EIE3109 Assignment Report

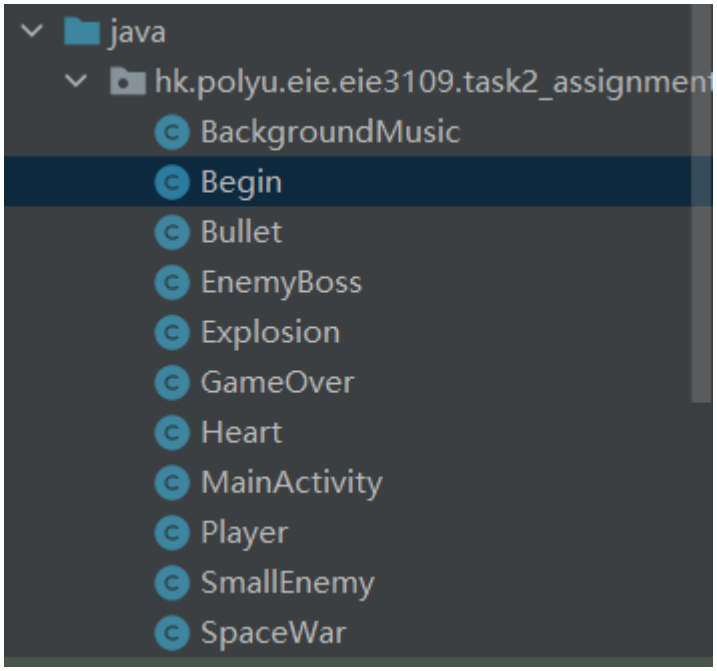
HAO Jiadong 20084595D

1. Introduction

For task 2, I design a game called Space War. In the game, the player should control a blue warcraft at the bottom of the screen by moving it to the left or right to attack the enemies to gain scores. The enemies (both the boss and the small enemies) can also attack the player so the player should make a good decision between scoring and being alive. Initially, the player has three lives. Each attack by the enemies reduces the player's lives by one. When the player has no lives left, the game ends.

2. Description

Classes Defined:



a. Activity-related classes

i. Begin: Display the initial interface. Enable the button to jump to MainActivity.

ii. MainActivity: Call out “SpaceWar” to start the game and control the start and stop of the background music.

iii. GameOver: Display the final scores. Enable the buttons to start a new game or quit the application.

iv: BackgroungMusic: Use “MediaPlayer” to start and stop the background music.

v: SpcaceWar: Implementation of the game logic.

b. Object-related classes

i. Player: Define the properties of the player (the blue warcraft at the bottom of the screen).

ii. EnemyBoss: Define the properties of the boss (the black aircraft at the top of the screen).

iii. SmallEnemy: Define the properties of the small enemies (the small grey ones).

iv. Bullet: Define the properties of bullets fired by both the player and the boss.

v. Heart: Define the properties of a game prompt “heart” which can increase the lives of the player.

vi. Explosion: Define the object to show exploding effect.

APIs Used (classified by functionalities):

a. Load resources, start an activity, and get a system service:

android.content.Context

android.os.Bundle

android.content.Intent

android.app.Activity

b. Interface related including displaying and receiving user actions:

android.view.View

android.widget.TextView

android.view.Display

android.view.MotionEvent

c. Display the sprites:

android.graphics.Bitmap

android.graphics.BitmapFactory

d. Set the painting properties:

android.graphics.Canvas

android.graphics.Color

android.graphics.Paint

android.graphics.Point

e. Specific tasks:

i. Realize the timer: android.os.Handler

ii. Play the background music : android.media.MediaPlayer

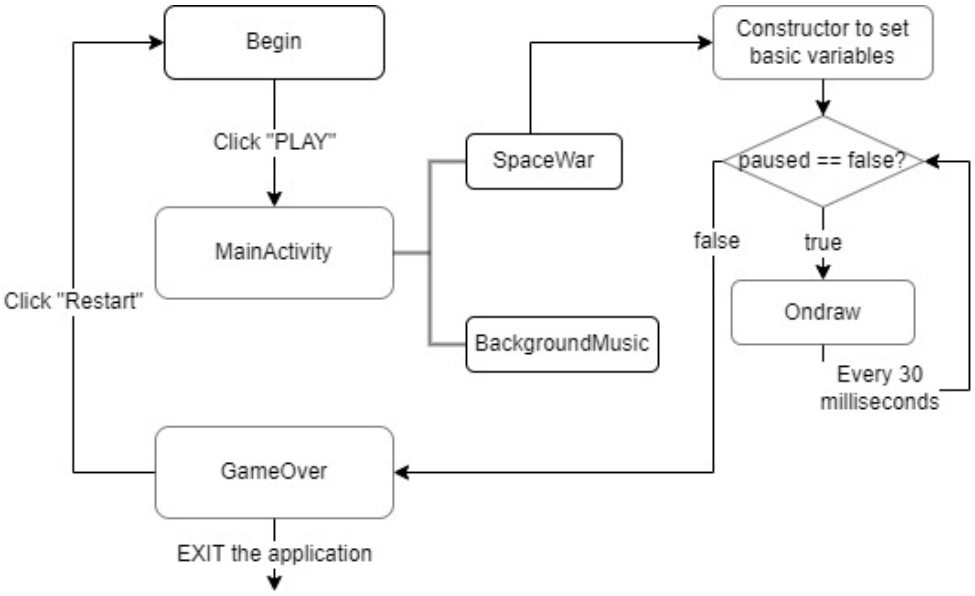
iii. Generate objects at random positions: java.util.Random

