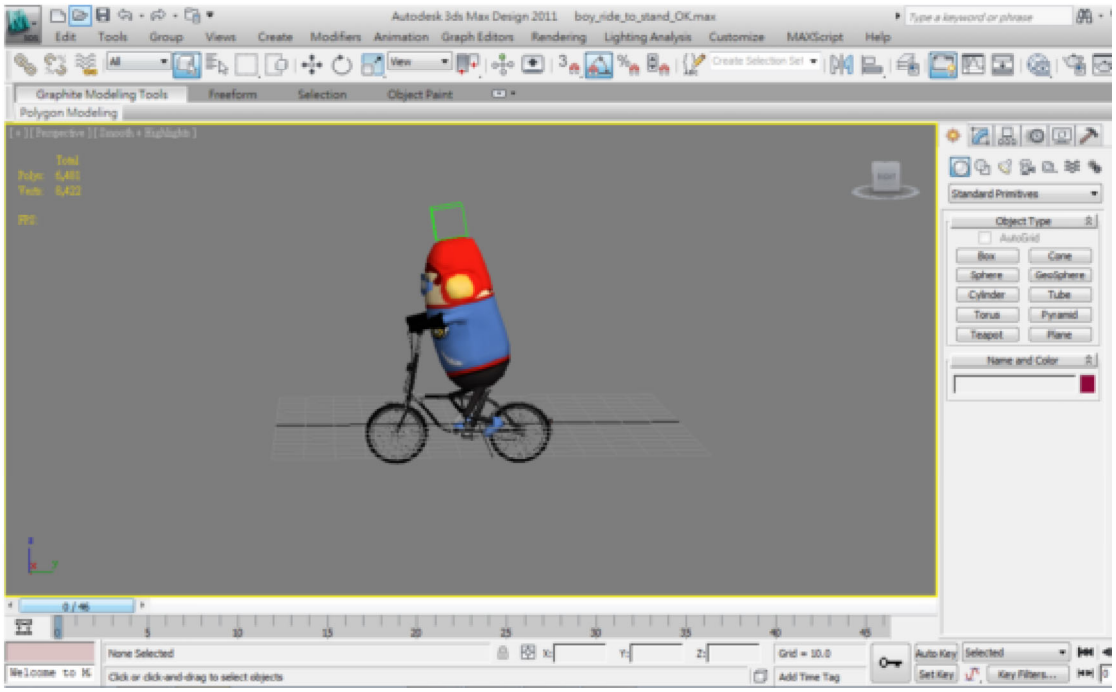
第4章角色匯入與控制

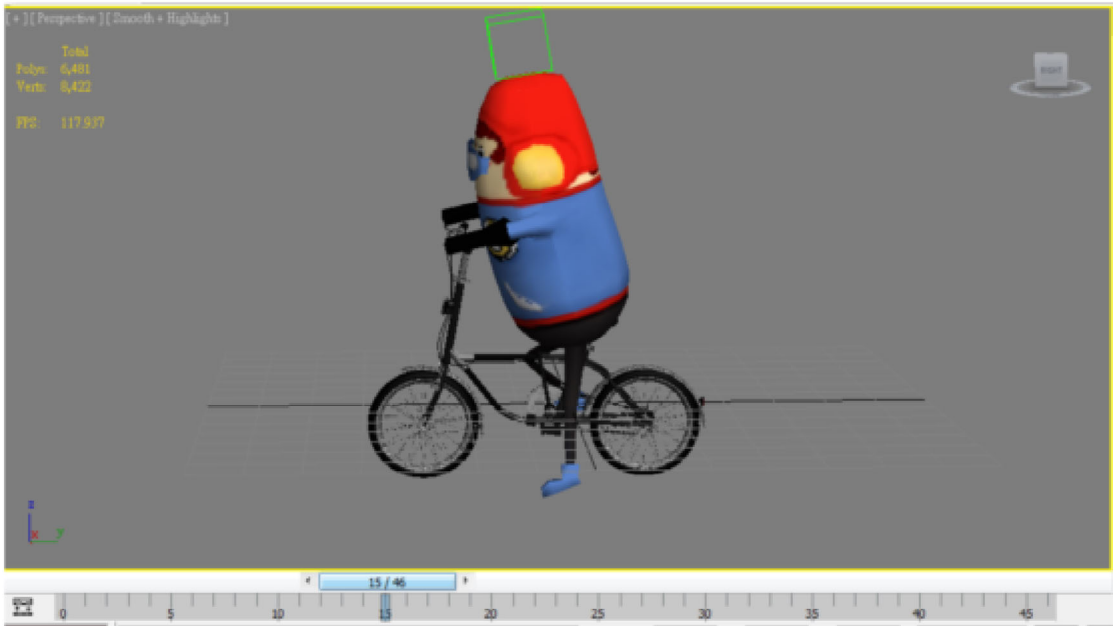
4.1 角色匯入Unity 3D

4.1.1 使用3DS Max建立角色

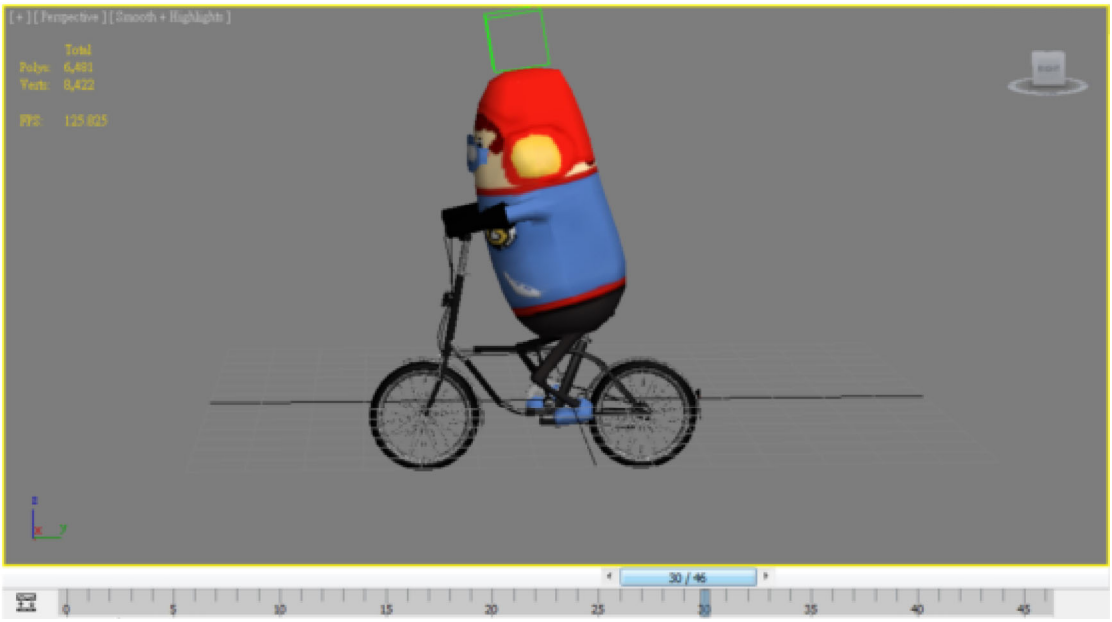
使用3DS Max將角色建模完成，綁上骨架，並製作動作。請參考本書提供之**boy.max**檔案，下圖所示為3DS Max開啟boy.max後在影格第0格的右視圖狀況。



