Resoluções Fichas CG

Francisco Ferreira

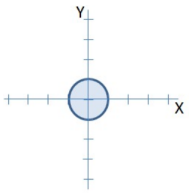
Atualizado 08/06/2024

Atenção! Resolução foi feita por um aluno, **não é oficial** e pode ter erros. A resolução foi confirmada com o Ramires, mas pode ter escapado alguma coisa. Se encontrarem algo mal, avisem.

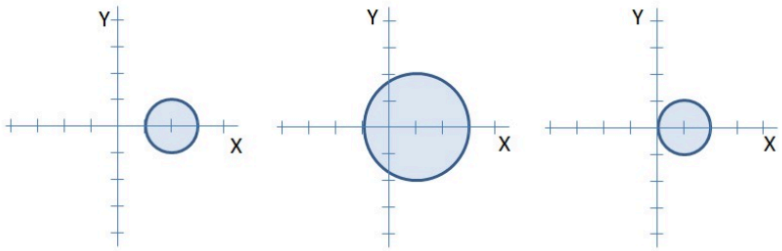
Ficha Transformações Geométricas I

1. Considere uma primitiva gráfica para desenhar uma esfera com centro na origem e raio unitário, e a aplicação da seguinte sequência de transformações geométricas à esfera:

```
glScale(2,2,2);
glTranslate(1,0,0);
glScale(0.5, 0.5, 0.5);
esfera();
```



Qual das seguintes opções corresponde à esfera transformada? Justifique, indicando cada um dos passos intermédios.



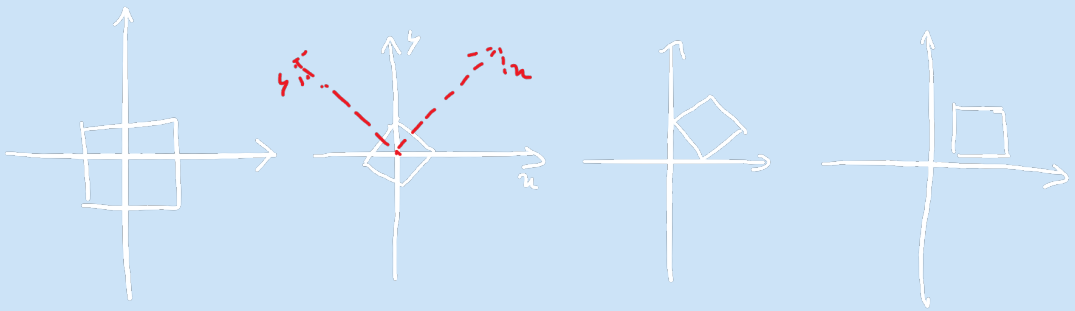
a) b) c)

Resposta: a)

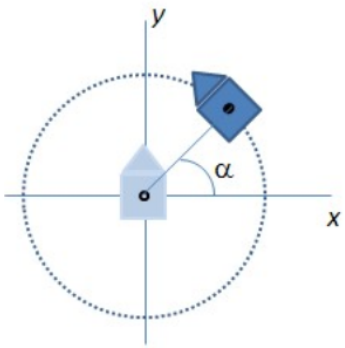
2. Considere uma primitiva gráfica para desenhar um cubo com centro na origem e lado com dimensão de 2 unidades, e a seguinte sequência de transformações geométricas a aplicar ao cubo:

```
glRotate(45, 0.0, 0.0, 1.0);
glTranslate(2.0 , 0.0 , 0.0);
glRotate(-45, 0.0, 0.0, 1.0);
```

Resposta:



3. Considere o objecto "casa" que por omissão é desenhado centrado na origem (casa clara). Considere que se pretende colocar o objecto na circunferência de raio unitário, com centro na origem, como ilustrado na figura (casa escura). Escreva os parâmetros das seguintes alternativas de sequências de transformações geométricas para obter o resultado pretendido:



Resposta: Notar que o eixo Z está a apontar para fora.

a)
```
glTranslate(cos(a), sin(a), 0)
glRotate(a, 0, 0, 1)
desenhaCasa()
```

b)
```
glRotate(a, 0, 0, 1)
glTranslate(1, 0, 0)
desenhaCasa()
```

4. Considere um conjunto matrizes representativas de transformações geométricas 3D básicas, em que translações são representados por Ti, rotações por Ri, e escalas por Si. Para cada afirmação que se segue indique se é verdadeira ou falsa. Apresente um contra-exemplo para as afirmações falsas e um exemplo ilustrativo para as verdadeiras.

   i. $T_1 \times R_1 = R_1 \times T_1$

   Resposta: Falso, ver exercício anterior.

   ii. $T_1 \times S_1 = S_1 \times T_1$

   Resposta: Falso, ver exercício 2.

iii. $T_1 \times T_2 = T_2 \times T_1$

Resposta: Verdadeiro, mover para a esquerda e mover para cima é o mesmo que mover para cima e mover para a esquerda.

iv. Para cada par (T_1, S_1) existe um par (T_2, S_2), tal que $T_1 \times S_1 = S_2 \times T_2$

Resposta: Verdadeiro, a nova matriz de translação será dividida pelo fator de escala.

v. $R_1 \times R_2 = R_2 \times R_1$

Resposta: Em 2D rodar 30º e rodar 70º é o mesmo que rodar 70º e rodar 30º, vão ser equivalentes a rodar 100º. No entanto o mesmo não se pode dizer em 3D (https://math.stackexchange.com/a/2016958).

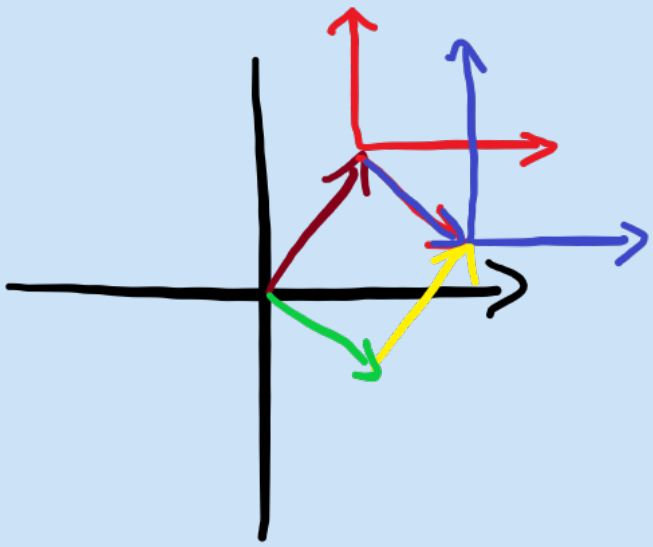
5. Considere a matriz A, obtida após uma sequência de transformações geométricas. Indique a sequência incorrecta para gerar a matriz A a partir da matriz identidade.

Resposta: Alínea c):

$$S(2,2,2) \times T(2,2,2) = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 4 \\ 0 & 2 & 0 & 4 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6. A composição de transformações geométricas é em certos casos comutativa, embora no caso geral não o seja.
   a. Mostre geometricamente que a composição de transformações geométricas compostas exclusivamente por translações é comutativa.

Resposta: A operação de translação consiste em deslocar a origem do referencial por um vetor, e a soma de dois vetores é comutativa.



   b. Mostre algebricamente que a composição de transformações geométricas compostas exclusivamente por escalas é comutativa.

Resposta:

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} ad & 0 & 0 & 0 \\ 0 & be & 0 & 0 \\ 0 & 0 & cf & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} da & 0 & 0 & 0 \\ 0 & eb & 0 & 0 \\ 0 & 0 & fc & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Leftrightarrow ad = da \wedge be = eb \wedge cf = fc$$

$$\Leftrightarrow \text{Verdadeiro}$$

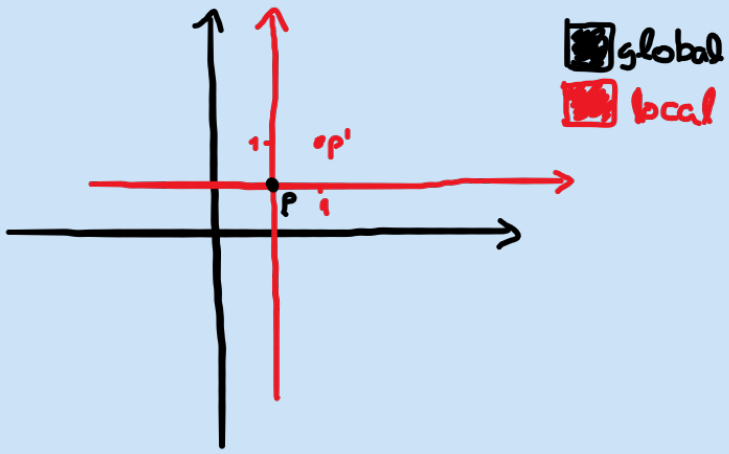
c. **Resposta:** Exercício 1 e 2

7. Considere a seguinte matriz 2D

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

a. Desenhe o sistema de coordenadas global e local (após transformação)
b. Desenhe o ponto $P(1,1)$ e a sua transformação ($p' = M \times p$). Verifique que o ponto transformado tem coordenadas $(1,1)$ no sistema local de coordenadas.

