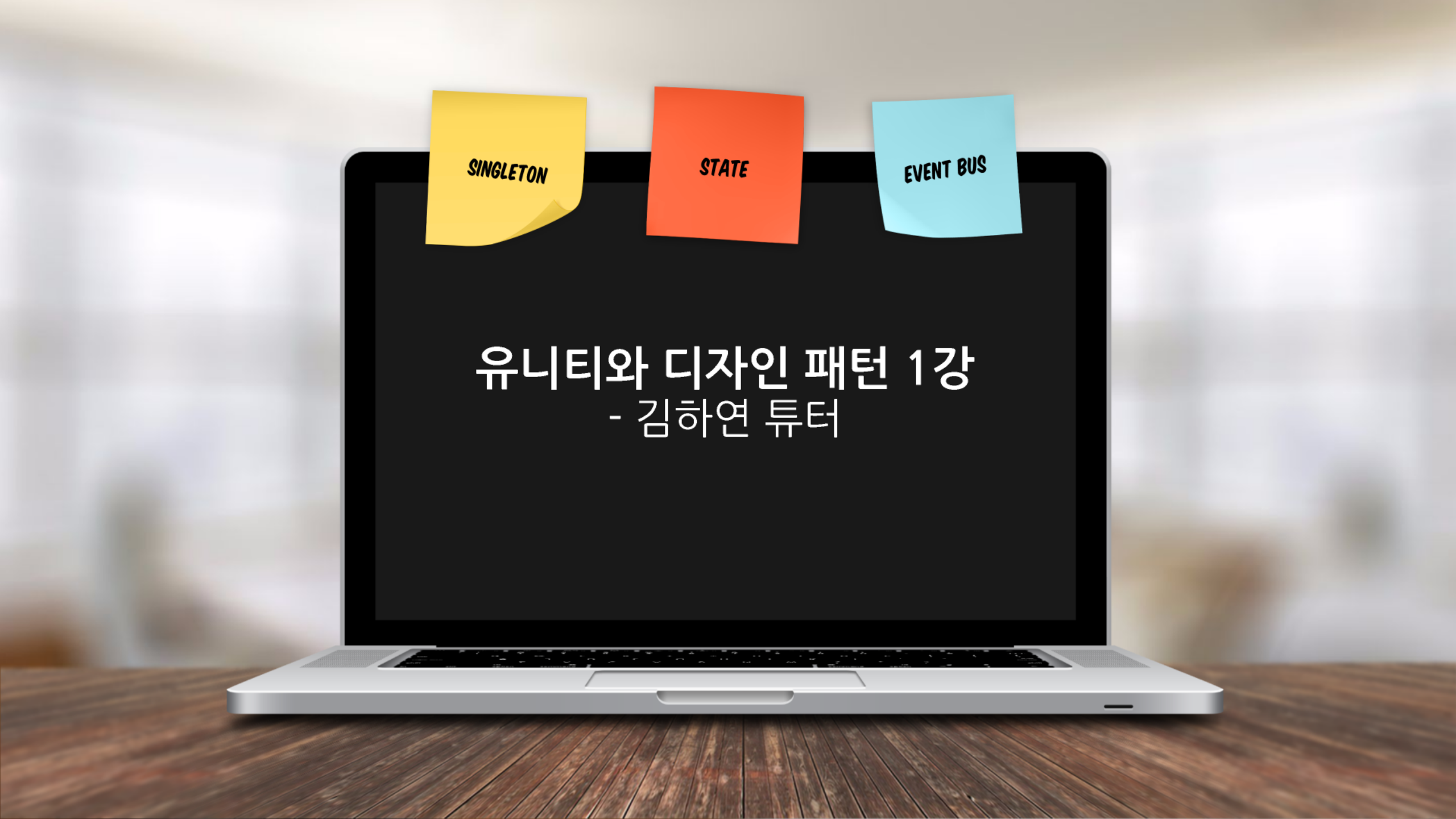

SINGLETON
STATE
EVENT BUS
유니티와 디자인 패턴 1강
- 김하연 튜터

싱글턴 패턴으로 게임 매니저 구현

- 다른 분야에서는 논란이 있는 패턴이지만, 유니티에서는 정말 많이 쓰이는 패턴
- 개별 매니저 클래스에서 핵심 시스템을 래핑하고 관리함
- 게임 내부를 캡슐화. 직관적인 인터페이스
- 핵심 구성 요소 간의 결합을 강력하게 하여 유닛 테스트를 어렵게 만듬

