

[2.5] O caso particular da transformação de escala `glScaled(1.0, 1.0, 1.0);` constitui um exemplo de

- i. Uma transformação identidade
- ii. Uma transformação rígida
- iii. Uma transformação homotética
- iv. Todas as anteriores

Homotética -> não muda os ângulos

Identidade -> A transformação não muda nada no objecto

Rígida -> Apenas porque o `glScaled` é aplicado com tudo a 1

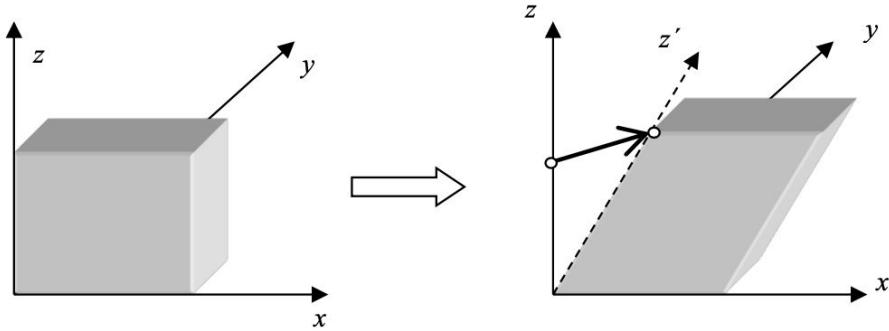
2016-2017 (Recurso)

[2.5] Qual das seguintes transformações é homotética?

- i. `glScaled(1.0, 2.0, 3.0);`
- ii. `glScaled(2.0, 2.0, 2.0);`
- iii. `glScaled(3.0, 2.0, 1.0);`
- iv. Nenhuma das anteriores

Porque não muda os ângulos do objeto.

[2.5] Qual das seguintes transformações permite deformar o objecto da maneira ilustrada na figura?



- i. Translação
- ii. Rotação
- iii. Escalamento
- iv. Nenhuma das anteriores

É uma shearing!

[2.5] Qual das seguintes equações descreve implicitamente a superfície de uma esfera unitária centrada na origem?

- i. $x^2 + y^2 - z^2 = 0$
- ii. $x^2 + y^2 + z^2 - 1 = 0$
- iii. $\frac{x^2}{4} + \frac{y^2}{9} + \frac{z^2}{16} = 1$
- iv. Nenhuma das anteriores

Como é uma esfera unitária, o raio vai ser 1. Então...

Em [geometria analítica](#), uma esfera com centro (x_0, y_0, z_0) e raio r é o [lugar geométrico](#) tal que:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2.$$

2016-2017 (Normal)

[2.5] Os formatos **BMP** e **SVG (Scalable Vector Graphics)** de representação de gráficos constituem exemplos de

- i. Representações matriciais
- ii. Representações vectoriais
- iii. Uma representação matricial e uma vectorial, respectivamente
- iv. Uma representação vectorial e uma matricial, respectivamente

2015-2016 (Recurso)

[2.5] A qual das seguintes sequências de transformações corresponde uma matriz de transformação composta igual à matriz identidade?

- i. glTranslated(1.0, 2.0, 3.0); glTranslated(-1.0, -2.0, -3.0);
- ii. glRotated(30.0, 0.0, 1.0, 0.0); glRotated(-30.0, 0.0, 1.0, 0.0);
- iii. glScaled(8.0, 4.0, 2.0); glScaled(0.125, 0.25, 0.5);
- iv. Todas as anteriores

A segunda em todas anula a primeira transformação.

2015-2016 (Recurso)

[2.5] Na representação de um ponto 3D em coordenadas homogéneas são usadas

- i. Apenas três componentes: x , y e z
- ii. Quatro componentes: x , y , z e w , em que $w = 0$
- iii. Quatro componentes: x , y , z e w , em que $w \neq 0$
- iv. Nenhuma das anteriores

Modelação

Técnicas de Codificação de Malhas Poligonais

Explícita

- Mais simples
- Cada face armazena a lista de vértices
- Redundância de informação

Ponteiros para listas de vértices

- Vértices armazenados separadamente

Ponteiros para listas de arestas

- Arestras armazenadas separadamente
- Introduzem-se referências para as duas faces que compartilham uma aresta
- Não desenha duas vezes a mesma aresta

Winged-Edge (Half-Edge, Face-Edge)

- Não desenha duas vezes a mesma aresta

Quad-Edge (Guibas-Stolfi)

Radial-Edge

Tipos de Representação de Sólidos

- **Por fronteira (B-rep – Boundary Representation)**
 - Intersecções estão representadas explicitamente
 - Mais fácil exibir um ponto sobre a superfície do objecto
 - Difícil determinar, dado um ponto, se ele está no interior, fronteira ou exterior do objecto
 - Operações booleanas são complicadas
- **Operações de conjuntos (CSG – Constructive Solid Geometry)**
- **Por enumeração do espaço em células (BSP-trees, Octrees, etc.)**
 - Poliedros convexos

Representação por Células

Dividem o espaço em sub-regiões convexas

- **Grelhas:** Cubos de tamanho igual
- **Octrees:** Cubos cujos lados são potências de 2
- **BSP-trees:** Poliedros convexos

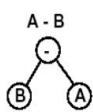
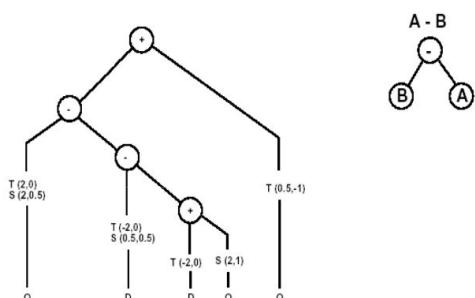
Árvore CSG (Constructive Solid Geometry)

Nós Internos

- Operações de conjunto
- Transformações lineares

Folhas

- Objectos primitivos (quadráticas)



T = Translação

S = Escalamento

+ = União de Conjuntos

- = Diferença de Conjuntos

Perguntas

2018-2019 (Recurso)

- d. [3.3] Numa árvore CSG (*Constructive Solid Geometry*)
- i. Os nós internos designam objectos primitivos
 - ii. As folhas designam operações booleanas ou transformações lineares afins
 - iii. Descer um nível corresponde a dividir o espaço 3D em oito octantes
 - iv. Nenhuma das anteriores

2018-2019 (Normal)

[3.3] Quais das seguintes técnicas de codificação de malhas poligonais permite desenhar eficientemente a malha sem que cada aresta seja desenhada duas vezes?

