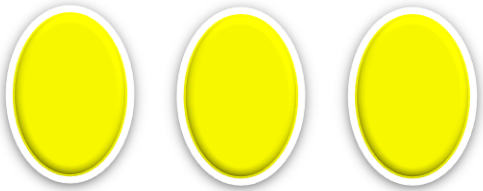
コイン集めゲームの作成



教職員対象パソコン活用研修会
R07/08/05 実施
HICS 広島情報専門学校
〒732-0816 広島市南区比治山本町 16-35
TEL (082) 252-4411 FAX (082) 256-4450
[URL] http://www.hi-joho.ac.jp/

作るゲームのルール

- フィールドに落ちているコインを
  全部集めたら「CLEAR！」と表示する。

組込むゲームルールについて（実装する内容）

①ゲームのマップに設定されている、コインの総数をカウントする変数を作成する(public static int coin_count)。

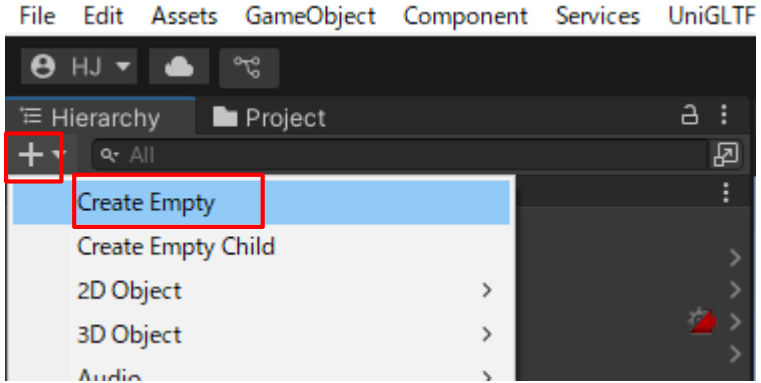
②コインをマップにセットするごとにコインの総数のカウントする変数を1加算するよう、プログラムする。

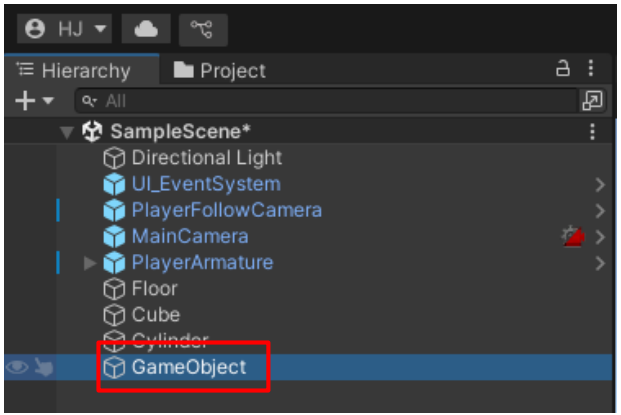
③プレイヤーがコインを取る(当たり判定処理)たびにコインの総数のカウントする変数を1減算するよう、プログラムする。

