Project2 report by F74046420 陳佑瑜

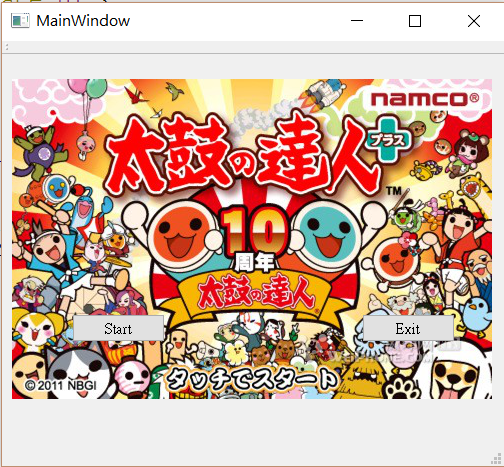
UML class diagram:

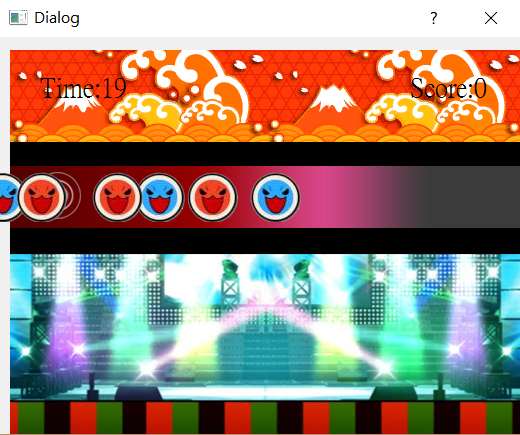
|  |
| --- |
| MainWindow |
| -gamewindow: game\*  -ui: Ui::MainWindow\* |
| -start(): |

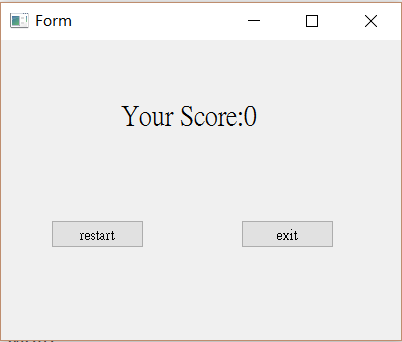
|  |
| --- |
| Game |
| +setxpos[16]:integer  +score:integer  -ui:Ui::game\*  -pretime:integer  -resultWindow:Result\*  -newgamewindow:game\*  -music:QMediaPlayer\* |
| +keyPressEvent(event: \*QKeyEvent)  +present()  -movement()  -countdown()  -show\_result()  -resetGame() |

|  |
| --- |
| Result |
| -ui:Ui::Result\* |
| +setscore(number:integer)  +restart()  +reset() |

Screen shot:







How to play:

Start the game with the start button and exit the game with the exit button.

When the drums move to grey position, press “k” to eliminate the blue drum and

press “j” to eliminate the red drum. You can get 1 point for eliminating the blue drum, and get 2 point for the red one. When it is time up(30sec), it will show you the total score you get and you can choose to restart or exit the program.

Program Architecture:

To build my first little game application, I mainly use the powerful UI design in Qt. Connect three different screens with signal and slots, in other words, create 3 different UI and to connect them with button and signal. To randomly create drums, I set up several labels in the invisible place of screen, and randomly set their position.

Each 25 milliseconds I decrease 5 from their current position. So in this tiny period of time, you can see the drums “seem to move”. To restart the program, the most difficult part I think, I designed that when you click the restart button, it will emit a reset signal. And if the signal&slots receive that signal, it will delete the result UI and recreate the game UI(game window). So this is how I make this program and how it works. Hope you will like it :) .