SimWorld – System explanation

CameraSystem

- **CameraFollowTarget** – has a reference which transform to follow each frame with a certain offset and speed. In SimWorld for example we use it to follow the player.

Interactions

- **IMouseInteractable** – an interface to be implemented to any class whom the player can interact with a mouse click. In SimWorld for example we use it on the shopkeeper to open the shop.
- **IMouseHoverable** – an interface to be implemented to any class whom the player can interact with a mouse hover. In SimWorld for example we use it on the shopkeeper to display a chat bubble when mouse is hovered.

Paperdolling

- **PaperdollParent** – apply this component to an animated *spriteRenderer* that uses a *spritesheet*. All it does is have info on which sprite in *spritesheet* corresponds to the current sprite in *spriteRenderer*.
- **PaperdollChild** – has a reference to a *paperdollParent* and its own *spritesheet*. Applies the same index of sprite in its own *spritesheet* as the *paperdollParent* has. For example in SimWorld it's used for clothing being animated along with the players base skin.

Player

- **PlayerController** – has references to other player components attached and has control over them such as enabling and disabling them.
- **PlayerMouseInputController** – sends raycasts to find any *IMouseInteractables* or *IMouseHoverables*.
- **PlayerKeyboardInputController** – contains info on axis input in current and previous frame.
- **PlayerMovementController** – uses *PlayerKeyboardInputController* to grab input and move player accordingly.
- **PlayerAnimationsController** – uses info from *PlayerKeyboardInputController* and *PlayerMovementController* to set up animations on the player.
- **PlayerClothingManager** – contains functionality to add or remove equippable items (in this case clothing) on the player according to the type of clothing (Hats, Hairs, Outfits).

InventorySystem

- **Inventory** – static class that contains a list of items inside it along with methods that manipulate the said list.
- **InventoryManager** – displays *inventory slots* and *items* contained inside of them based on data from *Inventory*. Also controls what happens when said slots are clicked (prompts for an option to equip or to sell).
- **InventoryItem** – an inventory-representation of data for an item. Has data such as icon, cost and a reference to the actual item gameObject.
- **InventorySlot** – a button that can contain an icon for an item and a reference to the *inventoryItem.*
- **ClothingItem** – a data container for an item gameObject which tells us which type of clothes it is (Hats, Hairs, Outfits).

ShopSystem

- **Shopkeeper** – a *mouseInteractable* that displays a chat bubble when hovered over and opens the shop when clicked on.
- **ShopManager** – displays *shop slots* and items contained inside of them. Also controls what happens when said slots are clicked (prompts for an option to buy in case there is sufficient gold and if the item isn't already bought).
- **ShopSlot** – a button that contains an icon for an item and a reference to the *InventoryItem*.

UI.TabSystem

- **TabGroup** – contains a list of *tabButtons* to toggle between their corresponding data containers and images based on selection.
- **TabButton** – a button with a reference to its tab data container.
-

SimWorld – Thought process

I started off thinking about the basic player controller scripts. First thing that came to mind was getting player's input via keyboard. Having input data I've worked on player's movement, along with his animations. While working on animations I've encountered the problem of modular clothes working with the player's animations. After I've solved it using the paperdoll technique, I've moved on to creating an inventory and shop system where we have data on what's bought and not, what's equipped and what's not.

SimWorld – Personal assessment

In general, I am glad with how modular the classes are and how most of them follow some principles such as the Single Responsibility Principle. If I were to change anything, it would be the inventory and shop system. Classes could've had better names and should've been organized better. I think that part was the toughest for a short timespan.