1.定时功能实现

在 Unity 开发中，延时是我们经常要实现的功能。

有点经验的开发者都能自己搞一个出来。

最常见的，一个延时功能的实现如下:

```
using UnityEngine;

public class CommonDelayExample : MonoBehaviour
{
    private float mStartTime;

    void Start()
    {
        mStartTime = Time.time;
    }

    void Update()
    {
        if (Time.time - mStartTime > 5)
        {
            DoSomething();
            // 避免再次执行
            mStartTime = float.MaxValue;
        }
    }

    void DoSomething()
    {
        Debug.Log("DoSomething");
    }
}
```

代码量比较多，后来接触到了 Coroutine（协程），更好的实现如下:

```
using System;
using System.Collections;
using UnityEngine;

public class CoroutineDelayExample : MonoBehaviour
{
    void Start()
    {
        StartCoroutine(Timer(5, DoSomething));
    }

    IEnumerator Timer(float seconds, Action callback)
    {
        yield return new WaitForSeconds(seconds);
        callback();
    }

    void DoSomething()
    {
        Debug.Log("DoSomething");
    }
}
```

使用 UniRx，则很简单，代码如下:

```
Observable.Timer(TimeSpan.FromSeconds(5)).Subscribe(_ => { /* do something
*/ });
```

当然以上代码是没有和 MonoBehaviour 进行生命周期绑定的。

要绑定很简单。

```
Observable.Timer(TimeSpan.FromSeconds(5))
        .Subscribe(_ => { /* do something */ })
```

```
        .AddTo(this);
```

只要加上一个 AddTo(this) 就可以了。

这样，当 this(MonoBehaviour) Destroy 的时候，这个延时逻辑也会销毁掉，从而避免造成空指针异常。

三行代码，大约 20 秒时间，就搞定了一个实现起来比较麻烦的逻辑。

今天的内容就这些。