Project Tutorials, Learning Journal & Source Files with a Demonstration Video

By Preet Sameer 4141851

GitHub Link for project:

<https://github.com/preetsameer/PreetTutorialsProject>

One Drive Link for project:

<https://stulsbuac-my.sharepoint.com/:f:/r/personal/sameerp_lsbu_ac_uk/Documents/Programming%20Assignment-Preet%20Sameer%204141851/Tutorials%20Oat?csf=1&web=1>

YouTube Link to Project Demonstration Playthrough Video::

<https://youtu.be/FRhHP890gYY>

Tutorials Include:

* Drag and Drop Inventory System
* Buttons
* Click to Move
* Dialogue System

Learning Journal is located at the end of the tutorials.

Drag and Drop Inventory System

For this tutorial, I have referenced from this YouTube Video: <https://youtu.be/kWRyZ3hb1Vc?si=xVyNZbGi0AbxXCAH> - Drag and drop in Unity UI - create your own inventory UI! By Coco Code

* **In this tutorial, you will learn how to create an Inventory menu with draggable items that you can drop in inventory slots.**

Step 1

Establish a space for your items; for this tutorial, I've chosen to implement a grid layout. A basic background and rectangular box will be enough. You will also need to import all necessary items, such as things you can put in an inventory for your game, into your assets, including the background.

Step 2

In your hierarchy, create a canvas by **right clicking>UI>Canvas**.Under your canvas, create another Canvas and name it Inventory. Under Inventory, **right click>UI>Image** and name it background, do this again and name it box. Select your background, then insert the image from your assets into the source image section within the inspector window.

Step 3

**Right click on box>UI>Image** to create a new shape that will be used to make a grid. Name this shape as ‘Grid’. Adjust the size of the grid and reduce its transparency.

Step 4

To create the inventory slot, right click on the grid and once again click on **UI>Image**. Name this as ‘InventorySlot’ and adjust the size of it. You can customise the appearance of the slot by incorporating your source image. In my case, I'll opt for simplicity and merely alter its colour.

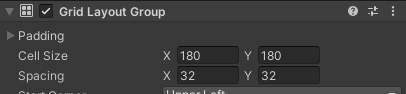
Step 5

Since we will be using this Inventory Slot multiple times, we will be creating a **prefab** out of it.

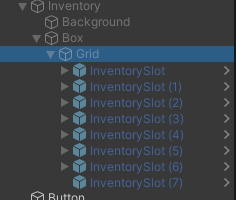
To do this; drag ‘InventorySlot’ from the hierarchy into the prefabs folder in assets. By doing this, we can generate numerous instances of this GameObject that will all share the same properties.

Step 6

We will be cleaning up our Inventory Slots as there is an issue with spacing. The solution to this is simply clicking on the Grid parent and adding a new component ,in the inspector window, called ‘**Grid Layout Group**’. Next, customise the Cell Size according to your preferences, I have set mine to 180 by 180. Additionally, I've configured the spacing to 32 by 32. Feel free to modify these values to better align with your game requirements.



From now on, whenever you add a new game object, it will automatically be positioned within your grid layout.



Step 7

To create the draggable items; **right click on the first inventory slot> select UI>Image**.This is done to generate a new object within the inventory slot. You can incorporate your image into the source image within your inspector window. Name this object 'Item' and resize it to fit within the slot.

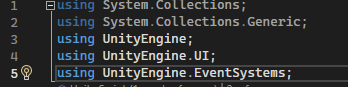
Step 8

Generate a new script by adding a new component through the inspector window. **Add component> type DraggableItem>Enter>Open your script.**

Step 9

To implement interfaces related to player actions, add this to your script;

using UnityEngine.EventSystems;



In this tutorial we will need three interfaces relating to dragging objects, these are:

public class DraggableItem : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler



Step 10

Right click on each interface separately and select ‘**Quick Actions and Refractorings**’**> Implement Interface.** These three methods will be triggered when the player starts the drag, during the drag, and when the player releases the mouse.

Step 11

To test this out, we need to replace ‘Throw New’ exceptions with:







After these changes, save your script.

Step 12

We need to be able to move our item with the mouse while we are dragging. To do this, in  type:



After this modification, the item will be copied from the current mouse position while dragging.

