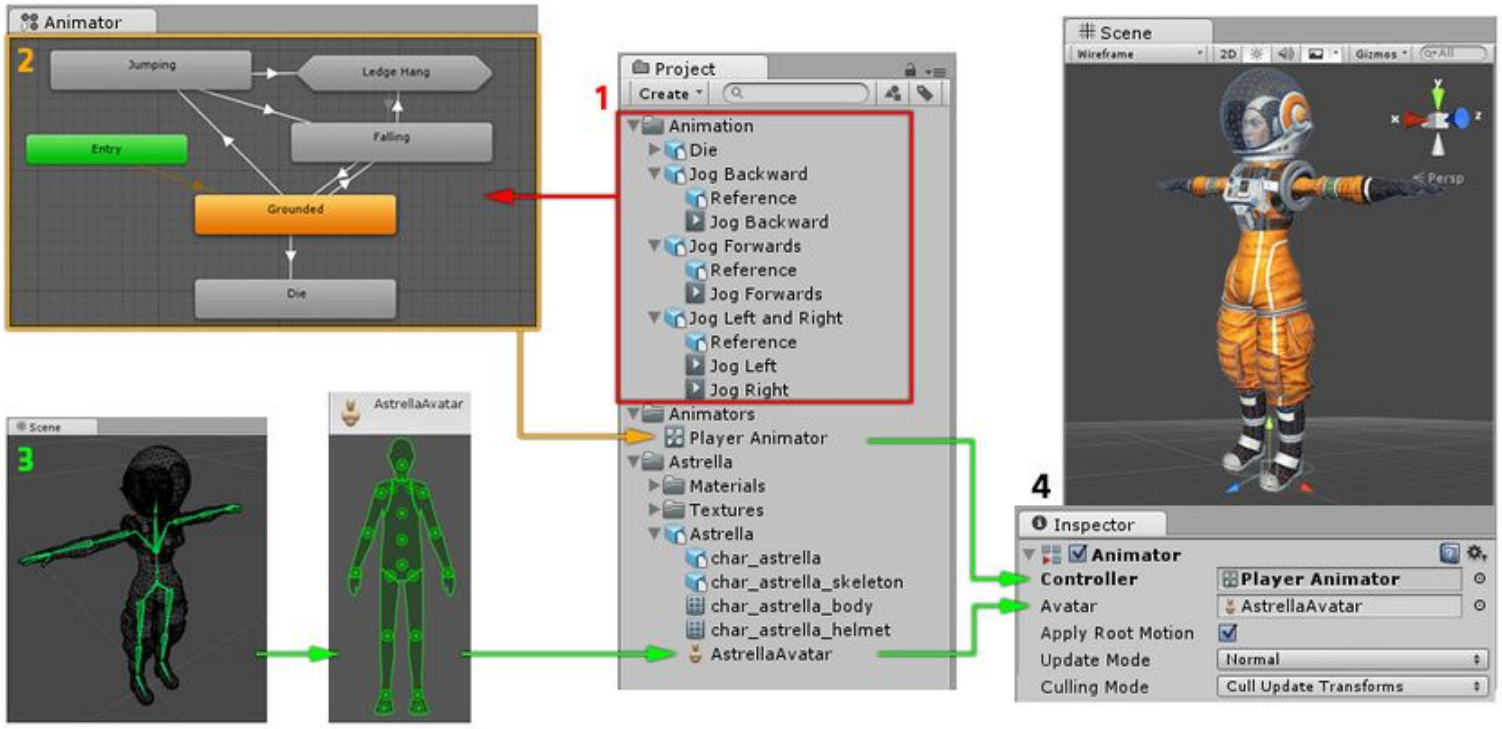
Unity LAB-3
Animator (2D)

Animator controller

Basic

Animation clip: how certain objects should change their position or other properties over time,such as walking,jumping .

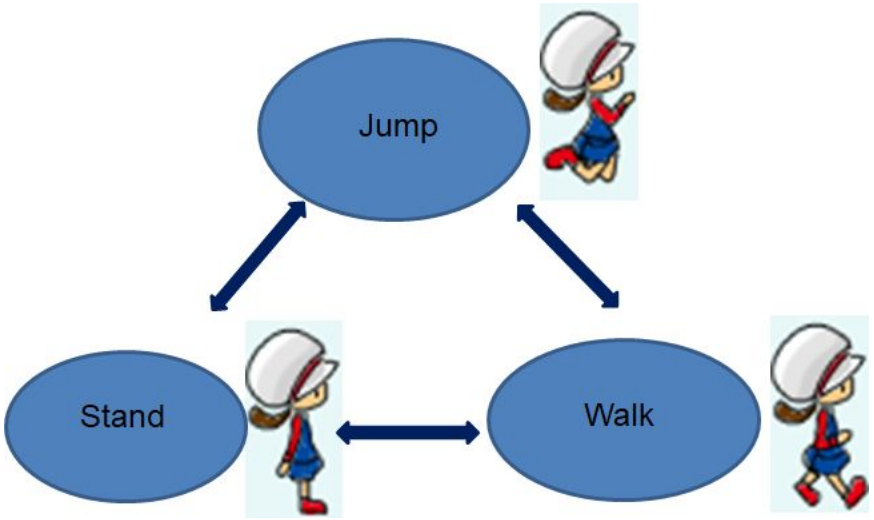
Animator controller: a flowchart-like system that control the transition of different animation clips.



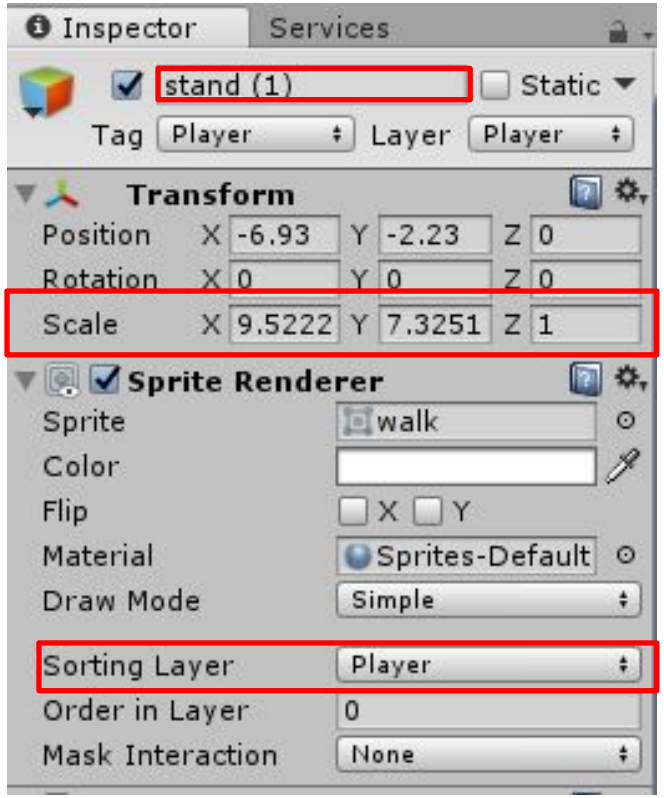
匯入素材

匯入第二週上課用的sprites
+本週上課用的sprites中的
fly.png檔

Import也可以直接從檔案總管
(資料夾)把檔案拖進Editor介
面的Project Tab



物件調整



改成player

大小 改成 X 10 Y 10

前後 設定至背景前

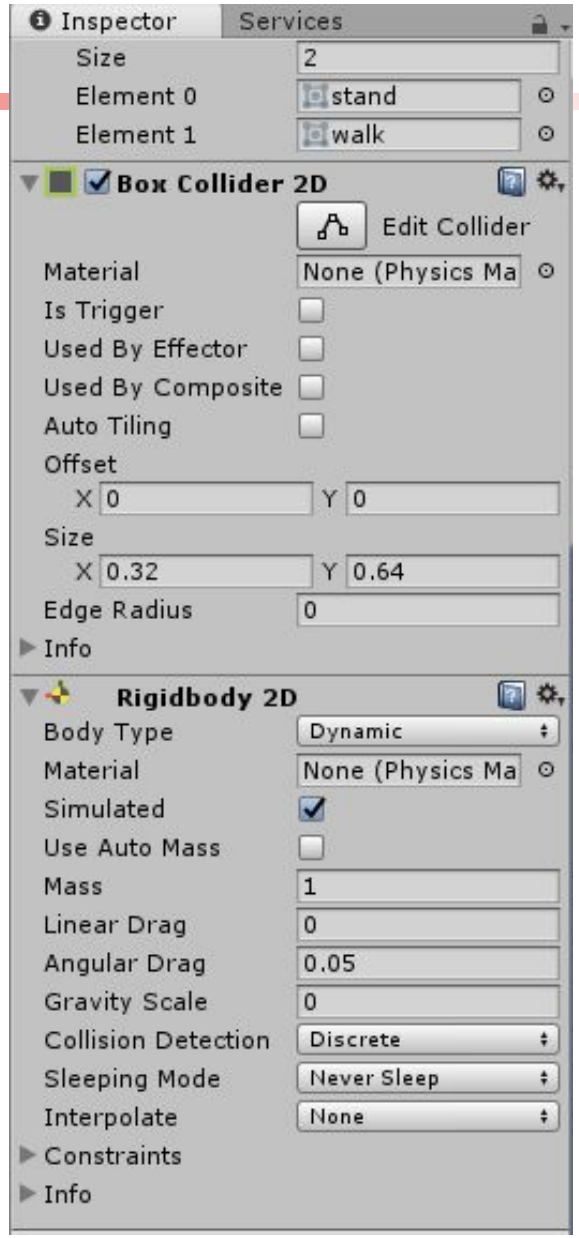
物體性質調整

找到右邊的 Add Component

Box Collider 2D：決定物體屬於可以碰撞的性質

Rigidbody 2D：使物體變成實體，並可以受到重力影響，趨近於真實情況，其中 Gravity 表示受重力大小

將 Gravity Scale 調整成 1



移動及跳躍

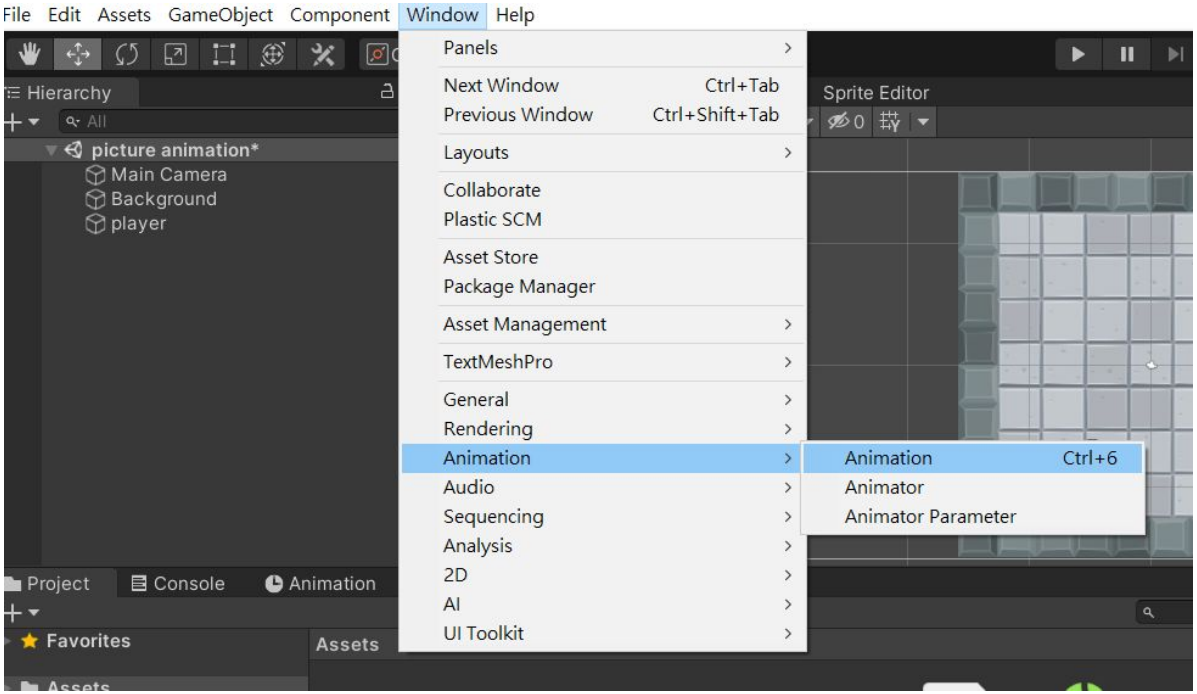
在 player 底下建立 new script 並將其命名為 playerController