iv: Store the existing objects: java.util.ArrayList

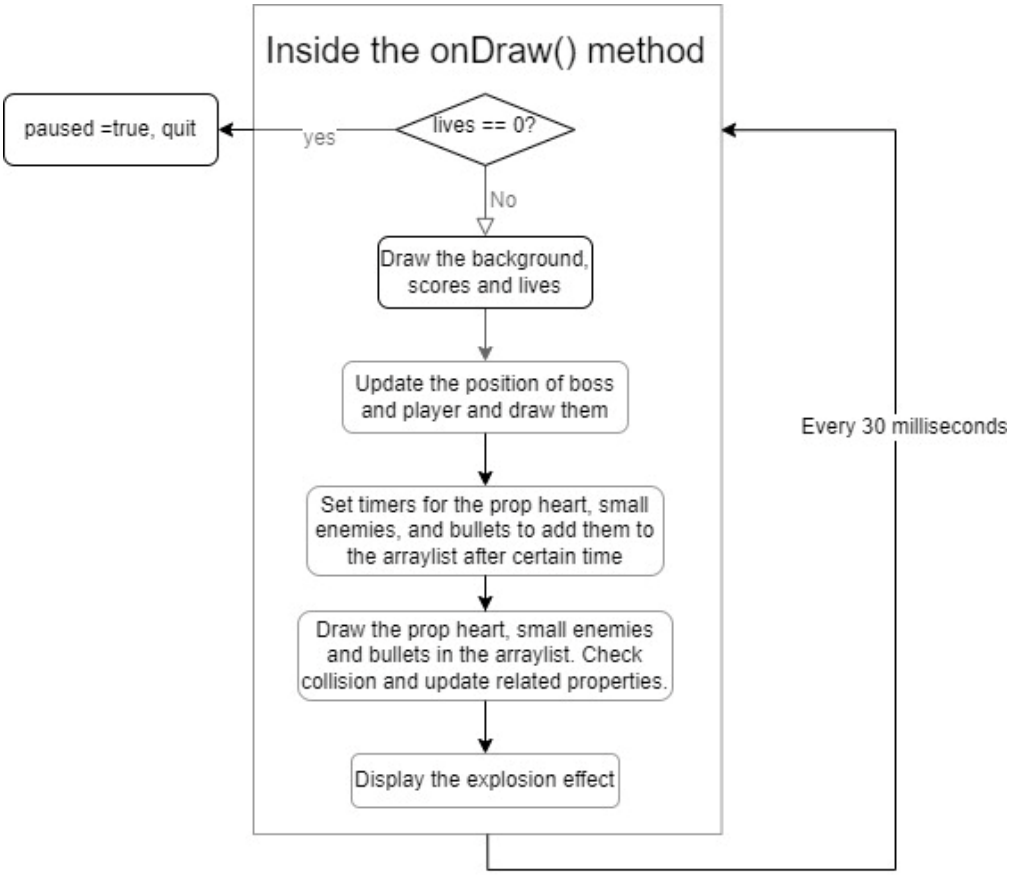
3. Methodology

Program flows:

a. Overall program flow:



b. Program flow inside onDraw():



Features：

a. **Use vivid sprites for animation**. The corresponding sources are cited in the reference part.

b. **Adopt collision-checking techniques to realize this game,** for example, to make sure the boss and the player can not move out of the screen. Also, we must constantly check whether the bullets or the small enemies successfully hit the target, if yes, we have to remove them and do corresponding updates.

c. **Use "android.media.MediaPlayer" to add a looping audio file** as the background music to add a unique atmosphere to the game.

d. **Use "android.os.Handler" to set timers for some objects** such as the small enemies and the prop heart so that their occurrences are under control.

e. **Use consecutive pictures to create a dynamic exploding effect** when

two objects have collisions.

Apart from the audio part, which is mostly referenced on a website cited in the reference part, all the other source codes are written by myself. The projects or tutorials listed in the reference are not directly copied, only for references of the basic framework or inspirations of certain designs.

How to Play:

a. Click the “PLAY” button in the initial interface to start the game.

b. The player: Move fingers on the screen to control the blue warcraft at the bottom of the screen to score and dodge attacks from the enemies. The warcraft will automatically shoot bullets. The current scores and remaining lives can be viewed on the top left and top right corner of the screen.

