Implementation Log

Challenge 1: Designing a Modular Event System

Problem Description

I wanted each lamp to dynamically enable or disable certain features (such as color changing or flashing) via buttons. This requires using UnityEvent to build an event system that can manage multiple listeners at runtime.

Why It's a Challenge

UnityEvent only allows static listener configuration in the Inspector. What I needed was the ability to add and remove different functional modules at runtime, not just change values.

My Solution

- Each LampController has an OnToggleLight UnityEvent.
- The default listener controls the basic on/off logic.
- Clicking the "Color" button dynamically adds or removes ChangeColor() as a listener.
- Listeners are managed using .AddListener() and .RemoveListener().

Sketch idea

Pseudocodec

```
Setup():
    OnToggleLight.AddListener(HandleToggle)
AddColorListener():
    OnToggleLight.AddListener(ChangeColor)
RemoveColorListener():
    OnToggleLight.RemoveListener(ChangeColor)
```

Challenge 2: Coroutine Management for Lamp Flashing

Problem Description

I needed to make the lamp toggle on and off every 0.5 seconds for a total of ? seconds after pressing a button. This behavior should use a coroutine and be safely stoppable if the lamp is destroyed.

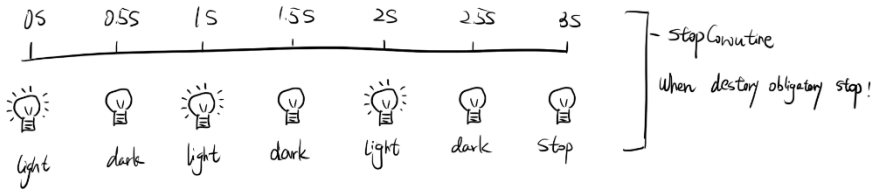
Why It's a Challenge

- Must prevent multiple coroutines from running at the same time
- Must allow clean termination from outside the coroutine
- Coroutine should not conflict with other lamp behaviors

My Solution

- I used a Coroutine flashRoutine variable to store the running coroutine.
- I checked if a coroutine was already running before starting a new one.
- When needed, I stopped it with StopCoroutine() and cleared the reference.

Sketch idea



Pseudocode

```
StartFlashing():
    If flashRoutine == null:
        flashRoutine = StartCoroutine(FlashCoroutine)
FlashCoroutine():
    timer = 0
        while timer < ? :
            ToggleLight() and timer += 0.5
StopFlashing():
    If flashRoutine == !null
        StopCoroutine(flashRoutine)
        flashRoutine = null
```

Challenge 3: Full Cleanup After Lamp Destruction

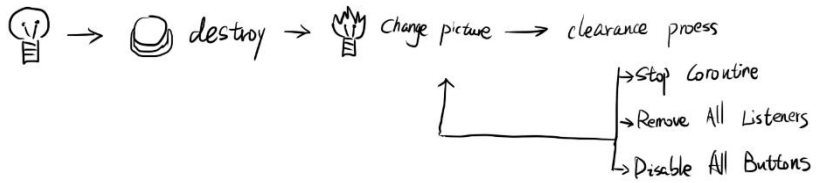
Problem Description

After clicking the "Destroy" button, the lamp should become broken, stop all activity, and ignore all further interaction. This means removing all listeners, stopping coroutines, disabling UI, and changing the lamp sprite.

Why It's a Challenge

- All modules might be active and must be fully shut down
- If listeners aren't removed, they'll still respond
- If coroutines aren't stopped, flashing might continue

Sketch idea



My Solution

I changed the sprite to a broken one, called RemoveAllListeners() to clear all UnityEvent listeners, used StopFlashing() to stop the coroutine, and disabled all UI buttons.

Pseudocode

```
BreakLamp():

    Setting images as broken

    Call StopFlashing()

    OnToggleLight.RemoveAllListeners()

    Disable all buttons
```