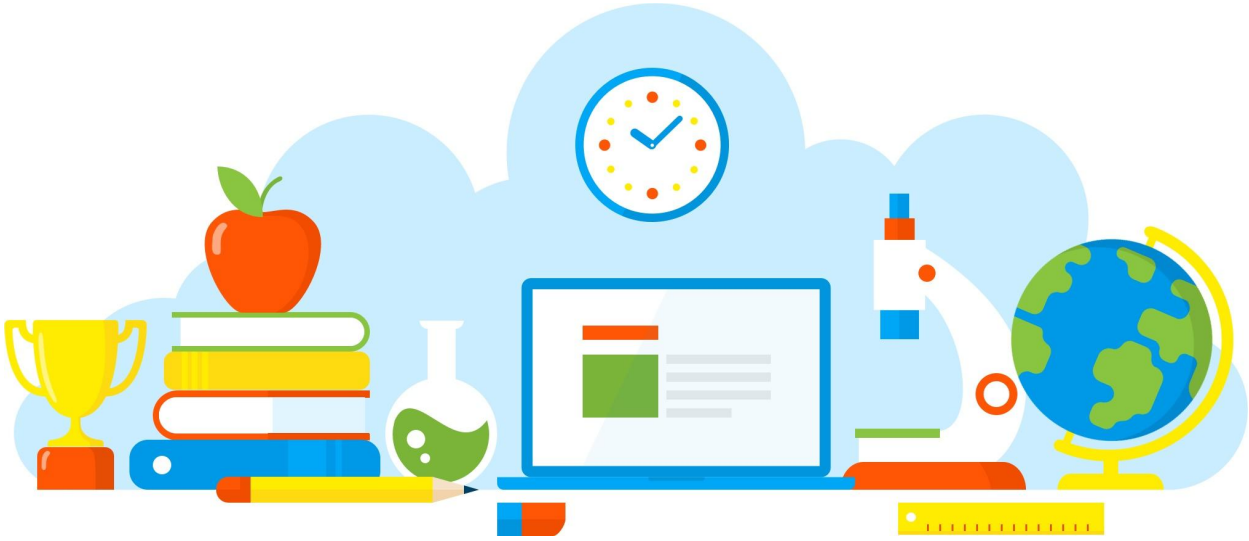


LSW Test Document

This document is submitted to fill the requirements of LSW Test

Ramez Al Tabbaa
13/06/2021

Introduction

Thanks for this great opportunity to join your ambitious great team and unique world of LSW, I did my best in this demo in hope it meets the expectations of your team standards , I enjoyed each minute working on this interview project, and I am so excited to join the team and work on LSW since games development is my passion.

LSW Demo Summary

Since the evaluation emphasis on both design and aesthetic choices, I searched carefully for the assets that ensure fulfilling those requirements.
List of premade assets used:

- Fantasy Heroes: Character Editor : used to create the characters in the prototype.
- 2D Village Tileset: used to create the level and environment.

Main features of the prototype:

- Character movement.
- Character intractability with intractable gameobjects.
- Dialogue system.
- Inventory system.
- Shop system.

I've started the planning from the items that will be available in this demo and created 2 different types of scriptable objects one for the outfits and another for other items.

The reason for differentiating the outfit from other other items is that the outfits consist of 12 sprites that cover the player's body and require differing treatment to apply them to the player when he equip the outfit.

The demo have 4 types of items and we created 3 items out of those types so in general have 13 items in this demo:

1. Outfits x3 + default outfit
2. Hats x3
3. Shields x3
4. Weapons x3

You can find the scriptable objects under Resources folder, those scriptable objects used as a basis for creating the inventory and the shop systems.

Player movement:

The character movement script allows the player to move freely in the level using the keyboard inputs which provides values for the horizontal and vertical axis.
You can use WASD or Arrow to move the player.

Player Detector:

Allow the player to detect and call the action of the interactable objects using Physics based on the last movement direction recorded.

Player Animation:

Controls the player moving animations & flipping the character face based on the movement direction.

