1942 Plane War Game

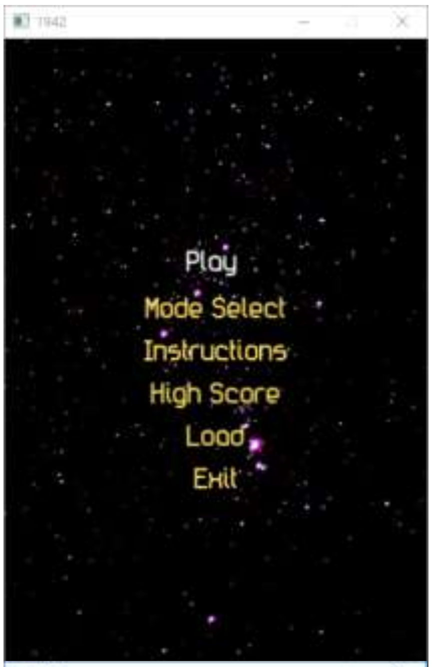
Introduction

This is our college's project that we need to finish one of the games provided by the teacher. I choose to accomplish a "Plane War" game named "1942".

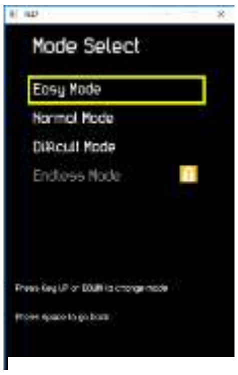
Because my project is not very complex, I choose the lightweight Framework HGE (http://kvakvs.github.io/hge/) recommended by my teacher.

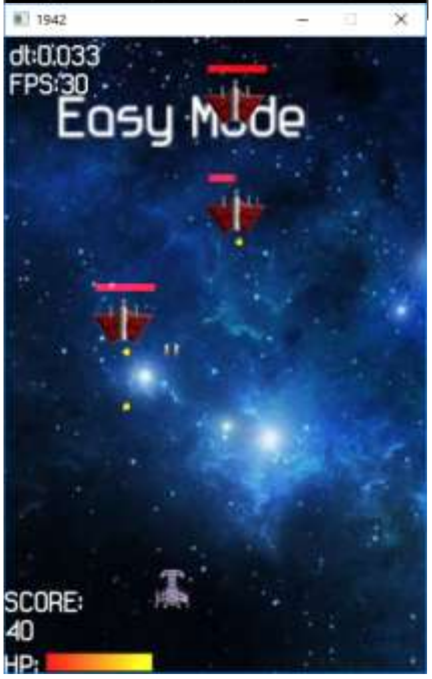
In this project, we need to design our own class design with proper hierarchy and inheritance. For instance, we need to isolate logic code from drawing code by ourselves.

Instruction



The first scene players see will be the main menu. The player can start a game, select mode, view the instructions and watch the record of his last time of play.



