**UIS Software Engineering Capstone**

**Tetris**

Software Design Document

Name (s):

Francisco Hernandez

Johnathan Tielebein

Zachary Trujillo

Date: (05/02/2018)

**TABLE OF CONTENTS**

[**1. INTRODUCTION 2**](#_Toc4987)

[1.1 Purpose 2](#_Toc4988)

[1.2 Scope 2](#_Toc4989)

[1.3 Overview 2](#_Toc4990)

[**2. SYSTEM OVERVIEW 2**](#_Toc4993)

[**3. SYSTEM ARCHITECTURE 4**](#_Toc4994)

[**4. DATA DESIGN 8**](#_Toc4998)

**INTRODUCTION**

**Purpose**

The purpose of this SSD is to explain the general design and architecture of the Tetris application developed. It’s intended audience is open to everyone that wants to get familiar with the software.

**Scope**

The application developed is the Tetris game with some modified features with respect to shapes, score, difficulty, and size of board.

**Overview**

The document will go over the development process and the architecture of the application. It will contain everything necessary for a user to play it and for a developer to modify code to it.

**SYSTEM OVERVIEW**

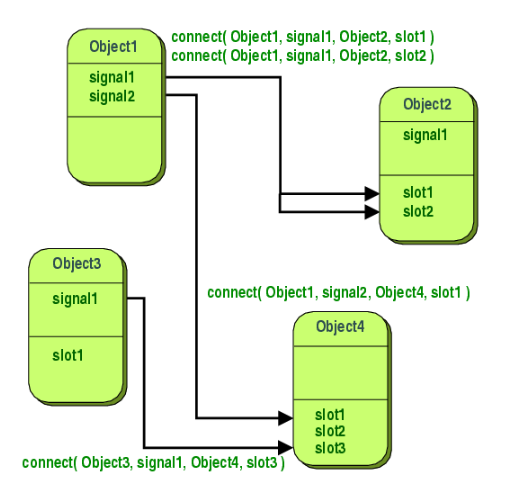
The general functionality is very similar to the classic Tetris game with a few enhanced features:

1. The dimensions of the board are different compared to the original Tetris.
2. Moving down speed of the shape changes depending on difficulty.
3. Pieces generated change depending on the difficulty.
4. Score multipliers take effect as the difficulty increases.

It is implemented using the Qt framework using QML for the front-end and C++ for the backend. Qt is cross-platform software framework with ready-made UI elements, C++ libraries, and a complete integrated development environment.

QML is a declarative language that allows user interfaces to be described in terms of their visual components and how they interact and relate with one another. It is a highly readable language that was designed to enable components to be interconnected in a dynamic manner, and it allows components to be easily reused and customized within a user interface. Using the QtQuick module, designers and developers can easily build fluid animated user interfaces in QML and have the option of connecting these user interfaces to any back-end C++ libraries.

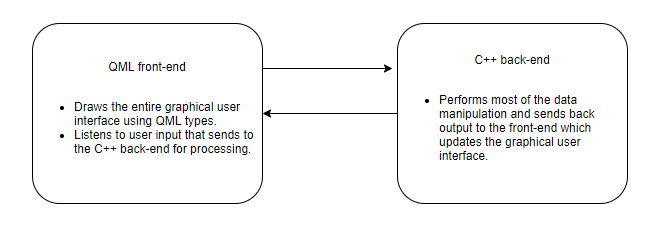
Signals and slots are used heavily in the Qt framework which are basically their own version o callbacks. Signals and slots are used for communication between objects. The signals and slots mechanism is a central feature of Qt and probably the part that differs most from the features provided by other frameworks. A signal is emitted when a particular event occurs. A slot is a function that is called in response to a particular signal.



Q Properties are used to pass information from QML to C++ and vice versa. Qt provides a sophisticated property system similar to the ones supplied by some compiler vendors. However, as a compiler- and platform-independent library, Qt does not rely on non-standard compiler features like \_\_property or [property]. The Qt solution works with any standard C++ compiler on every platform Qt supports. It is based on the Meta-Object System that also provides inter-object communication via signals and slots.