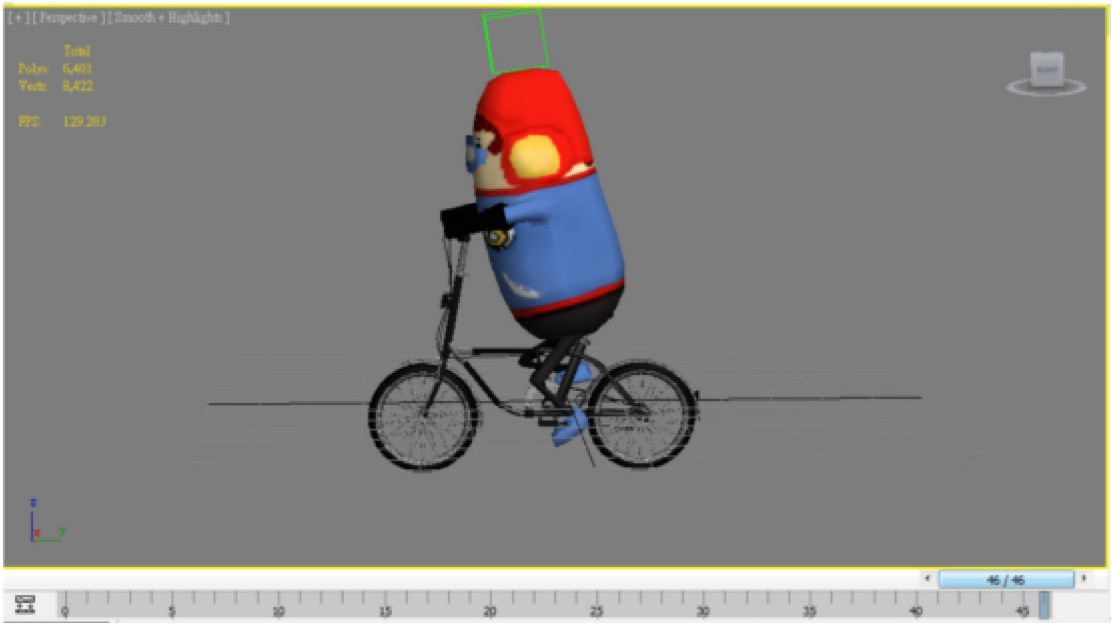
Boy.max內含一個角色並做3種動作，影格0-15做停車腳放下動作，下圖為第15格動作顯示角色的腳已放下。



