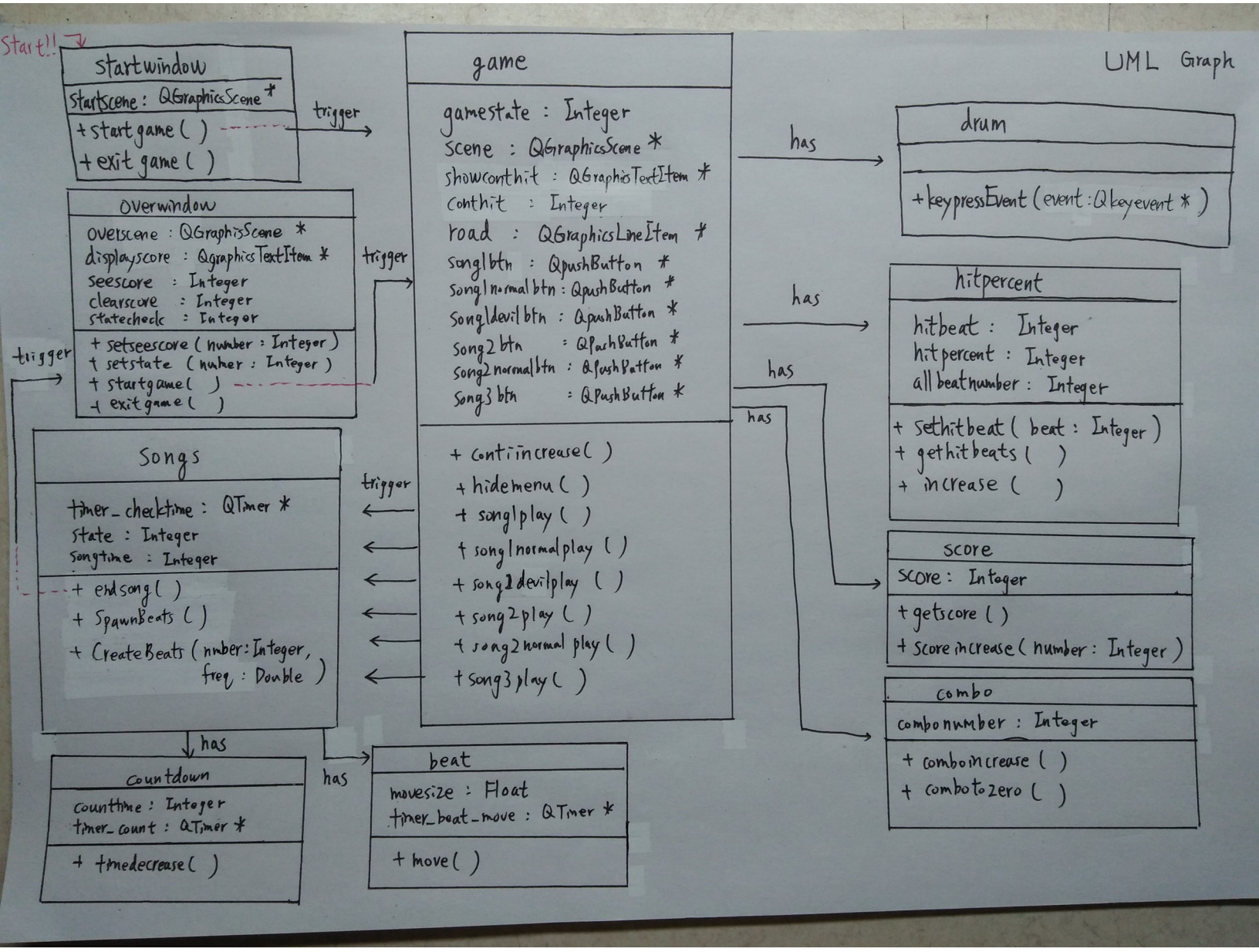
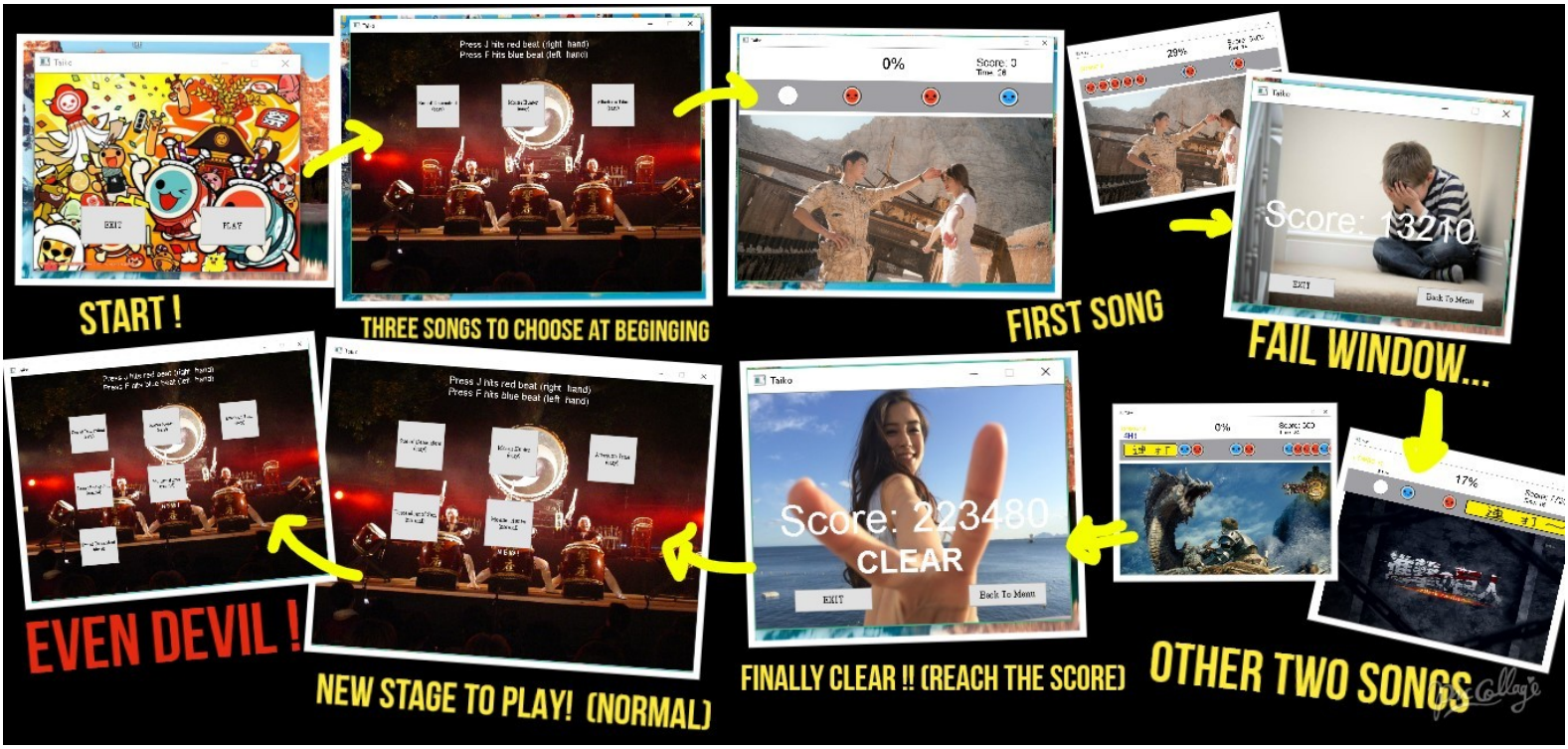
Taiko Report

