**PROGRAM DESIGN 2**

Name : 皮宜恩

Student ID : F74027036

Department : CSIE 106 甲

Assignment : Project 2

Environment : Qt 5.4 for Windows 8

Execution :

make

./pd2-2048

**DESCRIPTION**

This Program has been written for educational propose. The game name is 2048. It is perhaps one of the most addictive mathematical puzzle games. For starters, it is played on a 4 × 4 game board which begins with an initial configuration of two tiles, of value 2 or 4, placed at arbitrary locations on the grid.

**HOW TO PLAY**

The goal of the game is to combine equally-valued tiles to reach the 2048 tile without getting stuck. To do so, the player can select to move Up, Down, Left, or Right. Each move shifts all tiles on the grid in the direction chosen. If two adjacent tiles have the same value, they combine, and the single resulting tile after the combination will be the sum (double). Following the player’s move, a tile of value 2 or 4 will be generated at a random position on the board. Sure, the principal objective is to get as much point as possible

**PREPARATION**

To be able to write this program, I first start playing 2048 game to be more familiar with the game playing mode, However I have to tell you that I get addicted. hahaha … Well, Since the requirement was to program a GUI game, I have spent hours and hours , day and days reading QT documentation and practice some simple example providing by the QT teams. At the end I am so impressed by QT framework, I think QT is a really powerful one, and I plan to use more in the future. Furthermore, my real work began by the moment I was thinking about the game algorithm, I have try so many different one, but I keep getting stuck. Until, I mix up some idea finding online with my own logic to create that awesome work, I provide to you today.

**DESIGN**

I create the Game interface on QT using QML module, I clone the exact interface with the original one and also I add some additional option to get extra point. I will present those extra functions in the next section. Since, the class is c++ course, of course all the implementation is written in c++. To let the QML module interact with my c++ Object Class, I costume a library in main.cpp with my c++ Object in I import this Library in main.qml. By this technique is super easy to use both Module as if they were just one. All the methods and attributes that I use are include in the UML diagram section.

**MY PERSONAL POINT OF VIEW**

I enjoy every moment spending coding this game, I think it’s a really awesome project, and thanks to this project, now I have a so much better understanding of c++ language.

|  |
| --- |
| **GAMEPLAY** |
| - scoreVar : integer  - bestScoreVar : integer  - stepVar : integer  - prevIndex : integer  - nextIndex : integer  - totalStepVar : Boolean  - addPoint : Boolean  - movePoint : Boolean  - nouPanel : typedef <int>  - m\_number : nouPanel  - m\_index : nouPanel  - m\_state : nouPanel |
| <<constructor >> +GAMEPLAY ( \*parent : Qobject))  + getScore() const : integer  + getBestScore() const : integer  + getStep() const : integer  + getTotalStep() const : integer  + showBoard(&index :integer : integer  + undo() : void  + finishGame() : void  + restartGame() : void  - initBoard() : void  - addNumber(direction: Move\_Direction) : void  - moveNumber(direction: Move\_Direction) : void  - refreshBoard(fresh :Boolean) : void  + startUpFunction() : void Q\_INVOKABLE  + moveDIr(direction: Move\_Direction) : void Q\_INVOKABLE  + color(&index : integer constant0 : Qcolor Q\_INVOKABLE  + numcolor(&index : integer constant0 : Qcolor Q\_INVOKABLE |

**UML DIAGRAM**

|  |
| --- |
| SETTINGS |
| * \*settings\_ : QSettings * appVersion : QString |
| + setValue (&key : Qstring const , &value : Qvariant const) : void Q\_INVOKABLE  +value (&key : Qstring const , &defaultValue : Qvariant const) : Qvariant Q\_INVOKABLE  + getVersion() : QString Q\_INVOKABLE  + setVersion (version : QsSring const) : void  + contains ( &key :QString const) : boolean |