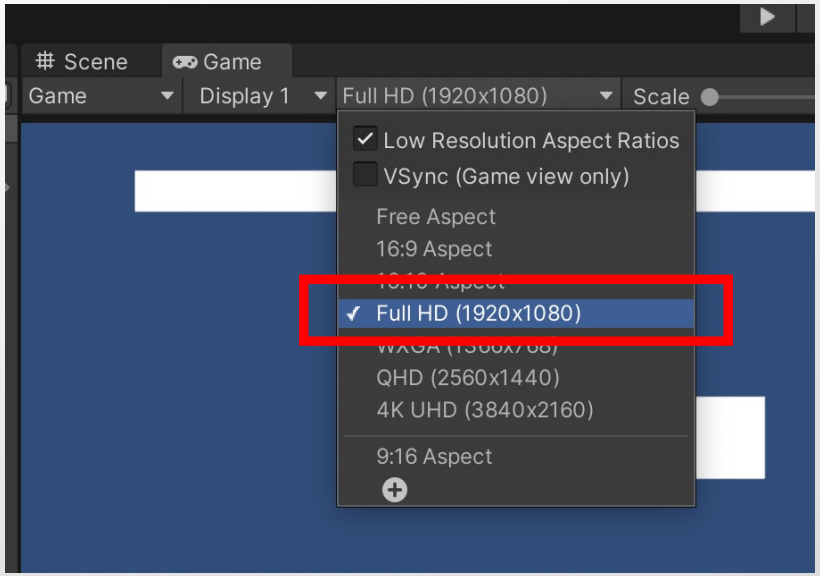
関数の復習+α

目次

１．関数の復習

「02_C#プログラミング.pdf」にて、関数とは何か？という話をしましたが、
理解出来ていますでしょうか？

具体的にどのように処理を分けるか、という練習が出来ていなかったのでここで
再度復習していきたいと思います。
「04_Data.unitypackage」をインポートしてください。
その後、以下の設定を忘れずにお願いします。



１．関数の復習

まずは、プレイヤーがどういった機能を持っているのかゲームを実行して試してみましょう！

確認出来たら、Move.csのスクリプトを確認してみてください。
全ての処理がUpdateの中に書かれているので、とても分かりにくい感じになってしまっています。。。
なので、Updateの中を自分なりに関数化して分かりやすく整えてみましょう！

必ずこういう関数に分けないといけない！
という正解は無いので、自分なりに考えてやってみてください。

１．関数の復習

＜自分の例＞

機能ごとに分けてUpdate内に関数を作成するかなぁ。。。。

・キー入力処理(横移動、ジャンプ)

・移動処理(rb2.velocity)

・背が伸びる

・色が変わる

・重力反転

２．GameObjectの取得方法

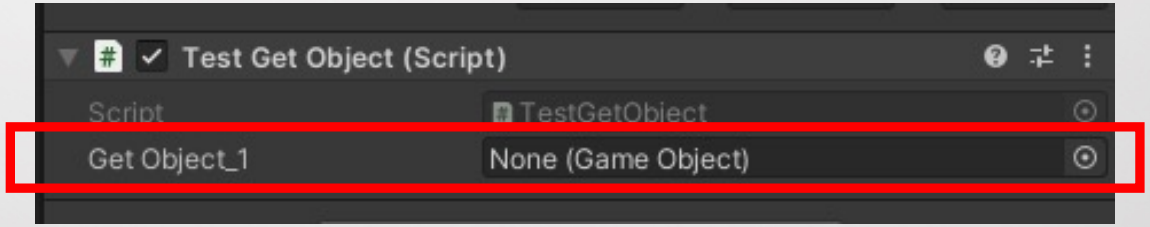
別のGameObjectを参照する場合、今まではアタッチ(Unity上でドラッグ)していましたが、プログラムの中からでも別のGameObjectを参照することが出来るため、ここではそのやり方を紹介していきます。

⇨TestGetObject.csを作成し、以下の①から記載をお願いします。
　空のGameObjectを作成し、上記のスクリプトをアタッチしてください。

①Unity上でアタッチ

⇨スクリプト内でPublicもしくはSerializeFieldを宣言し、Unity上からアタッチする方法。

```
public class TestGetObject : MonoBehaviour
{
    [SerializeField] private GameObject GetObject_1;
```



