カメラが Unity ちゃんを追いかけるようにする。

1. MainCamera

今回横スクロールのアクションゲームを作成するために、右方向にシーンが切り替わるようにしてみます。そのために、まず Unity ちゃんが右方向に移動したときに、カメラが Unity ちゃんを追いかけるようにしていきましょう。

そして、逆方向に移動した場合は、カメラを移動させず、Unity ちゃんが画面から出ないように、左端以上移動しないように設定してみましょう。

1. Player.cs からカメラが利用できるように、GameObject を public で作成する。
2. UnityChan のインスペクター▶Player(Script)で、先ほどの GameObject に MainCamera をバインドする。
3. Unity ちゃんの位置と、カメラの位置を比較し、変化があればカメラの位置を Unity ちゃんの位置に設定する。

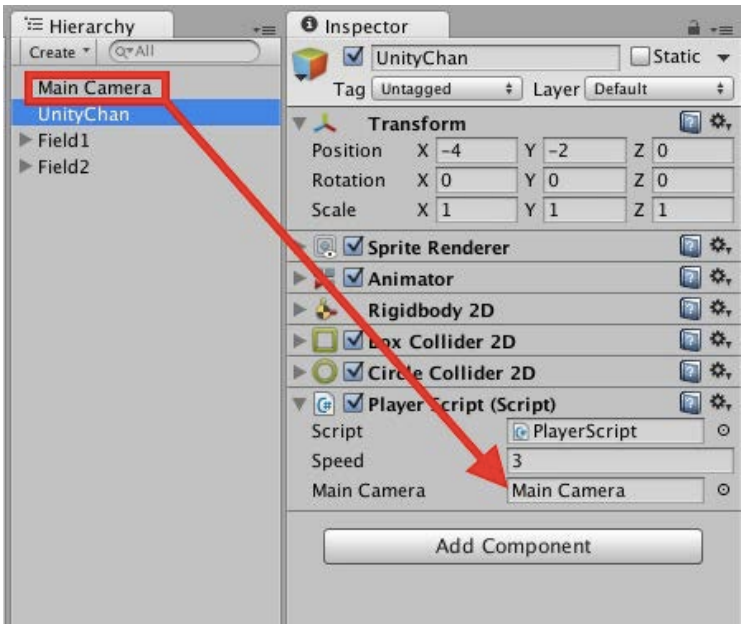
まずは、現在のソースに上記要件を追加します。
10 行目でカメラ用 GameObject を設定する。
12 行目で位置比較用のオフセット値を設定
キー入力後、Unity ちゃんのアニメーション切り替え後 27 行目から 49 行目がカメラの位置設定になります。

