# Unityで簡単なFPS

# １．準備

http://japan.unity3d.com/unity/download/

で、Unity4.3.4をダウンロード（無料）。
ダウンロードしたら、自分のわかりやすいところに好きな名前で保存する。（容量に空きがあるところがいい）
１０分ぐらい時間がかかる・・・
ダウンロードできたらインストールする。
ライセンスをとってない人はライセンスもとってください。

# Unityの紹介

Unityとは

「統合開発環境を内蔵し、複数のプラットホームに対応するゲームエンジン」

のこと。

iOS 、Android、Windows、Mac OS X、Web、Wii U、PlayStation3、Xbox 360 など

様々なプラットフォームへ向けた高度な 3D アプリケーションを制作することができます。

公式HP　：　http://japan.unity3d.com/

日本のゲームでUnityを使用しているゲームの紹介。

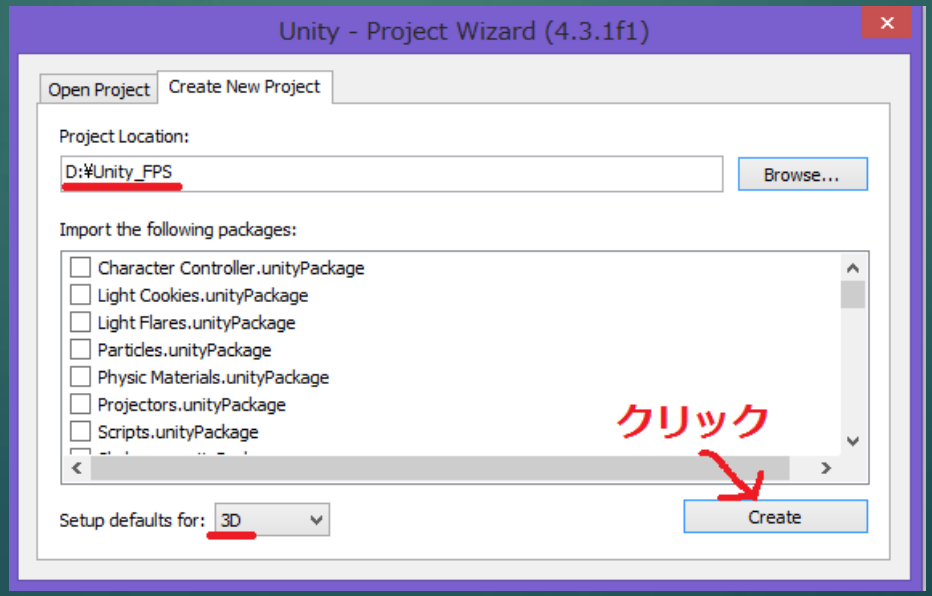
http://www.youtube.com/watch?v=k2PweddhqEc

他にも以下のページでいろいろ紹介されています。

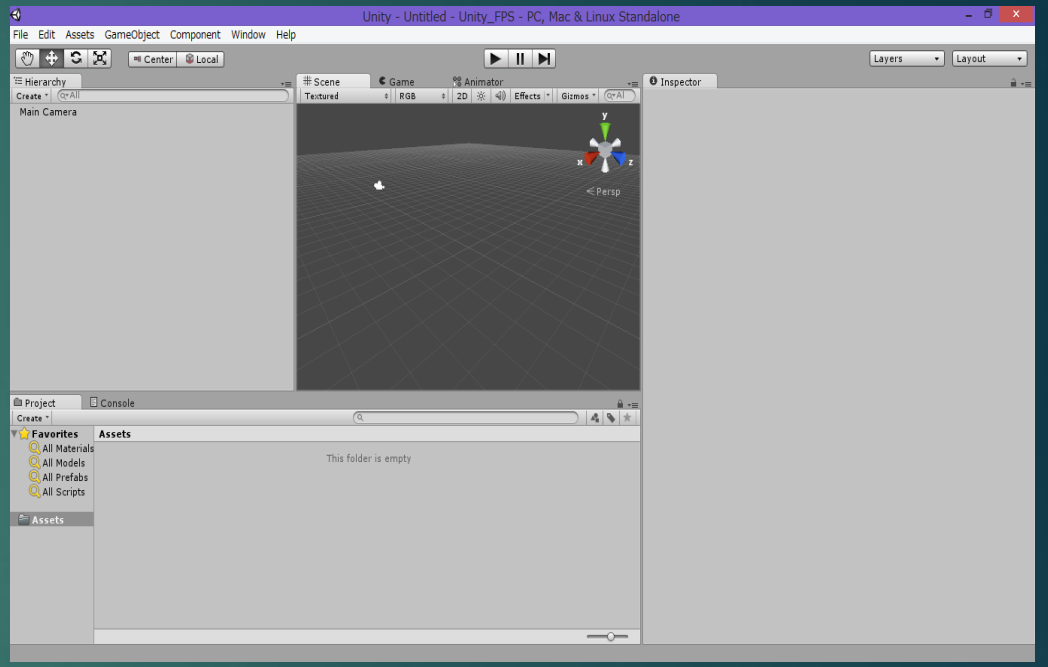
http://japan.unity3d.com/gallery/made-with-unity/game-list

# ２．起動してみる

起動したら、メニューの「File」→「New　Project」を選択する。
選択したら、下のような画面が出てくるので自分の好きな所にひと
まず、「Unity_FPS」というプロジェクトを作る。３Dとなってる箇所は変え
ずに。できたらCreateを押す。



# ３．画面構成




Createを押したら、初期起動画面が出てくる。

各ビューの説明

Scene（パーツを配置するための画面）

Game（実行状態を確認する画面）

Hierarchy（オブジェクトの階層構造を管理）

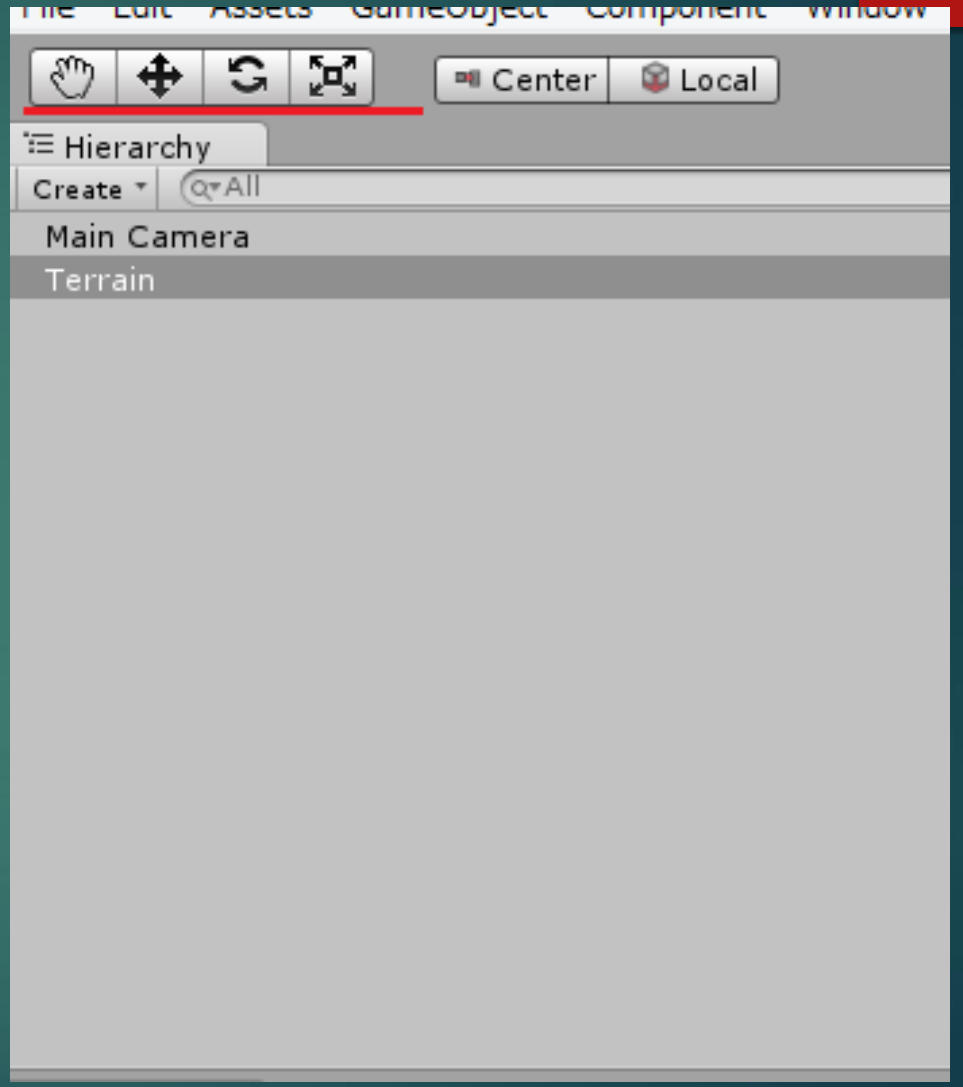
Project（オブジェクトの機能を管理）

Inspector（各オブジェクトの詳細情報を管理）

少なくともこの５つは出しとく。

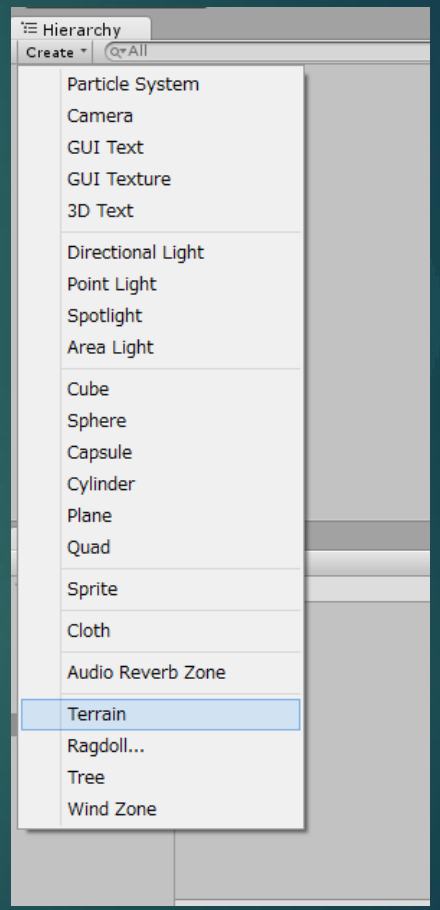
各ビューは好きなところに配置できるので自分にあった置き方をしてよい。

Hierarchyの上にある４つのボタンで
Scene上をいろいろいじれる。
左から、
並行移動、ｘｙｚ軸方向への移動、回転、
拡大・縮小
となってる。
また、ホイールで画面の拡大・縮小が
できる。
右ボタン押しっぱなしで視点変更。

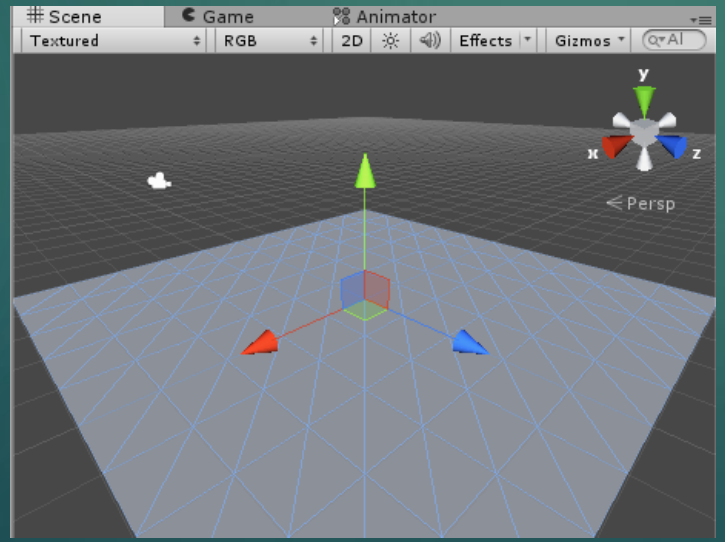


# 4　地面を作る

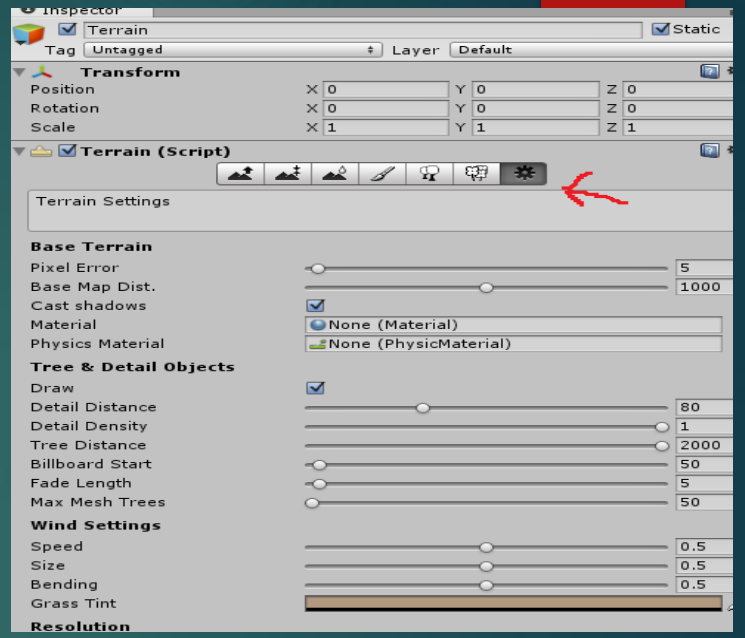
「Hierarchy」→「Create」→「Terrain」を選択。

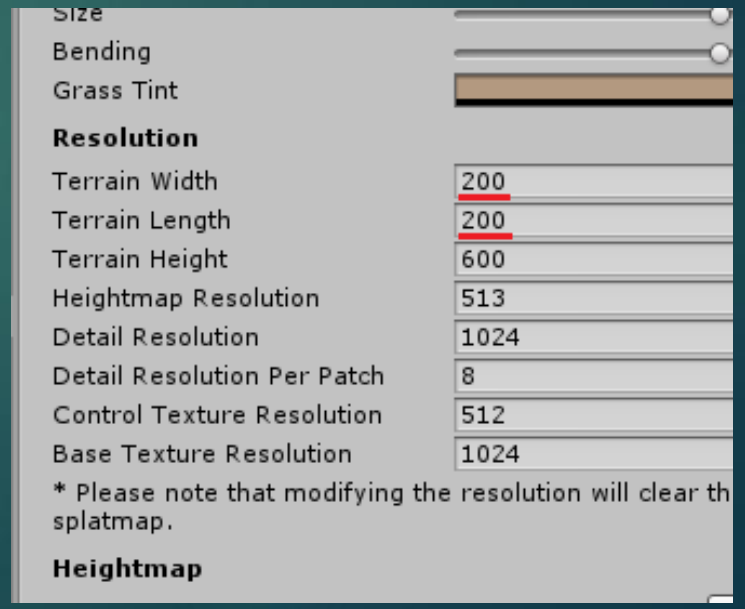



押すとSceneビューに正方形が作成される。
これで元の地面が完成！

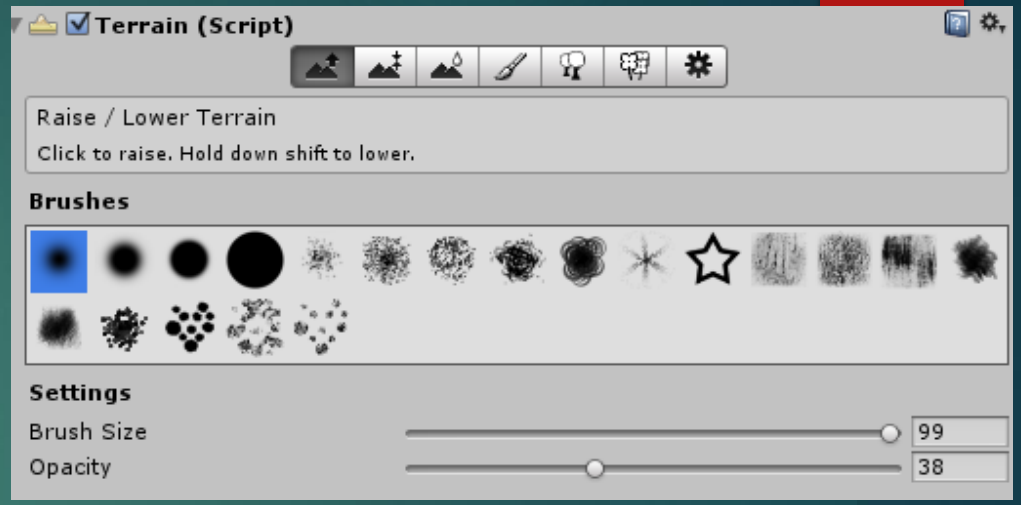


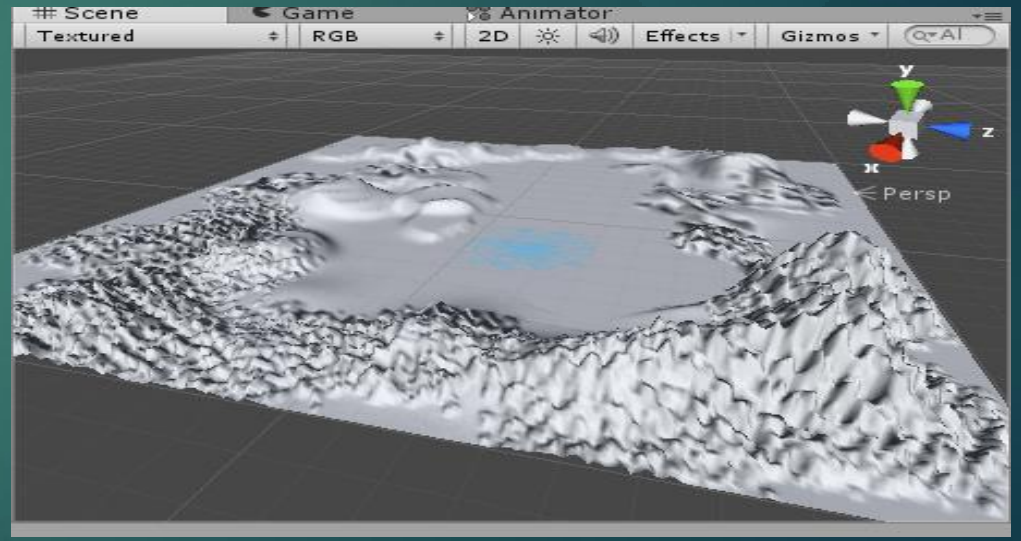
次に地面の大きさの変え方について。
TerrainのInspectorのTerrain（Script)のところの
一番右のマークを押すと、このTerrainの詳細を
見ることができる。
Resolutionという欄で大きさなどを決められる。
Terrain Width（幅）、Terrain Length（長さ）が2000と
初期状態ではなってるが、これは現実世界に
換算すると、２ｋｍと同じらしい・・・。
そんなに大きくなくていいので、どちらも200に
しとく。



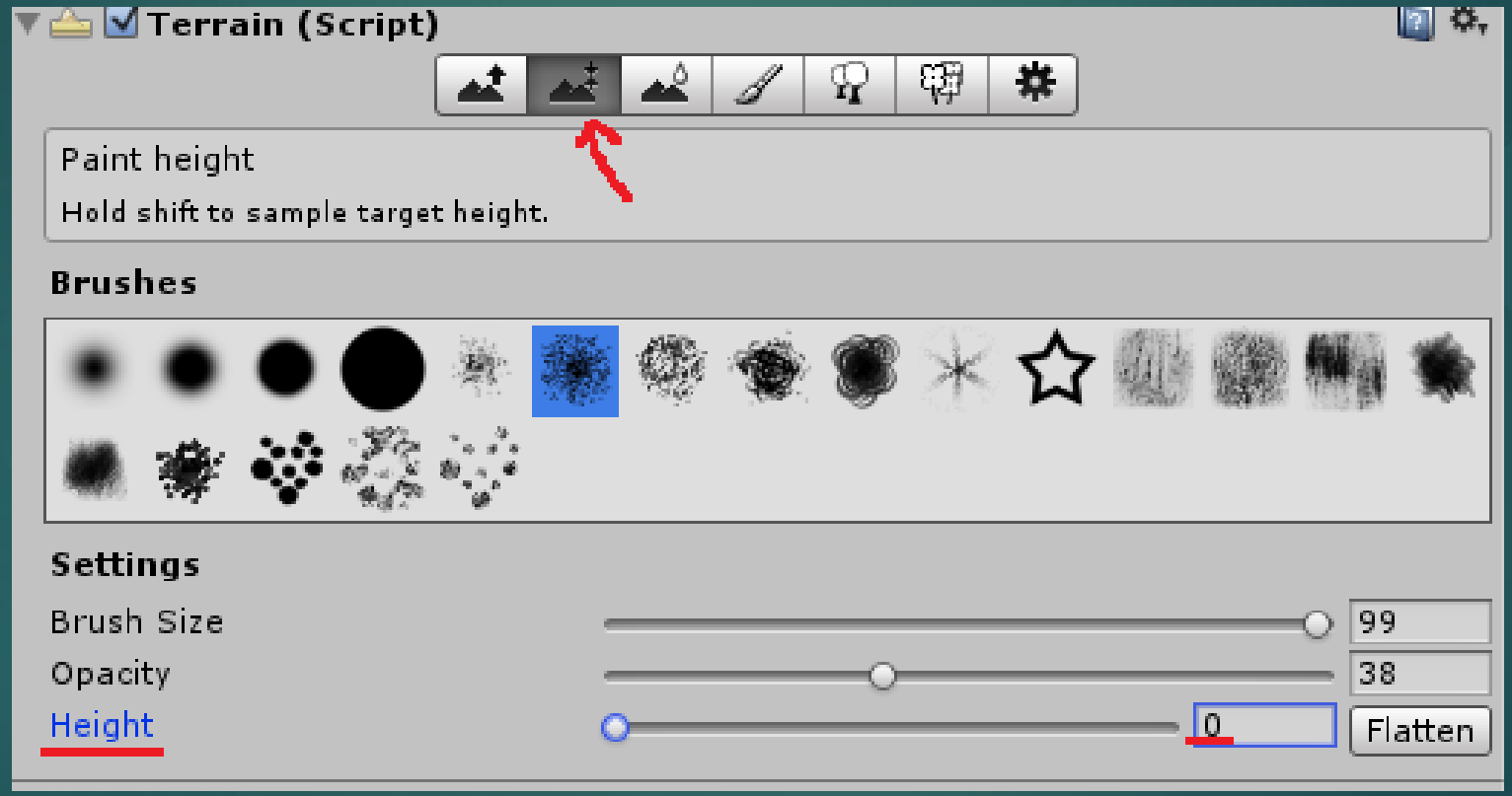



大きさが変更できたので、次に山の作り方。
さっきと同じところの今度は一番左の
マークを選択。
そしてScene画面上で左クリックや
左ボタンを長押しすると地面が変形
する！
Brushesをいろいろ変えることによって
表面が変わると思うのでいろいろ試し
てみる。



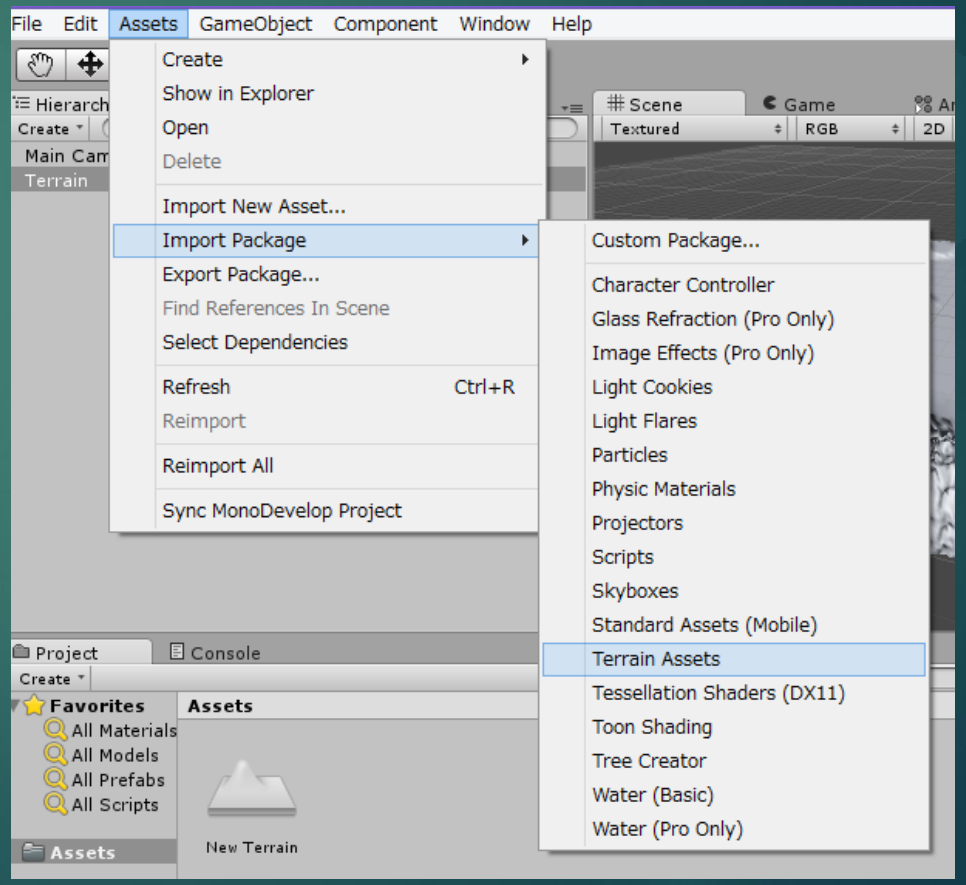



山を削る場合は左から２番目を選んで、Heightの値を０にして
さっきと同じようにScene上で長押ししたりするとへこむ。

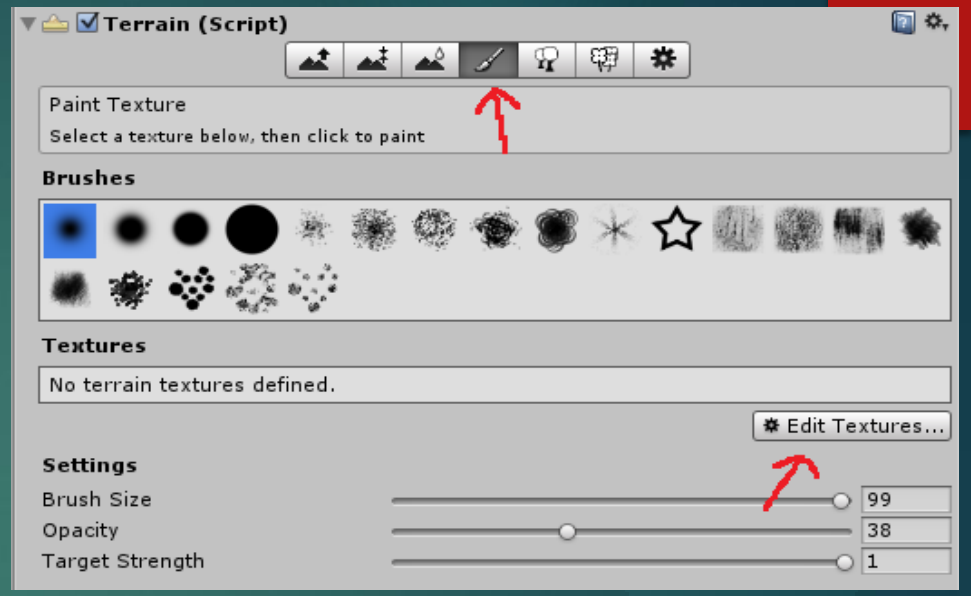


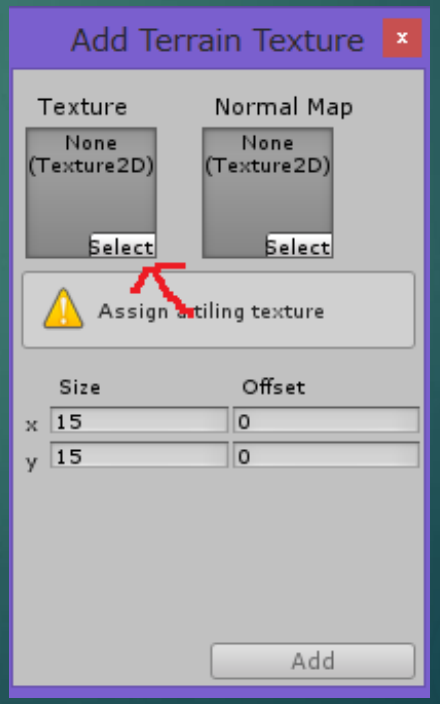
次に地面っぽくするためにテクスチャを貼る。
メニューの「Assets」→「Import Package」→「Terrain Assets」を選択。