④コインの総数のカウントする変数が0になったら、ゲームクリアを表示するようにする。

⑤完成したコインをプレハブ化して、マップ中に好きなだけ置けるようにする。

ゲームルールを制御する部分の作成

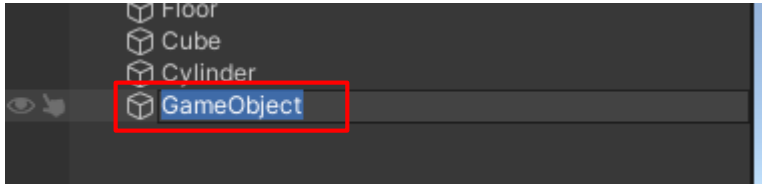




＋をクリックして、

CreateEmptyをクリックして、

空のゲームオブジェクトを作成します。

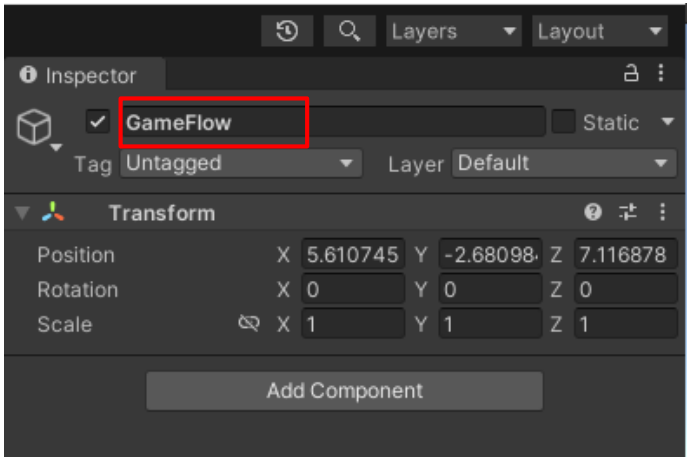


空のゲームオブジェクトの名前は
GameObjectとなっています。

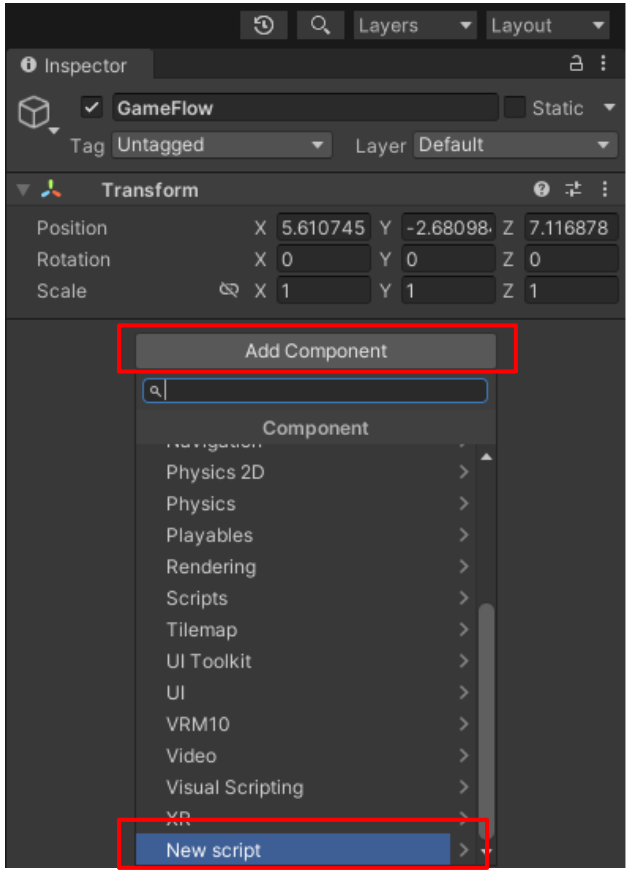
名前をクリックすると名前を

変更できるようになります。

名前はGameFlowに変更します。

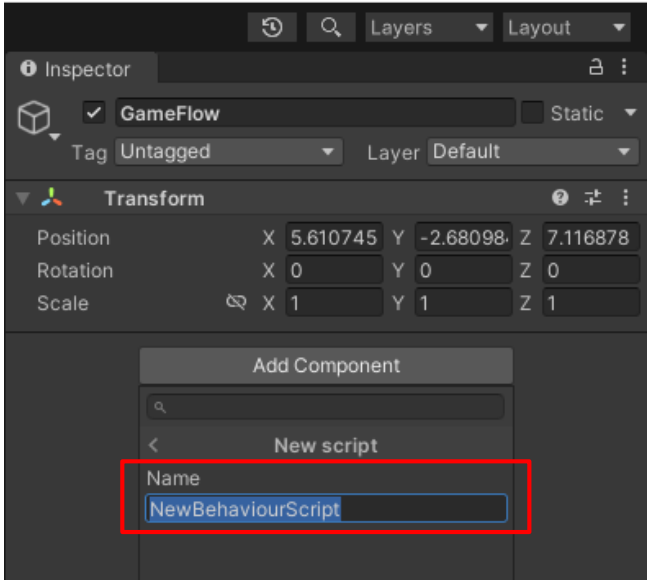


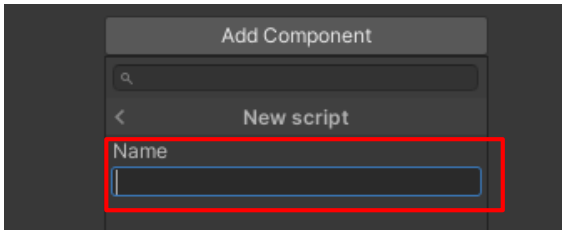
Inspectorの名前にもGameFlowと表示されます。

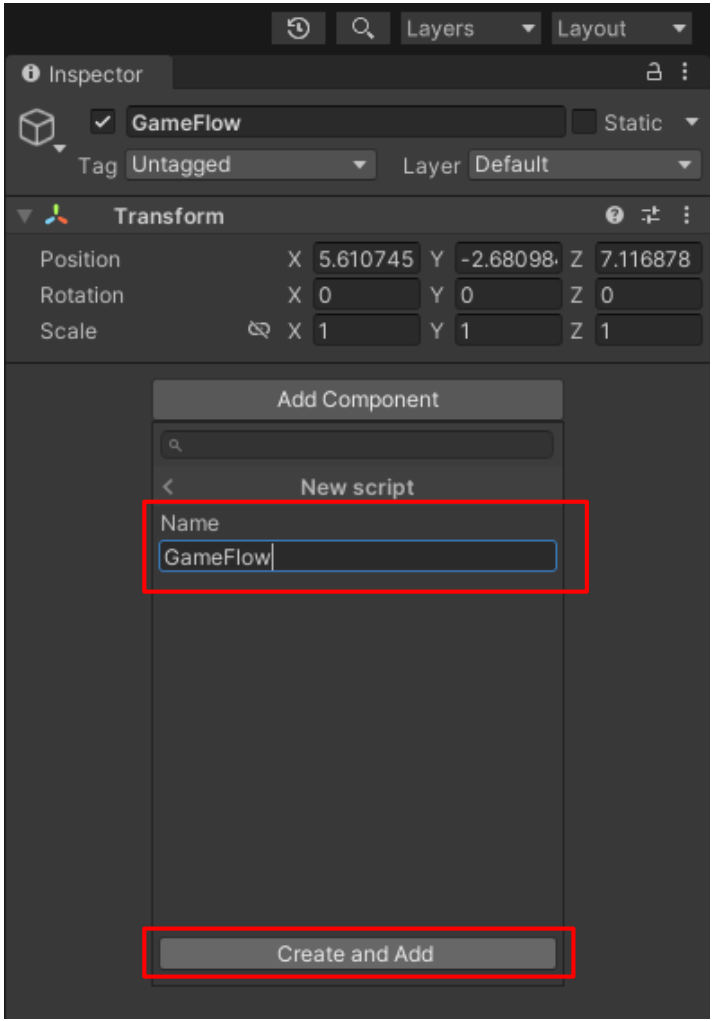


ゲームの流れのゲームオブジェクトにスクリプトを作成します。Add Conponentをクリックして、一覧の中のNew Scriptをクリックします。

Nameの欄にはNewBehaviourScriptと入っているので、デリートキーで消してください。

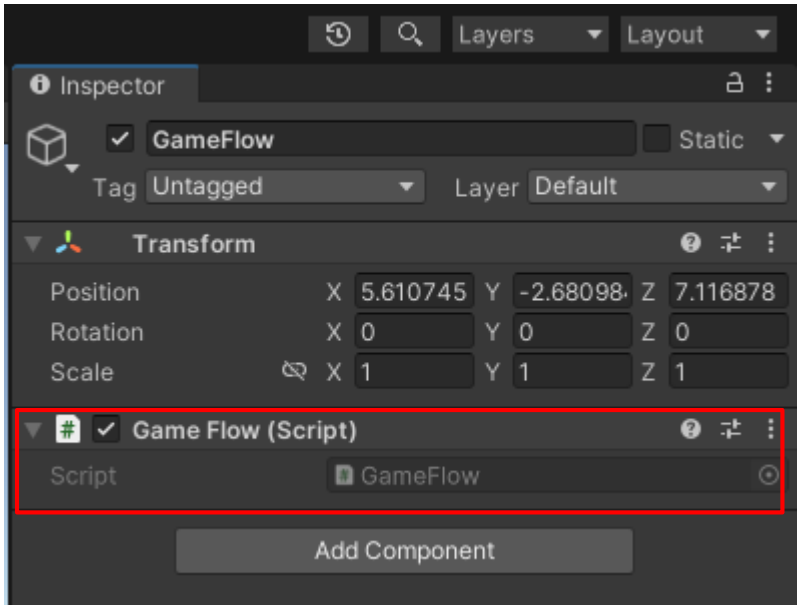


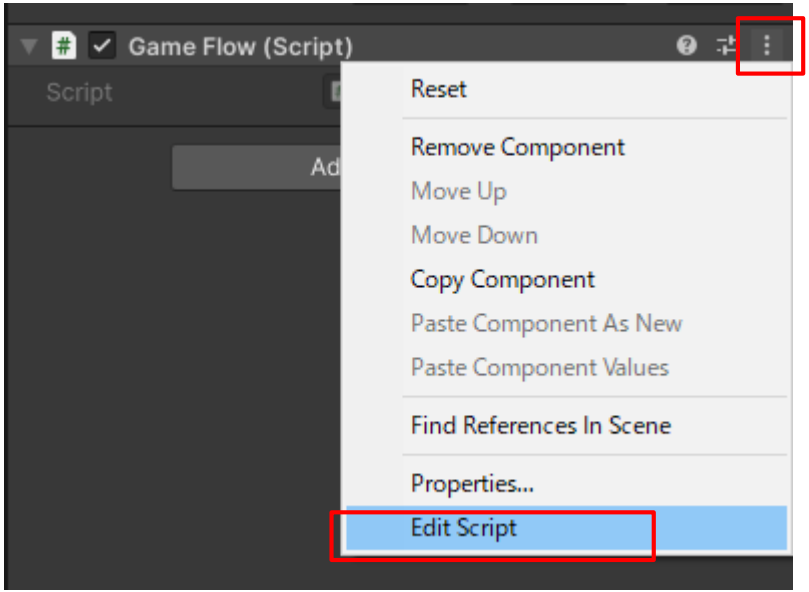




Nameの欄をGameFlowと入力して、Create and Addをクリックしてください。

しばらくすると、GameFlowというスクリプトのベースが作成されます。

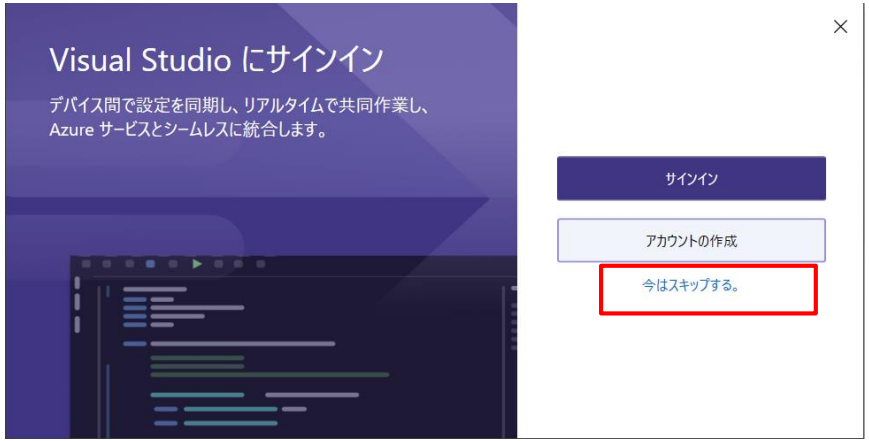


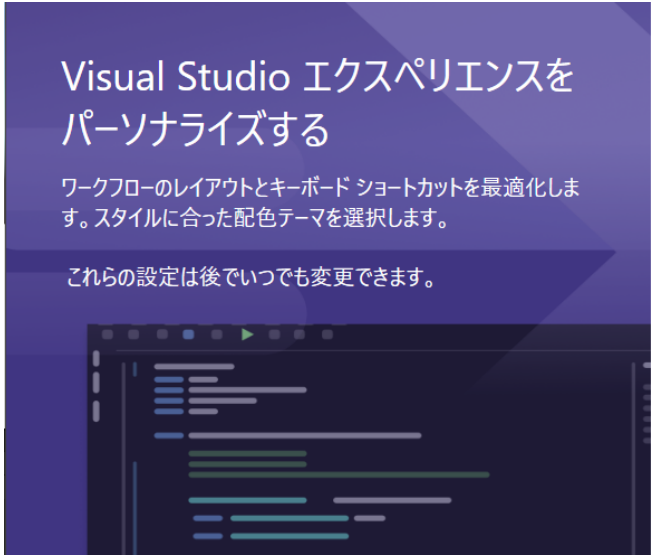


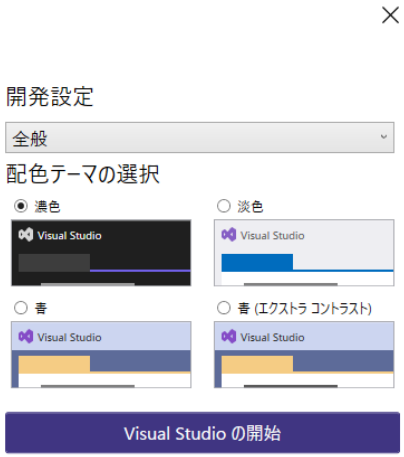
詳細ボタンをクリックして、メニュー一覧のEditScriptをクリックしてVisualStudioを起動します。