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

public class playerController : MonoBehaviour
{
    public float speed = 20;
    private Rigidbody2D rbody2D;
    private bool isOnGround;

    void Start()
    {
        rbody2D = gameObject.GetComponent<Rigidbody2D>();
        isOnGround = false;
    }

    void Update()
    {
        float dirx = Input.GetAxisRaw("Horizontal");
        rbody2D.velocity = new Vector2(dirx * speed, rbody2D.velocity.y);

        if (Input.GetKey(KeyCode.Space) && isOnGround)
        {
            isOnGround = false;
            rbody2D.velocity += new Vector2(0, speed);
        }
    }
    private void OnCollisionEnter2D(Collision2D collision)
    {
        isOnGround = true;
    }
    private void OnCollisionExit2D(Collision collision)
    {
        isOnGround = false;
    }
}

```

加入動畫 Animation clip

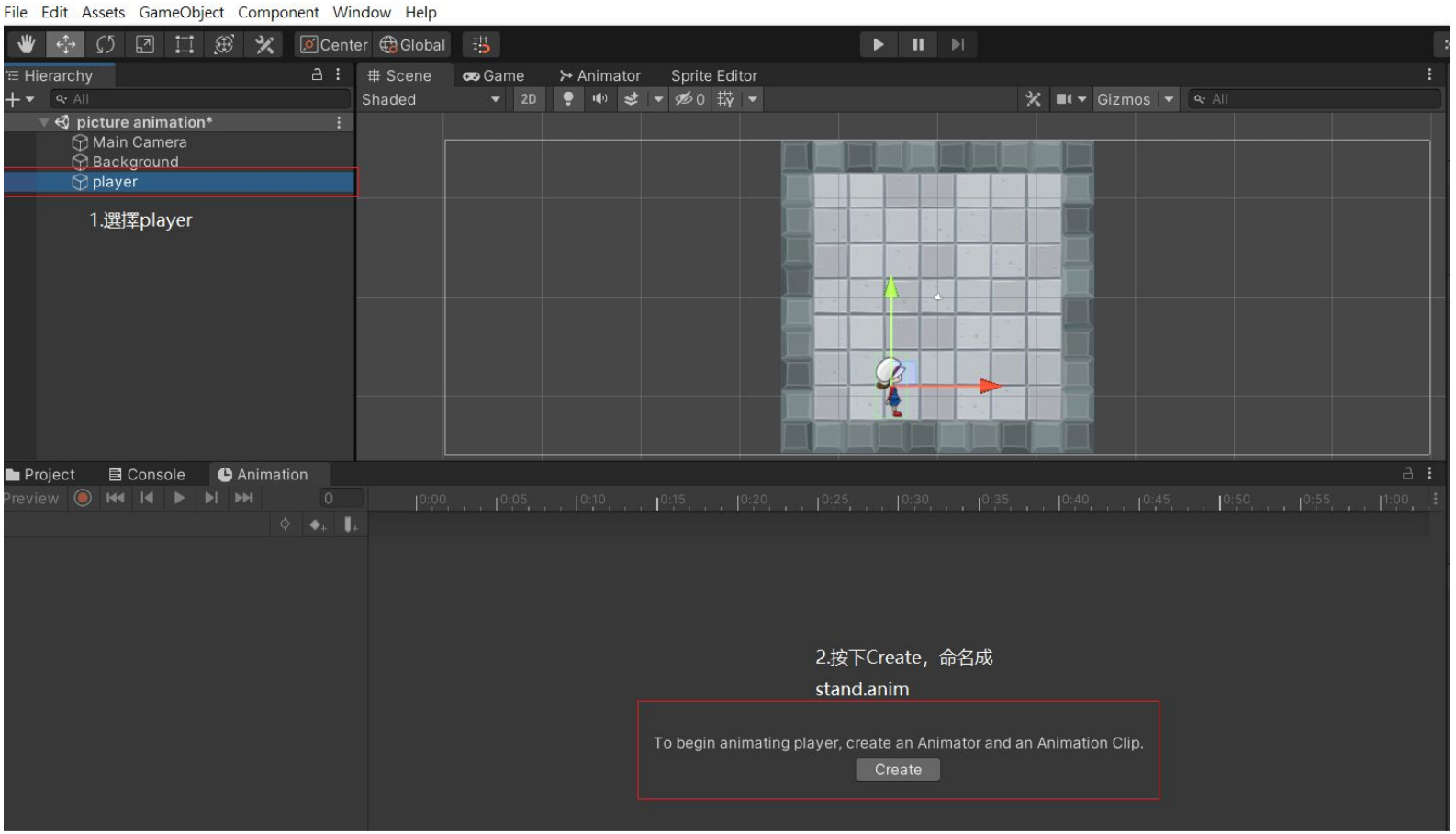
開啟動畫視窗

Window → Animation→ Animation



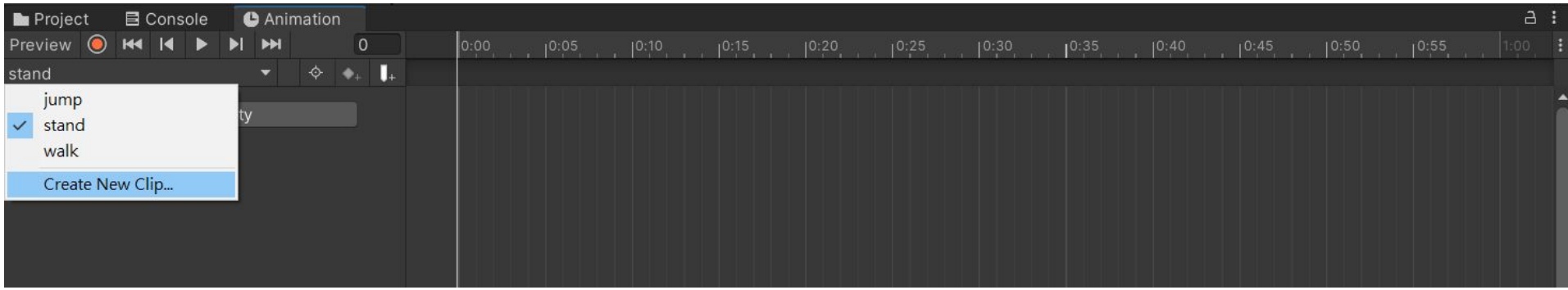
製作動畫

選擇左側的Player(因為 Animation 是綁定在物件底下，所以要先選 Player 才能將所新增的 Animation 與 Player綁定)，按下Create新增動畫



製作動畫

選擇Create New Clip，新增 jump move 動畫



stand 單獨放 stand 圖片

move 將 stand 、walk 、stand加入時間軸(等差時間軸)

jump 單獨放入 fly 圖片

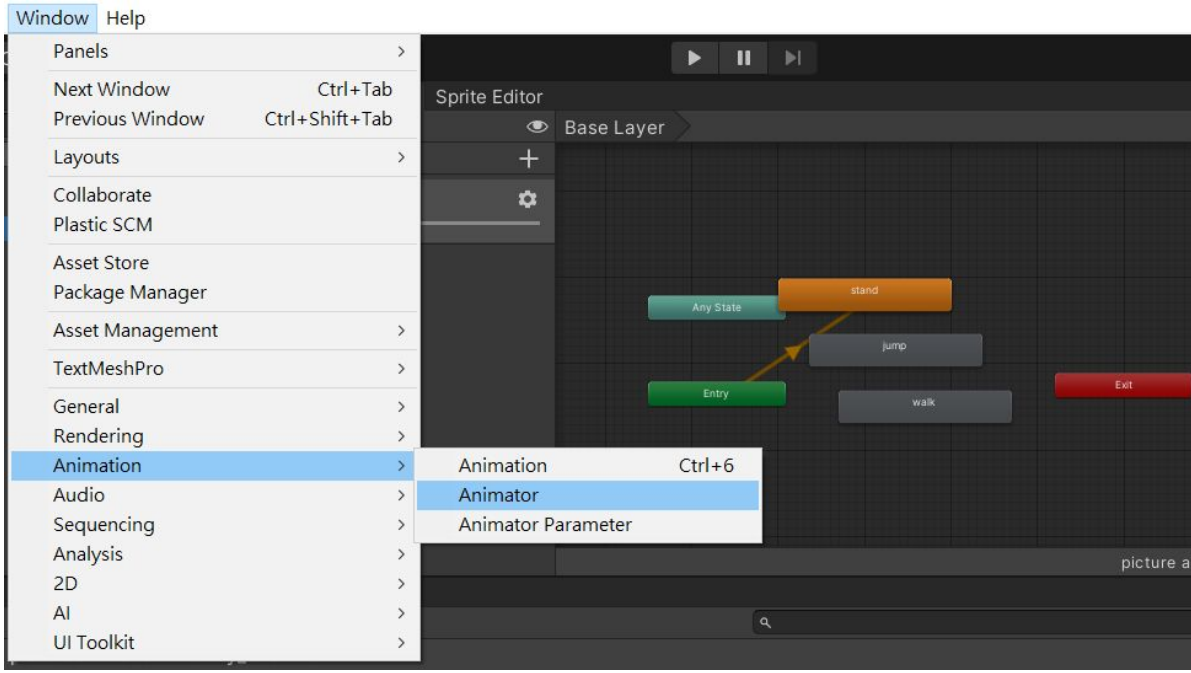
按下播放鍵可測試動畫