その後、Importする内容が見れる画面が出てくるので何も変えずに
Importボタンをクリック。

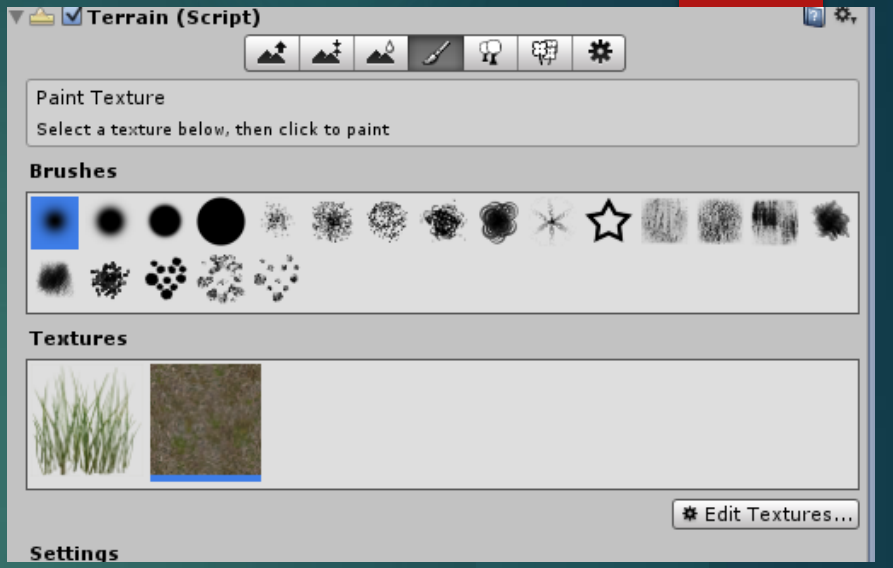


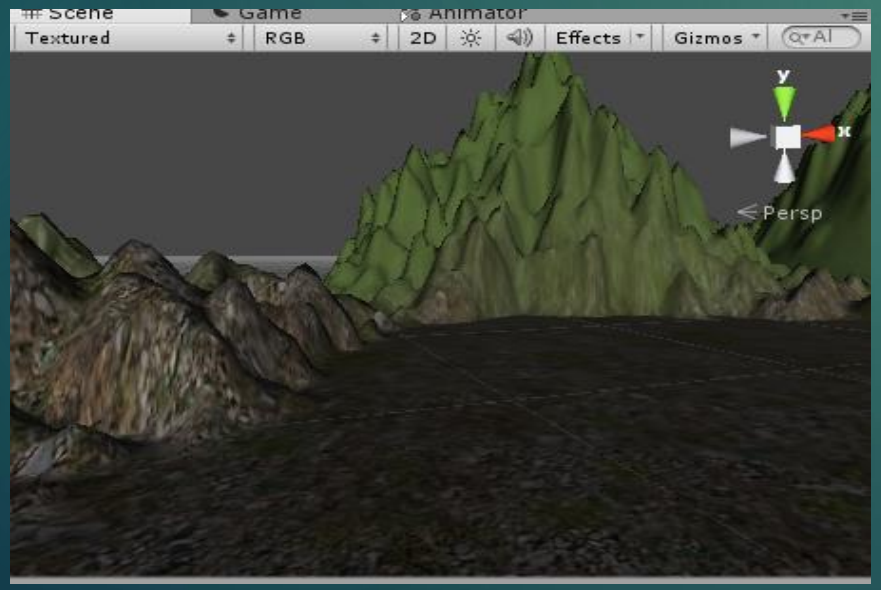
Importが終わったら、さっきのTerrainの
Terrain（Script)のところに戻る。
今度は真ん中を選択。
そしてEdit Textures→Add Texturesと選ぶ。
Add Terrain Textureというウィンドウが出る
ので、左のSelectをクリック。
そして好きなテクスチャを選択。
選んだらAddというボタンをクリック。
これで地面の表面が選んだテクスチャ
に変わる。

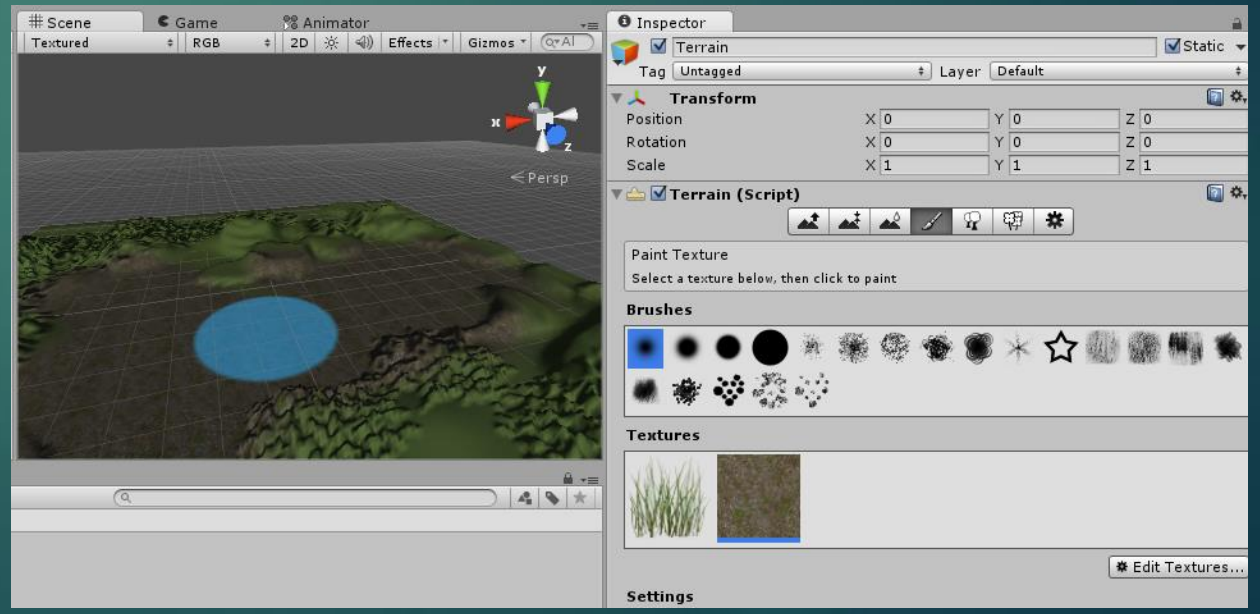



他のテクスチャを使う場合もさっきの作業を
繰り返せば大丈夫。Add Texturesをすることで
使用できるテクスチャが増える。
そして塗りたいテクスチャを選び、Scene上で
塗りたいところでクリックするとその部分が
選んだテクスチャになる。



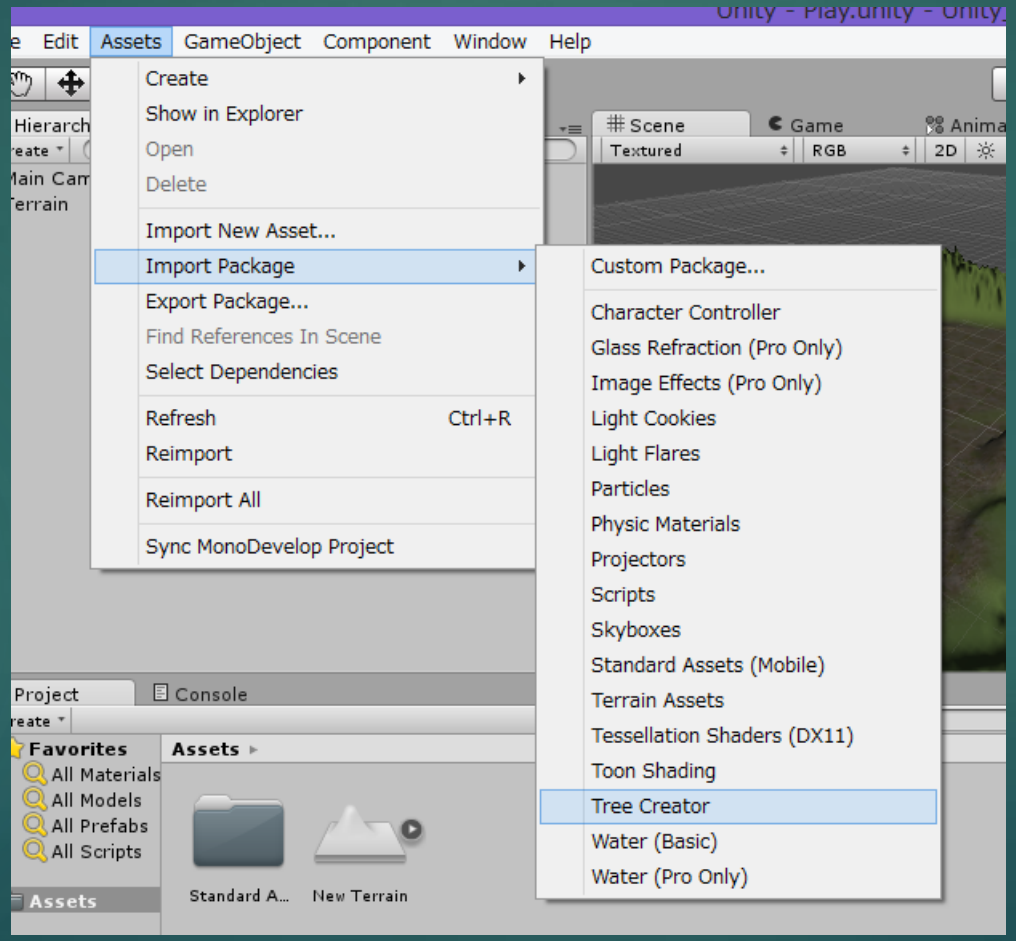



次に木を生やす。

メニューの「Assets」→「Import Package」→「Tree Creator」を選んで

何も変えずにImportする。



木を加えたいときも、テクスチャを貼るときと要領は同じ！

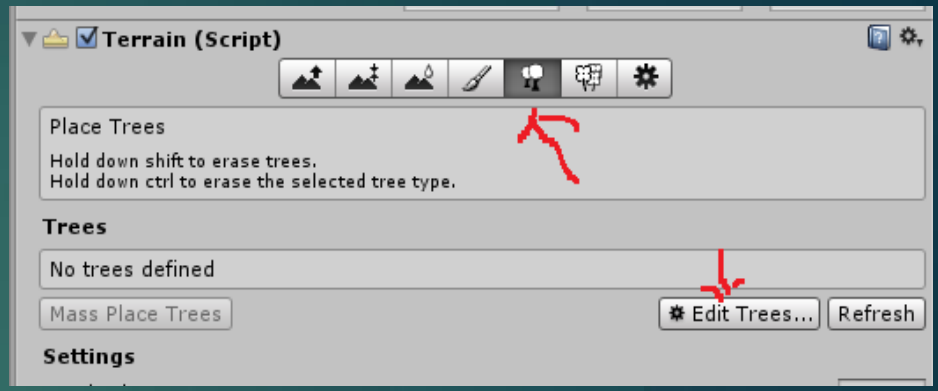
Terrain（Script)の右から３番目を選ぶ。

そしてEdit Trees→Add Treeをクリック。

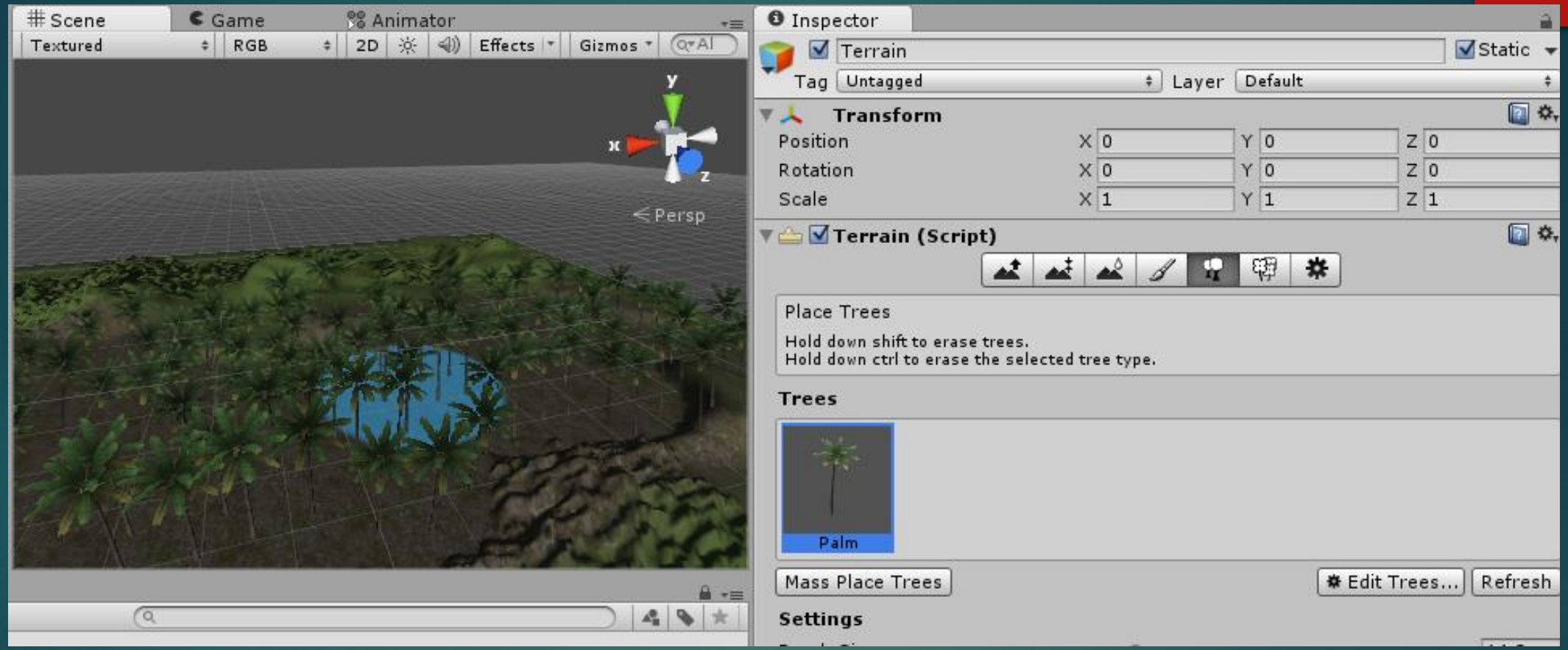
Add Treeというウィンドウが出るので、図で示して
いる小さい丸をクリック。

木を選べるので、ひとまず好きな木を選ぶ。

そしてAddを押す。



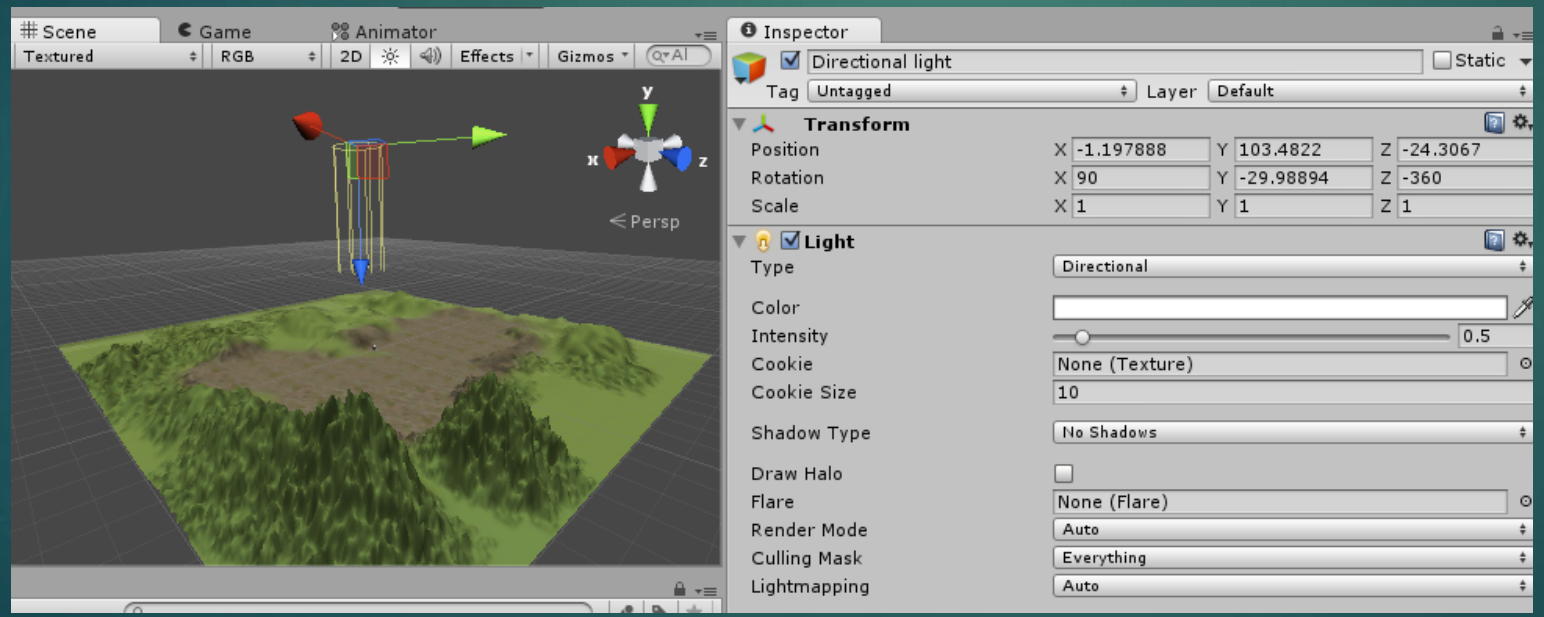

そしてScene画面でクリックすると青丸の部分に木が生える！

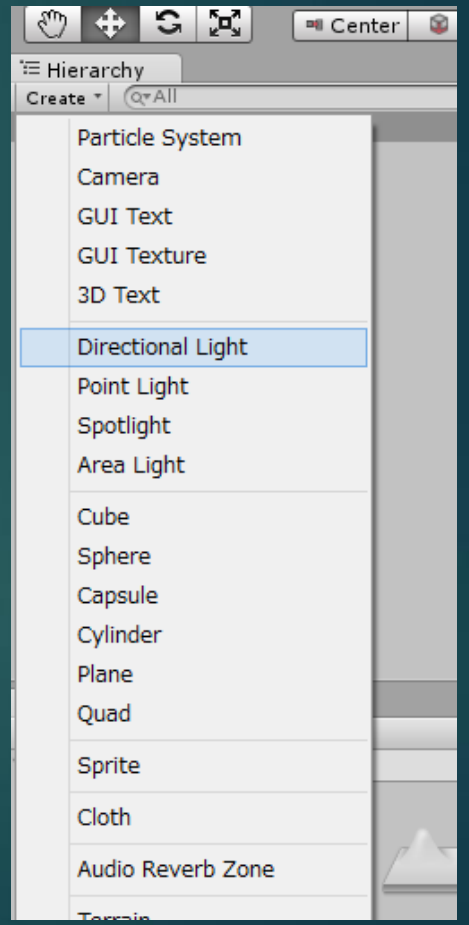



ちなみに木を取り除く場合は、「Edit Tree」→「Remove Tree」で消える。

# 5．ライトを置く＆空に模様を付ける＆水面を作る

Hierarchyの「Create」→「Directional　Light」を選択。
上の方から下を照らすように設置する。



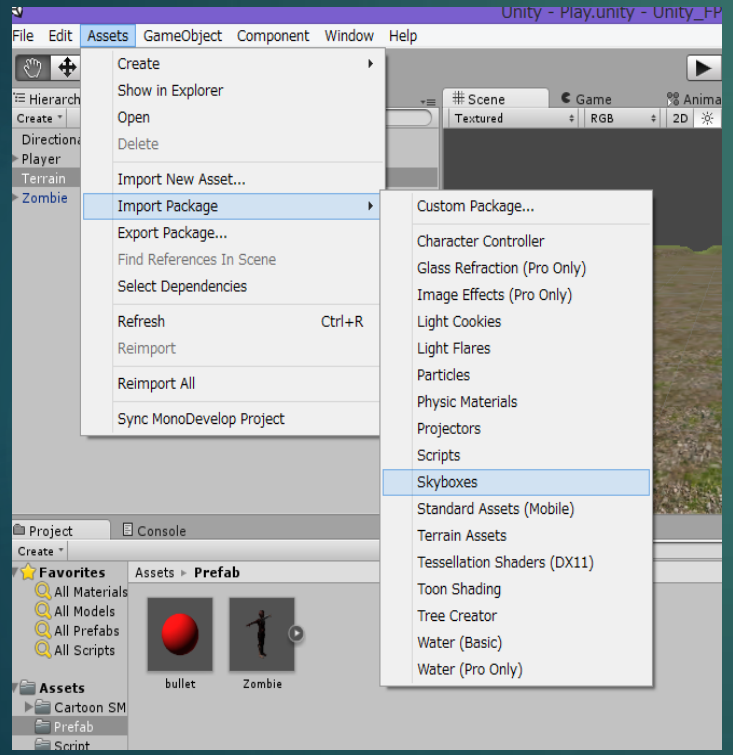



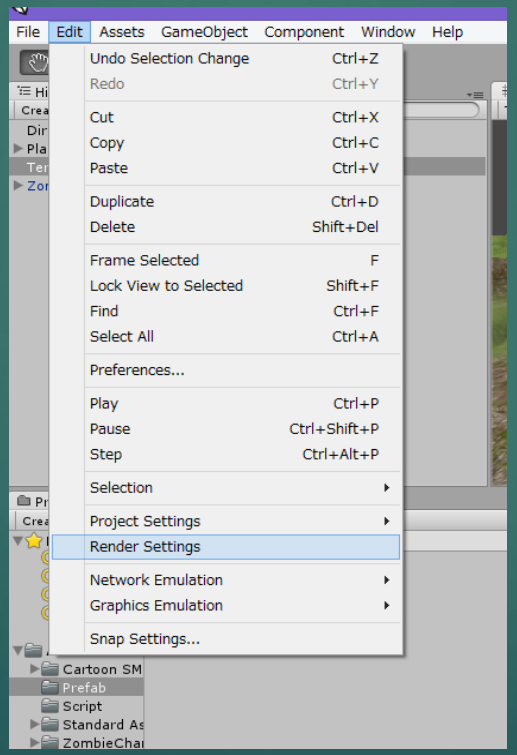
メニューの「Assets」→「Import Package」→「Skyboxes」をImportする。

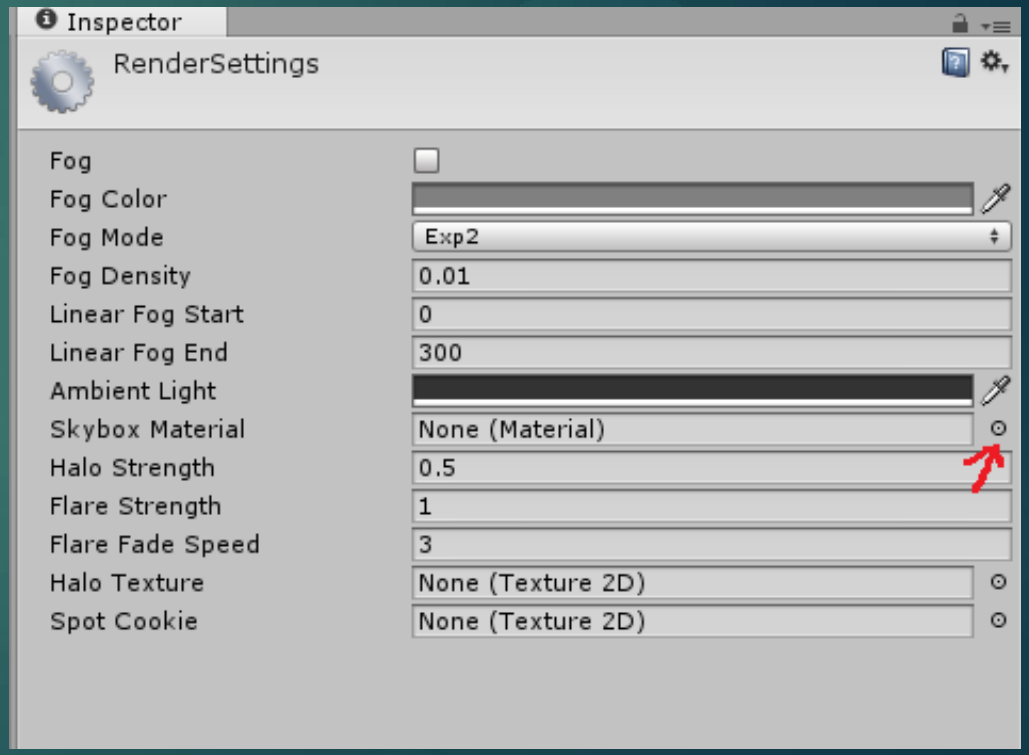
メニューの「Edit」→「Render Setting」をクリック

InspectorのSkybox Materialの右横の小さい丸をクリック

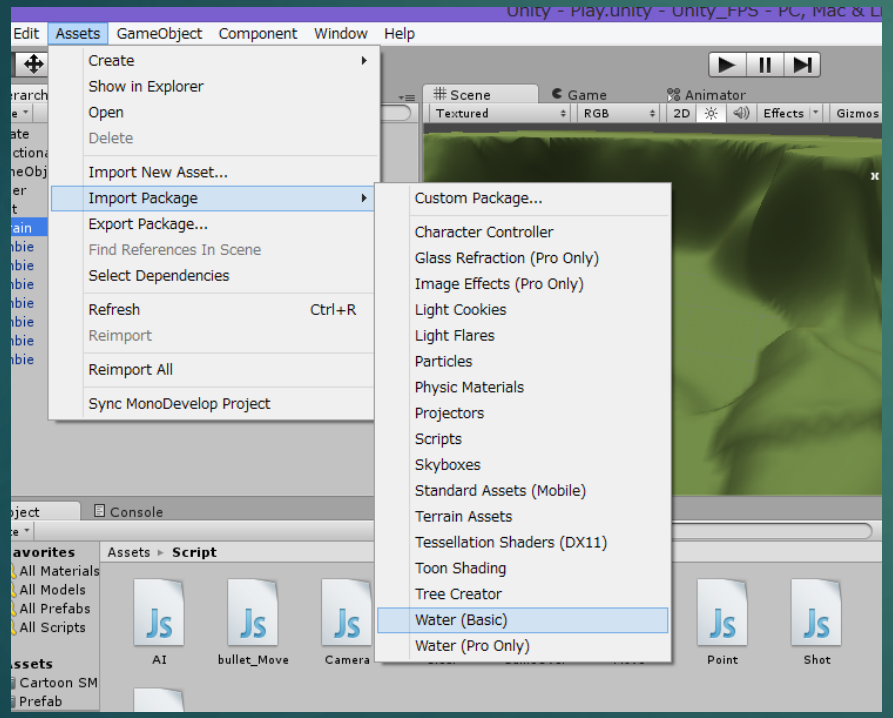
いろいろ素材がでるので好きなのを選ぶ。これで空模様が変わる。

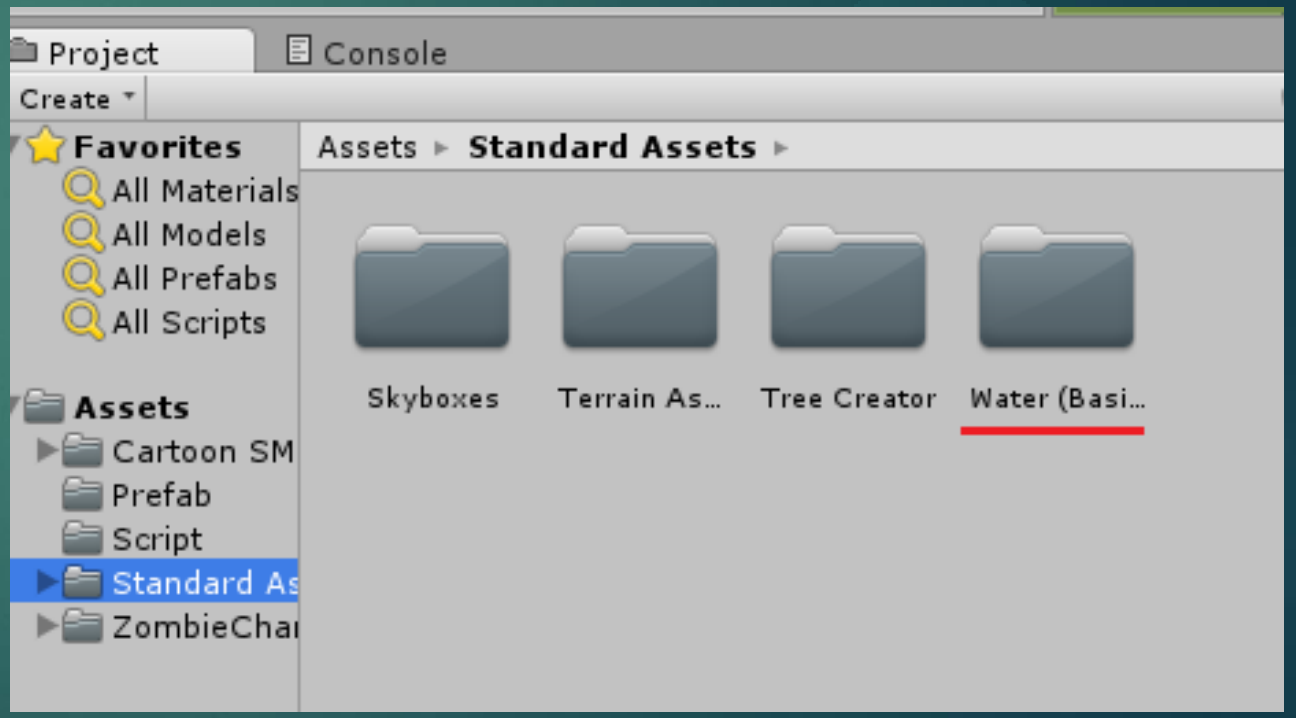


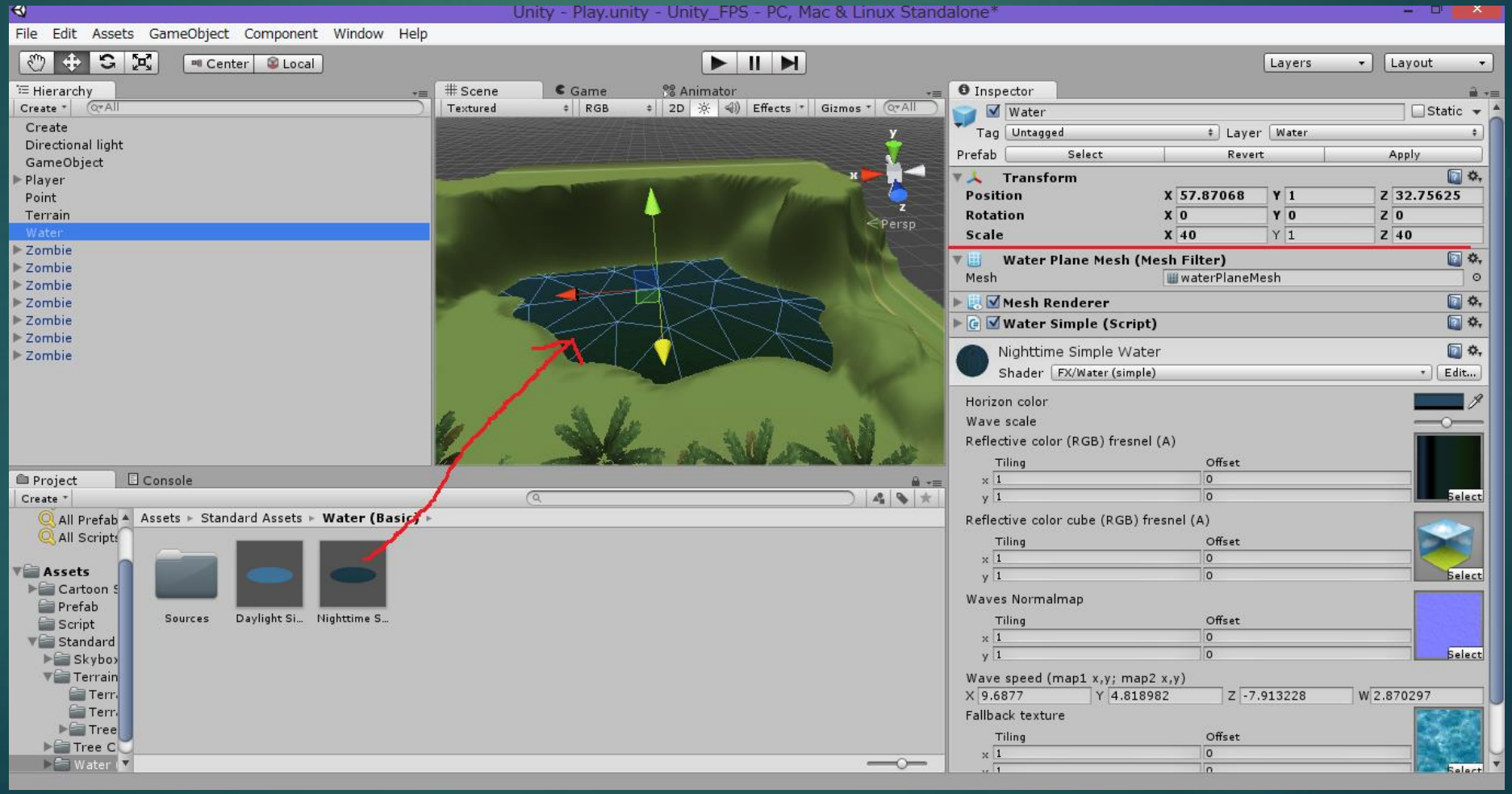



メニューの「Assets」→「Import Package」→「Water（Basic)」をImportする。
Importすると、「Projects」の「Asstes」の中に「Standard Assets」という
ファイルがあるので、それを開く。中に「Water（Basic)」というファイルが
あるのでそれも開く。



