Prototype Shop System

Rosa de Souza

How the prototype plays:

The main scene of the prototype contains quite a few mechanics for the player to mess around with. First off the controls:

- Move around with arrow keys and ‘WASD’;
- Interact with UI elements by clicking with the mouse;
- Quit the application by pressing ‘Esc’.

The first thing the player may notice is the inventory button. When it is pressed, it shows the inventory. With the inventory menu open, it’s possible to change the player character’s clothing.

You have 3 categories for clothing: Headgear, Bodywear and Footwear.

If you click on a piece of clothing when it’s already equipped, the character’s clothing will return to the default (first two slots of the inventory are the current default). Otherwise, it’ll change to the clothing piece shown in the image slot.

To the top right of the main game window, there’s the player’s coin count. With this currency you can buy any item from the store. If you decide to click on the coin count, your coin count will increase by 10.

If the player decides to start moving, they can interact with the shopkeeper by getting close to it.

The shopkeeper starts with two options: Open shop and Talk.

If you decide to Talk with the shopkeeper, a dialogue box will show up in the UI, this dialogue box disappears after a few seconds. After that the Talk option is replaced with Close (in case you need to close the menu that shows up after getting close to the shopkeeper).

And now for the most important part of the prototype: The shop.

Once the shop menu is open, the shop window and your inventory will be open. In the shop window, there are currently 3 items.

Beside them, there's the value of each one. If you got the right amount of cash, you can buy any of the items as much as you want. They'll be added to your inventory accordingly, and if you leave the shop menu, your player character will already be wearing them.

While in the shop menu, if you go to your inventory and press on any of the recently bought items, a new button will appear. This button can sell any item you want, except for the first two items, in which case the button will never appear. Once an item is sold you get your money back.

How the prototype works:

Clothing – Every clothing item is a scriptable object with the fields shown below:

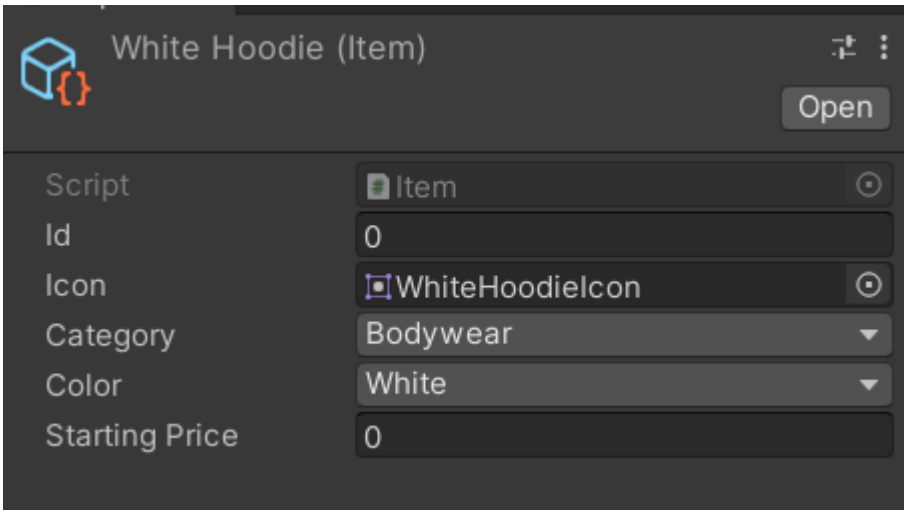


Figure 1 - Item Scriptable Object

Id – Unique ID for each item

Icon – The icon that's going to be shown in the shop and inventory

Category – This is used to determine which sprites will be changed from the player character

Color – This is used to determine which item color/type the item is. For example, if one more set of items were to be added for the character, it'd have a different name here.

Starting Price – The price of the item originally, used to calculate the currency.

Originally, I was going to save the items through JSON, because then the inventory could be saved at any point. I ended up not having enough time for that, so I decided to go with scriptable objects for now.

Character using item

At the start of the project, I tried utilizing sprites, just like Stardew Valley. Many issues arose while trying to come up with solutions for this, and it'd take way more time than required to work with, compared to something like rigging. So, I then decided to change by using skeletons. I had trouble finding good free assets for a while, but after some digging, I was able to find these characters made by Junhyuck Jang.

