**FINAL REPORT**

**1. Introduction**

The "Tanks 1990" game is a modernized version of the classic arcade tank battle game, inspired by the popular 1990s game where players control tanks on various battlefields, facing off against enemy tanks. Our project aimed to develop a functional and engaging 2D tank game using Java and Java Swing while applying the core programming techniques learned throughout the course.

The primary objective of this project was to create a game that captures the essence of the original "Tanks 1990" while introducing several improvements in gameplay, performance, and user experience. The game includes multiple levels, player-controlled tanks, various obstacles, power-ups, and enemies. Through this, we sought to integrate important Java concepts, including multi-threading, object-oriented programming, collision detection, and graphical user interface (GUI) design.

The challenge our team addressed was the development of an interactive and responsive tank game within the constraints of Java, ensuring smooth gameplay, accurate collision detection, and a robust game environment. We also aimed to optimize the game’s performance to minimize lag and maximize the player’s experience, especially during the most intense gameplay moments. By tackling these challenges, we set out to create a game that is not only fun to play but also demonstrates the capabilities of Java for developing interactive 2D games.

**2. Related Work**

The development of games like "Tanks 1990" often involves integrating several important game development principles, such as artificial intelligence, game physics, multiplayer networking, and real-time rendering. Several research projects and game development studies have explored these areas, offering insights into best practices, algorithmic solutions, and optimization techniques that can be applied in games similar to "Tanks 1990." Many of these projects focus on the challenges of real-time gameplay, efficient resource management, and the interaction between game mechanics and player behavior. By analyzing the methodologies of existing works, our team could better address issues such as collision detection, object interaction, and optimizing game performance in our project.

**3. Methodology**

The methodology of the Tank 1990 game revolves around combining key gameplay mechanics like tank movement, shooting, enemy behavior, environmental interaction, and power-ups. The core of the game's functionality lies within the **GamePanel** class, which manages the game state and updates in real-time.

The **GamePanel** class extends JPanel and implements Runnable and KeyListener. It is the main UI panel responsible for rendering the game world, processing input for the player’s tank, and updating the game state in a continuous loop. The panel includes features like tank movement, enem, collision detection, power-up management, and game flow control (pausing, win conditions), with a custom game loop running at 60 frames per second (FPS) for smooth performance.

Developed in **Java** with **Swing** for the graphical user interface (GUI), the game uses JPanel for rendering and KeyListener for input. A custom game loop runs at 60 FPS via the Runnable interface in a separate thread, ensuring smooth updates. Game objects like tanks, bullets, and obstacles are managed as instances of custom classes, allowing for efficient handling and rendering.

The game world consists of various environmental objects (e.g., **BrickWalls**, **SteelWalls**, **Water**, **Ice**, **Trees**) and tanks. The player’s tank and enemy tanks are represented by PlayerTank and EnemyTank classes, with shared attributes for movement but different behaviors. Power-ups are randomly spawned by the **PowerUpSpawner** class, enhancing gameplay.

Player movement is controlled with the arrow keys (UP, DOWN, LEFT, RIGHT), each mapped to a boolean flag (isUp, isDown, isLeft, isRight). A momentum feature on **Ice** tiles lets tanks continue sliding for a few frames after releasing the key, implemented using a slideMomentum variable. The player can fire bullets by pressing **SPACE**, with a cooldown to prevent rapid firing. Enemy tanks move randomly, shooting at regular intervals, while changing direction upon collisions.

The game includes a pause feature activated by pressing **P**, which stops game updates and renders a transparent overlay. The game ends when all enemy tanks are destroyed, or if the player’s tank is destroyed, triggering a "Game Over" or "Victory" screen.

Collision detection is a vital part of maintaining gameplay balance. The **CollisionHandling** class checks interactions between tanks, environment objects, and bullets. Bullets trigger explosions when hitting destructible objects like walls, and tanks react accordingly. The game loop runs in a separate thread to ensure a steady FPS. The paintComponent() method handles rendering, and the pause overlay prevents further input while the game is paused.

The **PowerUpSpawner** randomly places power-ups for the player to collect, while the **EnemySpawner** manages enemy tank spawning and behavior. Each active enemy is processed in the game loop, with updates to its movement, shooting, and collisions.

In summary, the methodology of Tank 1990 integrates object-oriented principles with real-time mechanics to provide an engaging experience. With dedicated classes and methods for movement, collisions, enemy behavior, and environmental interaction, the game remains dynamic and responsive. The continuous game loop ensures smooth gameplay, while power-ups and environmental effects keep the experience fresh and exciting.

