Styles

Programmer Interview for Blue Gravity Studios

Thought Process

On receiving the interview assignment I took a piece of paper and tried to break it down into smaller tasks. Such as:

- movable 2D top down character
- character inventory
- item trader
- character clothes
- interaction system

I created a simple board on Trello and moved tasks there. Besides the main repository I also created a sub repository which contains possible common stuff which may (or may not) “other” projects take advantage of.

For each task I created a test scene which dealt with a specific task. Most of the implementation took place at the test scene. Isolated from everything else.

Rough explanation of the systems

Inventory

Inventory can contain any range of Items. In editor you can specify size of an inventory and default items which will be initialized on awake. As API inventory provides following:

events:

- **EInventoryChanged**: notify if items changed in the inventory.
- **ESlotUsed**: notify if item at slot was used.

some of the functions:

- **void AddItem(Item item, GenericDelegate<InventorySlotCallback> callback**): Adds item to inventory if there is space.
- **Item RemoveItem(int slotID)**: removes an item on a specific slot.
- **void SortItems()**: sorts items
- **UseItem(int slotID, GenericDelegate<InventorySlotCallback> callback)**: Use item at specific slot. It may be selling an item, or wearing an item. Depends who calls it and what kind of item it is.

Item Shop

Shop has a list of items which can be purchased. Items can be set in the editor. For item shops there is another component implemented called Wallet. Wallet takes care of the transactions and holds the money value.

functions:
void PurchaseItem(Wallet wallet, int shopItemID, GenericDelegate<PurchaseCallback> callback):
void SellItem(Wallet wallet, Item item, GenericDelegate<SellCallback> callback)

To share items between different shopkeepers (imagine one store, but multiple shopkeepers) there is optional field called _sharedShop;

Character Skin

CharacterSkin takes care of how a character looks. Default looks can be set from an editor.

There are basically two types of looks. Body skin and outfit.
Each look has its own SpriteRenders and corresponding slot. Each sprite slot also has 3 variants of sprites for each facing side of a Character. Front, back and side facing.
When facing is changed, CharacterManager calls on CharacterSkin methods to update body and outfit renderers for corresponding facing.

Interactions

For interactions there is interface **IInteractable** and **CharacterInteractions** monobehaviour.

It has one method:
void Interact(MonoBehaviour actor)

CharacterInteractions has one method **TryInteract** which casts ray and checks if hit objects have implemented **IInteractable** interface. It then notifies through callback **CharacterManager** (or whoever calls TryInteract) about the interaction where it is taken care of it.

Personal assessment of my performance

I was feeling very comfortable when implementing assignment related programming tasks. It was fun.
From 0 to 10 bananas I would rate myself solid 7. Cut down 3 points because:

- On the start of the assignment I should specify scales of the sprites.
- GUI could be more consistent with how it looks.
- More item/outfit variants.
- Some of the code would need to be refactored.

Links

https://github.com/Jerichos/Styles
https://github.com/Jerichos/Styles.Common

https://trello.com/b/dz5JncSH/styles