Summary of Development

First Steps

The first steps where to in this case divide the works between the main components of this particular task which where:

1. Movement of the character
2. Shop creation

Having separated this tasks in general view, i proceed to work on the simpler first which was the movement of the character

Movement of the Character

For this I created 2 scripts that will handle the principal aspects of the movement.

This aspect was Camera and the Player.

I included the Camera cause I like to treat the camera as an extension of the Player, the way a camera moves and feels could be really appealing to a player.

So for starters a simple Camera Follow class that will take the target to move to its position , in this case the player, normally i would have to worry about rotation but with requested setup, it was easier to get it ready

Next was the Player Movement class, for movement I like to rely on Rigidbody physics movement, especially in 2D, there are options that apply well to it.

We add the Rigidbody to the player and then code the inputs for Horizontal and Vertical to grasp the directions.

On top of movement I added an animator, even though it is not like an animation, the management of the axis in the blend tree of an animator allows smooth transition of the sprites. I did not focus on this part so I could have more time in the shop.

Shop

For the shop I decided to distribute different classes for the purpose of readability of the code and understanding of the same.

I distributed the elements that would be in the store in the following classes:

- GameAssets
- Item
- ShopItem

GameAssets

This class was a handler object that would be available in game, in this class i put all the Sprites that were related to items to be used in the game and to be accessed via an Instance of this class.

Item

Inside Item, there was the template of what an item is and the method the item may have, sucha as GetItem, GetCost, GetSprite.

This is a class that does not need to inherit from monobehaviour.

ShopItem

The class shopItem is the representation in code of what the item in the shop is, has relation with the ui Shop and the Itemtype of Item

Inside this class there is only a method that is called when you click to buy.

UIShop and IShopCostumer

I created and simple Canvas to handle UI and the elements of the shop

I thought every shop item as a button and the way they need to appear or Instantiate and put the hierarchy in a way that satisfied this approach.

Inside UIShop i created most of the logic to be applied later by the Player Movement script

The most important method is the one that allows to create the Shop Item called **CreateItemButton** this method was the one in charge of creating the buttons to be available at the store whenever it shows

IShopCostumer for an Interface relation

While thinking about how to make the code a bit cleaner, i came up with the idea to have an interface to handle a couple of methods to be necessary

void BoughtItem(Item.ItemType itemType);

This method was to buy the item in question.

bool TrySpendGoldAmount(int goldAmount);

This method to subtract coins when it was possible to buy an item.

Both interfaces were implemented later by Player which was the class to have the final say.

Inside UIShop we have a reference to the shopCostumer interface just to call the methods implemented by the player.

Shop Trigger

The last of the classes i implemented was this Shop Trigger, as the name suggests, this was only in charge of managing Trigger enters and Trigger Exits

Depending on the one it was called, the uishop scripts is called to Show or Hide the store.