Programming Technology Documentation

Naveed Abdullah Task 01 OXRMEB

oxrmeb@inf.elte.hu

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 messagebox, where the player can type his name and save it to the database. Create a menu item, which displays a high score table of players for the 10 best scores. Also, create a menu item which restarts the game.

Analysis:

The "Yogi Bear" game is a 2D action game where players control Yogi Bear as he navigates a forest environment, avoiding obstacles, collecting picnic baskets, and evading rangers. The game operates across multiple levels, with increasing difficulty as the player progresses. The objective is to collect all picnic baskets while avoiding rangers and obstacles. If a ranger catches Yogi, the player loses a life. The game is designed to be fun and challenging, requiring both strategy (avoiding rangers and obstacles) and agility (collecting baskets).

The program provides an interactive graphical user interface (GUI) and implements game logic that handles player movement, collisions, and progression through levels. The game also includes a scoring system that tracks the player's progress, as well as high scores stored in a database. Upon winning or losing, the user is prompted to either restart the game or check the high scores.

Plan:

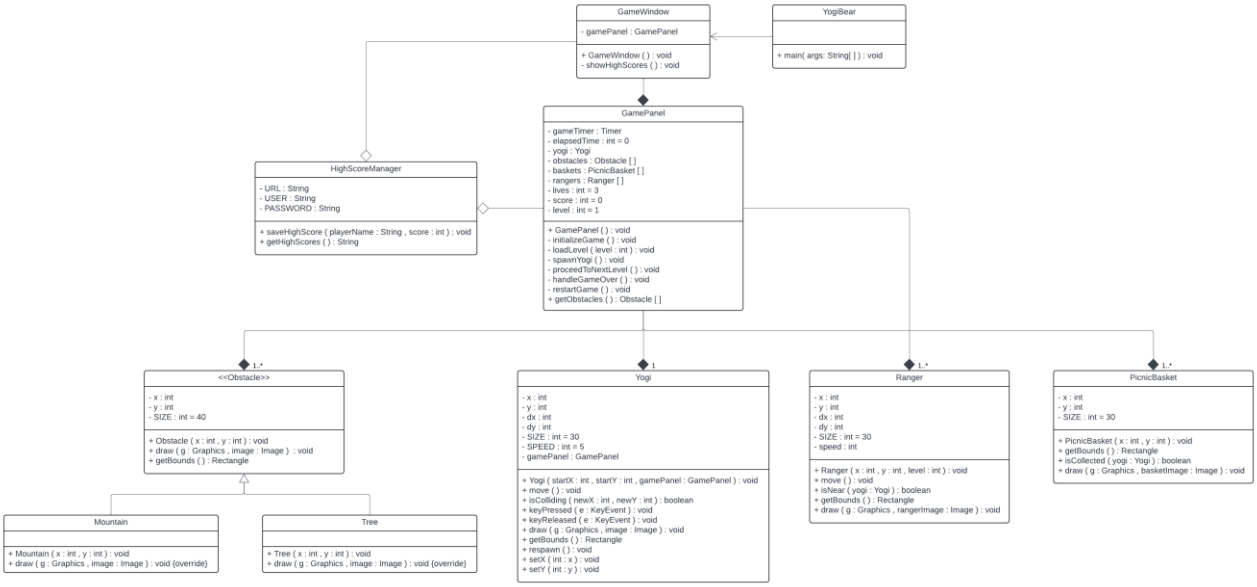
- **YogiBear (Main Class)**
  - Launches the application and initializes the main game GUI (GameWindow).
- **GameWindow (Graphical Interface)**
  - Handles user interface for displaying the game:
    - Sets up the window for the game (title, size, and menu).
    - Includes a menu with options like restarting the game and viewing high scores.
    - Displays the game panel (GamePanel) where the game logic is implemented.
    - Opens a window for the high scores, displaying top scores stored in the database.
- **GamePanel (Game Logic Controller)**
  - Controls game flow:
    - Initializes the game elements (Yogi, obstacles, picnic baskets, and rangers).
    - Manages game loops (such as updating the game state every frame with actionPerformed())
    - Detects collisions, collects baskets, and checks if the game is won or lost.
    - Handles transitions between levels, increasing the difficulty as the player progresses.
- **HighScoreManager (Database Integration)**
  - Manages saving and retrieving high scores from a MySQL database:
    - Saves the player's name and score when the game ends.
    - Retrieves and displays the top 10 high scores stored in the database.
    - Ensures the data is persisted for future game sessions.
- **Yogi (Character Representation)**
  - Represents Yogi Bear, including his position, movement, and actions:
    - Stores current position (X, Y) and movement direction.
    - Manages Yogi's movement in response to user input (via KeyListener).
    - Handles collision detection with obstacles and rangers, and updates his status when necessary (e.g., respawning after being caught by a ranger).
- **Obstacle (Obstacle Representation)**
  - Represents obstacles that Yogi must avoid:
    - Each obstacle has a defined position and size.
    - Handles collision detection with Yogi's movement
    - Draws obstacles on the game board.
- **PicnicBasket (Item Representation)**
  - Represents picnic baskets that Yogi needs to collect:

- Each basket has a position and checks if Yogi collects it.
- Draws baskets on the game board.
- Removes collected baskets from the list once Yogi picks them up.

Ranger (Item Representation)

- Represents rangers that chase Yogi:
  - Each ranger has a position and moves towards Yogi
  - Detects when they come near Yogi and reduces his lives if caught.
  - Can increase in number as levels progress, making the game more difficult.

UML Class Diagram:



Description of Methods

- YogiBear
  - **main**

    Initializes the game by creating an instance of the GameWindow class and setting it to be visible, starting the game.

- Obstacle

- **getBounds**
  Returns a Rectangle object representing the obstacle's bounding area, used for collision detection.
- **draw**
  Renders the obstacle as a gray square on the game panel using the provided Graphics object.

- PicnicBasket
  - **isCollected**
    Checks if the basket has been collected by Yogi (player character) by detecting an intersection between the basket's bounds and Yogi's bounds.

- Ranger
  - **isNear**
    Checks if the ranger's bounding area intersects with Yogi's, indicating proximity.

- Yogi
  - **isColliding**
    Checks if Yogi's new position would result in a collision with any obstacle on the game panel.
  - **keyPressed**
    Updates movement deltas (dx, dy) when an arrow key is pressed to move Yogi in the desired direction.
  - **keyReleased**
    Resets movement deltas when an arrow key is released, stopping Yogi's movement.
  - **respawn**
    Resets Yogi's position to a predefined starting point.

- GamePanel
  - **loadLevel**
    Loads a specific level of the game by:

    - Placing obstacles, picnic baskets, and rangers at predefined or randomized positions.
    - Adjusting ranger speed and number based on the level's difficulty.
    - Resetting Yogi's position to the starting point.
    - Clearing and reloading the `GamePanel` with the level-specific game elements.
      This method ensures that each level progressively becomes more challenging by scaling the difficulty dynamically.
  - **checkGameOver**
    Determines if the game has ended due to Yogi being caught by a ranger or if

the player has completed all levels.

- **proceedToNextLevel**

  Advances the game to the next level after completing the current one.

