LEVER PUZZLE SYSTEM V2 - V1.2
DOCUMENTATION

Contents

Introduction..........2

FAQ..........3

Manual Setup – Initial #1..........4

Extending the Lever System..........10

Contact..........11

Online Documentation..........11

Patch Notes..........12

MATTHEW RUKAS – SPEEDTUTOR
WWW.SPEED-TUTOR.COM
SPEEDTUTORUK@GMAIL.COM

Introduction

Thank you for purchasing the “Lever Puzzle System”. This is a Raycast based system where the aim is to pull the levers in the correct order, once the limit is reached you can press a button to test if the order is correct. Each of the levers has its own number that you can set, and the “LeverController” script controls the order you want. It is very easy to setup and even easier to add more switches.

The asset includes:
- Raycast system that detects the switches and buttons independently, so an infinite number of levers can be added as well as multiple versions of switches in the same scene.
- Add as many switches as are needed
- Limit the number of levers you can pull based on the value given
- Buttons, indicator lights and animated switches included
- Scripts which can be easily modified
- Sound effects for the system

See more detailed information, video and setup guides here: https://speedtutoruk.gitbook.io/lever-system/

FAQ

Q). How do I Import the asset?

A). Go to the Unity asset store and visit your “Download manager”. Download the asset if not already downloaded and click “Import”, import all required features of the asset for your use. It should have appeared in your project under “Lever Puzzle System”.

Q). Should I import “Project Settings” when choosing to import this asset?

A). It is always recommended NOT to import project settings unless important for your project, you can alternatively create an entirely new project and import this asset with the project setting, then transfer the package folder over to your current project. To save any issues! Always back up your project before importing any assets – If you have any issues, do let me know!

Q). Is there an example of this asset working?

A). Yes, you can open the “**Lever System Demo**” to see the lever system in action, or use this scene as your initial base of your project.

Q). How can I add more levers to the set I already have?

A). Just duplicate a lever you already have, make sure it has the appropriate tags. Edit the “LeverController” > “LeverOrder” to incorporate 6 numbers, and set pulls to 6. Refer to the “Extending the lever system” for more information.

Q). How can I add more than one lever set to my scene?

A). Check the “Extending the lever system”.

Q). Do you want to organise the lever elements into a empty GameObject?

A). When you create an empty GameObject, please make sure that the X, Y, Z coordinates of the empty GameObject are all 0, 0, 0 before dragging the parts into the GameObject.

Manual Setup – Initial #1

NOTE: See more detailed information, video and setup guides here:
https://speedtutoruk.gitbook.io/lever-system/

NOTE: You can get a quick start using the "Lever_Entire_DemoScene" prefab object and drag that straight into your scene. This will everyone to be already setup, you can delete the art assets as required but checkout the demo scene if you're confused!

Player:

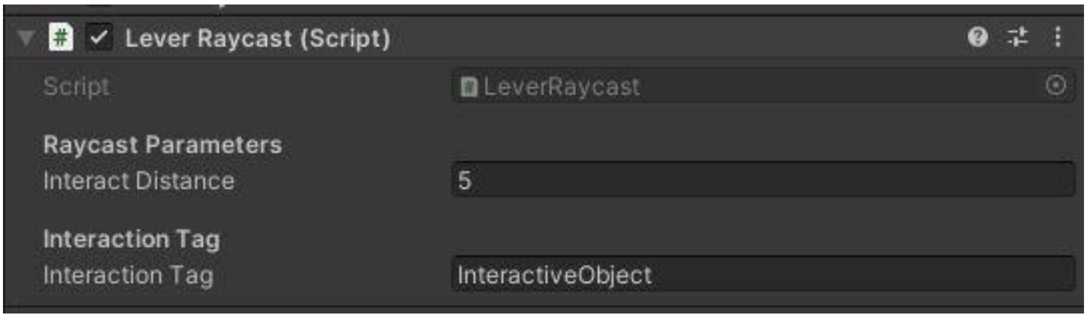
- Layer of: "Ignore Raycast"

Any Interactable Object:

- Tag "InteractiveObject" or as per the raycast script on the main camera, it will have the tag you need!

