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

# COROUTINE EXTENSIONS

BY SIMPLE MAN

Often, when writing code, there is a need to delay the operation for some time or until a certain condition is met. Our extensions are designed to simplify these operations to a single command (Delay, Wait Until, Repeat Until). The plugin does not require inheritance of your class from a special class or any other manipulations. You can easily upload coroutine extensions to an existing project and use it without restrictions right now!

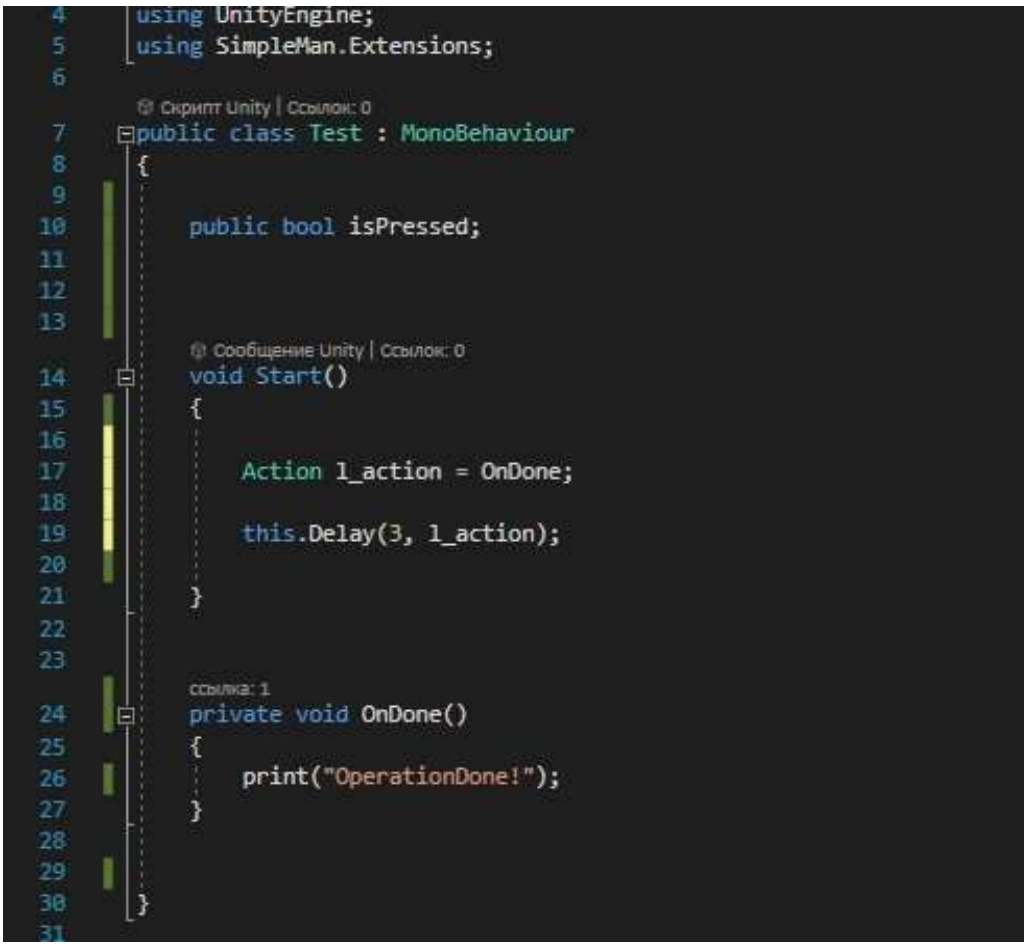
**HOW TO USE DELAY?**

To delay the operation for a certain time, use the Delay command.

For example, let's create a new Test script, declare a method of the void type in it, and call it OnDone (the name doesn't matter). This method outputs a message to the console.

Next, in the Start method, we call the Delay command, where the first parameter is responsible for the delay time in seconds, and the second parameter is the delegate of the method that should be called after the timer expires.