2．GameObjectの取得方法

②名前で検索

⇨スクリプト内でヒエラルキー上にあるGameObjectを名前で検索し、一致するGameObjectがある場合にアタッチする方法。

```
GameObject obj = GameObject.Find("Square");
```

③タグで検索(単体)

⇨GameObject内に設定しているtagで検索をかける方法。

複数ある場合は、ヒエラルキーの上から1つ目のみを取得する。

```
GameObject obj2 = GameObject.FindGameObjectWithTag("Player");
```

2．GameObjectの取得方法

④タグで検索(複数)

⇨GameObject内に設定しているtagで検索をかける方法。

複数ある場合でも、順に配列に格納される。

```
GameObject[] obj3 = GameObject.FindGameObjectsWithTag("Player");
```

⑤アタッチされているコンポーネントで検索

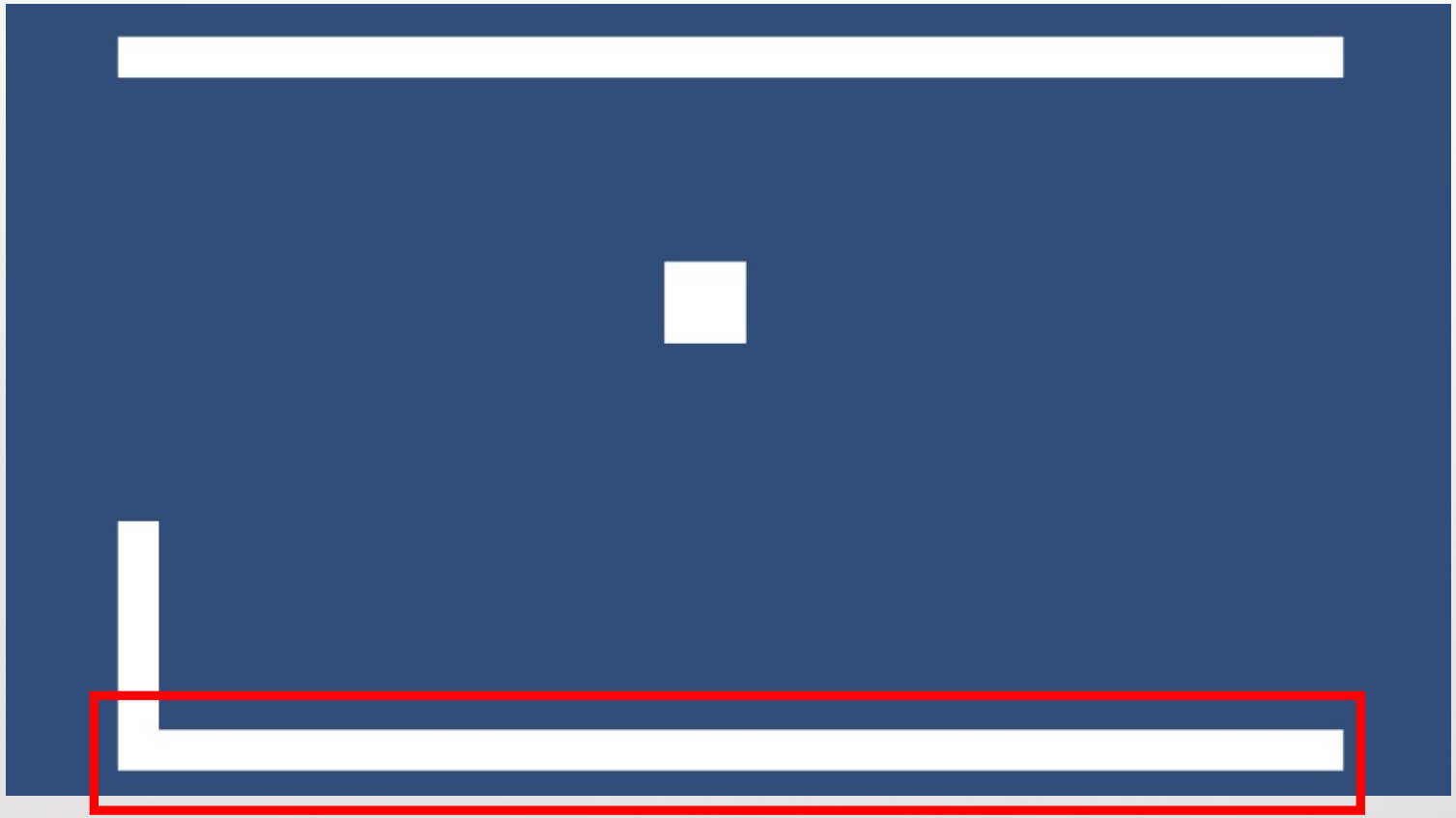
⇨Rigidbody2Dが付いているGameObjectを探す、など付与されている
コンポーネントで検索する方法。

```
GameObject obj4 = FindObjectOfType<Rigidbody2D>().gameObject;
```

