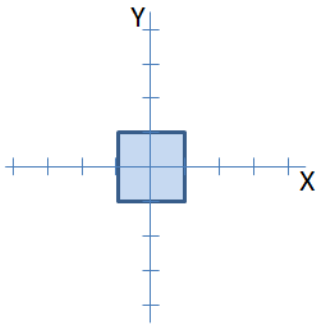
Prova Escrita
Notas sobre a resolução
CG (11/06/2013)

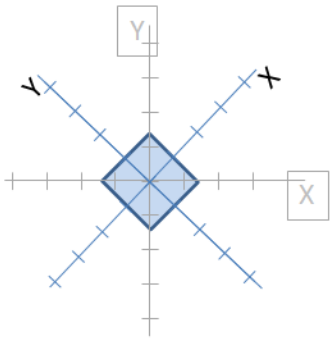
Questão 1.

A resposta correcta é a opção c).

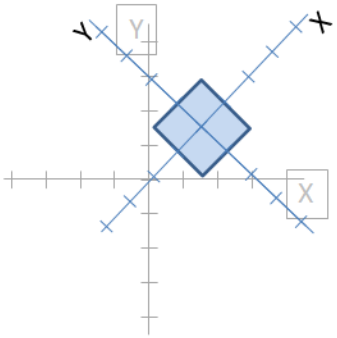
Os passos intermédios são os seguintes:



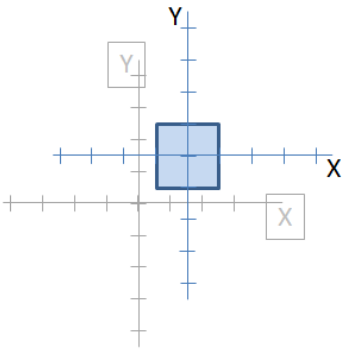
Posição inicial



1ª rotação



translação



2ª rotação

Questão 2.

2.1 Verdadeira. A única transformação que tem efeito no primeiro cubo é a primeira translação. A posição do centro será portanto 0,0,-3. Nota: esta também é a posição no espaço câmara.

2.2 Falsa. O cubo encontra-se a 7 unidades no eixo dos Zs, mas a câmara está a olhar para o ponto (0,0,-1), ou seja está de costas para o objecto, logo a coordenada será (0,0,7).

2.3 Verdadeira. O cubo não será visível porque a câmara esta de costas para o objecto. Ver resposta a 2.2

2.4 Verdadeira. O cubo não é visivel porque está tapado pelo primeiro cubo.

2.5 Falsa. É claramente falsa, os objectos nem estão à mesma distância da câmara, e estão em lados opostos da câmara.

Questão 3.

Para calcular a nova posição temos primeiro de descobrir os eixos da câmara. Temos portanto que calcular a direcção para onde a câmara esta a olhar, o vector right e o verdadeiro up.

$\overrightarrow{dir} = L - P$

$\overrightarrow{right} = \overrightarrow{dir} \times \overrightarrow{Up}$

$\overrightarrow{realUp} = \overrightarrow{right} \times \overrightarrow{dir}$

Uma vez calculados estes vectores, a nova posição é definida da seguinte forma

$newP = P + dh \cdot \overrightarrow{right} + dv \cdot \overrightarrow{up}$

Questão 4.

Esta questão também podia ser respondida através de um exemplo em que a multiplicação de matrizes não fosse comutativa.

Recorrendo a diagramas poderíamos utilizar o exemplo da pergunta 1. Caso a terceira linha trocasse de lugar com a segunda (uma troca da ordem entre uma rotação e uma translação) o resultado obtido seria a opção a) (as rotações anulam-se e portanto a transformação resultante seria só a translação). Na versão original o resultado é a opção c)

Questão 5.

Nesta questão era necessário referir a razão de ser do double buffering (não visualizar o processo de criação da imagem) e a utilização de dois buffers que alternadamente irão servir de escrita da imagem actual e de apresentação da imagem anterior, realizando-se a troca no final da frame.

Questão 6.

- Uma textura com mipmapping é uma sequência de imagens gradualmente mais pequenas, sendo a dimensão da imagem seguinte metade da anterior.

- Na aplicação é escolhida a imagem cuja resolução mais se aproxima da resolução indicada para o pixel em questão, ou uma combinação linear entre as duas imagens mais próximas.

Vantagens:

- melhor aspecto visual devido à consistencia entre a zona a texturizar e a imagem de onde são recolhidas as amostras;

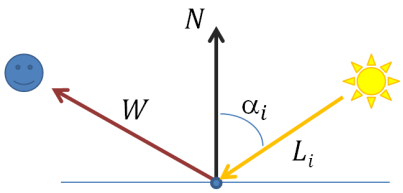
- potencialmente mais rapido quando as imagens selecionadas são as mais pequenas devido à cache de leitura.

Desvantagens:

- memória ocupada (< 1 + 1/3 da versão original)

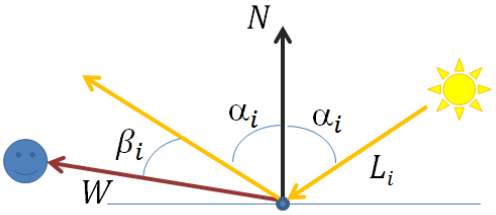
Questão 7.

Componente difusa



$$I = K^d \times I_i^d \times \cos(\alpha_i)$$

Componente especular



$$I = K^s \times I_i^s \times \cos(\beta_i)^{shininess}$$

Questão 8.

Processo de construção: Inicialmente determina-se o volume que contêm todos os triângulos do cenário. Seguidamente o volume é dividido em oito partes iguais, ou octantes. Os triângulos são então divididos pelos octantes que ocupam. Caso os vértices de um triângulo pertençam a mais que um octante então é possível dividi o triângulo de forma que cada parte só ocupe um octante; duplicá-lo e inserir em ambos os octantes; ou ainda guardá-lo numa lista associada ao volume pai dos octantes. O processo prossegue recursivamente criando uma árvore até que uma das três condições se verifique:

1. O volume é demasiado pequeno
2. O número de triangulos no volume é inferior a um determinado limíte
3. A profundidade da árvore criada excede um determinado valor.

Questão 9.

Sphere(1); // Sol

Rotate(alpha, 0, 1, 0) // rotação em torno do eixo dos YY

Translate(10, 0, 0); // distância entre o planeta e o sol

Rotate (10, 0, 0, 1); // inclinação orbital

Sphere(0.25); // planeta

Rotate(beta, 0, 1, 0); // Nova rotação em torno dos YY para afectar a posição da lua

Translate(1.5, 0, 0); // distância entre o planeta e a lua

Sphere (0.1); // lua