Java Project: Catch The Ball

Li Weijing
No.19204246
weijing.li@ucdconnect.ie

***Abstract*— In this project, I make a body sensing game, called catch the ball. To get the scores, you should try to catch balls as much as possible by moving your head before the "Timer" reaches 30. The user interface is implemented by Java Swing. The program use multithreading technique to control the position detection by face++ [1], the image taken by webcam and the ball falling in the JFrame. Since the API has a concurrency limit for free users, so the detection of the position has latency. The game has a database connectivity, so you can see the ID and scores of three top. The game is tested via JUnit.**

***Index Terms*— multithreading, Face++ API, Webcam, Java Swing, Database connectivity, JUnit.**

I. INTRODUCTION

Catch the ball is an interesting body sensing game which detects the position of your head and calculate the score of the balls you caught. Here is the guide for the game.

Firstly, as Fig.1 shows, you should fill in your name in the blank, and click "Start Game" button below.

Privacy Policy Reminder: Note that after this step, the camera of your laptop will be used to detect your head position, so please make sure your camera is available and is allowed to be called by the program. The latest image will be stored at the "\ java project\ camera pics", and won't be automatically deleted after the game.

Then, as Fig.2 shows, the background colour of the

Fig. 1. Login window

interface will change to black, and the little man raising both of his hands exists at the bottom, this man denotes the position of your heads, and it moves according to your head position. The place between two hands of the man is the valid catch ball position, the ball will disappear when it is caught. The score of the player is on the top left side of the interface, while the Timer is on the top right side. The game will exit when the timer reaches 30. **Note that** the larger balls falling quicker and have more scores than smaller ones.

After that, as Fig.3 shows, on the "game over" page, you

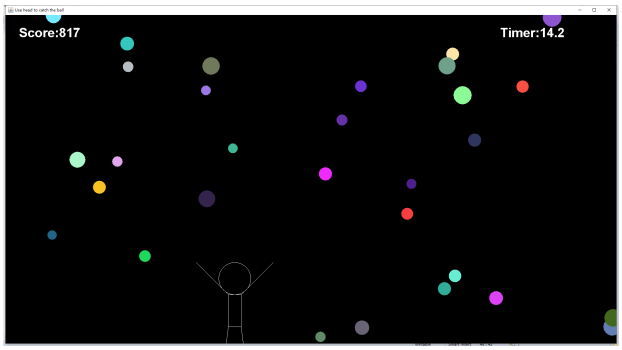

Fig. 2. Game window

can review your scores, and choose "View Scores" button to view the top three players or choose "Start game" button to restart the game. By clicking "View Scores" button, you

Fig. 3. Game over window

can see the top three scores of the game in the dataset, then you can click "Start Game" to restart the game.

The details of the implementation will be introduced in the following sections.

A. Framework

This program has 8 .java files, and this section listed the functionality of each file.

- Main_program.java

  This is the main class of the program, creating and initializing the JFrame window, and also manage the

Fig. 4. View Scores window

trigger of the button to switch the game interface.

- Get_answer_and_paint.java
  The text in Fig.1,3 and 4 is controlled by this class, it paints the text on the panel of the JFrame, and get the play_ID which filled in in the first step.

- Ball_fallen.java
  Ball_fallen is the most important class, which controls everything of the Game window. It uses three multithreading technique to control the whole game. The first thread is used to paint the ball, the little man, score and timer on the window by calling **run.java class**. The second one controls the camera, it takes pictures of the player every 100ms by webcam, and since the name of the image file is the same, so it will cover the last image, the program always keep only one latest image. The third thread used to call the **get_api.java class** to send the image to the remote server and get the position data back, it runs every 100ms. The api has a limit on the concurrency of the free user, sp part of the request will be denied, the thread returns data roughly every 400ms.

- Run.java
  This class is run with the Ball_fallen.java to form the multithreading technique, it is extended from the default Thread class.