1). Add an "FPSController" to your scene. (Unless you're using the demo which will already have one added, but the package will be required to be imported before use). Please navigate to the scripts folder and add the "LeverInteractor" script to your "MainCamera". (If not already.

- You can set the "Interact Distance"
- The tag used to detect object in-front



Lever Raycast (Script)

Script: LeverRaycast

Raycast Parameters

Interact Distance: 5

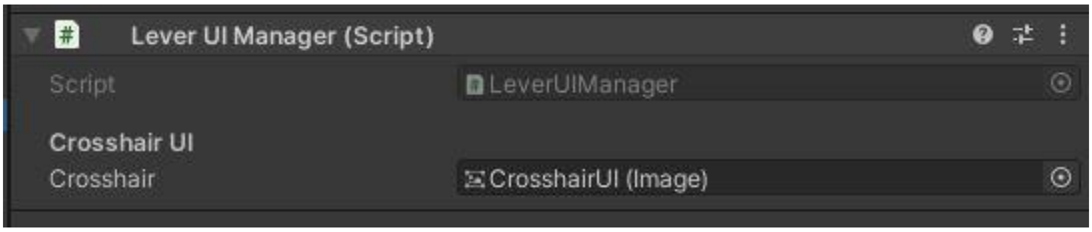
Interaction Tag

Interaction Tag: InteractiveObject

2). Now drag the managers from the "Prefabs" folder into the hierarchy. You will need to add a "UIManager" / "AudioManager" / "InputManager"

NOTE: Only drag this prefab into the hierarchy, not the scene.

3). UIManager: Add the crosshair image from the "CrosshairCanvas" components to the "LeverUIManager" crosshair slot in the inspector.



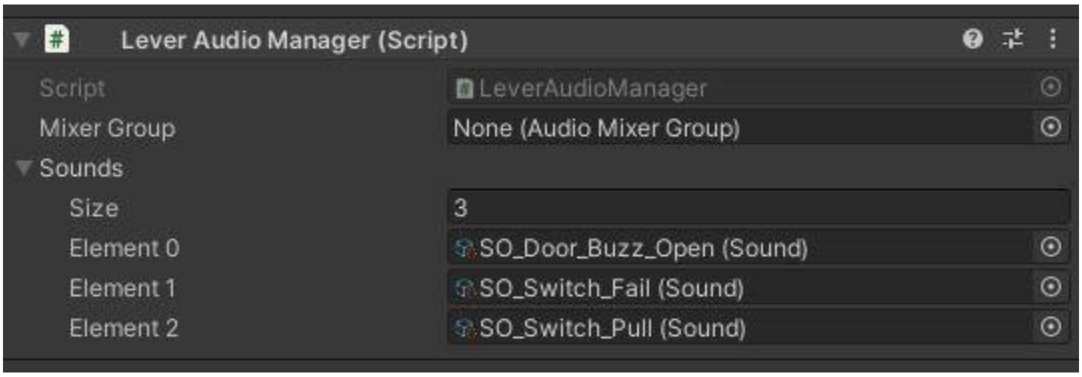
Lever UI Manager (Script)

Script: LeverUIManager

Crosshair UI

Crosshair: CrosshairUI (Image)