싱글턴 패턴

- 싱글턴 패턴은 유일성을 보장하는 것
- 런타임 동안 메모리에 오로지 하나의 인스턴스만 존재함
- 전역적으로 접근할 수 있는 시스템
- 싱글턴 인스턴스는 메모리 안에 한번 생성, 같은 유형의 인스턴스가 있으면 삭제함

장점
- 전역적 접근 가능
- 공유 자원에 동시 접근을 제한하거나 사용할 수 있음

단점
- 유닛 테스트가 어려움
- 잘못된 프로그래밍 습관을 유발함

게임 매니저 구현

```
using UnityEngine;

0 references
public class Singleton <T>: MonoBehaviour where T: Component
{
    private static T_instance;

    1 reference
    public static T instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = FindObjectOfType<T>();

                if (_instance == null)
                {
                    GameObject obj = new GameObject();
                    obj.name = typeof(T).Name;
                    _instance = obj.AddComponent<T>();
                }
            }

            return instance;
        }
    }
}
```

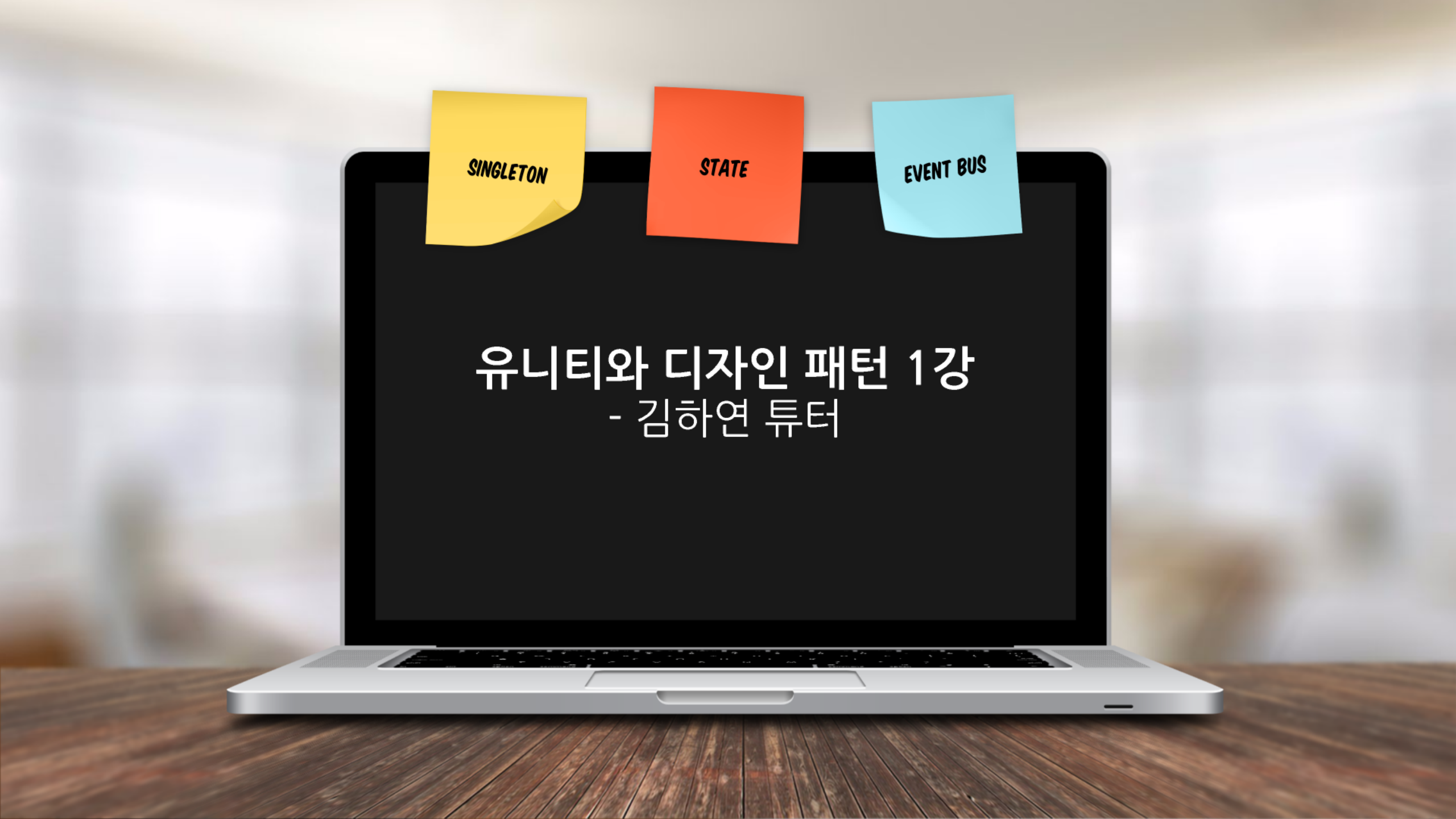
```
0 references
public void Awake()
{
    if (_instance == null)
    {
        _instance = this as T;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
    }
}
```

