Blue Gravity Task

Systems summary:

For a better understanding, I will divide the main features of the prototype into paragraphs. This is not a detailed project documentation!

I started by implementing a generic **State Machine System**. At first, I thought it would be more useful. But it ended up being used just for the player who also didn't have many states. I used an interface that I already used in other projects, but I adapted it a lot for this prototype.

For the **Player Controller System**, I used Unity's new Input System and the previously mentioned state machine. The idea was to make the game playable on a Gamepad Controller, but I noticed that I would waste a lot of time to make the menu inputs, so I discarded it. And I used velocity to perform the movement.

I already had the **Dialogue System** ready from other projects and I only needed to make a few small adjustments. It uses scriptable objects (SO), so it's very easy to implement new conversations.

For the **Interaction System**, I again used the codebase from another project and adapted it to use raycasts depending on the player's input direction. This system is very interesting because I can use Unity events to perform different types of functions, such as starting a dialog or opening a menu. I also used a shader asset (All In One Sprite) to show when an object can be interacted with.

Now getting into the Store part itself. First, the **Cloth Class** was created as a SO so I can have easy access to its information. Second, I created a **base class for the inventories** that would manage clothing (Inventory, Shopping Cart, Clothing Rack), as they all used similar functions and properties. Third, I created the **base class for the UIs for these Inventories**, which involve slots for storing and managing the clothes. These UI classes derive from yet another class used for Menus that close and open. (An asset called LeanTween was used to perform these animations).

And then we are left with the **Changing Clothes System**, which it was not possible to complete due to time factors but is 90% ready. Making a changing clothe system of when using pixel art sprites is not as simple as in a 3D game, where it is possible to just enable and disable the meshes of the clothes.

After doing a lot of research on ways to create this mechanic and not finding anything that seemed minimally quick to implement, I built a method that involves taking the spriteSheets of each outfit's animations and leaving them registered in their respective ScriptableObjects. During the animation, instead of changing to a specific sprite, it calls an Animation event that takes the sprite at a certain index of the spriteSheet in the OS of the equipped outfit. Taking into account that all sprite sheets have the same number of frames, the animations would line up.

The problem was that the animator uses blend trees, which causes the animations events to be called erratically. For the system to work I would have to redo the Animator operation, which I wouldn't have time to do before the deadline.

Finally, 90% of the mechanics were implemented, including **Buy, Equip and Sell clothes. Walk, Interact, Dialogue, Add/Remove clothes from cart and try them on.**

Development thoughts:

At first 96 hours seemed like enough time to develop the prototype, so I thought I'd polish things right from the start, but as development went along, I ended up realizing that I wasted a lot of time on small details.

The problem with making scalable code is that usually more time is required developing its codebase so that in the future this will speed up the creation of new features. For this prototype maybe I took it a little too seriously when I could have done something simpler to save time, since the idea is not to continue its development. In the end, the code was well-organized and scalable, making full use of OOP concepts, events, scriptable objects, etc.

Right from the start I went after a pack of sprites to use in the prototype since I don't have much experience in this regard, which saved me a lot of time. I also found a pack on the internet for the UI, but I had to make several changes to use it. I spent a lot of time designing the game's menus and their animations, but I liked the result. The source for these assets is in the page below.

Also I had some data corrupted in the middle of development and although I made backups regularly, I ended up losing about 4 hours of work.

Finally, I loved the experience. Despite not having time to polish as much as I would like, (additional animations, sound and visual effects), I believe I have done a good job and I hope you also think the same.

sincerely,

Ramon Carvalho

Source to the assets used:

UI sprites: https://www.spriters-resource.com/fullview/146710/

Font: https://www.cufonfonts.com/font/pokemon-fire-red

Character and level sprites: https://limezu.itch.io/moderninteriors

Arrow Icon:
https://www.iconfinder.com/icons/731839/arrow_game_mountain_pixel_art_pixelated_triangle_up_icon

BackgroundMusic: https://freesound.org/people/Jay_You/sounds/460432/

Doorbell sound : https://freesound.org/people/maisonsonique/sounds/196378/

Shopping car icon : https://en.wikipedia.org/wiki/File:Ic_shopping_cart_48px.svg

Trash can icon: https://www.flaticon.com/free-icon/trash-can_542724

Backpack Icon: https://www.flaticon.com/free-icon/backpack_3313637