水面のプレファブがあるので「Scene」上の置きたいところにドラッグ
してください。Scaleを変えれば大きさが変わるので自由に設定して
大丈夫です。

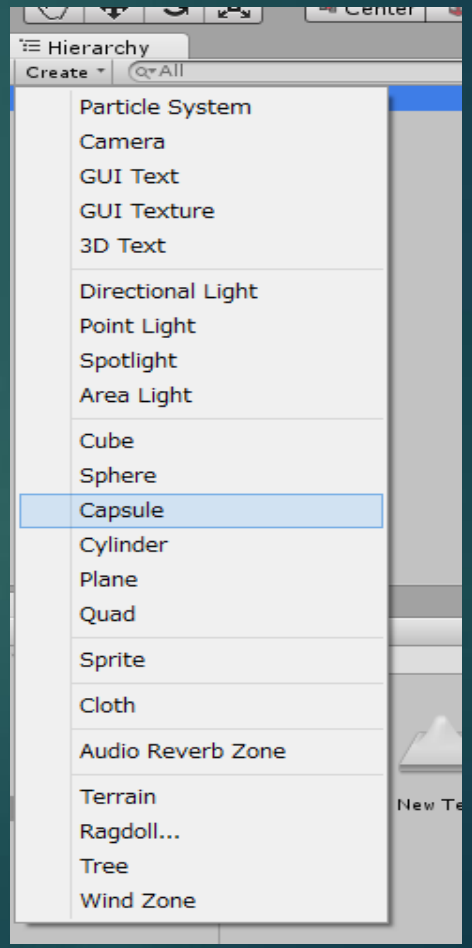


# 6．キャラクターを置く。

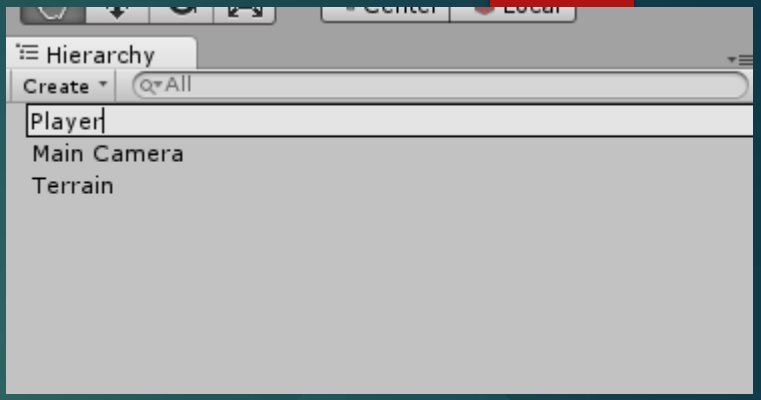
今回はFPSなので人型のキャラではなく、カプセルを動かす。

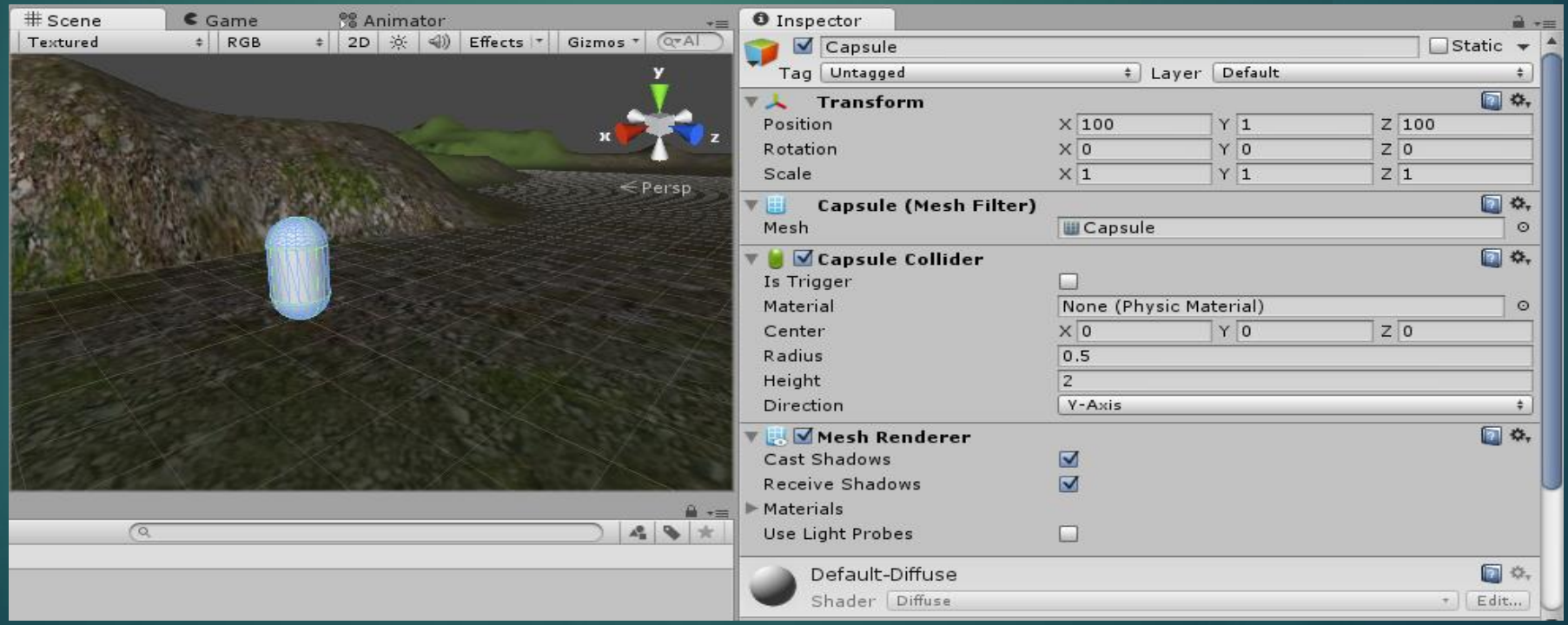
Hierarchyの「Create」→「Capsule」を選択。

Scene上にカプセルが現れる。

HierarchyのCapsuleをダブルクリックして名前を
「Player」に変える。
InspectorビューのTransformでPlayerの位置や
角度や大きさを変えられる。
PlayerのTransform→PositionのYの値を１にする。

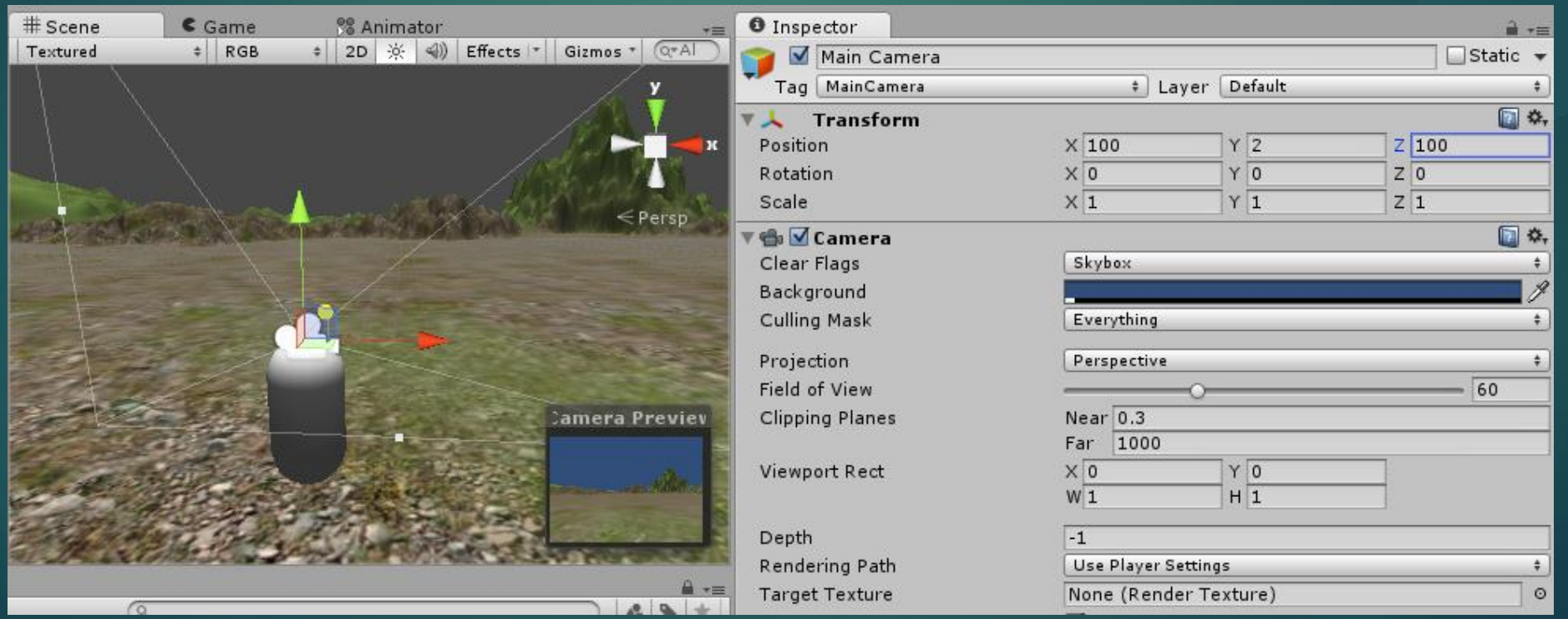


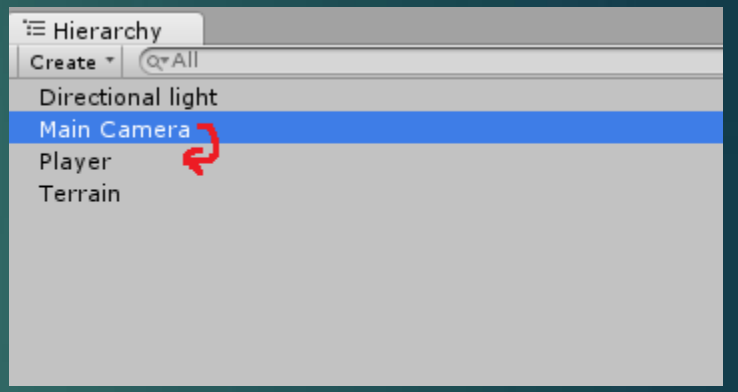
# 7 .カメラの設置

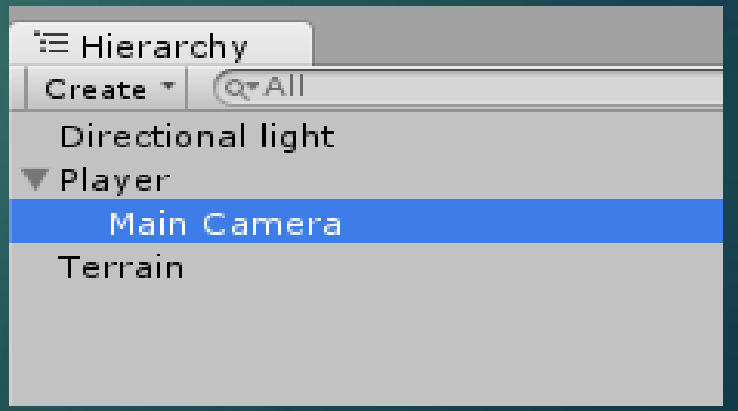
FPSを作るので、キャラクターの真上にMain Cameraを持ってくる。

画像の値を真似するのがはやい。



次にカメラがずっとキャラクターを写すように、カメラの構造を
変える。
HierarchyのMain CameraをドラッグしてPlayerに
重ねる。
重ねるとPlayerの下の階層にMain Cameraが
あることがわかる。
これでMain CameraはPlayerの一部になった。



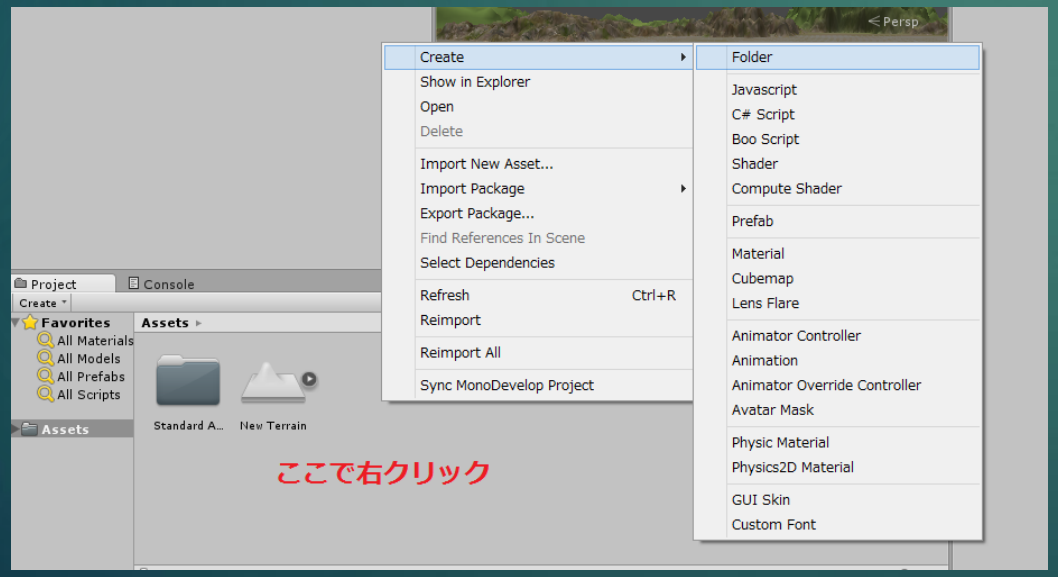
# 8．キャラクターを動かす

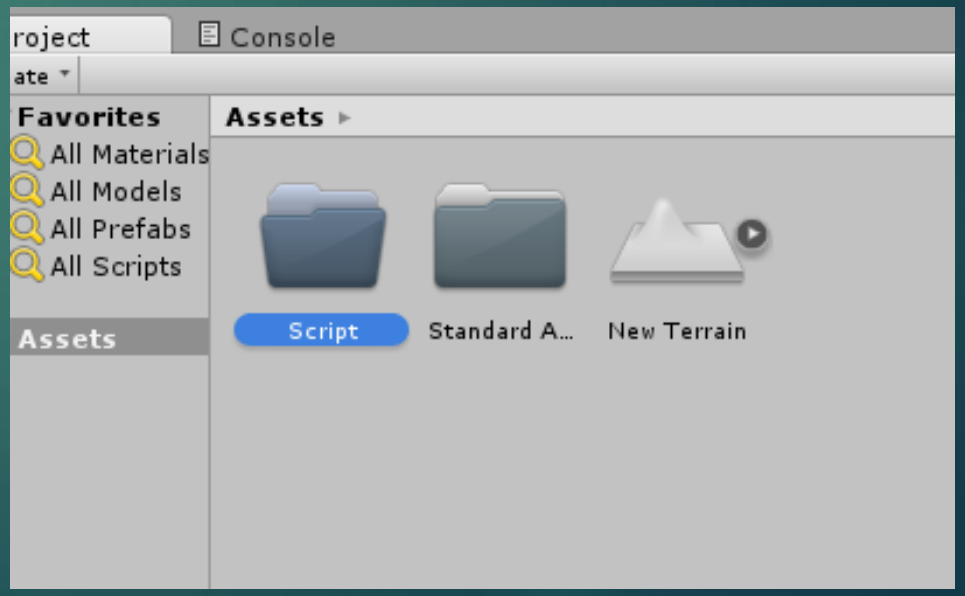
スクリプトを書きます。今回はJava Scriptで書きます。

ひとまずProject→右クリック→Create→Folder

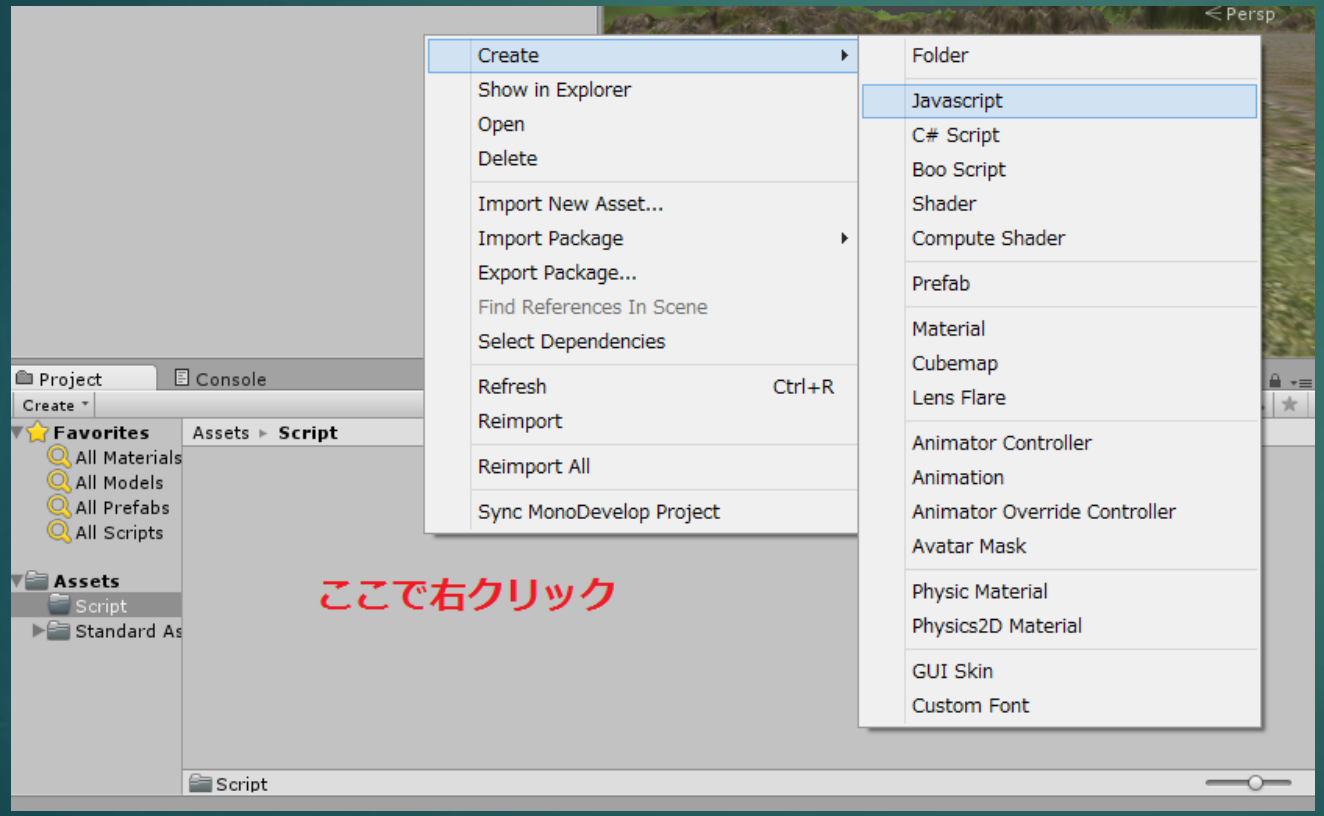
でフォルダを作成。

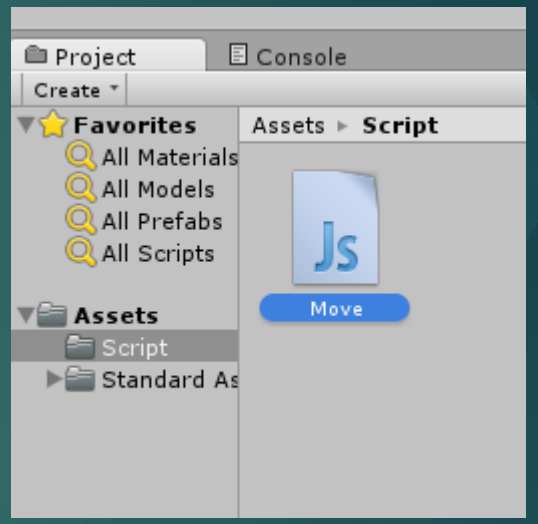
名前は「Script」とする。この中にスクリプトのコードをまとめる。



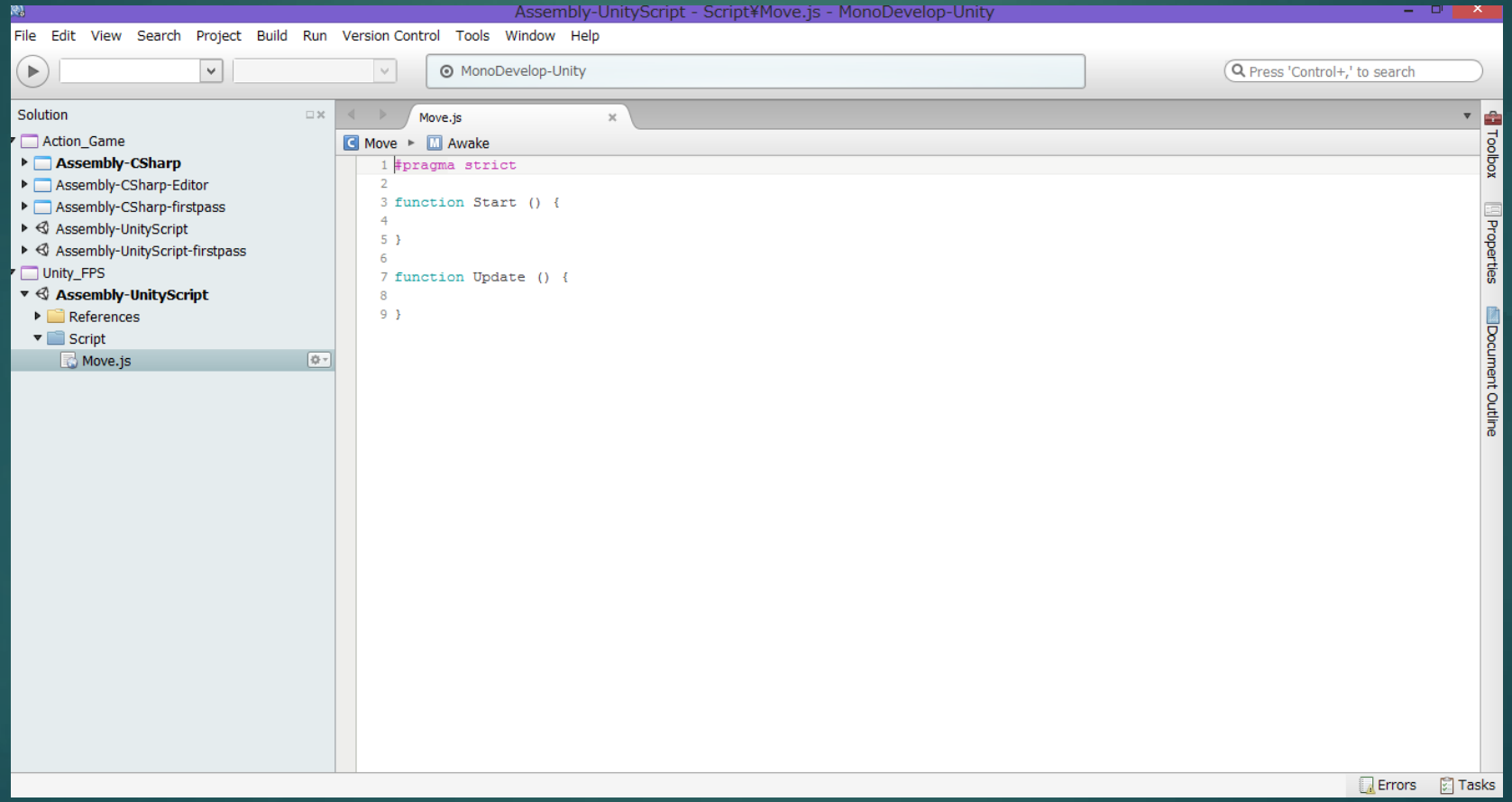



「Script」フォルダをダブルクリック→右クリック→Create→JavaScript
スクリプト名は「Move」とする。そして、「Move」をダブルクリック



「Move」をダブルクリックするとMonoDevelopが起動する。
ここにコードを書いていく。



「Move」の中に次のコードをそのまま記述。
コードが書き終わったらコードを保存（Ctrl+S）。

Move.js

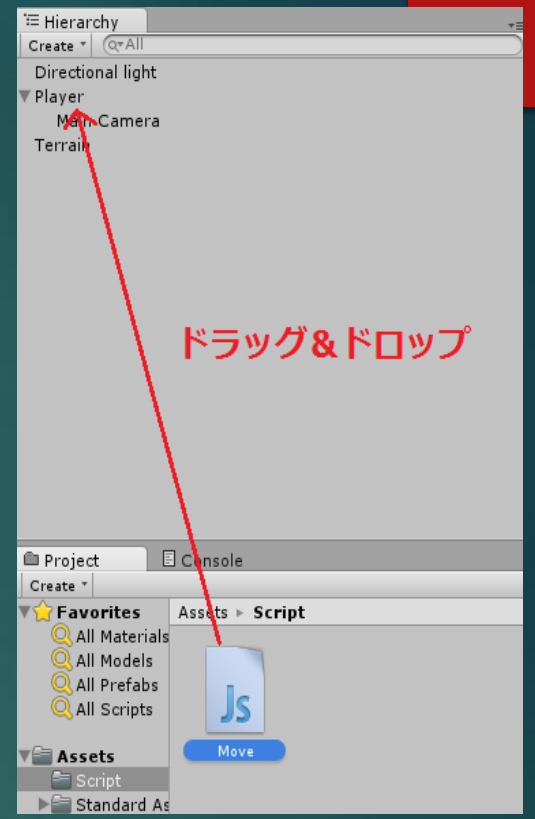
Move ▸ Awake

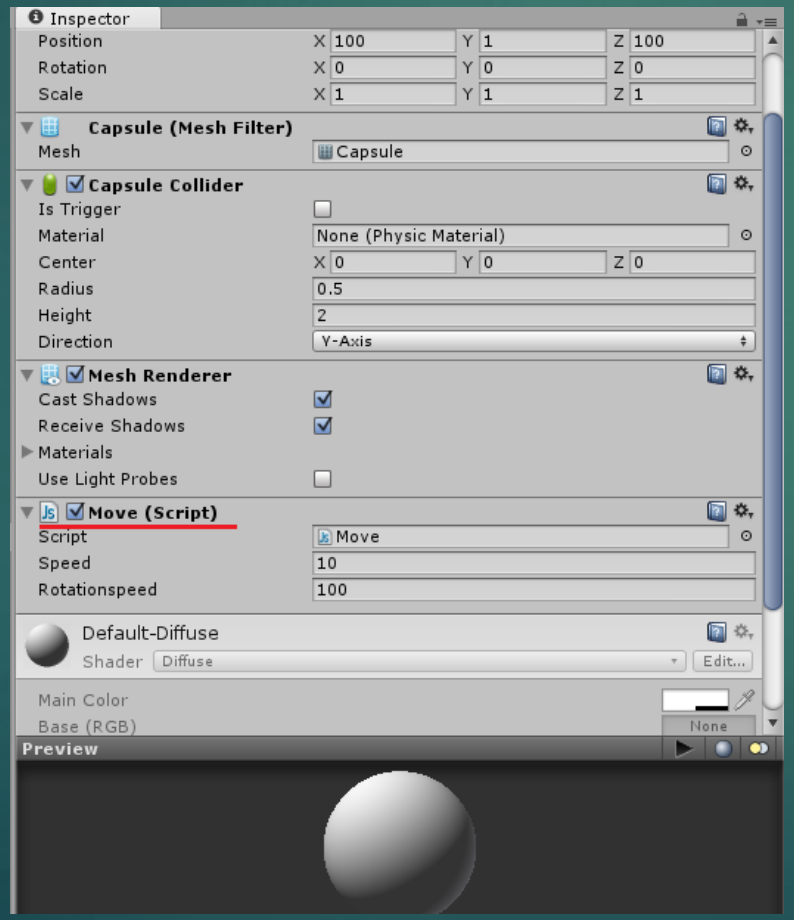
```
1  #pragma strict
2
3  var speed : float = 10.0;
4  var rotationspeed : float = 100.0;
5
6  function Start () {
7
8  }
9
10 function Update () {
11       var translation : float = Input.GetAxis("Vertical")*speed;
12       var rotation : float = Input.GetAxis("Horizontal")*rotationspeed;
13       var height : float = Terrain.activeTerrain.SampleHeight(transform.position)+1-transform.position.y;
14       translation*=Time.deltaTime;
15       rotation*=Time.deltaTime;
16       transform.Translate(0,height,translation);
17       transform.Rotate(0,rotation,0);
18 }
```

