1. Design and Event Flow of the System

This project implements an **event-driven system in Unity** to manage interactions such as inventory collection, environmental triggers, and UI updates. The system consists of five main components:

1. **EventManager (Central Hub)**
    - Manages event definitions (`onItemCollected`, `onDoorOpened`).
    - Stores inventory data.
    - Invokes events when relevant actions occur.
2. **ItemPickup (Event Invoker)**
    - Detects when the player picks up an item.
    - Adds the item to the inventory and triggers `onItemCollected`.
3. **DoorController (Event Listener)**
    - Listens for `onDoorOpened`.
    - Smoothly rotates and opens the door.
4. **UIManager (Event Listener)**
    - Listens for `onItemCollected`.
    - Updates the score and inventory UI.
5. **TriggerEvent (Event Invoker)**
    - Detects when the player enters a trigger zone.
    - Triggers `onDoorOpened` to open the door.

Event Flow:

- The player **picks up an item** → `ItemPickup` invokes `onItemCollected` → `UIManager` updates the UI.

- The player **enters a trigger zone** → `TriggerEvent` invokes `onDoorOpened` → `DoorController` opens the door.

2. How Events Improve Modularity and Gameplay Responsiveness

Modularity:

- **Loose Coupling:** Components interact via events rather than direct references. This makes it easy to add, remove, or modify features without affecting the entire system.

- **Reusable Code:** The `EventManager` acts as a central hub, making it easier to extend functionality without modifying multiple scripts.

Gameplay Responsiveness:

- **Instant Feedback:** When an event occurs (e.g., picking up an item), the UI updates immediately, improving player experience.

- **Efficient Communication:** Events allow game components to react instantly without constant checks or complex dependencies.

3. Challenges and Solutions

Challenge 1: EventManager Instance Not Found

- **Problem:** Some components tried to access `EventManager.Instance` before it was initialized.

- **Solution:** Ensured `EventManager` is a **Singleton** and initialized before any other scripts reference it.

Challenge 2: UI Not Updating Correctly

- **Problem:** The inventory UI sometimes didn't reflect newly added items.
- **Solution:** Added a **listener in `UIManager`** to update UI immediately after `onItemCollected` is invoked.

Challenge 3: Door Animation Delays

- **Problem:** The door rotation was abrupt instead of smooth.
- **Solution:** Used a **Coroutine with `Quaternion.Lerp`** to create a gradual rotation effect.

Conclusion

By implementing an **event-driven architecture**, the system achieves **modularity, efficiency, and responsiveness**. The use of UnityEvents and custom C# events allows **seamless interaction between components**, making the game system **more flexible and maintainable**