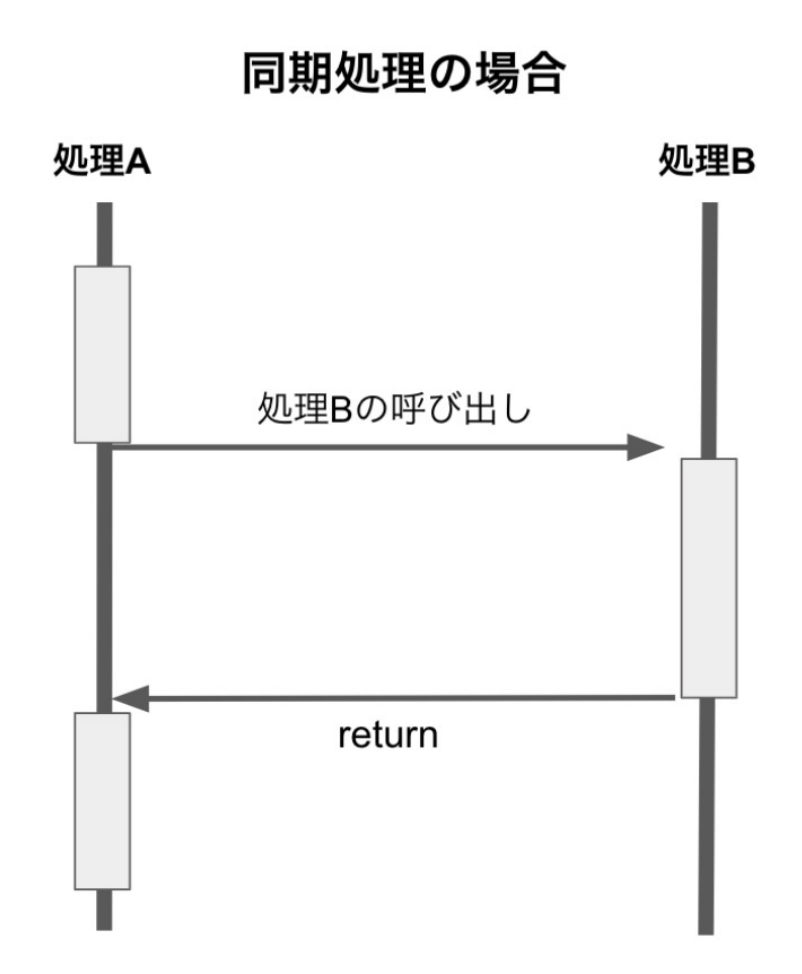
非同期処理とコルーチン

目次

１．非同期処理とは

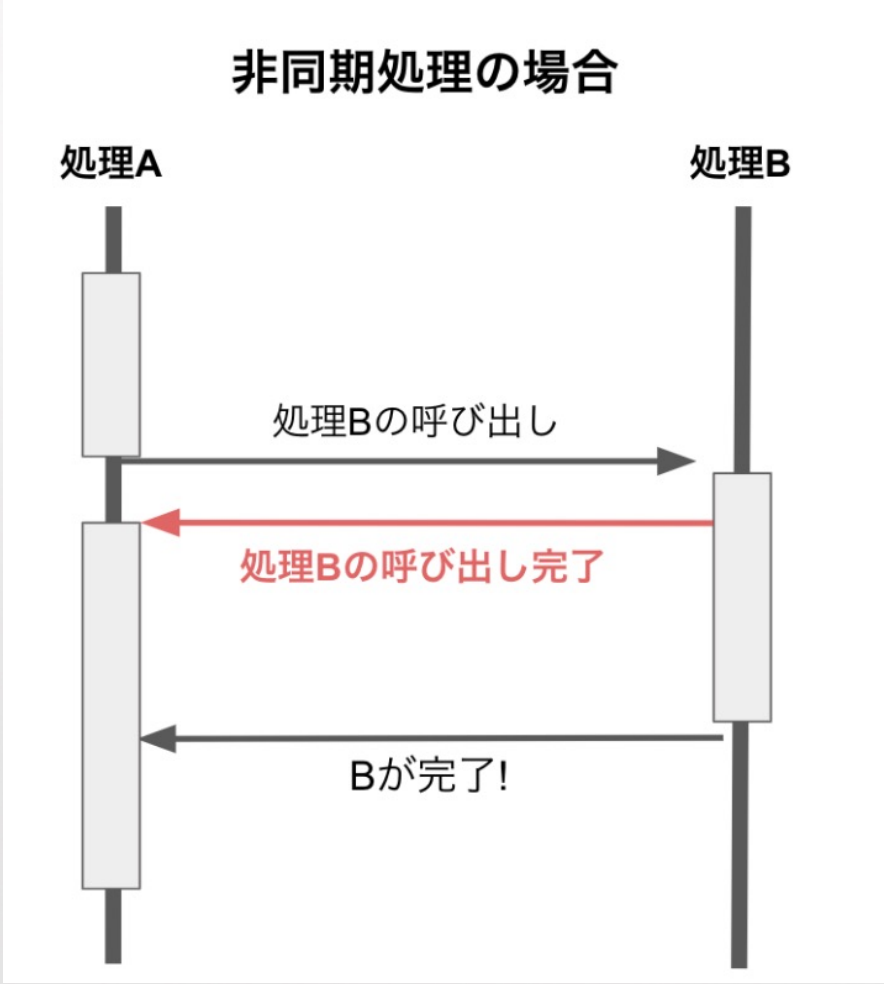
同期処理：複数の処理がある場合、一つずつ順番に処理していくもの。



処理Aの途中で処理Bが入った場合、
処理Aは停止し、処理Bを実行。
処理Bが完了しだい、処理Aの続き
から実行する。
(通常の関数呼び出しのイメージ)

１．非同期処理とは

非同期処理：複数の処理がある場合、完了を待たずに実行していくもの



処理Aの途中で処理Bが入った場合、
処理Aは停止せず、処理Bを実行。

２．コルーチンとは

Unityではこの非同期処理をコルーチンという独自のシステムで導入している。
(元々C#にはasybc/awaitという非同期処理があるが、とっつきにくく複雑なのでUnityが簡略化した機能)

このコルーチンがめちゃくちゃ優秀なので、ぜひ覚えてください！
コルーチンは１項で話をした非同期処理を実現できるのですが、それと同時にプログラムの中でn秒待つ、という待機処理を実装することが出来ます。

時間をカウントする処理については今まではUpdateの中で実施していましたが、このコルーチンを使用した方が分かりやすくかつ、簡単に記載することが出来ます。

２．コルーチンとは

では、実際に比較して書いていきたいと思います！
まずは、以下の内容を今までやってきたupdateの中でTime.deltaTimeを使用して以下の処理を実装してみてください。

⇨新規スクリプトファイル：test1.cs

ゲーム開始から5秒後に「5秒経過！」と1回だけDebug.Logを出力してください。

出来た人は、次にコルーチンを使用して作成してみましょう！

⇨新規スクリプトファイル：test2.cs

test2.csのページを見て実装してみてください。

test1.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class test1 : MonoBehaviour
6  {
7      private float timeSpan;
8      private bool isFinish;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         timeSpan = 5.0f;
14         isFinish = false;
15     }
16
17     // Update is called once per frame
18     void Update()
19     {
20         if (timeSpan <= 0)
21         {
22             if (isFinish == false)
23             {
24                 isFinish = true;
25                 Debug.Log("5秒経過!");
26             }
27         }
28         else
29         {
30             timeSpan -= Time.deltaTime;
31         }
32     }
33 }
34
```

test2.cs

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

public class test2 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        StartCoroutine("WaitSecond");
    }

    // Update is called once per frame
    void Update()
    {

    }

    IEnumerator WaitSecond()
    {
        //5秒停止
        yield return new WaitForSeconds(5);

        //出力
        Debug.Log("5秒経過!");
    }
}

