Elisabeth Mateu.

# UniVRse Dev Test: Project Overview

## CharacterController:

Father class of both “PlayerController” and “EnemyController”, is responsible for containing any information they may share.

It and its children are also responsible for calling the methods that control their actions, contained in their particular classes, so every class has a single responsibility.

There is no limit to the number of classes controlling the actions of a character, but for this test, only 3 were needed. They are the “CharacterGrounded”, which checks if a character is grounded; the “CharacterJump”, which controls the character jump; and the “CharacterMovement”, which controls how the character will move.

## Player:

The player is controlled by the “PlayerController” class. As previously said, it is a child from “CharacterController” and it is responsible for controlling the actions it has to do, in this case as the player.

Given that its actions are controlled by a user, it checks for any input and calls its bound action.

You can find the class in the prefab named “Player”, alongside other character components.

## Enemies:

They are controlled by 2 classes:

- The “EnemyController”: child from “CharacterController” and responsible for each enemy’s behavior.
- The “EnemyManager”: that manages the enemy spawning when pressing the ‘E’ key,, given the spawn area and the enemy’s prefab reference.

The “EnemyManager” class is also responsible for assigning each spawned enemy move area which, if the enemy was already in the scene instead of being spawned, would be assigned through the inspector.

Both spawn area and move area are assigned with a collider, marking its borders as the limits. For this example, I have assigned the same collider to both of them, although it could be a different one if desired.

You can find both of the classes in the prefabs “Enemy” and “EnemyManager”.

## Visibility Zones:

There are 3 classes responsible for managing the visibility zones.

First, there is “VisibilityZone”, which you can find in the “Zone” prefab. The prefab has:

- A mesh renderer, to visualize better where the zone starts and ends.
- A collider, that checks if the player or an enemy enters the zone
- And the “VisibilityZone” class itself, which manages who enters and exits the zone and the visibility of the player within that zone and its neighbors (assigned from the inspector with a list).

Secondly, there is the “EnemyVisibilityZone” class, that manages the visibility of each enemy given the information from each “VisibilityZone”. It also removes any reference to itself saved by a visibility zone when it is destroyed.

Finally, we have the “VisibilityZone_DebugManager” class. It is responsible for managing anything to do with the debugging of the zones. The debug options are:

- “Player Triggers Zone Message”, that logs a message each time a player triggers a zone, saying if it has entered or exited it and which zone is it.
- “Auxiliary Mesh Enabled in Edit Mode”, which enables or disables the auxiliary meshes from each zone prefab in the edit mode.
- “Auxiliary Mesh Enabled in Play Mode”, which enables or disables the auxiliary meshes from each zone prefab in the play mode.

For the debug options to work, every zone should be a child of a GameObject with the “VisibilityZones_DebugManager”. The user will also need to call the “RefreshDebugParameters” method from the inspector, so the class actualizes each parameter immediately.