This is the general plan for Tower Defence project.

The project has been divided into sub-tasks, which are discussed below.
The interplay between each part is illustrated in the flow chart.

Game

The top level of the application is *game*-entity that takes care of updating the game loop (we'll return to this later), and contains map, game statistics and menus.

Map

Each game level is read from an ASCII-file and a grid-based *map* is initialized on the basis of this. Each tile of the grid are either part of the enemies' path, or buildable / unbuildable terrain.
The map is also responsible for keeping and updating the containers for enemies and towers.

Path

The enemies' path always begins from the top of the map, and an enemy at a certain tile knows the next tile by reading the direction marking on its current tile.
Two enemies can occupy the same tile without collision.

Tower

Each tower has attack attributes *damage*, *speed* and *range*. They also have attributes such as *location on map, cost to build, cost to upgrade, health* and *cost to repair*.
There are different tiers of towers with increasing effectiveness and cost. An upgraded tower can be either straight-up bought, or upgraded from an existing lower-tier one.

Towers are damaged by the enemies.
When tower's health reaches zero, it stops shooting until repaired.

Towers start shooting the first enemy that comes within their range and keep on shooting the same target until it leaves the shooting range. Then the next target is picked. (On the basis of the distance they've traveled, maybe?)
The tower calculates its distance to enemies by polling their location from the map.

Enemy

There are different types of enemies with varying *speed* and *health*.
When it is attacked by a tower, it's *health* decreases gradually to zero, at which point the enemy disappears and the player is awarded *money* and *points*.

Game loop

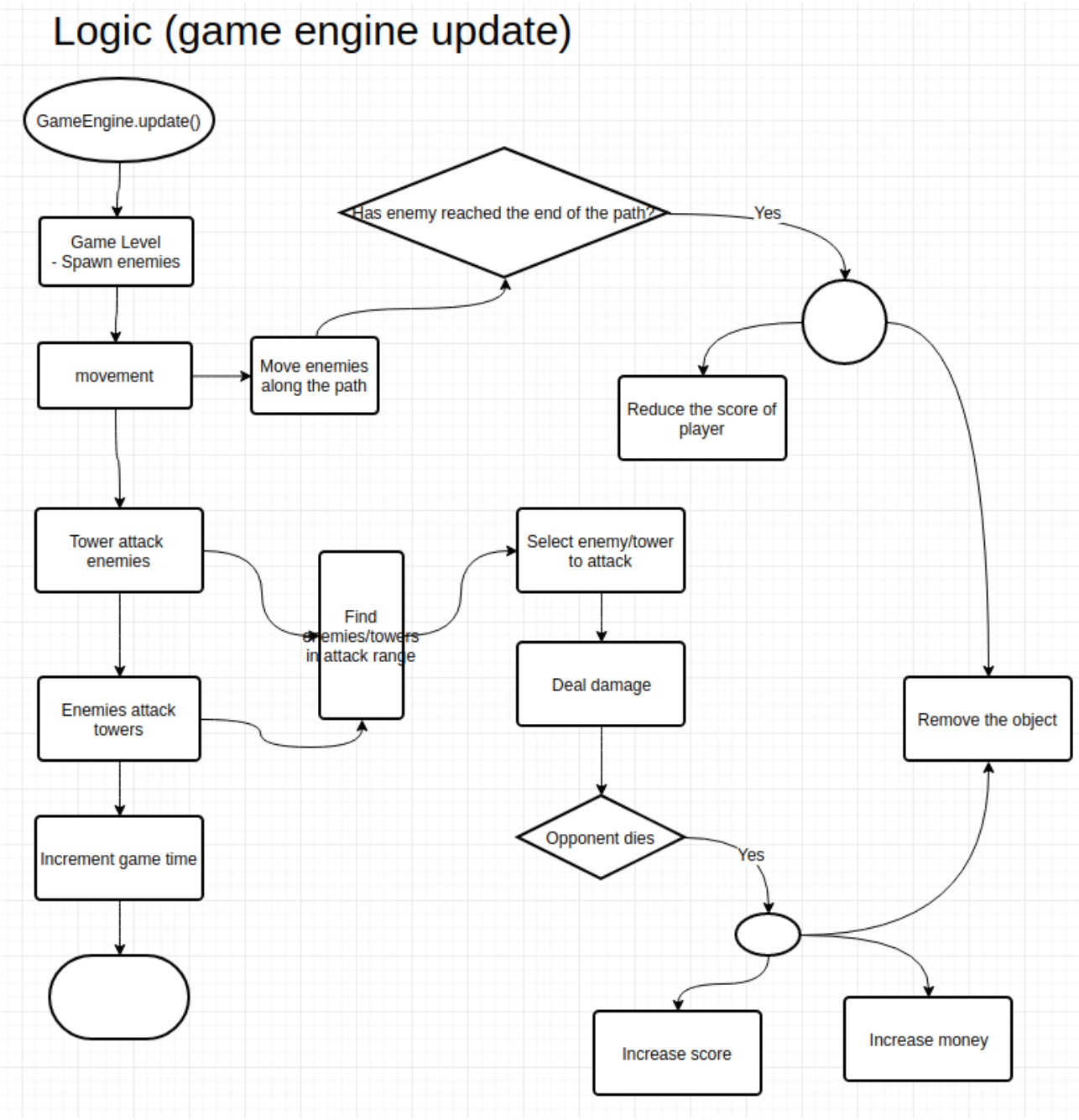
The game progression and possible user inputs are taken care of by a game-level loop that advances the game time, which in turn makes the units move.
In the meantime also the user inputs (mouse clicks) are listened to, and the units are updated / repaired / etc. accordingly.

