Blue Gravity Dev Assessment Documentation

Vincent Eugene Tan

Implementation Thought Process

After reading the specifications in this assessment, I immediately got excited about making a game similar in aesthetics to Stardew Valley. As such, I searched the Unity asset store for free 2D assets that would fit great to the pixel top-down aesthetic of the game world. However, the first problem I encountered was needing more options in the Unity asset store that matched my preferred aesthetic. Considering the time I spent searching for assets, I still did not manage to find a good player character asset that would fit both the aesthetic and requirements to implement a fitting skeletal system for player equipment. I decided to use the provided rogue character asset instead as it has numerous equipment options and it has proper sprite formatting to efficiently execute a player skeletal system.

To efficiently use my time, I started creating the tilemap level of the game world, while thinking of how I would implement the item system of the game. The tilemap creation for me was a tedious process as it would require me to paint the world map from scratch. I later realized that I could ease this by implementing a tile rule for painting, but with the remaining time left, I decided not to do it.

In scripting the item system of the project, I used scriptable objects to create each type of item that would have a distinction in its variables in each other. I decided to have 4 equipment slots (head, chest, arms, legs) for the player characters so that I could work with a manageable amount of items in the game. As such, each equipment type will have its distinct sprite types that are related to its body part. I design each item to have an ID, name, UI icon, body part sprites, cost, and item type. This equipment scriptable object is then passed to EquipmentManager for reference of the player’s currently equipped item. I also created two different types of items, that being ShopItem and InventoryItem classes. ShopItems can only be found and created in the ShopManger while InventoryItems in the InventoryManager. I decided that an interactive system for the inventory would be fitting for this demo, so I also implemented drag, and drop, and swapping of items for a more interactive UI.

After the complex implementation of the item system, I used the remaining time to implement the shopkeeper interaction and also simple NPC messages to give the game world a bit of life. I also added a MainMenu and AudioManager at the end to polish the demo.

There were some things that I could have implemented better in the demo, such as organizing the code a bit better, like abstracting InventoryItem and ShopItem to a base Item class. I could have also implemented an InputManager for all the game inputs so that there would be less hassle putting if statements in certain parts of the code. The player movement I implemented was a bit clunky and could be further improved as well. I also thought of creating a sorting system for both shop and inventory items but I didn’t have the remaining time to. The NPC behaviors could also be given more “spice” and more items and item types could be created with scriptable objects.