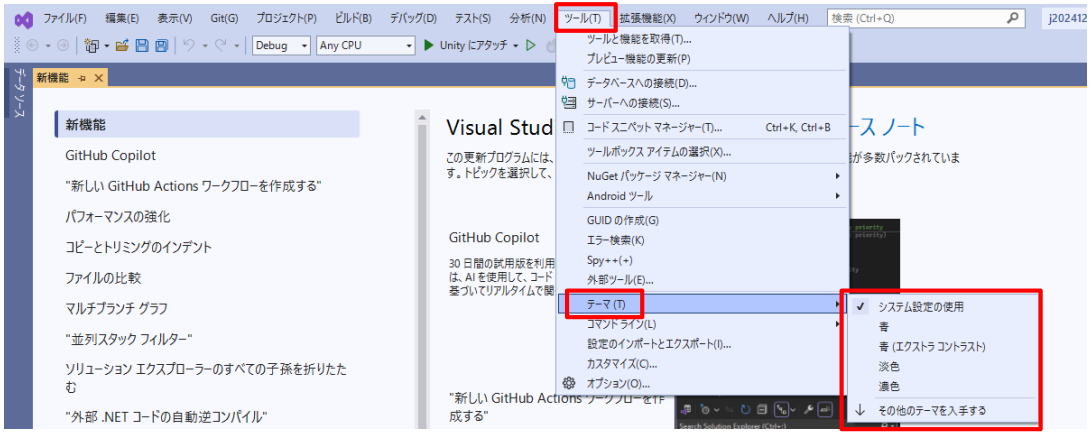
Unity2022はVisualStudio2022を使用します。

アカウントの入力を求められた場合は、今はスキップするをクリックします。



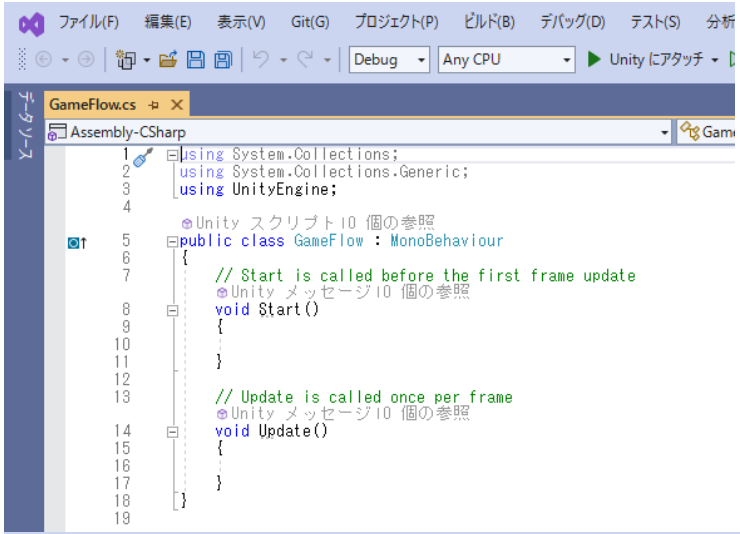


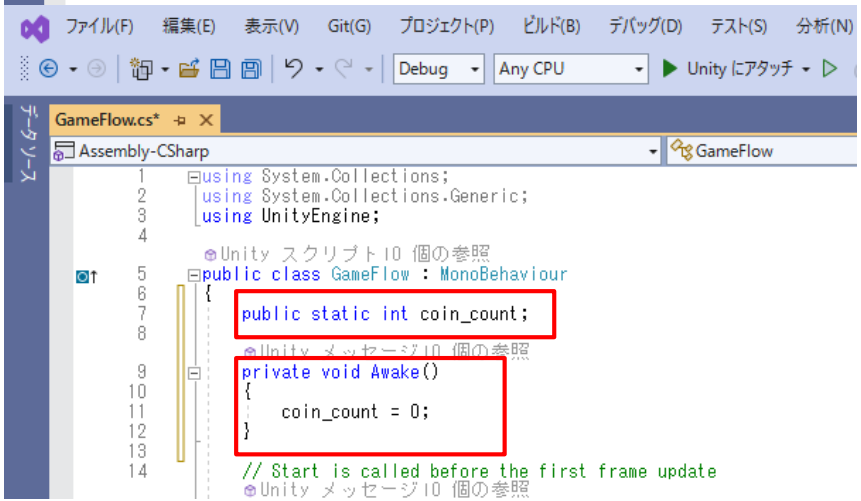




VisualStudioの初回起動の場合はテーマを選択する選択画面が出ます。

もし、後で変更する名合はVisualStudioのメインメニューのツール、テーマと選び、その中の希望のテーマを選ぶと画面の色などのテーマを変えることができます。



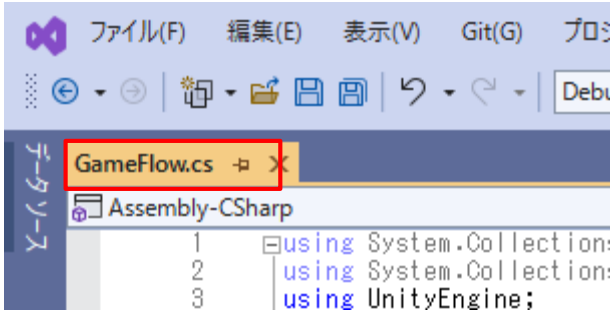


GameFlowを開くと、ベースのスクリプトができています。

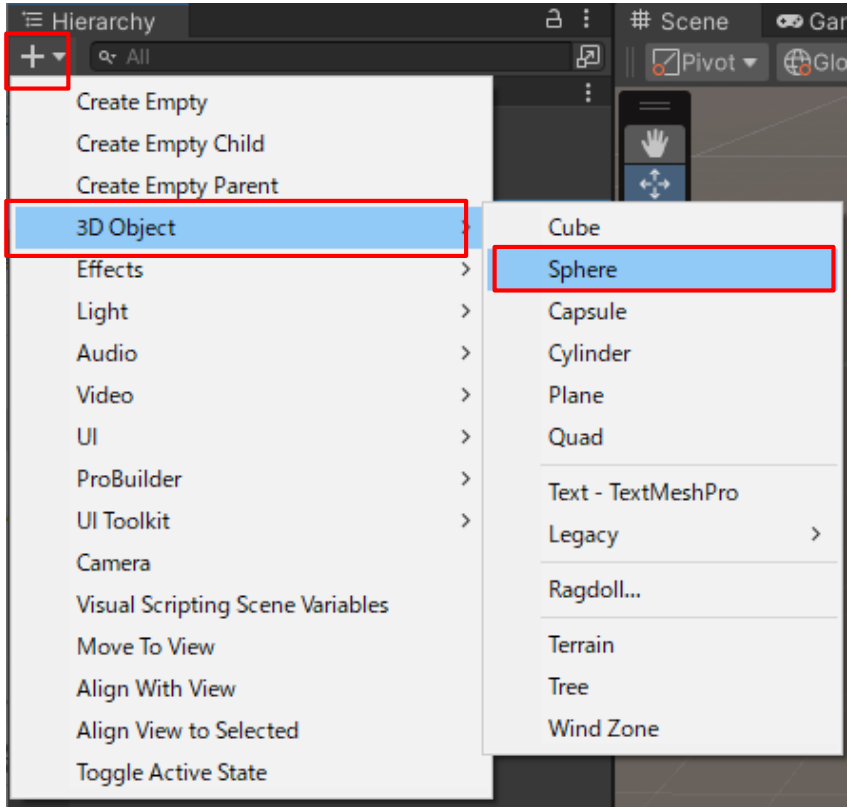
7行目以降を改行などで開けて、赤いブロックの部分を入力します。

入力したら、Ctrl+Sキーで保存してください。

下記のようにGameFlow.csの右についていた＊マークが消えます。消えると保存できたということになります。



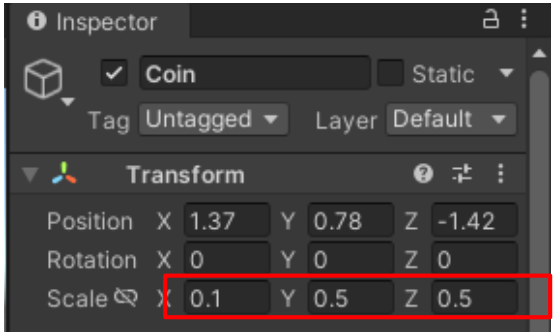
コインの作成



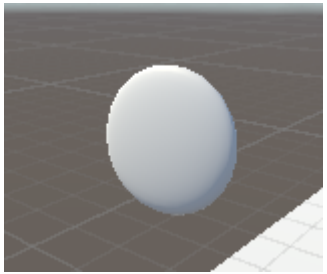
3DオブジェクトのSphere(球体)を変形させてコインのようなものを作ります。