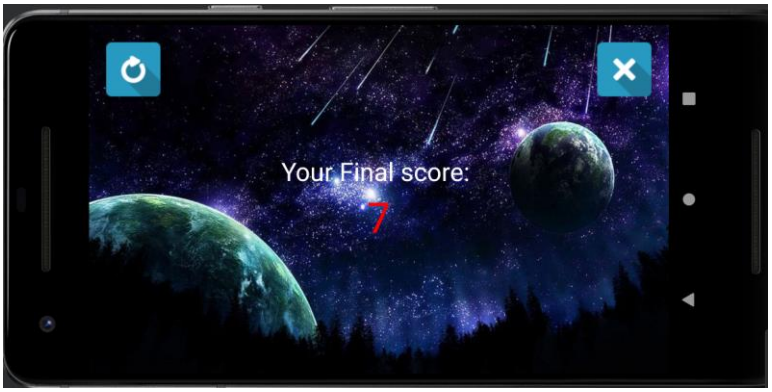
c. Lives: Initially, the player has three lives (also the upper limit value of lives). Each time the player gets attacked, the lives will be deducted by one. When the lives go to zero, the game ends. The player can obtain the prop heart randomly appearing on the screen to add the lives by one.

d. Boss: The big enemy at the top. It will move from left to right and escape from the player's attack and the speed will be faster and faster as time goes on. It will shoot bullets at the player randomly. It won't be destroyed by the player. Each time the player successfully shoots at the boss, the scores will add two.

e. Small enemies: The grey small warcraft appear randomly on the screen. They will try to crash on the player. If the player successfully shoots at it, it will be destroyed, and the score will add one. As time goes by, the fundamental velocity of small enemies will be higher and higher.

f. Each time a collision occurs(among bullet, enemies, and player), there will be an explosion effect to indicate it.

g. When the game ends, the player can view the final scores. Click the restart button at the top left corner to start a new game or click the close button at the top right corner to quit.



4. Conclusion

Problems faced and corresponding solutions:

a. The first problem is to decide when to create a new bullet. If we create a new bullet every time onDraw is invoked without any restrictions, then the entire screen would be filled with bullets because onDraw is called every 30 milliseconds. To solve this problem, my initial idea was to use a boolean variable “hasShot” to determine if there are any bullets on the screen. If hasShot==false, which means that there is no bullet on the screen, a bullet can then be generated. This doesn't generate a lot of bullets, but the downside is that there can be only one bullet on the screen at the same time, which makes the game boring. After looking up the information, I found that it was possible to add a timer to the bullets via the postDelayed method in "android.os.Handler", in that way we could freely change the firing rate of the bullets and hence the problem was solved.

b. Another problem is how to properly display the explosion effect. At first, I just displayed a single picture when exploding, which would create an awkward effect because the explosion is a dynamic process, which could not be shown merely in one picture. Then, I came up with the idea of using a video to show the explosion. After trying, I found that the background color of these videos was difficult to remove. Eventually, I found that I could build an exploding object to store sequences of the explosion pictures. This object has a variable “explosionFrame” that is used to record which image is going to be displayed next for a particular explosion instance. In that way, we can simulate the dynamic process of the explosion with a continuous play of images.

Future development:

a. Establish a more complex and interesting game structure. For example, we can set different levels for the game. The player can only move to the next level if he successfully beats the boss of the previous level. In lower levels, the enemies move slowly, and the attack is not intense, instead, in higher levels, the enemies are more powerful, and the boss can even have unique abilities such as freezing the player for a few seconds and so on. By setting the difficulty levels, we can make the game more challenging and attractive.

b. Add more fantastic visual and audio effects to create better user interaction. For example, when the player gets hit by a bullet or a small enemy, a short alarm can sound, and the screen border can turn red and flash to indicate the user is under attack.

c. Enable the Bluetooth or WIFI connection so that multiple players could group up and share the game.

d. Design more interesting objects to increase the playability of the game. For example, other than the prop heart, we can add a prop missile to the game. After the player obtains it, he can launch missiles at designated locations by tapping the screen to kill the enemies within that range.

e. We can display a global game leaderboard at the end of the game screen, which can motivate players to compete with other players.

References:

1. How to set timers in java: https://blog.csdn.net/u010687392/article/details/43955395

2. How to play audio:

https://www.tutorialspoint.com/how-to-play-background-music-in-android-app

https://blog.csdn.net/sweetlcw/article/details/121522374

3. Game framework references:

https://o7planning.org/10521/android-2d-game-tutorial-for-beginners#a1955354

https://github.com/sandipapps/SpaceShooter/tree/master/app/src/main

4. Background music: https://www.chosic.com/download-audio/28703/

5. Background image: https://www.pinterest.com/pin/501869952205364281/

6.Sprites:https://www.aigei.com/s?tab=file&type=2d&dim=aircraft_firing&term=album&page=2#resContainer

https://616pic.com/sucai/vj9i0g67z.html