- **spawnYogi**

  Resets Yogi's position to the starting location (default spawn point) at the beginning of a level or after a specific event, such as being caught by a ranger. Ensures consistent positioning for the character.

- **restartGame**

  Restarts the entire game.

- **handleGameOver**

  Handles the game's end, triggered by either a loss or win condition.

- HighScoreManager
  - **saveHighScore**

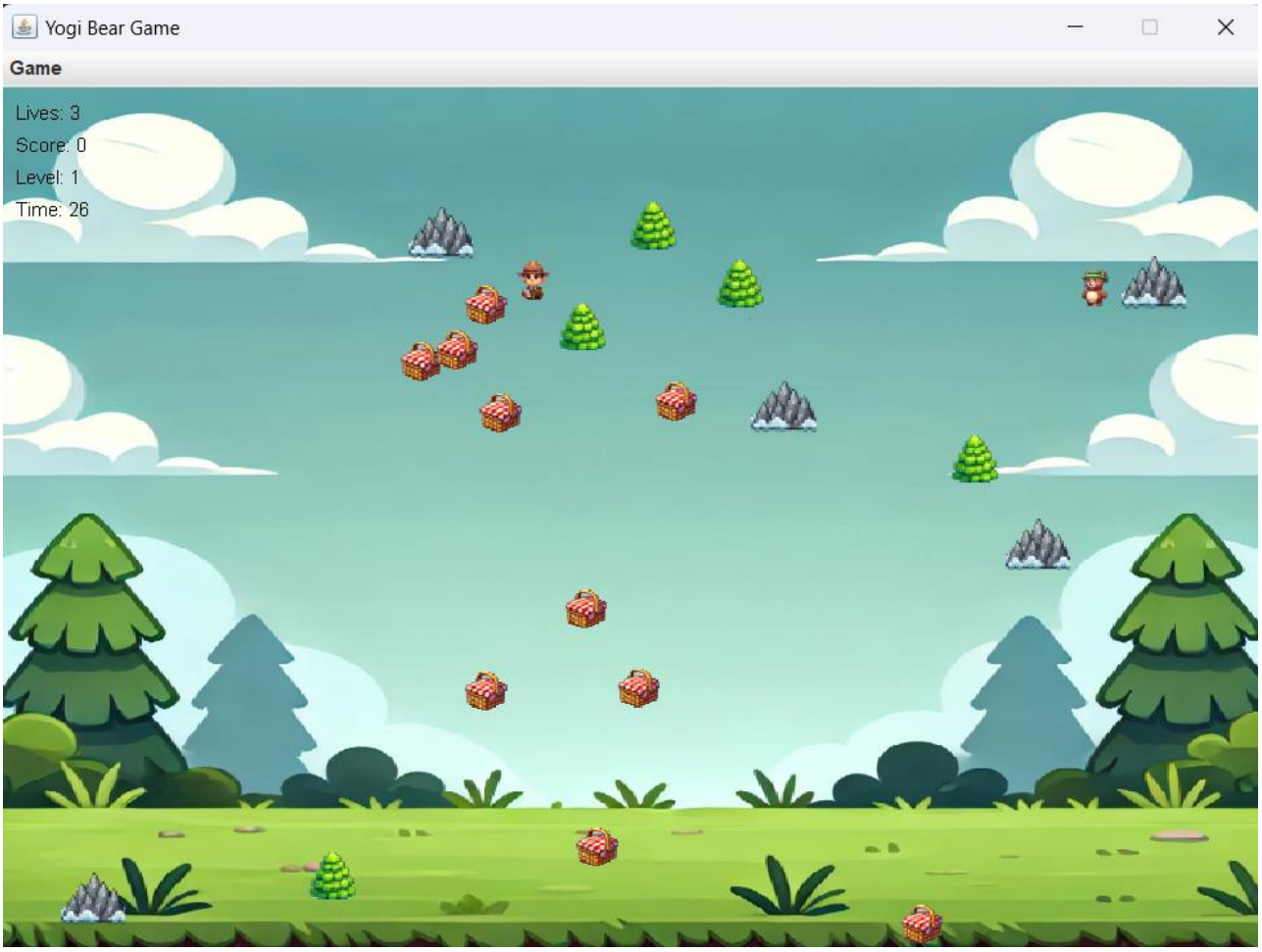
    Saves a player's high score to the database.
  - **getHighScores**

    Retrieves the top 10 high scores from the database, sorted by highest score.

White Box Testing:

1. Invalid Movement Test:

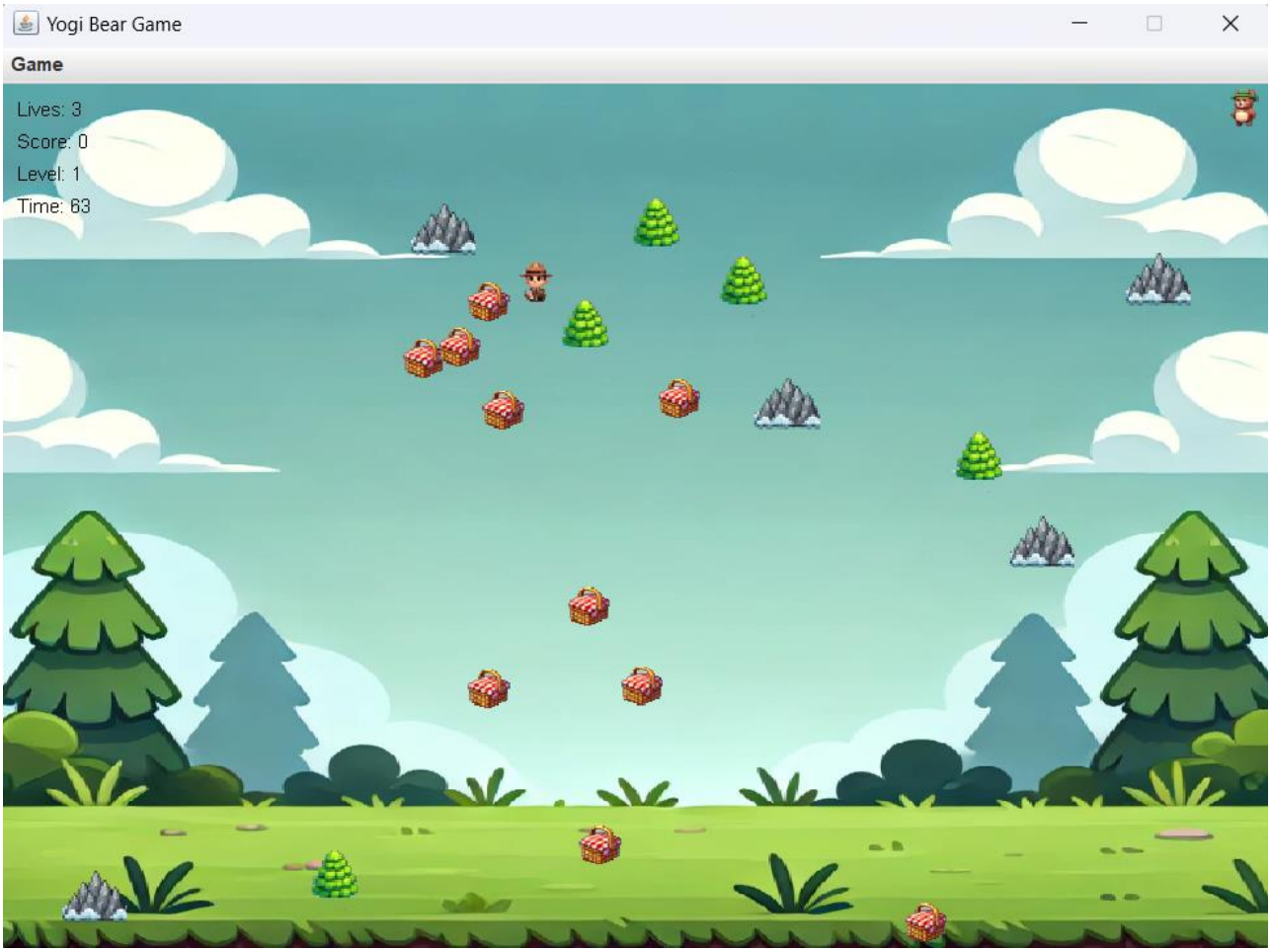
AS a player
I WANT TO be unable to move Yogi into an obstacle.
GIVEN there is an obstacle in the next tile
WHEN I press a movement key to move Yogi into the obstacle
THEN Yogi's position should remain unchanged, and no collision should occur