- i. Explícita e apontadores para uma lista de vértices
- ii. Apontadores para uma lista de vértices e apontadores para uma lista de arestas
- iii. Apontadores para uma lista de arestas e *Winged-Edge*
- iv. Nenhuma das anteriores

2017-2018 (Normal)

[2.5] Qual das seguintes técnicas de codificação de malhas poligonais permite desenhar a malha sem que cada aresta seja desenhada duas vezes?

- i. Explícita
- ii. Ponteiros para uma lista de vértices
- iii. Ponteiros para uma lista de arestas
- iv. Nenhuma das anteriores

2016-2017 (Normal)

[2.5] Na representação por fronteira de um objecto (*B-rep*)

- i. É difícil exibir um ponto sobre a superfície do objecto
- ii. É fácil determinar, dado um ponto, se o mesmo está no interior, na fronteira ou no exterior do objecto
- iii. As operações booleanas são complicadas de efectuar
- iv. Nenhuma das anteriores

Por fronteira (B-rep – Boundary Representation)

- Intersecções estão representadas explicitamente
- Mais fácil exibir um ponto sobre a superfície do objecto
- Difícil determinar, dado um ponto, se ele está no interior, fronteira ou exterior do objecto
- Operações booleanas são complicadas

[2.5] Nas representações de objectos por células

- i. As grelhas dividem o espaço em cubos cujos lados são potências de 2
- ii. As *octrees* dividem o espaço em cubos de igual dimensão
- iii. As *BSP-trees* dividem o espaço em poliedros convexos
- iv. Todas as anteriores

2015-2016 (Recurso)

[2.5] Na codificação de sólidos com base em ponteiros para uma lista de vértices

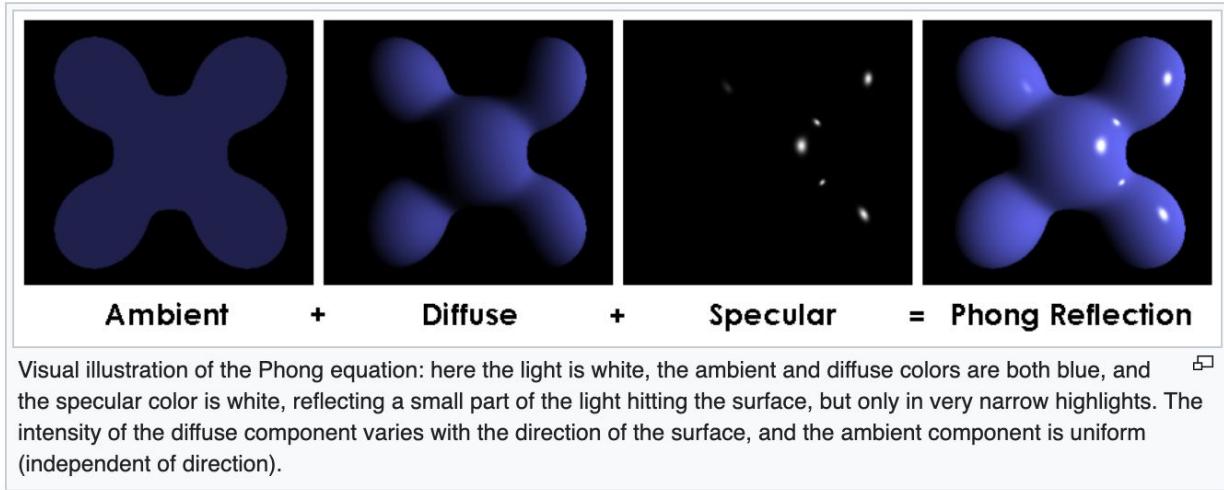
- i. Cada face do sólido armazena explicitamente a lista ordenada das coordenadas dos seus vértices
- ii. Há uma lista de vértices e as faces referenciam os seus vértices através de apontadores para essa lista
- iii. A redundância de informação é maior do que na codificação explícita
- iv. Todas as anteriores

[2.5] Nas representações de objectos por células

- i. As grelhas dividem o espaço em cubos de igual dimensão
- ii. As *octrees* dividem o espaço em cubos cujos lados são potências de 2
- iii. As *BSP-trees* dividem o espaço em poliedros convexos
- iv. Todas as anteriores

Illuminação

Modelo de Reflexão de Phong



Emissão

- Contribuição que não depende de fontes de luz (fluorescência)

Ambiente

- Contribuição que não depende da geometria

Difusa (ou reflexão lambertiana)

- É necessário o cálculo do vector normal
- Não depende da posição do observador

Especular

- É necessário o cálculo do vector normal
- Contribuição referente ao comportamento de superfícies polidas

Factor de atenuação da intensidade luminosa

- Constante (não depende da distância da fonte de luz ao objecto iluminado)
- Linear (proporcional à distância da fonte de luz ao objecto iluminado)
- Quadrático (proporcional ao quadrado da distância da fonte de luz ao objecto iluminado)

Perguntas

2018-2019 (Recurso)

- e. [3.3] O conhecimento do vector normal é necessário ao cálculo
- i. Das componentes ambiente e difusa de iluminação
 - ii. Das componentes ambiente e especular de iluminação
 - iii. Das componentes difusa e especular de iluminação
 - iv. Das componentes ambiente, difusa e especular de iluminação

2018-2019 (Normal)

- e. [3.3] A componente difusa do modelo de iluminação de Phong
- i. Só pode ser definida para as fontes de luz direcionais
 - ii. É característica de materiais tais como o metal brilhante
 - iii. Não depende da posição do observador
 - iv. Nenhuma das anteriores

2017-2018 (Recurso)

[2.5] Uma forma de determinar o vector normal a um polígono planar consiste em

- i. Calcular o produto escalar dos vectores definidos por duas arestas do polígono e dividir o resultado obtido pelo somatório dos comprimentos dos vectores
- ii. Projectar o polígono nos planos OYZ, OZX e OXY e calcular as áreas dos polígonos resultantes; as componentes da normal serão proporcionais a estes valores
- iii. As respostas i. e ii.
- iv. Nenhuma das anteriores

2017-2018 (Normal)

[2.5] No modelo de iluminação do OpenGL é possível definir um factor de atenuação da intensidade luminosa