コードが書けたら、それをPlayerに実装させる。

「Move」をドラッグして、Playerの上で離す。

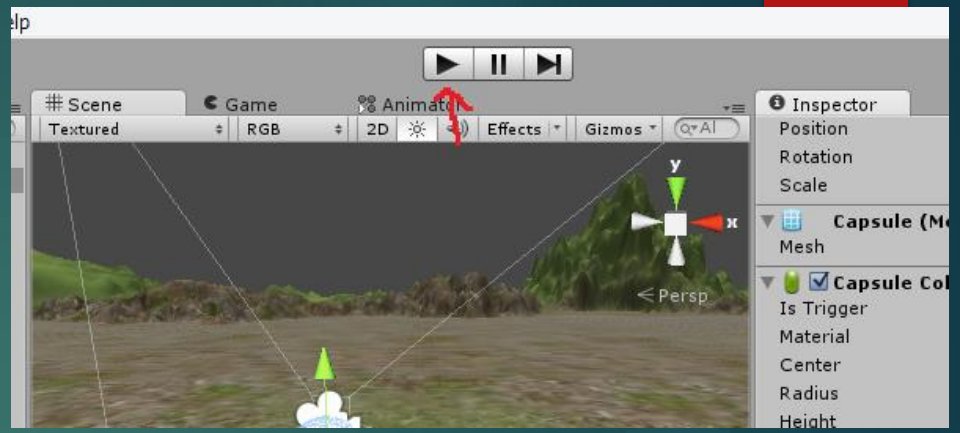
PlayerのInspectorに「Move」という項目が追加されていれば大丈夫。

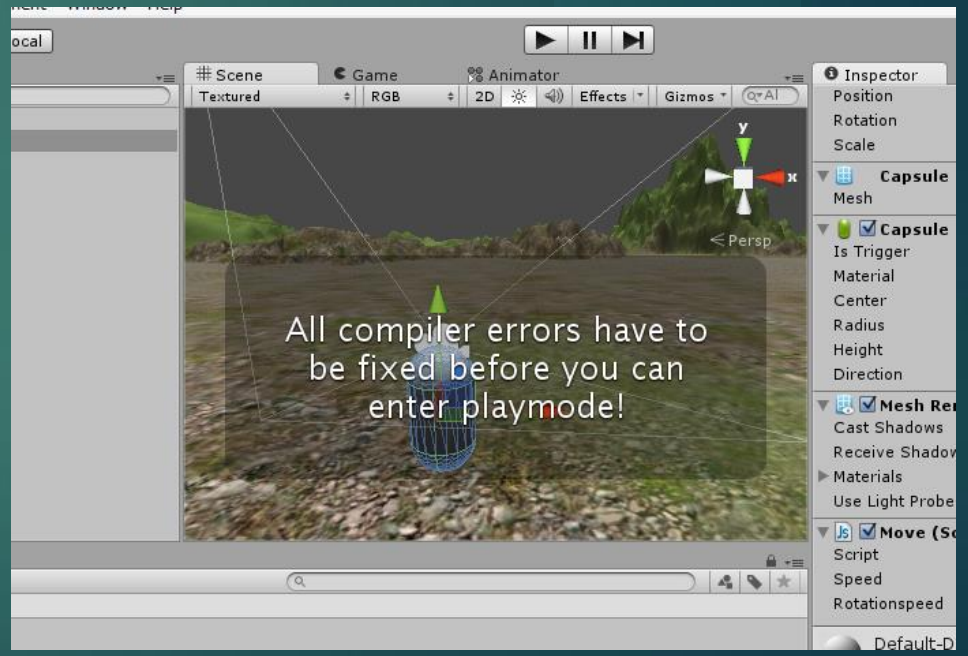


実行してみる。
図の再生ボタンを押すとScene画面が消えて
Game画面になる。
矢印キーで動けたら成功！(∩ﾟ∀ﾟ)
動かなかったり、右の図のようになってしまったら、スクリプトの書き間違えの可能性があるので、書き直す。
エラーはコンソールで確認できる。

これでキャラクターを動かすことは終わり！

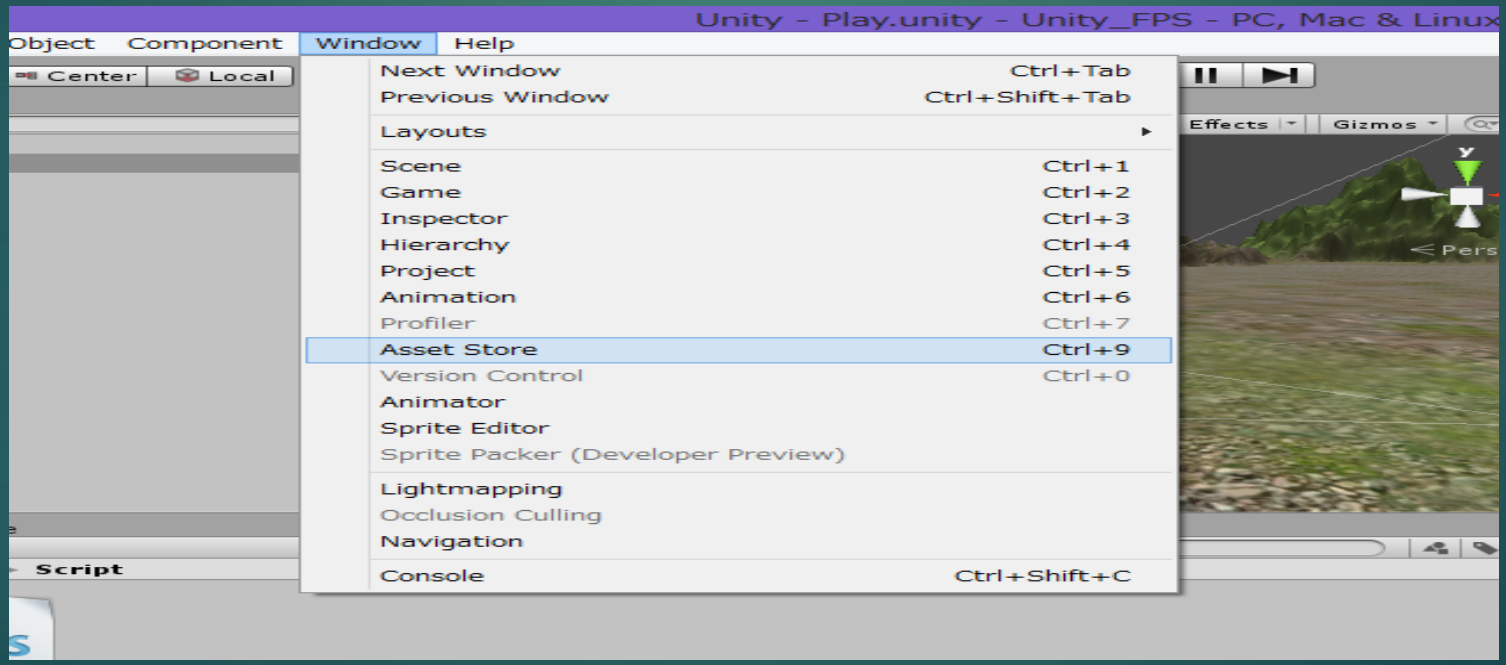


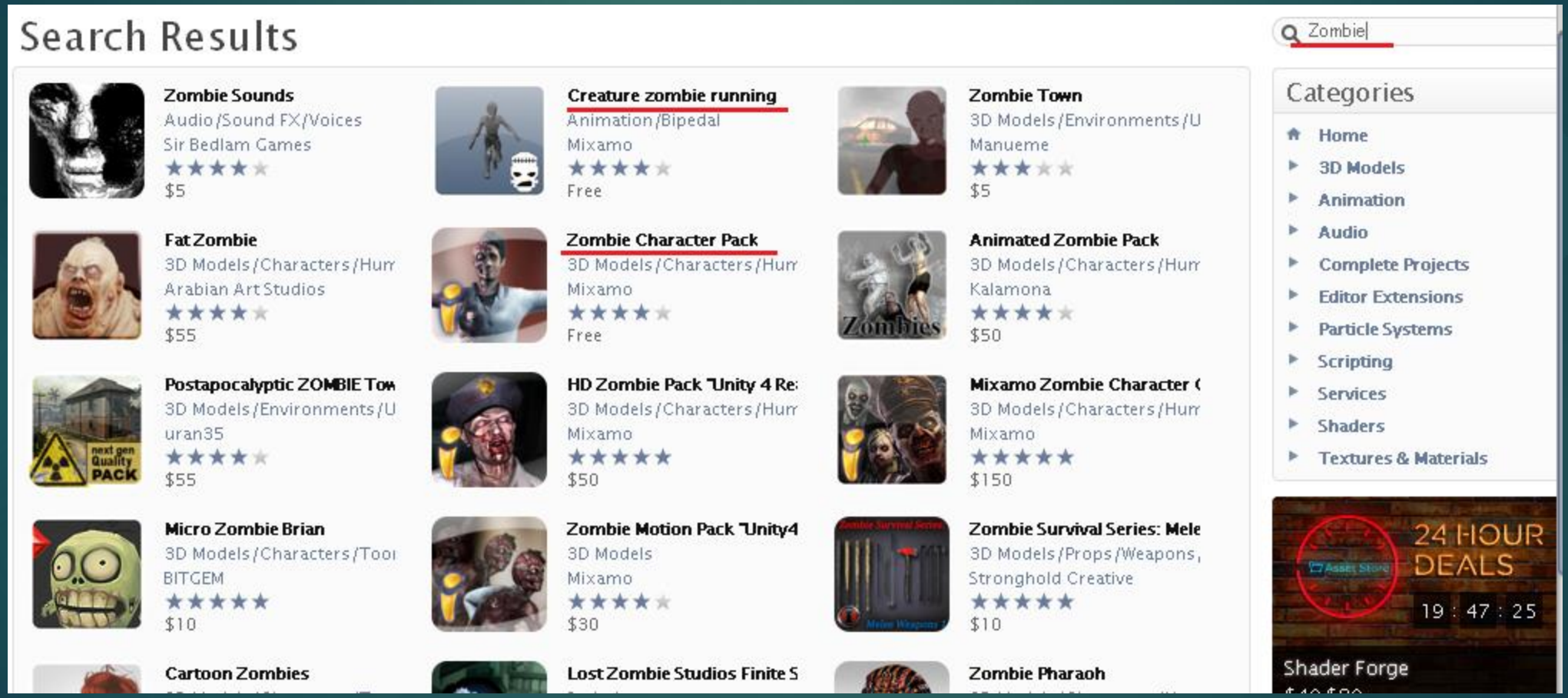
# 9．敵キャラを置く

敵キャラが欲しいので、モデルをAsset Storeから借りる。

メニューの「Window」→「Asset Store」をクリック。

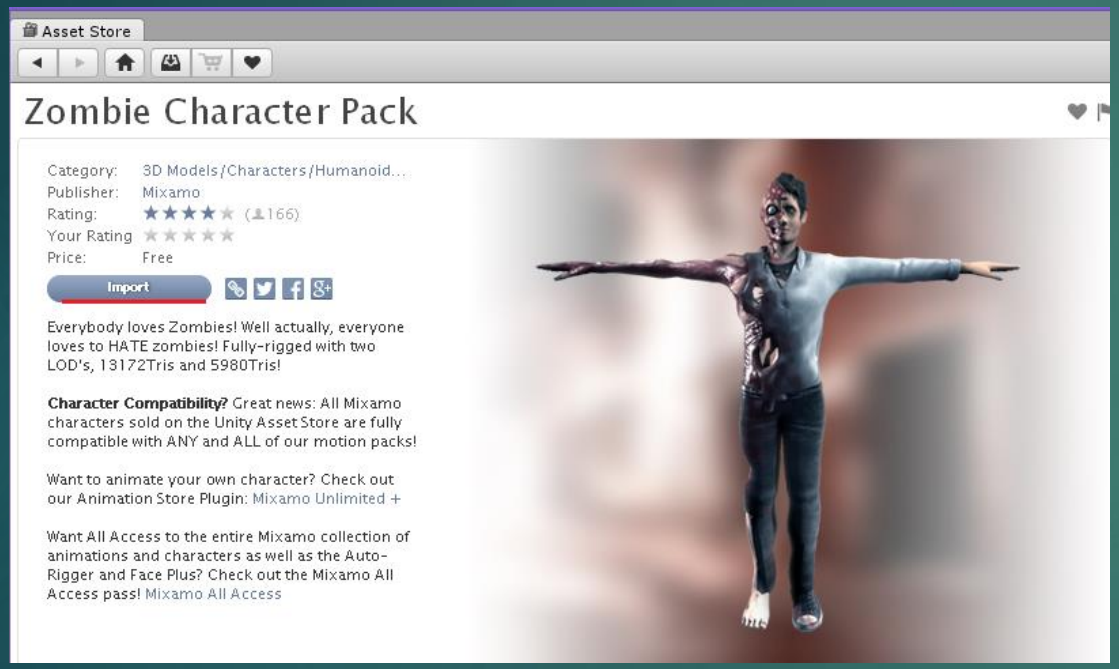


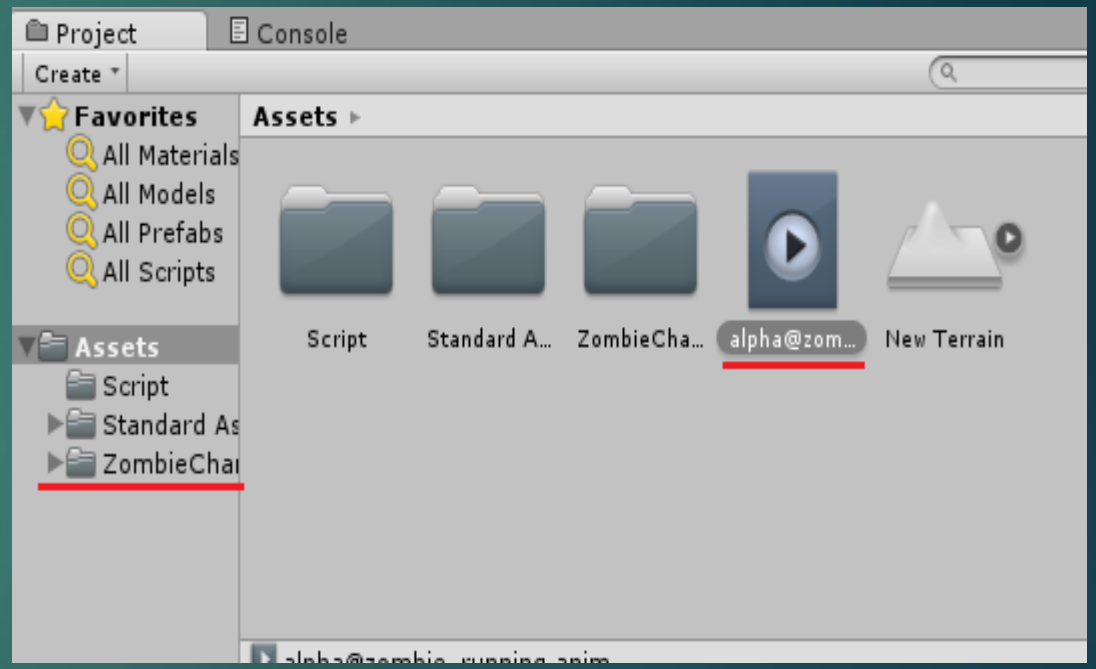
右上の検索のところで「Zombie」と入力。
今回はこの２つを使う（無料なので）。



両方ともクリックしてDownload→Importする。

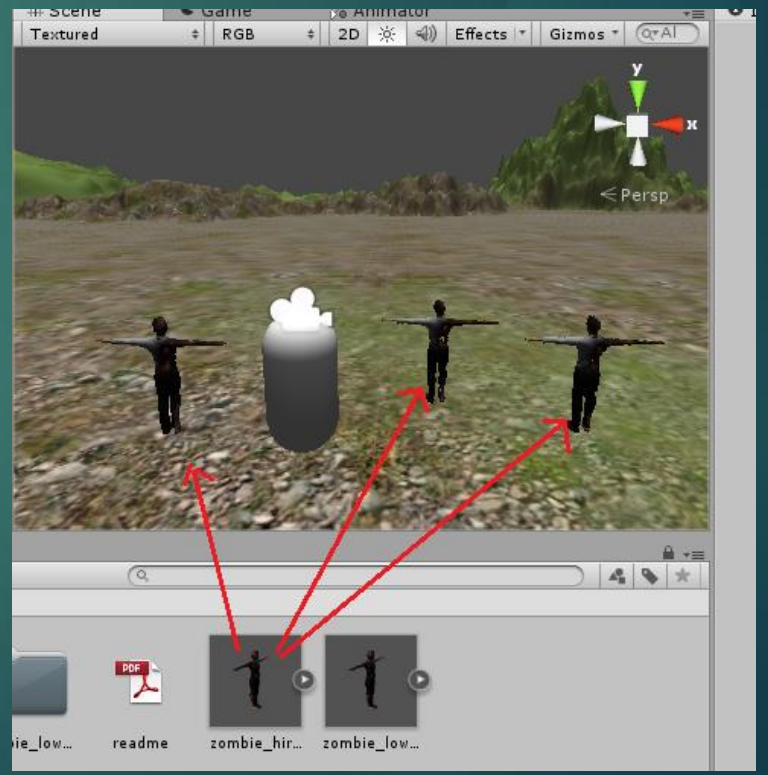
Importすると、勝手にUnityにImportされる。



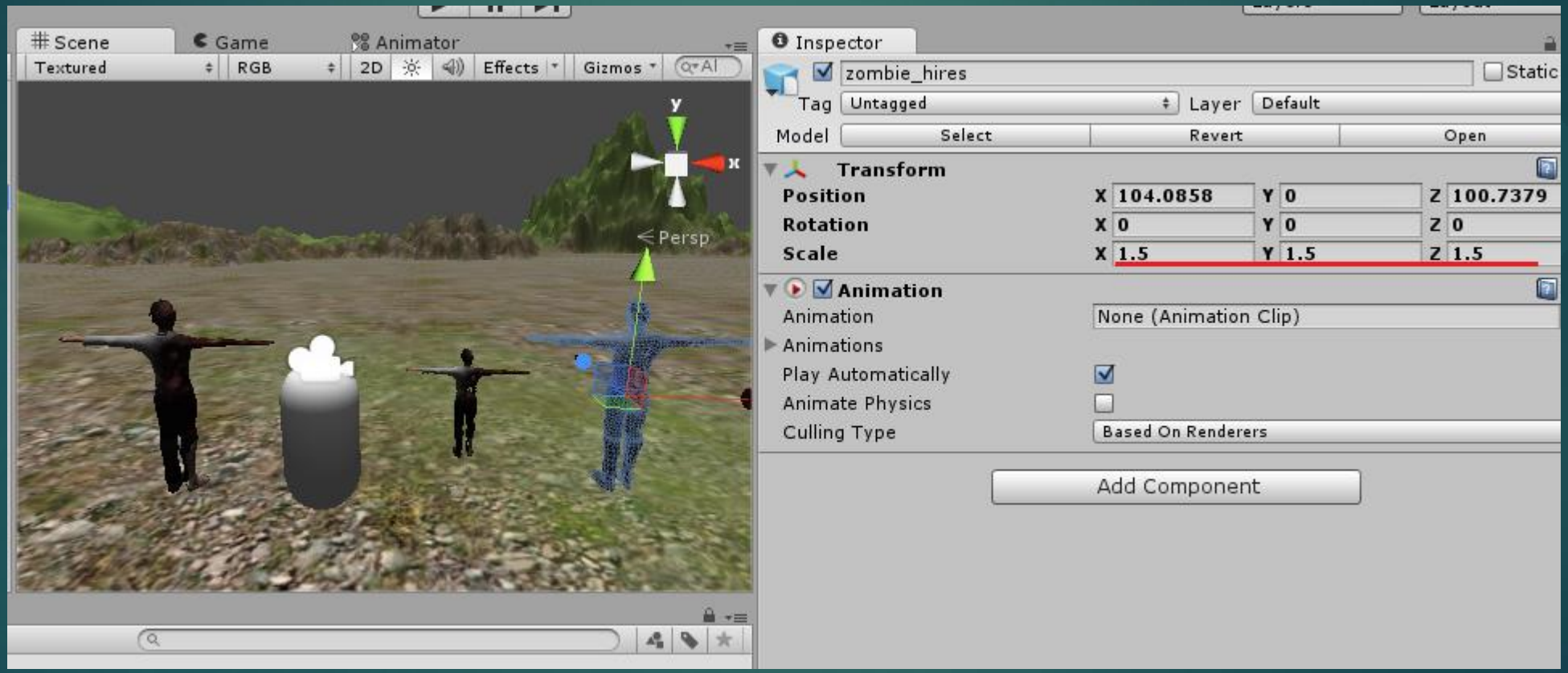



さっきImportしたゾンビのキャラクターはプレファブ化してる。
プレファブとは普通のJavaやC言語でいうクラスと同じ！
よって、たくさん複製したいものはプレファブ化して、プレファブ化し
たものからどんどん同じものを作ったほうが１つ１つ同じの作るより
楽になる。