게임 매니저 구현

```
1 reference
public class GameManager : Singleton<GameManager>
{
    0 references
    void Start()
    {

    }
}
```

- 빈 유니티 씬을 만듬
- GameObject 추가하고 GameManager 클래스를 추가함.
- 원하는 만큼의 유니티 씬을 File -> build settings 목록에 추가함.

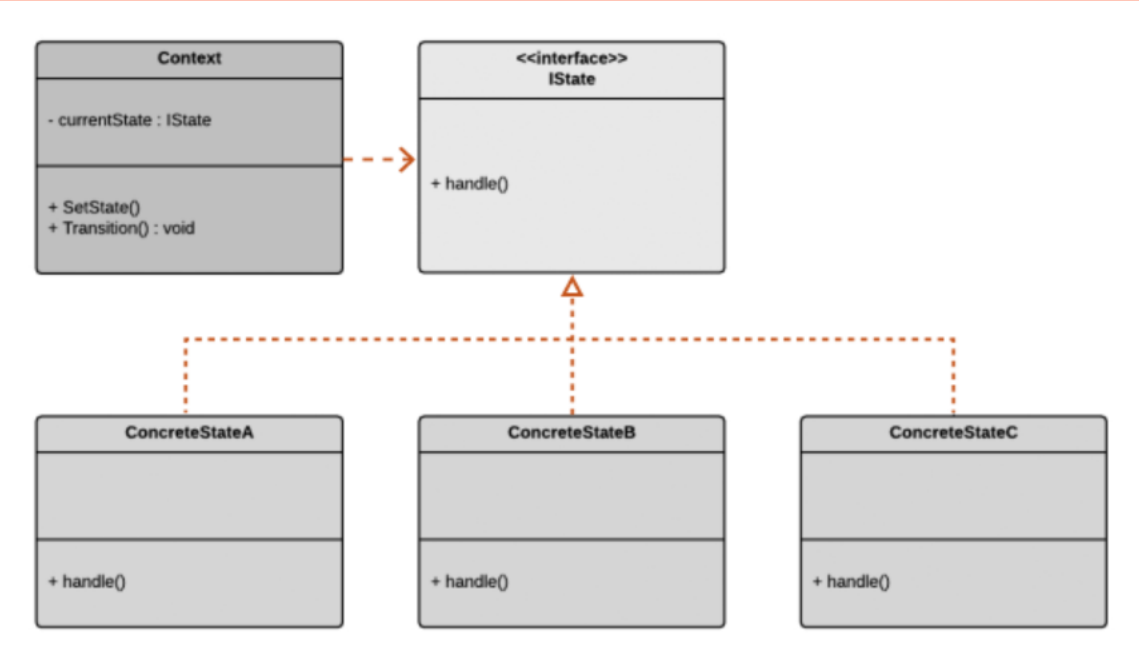

SINGLETON
STATE
