Shop Project
Made by: Willian Carvalho
Time spent: 13h 09min

Controls

Move: WASD (keyboard) / Left-joystick (control)
Inventory: I (keyboard) / start-button (control)

Introduction

The project was made thinking about create the most scalable and maintainable shop-system Program architecture. I was intending to apply Observer design pattern and SOLID principles to the architecture, but it wasn't fully applied. Because the project biggest focus was on the Shop system, the other scripts was made in a simpler way, and all the interactions with the player was simple. The game doesn't sound system due the lack of the time to develop the task. All the Unity project and code insides it (besides the art-assets) was made entirely by me while developing the task-project.

How to test the Game and Shop system

When you open the game, you can freely move around the tiny map. If you move up, you can see the Shop Keeper (the woman statue). When you get near to it, the shop will open and close automatically (open when gets near, and close when it's far). When the shop UI appears, you can buy and sell items based on you money balance (top-right corner of Shop UI). When you buy an Item, you can open your Inventory to equip the item bought. When you equip the item, it is removed from the inventory. To close the project, just press 'alt+F4'.

Game and Program architecture

The program architecture of the project was intended to have the most abstraction as possible. That way, if we need to apply some code maintenance, the other scripts of the project won't be affected, and the programmer doesn't need to see code who is not related to the maintenance task. Bellow, are a simple explanation of how the in-scene GameObjects and scripts was created:

Player character: Created using RigidBody2D (for collision), a Capsule collider, Animator (for animation handling), a PlayerData and PlayerBheaviour Scripts;

GameCamera: Used CinemachineBrain and VirtualCamera to make the Camera follow the player;

Shop Keeper interaction: Used CircleCollider2D to trigger the inventory and a ShopKeeper script to handle when the player triggers the ShopKeeper;

GameUI(Shop and inventoryUI): Used UIManager, ShopManager and PlayerInventoryManager scripts, for ShopUI and InventoryUI used a panel container with Grid Layout Group and content size Fitter to make the UI more responsive and dynamic. Some anchors where applied to make the UI more responsive, but was not fully tested;

Input system: Used the new input system to make the game to be easily built to other platforms (console, android, switch, etc.)

PlayerBehaviour(Script): Handle player move and animations

PlayerData(Script): Handle other player data like inventory, moneyBalance, Equip and Remove item from inventory, etc.;

PlayerInventoryManager(Script): Show to the screen, the items the player has in its inventory;

Shopitem(ScriptableObject): Created to make easier the creation of new equip items;

ShopItemDisplay(Script): Created to display the ShopItems in the store and inventory

08.30.2023

ShopManager(Script): Created to handle the Shop UI and its data (ShopInventory, Buy and sell containers, ShopResponseText, etc.)

UIManager(Script): Handle when the UIs need to be displayed.

Inventory(ScriptableObject): Created to ease the creation of an inventory (can be used to create other stores from other ShopKeepers and players too) ;

GameManager(Script): Is not being used, but was created to apply a better abstraction to the code and make it more organized (would be used for abstract the script events and handle game important information like number of shopKeepers, the player state, UI State, etc.)

Self Assessment

I am somewhat proud of the program architecture and project's organization, but I could make it better with a bit more of time. As said before, the abstraction and SOLID principles are not fully applied and this bother me a bit. I like to make all my codes clean and organized, and don't have time to refactor everything is a bit sad for me. But the shop system was really responsive, dynamic and scalable, and it makes me really happy and proud of myself. The game has some issues that need to be fixed, but nothing that can break the shop functionality. The equipped item needs to be animated to, so I started preparing the Animation controller to have the items' animations, but didn't finish.

The development of the project was really great and I enjoyed a lot, but I will refactor more the code to use GameManager, apply more abstraction and set variables and function more properly to make it a usable shop system for future games that will have this feature.

Hope Blue Gravity's Managers like the Project, because it was made with love and passion, and hope to work with you some day.

Sincerely,

08.30.2023