TOWER DEFENSE: DESIGN AND IMPLEMENTATION OF A STRATEGIC DEFENSE GAME

Crăciun Maria Cătălina

*Faculty of Automation, Computers and Electronics, ISM 2.1 A, Craiova, România,
craciun.maria.i4e@student.ucv.ro

Abstract

This paper briefly describes the steps for developing a classic Tower Defense game using Java. The player must strategically place and upgrade towers to stop waves of enemies from reaching the exit. The game includes components such as the grid-based map, towers with varying attributes (range, damage, fire rate), and enemy units that follow a defined path. The game progresses through multiple waves of increasing difficulty and incorporates a scoring system along with resource-based upgrades. The application was implemented over several weeks using object-oriented programming principles, and demonstrates core aspects of game development including real-time interaction, animation, event handling, and GUI design.

Keywords

Tower defense; strategy game; Java; OOP; GUI; object-oriented programming; Swing; scoring system; game loop; wave mechanics

1. Overview

A Tower Defense game is a subgenre of strategy video games where players must stop enemies from reaching a designated goal by placing defensive towers along a path. The towers attack enemies automatically when they come into range. This project aimed to develop such a game in Java using Swing, based on principles of object-oriented programming, modular structure, and interactive graphics.

1.1 Game Tools and Structure

The project was developed using the Java programming language and the Swing library for GUI. The program consists of multiple classes distributed across packages such as 'Game' and 'entitati', which separate the logic of the game board, towers, enemies, and projectiles. A timer-based loop updates the game state every 30 milliseconds. The game runs inside a JFrame window and uses mouse events to handle user interaction for tower placement and upgrades.

1.2 System Architecture

The game follows a modular, object-oriented structure:

Main class (TowerDefenseGame.java):
Extends JPanel, manages the game loop, user input, and overall game state (towers, enemies, projectiles).

Core classes in "entitati" package:
• Enemy.java – defines enemy attributes and movement
• Tower.java – handles placement, upgrades, and shooting
• Proiectil.java – manages projectile behavior and damage

The architecture ensures clear separation between logic, UI, and game elements, making the code easier to manage and extend.

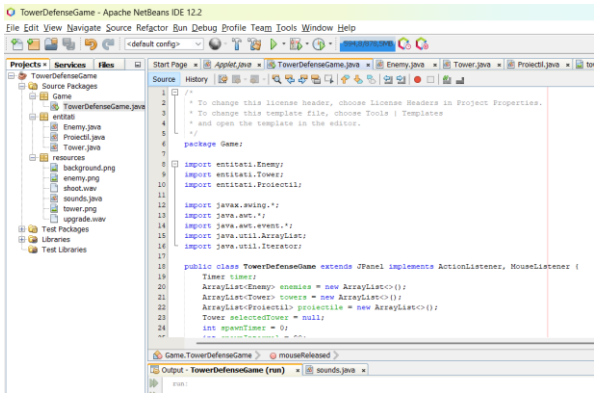

Figure. 1. System Architecture

1.3 Development and Information of the Application

The development of the Tower Defense game was carried out using the Java programming language, applying object-oriented programming principles to create a modular and maintainable structure. The application was built using the Swing library to design the graphical user interface and to handle user interactions in real time. A timer-based game loop updates the state of the game every 30 milliseconds, ensuring smooth animations and consistent gameplay.

• The application is composed of several interdependent components, each fulfilling a specific role:

• Enemies follow a predefined path and have properties such as health, speed, and reward points.

• Towers are placed by the player and automatically fire projectiles at enemies within a certain range.

• Projectiles are dynamically created and directed toward enemies, dealing damage upon impact.

• User Interface (UI) components include buttons for upgrading and buying towers, as well as visual indicators for score, wave number, and remaining lives.

1.4 Visual Assets

Background image (see Figure 2): A bright, grassy texture with subtle patterns was used to give the game a vibrant and friendly environment. It also serves as the base layer for the path and tower placement.

Figure. 2. Background image

Enemy design (see Figure 3): The enemy character is a cartoon-style creature with horns, designed to be both expressive and easily identifiable. It includes health points (HP) displayed above it and follows a predefined path from left to right.

Figure. 3. Enemy design

Tower design (see Figure 4): Towers are shown as stylized medieval structures. When upgraded, they are highlighted with a pink glow and shoot special projectiles to indicate increased power and range.

Figure. 4. Tower design

The visual feedback is enhanced with custom images and simple animations. For example, Figure 5 below shows an active game state where the player has several towers placed, enemies with 200 HP are approaching, and one upgraded tower is highlighted with a pink glow and special projectiles. The interface also displays the current score, wave, and lives.

