

author: Igor Chebotar
contact: wolk.fo@gmail.com

COROUTINE EXTENSIONS

BY SIMPLE MAN

Часто во время написания кода возникает потребность отсрочить выполнение операции на какое-то время или до момента, пока некое условие не будет выполнено. Наши расширения призваны упростить эти операции до вызова одной команды (Delay, WaitUntil, RepeatUntil...). Плагин не требует наследования вашего класса от некоего специального и каких-либо других манипуляций. Вы спокойно можете загрузить coroutine extension в существующий проект и без пользоваться без ограничений уже сейчас!

КАК ИСПОЛЬЗОВАТЬ DELAY?

Чтобы отсрочить выполнение операции на определенное время воспользуемся командой Delay.

Задача: отсрочить выполнение метода, который выводит сообщение, на 2 секунды после старта.

Для примера создадим новый скрипт Test, в нем объявим метод типа void и назовем его OnDone (название роли не играет). Этот метод выводит сообщение в консоль.

Далее в методе Start вызовем команду Delay, где первый параметр отвечает за время задержки в секундах, а второй - делегат метода, который должен быть вызван по истечению времени таймера.

Пример:



```
2 3
Сообщение Unity | Ссылка: 0
private void Start()
{
    //Set delay in two seconds
    this.Delay(2, OnDone);
}

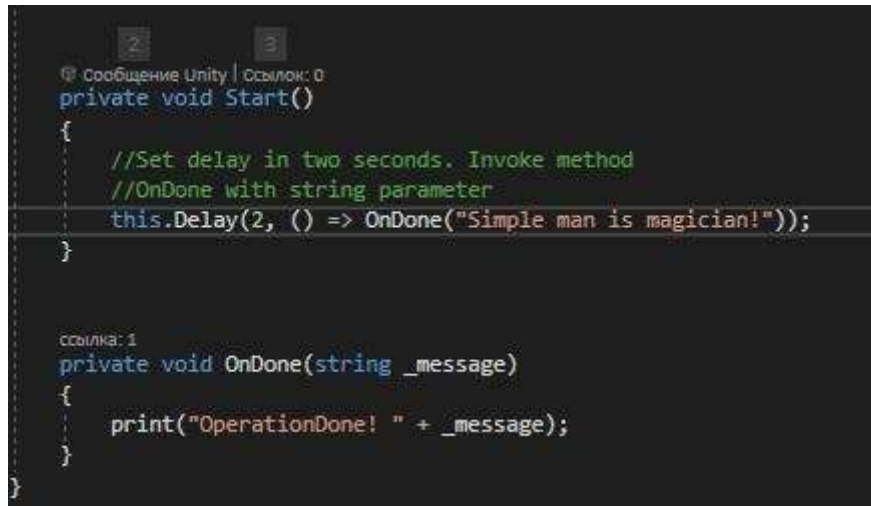
ссылка: 1
private void OnDone()
{
    print("OperationDone!");
}
```

Итог: при нажатии на “Play” через 2 секунды метод OnDone выведет сообщение в консоль.

КАК ИСПОЛЬЗОВАТЬ DELAY С ПЕРЕДАЧЕЙ ПАРАМЕТРА?

Для передачи аргумента в метод OnDone(string _message) воспользуемся лямбда-выражением.

Пример:



```
2 3
Сообщение Unity | Ссылки: 0
private void Start()
{
    //Set delay in two seconds. Invoke method
    //OnDone with string parameter
    this.Delay(2, () => OnDone("Simple man is magician!"));
}

ссылка: 1
private void OnDone(string _message)
{
    print("OperationDone! " + _message);
}
}
```

КАК ИСПОЛЬЗОВАТЬ WAIT UNTIL?

Иногда нам нужно отсрочить выполнение операции не на определенное время, а до того момента, пока не выполнится некое условие. В этом нам поможет Wait Until.

Задача: вызвать метод OnDone сразу после того, как поле isPressed примет значение true.

Для этого в методе Start вызовем команду WaitUntil и через лямбда выражение передадим IsPressed в качестве условия.

Пример:

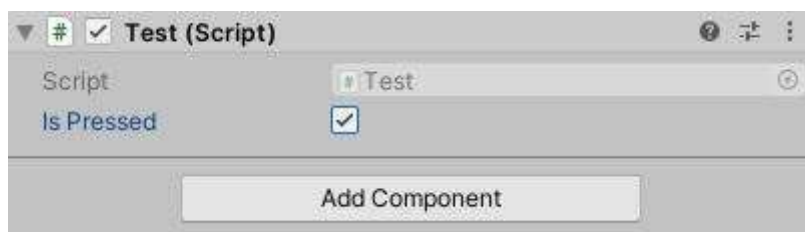
```
public bool isPressed = false;

Сообщение Unity | Ссылка: 0
private void Start()
{
    //Wait until is pressed will be true
    //then invoke OnDone
    this.WaitUntil(() => isPressed == false, OnDone);
}

Ссылка: 1
private void OnDone()
{
    print("OperationDone!");
}
```

Проверка:

По нажатию на IsPressed в консоли появляется сообщение



КАК ИСПОЛЬЗОВАТЬ WAIT UNTIL С ЗАДЕРЖКОЙ?

Встречаются также ситуации, которые требуют ресурсозатратных операций, которые могут выступать в качестве условий. Например если мы хотим взаимодействовать с компонентом объекта, но по какой-то причине этот компонент пока отсутствует на объекте.

Известно, что GetComponent - дорогая операция и мы не имеем права проверять наличие компонента каждый кадр, пока ждем его появления. Однако можем проверять это условие каждые полсекунды, без особых потерь производительности.

В таких случаях нам поможет WaitUntil с задержкой.

Работает он так же как и обычный, но в качестве последнего аргумента при вызове метода можно выбрать время задержки в секундах.

Пример:

```
public bool isPressed = false;

Сообщение Unity | Ссылка: 0
private void Start()
{
    //Wait until is pressed will be true
    //then invoke OnDone
    this.WaitUntil(() => isPressed == false, OnDone, 0.5f);
}

Ссылка: 1
private void OnDone()
{
    print("OperationDone!");
}
```

КАК ИСПОЛЬЗОВАТЬ REPEAT UNTIL?

Часто возникают ситуации, когда нужно повторять какое либо действие до определенного момента или же вовсе бесконечно. Примером может служить использование Raycast'a. RepeatUntil позволяет это реализовать путем написания одной строчки.

Задача: каждый кадр вызывать метод Repeat пока IsPressed не примет значение true, затем вызвать метод OnDone.

Пример:

```
© Сообщение Unity | Ссылки: 0
private void Start()
{
    this.RepeatUntil(() => isPressed == false, Repeat, OnDone);
}

ссылка: 1
private void Repeat()
{
    print("Repeat");
}

ссылка: 1
private void OnDone()
{
    print("OperationDone!");
}
```

КАК ИСПОЛЬЗОВАТЬ REPEAT FOREVER?

Очень похожим образом вызывается команда RepeatForever. Это упрощенная версия RepeatUntil.

*RepeatUntil и RepeatForever также как и WaitUntil могут иметь задержку выполнения.

КАК ОСТАНОВИТЬ ВЫПОЛНЕНИЕ ОПЕРАЦИИ?

Все указанные здесь методы возвращают Coroutine. Остановить выполнение операции можно путем кэширования полученной корутины и ее последующей остановки стандартным Unity методом "StopCoroutine".