EVENT BUS
유니티와 디자인 패턴 1강
- 김하연 튜터

상태 패턴으로 캐릭터 상태 관리하기

- 캐릭터는 상태가 끊임없이 전환됨.
- 상태를 관리하는게 상태 패턴

상태 패턴 개요

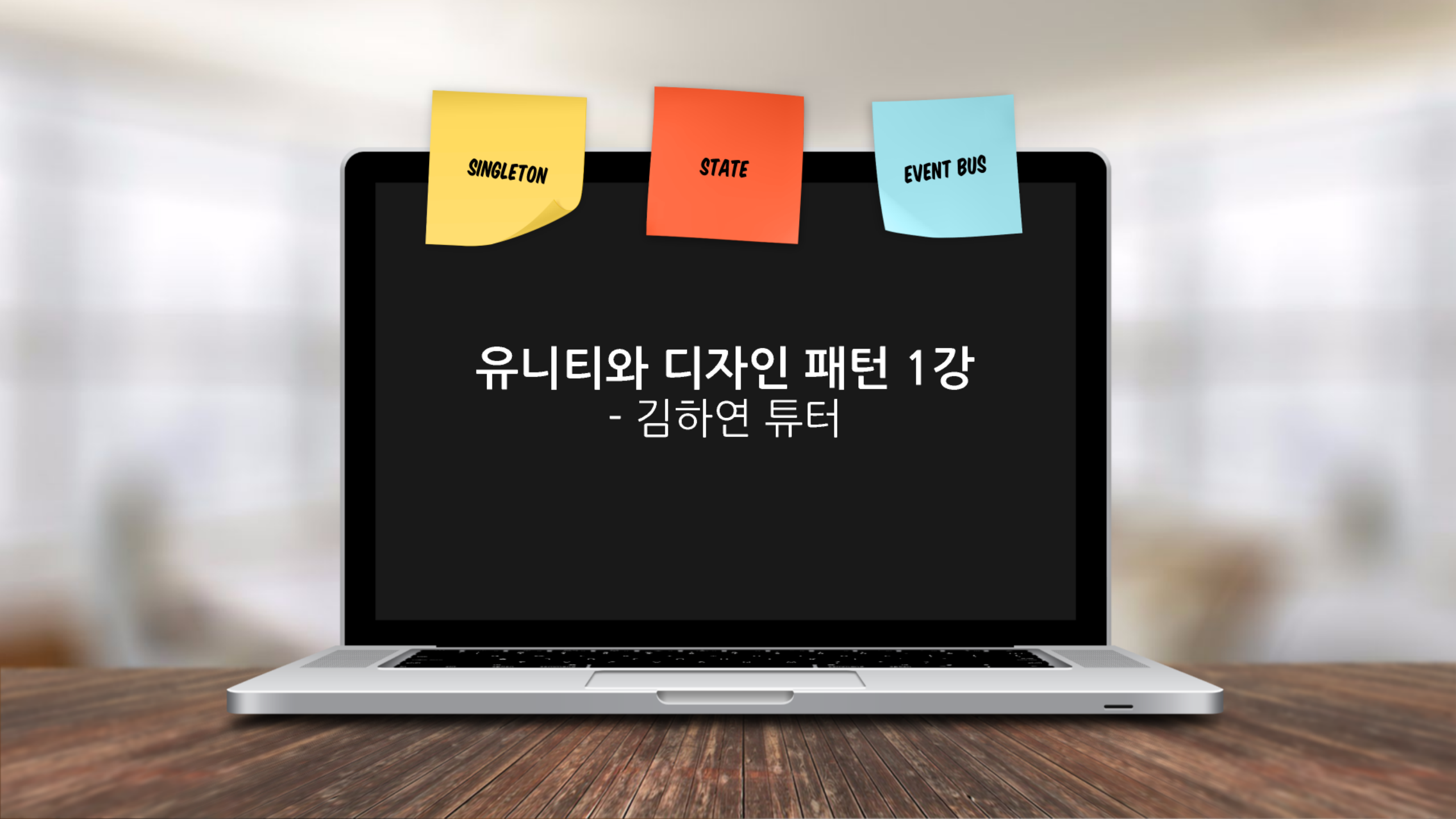
- 객체가 내부 상태를 기반으로 움직이는 시스템
- Context 클래스는 객체의 내부 상태를 변경하도록 요청하는 클래스
- IState 인터페이스는 구체적인 상태 클래스로 연결할 수 있도록 설정함
- Context 오브젝트가 IState 인터페이스를 구현한 클래스들을 호출함



