Project Overview: Tank 1990 Game in Java Swing

I. Introduction

1. Objective

This project recreates the classic game *Tank 1990* using Java Swing. The primary goal is to build a structured game framework that includes key gameplay features such as player and enemy tank interactions, level progression, score management, and power-ups, while also exploring core Java Swing functionalities for UI, event handling, and animations.

2. Key Features and Components

- **Gameplay Management**: Central control for game state, level progression, scoring, and health.
- **Tank and Enemy AI**: Player-controlled tank, enemy spawning, and tank movement, with particular attention to enemy behavior and collision handling.
- **Power-Ups and Effects**: Includes features like pausing enemy movement or granting health boosts to the player.
- **Animations**: Effects for spawning, explosions, and bullet impacts, providing visual feedback to the player.
- **User Interface**: Menu navigation, level display, and score panels created with Swing components.

3. Challenges and Solutions

- **Tank Positioning and Collision:** Ensuring tanks spawn only on walkable areas, which involved debugging *NullPointerException* issues when tanks spawned in inaccessible areas.
- **Event Timing and Delays**: Synchronizing game start delays, power-up effects, and animations required precise timer management using *TimerManager* and custom delays.
- **Shared Timer Management**: Implementing a single shared timer across components to keep gameplay synchronized, handling pause/resume functionality and level reset.
- **Focus and Key Bindings**: Key controls for menu navigation and triggering actions with the Enter key were achieved by configuring key bindings for button focus traversal and event triggering.
- **Level Resetting and State Management**: Ensuring level progression resets properly without carrying over unintended states, which involves extensive checks in the *resetGame()* method.

4. **Related Work** **(Flappy bird with java swing)**

- **Scene Management**: In Flappy Bird, scene transitions (e.g., from the start menu to gameplay and game-over screens) are simple and linear. For Tank 1990, scene management expands, incorporating menus, level transitions, and more advanced control of in-game panels like scoreboards, pause screens, and level-specific content.
- **Physics**: In Flappy Bird, physics primarily involves the bird's gravity. jump forces and simple collision. Tank 1990, however, introduces more diverse interactions, such as collision detection between tanks and obstacles, bullet trajectories, and environmental effects like ice, where tanks slide.
- **GUI**: The experiences from Flappy Bird’s button handling and score display can inform the setup of Tank 1990’s user interface, such as health displays, level numbers, and dynamically updated scoreboards. This project would involve handling focus traversal across buttons (e.g., for pause, save, exit) and improving accessibility with keybindings for smoother control flow.

II. Methodologies

- **Modular Architecture**: Each game component, such as *TankManager*, *PowerUpsManager*, and *GameplayManager*, is developed as a separate module to handle specific functions like tank management, power-ups, and overall gameplay. This separation allows isolated testing and easier debugging.
- **Object-Oriented Design**: Key OOP principles like encapsulation (separate classes for *PlayerTank* and *EnemyTank*), inheritance (shared tank behaviors), and polymorphism (unique player/enemy actions) support reusability. Singleton and Factory patterns handle shared resources and standardized object creation.
- **Dynamic UI**: Scoreboards, health indicators, and power-up icons are dynamically updated during gameplay, with layouts designed for accessibility and quick information retrieval.

III. Implementation

- **OOP**: Object-oriented principles are fundamental to structuring the Tank 1990 game. For example, tanks, power-ups, and projectiles can each be represented as separate classes with properties and behaviors, utilizing encapsulation, inheritance, and polymorphism to create flexible and reusable components.
- **Java I/O**: Java I/O is useful for saving and loading game data, like high scores or level progress. File I/O allows the game to persist data between sessions, giving users a sense of continuity. For example, when the player selects "Save" on the pause menu, the game state could be written to a file.

- **Multi-threaded Programming**: Multithreading is essential for handling concurrent processes such as animations, game loops, and event handling. In Tank 1990, a separate thread could manage the main game loop while others manage individual tank behaviors or animations. This keeps the game responsive, even when multiple actions are happening at once.
- **GUI Programming**: The game's user interface is created using Java Swing, covering basic components (like JButton for game control) and more complex layout management. With Swing, various panels display scores, health, and power-up information dynamically, keeping the UI synchronized with gameplay.
- **Java Functional Programming**: Functional programming streamline code, especially in areas involving event handling and processing collections of data. In Tank 1990, lambda expressions and method references could simplify code for handling game events, filtering tanks, or updating power-ups. Functional programming also supports clean code practices, making the game code easier to maintain and extend.

