ELEC-A7151 fall 2023 Project Plan

Akseli Tuominen, Niilo Siren, Noel Nironen, Nandu Jagdish

Scope of Work

Our project involves the development of a tower defense game.

Additional Requirements

- **Upgradeable Towers:** Players will have the ability to upgrade their towers, enhancing their power and capabilities over time.
- **Variety of Enemies and Towers:** We will introduce multiple types of enemies and towers to create engaging gameplay and strategic depth.
- **High Score Tracking:** We will implement a high score system to track and display the best performances of players.
- **Sound Effects:** While optional, we may consider adding sound effects to enhance the gaming experience.

How the Program is Used

Users will interact with the program by following these general steps:

- **Install:** Players start by downloading SFML, which the graphics depend on. Library is installed using *sudo apt-get install libsfml-dev.* After this the code needs to be pulled from GitLab and opened in VSCODE
- **Launching the Game:** Players continue by launching the tower defense game on their computer by running the code
- **Main Menu:** Upon launching, the game presents a main menu with options for starting a new game, accessing high scores, adjusting settings (if available), and quitting the game.
- **Starting a New Game:** Players select the "New Game" option to begin playing. Player chooses if he wants to play by placing towers or running a wave of enemies. After this the player chooses the map he wants to play and the difficulty
- **Gameplay:** The gameplay is that of a classic tower defense game. Players places towers or runs the enemies through the map while managing his recourses
- **Winning and Losing:** The game ends if a balloon gets to the endpoint.
- **High Score Tracking:** The game keeps track of players' high scores, which are displayed on the high scores menu.

How the Program Works:

The tower defense game's operation is driven by a series of key components and processes:

- **Game Engine:** The core game engine manages the rendering, physics, and game logic. It handles the display of game elements, user input, and interactions.
- **Level Design:** The game utilizes level files that contain information about the map layout, including paths for enemy movement, tower placement zones, and the configuration of enemy waves.
- **Compile and Run:** The game is built using CMakelist

High-Level Structure

We are using C++ 17 and by using Cmake the game can be ran cross platform on windows or Linux. The software will be organized into the following main components:

- **Engine File:** The core engine of the game, responsible for rendering, physics, and game logic.
- **Geometry Files:** Handling geometric data for game objects and terrain.
- **Level Files:** Storing information about the game levels, including the map layout and enemy waves.
- **Object Folder:** A directory for managing game objects and their properties.
- **Map Folder:** Containing resources related to the game map.
- **Graphics Folder:** Storing graphical assets such as sprites, textures, and animations.
- **Main:** Used for initializing game engine and running game

Planned Use of Libraries

We will make use of the Simple and Fast Multimedia Library (SFML) to facilitate game development. SFML provides robust support for graphics, audio, and user input, which will significantly streamline our development process.