**Detailed example:**



**Fast:**

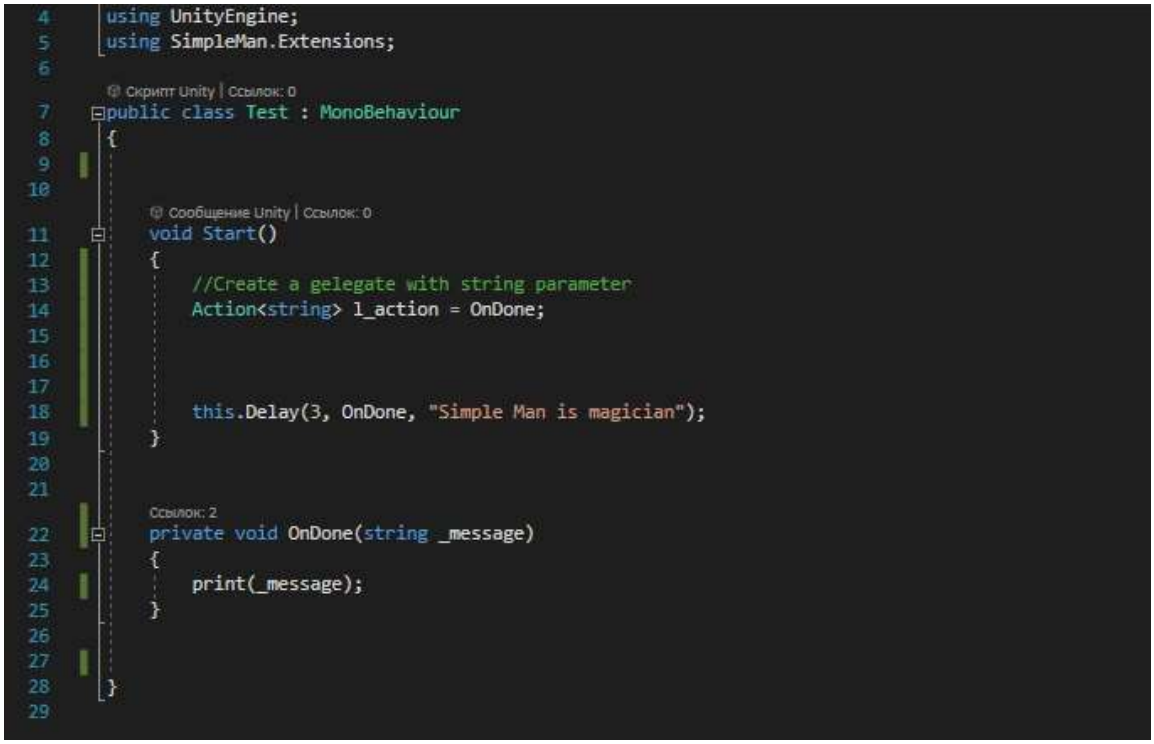
```
using UnityEngine;
using SimpleMan.Extensions;

Скрипт Unity | Ссылок: 0
public class Test : MonoBehaviour
{

    public bool isPressed;



    Сообщение Unity | Ссылок: 0
    void Start()
    {


        this.Delay(3, OnDone);

    }



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


}
```

Bottom line: when you click "Play" after 3 seconds, the OnDone method outputs a message to the console.

**HOW TO USE DELAY WITH PARAMETER?**

To pass an argument to the OnDone(string_message) method, before calling the Delay command, you must create an appropriate delegate instance that corresponds to the data type that the final method accepts, in our case, string.

Next, as usual, we call the Delay method and pass it our delegate as the second argument and the string as the third.

