TÉCNICO LISBOA

Animação e Visualização Tridimensional

Mestrado em Engenharia Informática e de Computadores
Alameda

2nd Mini-Test
28th November 2018

The mini-test has a maximum duration of 45 minutes. Answer with black or blue pen to the following questions and **justify in detail** all the answers. If necessary you can use the back of the respective sheet to complete the answer. Calculators, cell phones or other mobile devices are not allowed. Identify all the sheets of your mini-test. **Good luck!**

1. Consider the following OpenGL 3.3 code sample.

```
glGenTextures(3, TextureArray);
LoadTexture(TextureArray[0], "wood.bmp");
LoadTexture(TextureArray[1], "steel.bmp");
LoadTexture(TextureArray[2], "orange.bmp");
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, TextureArray[1]);
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, TextureArray[2]);
glActiveTexture(GL_TEXTURE2);
glBindTexture(GL_TEXTURE_2D, TextureArray[0]);
tex_loc = glGetUniformLocation(shader.getProgramIndex(), "texmap");
……
void render_scene() {
        ……
        glUniform1i(tex_loc, 0);
        drawObject1();
        glUniform1i(tex_loc, 2);
        drawObject2();
        glUniform1i(tex_loc, 1);
        drawObject3();
        …… }
```

a) **[1.5v]** The code sample above does not render multitextured objects. Justify this by referring what texels are used to shade the three objects.

No multitexturing since each object is just rendered with one GLSL uniform sampler2D variable texmap: Object 1 with steel.bmp; object 2 with wood.bmp, object 3 with orange.bmp

b) **[1.5v]** Adapt the code in order to allow multitexturing.

Add a second GLSL uniform sampler2D variable texmap1 that is accessed in OGL by the tex_loc1 containing its location:

tex_loc1 = glGetUniformLocation(shader.getProgramIndex(), "texmap1");

and before drawing an object, for instance

glUniform1i(tex_loc, 0);
glUniform1i(tex_loc1, 1);
drawObject1();

Aluno: _____ _____

2. You want to map a 128x64 texture image into a rectangle whose texture coordinates are: {(0, 0), (0, 4), (4, 4), (4, 0)}. And you set up a texture object with the following calls:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR_MIPMAP_NEAREST);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
```

a) **[1.0v]** How many times will be the texture tiled in the rectangle? Why?

GL_REPEAT in each direction, so 4 x 4= 16 tiles

b) **[1.5v]** In what situation, the filter of the last command will be used? Is it a point or an area sampling technique? Why?

Magnification: one texel mapped to multiple pixels. Area sampling: to choose the nearest 2x2 texels, average them and in the pixel

c) **[2.5v]** Consider that a LookAt() call made the rectangle to be projected in the screen with the size of 8x4 pixels. Describe, in detail, the filtering technique that will be used by OpenGL in this situation.

mipmapping caracteriza-se pela construção de níveis de mapas de textura cuja resolução é progressivamente ¼ (1/2 em cada direcção) da resolução anterior até se atingir a resolução 1x1. Neste caso ter-se-ia em termos de memória os seguintes mapas de textura:
128x64 (nível 0); 64x32 (nível 1); 32x16 (nível 2); 16x8 (nível 3); 8x4 (nível 4); 4x2 (nível 5), 2x1 (nível 6), 1x1 (nível 7)

Com a opção GL_LINEAR_MIPMAP_NEAREST, apenas seleciona um mapa de textura e realiza um area sampling 2x2 nesse mapa e mapeia o valor calculado no pixel.
O mecanismo do OpenGL que realiza o acesso ao mapa de textura correcto,
baseia-se no cálculo do ρ que vale o máximo de (128/8; 64/4), ou seja 16.
Significa deste modo que λ =log2 ρ =log2 16 = 4: portanto é selecionado o mapa de nível 4: 8x4

3. Consider the rendering of a 3D scene with opaque and translucent objects where the visibility problem will be solved by using the Zbuffer-based two-step approach studied in this Course.

a) **[2.5v]** Why two steps? And how do you would implement it?

Firstly, a step to draw the opaque objects. Then, another step to draw the translucent objects with blending mechanism enabled. In the fragment shader, for each step, I would check the alpha channel of the incoming fragment, and, accordingly, I would discard it (with discard GLSL built-in function) or accept it. In the first step fragments with alpha != 1 would be discarded. In the second step,

only the fragments with alpha]0, 1[would be accepted (alpha == 0 would be also discarded for optimization)

b) **[2.5v]** In the second step, describe how the OpenGL depth-test should be configured?

Desactiva-se a escrita no z-buffer (mas o teste z-buffer continua activo) para evitar que em caso de um translúcido à frente de um objecto opaco altere o valor da profundidade armazenada no z-buffer removendo este último o que seria indesejável. Ou seja, o z-buffer só guarda os valores de profundidade dos opacos. Mantém-se, no entanto, o teste de visibilidade activo para evitar que objectos translúcidos que estão atrás de objectos opacos sejam desenhados.

4. You implemented the 2D Lens Flare algorithm to simulate an optical effect created by inter reflections between elements of a lens when the camera is pointed toward a bright light.

a) **[1.0v]** Why do you turn off the depth-test?

To guarantee that the flare will be always drawn in FRONT of the scene.

b) **[2.0v]** How did you position the several elements of the flare in the screen?

The position of each element of a flare is interpolated along a line from the projected position of the light to a point opposite it across the center of the screen, using its parametric position (a value between [0,1].

5. You want to implement planar reflections on a mirror wall placed at x = 0. All the other objects of the scene, including a point light, are placed on the left side of the wall.

a) **[2.0v]** Which objects should be reflected and how do you calculate them?

All the objects, including the light and excluding the mirror wall.
Use the scale S(-1, 1, 1)

b) **[2.0v]** How do you shade the reflected geometry and avoid that it can appear at places where there is no wall?
By using the blending mechanism for shading and the stencil to restrict the drawing to the wall. The students should detail both mechanisms