Player.cs

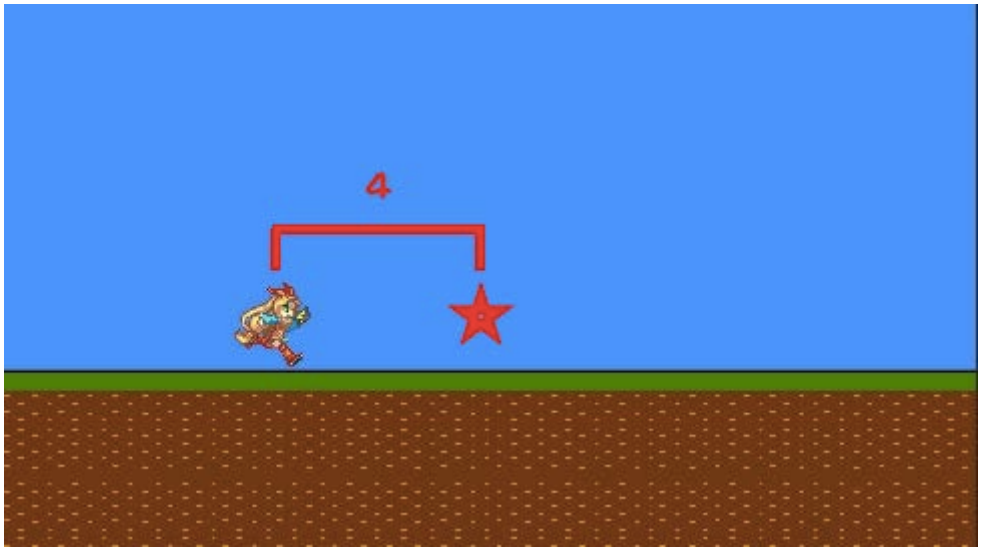
```
 1        using UnityEngine;
 2        using System.Collections;
 3
 4        public class Player : MonoBehaviour {
 5
 6            public float speed = 4f; //歩くスピード
 7            private new Rigidbody2D rigidbody2D;
 8            private Animator anim;
 9            // 1.
10            public GameObject mainCamera;
11            // 3. 位置比較のためのオフセット値
12            public readonly int CAMERA_OFFSET=0;
                                  :
                                  :
                                  :
23                  transform.localScale = temp;
24                      //Wait→Dash
25                  anim.SetBool ("Dash", true);
26
27                  //3. Camara の追従
28                  // 画面中央から左に CAMERA_OFFSET 以上移動するとカメラが追いかける。
29                  if (transform.position.x >
30                          mainCamera.transform.position.x - CAMERA_OFFSET)
31                  {
32                      //カメラの位置を取得
33                      Vector3 cameraPos = mainCamera.transform.position;
34                      //ユニティちゃんの位置から右に CAMERA_OFFSET 移動した位置を
35                      //画面中央にする
36                      cameraPos.x = transform.position.x + CAMERA_OFFSET;
37                      mainCamera.transform.position = cameraPos;
38                  }
```

```
39                  //カメラ表示領域の左下をワールド座標に変換
40                  Vector2 min =
41                      Camera.main.ViewportToWorldPoint(new Vector2(0, 0));
42                  //カメラ表示領域の右上をワールド座標に変換
43                  Vector2 max =
44                      Camera.main.ViewportToWorldPoint(new Vector2(1, 1));
45                  //ユニティちゃんのポジションを取得
46                  Vector2 pos = transform.position;
47                  //ユニティちゃんの x 座標の移動範囲を Clamp メソッドで制限
48                  pos.x = Mathf.Clamp(pos.x, min.x + 0.5f, max.x);
49                  transform.position = pos;
50
51              } else {
52                  //左も右も入力していなかったら
53                  //横移動の速度を 0 にしてピタッと止まるようにする
54                  rigidbody2D.velocity = new Vector2 (0, rigidbody2D.velocity.y);
55                  //Dash→Wait
56                  anim.SetBool ("Dash", false);
57              }
58          }
59      }
```

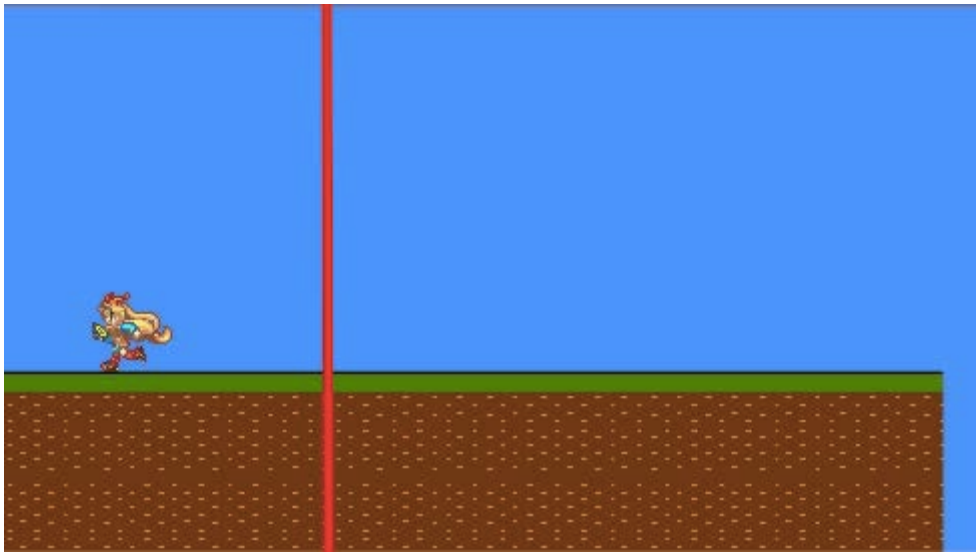
ソース入力後、UnityChan のインスペクター▶Player(Script)で、作った public 変数 mainCamera に、Main Camera を指定します。