상태 패턴 개요

장점
- 캡슐화가 잘 되어있음
- 긴 조건문이나 방대한 클래스를 구현하는 것을 막음

단점
- 만약 상태가 전환되는 사이에서 발생하는건 따로 구현을 해야함.

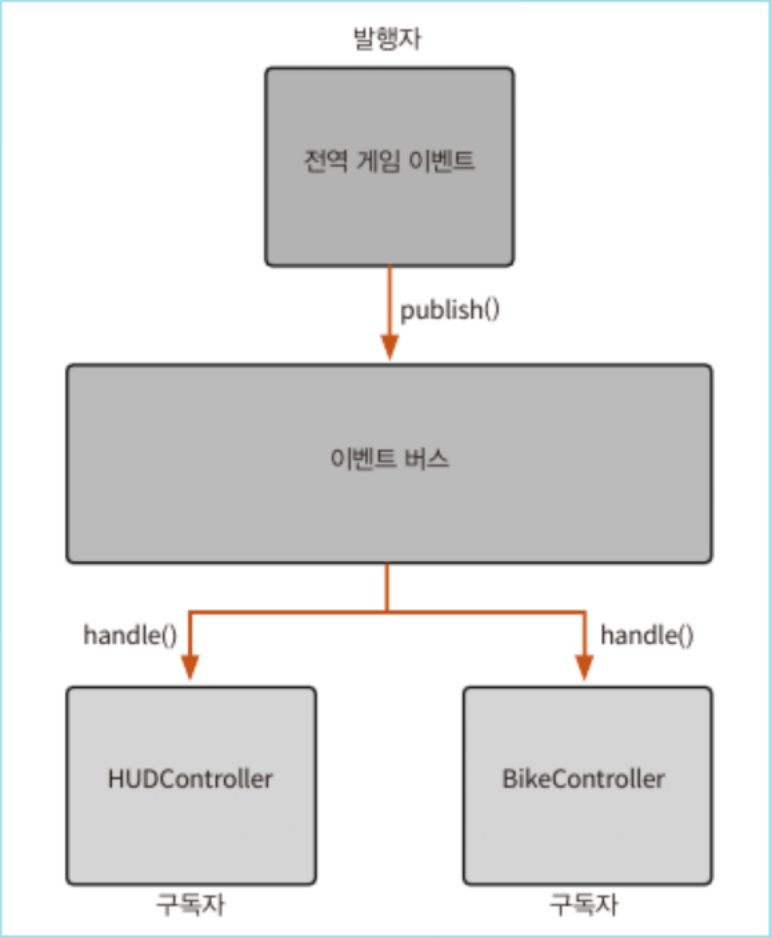

SINGLETON
STATE
EVENT BUS
유니티와 디자인 패턴 1강
- 김하연 튜터

이벤트 버스 패턴으로 게임 이벤트 관리

- 이벤트 버스는 전역 이벤트를 관리하는 중앙 허브 개념
- 게임에서 월드 이벤트 발생시 해당 캐릭터들에게 이벤트를 발송하는 식
- 구현하기 의외로 매우 간단해서 많이 쓰임

