Chyong Artur
XOG5D9
xog5d9@inf.elte.hu
Group 5

3. assignment/1. Task

10th December 2023

Task

Yogi Bear wants to collect all the picnic baskets in the forest of the Yellowstone National Park. This park contains mountains and trees, that are obstacles for Yogi. Besides the obstacles, there are rangers, who make it harder for Yogi to collect the baskets. Rangers can move only horizontally or vertically in the park. If a ranger gets too close (one unit distance) to Yogi, then Yogi loses one life. (It is up to you to define the unit, but it should be at least that wide, as the sprite of Yogi.) If Yogi still has at least one life from the original three, then he spawns at the entrance of the park.

During the adventures of Yogi, the game counts the number of picnic baskets, that Yogi collected. If all the baskets are collected, then load a new game level, or generate one. If Yogi loses all his lives, then show a popup message box, where the player can type his name and save it to the database. Create a menu item, which displays a high score table of the players for the 10 best scores. Also, create a menu item which restarts the game.

Objectives

Game Mechanics:

Design Yogi Bear's movement within the Yellowstone National Park, considering obstacles like mountains and trees.

Implement the ranger movement logic allowing it to move only horizontally or vertically.

Establish collision detection to ensure Yogi Bear loses a life and respawns (if any more live available) if a ranger comes within one unit distance.

Scoring and Level Progression:

Track the number of picnic baskets collected by Yogi Bear throughout the game.

Implement a level completion condition when all picnic baskets are collected, triggering a new game level loading.

Player Lifespan and Restart Functionality:

Define Yogi Bear's life system, enabling a respawn at the park's entrance if lives are remaining.

Integrate a mechanism to handle Yogi Bear losing all lives, prompting a popup for the player to input their name and save it to a database.

Menu Items and Interface:

Create a menu displaying a high score table for the top 10 players, incorporating database retrieval and presentation.

Implement a menu option to restart the game, providing seamless navigation within the game interface.

User Experience and Interface Design:

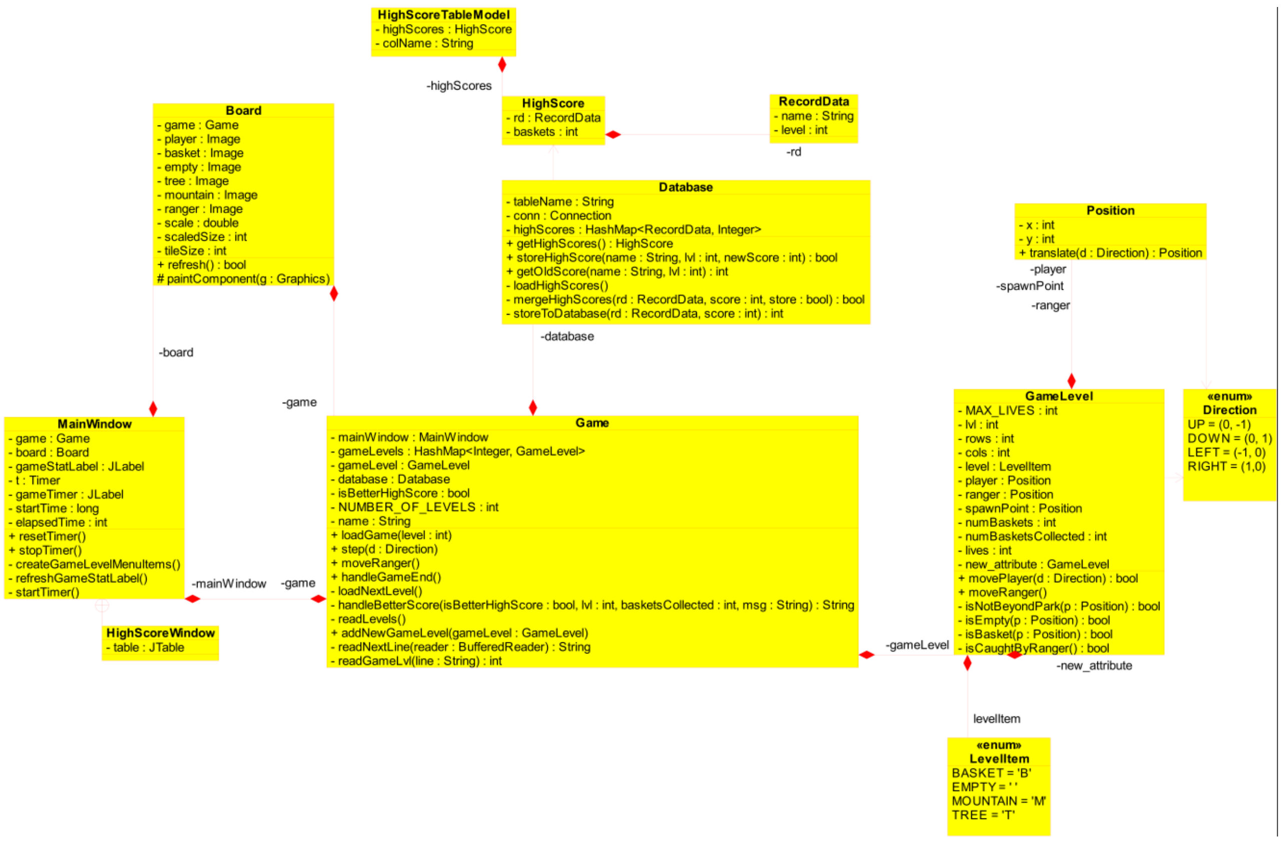
Develop a user-friendly interface with intuitive controls and clear visual feedback for game elements.