Step 13

There is a slight issue that needs to be fixed, when you drag your item across the grid, it disappears under the other slots. This is because in Unity, objects that are placed last are placed at the very top of the scene, setting the first slot as the parent positions our item below everything else in the hierarchy. To address this issue, we need to unlink our item while it's being dragged and reposition it at the bottom of the hierarchy. This way, it will be positioned above every other object in the scene. To do this we need to assign a new variable;

The purpose of this variable is to save what was the original parent of the object, then assign the parent again after the player finishes dragging.

Also remember to hide this in the inspector; 

Step 14

In , type 

And, in , type 

In , we need to set the canvas as the parent of our object during the drag, do this by adding;



This points to the canvas wherever we are in the hierarchy.

Step 15

You can now test dragging your item; it will follow the mouse as you drag it. You can see your item does not go back to its original slot, this is because we have yet to implement **Snapping**.

Step 16

We now need to figure out how to snap items to slots. To do this; click on the inventory slot and add another grid layout group, this way we’ll create a one item grid inside every inventory slot.

Resize the cell to your desired dimensions; I'll maintain it at 180 by 180 to match the item object. Then, modify the child alignment to middle centre. This process will always keep the item object in the middle of the slot.

Step 17

Before testing this out, make sure you update your prefab, every modification you have made will be added to every other inventory slot. To do this, simply **click Overrides in the top right of the inspector window>apply Grid Layout Group**. Now, every inventory slot incorporates the grid we've implemented.

Step 18

Press play and click on the item; you can now drag it around the grid. Upon releasing the item, it will snap back to its original slot.

Step 19

For dragging items across the grid, you'll need to insert a few lines of code. Click on your inventory slot, within the inspector window **add component> type InventorySlot>Enter>Open your script.**

Once again, add 

And implement IDropHandler like this:



**>Quick Actions>Implement Interface**

This method will be triggered when an item will be dropped in the inventory slot.

Type This is necessary to adjust the `parentAfterDrag` variable of the draggable item. This ensures that in the `onEndDrag` method, the item will be assigned to the new parent.

Lastly, in the InventorySlot script type;

Step 20

Upon testing your game, you may observe that the item does not remain in the slot where you dragged it. This occurs because, upon releasing the mouse pointer, the game checks what is directly under the mouse. As it turns out, when the mouse is released, there is already an item under it. Since it's not possible to drop an item onto another item, the dragged item returns to its original parent. The solution to this is to hide this object from any other mouse inputs during the drag process, so that when the mouse pointer is released it will ignore the item underneath and will do all the checks with the inventory slot as intended.

To do this we will be adding a Raycast.

At the top of the DraggableItem script, create a new variable called;



To make this work we will need to add;



Under BeginDrag type;



And in OnEndDrag type;



This is so that any interaction with the item will be possible after it is placed in the slot.

Step 21

Save your script and assign it to the Item image in the inspector window. Press play and your item is able to be dragged and dropped into different inventory slots.

Step 22

Create a prefab of the Item and duplicate it, putting all the items within the allocated inventory slots. Once you press play, your items all work together and you are able to drag and drop multiple items into different slots.

Step 23

If you notice once there is a slight issue; if you want to drag an item into the same slot of an occupied slot, it will place the item underneath the grid. To fix this we need to generate an if statement in the InventorySlot script. The purpose of this statement is to check if there are any children in this slot.

Type;



Save your script and hit play once again, you will see that it is now no longer possible to add two items into the same slot.

**You now have a fully functional inventory system!**

Buttons

* **In Unity, adding a button to trigger a specific function, such as turning something on or off, is a straightforward process. Follow these simple steps:**

Step 1:

Launch up Unity and start a 2d scene, make sure you have a **canvas** in your hierarchy. To add a canvas you must right click in the hierarchy window and select **UI>Canvas**. This will create your canvas.

Step 2:

In the hierarchy window, under **canvas**, right click and select **UI>Button TextMeshPro. This will add your button.**

Step 3:

Adjust your button the way you need it to be shaped for your project. You can do this by modifying the button in the inspector window or selecting the button within the scene view and drag your button to the size and shape you want. If you want to add an image, click on the button and in the inspector window select source image. There you can add your image, make sure your image is already in your assets.

Step 4:

Create a script by adding a component in the inspector window. Name this script something that will suit your needs, I will be naming my script InventoryButton. This button’s function is to turn my inventory menu on and off, but you can apply this to anything you would like to turn on and off.