Resposta:



8. Considere a seguinte matriz 2D

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

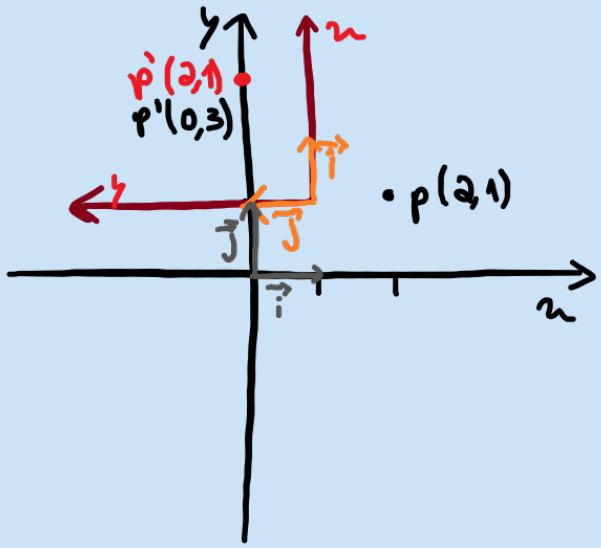
a. Aplique a matriz ao ponto $(2,1)$.

Resposta: $\begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 1 \end{bmatrix}$

b. Desenhe o sistema de coordenadas global e local (após transformação).

c. Desenhe o ponto $(2, 1)$ e a sua transformação. Verifique que o ponto transformado tem coordenadas $(2, 1)$ no sistema local de coordenadas.

Resposta: Notar que, para uma matriz $M = \begin{bmatrix} i_1 & j_1 & o_1 \\ i_2 & j_2 & o_2 \\ 0 & 0 & 1 \end{bmatrix}$, ao ser aplicada, colocará o eixo x na direção $\vec{i}(i_1, i_2)$, eixo y na direção $\vec{j}(j_1, j_2)$ com origem em $\vec{o}(o_1, o_2)$.[1]



9. Construa a matriz de rotação em torno do eixo do Z com um ângulo de 45º.

Resposta:

$$\begin{bmatrix} \cos(45) & -\sin(45) & 0 & 0 \\ \sin(45) & \cos(45) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a. Aplique a matriz ao ponto $(0.707, 0.707, 0)$.

Resposta: Sabendo que $\frac{\sqrt{2}}{2} \approx 0.707$:

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2} \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

b. Sem realizar cálculos, calcule a inversa da matriz construída.

[1] O mesmo pode ser aplicado para matrizes 4x4. Para melhor compreensão, recomendo a série de álgebra linear do *3Blue1Brown*.

Resposta: A inversa da matriz construída será igual à matriz de rotação em torno do eixo do Z mas com ângulo de **−45º**. Logo:

$$M = \begin{bmatrix} \cos(-45) & -\sin(-45) & 0 & 0 \\ \sin(-45) & \cos(-45) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c. Aplique a inversa ao ponto transformado e verifique que o resultado é o ponto original.

Resposta:

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \\ 1 \end{bmatrix}$$

10. Considere o ponto $p(1, 2, 3)$ e o ponto $q(3, 4, 3)$.
    a. Defina uma matriz de escala S tal que $q = Sp$.

Resposta:

$$\begin{aligned} & q = Sp \\ \Leftrightarrow & \begin{bmatrix} 3 \\ 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} \\ \Leftrightarrow & \begin{bmatrix} 3 \\ 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ 2b \\ 3c \\ 1 \end{bmatrix} \\ \Leftrightarrow & a = 3 \wedge 4 = 2b \wedge 3 = 3c \wedge 1 = 1 \\ \Leftrightarrow & a = 3, b = 2, c = 1 \end{aligned}$$

$$S = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

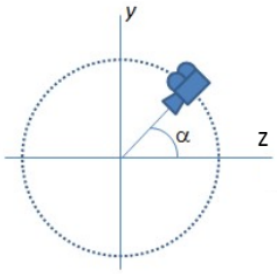
    b. Defina uma matriz de translação T tal que $q = Tp$

Resposta:

$$
\begin{aligned}
& q = Tp \\
\Leftrightarrow & \begin{bmatrix} 3 \\ 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} \\
\Leftrightarrow & \begin{bmatrix} 3 \\ 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} a+1 \\ 2+b \\ 3+c \\ 1 \end{bmatrix} \\
\Leftrightarrow & \, a+1 = 3 \wedge 2+b = 4 \wedge 3+c = 3 \wedge 1 = 1 \\
\Leftrightarrow & \, a = 2 \wedge b = 2 \wedge c = 0 \\
& T = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Ficha Transformações Geométricas II

1. Considere que se pretende colocar uma câmara na circunferência de raio unitário, com centro na origem, como ilustrado na figura.



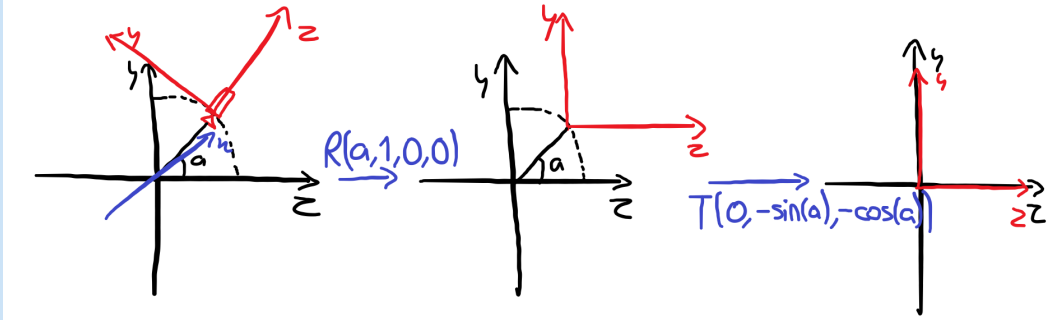
   a. Escreva os parâmetros da função gluLookAt, sabendo que os três primeiros parâmetros representam a posição da câmara, os três seguintes indicam um ponto para onde a câmara está a apontar, e os três últimos parâmetros definem o vector "up";

   Resposta: `gluLookAt(0, sin(a), cos(a), 0, 0, 0, 0, 1, 0);`

   b. Recorrendo somente a rotações e translações, escreva a sequência de transformações geométricas apropriadas para obter exactamente a mesma definição da câmara (pode utilizar funções como sin e cos).

   Resposta:

```
glRotate(a, 1, 0, 0)
glTranslate(0, -sin(a), -cos(a))
```

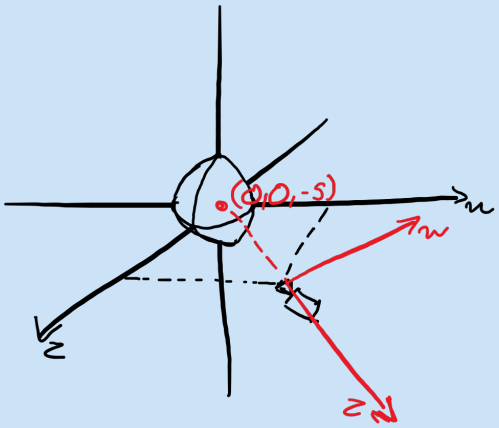


2. Considere o seguinte excerto de código:

```
gluLookAt(5, 0, 5, 0, 0, 0, 0, 1, 0);
drawEsfera(); // desenha esfera de raio 1 centrada na origem
```

   De acordo com o seguinte código, assinale as afirmações verdadeiras:

Resposta: Primeira análise:



a. No espaço global a esfera é desenhada com o centro em $(0, 0, 0)$.

Resposta: Verdadeiro

b. No espaço câmara a esfera é desenhada com o centro em $(-5, 0, -5)$.

Resposta: Falso, terá centro em $(0, 0, -5)$

c. No espaço câmara a esfera é desenhada com o centro no eixo Z.

Resposta: Verdadeiro

3. Considere que uma câmara está definida com a seguinte instrução:

```
gluLookAt(p1, p2, p3, l1, l2, l3, u1, u2, u3);
```

a. Apresente o processo de cálculo para mover a câmara para a esquerda uma unidade, mantendo a direcção do olhar, recorrendo somente à informação fornecida na instrução.

