Dereje Tlahun

Due date 7/12/23

IS413

The Invader_Move class is an essential component of the Space Invaders game implementation. It extends the Rectangle class from JavaFX and represents both the player and the invader objects in the game.

The Invader_Move class includes the following key components:

1. Instance Variables:
    - dead (Boolean): Represents the state of the invader. It indicates whether the invader is dead or alive.
    - type (String): Stores the type of the invader.
2. Constructor:
    - The constructor takes parameters for the initial position (x and y), width, height, type, and color of the invader.
    - It calls the superclass (Rectangle) constructor with the width, height, and color parameters.
    - The constructor sets the initial translation (setTranslateX() and setTranslateY()) of the invader based on the provided position.
3. Movement Methods:
    - moveLeft(), moveRight(), and moveUp(): These methods update the invader's position by adjusting its translation (setTranslateX() and setTranslateY()). Each movement method moves the invader by a fixed amount in the corresponding direction.
4. Collision Detection:
    - intersects(Invader_Move other): This method checks if the current invader intersects with another Invader_Move object. It utilizes the getBoundsInParent() method to obtain the bounding box of each invader and checks if they intersect.

The Invader_Move class provides the necessary functionality for movement, collision detection, and state management of the player and invader objects in the Space Invaders game.

The SpaceInvadersModel class plays a crucial role in the Space Invaders game implementation. It represents the game's model, which encompasses the game state, logic, and behavior. Let's dive into the details of the SpaceInvadersModel class:

1. Instance Variables:
    - Pane root: This variable represents the root container of the game's graphical elements.
    - double t: It is a timer variable used to control the movement of the invader.

    - Invader_Move player: An instance of the Invader_Move class representing the player object.
    - Invader_Move invader: An instance of the Invader_Move class representing the invader object.
    - int invaderDirection: A variable indicating the direction of the invader's movement.
    - boolean gameOver: A flag to determine if the game is over.
2. Constructor:
    - The constructor initializes the instance variables of the SpaceInvadersModel class.
    - It creates a new Pane object as the root container.
    - The player and invader objects are instantiated with their initial positions, sizes, types, and colors.
    - The player and invader objects are added to the root pane.
    - The background of the root pane is set to a light blue color.
3. Methods:
    - Pane getRoot(): Returns the root pane of the game.
    - void update(): Updates the game state, including invader movement and collision detection.
    - void displayGameOverMessage(): Displays a "Game Over" message if the game is over.
    - void movePlayerLeft(): Moves the player object to the left if it is within the game boundaries.
    - void movePlayerRight(): Moves the player object to the right if it is within the game boundaries.
    - void shootPlayer(): Initiates a player's missile shooting action.
    - void startGameLoop(AtomicBoolean moveLeft, AtomicBoolean moveRight): Starts the game loop, which continuously updates the game state based on player input and triggers the display of the "Game Over" message when appropriate.
    - private void shoot(Invader_Move who): Handles the shooting action of a player or invader by creating and moving missiles, checking for collisions, and updating the game state accordingly.

The **SpaceInvadersModel** class acts as the core of the game, managing the game objects, updating their positions, detecting collisions, and controlling the game flow. It encapsulates the logic necessary for a functional and interactive Space Invaders game experience.

The SpaceInvadersController class is a crucial component of the Space Invaders game implementation. It serves as the entry point for the game and is responsible for initializing the game model (SpaceInvadersModel) and the game view (SpaceInvadersView).

Let's delve deeper into the responsibilities and functionality of the SpaceInvadersController class:

1. Main Method: The main method is the entry point of the Java program. It calls the launch method, which is provided by the Application class from JavaFX, to start the JavaFX application and invoke the start method.

2. Start Method: The start method is an override of the start method from the Application class. It receives a Stage object as a parameter, representing the main window of the application. Inside the start method, the following tasks are performed:
   - Set the title of the game window to "Space Invader Game" using the setTitle method of the Stage class.
   - Create an instance of the SpaceInvadersModel class, representing the game's model.
   - Create an instance of the SpaceInvadersView class, passing the Stage object and the SpaceInvadersModel object as parameters.
   - Call the initialize method on the SpaceInvadersView instance to set up and display the game view.

The SpaceInvadersController class effectively connects the game model and view, facilitating the smooth flow of data and interactions between the two. It sets up the game window, initializes the necessary components, and launches the game view. This class plays a vital role in orchestrating the overall structure of the Space Invaders game and ensuring its proper functioning.

The SpaceInvadersView class is responsible for managing the view aspect of the Space Invaders game. It handles the initialization of the game scene, captures user input, and starts the game loop for updating the game state and rendering changes on the screen.

The class has two main instance variables: stage and model. The stage variable represents the JavaFX Stage object, which is the top-level container for displaying the game window. The model variable is an instance of the SpaceInvadersModel class, which contains the game logic and state.

The initialize() method is the entry point for setting up the game view. It creates a new instance of the Scene class, passing the root pane from the model as a parameter. The root pane contains all the graphical elements of the game.

The method also sets up event handlers for user input. It creates two AtomicBoolean variables, moveLeft and moveRight, to track the state of the left and right arrow keys. When a key is pressed or released, the event handlers modify the values of these variables accordingly.

The model.startGameLoop(moveRight, moveLeft) method is called to start the game loop. It takes the moveRight and moveLeft variables as parameters. The game loop continuously updates the game state based on user input and renders the changes on the screen.

Finally, the scene is set on the stage using stage.setScene(scene), and the game window is displayed using stage.show().

Overall, the SpaceInvadersView class manages the initialization of the game scene, captures user input, and starts the game loop for updating and rendering the game state. It acts as the bridge between the graphical representation of the game and the underlying game logic in the SpaceInvadersModel.

Our implementation utilizes the JavaFX framework, taking advantage of its rich set of graphical and event handling capabilities. The game's visuals are created using `Rectangle` objects, with different colors representing the player, invader, and other game elements. The game loop, implemented using the `AnimationTimer` class, continuously updates the game state and renders the changes on the screen, ensuring smooth and responsive gameplay.

In conclusion, our JavaFX implementation of the Space Invaders game successfully recreates the essence of the original game. By leveraging the capabilities of JavaFX, we have created an engaging and visually appealing gaming experience. The combination of the `Invader_Move`, `SpaceInvadersController`, `SpaceInvadersModel`, and `SpaceInvadersView` classes forms a robust and modular structure that adheres to the principles of object-oriented design. This project serves as a testament to the versatility and power of JavaFX for game development. It provides a solid foundation for further enhancements and customization, enabling the addition of new features and gameplay elements. Overall, our Space Invaders implementation showcases our technical skills and passion for recreating beloved classics in a modern context.