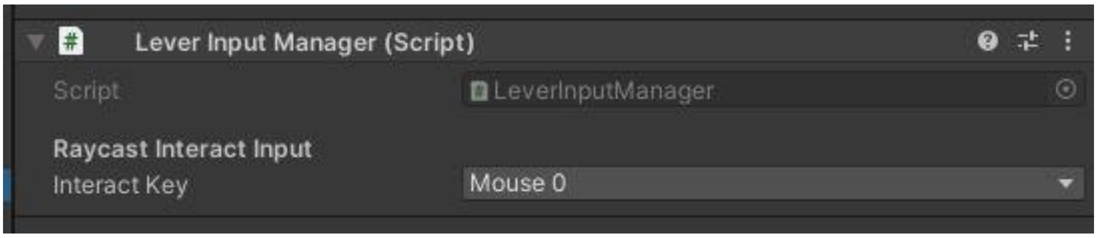
4). AudioManager: Add the ScriptableObjects sound effects, found in the ScriptableObject folder to the size array of sounds. This should already be setup for you:



Lever Audio Manager (Script)

Script	LeverAudioManager
Mixer Group	None (Audio Mixer Group)
Sounds	
Size	3
Element 0	SO_Door_Buzz_Open (Sound)
Element 1	SO_Switch_Fail (Sound)
Element 2	SO_Switch_Pull (Sound)

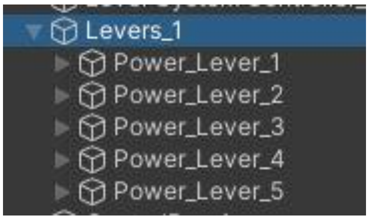
5). InputManager: Select the Input you wish to use for the system



Lever Input Manager (Script)

Script	LeverInputManager
Raycast Interact Input	
Interact Key	Mouse 0

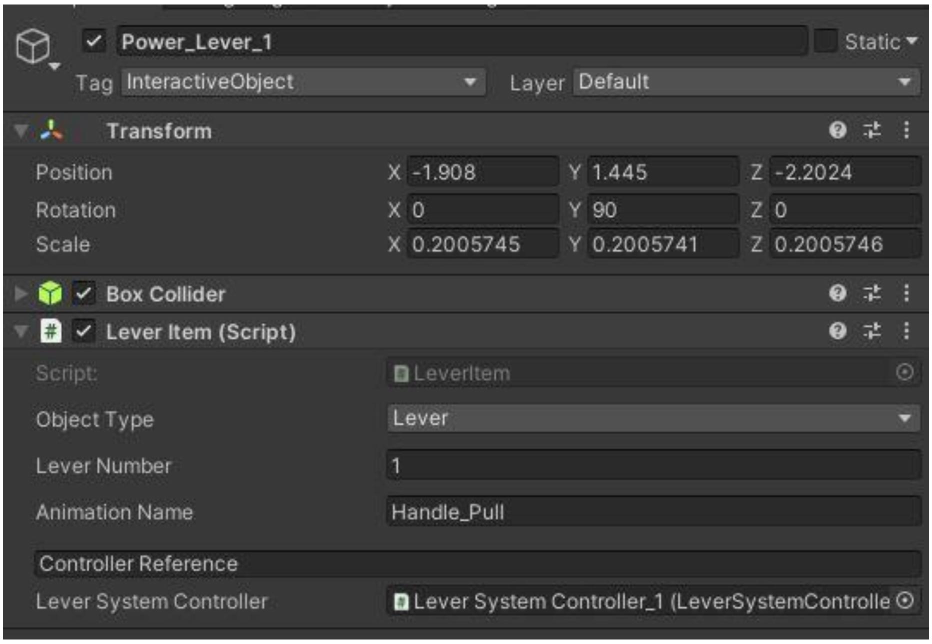
6). Add a set of levers to your scene, the prefab will have a folder called "Levers_1" - which will include 5 levers. You can duplicate these objects if you wish to have more or less for ONE SINGLE controller:



Levers_1
Power_Lever_1
Power_Lever_2
Power_Lever_3
Power_Lever_4
Power_Lever_5

7). You can see the individual lever setup below:

- Set the tag to "**InteractiveObject**" (The tag you specified in the Raycast script on the Main Camera)
- A box collider
- Add a "**LeverItem**" script to this object (If not already)
    - Set the **ObjectType** to: "**Lever**"
    - Set the number you want this lever to have, based on your order
    - Set the animation name for this item, the default is "**Handle_Pull**"
    - Add the "**Controller**" reference, which we will add in the later stage of this setup