Graphics

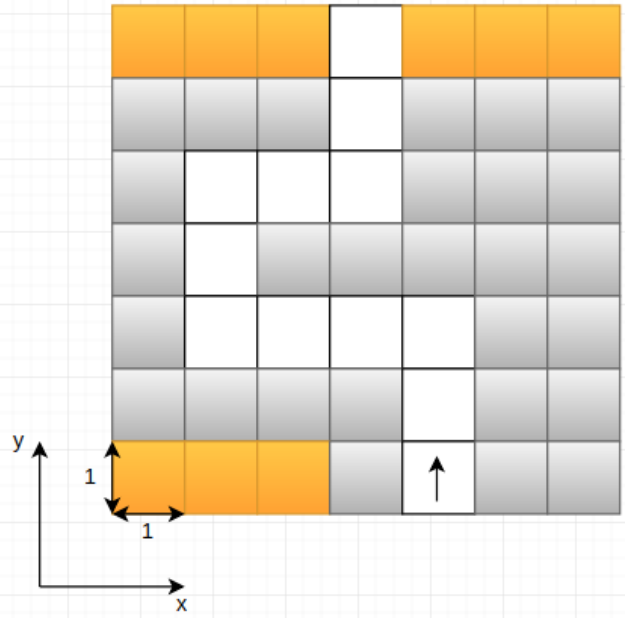
There are at least two screens: The title screen for choosing the level and starting the game, and the game screen where the game is played.
Graphics part of the game is also responsible for obtaining the user input in form of positions clicked, which are then translated to game controls.

The graphics are updated on the basis of info in *map* and *game* each loop.
The graphics are 2D, and they're created with FSML. Primitives are used for initial graphics, but possibility for sprites is explored later.

<u>Below graphs and sketches from the planning session of the project</u>



Game Map

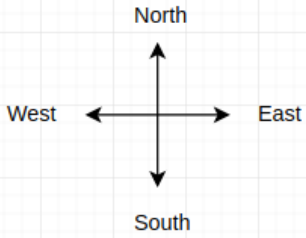


Tiles

Path that enemies follow

Tower can be places on these tiles

Nothing can be places on these tiles



Objects

- Game Engine
- Game Level
- Game Map
- Tile
- Towers
- Enemies
- GameStats

Graphics

1) Game engine update

2) Update graphics

Reponds to user input such as placing new towers, upgrading tower etc