Stick Hero Game

Group No: 64
Ashish Bargoti (2022114)
Aman Tomar (2022061)

This repository contains a simple implementation of the Stick Hero game using JavaFX. The game involves stretching a stick between two pillars to bridge a gap. The goal is to make the stick long enough for the character to cross the gap and reach the other side.

Features

- Stretch and elongate the stick using mouse events.
- Randomized pillars with different images.
- Hero falls if the stick length is less than the distance between pillars.
- Score tracking and display.
- Revival feature using cherries collected during gameplay.
- Background music during gameplay.

How to Play

1. Click and hold the mouse to stretch the stick.
2. Release the mouse to stop elongating the stick.
3. The hero will traverse the stick to cross the gap.
4. Score increases for successful crossings.
5. Collect cherries to earn revivals.

How to run:

- Unzip the folder
- Open the project in Intellij
- Go to src/main/java/com/example/demo/HomePage.java
- Adjust the relative path of the images and mp3 files if required
- Press run

Design Patterns Used -

1. Singleton -

   Player Class

   The Playerr class, residing within the `com.example.demo` package, plays a crucial role as a representation of a player entity within a game.

   Features

   Singleton Design Pattern

   The Playerr class is meticulously implemented following the Singleton Design Pattern. This distinctive pattern guarantees the existence of only one instance of the Player class throughout the entire application. The key method facilitating this design is `getInstance()`, which is responsible for returning the singular instance of the Player class.

   Usage

   Utilizing the Player class is a straightforward process. To access the singleton instance of the Player, simply make a call to `Player.getInstance()`. This method ensures that only one instance of the player is maintained within the application, providing a centralized point for player-related operations and data.

2. Factory -
   SceneFactory Class

This is the SceneFactory class. It is an implementation of the Factory design pattern and is used to create Scene objects based on a given type.

Features

- Factory Design Pattern: The `SceneFactory` class is implemented as a Factory. This means it provides a way to delegate the instantiation logic to child classes.

- **Scene Creation**: The `SceneFactory` class has a `getScene` method that creates and returns `Scene` objects based on the given type.

Methods

- `getScene(String type, ActionEvent event)`: Returns a `Scene` object of the given type. The type can be "Playground", "Playground1", or "Help". Any other type is considered invalid and will throw an `IllegalArgumentException`.

Usage

To use the `SceneFactory` class, create an instance of `SceneFactory` and call the `getScene` method with the desired type. For example:

```
SceneFactory sceneFactory = new SceneFactory();
Scene scene = sceneFactory.getScene("Playground", event);
```

Threads (used) -

We have used thread by making a Class Hero and run them in Controller class using thread.start().

We have used thread for taking / moving TheHero from 1 position to another in some cases.

Also when Hero falls into the abyss / through our reviving feature we brings the hero imageView using thread for 1 image and another thread for another image.

Like for TheHero - t1 thread
TheHero1 - t2 thread

InnerClasses -

We have also implemented inner Classes as in HomePage where we have made Scenefactory as inner class.

This helps in implementing Factory design method easily.

Game Screens

- Game Screen (game1.fxml): The main gameplay screen.
- Scorecard Screen (scorecard.fxml): Displayed after the game ends, showing the final score and option to play again.
- Entry Screen (game.fxml): Initial screen with a "Play" button to start the game.
- Exit Screen (exit.fxml): Displayed when the game is paused or exited.

Updates

- Added background music.
- Implemented falling of the hero when the stick length is insufficient.
- Added pillars to determine the required stick length.
- Score is displayed and updated during gameplay.
- Revival feature using collected cherries.
- Best score is not updated or stored yet.

.

Dependencies

- JavaFX for the graphical user interface.

Collaborators

This game was developed by Ashish Bargoti and Aman Tomar as a simple JavaFX project.

Have fun playing Stick Hero!