- i. Constante, isto é, que não depende da distância da fonte de luz ao objecto iluminado
- ii. Linear, isto é, proporcional à distância da fonte de luz ao objecto iluminado
- iii. Quadrático, isto é, proporcional ao quadrado da distância da fonte de luz ao objecto iluminado
- iv. Todas as anteriores

2016-2017 (Recurso)

[2.5] Uma forma de determinar o vector normal a um polígono planar consiste em

- i. Calcular o produto vectorial dos vectores definidos por duas arestas do polígono e dividir as componentes do vector resultante pelo comprimento deste último
- ii. Projectar o polígono nos planos OYZ, OZX e OXY e calcular as áreas dos polígonos resultantes; as componentes da normal serão proporcionais a estes valores
- iii.** As respostas i. e ii.
- iv. Nenhuma das anteriores

2016-2017 (Normal)

[2.5] O modelo de iluminação do OpenGL

- i. Requer a especificação de normais
- ii. Constitui um exemplo de um modelo de iluminação local
- iii. Contempla apenas os caminhos para a luz do tipo fonte luminosa → superfície → observador
- iv.** Todas as anteriores

[2.5] Para iluminar uma cena com uma fonte de luz direccional, deverá

- i.** Activar o modelo de iluminação do OpenGL
- ii. Especificar para a posição da fonte de luz um conjunto de coordenadas homogéneas tal que $w = 1$
- iii. Especificar para o ângulo de *cutoff* o valor de 180°
- iv. Todas as anteriores

2015-2016 (Recurso)

[2.5] Em OpenGL, a alteração da propriedade GL_SHININESS de um material afecta a forma como é reflectida

- i. A componente ambiente de iluminação
- ii.** A componente difusa de iluminação
- iii. A componente especular de iluminação
- iv. Todas as anteriores

[2.5] Um projector constitui um exemplo de uma fonte de luz

- i.** Posicional
- ii. Direccional
- iii. Omnidireccional
- iv. Nenhuma das anteriores

Texturas

Propriedades Mapeáveis

- Cor (coeficientes de reflexão difusa)
- Coeficientes de reflexão especular e difusa
 - “Environment Mapping”
- Perturbação do vector normal
 - “Bump Mapping”
- Perturbação da superfície na direção da normal
 - “Displacement Mapping”
- Transparência / opacidade

Função de Mapeamento de uma Textura

- Devolve o ponto do objecto correspondente a cada ponto do espaço de textura
- Corresponde à forma com que a textura é usada para “embrulhar” (wrap) o objecto
- Se a superfície do objecto puder ser descrita na forma paramétrica, esta pode servir como base para a função de mapeamento

$$(x, y, z) = F(s, t)$$

Mipmapping (Mapeamento de Texturas)

- É suportada pela generalidade das API gráficas
- Permite que texturas de diferentes níveis de resolução sejam aplicadas de forma adaptativa
- Reduz os efeitos de discretização que decorrem da interpolação

Correcção Perspectiva

Permite corrigir o efeito de deformação que decorre da utilização de técnicas simples de interpolação linear no mapeamento de texturas em polígonos.

Perguntas

2018-2019 (Recurso)

[3.3] A técnica de *mipmapping* de mapeamento de texturas

- i. É suportada pela generalidade das API gráficas
- ii. Permite que texturas de diferentes níveis de resolução sejam aplicadas de forma adaptativa
- iii. Reduz os efeitos de discretização que decorrem da interpolação
- iv. Todas as anteriores

2018-2019 (Normal)

[3.3] Uma função de mapeamento de textura

- i. Devolve, para cada ponto do espaço de textura, o ponto correspondente da superfície do objecto
- ii. Corresponde à forma com que a textura é usada para “embrulhar” (*wrap*) o objecto
- iii. Pode basear-se na descrição paramétrica da superfície do objecto ao qual a textura está a ser aplicada
- iv. Todas as anteriores

2017-2018 (Recurso)

[2.5] No mapeamento de texturas em OpenGL, o processo de filtragem designado por GL_NEAREST_MIPMAP_LINEAR

- i. Escolhe o texel que mais se aproxima do centro do pixel no mipmap que melhor se adequa ao contexto de minificação existente
- ii. Calcula uma média pesada da matriz de 2 x 2 texels que mais se aproxima do centro do pixel no mipmap que melhor se adequa ao contexto de minificação existente
- iii. Escolhe o texel que mais se aproxima do centro do pixel em cada um dos dois mipmaps que melhor se adequam ao contexto de minificação existente; em seguida, efectua uma interpolação linear destes dois valores
- iv. Calcula uma média pesada da matriz de 2 x 2 texels que mais se aproxima do centro do pixel em cada um dos dois *mipmaps* que melhor se adequam ao contexto de minificação existente; em seguida efectua uma interpolação linear destes dois valores

2017-2018 (Normal)

[2.5] De que forma ou formas permite o mecanismo de mapeamento de texturas do OpenGL aplicar uma textura à superfície de um objecto?

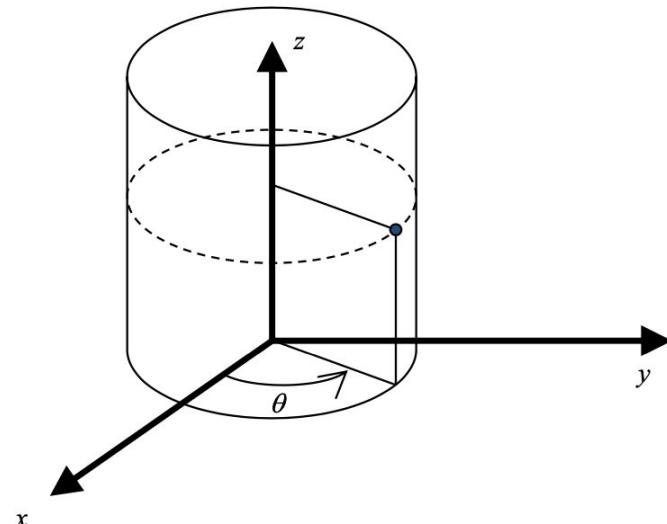
- i. Misturando a cor da superfície com uma cor predefinida
- ii. Modulando a cor da superfície com a dos téxeis
- iii. Substituindo a cor da superfície pela dos téxeis
- iv. Todas as anteriores

[2.5] A correção perspectiva permite

- i. Corrigir o efeito de discretização (*aliasing*) que decorre da utilização de *frame buffers* de baixa resolução
- ii. Corrigir o efeito de diminuição da dimensão aparente de um objecto quando a distância do mesmo à câmara aumenta
- iii. Corrigir o efeito de deformação que decorre da utilização de técnicas simples de interpolação linear no mapeamento de texturas em polígonos
- iv. Nenhuma das anteriores