Resposta:

$$\vec{p} = (p_1, p_2, p_3)$$
$$\vec{l} = (l_1, l_2, l_3)$$
$$\vec{u} = (u_1, u_2, u_3)$$
$$\vec{d} = \frac{\vec{l} - \vec{p}}{\left\|\vec{l} - \vec{p}\right\|}$$
$$\vec{e} = \frac{\vec{u} \times \vec{d}}{\left\|\vec{u} \times \vec{d}\right\|}$$
$$\overrightarrow{p_{\text{novo}}} = \vec{p} + \vec{e}$$
$$\overrightarrow{l_{\text{novo}}} = \vec{l} + \vec{e}$$

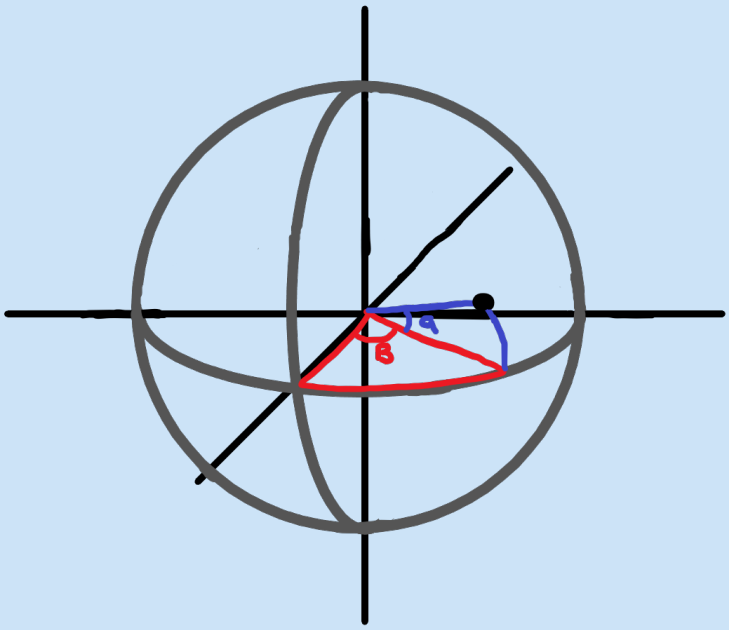
b. Apresente o processo de cálculo para mover a câmara para cima uma unidade, mantendo a direcção do olhar, recorrendo somente à informação fornecida na instrução.

Resposta: Pegando nos cálculos da resolução do exercício anterior:

$$\overrightarrow{up} = \vec{d} \times \vec{e}$$
$$\overrightarrow{p_{\text{novo}}} = \vec{p} + \overrightarrow{up}$$
$$\overrightarrow{l_{\text{novo}}} = \vec{l} + \overrightarrow{up}$$

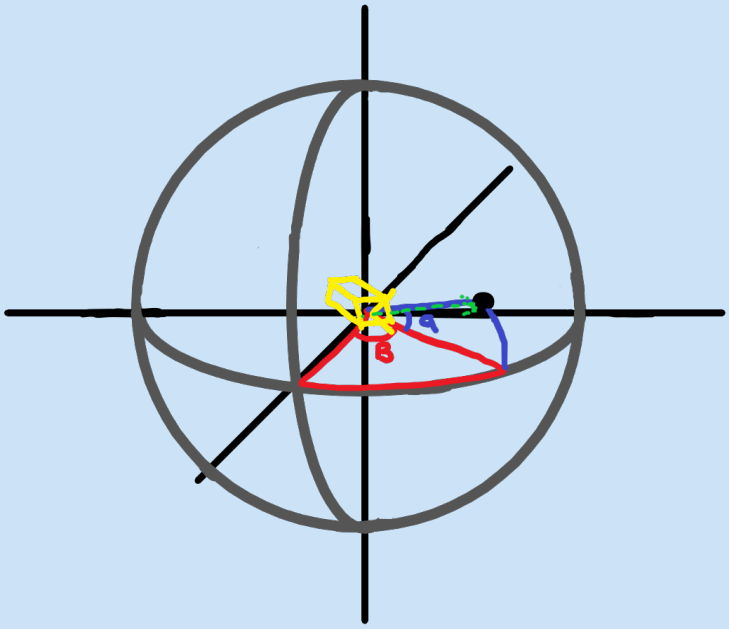
4. Considere que se pretende adicionar uma câmara no modo explorador numa aplicação em OpenGL. Apresente os cálculos, considerando coordenadas esféricas, para determinar as a primeira componente da função gluLookAt (a posição da câmara) assumindo que a câmara está sempre a olhar para a origem. Considere um ângulo vertical alpha, e um ângulo horizontal beta. Ilustre graficamente os cálculos efectuados.

Resposta: $(\sin\beta \times \cos\alpha, \sin\alpha, \cos\beta \times \cos\alpha)$



5. Considere que se pretende adicionar uma câmara no modo FPS numa aplicação em OpenGL. Apresente os cálculos, considerando coordenadas esféricas, para determinar a segunda componentes da função gluLookAt (o ponto para onde está a olhar) considerando um ângulo vertical alpha e um ângulo horizontal beta. Assuma que a câmara se encontra posicionada no ponto P(x,y,z). Ilustre graficamente os vectores e pontos considerados.

Resposta: $(x + \sin\beta \times \cos\alpha, y + \sin\alpha, z + \cos\beta \times \cos\alpha)$



6. Considere o seguinte excerto de código:

```
translate(0, 0, -3);
drawEsfera();
translate(0, 0, 3);
gluLookAt(px, py, pz, 0, 0, -1, 0, 1, 0);
translate(0, 0, -10);
drawEsfera();
```

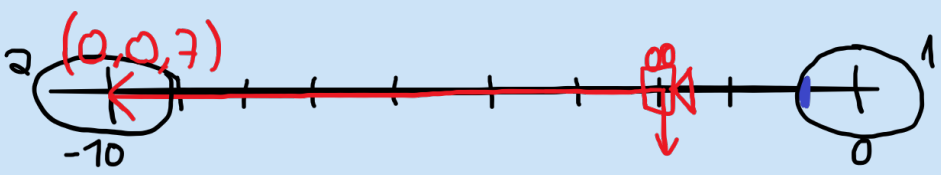
De acordo com o código acima comente as seguintes afirmações (recorra a diagramas para suportar a resposta):

a. A posição do centro da primeira esfera no espaço câmara é $(0, 0, -3)$.

Resposta: Verdadeiro, a primeira esfera já está desenhada após a transformação da câmara, logo ela vai ter sempre essa posição independentemente de onde a câmara ocupa no espaço global.

b. Caso $(p_x, p_y, p_z) = (0, 0, -3)$, a posição do centro da segunda esfera no espaço câmara é $(0, 0, -7)$.

Resposta: Falso, a câmara, como o *looking at* está a $(0, 0, -1)$, está a apontar para o lado positivo, logo, a segunda esfera terá coordenadas $(0, 0, 7)$ e não $(0, 0, -7)$.

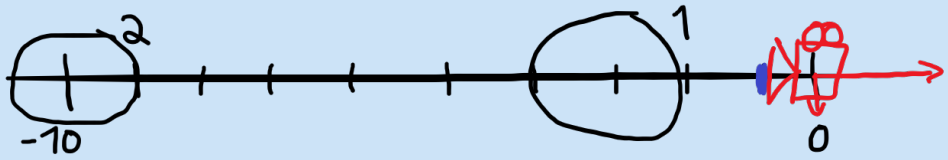


c. Caso $(p_x, p_y, p_z) = (0, 0, -5)$, a segunda esfera não é visível.

Resposta: Verdadeiro, com a mesma lógica da pergunta anterior, a câmara está a apontar para o lado positivo, portanto a segunda esfera está atrás da câmara, não sendo visível.

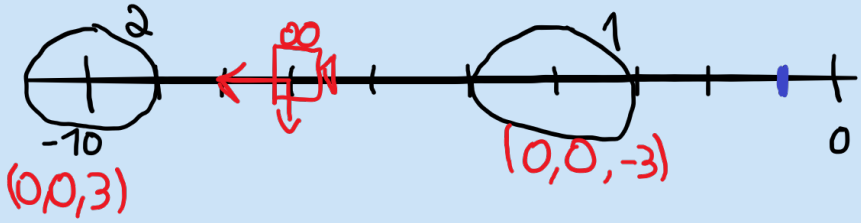
d. Caso $(p_x, p_y, p_z) = (0, 0, 0)$, a segunda esfera não é visível.

Resposta: Falso, com esta posição a câmara já vai olhar para o lado negativo, mostrando a segunda esfera (apesar que esta esteja tapada pela primeira esfera).

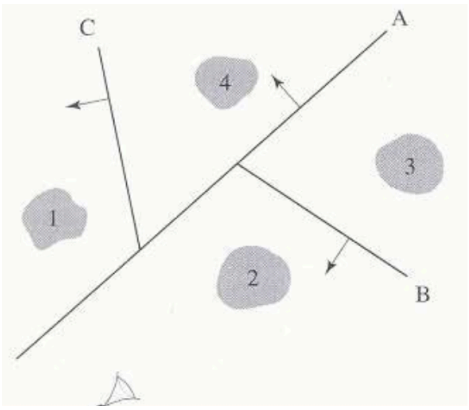


e. Caso $(p_x, p_y, p_z) = (0, 0, -7)$, as esferas ocupam a mesma posição no espaço câmara.

