INF2110 - Introduction à l'Informatique Graphique **M1 Informatique**
TP9 **Fall 2018**
Prof. Caroline Larboulette

Texture Mapping

What you need to turn in at the end of the lab session
At the end of the lab session, you should upload a .zip file containing the source code .cpp (+ eventually other classes you have used) of each exercice as well as a .pdf shortly describing what you have done for each question with screenshots to illustrate the results you have obtained. Remember the assignment may be graded. ***Read the entire document before you start coding***.

Program using the *libqglviewer* library. Start from TP6 or TP8 (you need to be able to load and draw a polygonal model). Use a model that contains texture information (lines *vt* in the .obj file). To create an .obj file with texture information, use a software such as Autodesk Maya.

1. Load texture UV coordinates and image file

Load the UV coordinates for each vertex of your object and store them in appropriate data structures. Load the image corresponding to the texture using the **`QImage`** class of the Qt Library (see online documentation and lecture for details). An alternative could be to generate / compute the texture coordinates using a projection.

2. Create a texture object

Create a texture object for this texture. This needs to be done only once (i.e., in the *`init()`* function).

1. Use `void` **`glGenTextures`**`(GLsizei` *`n`*`, GLuint *`*`textures`*`)` to generate texture IDs. **`glGenTextures`** returns *`n`* texture names in *`textures`*. Those names are integers, you need one per texture file.
Before you define texture parameters or use the texture to draw your objects, you need to enable texturing with **`glEnable`**`(GL_TEXTURE_2D)` and select the texture you are working on with `void` **`glBindTexture`**`(GLenum` *`target`*`, GLuint` *`texture`*`)`.

2. Specify the texture parameters to define how the texture is used with `void` **`glTexParameterf`**`(GLenum` *`target`*`, GLenum` *`pname`*`, GLfloat` *`param`*`)` and how the parameters are interpreted with `void` **`glHint`**`(GLenum` *`target`*`, GLenum` *`mode`*`)`. Indicate how the texture is applied to each pixel by specifying a texture environment with `void` **`glTexEnvf`**`(GLenum` *`target`*`, GLenum` *`pname`*`, GLfloat` *`param`*`)`. Use different values for the parameters to see their effect.

3. To specify an image for a texture, use the `void` **`glTexImage2D`**`(GLenum` *`target`*`, GLint` *`level`*`, GLint` *`internalFormat`*`, GLsizei` *`width`*`, GLsizei` *`height`*`, GLint` *`border`*`, GLenum` *`format`*`, GLenum` *`type`*`, const GLvoid *`*`pixels`*`)`. The *`*pixels`* can be obtained from the **`QImage`** object.

3. Apply the texture to your object

Before you start drawing, you need to enable texturing and bind the texture you want to use for this object:

```
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture_name);
```

Use `void` **`glTexCoord2f`**`(GLfloat s, GLfloat t)` function in `draw()` to specify the UV coordinates of each vertex.
Use `glDisable(GL_TEXTURE_2D)` to disable texturing, for drawing other objects for example.

4. Additional models

Now that it is functional for a single object, load additional objects with different textures.