2016-2017 (Recurso)

[2.5] A função de mapeamento de texturas que a seguir se discrimina baseia-se numa parametrização



$$\begin{aligned}x &= \cos\theta & \theta &= 2\pi \cdot s \\y &= \sin\theta & z &= t \\z &= z\end{aligned}$$

- i. Esférica
- ii. Cilíndrica
- iii. Cúbica
- iv. Nenhuma das anteriores

[2.5] O mecanismo de mapeamento de texturas do OpenGL permite

- i. A geração automática de coordenadas de textura
- ii. Diversos modos de filtragem
- iii. Que texturas de diferentes níveis de resolução sejam aplicadas de forma adaptativa
- iv. Todas as anteriores

2016-2017 (Normal)

[2.5] No mapeamento de texturas em OpenGL, o processo de filtragem designado por `GL_NEAREST_MIPMAP_NEAREST`

- i. Escolhe o *texel* que mais se aproxima do centro do pixel no *mipmap* que melhor se adequa ao contexto de minificação existente
- ii. Calcula uma média pesada da matriz de 2 x 2 *texels* que mais se aproxima do centro do pixel no *mipmap* que melhor se adequa ao contexto de minificação existente
- iii. Escolhe o *texel* que mais se aproxima do centro do pixel em cada um dos dois *mipmaps* que melhor se adequam ao contexto de minificação existente; em seguida, efectua uma interpolação linear destes dois valores
- iv. Calcula uma média pesada da matriz de 2 x 2 *texels* que mais se aproxima do centro do pixel em cada um dos dois *mipmaps* que melhor se adequam ao contexto de minificação existente; em seguida efectua uma interpolação linear destes dois valores

2015-2016 (Recurso)

[2.5] A correcção perspectiva permite

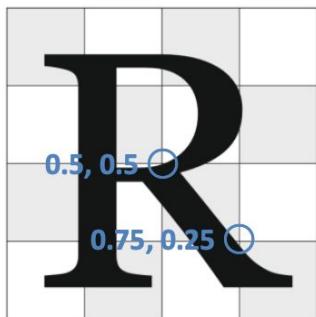
- i. Corrigir o efeito de discretização (*aliasing*) que decorre da utilização de *frame buffers* de baixa resolução
- ii. Corrigir o efeito de diminuição da dimensão aparente de um objecto quando a distância do mesmo à câmara aumenta
- iii. Corrigir o efeito de deformação que decorre da utilização de técnicas simples de interpolação linear no mapeamento de texturas em polígonos
- iv. Nenhuma das anteriores

Mapeamento de Texturas

2018-2019 (Recurso)

[4.0] Pretende-se mapear a textura representada na Figura 1 num rectângulo, de modo a que este fique com o aspecto ilustrado na Figura 2. Indique as coordenadas (s, t) de textura correspondentes a cada um dos vértices do polígono.

Figura 1



v0: 0.75, 0.5

v1: 0.75, 0.25

v2: 0.5, 0.25

v3: 0.5, 0.5

Figura 2

0.5, 0.5

v3

0.75, 0.5

v0



0.5, 0.25

0.75, 0.25

2018-2019 (Normal)

[4.0] Pretende-se mapear a textura representada na Figura 1 num rectângulo, de modo a que este fique com o aspecto ilustrado na Figura 2. Indique as coordenadas (s, t) de textura correspondentes a cada um dos vértices do polígono.

Figura 1

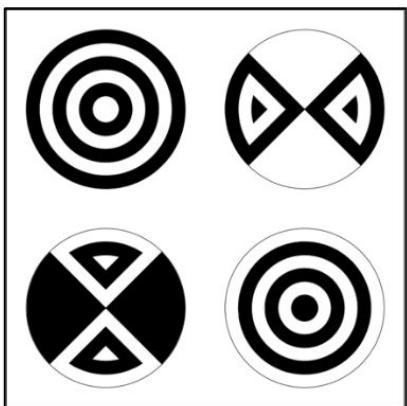
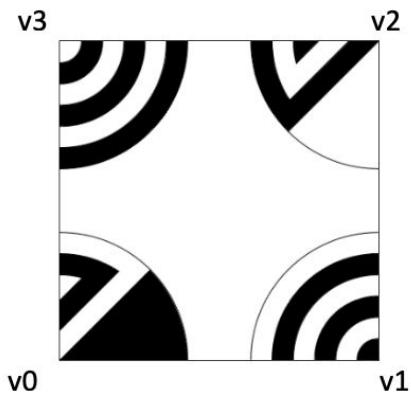


Figura 2



v0: **0.25, 0.25**

v1: **0.75, 0.25**

v2: **0.75, 0.75**

v3: **0.25, 0.75**

2017-2018 (Recurso)

[3.0] Aplique a textura apresentada na Figura 1 a um quadrado, de modo a ficar com o aspecto apresentado na Figura 2.

Figura 1

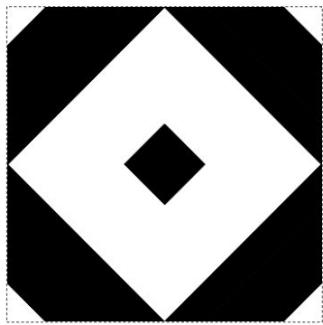
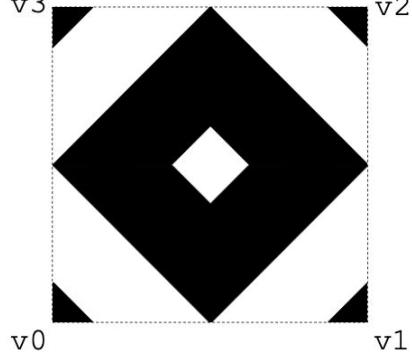


Figura 2



```
glTexCoord2f(0.5, 0.5);
glVertex3fv(v0);
glTexCoord2f(1.5, 0.5);
glVertex3fv(v1);
glTexCoord2f(1.5, 1.5);
glVertex3fv(v2);
glTexCoord2f(0.5, 1.5);
glVertex3fv(v3);
```

```
// ou qualquer outro mapeamento em que
// as coordenadas terminem em 0.5 e
// tenham dimensão 1.0 por 1.0
```

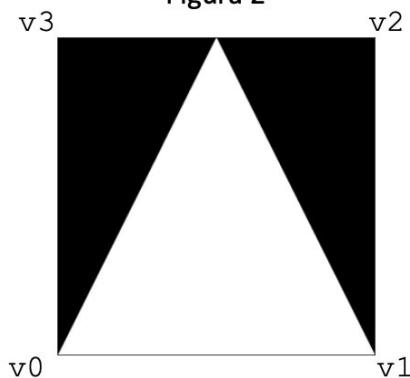
2017-2018 (Normal)

[3.0] Aplique a textura apresentada na Figura 1 a um rectângulo, de modo a ficar com o aspecto apresentado na Figura 2.