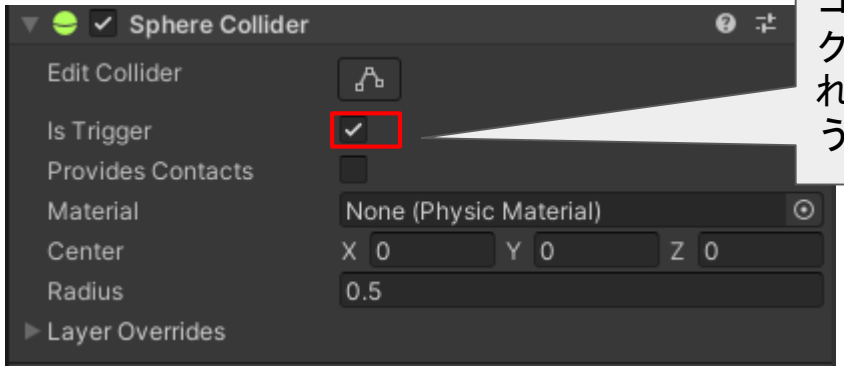
まず、球体を作成します。

作成したら、名前をCoinに変えます。

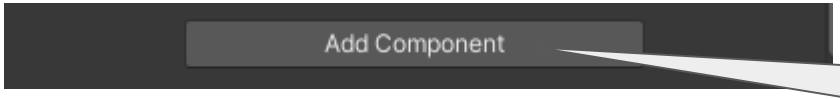


スケールをXを0.1、Yを0.5、Zを0.5にして薄くするとコインのようになります。

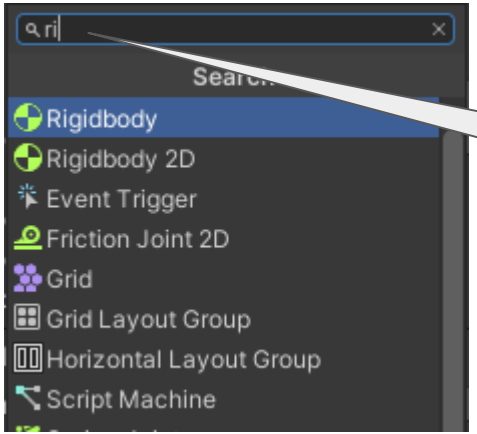




コインのSphereColliderのIs Triggerはチェックを入れます。これにより、スクリプトに入れるOnTriggerEnterの割り込みに反応するようになります。