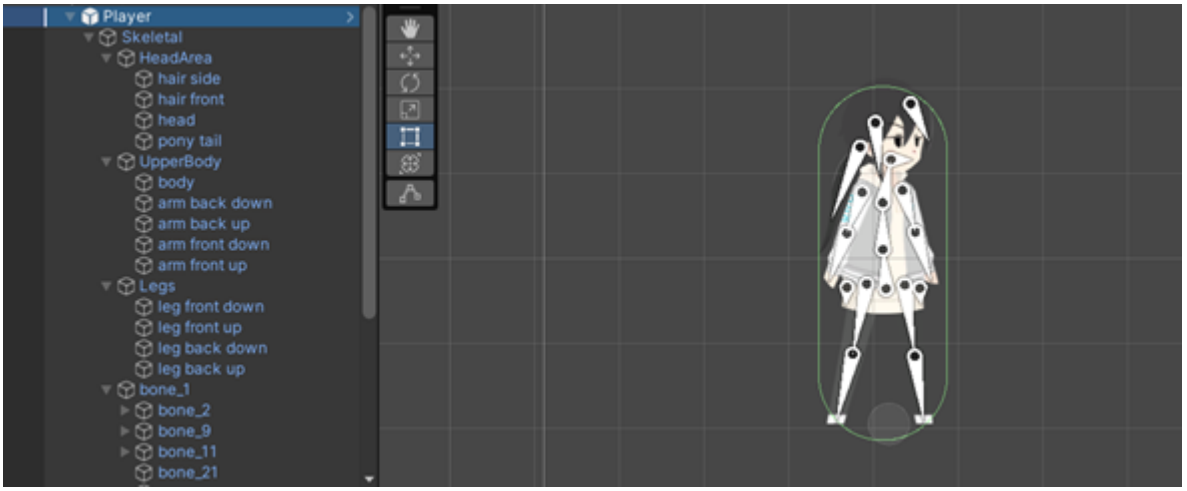


Figure 2 - Player Prefab and Hierarchy

As you can see, the character already had some bones and even animations prepared, although I had to change some of them for the headgear piece to fit.

The biggest issue overall with this asset was the way the sprites were divided. At first, none of the parents for each body member were there. And the worst part, if I wanted to design clothes I'd need to change all of these assets.

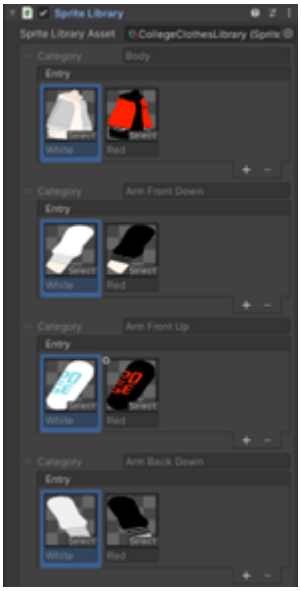


Figure 3 - Sprite Library

What I did then, was create a sprite library of the main sprite PSB file, with that it's possible to create many clothing pieces for the character. Each piece has a category and name, this is where I used the scriptable objects variables.

If your category was Bodywear for example, it'd take the Game Object UpperBody (shown in Figure 2) and get the sprite resolver of every child in UpperBody. This way, it's possible to change the entire upper body of the character with just one click.

```
// Gets the sprite resolvers from the body part
_sprites = _correctBodyPart.GetComponentsInChildren<SpriteResolver>();

// The item's color
var color = itemInSlot.item.color.ToString();

// Changes character sprite by changing the category of the item + the color
foreach (SpriteResolver sprite in _sprites)
{
    // If the current player's color is equals to the item's color, then it'll reset back to default.
    // Otherwise, change to the item's color.
    // Also checks if it's wearing the item while selling it.
    // If it is then item's color will reset back to default
    if(sprite.GetLabel() == color) sprite.SetCategoryAndLabel(sprite.GetCategory(), "White");
    else if(!_sellScript.beingSold) sprite.SetCategoryAndLabel(sprite.GetCategory(), color);
}
```

Figure 4 - Code for body sprites

The function SetCategoryAndLabel is then responsible for changing the sprite resolvers to the correct Category and Type of the item.

This code is used in the ChangeCloth() function in the InventorySlot script, which is why there's some code for checking whether the player is wearing the item or not. This is used for when you sell an item, pick it in the inventory or even whenever you buy it.

