Stardew Valley Clone

by Cristian Scalia

First Steps

First of all, I begin by reading the document. Then I watch some videos of the game because I've never played it before. I had heard about it, but I had never actually played it. This way, I can understand the mechanics of the game a little better

Then, I create some macro tasks that will be involved in the implementation of the features

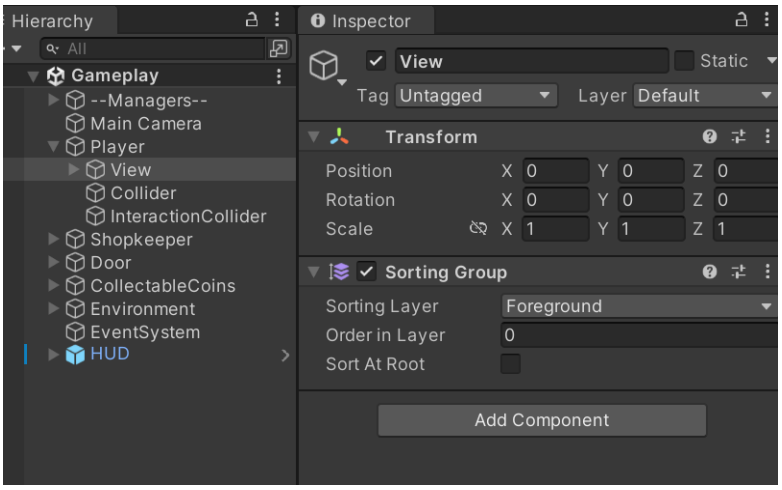
1 Player Character Movement
2 Player View Layers of Sprites
3 Camera Movement
4 Interaction System
5 Inventory
6 shopkeeper interaction (buying/selling items)
7 Create PDF File Upload To GitHub
8 Build Executable and Upload To GitHub

Starting With The Project

I Started adding the Asset Provided in the task's Documentation,

Then I Continue Creating the Character Movement, this is in Charge of the 'PlayerMovement' Component. To obtain inputs, I use the new Input System. I created an 'InputsManager' class that handles this system and it is also responsible for blocking the inputs in case someone is interfering with them (for example, when the Shop is open).

To ensure that a part of the environment occlude the player, I use two layers: Background and Foreground. The objects that need to occlude the player should be placed in the Foreground layer and must use the same Order in Layer value as the Sorting Group assigned to the player (in my project, it's set to 0).



Once I completed that, I started checking the Equipment View System. I created an Equipment Type for each body part: Body, Clothes, and Hair. Each part's view contains a Sprite Renderer that has its own Sort Order to ensure correct rendering order

There is an enum called 'EquipmentType' that corresponds to each Customizable Part (Body, Clothes, Hair). Then, we have the 'EquipmentView,' which is a ScriptableObject containing the definition of the affected Body Part and a 'runtimeController' that contains the animation clips of the corresponding Asset. I have created a base animator controller that contains 4 states, Up right down left, and then each Cloths or part need to create an Override controller and set the corresponding animations clips. The class 'Character View' is responsible for changing those controllers and playing the corresponding animations depending on the player's movement.

Then I created the Interactable System. We have an Interactor and Interactables. The Interactor collects interactables using triggers, and when the user presses the Interact Key 'E', the inheritances of the interactable will handle the interaction accordingly.

Then I created the Inventory System. The class 'ItemDefinition' contains the definition (Name, Price, Image, and an ID). The Inventory handles the logic for adding, removing, and notifying the InventoryUI to generate the view updates.

Once this is done, I create the ItemDefinition for the EquipmentItems. It inherits from ItemDefinition and adds the information of the Equipment View. This way, I merged the two systems: Items and Equipment.

Then I added a Character Equipment Script that will store information about the equipped items, allowing for unequipping in case an item is sold

Then I Create the shopkeeper using the Interactable System that triggers the opening of the Shop. And the Shop UI.

Finished adding a Simple Currency Manager and added some basic tutorials to show how to move and equip the items.

Conclusión

- I love creating prototypes, so I had a lot of fun. It's been a while since I worked on the 2D aspect of Unity. I dedicated a lot of time. Definitely, I'll continue with this prototype in order to finish pending tasks.
- I think that a lot of things could be done with a better approach. However, I don't have enough time to think of a better solution. In this way, you would need to create a lot of assets to design a new Cloth (Equipment Part), or at least create a tool that handles it. Also, I thought about implementing asset bundles to dynamically add clothes, but I didn't have the time.
- I Didn’t Localize the Texts in the Game .
- I didn't manage to create a basic popup system in the given time, nor did I create unit tests for the base classes such as Inventory and CurrencyManager.
- It took me more time than I initially thought
- I'm not used to adding assets that I will not use in the project. In this case I decided to leave it there because of the deadline. Shouldn't be bundled in the build because I'm not using it, but it generates a lot of project size unnecessarily

Hope you enjoy the code and the project, Looking forward to hearing from you son.

Best regards,
Cristian.