Step 5:

The function you want to create is a toggle on and off menu, to do this we must reference the game object, the inventory in this instance. Then we add an if statement;

**if (inventory.activeSelf)**

**{**

**inventory.SetActive(false);**

**}**

**else**

**{**

**inventory.SetActive(true);**

**}**

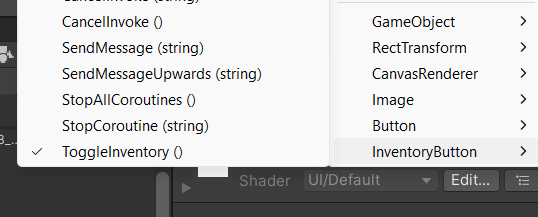
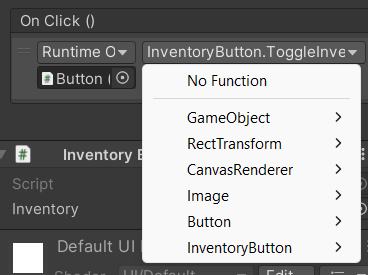
This means that the inventory will be set as active when you click on the button, and when you click off it, it will not be active.

Step 6:

Once you have written your script, save it and your project.

Step 7:

Drag your object, in my case this would be the inventory object, into the script slot of your button, under **On Click ()**. There is also a drop down menu under On Click (),this is your **script behaviour**. Click on script behaviour and find the option you just created in your script, for me I would have to **select ‘InventoryButton’** then **‘ToggleInventory’**.



**Remember to save all your work and you should end up with a working button!**

Click to Move

For this tutorial I have referenced this video:

<https://youtu.be/5KLV6QpSAdI?si=DKJq5iu8x4fti8eF> - Simple 2D CLICK to MOVE Unity Tutorial by BMo

* **Click to move will allow your player to move wherever you right click on the screen, here’s how to do it:**

Step 1

In your Unity project, add a square in your assets, this will be your player. You can also change its appearance, add your own image in your assets and change what your source image is in our inspector window.

Step 2

Adjust the size of your object and name it anything you like, I will be naming it **player**.

Step 3

Drag your player into the scene

Step 4

Create a new script in your assets and name it **MoveToMouse**.

Step 5

In your script, we will be starting with controlling the move speed of the player, how fast it will move to the mouse. To do this we will need to enter a

**public float speed**. You can add any speed you like and can adjust it later but my player’s speed will be **5f;**.

Step 6

We will now need a **PrivateVector3 target;** and in Start we can set the target =. Transform.Position. This allows the target to be in its original position, it will not be moving at first.

Step 7

In Update, we will be creating an if statement. This will be **if(Input.GetMouseButtonDown(1)).** 1 meaning right click button and 0 meaning left click button, in this case I will be using the right click mouse button.

Step 8

You can't just finish with mouse position**,** you need to be able to convert where you are clicking into a **Vector3** position>to feed to the transform. To do this, it's fairly simple;

**if(Input.GetMouseButtonDown(1))**

**{**

**Target = Camera.main.ScreenToWorldPoint(Input.mousePosition);**

**}**

\*this will target the camera you are using, you can enter any other cameras you have set up but the purpose of this tutorial I will be using the main camera\*

This line of script will convert where you click on the screen into a world position that you can feed to the transform.

Step 9

As this is a 2D game, we aim to prevent the sprite from moving within the third dimension. In a 2D environment we don't really want to touch the z-axis. To do this enter:



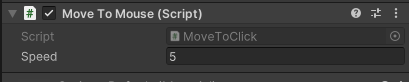
Step 10

This is the last line of code, it includes the speed at which we want to traverse Vector3.



Step 11

The last step involves dragging the script to the player. Start your game and once you left click or right click, depending on what you chose, your player should be moving to your mouse at the speed of what you chose.



**Dialogue System**

For this tutorial I have referenced this YouTube video: <https://youtu.be/1nFNOyCalzo?si=0dCfZDiTQZnt80Qq> - UNITY 2D NPC DIALOGUE SYSTEM TUTORIAL

* In this tutorial, you will learn how to make a fully functional dialogue system with NPC characters in your game.

Step 1

In your asset library, include a player character that you can move around. For this tutorial , I'll be using my player character introduced in my "Click to Move" tutorial. You will also need to import an NPC sprite and a close up of the NPC sprite, this is optional but in this tutorial I will be using the same image for my NPC for my close up image.

Step 2

Drag your NPC sprite into the hierarchy, this will create a new GameObject. Name this whatever you like but I will be naming it ‘**NPCBilly’** as it is the name of my character.

