Angry Birds Game Project

A LibGDX project inspired by the classic Angry Birds gameplay, featuring physics-based mechanics, diverse bird and pig types, multiple levels, and an immersive gaming experience. This project was generated using gdx-liftoff.

Game Features

- **Main Menu**: Navigate to start a new game, load saved games, or adjust settings.
- **Gameplay**: Launch birds with unique abilities to destroy obstacles and defeat pigs.
- **Level Progression**: Three levels of increasing complexity, featuring different birds and pigs.
- **Screens**: Includes distinct screens for Main Menu, Gameplay, Level Selection, Pause, Game Won, Defeated, Load Saved Games, and Settings.
- **Background Music**: Plays continuously across screens except during gameplay for an uninterrupted experience.
- **Custom Animations**: Includes particle effects for enhanced visuals and dynamic updates for in-game objects.

Game Elements

Birds

1. **Red Bird & Blue Bird**: Regular birds with standard speed and damage of 1 hit.
2. **Yellow Bird**: Deals 2 hits worth of damage and is faster than regular birds.
3. **Black Bird**: The most powerful bird, dealing 3 hits worth of damage with triple the speed of a Red Bird.

Pigs

1. **Basic Pig**: Standard Pig with 1 health point.
2. **Crown Pig**: Tougher Pig with 2 health points.
3. **Flying Pig**: Moves dynamically across the screen with 1 health point, protecting the main structure.

Blocks

- **Glass Blocks**: Weak, withstanding 1 hit.
- **Wooden Blocks**: Medium strength, withstanding 2 hits.
- **Cement Blocks**: Strong, withstanding 3 hits.

Physics & Collision Handling

Collisions are managed through custom contact listeners designed for each level:

- Updates the health of impacted bodies.
- Deactivates objects with zero health.
- Implements realistic physics for bird trajectories and obstacle destruction.

Saved Games and Serialization

The game allows users to save their progress across multiple runs.

- **Implementation**: A container class stores all game state data (e.g., body positions, health, bird status). This data is serialized using FileWriter and deserialized with FileReader.
- **Level-Based Saves**: Saved games are organized into separate files by level, allowing users to load the last saved state of each level.

Open-Level Design

Users have the freedom to select and play at any level without restrictions. There are no locked levels, allowing players to explore the game in any order they prefer.

Design Patterns

Factory Method Design Pattern in the Project

The project employs the Factory Method design pattern to create gameplay instances for two different modes: starting a new game and loading a saved game. Depending on the mode selected, different constructors are used to instantiate the gameplay class. This approach

centralizes the creation logic, ensuring flexibility in initializing the game and simplifying the process of managing and extending the game's initialization workflow.

Memento Design Pattern in the Project

The project utilizes the Memento design pattern to manage the game's state, enabling progress to be saved and resumed seamlessly. Each Memento object captures the current state of the game, including object positions, scores, and other relevant details. When reloading a saved game or undoing an action, the stored memento is used to restore the game state. This design pattern separates state management from the main game logic, ensuring a clean architecture while providing robust state persistence and rollback capabilities.

JUnit Testing

Unit tests were written to ensure the reliability and accuracy of core functionalities using the JUnit framework. Dependencies for JUnit testing were included in `build.gradle`, such as `junit-jupiter` and `mockito`. Key tests include:

- **Collision Handling**: Ensuring health reduction of birds and pigs upon collision.
- **Trajectory Calculation**: Validating the bird's movement based on physics formulas.
- **Bird Launch**: Testing the velocity of birds upon launch.
- **Health Updates**: Verifying multiple collisions reduce pig health accurately.

Example Code:

```
@Test
void health_reduces_when_collides() {
   // Simulate collision
   world.step(1 / 60f, 6, 2);
   contactListener.beginContact(createMockContact(birdBody, pigBody));

   assertEquals(0, bird.getHealth(), "Bird's health should reduce after
collision.");
   assertEquals(0, pig.getHealth(), "Pig's health should reduce after
collision.");
}
```

Bonus(expecting):

1. We have made a bonus bird (we name it “Sonic Bird”), this is the black bird in the Level-3, which has the velocity of approx 3 times more than a normal bird which makes its strength very high and lethal. Only Sonic bird is the one which can break the cement block in Level-3.
2. The “Floating Pig” in Level-3, This is a special pig which has feathers and can fly in the sky, the user have to aim at it perfectly to destroy it. For this, we are also updating its body positioning dynamically (when the texture moves up and down) for the collision to occur smoothly. We also used circular-shaped body instead of Box for this pig.
3. Used the “Particle Effect” feature of LibGdx for the animation of Black bird in Lvl-3(Sonic Bird) such that because of its velocity it looks like the bird has a nitro booster on its back, which increases the aesthetic look.
4. Incorporated the Sound of angry birds in some screens for a better experience.