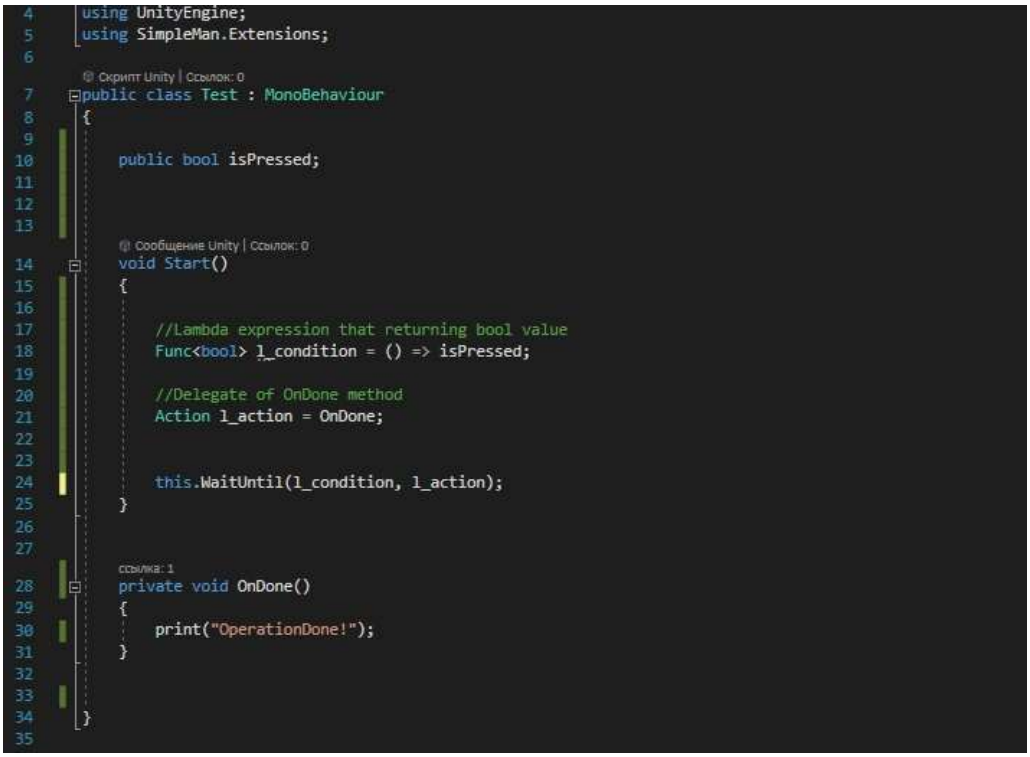


```
using UnityEngine;
using SimpleMan.Extensions;

Скрипт Unity | Ссылок: 0
public class Test : MonoBehaviour
{


    Сообщение Unity | Ссылок: 0
    void Start()
    {
        //Create a gelegate with string parameter
        Action<string> l_action = OnDone;



        this.Delay(3, OnDone, "Simple Man is magician");
    }


    Ссылок: 2
    private void OnDone(string _message)
    {
        print(_message);
    }


}
```

## HOW TO USE WAIT UNTIL?

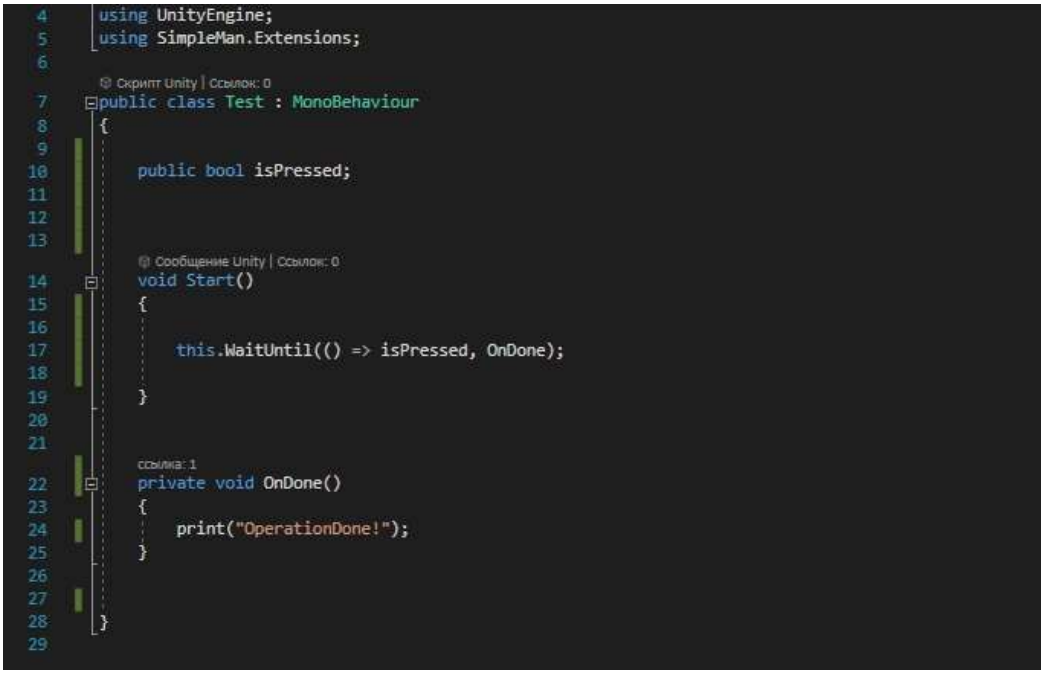
Sometimes we need to delay the operation not for a certain time, but until a certain condition is met. Wait Until will help us with this.