**4. Implementation**

During the development of the game, we applied various concepts from the course, including Object-Oriented Programming (OOP), GUI Programming, Collision Handling, State Management, Multithreading, and Timer functionality to build core components and ensure smooth gameplay.

* **Object-Oriented Programming (OOP)**: We used OOP principles to create the game’s core components, such as **Tank**, **PlayerTank**, **EnemyTank**, and **PowerUps**. Each of these components was designed as a separate class with specific attributes and methods, making the codebase more manageable and scalable. The **PlayerTank** and **EnemyTank** classes inherit from the base Tank class, while various power-ups provide players with temporary advantages, such as enhanced firepower or protection.
* **GUI Programming**: GUI Programming techniques enabled us to design an intuitive interface for the player, including components like the **MenuPanel** for mode selection, the **GamePanel** for displaying the map and updating game objects, and the **StatusPanel** to show score and remaining lives. We also incorporated **sound samples** to create sound effects for actions such as shooting, collisions, and collecting power-ups, enhancing the game’s immersive experience. Additionally, we added animations for effects like shooting, collisions, and power-up collection, making the game visually engaging as well as audibly interactive.
* **Collision Handling**: To manage interactions between objects, we implemented collision detection using distance calculations and bounding boxes. Collisions between bullets and obstacles trigger effects, such as breaking bricks or stopping bullets at steel walls, while collisions between tanks prevent them from overlapping. When the player collects power-ups, the system also handles collisions to activate benefits like enhanced power or temporary protection.
* **State Management**: We utilized a GameStateManager to manage the game’s states, including game start, pause, level completion, and victory screen. This state management ensures a smooth and organized flow, allowing seamless transitions between different stages and events.
* **Multithreading and Time**r: To improve performance, we used multithreading for heavy tasks like updating enemy positions and handling collisions, ensuring smooth gameplay without lagging. Additionally, we implemented a timer to update periodic events such as score, lives, and the number of remaining enemies every 100 milliseconds in the StatusPanel. The timer also manages the duration of power-ups and respawns enemies as needed.

By combining these techniques, our game delivers a smooth, interactive experience with visually engaging elements, providing players with an intuitive and enjoyable gameplay experience**.**

**5. Results**

Our project, Tanks 1990, achieved its primary objective of delivering a classic arcade-style tank battle game with engaging gameplay, smooth controls, and interactive visuals. Through extensive testing and iteration, we ensured that core mechanics like player movement, collision detection, and power-up activation worked reliably. Key outcomes include a responsive game interface, visually appealing animations, and sound effects that enhance the player experience.

However, we encountered challenges related to collision handling between multiple objects, particularly when managing simultaneous interactions of tanks, bullets, and power-ups. To address this, we refined our collision detection algorithm, implementing bounding boxes and specific collision zones for precise object interaction. Additionally, optimizing performance to prevent lag was a concern, which we managed by using multithreading to handle resource-intensive processes. These adjustments helped us maintain consistent gameplay quality, ultimately creating an enjoyable and smoothly functioning game that closely resembles the original Tanks 1990 experience.

**6. Conclusion**

The "Tanks 1990" project was a valuable learning experience that deepened our understanding of Java programming, object-oriented design, and real-time game development. Through this project, we enhanced our skills in managing multi-threaded processes, optimizing performance, and designing interactive GUIs. We also learned the importance of structured teamwork, efficient debugging, and continuous testing to refine gameplay and user experience. While we achieved our goal of creating an engaging and technically sound game, several areas present opportunities for future improvement, such as adding networked multiplayer, enhancing AI for more challenging enemy behavior, and expanding the game with additional levels and power-ups. Overall, this project not only allowed us to apply key Java techniques but also inspired ideas for future enhancements, setting a foundation for further exploration in game development.

**7. Contributions**

In our project, each team member took responsibility for key areas of development, ensuring smooth collaboration and comprehensive coverage of the game features. Minh and Đăng focused on the utility classes and UI panels, implementing essential game mechanics and designing the interface for player interaction. Chi managed the creation and functionality of the game environments, designing each element to enhance gameplay. Hung and Huyền worked on the power-up system, developing the different types of power-ups and their unique effects to add strategic depth.

Together, the combined efforts of Minh, Đăng, Chi, Hùng, and Huyền have culminated in a cohesive and immersive "Tanks 1990" gaming experience that seamlessly integrates gameplay mechanics with captivating visuals, setting a strong foundation for player enjoyment and engagement.