下圖為第30格動作顯示角色腳已抬起，影格15-30做準備騎車腳提起之動作；

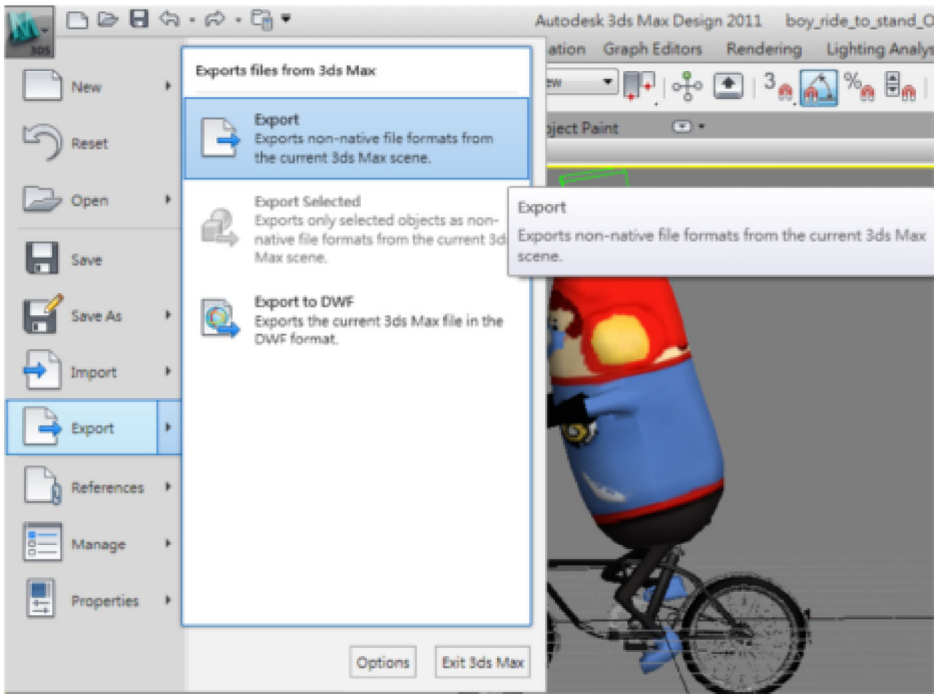


下圖為影格第46格動作做踩踏的動作，影格30-46做腳踩騎車的動作正好做一圈踩踏。

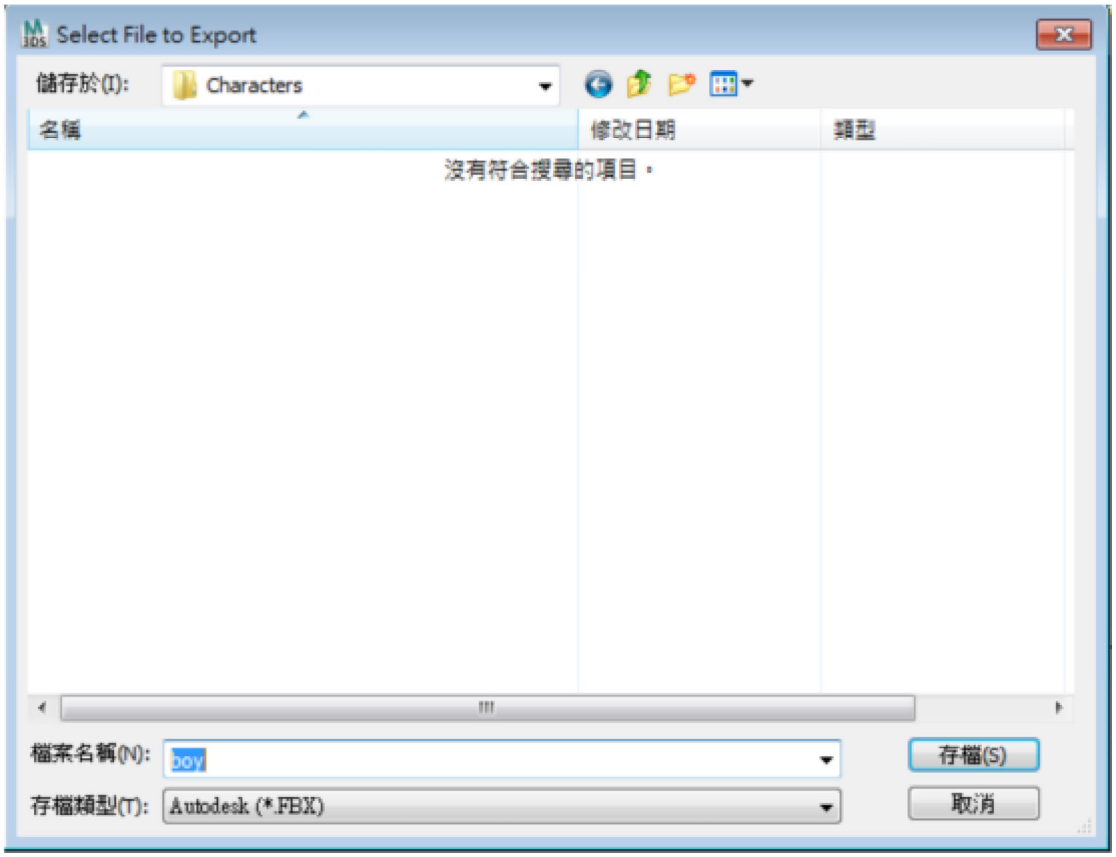


4.1.2 3DS Max角色匯出與匯入Unity 3D

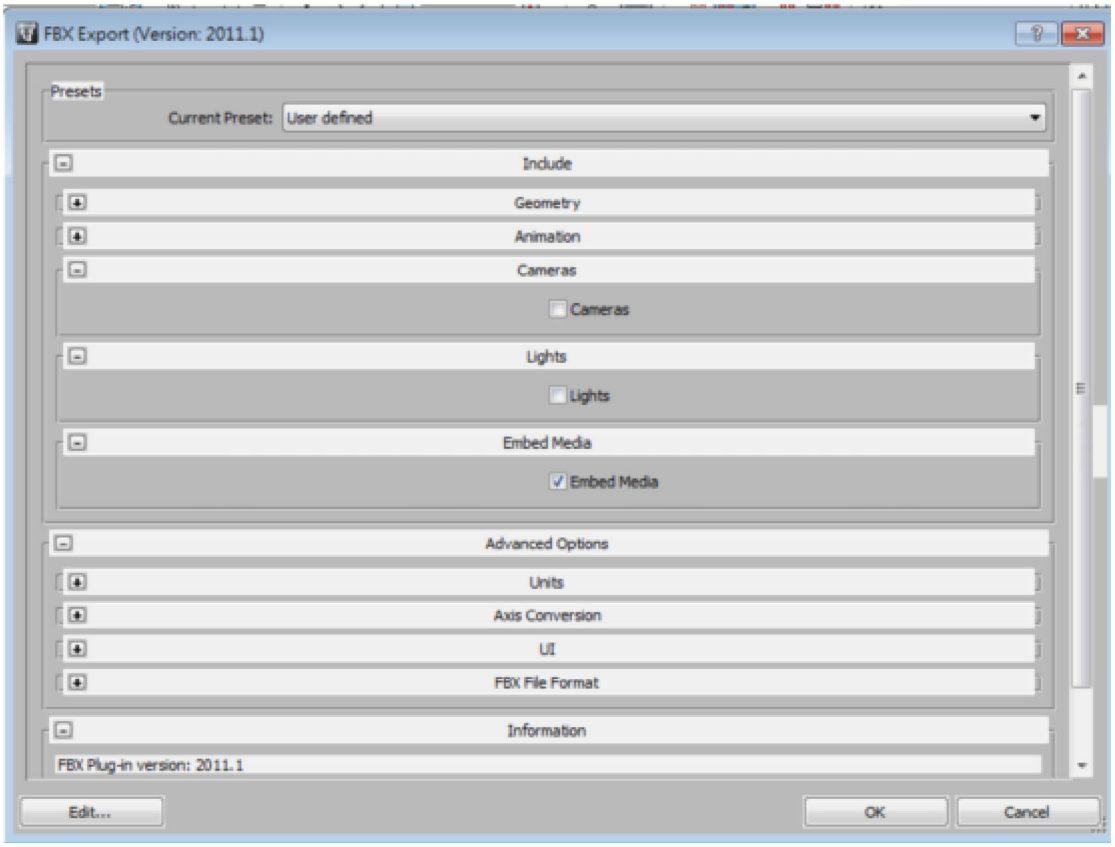
選擇選單Export/Export輸出角色



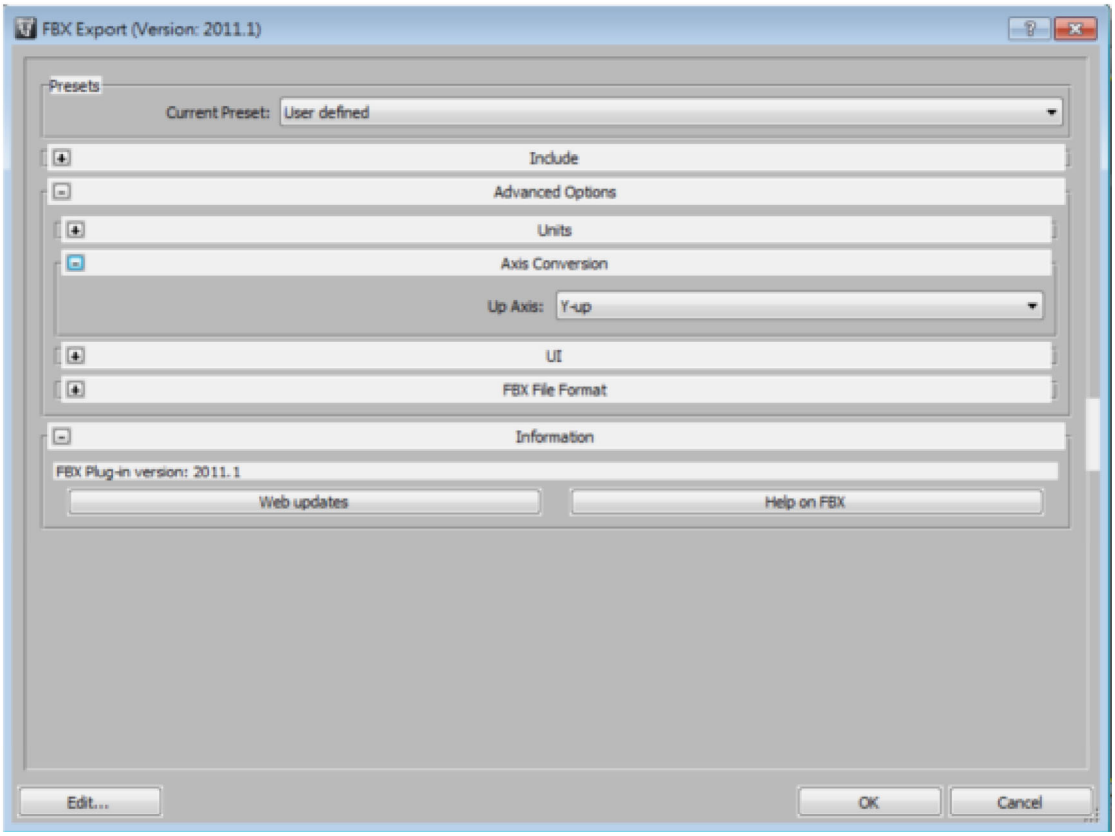
在Unity 3D的專案檔案夾匣U04中建立一個新的檔案匣Assests/Characters匯出命名為boy.fbx存檔。



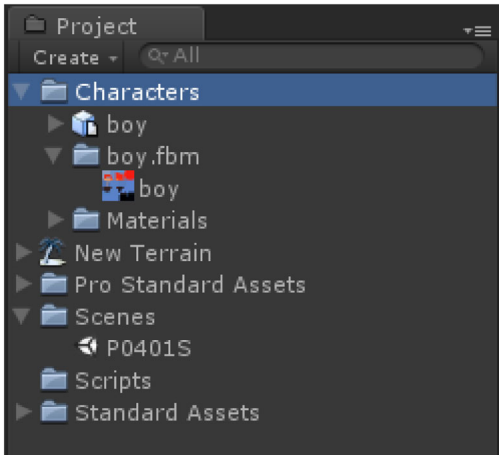
匯出時會出現匯出選單提供使用者選擇，在Include選項下共有Geometry，Animation, Cameras, Lights, Embed Media共五種資料可以匯出，在本範例中我們只要匯出角色的模型(Geometry)、動作(Animation)與內嵌媒體(Embed media)做使用，其餘的攝影機(Cameras)、燈光(Lights)可以不用輸出，所以把不用的項目勾選清空如下圖。