Ensure the popup message box for entering player name is easily accessible and navigable.

Database Integration:

Implement database connectivity for saving player names and scores upon game completion or failure.

Class diagram



Methods

Board class:

paintComponent(Graphics g) – overridden method. Based on the game, it creates and draws the elements such as BASKET, MOUNTAIN, TREE, EMPTY.

Game class:

loadGame(int level) – public method which loads the level based on the given parameter. It resets all the fields needed.

step(Direction d) – public method which calls movePlayer(d) method and checks if the game is finished.

moveRanger() – public method which calls moveRanger() method and checks if the game is finished.

handleGameEnd() – public method which handles the end of the game. It checks if the game is over and saves the new record when it's applicable.

readLevels() – private method which reads new levels (text files with the description of the levels) and saves them into gameLevels.

GameLevel class:

movePlayer(Direction d) – public method which checks if the movement to the given direction is possible and if it is it changes the player's location.

moveRanger() – public method which generates the random number and based on the number decides to which direction the ranger will move.

isCaughtByRanger() – private method which checks if the ranger and the player are in the distance of one unit from each other.

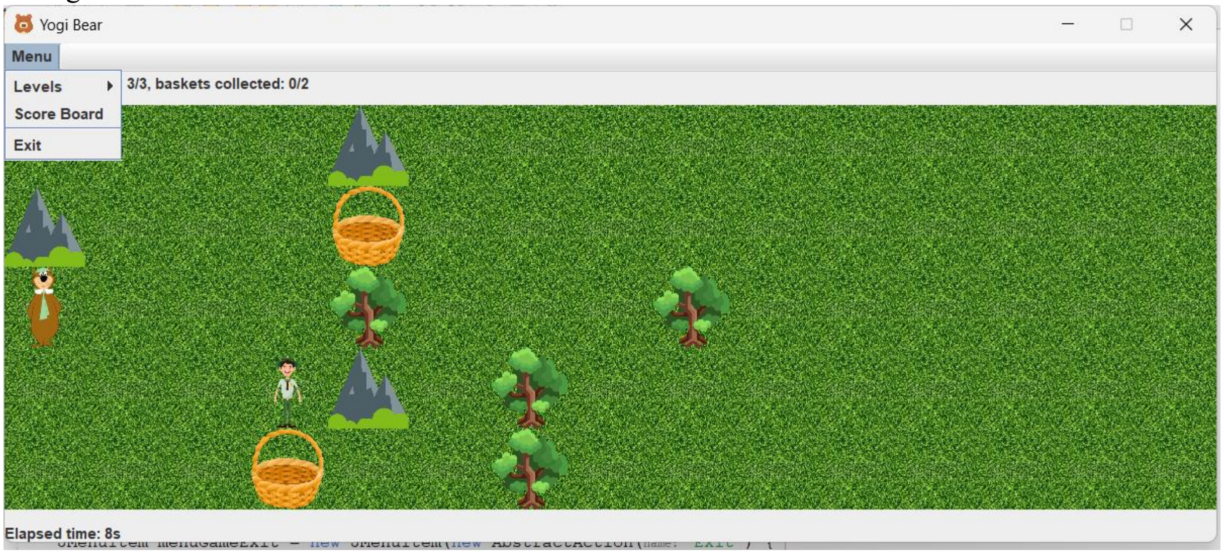
Database class:

getHighScores() – public method which returns ArrayList of 10 elements of HighScore with the highest number of collected baskets from the database table.

