Blue Gravity Task - Development Process

So when I got the task it was dictated to create a 2D game with a buying/selling items from a vendor, where the player can interact with the vendor and buy items (clothes to be specific) from that vendor, and sell items to the vendor.

I first started with looking for a 2D character asset with multiple sprites for that character and animations so I can save time. I found this **asset** and used the knight as the player sprites. Since I used a knight I felt it was fitting to put him in a forest, I got a **free forest tile asset** from the asset store and put a merchant that sells armour and helmets in the scene. I used one of the sprites from the character asset above for the merchant as well.

I started by creating a simple 2D character controller. I move the characters RigidBody2D velocity by getting the input vector from the player input using the new Input System and multiplying that by a speed variable.

After I got the character moving in the scene, I started working on the merchant, I made a script for him which basically checks by a trigger if the player is in range and starts a dialogue with him **using a dialogue system I had coded for some of my previous projects, so I reused it for this task.**

After that I created an **Item Scriptable Object** which contains data for items that we can buy/sell/equip. It contains variables such as:

- **Item name(string)** (used as ID in player prefs to save the state of the item, if the player has it bought/equipped).
- **Item Sprite(sprite)** which tells the UI buttons in the merchant window and the inventory window alongside the player sprite renderer what sprite to equip
- **An enum of Item Type which had 2 constants Helmet and Body** which dictated what type of item it is, does it go on the head or the body sprite renderer
- **Item Cost(int)** which is used to dictate how much the buying cost of the item is, basically how much coins the player will spend to buy this item
- **Selling Cost(int)** which dictates how much the player can sell the item for to the vendor.

Alongside these base variables I also made some helper variables for the scriptable object to speed up the game design process for the sake of saving time, these are:

- **Use Same Name As Sprite(bool)** which if toggled true changed the **Item Name** variable to be the same name as the **Item Sprite**
- **Randomize Price(bool)** which if toggled true will reveal a **Vector2Int** variable of **Random Price Range** using **Odin Inspector**, using the Random Price Range variable the **Item Cost** will get a random price between the X and Y values of the Vector2Int variable.
- **Set Selling Price By Factor Of Buying Price(bool)** if toggled true this will reveal a float variable **Selling Price Factor** which will be multiplied by the **Buying Price** to set the **Selling Price** (basically the idea is, I set the selling price factor to be like 0.75, so the player sells the items back to vendors at 75% of the buying price)

I also created functions in the scriptable object script to **Get and Set the state of the item**(if the item is bought and/or equipped) **using player prefs.** Basically checking if a PlayerPref integer with ID of name is equal to 1 and if so the item is Bought, and if it's 0 the item is not bought/the item is sold. Same logic if the item is equipped or not.

After I created the scriptable object script I started working on an ItemButton script which controls the UI buttons,it contains variables for the **Item Image, Item Cost Text** and **Item Button.** I created a public function that sets the **sprite** of the image, and **string** of the text field. I also created 2 Getter functions that get the Item Cost and the button component.
All text in the game uses **Text Mesh Pro** and **Immortal Font**

Afterwards I created a **Player Inventory** which contains a list of bought items and spawns buttons in the player inventory windows (both in shop window inventory and in normal inventory) and is responsible for the functions of what those buttons do (Inventory buttons changing the sprite of the renderers to the new item sprites, Shop Window Inventory buttons to sell the item, remove the item from the list and add them to the merchant)

I made Scriptable Objects for each armour item and helmet item for the knight and assigned them to a list in the **Shop Keeper** script from the inspector. On Awake I loop through the list in reverse and check if the Item is bought if so I remove the item from the list and add it to the list of items in the **Player Inventory** script. I also created a Unity Event for the dialogue so when the dialogue ends it opens up the Shop window which is populated with buttons of the items the merchant has based on the list of scriptable object items.

The **Shop Manager (this uses the Singleton pattern so the Shop Keeper script and the Player Inventory script can use it's functions without needing a direct reference to it)** script controls the buttons that get spawned in the Shop Window and get added to a list in the Shop Manager, sets the functions of those buttons (to spend money, to add the item to the player inventory, and setting the state of the specific scriptable object to bought), It also has a public function to check if the player has enough money to buy items which basically loops through the list of buttons and sets the button to interactable based on if the price of the item is less or equal to the amount of money the player has.

I polished the spawning of the buttons using **DoTween** by basically making them pop as they spawn, and also used a pooling system instead of Instantiating them using **Lean Pool**

The **Game Manager** for this project was very simple, I mostly used it as a communicator script using the **Singleton pattern** where I set scene variables as public (such as the player inventory), and basically control if the player toggles the Inventory screen off/on

And finally - the **Coin Manager(also a Singleton)** script controls the economy of the player, it has a coinAmount variable of type Integer and a coinAmountText variable of type Text Mesh Pro (to show the amount of coins in the UI)
On Start I set the amount based on a **Player Pref** with a default value of 100 coins and update the UI text based on the amount.
I also created 2 functions for **Adding Coins** with an argument of **amount** which basically increments the coinAmount value based on the amount and saves the new coinAmount in **Player Prefs** this is used when the player sells items to the vendor**.**
Similarly a **Remove Coin** function with an argument of amount which removes coins from the currentAmount variable and checks if the current amount is less than or equal to 0, if so it sets the current amount of coins to 0 and saves the value to **Player Prefs**. This script contains a **getter** for the CurrentCoinAmount which is used in the function to check if the player has enough coins to buy items. It also contains a helper function which saves all player prefs based on a const COIN_AMOUNT_KEY of type string used as the player pref ID and at the end of the function checks if the player has enough money to buy any of the items in the shop window. This function is called every time the player Spends/Gains coins.

All in all I'm very proud of the system I came up with, even with the time constraint. It works as intended and it's very scalable.

P.S. Some other assets I used that I couldn't find a way to mention above:

- A chest prop from the **Pixel Art Platformer** asset that I used to decorate the shop keeper so it feels like he's giving you items from a chest
- For the UI I used the **Fantasy Wooden GUI**