Figura 1



Figura 2



```
glTexCoord2f(1.0, 0.0);  
glVertex3fv(v0);  
glTexCoord2f(1.0, 1.0);  
glVertex3fv(v1);  
glTexCoord2f(0.5, 1.0);  
glVertex3fv(v2);  
glTexCoord2f(0.5, 0.0);  
glVertex3fv(v3);
```

ou

```
glTexCoord2f(1.0, 1.0);  
glVertex3fv(v0);  
glTexCoord2f(1.0, 0.0);  
glVertex3fv(v1);  
glTexCoord2f(0.5, 0.0);  
glVertex3fv(v1);  
glTexCoord2f(0.5, 1.0);  
glVertex3fv(v1);
```

2016-2017 (Recurso)

[3.0] Aplique a textura apresentada na Figura 1 a um rectângulo, de modo a ficar com o aspecto apresentado na Figura 2.

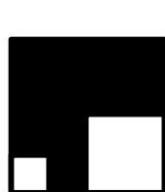


Figura 1



Figura 2

```
glTexCoord2f(3.0, 0.0);
glVertex3fv(v0);
glTexCoord2f(0.0, 0.0);
glVertex3fv(v1);
glTexCoord2f(0.0, 1.0);
glVertex3fv(v2);
glTexCoord2f(3.0, 1.0);
glVertex3fv(v3);
```

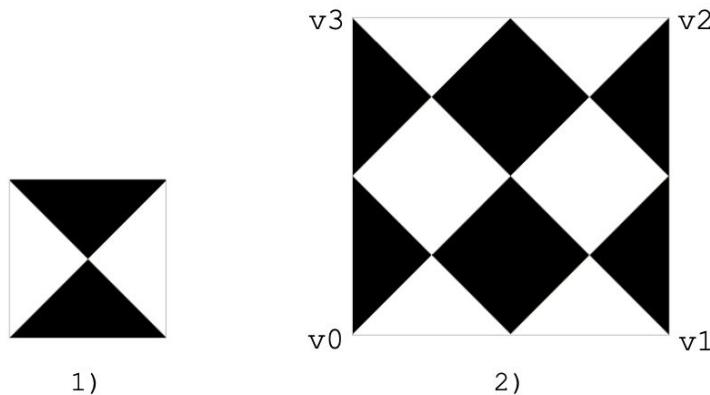
ou qualquer outra solução idêntica, desfasada em valores inteiros em s ou t

[2.0] Indique os parâmetros de configuração de texturas que é necessário realizar para o código acima funcionar correctamente.

Activar o modo de repetição do padrão de textura no eixo s (ou seja, configurar `GL_TEXTURE_WRAP_S` como `GL_REPEAT`).

2016-2017 (Normal)

[3.0] Aplique a textura apresentada na figura 1 a um quadrado, de modo a ficar com o aspecto apresentado na figura 2.



```
...
glTexCoord2f(0.5, 0.5); // como alternativa
glVertex3fv(v0);      // usar outros valores
glTexCoord2f(2.5, 0.5); // desfasados de 2 unidades
glVertex3fv(v1);      // e com parte decimal .5
glTexCoord2f(2.5, 2.5); // ou então trocar s com t
glVertex3fv(v2);      // e usar valores
glTexCoord2f(0.5, 2.5); // com parte decimal .0
glVertex3fv(v3);      // (isto é, rodar a imagem)
...

```

[2.0] Indique os parâmetros de configuração de texturas que é necessário realizar para o código acima funcionar correctamente.

Activar o modo de repetição do padrão de textura nos eixos s e t (ou seja, configurar `GL_TEXTURE_WRAP_S` e `GL_TEXTURE_WRAP_T` como `GL_REPEAT`).

Polígonos e Primitivas do OpenGL

2017-2018 (Recurso)

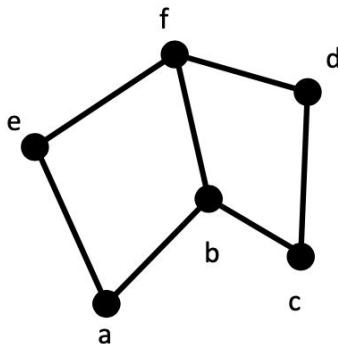
[3.0] Pretende-se modelar um terreno a partir de uma matriz que contém o valor da cota em vários pontos do terreno. Que tipo de polígonos e qual das primitivas de desenho do OpenGL serão mais indicados para fazer a modelação?

Tipo de polígonos: triângulo _____

Primitiva de desenho: `GL_TRIANGLE_STRIP` _____

2015-2016 (Recurso)

[2.0] Suponha que pretende modelar o objecto ilustrado na figura. Comente os dois extractos de código apresentados.



```
glBegin(GL_QUADS);  
    glVertex3fv(a);  
    glVertex3fv(b);  
    glVertex3fv(f);  
    glVertex3fv(e);  
    glVertex3fv(b);  
    glVertex3fv(c);  
    glVertex3fv(d);  
    glVertex3fv(f);  
    glEnd();  
  
glBegin(GL_LINE_STRIP);  
    glVertex3fv(f);  
    glVertex3fv(d);  
    glVertex3fv(c);  
    glVertex3fv(b);  
    glVertex3fv(a);  
    glVertex3fv(e);  
    glVertex3fv(f);  
    glVertex3fv(b);  
    glEnd();
```

O extracto de código da esquerda desenha dois polígonos quadriláteros; o da direita desenha uma linha contínua.