getOldScore(String name, int lvl) – public method which gets the old score based on the name and the level. If such a record doesn't exist, it returns 0.

mergeHighScores(RecordData rd, int score, Boolean store) – private method which checks if the new score is higher than the old score and if it is it updates the record in the database table.

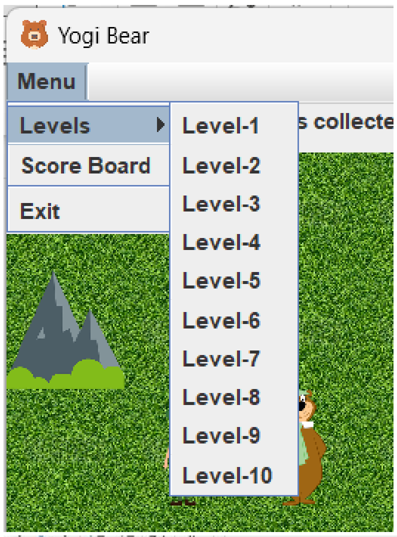
Manual testing:

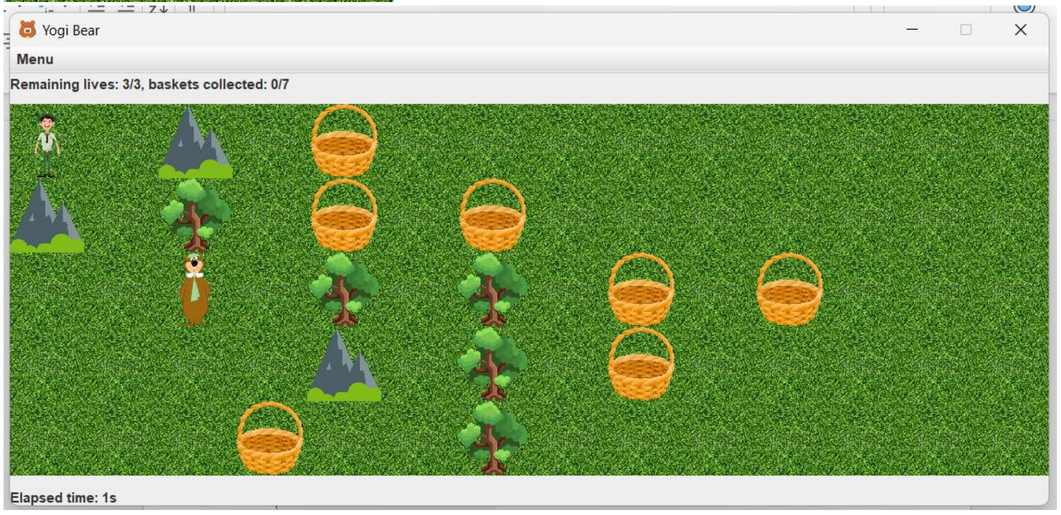
1. The game is initialized. The levels are read from the files. The icons are loaded. The timer is working. The player cannot go through the mountains and trees, only on the empty spots and through baskets.