Inventory System

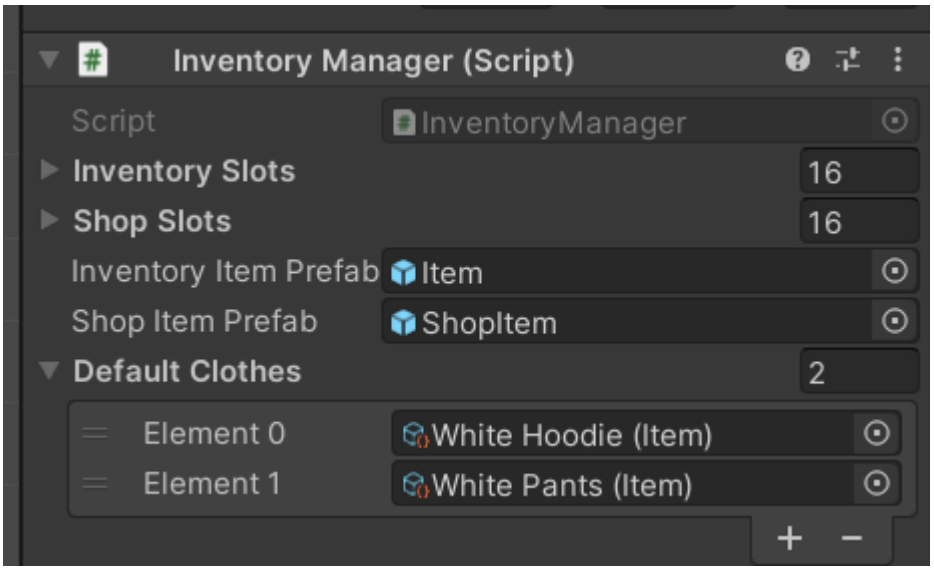


Figure 5 - Inventory Manager in Hierarchy

The image above shows the Inventory Manager. This is where the game will check all the slots in the inventory and shop. It also spawns the item assets accordingly whenever asked to do so. The default clothes will stay in the character's inventory no matter what.

The one difference between the inventory item and the shop item prefab is that the shop item will show you the item's price, while the inventory is just the icon.

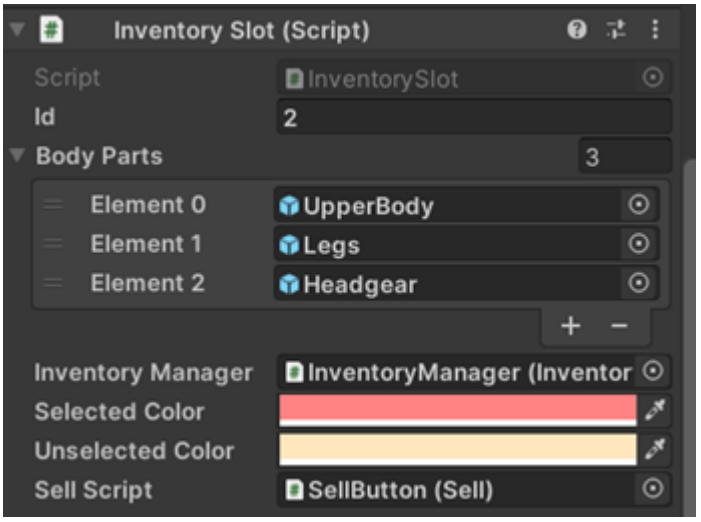


Figure 6- Inventory Slot Script in Hierarchy

Each slot contains their own set of characteristics:

The Id is the slot's own unique ID. This is mostly used for the ChangeSelectedSlot() function, which will change the slot's color once clicked.

The body parts are used for the character's sprites.

Selected and Unselected colors are there to change the color of the slot whenever it's selected or not.

And at last, the Sell Script is to get the Sell button's script, in this case it's used to check if you're wearing the item that's being sold.