Power_Lever_1 — Static
Tag: InteractiveObject — Layer: Default

Transform

	X	Y	Z
Position	-1.908	1.445	-2.2024
Rotation	0	90	0
Scale	0.2005745	0.2005741	0.2005746

Box Collider

Lever Item (Script)

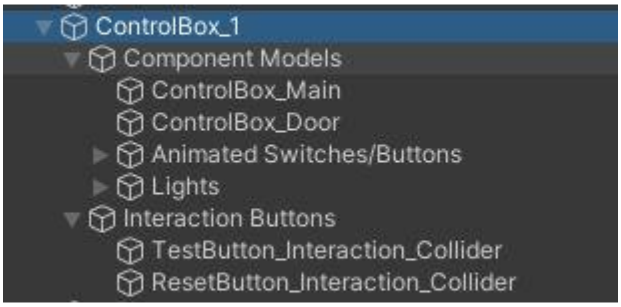
Script:	LeverItem
Object Type	Lever
Lever Number	1
Animation Name	Handle_Pull

Controller Reference

Lever System Controller	Lever System Controller_1 (LeverSystemControlle

8). Add a "**ControlBox**" prefab to your scene, which will have:

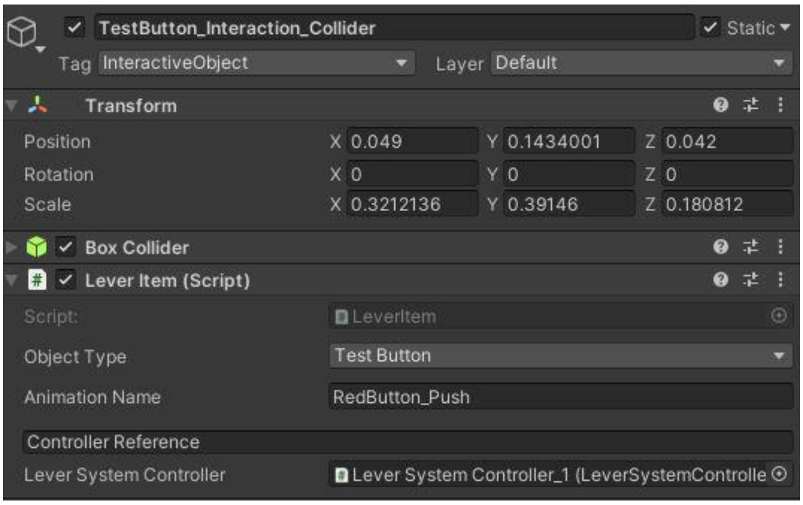
- Component Models (These are added to the System controller, scroll down further)
- Interaction Buttons (Setup below)



- ControlBox_1
    - Component Models
        - ControlBox_Main
        - ControlBox_Door
        - Animated Switches/Buttons
        - Lights
    - Interaction Buttons
        - TestButton_Interaction_Collider
        - ResetButton_Interaction_Collider

8b). Interactive Button Setup:

- Tag: "**InteractiveObject**"
- Box Collider
- "**LeverItem**" script
  - Test Button or Reset Button depending on the button type
- Animation Name: "**RedButton_Push**"
- Add the controlof the system below:
-