## Detailed example:



```
using UnityEngine;
using SimpleMan.Extensions;

Скрипт Unity | Ссылок: 0
public class Test : MonoBehaviour
{

    public bool isPressed;



    Сообщение Unity | Ссылок: 0
    void Start()
    {

        //Lambda expression that returning bool value
        Func<bool> l_condition = () => isPressed;

        //Delegate of OnDone method
        Action l_action = OnDone;


        this.WaitUntil(l_condition, l_action);
    }


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


}

```

## Fast:



```
using UnityEngine;
using SimpleMan.Extensions;

Скрипт Unity | Ссылок: 0
public class Test : MonoBehaviour
{

    public bool isPressed;



    Сообщение Unity | Ссылок: 0
    void Start()
    {

        this.WaitUntil(() => isPressed, OnDone);

    }


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


}

```

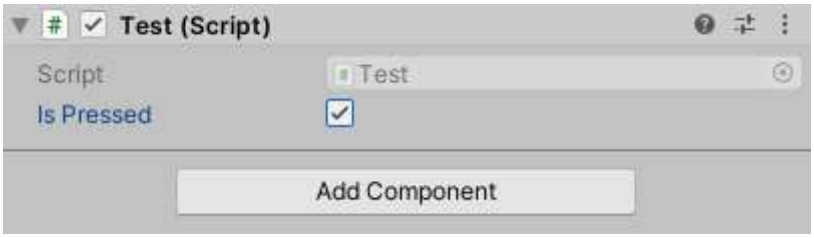
In this example, the condition is the public field IsPressed.
Task: call the OnDone method when isPressed is set to true.

To do this, create an instance of the Func<bool>delegate in the Start method and using a lambda expression, we will pass IsPressed as a condition.

Then, similar to Delay, we call the Wait Until command and pass the condition delegate as the first argument, and the ondone method delegate as the second.

You can also use overloading and pass a parameter for the final method, similar to the previous example.