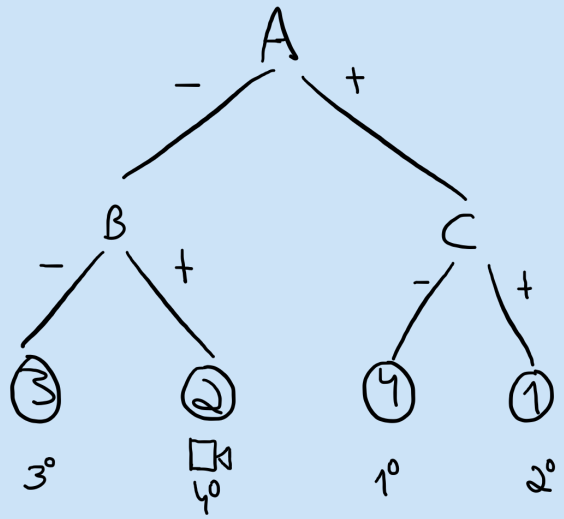
Resposta: Falso, a câmara está a apontar para o lado positivo, logo a segunda esfera terá centro na posição $(0, 0, 3)$ no espaço câmara e a primeira esfera está sempre na posição $(0, 0, -3)$ no espaço câmara.



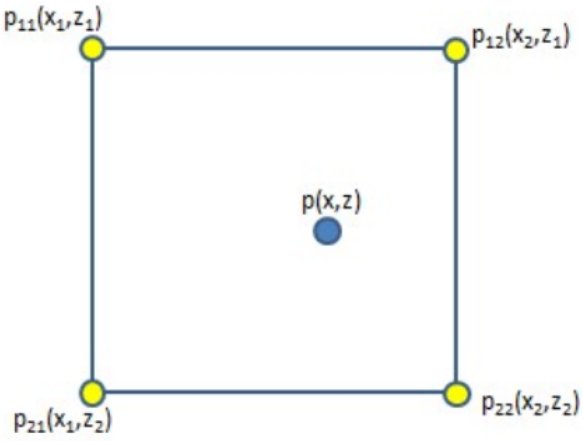
7. Considere a seguinte divisão do espaço utilizando uma BSP. Construa a árvore correspondente e, dada a posição da câmara indicada na figura, apresente a ordem de desenho dos objectos de forma a garantir a ordem de escrita dos pixels.



Resposta: $4 \rightarrow 1 \rightarrow 3 \rightarrow 2$[2]



8. Considere que se pretende usar uma grelha para representar um terreno, à semelhança do que foi pedido no trabalho prático. As coordenadas dos pontos da grelha são números inteiros e a dimensão dos lados de cada quadrícula da grelha é uma unidade. Para obter a altura dos pontos da grelha é disponibilizada a função $h(p_{ij})$, sendo p_{ij} um ponto da grelha. Com base na figura, indique como proceder matematicamente para calcular a altura do ponto p.



[2]Uma explicação detalhada deste exercício pode ser encontrada no vídeo das aulas teóricas dos *Depth Buffers*.

Resposta: Sendo $\lfloor a \rfloor$ a operação para arredondar a para baixo:

$$p = (x, z)$$

$$\begin{aligned} x_1 &= \lfloor x \rfloor & z_1 &= \lfloor z \rfloor \\ x_2 &= x_1 + 1 & z_2 &= z_1 + 1 \end{aligned}$$

$$f_z = z - z_1$$
$$h_1 = h(x_1, z_1) \times (1 - f_z) + h(x_1, z_2) \times f_z$$
$$h_2 = h(x_2, z_1) \times (1 - f_z) + h(x_2, z_2) \times f_z$$
$$f_x = x - x_1$$
$$h = h_1 \times (1 - f_x) + h_2 \times f_x$$

A lógica, sendo *lerp* uma interpolação linear entre dois pontos, é:

$$lerp(lerp(p_{11}, p_{12}), lerp(p_{21}, p_{22}))$$

9. Considere a biblioteca gUM que contem primitivas gráficas para cadeiras e mesas como se ilustra nas figuras. Escreva uma função em C que permita construir em OpenGL um modelo semelhante ao apresentado na figura com a cena das mesas e cadeiras. Como referência, em termos de medidas, considere que a mesa tem um raio de 1 unidade, e que as cadeiras têm os lados do tampo com 0,4 unidades.

Resposta: `POR FAZER`

Ficha de Curvas e Superfícies

1. Considere que se pretende unir duas curvas cúbicas de Bezier. Quais são as restrições que devem ser impostas aos pontos de controlo de cada curva para:
   a. ter continuidade na linha;

      Resposta: O começo da segunda curva tem de estar no fim da primeira curva.

   b. ter as tangentes à curva no ponto final da primeira curva na mesma direcção que no ponto inicial da primeira curva;

      Resposta: O segundo ponto da segunda curva tem de estar na reta formada pelos pontos P_2 e P_3 (penúltimo e último ponto de controlo da primeira curva, respetivamente).

   c. ter continuidade da derivada na linha.

      Resposta: $Q_1 = Q_0 + P_3 - P_2$, sendo P_i o ponto com índice i da primeira curva e Q_j o ponto com índice j da segunda curva.[3]

2. Considere uma curva de Bezier. De um ponto de vista geométrico qual a relevância da soma dos pesos atribuídos a cada ponto de controlo ser sempre 1 para todo o t, sendo todos os pesos positivos?

   Resposta: É importante, já que, desta forma, a curva não saí da figura formada pela união dos pontos de controlo, que pode ser útil para, por exemplo, *culling*.

3. Considere um ponto numa curva de *Catmull-Rom.* para orientar correctamente um modelo cuja "frente" esteja orientada para o eixo dos Z, é necessário construir uma matriz de rotação, partindo do valor da derivada da curva e de um valor para o vector "up" inicial.
   a. Descreva matematicamente os passos necessários para construir a matriz.

      Resposta:

      $$\begin{aligned} \vec{Z_i} &= P'(t) \\ \overrightarrow{X_i} &= \overrightarrow{Y_{i-1}} \times \vec{Z_i} \\ \vec{Y_i} &= \vec{Z_i} \times \overrightarrow{X_i} \\ \vec{Y_0} &= \text{"up" inicial} \end{aligned} \qquad M = \begin{bmatrix} \overrightarrow{X_i}_x & \vec{Y_i}_x & \vec{Z_i}_x & 0 \\ \overrightarrow{X_i}_y & \vec{Y_i}_y & \vec{Z_i}_y & 0 \\ \overrightarrow{X_i}_z & \vec{Y_i}_z & \vec{Z_i}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   b. Utilizando esta matriz, qual o efeito que se obtém se o objecto estiver inicialmente virado para o eixo do X? E como lidar com esta situação?

      Resposta: O objeto ficará rodado 90º (vai andar de lado). Para resolver basta aplicar também uma matriz de rotação.

4. Descreva matematicamente o processo da obtenção do vector normal a uma superfície cúbica de Bezier.

[3] Uma melhor explicação deste exercício pode ser vista no vídeo teórico das Curvas de *Bezier*.

Resposta: Para um ponto $A(u,v)$, sabemos que:

$$p(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

Sendo $M = M^T$ a matriz de coeficientes das superfícies de *bezier* e P_{ij} o ponto (i,j) da superfície.

$$\vec{u} = \frac{\partial p(u,v)}{\partial u} = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

$$\vec{v} = \frac{\partial p(u,v)}{\partial v} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix}$$

$$\vec{n} = \vec{v} \times \vec{u}$$

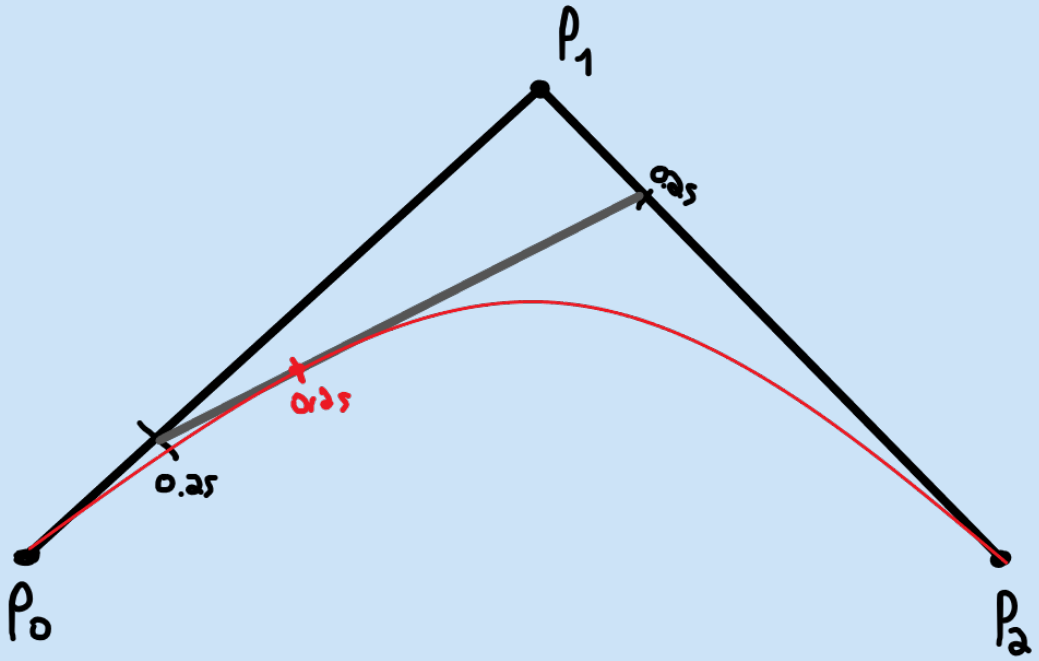
5. Uma curva quadrática só tem três pontos de controlo. Derive a fórmula do cálculo dos pontos para curvas de grau 2.

