CSC 413 Tank Project Documentation

Semester 2023

Juan Estrada

Student ID: 923058731

-Summer-2023-R4

https://github.com/csc413-SFSU-Souza/csc413-tankgame-jjestrada2

Table of Contents

1 Introduction

1.1 Project Overview

The Tank Wars Game project is a big assignment for the CSC 413 course. It's all about making a fun tank game using Java. The main goal is to learn how to use Object-Oriented Programming (OOP) and to create a game that's exciting to play.

1.2 Technical Overview

In developing the Tank Wars Game, I delved into a range of technical aspects to bring the project to life. Here's an overview of the technical details involved in crafting this Java-based 2D tank battle game:

Programming Language and Framework:

I employed Java as the primary programming language for the game. Java's versatility and object-oriented nature lent themselves well to implementing the game's mechanics and functionalities.

Object-Oriented Design (OOD):

I embraced Object-Oriented Programming (OOP) principles to architect the game. This involved creating classes, interfaces, and abstract classes to model the various components of the game, such as tanks, bullets, power-ups, and the game world itself.

Game Loop:

I established a game loop to handle the continuous updating and rendering of game elements. This loop ensured that the game logic ran smoothly and consistently, resulting in seamless gameplay.

Player Controls:

Implementing player controls was a key technical aspect. I used event listeners to capture keyboard inputs, enabling tank movement, rotation, and shooting actions based on player interactions.

Collision Detection and Response:

Accurate collision detection between game entities, such as tanks and walls, was essential for a realistic gameplay experience. I employed algorithms to detect collisions and then handled the resulting interactions, such as ricocheting bullets or tank movements upon impact.

Animation and Graphics:

To bring the tanks and game environment to life, I utilized animation techniques. I integrated sprite sheets to create smooth animations for tank movement and rotation, as well as for various in-game actions.

User Interface (UI):

The game's user interface included elements such as health bars, life counters, and a mini-map. I used Java's graphical libraries to design and display these elements on the screen, enhancing the player's understanding of the game state.

Sound Effects:

Integrating sound effects contributed to the game's immersive quality. I used libraries to trigger sound clips for events like shooting, collisions, and power-up pickups, enhancing the overall player experience.

Game Mechanics and Logic:

I formulated the game's mechanics, including tank movement restrictions during rotation, power-up effects, bullet behavior, and scoring systems. The logical underpinnings of the game ensured balanced and engaging gameplay.

Building into a JAR File:

As a culminating step, I compiled the game into a Java Archive (JAR) file. This packaged the game's components, ensuring that it could be executed as a standalone application.

1.3 Summary of Work Completed

What I Achieved:

1. Creating a Class Diagram (The Blueprint):I drew a class diagram to plan how different parts of the game would fit together. This helped me organize my thoughts before writing the actual code.
2. Building the Game:I coded the game based on the class diagram. The game turned out to be quite exciting with various features.
3. Start and End: I added a starting screen and a screen showing the winner at the game's end.
4. Players in Action: The game supports two players, and their tanks can move, turn, and shoot.
5. Visual Appeal: Tanks move smoothly, and a mini-map shows the game's action.
6. Cool Extras: I introduced health bars for tanks and power-ups to enhance their abilities.
7. Obstacles: The game includes walls that some tanks can break and others can't.
8. Hits and Sounds: Bullets interact with walls and tanks, and I included sound effects for actions like shooting and grabbing power-ups.

9. Game Packaging: I converted the game into a special file called a JAR and placed it correctly in the GitHub repository.
10. Documentation and Presentation:I created detailed documentation explaining my design choices, coding methods, and how players can enjoy the game. Additionally, I prepared a brief presentation showcasing the game's features.

2 Development Environment

a) Java 20.0.1

b) IntelliJ

3 How to Build/Import your Project

Step 1: Cloning the Repository

- Clone this repository using Git.
- Open IntelliJ IDEA and click the "Open" button in the top right corner.
- Choose the folder named "csc413-tankgame-jjestrada2" from the cloned repository.

Step 2: Configuring Project Settings

- Open the "Project Structure" using Ctrl+Alt+Shift+S.
- In "SDK and Language Version," select a Java version that's at least Java 16.
- In "Project Settings," choose "Modules" and mark the "resources" folder as Resources.

Step 3: Building the Project

- Click the "Build" button at the top of the IDE (CTRL+F9).

Step 4: Creating a JAR Artifact

- Go back to the "Project Structure" window.
- Choose "Artifacts," then click the + button and select "JAR" > "Module with dependencies."
- Select "csc413-tankgame-jjestrada2" as the Module and "Launcher.java" as the Main Class.
- Build the JAR by clicking the "Build" button and selecting "Build Artifacts."

4 How to Run your Project

- Run the JAR by either double clicking the JAR
- OR typing into a console located in the same directory as the jar --> java -jar csc413-tankgame-jjestrada2.jar
- Run via IDE by hitting the play button or right clicking Launcher.java and select RUN

5 Assumption Made

Assumed World Boundaries: I could assume that the game area has fixed boundaries, making collision detection and movement logic simpler.

Assumed Bullet Speed: I might assume a constant speed for bullets, simplifying their movement and collision calculations.

Assumed Graphics Resources: I might assume that I can find graphics resources such as tank sprites, wall textures, and power-up icons for use in the game.

Assumed Single Level: I might assume that the game will have a single level instead of multiple levels, reducing the complexity of level design and transitions.

Assumed Win Condition: I could assume that the game ends when one tank's lives reach zero, making the win/loss conditions straightforward.

6 Implementation Discussion

6.1 Class Diagram

7 Project Reflection

The Tank Wars Game project wasn't just about creating a game; it was about exploring programming concepts, bringing virtual tanks to life, and sharing my achievements through documentation and presentation. It was both challenging and rewarding, offering an incredible learning experience and the satisfaction of building something truly enjoyable!

8 Project Conclusion/Results

The Tank Wars Game project was a big challenge where we learned to use special programming ideas and made a cool game in the process. We made tanks battle, added all sorts of things like power-ups and walls, and explained everything in our document and presentation. It was a fun way to learn and create something exciting at the same time!