ELEC-A7151 Course project documentation - “Bloonless Tower Defense”

900650 Nuutti Nykänen

793595 Elias Kauranen

591111 Patrik Ahola

Tino Korpelainen 896421

1. Overview

We created a tower defense game as a project for the Object-oriented programming with C++ course. In a tower defense game, a player is tasked with managing a set of purchasable towers that are placed on a map. These towers, different in nature, attack enemies that cross a predetermined path across the LevelMap. Some towers support friendly attacking towers.

The game starts with a map that contains a predetermined enemy path. For the player to start the actual gameplay, they are required to make a call for the game to launch a wave. This kickstarts an unstoppable wave of enemies that ends only when the wave’s enemies either are defeated or they get to the end of the enemy path. As stated before in the additional features, the player will be able to interact with the store and the map mid-wave (also between waves).

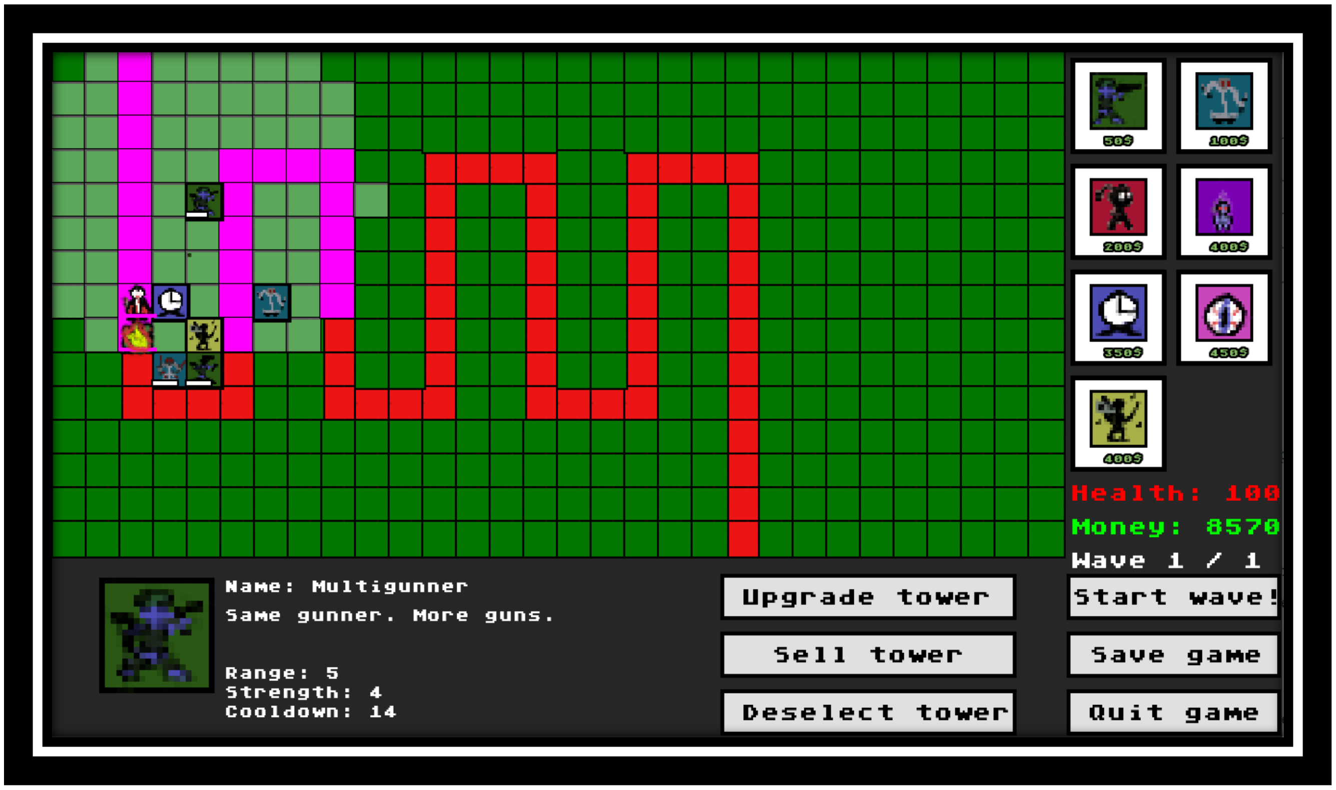
All interactivity is done via the GUI, via clickable buttons. The buttons’ functions are clearly stated by text inside them, for example “START WAVE”. The store is also image-based, which means the towers are displayed in a grid at the side of the GUI. The game ends when either the player wins or loses. This prompts the game to asks a username that it uses to record the player’s score to a text file for later inspection.

