Sistemas Gráficos e Interacção

Época de Recurso 2023-02-10

N.º _____ Nome _____

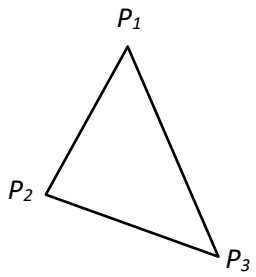
Duração da prova: 45 minutos
Cotação de cada pergunta: assinalada com parêntesis rectos
Perguntas de escolha múltipla: cada resposta incorrecta desconta 1/3 do valor da pergunta

Parte Teórica **10%**

a. **[3.3]** Qual a dimensão em bytes de um *frame buffer* RGBA de 1024 x 1024 x 16 bits?

i. 1 Megabyte
ii. 2 Megabyte
iii. 4 Megabyte
iv. Nenhuma das anteriores

b. **[3.3]** Dados três pontos P_1, P_2 e P_3 e a combinação afim $Q = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$, em que $\alpha_1 = 0.2$ e $\alpha_2 = 0.3$, é possível afirmar que



i. $\alpha_3 = 0.5$
ii. A combinação afim é convexa
iii. O ponto Q pertence à envolvente convexa dos pontos P_1, P_2 e P_3
iv. Todas as anteriores

c. **[3.3]** Dadas três transformações genéricas T (translação), R (rotação) e S (escala), a composição de transformações

i. T seguida de R produz o mesmo resultado que R seguida de T
ii. T seguida de S produz o mesmo resultado que S seguida de T
iii. R seguida de S produz o mesmo resultado que S seguida de R
iv. Nenhuma das anteriores

d. **[3.3]** Considere o objecto delimitado pela superfície descrita pela seguinte equação:

$$(x-1)^2 + (y-2)^2 + (z-3)^2 - 1 = 0$$

O ponto de coordenadas (1.0, 2.0, 4.0) encontra-se

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

e. **[3.3]** Qual das seguintes componentes do modelo de iluminação de Phong depende da posição do observador?

i. Ambiente
ii. Difusa
iii. Especular
iv. Todas as anteriores

f. **[3.3]** No mapeamento de texturas fará sentido definir um filtro de magnificação do tipo *Nearest Mipmap Linear*?

i. Sim. Será calculada 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 existente
ii. Sim. Será escolhido o *texel* que mais se aproxima do centro do pixel em cada um dos dois *mipmaps* que melhor se adequam ao contexto existente; em seguida, é efectuada uma interpolação linear destes dois valores
iii. Não. Será sempre usado o mapa de maior resolução
iv. Nenhuma das anteriores

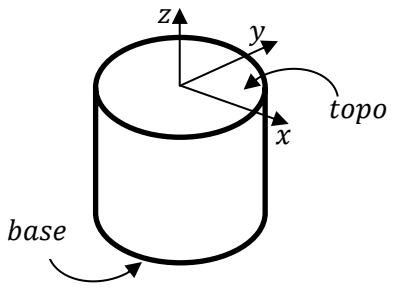
Sistemas Gráficos e Interacção

Época de Recurso 2023-02-10

N.º _____ Nome _____

Parte Teórico-Prática 20%

a. **[4.0]** Pretende-se definir os vectores normais para o cilindro da figura. Indique os valores para as normais do topo e da base, e a fórmula para a normal lateral, em função da variável *theta*, que contém o ângulo no plano OXY para o qual se está a calcular a normal.



Normal do topo: 0.0, 0.0, 1.0

Normal da base: 0.0, 0.0, -1.0

Normal lateral: cos(*theta*), sin(*theta*), 0.0

b. **[4.0]** Pretende-se definir uma câmara de topo sobre a personagem de um videojogo, alinhando a parte superior da câmara com o eixo dos xx.

Considerando que a altitude na cena está definida segundo o eixo dos zz, que a posição da personagem está guardada em $pers.x$, $pers.y$ e $pers.z$, e que a distância da câmara à personagem é dada pela constante $DIST$, preencha os parâmetros nas seguintes funções.

```
camera.position.set(pers.x, pers.y, pers.z + DIST);

camera.lookAt(pers.x, pers.y, pers.z);

camera.up.set(1.0, 0.0, 0.0);
```

c. **[1.2]** Para desenhar um círculo em *Three.js*

i. Tem-se obrigatoriamente de usar as equações paramétricas da circunferência
ii. Pode-se usar a *DiskGeometry*
iii. Pode-se usar a *CircumferenceGeometry*
iv. Pode-se usar a *CircleGeometry*

d. **[1.2]** Numa *PerspectiveCamera*, os parâmetros *near* e *far*

i. Servem para definir o campo de visão (*fov*)
ii. São obrigatoriamente valores maiores do que zero
iii. Podem assumir valores negativos
iv. Servem para definir o *aspect ratio* da câmara

e. **[1.2]** A actualização dos valores alterados por programação numa entrada da interface *lil-gui*

i. Não é possível
ii. É automática
iii. Tem de ser requerida, invocando o método *listen()* aquando da criação da interface
iv. Nenhuma das anteriores

f. **[1.2]** O método mais eficiente de definir uma *geometria* em *Three.js*, é definindo os polígonos

   i. Directamente a partir da lista de vértices
   ii. Usando uma lista de arestas
   iii. Usando uma lista de índices e uma lista de vértices
   iv. Todos os métodos anteriores têm a mesma eficiência

g. **[1.2]** As fontes de luz em *Three.js* que podem projectar sombras são:

   i. *DirectionalLight* e *PointLight*
   ii. *DirectionalLight*, *SpotLight* e *PointLight*
   iii. *AmbientLight*, *SpotLight* e *PointLight*
   iv. *AmbientLight*, *DirectionalLight*, *SpotLight* e *PointLight*

h. **[1.2]** Um *Render Target*, é

   i. O *canvas* onde o *Three.js renderiza* a *frame*
   ii. É uma aplicação, para a qual é passado o resultado da *renderização*
   iii. É um nome alternativo dado aos processadores gráficos
   iv. Uma textura especial, para a qual se pode *renderizar*

i. **[1.2]** A camara por omissão no *Three.js*

   i. Está na origem a olhar para a parte negativa do eixo dos xx
   ii. Está na origem a olhar para a parte negativa do eixo dos yy
   iii. Está na origem a olhar para a parte negativa do eixo dos zz
   iv. Está no ponto (0, 0, 5) a olhar para a origem

j. **[1.2]** Na utilização de um *RayCaster*, as coordenadas normalizadas do rato são valores entre:

   i. 0 e 1
   ii. -1 e 1
   iii. 0 e *largura* (ou *altura*) da janela
   iv. -0.5 * *largura* (ou *altura*) da janela e 0.5 * *largura* (ou *altura*) da janela

k. **[1.2]** O *AlphaMap* permite

   i. Ajustar a altura dos vértices de uma *mesh*
   ii. Ajustar a posição (*x*, *y*) dos vértices de uma *mesh*
   iii. Definir áreas mais ou menos brilhantes/baças
   iv. Definir áreas mais ou menos transparentes/opacas

l. **[1.2]** A maneira mais simples de criar um *billboard* (*sprite*) em *Three.js* é criar um objecto

   i. *Mesh* a partir de um *PlaneGeometry* e um *StandardMaterial*, e orientá-lo a cada *frame*
   ii. *Mesh* a partir de um *PlaneGeometry* e um *SpriteMaterial*
   iii. *Sprite* a partir de um *SpriteMaterial*
   iv. *Sprite* a partir de um *PlaneGeometry* e um *SpriteMaterial*