Here, the yogi sprite cannot move from this obstacle, even on the right key.

2. Valid Movement Test:

AS a player
I WANT TO move Yogi to a valid tile
GIVEN the next tile is walkable
WHEN I press a movement key
THEN should move to the new position, and the screen should update

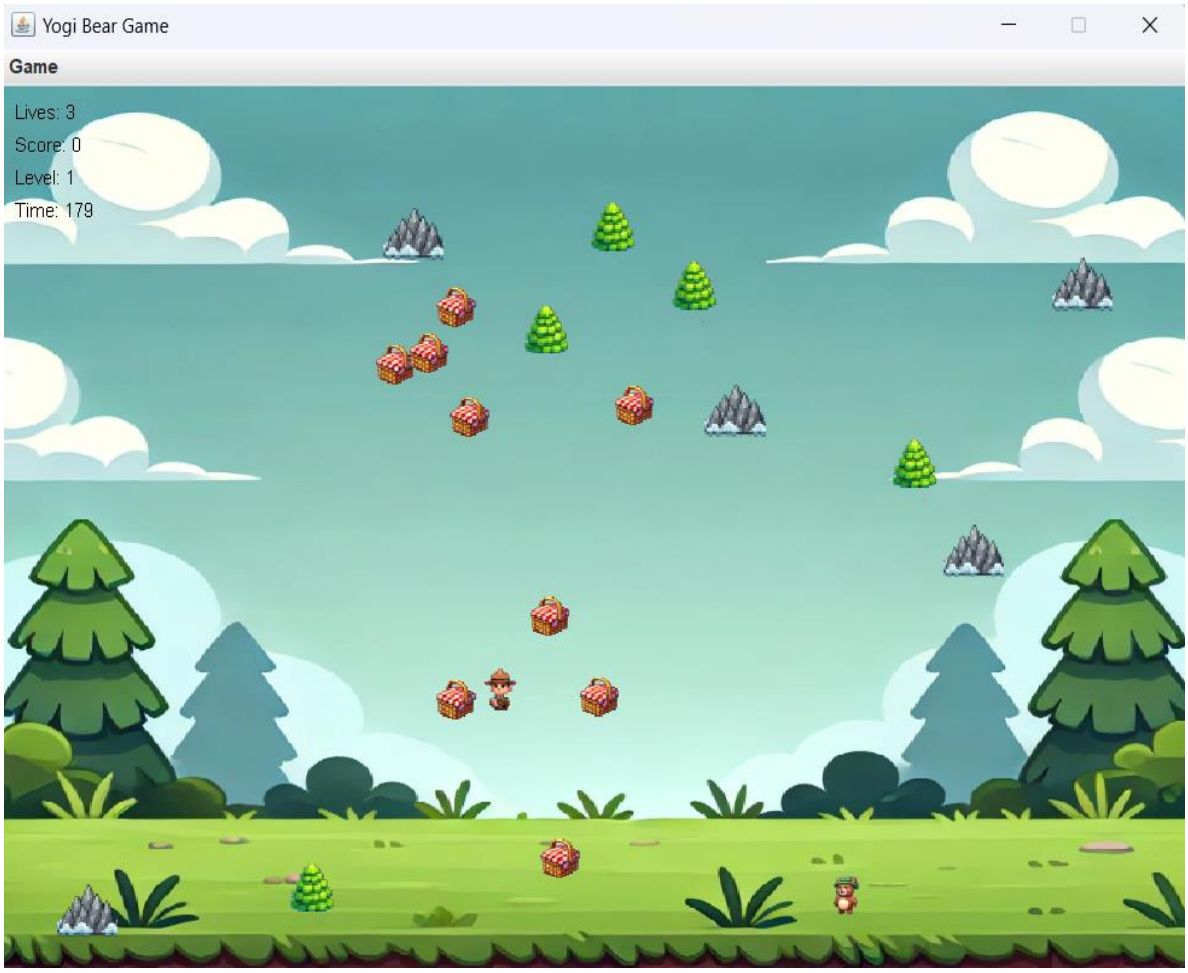


I moved the blue sprite from the previous game to this new position.

3. Collecting Picnic Baskets

AS a player
I WANT TO collect picnic baskets placed on the map
GIVEN Yogi's position overlaps with a picnic basket
WHEN Yogi moves onto the jar
THEN the picnic basket should disappear, and my score should increase