Esfera Iluminada

2018-2019 (Recurso)

[3.0] Considere uma esfera constituída por um material amarelo claro (1.0, 1.0, 0.5) iluminada por uma única fonte de luz verde clara (0.0, 1.0, 0.5). Quais as componentes primárias (R, G, B) da cor resultante? Indique os cálculos efectuados.

$$R = 1.0 * 0.0 = 0.0$$

$$G = 1.0 * 1.0 = 1.0$$

$$B = 0.5 * 0.5 = 0.25$$

2018-2019 (Normal)

[3.0] Considere uma esfera constituída por um material cinzento escuro (0.1, 0.1, 0.1) iluminada por uma única fonte de luz verde clara (0.75, 1, 0.75). Quais as componentes primárias (R, G, B) da cor resultante? Indique os cálculos realizados.

$$R = 0.75 * 0.1 = 0.075$$

$$G = 1.0 * 0.1 = 0.1$$

$$B = 0.75 * 0.1 = 0.075$$

2017-2018 (Normal)

[3.0] Considere um cubo com material cor-de-laranja (1.0, 0.5, 0.0) iluminado por uma única fonte de luz azul-clara (0.0, 0.5, 1.0). Quais as componentes primárias (R, G, B) da cor resultante? Indique os cálculos realizados.

$$R = 1.0 * 0.0 = 0.0$$

$$G = 0.5 * 0.5 = 0.25$$

$$B = 0.0 * 1.0 = 0.0$$

2016-2017 (Recurso)

[2.0] Considere um cubo com material laranja (1.0, 0.5, 0.25) iluminado por uma única fonte de luz verde claro (0.5, 1.0, 0.0). Qual as componentes primárias da cor resultante?

$$R = 0.5 \quad G = 0.5 \quad B = 0.0$$

2016-2017 (Normal)

[2.0] Considere um cubo com material azul (0.25, 0.25, 1.0) iluminado por uma única fonte de luz amarela (1.0, 1.0, 0.0). Qual será a cor (em termos das suas componentes primárias RGB) resultante?

$$R = 0.25 \quad G = 0.25 \quad B = 0.0$$

[2.0] Um objecto é iluminado por duas fontes de luz: uma emite componente especular vermelha; a outra emite componente especular verde. Indique como terá de configurar o material para que o reflexo especular fique com a cor da luz emitida.

Para a reflexão especular apresentar a cor da fonte de luz, a componente de reflexão especular do material tem que ser branca (independentemente da cor do material).

Normal Unitária

2018-2019 (Recurso)

[4.0] Determine as componentes da normal unitária do quadrilátero apresentado na Figura 3. A face visível do polígono é a face da frente, assinalada pela semiesfera branca.

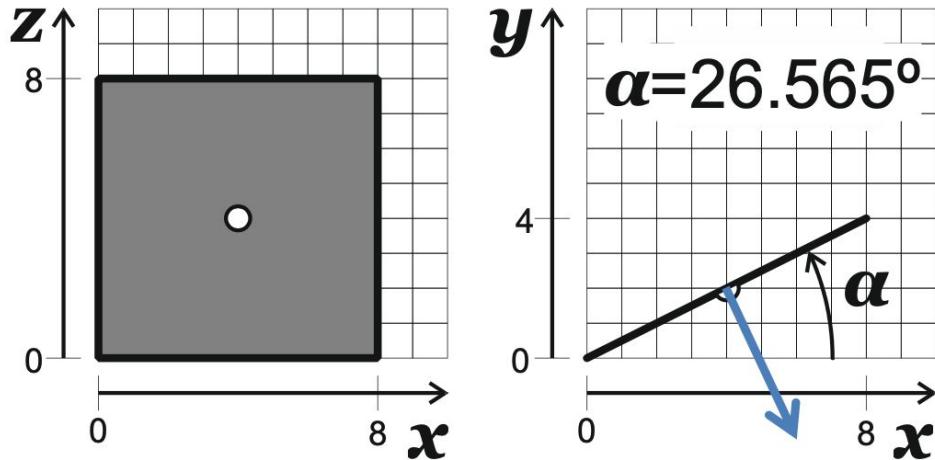


Figura 3

Normal não unitária: $(4, -8, 0)$ ou $(1, -2, 0)$

Normal unitária: $(4 / \sqrt{4^2 + (-8)^2 + 0^2}, -8 / \sqrt{4^2 + (-8)^2 + 0^2}, 0)$ ou
 $(1 / \sqrt{1^2 + (-2)^2 + 0^2}, -2 / \sqrt{1^2 + (-2)^2 + 0^2}, 0)$ ou
 $(\sin(26.565^\circ), -\cos(26.565^\circ), 0)$ ou
 $(0.4472, -0.8944, 0)$

2018-2019 (Normal)

- c. [4.0] Determine as componentes da normal unitária do quadrilátero apresentado na Figura 3. A face visível do polígono é a face da frente, a qual faz um ângulo de 45° com o eixo dos X.

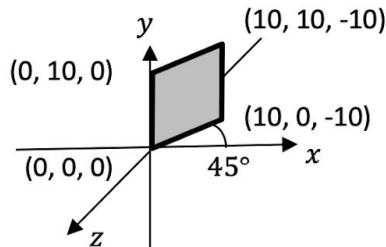


Figura 3

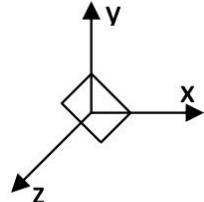
Normal: $\cos(45^\circ), 0.0, \sin(45^\circ)$ ou $\sin(45^\circ), 0.0, \cos(45^\circ)$ ou $\sqrt{2.0} / 2.0, 0.0, \sqrt{2.0} / 2.0$

2017-2018 (Recurso)

- [3.0] Considerando as definições por omissão do OpenGL, pretende-se definir a normal para o quadrilátero desenhado pelo seguinte extracto de código. Qual a normal unitária perpendicular ao quadrilátero?