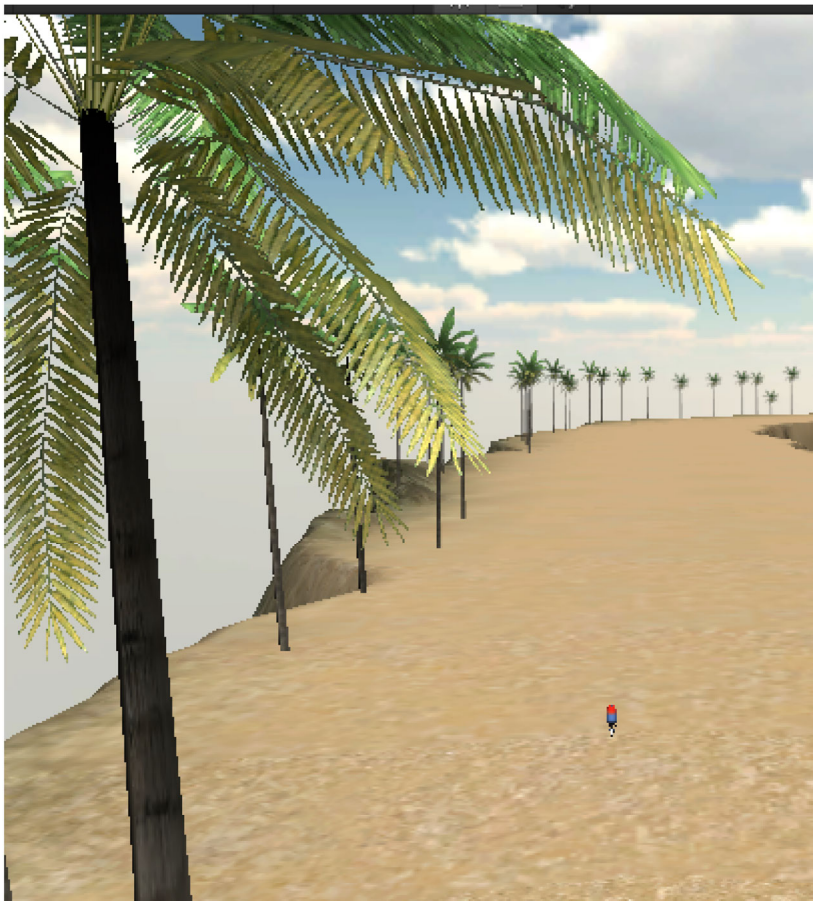
另外在進階選項(Advanced Options/Axis Conversion)中預設值使用Y-up，意思是角色的上視角為Y軸，所以在建立角色模型時要特別注意角色上視角是Y軸並且角色正前方向Z軸方向。



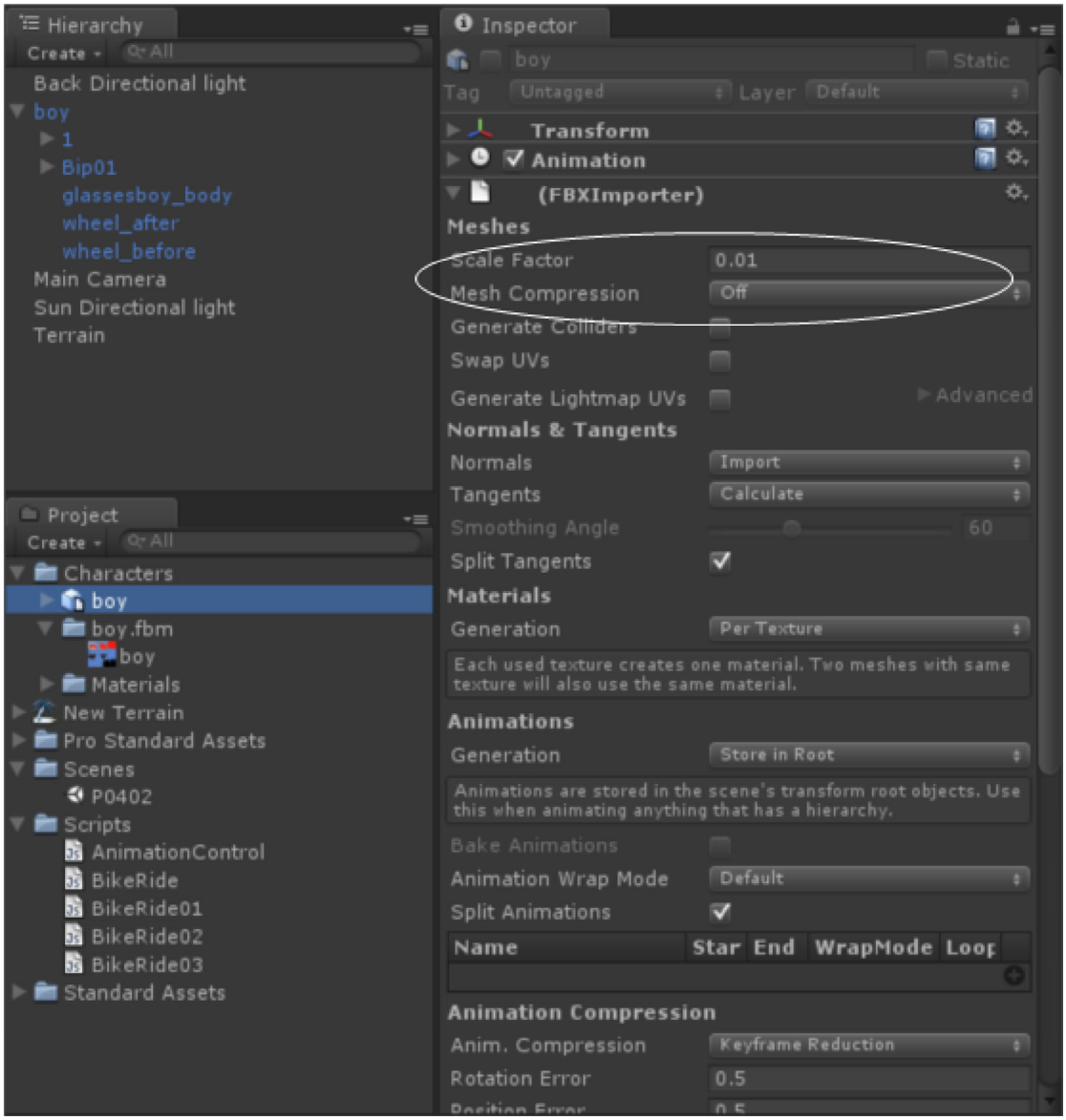
在Project/Characters檔案匣會出現boy預製物、內嵌媒體(boy.fbm/boy.psd)是boy角色使用的貼圖與Materials是角色使用的材質球。



