Space Invaders in Java

Introduction and Problem Statement

This project is a simple implementation of the classic Space Invaders game, but in Java using Swing. It includes features such as player movement, shooting, alien movement, and a game over screen.

Some features

- Player movement (left and right)
- Player shooting
- Alien movement
- Alien shooting
- Collision detection
- High score display
- Game over screen

Requirements for playing

- Java Development Kit (JDK) 8 or higher
- Maven

Running the Game

1. **Compile the code:**
   - Run `mvn compile` to compile it.
2. **Run the main class:**
   - Execute the `run` file

Project Structure

- **src/**
  - **score.txt**: All scores.
  - **Images/**: Contains all the image resources used in the game.
    - `alien.png`
    - `rocket.png`
    - `GameOver.png`
    - `ikona.png`
    - `logo.png`
  - **Font/**: Contains the custom font used in the Main Menu.
    - `font.ttf`
  - **Audio/**: Contains the custom sounds used in game.
  - `SPACE INVADERS SHOOT - Gaming Sound Effects HD FREE NO Copyright.wav` **UnitTest/**: Contains tests for some classes.
- **src/main/java/**

- Game/**:Contains classes that makes the game .
  - **MainFrame.java**: The main frame that initializes the game (Main menu).
  - **MainMenuPanel.java**: The main menu panel with buttons to start the game, show the high score, show controls and quit the game.
  - **AlienShot.java**: Represents a shot fired by an alien.
  - **GameOver.java**: Frame displayed at the end of the game with options to return to the menu or quit.
  - **Sprite.java**: Base class for all moving objects in the game (aliens and player).
  - **Player.java**: Represents the player's character.
  - **Alien.java**: Represents an alien enemy.
  - **Shot.java**: Represents a shot fired by the player.
  - **GameOver.java** : Frame for game over.
- **MainMenu/**:Contains menu buttons, panel, and frame. Here is Main class too.
  - **HighScoreButton.java**: Button to display the high score from a text file.
  - **PlayButton.java**: Button to start the game.
  - **OptionsButton.java**: Button to show the game controls.
  - **ExitButton.java**: Button to quit the game.
  - **MenuButton.java**: Base class for menu buttons.
- **Handlers/**: Handlers for keyboard inputs, score writing and sound playing.
  - **KeyHandler.java**: Handles keyboard input.
  - **SoundManagers.java**: Handles sounds.
  - **ScoreManager.java**: Writes scores to score.txt.
- **vendor/**:Contains not my code.
  - **BasicBlocks.java**:Represents houses on game screen(GamePanel).

Description of Game Classes

Sprite Class

The `Sprite` class is the base class for all moving objects in the game. It contains common properties and methods.

Properties: - `x`, `y`: Position of the sprite. - `width`, `height`: Dimensions of the sprite. - `image`: Image representing the sprite. - `destroyed`: Boolean indicating if the sprite is destroyed.

Methods: - `draw(Graphics2D g)`: Draws the sprite on the screen. - `getBounds()`: Returns a rectangle bounding the sprite. - `checkCollision(int shotX, int shotY, int tileSize)`: Checks if the sprite collides with a shot.

Player Class

The `Player` class represents the player's character. It extends the `Sprite` class and adds specific properties and methods.

Properties: - `lives`: Number of lives the player has. - `playerSpeed`: Speed of the player's movement.

Methods: - `moveLeft(int speed)`: Moves the player to the left. - `moveRight(int speed, int maxWidth, int tileSize)`: Moves the player to the right. - `playerMoving(KeyHandler keyHandler, int panelWidth, int tileSize, int speed)`: Moves the player based on keyboard input.

Alien Class

The `Alien` class represents an alien enemy. It extends the `Sprite` class and adds specific properties and methods.

Properties: - `speed`: Speed of the alien's movement.

Methods: - `move(CopyOnWriteArrayList<Alien> aliens, int panelWidth)`: Moves all aliens and checks for direction change. - `update(Shot shot, int tileSize)`: Updates the alien's state based on collisions with shots. - `shoot()`: Creates and returns a new `AlienShot`. - `checkCollision(int shotX, int shotY, int tileSize)`: Checks for collisions with player shots.

Shot Class

The `Shot` class represents a shot fired by the player.

Properties: - `x`, `y`: Position of the shot. - `isShooting`: Boolean indicating if the shot is currently active.