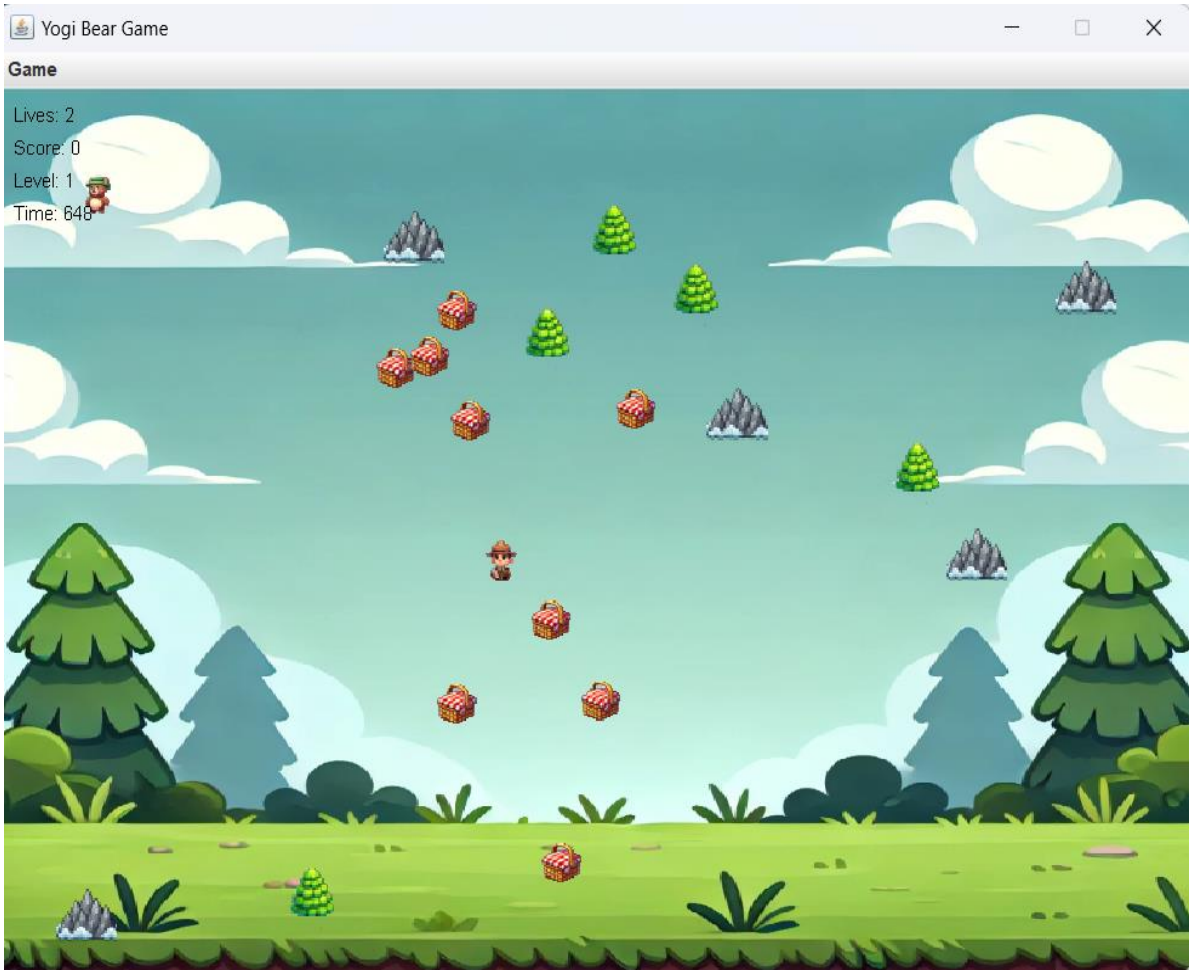
I moved the yogi sprite from the previous game to this new position where it was coinciding with the picnic basket, but it collected it, that's why it is disappear in the below picture.



4. Collision Test with Ranger:

AS a player
I WANT TO avoid being caught by Ranger
GIVEN Ranger position overlaps with Yogi's
WHEN the collision occurs
THEN the players loses a life and moves to start position

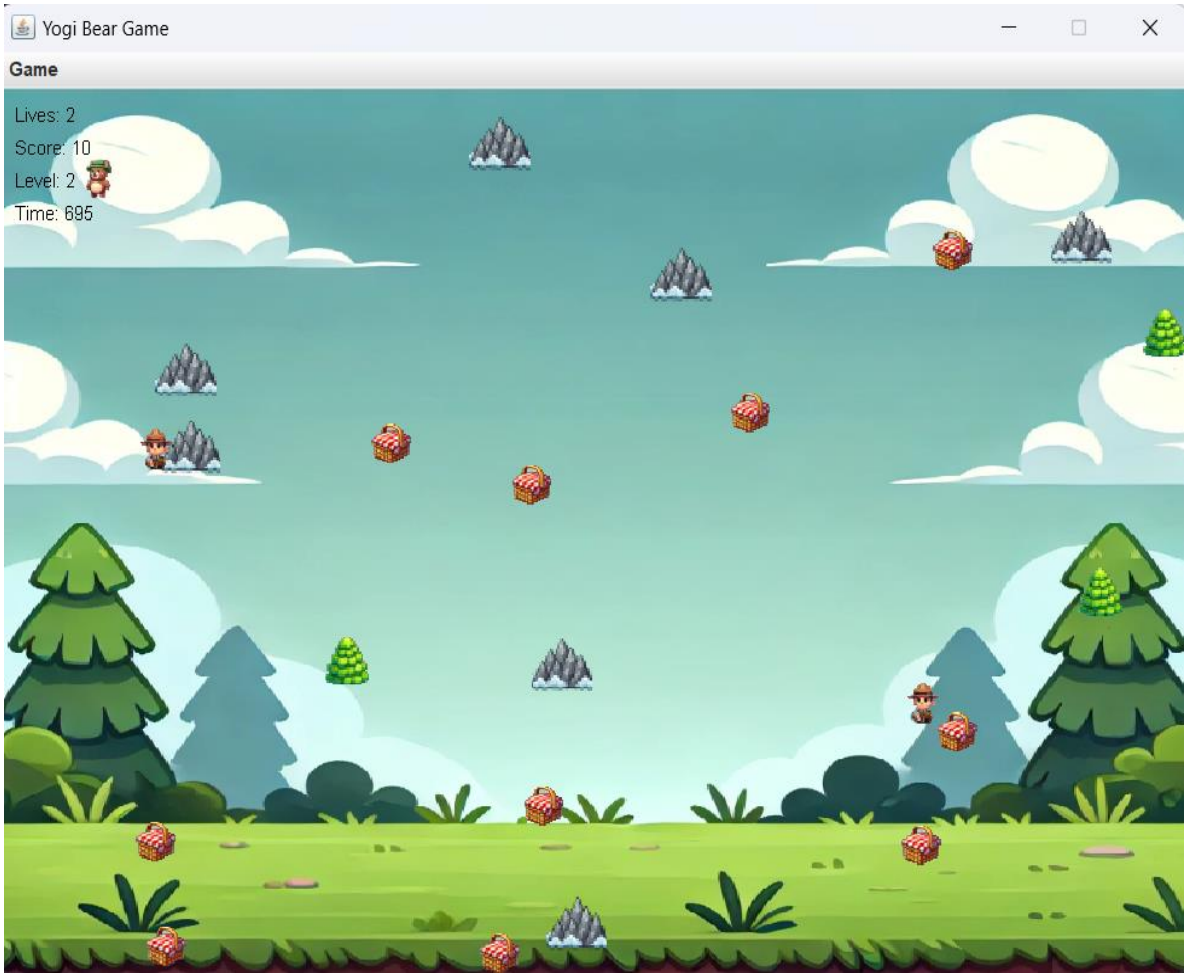
The life has been reduced by one when the yogi touches the ranger, and the yogi moves to the first position.



