**Black Light Project’s Systems Design Document**

By SERGIO MADERA

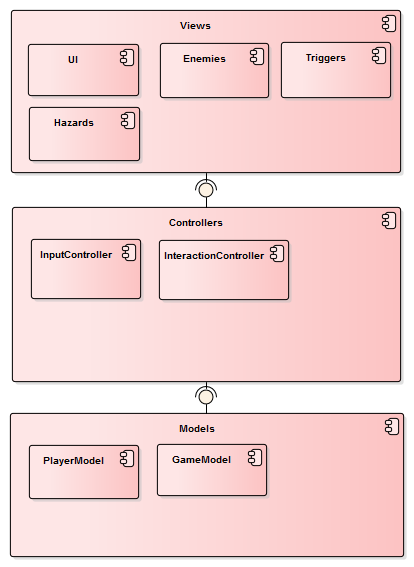
|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Revision date** |
| 0.1 | Sergio Madera | 17/10/2016 |
|  |  |  |
|  |  |  |

**Introduction**

This document’s purpose is to describe in detail and justify the design for certain systems used in the Black Light Project regarding interaction between components. Diagrams and similar graphics will be used when required.

**Components**

The following is the component diagram of logic components scripted by the developer (author of this document) in the Unity project. It is not strictly correct, and some components just describe aggrupation of other components.

****

Now, a brief description of each of them.

**Views**

**UI**

It’s the UI Script, handles what is showed, hidden or updated in the User Interface.

**Controllers**

**InputController**

Receives all input from the player and acts as a controller for the logic of the game, meaning it coordinates other components necessary to show the result of the input to the user. It also handles input related logic such as raycasting to see if the player is trying to interact with something.

**InteractionController**

Manages interaction with interactable objects in the game. Has a list of all accepted (programmed) interactions. The interactable objects tell they want to interact with the player and the parameters with which they want to do so, and the InteractionController coordinates the corresponding components so it is done

**Models**

**PlayerModel**

Handles logic regarding changes in the player's position, rotation, health, stacks, abilities, points and general status in the game world. This means it contains all player movement related logic.

**GameModel**

Contains information regarding current game world status, this is, what NPC should say currently, what quests are available, what enemies spawn, what event should happen next in the story.

**General Player Input Flow**

1. Player inputs
2. InputController gets input
3. InputController decides from what’s information is available to him what the player wants to do (following the game’s control scheme)
4. InputController calls other components so the input can be answered

**Interaction Flow with Interactable Object**

1. Player inputs a certain key
2. InputController gets input, checks if there’s an active interactable object under raycasting.
3. InputController calls Interactable Object (View)
4. Interactable Object tells the InteractionController the way it wants to interact with the player following a certain format
5. InteractionController calls the Model components necessary to make the interaction happen and tells the UI (View) to update itself accordingly.

**Interaction Flow when talking with NPC**

1. Player inputs a certain key
2. InputController gets input, checks if there’s an active interactable object under raycasting.
3. InputController calls NPC<Interactable Object> (View)
4. NPC tells the InteractionController it wants to talk with the player following a certain format
5. InteractionController calls the Model components necessary to make the interaction happen and tells the UI (View) to update itself accordingly.