TÉCNICO LISBOA

Animação e Visualização Tridimensional

Mestrado em Engenharia Informática e de Computadores
Alameda

2nd Mini-Test
29th November 2017

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], "orange.tga");
LoadTexture(TextureArray[1], "steel.tga");
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.

There is no multitexturing since there is only one glsl uniform sampler2D variable, the texmap variable which is accessed in the OGL program through the variable tex_loc

The Object 1 is textured with steel.tga; and object 2 with orange.tga

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

For instance, Create another glsl uniform sampler2D variable, the texmap1 variable which is accessed in the OGL program through the variable tex_loc1

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

and before drawing an object:

glUniform1i(tex_loc, 0);

glUniform1i(tex_loc1, 1);

drawObject1();

Aluno: _________ ____

2. You want to map a 256x128 texture image into a rectangle whose texture coordinates are: {(0, 0), (0, 3), (3, 3), (3, 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 3 x 3= 9 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: GL_LINEAR means to choose the nearest 2x2 texels, average them and allocate the resulting color to 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)

With the option GL_LINEAR_MIPMAP_NEAREST, OGL just select one mipmap and then performs an area sampling 2x2 to calculate the color to shade the fragments.

OGL select the right mipmap by calculating ρ = max (256/126; 128/8), or 16.
Then calculates $\lambda = \log_2 \rho = \log_2 16 = 4$:
Thus level 4 mipmap is chosen which has 16x8 resolution

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) **[2.0v]** 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. In the first step, the fragments with alpha != 1 would be discarded; in the second step the fragments with alpha ==1 would be discarded.

b) **[2.0v]** In the second step, describe how the OpenGL depth-test should be configured?

The Z-buffer test remains enabled but the writing is disabled (gl_depth_mask(GL_FALSE) in order to guarantee that the z-buffer only stores the information of opaque objects. This way, translucent objects behind opaque objects will be not drawn

4. You want to implement planar reflections on a mirror floor placed at y = 0. All the other objects of the scene are placed above the floor including a point light.

a) **[1.5v]** Which objects should be reflected and how do you calculate them?
All the geometric objects except the floor should be reflected by using a YY' symmetric scale transformation S(1, -1, 1). The light source should be also reflected.

b) **[1.0v]** Before rendering the reflected geometry, explain why is it necessary to set up properly the backface culling feature?

Because the scale transformation changes the orientation of vertices order of the polygons back_faces will become front faces and vice-versa

c) **[1.5v]** How to avoid that reflected geometry can appear at places where there is no floor?

By using the stencil buffer. Explain it

d) **[1.0v]** Is it necessary to use the OpenGL blending mechanism? Why?

Yes. The floor should be translucent and blended (GL_SRC_ALPHA, ONE_MINUS_SRC_ALPHA) with the reflected geometry already rendered in order to cause the illusion of reflection in the specular floor.

Aluno: ________ ____