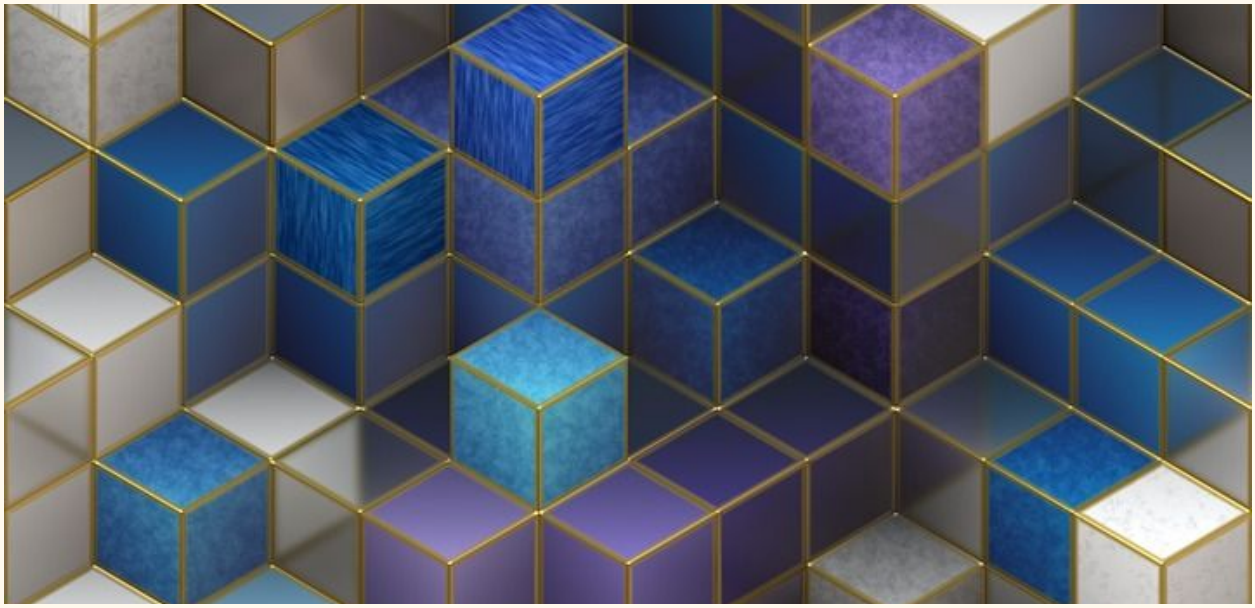
SCHOOL OF COMPUTING SCIENCE AND DIGITAL MULTIMEDIA

3D COMPUTER GRAPHICS DEVELOPMENT

Anthony Sébert



ABSTRACT

This report gives an overview of the project completion & prospects. The subject was the creation of a simple chess game in C++ using OpenGL. It contains references to the documentation about the languages and technologies included, the code hierarchy, and the problems encountered.

Keywords

C++, OpenGL, GLUT, JSON, 2D graphics, chess game

INTRODUCTION

The application detailed here has been developed as part of the module CM3115. It has been written in C++11, and use the following libraries: *freeGLUT*, *GLEW* and *JSON for Modern C++*.

The chosen IDE is *Visual Studio 2017 Enterprise Edition*, with the version control system tool *Git*, and the online storage service *GitHub*. A Visual Studio add-on bind all these elements to work together.

The purpose of the application is to create a simple chess game.

LITERATURE REVIEW

The sources employed to research information about the subject and the libraries are exclusively composed of online resources.

C++ resources

https://en.wikibooks.org/wiki/Optimizing_C%2B%2B

https://en.wikibooks.org/wiki/C%2B%2B_Programming

http://www.cplusplus.com/

http://en.cppreference.com/w/

OpenGL/GLUT

https://en.wikibooks.org/wiki/OpenGL_Programming

https://www.khronos.org/registry/OpenGL-Refpages/gl4/

http://freeglut.sourceforge.net/docs/api.php

http://campusmoodle.rgu.ac.uk/course/view.php?id=91125

JSON for Modern C++

https://json.readthedocs.io/en/latest/

JSON

https://json.org/

RESULTS

The program can be divided in three sections, corresponding to the MVC design pattern. For a global overview, see the diagram below.

Model

The Model section encapsulates the data: board cells and pieces. All the pieces inherit from the abstract class Piece, which contains purely virtual functions.

The pieces are contained in a Army object, in order to provide a simplified way to handle multiple Piece objects (Mediator design pattern).