動畫播放控制 Animator

開啟 Animator 視窗

Window → Animation →Animator

針對這個player Animator點開來會有五個東西(如右圖)(沒有的話可自行將剛剛創建的三個動畫拖曳進這裡)



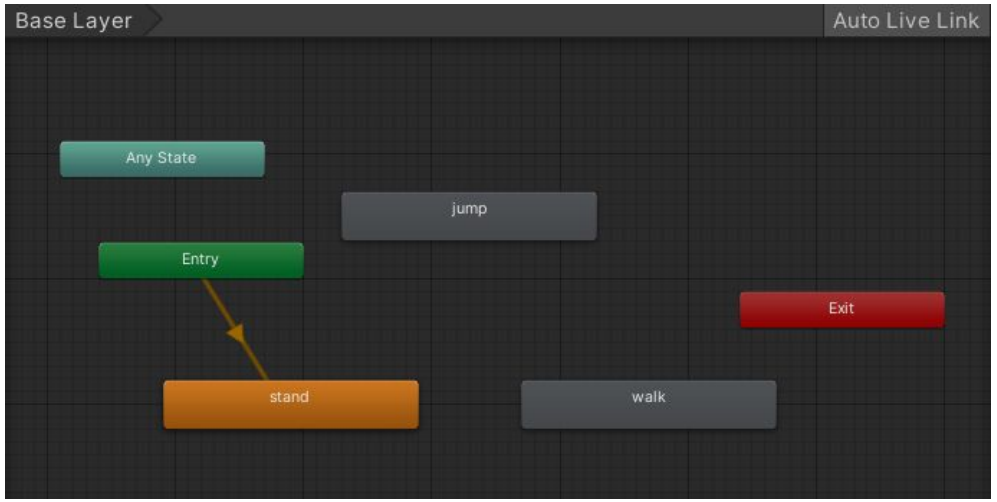
Make transition

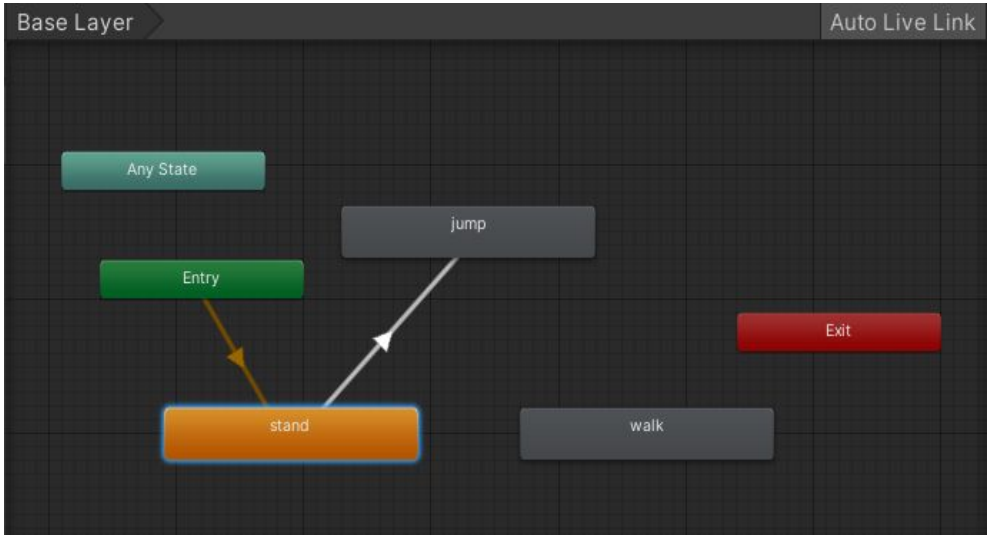
點選stand，按右鍵

→ Make Transition

→左鍵點選另一個動畫

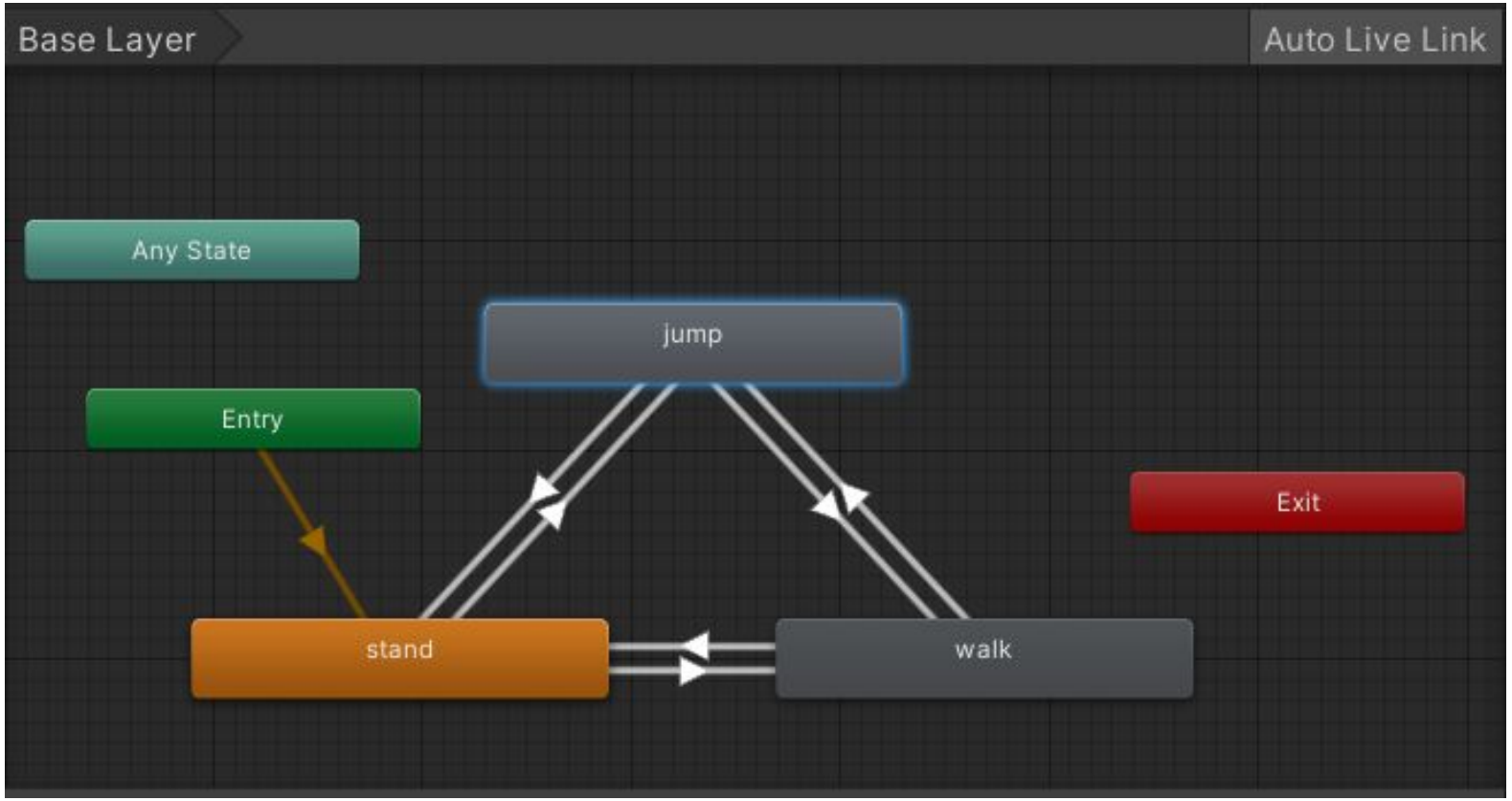
→ 有箭頭出現就成功





Make transition

用同樣的方法，讓所有的動畫之間都有transition(箭頭連接)



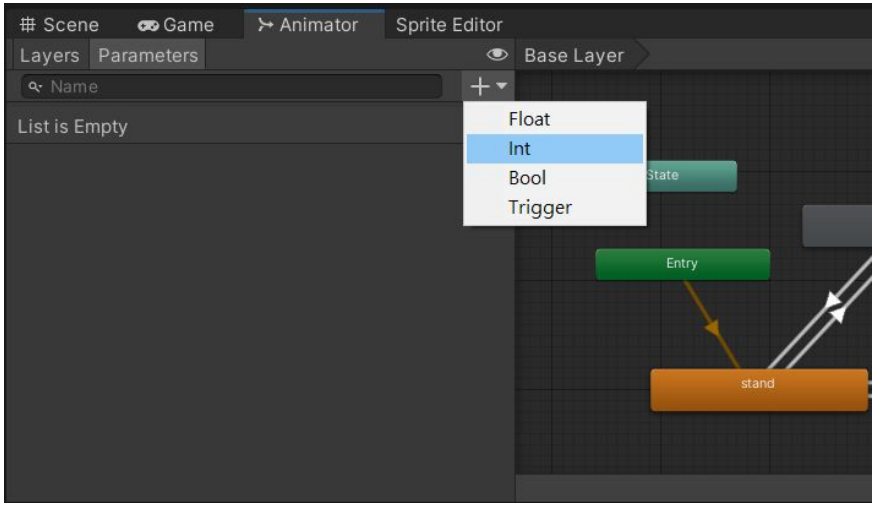
使用變數控制狀態

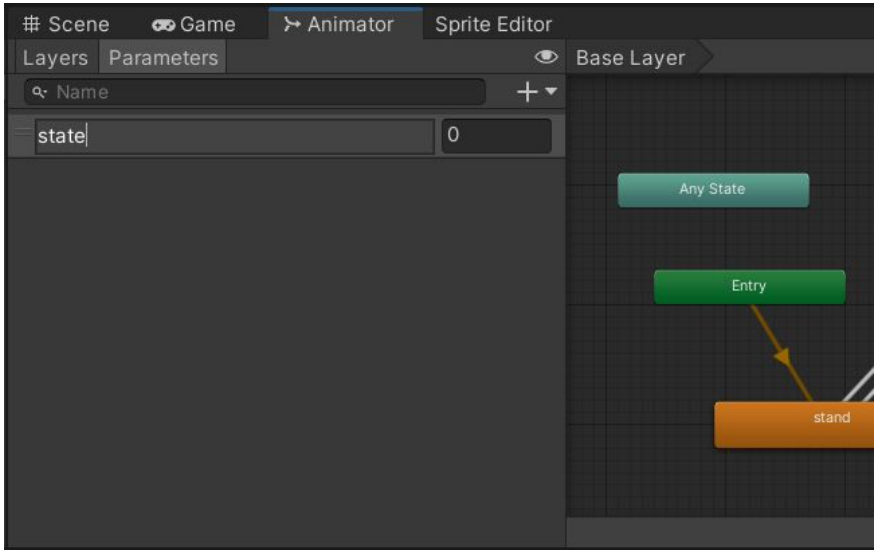
點選Parameters，按+

增加一個 Int

作為狀態判定的參數

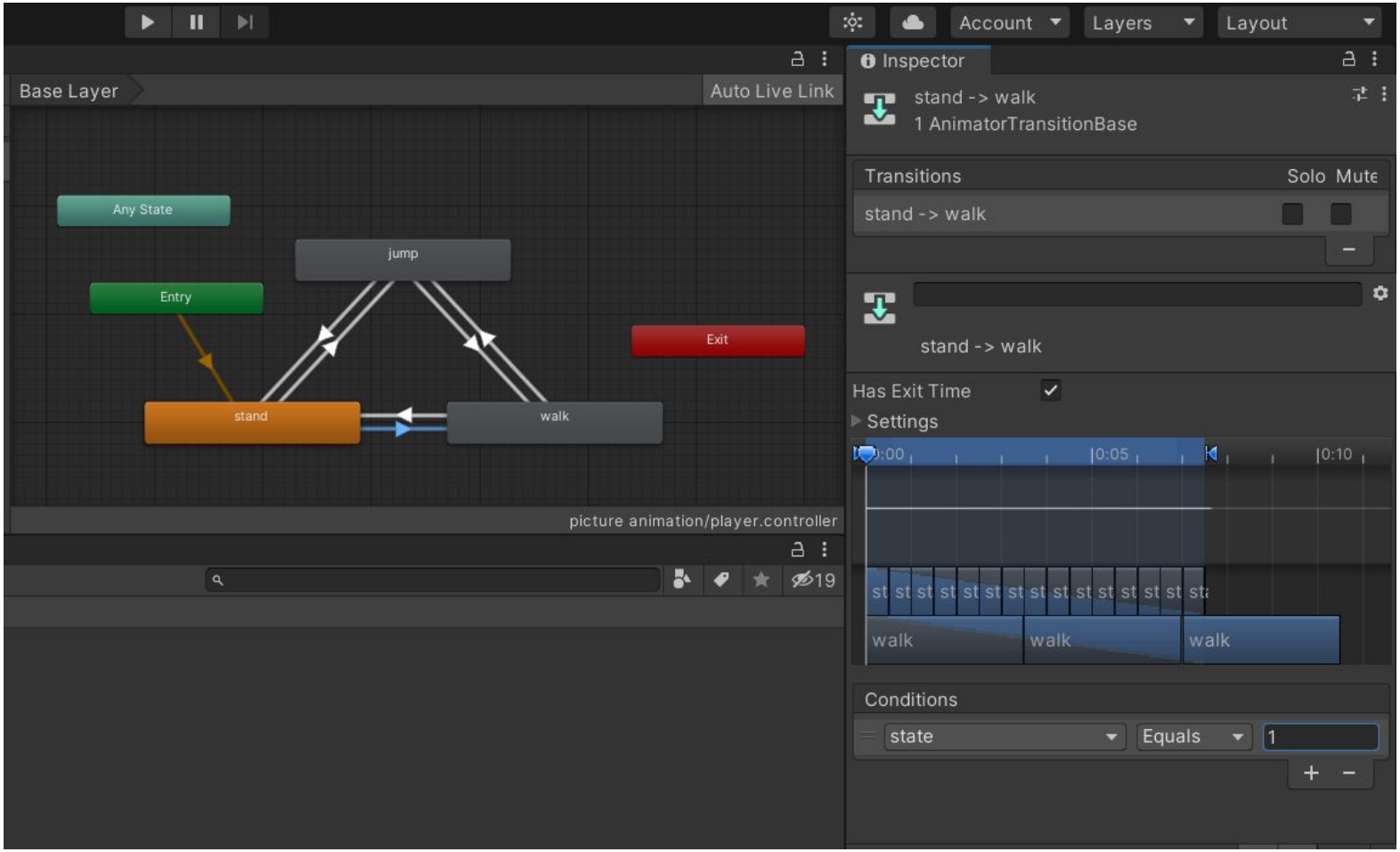
並命名為 state





狀態控制變數

點選stand往walk箭頭，在Conditions那一欄按+新增條件判斷，把條件設成state equals 1



狀態控制變數

將所有往 stand 箭頭的 condition 皆設為 Equal 0

將所有往 move 箭頭的 condition 皆設為 Equal 1

將所有往 jump 箭頭的 condition 皆設為 Equal 2

在 script 中改變狀態控制變數

playerController 新增 animator 讓 script 可藉此操作變數

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

Unity 指令碼|0 個參考
public class playerController : MonoBehaviour
{
    public float speed = 20;
    private Rigidbody2D rbody2D;
    private bool isOnGround;
    private Animator animator;

    Unity Message|0 個參考
    void Start()
    {
        rbody2D = gameObject.GetComponent<Rigidbody2D>();
        animator = gameObject.GetComponent<Animator>();
        isOnGround = false;
    }
```