5. Next Level Transition Test:

AS a player
I WANT TO proceed to the next level after collecting all honey jars
GIVEN there are no picnic baskets left on the map
WHEN the condition is met
THEN the game should transition to the next level with increased difficulty

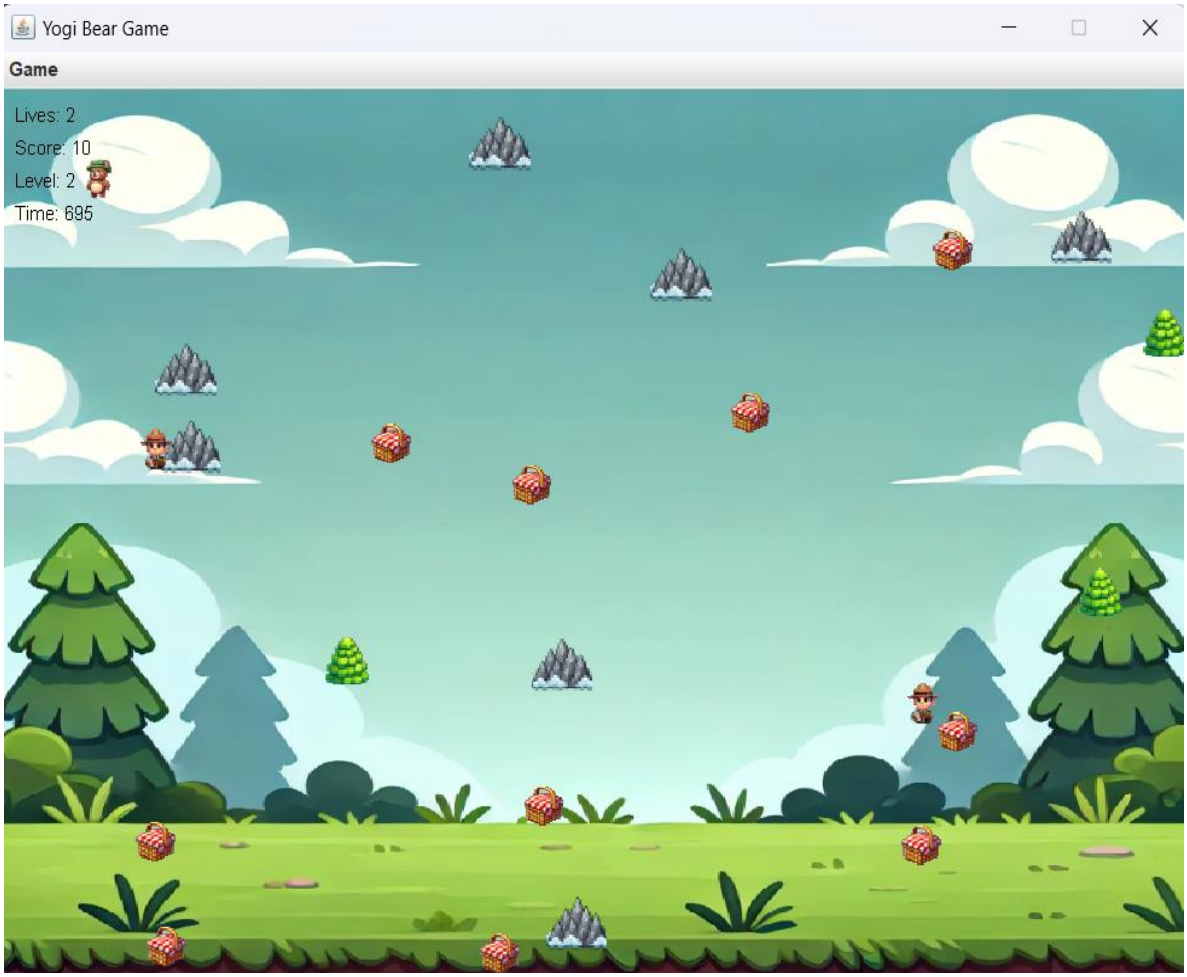
The level has been changed to level 2 and the number of rangers has also increased by 1.



6. Score Update Test:

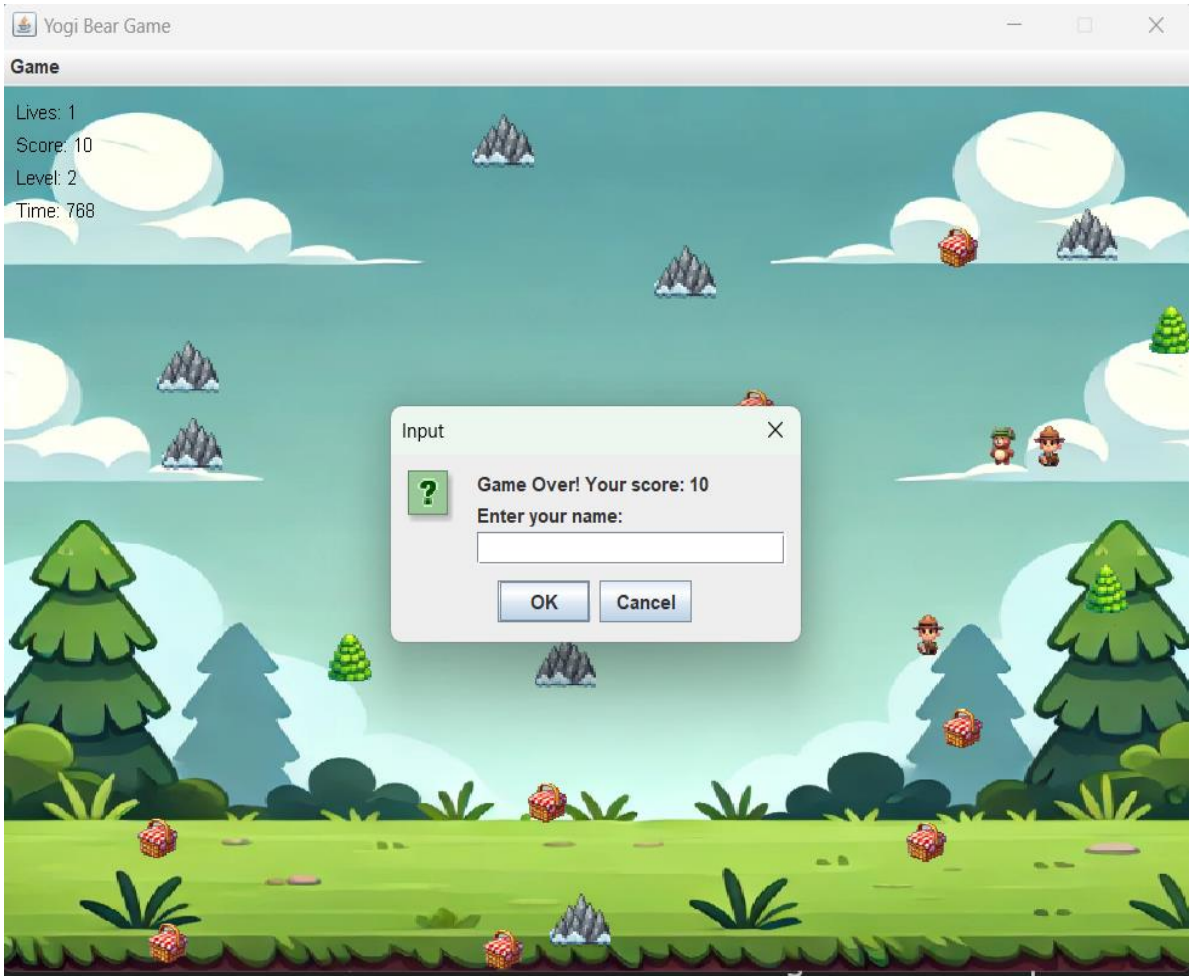
AS a player
I WANT TO see my score update in real-time
GIVEN I collect all picnic baskets
WHEN my score increases
THEN the updated score should display correctly on the screen.