Methods: - `draw(Graphics2D g, int width, int height)`: Draws the shot on the screen. - `shooting(KeyHandler keyHandler, Player player, int shotSpeed)`: Updates the shot's position based on player input.

The `AlienShot` class represents a shot fired by an alien. It extends the `Shot` class and adds specific methods.

Methods: - `move(int shotSpeed)`: Moves the shot down the screen.

GamePanel Class

Properties

- **originalTileSize**: The original size of a tile (16 pixels).
- **scale**: The scale factor for resizing the tiles (3).
- **tileSize**: The scaled size of a tile.
- **maxScreenCol**: The maximum number of columns on the screen (16).
- **maxScreenRow**: The maximum number of rows on the screen (12).

- **width**: The width of the game screen.
- **height**: The height of the game screen.
- **FPS**: Frames per second (60).
- **gameThread**: The main game thread.
- **keyHandler**: Handles keyboard inputs.
- **soundManager**: Manages game sounds.
- **isGameOver**: Indicates if the game is over.
- **shotSpeed**: The speed of the player's shot.
- **playerSpeed**: The speed of the player.
- **alienSpeed**: The speed of the aliens.
- **player**: The player's character.
- **shot**: The player's shot.
- **numberOfDestroyedAliens**: The number of destroyed aliens.
- **bb**: The basic blocks in the game.
- **aliens**: A list of aliens.
- **alienShots**: A list of shots fired by aliens.
- **playerX**: The initial X position of the player.
- **playerY**: The initial Y position of the player.
- **alienShotTimer**: Timer for alien shots.
- **score**: The player's score.
- **gameFrame**: The game frame.
- **scoreManager**: Manages the score system and stores high scores.

Methods

- **GamePanel()**: Constructor that initializes the game panel.
- **startGameThread()**: Starts the main game thread.
- **gameOver()**: Handles the game over state.
- **run()**: Main game loop.
- **update()**: Updates the game state.
- **checkBlockCollision(Shot shot)**: Checks collision between a shot and blocks.
- **checkBlockCollision(AlienShot alienShot)**: Checks collision between an alien shot and blocks.
- **createEnemies()**: Creates the initial set of aliens.
- **alienShoot()**: Handles alien shooting.
- **checkCollision(AlienShot alienShot, Player player)**: Checks collision between an alien shot and the player.
- **paintComponent(Graphics graphics)**: Renders the game components.

Usage

- **Start the game**: Compile and Run the `Run` File
- **Move the player**: Use the left/right arrow keys or A and D keys to move the player.
- **Shoot**: Press the spacebar to shoot.

- **View high score**: Click the "High Score" button in the main menu.
- **View controls**: Click the "Controls" button in the main menu.
- **Quit the game**: Click the "Quit" button in the main menu or on the game over screen.

Screenshots

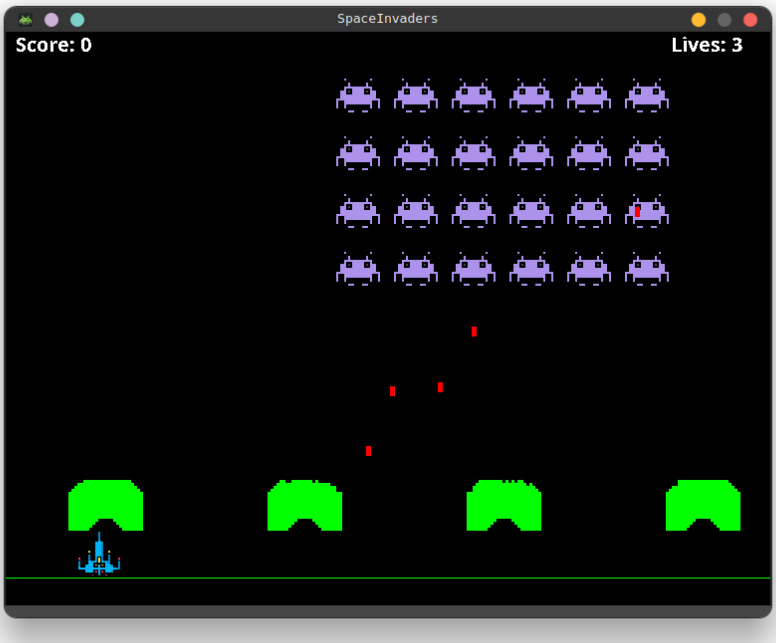

Figure 1: Game