在 script 中改變狀態控制變數

運用 SetInteger(變數名稱, 狀態值) 來控制變數

```
void Update()
{
    float dirx = Input.GetAxisRaw("Horizontal");
    rbody2D.velocity = new Vector2(dirx * speed, rbody2D.velocity.y);

    if (Input.GetKey(KeyCode.Space) && isOnGround)
    {
        isOnGround = false;
        rbody2D.velocity += new Vector2(0, speed);
    }

    if (!isOnGround)
    {
        animator.SetInteger("state", 2);
    }
    else if (dirx != 0)
    {
        animator.SetInteger("state", 1);
    }
    else
    {
        animator.SetInteger("state", 0);
    }
```

跳躍

走路

站立

變換方向

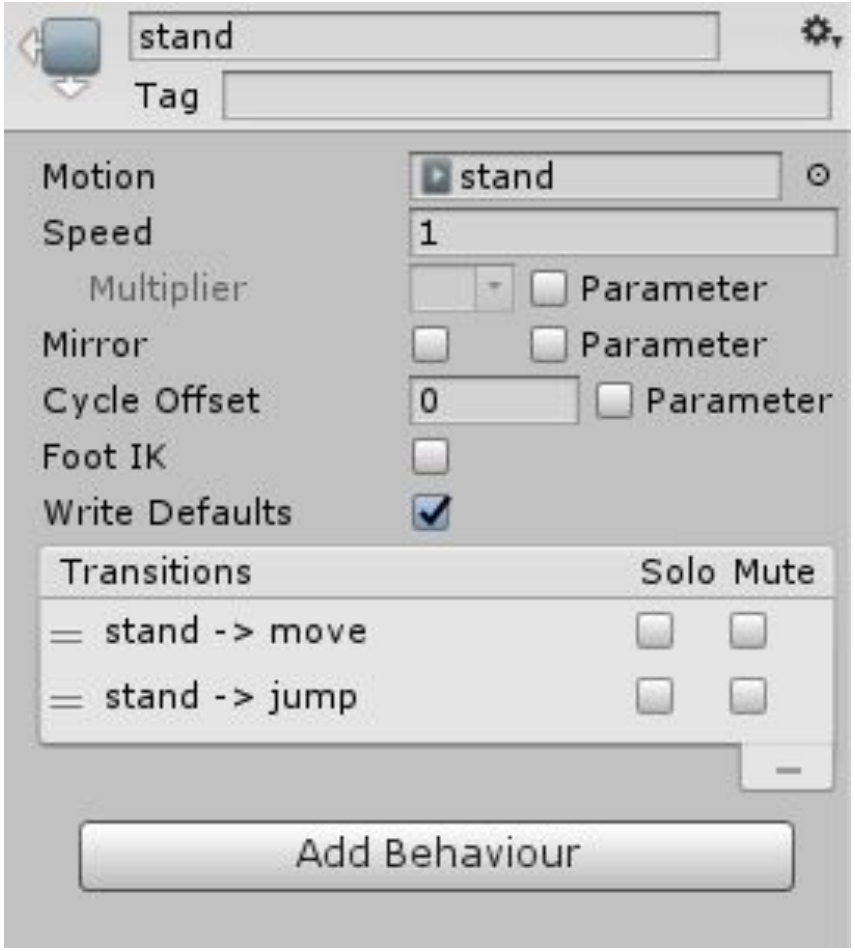
利用改變 scale X 的數值來達成轉向

```
void Update()
{
    float dirx = Input.GetAxisRaw("Horizontal");
    rbody2D.velocity = new Vector2(dirx * speed, rbody2D.velocity.y);

    if (Input.GetKey(KeyCode.Space) && isOnGround)
    {
        isOnGround = false;
        rbody2D.velocity += new Vector2(0, speed);
    }

    if (!isOnGround)
    {
        animator.SetInteger("state", 2);
    }
    else if (dirx != 0)
    {
        animator.SetInteger("state", 1);
    }
    else
    {
        animator.SetInteger("state", 0);
    }

    if (dirx * transform.localScale.x < 0)
    {
        this.transform.localScale = new Vector3(-1 * this.transform.localScale.x, this.transform.localScale.y, this.transform.localScale.z);
    }
}
```

Run

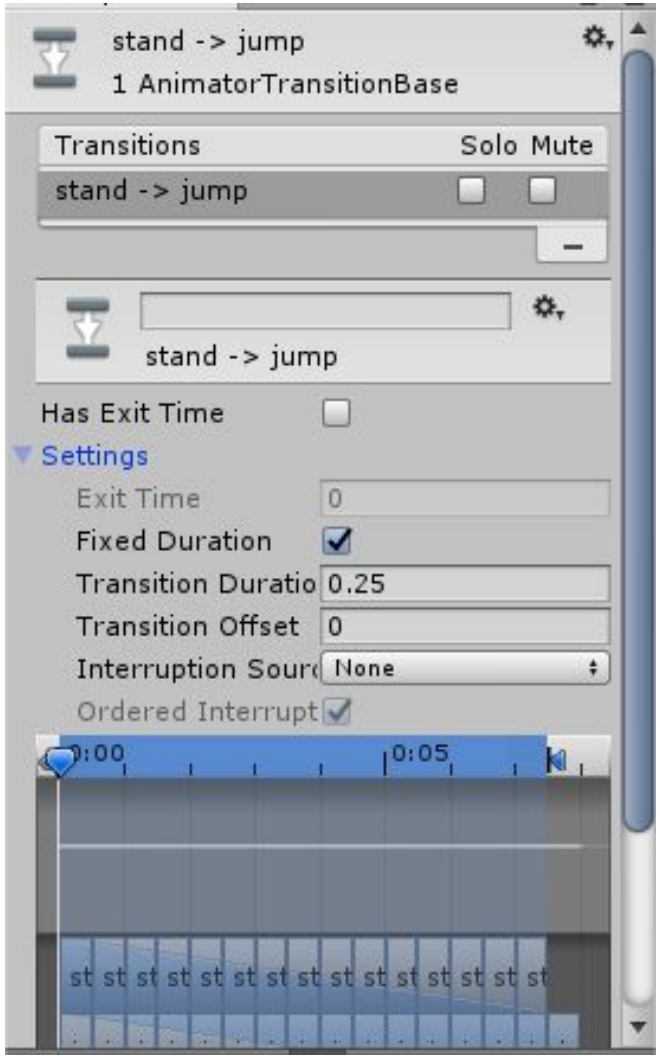
測試

補充(animation)



- **Speed**：動畫撥放的速度
- **Solo**：勾選後，只能通過有勾選 Solo 的狀態，其餘會被當作 Mute
- **Mute**：勾選後，不會切入到該狀態

補充(transition)



Has Exit Time：打勾後 Exit Time 會啟用

關閉 Exit Time >> 目前動畫直接切掉換成下一個動畫

開啓Exit time >>會等目前動畫播到Exit Time再切換到下一個動畫並播放

Exit Time可設定 0 到 1 之間, 1 為完整撥放

Fixed Duration：打勾後transition duration的單位為秒反之則為動畫長度的百分比
Trasition Offset：改變下一個動畫播放的時間點

挑戰

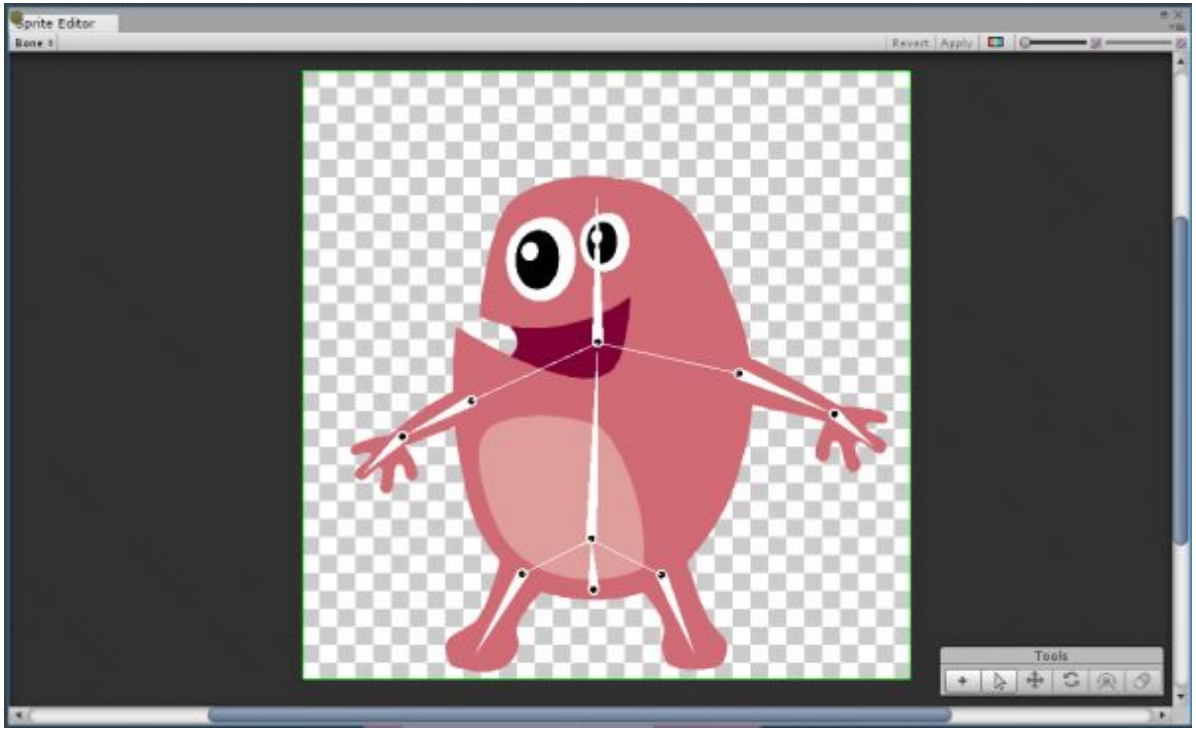
利用asset store素材，練習animation機制

- Sunny Land
- 試著完成idle、移動、跳躍、(蹲下)的動作

(補充)骨骼動畫製作

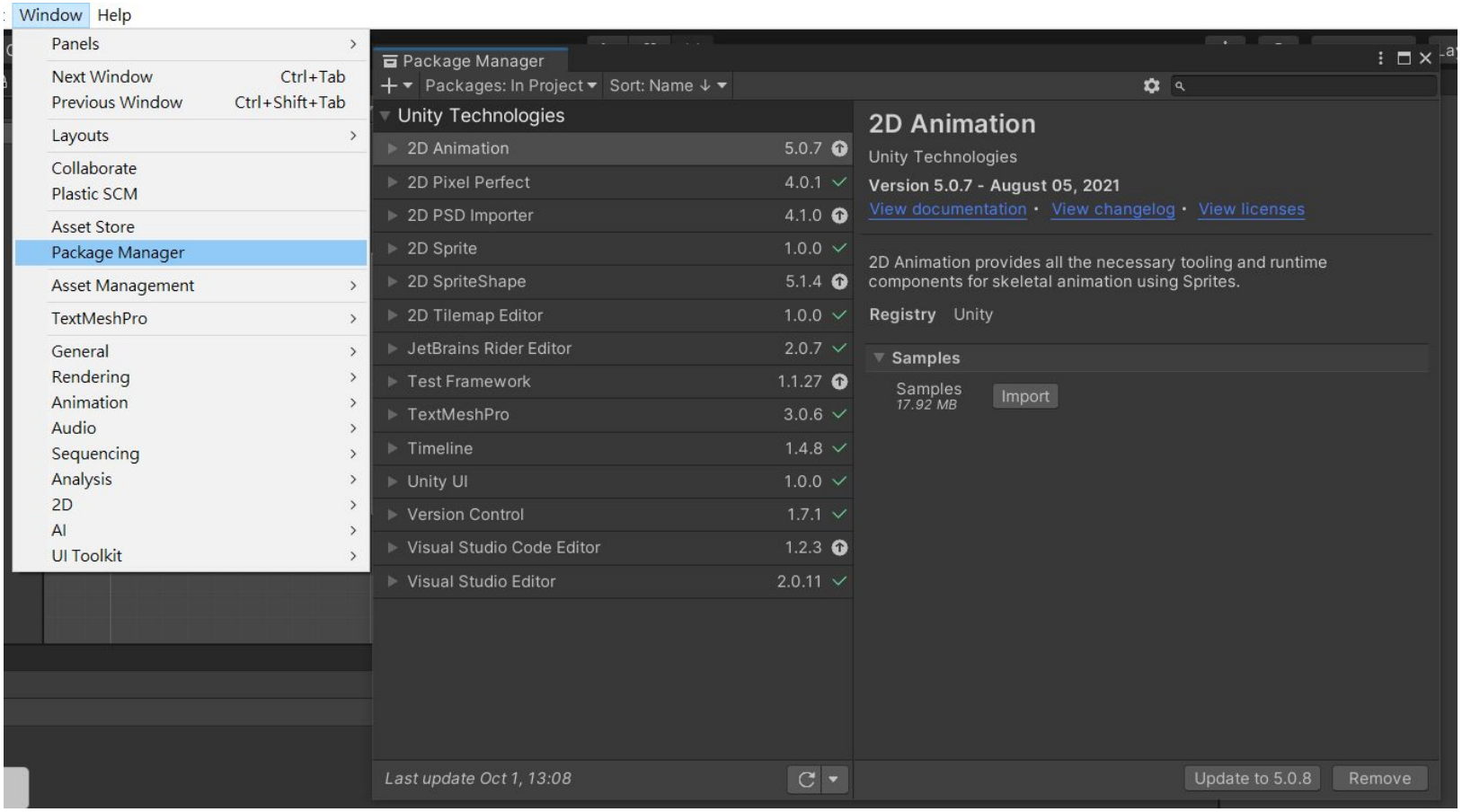
骨骼動畫(Skeleton animation)

