GameStateController class controls the game state and has references to some importante object. It controls Input according to its state, this way it must hold reference to objects that interact with input: Player, Message Panel and a dynamic reference to UI panels like Inventory and Shop.

The Player class controls the movement, animations and the Game Object that it interacts with. The class holds a reference to the InteractableObject that is attached when the player's game object enters the trigger and detaches when it leaves.

The Player movement is controlled with WASD and the inventory is opened and closed with the key Esc. Until now, there is no action to close the game. You can close with Alt + F4.

The Store

In the store the player can buy items using the "Buy" button of the item slot. It calls the Buy method of the "Shop" class, that checks the player's money and then calls "Inventory.Buy", changing the money and calling "Inventory.AddItem" that calls "InventoryWindow.FillSlot". This last method fills the slot with the item icon.

The player also can sell items, selecting them in the InventoryWindow and clicking on the "Sell" button. The button calls the "Sell" method in the "Shop" class that calls the method "Inventory.Sell". This method checks if the selected slot has an item. If it does, it calls "Inventory.RemoveItem" and adds half the item price to the player's money, then clears the selection.

The Inventory

The Inventory is opened by the Esc key. It uses two classes: Inventory that controls the Items and coins and InventoryWindow which controls the UI that shows the Inventory elements.

The Inventory Canvas has a panel that holds the title, the "Close" button, the "Equip" button and the Inventory Window. The Inventory Window is shifted to the Shop Canvas and Inventory Canvas when they are opened avoiding that the Inventory is builded more than once.

There is a "Equip" button in Inventory Canvas. This button calls the Equip method that checks the Item type and calls the appropriate method to equip the Item in the right slot. After this, the equipment slot is filled in InventoryWindow and an event is raised and is listened to by "Player" class that uses it to instantiate the Item prefab.

The player cannot sell an equipped item.
A function to unequip is not implemented, so, after equipping an Item, it is only released if another equipment is equipped.

I tried to use events as much as possible to avoid holding references. I know the event could be used more widely, but I think it was enough. Whenever was possible I preferred to use c#

events over UnityEvents. UnityEvents have cleaner implementation, but are harder to debug and harder to understand when there are more than one programmer in the project.

I did the best I could with the time provided given that I also had to take some time to find visual assets to use that could be used in this project.