The game loads a map from a file. This file contains the non-branching enemy path and included enemy waves. Enemies differ in the sense that they each have their own speed, health and possible enemies inside them that pop out upon death. A Player has a money system and they can purchase towers from a store in the GUI. Towers can be purchased whenever, even with a wave in progress.

We implemented a total of seven different towers and their upgrades – the requirement was three. There’s an attacking tower that hits multiple enemies at once, another that teleports enemies back the path, a supporting tower that slows down enemies or reduces tower cooldowns etc.

Additional features we implemented on top of previously mentioned include non-hardcoded maps that can be modified within a file. Towers can be upgraded with a dedicated button and tower evolution lines consisting from up to three towers. Sound effects are implemented.

Additional features we decided upon but didn't implement include a tower/enemy weakness system and a high-score tracker.



```
Health: 100
Money: 8570
Wave 1 / 1

Name: Multigunner
Same gunner. More guns.

Range: 5
Strength: 4
Cooldown: 14
```

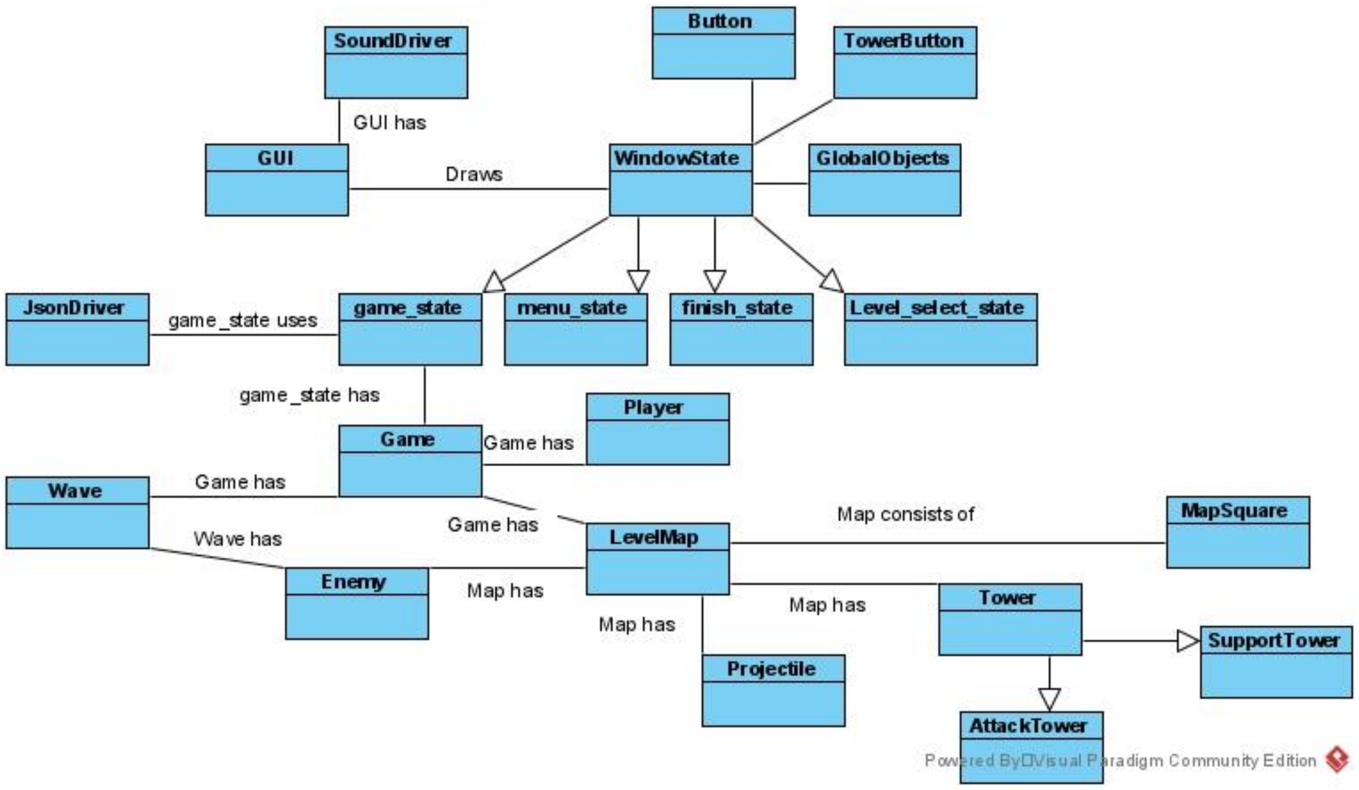
2. Software structure

The game has a GUI-class for visual presentation. The object of this class has a WindowState-object which determines the state that is drawn. WindowState has a child class for each state in the game (Menu, Level select, Game, Game finish...)

Game_state has the instructions for GUI to draw the game itself, when a level is played. It uses JsonDriver to load or generate a map, enemy waves or a save from JSON formatted data.

Game class is responsible for handling the game logic and updating its state. It does this mainly with the help of a LevelMap object. The levelmap consists of mapsquares that are either free, meaning that towers can be placed on them or enemy squares, in which case they are used by a moving enemy.

The Game uses levelmap to get information about all the towers, enemies and projectiles on the map, as well as to update them. When enemies pass the map or are killed, the player health and money are correspondingly updated by Game.



3. Instructions for building and using the software

Instructions for windows with CLion IDE (available as a free license to Aalto students):

