# Animação e Visualização Tridimensional

Mestrado em Engenharia Informática e de Computadores
*Alameda*

**2nd Mini-Test**
30th November 2016

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

**a)** **[2.0v]** The code sample above does not render multitextured objects. Justify this by referring what texels are used to shade both Object1 and Object2.

Object 1 with steel.bmp; object 2 with wood.bmp

**b)** **[2.0v]** Adapt the code in order to allow multitexturing.

Add :

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

and before drawing an object, for instance

glUniform1i(tex_loc, 0);

glUniform1i(tex_loc1, 1);

drawObject1();

**2.** You want to map a 256x128 texture image into a rectangle whose texture coordinates are: {(0, 0), (0, 2), (2, 2), (2, 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)** **[2.0v]** How many times will be the texture tiled in the rectangle? Why?

GL_REPEAT in each direction, so 2 x 2= 4 tiles

**b)** **[2.0v]** In what situation, the filter of the last command will be used? Is it a point or an area sampling technique? Why?

Magnification. Area sampling: to choose the nearest 2x2 texels, average them and in the pixel

**c)** **[3.0v]** Consider that a LookAt() call made the rectangle to be projected in the screen with the size of 16x8 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:
256x128 (nível 0), 128x64 (nível 1); 64x32 (nível 2); 32x16 (nível 3);
16x8 (nível 4); 8x4 (nível 5); 4x2 (nível 6), 2x1 (nível 7), 1x1 (nível 8)

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 (256/126; 128/8), ou seja 16. Significa deste modo que $\lambda = \log_2 \rho = \log_2 16 = 4$: portanto é selecionado o mapa de nível 4: 16x8

**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. 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.

**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?

Since the flare is rendered in last place, with z-buffer off, it will appear always in FRONT of the scene.

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

Elements of a flare are rendered along a line from the projected position of the light to a point opposite it across the center of the screen.

Effect rendered with a small collection of textures: circles, rings, hexagons, sunbursts, so on.

**c) [1.0v]** The lens flare effect must animate convincingly with the camera movement. Describe such animation by referring the size and opacity of each flare element.

Light is farther from the center, the elements are smaller and more transparent; when closer, the elements become larger and more opaque.