TinySimWorld:Medieval
Documentation

Introduction

From the start of development, the core idea of the systems here is to use one shop script for multiple shops, each having a dynamic and easy to edit item list. Also to showcase the use of scriptable objects and how powerful they can be in these types of situations where you might want many resources with different requirements to be used with a single system seamlessly. Each item config is used across the board providing systems with the data they need for each item.

Item Configs

This demo relies heavily on scriptable objects in the form of Item Configs to provide the game with a config for each item we want to add to the game. This allows us to easily add new items and keep track of each asset linked to items.

Item properties:

- Active - Is this item active? And should it be loaded by shops
- Name - This will be shown in the store and inventory
- Description - This will be shown in the store and inventory
- Cost - this is the price you pay for the item (Sell price is calculated as half this value)
- Icon - this is the items icon, this will be shown in the shop and inventory
- Type - this is the item type, this is required for selecting the correct action for the item. See below for more detailed explanation.

Items each have a Type (ItemType enum) that can be easily added to. This property is mainly used for detecting what we want to do with each item for example a Clothing item would need to be equipped while a consumable would need a different action.

Item config constraints/requirements

- Clothing Item - Clothing items require 3 sprites to be added to its sprite list, one for each direction of the player each with the following index (Front = 0, Side = 1, Back = 2).

Store System

The store system has the ability to be used for multiple store types and sell many different item types such as Clothing, Food, Weapons etc.

When creating a new store object in the world you will need to pay attention to the following;

- (Variable) Shop_Type - This is an enum used for setting the type of shop we are creating, This is then used to load the correct items in the resources folder.
- (Variable) Name - This will be displayed on the shop interface
- (Component) 2D Collider (Trigger zone) - Each shop requires a trigger zone for detecting when the player is in range and able to interact with the shop.

Each shop will load up all the item resources it needs using the Shop_Type that has been set. For instance if the shop is set to Clothing then it will load up all Item resources in the following folder “/Resources/Items/Clothing/”

You can add more items easily by right clicking in the desired resources folder and selecting the top most context menu item for example (“Shop/Item/Clothing”) and this will create a new Item template in the folder.
When the player initiates the shop interface using the action button, the needed items are loaded into the list, each item has two actions - buy and sell.
The player can only buy items if they have enough coins to do so, when this does happen the item is added to the players inventory.

The player can only sell items that they have in their inventory and is not equipped. The sell price for each item is calculated as half the buy value.

Inventory System

While the inventory system is basic, it provides a good example of how each item can be equipped based on the item config data.

When the player pressed the Inventory key, the game loads up all the items in the players inventory but not the ones currently equipped (design decision). The player can then choose to equip items by pressing the button on the item in the list. The players character script will be requested to update clothing and passed the item config.

The previously equipped items will now be unequipped and will now show in the inventory list.

Player Controller

The player is driven using a very basic controller, while there is not a great deal to go into here I do want to mention the importance of using FixedUpdate when dealing with object movement and Update when listening for player input. This will give the player the most clean experience and will have nice smooth movement with snappy input.

Closing Thoughts

While pressed for time over the past two days I believe I made good progress with the end result being a bug free experience with a good example of a basic shop and item system. If time were permitting I would have liked to have done the following:

- Add some more examples of different shops and items.
- Added some more sounds
- Added some NPC movement and interactions
- Added some ambient interactions

Overall I am quite happy with the end result. I have not taken part in a game jam for quite some time so I quite enjoyed myself during this interview since it had similar time constraints and requirements.