3/3, baskets collected: 0/2

Elapsed time: 8s

2. The levels can be chosen.

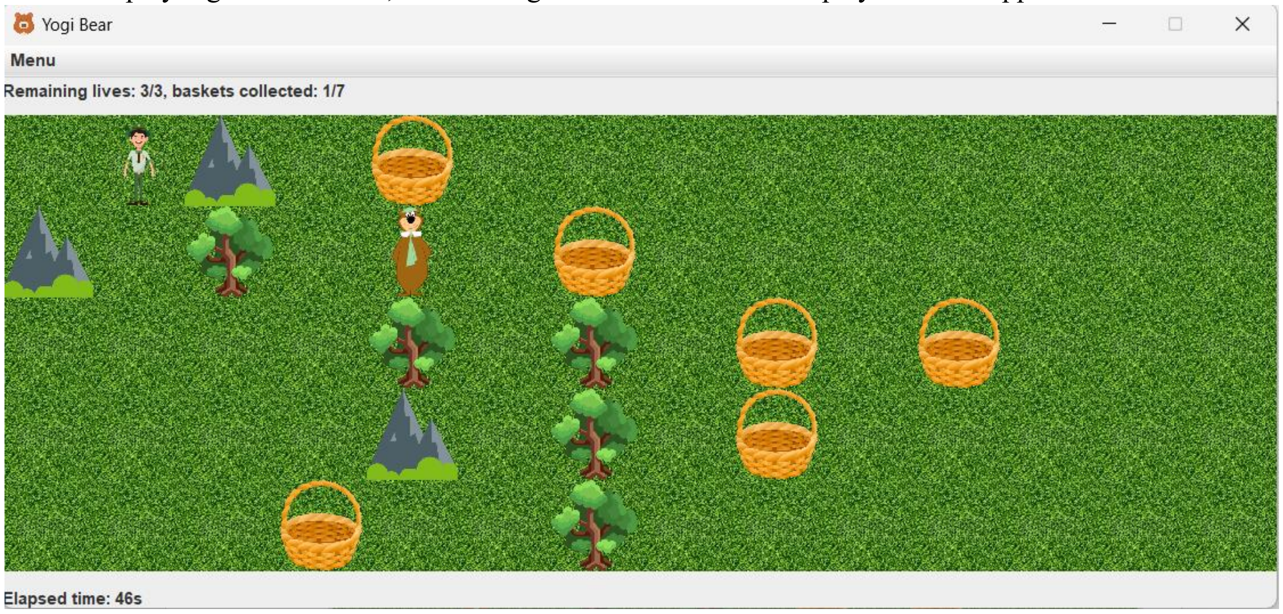




Remaining lives: 3/3, baskets collected: 0/7

Elapsed time: 1s

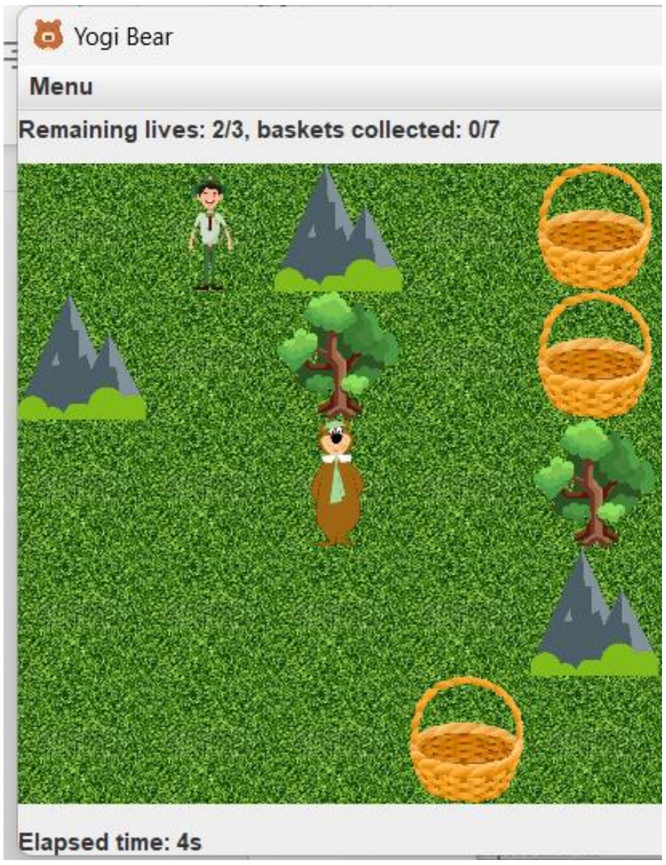
3. When the player gets to a basket, the basket gets collected and it’s displayed on the upper label.



