**Max Wilson Game Programming CW1 24/25**

**Link to project:** <https://stulsbuac-my.sharepoint.com/:u:/g/personal/wilsom34_lsbu_ac_uk/Ec1WmFxoP0lDnvC8hzbtLUUBoNCzaF215jigkgw2NYeI5A>

**Progress Journal**

**My Programming Background**

-My background is mostly in Luau (Roblox Lua) making games/exploits on Roblox

*(Luau is a modified version of Lua 5.1 adapted by Roblox*

*for use with the Roblox Studio game engine.)*

-Designing systems from that perspective so will have to adapt them to work for C#

-Also more used to multiplayer development so in theory this should be easier since I don’t

have to account for client-server interaction

**Research & Building Unity/C# Knowledge**

-I’m most proficient in Roblox Luau and am currently working on a solo project which some of

these systems could be adapted for.

-When planning my systems it’s following the programming paradigms of Luau and I may not

know how/be able to script it that way 1:1 in Unity.

-My main goal is to familiarise myself with the UI design tools in Unity while using robust

programming logic that I can adapt for the player inventory UI system in my Roblox project.

**Differences Between Languages**

-Unity is more object oriented with more complex hierarchy (object-component based)

-Roblox makes use of ModuleScripts as an efficient way to create global functions and share

data between scripts.

-Upon some research [this reddit post](https://www.reddit.com/r/unity/comments/1ecpwcw/im_a_roblox_developer_how_do_i_add_a_shared/?utm_source=share&utm_medium=web3x&utm_name=web3xcss&utm_term=1&utm_content=share_button) on r/Unity recommends using plain classes and static

functions to achieve similar results to ModuleScripts.

-[This tutorial](https://learn.unity.com/tutorial/inheritance?signup=true#5c8924f2edbc2a0d28f48439) on learn.unity did a good job of explaining how to apply those concepts myself.

-Roblox Studio offers a more straightforward UI system with specialized UI elements that come with stricter constraints, while Unity provides greater flexibility with its more general UI objects, though this requires more effort to script functionality.

**Tutorial Concepts**

-4 Different tutorials detailing how to make 4 game systems in Unity

-They all should come together to form a final coherent prototype

**Ideas**

-Game Title screen (play button that spawns in your character, options button, etc.

-GUI with multiple frames you can cycle through to display different menus

-Respawnable instanced object with Healthpoints

-Sorting filter for player inventory GUI

**Possible Extras**

-Sword swing with hitbox/hurtbox that deals damage

**How Would They Connect**

-Title screen with options menu can make use of GUI with multiple frames

-Press play on title screen to transition to level with new HUD UI and playable character

-HUD contains inventory icon with keybind label to open inventory GUI

-Player inventory GUI contains buttons to cycle through frames.

-Frames contain scrolling frames displaying player inventory corresponding to item type

(weapons, items, potions, materials, etc)

-Scrolling frames can have their display order sorted by selecting a condition from a drop-down

(by rarity, by damage, by level, etc)

-Scene contains a module for creating instanced destructible objects with selectable

parameters, an example of which will exist in the level.

(parameters: health, auto-respawn, auto-respawn timer)

**System Design**

-Follow good coding practices

(Consistent script/variable naming, clean hierarchy structure, script structure)

-All the systems to be able to function as independently as possible as modules

(Don’t hardcode values, don't tangle up scripts/functions)

**Title Screen**

-Could have its own scene or could just be a UI overlay with the level scene loaded.

**Pros & Cons To Each Approach**

**UI Overlay:**

* Easier to implement
* This type of implementation would be useful for online games which is more applicable to my own projects
* Not an over-engineered solution for my problem

**Multiple Scenes:**

* Good utility for use in single player games with multiple levels (e.g. traditional platformer games)
* Can still be adapted for multiplayer games easily enough
* *“Shouldn’t be too much extra work”* ~ 😬

**Design**

-Multiple scenes, Title Scene and demonstration level scene

-‘LevelManager’ script to handle scene switches

-When launched, game will load Title Scene

-Title Scene will be easily customisable to display title art of devs choice.

-Title Scene will also load Title UI Overlay (Canvas) which will contain customisable UI buttons.