ゾンビをScene上へドラッグしてやれば
どんどんゾンビがつくれる！



サイズが微妙なのでゾンビのScaleを１．５にする。

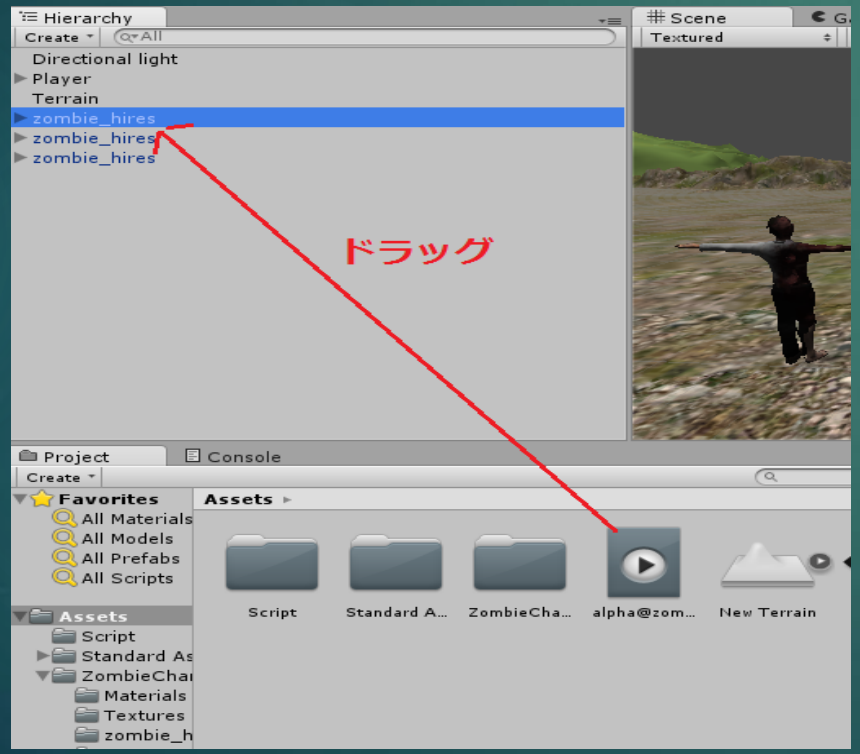


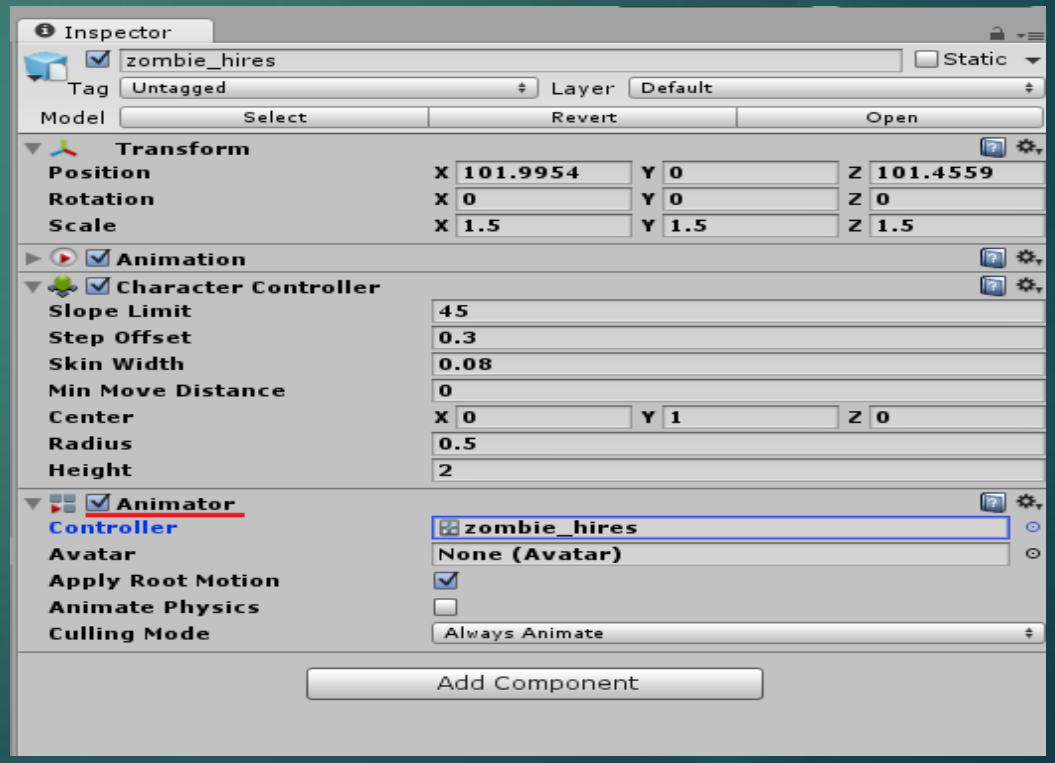
# １０．ゾンビを動かす

Asset Storeから持ってきたゾンビのモーションを入れる。

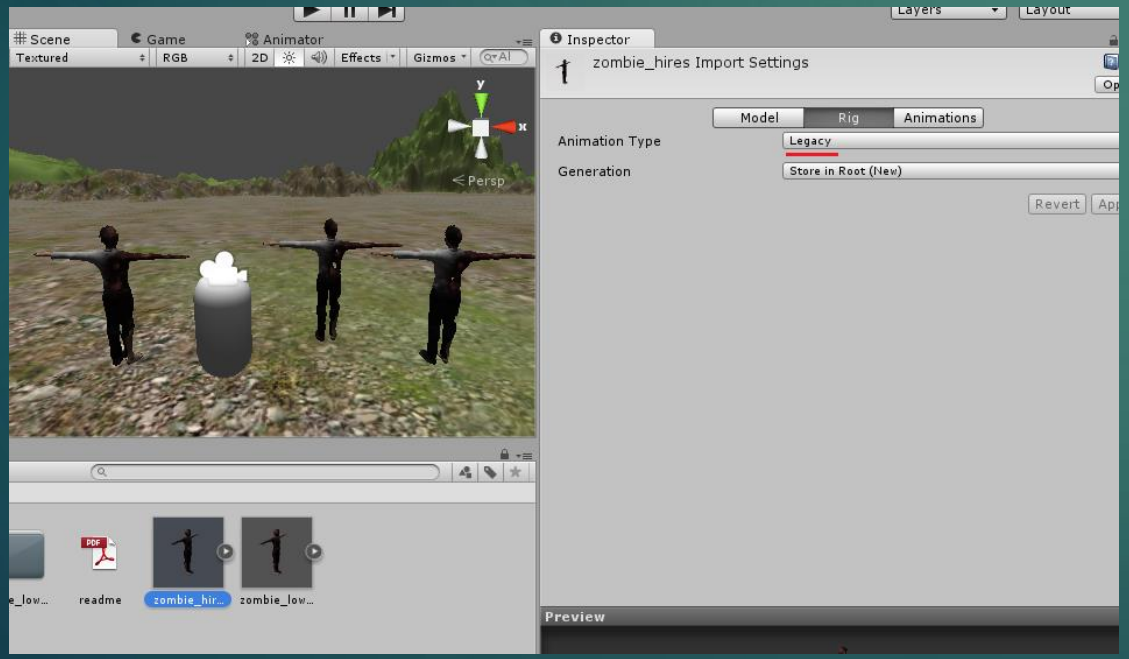
モーションデータをHierarchyのゾンビにドラッグする。

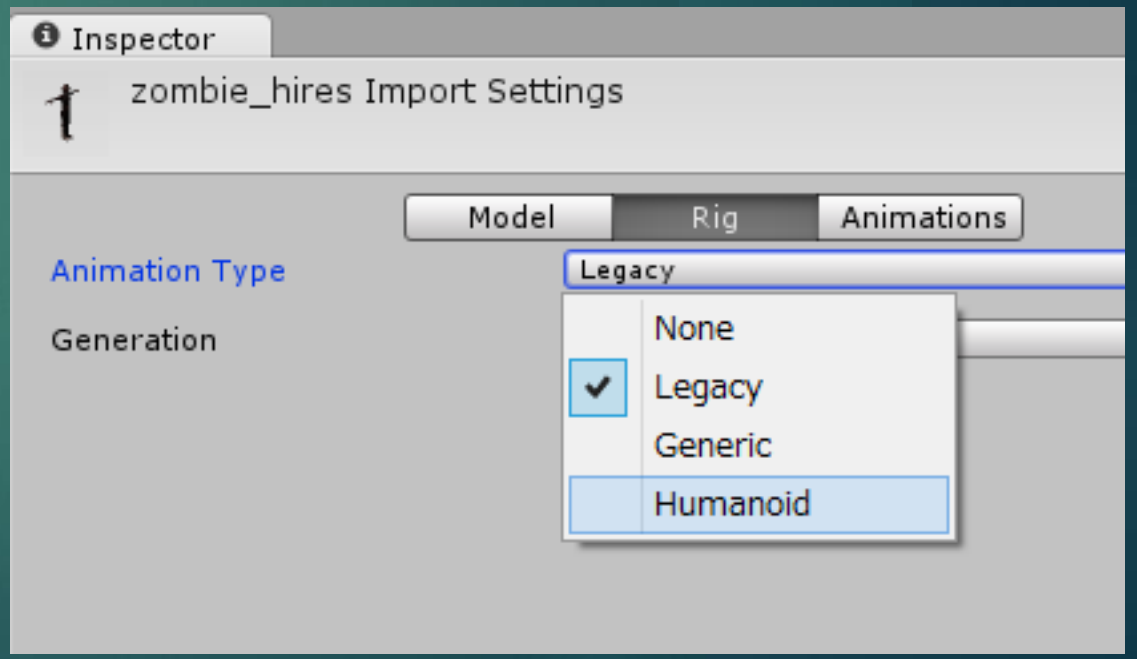
ゾンビのInspectorにAnimatorという項目が追加されれば成功。



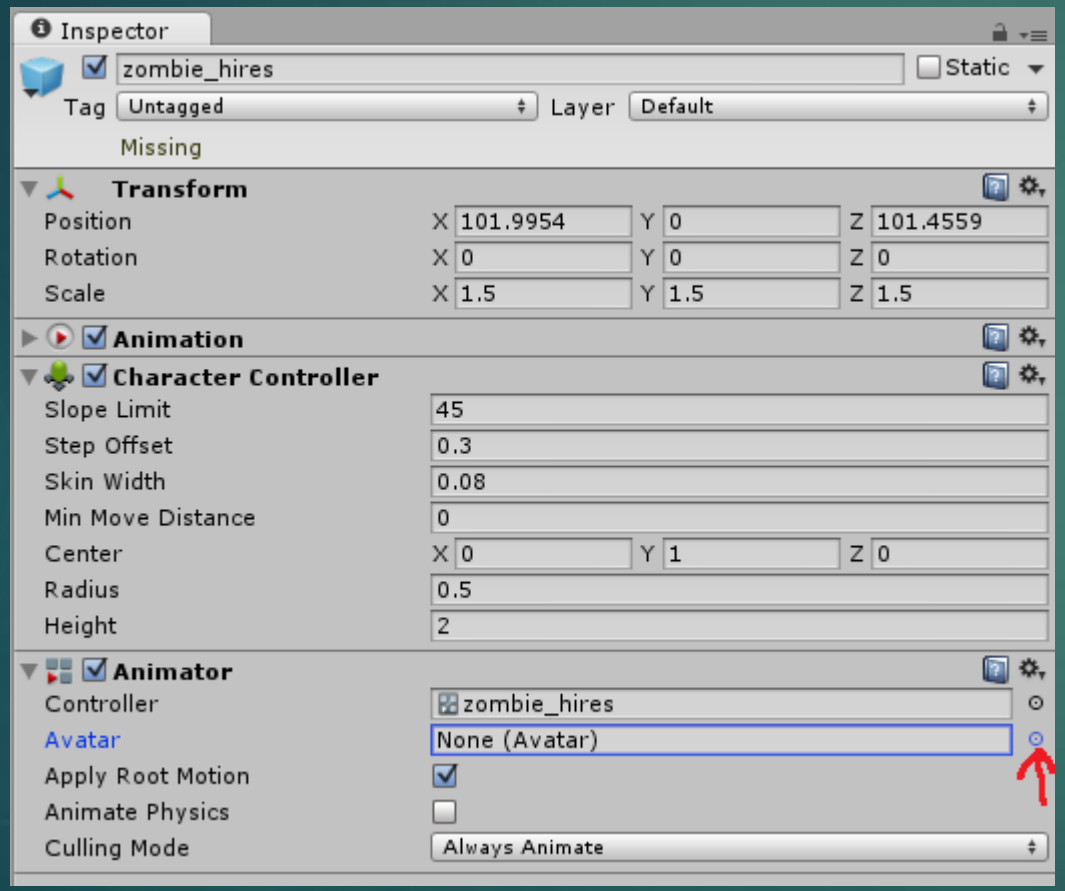



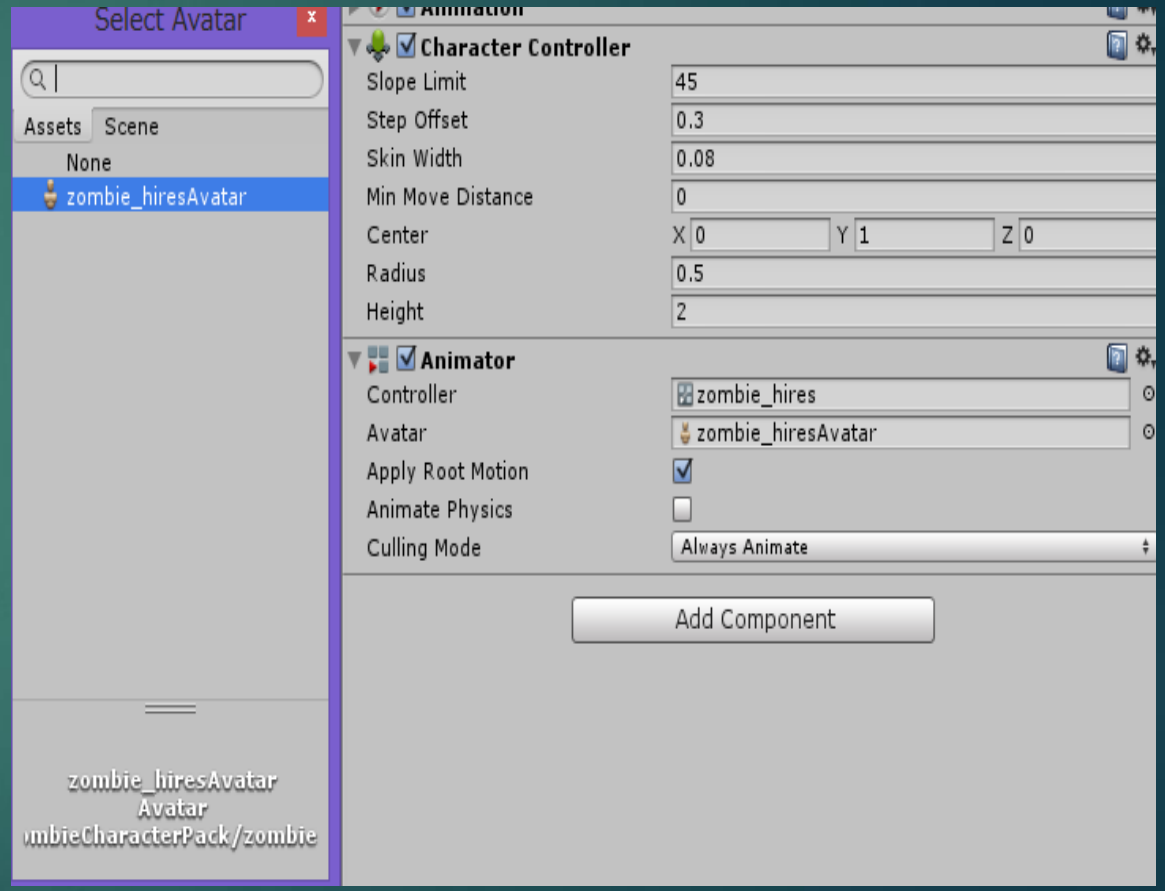
Projectのゾンビをクリックし、Inspectorに
詳細をだす。
Animation TypeがLegacyになってるので
Humanoidにする。
そしてApplyを押す。これでゾンビが人型になる。



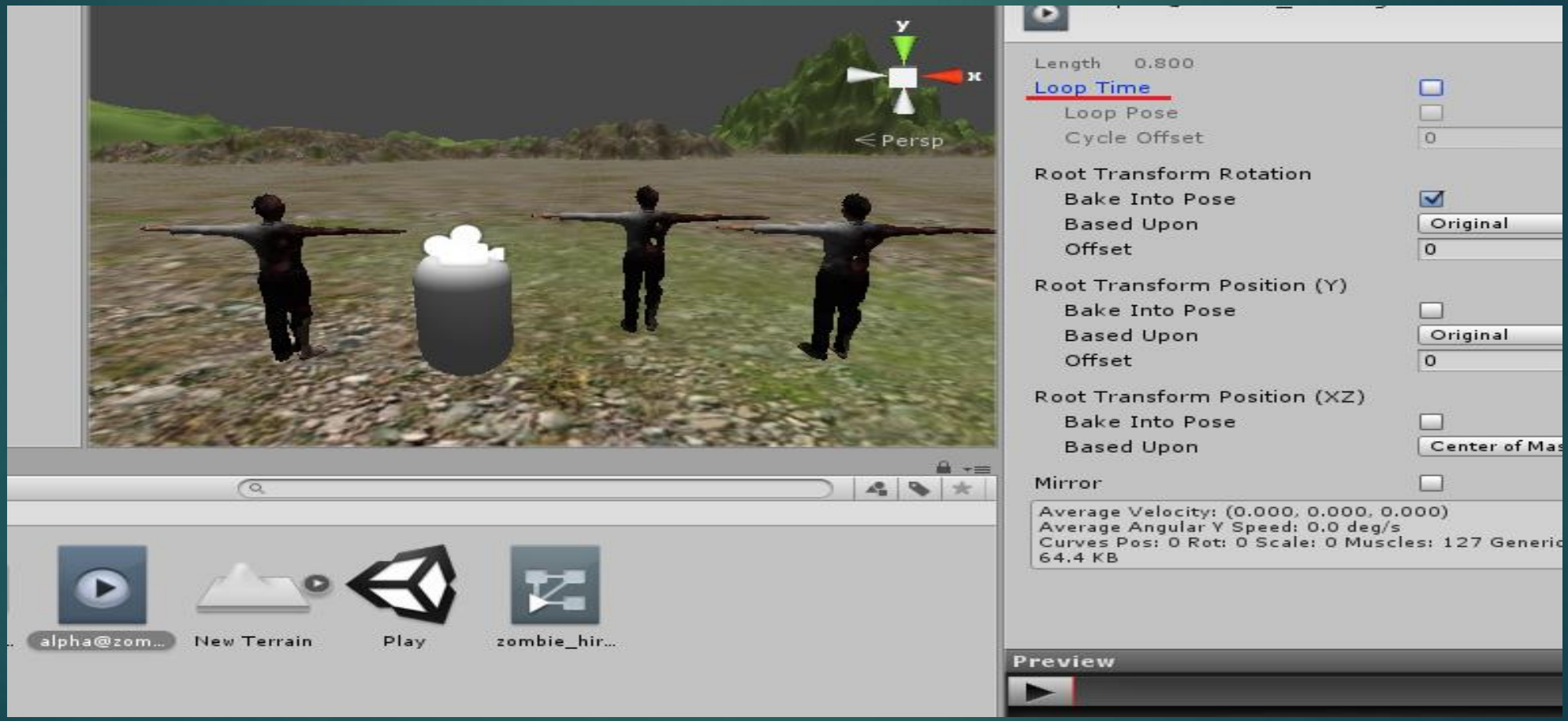



ゾンビのInspectorに戻り、AnimatorのAvatar欄の小さい丸をクリック。
ゾンビのアバターがあるのでそれを選択する。



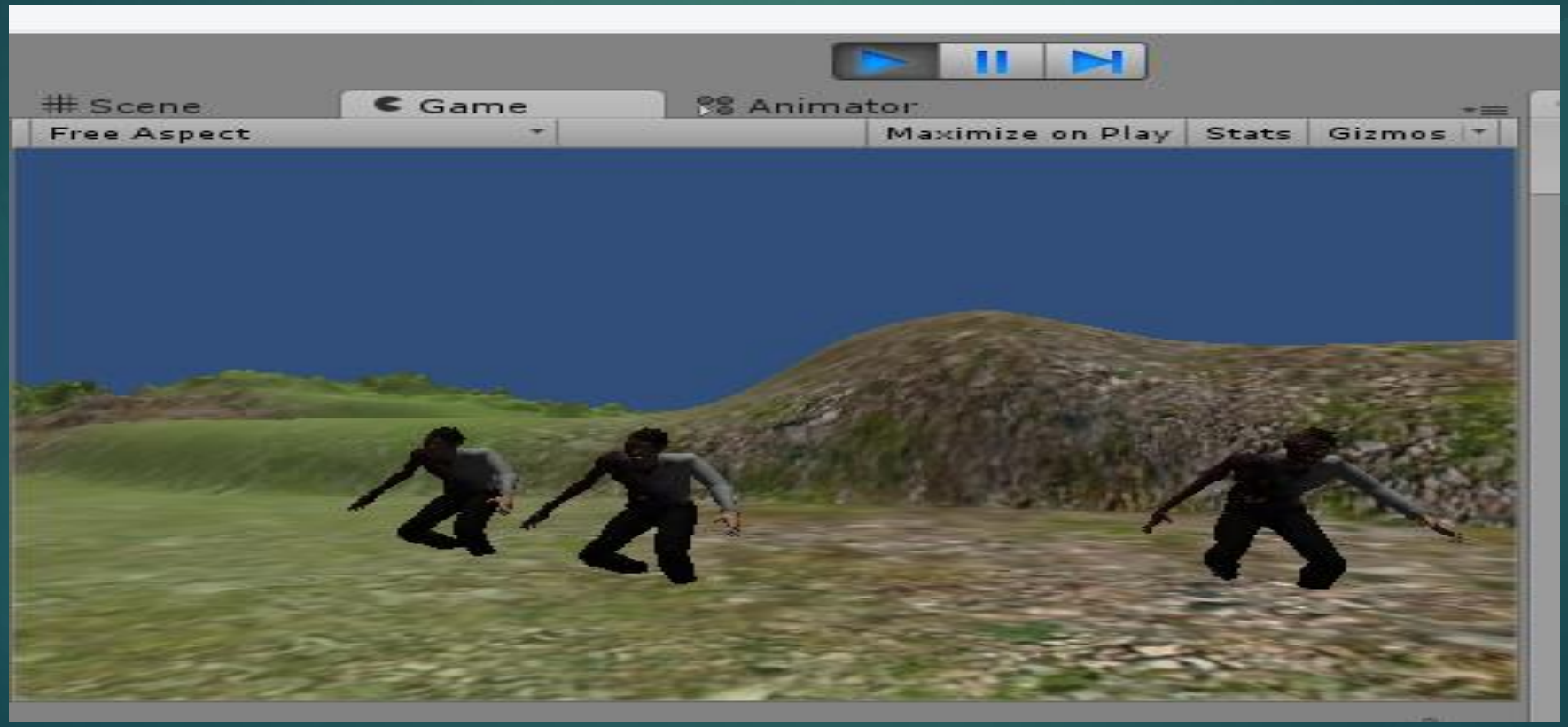
Projectのモーションモデルをクリックし、Loop Timeのチェックを入れる。
これでループができる。



実行すると・・・

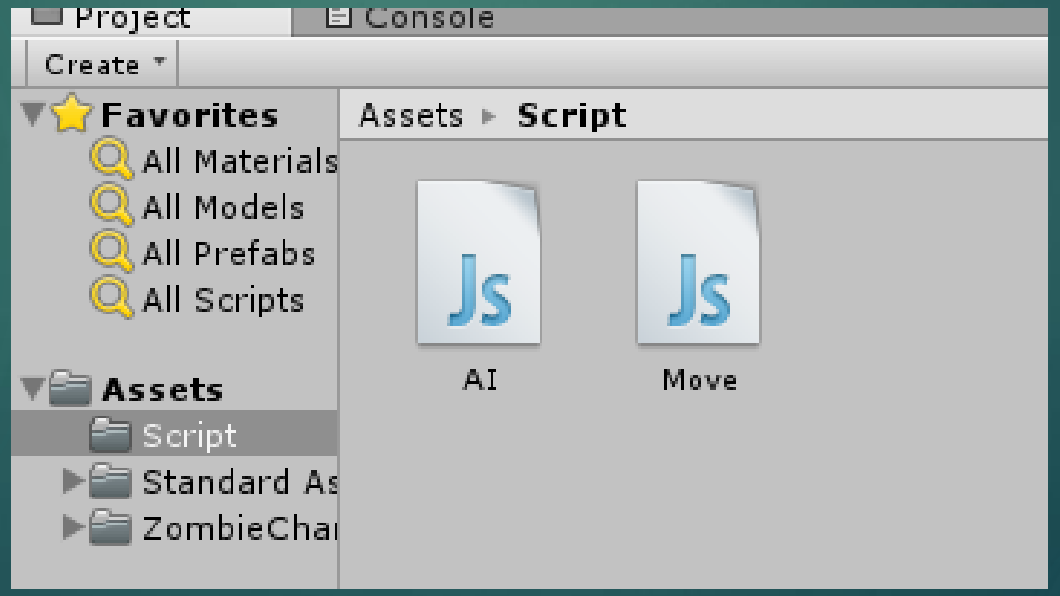
ゾンビが走ってる！！　追いかけては来ないけど・・・

ひとまずゾンビにモーションを付けることができた！



# １１．ゾンビにAIを付ける

ゾンビに自分を追いかけてくるようなAIを付ける。
またスクリプトを書くのでScriptフォルダ内で右クリックし、
JavaScriptを１つ作る。
名前は「AI」で。



# 「AI」の中にそのまま記述

Move.js | AI.js

AI ▸ Awake

```
#pragma strict

private var isEnable = false;

var target : GameObject;
var speed : float = 5.0;
var gravity : float = 20.0;

function Start () {

}

function Update () {
      var controller : CharacterController = GetComponent(CharacterController);
      var moveDirection = Vector3.zero;

      if (isEnable == true){
           if (2<Vector3.Distance(this.transform.position,target.transform.position)) {
               var targetDirection : Vector3 = Vector3(target.transform.position.x,
               this.transform.position.y,target.transform.position.z);
               this.transform.rotation = Quaternion.Slerp(this.transform.rotation,
               Quaternion.LookRotation(targetDirection-this.transform.position),Time.time*0.1);
               moveDirection += transform.forward*1;
               moveDirection.y-=gravity*Time.deltaTime;
               controller.Move(moveDirection*Time.deltaTime*speed);
           }
      }
}

function OnTriggerEnter(c : Collider) {
     if (c.name=='Player') {
          isEnable = true;
```

# 続き

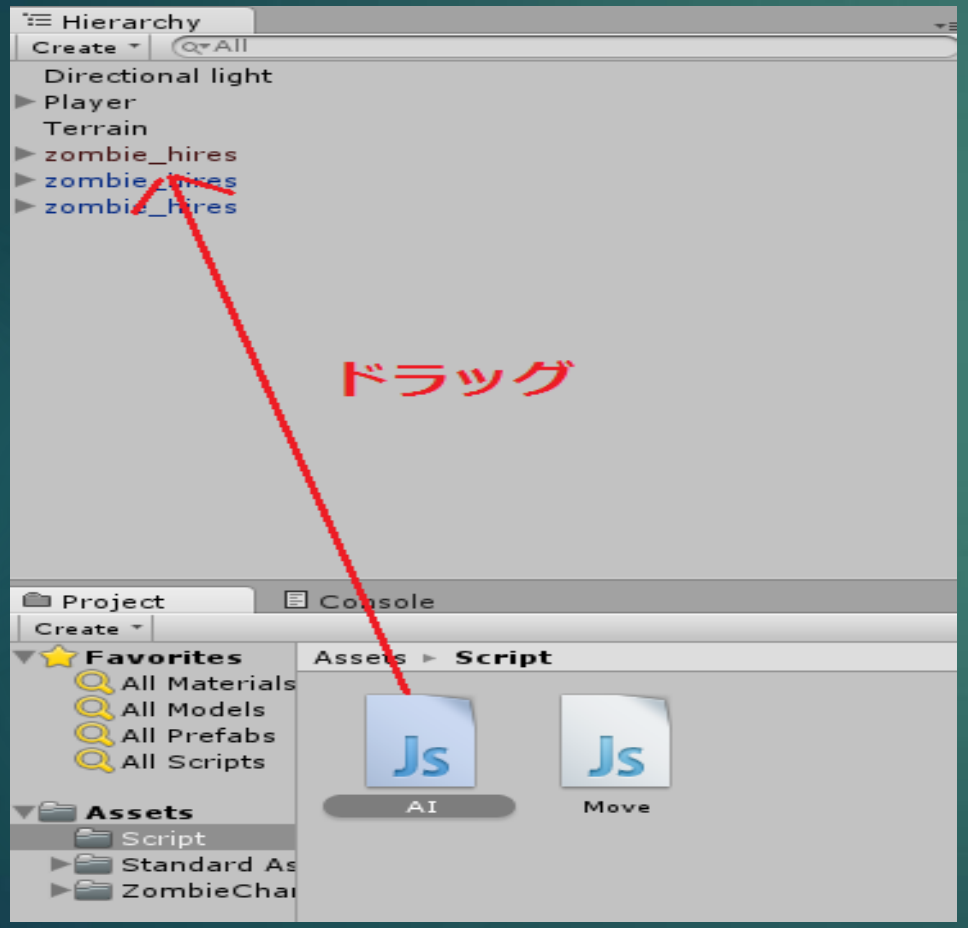
Move.js | AI.js

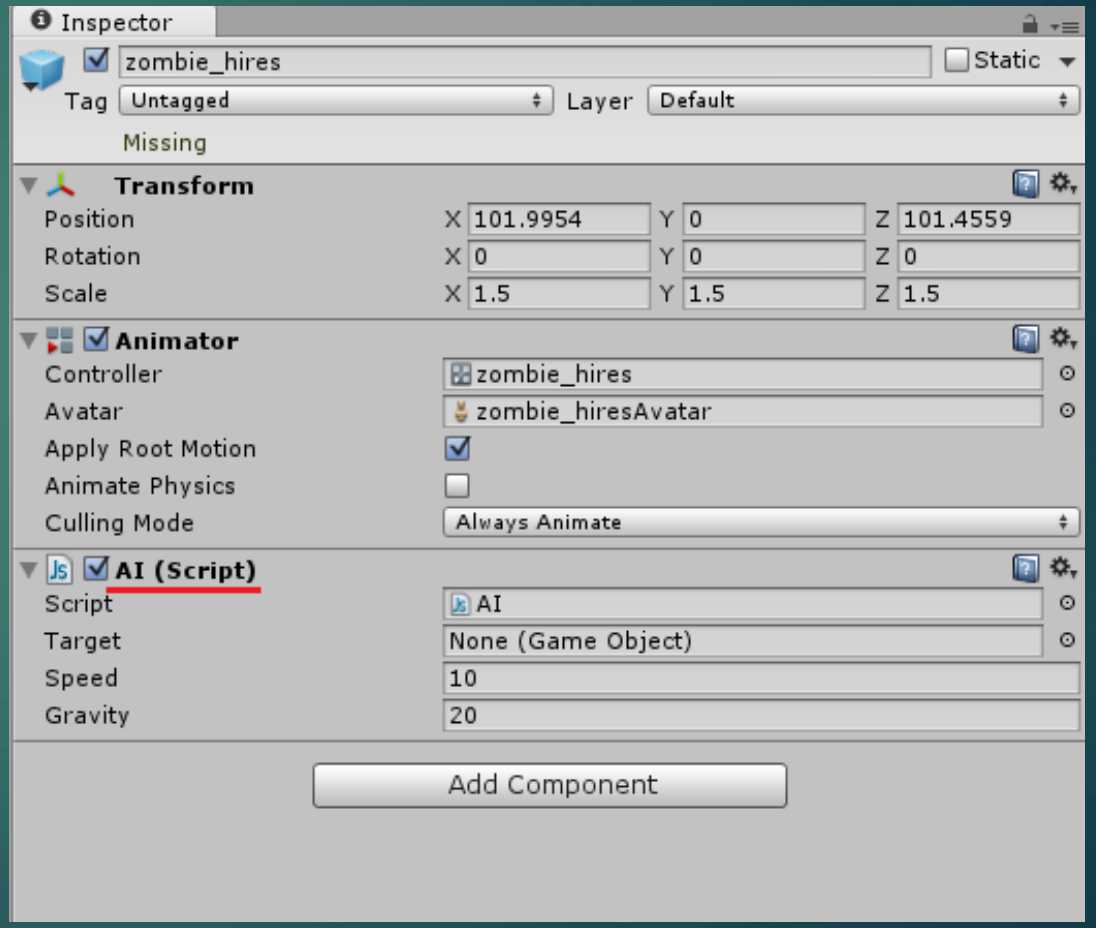
AI ▸ Awake

```
10
11 }
12
13 function Update () {
14       var controller : CharacterController = GetComponent(CharacterController);
15       var moveDirection = Vector3.zero;
16
17       if (isEnable == true){
18            if (2<Vector3.Distance(this.transform.position,target.transform.position)) {
19                var targetDirection : Vector3 = Vector3(target.transform.position.x,
20                this.transform.position.y,target.transform.position.z);
21                this.transform.rotation = Quaternion.Slerp(this.transform.rotation,
22                Quaternion.LookRotation(targetDirection-this.transform.position),Time.time*0.1);
23                moveDirection += transform.forward*1;
24                moveDirection.y-=gravity*Time.deltaTime;
25                controller.Move(moveDirection*Time.deltaTime*speed);
26            }
27       }
28 }
29
30 function OnTriggerEnter(c : Collider) {
31      if (c.name=='Player') {
32          isEnable = true;
33          target = c.gameObject;
34      }
35 }
36
37 function OnTriggerExit(c : Collider) {
38      if (c.name == 'Player'){
39          isEnable = false;
40      }
41 }
```