Remaining lives: 3/3, baskets collected: 1/7

Elapsed time: 46s

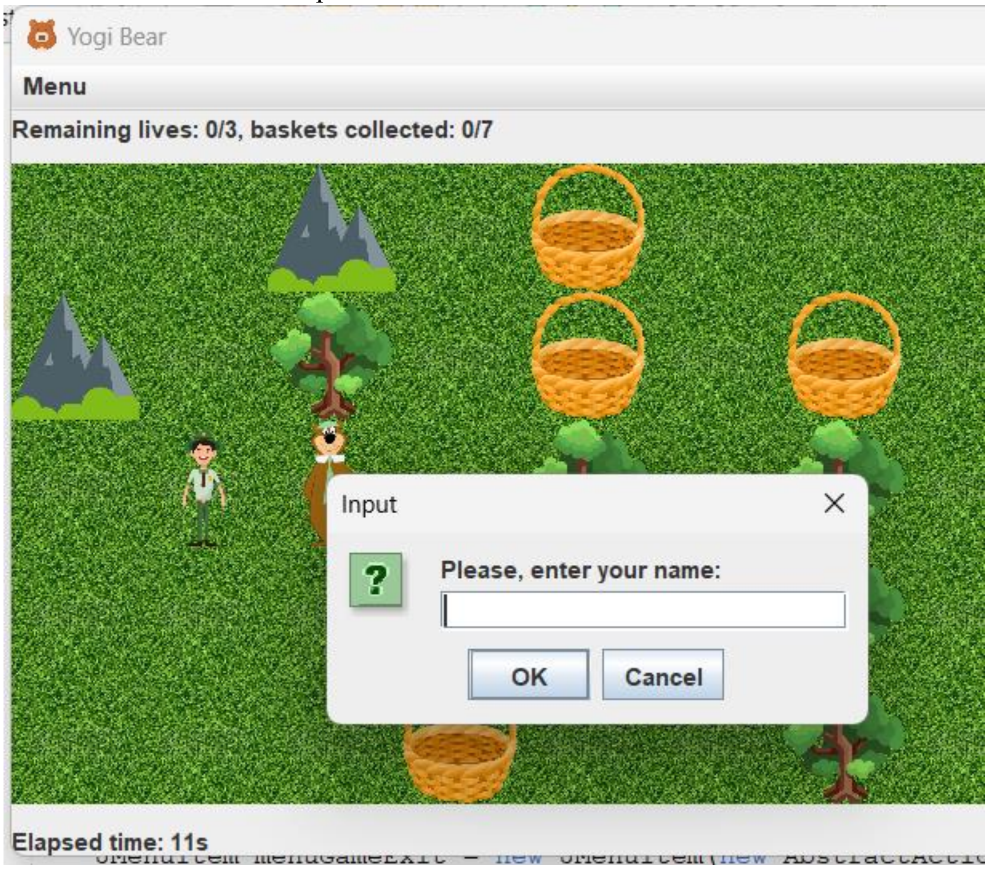
4. When the player gets close to the ranger, the number of lives gets decremented and the player respawns at the initial point.



Remaining lives: 2/3, baskets collected: 0/7

Elapsed time: 4s

5. When the number of lives gets 0 or all the baskets are collected, the game finishes and the input field for the name shows up.

