A Game That Needs A Name

# Basic Architecture

The game follows a basic layered architecture using Services to interface the Domain Logic this being split into various systems. The Services that interface the different Systems are used to simplify their access and help developers interact whit each other. Subsystems can use the interfaces as well as the UI. The communication of internal state of subsystems is done using views and not directly.

# Directory mANAGEMENT

**GameModels**

Modelation software files containing the game assets.

**Documentation**

Documentation for the project.

**UnityProject**

Unity project files.

# Organization IN UNity Editor

The file organization of the game follows the following scheme:

**ReadyToInstatiate**

GameObjects with components added and ready to be instantiated into the game

**Tests**

Test Scripts that can be added to test specific functionalities

**UI**

The UI scripts used in the game.

**Scenes**

The different Scenes of the game.

**Dependencies**

Outside libraries and assets used by the game.

**Materials**

The Materials used by the GameObjects.

**GameObjects**

The different GameObjects used in the game. The Model represents the model imported from a Modeling Software. The InGameModel is a Model ready to be instantiated in game whit the scripts and components needed already attached.

**Systems**

The different systems of the game, the idea is for the developer to not have to look at this directory to implement new functionality (using exclusively the Services Directory).

**Services**

The Services are stored in the Services directory in a way that if a developer wants to add a functionality in the code it can consult the directory and make easy use of the different systems already in place. The Services NEED TO BE DOCUMENTED along with the way they work and the pre-requisites of each functionality.