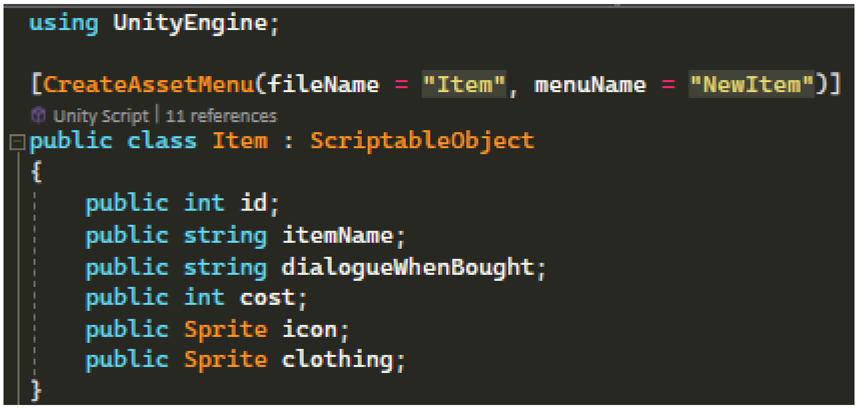
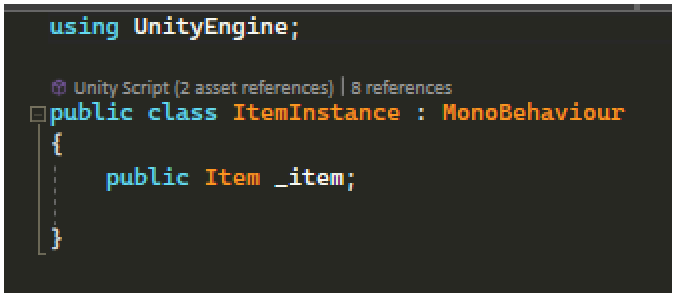
System

Main scripts: InventoryHandler, ShopHandler. Both the inventory and shopping systems are very similar and are based around the “Item” scriptable object. Each script has a list of items and is in charge of adding/removing them and updating the visuals of the respective UI, this is easy as we have the “item Instance” script which let us access anything we need easily. Additionally, the inventory has some added methods such as changing the players clothing and raises some events that the ShopHandler listens to, aiming for interesting dialogue interactions related to what the player buys and wears.



```
using UnityEngine;

[CreateAssetMenu(fileName = "Item", menuName = "NewItem")]
public class Item : ScriptableObject
{
    public int id;
    public string itemName;
    public string dialogueWhenBought;
    public int cost;
    public Sprite icon;
    public Sprite clothing;
}
```



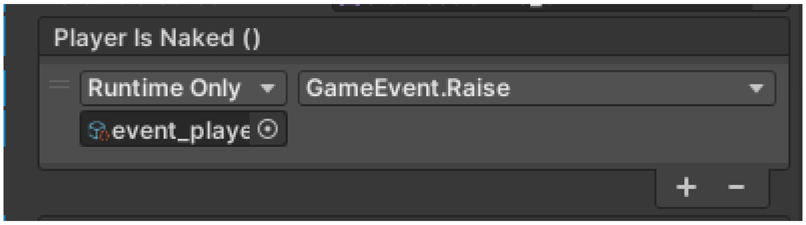
```
using UnityEngine;

public class ItemInstance : MonoBehaviour
{
    public Item _item;

}
```

Events are raised on a modular way, the ShopHandler never references or subscribes to the Inventory events. We do this using the “Game Event” and “Game Event Listener” scripts.

The inventory raises the scriptable object event through and unity event

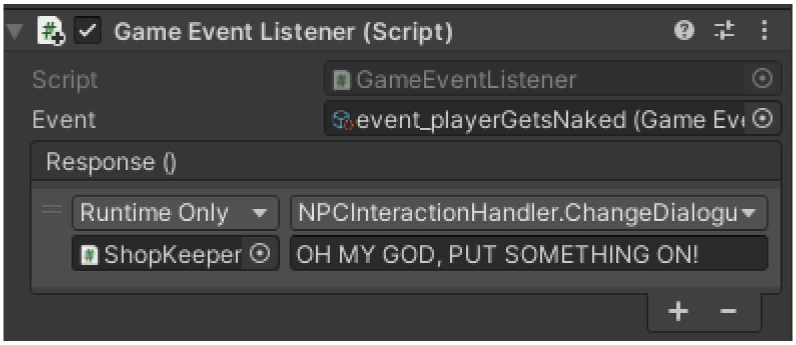


Player Is Naked ()

Runtime Only | GameEvent.Raise

event_playe

And then an event listener can make anything we need, without them referencing each other in any way.



