Requirements

Group 14

Tecch Titans:

Bradley Mitchell

Daniz Hajizada

Ellie Gent

Joel Crann

Keela Ta

Leo Crawford

Lukas Angelidis

Architecture

Diagrammatic Representations:

Structural

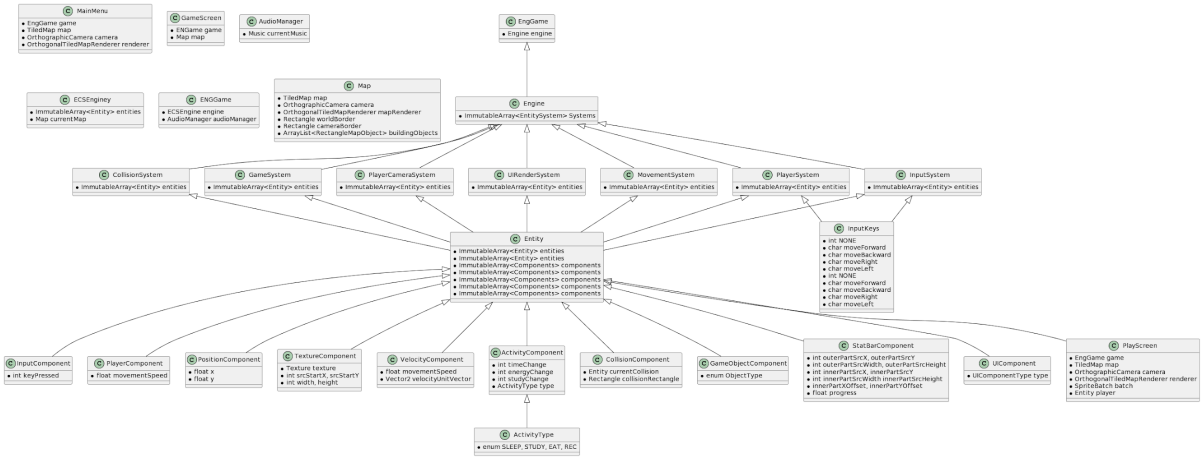
Message-Based Architecture *(Visible on the website)*

Basic architecture for a game involving a player that can move around a map, using messages to communicate input data with the display.

We've decided on using the LibGDX framework upon reflecting on this UML diagram, utilising a message based architecture to communicate inputs with a game screen.

Entity-Component System

This is the finalised Entity-Component System (ECS) diagram of our project. The previously implemented version of it can be found on our website.



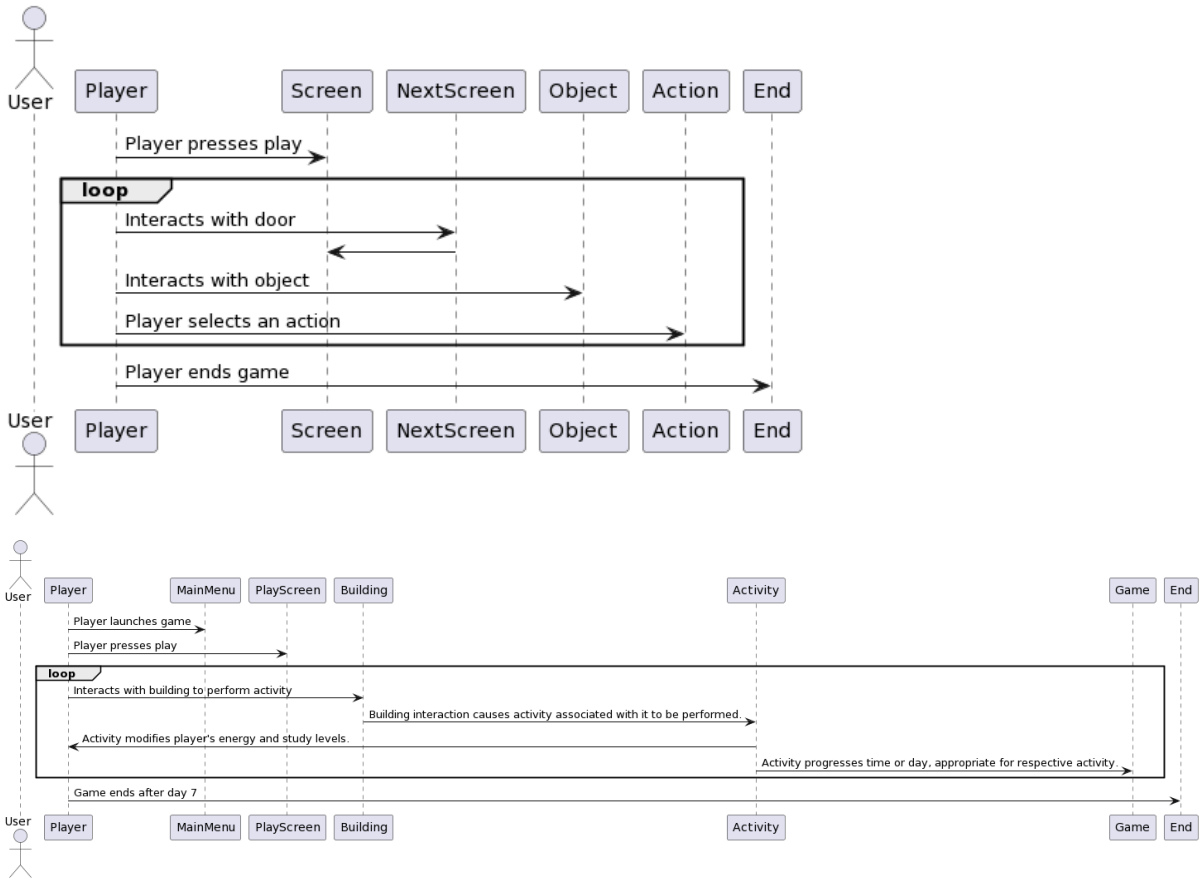
The reason we had decided upon using an Entity Component System architecture is because it allows a more flexible and modular design, allowing entities to easily reuse components to create new entities, giving us the ability to satisfy the user requirement [UR_AVATARS_ON_MAP] with relative ease. This architecture builds upon the previous architecture of being purely message based, with messages being sent between systems via the components which contain data related to the game. Each system can find particular entities with specified components, allowing the systems to interoperate while also retaining