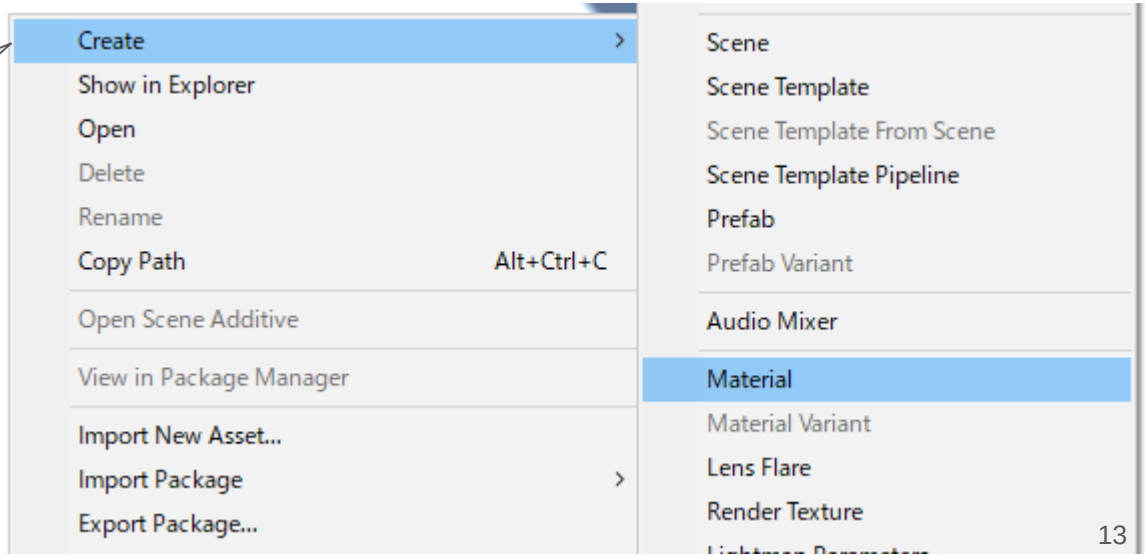


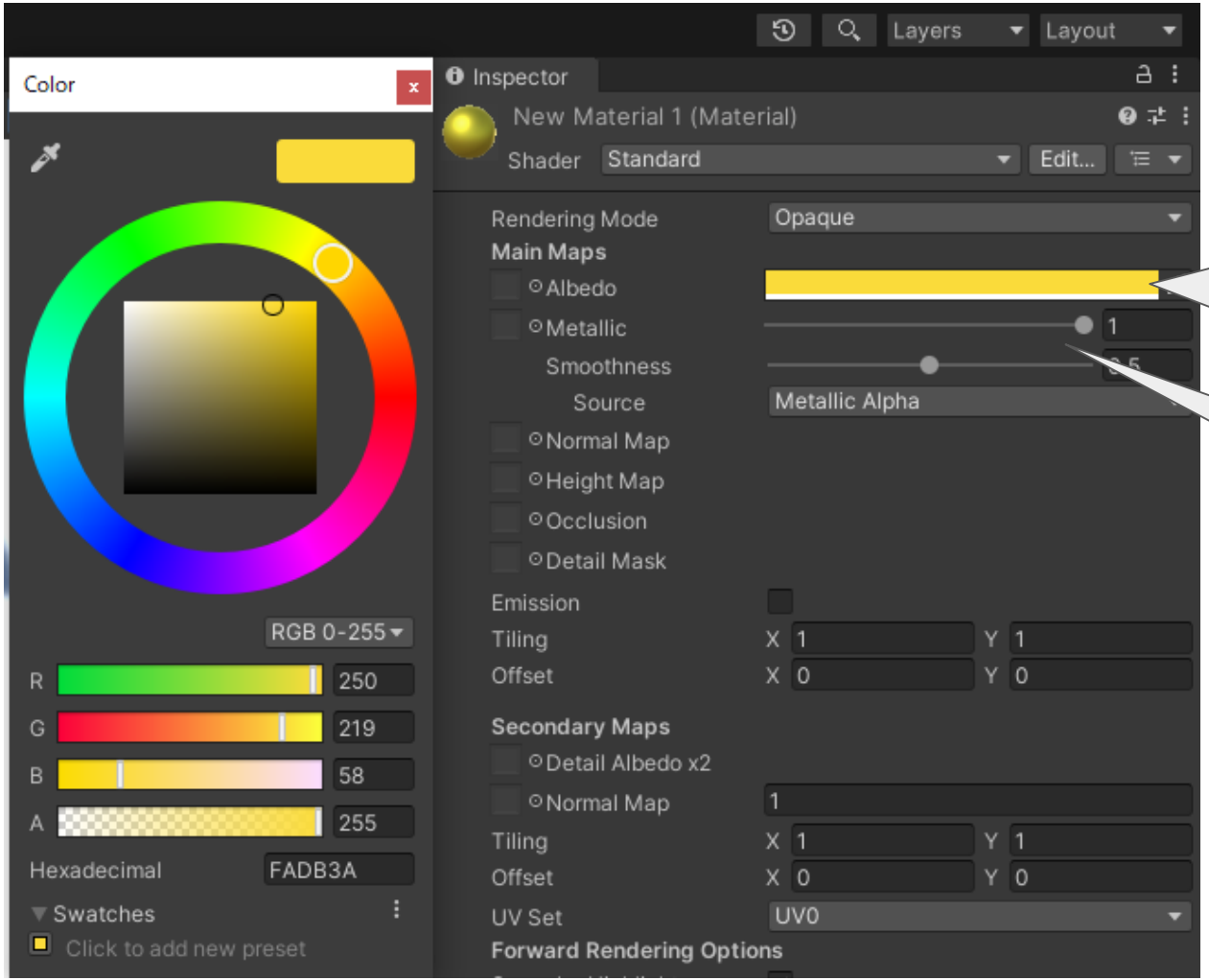
Add Componentをクリックして、
Rigidbodyを入れます。



検索で ri と入れるとRigidbodyを素早く出すことが出来ます。

コインの金色をだすためのマテリアルを作成します。
Assetsの中で右クリックしてメニューをだし、Create、Materialをクリックいます。

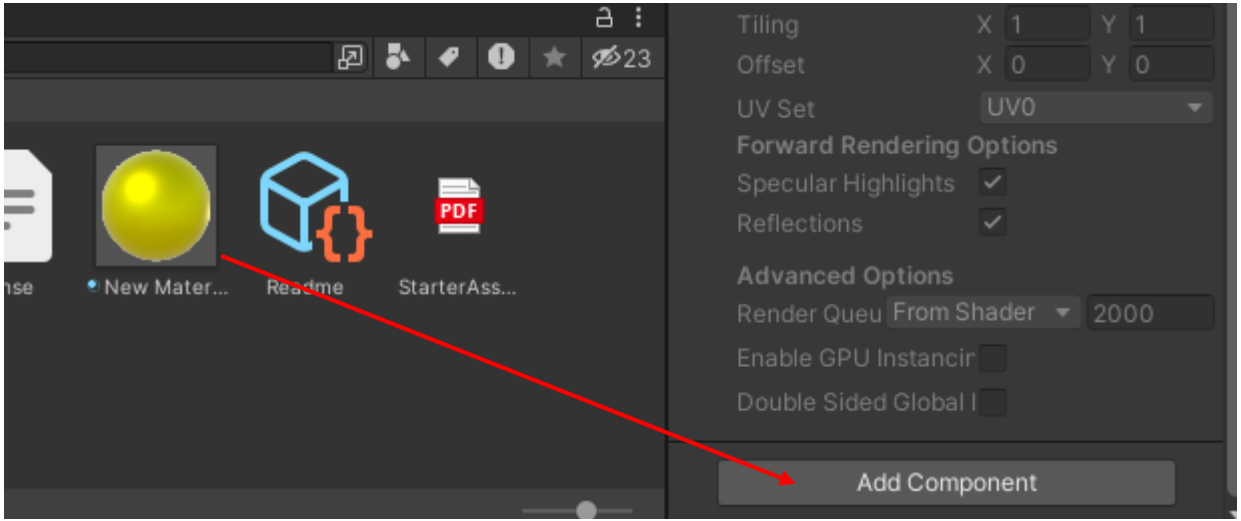


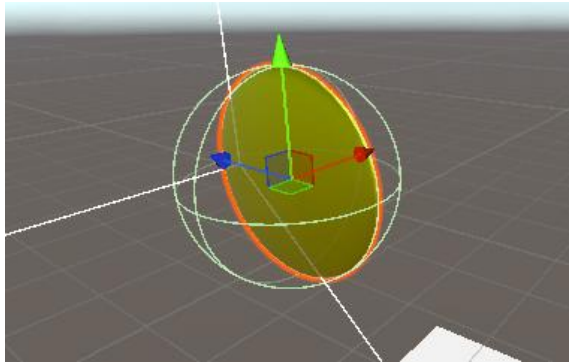


カラーバーをクリックすると色を変更することが出来ます。

Metallicのレベルを上げると金属感を出すことが出来ます。

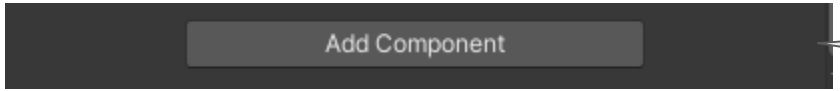
完成したマテリアルをCoinにドラッグ＆ドロップするとマテリアルを設定することが出来ます。



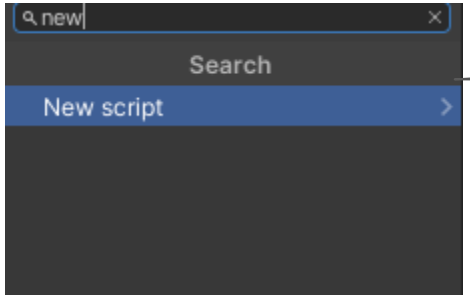


作成したマテリアルをAddComponentにドラッグ＆ドロップします。白色のコインが金色になります。

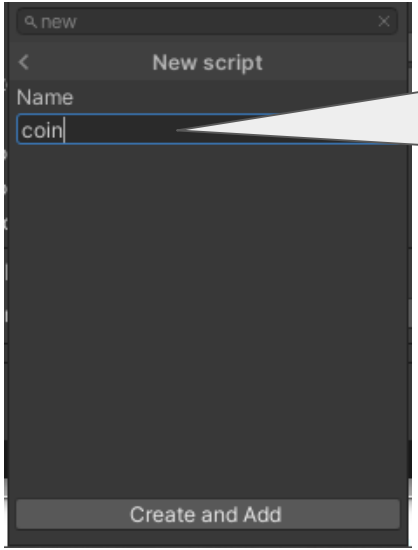
スクリプトの組み込み



ヒエラルキーにのCoinを選択してから、インスペクターのAddComponentをクリックします。



検索でnew と入力するとnew scriptがでます。



nameで coin 入力して 下のCreate and Add をクリックします。
coinにC#スクリプトのベースが組み込まれます。

コインのスクリプトを入力します。（coin.csファイルのコーディング）

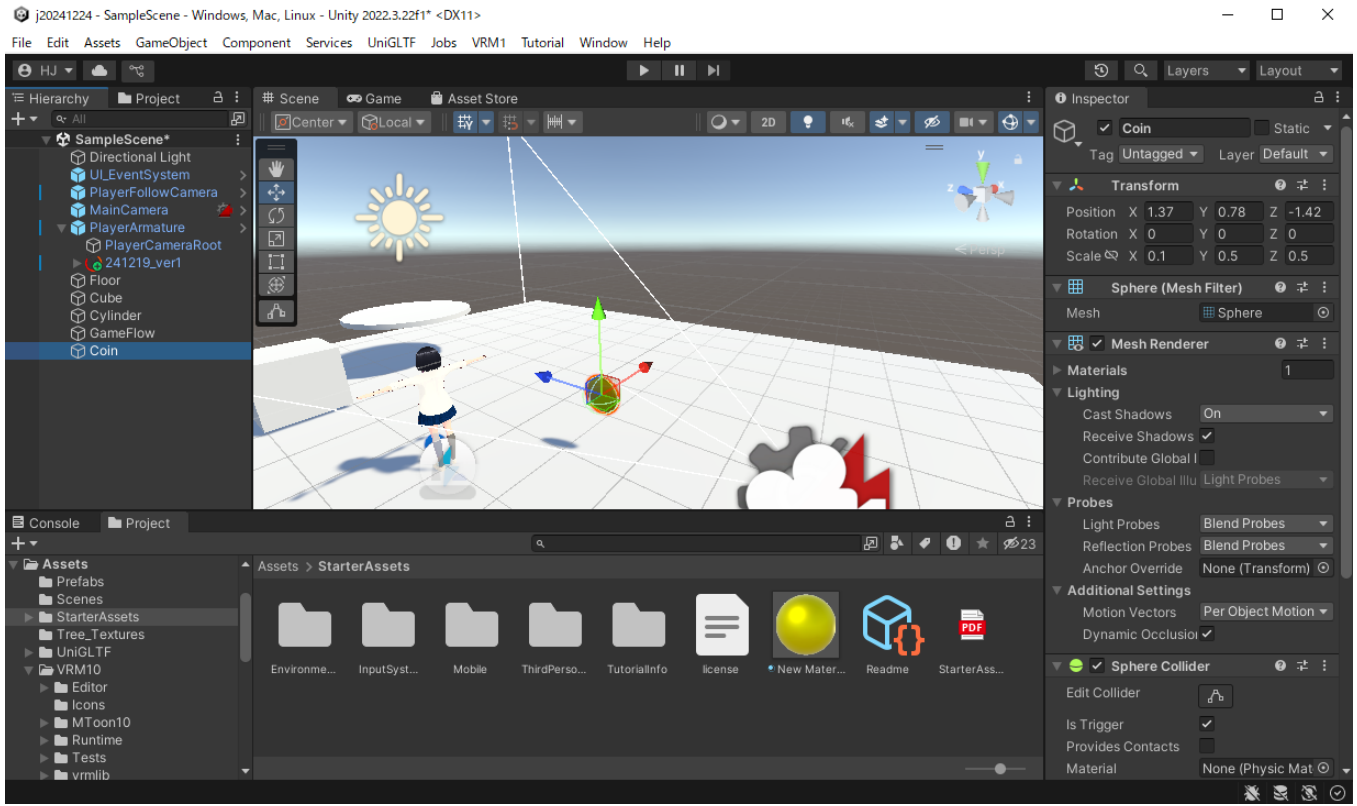
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class coin : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        GameFlow.coin_count = GameFlow.coin_count + 1;
    }