-For the tutorial the Title UI will be configured with both ‘Play’ and ‘Options’ buttons.

-Upon LClick, Play Button will use the LevelManager script to load the demonstration level

scene.

-Upon LClick, Options Button will load a GUI that can be customised to contain game options.

*(See Multiple Frames GUI section)*

**Multiple Frames Inventory GUI**

**Design**

-‘UI\_Handler’ script to control the functionality of the UI.

*(Currently displayed frame, linking buttons to frames, switching displayed frame)*

-UI.Refresh() function to refresh the UI in case of player changing inventory sort method, etc.

-Table containing player inventory data.

*(Probably a ScriptableObject with example inventory depending on time*

*constraints could make proper database with functions to parse data)*

-Objects scripted to function similar to Roblox specialised UI elements to create the inventory

UI frames.

*(Mainly an equivalent to Roblox ScrollingFrame UI Object)*

-Scrolling Frame:

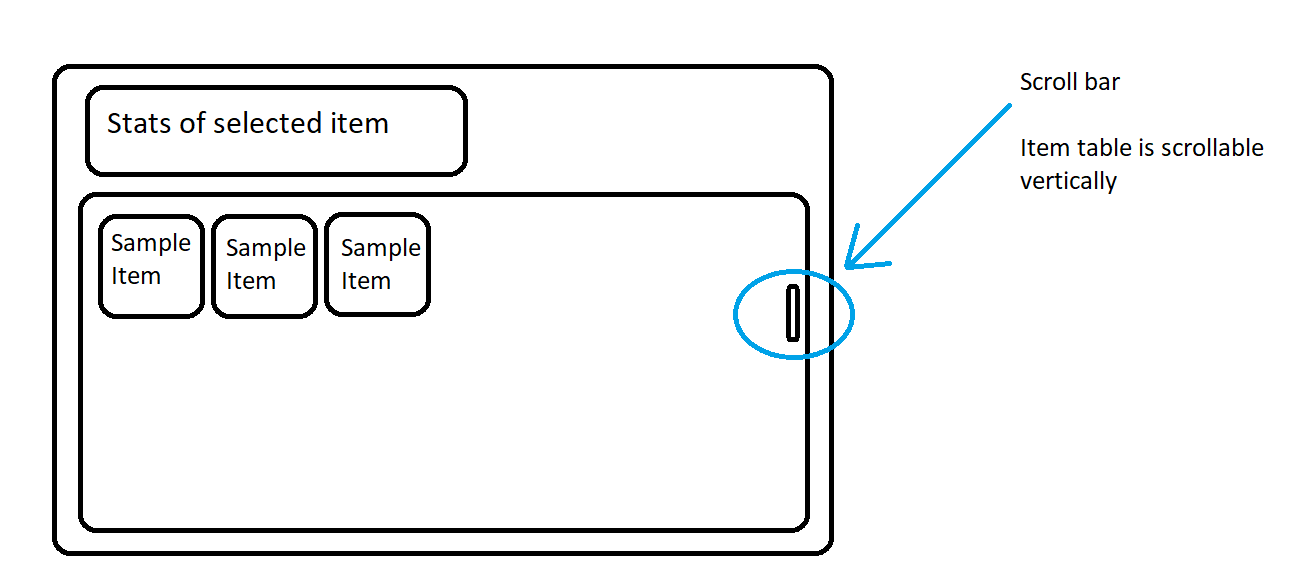
* Frame containing a vertically scrollable table of player items
* Function to populate list with buttons representing player inventory items

**UX**

* On screen inventory icon *(could be clickable button)* with labelled keybind
* On keybind press, open UI frame containing player weapon inventory
* List of image buttons adjacent to UI frame.

*(button icons representing item types: weapons, armor, potions, etc)*

* On click, image button closes current on screen UI frame and replaces it with the frame corresponding to the icon on the button.
* If pressed button corresponds to currently active UI frame, or the initial inventory keybind or escape key are pressed, inventory UI is closed.