Check:
When you click on IsPressed a message appears in the console



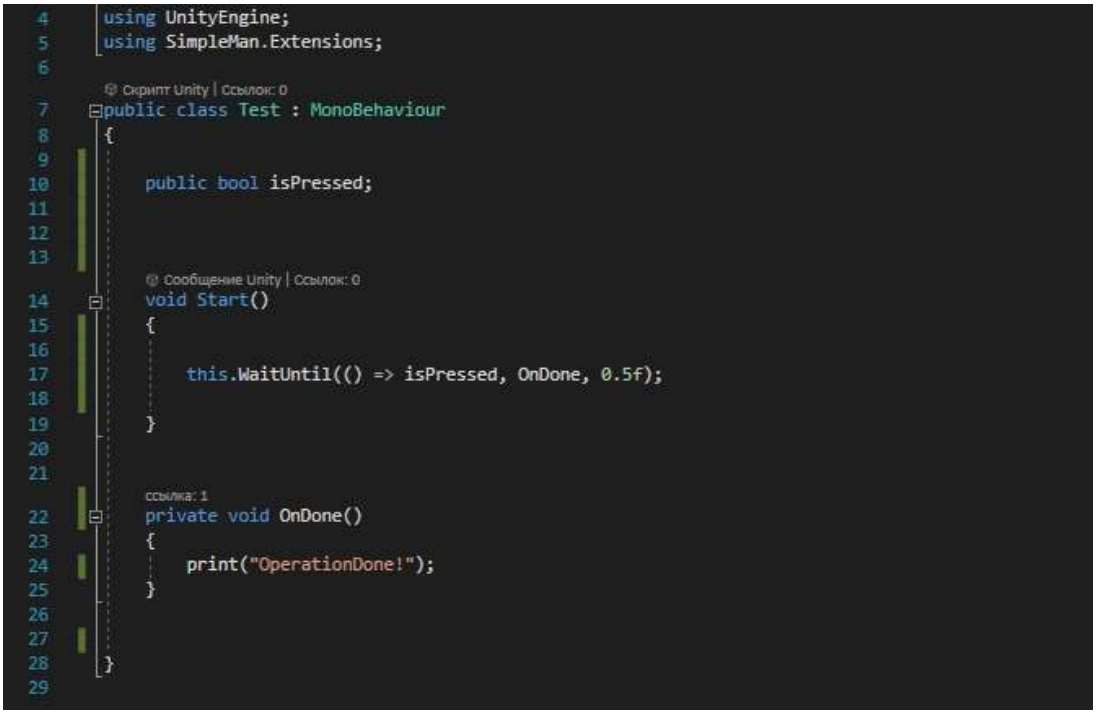

**HOW TO USE WAIT UNTIL WITH DELAYING?**

There are also situations that require resource-intensive operations, which can act as conditions. For example, if we want to interact with an object component, but for some reason this component is not yet present on the object. It is known that GetComponent is an expensive operation and we do not have the right to check the presence of a component every frame while waiting for it to appear. However, we can check this condition every half a second, without much loss of performance.

In such cases, Wait Until will help us with the delay.

It works the same as normal, but you can choose the delay time in seconds as the last argument when calling the method.

Example:



**HOW TO USE REPEAT UNTIL?**

Often there are situations when you need to repeat an action up to a certain point or even indefinitely. An example is the use of Raycast. It is known that using Raycast in each frame will seriously reduce the performance of the game, and often it is not necessary to do this, because the same effect can be achieved using Raycast every 0.2 seconds or less.