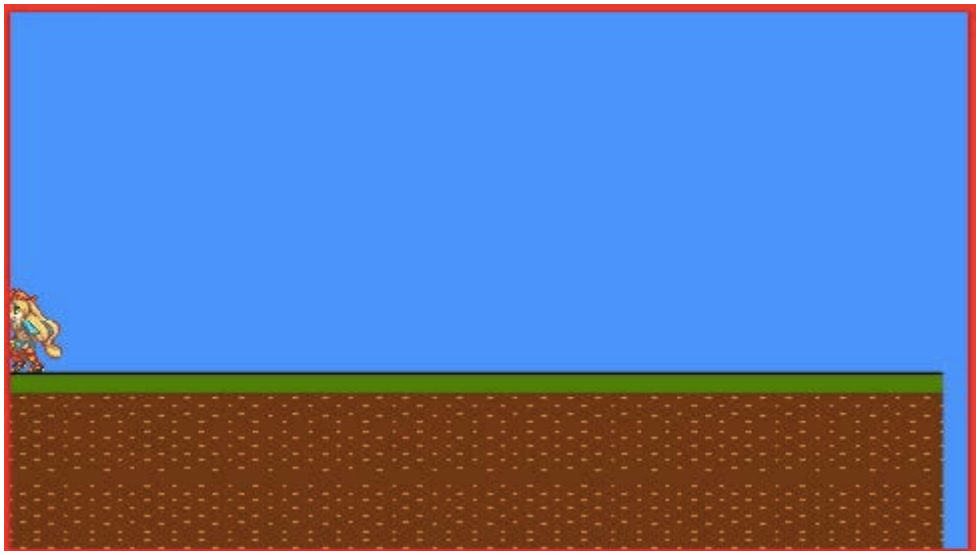
これで、画面中央から CAMERA_OFFSET 離れた位置をユニティちゃんが超えると、Main Camera もそれに合わせて移動するようになります。下記例では 4 に設定した場合です。



また左に移動した場合、カメラ位置は動かしません



そして Mathf.Clamp メソッドで Unity ちゃんの移動位置を制限しているため画面からはみ出ません。



ポイント

Transform： オブジェクトの位置、回転、スケールを扱うクラス
Unity Documents
https://docs.unity3d.com/jp/current/ScriptReference/Transform.html

シーン内のすべてのオブジェクトは Transform を持ちます。 Transform はオブジェクトの位置、回転、スケールを格納し、操作するために使用されます。 すべての Transform は親を持ち、階層的に位置、回転、スケールを適用することができます。これはヒエラルキーウインドウで階層を見ることができます。

Mathf.Clamp： 与えられた最小 float 値と最大 float 値の範囲に値を制限します。
Unity Documents
https://docs.unity3d.com/ja/540/ScriptReference/Mathf.Clamp.html

読み込み専用変数の設定。
const と static readonly の使い分け

const フィールドは、コンパイル時定数の扱い。（MSDN より）
〇変数のように扱える定数（暗黙的 static）
〇宣言時にのみ初期化可能（コンパイル時に値が埋め込まれる）
〇readonly より実行速度が速い
〇switch 文やデフォルト引数に使える
〇インスタンスを new した結果は割り当てられない（C#の組み込み型のみ）

readonly フィールドは、実行時定数の扱い。（MSDN より）
〇実際は、読み取り専用の代入不可な変数
〇宣言時の他に、コンストラクタ内でも初期化可能
〇定数である const よりは、僅かに実行速度が遅い
〇switch 文やデフォルト引数には使えない
〇インスタンスを new した結果を割り当てられる

static readonly
〇const が使いたいけど、使えない場合に、static readonly を使用する。

定数値のシンボル名が必要で、その値の型を const 宣言で使用できない場合、またはその値をコンパイル時に計算できない場合は、static readonly フィールドが役に立ちます。(MSDN – 定数用の static readonly フィールド)

まとめ
〇基本的に、static readonly を使用する。
〇const は、属性に指定するパラメータや列挙型の定義など、コンパイル時に値が必要な場合にのみ使用する。
〇Effective C# でも、const よりも readonly の使用が推奨されている。

高いパフォーマンスが求められていて、なおかつ将来にわたって変更されることがないことが明らかな場合にのみコンパイル時定数を使用するべき。