Nota: Pode usar funções trigonométricas com ângulos expressos em graus ou em radianos.

```
glBegin(GL_QUADS);
    glNormal3f(-sin(45.0), -sin(45.0), 0.0);
    // ou (-cos(45.0), -cos(45.0), 0.0)
    // ou (-sqrt(2.0) / 2.0, -sqrt(2.0) / 2.0, 0.0)
    glVertex3f(1, 0, 0);
    glVertex3f(1, 0, 1);
    glVertex3f(0, 1, 1);
    glVertex3f(0, 1, 0);
glEnd();
```



2017-2018 (Normal)

[3.0] Considerando as definições por omissão do OpenGL, pretende-se definir a normal para o quadrilátero desenhado pelo seguinte extracto de código. Qual a normal unitária perpendicular ao quadrilátero?

```
glBegin(GL_QUADS);
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(0, 1, 1);
    glVertex3f(1, 1, 1);
    glVertex3f(1, 1, 0);
    glVertex3f(0, 1, 0);
glEnd();
```

Observações:

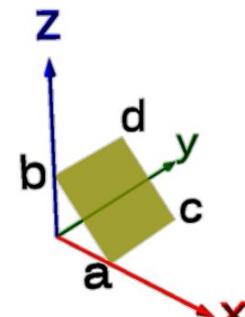
- Cada coordenada correcta vale 1.0 valores
- Se for indicado $\sin()$ / $\cos()$ correcto, mas não o resultado, então vale 0.5 valores

2016-2017 (Normal)

[3.0] Complete o código para modelar o quadrado apresentado na figura e defina a **normal unitária** correctamente, assumindo que a orientação por omissão dos polígonos no OpenGL não foi alterada.

Os arrays a , b , c e d já estão definidos; os segmentos \overline{ab} e \overline{cd} fazem 45° com o plano oxy ; e os segmentos \overline{ac} e \overline{bd} são paralelos ao eixo dos yy .

```
glBegin(GL_QUADS); // ou GL_POLYGON
    glNormal3f(cos(M_PI / 4.0), 0.0, cos(M_PI / 4.0));
    // ou sqrt(2.0) / 2.0
    glVertex3fv(a); // ou c, d, b, a
    glVertex3fv(c); // ou d, b, a, c
    glVertex3fv(d); // ou b, a, c, d
    glVertex3fv(b);
glEnd(); // se o quadrado fosse desenhado no sentido oposto (retrógrado ou horário), o sentido da normal também seria o oposto
```



Posicionamento da Câmara

$$\text{seno } \alpha = \frac{\text{cateto oposto}}{\text{hipotenusa}}$$

$$\cos \text{seno } \alpha = \frac{\text{cateto adjacente}}{\text{hipotenusa}}$$

$$\tan \text{gente } \alpha = \frac{\text{cateto oposto}}{\text{cateto adjacente}}$$

2018-2019 (Recurso)

[4.0] Pretende-se simular a visão de um utilizador numa bicicleta que está a olhar sempre para o guiador da mesma. A posição do utilizador é dada por `mod.x`, `mod.y` e `mod.z`, a direcção para a qual a bicicleta está orientada é dada por `mod.dir`, a altura dos olhos do utilizador relativamente à sua posição é dada por `A_OLHOS`. Relativamente à posição do utilizador, a altura do guiador é dada por `A_GUIA` e encontra-se à distância de `D_GUIA`.

Complete a informação seguinte de modo a obter a câmara pretendida, considerando como eixo vertical o eixo dos Z (positivo para cima).

Eye: `mod.x, mod.y, mod.z + A_OLHOS`

Center: `mod.x + D_GUIA * cos(mod.dir), mod.y + D_GUIA * sin(mod.dir), mod.z + A_GUIA`

Up: `0, 0, 1`

2018-2019 (Normal)

[4.0] Pretende-se simular uma câmara montada no guiador de uma bicicleta, orientada para a frente, e que roda com o guiador. A posição do guiador é dada por `modelo.x`, `modelo.y` e `modelo.z`, a direcção para a qual o guiador está orientado é dada por `modelo.dir`, e a altura da câmara em relação ao guiador é dada por `ALTURA_CAMARA`.

Complete a informação seguinte de modo a obter a câmara pretendida, considerando como eixo vertical o eixo dos Z (positivo para cima).

Eye: `modelo.x, modelo.y, modelo.z + ALTURA_CAMARA`

Center: `modelo.x + cos(modelo.dir), modelo.y + sin(modelo.dir), modelo.z + ALTURA_CAMARA`

Up: `0.0, 0.0, 1.0`

2017-2018 (Recurso)

[3.0] Pretende-se simular uma câmara montada num helicóptero a olhar directamente para baixo. O *up* da câmara está alinhado com a direcção em que o helicóptero segue. A posição do helicóptero é dada por *modelo.x*, *modelo.y* e *modelo.z*, e a direcção em que o helicóptero está a seguir é dada por *modelo.dir*. Complete a instrução seguinte de modo a obter o resultado pretendido, considerando como eixo vertical o eixo dos Z (positivo para cima).

```
gluLookAt(modelo.x, modelo.y, modelo.z,  
          modelo.x, modelo.y, modelo.z - 1.0,  
          cos(modelo.dir), sin(modelo.dir), 0.0);
```

2017-2018 (Normal)

[3.0] Pretende-se simular uma câmara em cima de um piloto de um carro de Fórmula 1 a olhar para a frente do carro. A posição do piloto é dada por *modelo.x*, *modelo.y* e *modelo.z*, a direcção que o carro está a seguir é dada por *modelo.dir*, e a altura a que a câmara fica da posição do piloto é dada por *ALTURA_CAMARA*. Complete a instrução seguinte de modo a conseguir a câmara pretendida, considerando como eixo vertical o eixo dos Z (positivo para cima).

Nota: A orientação da câmara é similar à visão normal do piloto durante a condução, olhando para a frente do carro.

```
gluLookAt(modelo.x, modelo.y, modelo.z + ALTURA_CAMARA,  
          modelo.x + cos(modelo.dir),  
          modelo.y + sin(modelo.dir),  
          modelo.z + ALTURA_CAMARA,  
          0.0, 0.0, 1.0);
```

Observações:

- 1.0 valores para cada linha
- **Linha 1:**
 - Falta “modelo...”: 0.25 valores
 - Falta apenas “+ ALTURA_CAMARA” em Z: 0.75 valores
- **Linha 2:**
 - Falta “modelo....” ou sin()/cos(): 0.25 valores
 - Falta apenas “+ ALTURA_CAMARA” em Z: 0.75 valores
 - Usar “x * cos()” em vez de “x + cos()”: 0.75 valores
 - Usar “DIST + cos()” em vez de “DIST * cos()”: 0.75 valores
- **Linha 3:**
 - (1.0, 0.0 ,0.0) ou (0.0, 1.0, 0.0): 0.25 valores
 - Trocar primeira e segunda linhas: 75%
 - Trocar Y e Z: 66%

2016-2017 (Recurso)

[3.0] Observe o seguinte objecto 3D articulado.

