TDA367 System Design Document

Smurfs vs. Gargamel

Group 9

Introduction

This System Design Document describes the architecture and the design of Smurfs vs. Gargamel. The document is intended to serve as a guide for the development team to ensure consistency in implementation and make sure the project is adjusted for future enhancements of the system. It provides a overview of the design through class diagrams, sequence diagrams, and state diagrams.

Smurfs vs. Gargamel is a tower-defense game where players strategically place smurf units to defend against waves of Gargamel's forces. The game emphasizes resource management, fun and innovative units, together with increasingly challenging levels.

The goal with the project is to provide an engaging and strategic gameplay experience, with the benefit of introducing players into the smurf universe.

System architecture

The system uses a modular, object-oriented design with several high-level modules. The main module for game logic is the `Model`. Within the `Model` module, there is information about where all sprites currently are together with their projectiles, and the functions to add new sprites, such as attackers or defenders. `Model` makes sure to avoid unnecessary complexity by using several different managers for different responsibilities.

Another module is `Board`, containing information about the different lanes with their respective cells. There also a module for the panels, together with one for the renderers. The panels store input recievers, notifying `Model` about specific inputs from the user.

High-level Architecture

A high-level architecture can be diagrammed like this:

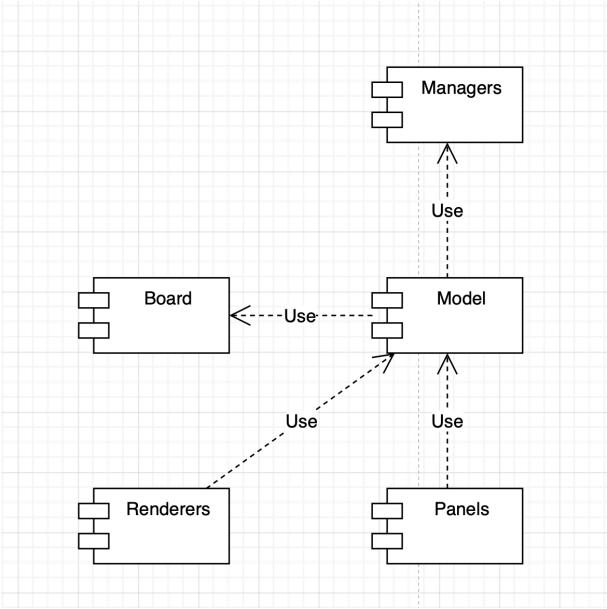

Figure 1: High-Level Architecture of the System

Detailed design

The high-level diagram may be decomposed into smaller, more detailed diagrams. Starting off with the way the app is launched:

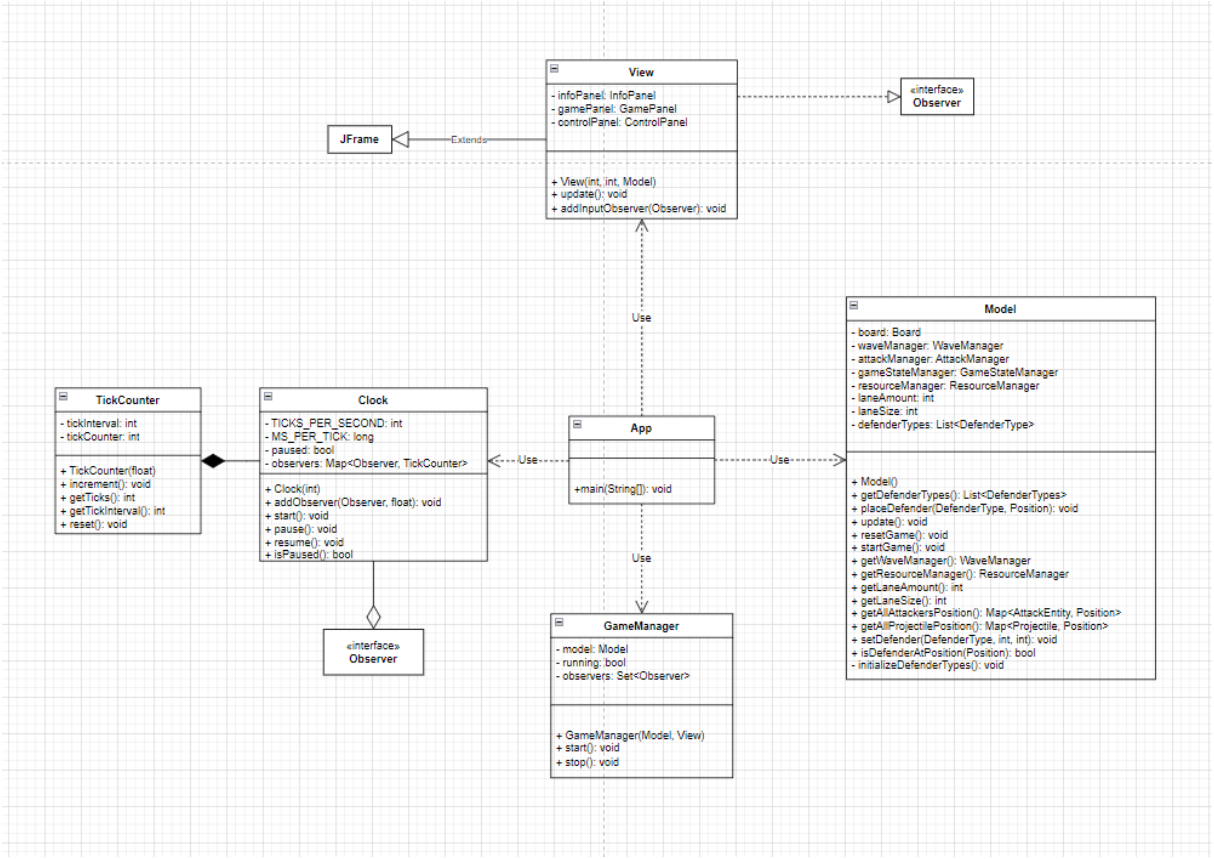

Figure 2: Starting the App

The view should initialize the different panels and the respective renderers, together with the controller. Decomposing the process into a class diagram yields:

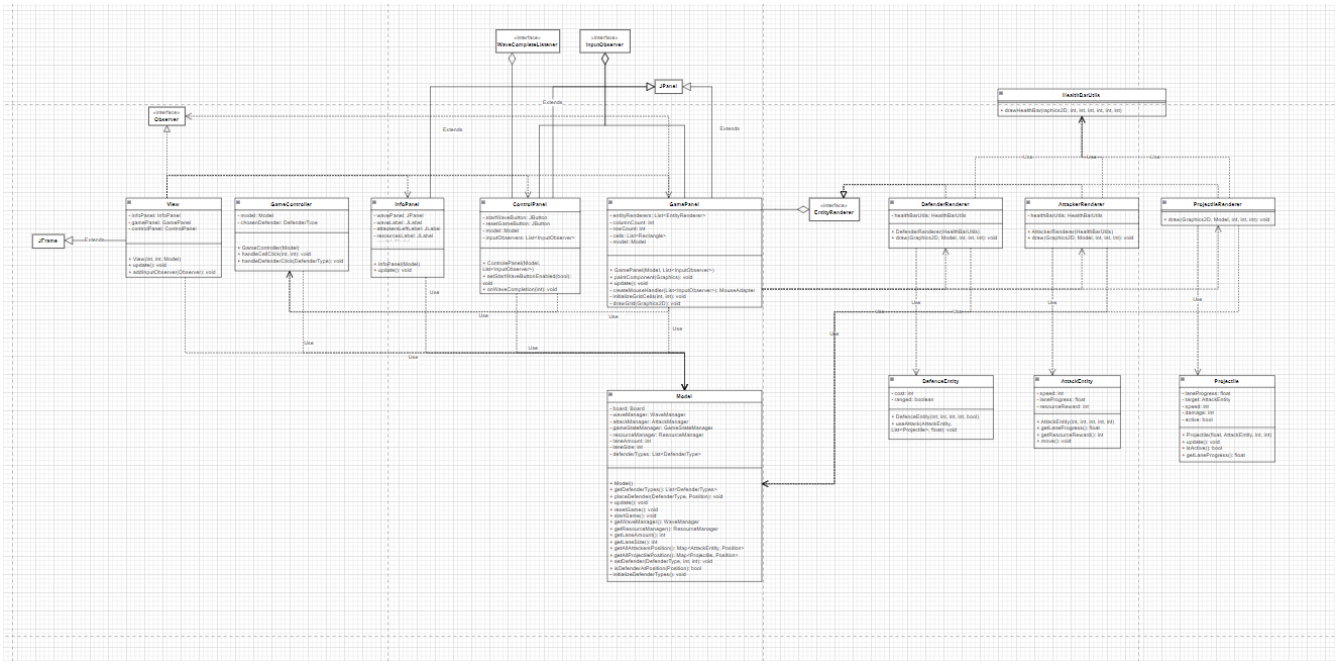

Figure 3: Initializing the View

The final part to showcase is the model, and its communication with the board, the managers, and the entities. The diagram looks like:

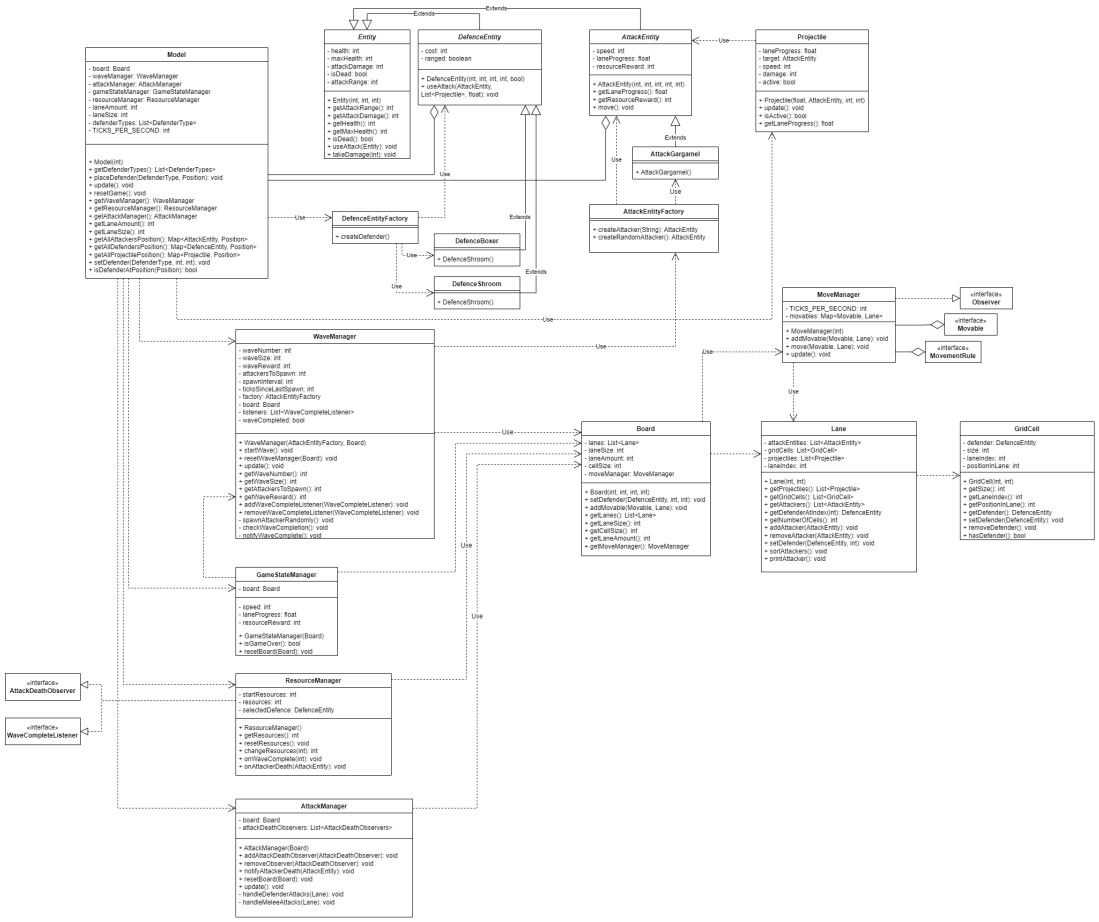

Figure 4: The model with dependencies