IV. Conclusion

1. Key Learnings

- **Complexity of Timer-Based Gameplay**: Implementing smooth gameplay heavily relies on efficient timer management. Through trial and error, we learned that timers in Java Swing (like javax.swing.Timer) need to be used carefully to prevent frame drops, lag, and desynchronization of animations. Optimizing the timer management was essential to achieve consistent animation and fluid player movement.
- **Object-Oriented Design and Structure**: This project reinforced the importance of a modular object-oriented approach in game development. We utilized encapsulation and inheritance, structuring components like tanks, power-ups, and levels as distinct classes, which made managing interactions and changes much easier. This design choice allowed us to isolate and troubleshoot specific game features without affecting the entire codebase.
- **Multithreading Challenges**: We explored multi-threading to handle concurrent game events and animations, but balancing responsiveness with stability proved challenging. We learned the importance of handling concurrency issues carefully, particularly when multiple tanks and projectiles interact within the same game loop.
- **Data Persistence and Database Integration**: While we implemented basic data persistence through Java I/O, we realized the limitations of file-based storage for high scores and game state. This highlighted the value of database integration for more robust and scalable data management, especially for handling multiple player profiles or high score tracking in a centralized manner.
- **Debugging GUI and Animation Synchronization**: Developing a smooth, user-friendly GUI using Java Swing and ensuring that animations

synchronize well with gameplay were challenging but crucial steps. We learned how small adjustments to event handling, frame rates, and animations could significantly impact the player's experience, making debugging and testing essential.

2. Potential Future Work

To further enhance the gameplay experience the functionality of Tank 1990, we can focus on the following:

- **Smoothing Gameplay with Advanced Timer Techniques**: In future work, improving the smoothness of the gameplay will be a priority. This can involve exploring more advanced timing solutions, such as using the *ScheduledExecutorService* instead of *javax.swing.Timer*, or even looking into game engines that offer better built-in frame management.
- **Implementing Two-Player Mode**: Introducing a two-player mode will add a new layer of excitement and complexity. This would involve managing additional inputs, tank objects, and potentially new controls for each player, requiring careful design to prevent conflicts and ensure that the experience remains smooth and fair.
- **Database Integration for Game State Saving**: Adding a database to store player profiles, high scores, and save states would allow players to resume games seamlessly, track achievements, and compare scores. By using a relational database (such as MySQL or SQLite), we can provide a persistent, reliable storage solution that integrates well with the game's backend.
- **Enhanced AI for Enemy Tanks**: Developing a more sophisticated AI for enemy tanks could improve the challenge and engagement of gameplay. Implementing A* pathfinding algorithms or more complex behavior patterns could give enemy tanks smarter movement and responses based on the player's actions.
- **Optimizing Graphics and Animations**: Finally, optimizing graphics and animations would contribute to a more immersive experience. Transitioning to more efficient sprite handling and animation techniques (e.g., sprite sheets) would reduce memory usage and potentially allow for more detailed animations without compromising performance.

Contributions

Date	Activities	Team members
21st October, 2024	Implemented base code (project’s structure, classes,...)	Tran Trung Quoc
29th October, 2024	Fixed base code & added texture	Phan Thanh Hai
30th October, 2024	Added sidebars and pause panel	Vo Duy Tien
30th October, 2024	Added texture	Vo Duy Tien
3rd November, 2024	Player movement	Tran Trung Quoc
3rd November, 2024	Player & Enemy & Bullet classes	Phan Thanh Hai
4th November, 2024	Shooting function	Phan Thanh Hai
4th November, 2024	Load map from resources folder	Tran Tien Dat
4th November, 2024	Draw map	Vo Duy Tien
5th November, 2024	Movement Collision	Tran Tien Dat
5th November, 2024	Ice logic	Tran Trung Quoc
5th November, 2024	Add game border & change game size	Hoang Minh Duc
6th November, 2024	Powerups Manager	Tran Trung Quoc
7th November, 2024	Destroy tank animation	Tran Tien Dat
8th November, 2024	Complete sidebars & add save panel	Hoang Minh Duc
8th November, 2024	Scoreboard	Hoang Minh Duc
8th November, 2024	Test game	Hoang Minh Duc
9th November, 2024	Fix logic and finalize	Tran Trung Quoc