The score is also changed by 10 when moving to the next level.



7. Game Over Condition Test:

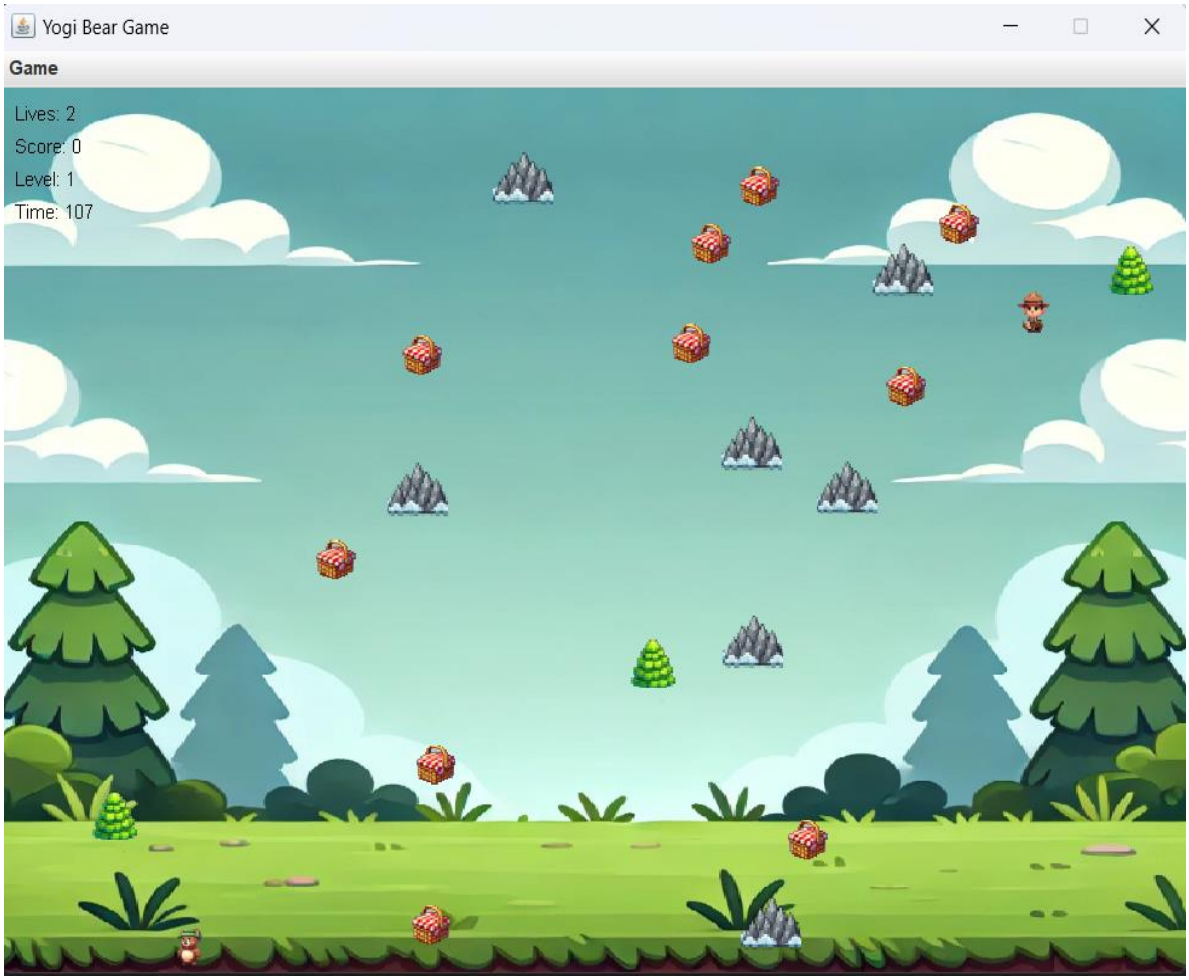
AS a player
I WANT TO lose two the game if I get caught by Ranger 3 times
GIVEN a collision with Ranger occurs
WHEN the game ends
THEN a “Game Over” dialog box should appear.



8. Movement Constraints Test:

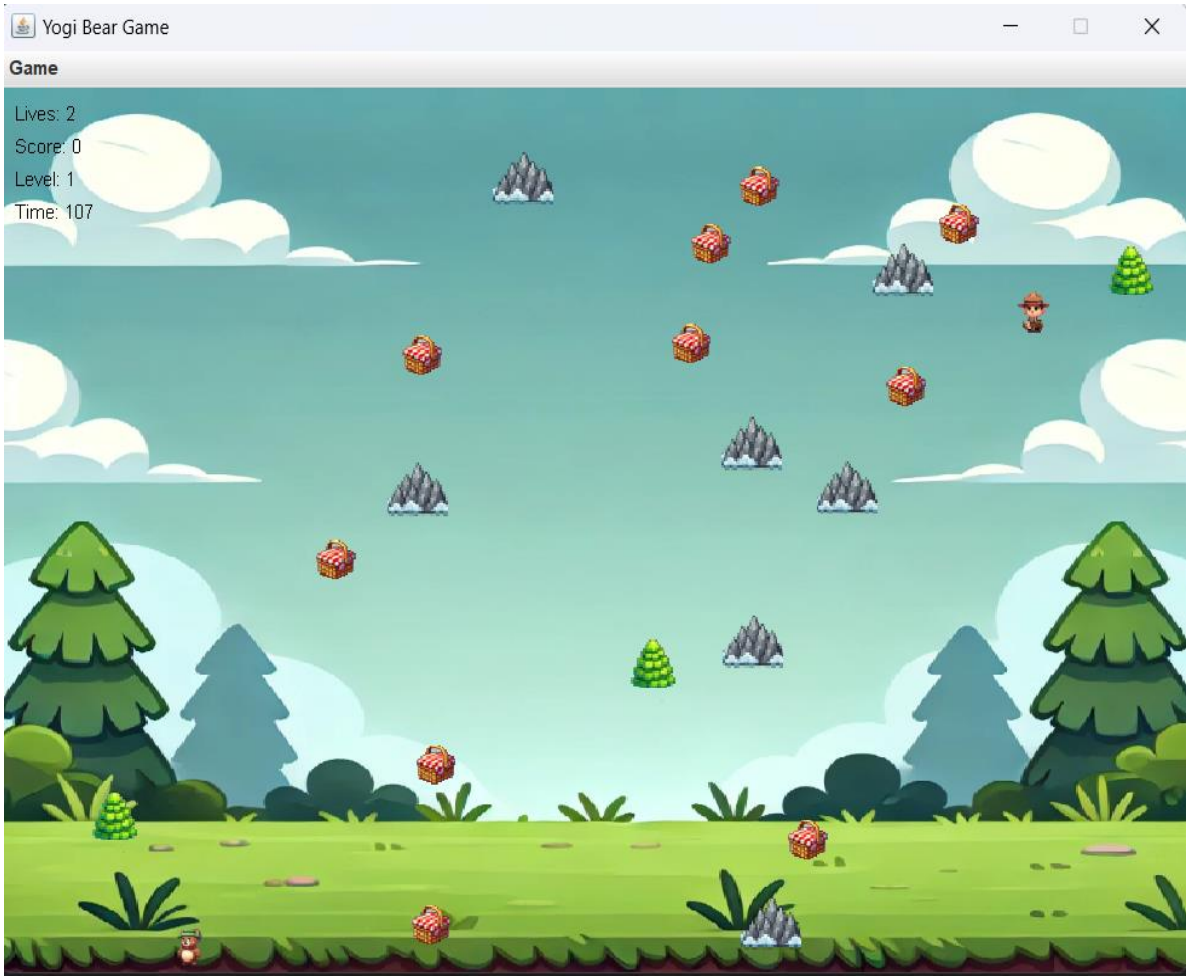
AS a player
I WANT TO be unable to move Yogi out of the map boundaries
GIVEN Yogi is at the edge of the map
WHEN I press a movement key in the direction of the boundary
THEN Yogi's position should not change