骨骼動畫是利用建立好的骨架，套用到圖片或模型使其動作，跟傳統的圖片動畫相比，能夠更好和精確的去控制動畫裏的動作，做出更加生動的動畫。



匯入素材

點選Window→Package Manager→2D Animation，裏面有Samples，按download(import)，把素材匯入Unity，

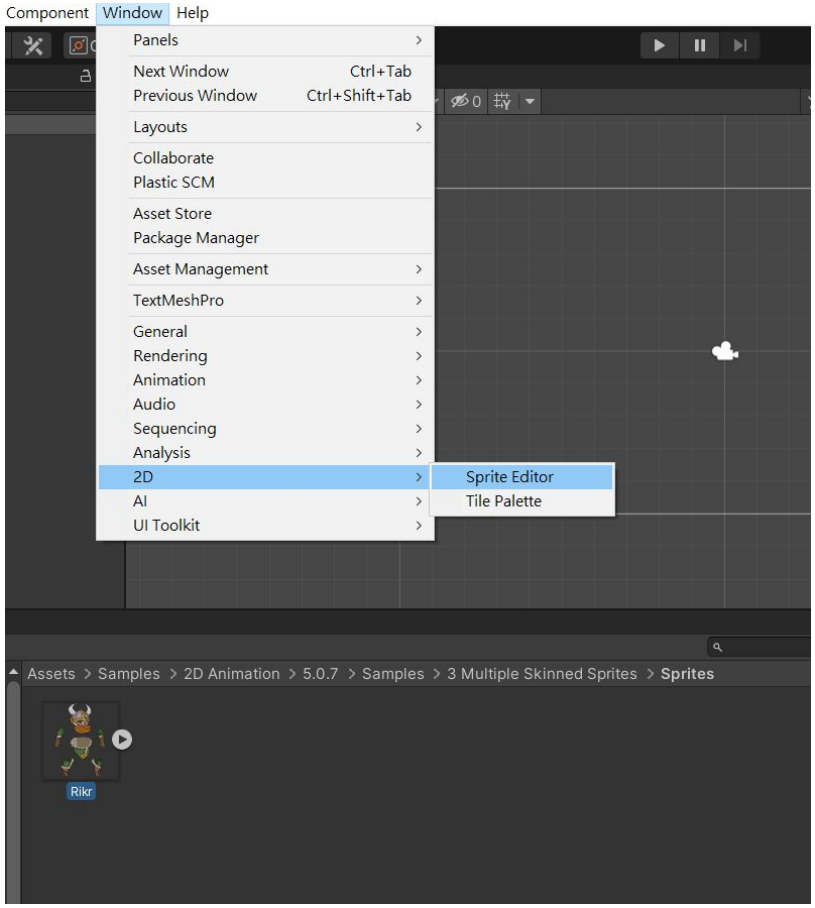


匯入素材

把Rikr_with_bone.unitypackage拉到unity裏面，或者從Asset>Import Package引入，裏面有Rikr_with_bone.prefab

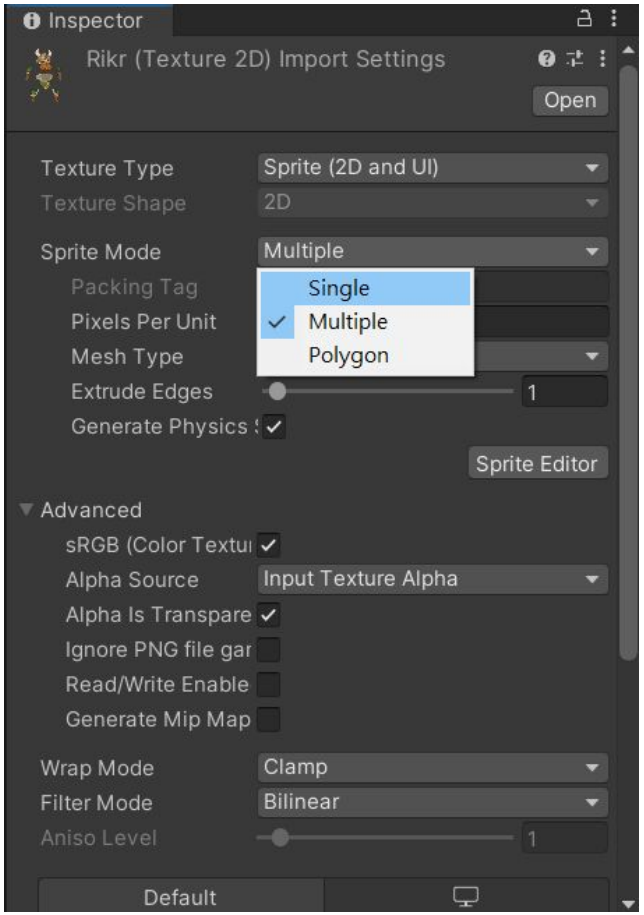
製作骨架

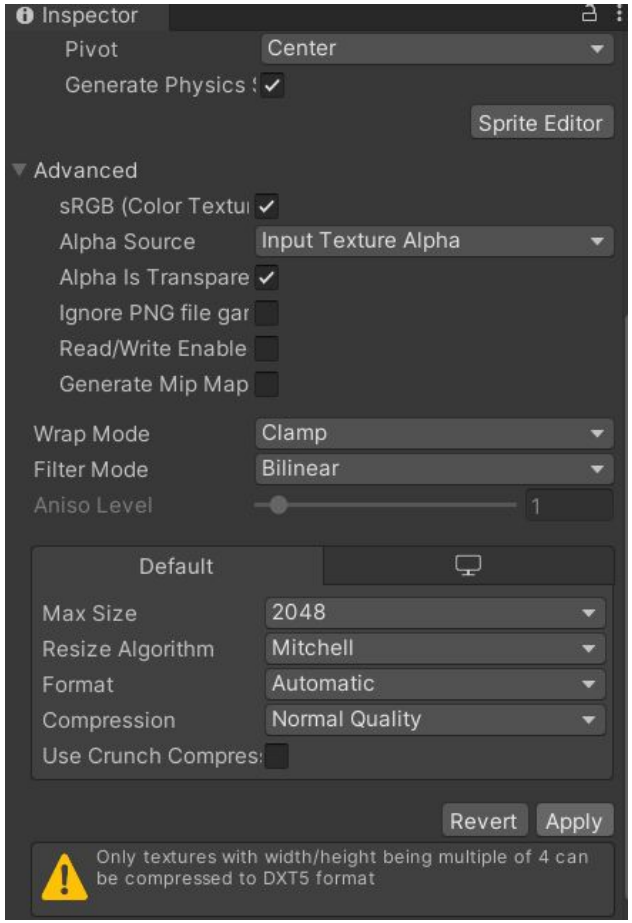
選擇Samples->3 Mutiple Skinned Sprite→Sprite的Rikr圖片，然後點Window→2D→Sprite Editor，打開Sprite Editor的視窗



製作骨架

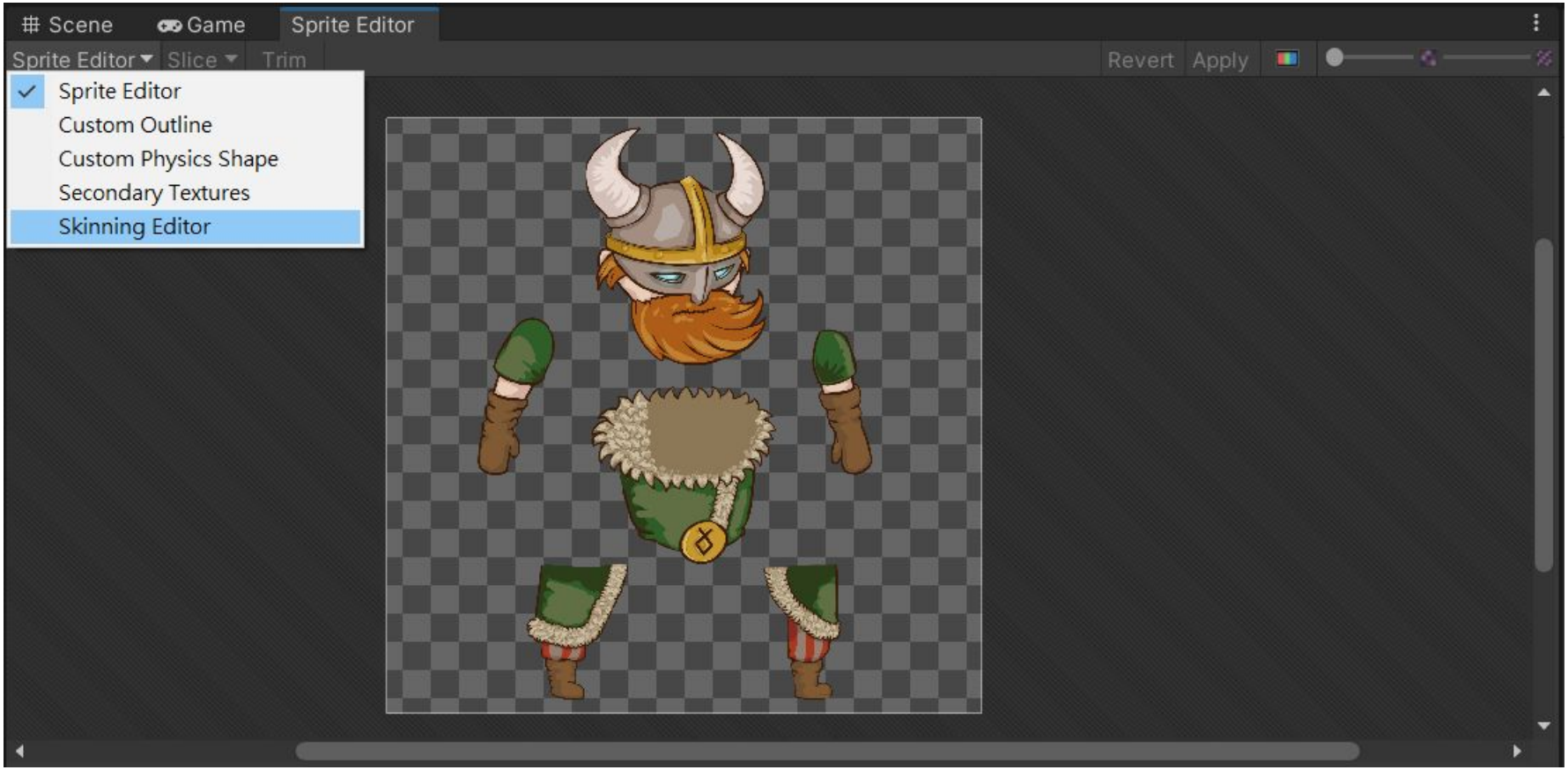
把Inspector裏的Sprite Mode從Multiple改成Single，並按下Apply。