2．GameObjectの取得方法

＜練習問題＞

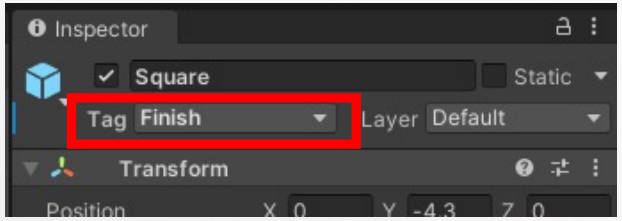
先ほどのGameObjectの取得方法の②、③のそれぞれのやり方で、
下の床の色を黒色にしてみてください。(tagは自由につけてください)



2．GameObjectの取得方法

＜解答＞

下の床はSquareのGameObjectになる。Tagもつけないといけないので、「Finish」に設定しておく。(何でも大丈夫です。。。)




```
//②のやり方
GameObject obj = GameObject.Find("Square");
obj.GetComponent<SpriteRenderer>().color = Color.black;

//③のやり方
GameObject obj2 = GameObject.FindGameObjectWithTag("Finish");
obj2.GetComponent<SpriteRenderer>().color = Color.black;
```

３．ランダムな値

Unityではランダムな値を簡単に取得することが出来ます！
ランダム性を加えるとゲームが一気に面白くなるので、ぜひどんどん使用してください！

使い方は以下の通りです。

```
int rnd = Random.Range(0,10);
```

⇨ 0から9までの10個の中からランダムな値を返す

これだけでも十分使用出来ますが、もう1歩進んで次ページの内容も理解してもらうと良いかと思います！

３．ランダムな値

Unityに限らず、プログラムの中では完全なランダム値というのは存在しません。
今回のランダム値も同じで、seed値という値を基準にランダム値が作られています。
Seed値が同じ値だった場合、ランダム値は同じ値を出します。(以下、確認)

```
int value;

//シード値を10に指定
Random.InitState(10);

//ランダム値の生成・出力
for(int i = 0; i < 5; i++)
{
    value = Random.Range(0, 10);
    Debug.Log(value);
}
```

３．ランダムな値

なので、毎回異なるSeed値を出す必要があります。
よくやるやり方は、以下のものがあります。(今の時刻のミリ秒を返す)
これで、まずかぶることが無い値を出すことが出来ると思います。

```
//現在時刻のミリ秒でシード値を初期化
Random.InitState(System.DateTime.Now.Millisecond);
```

３．ランダムな値

＜練習問題＞

①80パーセントの確率で「当たり」と表示し、それ以外は「外れ」とDebug.Logに出力させてください。

②0.01パーセントの確率で「当たり」と表示し、それ以外は「外れ」とDebug.Logに出力させてください。

③30パーセントの確率で「当たり」と表示し、20.5パーセントの確率で、「外れ」、49.5パーセントの確率で「もう1回」とDebug.Logに出力させてください。

３．ランダムな値

＜解答＞

①

```
int value = Random.Range(0, 100);

if (value < 80)
{
    Debug.Log("当たり");
}
else
{
    Debug.Log("外れ");
}
```

②

```
int value = Random.Range(0, 10000)

if (value < 1)
{
    Debug.Log("当たり");
}
else
{
    Debug.Log("外れ");
}
```

③

```
int value = Random.Range(0, 1000);

if (value < 300)
{
    Debug.Log("当たり");
}
else if (value < 505)
{
    Debug.Log("外れ");
}
else
{
    Debug.Log("もう1回");
}
```