Resposta: Sendo P_i o ponto de controlo de índice i:

$$C_1(t) = P_0 \times (1-t) + P_1 \times t$$
$$C_2(t) = P_1 \times (1-t) + P_2 \times t$$
$$C(t) = C_1(t) \times (1-t) + C_2(t) \times t$$

6. Considere os seguintes pontos de controlo (em 2D) de uma curva quadrática: $P_0(0,0)$, $P_1(1,1)$, $P_2(2,0)$. Utilizando o método de *De Casteljau* apresente o diagrama para o cálculo do ponto quando $t = 0.25$.

Resposta:



Visualização interativa em:

- https://webglfundamentals.org/webgl/lessons/webgl-3d-geometry-lathe.html
- https://www.desmos.com/calculator/s78usaowv9?lang=pt-PT

Ficha Iluminação

1. Considere o seguinte excerto de código:

```
float p[4] = {0.0, 1.0, 0.0, 1.0};
glLightfv(GL_LIGHT0, GL_POSITION, p);
gluLookAt(5, 0, 5, 0, 0, 0, 0, 1, 0);
glLightfv(GL_LIGHT1, GL_POSITION, p);
drawEsfera(); // desenha esfera de raio 1 centrada na origem
```

   De acordo com o seguinte código, assinale as afirmações verdadeiras:

   a. A posição da luz 1 no espaço global é dependente da posição da câmara.

   Resposta: Falso.

   b. A posição da luz 1 no espaço câmara é fixa.

   Resposta: Falso.

   c. A posição da luz 0 no espaço global é dependente da posição da câmara.

   Resposta: Verdadeiro.

   d. A posição da luz 0 no espaço câmara é fixa.

   Resposta: Verdadeiro.

   e. No espaço global, a posição da luz 0 é idêntica à posição da luz 1 se a câmara for posicionada com `gluLookAt(0, 0, 0, 0, 0, 1, 0, 1, 0)`.

   Resposta: Verdadeiro, como a posição da câmara é $(0, 0, 0)$, a posição no espaço global não alterará após a transformação.

   f. No espaço global, a posição da luz 0 é idêntica à posição da luz 1 se a câmara for posicionada com `gluLookAt(0, 0, 0, 0, 0, -1, 0, 1, 0)`.

   Resposta: Verdadeiro, mesma lógica da alínea anterior.

   g. No espaço global, a posição da luz 0 é idêntica à posição da luz 1 se a câmara for posicionada com `gluLookAt(0, 0, 0, 0, 0, 1, 0, -1, 0)`.

   Resposta: Falso, a posição da luz 1 estará em $(0, -1, 0)$ no espaço global visto que o referencial agora está a "apontar para baixo".

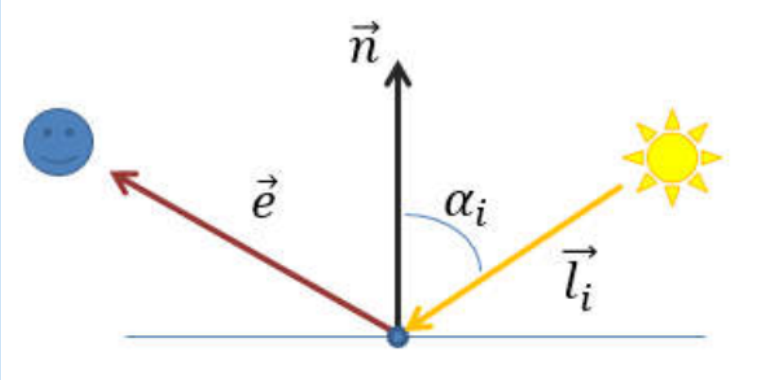
2. Enumere e caracterize as diferentes componentes da cor utilizadas nos materiais em OpenGL.

Resposta:
- `GL_DIFFUSE`: Cor difusa do material (cor com luz);
- `GL_AMBIENT`: Cor ambiente do material (cor sem luz);
- `GL_EMISSIVE`: Cor emitida pelo material (cor independente da luz);
- `GL_SPECULAR`: Cor especular do material (cor brilhante do objeto);
- `GL_SHININESS`: Fator de *shininess* (mancha brilhante) do material, o expoente especular do material.

3. Considere duas das componentes da equação de iluminação: difusa e especular. Apresente a equação de cada componente suportada por um diagrama indicando claramente os elementos envolvidos na equação.

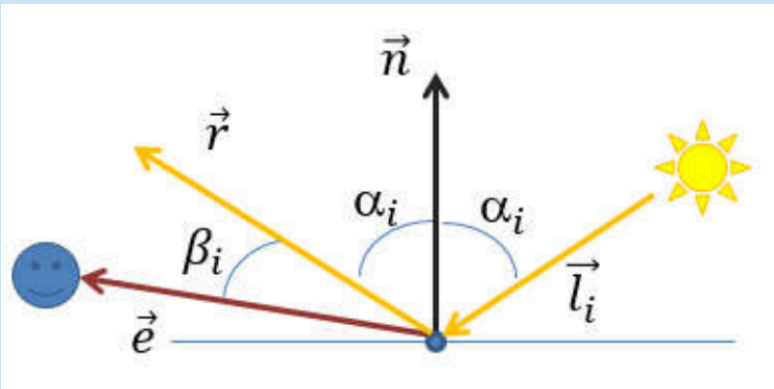
Resposta:

- **Luz difusa**
  Sendo:
  - $\vec{l_i}$ o vetor que vai desde o ponto de luz até à superfície refletora;
  - $\vec{e}$ o vetor que vai desde o ponto de contacto da luz na superfície refletora até à câmara;
  - $\vec{n}$ o vetor normal à superfície;
  - α_i o ângulo que o vetor $\vec{n}$ faz com $-\vec{l_i}$;
  - K_d a cor difusa do material do objeto
  - I_i a intensidade da luz



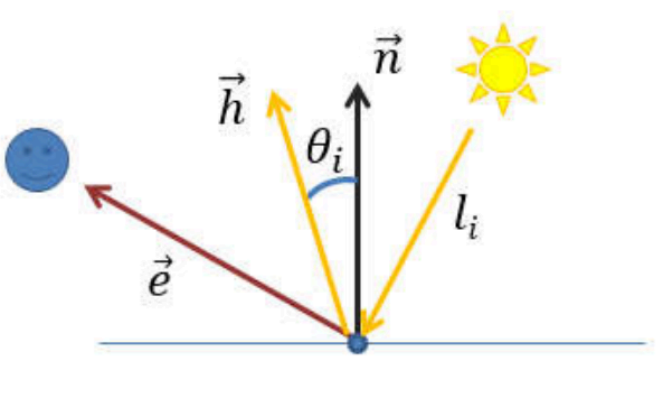
$$I_d = K_d \times I_i \times \cos(\alpha_i)$$

- **Luz especular (Phong)**
  Sendo:
  - $\vec{r}$ o vetor $\vec{l_i}$ refletido a partir do vetor $\vec{n}$
  - β_i o ângulo que o vetor $\vec{r}$ faz com o vetor $\vec{e}$
  - K_s a cor especular do material do objeto
  - $shininess$ o valor do *shininess* do material do objeto



$$I_e = K_s \times I_i \times \cos(\beta_i)^{shininess}$$

- **Luz especular (Blinn)**
  Sendo:
  - $\vec{h}$ o vetor médio entre $\vec{e}$ e $-\vec{l_i}$
  - θ_i o ângulo formado entre $\vec{h}$ e $\vec{n}$



$$I_e = K_s \times I_i \times \cos(\theta_i)^{shininess}$$

4. Os cálculos de iluminação beneficiam do facto de os vectores envolvidos serem vectores unitários. Justifique porquê.

Resposta: Atentando nas fórmulas da luz apresentadas no exercício anterior, é notável que é utilizado um cosseno com um ângulo entre dois vetores.

Relembrando a fórmula do produto interno:

$$\vec{a} \cdot \vec{b} = |\vec{a}|\left|\vec{b}\right| \cos(\alpha)$$

Ora, se os dois vetores forem unitários, isto é, $|\vec{a}| = 1$ e $\left|\vec{b}\right| = 1$, temos que:

$$\vec{a} \cdot \vec{b} = \cos(\alpha)$$

Ora, relembrando que o produto interno também pode ser calculado a partir da multiplicação matricial entre os dois pontos:

$$\vec{a} \cdot \vec{b} = \begin{bmatrix} \vec{a}_x & \vec{a}_y & \vec{a}_z & \dots \end{bmatrix} \begin{bmatrix} \vec{b}_x \\ \vec{b}_y \\ \vec{b}_z \\ \dots \end{bmatrix}$$

Podemos poupar o custo de cálculo de cossenos (que é computacionalmente custoso), obtendo o valor dele a partir de multiplicação de matrizes, melhorando drasticamente a eficiência dos cálculos das luzes.