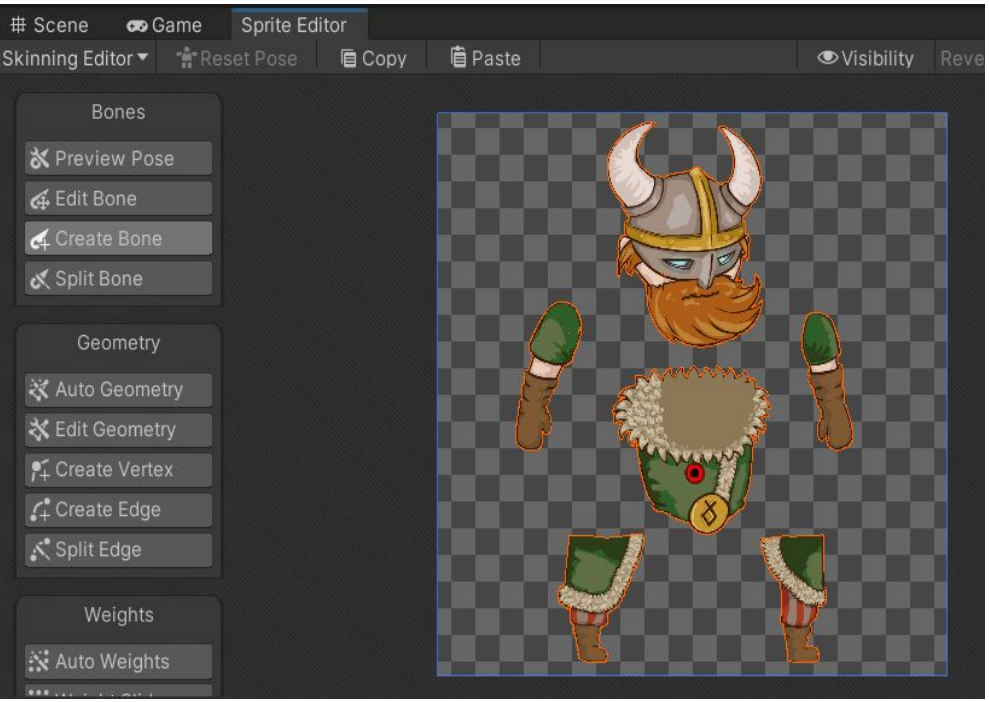
製作骨架

回到Sprite Editor，把Sprite Editor的選項打開，換成Skinning Editor



製作骨架

選擇Create Bone，並對圖片雙擊左鍵打開骨架預覽模式。



Edit bone 模式
移動關節：左鍵點選黑點移動
移動骨骼：左鍵點選黑點以外地方移動
連接關節：左鍵點選黑點移動到對應骨骼的黑點

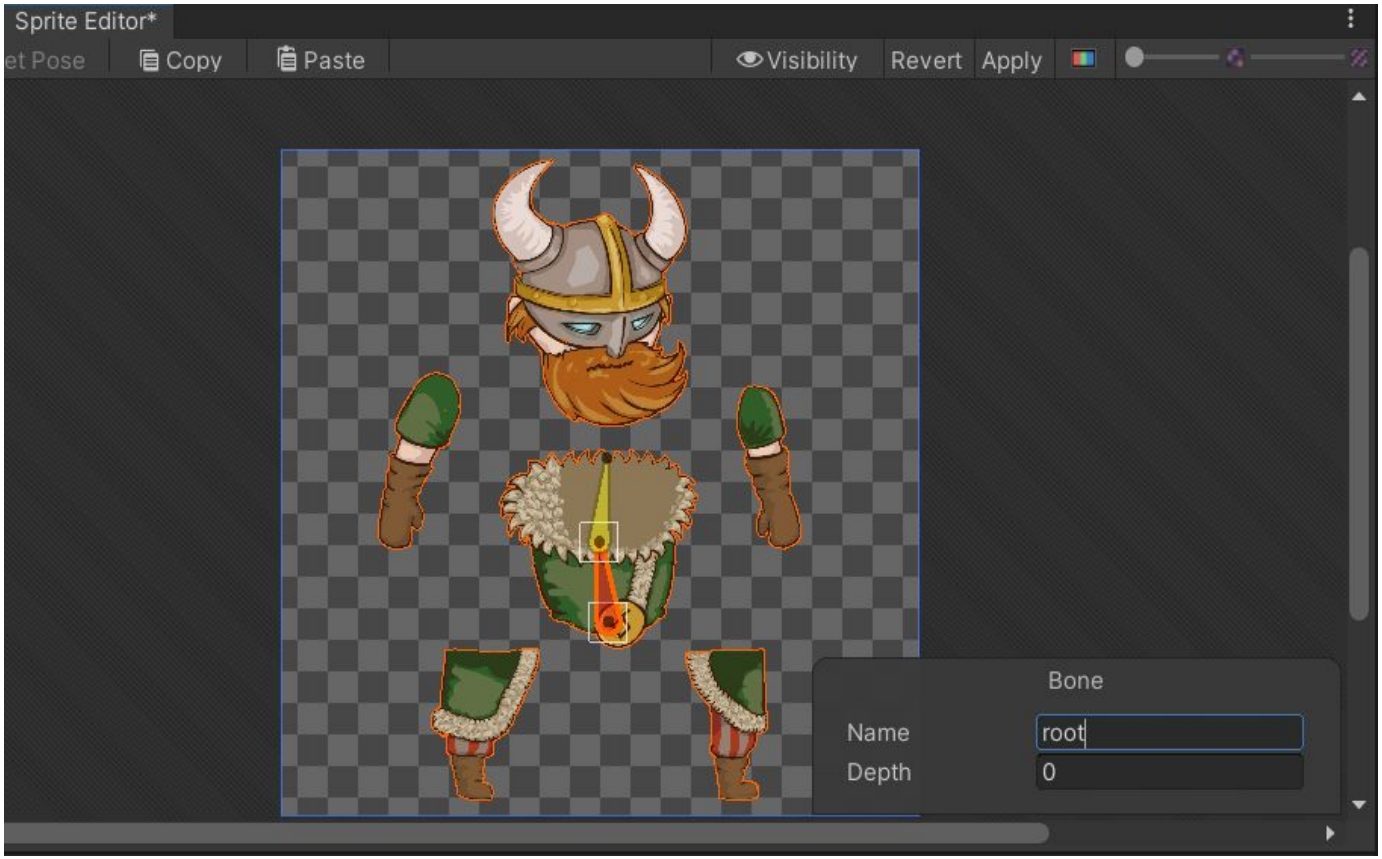
Create bone模式
創造骨骼：左鍵
取消創建：右鍵/Esc

Split bone 模式
插入骨骼：在骨骼中左鍵點選黑點以外地方
移動關節：左鍵點選黑點移動

製作骨架(軀幹)

在肚子部分左鍵新增一個bone(紅)，然後移動到胸口再左鍵新增一個bone，再移到脖子部分按下左鍵完成上半身骨骼(黃)的創建，并按右鍵取消創建新骨骼。

點選紅色骨骼，並把Name改成root，同樣動作，把黃色骨骼命名成body



製作骨架(腿)

點選紅色骨骼的黑點，之後移動到左腳上半部創建骨骼(綠)，再移動到膝蓋部分創建骨骼(天藍)，最後移動到脚踝點左鍵完成腳骨骼的製作，並點選右鍵取消創造。

之後再點選紅色骨骼黑點，對右脚做一樣的動作。并把骨骼命名成對應的名稱。
綠色：left hip
天藍色：left leg
藍色：right hip
粉色：right leg

製作骨架(手、頭)

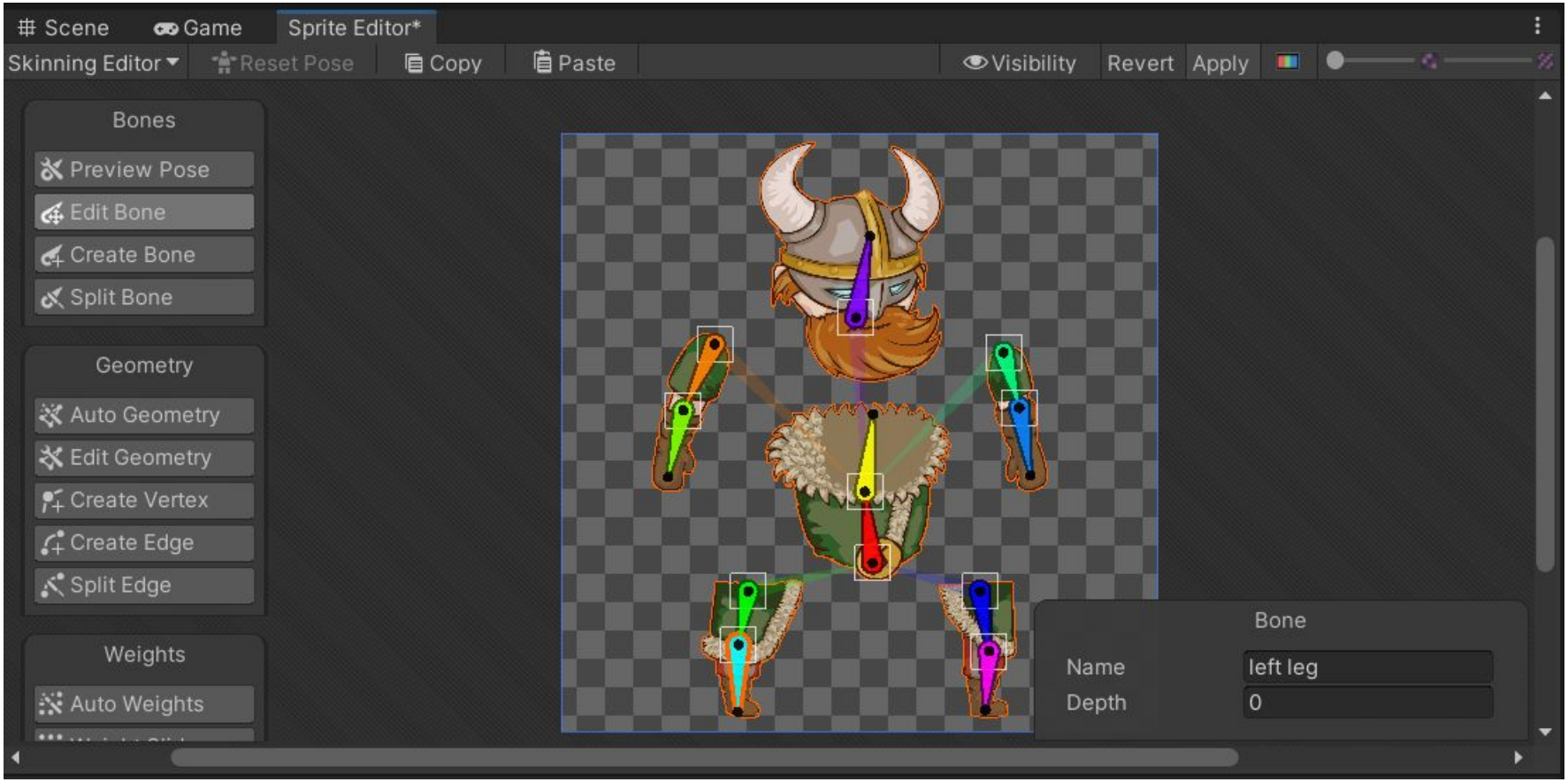
點選黃色骨骼的黑點，之後移動到左手上半部創建骨骼(橘)，再移動到手肘創建骨骼(綠)，最後移動到手腕點左鍵完成手部骨骼的製作，並點選右鍵取消創造。

之後再點選黃色骨骼黑點，對右手做一樣的動作。完成后點選黃色骨骼黑點移動到頭部再拉出一個骨骼，并把拉的骨骼命名成對應的名稱。

紫色：head
橘色：left shoulder
綠色(左)：left arm
綠色(右)：right shoulder
藍色：right arm

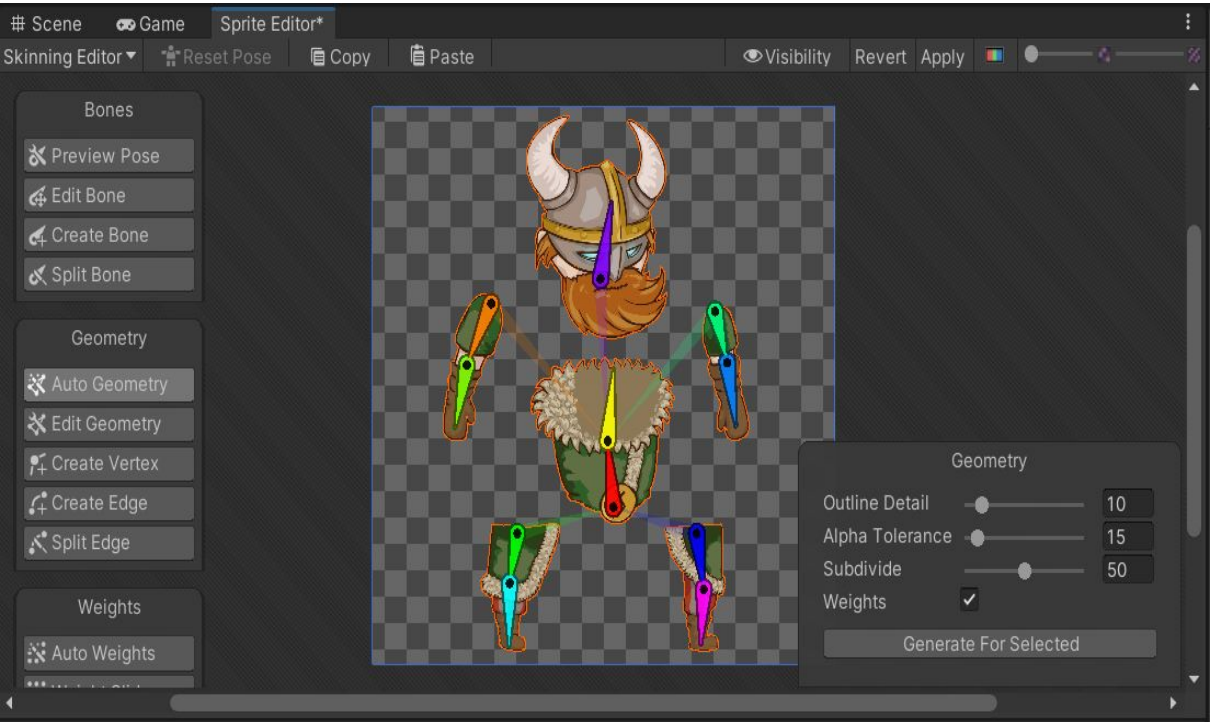
製作骨架

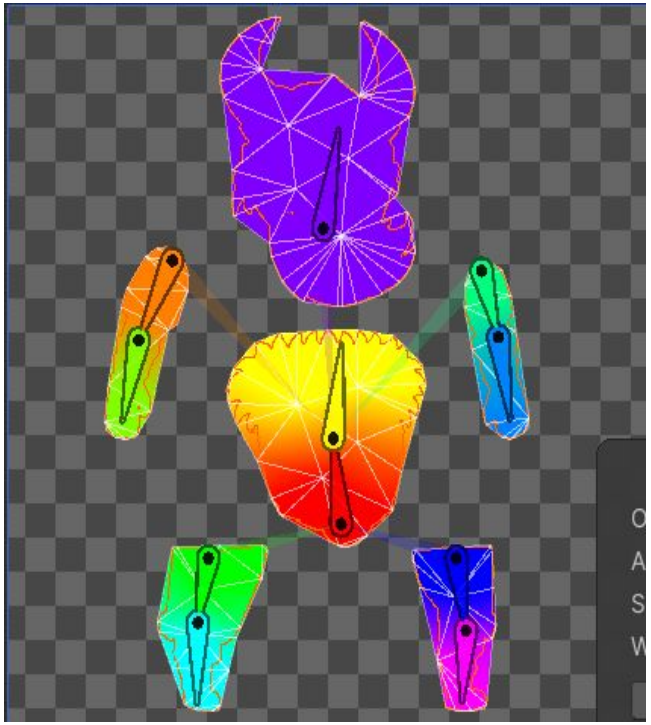
做完后記得右上角的Apply一定要按下去，不然剛剛拉的骨骼會全部不見。



骨架綁定圖片(skining)