- Download SFML: https://www.sfml-dev.org/files/SFML-2.5.1-windows-gcc-7.3.0-mingw-64-bit.zip
- Extract SFML-2.5.1 folder to C:\ and rename it to SFML

- Download MinGW 7.3.0 (64-bit): https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/Personal%20Builds/mingw-builds/7.3.0/threads-posix/seh/x86_64-7.3.0-release-posix-seh-rt_v5-rev0.7z/download
- Extract MinGw to C:\
- Install and open CLion
- From the welcome screen click "Get from vsc" and write to url: git@version.aalto.fi:cpp-autumun-2021/tower-defense/tower-defense-12.git
- (If you don't get the welcome screen close the current project from file -> close project)
- Start the project from upright panel

Errors:

- 0xC0000135 exit code
    - Download "libgcc_s_seh-1", "openal32", "libstdc++-6" extract them to the same directory as the .exe -file.

4. Testing

GUI testing was done almost exclusively visually due to it being the best way. Visual testing included creating a level map with dummy data and automatically adding different towers to it. Some GUI functionalities like event polling was tested using print commands and breakpoints.

LevelMap / Tower / Projectile interaction testing was done at the beginning with std::cout << std::endl prints. This proved to be an efficient demonstration of what was happening within a map as printing coordinates felt natural. After a GUI was implemented, the testing was much easier as gameplay could be seen in a natural, visual way.

The last week consisted heavily of testing and trying to break the game. This allowed us to eliminate most errors and the game runs correctly in its finished state.

5. Work log

First week:

Tino Korpelainen – 8 hours

- Added the SFML -library to the project
- Made a prototype SFML -window
- Added boilerplate Gui -class

Nuutti Nykänen & Elias Kauranen – 8 hours

- LevelMap class base laid out
    - Constructor
        - Initializes the map to be full of FreeSquares
        - Initializes the enemy path from given (edge) coordinates No validity checks yet
    - Get() any subclass of MapSquare
    - Can Get() enemies or a tower at given (x, y)
    - Can change MapSquares on the map
- MapSquare (abstract)
    - Subclasses FreeSquare, EnemySquare, TowerSquare
- Enemy
    - Base class and a few enemy subclasses
- Tower
    - Base class