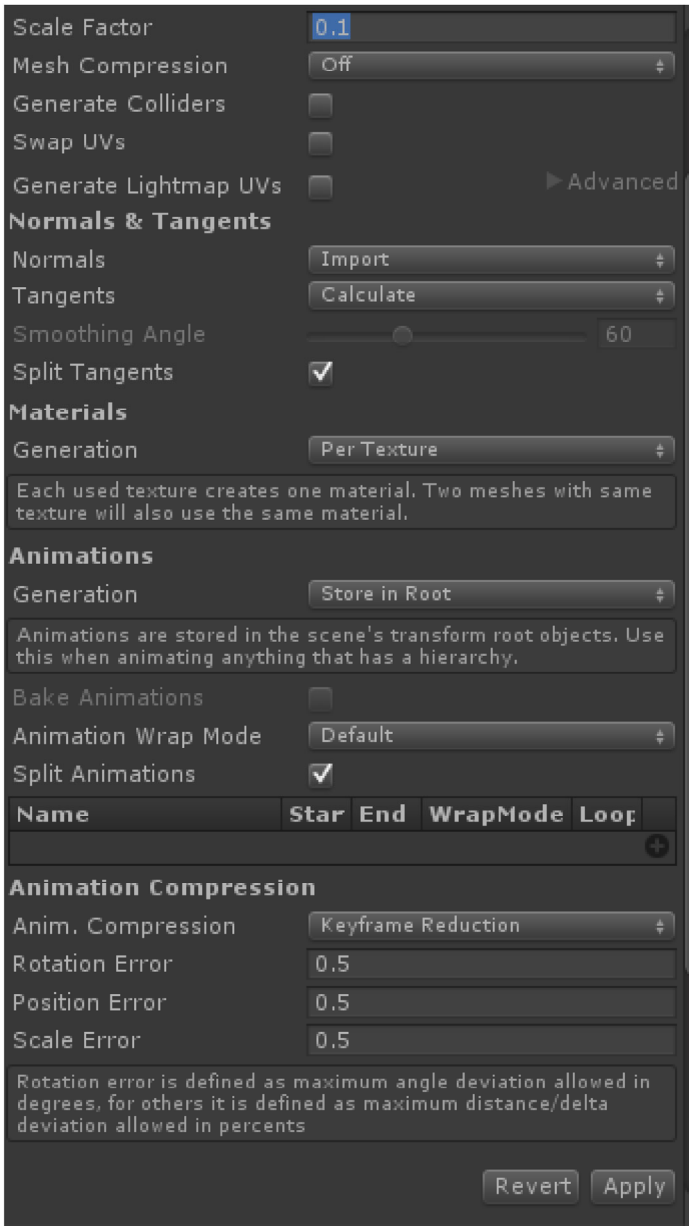
我們將boy預製物拉入場景中，會發現boy非常地小與場景樹木相對大小不對稱。



由選擇Project/Characters/boy之後出現在右方的Inspector/FBXImporter可以發現Scale Factor=0.01，這意謂著原始模型大小被縮小為0.01。



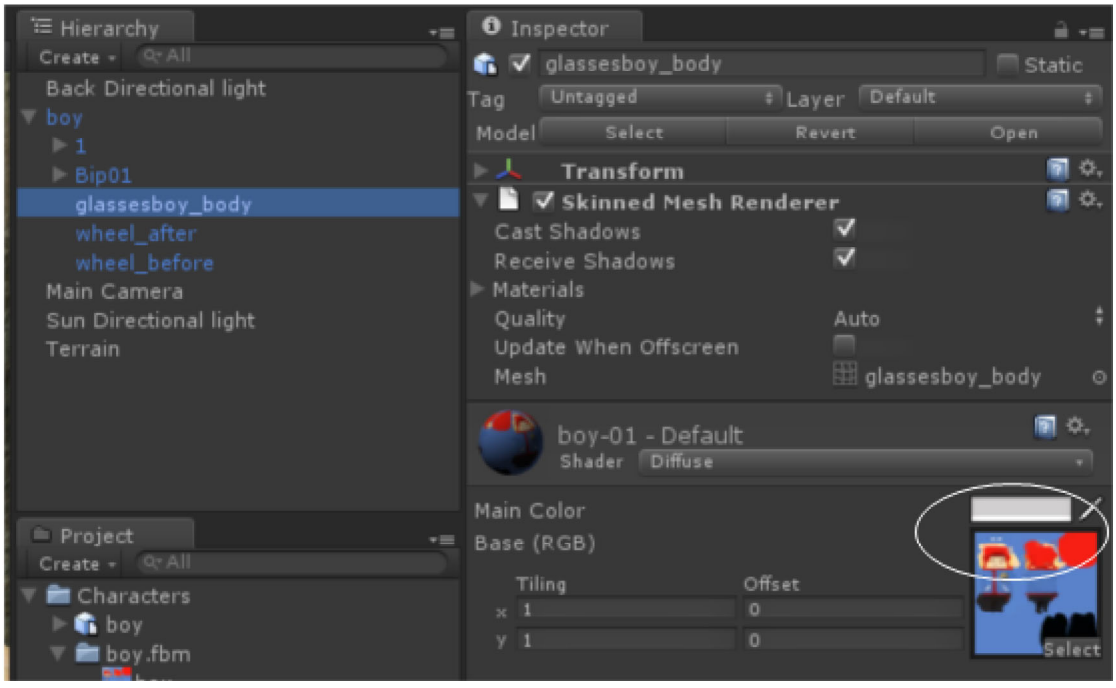
我們將此值更改為0.1並且按下右下方Apply(套用)。



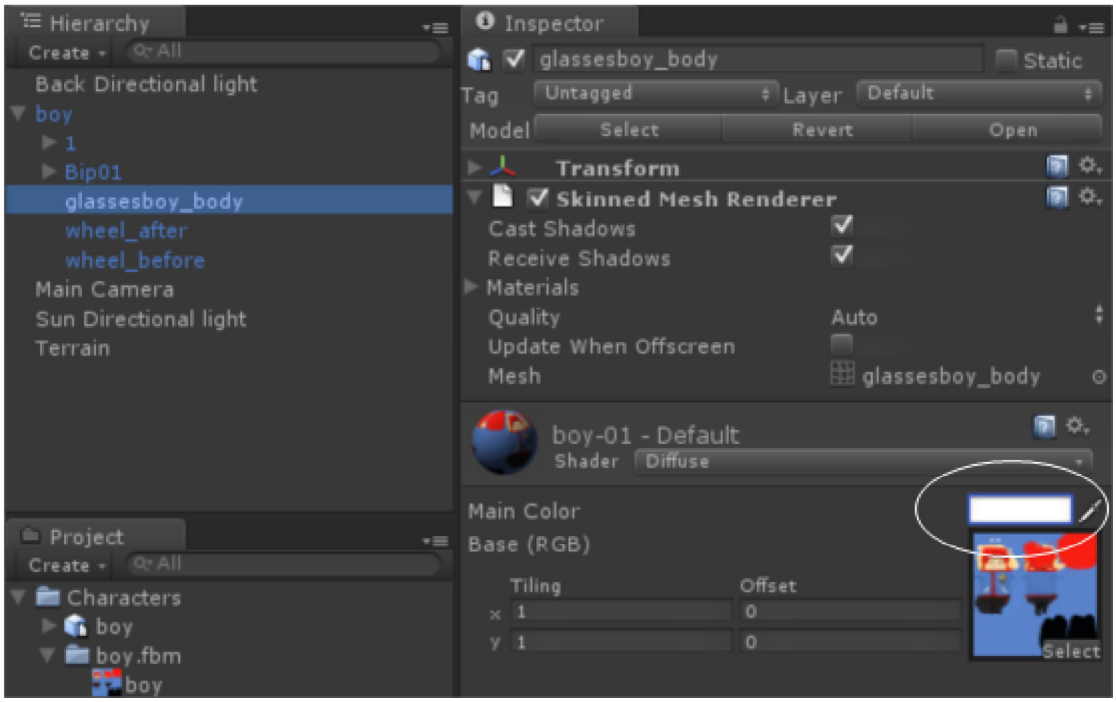
Boy被更新放大10倍如下圖所示。

4.1.3 調整角色貼圖亮度

當匯入boy角色之後，角色貼圖的亮度會稍嫌太暗，選擇Hierachy/boy/Bip01/glassesboy_body可顯示boy身體目前使用材質的狀態。從Inspector/Skinned Mesh Renderer可以看到boy使用boy-01-Default材質球，我們調整其下的Main Color可以控制貼圖的亮度。