Step 3

Click on your player GameObject and create a player tag in the inspector window. This is important as it will be used in our script.

Step 4

Click back on NPCBilly (or your chosen name) and attach a Capsule Collider 2D from the components in the inspector window. Customise its size according to your preferences.

Then, add another collider, this will be a Box Collider 2D and once again adjust its size according to your preferences.

Make sure both of these colliders are set to **‘Is Trigger’.** This step ensures that when your player is within the NPC's range, it triggers the activation of the dialogue box.

Step 5

Create a canvas by **right clicking on the hierarchy>UI>Canvas**. Under canvas, **right click again>UI>Panel** to create a new object called ‘**DialoguePanel’**. Resize and position the panel according to your preferences. , you are able to add any background images but I will keep mine basic for the purpose of this tutorial.

Step 6

To design the contents of the panel,within the panel as a child, **right click>UI>Image**. This will be the profile close up image of the NPC character although this step is completely optional. Once you add your image and name it **‘NPCImage’** and add your image as a source image within the inspector window. Scale and position it in the panel.

Step 7

To add text in the dialogue panel, right click on **DialoguePanel>UI>Text-TextMeshPro.** Scale and position in the panel and leave no text.

Step 8

Head back to your canvas and scale it with the screen size in the **Inspector window>Canvas Scaler> Reference Resolution>x1920>y1080.**

Step 9

Disable the dialogue panel in the hierarchy.

Step 10

Click on NPCBilly and add a new script called **NPCBilly**.

Step 11

Start off by deleting the start functions and insert;



for this code to work.

Step 12

We are going to create some variables, these are;



public string[] dialogue; will contain all of the sentences for the dialogue we will write.

Step 13

Type;



Which will help us find a position in the string.

Step 14

We will be creating this;



To type our text.

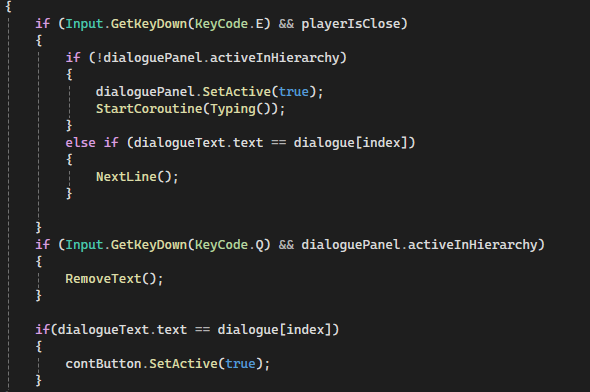
And;



When our player is in range

Step 15

We will then be adding these lines of code;



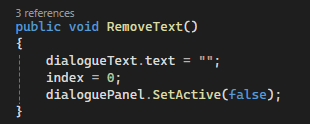
You are able to change the KeyCode to any key, for this tutorial I will be using the letter E to trigger the text.

Step 16

Add this . By using TMPro; at the beginning of a script, you are importing the TextMesh Pro namespace into your script. It streamlines the code and enhances readability while handling text-based components within Unity.

Step 17

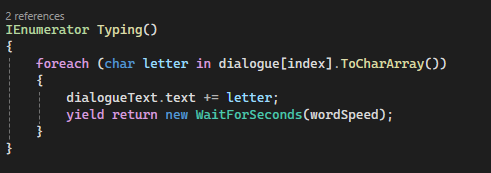
To make your exit function, where your text will reset when clicking off your dialogue, type;



Add  to your else statements.

Step 18

To create an interesting typing effect where one letter follows the other, you need to add an Enumerator.



You can change the speed of your word speed to whatever you would like.

Step 19

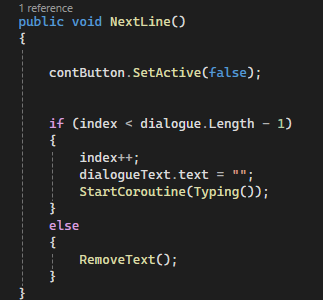
Inside of our update function, we will add;



StartCoroutines are usually used for tasks that involve animations, delays or other processes that gradually unfold. In this case it is ‘Typing’ which is why we have added it to the StartCoroutine function.

Step 20

Type this code;



The purpose of this is to add lines to our text, one line after another. You will be able to type in your text in each instance you create. This is for your continue button, after you press continue, your next line will appear.

