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 calling a single command (Delay, Wait Until, Repeat Until...). the Plugin does not require inheriting your class from a special one 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?

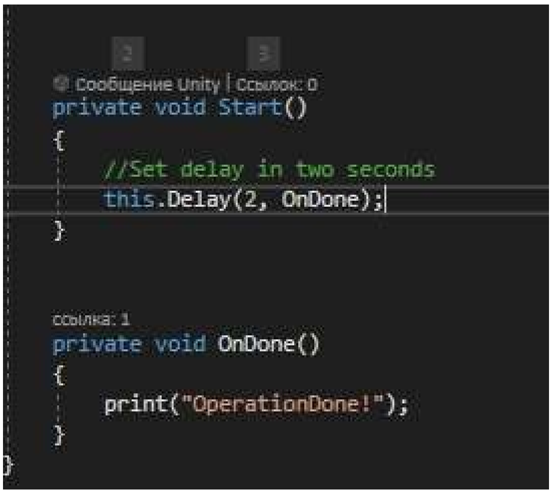
To delay the operation for a certain time, use the Delay command.

Task: delay the execution of the method that outputs the message for 2 seconds after starting.

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.

Example:



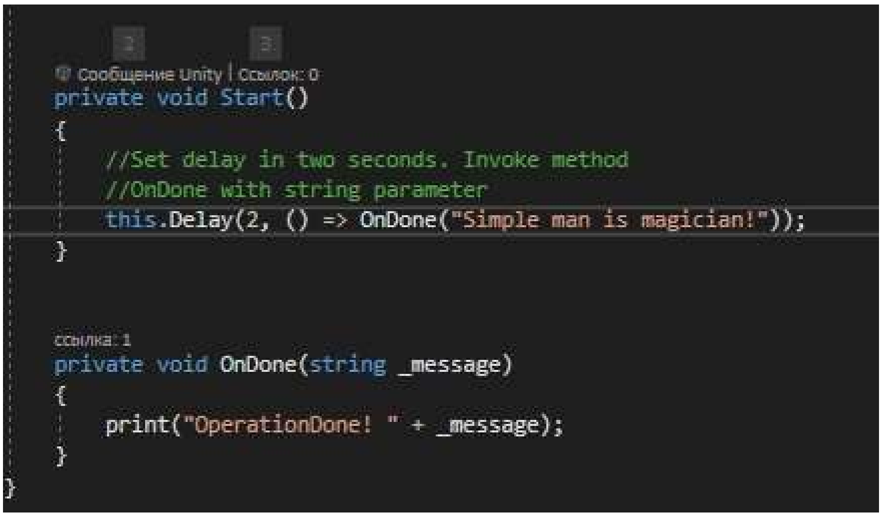
```
Сообщение Unity | Ссылок: 0
private void Start()
{
    //Set delay in two seconds
    this.Delay(2, OnDone);
}


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

Bottom line: when you click on "Play" after 2 seconds, the OnDone method displays a message in the console.

HOW TO USE DELAY WITH A PARAMETER?

To pass an argument to the OnDone(string_message) method, use a lambda expression.

Example:



```
    Сообщение Unity | Ссылок: 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);
    }
}
```

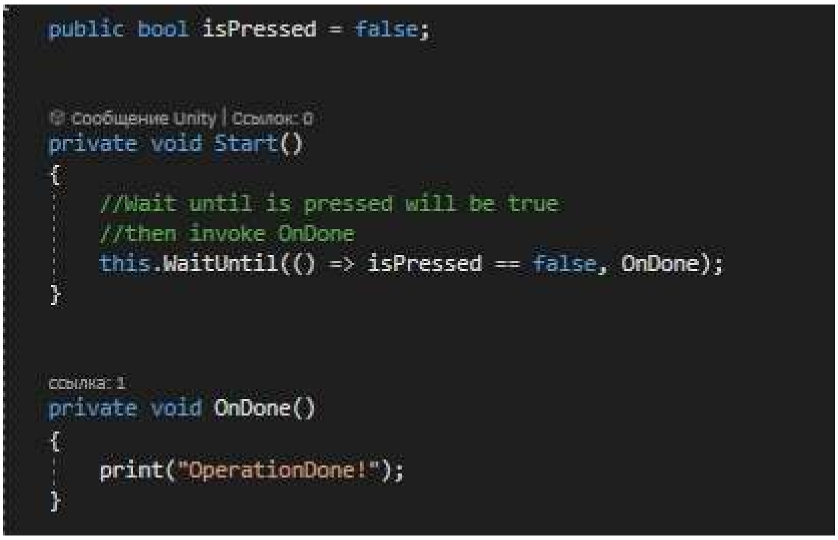
HOW TO USE WAIT UNTIL?

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.

Task: call the OnDone method after the isPressed field is set to true.

To do this, call the Wait Until command in the Start method and pass IsPressed as a condition via a lambda expression.

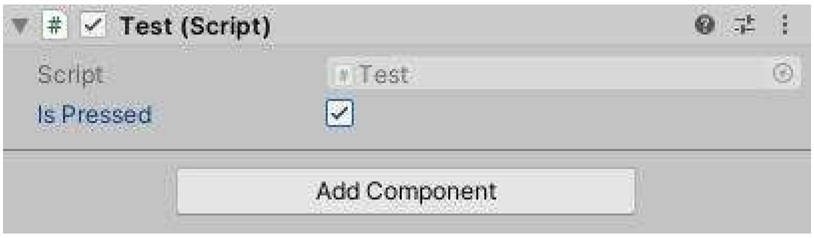
Example:



```
public bool isPressed = false;


Сообщение Unity | Ссылок: 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!");
}
```

Check:

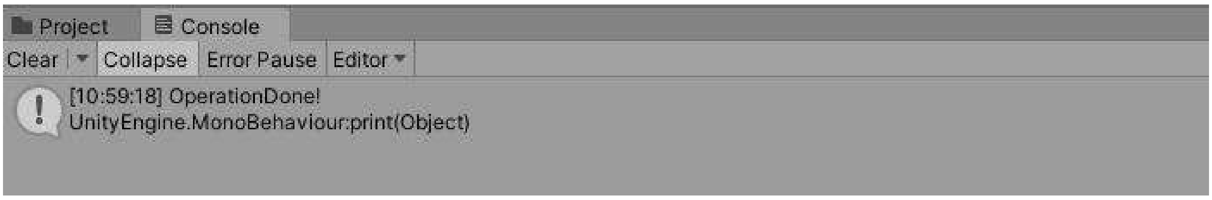
When you click on IsPressed a message appears in the console



Test (Script)

Script Test

Is Pressed ☑



```
[10:59:18] OperationDone!
UnityEngine.MonoBehaviour:print(Object)
```

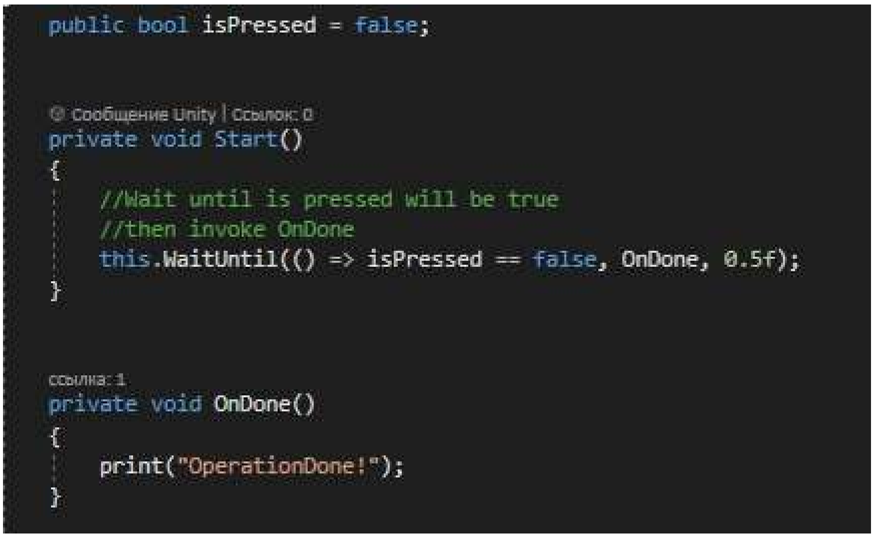
HOW TO USE WAIT UNTIL WITH DELAY?

There are also situations that require resource-intensive operations that 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 performance loss.

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

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

Example:



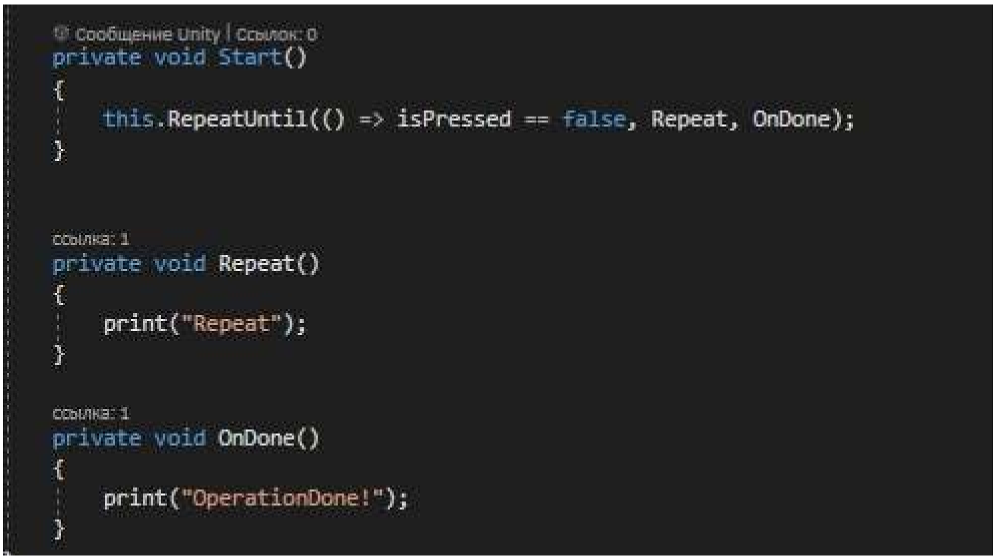
```
public bool isPressed = false;


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


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

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

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

Example:



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


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

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

HOW TO USE REPEAT FOREVER?

The Repeat Forever command is called in a very similar way. This is a simplified version of Repeat Until.

* Repeat Until and Repeat Forever as well as Wait Until may have a delay in execution.

HOW TO STOP OPERATION?

All the methods specified here return Coroutine. You can stop the operation by caching the received coroutine and then stopping it using the standard Unity "StopCoroutine" method.