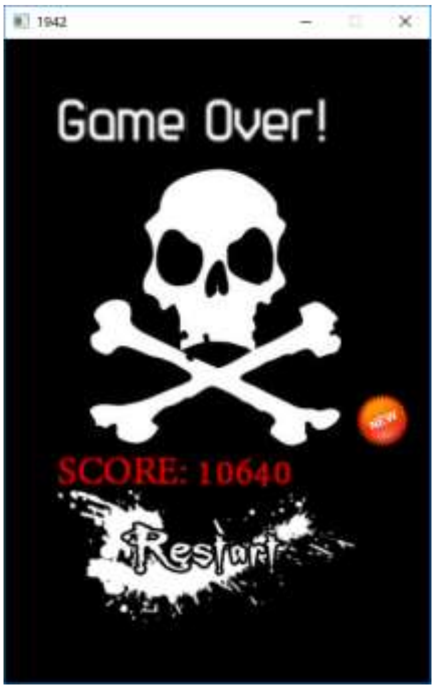


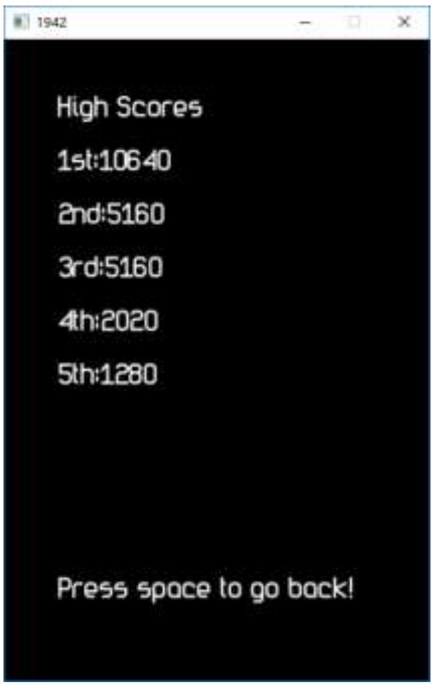
After starting the game, the player can use direction keys "↑↓←→" to move and "space" key to attack.

At the bottom, there are indicators of score and HP. If the player's plane is attacked by enemy or its bullet, the HP will get down. The game will come to an end if the HP is lower or equal to zero.

When an enemy plane crashes, there may be a random remain left. It may be a heart, which can recover your "HP", a green "P", which can protect you against the bullet of enemy and itself, a yellow "C", which can make all enemies on the screen crash and a poison, whose effect equals the damage brought by the bullet.

For further instructions, players can go to the instruction



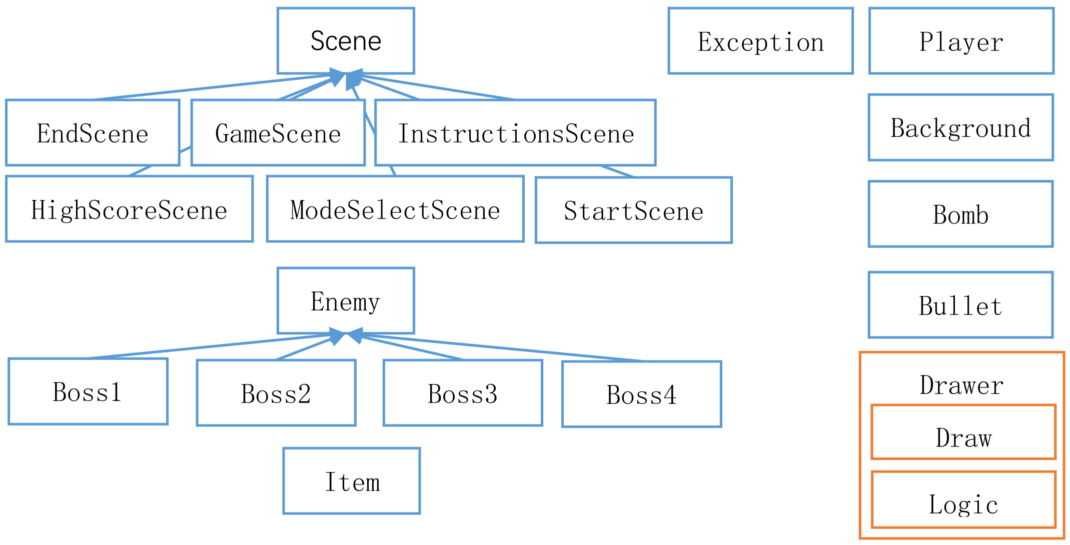


High Scores
1st:10640
2nd:5160
3rd:5160
4th:2020
5th:1280
Press space to go back!

Difficulty

1. To make the game scalable, I design the base class plane and other planes are inherited from it. If I want to design some planes in future, I can just write a new class inherited from the base class. So as other things, such as scenes.
2. When implementing the "save/load" function(the player can press "Load" to see the record of last time), I originately record each keyboard activity of the player with time stamp, however, it has lots of little offset caused by round-off of time which will finally cause the record is different from the originate scenes. Finally I choose the time stamp whose unit is clock, whose type is integer to avoid the problem.
3. To make the code readable and easy to maintain, I separate the logic code and the drawing code.

Architecture



The most basic class is the **Drawer**, which is composed by the **Draw** class which responses for drawing the elements and the **Logic** class which is used to move elements, check whether some events happen and change the scenes.
All boss inherits from the **Enemy** which shows a strong scalability considering about adding new kinds of boss. So as the scenes.
Because differents items resemble each other, I add a member varible in the **Item** class to represent the type of the item instead of a new class inherit from Item.