Shop System

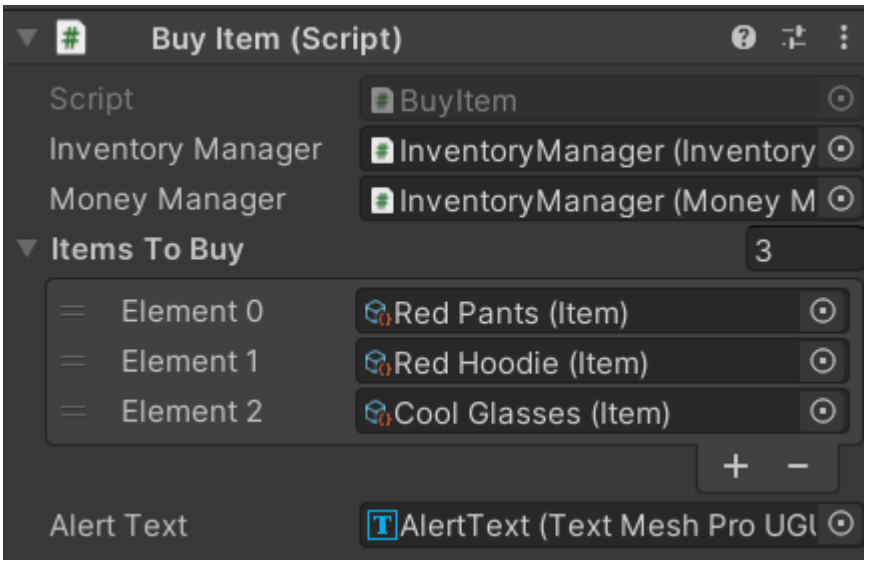
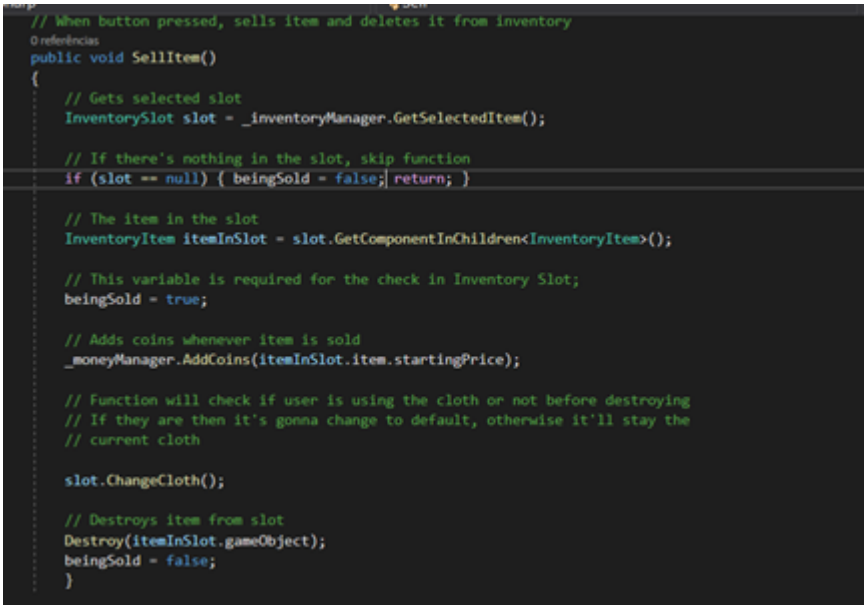


Figure 7 - Buy Item Script in Hierarchy

The Buy Item script requires a set of items to function, each of these items will be turned into an option in the shop. The script just basically adds an item to the inventory if the player is unable to buy it.

Sadly, I couldn't get the time to perfect the system, so the buttons for purchasing the item will take the position of the item in the array shown in the picture, instead of the actual item's Id.

The alert text is just there for feedback whenever a player buys something or to show they don't have enough money to buy the item.



```
// When button pressed, sells item and deletes it from inventory
0 referências
public void SellItem()
{
    // Gets selected slot
    InventorySlot slot = _inventoryManager.GetSelectedItem();

    // If there's nothing in the slot, skip function
    if (slot == null) { beingSold = false; return; }

    // The item in the slot
    InventoryItem itemInSlot = slot.GetComponentInChildren<InventoryItem>();

    // This variable is required for the check in Inventory Slot;
    beingSold = true;

    // Adds coins whenever item is sold
    _moneyManager.AddCoins(itemInSlot.item.startingPrice);

    // Function will check if user is using the cloth or not before destroying
    // If they are then it's gonna change to default, otherwise it'll stay the
    // current cloth

    slot.ChangeCloth();

    // Destroys item from slot
    Destroy(itemInSlot.gameObject);
    beingSold = false;
    }
```

Figure 8 - Sell Item Script

The sell item script is simple, it just takes the item that's selected currently and sells them whenever the button is pressed, destroying the item's Game Object.

Final Thoughts

Overall, I think I did a pretty good job. Obviously, if it wasn't for the first day where I had to restart the character animations, plus the time I spent trying to find good free assets. Then maybe the project would be prettier, with some better-looking UI elements and animations. And maybe some more mechanics for the shop.

But even so I tried my best to make the prototype as polished as possible with what I had, and with the time required. Which turned out to be an amazing experience.