Storedirt Volley

by Juan David Payán

✨Description

"Storedirt Volley" is a small prototype that places players in control of a character who becomes ensnared by consumerism, resorting to unpaid labor in mining activities solely to acquire luxury items like Gucci clothing.

❓Q/A

What to Focus On:

- A comprehensive examination of the Player prefabs and their constituent components is recommended. Pay particular attention to the PlayerController, which utilizes Dependency Injection to delegate responsibilities to dedicated C# classes.
- Noteworthy features include the functional mechanics of the Stores/Inventory dashboards, which were developed under time constraints through iterative coding.

Achievements to Be Proud Of:

- A notable accomplishment in this project is the successful implementation of a Dependency Injection architecture within the Unity framework. This architectural choice was made to ensure high decoupling between system components.
- The development of layered Animators, although challenging, adds to the project's flexibility and scalability, especially when introducing new clothing items.

Areas for Improvement:

- It is worth considering a more streamlined approach to the layered sprite-sheet character. This could involve direct resource folder access and programmatic index changes instead of employing a separate Animator for each layer.

- There was an intention to implement a Hold-E mechanic for mining Gold, which remains a potential improvement for future iterations.

Time Allocation:

- The most time-consuming aspect of the project was devising a method to create modular and customizable sprite-sheet art in a layered fashion. This consumed approximately 70% of the development time. Other project components progressed smoothly.

Customization

The characters are structured into four distinct layers: the body, underwear, armor, and head accessories. Each of these layers possesses its own Override Animator Controller, all of which supersede an Empty Master Animator state machine. This master animator defines the available character states, encompassing idling, walking, and running in four distinct directions.

In order to achieve seamless synchronization across these layers, each clip within every layer must precisely match keyframes. Given that a skeletal approach is not employed, the standard animator parameters and transitions are not utilized. Instead, a custom script has been implemented to instruct all Animator layers to play a specific state, ensuring effortless synchronization.

Store/Inventory

To facilitate the purchase and equipping of items, Scriptable Objects were developed to serve as representations of articles that can be equipped, added to an inventory, or purchased. These Scriptable Objects, aside from containing fundamental attributes like a name, thumbnail sprite, and price (as expected), also include fields for defining the accessory type and its associated Animator Controller. This design approach streamlines the addition of new skins, requiring only the synchronization of keyframes with the aforementioned layers.

Implementing these Scriptable Objects in this manner, allows for the seamless expansion of the game's skin offerings, with the primary development effort centered around ensuring alignment with the existing animation layers.

.Interactuable/Interaction/Interaction Manager

A streamlined and straightforward bridge mechanism has been established to facilitate interactivity through the utilization of interfaces. This design philosophy is exemplified in both the gold mine and the cloth seller, both of which implement their own interactable interfaces. Even the gold ores that appear during mining adhere to this structure. These entities are responsible for detecting the player's proximity and subsequently initiating a request for interaction. This interaction request is conveyed to the player via a context-aware prompt, displaying options such as "Press E to do X." It is noteworthy that the UI panel, once active on the player's interface, receives injected information from the interactable, including the interaction name and the entire process. As a result, the player's role is simplified to merely pressing the 'E' key to engage in interactions.