OpenGL - Lab Session 1

First steps in OpenGL and modeling

During this lab session, we will make our first steps in OpenGL and practice on modeling objects from geometric primitives. For simplicity, we will use two standard wrappers for OpenGL called GLUT, and QGLViewer, with simple windowing and callback functionalities. (Here, a wrapper means a library whose main purpose is to increase the usability of a second library.)

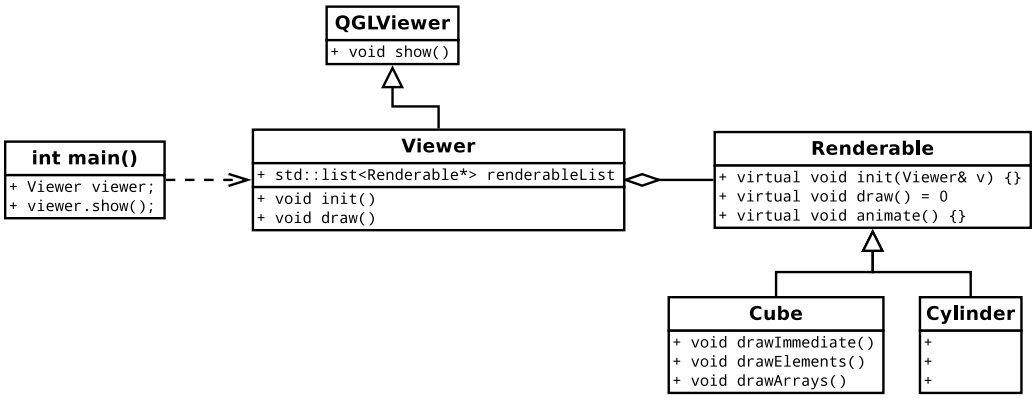
1 Introduction

To start this practical, you will need to download the following code (QGLViewer skeleton):

https://intranet.ensimag.fr/KIOSK/Matieres/4MMG3D/Practicals/TP1.tgz

Have a look at the skeleton and Makefile to understand the basic functionalities and build options of QGLViewer. You don't need to precisely understand all of the code and initialization steps for now.

The code organization is illustrated by the following class diagram



which can be read by saying that the `main` function implements an object called `viewer` of the class `Viewer`, which derives from the class `QGLViewer`. Thus `viewer` contains the function `show()`. The `Viewer` class also contains several objects of the class `Renderable`, stored in the attribute `renderableList`, which is of type `std::list`.

The class `Renderable` has three `virtual` functions, which means that these functions aim to be reimplemented by the classes derived from `Renderable`, such as the two classes `Cube` or `Cylinder`. The mark "`= 0`" put after the `virtual` function `draw` means that any derived class **must** reimplement this function.

In the remaining of this practical, you will need to understand the several drawing functions of the class `Cube`. You will then use your understanding to implement the class `Cylinder`.

2 Code analysis

Open the files `cube.h` and `cube.cpp`. These two files implement the class `Cube` which aims to draw a cube with three different methods. The following of this section enumerate these method. For the next section, you will need to understand each of them to be able to reproduce them.

The three methods to observe are:

- `void drawImmediate()`
- `void drawElements()`

- `void drawArrays()`

In all practicals, we will be using the graphical interface library Qt, version 4, upon which QGLViewer is based. Qt comes with its own build system called `qmake`. To invoke the Qt4 build system and compile the code on Ensimag machines, type the following commands on Unix shell prompt:

```
qmake-qt4 cg3D.pro
make
```

The first command create a Makefile from the .pro file. The second command execute the Makefile to compile the code. If you are developing on an external platform and Qt4 is the only version of Qt installed, then you may type `qmake` instead of `qmake-qt4` as Qt4 will be your default Qt version in this case.

3 Cylinder drawing

From the observation of the class `Cube`, create the files `cylinder.h` and `cylinder.cpp` to implement a class `Cylinder` that performs the drawing of a cylinder. As for the cube, try to implement several methods, and compare their efficiency.

For the compilation, you will need to add your 2 files in the `HEADERS` and `SOURCES` file lists in the configuration file `cg3D.pro`, and then execute the compilation commands above.

4 To go further

If you have time, you can implement other classes to draw primitives from the library `glut`.