- Get_api:
  The class is copied from the official sample to send the image and return the position data. [1]

- Ball.java
  The setting of the ball is getting from this class, it randomly set the number and size of the falling balls, and have the draw method to draw it on the graphic.

- player.java
  The setting of the little man is getting from this class, it gets the position from $run.javaclass$ and have the draw method to draw it on the graphic.

- create_database.java
  Creating database and initializing it, not used in the running of the game program.

B. Requirements satisfiction

In this section, I will list all basic and advanced requirements that the program has satisfied along with some other interesting ideas it has implemented.

Basic Requirements:

- The game should have a user interface, ideally a 2D graphical user interface (which can be quite simple)
  Satisfied: This game has an interface.

- Two players should be able to take turns (either consecutively or in parallel as appropriate for the game chosen)
  Satisfied: This game can return and record the scores of each player, it can be played in turn.

- When some win condition is met, the game should end
  Satisfied: This game can switch to the game_over page after the timer reaches 30, and on the game_over page, the player can choose to play the game again or view the scores. Note that the timer doesn't mean 30 seconds, it means the

- Players should be able to specify names, these should be displayed to indicate specific events, e.g.whose turn it is
  Satisfied: The player can fill in the game at the login page Fig.1, and the player name and scores will be stored in the database after game over.

- The game must well unit tested
  Satisfied: I have set a unit test for some class method.

- You must be able to record (persistently) previous results of games, if your game has a score this could be the highest scoreboard, otherwise a list of previous game outcomes, or as appropriate for your game
  Satisfied: This game has a highest scoreboard like Fig.4.

- All code should be placed within a well-structured Java package with accompanying JavaDoc
  Satisfied: The Javadoc has been generated, please refer to the java project folder.

Advanced Requirements:

- Database connectivity – you can use the database made for you in the DB lab.
  Satisfied, in the view_score window(Fig. 4), the program has connected the database and get the top three scores.

