Secondary Technical School of Electrical Engineering Jecna 30, Prague, Czech Republic

Information Technology

Jecna 30, 121 36, Prague 2

Tower Defence Game

Dominik Pour
Information Technology
2024

Obsah

1 Introduction

The Tower Defense Game is a strategic game where players build and upgrade towers to defend their base from waves of enemies. The game features various types of towers and enemies.

2 Software

- Java Runtime Environment (JRE) 21 or higher
- Windows operation system

3 Game Mechanics

3.1 Towers

- Types of Towers:
  - **Cannon Tower:** A simple tower with moderate attack power and range.
  - **Ninja Tower:** A long-range tower with low attack power but fast firing rate.
  - **Mage Tower:** A short-range tower with high power bus slow firing rate.
- Upgrading Towers:
  - Towers can be upgraded to increase their attack power, range, and firing rate.
  - Each upgrade requires a certain amount of in-game currency.

3.2 Enemies

- Types of Enemies:
  - **Slime:** A basic enemy with moderate speed and health.
  - **Spider:** A fast enemy with low health.
  - **Worm:** A slow enemy with medium health.
  - **Snake**: An extra fast enemy with medium health.
- Waves:
  - Enemies come in waves, with each wave increasing in difficulty.
  - The goal is to survive as many waves as possible without letting enemies reach the base.

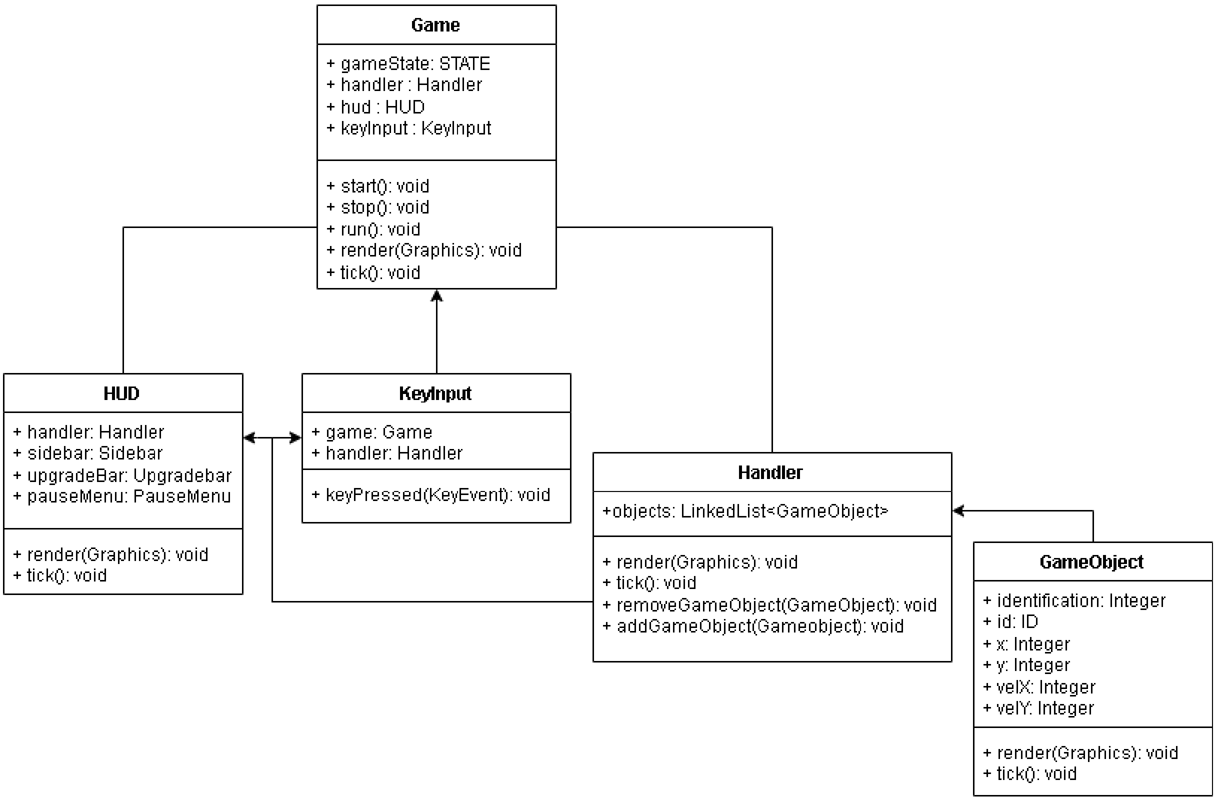
3.3 Maps

- Map Layouts:
  - Different maps offer various paths and strategic points for placing towers.
  - Each map has unique challenges and optimal strategies.

4 Code Structure

4.1 Key Classes

- **Game.java:**
  The main class that initializes and starts the game. It sets up the game window, initializes game components, and handles the main game loop.
- **Handler.java:**
  Manages all game objects, including updating their state and rendering them on the screen.
- **HUD.java:**
  Handles the display of game information such as health, coins, and wave counter. It also manages the pause menu and other UI elements.
- **GameObject.java:**
  The base class for all game objects, including towers, enemies, and projectiles. It defines common properties and methods that all game objects share.
- **KeyInput.java:**
  Handles keyboard input, allowing the player to interact with the game. It processes key presses and releases, enabling actions like pausing the game and controlling certain game elements.



4.2 Directory Structure

- **src/Game**: Contains the main game logic and entry point.

- **src/Game/HUD**: Contains the Heads-Up Display (HUD) classes and UI elements.
- **src/Game/Towers**: Contains the classes related to different tower types.
- **src/Game/Map**: Contains map-related classes and logic.
- **src/Game/Audio**: Contains the audio management classes.

5 Manual

5.1 Starting a Game

- **Select a Map:** Choose a map to play on.
- **Place Towers:** Use your initial coins to place towers strategically on the map.
- **Start the Wave:** Click the "Next Wave" button to begin defending against the first wave of enemies.

5.2 Tower Placement

- Click on a tower from the sidebar.
- Click on the map where you want to place the tower.
- Make sure the placement is strategic, covering paths where enemies will travel.

5.3 Defending Against Waves

- Each wave consists of a number of enemies that follow a path on the map.
- Towers automatically attack enemies within their range.
- Earn coins by defeating enemies.
- Use earned coins to place more towers or upgrade existing ones.

5.4 Upgrading Towers

- Click on a tower to open the upgrade menu.
- Spend coins to enhance tower attributes such as range, damage, and attack speed.

5.5 Winning and Losing

Winning

- Successfully defend against all waves of enemies.
- Keep your health above zero throughout the game.

Losing

- If enemies reach the end of the path, your health decreases.
- Lose the game if your health drops to zero.

6 Conclusion

The game successfully incorporates all planned features, including a variety of towers, enemies, and upgrade paths. There was a performance problem. Particularly with memory management and rendering efficiency. Also, during the early stages of development, the game encountered several bugs and glitches, particularly in handling user inputs and rendering graphics. These issues required significant debugging efforts.

7 Sources

7.1 Used libraries:

- **FlatLaf**: FlatLaf releases
- **Hamcrest Core**: Hamcrest Core
- **JOGG**: JOGG
- **JOrbis**: JOrbis
- **JUnit**: JUnit releases
- **LWJGL**: LWJGL releases
- **Slick2D**: Slick2D