INF2110 - Introduction à l'Informatique Graphique **M1 Informatique**
TP3 **Fall 2019**
Prof. Caroline Larboulette

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 .pdf shortly describing what you have done for each question : source code extractions with associated screenshots to illustrate the results you have obtained. Remember the assignment may be graded.

For this lab, first download the file tp3_squelette.cpp from the lecture's webpage on Moodle.
Have a look at the code and the comments.

1. Drawing and Filling Polygons

The command `glBegin(GL_POLYGON)` draws filled polygons by default. However, options for drawing polygons can be specified by `glPolygonMode(GLenum face, GLenum mode)`. The parameter *face* controls which faces are drawn: `GL_FRONT_AND_BACK`, `GL_FRONT` or `GL_BACK`. The parameter *mode* indicates whether the polygon should be drawn as points, outlined or filled (`GL_POINT`, `GL_LINE`, `GL_FILL`). Using keyboard keys (see `void keyboard()`), make it possible to switch from one mode to another. Use the key binding of your choice.

2. Modeling 2D Transformations

To position, orient and scale a model, OpenGL offers three routines:
- `void glTranslate{fd}(TYPE x, TYPE y, TYPE z);`
Translates an object by the given x, y and z values.
- `void glRotate{fd}(TYPE angle, TYPE x, TYPE y, TYPE z);`
Rotates an object in the ccw direction about the ray from the origin through the point (x,y,z). As we work in 2D, what should you use for point(x,y,z) ?
- `void glScale{fd}(TYPE x, TYPE y, TYPE z);`
Scales with factors x, y and z along each direction.

Use those functions to apply transformations to an object of your choice (you may use the little house example). Try the configurations seen in class.
- What happens if you make several transformations (translate + rotate) ?
- What happens if you draw all of the transformed objects on the same drawing ? (To correct the problem, use `glLoadIdentity();` Explain why.)

Make sure you are modifying the correct **(object)** matrix anytime you make a transformation call by specifying `glMatrixMode(GL_MODELVIEW);` (the other main matrix is `glMatrixMode(GL_PROJECTION);` and deals with the camera transformations).

3. Modeling 2D Transformations using Matrices

Instead of using OpenGL transformation calls, you may directly specify the transformation matrices to apply to your objects. You may even create a matrix to represent composed transformations. Recreate the above examples in matrix form, both with several matrices and with the compound matrix. Show the detailed computations in the report for the compound matrix. *As your work is individual, you should all have different matrix sequences and compound matrices.*

```
void glLoadMatrix{fd}(const TYPE *m); //loads a matrix
void glMultMatrix{fd}(const TYPE *m); //multiplies the current matrix
with m
```

You should declare your matrices as m[16]. Be careful, in OpenGL, the rows and columns are transposed compared to the standard C++ convention !