Step 21

Save your script. In the inspector window, there will be new slots that you have created in your script for your NPC.

Drag your GameObjects to the respective slots, they will all have the same names;

* DialoguePanel>Dialogue Panel
* DialogueText> Dialogue Text

Step 22

Set the amount of sentences you want in the dialogue array, I have set mine as 2 but you can do as many or as little as you’d like.

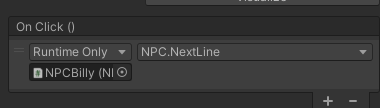
Enter your lines in each element and set the Word speed to your liking for example 0.06.

Step 23

Enable your DialoguePanel so we are able to create a new button. Right click on DialoguePanel>UI>Button-TextMeshPro. Arrange and place your continue button according to your preferences.

Step 24

Add a new OnClick() event to your Continue Button in the inspector window.

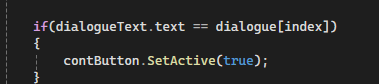


Drag in your NPC GameObject and add in the NextLine function we have just created in the script.

Step 25

Go back into your NPC script and create a new GameObject so that we are able to refer to our button. At the bottom of the Update function, create another if statement.

We want to set our button to active once our text has finished typing;



Step 26

At the start of our NextLine function, we'll set our continue button to false. This prevents the player from skipping ahead in the dialogue.



Step 27

Save your script and go back to your project in Unity. Drag your continue button into ‘NPCBilly’ and disable the continue button, then disable the dialogue panel.

**Press play and test out your dialogue, you should be able to move around and once you get close to the NPC you can activate your dialogue by pressing E, hit continue and more lines of dialogue will type out.**