*(example sketch)*

**Sorting Filters**

-Change the order that UI elements are displayed e.g. sorting inventory items by rarity/name/lvl

-Lightweight modular design that you can easily add your own filters to

-For tutorial purposes I’ll create 2 filters (By Rarity & by Level) which can be selected from a

dropdown menu in the inventory UI frames to sort player inventory items.

**Design**

-Player items must all contain values representing their data (rarity, lvl, name)

-Add a check so no invalid data gets parsed.

*(check for valid data in each value e.g. make sure item.rarity = a string*

*in the ‘Rarities’ table)*

-‘SortHandler’ script containing functions to handle the logic for each sorting algorithm.

-SortHandler:

* Required data for each sorting algorithm to function.

*(e.g. sort by rarity needs extra data in the form of a table containing the names*

*of each rarity with an ‘order’ value assigned to them to know what order to*

*sort compared to sort by level which utilises numeric values to determine order.)*

* Parse the sort algorithm the player selected in form of a string e.g “byRarity”
* Sort functions:
  + Add items to the table in groups

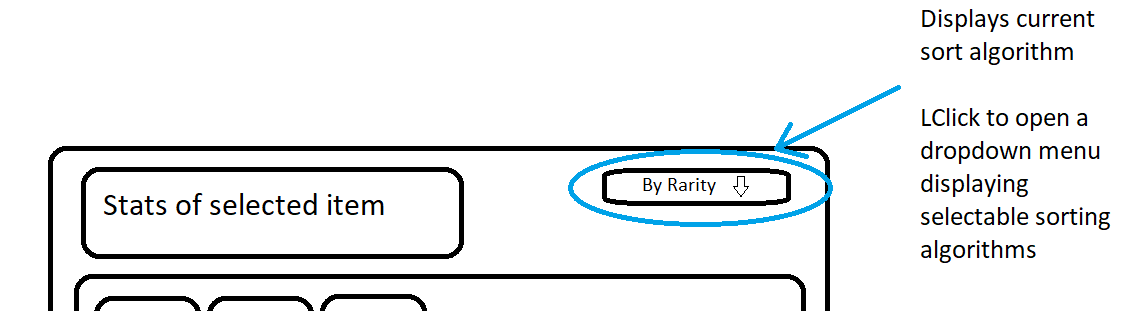
*(e.g. if Common.Order = 1 the items with Rarity.Common will be added*

*first and so on)*

* Update player data with the new sorted table and use UI.Refresh() so that the inventory UI accurately displays the updated order

**UX**

* Button with text label in top left or right *(TBD)* of inventory UI frames containing the name of currently applied sort filter (e.g “By Rarity”)
* Upon LClick, dropdown list containing buttons with text labels representing each selectable sorting algorithm.
* Upon LClick, UI is refreshed to accurately display the newly sorted item order and dropdown menu is closed. Dropdown button label text is updated to display newly selected sorting algorithm.



*(example sketch)*

**Implementation**

-Created new 2D project in Unity 2021.3.11f1

-Made a group of new folders inside the assets folder to keep stuff organised.

*(Scenes, Scripts, Prefabs, Materials, UI, etc)*

-Text Mesh Pro & Unity UI packages downloaded from the package manager

**Title Screen**

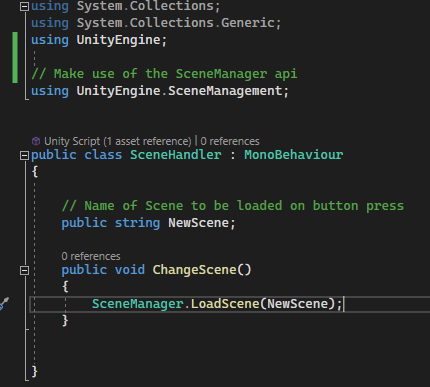
-Create a ‘Title Scene’ in Scenes folder

-‘Sample Scene’ will be the level scene so rename it accordingly

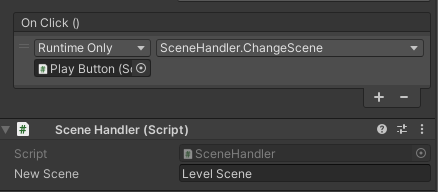
-Open Title Scene, select Main Camera and set environment to Solid Color