Division of Work and Responsibilities

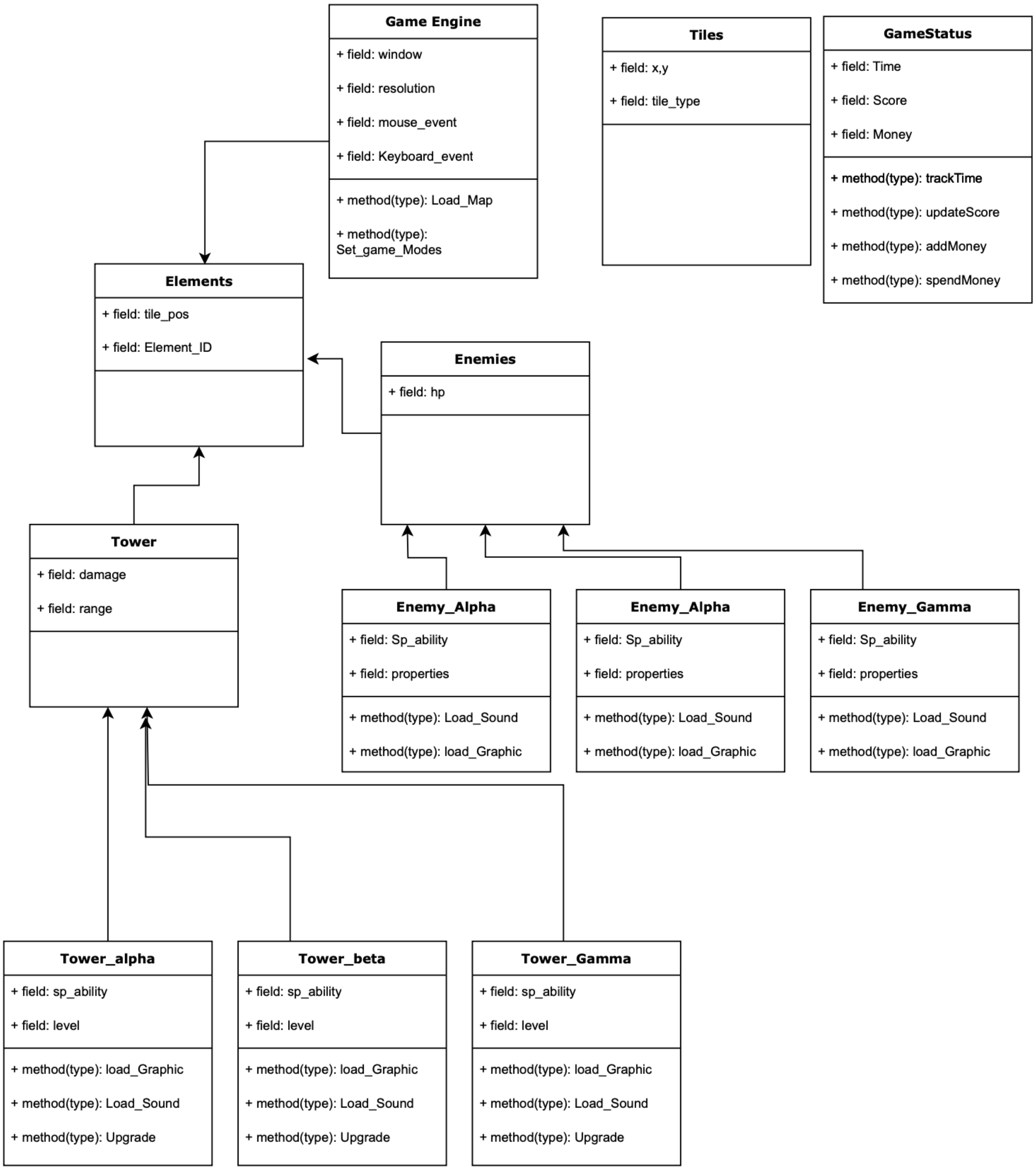
- **Game Logic and Engine Development:** Akseli
- **Graphics and Animation:** Noel
- **Level Design and Balancing:** Niilo
- **High Score System and sound Implementation:** Nandu

Planned Schedule and Milestones

We have established the following schedule and milestones:

- **Week 1:** Project planning and initial design phase
- **Week 2:** Game engine development and basic graphics
- **Week 3-4:** Implementation of tower and enemy mechanics
- **Week 5:** Level design, balancing, and high score system
- **Week 6:** Final polish, sound integration (if applicable), and testing
- **Week 7:** Bug fixing, optimization, and final testing
- **Week 8:** Submission of the final project

Each milestone will be reviewed and assessed, with the flexibility to adjust as necessary to meet the project's objectives. We aim for at least the grade 3


Game Engine
+ field: window
+ field: resolution
+ field: mouse_event
+ field: Keyboard_event
+ method(type): Load_Map
+ method(type): Set_game_Modes
Tiles
+ field: x,y
+ field: tile_type
GameStatus
+ field: Time
+ field: Score
+ field: Money
+ method(type): trackTime
+ method(type): updateScore
+ method(type): addMoney
+ method(type): spendMoney
Elements
+ field: tile_pos
+ field: Element_ID
Enemies
+ field: hp
Tower
+ field: damage
+ field: range
Enemy_Alpha
+ field: Sp_ability
+ field: properties
+ method(type): Load_Sound
+ method(type): load_Graphic
Enemy_Alpha
+ field: Sp_ability
+ field: properties
+ method(type): Load_Sound
+ method(type): load_Graphic
Enemy_Gamma
+ field: Sp_ability
+ field: properties
+ method(type): Load_Sound
+ method(type): load_Graphic
Tower_alpha
+ field: sp_ability
+ field: level
+ method(type): load_Graphic
+ method(type): Load_Sound
+ method(type): Upgrade
Tower_beta
+ field: sp_ability
+ field: level
+ method(type): load_Graphic
+ method(type): Load_Sound
+ method(type): Upgrade
Tower_Gamma
+ field: sp_ability
+ field: level
+ method(type): load_Graphic
+ method(type): Load_Sound
+ method(type): Upgrade