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

    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            GameFlow.coin_count = GameFlow.coin_count - 1;
            Destroy(gameObject);
        }
    }
}
```

コインを一枚設置するごとに配置された
コインの枚数をカウントアップします。

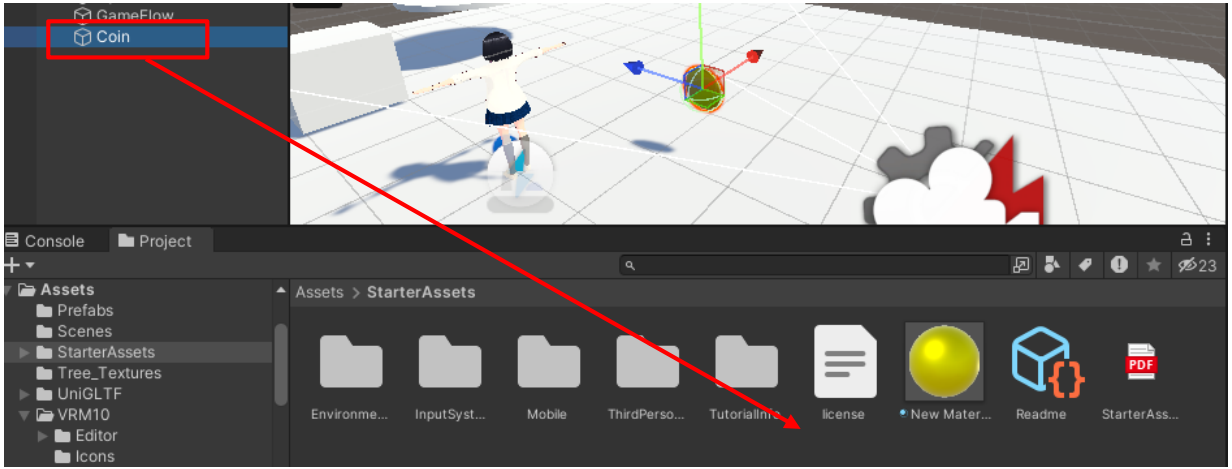
プレイヤーがコインに触れた時に
設置されているコインの枚数を減らし、
触れたコインをマップ上から消去します。

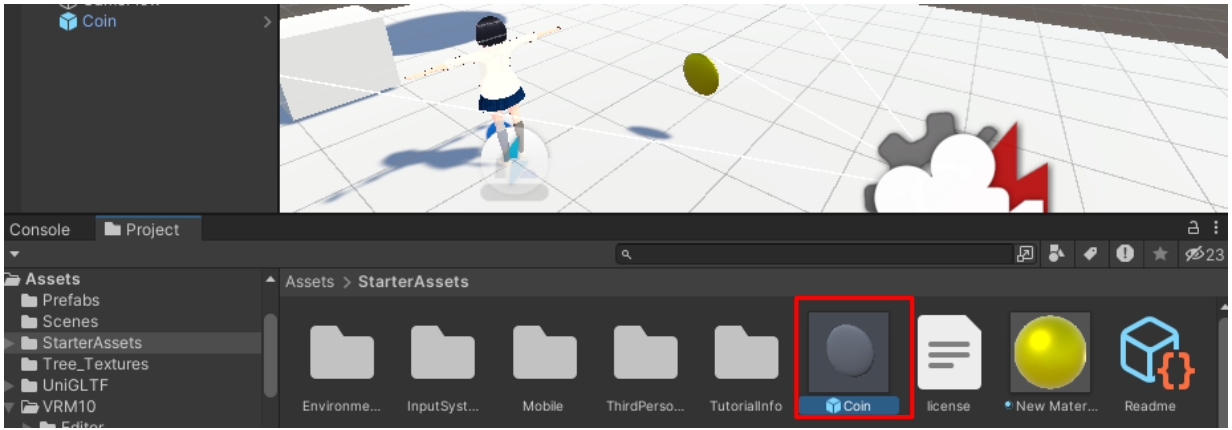
スクリプトが完成したら、ゲームを実行してコインを取ってみましょう。

コインが消えたら処理の完成です。

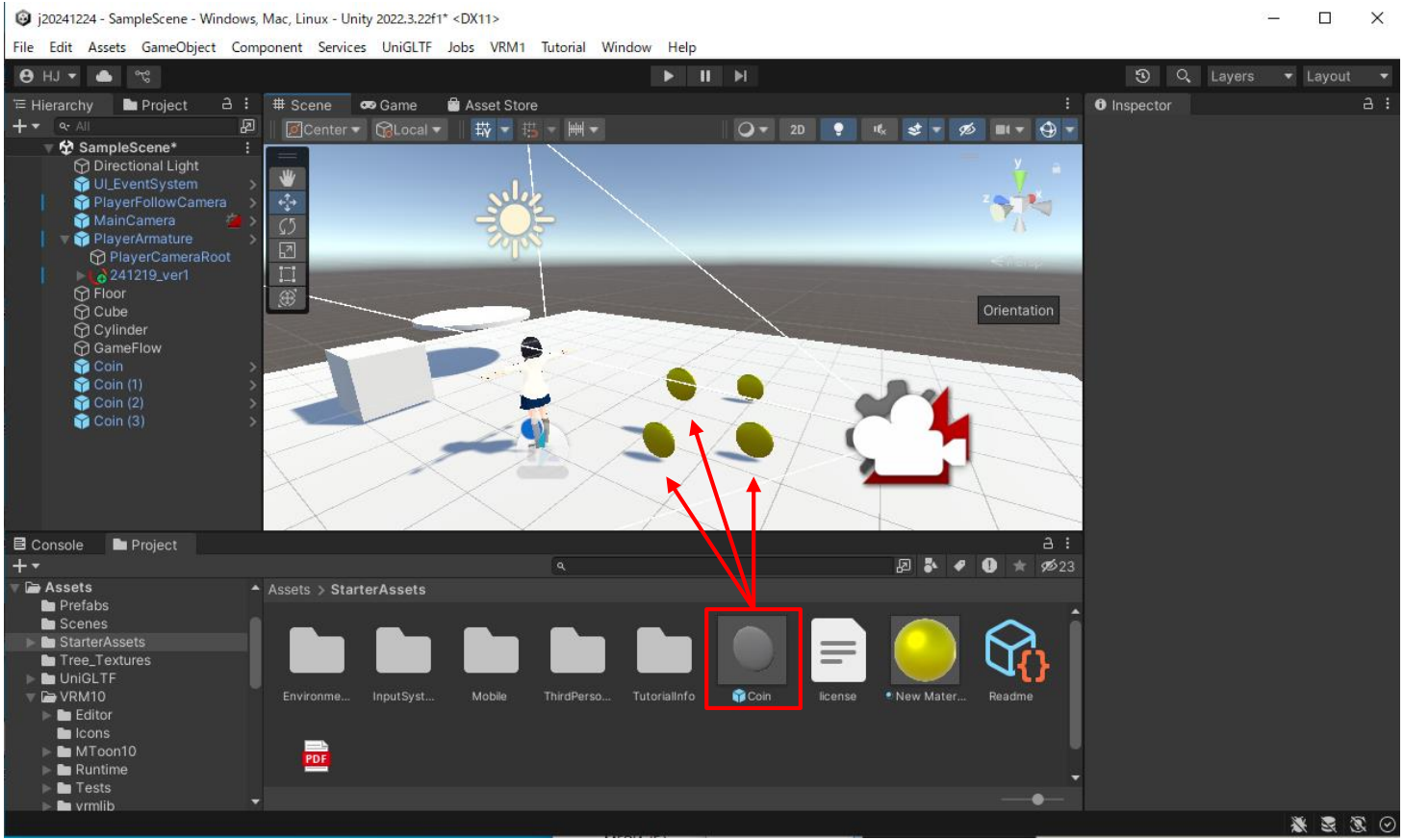


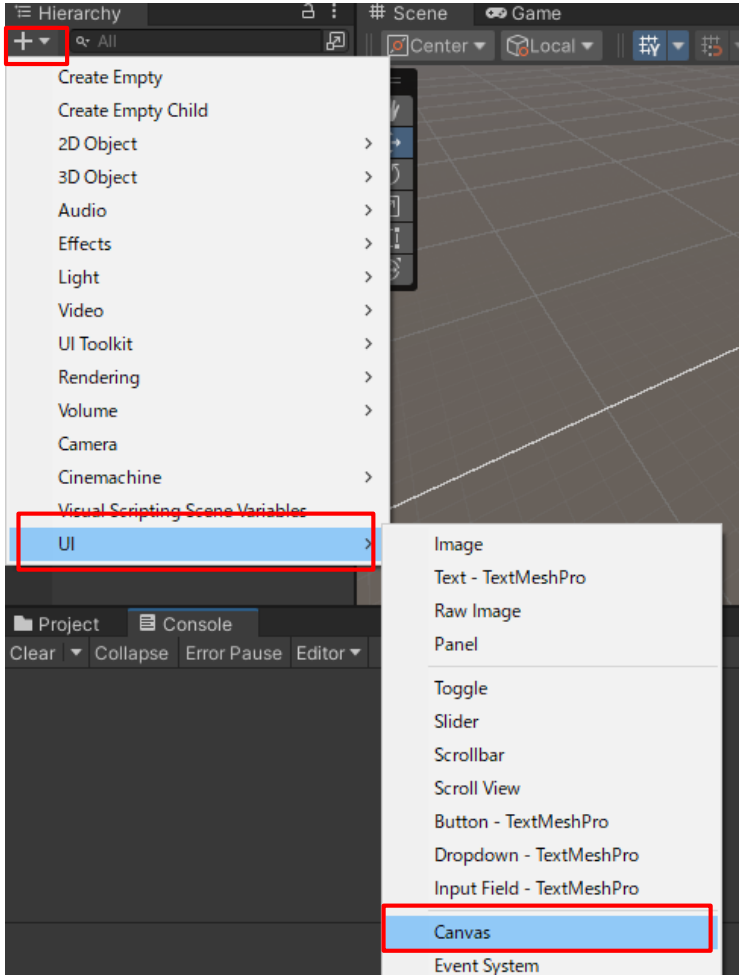
完成したコインをAssetsフォルダに入れてプレハブ化します。



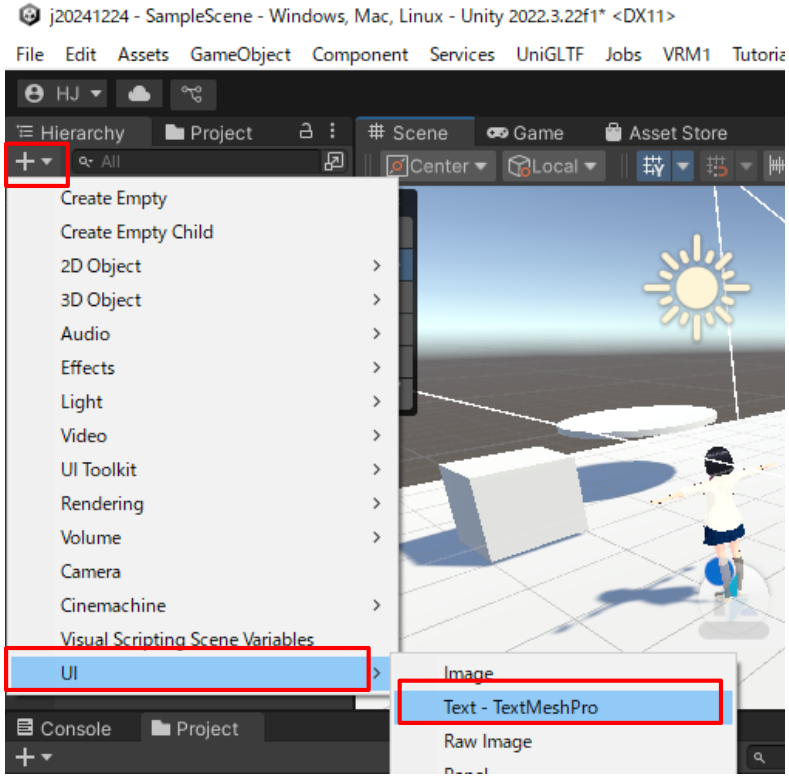


プレハブ化したものはいくらでもステージにドラッグ＆ドロップして置いて複製できます。



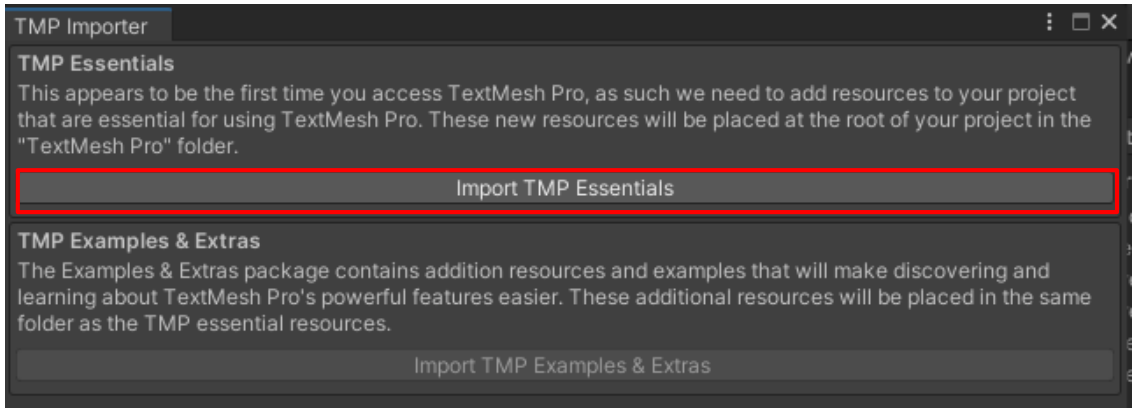


すべてコインを取ったらCLEARを表示するように変更します。文字表示のためのCanvasを作成します。＋をクリックして、UI、Canvasとクリックして、Canvasを作成します。

