코루틴이란?
in Unity

박세웅

목차

1. 코루틴이란?
2. 사용상황 / 이유
3. 사용방법
4. 코루틴에 대한 이야기할 요소

코루틴(Coroutine)이란?

- Unity에서 코루틴은 실행을 일시 정지하고 제어를 Unity에 반환하지만 중단한 부분에서 다음 프레임을 계속할 수 있는 메서드입니다.

- 유니티 메뉴얼

사용상황 / 이유

```
public class test1 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        Fade();
    }

    //오브젝트의 알파(불투명도) 값을 보이지 않을 때까지 점차 줄이는 작업
    void Fade()
    {
        Color c = renderer.material.color;
        for (float alpha = 1f; alpha >= 0; alpha -= 0.1f)
        {
            c.a = alpha;
            renderer.material.color = c;
        }
    }
}
```

일반적으로 메서드는 호출된 메서드에 제어와 선택적 반환 값을 반환

→ 메서드 내의 모든 행동은 단일 프레임 업데이트에서 발생

Update문으로 구현 가능

하지만 장기적으로 구현과 유지보수 관점에서 문제 발생

사용방법

```
private void Start()
{

    StartCoroutine(Myco());

}
IEnumerator Myco()
{

    yield return new WaitForSeconds(1f);
    print("Myco");
}
```

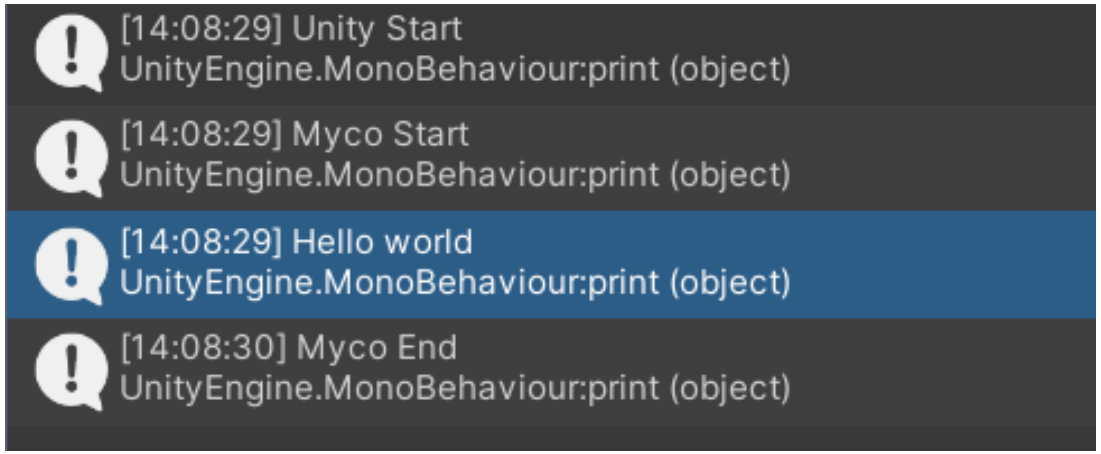
1. 코루틴은 StartCoroutine()으로 부르고 StartCoroutine()은 매개변수로 IEumerator로 받음

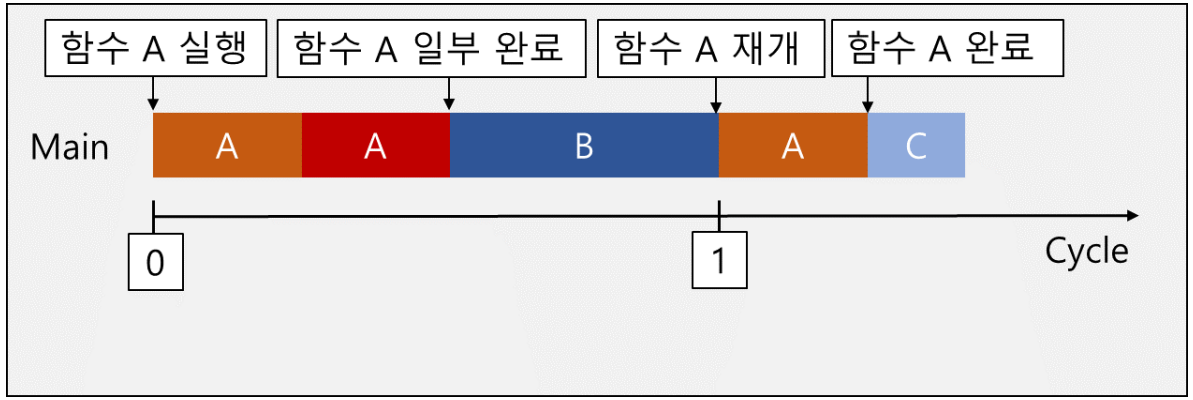
2. 코루틴은 IEumerator를 반환하며 내부에는 최소 1개 이상의 yield return문이 있어야함