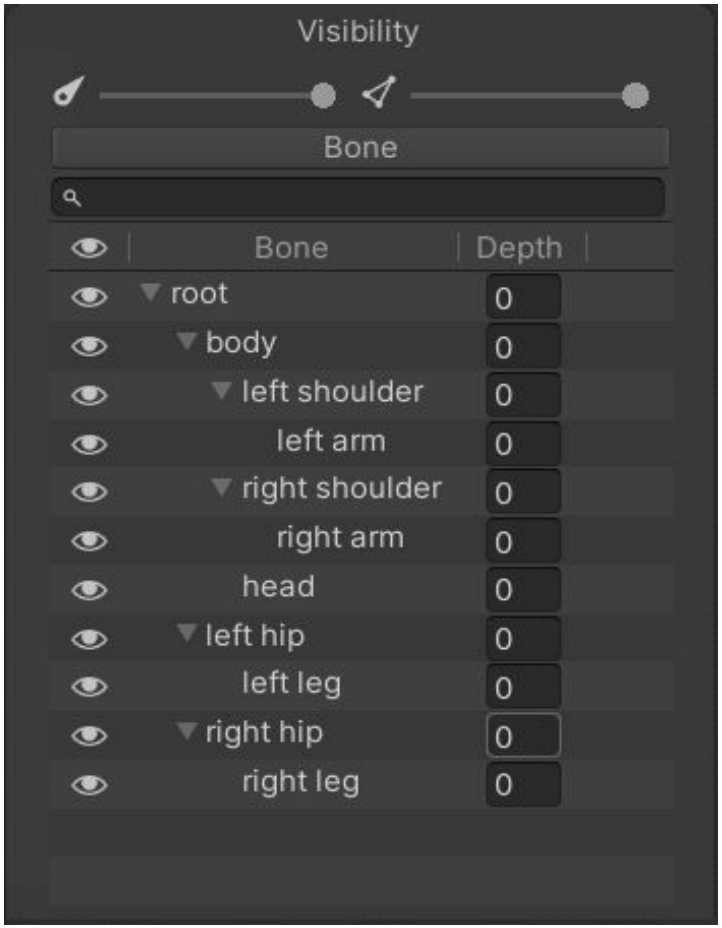
點選Auto Geometry，按下Generate for selected，Unity會根據圖片和關節的位置進行sprite的分割和權重設定，然後可以按preview pose 看骨架綁定的結果，沒問題按Apply。





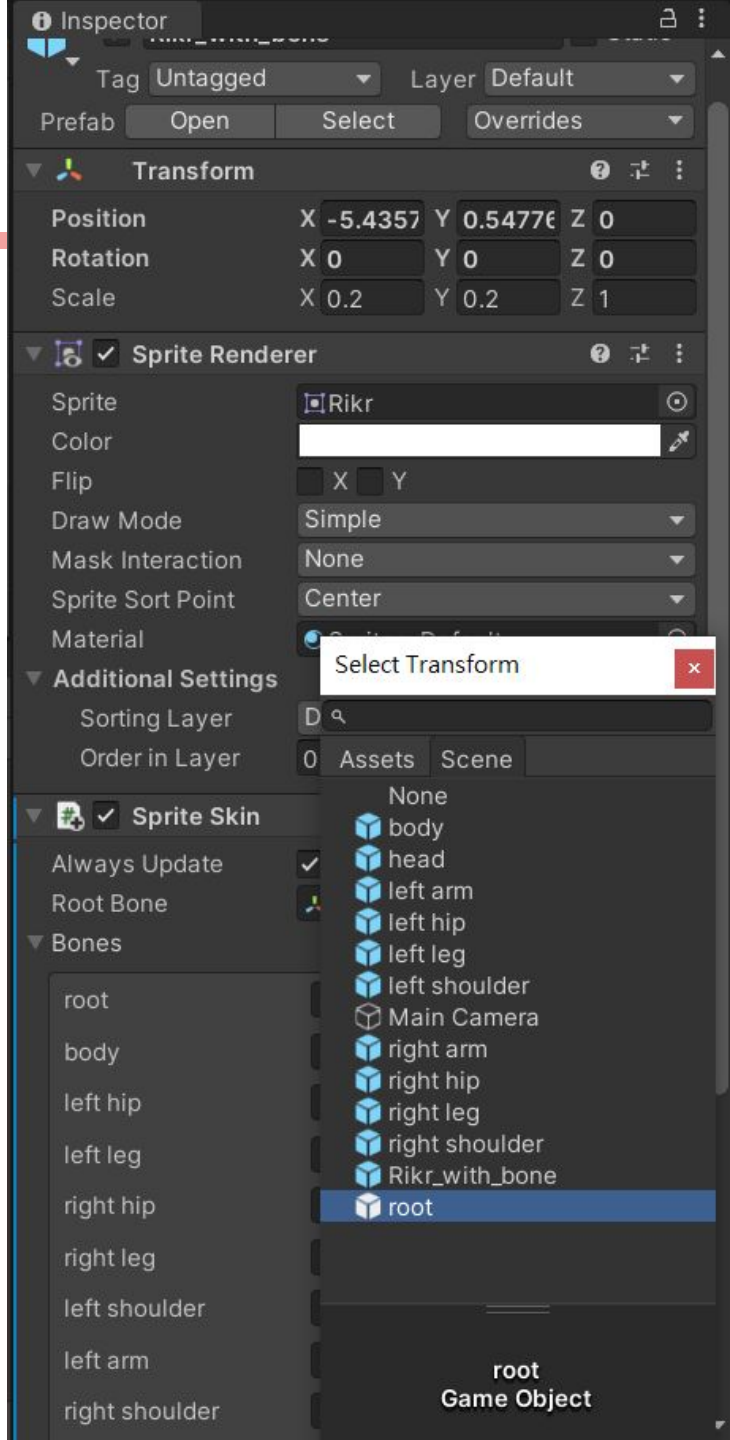
骨架綁定圖片(skining)

點選Visibility，可以看到整個骨架的架構，把Depth調成右圖，然後按Apply



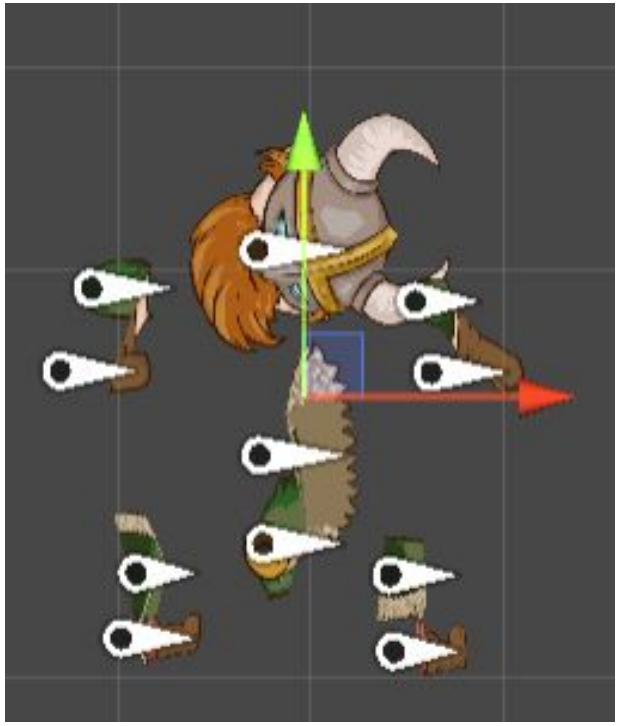
骨架圖片製作

回到Scene，把Rikr_with_bone.prefab拉到場景裏，然後在Inspector裏Add Component，搜尋Sprite Skin添加，之後點選Root Bone這一欄右邊的小圈圈，選擇root。