As coordenadas do ponto central do corpo são indicadas por:

```
(modelo.corpo.x, modelo.corpo.y,  
modelo.corpo.z);
```

o corpo roda apenas em torno do eixo dos Z:

```
modelo.corpo.dir (em radianos);
```

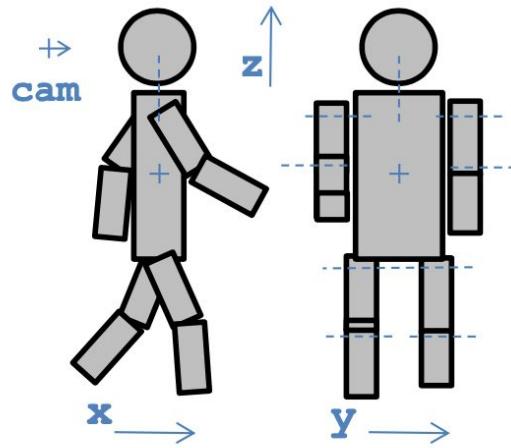
a cabeça roda apenas em torno do eixo dos Z relativamente ao corpo:

```
modelo.corpo.head_dir (em radianos);
```

os braços e as pernas são articulados apenas na direcção do eixo dos Y (linhas tracejadas na imagem).

Imagine que quer colocar uma câmara por trás da cabeça do modelo (ver figura) a olhar em frente. Indique os parâmetros a utilizar em `gluLookAt()`.

```
gluLookAt(  
    modelo.corpo.x - DIST_CAM * cos(modelo.corpo.dir +  
        modelo.cabeça.dir),  
    modelo.corpo.y - DIST_CAM * sin(modelo.corpo.dir +  
        modelo.cabeça.dir),  
    modelo.corpo.z + ALTURA_CAM,  
  
    modelo.corpo.x,  
    modelo.corpo.y,  
    modelo.corpo.z + ALTURA_CAM,  
  
    0,  
    0,  
    1);
```

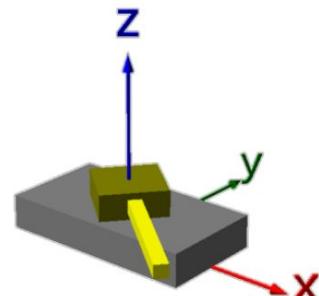


2016-2017 (Normal)

[3.0] Pretende-se implementar a câmara de um tanque. A câmara está colocada por cima da torre e aponta sempre na direcção do canhão (paralela ao plano *oxy*).

A posição do tanque é definida por `modelo.pos.x`, `modelo.pos.y` e `modelo.pos.z`; o ângulo do tanque por `modelo.tanque.direccao`; o ângulo da torre em relação à base por `modelo.tanque.angTorre`; e a altura da câmara em relação à base do tanque pela constante `ALTURA_CAM`.

Nota: Quando os ângulos são de 0° , a base, a torre e o canhão são desenhados alinhados com o eixo dos *xx* (o canhão aponta para a parte positiva do eixo).



```
gluLookAt(modelo.pos.x, modelo.pos.y, modelo.pos.z + ALTURA_CAM,
    modelo.pos.x + cos(modelo.tanque.direccao + modelo.tanque.angTorre),
    modelo.pos.y + sin(modelo.tanque.direccao + modelo.tanque.angTorre),
    modelo.pos.z + ALTURA_CAM,
    0.0, 0.0, 1.0);
```

Random

2016-2017 (Recurso)

[3.0] Considere o modelo de um cilindro centrado no eixo dos Z. Complete o seguinte excerto de código em C, na parte assinalada com `/* CÓDIGO */`, para desenhar as paredes laterais do referido modelo. Defina normais unitárias de forma a que a superfície não exiba arestas significativas. Use apenas as seguintes funções do OpenGL: `glBegin()`, `glVertex3f()`, `glNormal3f()`, `glEnd()`.

```
z1 = 0.0; z2 = 10.0;
alpha = 0.0; alpha_inc = (2.0 * M_PI) / slices;
x1 = radius * cos(alpha); y1 = radius * sin(alpha);

for (i = 0; i < slices; i++) {
    alpha += alpha_inc;
    x2 = radius * cos(alpha); y2 = radius * sin(alpha);

    glBegin(GL_QUADS);
        glNormal3f(x1 / radius, y1 / radius, 0.0);
        glVertex3f(x1, y1, z2);
        glVertex3f(x1, y1, z1);
        glNormal3f(x2 / radius, y2 / radius, 0.0);
        glVertex3f(x2, y2, z1);
        glVertex3f(x2, y2, z2);
    glEnd();
```

[2.0] Imagine dois objectos: uma parede branca mate e uma esfera de plástico brilhante verde. Indique as instruções do OpenGL e os parâmetros respectivos que usaria para configurar as características diferentes entre eles.

```
GLfloat[] mat_black = {0.0f, 0.0f, 0.0f, 1.0f};  
GLfloat[] mat_white = {1.0f, 1.0f, 1.0f, 1.0f};  
GLfloat[] mat_green = {0.0f, 1.0f, 0.0f, 1.0f};  
  
/* branco mate */  
  
glMaterialfv(GL_FRONT, GL_DIFUSE, mat_white);  
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_black);  
glMaterialf(GL_FRONT, GL_SHININESS, 0.0);  
  
/* verde brilhante */  
  
glMaterialfv(GL_FRONT, GL_DIFUSE, mat_green);  
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_green);  
glMaterialf(GL_FRONT, GL_SHININESS, 100.0);
```

[2.0] Suponha que pretende animar um objecto ao longo do eixo dos X. Indique qual o **callback GLUT** que deve utilizar para conseguir realizar a animação e escreva o código correspondente.

callback:

```
	glutTimerFunc()
```

código:

```
void Timer(int value) {  
    modelo.x++;  
    glutPostRedisplay();  
    glutTimerFunc(delay, Timer, 1);  
}
```

2016-2017 (Normal)

[2.0] Suponha que pretende saber qual o objecto que está a ser desenhado por baixo do cursor do rato. Indique qual o **callback GLUT** e a **técnica do OpenGL** que deve utilizar para conseguir identificar o objecto.

O **callback** a utilizar seria o `glutPassiveMotionFunc()` (ou o `glutMouseFunc()`, caso estivesse a pressionar um botão do rato) e a **técnica do OpenGL** seria o **picking** (modo de selecção).

2015-2016 (Recurso)

[2.5] As vulgares impressoras de jacto de tinta constituem exemplos de dispositivos

- i. Vectoriais
- ii. Tensoriais
- iii. Matriciais
- iv. Nenhuma das anteriores