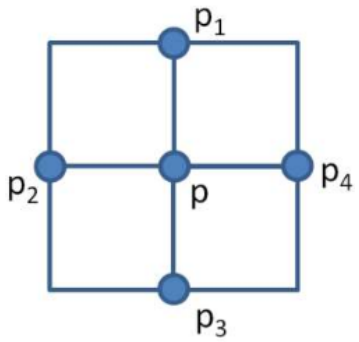
5. Descreva as características e limitações do modelo de iluminação de Gouraud com interpolação. De que forma o modelo de Phong resolve os problemas associados a essas limitações?

Resposta: O modelo de Gouraud diz que cada vértice de cada triângulo tem uma normal associada (em vez de apenas uma normal por triângulo em modelo mais simples). Esta normal, em casos de superfícies curvas, será igual à normal da superfície curva que estamos a tentar aproximar, de forma que, triângulos juntos que aproximem uma superfície curva tenham a mesma normal nos vértices adjacentes. Após termos a normal, as intensidades de luz são calculadas para cada vértice, e para cada pixel do triângulo estas intensidades são interpoladas. Isto resolve o problema de modelos de luz mais simples que faz com que se note diferenças de luzes entre cada triângulo, em vez de ser uma transição contínua.

Mesmo assim, este modelo tem alguns problemas. Se uma luz incidir apenas a meio do triângulo (por exemplo uma *spotlight*), essa luz não vai ser registada em nenhum dos vértices, ou seja, não vai ser captada, o que não é esperado. Outro problema é que, devido à interpolação, detalhes especulares podem não ser representados corretamente, devido a também só serem calculados uma vez por vértice.

O modelo de Phong já leva mais detalhe em conta. No caso, em vez de ser calculado uma normal para cada vértice, essa normal também é calculada, mas para cada pixel, essa normal é interpolada com as dos vértices, e a luz é calculada para cada pixel. Ora isto resolve o problema da *spotlight* enunciada anteriormente já que a intensidade da luz será calculada novamente para cada pixel, captando assim essa luz. O mesmo pode ser dito para os detalhes especulares que também vão ser calculados para cada pixel e não serão interpolados.

6. Considere que se pretende usar uma grelha para representar um terreno, à semelhança do que foi pedido no trabalho prático. As coordenadas dos pontos da grelha são números inteiros e a dimensão dos lados de cada quadrícula da grelha é uma unidade. Para obter a altura dos pontos da grelha é disponibilizada a função $h(p_i)$, sendo p_i um ponto da grelha. Para se poder calcular a iluminação dos pontos da grelha é necessário calcular a normal em cada ponto. Com base na figura, indique como proceder matematicamente para calcular a normal do ponto p.



Resposta: Sendo

$$\vec{p} = (p_x, h(p), p_z)$$
$$\vec{p_1} = \left(p_{1_x}, h(p_1), p_{1_z}\right)$$
$$\vec{p_2} = \left(p_{2_x}, h(p_2), p_{2_z}\right)$$
$$\vec{p_3} = \left(p_{3_x}, h(p_3), p_{3_z}\right)$$

Calcular vetores desde $\vec{p}$ até $\vec{p_i}$:

$$\vec{v_1} = \vec{p} - \vec{p_1}$$
$$\vec{v_2} = \vec{p} - \vec{p_2}$$
$$\vec{v_3} = \vec{p} - \vec{p_3}$$
$$\vec{v_4} = \vec{p} - \vec{p_4}$$

Calcular normais entre esses vetores:

$$\vec{n_1} = \frac{\vec{v_1} \times \vec{v_2}}{\|\vec{v_1} \times \vec{v_2}\|}$$
$$\vec{n_2} = \frac{\vec{v_2} \times \vec{v_3}}{\|\vec{v_2} \times \vec{v_3}\|}$$
$$\vec{n_3} = \frac{\vec{v_3} \times \vec{v_4}}{\|\vec{v_3} \times \vec{v_4}\|}$$
$$\vec{n_4} = \frac{\vec{v_4} \times \vec{v_1}}{\|\vec{v_4} \times \vec{v_1}\|}$$

Calcular média entre todas as normais:

$$\vec{n} = \frac{\vec{n_1} + \vec{n_2} + \vec{n_3} + \vec{n_4}}{4}$$

(Como a grelha é regular, também era possível apenas calcular dois vetores, de p_2 até p_4 e de p_1 até p_3 e fazer o produto externo desses dois vetores para obter a normal. Manteve-se a resolução que também é compatível com grelhas irregulares que aparece em alguns testes.).

7. A equação de iluminação contempla 3 componentes: ambiente, difusa e especular. Classifique, e justifique tendo em conta os vários modelos de shading e as várias componentas da equação de iluminação, cada uma das seguintes afirmações como sendo verdadeiras ou falsas:

Atenção! Perguntas como estas não foram para ser exatas e sim para se discutir sobre o assunto de luzes. Várias respostas podem estar certas dependendo da justificação.

a. Com uma luz direccional, a intensidade emitida por todos os pixels de um triângulo é sempre igual.

Resposta: Verdadeiro com Flat Shading e Interpolation Shading, falso com o método de Gouraud e Phong, já que estes poderão ter normais diferentes para cada vértice, assim alterando a intensidade de cada pixel.

b. A componente difusa da iluminação depende somente do vector da direcção da luz.

Resposta: Falso, também depende da intensidade da luz, da cor especular do material do objeto e da normal.

c. A componente especular depende somente da posição da câmara.

Resposta: Falso, mesma justificação da alínea anterior.

d. A intensidade da componente especular é mínima quando a posição da luz coincide com a posição da câmara.

Resposta: Falso, no caso que também coincide com a normal, o ângulo será 0, $\cos(0) = 1$, logo será máxima neste caso. Poderá ser um pouco verdade noutras situações, mas também depende do ângulo.

e. A intensidade da componente difusa é máxima quando a normal e a direcção que aponta para a luz coincidem.

Resposta: Verdadeiro, o ângulo formado será 0, $\cos(0) = 1$, logo será máxima.

f. Uma luz pontual nunca ilumina de forma igual todos os vértices de um triângulo.

Resposta: Falso, se for um triângulo equilátero e a luz estiver por cima do centro do triângulo, todos os pontos terão a mesma distância à luz, ou seja, a mesma intensidade de iluminação.

8. Distinga, de um ponto de vista computacional, os modelos de shading de Phong e Gouraud.

Resposta: Como enunciado na resposta da pergunta 5, o modelo de Gouraud, calcula a intensidade da luz para cada vértice do triângulo, fazendo interpolações da intensidade para cada pixel. Já o modelo de Phong, faz interpolação das normais dos vértices para cada pixel e calcula a intensidade da luz para cada um. Ora, desta forma, o modelo de Phong será normalmente computacionalmente mais custoso do que o modelo de Gouraud, já que mais cálculos de luz terão que ser feitos, em casos que hajam mais píxeis do que vértices.

9. Distinga de um ponto de vista qualitativo, considerando a componente especular, os modelos de shading de Phong e Gouraud.

Resposta: Como enunciado na resposta da pergunta 5, o modelo de Phong terá uma qualidade melhor sobretudo na componente especular já que esta não será interpolada a partir da sua intensidade nos vértices, mas sim calculada para cada pixel.

10. O modelo de Gouraud apresenta problemas quando nenhum dos vértices de um triângulo parcialmente iluminado recebe luz. Diga de que forma o modelo de Phong resolve este problema.

Resposta: Explicado na resposta da pergunta 5.

11. O modelo Flat assume que a luz está infinitamente distante. Justifique porquê.

Resposta: O modelo flat calcula a intensidade da luz apenas uma vez por triângulo. Ora para isto estar visualmente correto, era necessário que todos os vértices tenham a mesma normal, tendo assim a mesma intensidade da luz. Isso só acontece caso a luz esteja infinitamente distante.

Relembrar que a equação da luz difusa tem em conta o ângulo entre a normal e a direção da luz.

12. O modelo Flat assume que a câmara também está infinitamente distante. Justifique porquê.

Resposta: De razão semelhante à resposta da pergunta anterior, como o modelo Flat calcula apenas uma normal por cada triângulo, a componente especular será a mesma para todo o triângulo. Isto só faz visualmente sentido se a câmara estiver infinitamente distante já que só assim é que todos os píxeis do triângulo têm a mesma normal, e assim a mesma intensidade especular.

Relembrar que a equação da luz especular tem em conta o ângulo entre a direção da câmara a partir do vértice e o raio refletido (ou o meio-vetor no caso da equação de Blinn).

Ficha Texturas

1. Para obter transparências parciais é necessário ordenar os triângulos de modo a que os triângulos transparentes sejam desenhados só no final, ordenados por distância decrescente à câmara. Justifique esta necessidade.