Installation Instructions

1. **Clone the Repository**: Clone the project repository into your preferred IDE.
2. **Dependencies**: Install the LibGDX library.
3. **Run the Game**:
    - Open the project in your IDE.
    - Run **`AngryBirdsGame.java`** using the Lwjgl3Launcher.

File Structure

- **core**: Contains shared application logic across platforms.
- **lwjgl3**: Desktop platform implementation using LWJGL3.
- **Screens**: Each screen (e.g., MainMenuScreen, GamePlayScreen) has a dedicated class.
- **Assets**: Contains textures, sounds, and fonts.

Credits

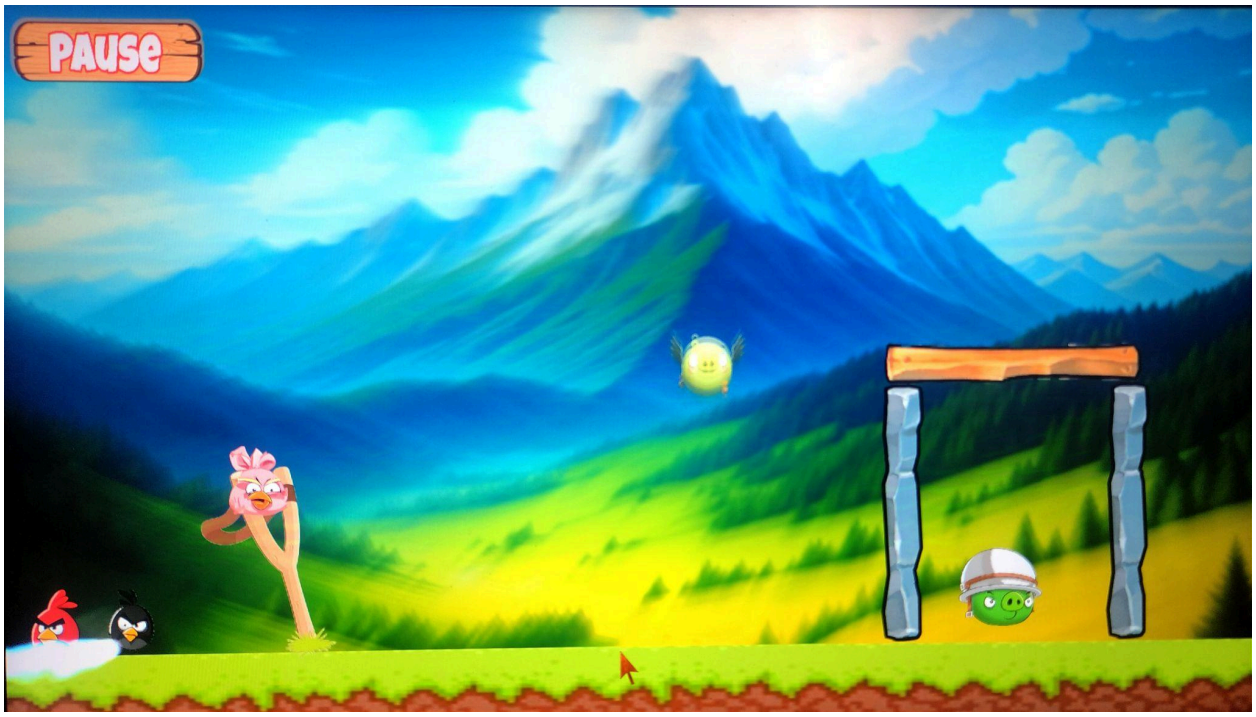
1. **LibGDX**: For the robust game development framework.
2. **Graphics & Sounds**: Sourced from public assets, enhanced with AI tools for quality.
3. Some online sources(like W3Schools etc) are used for developing the project. Like we used GeeksForGeeks for JUnit testing in IntelliJ using Mockito.

4. The **hovering effect of buttons**, this idea was taken from senior's projects. Implementation was ours.
5. **Contributors**:
    - Arpit Raj (2023132)
    - Madhav Maheshwari (2023304)

    Both of us equally contributed to this project.

MainMenu Screen:

GamePlayScreen3:



GitHub Repository: [Angry Birds Game Repository]

Thank you for playing!