이벤트 버스 패턴 개요

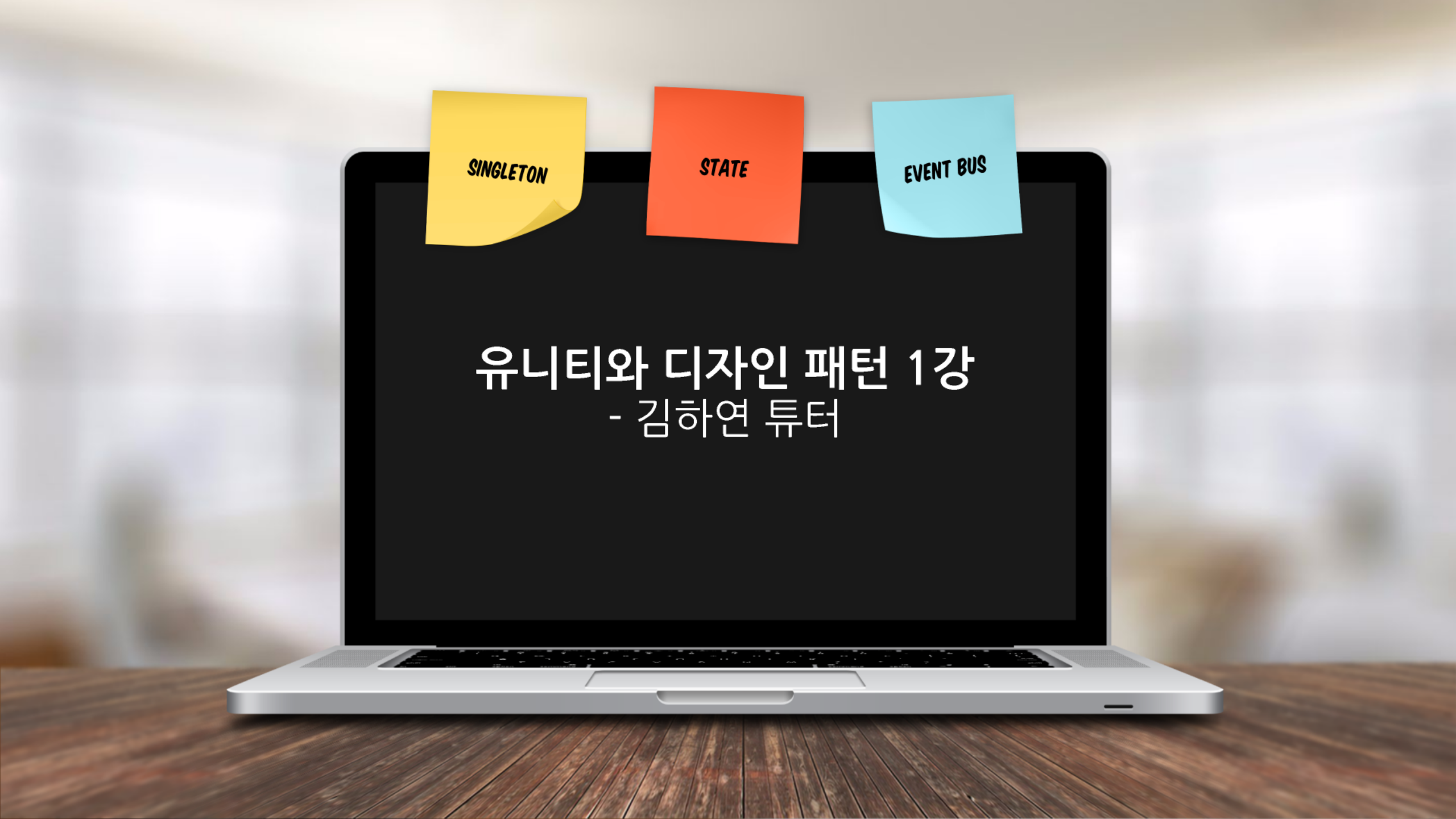
- 어떤 객체에서 이벤트가 발생하면 다른 구독자가 신호를 받는 시스템
- 발행/구독 패턴
- 발행자랑 구독자는 서로 인식하지 못함. 중간에 이벤트 버스가 이를 관리하기 때문임.



이벤트 버스 패턴 개요

장점
- 오브젝트를 직접 참조하지 않고, 이벤트를 통신할 수 있음
- 이벤트 버스는 구독 시스템을 쉽게 구현하게 만듬
- 게임 프로토타입 만들때 많이 쓰임. 쉽고 빠르기 때문.

단점
- 약간의 성능 비용
- 이벤트 버스가 static 전역 변수라서, 전역변수의 단점을 모두 가지게 됨


SINGLETON
STATE
EVENT BUS
유니티와 디자인 패턴 1강
- 김하연 튜터