つくったスクリプトをゾンビに付ける。

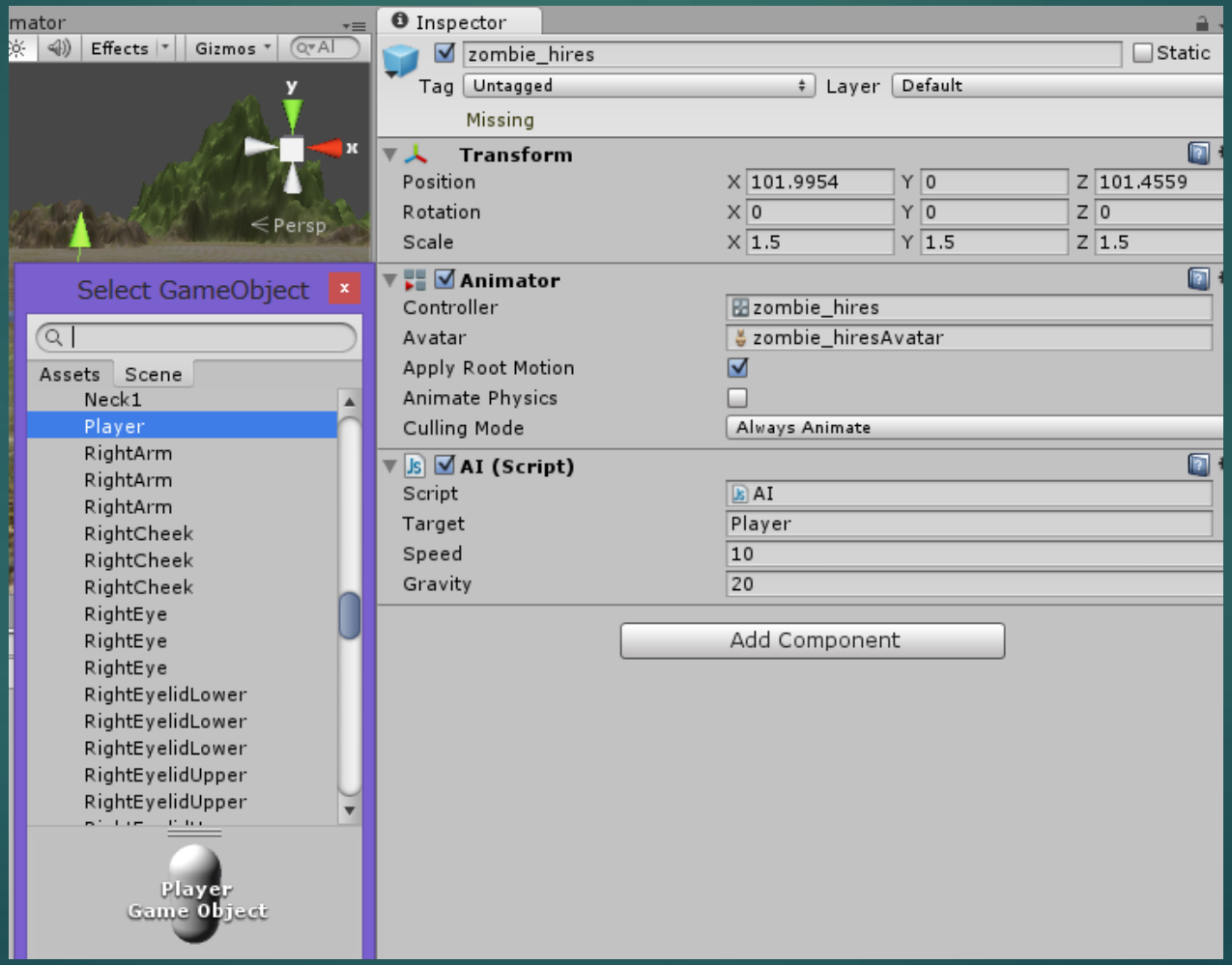
ゾンビのInspectorにAI（Script）の項目が追加されればOK！



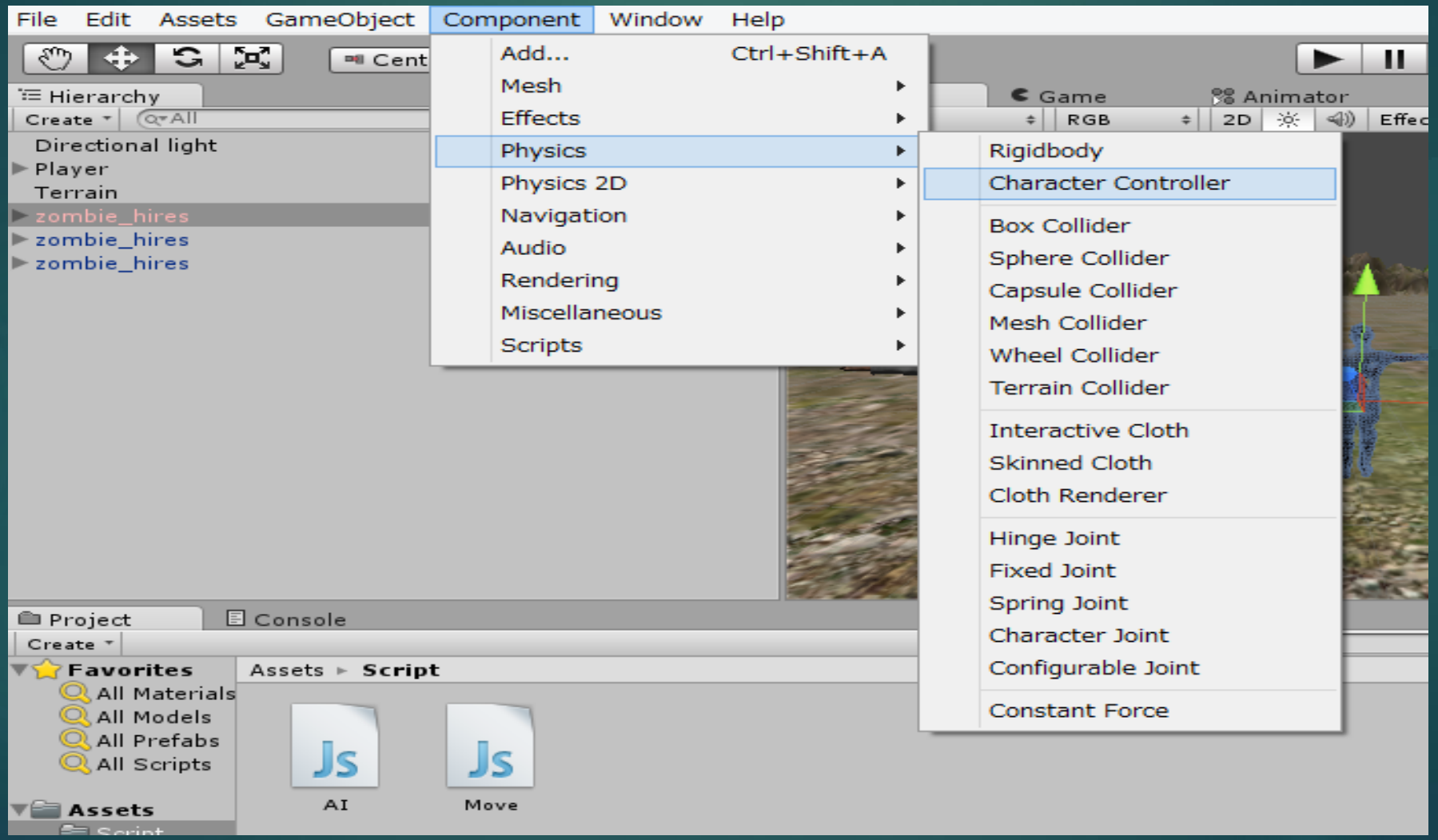



ゾンビのInspectorのAI（Script）のTarget欄の小さな丸をクリック

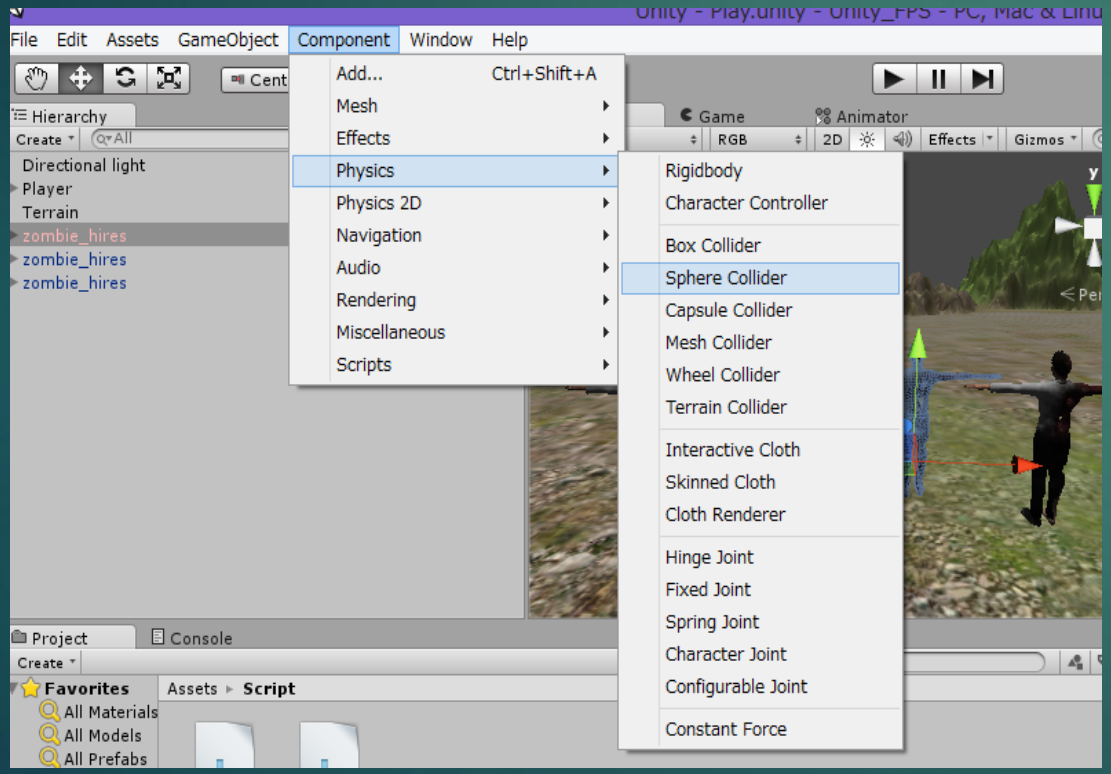
Playerを探してクリックすると、TargetのNoneがPlayerになる。

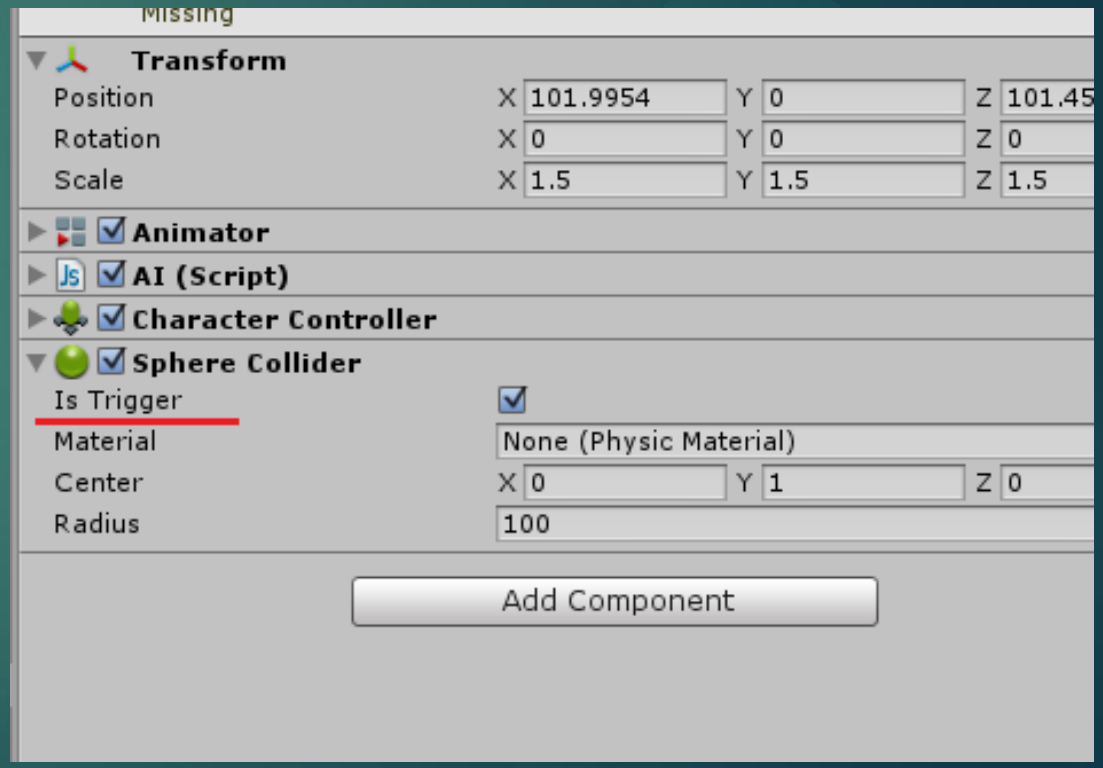


ゾンビを選びながらメニューの「Component」→「Physics」→「Character Controller」をクリック



ゾンビを選びながらメニューの「Component」→「Physics」→
「Sphere Collider」をクリック
ゾンビのInspectorのSphere Collider内のIs Triggerに印をつけて
Radiusを１００、CenterのYを１に変える

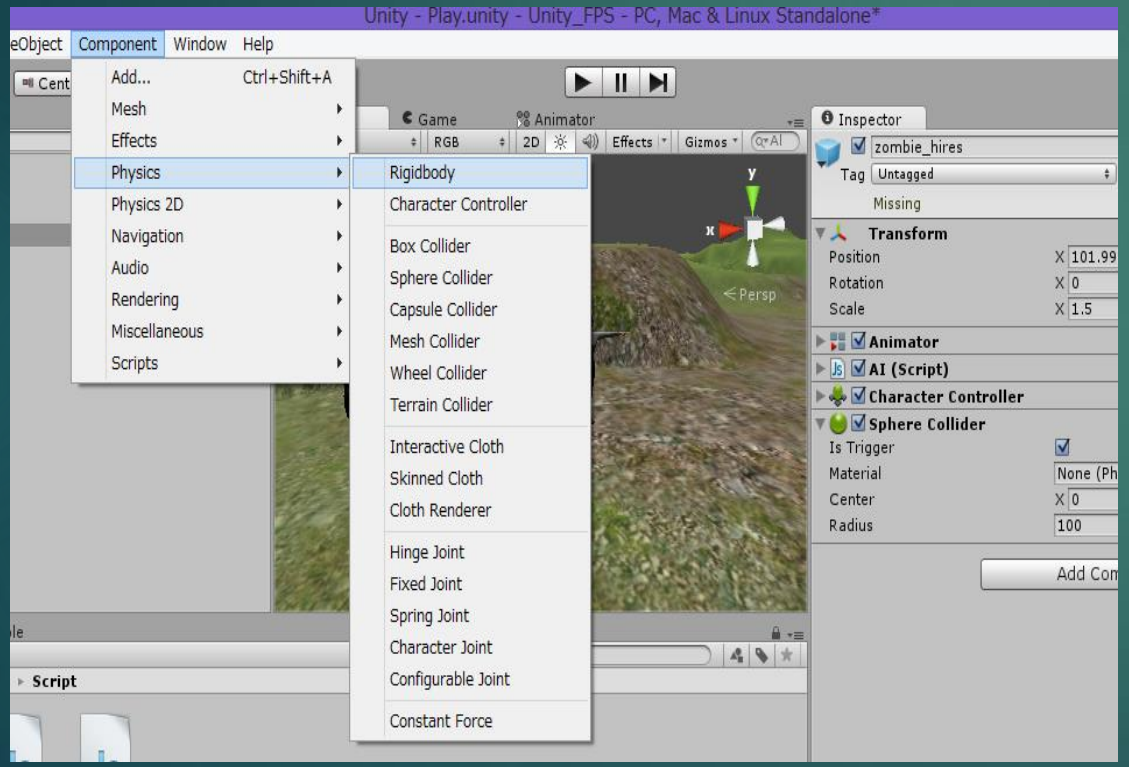


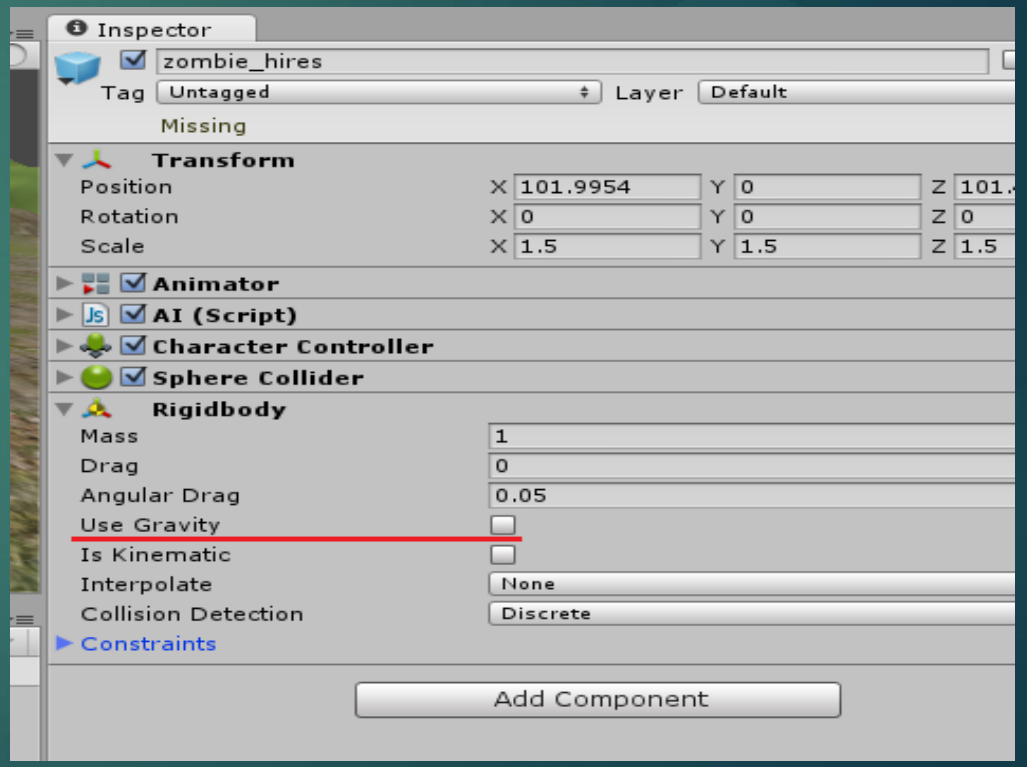



ゾンビを選びながら、メニューの「Component」→「Physics」→「Rigidbody」を選択

そしてゾンビのInspectorのRigidbody欄の

Use Gravityのチェックを外す。

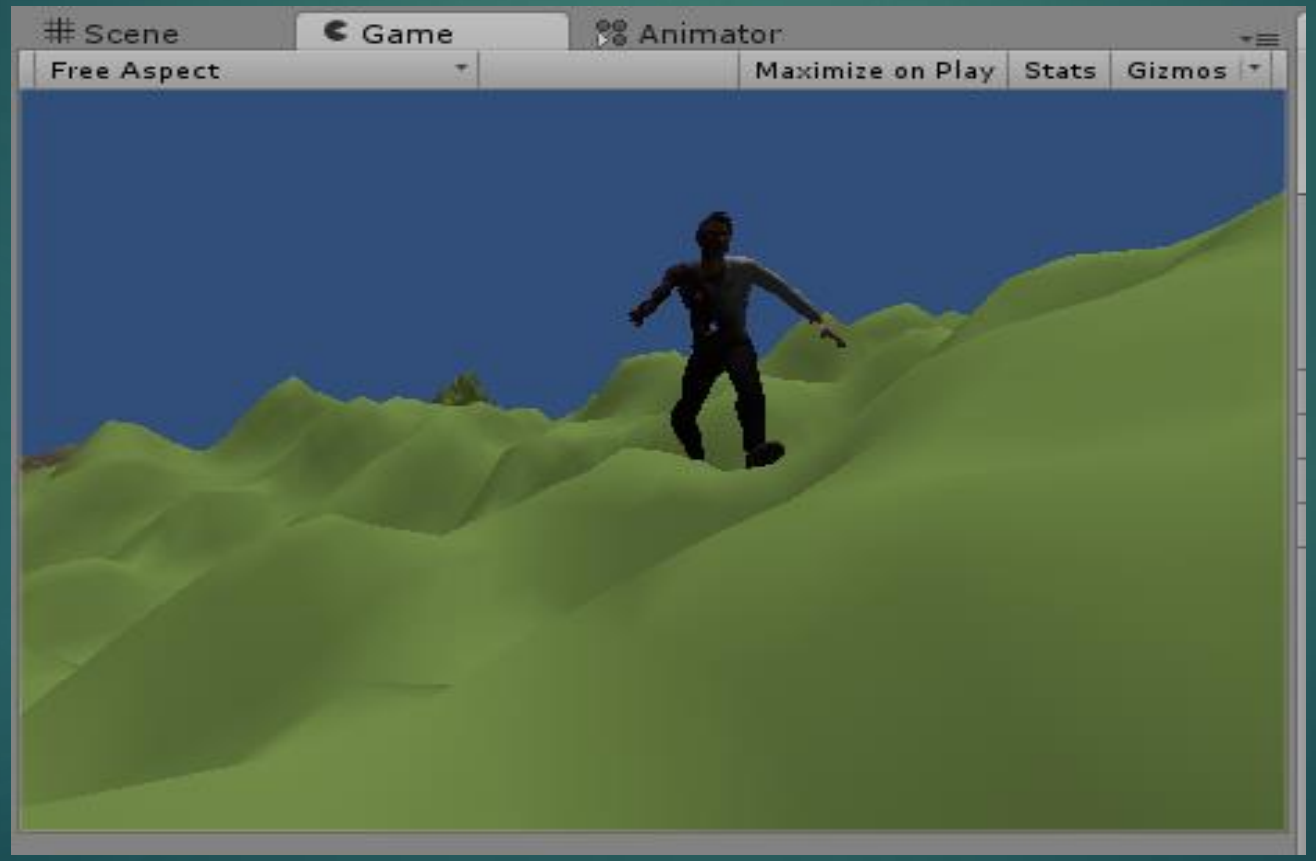


実行してみる・・・

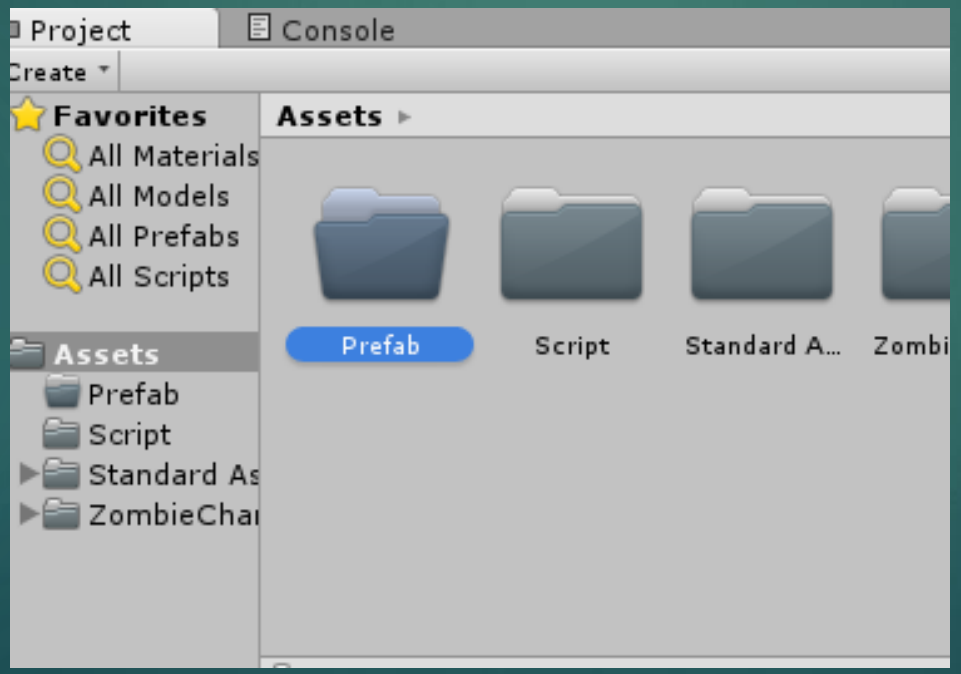
ちゃんと追いかけてきたら成功！

動かなかったらどこかミスしてるはず！

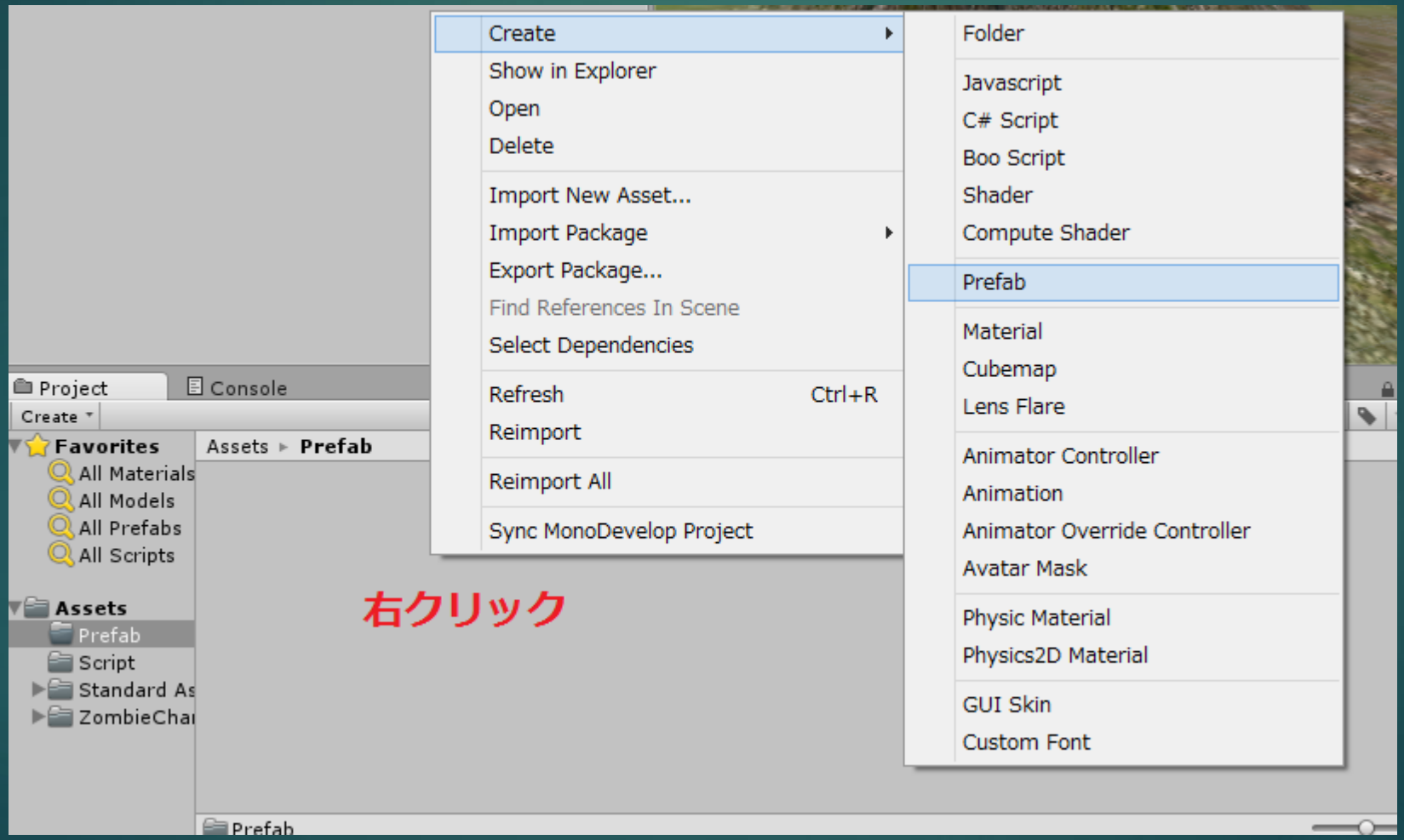


# １２．プレファブの作り方

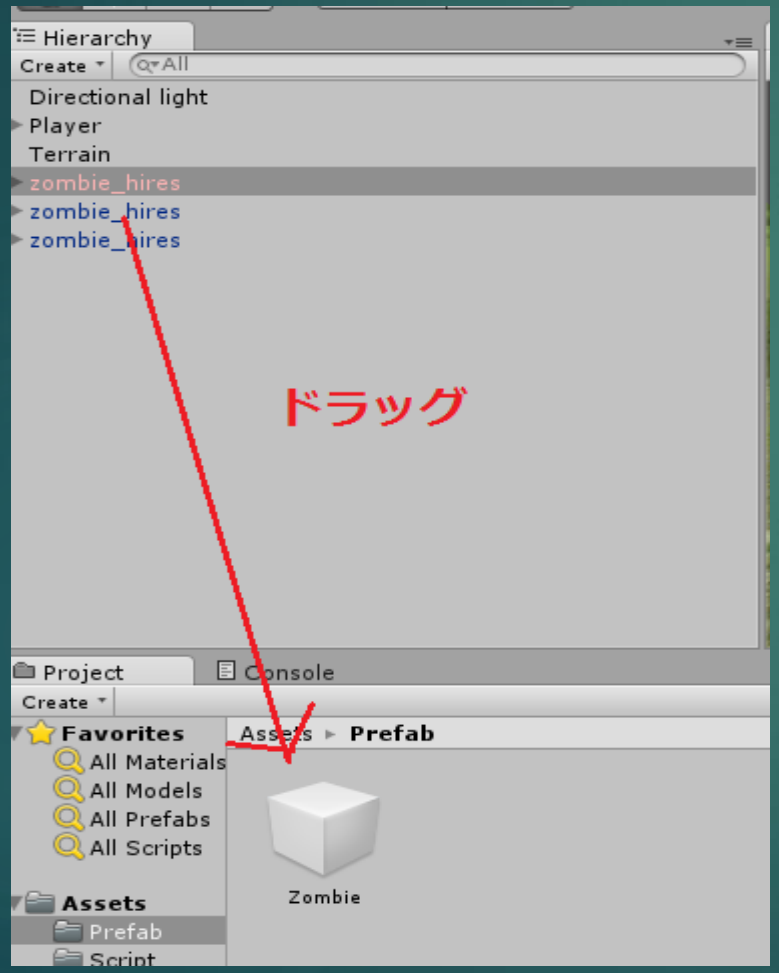
ここまでたぶん１体のゾンビを作ってきたけどこれを何体もつくるのは
めんどい→プレファブ使う
ひとまずプレファブのフォルダを作る。