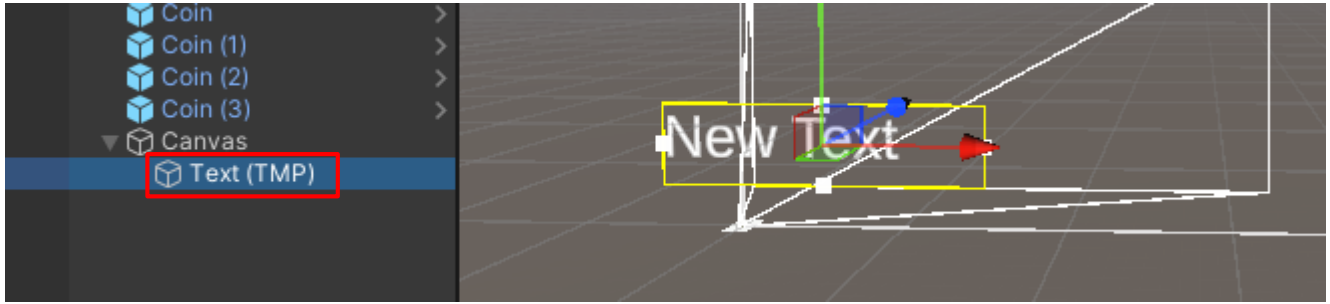


テキスト表示のためのテキストボックスを作成します。

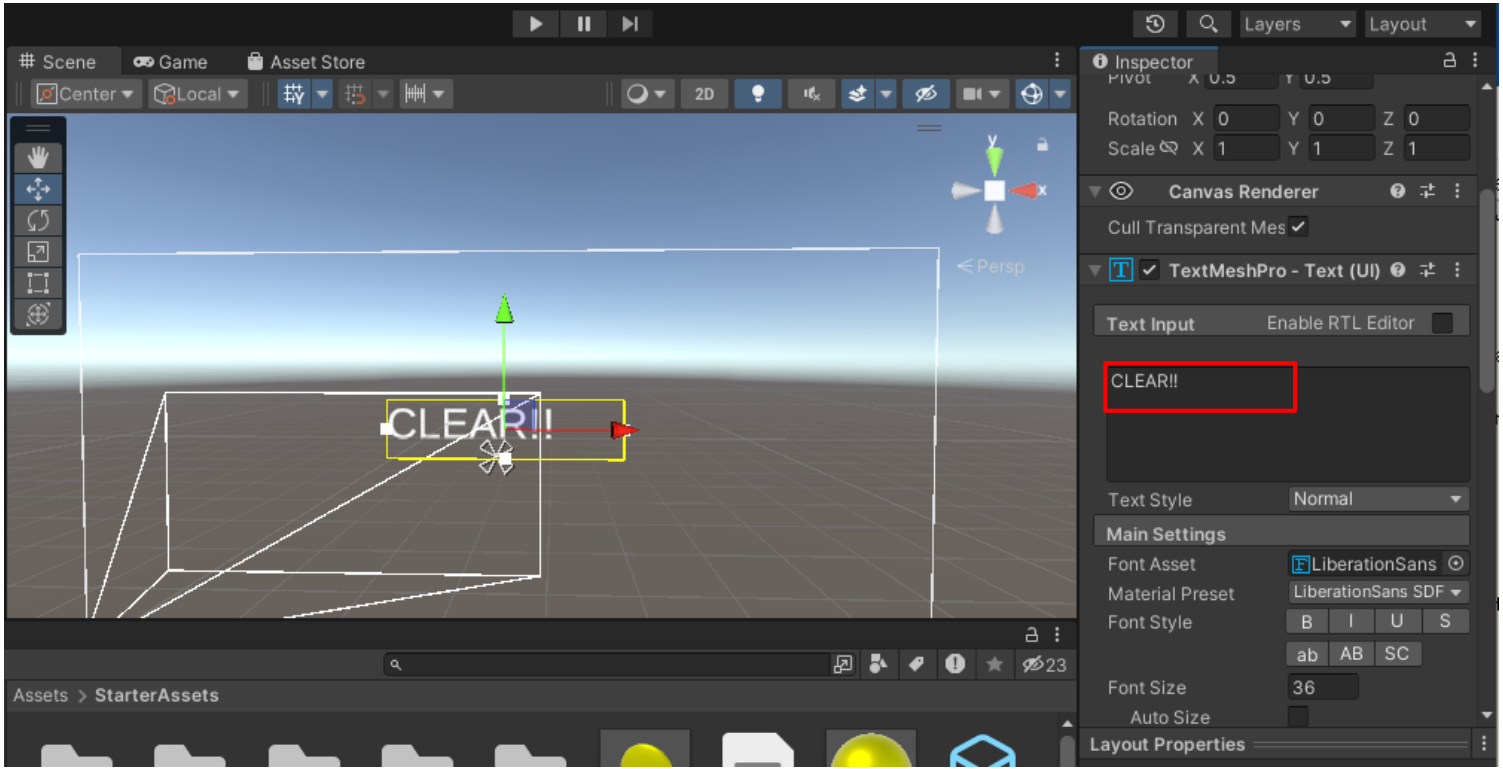
＋をクリックして、UI、Text-TextMeshProをクリックします。



最初にTextMeshProを入れた場合はこのようなダイアログが出ます。
Import TMP Essentialsをクリックします。



Text(TMP)はclearに変更します。
クリックすると名前変更ができます。



表示テキストを変更します。

GameFlowスクリプトの追加・変更箇所を行います。

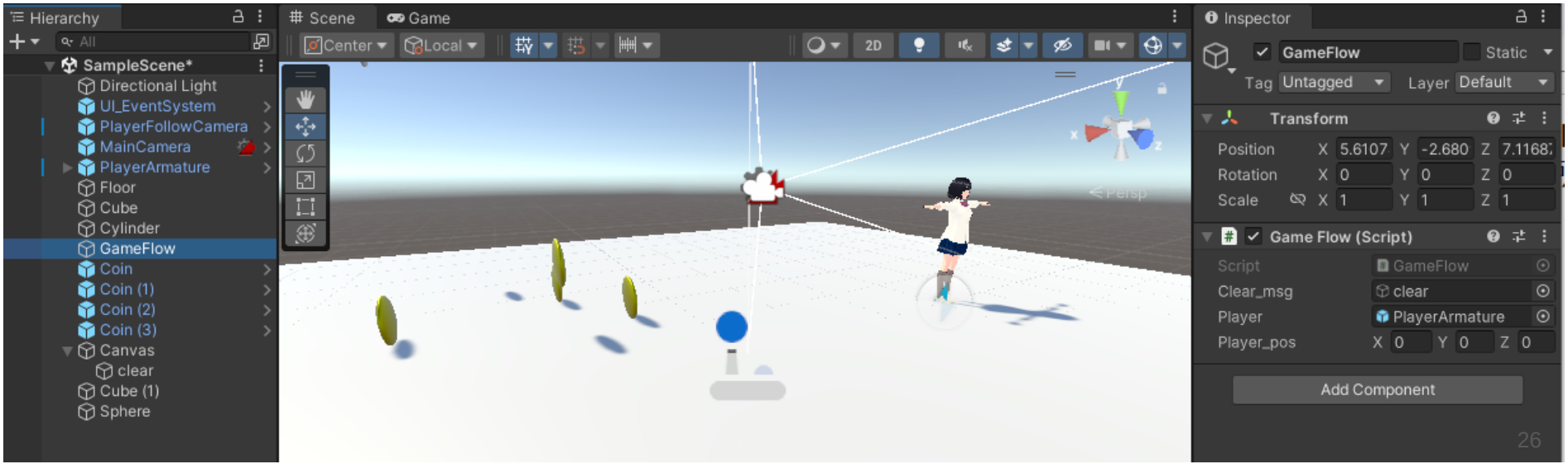
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameFlow : MonoBehaviour
{
    public static int coin_count;
    public GameObject clear_msg;

    private void Awake()
    {
        coin_count = 0;
        clear_msg.SetActive(false);
    }

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (coin_count<=0)
        {
            clear_msg.SetActive(true);
        }
    }
}
```