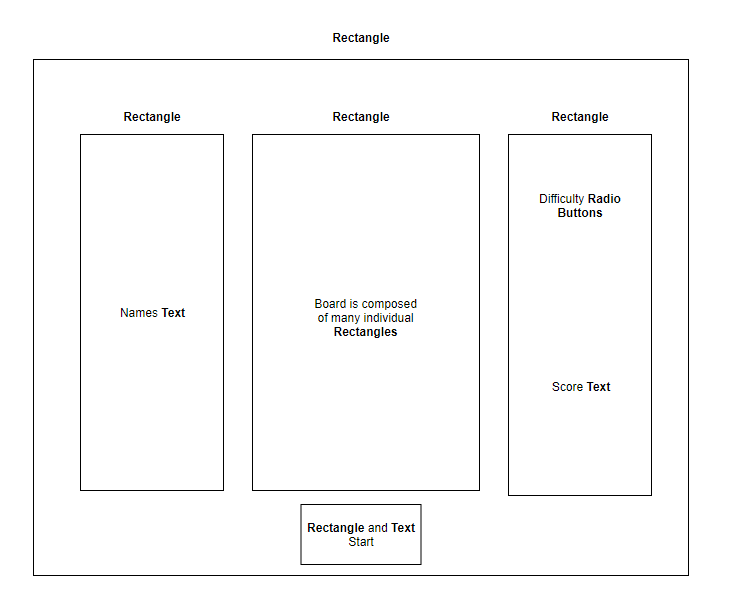
**SYSTEM ARCHITECTURE**

The high-level architecture of the software is divided into a QML front-end and a C++ back-end.



**QML Front-end**

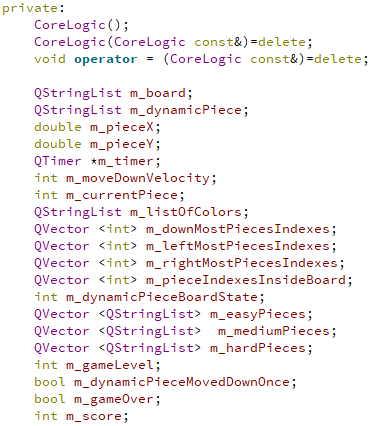
The front-end is entirely build with QML types:



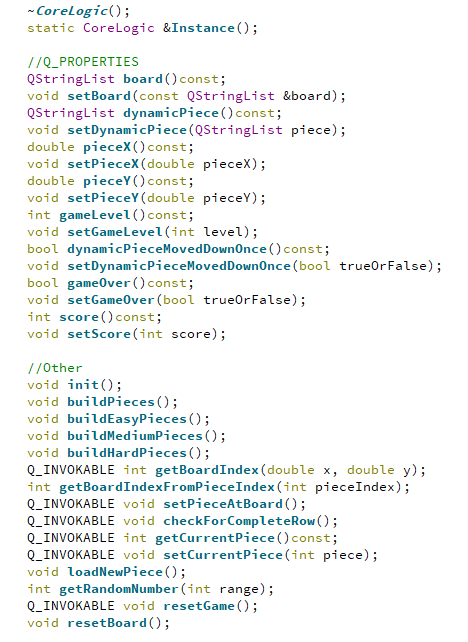
**C++ Back-end**

The back-end is built with C++, having one singleton class. There was no need for sub classing:

Private member variables:



Public methods:



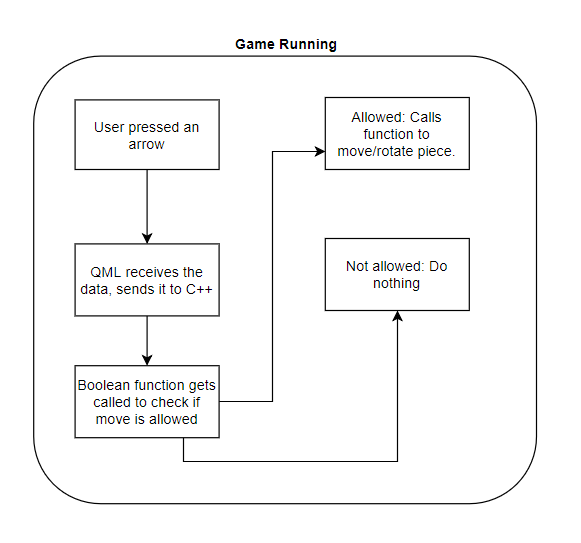


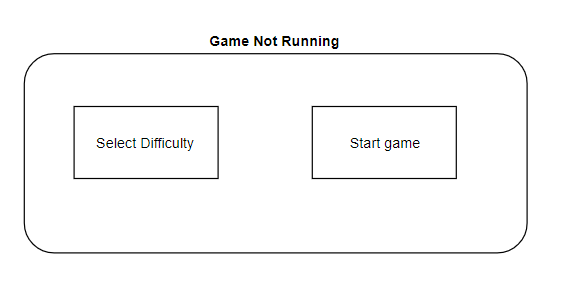
**DATA DESIGN**

Data can be divided into two categories. Data that comes from user input and data that the application itself generates for its own use.

The data that the application generates are the board, the shapes, and everything that has some effect on them. Data that comes from user input is simply a key press (up, down, left, right) and a mouse click (to select difficulty and start/pause the game).

When the game is running, the application listens to keys. When the application receives a key pressed, it calls a series of functions to see what needs to be performed that can be resumed to first check if the move is allowed with a boolean function and then call another function that performs the action if the previous function returns true.





.