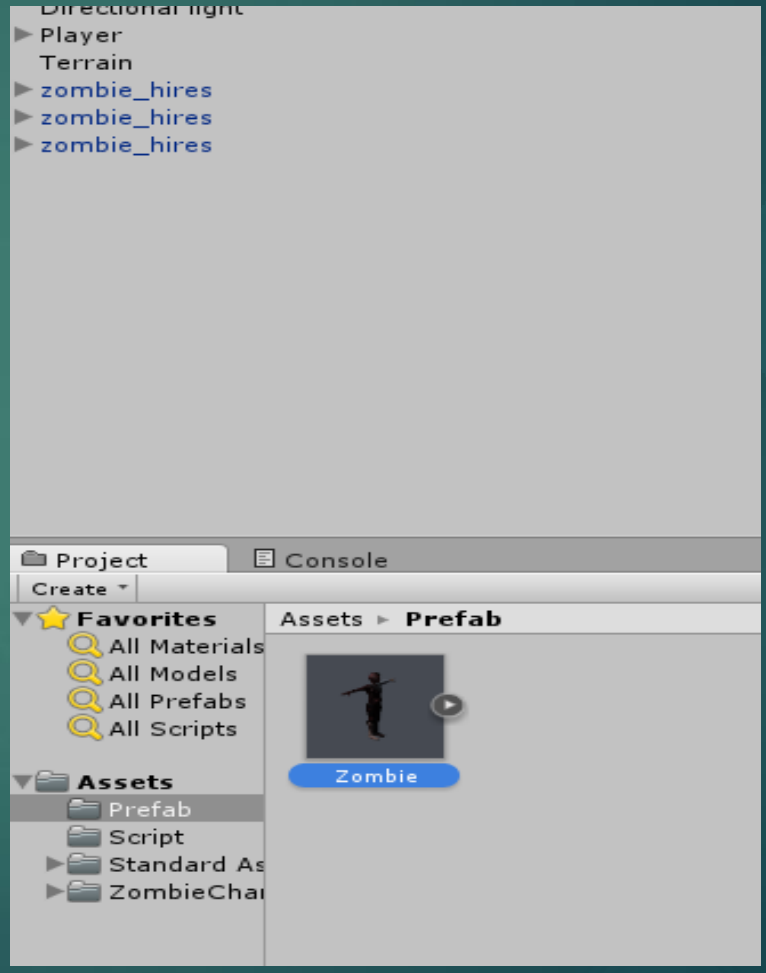


Prefabフォルダ内で右クリック→「Create」→「Prefab」
名前は「Zombie」にする。

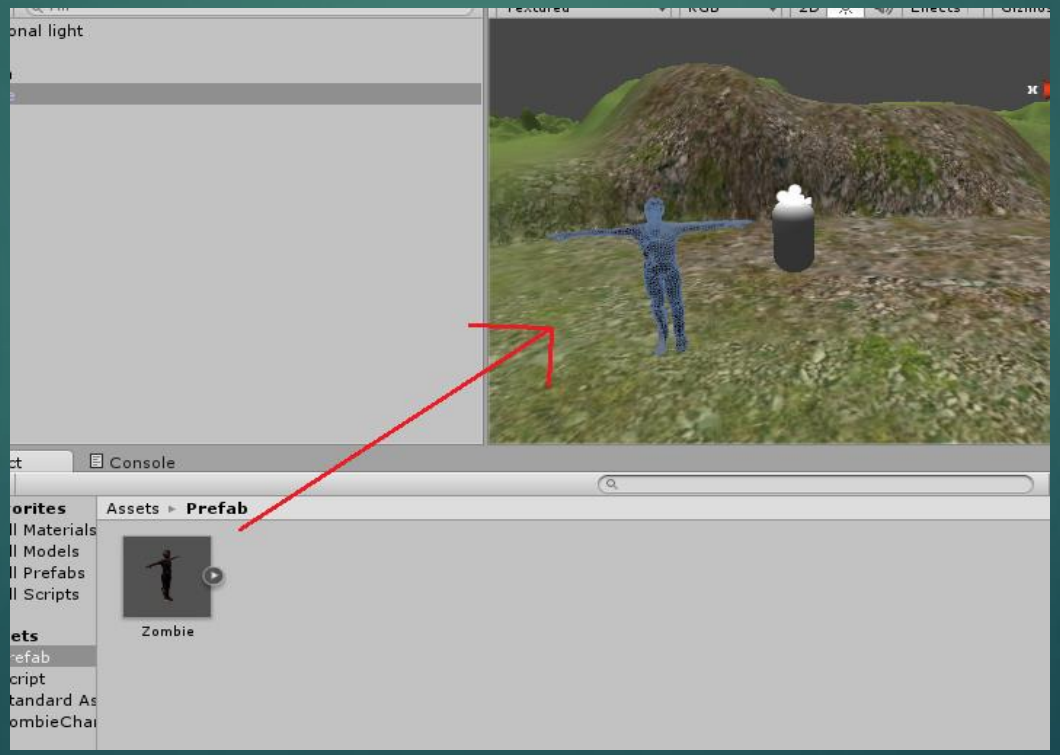


さっきつくったゾンビをプレファブの「Zombie」にドラッグ
これでゾンビのプレファブ完成



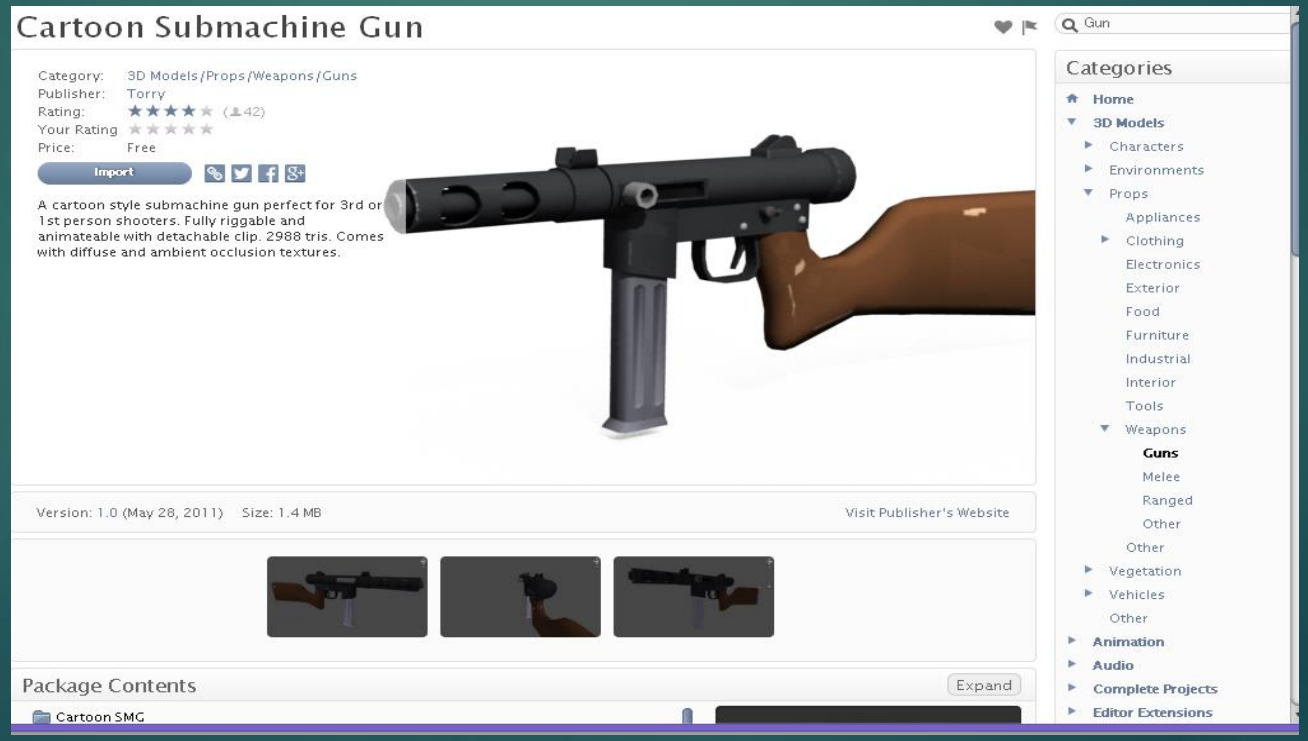
今、Scene上にいるゾンビ達はいらないので消す。
これからゾンビを使うときはプレファブの「Zombie」をScene上へドラッグすることで、さっき作った追いかけてくるゾンビが何体も簡単にできる！プレファブの中身を変えれば、Scene上にいるプレファブから作ったオブジェクトの中身も勝手に変わる！これがプレファブの特徴！



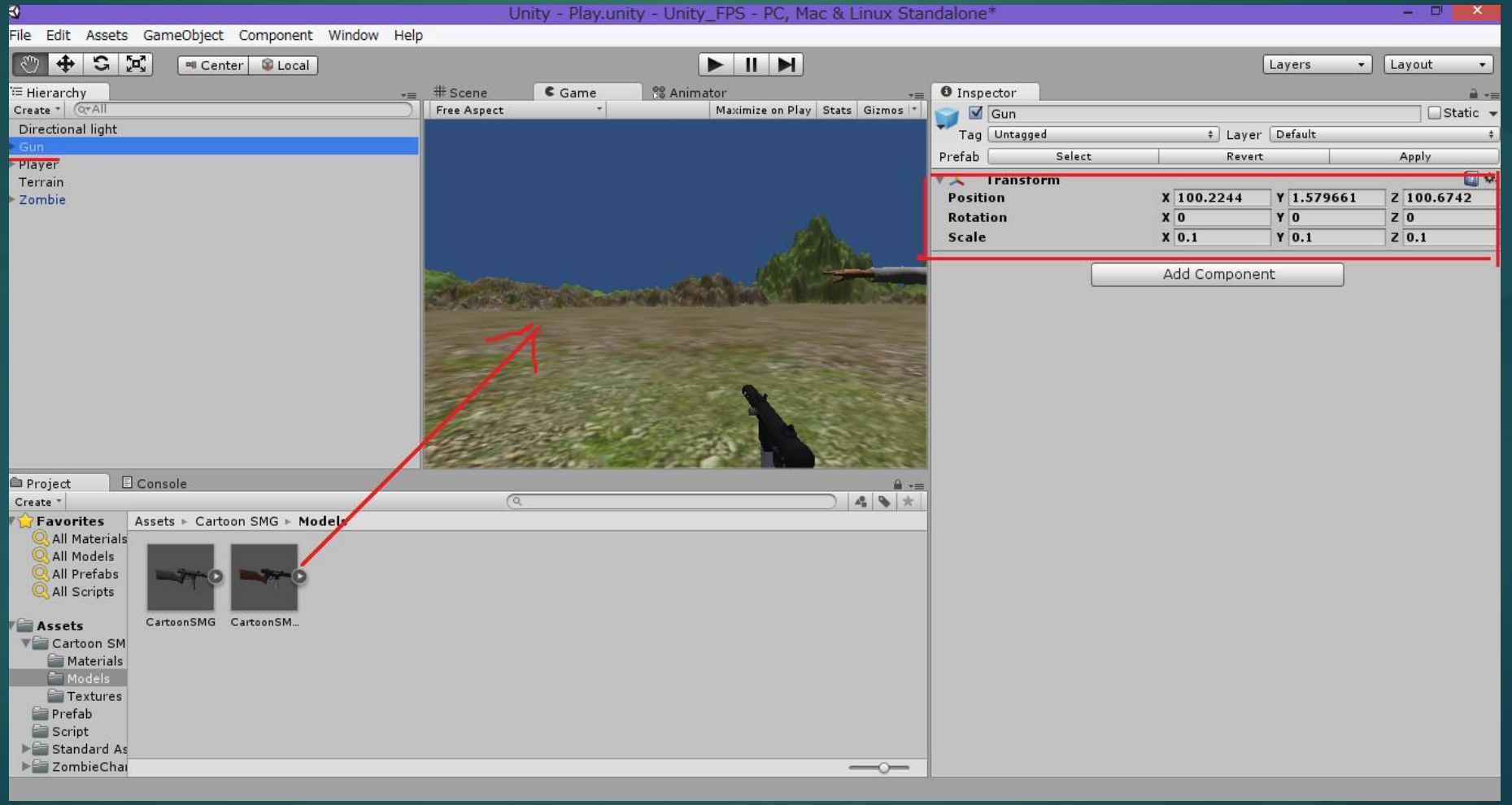
# １３．銃を持たせて弾発射

またAsset Storeからとってくる。

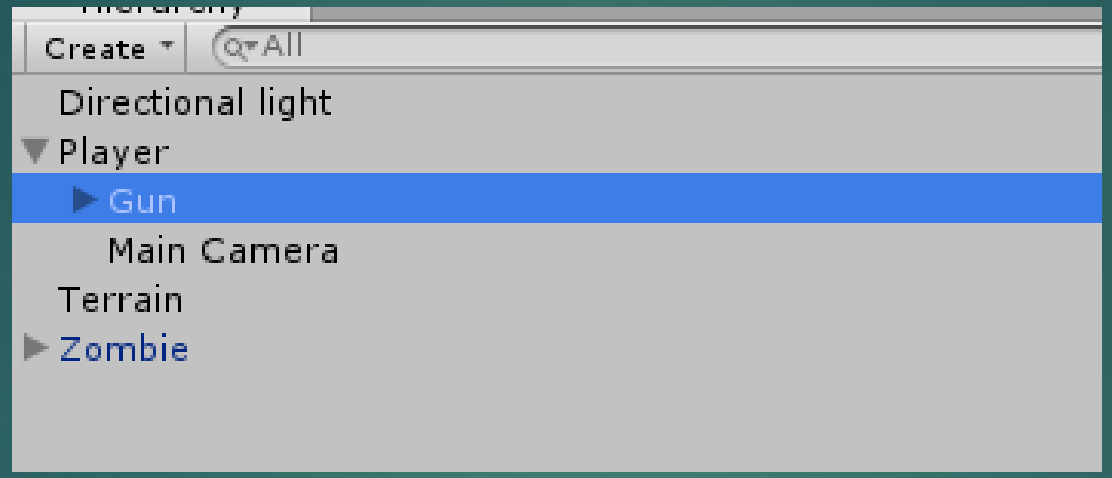
「Gun」で検索したら出てくる。今回はこれをImportする。



銃をImportしたらプレファブ化した銃をScene内へドラッグし、名前を「Gun」に変え、大きさを1/10にし、位置をFPSっぽくなるように配置する。



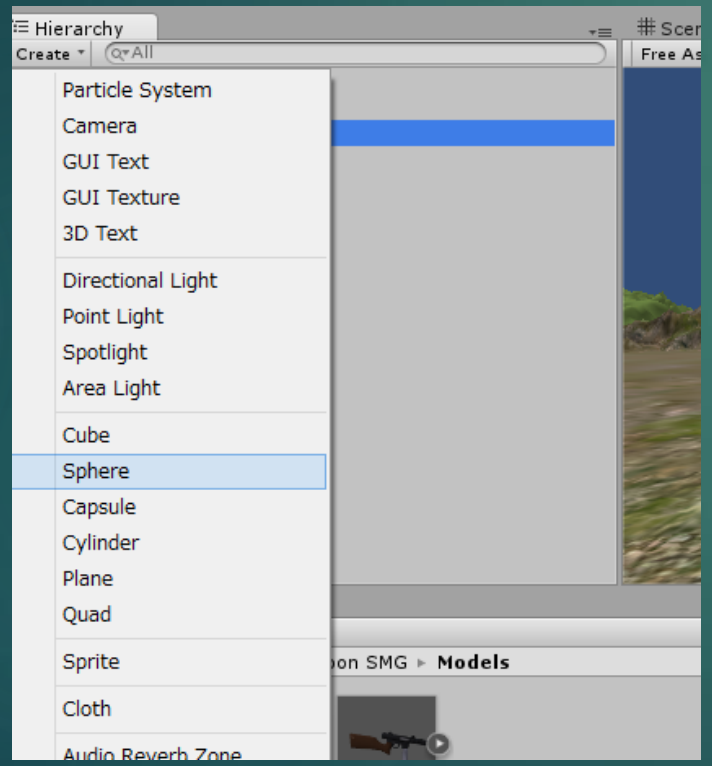
そして、銃をPlayerの一部としたいので、「Gun」を「Player」へドラッグする。

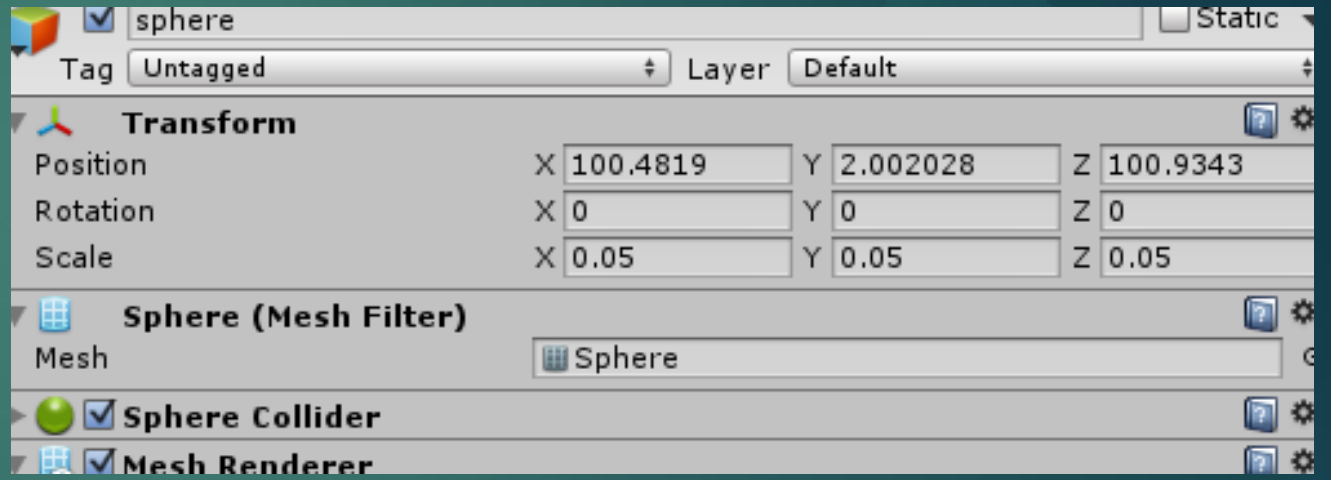



これで持たせるのは完了！

次に弾をつくるので、「Hierarchy」→「Create」→「Sphere」を選択

大きさを1/20にする。

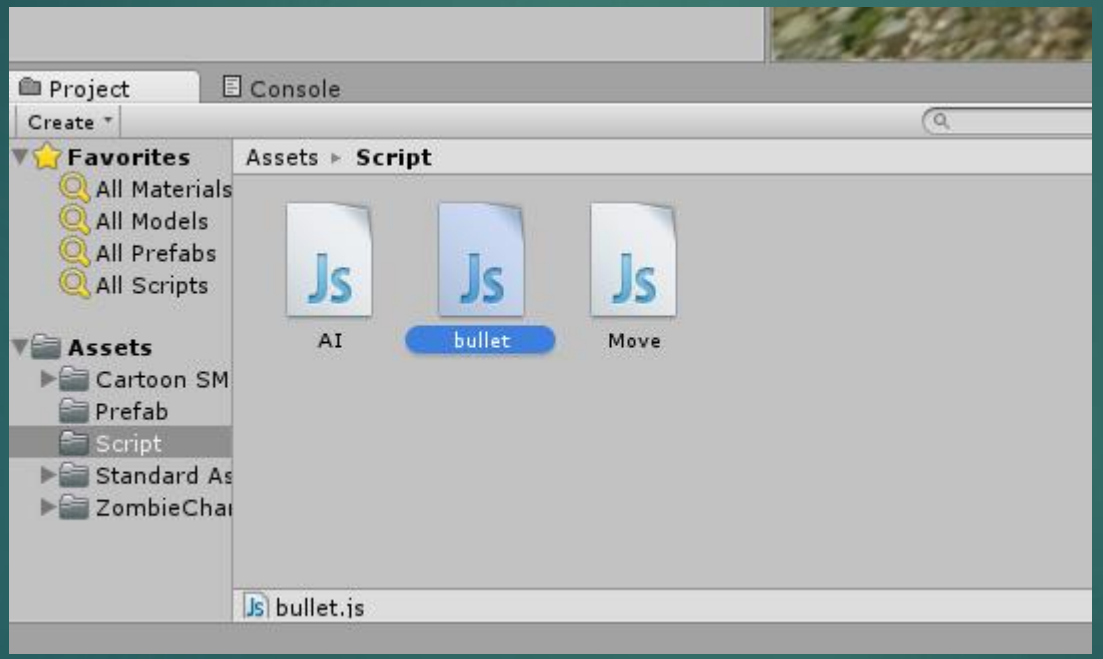


今度は弾の移動するスクリプトを書く。

「Script」フォルダで右クリックしてJavaScriptを作成

名前は「bullet_Move」とする。



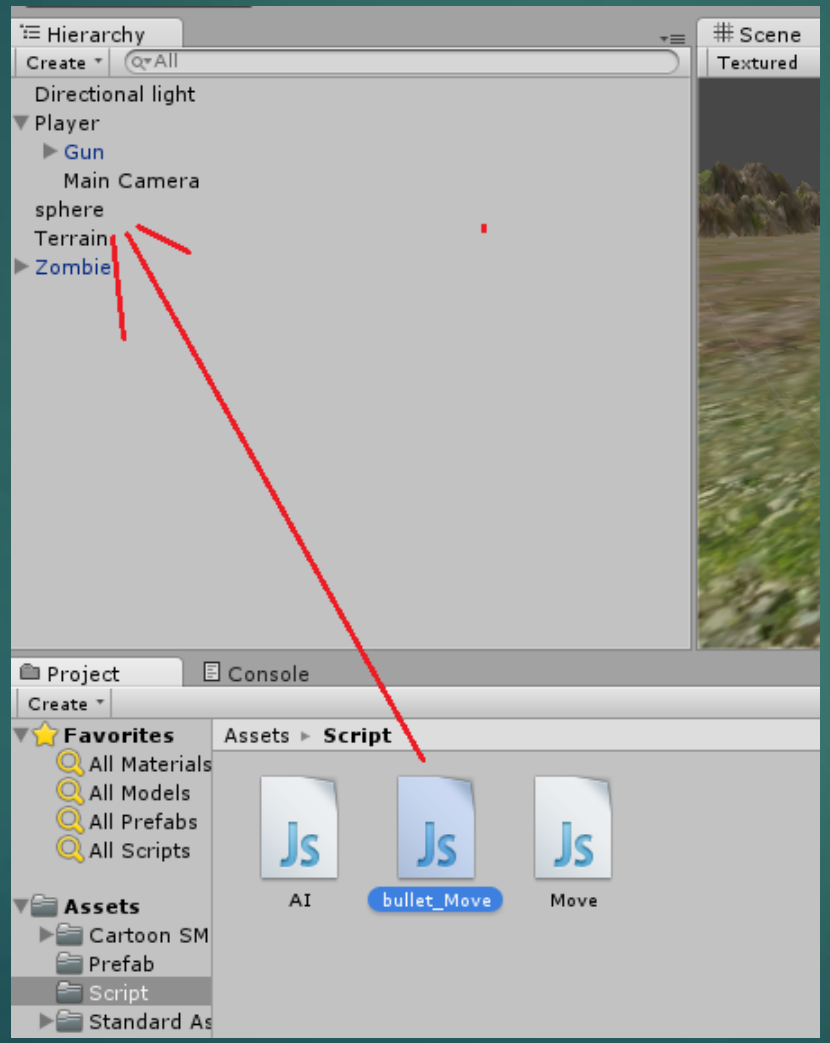
# 「bullet_Move」の中身　translationの値が弾の速さ

Move.js | AI.js | bullet_Move.js

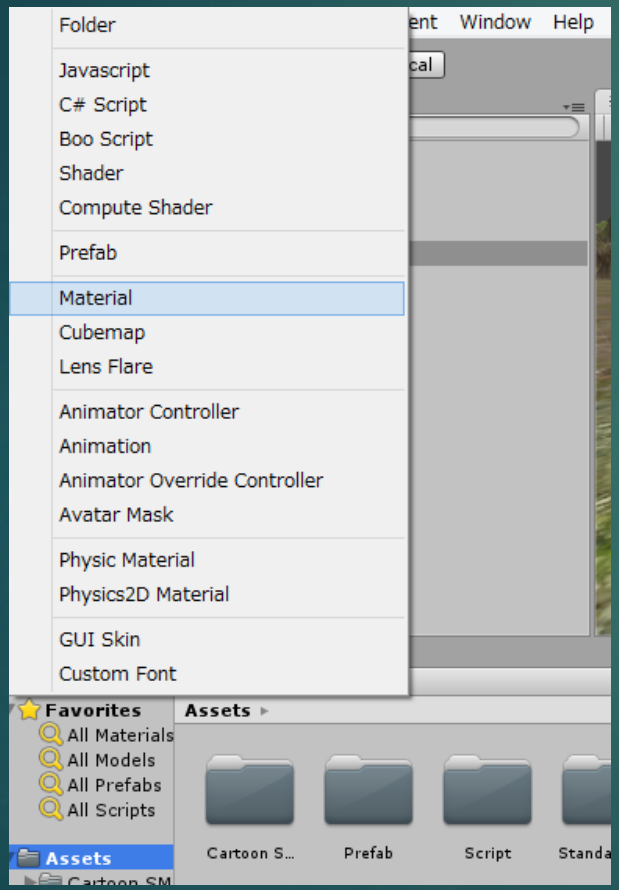
bullet_Move ▸ Awake

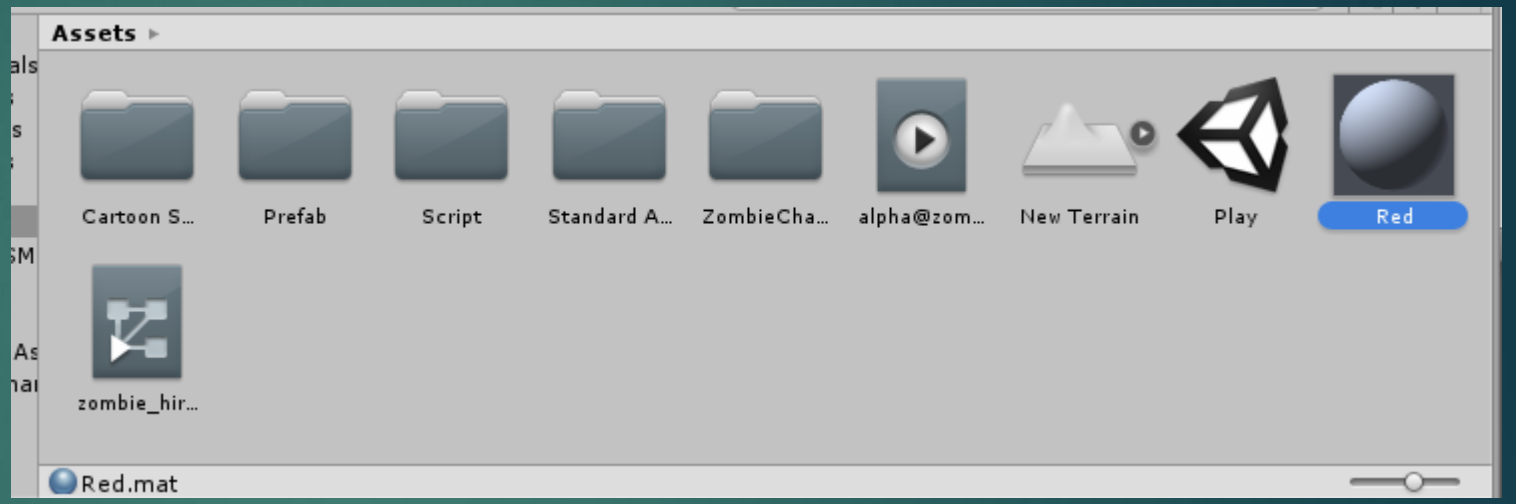
```
1  #pragma strict
2
3  var translation = 0.1;
4
5  function Start () {
6
7  }
8
9  function Update () {
10       transform.Translate(0,0,translation);
11 }
```

これを「Sphere」へドラッグする。



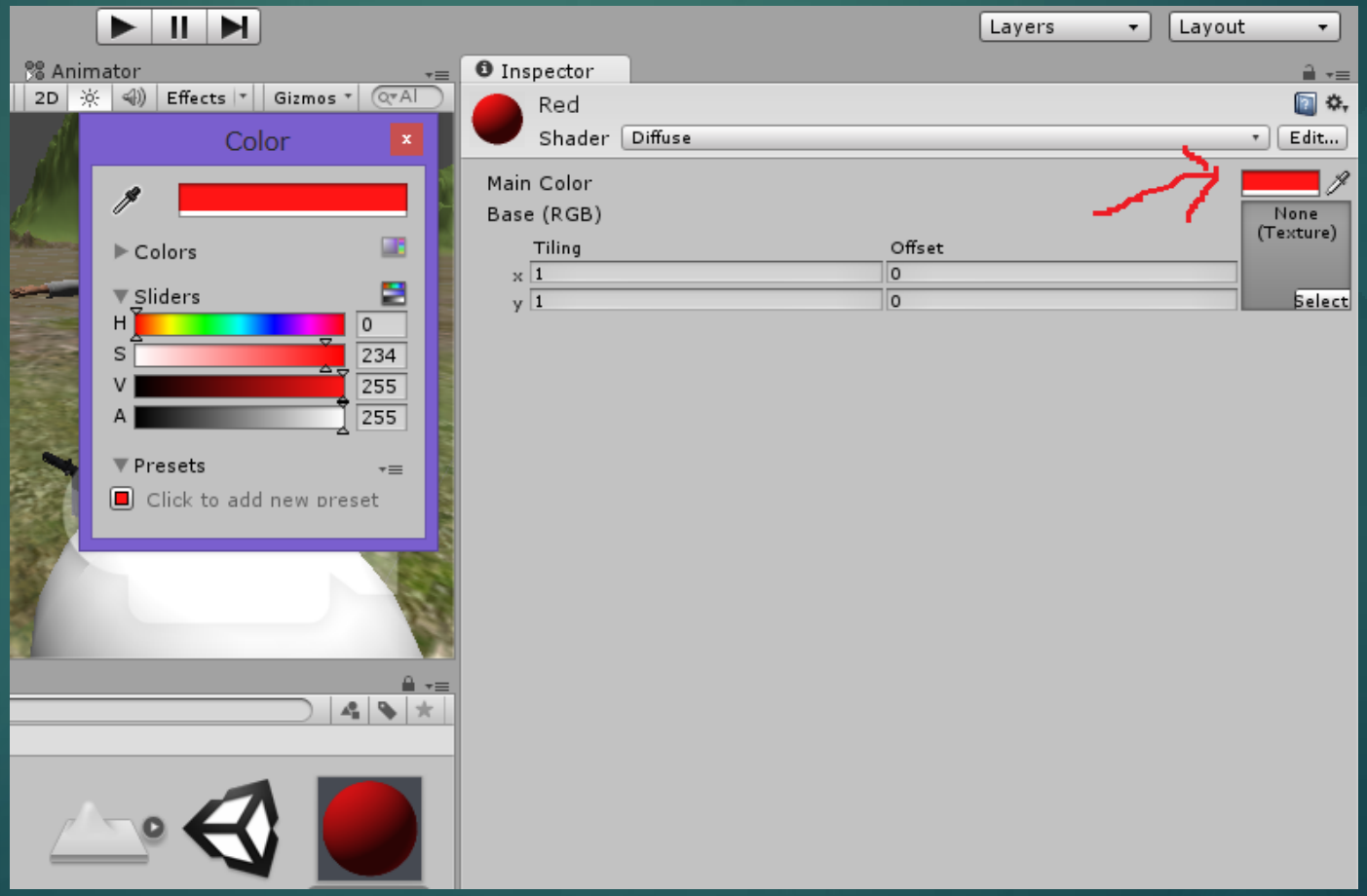
# ProjectのCreateをクリック→Materialをクリック　名前は「Red」にする。