-Create a Canvas containing a Text Button

-Create new ‘SceneHandler’ script



-Add the script to the button and set it up in the inspector like so



**Inventory GUI**

-Create Canvas in LevelScene

-Inside Canvas create GameObject renamed to PlayerInventory and create TextButton

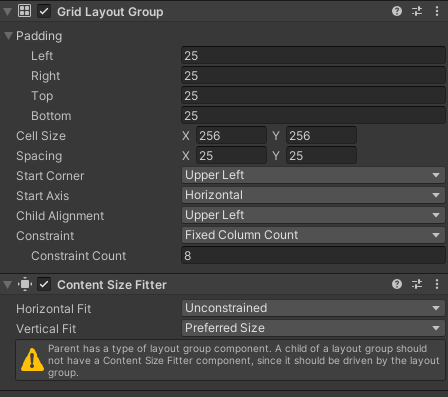
renamed to Inventory Button

-Inside PlayerInventory Create UI Panel named Weapons Panel with empty ‘Content’ GameObject inside of it.

-Create Button inside Content and set it up how you want your inventory item frames to look.

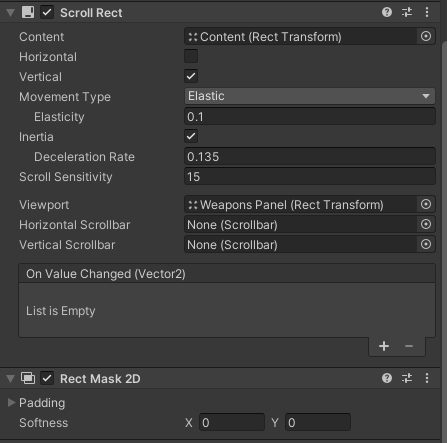


-Add GridLayout & Content Size Filter Components to Content and set them up in the inspector like so



-Duplicate your Sample frame to populate the grid and adjust the GridLayout settings where necessary.

-Add ScrollRect & RectMask Components to Weapons Panel and set them up like so

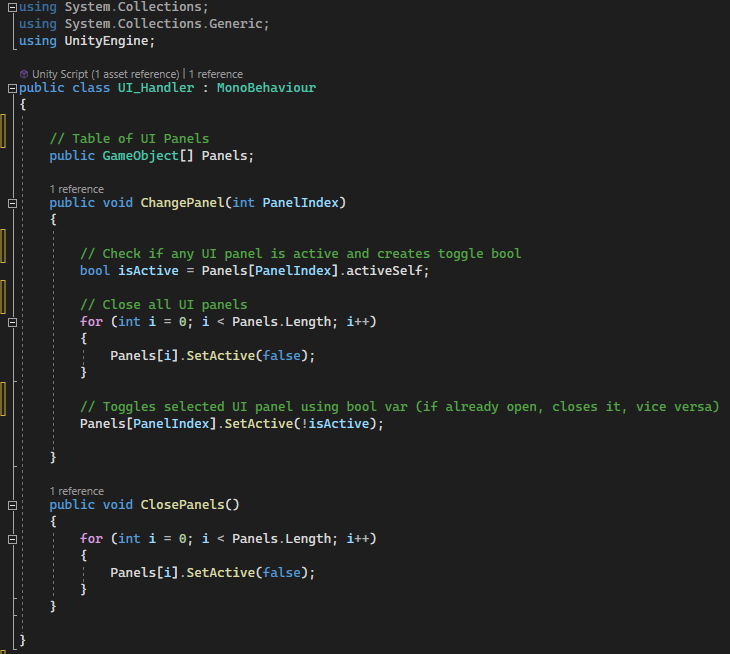


-This makes the list scrollable and keeps the inventory table within the confines of the UI

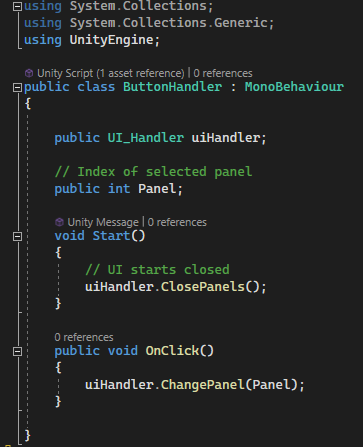
-Duplicate Weapons Panel twice and rename to Items Panel & Potions Panel, recolor the

sample item frames for visual distinction.