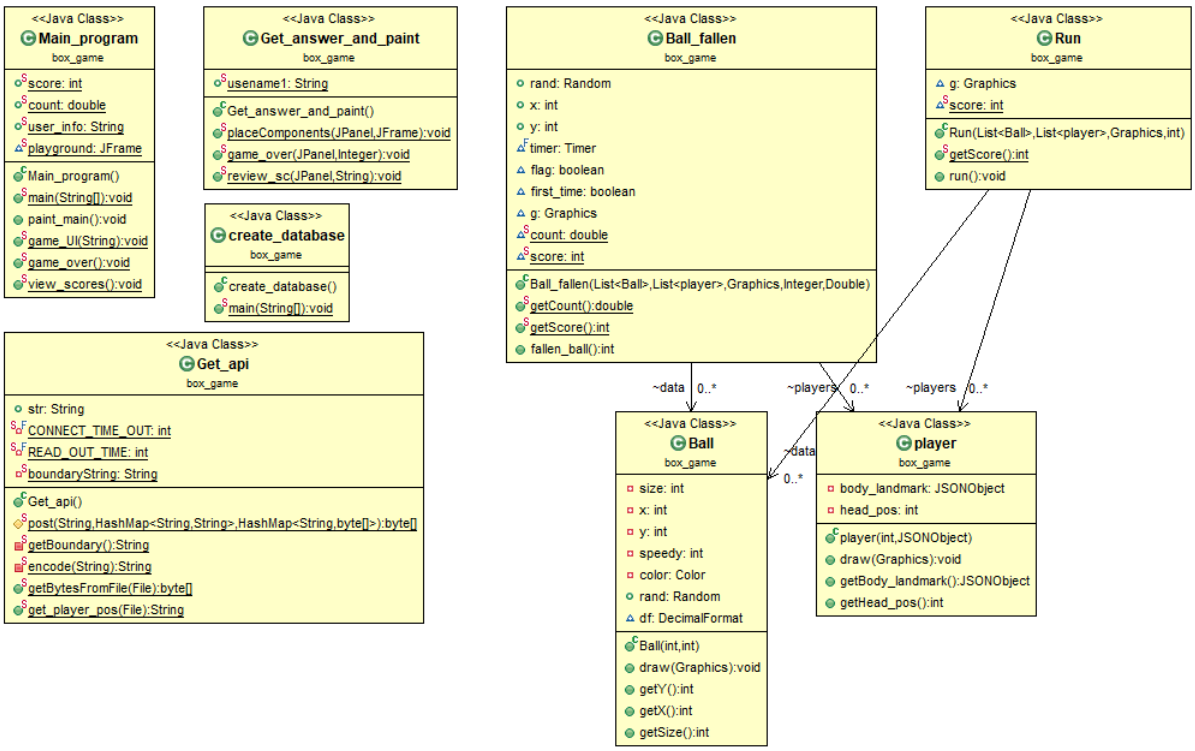

Fig. 5. class diagram

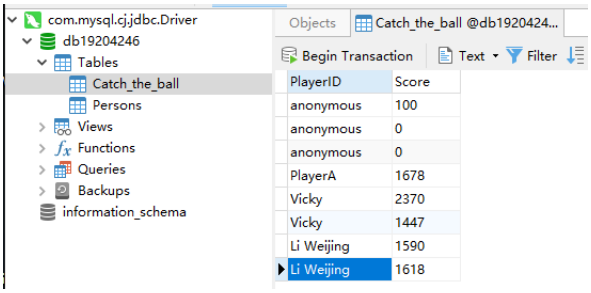

Fig. 6. Database

- **Own Idea:** Multithreading:
  In this program, I have implemented three different thread runs parallel to controls the window, the camera and the api.

- **Own Ideas:** Face++ API
  The skeleton detection API use to get the position of head of player [1].

- **Own Ideas:** Webcam Capture API
  This library allows you to use your build-in webcam directly from Java. [2]

II. Structure and Implementation Details

In this section, I will introduce the Technical approach use in the project in detail as well as the structure of the program with class diagram.

A. Class diagram

In the class design, it can be divided into two parts, the first part is the main game program includes Ball_fallen.java, Ball.java, player.java, Run.java and Get_api.java. These five classes runs together.

Firstly, the Ball_fallen.java start three thread, thread 1 start at 100ms and runs every 100ms. It starts later than the other two threads because the program needs to start the camera and API first to get the data and pass it to thread 1 to draw the little man. In the thread 1. The thread calls Run.java once to start the thread to paint. Then Run.java get the setting of the balls from the Ball.java and call the draw method in Ball.java and player.java to paint them. Besides, it calculates the scores by the size of the Ball and paint the Timer on the top right side of the window. While the counter(timer*10) reaches 275 the ball will stop falling, and all other thread will stop at the counter(timer*10) 300. The balls stop 25ms before to let the player catches all of the balls on the screen before the game over.

Thread 3 start at 0ms and runs every 100ms, thread 3 is used to control the camera to take the image of the player and to reduce the size of the file, it only stores one latest image of the player.

The thread 2 starts at 5ms, it leaves a little time for thread 3 to take the image, then started to call the Get_api.java to

connect the API, send the image and get the return value in java format. Although it runs every 100ms, we can only get data every 400ms because of the concurrency limit of the API, which may cause the latency of the game.

All of the other classes belongs to the second part, they control the main process of the game, including the login, main game, game over, review score, connect the database and restart the game.

B. Unit testing strategy

Since most of the classes are based on Graphical user interfaces, like input text or camera, so I only test the get API class. By using an image as the input, the test shows that it can get the JSON from the remote API side.

III. CONCLUSIONS

This simple body sensing game, catch the ball, contains many java techniques such as multithreading, database, extend class, etc. Players can get scores by trying to catch the ball through moving in front of the camera. Thanks to many useful APIs like webcam and face++, this game is not that boring, but it still has many defects like window refreshing and latency. I have to say I like this project, and really learn a lot through coding.

REFERENCES

[1] https://www.faceplusplus.com/skeleton-detection/.

[2] https://github.com/sarxos/webcam-capture.