LSW Programing Interview Documentation

Gabriel Machado de Carvalho Gomes
(git repo)

System Architecture

The system uses singletons and scriptable objects as well as object oriented design patterns.

Player movement and animation are very simple, reading from Input.Axis and changing Rigidbody2D velocity and updating blend tree animator float.

The buy / sell system is done through a static class TransactionManager.cs that handles checking the requested transactions and triggers the effect on both buyer and seller.

On the other hand, the inventory of both shop and player is handled by InventoryManager.cs. This class not only coordinates money transfers but also inventory addition and removal of items.

The equip / unequip systems is done by replacing the sprite gameobject of player and updating PlayerManager's reference to the new animator. This way the sprite is changed and controlled seamlessly by the manager.

The panel UIs are configured referencing a InventoryManager.cs and are updated every time the inventory is changed. That way the panel are consistently showing correct information about item listing, used both at shop's and player's inventory.

Items and its data are implemented with Scriptable Objects interacting with almost every part of the cloth shop system, from UX to gameplay.

Design Process

The design process started in my personal notebook. Some general organograms and class concepts started to take form and fit together.

Initially the idea is to create a shop system that could interact with the player but most of all could be easily expandable and customizable in terms of item data and item listing. That's when I decided to implement the items with scriptable objects. This way not only me but any developer (specially non-coders) can create, visualize, edit and remove items from the game with just a few clicks. (eg. Create->Scriptable->Item)

When I started to implement the player the simpler and smoother I could, I realized the layer ordering mechanics that should be in place for every sprite. This way I've created a script that orders the sprite accordingly to their y position + offset. Applying not only to the player but to other sprites at the scene worked like a charm.

Moving on to UI panels, I started first creating the shop UI. By configuring the available items reading from a separate inventory manager I was able to update the UI every

time a transaction between the inventory occurs. Applying this model to player's inventory was very simple since the shop inventory and panel implementations were done generically.

Creating the buy / sell mechanics required a static class that could overview any requested transactions. With it I was able to invoke an event every time a transaction was validated and every component that needed were updated with correct inventory data.

Implementing the equip / unequip system was very straightforward since it was only a matter of removing previous skins and instantiating the new skin prefab while updating player manager reference to the animator.

Self Evaluation

This is my third time applying to Blue Gravity. First one was in 2020. That being said, I think I'm doing very well at persistence and confidence.

Regarding the technical aspects of programming I think I've managed to create an interesting modular system that not only updates itself after transactions but also can be easily scaled to new and bigger collections of items, even by no-coder developers at team. Seems a good fit between simplicity and scalability.

In terms of following the interview guidelines I think I've managed to follow every topic requested. I created the git repo from the start and made sure to push changes and updates with comments every time it was possible. From here this documentation exceeds 600 words (double of what was requested), but always taking care to be objective and on point.

But overall, I think I've achieved quality. I paid a lot of attention to getting the animations, inputs and UIs to feel right and smooth. Also made sure the art and level design were creative while also consistent.