提昇亮度。



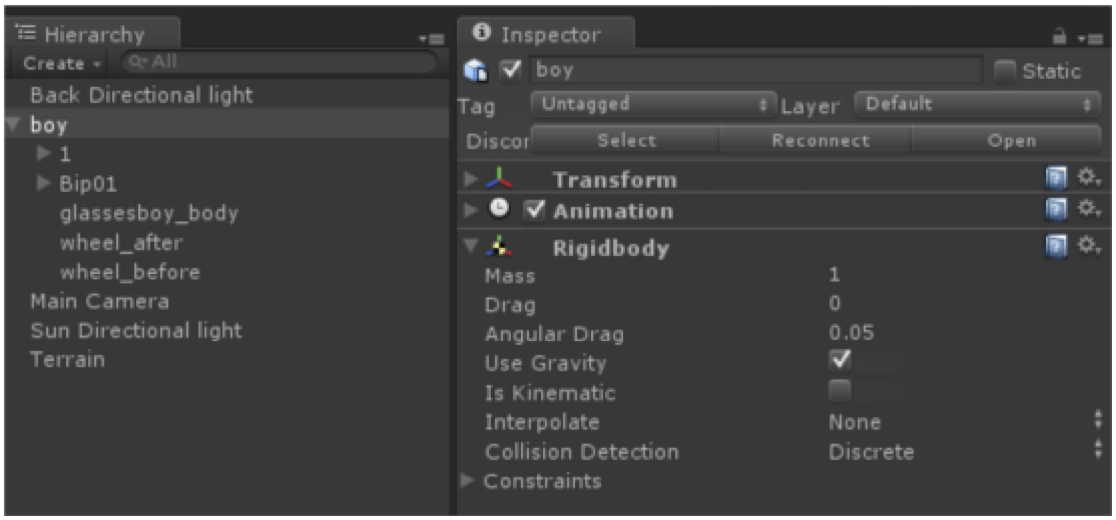
亮度提昇後如下圖所示。

4.2 加入角色物理特性

我們欲增加角色的重力與碰撞能力，做以下動作:

- 選擇Hierachy/boy遊戲物件
- 由選單中選擇Component/Physics/Rigibody加入boy

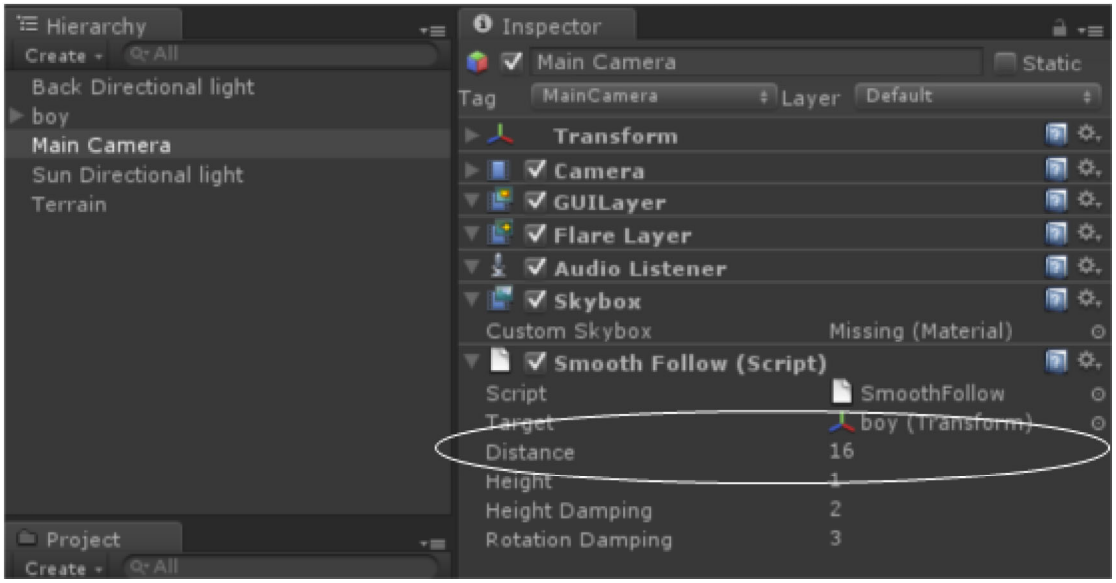
其結果如下圖所示，在boy物件上增加了Rigidbody屬性。



4.3 攝影機追隨角色

欲讓攝影機追隨角色請執行以下動作:

- 在Project中匯入套件(Import Package)動作腳本Scripts。
- 找出Project/Standard assets/Scripts/Camera Scripts/SmoothFollow腳本並套用到Main Camera。
- 選擇Main Camera遊戲物件，在Inspector/Smooth Follow Script下的Target屬性設定為boy遊戲物件。

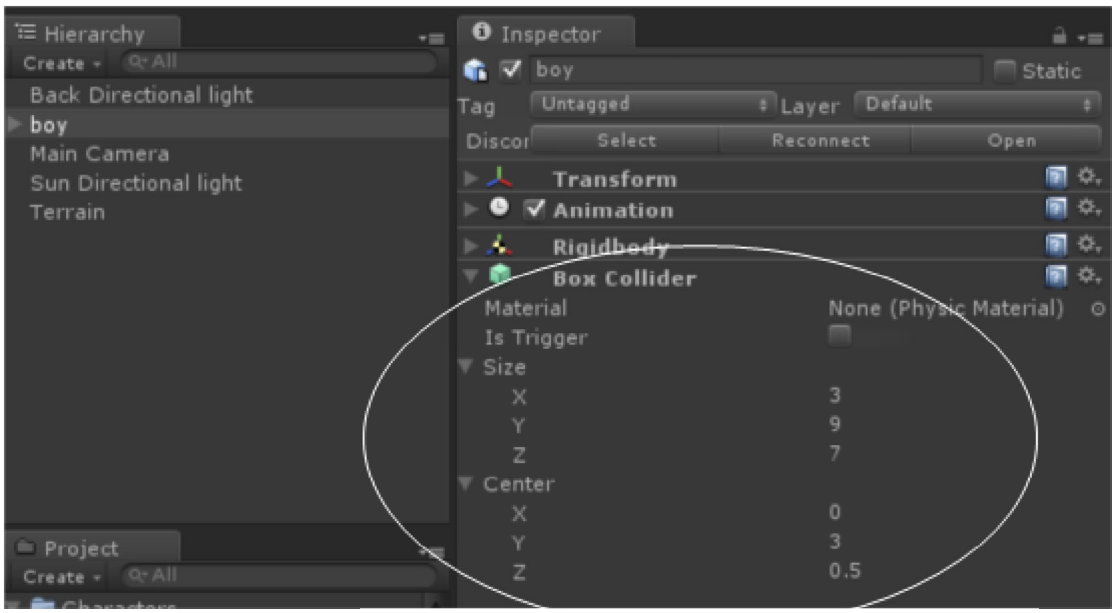


執行遊戲會發現鏡頭跟隨在角色背後，此時角色因有Rigidbody關係具備有重力，但是角色會穿透地形往下掉。

4.4 加入碰撞屬性

為boy角色加入碰撞屬性可以防止boy穿透地形掉入與週遭地形可產生碰撞，執行以下動作:

- 針對boy新增Component/Physics/Box Collider
- 在Inspector/Box Collider中調整參數Size與Center如下圖所示。

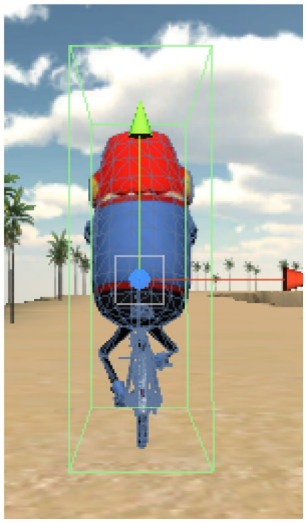


在boy外觀上新增一個透明的box collider罩住。

執行遊戲可以發現boy已可以站立在地形上，但我們發現鏡頭角度似乎太低。

4.5 調整追隨鏡頭視角

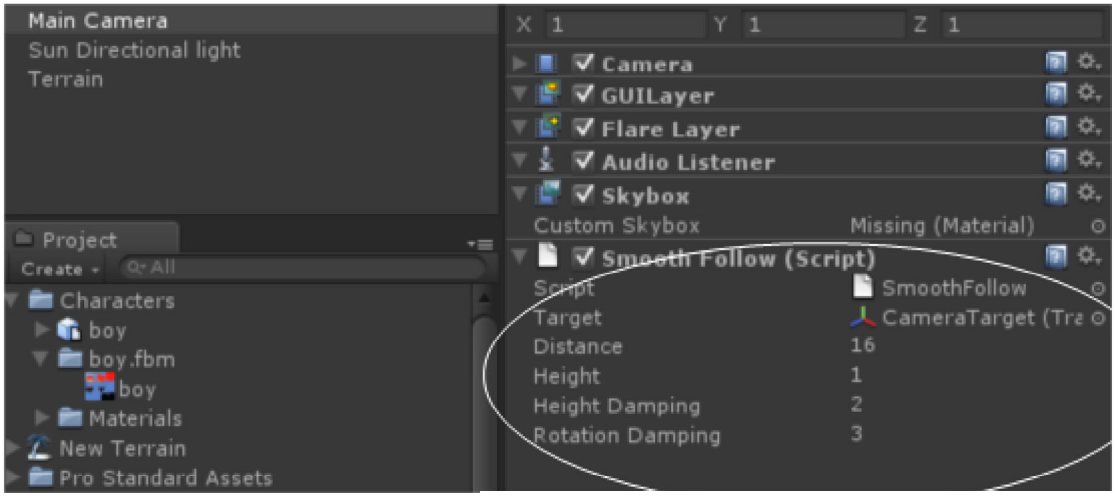
如下圖所示，讓我們觀察一下boy遊戲物件的transform中心點，可以發現位於臀部上方，當我們將Main Camera/Smooth Follow Script/Target設定為boy時，攝影機鏡頭就會指向boy的transform中心點，此結果會造成視角過低。



欲調整尾隨鏡頭的視角，做以下動作:

- 選取選單GameObject/Create Empty新增一個空的遊戲物件。
- 按下鍵盤F2更新物件名稱為CameraTarget。
- 將CameraTarget移入boy遊戲物件之下，調整其Transform座標為x=0, y=6, z=0(如下圖所示)。

- 調整Smooth Follow Script下Distance=16, Height=1



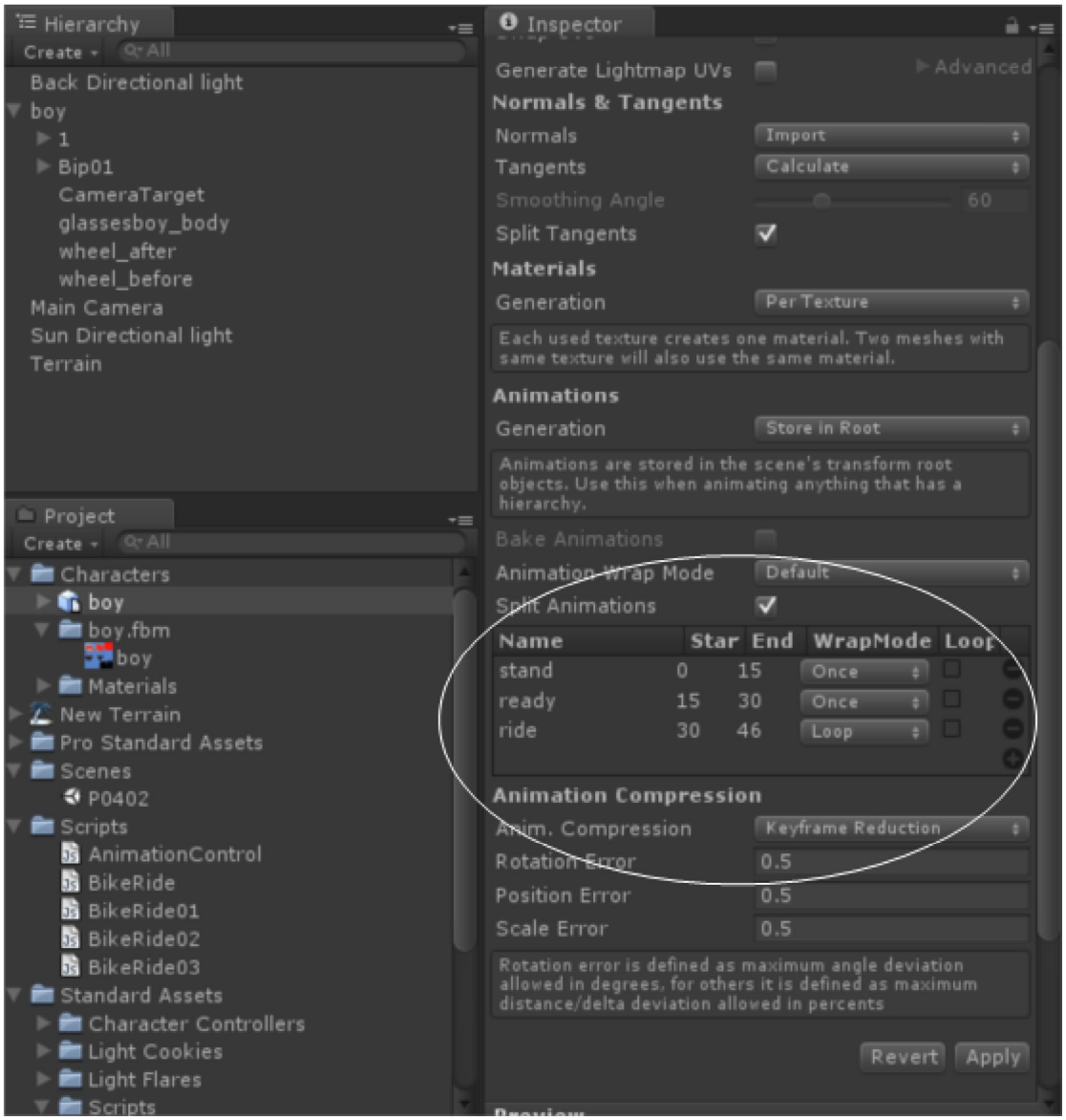
執行遊戲結果如下圖，角度正好看到全景與角色的全身。

4.6 角色動作設定

角色的動作設定分別要從Project的boy與Hierachy的boy做分別的設定，步驟如下:

