SPŠE Ječná

Field of Study: Computer Science

Address: Praha 2, Ječná 30

Project Title: Tower Defense – A Java Strategy Game

Student: Ivo Faltus

Subject: Informační technologie

Year: 2025

School Name: SPŠE Ječná

Contents

---

1. Project Objective

The objective of this project is to create a tower defense game in Java. The player places towers to defend against waves of enemies on an expanding map. The game includes levels

of increasing difficulty, animations, sound effects, win/lose conditions, and strategic gameplay elements. Background music is included and can be muted in settings. Player can choose what difficulty he wants to play the ongoing wave on.

2. Game Description

2.1 Story/Algorithm

The player defends a base against attacking knights. Each level introduces a more complex map and more enemies. The player places towers at strategic positions to stop the knights from reaching the end of their path. The game evaluates whether all knights were stopped or if any reached the base, reducing the player's life count.

2.2 Characters

- ***Knight***: The enemy that follows a path on the map and reduces player's lives if it reaches the end.
- ***User***: Starts with set amount of towers, different within each level.
- ***Tower***: An automated defense unit that attacks nearby knights. Is able to kill 2 knights.

2.3 Mechanics

- Place towers by clicking on the map
- Towers automatically shoot enemies in range
- Display of player lives and number of remaining knights
- Enemies follow predefined paths based on level
- Game includes win and lose states
- GUI interface for game levels, controls, and outcomes

3. System Requirements

- ***Programming Language***: Java SE 17
- ***IDE***: IntelliJ IDEA
- ***Libraries***: Only standard Java libraries are used
- ***Execution***: Run `Main.java` to launch GUI
- ***Audio***: Utilizes Java Sound API for .wav playback

4. Basic Structure

- `***Main.java***`: Entry point, initializes the game
- `***Map.java***`: Defines level maps and enemy paths
- `**Tower.java**`: Defines tower behavior and shooting
- `**Knight.java**`: Represents the enemy logic
- `**Wave.java**`: Handles wave behavior and scheduling
- `**Menu.java**`: Displays main menu
- `**Player.java**`: Manages lives and player data

- `**Audio.java**`: Manages background music and sound effects
- `**ProgramToggle.java**`: Global flags and state
- `**UnitTests.java**`: Contains test cases for logic validation

4.1. Structure explained further

Firstly, the map for each wave is “drawn” in the Map class, after that method that creates bottom segment with buttons “place tower” “pick tower up” etc. The whole game flow starts when player clicks “Play” in the Main Menu. That causes execution of the wave method, which class the map draw method along with the method that executes the knights. Throughout the game flow, method that runs on a separate thread, watches whether the knight is adjacent to tower. If so, it kills the knight and damages the tower. It keeps looping through the ArrayList of knights, in case all of them are dead user wins. In case some of the XY coordinates of knight are equal to the finish coordinates a lost is announced.

5. Test Data

UnitTests.java contains unit tests for tower behavior, knight movement, tower-knight interaction, player life reduction, and wave progression. These cover typical and edge cases to ensure stable game logic. W/L logic cannot be tested since is too complex and requires various methods to be called.

6. User Manual

1. Run `Main.java` from an IDE or terminal
2. Choose a level from the menu
3. Click on map locations to place towers
4. Watch towers attack automatically
5. Prevent knights from reaching the end to avoid losing lives
6. Progress to the next level upon victory
7. A 'You Win' or 'Game Over' screen will show the outcome

7. Conclusion

It was necessary for me to apply all of the knowledge I’ve become aware of so far, therefore I was able to build the game, I would say, as good as I had imagined it initially. I learnt various java mechanics during the building, such as lambda expressions, swing timer and much more. I encountered many logic issues, all of which I eventually figured out on my

own. I didn't follow any tutorial whatsoever, therefore all the logic used is from the top of my head, which I am proud of. I am also glad for being able to divide the time effectively and there was no hurry at any juncture. But otherwise, the thing I honestly yearn to improve at is that, when some problem occurred I was not able to continue doing anything else in the project than solving it, which took solid amounts of time.

8. References

[1] Java SE 17 Documentation – [2] Java Swing UI Tutorial – [3] Java Sound API –