1. PROTOTYPE IDEA

After receiving the test instructions, I had an idea to create a survival horror prototype slice. My goal was to replicate elements from older titles in the genre, and since some of the test requirements could be interpreted this way, I decided to give it a shot.

2. SYSTEMS

1.1. GAMEPLAY

The character movement, unlike older survival horror titles, are not tank controls nor have fixed cameras through the level. For this, I took inspiration on SIGNALIS and made the camera follow the character around.

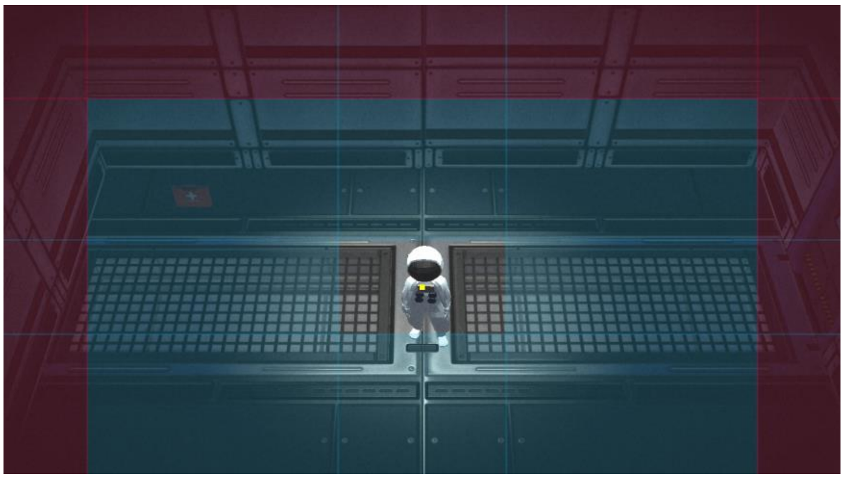

Figure 1: Screenshot showing the Cinemachine camera configuration.

This allowed me to keep the same movement orientation in the whole project without worrying about any input direction changes based on the camera position.

The animations were all imported from Mixamo's store and were just used to demonstrate how the character react to the world. Although this is a really simple animator setup, the polish came as the animation layers and on scripted movement. For the animation layers, the player has two that can be active. The first one is while the player has his weapon equipped and the second one are if the player are below half heath.

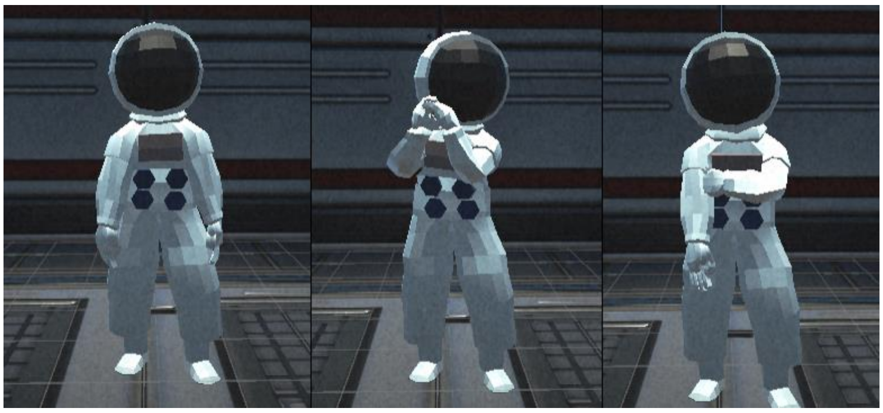

Figure 2: Screenshots from the different animation layers, being the idle with no extra layers, idle while holding a weapon and idle while injured respectively.

As for the scripted movement, this was the player character looking into objects that are “objects of interest”. These objects are all the interactable elements that were in the game, being collectables, doors or save points for example.

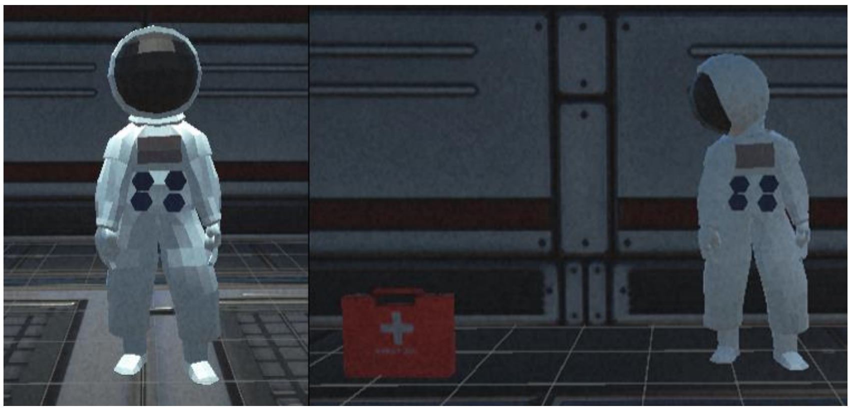

Figure 3: Screenshot showing the difference between the player looking and not looking to an “object of interest”.

The interactable objects were just an interface that the player could call the ”Use” function and had two triggers divided into pickup zone and interest zone. The pickup zone is the smaller one, granting the player the ability to call the interface function. The interest zone was the bigger one, serving as the callback to the player to register that object as an “object of interest”, thus making the player look at it.

1.2. INVENTORY STRUCTURE

The inventory was simple in concept, just modernize how old resident evil games worked with a functional drag and drop system. Although I’m not a good graphic designer, the inventory came close as to represent an older resident evil game. The data for the player inventory was divided into two parts: the player inventory itself and the box.

This was made to make a distinction from what the player is carrying (that is administered on the "PlayerController" script), and what is in the box (that is administered on the "GameManager" script). Both of these hold data from the "Item Inventory" struct that have some information like the item's name, description, type and so on. This data is fed into the "UIManager" and then into the "UISlots", showing the items and managing the operations like use the item or the drag and drop itself.

1.3. SAVE AND LOAD SYSTEMS

The save and load system is just a "PlayerData" class that hold some information about the progress of the game. The thing that makes this interesting in the point of view of a project that usually have a lot of items, enemies and level states to keep track of, is the UUID system.

Every object that have this unique id, can be kept track of just on a string list system. For example, the "GameManager" keeps a list of all the items UUID that have been picked up by the player, so when the level is loaded, if this unique id is in the list, the object is destroyed.

Using this in the save system, allows for a quick verification of what each UUID made, putting them in the correct state once A: the game is loaded or B: the scene is loaded.

3. PERSONAL PERFORMANCE

I really liked making a prototype of this genre but unfortunately I did not have time to implement half of the features that I have initially planned. Things like the weapon working, the keycard for opening other areas and an enemy, for example. But the requested features itself, they were things that I already done before in some projects, so it was easier to plan how they would work from the start.