   Resposta: A necessidade vem das transparências parciais precisarem de ser aplicadas por ordem, já que se for desenhado por exemplo um vidro, a cor do vidro tem de ser combinada com que está atrás do vidro. Se não houver ordenação, essa combinação pode ser feita com triângulos errados.

2. Descreva como funciona o mecanismo de transparências totais utilizando o teste do canal alpha.

   Resposta: O teste do canal alpha é um teste configurável, que pode passar se o valor α da textura for maior ou menor do que um determinado valor, ou não passar se for maior ou menor que um determinado valor. Se o teste passar e a distância do que estamos a renderizar à câmara for menor do que o valor calculado que está no *z-buffer* atualmente, renderiza e atualiza o *z-buffer* com a nova distância. Se não passar, é ignorado e não é atualizado o *z--buffer*.

3. Descreva o problema de amostragem resultante de se projectar uma textura no ecrã numa área com um número de pixels muito inferior à dimensão da textura.

   Resposta: Como o ecrã não tem detalhe o suficiente para representar a textura na sua totalidade, píxeis dela têm de ser escolhidos para serem mostrados no ecrã. Isto torna-se um problema, já que, ao fazer a transformação de coordenadas no ecrã para coordenadas de textura, como a textura é maior que o ecrã, píxeis do ecrã adjacentes terão coordenadas de textura demasiado distantes. Ora, se a distância for grande o suficiente para calhar em outra parte da textura, outra cor completamente diferente será escolhida, o que fará existir uma transição pouco suave entre píxeis do ecrã. Este problema, juntando com um movimento ligeiro da câmara, fará com que muitos píxeis troquem de valor de forma não suave, dando o efeito de *flickering*.

4. Descreva o processo de amostragem utilizando o filtro `GL_LINEAR` e `GL_NEAREST`.

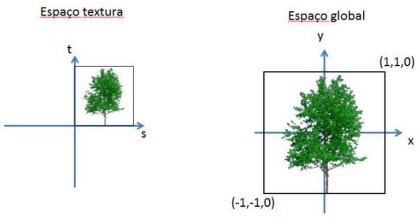
   Resposta: `GL_NEAREST`: para um pixel do ecrã, o valor de textura é calculado a partir do centro do pixel da textura mais próximo nas coordenadas de textura. `GL_LINEAR`: para um pixel do ecrã, o valor de textura é calculado a partir dos 4 píxeis da textura mais próximos da coordenada de textura, fazendo interpolação linear com a distância ao centro deles.

5. Descreva o mecanismo de *mipmapping*, indicando as suas vantagens e desvantagens.

   Resposta: O mecanismo de *mipmapping* faz com para uma textura, várias resoluções sejam criadas para ela. Isto resolve o problema enunciado na pergunta 3, já que, no caso de projetar uma textura maior do que o tamanho do ecrã, uma textura com menor tamanho será escolhida para minimizar o *flickering*. No entanto, essas mesmas texturas terão de ser geradas e colocadas em memória gráfica, gastando mais recursos. Em tempo real, existe um custo computacional (apesar de pouco notável, devido ao uso forte da *cache* da GPU), ao escolher a textura, baseando-se na distância, mesmo que ela já esteja em memória da GPU.

6. Considere que se pretende mapear uma textura num *QUAD*. Complete o código seguinte para obter o resultado da figura.

```
glBindTexture(GL_TEXTURE_2D,texID);
glBegin(GL_QUADS);
  glTexCoord2f(___,____); glVertex3f(___,___,___);
  glTexCoord2f(___,____); glVertex3f(___,___,___);
  glTexCoord2f(___,____); glVertex3f(___,___,___);
  glTexCoord2f(___,____); glVertex3f(___,___,___);
glEnd();
```



Resposta:

```
glTexCoord2f(0, 0); glVertex3f(-1, -1, 0);
glTexCoord2f(1, 0); glVertex3f(1, -1, 0);
glTexCoord2f(1, 1); glVertex3f(1, 1, 0);
glTexCoord2f(0, 1); glVertex3f(-1, 1, 0);
```

Ficha Culling

1. Compare em termos computacionais os três tipos de *culling* apresentados na disciplina.

   Resposta:
   - *Back face Culling*: *culling* de triângulos que não estejam virados para a câmara, computacionalmente muito leve;
   - *View frustum Culling*: *culling* de modelos que não estejam visíveis no *frustum* da câmara, computacionalmente mais pesado que *Back face Culling* devido a cálculos extras de planos e de que lado os modelos se encontram neles;
   - *Occlusion Culling*: *culling* de modelos que aparecem por trás de outros modelos, computacionalmente o mais pesado.

2. Descreva o processo matemático para obter a equação normalizada do plano que contem os pontos p_1, p_2 e p_3.

   Resposta:
   Calcular normal do plano:

   $$\vec{v_1} = p_1 - p_2$$
   $$\vec{v_2} = p_3 - p_2$$
   $$\vec{n} = \frac{\vec{v_1} \times \vec{v_2}}{\|\vec{v_1} \times \vec{v_2}\|}$$

   Logo o plano será da forma:

   $$\vec{n}_x x + \vec{n}_y y + \vec{n}_z z + d = 0$$

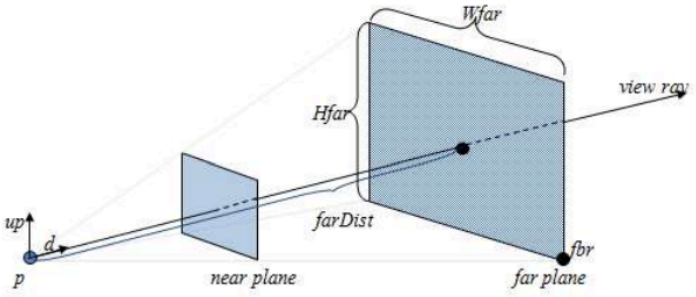
   Calcular $d \rightarrow$ Substituir (x, y, z) por um ponto do plano qualquer (escolhemos p_1):

   $$\vec{n}_x p_{1_x} + \vec{n}_y p_{1_y} + \vec{n}_z p_{1_z} + d = 0$$
   $$d = -\left(\vec{n}_x p_{1_x} + \vec{n}_y p_{1_y} + \vec{n}_z p_{1_z}\right)$$

   Logo o plano terá a forma:

   $$\vec{n}_x x + \vec{n}_y y + \vec{n}_z z - \left(\vec{n}_x p_{1_x} + \vec{n}_y p_{1_y} + \vec{n}_z p_{1_z}\right) = 0$$

4. Considere os vectores $\vec{d}$ e $\overrightarrow{up}$, o ponto p, e as distâncias $farDist$, $Wfar$ e $Hfar$, apresentados na figura. Descreva o processo matemático para obter o ponto fbr.



Resposta: Assume-se que $\overrightarrow{up}$ e $\vec{d}$ sejam vetores normalizados e $\overrightarrow{up}$ seja o vetor real para cima da câmara (e não um aproximado) (ver próximo exercício):

$$\vec{r} = \vec{d} \times \overrightarrow{up}$$

$$fbr = \vec{d} \times farDist + \vec{r} \times \frac{Wfar}{2} - \overrightarrow{up} \times \frac{Hfar}{2}$$

5. Considere agora que tem somente os dados presentes nas seguintes instruções:

```
gluPerspective(fov, ratio, nearDist, farDist);
gluLookAt(px,py,pz, lx,ly,lz, ux,uy,uz);
```

Descreva o processo matemático para obter os dados referidos na pergunta anterior: vectores $\vec{d}$, $\overrightarrow{up}$ e $\vec{r}$, e as distâncias $Wfar$ e $Hfar$.

Resposta:

$$p = (p_x, p_y, p_z)$$

$$\vec{l} = (l_x, l_y, l_z)$$

$$\vec{u} = (u_x, u_y, u_z)$$

$$\vec{d} = \frac{\vec{l} - p}{\left\|\vec{l} - p\right\|}$$

$$\vec{r} = \vec{d} \times \frac{\vec{u}}{\|\vec{u}\|}$$

$$\overrightarrow{up} = \vec{r} \times \vec{d}$$

$$\frac{Hfar}{2} = \tan\left(\frac{fov}{2}\right) \times farDist$$

$$Hfar = 2 \times \tan\left(\frac{fov}{2}\right) \times farDist$$

$$Wfar = Hfar \times ratio$$

6. Apresente o algoritmo para extrair os planos do *view frustum* segundo a visão geométrica.

Resposta: Como um plano pode ser calculado a partir de uma normal $\vec{n}$ e um ponto o:

$$\vec{m} = \vec{d} \times farDist$$

Plano near

$$o = p + \vec{d} \times nearDist$$
$$\vec{n} = \vec{d}$$

Plano *far*

$$o = p + \vec{d} \times farDist$$
$$\vec{n} = -\vec{d}$$

Plano *left*

$$o = p$$
$$\vec{n} = \left(\vec{m} - \vec{r} \times \frac{Wfar}{2}\right) \times \overrightarrow{up}$$