Patrik Ahola - 6 hours

- Game class basic structure
- Definition of some basic functions and variables (mostly pseudo code)
    - Constructor
    - UpdateState-function to update the game's state
    - Run-function to run the game loop
    - SpawnEnemy, SpawnTower, StartWave

Second week:

Nuutti Nykänen & Elias Kauranen – 12 hours

- Enemy movement
    - works correctly, from start to finish
    - bidirectional linked-list enemy path
        - allows for "area of effect" damage (todo)
- Further tower implementation
    - placement on map
    - no attack function yet

    - subclasses SupportTower & AttackTower
        - subclasses not implemented
- Projectiles
    - movement on map towards an enemy target works
    - hits an enemy, damaging it
    - subclasses possible for different towers
    -

Tino Korpelainen – 7 hours

- Added a parent window state class from which different child classes extend
- Added child classes: GameState, FinishState, LevelSelectorState, MenuState
- Gui loop calls different states loops. State loops first advance the game state and then draw the updated game stat

Patrik Ahola 6 - hours

- Wave and player classes first versions
- Game class - SpawnEnemy, SpawnTower, Startwave functions.
- Game constructor initializes a player and waves.

Third week:

Patrik Ahola - 9 hours

- Further Game, player and wave class implementations and refactoring
- Logic for selling and buying player's towers and managing money
- Removing enemies from waves
- KillEnemy in Game class
- Added projectiles and the logic for buying and selling player's towers in game

Nuutti Nykänen & Elias Kauranen – 12 hours

- Further implementation and debugging of enemy movement and projectile interaction
- Different projectile types and effects
    - BombProjectile hits multiple enemies on hit
    - Easily can create more projectiles
- Different support towers

Tino Korpelainen – 6 hours

- Implemented working buttons, hit detection, global object, text
- Added a global font object
- Separated event polling to each of the states.
- Working state changes through buttons

Fourth week:

Tino Korpelainen – 9 hours

- Added temporary sprites to game objects
- Added buttons for starting wave, upgrade tower, sell tower
- Added buttons for buying different towers
- Bugfixes
- Joined the gui and the game

Nuutti Nykänen & Elias Kauranen – 8 hours

- Worked a little on GUI
- Overall debugging of LevelMap
- Validity checks for enemy path in LevelMap initialization

Patrik Ahola -

- Prototype UpdateState function
- Enemies move
- Attacktowers shoot
- Projectiles move
- Game refactoring and debugging

Fifth week:

Nuutti Nykänen - 28 hours

- GUI work
    - Enemies, towers and their ranges drawn on map
    - Sprites created and implemented
- SupportTower
    - virtual function implementation
    - SupportTower Clocker created, does not work correctly at the moment
- Projectile debugging and optimization
- Changing enemy speed temporarily with modifiers
- Rest of the SupportTowers
    - Clocker
    - DJDude
    - Seer
- Rest of the AttackTowers
    - KnifeBot
    - CursedKid
- Rest of the sprites
- GUI work
    - health bars

    - cooldown bars
    - drawing selected / buying candidate tower range
- Enemies within enemies
- General debugging

Elias Kauranen – 15 hours

- Made some changes in some classes (fixing bugs)
- GUI work
    - Tower shop
    - Player can buy towers
    - Player can sell towers
    - Player can upgrade towers
    - Fixing mistakes
- General refactoring

Patrik Ahola - 8 hours

- Game work
    - UpdateState: Enemies, towers, projectiles updated (With some help).
    - Enemy kill and damage detection (Might need fixing)
- Overall refactoring and fixing
- GameState
    - Player health and money drawn on screen

Tino Korpelainen 30 - hours

- Added the json library
- Added the level selector
- Added different json files for different levels
    - Different sizes, enemy paths and waves.
    - Each level is loaded from the json file.
- Prettied up the gui, small refactoring.
- General refactoring and removing duplicates
- Saving and loading active games.
- Prettying up the gui
- Added money icons to the tower selector
- Added tower selection with info and buttons
- Improvements to level select
- Added a finish state
- Made new levels
- Added infrastructure to add different sound effects