Game Event Listener (Script)

Script GameEventListener

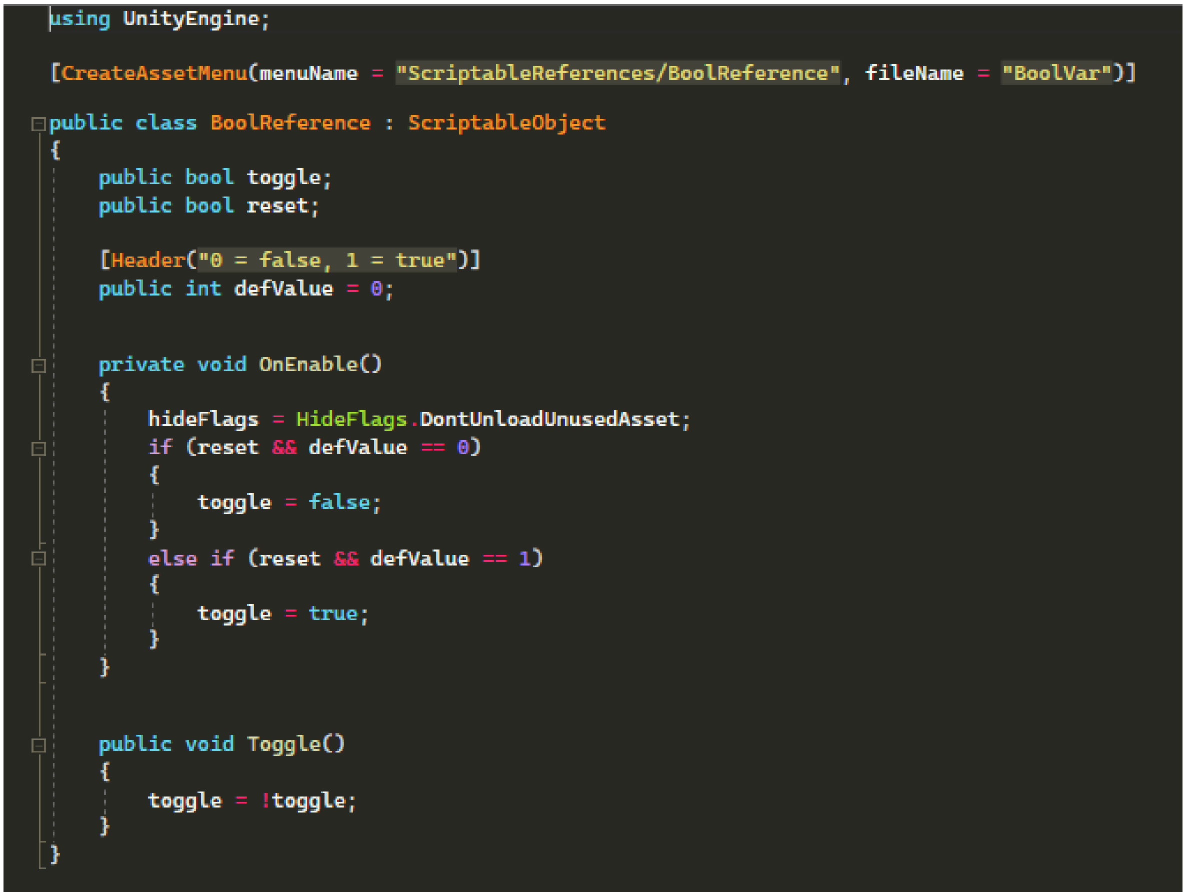
Event event_playerGetsNaked (Game Ev

Response ()

Runtime Only NPCInteractionHandler.ChangeDialogu

ShopKeeper OH MY GOD, PUT SOMETHING ON!

For the sake of a modular architecture, I also created some scriptable object “References”. The main use was so other scripts can know when the shop is opened using a “BoolReference”



```
using UnityEngine;

[CreateAssetMenu(menuName = "ScriptableReferences/BoolReference", fileName = "BoolVar")]

public class BoolReference : ScriptableObject
{
    public bool toggle;
    public bool reset;

    [Header("0 = false, 1 = true")]
    public int defValue = 0;


    private void OnEnable()
    {
        hideFlags = HideFlags.DontUnloadUnusedAsset;
        if (reset && defValue == 0)
        {
            toggle = false;
        }
        else if (reset && defValue == 1)
        {
            toggle = true;
        }
    }


    public void Toggle()
    {
        toggle = !toggle;
    }
}
```

All the “Reference” scriptable objects and the game event scripts were brought from personal projects. The “Player movement” script was the only other script brought from another project.

Thought process

My thought process centered around this sentence of the task “it is crucial to consider both design and aesthetics in addition to the playability of the prototype you create.” I decided to first look for the art assets. Then build a story with the visuals I had at hand with fun interactions, around the buy/sell system, between the player and the shop owner creating.

Personal Assessment

More than a task with requirements I’d say I achieved a little fun with room for stories on each player’s mind depending on the interactions they discover. I did what I planned so I would give me a nice assessment on design, programming and quality.