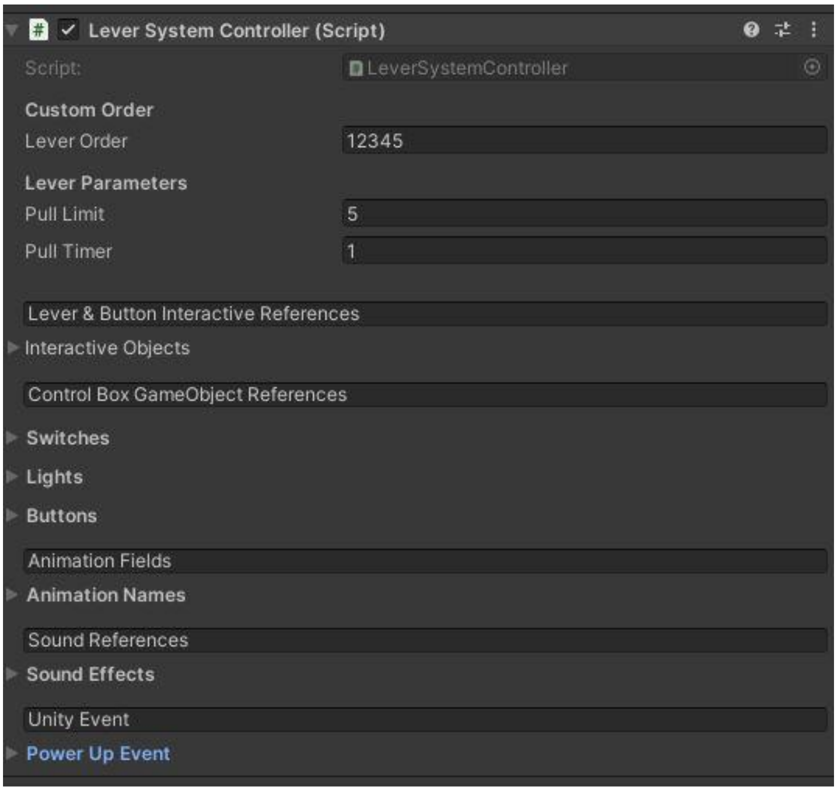
TestButton_Interaction_Collider ☑ Static
Tag InteractiveObject Layer Default

Transform
Position X 0.049 Y 0.1434001 Z 0.042
Rotation X 0 Y 0 Z 0
Scale X 0.3212136 Y 0.39146 Z 0.180812

☑ Box Collider

☑ Lever Item (Script)
Script: LeverItem
Object Type: Test Button
Animation Name: RedButton_Push
Controller Reference
Lever System Controller: Lever System Controller_1 (LeverSystemControlle

9). Add a "**Lever System Controller**" to your scene, this will have a "**Lever System Controller**" script attached, see the setup below:



☑ Lever System Controller (Script)
Script: LeverSystemController

Custom Order
Lever Order: 12345

Lever Parameters
Pull Limit: 5
Pull Timer: 1

Lever & Button Interactive References
Interactive Objects

Control Box GameObject References
Switches
Lights
Buttons

Animation Fields
Animation Names

Sound References
Sound Effects

Unity Event
Power Up Event

Lever Order: The order the levers need to be pulled to complete the sequence, match this will the numbers you set on the lever objects

Pull limit: What is the limit of the pull order, make sure it's the number of levers you have

Pull Timer: The timer you must wait before another lever can be pulled, can stop unnecessary double interactions

Interactive Objects: Add all the the levers and interactive buttons to this array. By default there will be 7 slots

Switches: Add all of the switches inside the "Control box"

Lights: Add all the lights inside the "Control box"

Buttons: Add all the buttons inside the "Control Box"

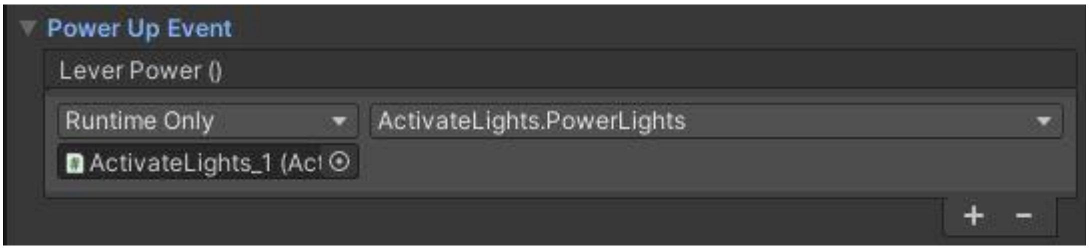
Animation Names: These are the animation names for the objects, defaults are usually ok

Sound Effects: Add the ScriptableObject audio effects for the system

Power Up Event: This is the event that will happen when the levers are pulled in the correct order and tested

10). Power up event

This is the power up event, I have created a basic script called "ActivateLights", which is a public method to change emission and activate some GameObjects. You can add whatever you want here, whether that's opening a door, turning on a generator or more!