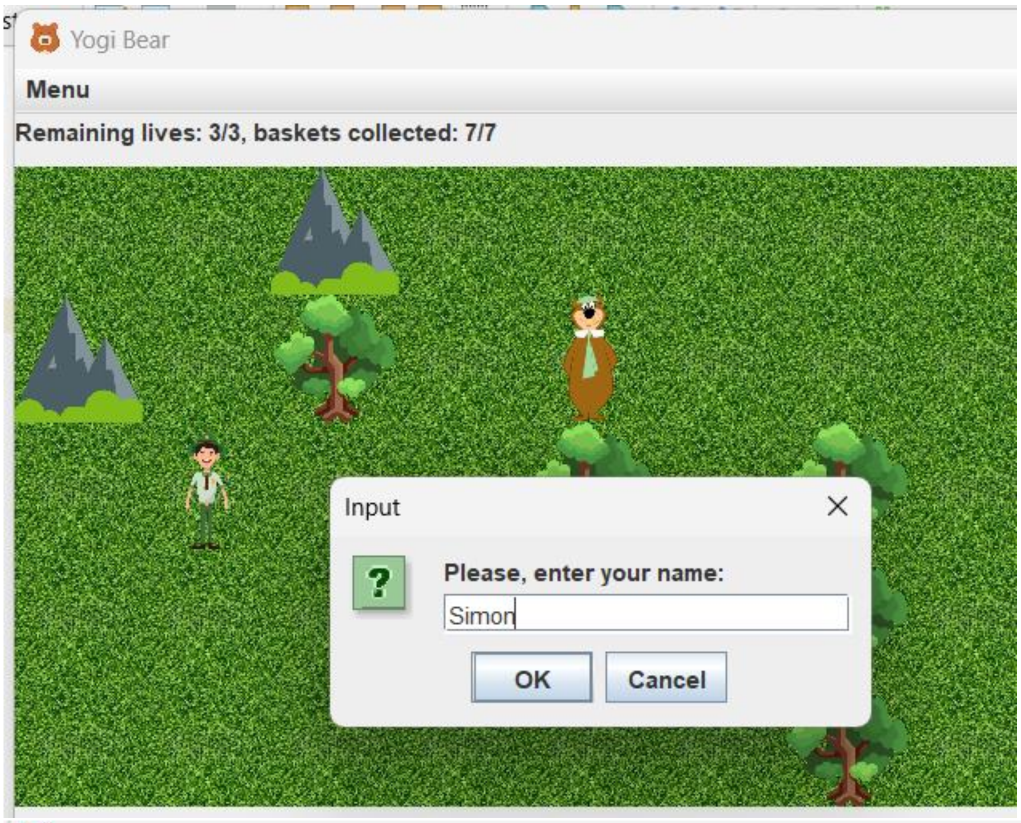


Remaining lives: 0/3, baskets collected: 0/7

Please, enter your name:

Elapsed time: 11s

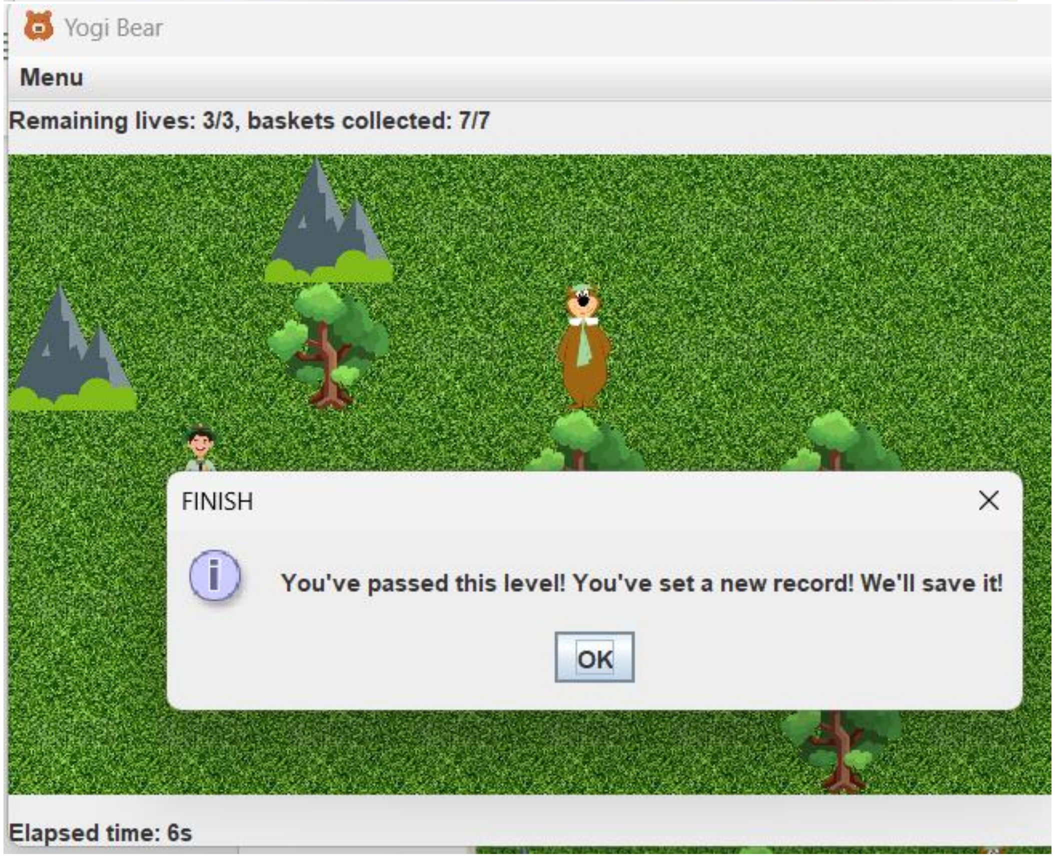
6. If the new score is better than the old one, it gets saved. Otherwise, no.



