Blue Gravity Studios Test Project

By Kaz Born

Dependency Injection

I always opt to use Zenject in my projects, as it allows for more independent components. This way I can, for example, send off a signal that an item was bought, then other existing components hear this signal and add this item to an inventory, take away the cost of the item from a wallet, update the UI to reflect these changes, all without having to depend on all these components existing in the moment the signal was sent off.

Character Behaviors

Characters are made with various components, each controlling a single part of the character's behavior. The Character prefab has the basic behaviors of movement and animation, and other prefabs can build on top of this.

The Shopkeeper prefab adds it's own script to handle the shop item inventory and opening the shop when interacted with, while the Player prefab can interact with any interactable, as well as forwarding the player inputs to the movement component.

Interaction

Characters' interactions with the world is made through an interface named IInteractable and a component named Interactor. Any object can interact and be interacted with if these are implemented.

Character Sprite Layering System

I made the character with layers for each slot of item, like a paper doll. This way I can activate and deactivate each layer according to what the character has equipped.

Dynamic Sprite System for Characters

As for animating each layer, I figured it's better to have a single character animator with one of each animation and dynamically change the sprites for each layer. The alternative would be creating every single animation for every single item in the game, and this has very bad scalability.

To make the animation system with dynamically changing sprites, I used the code contained in this tutorial:

Unity3D: Replace Sprite Programmatically in Animation - Erik Moberg's personal homepage - photography, gadgets, DIY, and more

I heavily modified the code to pre-load every sprite sheet before the game starts instead of having to load them at run-time in the class SpriteSheetManager. The code that updates the characters' sprites is held inside the character's animation controller: the SpriteSwapper.