Power Up Event

Lever Power ()

Runtime Only

ActivateLights.PowerLights

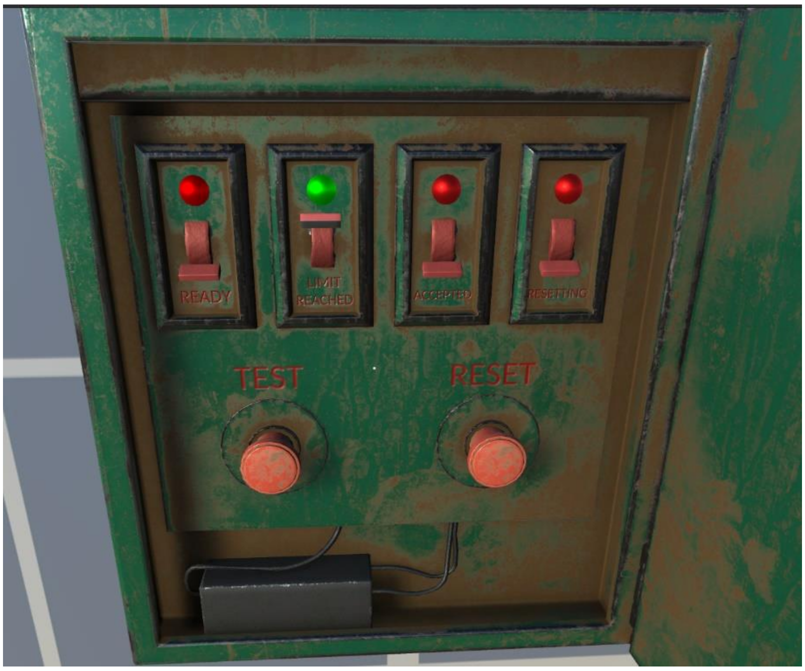
ActivateLights_1 (Act

Using the Lever system:

- Pull the levers in the order you wish
- When the limit is reached, the system must be reset you will see the button light. No interactions can happen until "**Tested**" or "**Reset**".
- Use the "**Test**" button to test the current order you have set, resetting will happen if the order is incorrect
- Use the "**Reset**" button to reset your current order

The indicators at the top show what's current active:

- **Ready:** You can start pulling levers
- **Limit Reached:** When the pull limit is reached
- **Accepted:** When the order has been "Tested" and correct
- **Resetting:** The system is resetting the order, ready to use again

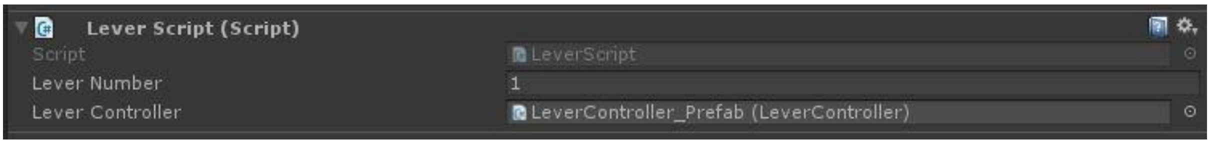


NEED Additional HELP? UPDATED INFORMATION OR VIDEOS? Check out the online documentation here: https://speedtutoruk.gitbook.io/lever-system/