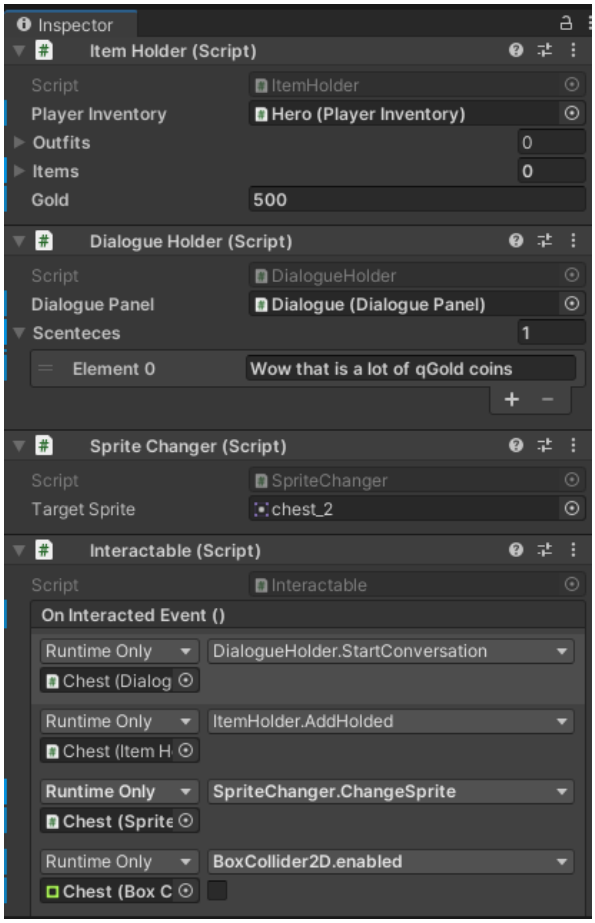
Player Inventory:

Is where all the player items / outfits / gold is stored, it is responsible for launching the InventoryItem + while also granting access to Add / Remove any item and gold.

Character Intractability:

In order to make any object interactable you need to make it an interactable layer, then add Interactable.cs and list all the events you want to happen in OnInteractedEvent.

Taking the Chest as an example, you can see that all the actions are listed in the Event in Interactable.cs giving any game designer the ability to add / make any interaction he/she wants.



Dialogue System:

It is a very basic dialogue system within the time limit that allows adding sentences easily then they are rendered with typewriter effect.

There is also a continue button to start the next sentence.

Item Loader:

ItemLoader.cs is responsible for cashing all items / outfits from the Resources folder for any other class to access based on ID.

Q: Why not just send the scriptable objects between classes?

It was important for me to make sure the class won't have to be changed much (if at all) in case of adding more types of scriptable objects (scrolls, potions…)

So I keeped all the communications when sending / getting scriptable objects based on ID, and each class can access what it needs from ItemLoader (e.g type, icon, price...)

Inventory System:

Equip / Remove any item from your inventory, Equipped Items / Outfits are not mentioned in the Inventory.

You can open the Inventory Panel with the I button.

Shop System System:

You can buy / sell any item in the game in the shop.

Both Inventory system & Shop system use InventoryBag.cs as a holder for all they need to Show / Buy / See / Select any item in the game.

Both Inventory system & Shop system use InventoryItem.cs as a UI button to show items and allow their functionalities by adding Listeners on the button when instantiating one.

```
GameObject itemObj = Instantiate( inventoryItemPrefab, shopkeeperItemsHolder );
InventoryItem item = itemObj.GetComponent<InventoryItem>();
item.button.onClick.AddListener( delegate{ SetSelectItem( item ); } );
item.button.onClick.AddListener( delegate{ ShowBuyButton(); } );
```

Normally I would make an Object Pooling system so I won't have to Instantiate & Destroy InventoryItem between the two Systems.

Conclusion:

Overall, based on the test time limit I believe I did a fair job in terms of the prototype functionality , game design and ashestics. The prototype is eye-catching and provides a good core basis for creating a simulation genre game that is similar to that of 'The Sims' and 'Stardew Valley'.

Everything uploaded to github including the executable build as required.

Links:

Github: https://github.com/RamezTa/LSW-Demo/

Additional build download link:
https://mega.nz/file/H1gA3QYT#zz67mjSZaTxJmED80sTlEOvNA2KGQT1jp9VL8KAHi3U

Thanks for this great and fun opportunity and looking forward to getting a chance to join your team soon.

Regards

Ramez Al Tabbaa