The Yogi does not move out of the screen boundary on all sides, even on keep on pressing the movement button.



9. Random Picnic Basket Test:

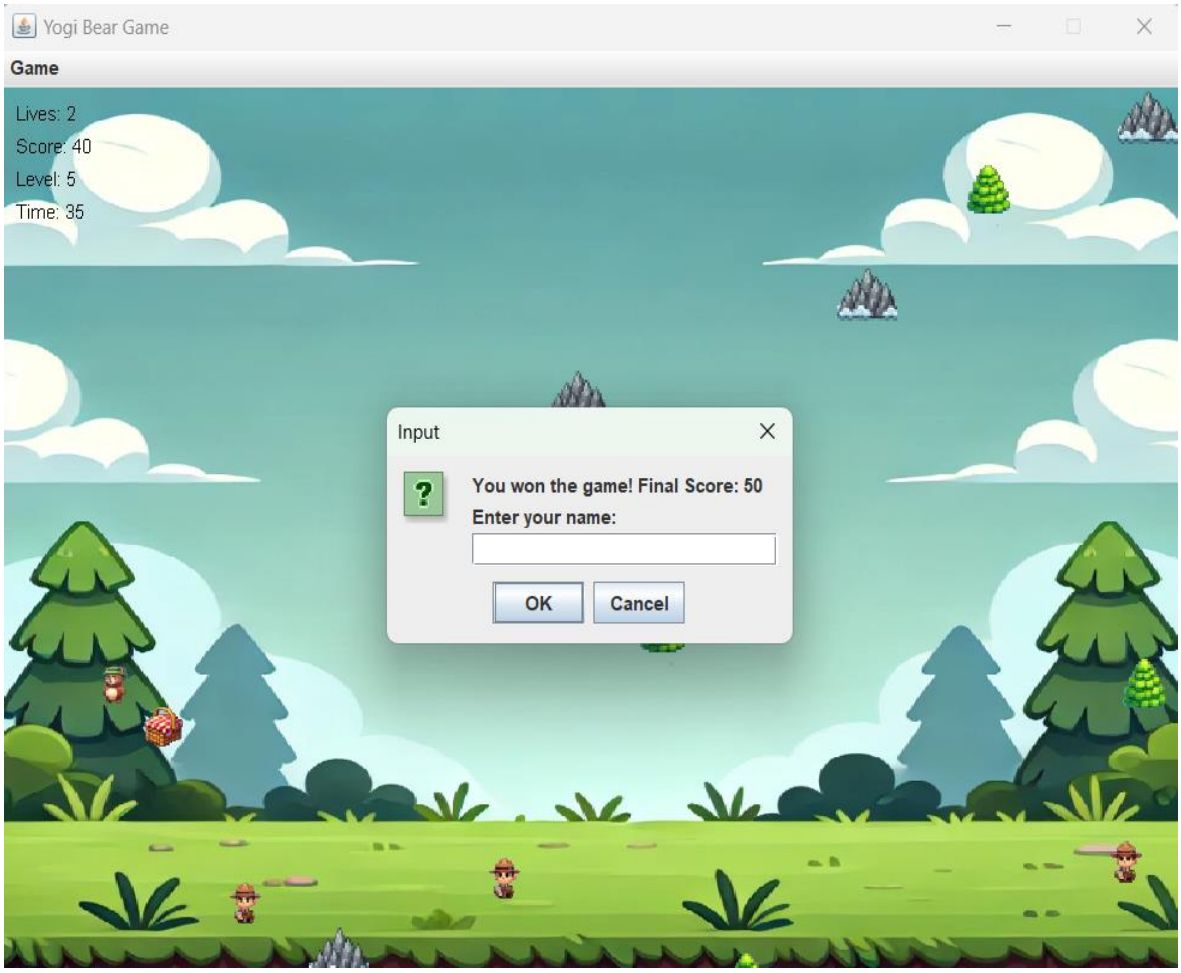
AS a player
I WANT TO see picnic baskets placed randomly at the start of the game
GIVEN the level starts
WHEN the game initializes the level
THEN picnic baskets should appear at random, valid location on the map



10. Win Game Test:

AS a player
I WANT TO win the game
GIVEN all levels are won
WHEN the player finishes the last level
THEN the win game message should come

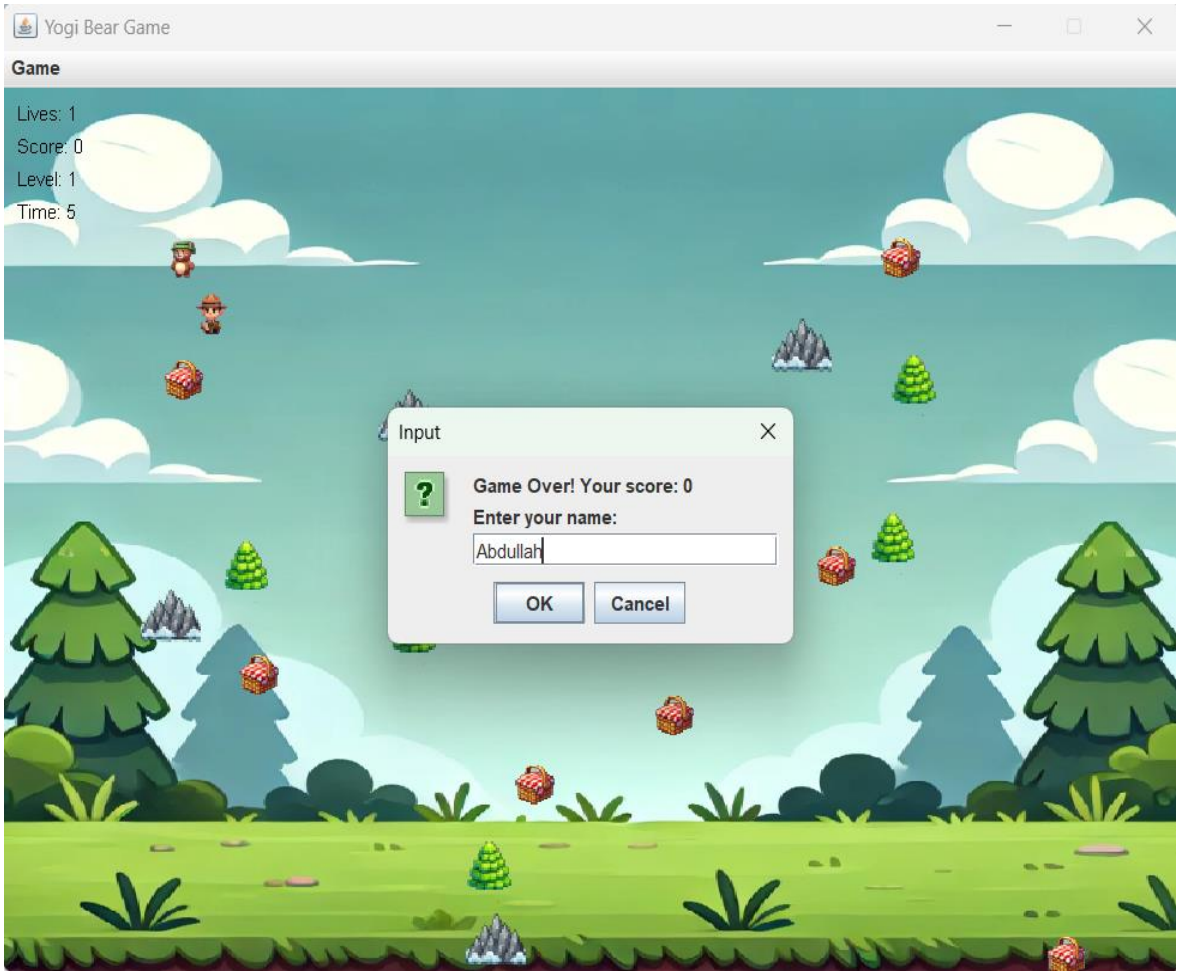
The player wins the game after finishing the required number of levels.