여기서 yield의 의미는 '일시적으로 cpu 권한을 다른 함수에 위임한다'라는 의미

사용방법

```
private void Start()
{
    print("Unity Start");
    StartCoroutine(Myco());
    print("Hello world");
}
IEnumerator Myco()
{
    print("Myco Start");
    yield return new WaitForSeconds(1f);
    print("Myco End");
}
```



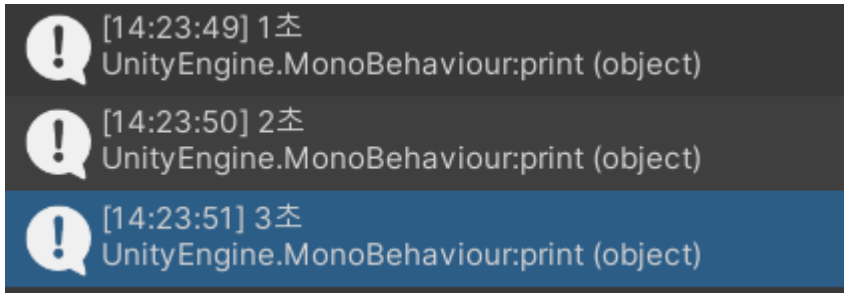


코루틴(Coroutine)

```
private void Update()
{
    if(Input.GetKeyDown(KeyCode.Keypad1)){
      StartCoroutine(Myco());
    }
}
IEnumerator Myco()
{
    yield return new WaitForSeconds(1f);
    print("1초");
    yield return new WaitForSeconds(1f);
    print("2초");
    yield return new WaitForSeconds(1f);
    print("3초");
}
```

코루틴은 실행을 일시 정지하고 제어를 Unity에 반환하지만

중단한 부분에서 다음 프레임을 계속할 수 있는 메서드입니다.



사용방법 - 명령어

1. yield return null
2. yield return new WaitForSeconds(float)
3. yield return new WaitForSecondsRealtime(float)
4. yield return new WaitForFixedUpdate()
5. yield return new WaitForEndOfFrame()
6. yield return new WaitUntil(조건문)
7. yield return new WaitWhile(조건문)
8. yield return StartCoroutine(string).
9. yield break 등등

이야기 할 요소

1. 코루틴은 비동기처럼 **여러 개의 쓰레드가 동시에 동작하는 것과 같이 보인다**

→ 하지만 동기 방식으로 작동

2. 유니티는 한번에 하나의 작업만 처리하는 단일 스레드 방식을 사용

코루틴은 동기 방식으로 작동하기 때문에 정말 무거운 작업을 코루틴으로 처리할 시 성능저하를 야기합니다 (렌더링 프레임 수치에 직접적인 영향력)

→ 잘 이용되진 않지만 필요한 경우 비동기 방식 로직으로 작성 (async/await)

3. 스크립트가 들어있는 오브젝트가 setActive(false)등으로 비활성화되면 작동하지 않는다는 점

이야기 할 요소

4.코루틴은 C# 컴파일러에 의해 자동 생성된 클래스 인스턴스 형태로 힙 메모리에 저장되고 가비지를 많이 생성함 → 캐싱을 통해 최적화 해야함

5. StartCoroutine에서 IEnumerator를 매개변수를 받는 이유 (기술 면접에서 나옴)