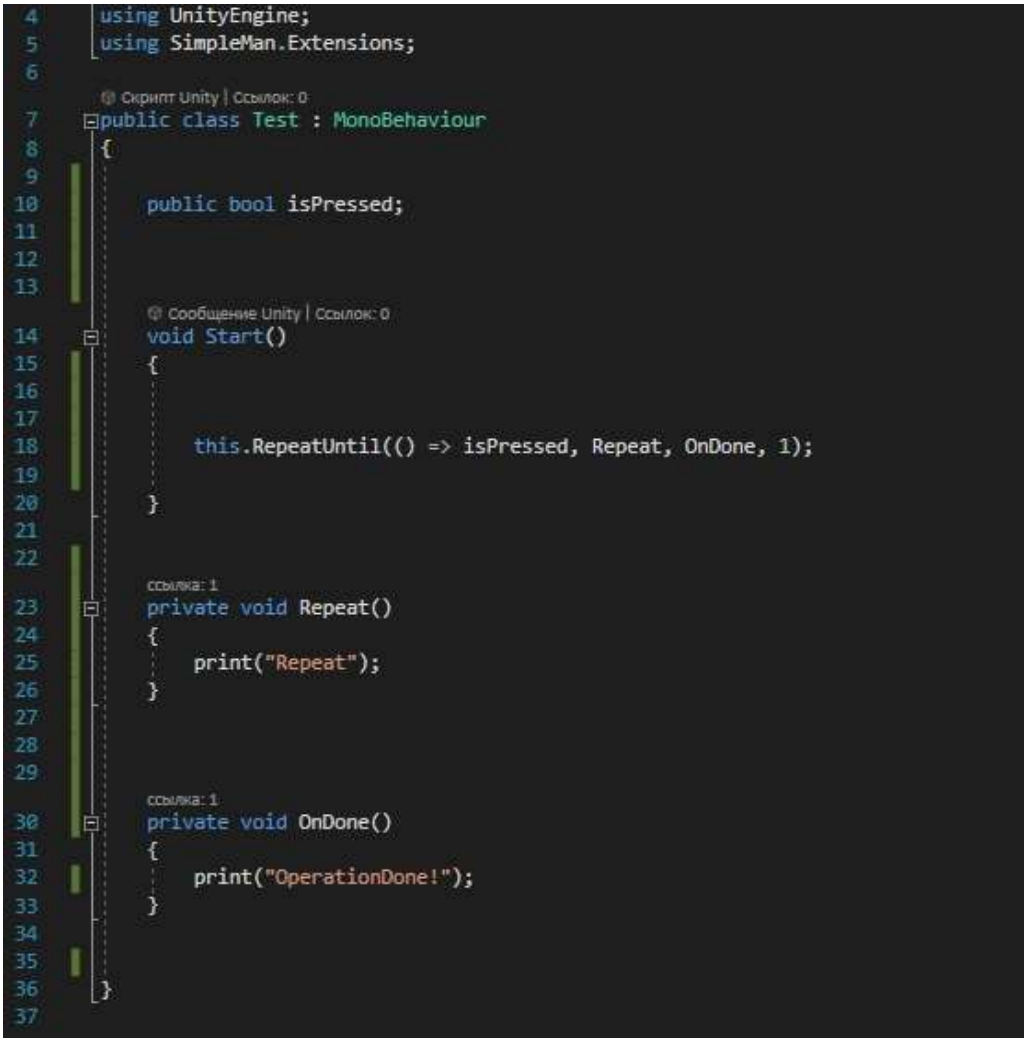
Repeat Until allows you to implement this approach by writing a single line.

Task: call the Repeat method every second until IsPressed is set to true, then call the OnDone method.

**Detailed example:**

```
using UnityEngine;
using SimpleMan.Extensions;

Скрипт Unity | Ссылок: 0
public class Test : MonoBehaviour
{

    public bool isPressed;



    Сообщение Unity | Ссылок: 0
    void Start()
    {

        Func<bool> l_condition = () => isPressed;
        Action l_repeatAction = Repeat;
        Action l_doneAction = OnDone;
        float l_delay = 1;


        this.RepeatUntil(l_condition, l_repeatAction, l_doneAction, l_delay);

    }


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



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


}
```

## Fast:



```
using UnityEngine;
using SimpleMan.Extensions;

Скрипт Unity | Ссылок: 0
public class Test : MonoBehaviour
{

    public bool isPressed;



    Сообщение Unity | Ссылок: 0
    void Start()
    {


        this.RepeatUntil(() => isPressed, Repeat, OnDone, 1);

    }


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



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


}

```

**Task:** Endlessly call the Repeat method every second

```
using UnityEngine;
using SimpleMan.Extensions;

Скрипт Unity | Ссылок: 0
public class Test : MonoBehaviour
{

    public bool isPressed;



    Сообщение Unity | Ссылок: 0
    void Start()
    {


        this.RepeatUntil(() => true, Repeat, null, 1);

    }


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



    Ссылок: 0
    private void OnDone()
    {
        print("OperationDone!");
    }


}
```

**HOW TO STOP OPERATION?**

The Delay, Wait Until, and Repeat Until methods return Coroutine. You can stop the operation by caching the resulting coroutine and then stopping it using the standard Unity method "StopCoroutine".