Implementation Log

Challenge 1: How to design a modular system allowing each lamp to combine functions freely

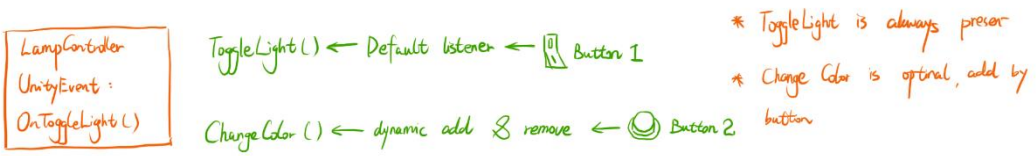
Problem

I want each generated lamp to have different functions like color changing or flashing. These functions should not activate by default, but only when the player clicks the corresponding button. The functions need to be independent, flexible to combine, and should not interfere with each other.

Solution

I plan to use an Unityevent system to decouple the logic between the lamp and its function modules. Each module can register as a listener, and when a specific button is clicked, the event will trigger its associated behavior.

Sketch idea



Pseudocode

```
WHEN button is pressed:
    trigger a modular event
EACH function module:
    listens to that event
    performs its behavior if triggered
Each lamp:
    holds its own set of function modules
```

Challenge 2: How to make the lamp's light flash briefly and then stop automatically

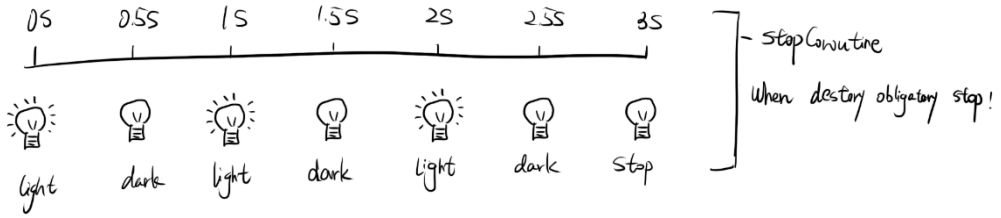
Problem

I want the light on the lamp to flash briefly when a button is clicked, then stop automatically. The flashing should not affect other lamps, should not be triggered repeatedly, and the lamp should return to its normal state afterward.

Solution

I plan to use a non-blocking time-based control method, such as a Coroutine, to handle the on-off behavior of the light. During flashing, the light will alternate between on and off. Once the time is up, it will be forced off and the flashing state will be cleared.

Sketch idea



Pseudocode

```
WHEN flash is triggered:
    IF lamp is on AND not already flashing:
        START time-based loop:
            toggle light on/off
            wait briefly each loop
        END
        reset light to off
        clear flashing state
    IF END
```

Challenge 3: How to design a “destroy lamp” mechanism that completely stops all behaviors

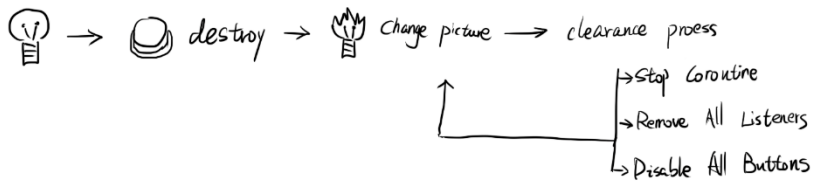
Problem

I want the lamp to stop responding to any buttons after being destroyed, and all ongoing behaviors (like flashing) should immediately stop. The UI and light should also be hidden or turned off, visually indicating that the lamp is broken.

Solution

I plan to use a “destroyed” flag variable to prevent repeated destruction. When triggered, I will disable components related to light and functional control, stop any time-based processes (like Coroutines), and hide all UI buttons to prevent further interaction.

Sketch idea



Pseudocode

```
WHEN destroy is triggered:
    IF lamp is not already destroyed:
        mark as destroyed
        stop all time-based processes
        disable light control modules
        hide all buttons and visuals
        show broken appearance
    IF END
```