**INF2110 - Introduction à l'Informatique Graphique** **M1 Informatique**
**TP1** **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 of exercices 1 to 3 (can be in multiple files) as well as a .pdf report describing what you have done: source code extractions with associated screenshots to illustrate the results you have obtained. Remember the assignment may be graded. Work is individual.

All the details of the OpenGL library can be found in the OpenGL Programming Guide:
https://sgar91.files.wordpress.com/2010/12/opengl-programming-guide-7e.pdf

For this lab, first download the file tp1_squelette.cpp from the lecture's webpage on Moodle.

Have a look at the code and the comments. Then, change the includes to be compatible with your platform and compile.

**1. Describing Points, Lines and Polygons**

Vertices **positions** are specified with `void glVertex{2,3,4}{sifd}[v](TYPE coords);`

Try the different drawing options:
`GL_POINTS` : individual points
`GL_LINES` : pairs of vertices interpreted as individual segments
`GL_LINE_STRIP` : series of connected line segments
`GL_LINE_LOOP` : same as above, with a segment added between last and first vertices
`GL_TRIANGLES` : triples of vertices interpreted as triangles
`GL_TRIANGLE_STRIP` : linked strip of triangles (each series of 3 points create a triangle)
`GL_TRIANGLE_FAN` : linked fan of triangles (first point is center, other points are around)
`GL_QUADS` : quadruples of vertices interpreted as four-sided polygons
`GL_QUAD_STRIP` : linked strip of quadrilaterals
`GL_POLYGON` : boundary of a simple, convex polygon

**2. Displaying Points, Lines and Polygons**

**2.1 Color**

Use `glColor*();` to change the color of primitives or individual points.
Use different colors for different vertices of a polygon. What do you observe ?

**2.2 Point Details**

Use `glPointSize(GLfloat` *`size`*`);` to control de *size* (in pixels) of a rendered point. *size* must be greater than 0.0 and is 1.0 by default (if not specified).

**2.3 Line Details**

Use `glLineWidth(GLfloat width);` to set the width in pixels for the rendered lines.
Use `glLineStipple(GLint` *`factor`*`, GLushort` *`pattern`*`);` to define a stipple pattern.
`glEnable(GL_LINE_STIPPLE);` must be enabled.
Call `glDisable(GL_LINE_STIPPLE);` to disable line stippling after your have finished drawing.

**3. Implement the Midpoint Scan Line Conversion Algorithm**

Now that you can draw lines and points, create and draw a grid, and a function to "draw a line" by coloring pixels on this grid from the specification of its end points both with integer coordinates. The implementation details are up to you !