Plano *right*

$$o = p$$
$$\vec{n} = \overrightarrow{up} \times \left(\vec{m} + \vec{r} \times \frac{Wfar}{2}\right)$$

Plano *top*

$$o = p$$
$$\vec{n} = \left(\vec{m} + \overrightarrow{up} \times \frac{Hfar}{2}\right) \times \vec{r}$$

Plano *bottom*

$$o = p$$
$$\vec{n} = \vec{r} \times \left(\vec{m} - \overrightarrow{up} \times \frac{Hfar}{2}\right)$$

7. Descreva o algoritmo para extrair os planos do *view frustum* em *clip space*.

Resposta: A partir das matrizes de *model view*, M e de projeção, P:

Um ponto em *clip space*, p_c, pode ser calculado a partir do mesmo em *object space*, p_m:

$$p_c = P \times M \times p_m$$

Este ponto p_c está dentro do *frustum* se:

$$-w_c \leq p_{c_x} \leq w_c$$
$$-w_c \leq p_{c_y} \leq w_c$$
$$-w_c \leq p_{c_z} \leq w_c$$

Sendo w_c o tamanho do *frustum* do *clip space*,

Esse p_c também pode ser escrito de forma matricial:

$$A = P \times M \Rightarrow p_c = A \times p_m$$

Este A pode ser visto com uma matriz de 4 linhas:

$$A = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix}$$

Então:

$$p_c = A \times p_m = \begin{bmatrix} l_1 \times p_m \\ l_2 \times p_m \\ l_3 \times p_m \\ l_4 \times p_m \end{bmatrix} = \begin{bmatrix} p_{x_c} \\ p_{y_c} \\ p_{z_c} \\ w_c \end{bmatrix}$$

Substituindo estes novos valores numa das inequações acima temos:

$$\begin{aligned} & -w_c \leq p_{c_x} \\ \Leftrightarrow\ & -l_4 \times p_m \leq l_1 \times p_m \\ \Leftrightarrow\ & 0 \leq l_1 \times p_m + l_4 \times p_m \\ \Leftrightarrow\ & 0 \leq (l_1 + l_4) \times p_m \\ \Leftrightarrow\ & 0 \leq (a_{11} + a_{41})x_m + (a_{12} + a_{42})y_m + (a_{13} + a_{43})z_m + (a_{14} + a_{44})w_m \end{aligned}$$

E assim temos a equação de um dos planos do *frustum*. O processo pode ser repetido para o resto das inequações, tendo assim acesso aos 6 planos do *frustum*.

8. Por forma a tornar eficiente o algoritmo de view frustum culling é necessário implementar algum mecanismo de agrupamento de triângulos. Descreva o processo de partição espacial baseado em k-D trees.

Resposta: O processo baseia-se na divisão recursiva de conjuntos de triângulos da cena conforme a sua posição. Inicialmente, cria-se um plano paralelo ao eixo do X (ou Y) que tenta dividir a cena em metade dos triângulos para um lado, e outra metade para a outra, e para cada uma delas agora dividir metade dos triângulos com um plano paralelo ao eixo do Y. A cada divisão, o eixo paralelo troca, e o processo continua recursivamente.

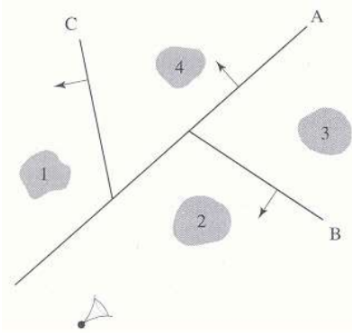
9. Os processos de partição espacial são em regra recursivos na construção da estrutura de dados. Indique três critérios possíveis para terminar a recursividade.

Resposta:
- Número de triângulos numa célula for menor que um determinado valor;
- Quando o volume da célula for menor que um determinado valor;
- Quando a árvore atingir uma certa profundidade.

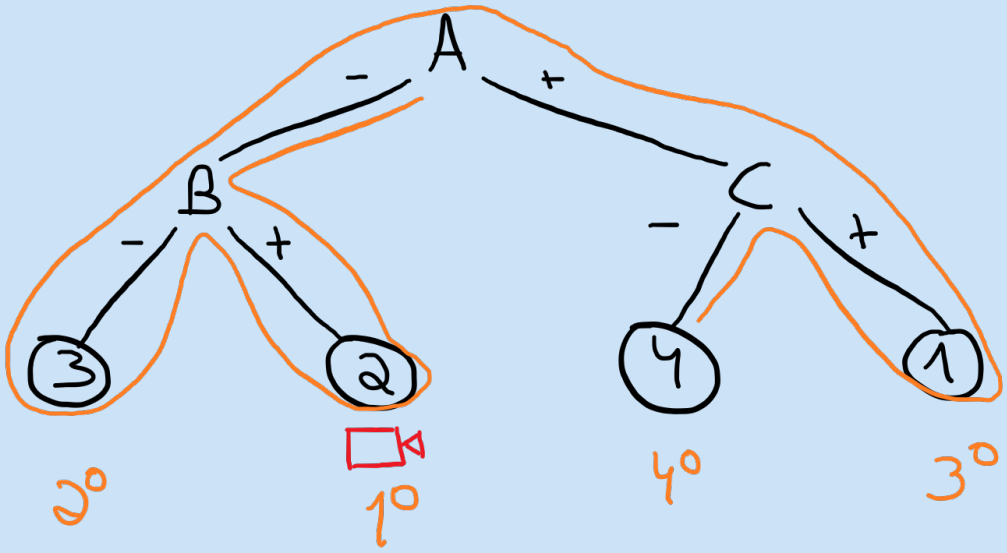
10. Num processo de partição espacial é possível que um triângulo pertença a mais que um filho. Indique quais as opções disponíveis nestes casos apresentando as vantagens e desvantagens de cada uma.

Resposta:
- Opção 1: colocar o triângulo que foi particionado no nodo pai;
  Vantagens: Menos memória usada;
  Desvantagens: Poucos triângulos vão ficar associados ao pai, já que poucos triângulos irão ficar entre bordas de divisões (subutilização da placa gráfica).
- Opção 2: colocar cada uma das partições do triângulo em cada um dos filhos;
  Vantagens: Menos píxeis podem ser renderizados;
  Desvantagens: Mais memória usada, mais vértices a serem processados.
- Opção 3: duplicar o triângulo nos dois filhos;
  Vantagens: Mais simples de programar e o processo torna-se mais eficiente, já que, mesmo que renderize os dois triângulos, o segundo não vai ser processado pois o primeiro já ocupou o espaço no *z-buffer*.
  Desvantagens: Mais memória usada (em termos de índices).

11. Considere a seguinte divisão do espaço utilizando uma BSP. Construa a árvore correspondente e, dada a posição da câmara indicada na figura, apresente a ordem de desenho dos objectos de forma a minimizar a escrita de pixels.



Resposta:



12. Indique os tipos de volumes envolventes que poderiam ser utilizados numa partição hierárquica, comparando a sua eficiência em termos de culling e complexidade algorítmica.

Resposta:

AABB (Axis Aligned Bounding Box)

Uma *bounding box* alinhada com os eixos.

- Espaço pouco otimizado;
- Eficiente na construção;
- Eficiente nos testes.

OBB (Object Bounding Box)

Uma *bounding box* sem estar alinhada com os eixos.

- Espaço mais ou menos otimizado;
- Mais ou menos eficiente na construção;
- Mais ou menos eficiente nos testes.

Sphere

Uma esfera que contém o objeto.

- Espaço pouco otimizado;
- Eficiente na construção;
- Eficiente nos testes.

Convex Hull

Figura complexa que contém o objeto com várias divisões.

- Espaço muito otimizado;
- Pouco eficiente na construção;
- Pouco eficiente nos testes.

13. Descreva detalhadamente o processo optimizado de teste de inclusão no VFC com paralelepípedos alinhados com os eixos.

Resposta: O teste de inclusão passa por testar se o paralelepípedo está dentro ou fora do *frustum*.

Para isso, podemos determinar qual é o *p-vertex* do paralelepípedo para cada plano. Este *p--vertex* será o vértice que terá maior valor no teste da distância ao plano. Desta forma, podemos olhar para a normal do plano e ver se cada um dos seus componentes são positivos ou negativos. O *p-vertex* terá componente máxima se o mesmo componente na normal for positivo, e a componente mínima se for negativo. Ou seja, se a normal tiver $x < 0$ e $y > 0$, *p-vertex* $= \left(\min_x, \max_y\right)$.

Agora com o *p-vertex* conseguimos calcular a distância ao plano, substituindo esse ponto na equação do plano. O resultado, como o plano é normalizado, será a distância ao plano, e será positiva se estiver do lado positivo do plano, e negativa caso esteja no lado negativo do plano. Caso esteja do lado negativo, está fora do *frustum* e podemos não renderizar esse paralelepípedo (não necessitando de testar outros planos). Se o teste para todos os planos do *frustum* der positivo, concluímos que o paralelepípedo está dentro do *frustum* (pelo menos parcialmente) e pode ser renderizado.