Black Box Testing:

1. High Score Saving Test:
   AS a developer
   I WANT TO ensure player scores are saved to the database
   GIVEN the game ends with a valid score
   WHEN the *saveHighScore()* method is called
   THEN the score should be stored in the database with the player's name

The name gets stored to the database here, when the game ends.

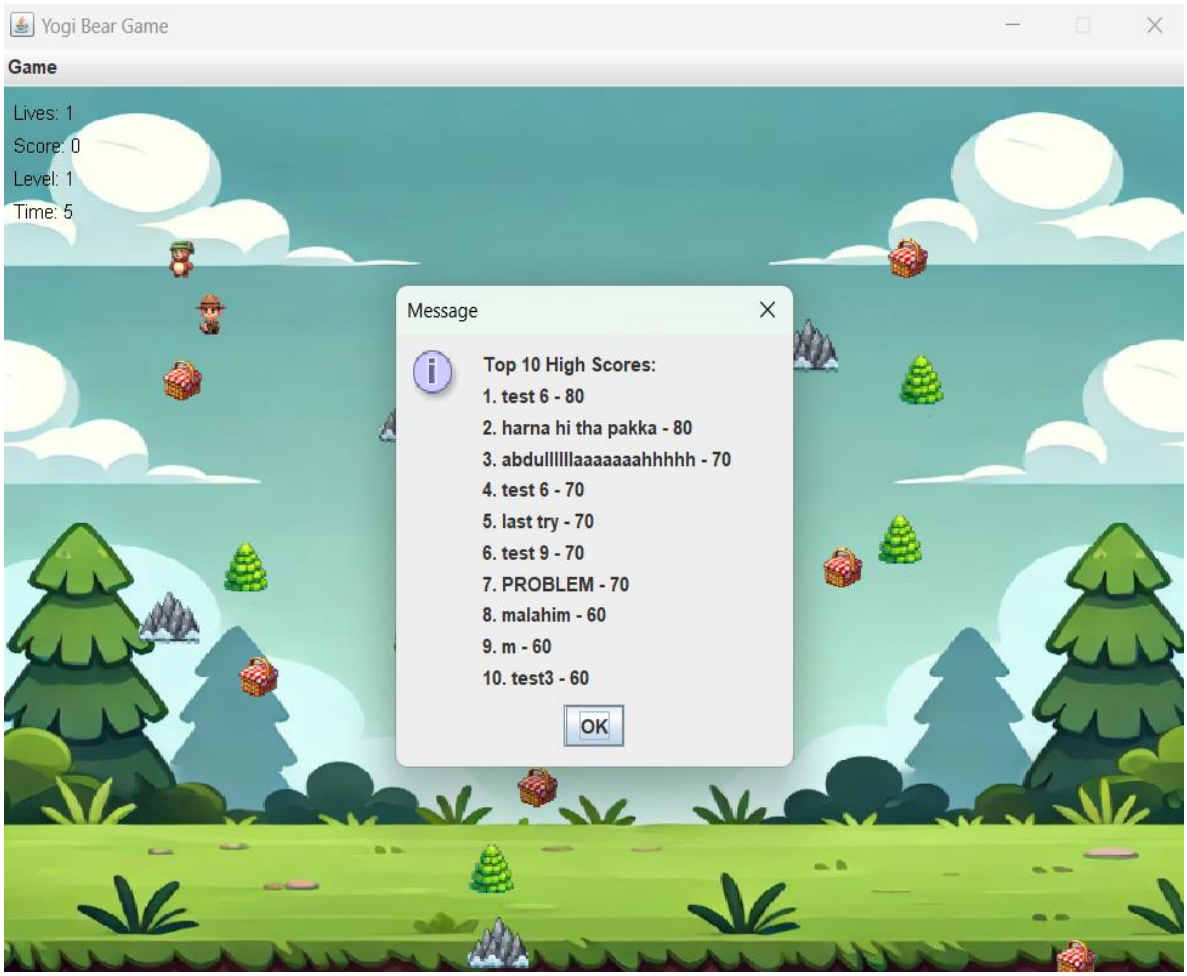


2. High Score Retrieval Test:

AS a developer
I WANT TO retrieve and display the top 10 high scores
GIVEN the game calls *getHighScores()*
WHEN the method executes
THEN the top 10 scores, ordered by rand, should display correctly

This window can be fetched from the dropdown menu at the top

of the screen, or when the game ends.



3. Picnic Basket Display Test:

AS a developer
I WANT TO display the remaining picnic baskets on the screen
GIVEN the player has not yet collected all jars
WHEN the game updates
THEN the count should decrease as baskets are collected

4. Level Load Test:

AS a developer
I WANT TO load level dynamically
GIVEN the player advances to the next level

WHEN the *loadlevel()* method is called
THEN the new level should load, with appropriate map layout and obstacles

5. Start New Game:
AS a developer
I WANT TO check whether there is an option to start a new game
GIVEN the *restartgame()* method is called
WHEN the games ends or button is pressed from the menu
THEN the game restarts, the timer, the level, the lives and the scores resets

6. Spawn Yogi:
AS a developer
I WANT TO check whether the yogi gets spawned at a valid position
GIVEN the *spawnYogi()* method is called
WHEN the life is lost, or a new game is started
THEN the yogi should get spawned at the starting position and not coincide with an obstacle or any Ranger.