Controller

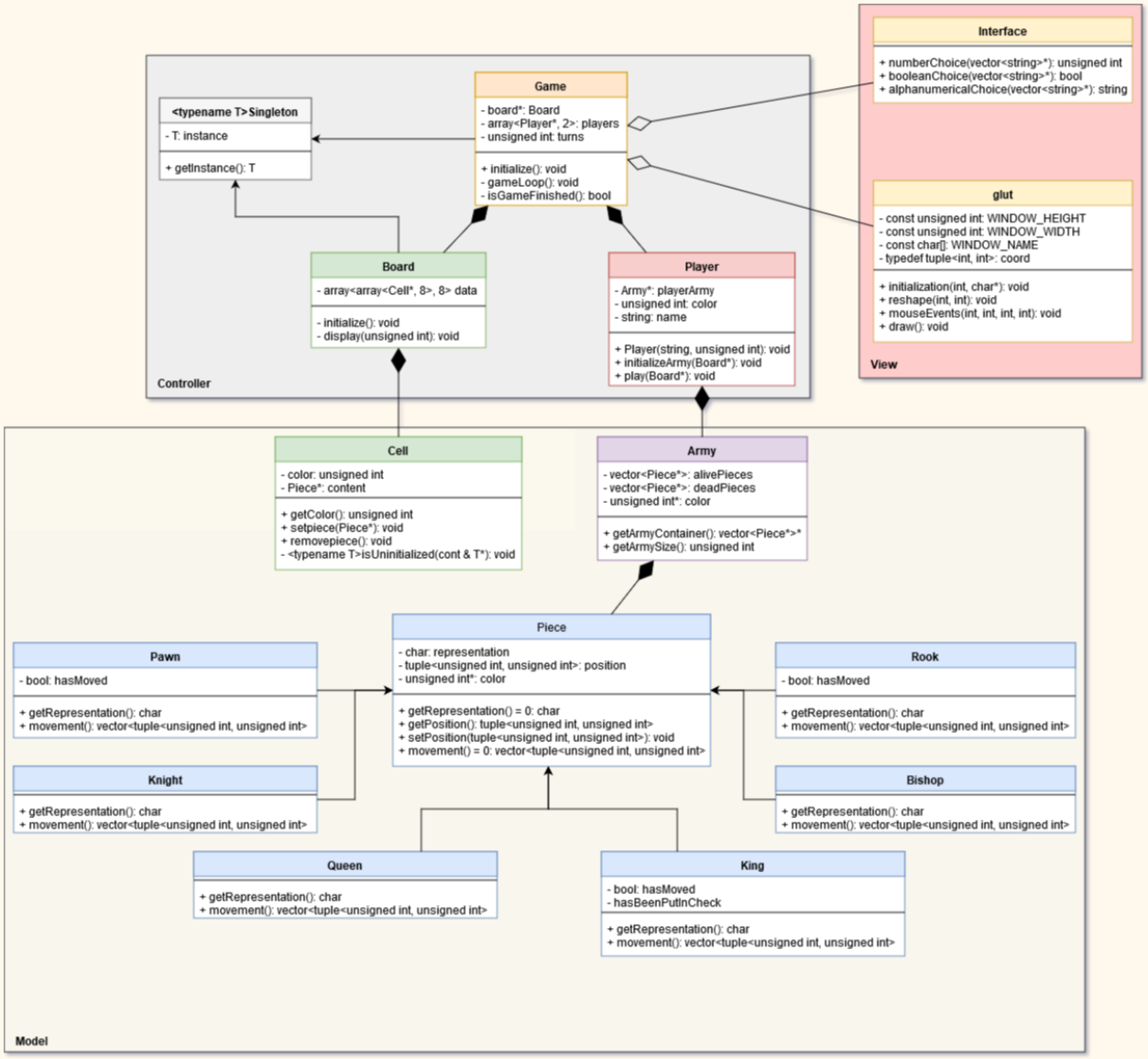
The Controller section, in grey in the diagram, ensure the creation of the following objects: game, players, board, pieces (contained in armies), and manage the graphic interface, the game progress, and the victory condition.

It also contains a Singleton class, which allow any derived class to be a singleton.

The import/export to JSON features are also available from there.

View

The view section manage the user input/output. The class Interface provide a console interface, used for test purposes. The glut files, which are not part of an object due to the *GLUT* library limitations, provide drawing functions and events handlers for the graphic interface.



Objects and their relationships (smart pointers are written as raw pointers)

DISCUSSION

The *OpenGL* API, and therefore the *freeGLUT* library, is based on callbacks functions, which the prototype is determined and only accept a corresponding function. This results a structural rigidity, that for example prevent to encapsulate dedicated functions in an object (the prototype is *void(object::*callback)()* instead of *void(*callback)()*).

It is also provided large range of unnecessary functions that only differs regarding to the parameter (*glColor* derived classes for example), bringing a certain confusion.

The documentation is often in plain text and poorly structured, and only concerns small programs, and does not talk about integration in a larger project.

CONCLUSION

Recommandations

An axis of progression would be the implementation of the replay of a game, the coloration of the cells where it is possible to move, and the drag & drop feature.

Also, graphic equivalents to the pre-game menus are planned.

Moreover, it is apparently possible to use the SDL with OpenGL. This library is more easy to use, and allow more coding liberties, even if the 3D support seems to be a little less efficient.