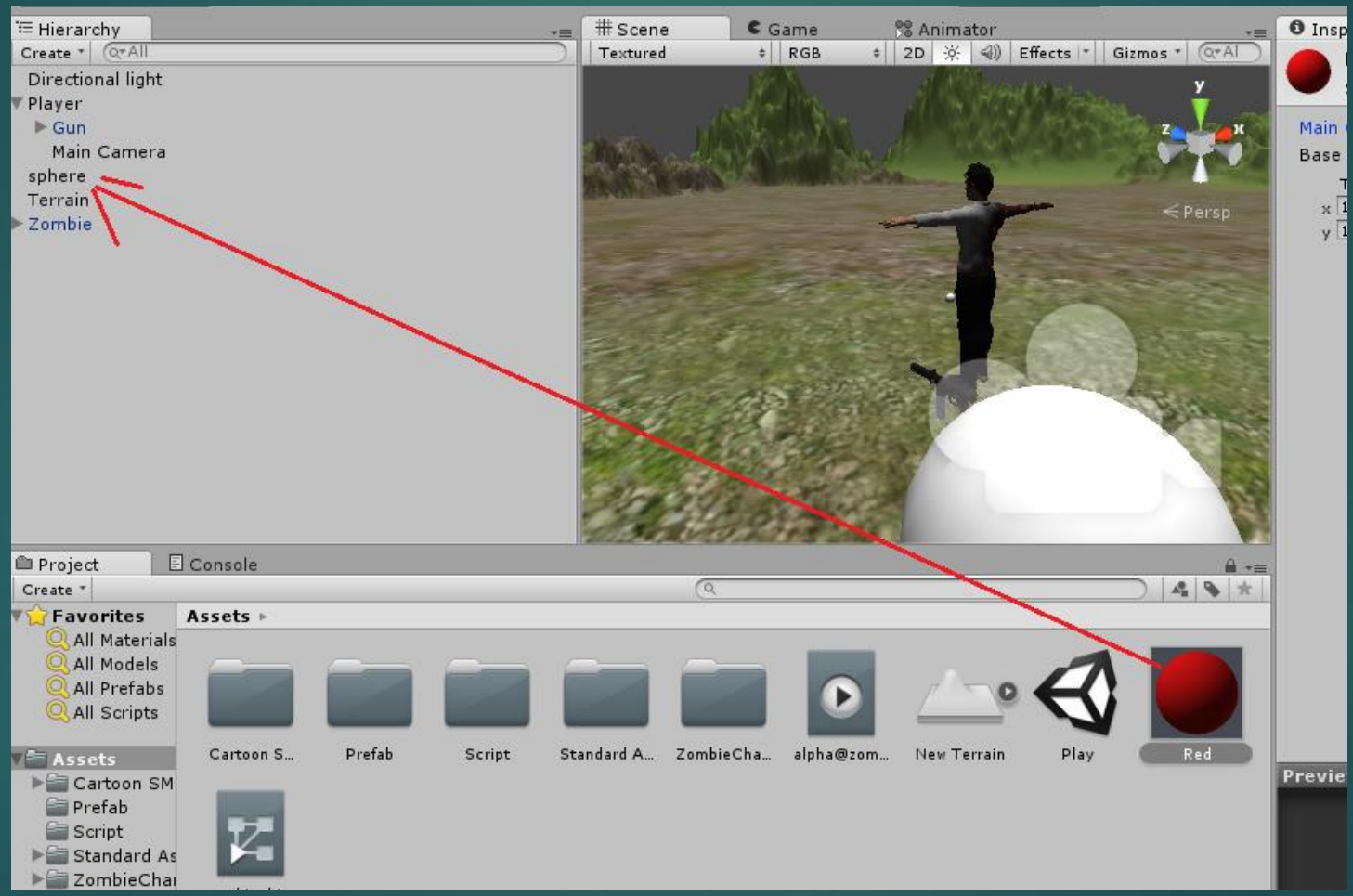



「Red」のInspectorのスポイトマークの隣をクリックすると色が選べるので、赤を選ぶ。

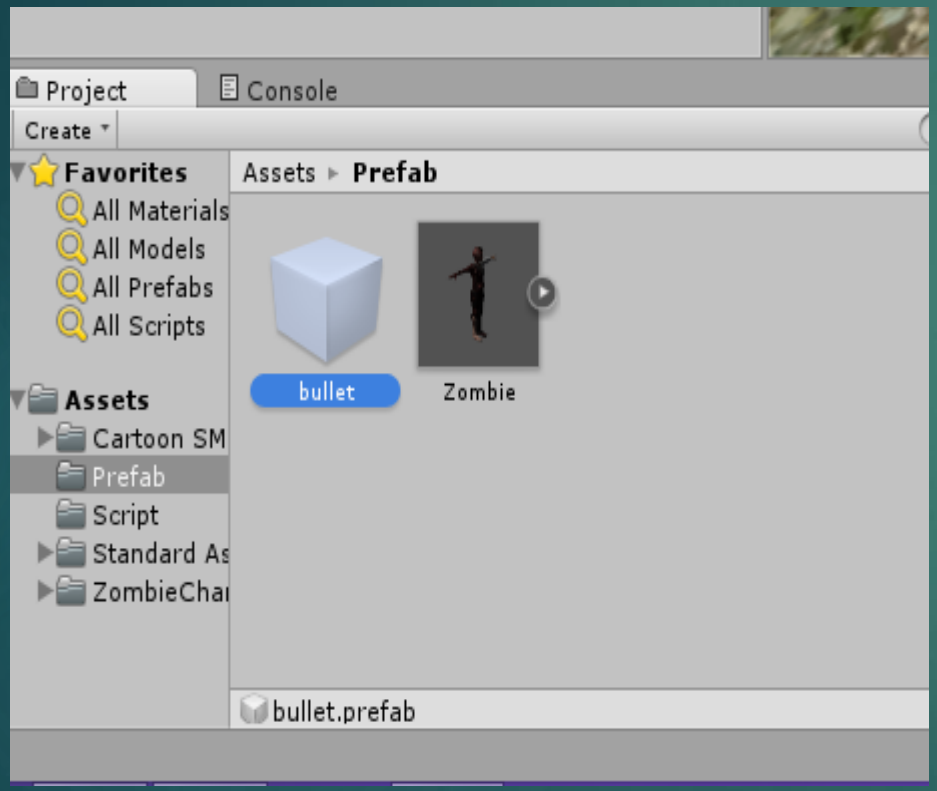
そうすると、Materialの色が赤になる。

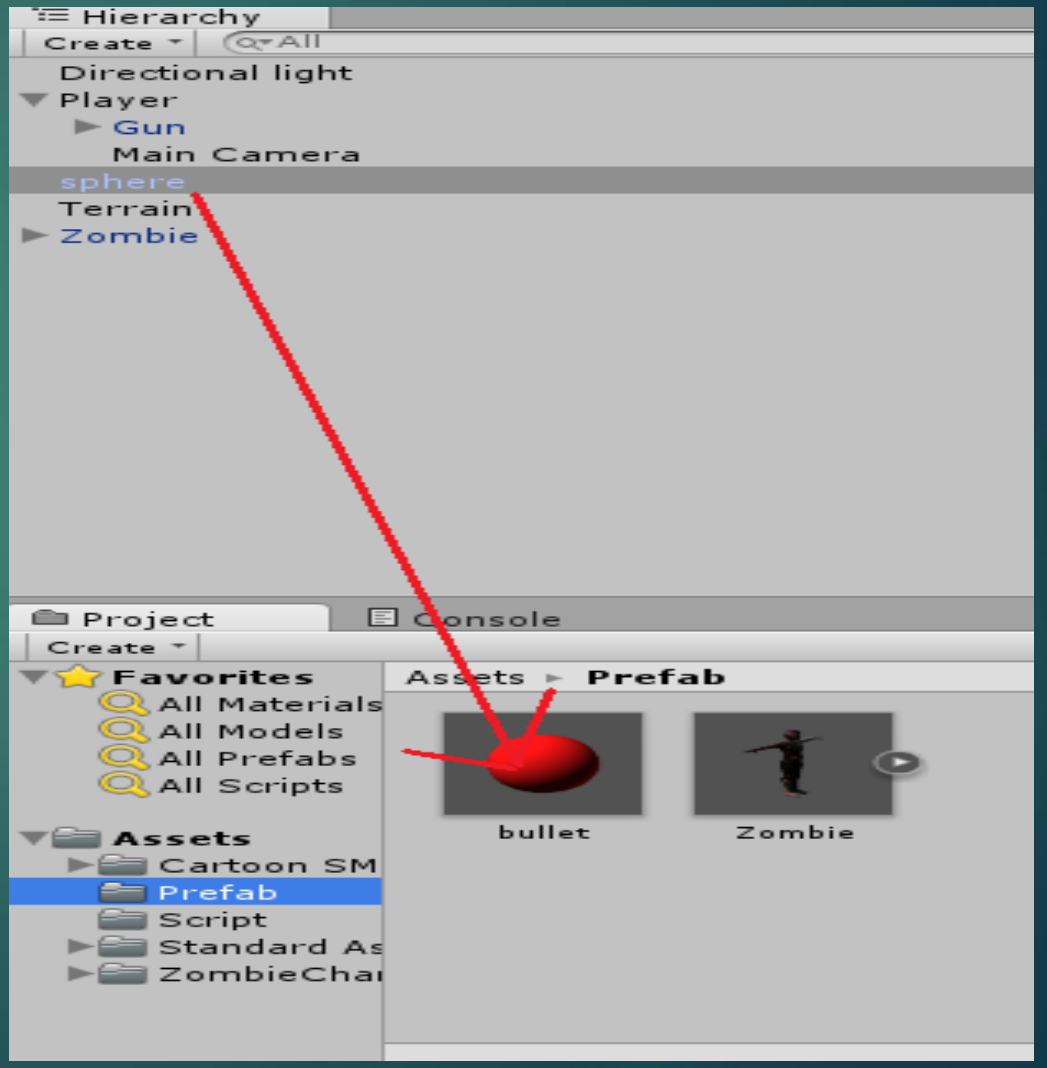


「Red」を「Sphere」にドラッグすることで、赤色に変えることができる。



「Prefab」フォルダに新たにプレファブを作る。名前は「bullet」にする。
「Hierarchy」にある「Sphere」を「bullet」へドラッグする。
そして「Sphere」は削除する。



次に弾を発射するスクリプトを書く。

「Script」フォルダにJavaScriptを作成する。名前は「Shot」とする。

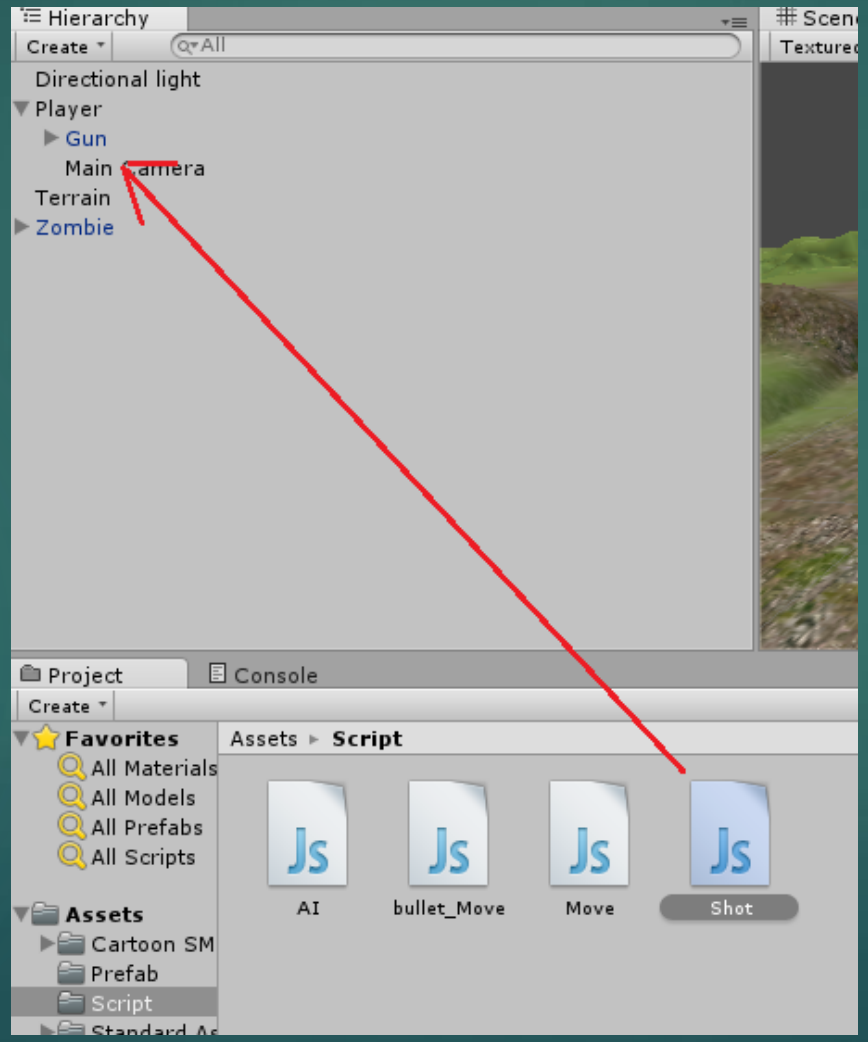
# 「Shot」の中身

Move.js | AI.js | bullet_Move.js | Shot.js

Shot ▸ Awake

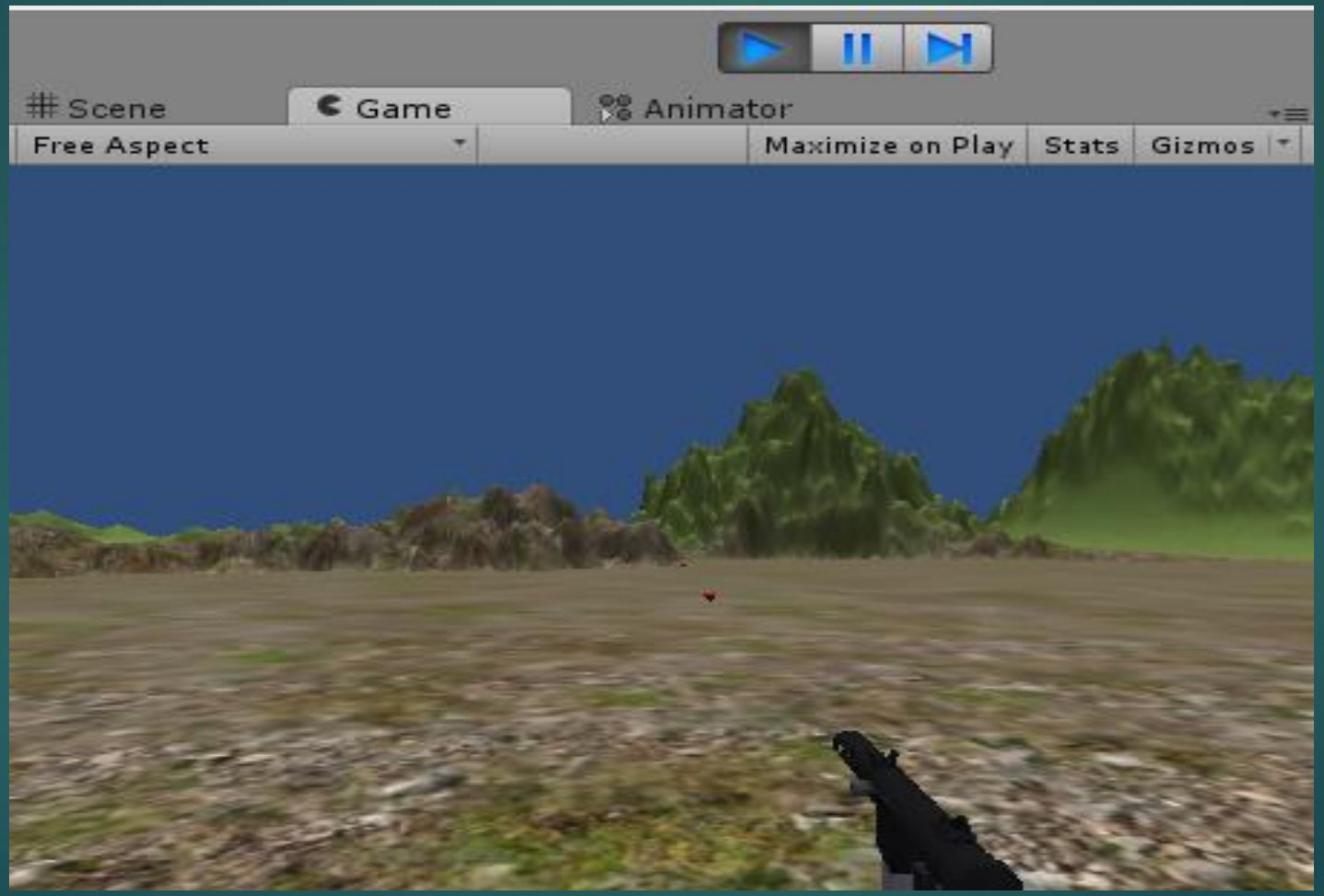
```
1  #pragma strict
2
3  var projectile : GameObject;
4  var fireRate : float = 0.6;
5  private var nextFire : float = 0.0;
6
7  function Start () {
8
9  }
10
11 function Update () {
12      if (Input.GetKey("space")&& Time.time>nextFire) {
13              nextFire = Time.time + fireRate;
14              var clone : GameObject =
15              Instantiate(projectile,transform.TransformPoint(0,1,3),transform.rotation) as GameObject;
16      }
17 }
```

書いたらGunへドラッグする。その後、GunのInspectorの「Shot」のGame Object欄にプレファブの「bullet」を指定する。



実行してみる・・・

スペースを押して弾が発射されれば成功！



# １４．ゾンビと弾を消す

弾にタイマーを付ける＆ゾンビに当たったら消えるようにする。

スクリプト「bullet_Move」に書き加える。

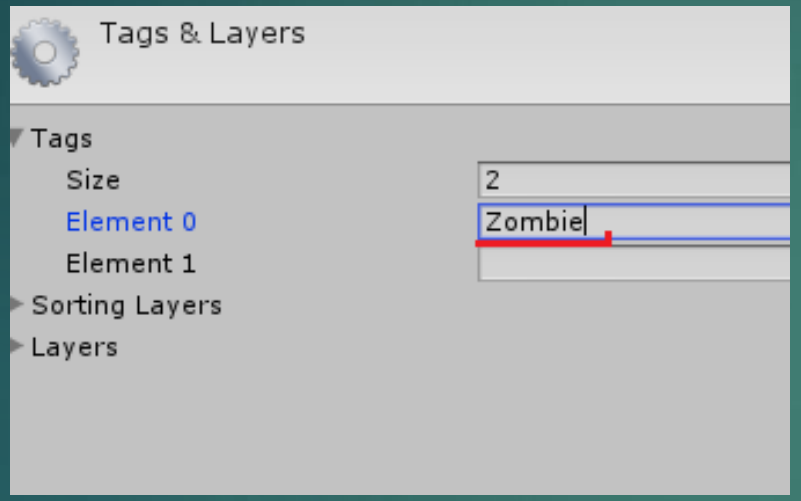
Move.js | AI.js | bullet_Move.js | Shot.js

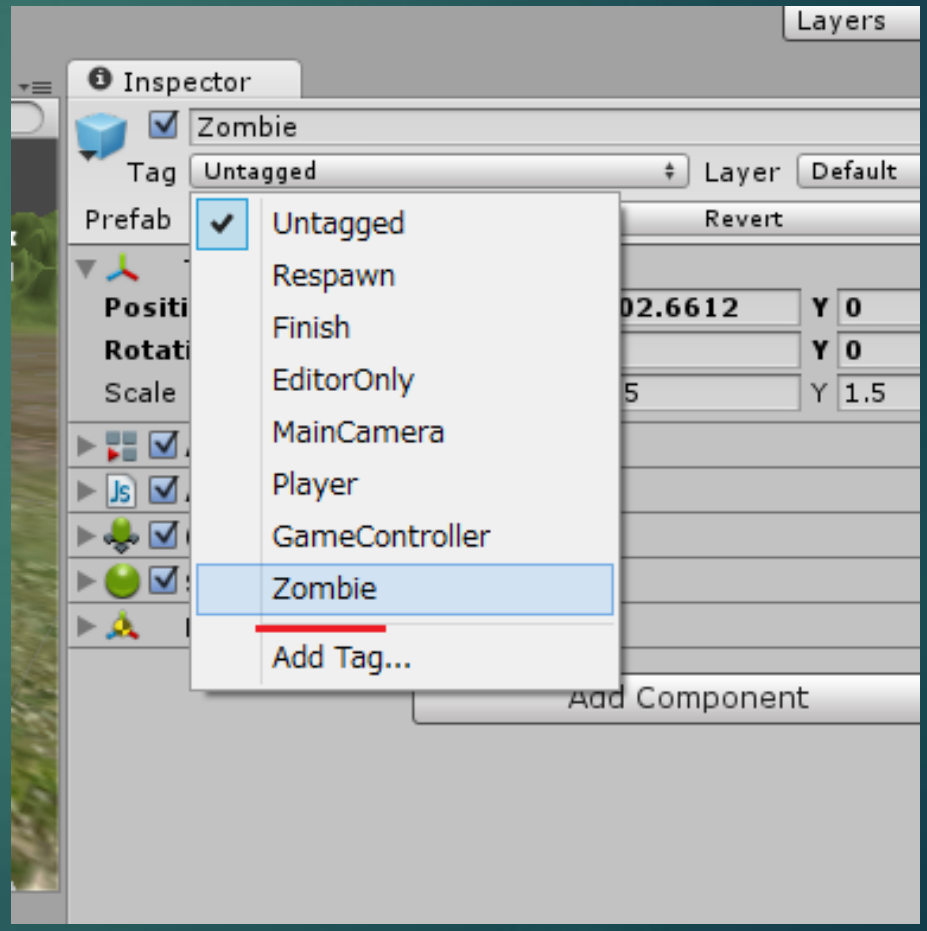
bullet_Move ▸ OnCollisionEnter

```
#pragma strict

var translation = 0.5;
var timer : float = 1.0;

function Start () {

}

function Update () {
	timer -= Time.deltaTime;
	transform.Translate(0,0,translation);
	if (timer<=0) {
		Destroy(gameObject);
	}
}

function OnCollisionEnter(collisionInfo : Collision) {
	if (collisionInfo.gameObject.tag == 'zombie') {
		Destroy(gameObject);
	}
}
```

大文字のZでしたm(_ _)m

ゾンビのInspectorのTagをクリック→Add Tag →Element0 をZombieにする→タグにZombieが増えるのでそれを選択→PrefabのZombieに上書きする（Hierarchyでゾンビを書き換えた場合、上書きしとく）




これで弾は無駄なく消えるようになった。

次にゾンビが弾に３回触れたら消えるようにする。

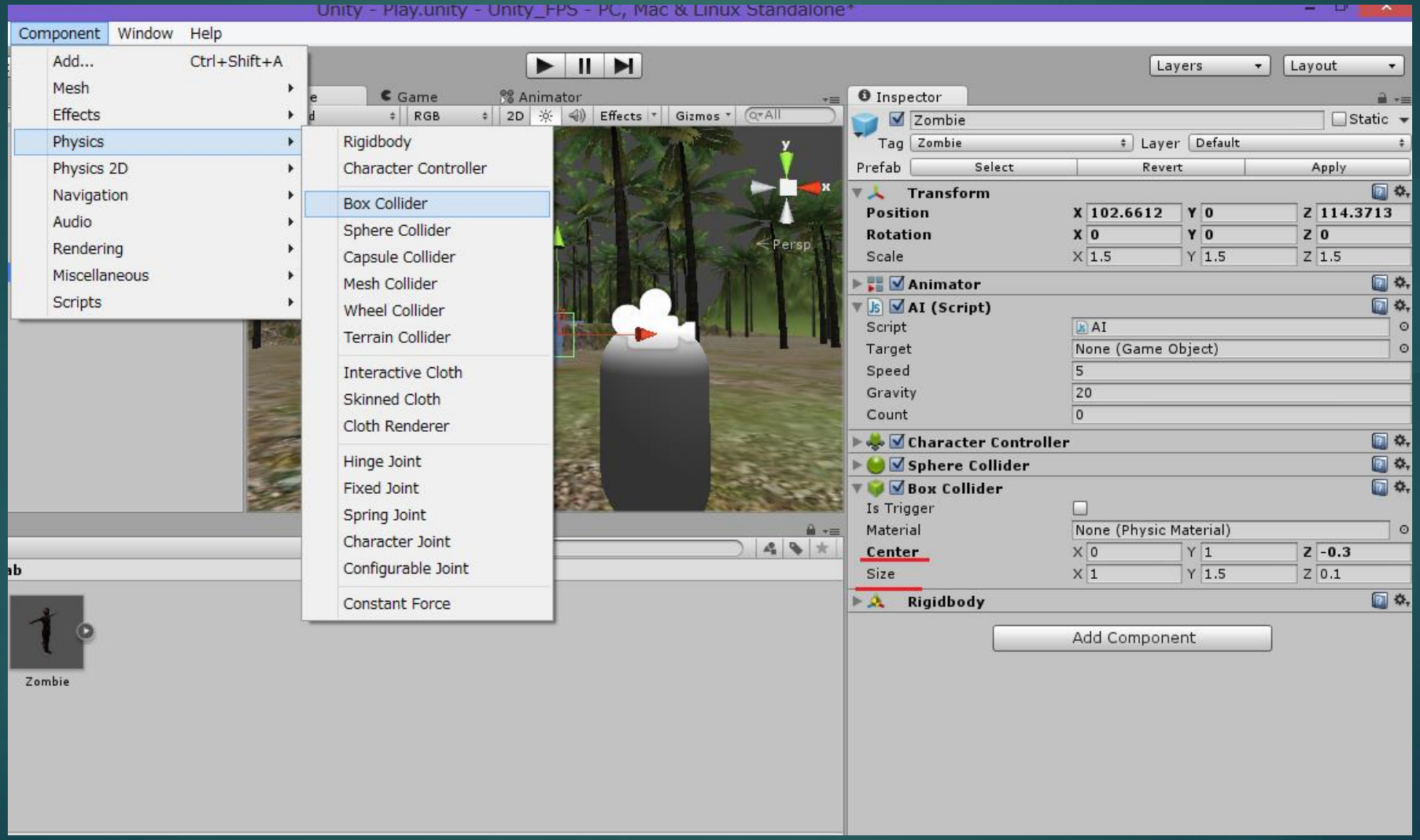
スクリプト「AI」に書き加える。

Move.js | AI.js | bullet_Move.js | Shot.js

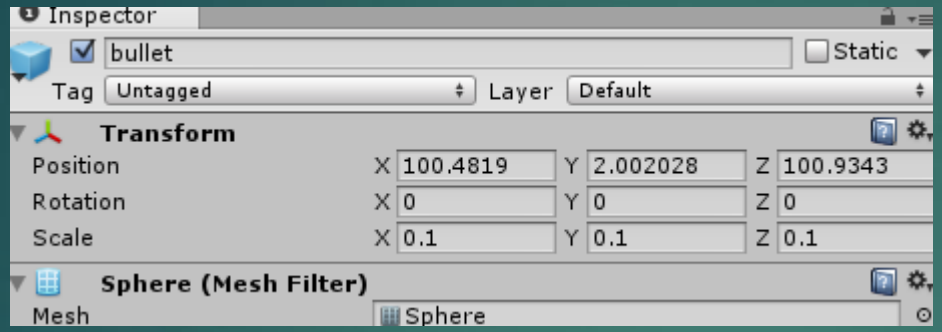
AI ▸ OnCollisionEnter

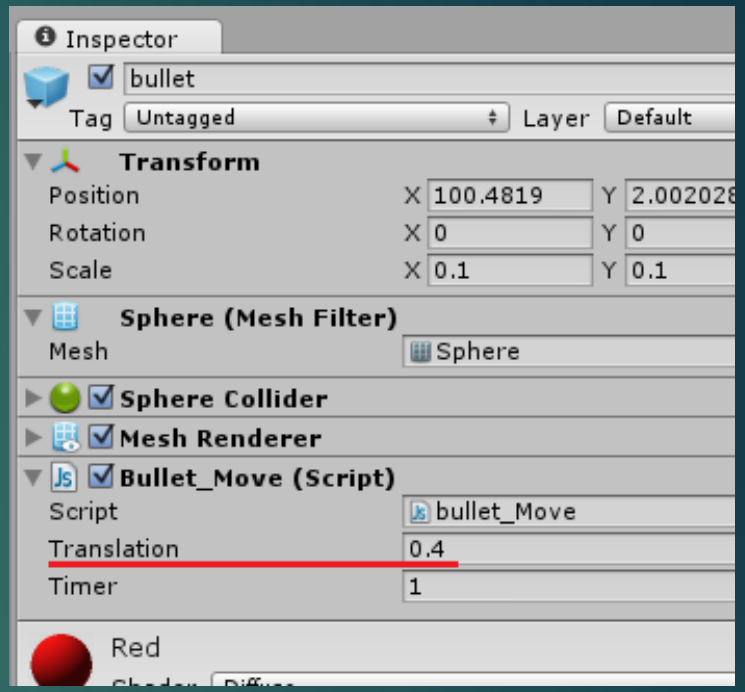
```
                this.transform.position.y,target.transform.position.z);
                this.transform.rotation = Quaternion.Slerp(this.transform.rotation,
                Quaternion.LookRotation(targetDirection-this.transform.position),Time.time*0.1);
                moveDirection += transform.forward*1;
                moveDirection.y-=gravity*Time.deltaTime;
                controller.Move(moveDirection*Time.deltaTime*speed);
            }
        }
}

var count : int = 0;

function OnCollisionEnter(collision : Collision) {
    if (collision.gameObject.name == "bullet(Clone)") {
        count = count + 1;
        if (count == 3 ) {
            Destroy(gameObject);
        }
    }
}

function OnTriggerEnter(c : Collider) {
    if (c.name=='Player') {
        isEnable = true;
        target = c.gameObject;
    }
}

function OnTriggerExit(c : Collider) {
    if (c.name == 'Player'){
        isEnable = false;
    }
}
```

ゾンビを選択したまま、「Component」→「Box Collider」で下の図のような配置と大きさにする。



プレファブのbulletのScaleも少し変える。




これで実行すると、ゾンビが消えてくれるはず・・・

（もし消えなかったら弾のスピードとかを遅くしてみる。

右の値を変えれば変わる！）

いろいろと調整が必要・・・

これで基礎部分が完成！

どこか間違ってたらすいませんm(_ _)m