Learning Journal

| DATE | PROBLEM/UPDATES | SOLUTION |
| --- | --- | --- |
| 10/10/23 | I have never properly coded before and I was very afraid of using C Sharp on Unity. It was a very intimidating experience for me, so I had to start learning about Unity and coding in general. I was completely new to this and had absolutely no idea how I would start my project. | I started watching beginner tutorials on YouTube so I could gain a general understanding of Unity and its many elements. I took a liking to 2D, because it is very beginner friendly. |
| 17/10/23 | Understanding Unity’s interface was a challenge in itself, I was very confused. It is not like Photoshop or Maya. I was very stuck on how I would manage my assets and prefabs etc. Even starting a new project and saving it in my folder, One Drive, and Hard Drive was a struggle.  I found myself drowning in the amount of Unity Learning resources, there's too many and yet my questions were still not answered. I still felt very lost and not in my element. | I just had to stick it through and find the correct resources to help learn how to save my projects and learn Unity’s interface. I also asked for my professor’s help as well as the teaching assistant’s help. |
| 24/10/23 | At this time I still did not understand what the assignment was about and what I had to create in my project. I also did not know the criteria at which I had to achieve to do well in this module. | I went on the Moodle and reread all of the assignment details and took notes on the most important parts of the assignment.  I asked my professor what I should be doing and was given some good advice. |
| 31/10/23 | I started looking for games that I enjoy playing and took elements from these games, this was part of the advice I was given. These games included Sticky Business on Steam, Animal Crossing on the Switch and other inspirations I got from Internet meme culture. | /// |
| 07/11/23 | I searched for specific elements that I wanted to create. I had too many that were too difficult to create with the time span and little knowledge that I had. I wanted to create walk cycles, multiple NPC characters that the player could interact with, moving NPCs that walk around, a custom cursor, dialogue with multiple choices, an interactive inventory menu where you can drag and drop items to a player into the game and lastly a title menu screen. I had overscoped my project and I had to narrow everything down. | Once I had narrowed my 4 tutorial elements down I had come to the conclusion to make click to move, a dialogue system, a drag and drop inventory system and a title menu screen. |
| 14/11/23 | I had realised that a title menu screen was too similar to the inventory system and I had already decided to make 2 other UI elements of games so I decided to drop the title menu screen. However, this meant I had to look for another element but I had no idea what I could do. | I thought, while I am in the process of creating the other elements I will be able to find the fourth game element to create in Unity. |
| 21/11/23 | I started working on my click to move element and right from the get go I was struggling. I watched a YouTube tutorial I was using to reference from multiple times to understand what was going on. I finally found my footing and started to get to work. The first issue I encountered was how to insert an image in source images for my player. These YouTube tutorials don’t completely start from the beginning and go very fast, they expect the viewer to know the basics straight away, and clearly I did not.  The name of the script should have no spaces, nor the GameObject, this caused an issue with starting the script but it was a quick fix. Another issue was after completing the script, I dragged the script onto the Player, it bounced back and didn't let me implement it on the GameObject.  At first I did not understand my errors and I kept trying to change the script, thinking that would work but it actually made it worse. I restarted the script however the outcome was the same as before and I was stuck.  Then I was stuck on how to actually play and test my game, I could not find the play button and was troubled once again.  When creating my inventory system, it was relatively easy, despite the lengthy process. I had already gained some confidence from the click to move tutorial. I was ready to start making my inventory system.  When it came to the script, I ran across a few issues. One of them was my Debug.Log function not working, it stated that this ‘does not exist in the current context’. I was struggling to find a solution for this and had to ask for help. | To insert an image in the source images, I just had to experiment with the hierarchy and the inspector window and I eventually figured out how to do it.  Most of the errors involved ‘;’ missing from lines of script as well as missing ‘()’.  I had not realised that at the bottom of Microsoft Visual Studio, there is a window where you can see the errors you have made and what line so you can easily point out the mistake you made.  I went back into my code and cleaned up my script and dragged it into the player, it ended up working.  I went on YouTube and found where the play button was, it was an easy fix.  The solution to the Debu.Log issue was adding brackets that were missing in my code. This then made my function exist in the current context. This error took up 21 errors, that were all fixed just by correcting the script by adding brackets. |
| 28/11/23 | To complete my Inventory system, I had to fix a couple more issues that I had. One of these was that when you pressed play, my inventory screen was not popping up, it was nowhere to be seen in the scene.  My canvas was not in the game screen and it was much larger, I was not able to change the size, that's why I had to experiment with changing the screen set.    (The small circle at the bottom left is my game scene)  The last issue I encountered with my inventory system was that I was struggling to figure out why my items disappeared when I duplicated my inventory slots after creating a prefab for the slots.  The biggest issue, that was also the most stressful, was how I would be combining all my game elements in one scene on Unity. This took me a whole week to wrap my head around and still it confused me. I asked around for help from my peers yet no one really gave me good answers and just expected me to figure it out on my own. I tried to find answers online through YouTube videos however I didn't get much help from them. I even posted my questions on the Unity forum.I got some answers however they were not relevant enough to answer my question as I didn't really know how to ask the right questions about my specific issue in my project.  Lastly, creating my dialogue system. This was the hardest element to create. Some tutorials for Unity on YouTube were either too detailed or too old to reference from. The one I had used, even though it was only a year and a half old, had used an old version of Unity. I noticed this quickly as some functions and lines of script were not the same as what I had to use. When dragging my DialogueText into the script slot, it did not let me and bounced back. I understood that this was an issue with the script not being up to date. | This was a simple fix and all I had to do was change the screen set. To Screen Set-Overlay from Screen Set-Camera.  This was not a part of the tutorial on YouTube as my issue was not the same nor was my inventory the same as the tutorial.  All I had to do was move the items under each respective GameObject, in my case under each inventory slot, and this issue was fixed. Rearranging parent and child objects was all it took to fix this issue.  I took a short break from my work and later spoke with my professor and teaching assistant and got valuable information. In the end I was able to create a cohesive project.  I looked through the comments to see if anyone else had the same issue as me. Someone in the comments had typed out code that they used because they had the latest version of Unity. I had to add ‘using TMPro;’ in my script as well as ‘public TextMeshProUGUI dialogueText;’. In the YouTube video, they had not used these lines of code but it was necessary for me to add. Once I added these, my script was working and I was able to drag DialogueText into the script. |
| 05/11/23 | This week I had finally found what my fourth element would be and that was buttons. I had no idea how to create a button and it was very hard for me to find something on YouTube or on the internet that would fit what exactly I needed. | I had asked my professor to help me create a button for my inventory system and I wrote down what I had learnt to use for my tutorial. It was easy enough to create so I didn't really have much trouble with it. |

Links:

GitHub Link for project:

<https://github.com/preetsameer/PreetTutorialsProject>

One Drive Link for project:

<https://stulsbuac-my.sharepoint.com/:f:/r/personal/sameerp_lsbu_ac_uk/Documents/Programming%20Assignment-Preet%20Sameer%204141851/Tutorials%20Oat?csf=1&web=1>

YouTube Link to Project Demonstration:<https://youtu.be/FRhHP890gYY>