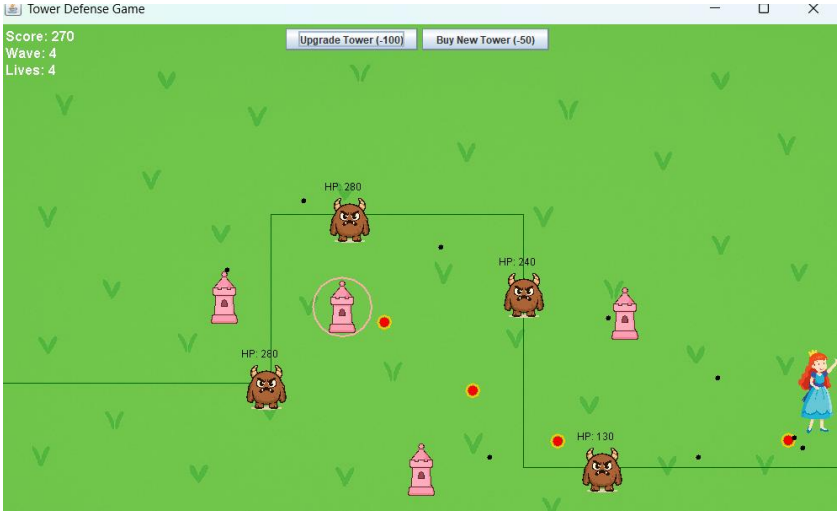

Figure.5. Game.

As the game progresses, the waves become increasingly difficult, with enemies gaining more health and speed. Figure 6 illustrates the Game Over state after the player has lost all lives during wave 5, while reaching a score of 460. The message is clearly displayed in red, and the user is given the option to restart the game from the interface.

Figure. 6. Game over!

These screenshots highlight key stages of the gameplay and demonstrate how the player must strategically manage resources and tower placements to survive increasingly challenging waves.

1.5 Audio Assets

Upgrade sound (upgrade.wav): This sound is played when a tower is successfully upgraded, offering satisfying feedback to the player.

Shooting sound (shoot.wav): Triggered each time a tower fires at an enemy, this sound effect makes the gameplay feel more dynamic and alive.

All assets were integrated into the game using relative paths and loaded via Java's ImageIcon and AudioSystem APIs. These resources contribute to a more immersive and polished experience, despite the game's relatively simple structure.

Equations:

The speed of an enemy increases with each wave and is calculated as:
$v = \min(3 + w * 0.3, 10)$ (1)
where *v* is the speed of the enemy and *w* is the current wave number.
The health of an enemy grows linearly with wave number:
$H = 100 + (w - 1) * 20$ (2)
where *H* is the health of the enemy and *w* is the wave number.
Projectile damage is increased when special effect is active:
$D = d + 10$ (3)
where *D* is the total damage and *d* is the base damage of the projectile.

2. Overview

• Towers: Placed by the player and fire at enemies within a defined radius. They can be upgraded for higher damage and longer range.
• Enemies: Spawn at fixed intervals and follow a predefined path. Each enemy has health points, speed, and a reward value when destroyed.
• Projectiles: Fired by towers to deal damage to enemies. Upgraded towers shoot projectiles with special visual effects and enhanced damage.
• Game UI: Displays score, wave number, and player lives. Provides buttons for buying and upgrading towers and restarting the game.

3. Game Development and Logic

The development followed a modular approach:
• Designing the main window and background image;
• Implementing the classes Tower, Enemy, and Projectile;
• Creating enemy pathfinding with waypoints;
• Managing tower placement and shooting mechanics;
• Implementing wave progression and score/life tracking;
• Adding interactive buttons (Buy, Upgrade, Restart) and visual feedback.

4. Interface Layouts

• Start Layout: Displays the initial tower and a background image, allowing the user to begin placing new towers using the Buy button.

• Game HUD: Shows real-time score, wave number, and remaining lives. Buttons are placed for game interaction, including tower upgrades and restarting after game over.

Conclusions

Through this project, I gained practical experience in designing game logic, handling animations and events, and organizing Java code using object-oriented principles. The Tower Defense game demonstrates how core programming and game development techniques can be applied to build a functional, interactive strategy game.

Acknowledgment

I would like to thank Professor Nicu George Bizdoacă for his patience and dedication during this semester. We learned a lot thanks to his support and clear explanations.

I also want to thank our laboratory teacher, Florina Petcu, for her professionalism and the way she guided us during the Java lab classes. Her help and involvement were very important throughout these months.

References:

1. ZetCode. Java 2D games tutorial. Available at: https://zetcode.com/tutorials/javagamestutorial/ [Accessed: 5 June 2025].
2. CodeGym. Learn Java by Creating Games. Available at: https://codegym.cc [Accessed: 5 June 2025].
3. Java Code Geeks. Java Game Development Articles. Available at: https://www.javacodegeeks.com [Accessed: 5 June 2025].
4. GameDev Academy. Java Game Programming Tutorials. Available at: https://gamedevacademy.org [Accessed: 5 June 2025].
5. YouTube – RyiSnow. Java Game Development Tutorial Series. Available at: https://www.youtube.com/@RyiSnow [Accessed: 5 June 2025].
6. Oracle. Java SE 8 Documentation. Available at: https://docs.oracle.com/javase/8/docs/ [Accessed: 5 June 2025].