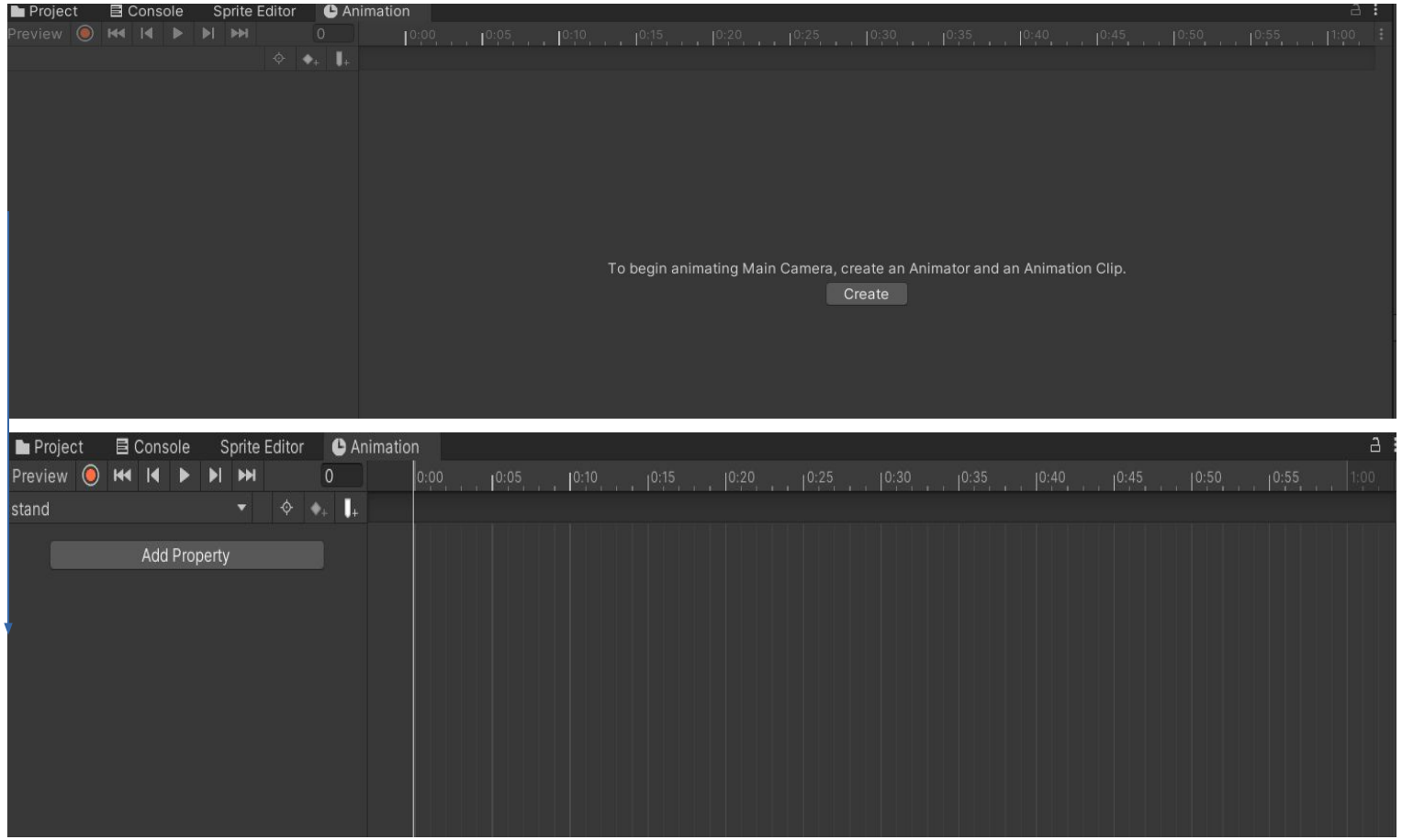
骨架圖片製作

做完上述步驟應該會看到左圖的畫面，因爲方向全部朝右所以會有不自然的扭曲，把骨骼變成之前做的時候的方向，然後把各部位擺到正確的位置



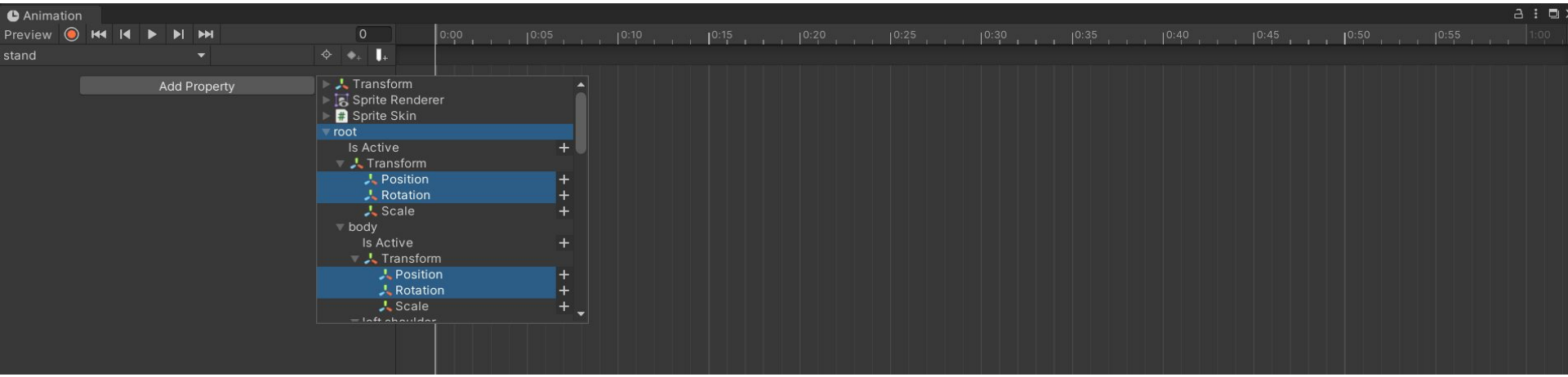
骨骼動畫製作

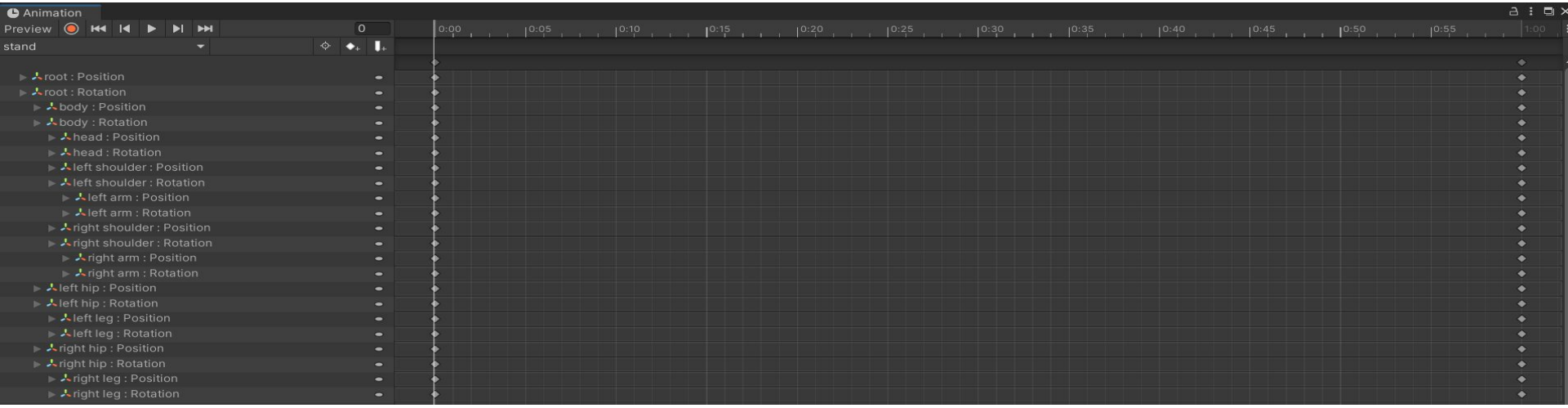
點選Rikr_with_bone，然後點window->Animation->Animation，按create新增兩個動畫，取名為stand和jump



骨骼動畫製作

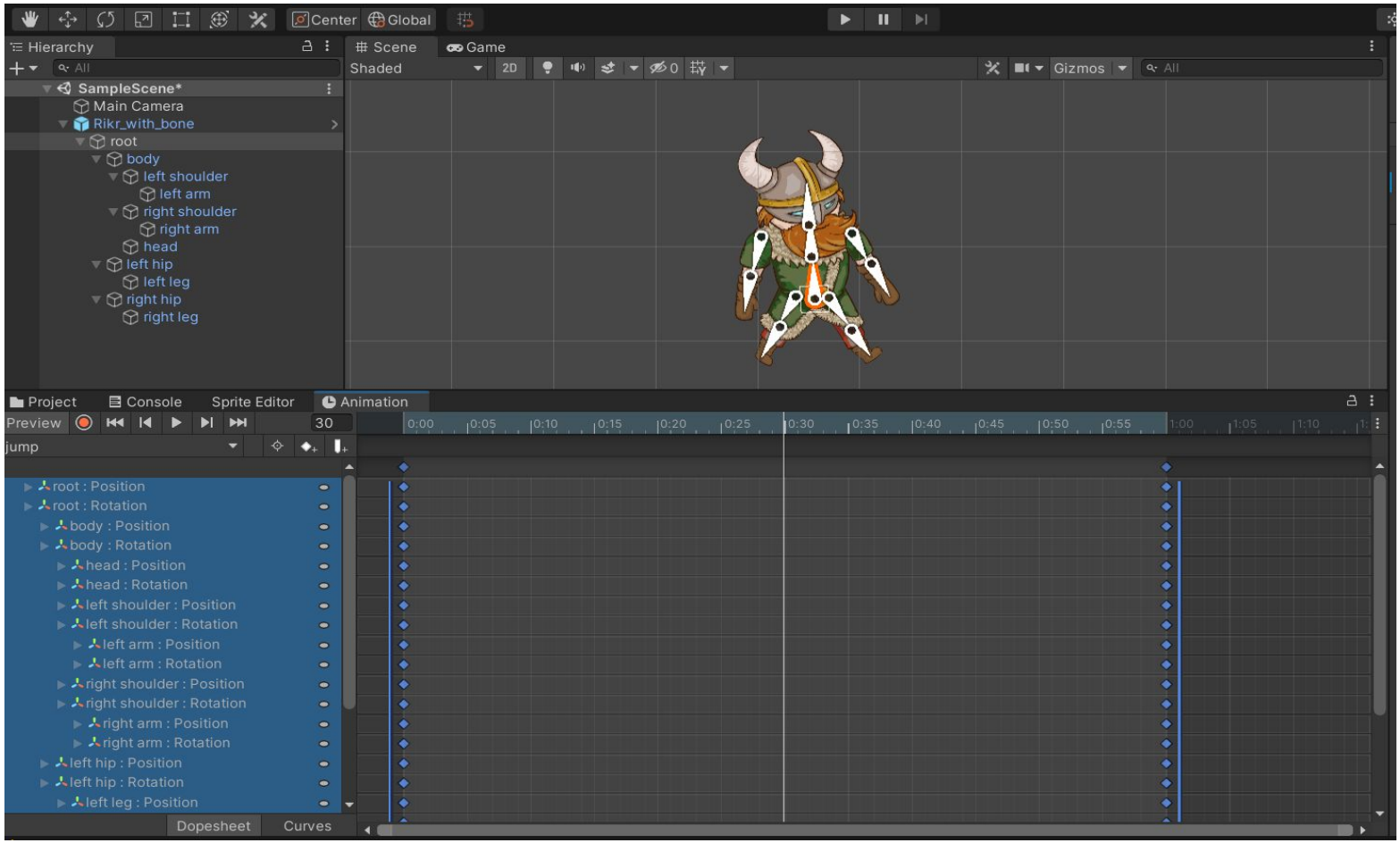
點Add Property，選擇root，然後按Alt+右方向鍵展開所有關節。按住ctrl複數選擇，把每一個關節的Transform選擇，全部選完后按滑鼠右鍵Add Properties，會變下圖這樣。





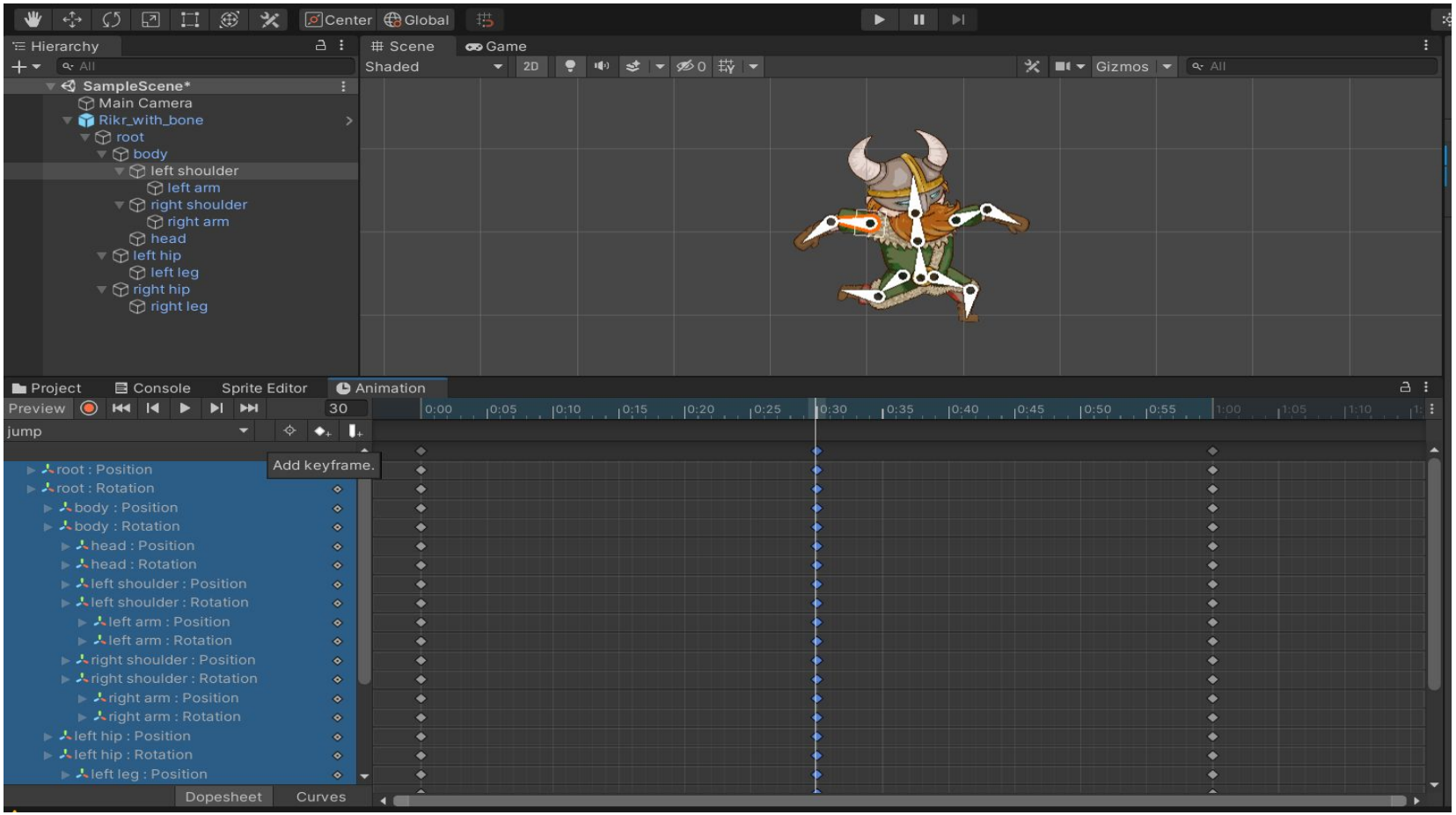
骨骼動畫製作

點root:Position，按ctrl+A全選，點選時間軸的中間部分(0:30)，應該會看到一條白線在中央



骨骼動畫製作

移動場景裏的關節，使其變成跳躍最高點的動作，然後按Add keyframe，此時中間應該會多出一堆方塊，然後按播放鍵可以測試動畫播放並進行微調。



骨骼動畫製作

剩下的走路等動畫也可以利用更改關節位置製作，只要在對應的時間點擺好想要的pose，並按下add keyframe，至於keyframe之間的變化的位移和旋轉會由Unity自動計算，keyframe和關節數愈多所做出來的動畫會愈精緻。