```

2．コルーチンとは

＜プログラム解説＞

```
StartCoroutine("WaitSecond");
```

コルーチンを実行するという命令。引数は関数名を指定する。

```
IEnumerator WaitSecond()
{
    //5秒停止
    yield return new WaitForSeconds(5);

    //出力
    Debug.Log("5秒経過！");
}
```

戻り値はUnityが使っているので、使えない(IEnumerator固定)

待つ処理は1行のみで指定できる。

(yield return new WaitForSeconds(n);)

３．コルーチンの使い方

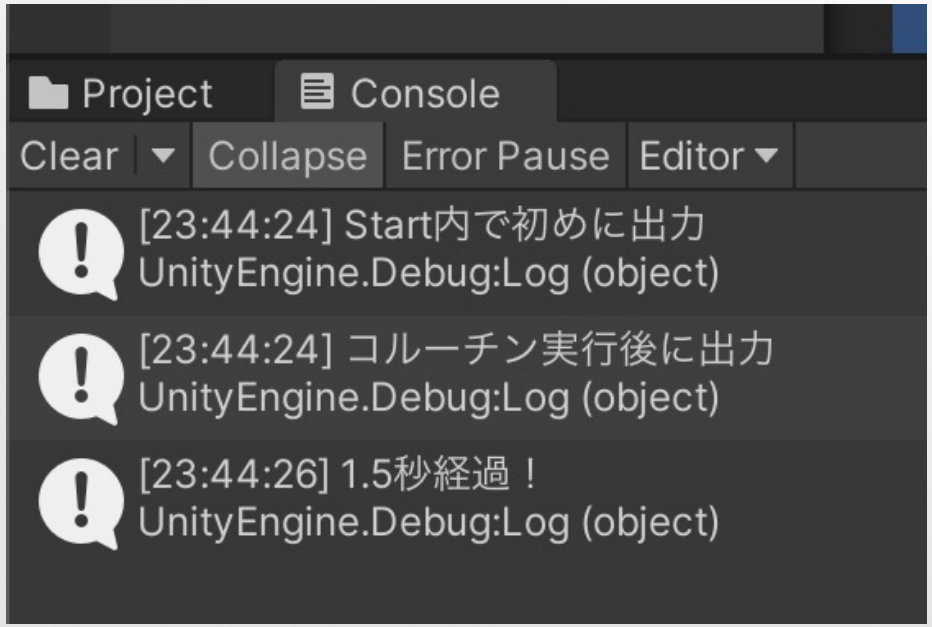
では、次に非同期処理の確認を実施していきましょう！

以下のプログラムを作成してみてください。

⇨新規スクリプトファイル：test3.cs

test3.csのページを見て実装してみてください。

Debug.Logが出力される順番を確認してください！



test3.cs

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

public class test3 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        Debug.Log("Start内で初めに出力");
        StartCoroutine("WaitSecond");
        Debug.Log("コルーチン実行後に出力");
    }

    // Update is called once per frame
    void Update()
    {

    }

    IEnumerator WaitSecond()
    {
        //1秒停止
        yield return new WaitForSeconds(1.5f);

        //出力
        Debug.Log("1.5秒経過!");
    }
}
```

３．コルーチンの使い方

非同期処理なので、Start関数自体はすぐに抜けて終了。
WaitSecondの関数はまた別で1.5秒待って、その後実装される。

コルーチンでは同期処理を実施出来ないので注意！！
(Start関数の中で1.5秒待つみたいな)
それはC#のasync/awaitの機能を使えば良いが、また別の機会に紹介しますね！

また、n秒待つという他に特定を満たすまで待機することも可能です。
⇨新規スクリプトファイル：test4.cs
test4.csのページを見て実装してみてください。

test4.cs

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class test4 : MonoBehaviour
{
    [SerializeField] private int testValue;
    // Start is called before the first frame update
     void Start()
    {
        StartCoroutine("WaitSecond");
    }

    // Update is called once per frame
    void Update()
    {

    }

    IEnumerator WaitSecond()
    {
        Debug.Log("コルーチン実行中");

        //10以上になるまで待機する
        yield return new WaitUntil(() => testValue >= 10);

        //出力
        Debug.Log("10以上");
    }
}