- 選取Project/Characters/boy預製物，在Inspect/FBXImport之下使用”+”來新增三項動作的起始及結束影格後，選擇Apply進行套用。

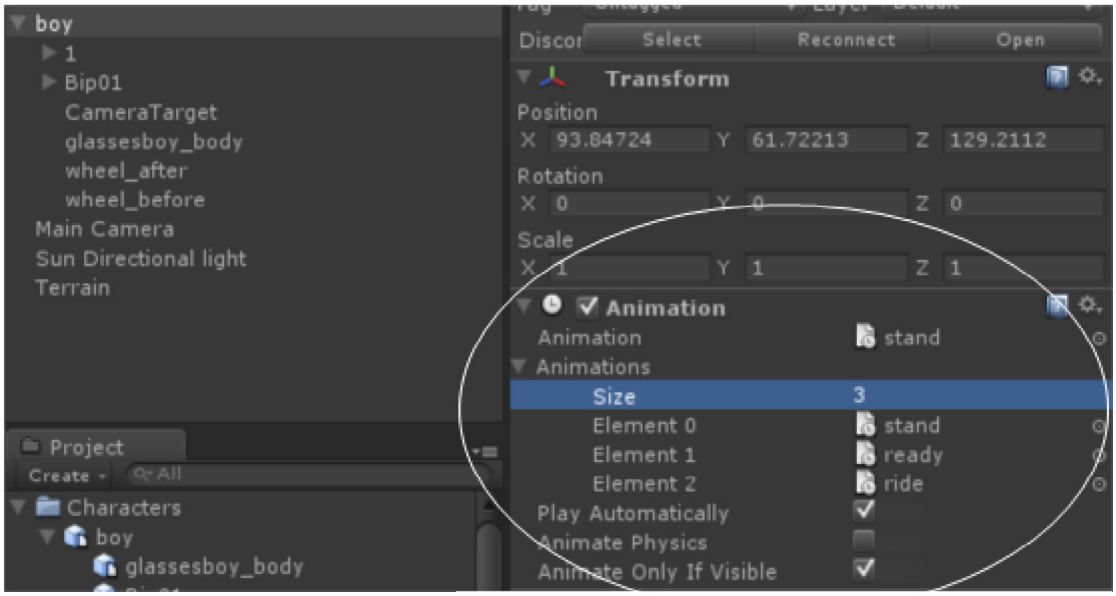
Name	**Start**	**End**	**WrapMode**
stand	0	15	once
ready	15	30	once
ride	30	46	loop



- 選取Hierarchy/boy設定Inspect/Animation如下圖。

Animation下的第1個參數Animation為預設動作，設定為stand。

Animations設定3個動作順序分別為stand, ready, ride。



4.7 角色動作控制

新增一個Javascript動作腳本命名為AnimationControl.js套用在Hierarch/boy上。

```
function Update () {
	if(Input.GetKey("0"))
		animation.CrossFade("stand");
	if(Input.GetKey("1"))
		animation.CrossFade("ready");
	if(Input.GetKey("2"))
		animation.CrossFade("ride");
}
```

使用animation.CrossFade函式輸入動作名稱即可播放該角色動作。本範例根據使用者所輸入0,1,2分別播放動作stand, ready, ride等三項動作做測試。測試結果可以發現當按下0做stand動作後若直接再按下2做ride動作會發現腳踏車並沒有正確地回歸位置，原因是stand的動作會將腳踏車左傾，若沒有透過做ready的動作並無法將腳踏車正確歸位，便造成ride動作人物與腳踏車定位錯誤。

4.8 製作角色向前的推力

本範例做以下動作

- 在Start函式中先播放ready動作
- 等待讀取向上方向鍵。
- 若使用者按向上方向鍵便執行rigidbody.AddForce(transform.forward*power)其中power為推力設定為200。Transform.forward為目前角色的正前方方向。

```
var power:float=200;
function Start()
{
	animation.Play("ready");
}
function Update () {
	if(Input.GetKey("up"))
		rigidbody.AddForce(transform.forward*power);
}
```

4.9 顯示速度

使用rigibody.velocity.magnitude可以顯示rigidbody運動中的純量值，此值可視為其運動速度。

```
function Update () {
	print(rigidbody.velocity.magnitude);
	if(Input.GetKey("up"))
		rigidbody.AddForce(transform.forward*power);
}
```

透過print可以將速度顯示在遊戲視窗左下角位置。

4.10 左右轉與傾斜運動

我們重新宣告4個變數，power為推力，speed為rigidbody目前的運動速度，direction為轉動力矩，inclination為腳踏車轉向時的傾斜度。

```
var power:float=50;
var speed:float=0;
var direction : float;
var inclination : float;
```

```
function Update () {
	var _deltaSpeed:float;
	speed=rigidbody.velocity.magnitude;
	_deltaSpeed = Mathf.Clamp01(speed/30);
	if(Input.GetKey("up"))
		rigidbody.AddForce(transform.forward*power);
	if(Input.GetKey("right"))
	{
		direction += 1.2*_deltaSpeed;
		inclination = Mathf.Lerp(inclination,-40*_deltaSpeed,Time.deltaTime*2);
	}
	if(Input.GetKey("left"))
	{
		direction -=1.2*_deltaSpeed;
		inclination = Mathf.Lerp(inclination,40*_deltaSpeed,Time.deltaTime*2);
	}
	inclination = Mathf.Lerp(inclination,0,Time.deltaTime);
	transform.rotation.eulerAngles.z = inclination;
	transform.rotation.eulerAngles.y = direction;
}
```

- _deltaSpeed = Mathf.Clamp01(speed/30);其中Mathf.Clamp01把speed/30後的值限制輸出為0~1的值，若speed/30大於1則輸出1，若小於0則輸出0，若介於0~1之間則輸出原值。
- 使用者按下向右鍵則執行

direction += 1.2*_deltaSpeed;目前的方向轉矩做正向遞增

inclination = Mathf.Lerp(inclination,-40*_deltaSpeed,Time.deltaTime*2);傾斜度為目前inclination與-40*_deltaSpeed值的內差。此方式會讓腳踏車做較緩慢地向右傾斜。

- 使用者按下向左鍵則執行

direction -=1.2*_deltaSpeed;轉向累加相反值。

inclination = Mathf.Lerp(inclination,40*_deltaSpeed,Time.deltaTime*2);傾斜度向左傾斜。

- 當腳踏車傾斜後，我們採用inclination = Mathf.Lerp(inclination,0,Time.deltaTime);做傾斜度逐漸歸零的動作，可讓腳踏車回正
- 傾斜度可利用設定eulerAngles.z軸設定；腳踏車轉向可利用eulerAngles.y設定。

```
transform.rotation.eulerAngles.z = inclination;
transform.rotation.eulerAngles.y = direction;
```

4.11 設定阻力與凍結彈跳

Rigidbody下有一個參數Drag設定為1可以增加地面的摩擦阻力。另外為了防止腳踏車碰撞石頭造成翻車，我們設定Constrants/Freeze Rotation將x,y,z都勾選起來可以凍結碰撞翻滾。