Remaining lives: 3/3, baskets collected: 7/7

Please, enter your name:

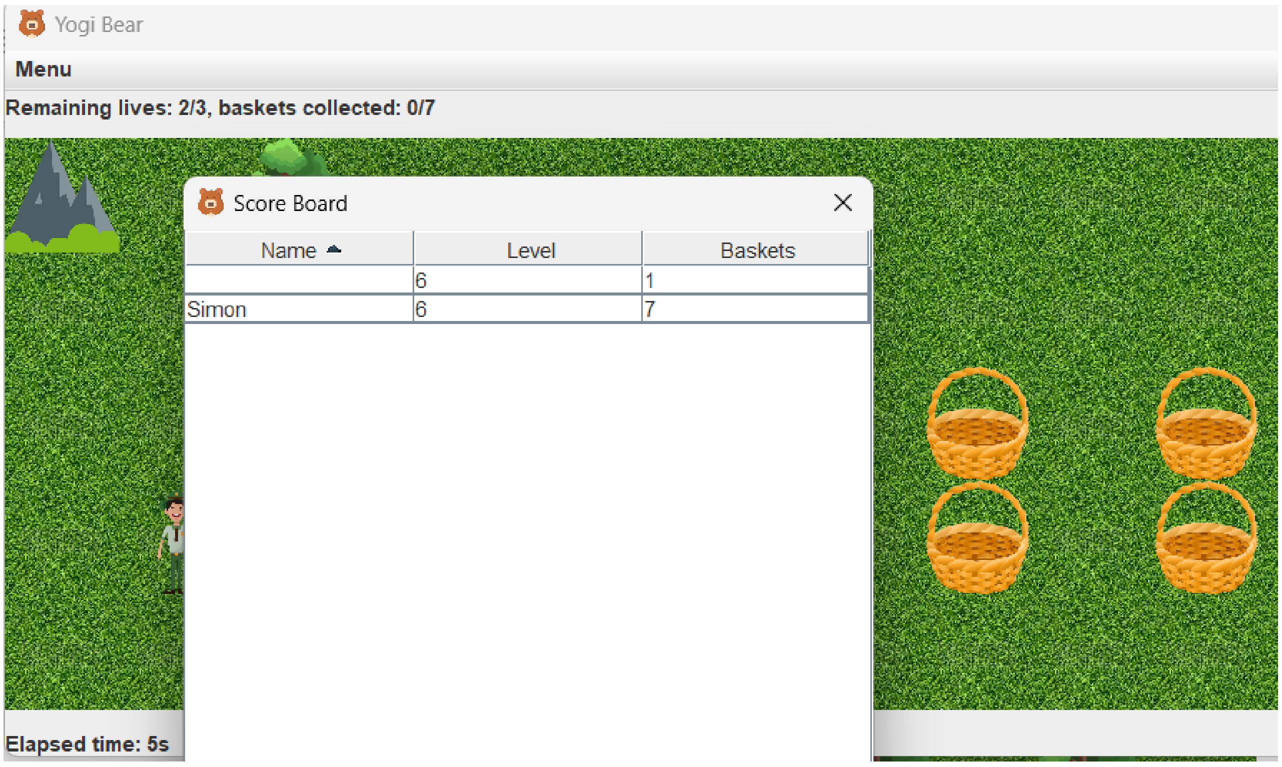
Simon



Remaining lives: 3/3, baskets collected: 7/7

You've passed this level! You've set a new record! We'll save it!

Elapsed time: 6s



Remaining lives: 2/3, baskets collected: 0/7

Name	Level	Baskets
	6	1
Simon	6	7

Elapsed time: 5s

Connection between the events and event handlers

ActionListener for Timers

Timer gets the delay time (in milliseconds) as a parameter, and it executes actionPerformed based on the given delay time. One is used to move ranger all the time while the game is not finished. One more is used to update the top label and GUI board. One more is used to update the elapsed time.

KeyAdapter for MainWindow

It is used to read the key given by the user and based on them to move the player.