To describe the stages of production/development of the Gamejam project, I will try to summarize as much as possible but punctuating the most important points. All the visual and audio (assets) were taken from the internet free of charge and free to use.

The first focus was the creation of the scenery. With everything already configured and tidy, I went to the creation of the character, where I created three scripts, one for movement, one for animation and another to know when to look left or right.

The animation script has a controller based on the animator and a function where it receives the input values based on the new unity input system, and if the value being compared is greater than zero, it activates the running animation and if not, activate the idle animation, already in the playerController, the logic is quite the same, just being necessary to save the vector2 because I'm dealing with physics and rigidbody and the last script, which determines the position where the player is looking, just compares the input vector2 of player and position x, being greater than 0, looks right, less than 0, looks left.

After that, I configure the camera with the Cinemachine package, and in addition, I implemented the dynamic scene change, again three scripts were created, where they behave like ScriptableObjects, one of them stores the scene name, another is empty and the third one saves the scene information loaded, in addition, I created a SceneInitializer script, for when the player enters an area/scene, it loads the right scene, when the scene loading finishes, the PlayerSpawner script is invoked, where we basically invoke the player and check by the tag if it is already instantiated or not, in addition, there is the LoadSceneRequest script, where there are only two variables, one that stores the scene to be loaded and the other that simulates a loadscene effect.

With the scenes configured, I started to create the interfaces/UI and the gamemanager. The gamestateSO only stores a name, the GameManagerSO is responsible for receiving the current state of the game and the previous state, in the GameStateChanger, it basically receives the GameManagerSO and finally, there is the timeManager, which manages the pause of the game in certain states about the game.

In the dialog system, CharacterSO receives the name and the desired sprite, in ConversationSO it is where the dialog file is created, in DialogueManager, on the other hand, there is all the control of the dialog, from the beginning of the conversation, the passing of sentences and so on.

In the combat system there is in fact a slightly more complex and well-connected logic, having integration with the inventory, in addition to having a small turn-based combat system.

And finally, the implementation of the store and all the buying and selling logic, in addition to the communication with the inventory system and cash.

Particularly, to explain the processes through the document, it was a little complicated but anyway, the project and all the source files will be easy to understand, besides, the only system that I couldn't implement was the skin's, in fact, I managed to to find a logic to implement, I tried to use an overrideAnimator and activate it during the runtime, however, due to the state logic that was implemented and the script that controls the pause system as well, as described above, the skin's system did not occur. But in general, I believe that I managed to implement most of the necessary items, in addition to having a very beautiful and interesting project.