F44044066 航太 108 龔俊瑋

1. UML Graph :



2. Gameplay screenshot & How to play



Start with three basic songs (easy stage).
Red beat, Blue beat and continuous-hit beat to hit.
Right corner shows your score and time left. Hit at the right timing, delete the beats, and get scores.
If you are good enough, hit at the perfect timing, a nice-hit is made, much score added!
And be sure to keep the combo number. Do not break it, or you might get failed.
Once success. New stages pops out ! Normal stage to play and even devil one may be available!

3. program architecture

First, I make a drum class, which can track on the time F or J key are hit.
And it needs to have information of the colliding items, in order to cancel the beats.

So next, I make a beat class, which can move horizontally.

Then, I come to the gamewindow.
For the gamewindow, I make a countdown class for showing the game time.
A score class which displays scores. A hitpercent class, showing hit percent.
And several buttons for choosing songs.

Last. Instead of spawning beats in the gamewindow directly, I decide to make “Song Classes”,which contains all the information about the song.
By doing this, I can easily design numerous songs, making each songs independent, so that while making this game larger, each songs can be designed by different groups and added to game once finished.

For the Song class
I've tried various ways to spawn the beats.
-Spawn the beats with limited random position
-Display beats by complicated algorithm with multiple functions and slots
-Create beats precisely by controlling the time they should be create

For the first approach, It is the simplest way, and it perfectly meets this project's requirement. However, because the beats are spawned in random, the song then has no tempo and makes the game far away from a tempo-hitting game.

For the second approach. Yet it really gives the song right tempo. The whole process is too complicated and it takes too much time and testing. Besides, after testing for many turns, this approach has a serious problem that “It is not steady”. Sometimes, it will lost the tempo, making the beats either too early or lag behind the rhythm, which I still don't have an adequate answer.

For the third approach. It's done by using Qtimer::singleshot function. The advantage of this approach is that it can place the beat whenever timing you want, the algorithm of this approach is much simple. And it provides better steadiness. It is by far the greatest method I find. Yet, it still has a steady problem that : when the song plays for a while, the preciseness of the beats-creating-time gets off. I believe the problem is result from Qtimer::singleshot function. Therefore, modification of this approach is needed. And, brand new approach should still be tried.