independence from each other, as they do not need to directly interact to function. This allows a player to make any choice of their own leading to their own individual outcome - [UR_ACTIVITIES].

One particular instance of this can be seen in the input system. When a player interacts with an interactable entity on the map, i.e. the piazza building, the player component's current activity is updated to the desired activity. The game system independently constantly checks the entities with a player component for an activity to perform. When one arises, the game system handles the appropriate logic for the activity, i.e. progress the day and reset the time and energy when sleeping. The systems remain independent while also sharing information in a message based way. Overall, ECS has led to a more manageable and versatile codebase.

Behavioural

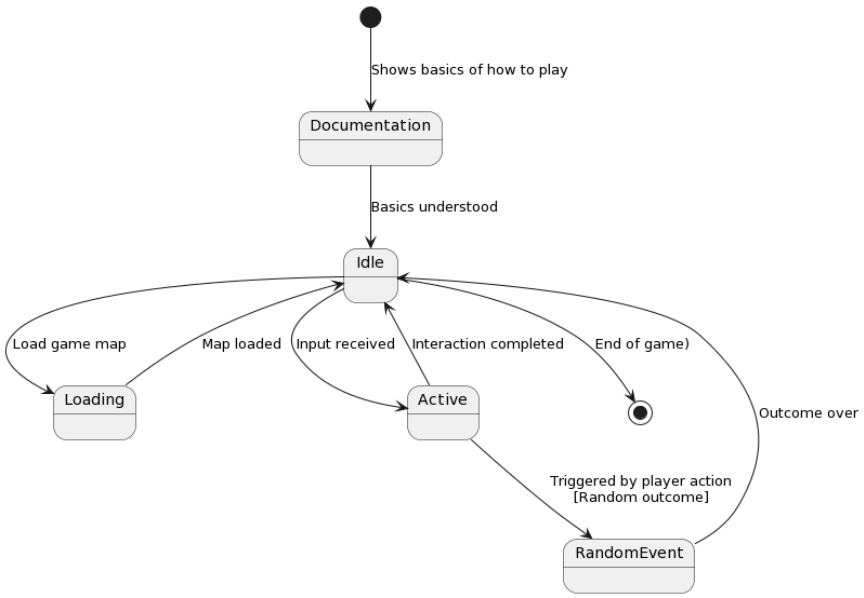
Sequence Diagram



Using an UML Sequence diagram, we've illustrated the sequence of interactions a player can make, as well as the flow of events which could happen throughout the game. This diagram includes components such as the player, the screen(s), objects and actions which interact with each other.

This behavioural diagram shows a loop which represents a quick and simple way of playing the game. Satisfying the [UR_GAME_PLAY] requirement, ensuring the game is easy to pick up and play.

State Diagram



This diagram represents the transitions in the game where an input is received and processed, leading to a change in game state, and then returning to the idle state in order to start again. The documentation at the start holds basic instructions of how the game works. This ensures that users who don't have much game experience and have not been trained are clear on how to play, which suits the customers needs [UR_GAME_PLAY].

The RandomEvent links to how the system should provide favourable outcomes to tailor the player's enjoyment and overall satisfaction [UR_USER_EXPERIENCE].