-Create UI\_Handler script

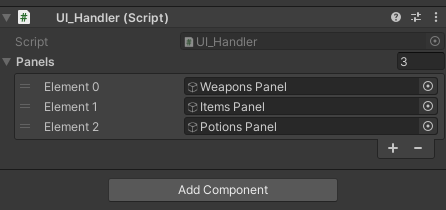


-Create ButtonHandler script

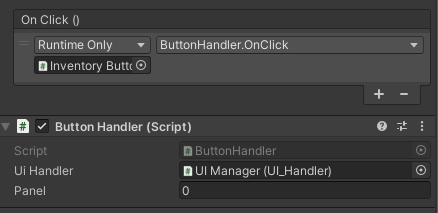


-Inside Level Scene Create ‘UI Manager’ GameObject, add UI\_Handler script as a component

and setup like so



-Setup Inventory Button like so



-Panel = 0 because index 0 in Panels table is Weapons Panel and we want that as the default

inventory page

**Inventory Switcher**

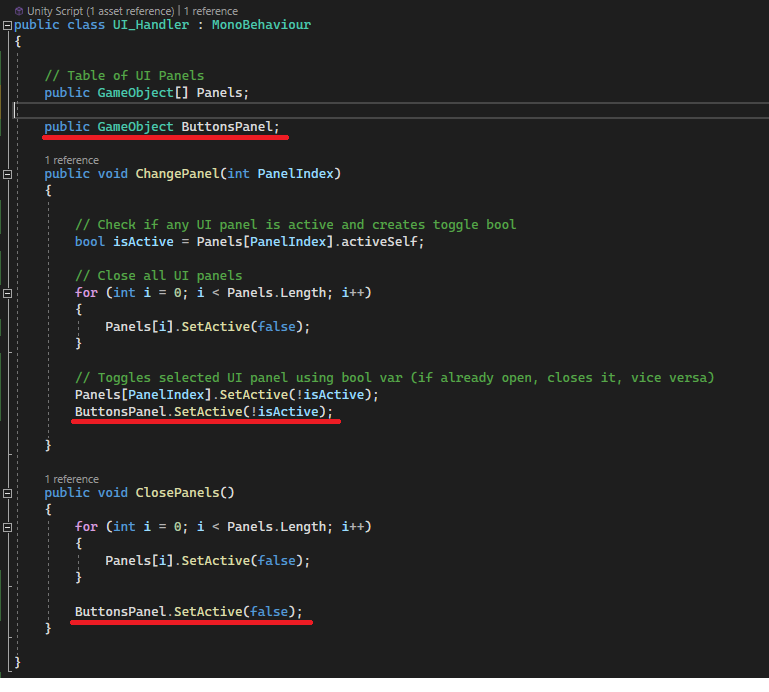
-Create new Panel in Canvas ‘Buttons Panel’

-Inside Buttons Panel add VerticalLayoutGroup Component and create 3 buttons; Weapons,

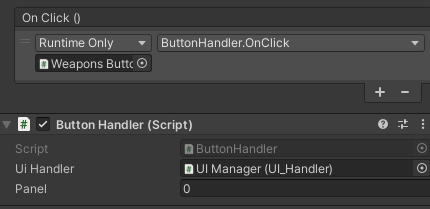
Items, Potions



-Make these changes to UI\_Handler Script to handle toggling visibility of Buttons Panel.



-Setup each Button in Buttons Panel like so



-Ensure the correct Panel index is selected for each button

-Would have been more efficient to handle it all through the buttons panel with scripts rather

than each button individually.

**Conclusion**

Overall I didn’t achieve all the functionality that I wanted and I wasn’t able to go into as much detail in the tutorials as I would have wanted to because of my poor health/low attendance.

I would have liked to implement more UI functionality such as the inventory grids being populated with PlayerData from a database rather than just sample item frames.

Also I wasn’t able to implement the sorting filters due to the lack of proper data because there was no data for me to sort or functions to handle populating the inventory grid with item frames.