Documentation

Thought Process

First, I started by implementing the basics of the game which were the Player and the Clothes. After that was done I started implementing the Inventory system, in which I created a class for each Item (Clothes), and executed logic with the OnTriggerEnter2D function to add that item to the player’s inventory. Once that was working, and the items were being correctly added to the inventory, I created the Equip System, creating a second inventory for the player to serve as the equipment inventory, allowing the player to equip 1 item type at a time (1 outfit and 1 hat). With the equip system working, I started developing the Shop. The player can buy 4 items from the Shopkeeper, which are 2 outfits and 2 hats, so when clicking to purchase a certain item, the ShopUI class reads which slot index was clicked and sends the corresponding item class to the player’s inventory. Then I use the GameManager to manage the player and shopkeeper’s coins, adding and removing respectively when buying or selling items. To sell items, I made the Shop display all the currently non-equipped items the player has in the inventory, and the Shopkeeper buys those items, not with the original item price, but with a modifier, which is lower than the original price. When selling, the item that is sold in the Shop, is read, and through its index, I remove the corresponding item from the slot in the player's inventory.

Player

PlayerCharacter

The PlayerCharacter class represents the player in the game and manages its inventor. It has an Inventory and an EquipInventory that handles collected and equipped items. The class provides easy access to these inventories, simplifying interactions with the game world.

PlayerMovement:

The PlayerMovement class controls the player's movement in the game. It handles parameters like movementSpeed and tracks player input through horizontalAxis and verticalAxis. The class updates the player's position based on the calculated movement direction and speed. It also communicates with the PlayerAnimator to animate the player's movements and provides a method to determine whether the player is currently in motion.

Items

CollectableItem:

The CollectableItem class represents the various items that the player can use in the game. It has attributes such as collectableType to categorize the item, itemName for identification, and icon for visualization. When the player character collides with the item, it is added to the player's inventory, and the item object is then destroyed. This was initially used to test the inventory system, but then since the items are bought from the shop, colliding with the item becomes irrelevant unless a drop and pickup system was to be made.

ClothesAnimator:

The ClothesAnimator class manages the animation of clothing items worn by the player, like the outfit and hat. It is responsible for animating the player's movements by utilizing the PlayerMovement class to determine the player's direction and magnitude of movement. The class interacts with the attached animator to set parameters such as the player's movement state, and horizontal and vertical values, based on the calculated movement direction.

Inventory System

Inventory:

The Inventory class manages the player's inventory by utilizing slots to store collectible items. Each slot has attributes such as type, count, maxAllowed, and itemPrice. The class provides methods to add and remove items, checking constraints to ensure a slot can accommodate the item being added. It also allows items to be added or removed from specific slots, facilitating a controlled management system for the player's inventory.

InventorySlotUI:

The InventorySlotUI class is associated with the user interface for inventory slots. It is responsible for updating the UI elements, such as the item's icon and item quantity text. The SetItem function sets the item's icon and quantity text based on the provided slot, while SetEmpty clears the UI elements to indicate an empty slot.

InventoryUI:

The InventoryUI class manages the user interface for the player's inventory, providing functionality to toggle the inventory display on and off. It also facilitates the refreshing of the inventory UI, updating the slots with the relevant information from the player's inventory and equipment inventory. The class calls the EquipManager to handle equipping and unequipping items, with corresponding updates made to the inventory UI and the shop UI.

Equip System

EquipManager:

The EquipManager class is responsible for managing the equipment of the player. It handles the equipping and unequipping of different items, such as outfits and hats, by instantiating or destroying the corresponding prefabs. It checks the type and name of the item to be equipped to determine the appropriate prefab to use. The HasItemEquipped function checks if a certain type of item is already equipped, preventing the player from equipping the same item type again.

EquipSlotUI:

The EquipSlotUI class manages the user interface for equipment slots, with functionality to display placeholders for outfit and hat slots. This class provides methods to set the item in the slot or to indicate an empty slot, making appropriate adjustments to the placeholder image as necessary.

Shop System

Shopkeeper

The Shopkeeper class manages the interactions between the player and the shop. It tracks whether the player is interacting with the shop and listens for the "E" key to open and close the shop interface. Additionally, it toggles the shop interface when the player enters or exits the shop area.

ShopItemUI

The ShopItemUI class represents the user interface for shop items. It handles setting the item properties and displaying the buy and sell prices. Depending on the availability of the item, it sets the appropriate price for the item. The class provides methods to obtain the buy and sell prices of the items.

ShopUI

The ShopUI class controls the shop's user interface. It manages the visibility of the shop panel and provides functionality to toggle between the buy and sell panels. The class is responsible for updating the shop's inventory and handling the buying and selling of items. It interacts with the player and the shopkeeper to manage the coins during the buying and selling transactions.

Menu System

MainMenuUI

The MainMenuUI class is responsible for managing the user interface of the main menu. It contains methods for starting the gameplay, toggling the options panel, quitting the game, and adjusting the music and sound effects volumes. The Awake method initializes the UI elements, the Start method plays the menu music, and the Play method loads the gameplay scene. Additionally, the class handles UI events for managing options, confirming and declining the quit action, and adjusting the music and sound effect volumes. It interacts with the AudioManager to control audio functionalities effectively.

PauseMenuUI

The PauseMenuUI class is in charge of managing the pause menu user interface during gameplay. It handles the functionalities to pause and resume the game, toggle the options panel, return to the main menu, and adjust the music and sound effect volumes. The Awake method initializes the UI elements and sets the time scale to 0 during pause. The class also communicates with the AudioManager to control the music and sound effect volumes based on the slider values.

Managers

AudioManager

The AudioManager class manages the audio elements within the game. It contains methods for playing music and sound effects, adjusting the volume, and triggering footstep sounds. The PlayMusic and PlaySound methods allow for the playing of specified music and sound effects with an option to adjust the pitch. The MusicVolume and SoundVolume methods enable the dynamic adjustment of the volume for music and sound effects. Additionally, the class handles the playing of footstep sounds by randomly selecting a footstep sound clip from the available array of footstep sounds. The GetMusicVolume and GetSFXVolume methods allow access to the current music and sound effect volumes, respectively.

GameManager

The GameManager class, on the other hand, handles the game management aspects, such as managing the player and shopkeeper's coins, applying changes in the coins based on specific events, and keeping track of the shopkeeper's sell modifier. Additionally, the class ensures that the game starts with a standard timescale and triggers the appropriate gameplay music. These classes work together to ensure efficient control over the game's UI and management systems.

UIManager

The UIManager class manages various UI components, including the inventory, shop, and pause menu interfaces. It provides methods to access each of these UI elements separately, allowing for interaction and modification of the UI components during gameplay.

Known Bugs

When walking while equipping an item at the same time, the item's animation does not match the player's animation if we keep walking. When we stop walking and start walking again after equipping it works fine.

Overall Self-Assessment

Overall I am very satisfied with my performance on this project, taking into consideration the time limit to execute it. Some aspects, such as perhaps a more dynamic inventory and shop system where the Sell Inventory of the shop could automatically fill the slots according to the player’s inventory slot size.