Extending the Lever System

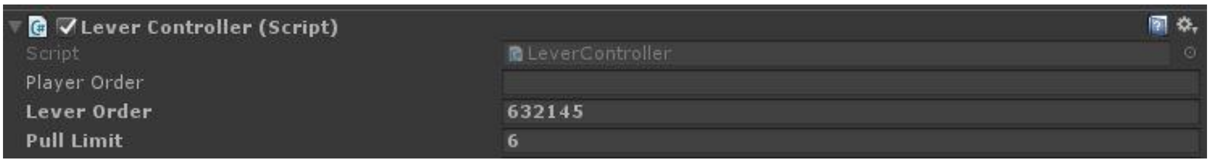
Adding more than 5 levers to a setup?

Just add another lever by duplicating one you already have. Give the lever another number. If you have added your 6th lever, make this number 6. Do this in the “**LeverScript**” by changing the “LeverNumber” value in the inspector.



Lever Script (Script)
Script: LeverScript
Lever Number: 1
Lever Controller: LeverController_Prefab (LeverController)

Increment the pulls and order number on the “**LeverController**” GameObject, dependant on how many levers you have. For Example; If you have 6 levers, make sure the “**LeverOrder**” has 6 numbers: 1,2,3,4,5,6 and the pull limit also has “6” in it.



☑ Lever Controller (Script)
Script: LeverController
Player Order:
Lever Order: 632145
Pull Limit: 6

How to add another lever set to the same scene?

Duplicate the entire “**LeverSystemController**” and “**SwitchBox_Model**” parent GameObject in your scene and reposition wherever you want. Check all scripts are linked and it should be working fine.

Contact

Thank you very much for downloading! Please be sure to leave a **5*** rating and **review** if you liked the package! Support me on

Patreon: https://www.patreon.com/SpeedTutor
Make a small donation: http://www.paypal.me/speedtutor

If you have any problems about the setup, usage, customisation or have any suggestions for future updates of this kit, please feel free to contact me.

Email: speedtutoruk@gmail.com
Discord: https://discord.gg/FguwaJHfSY
YouTube: https://www.youtube.com/user/speedtutor
Website: http://www.speed-tutor.com

Online Documentation

See more detailed information, video and setup guides here: https://speedtutoruk.gitbook.io/lever-system/

Patch Notes

Version 1.2 - July 2023

- Duplicated all scripts to fix GUID issues when using my other assets
- Made the **Sound** script generic
- Renamed **LeverRaycast** to **LeverInteractor**
  - Removed redundant singleton code from this script
  - Edited variables names for more consistency
- Refactored **LeverSystemController** script for further optimisation
  - Added new methods for easier understanding of repeated usage
  - Added new tooltips
- Added **Edit Custom Inspector** script to then bottom of custom inspector scripts
- Edited Custom inspectors for easier readability by splitting into multiple methods

Version 1.1 – July 2022

- Created new scene organisation.
- Refactored all code to make it shorter, easier to read and optimised
- Created new ScriptableObjects for audio effects
- Created Additional Packages folder for organisation
- Renamed all Textures, models and animations for ease of use
- Created new UIManager
  - Moved "Crosshair UI" from raycast and into the UIManager
- Edited Raycast script to 2022 version
- Changed "LeverItemController" to "LeverItem" for consistency
  - Morphed "LeverItem" with old lever controller and refined the script
  - New custom inspector for this script
- LeverSystemController script changes:
  - Added SO to this script for audio control
  - Created a custom inspector
- Added a new "Power up" or "On" event, so you can obviously see some lights switch on when the levers are pulled in the correct order

Version 1.0 – January 2021

- Created a new “LeverController” and “LeverItemController” to add to each interactive object to make it easier
- Refactored the “LeverSystemController” to incorporate refinements to the code and workings
- Added new managers for: Sound, Input and Raycast
- Re-written the raycast script to be more optimised and refined
- Added a UnityEvent for final interaction, so users can easily create different outcomes
- Added: New PBR lever
- Added: New PBR Control box with interaction
- Added: New animations for levers and control box