コイン枚数は0とします。
（配置枚数はcoin.csで数えます）
クリアメッセージは最初表示しないようにします。

配置されたコインの枚数が0になった時、
クリアメッセージを表示します。

GameFlowをクリックして、InspectorのClear_msgの欄にCanvasのclearを
ドラッグ＆ドロップしてリンクします。

すべてのコインを取ってクリアメッセージが出るか確認しましょう。



プレイヤーがステージから落下した場合の復帰処理をつくる

1.ゲーム開始時にプレイヤーの初期座標値を別の変数に保存しておく。

2.ゲーム中、プレイヤーのY座標が-20(あまり、下の座標になったら落ちたと判定する)になったら、プレイヤーの初期座標値をプレイヤーに代入する。

プレイヤーが落下した時の処理を作成する。

GameFlow.csのソースコードの赤い部分を追加変更します。

```
public class GameFlow : MonoBehaviour
{
    public static int coin_count;
    public GameObject clear_msg;
    public GameObject player;
    public Vector3 player_pos;

    public bool fall_flg;

    private void Awake()
    {
        coin_count = 0;
        clear_msg.SetActive(false);
        player_pos = player.transform.position;
        fall_flg = false;
    }

    // Start is called before the first frame update
    void Start()
    {

    }
```

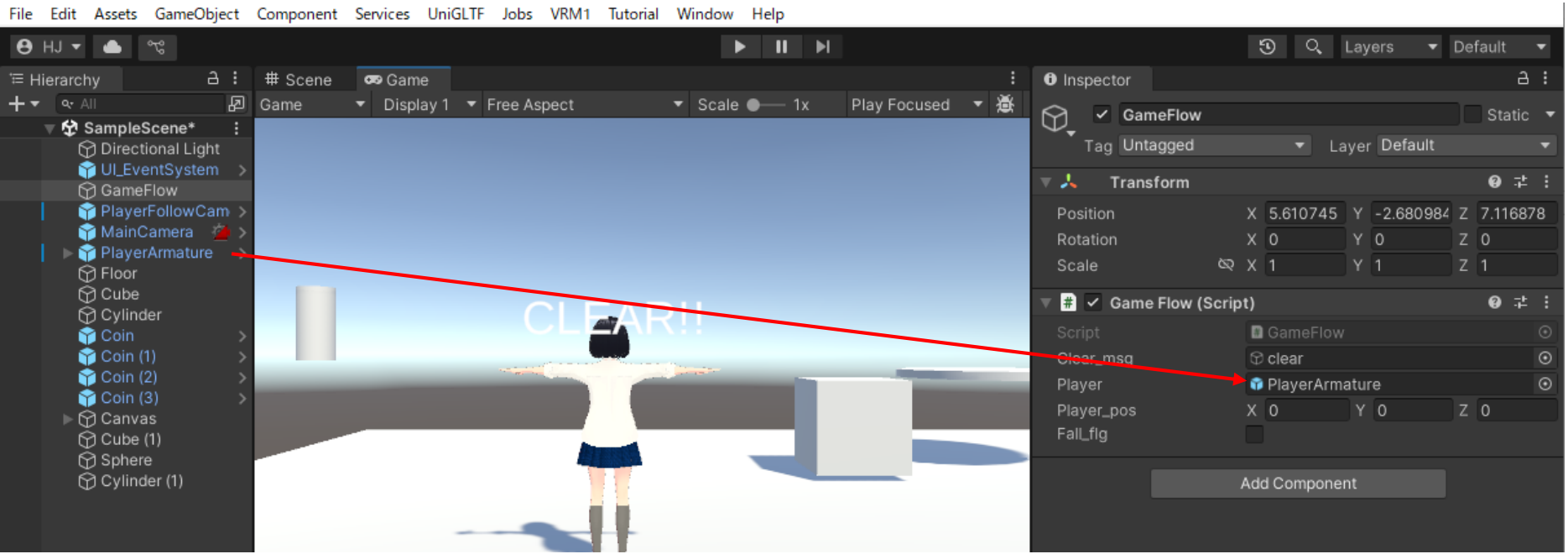
プレイヤーの初期位置を記録しておきます。
落下判定フラグはfalse（偽）とします。

変更のつづき

```
void Update()
    {
        if (coin_count <= 0)
        {
            clear_msg.SetActive(true);
        }
        if (player.transform.position.y < -20 && fall_flg == false)
        {
            Debug.Log("落ちた！！");
            player.SetActive(false);
            fall_flg = true;
            //player.transform.position = new Vector3(0, 1, 0);
        }
        else if (fall_flg == true)
        {
            player.transform.position = player_pos;
            player.SetActive(true);
            fall_flg = false;
        }
    }
```

プレイヤーの位置が高さ(y)-20を下回った場合
落下判定をtrue（真）とします。
一時的にプレイヤー操作を受け付けないようにします。

その後、初期位置に戻す処理を行い、
落下判定を再びfalse（偽）とします。
その後、プレイヤー操作を受け付けるようにします。



GameFlowのソースコードを新しい部分と、プレイヤーを繋げます。

PlayerArmatureを線の図のようにドラッグ＆ドロップして入れます。

これでステージから落下するとスタート地点に戻ることができます。