```

３．コルーチンの使い方

＜プログラム解説＞

```
IEnumerator WaitSecond()
{
    Debug.Log("コルーチン実行中");

    //10以上になるまで待機する
    yield return new WaitUntil(() => testValue >= 10);

    //出力
    Debug.Log("10以上");
}
```

yield return new WaitUntil(() => testValue >= 10);

⇨赤線部分が条件式。「() =>」はラムダ式というC#独特の書き方。

赤線の条件を満たすまで待機しているはずです！

４．演習問題

＜問題1(test5.cs)＞

コルーチンの中で「Aが押されるまで待機」と表示し、Aが押されたら「Aが押された」と表示させてください。

(1回のみ実行可能としてください)

＜問題2 (test6.cs) ＞

コルーチンを使って1秒ごとに1⇨2⇨3…というようにDebug.Logに出力させてください。(無限に)

＜問題3 (test7.cs) ＞

3秒後にbool型の変数をfalseからtrueにしてください。update内でbool型の変数がtrueになった際に「3秒経過」と表示してください。

４．演習問題

＜問題4(test8.cs)＞

Bを押した後、2秒後に「Bが押された」と出力してください。

1回のみではなく、何回も実行出来るようにしてください。

＜問題5 (test9.cs) ＞

コルーチンを使って3秒後に「3秒経過」と表示し、さらに2秒後に「5秒経過」と表示させてください。

test5.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using Unity.VisualScripting;
4  using UnityEngine;
5
6  public class test5 : MonoBehaviour
7  {
8      private bool isPushKey;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         isPushKey = false;
14
15         //コルーチン実行
16         StartCoroutine("WaitSecond");
17     }
18
19     // Update is called once per frame
20     void Update()
21     {
22
23
24
25
26     }
27
28     IEnumerator WaitSecond()
29     {
30         Debug.Log("Aが押されるまで待機");
31
32         //10以上になるまで待機する
33         yield return new WaitUntil(() => Input.GetKey(KeyCode.A));
34
35         //出力
36         Debug.Log("Aが押された");
```

```
37    }
38 }
```

test6.cs

```cs
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class test6 : MonoBehaviour
{
    private int timeCount;

    // Start is called before the first frame update
    void Start()
    {
        timeCount = 0;

        //コルーチン実行
        StartCoroutine("WaitSecond");
    }

    // Update is called once per frame
    void Update()
    {

    }

    IEnumerator WaitSecond()
    {
        //コルーチンの中だと無限ループでも大丈夫です!
        while (true)
        {
            //1秒停止
            yield return new WaitForSeconds(1);

            //カウント
            timeCount++;
            Debug.Log(timeCount);
        }
    }
```

```
}

```

test7.cs

```cs
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class test7 : MonoBehaviour
{
    private bool isFlag;

    // Start is called before the first frame update
    void Start()
    {
        isFlag = false;

        //コルーチン実行
        StartCoroutine("WaitSecond");
    }

    // Update is called once per frame
    void Update()
    {
        //trueかどうか判定
        if (isFlag == true)
        {
            Debug.Log("3秒経過");
        }
    }

    IEnumerator WaitSecond()
    {
        //5秒停止
        yield return new WaitForSeconds(3);

        //フラグ切り替え
        isFlag = true;
    }
```

```
37 }
38
```

test8.cs

```cs
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class test8 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        //Bが押されたか判定
        if (Input.GetKey(KeyCode.B))
        {
            //コルーチン実行
            StartCoroutine("WaitSecond");
        }
    }

    IEnumerator WaitSecond()
    {
        //2秒停止
        yield return new WaitForSeconds(2);
        Debug.Log("Bが押された");
    }
}

```

test9.cs

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class test9 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        //コルーチン実行
        StartCoroutine("WaitSecond");
    }

    // Update is called once per frame
    void Update()
    {

    }

    IEnumerator WaitSecond()
    {
        //3秒停止
        yield return new WaitForSeconds(3);
        Debug.Log("3秒経過");

        //2秒停止
        yield return new WaitForSeconds(2);
        Debug.Log("5秒経過");
    }
}

```