行動APP設計

E-portfolio

ADT106110_林子涵

Week 01

【前導基礎知識】

◆ 什麼是APP？

全名 Application，專指行動裝置上的應用程式。行動裝置則是指舉凡可隨身攜帶，擁有運算能力的裝置

◆ 數位學習與APP的關係

Elearning → mlearnig → ulearning
mlearnig：mobile learning
Ulearning：無所不在的學習

◆ APP+數位學習+遊戲

遊戲比重，低至高

嚴肅遊戲：設計主要目的非純粹是娛樂的遊戲，較強調趣味與競爭性帶來的教育價值。
遊戲式學習：挑選學習主題去做遊戲。
遊戲化：好玩優先，但依然能從中學習到知識。

◆ Unity相關

- 安裝課程要求版本的Unity：2018.3.5
  下載unity hub→安裝→下載版本
- 套用版本
  專案→新專案→下拉選單→選擇版本
- 增加可製作安卓app的設定
  新增模組→打勾Android Build Support 1
- 其他裝置設定
  WebGL Build Support　（網遊用）
  Windows Build Support（PC Game）
- 其他
  更改版本時，記得備份，以免出錯導致無法開啟專案。

Week 02

【課程分數紀錄】

【Unity的注意事項】

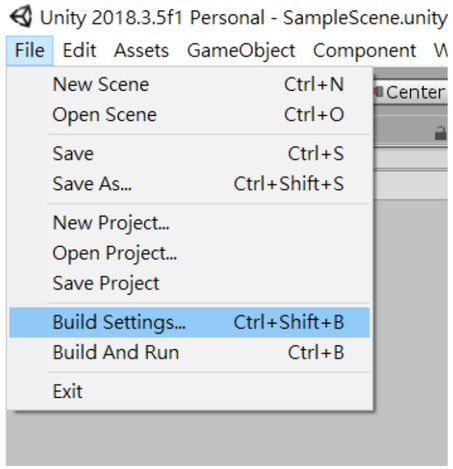
1. Unity的目錄除了Assets外都不能動，否則Unity會混亂
2. 養成個人習慣創建好資料夾並確實做好命名

【切換平台】

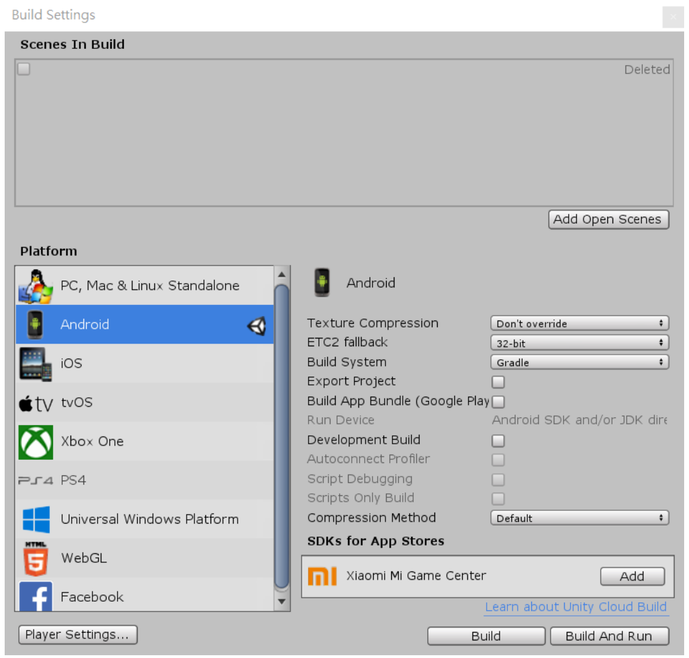
在開始一個新專案時，就要在Build setting裡決定platform：PC or Android
就算沒選之後依然能改但轉換過程相當費時還容易當機

---

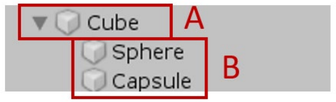
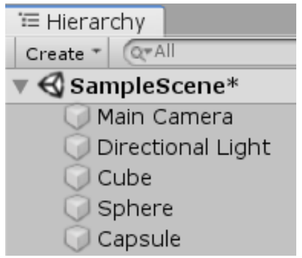
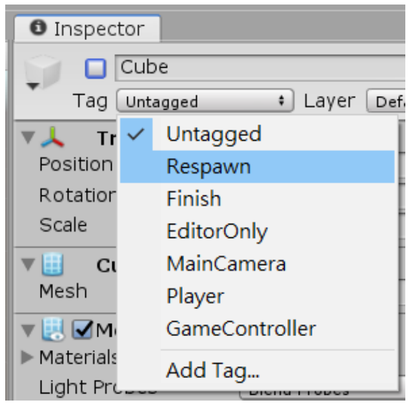
1. 從file找到Biuld setting



2. 預設為PC，將其轉為Android
選擇Android，點右下Switch Platform後變成下圖
(由原本的Switch Platform→Build)



【功能】

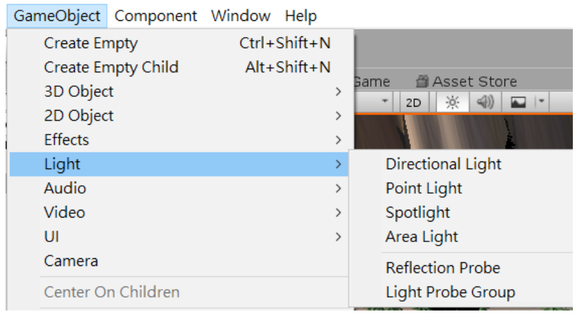
名稱	功能	圖示
依存	將物件托放到某物件中會依存，當B依存A，A動則B動	
物件順序	上下順序有差，在序列中的位置代號不同	
物件存在但沒作用	1. 關掉Mesh Renderer：物件會變透明，但它的物理效果還在 2. 關掉名稱旁勾勾，變成Disable(物理效果也消失)	
一對多	貼標籤，既屬於某物件底下也屬於某標籤底下	

【GameObject】

空物件，僅位置資訊。

【Light 光】

位置：



Directional Light 太陽光

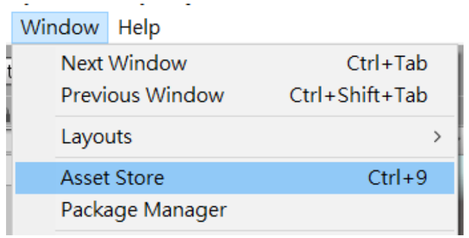
1. 位置不影響光影，僅方向有影響
2. 可以拖到物件底下變成探照燈

Point Light 點光源

1. 位置影響光影

【Assets Store】

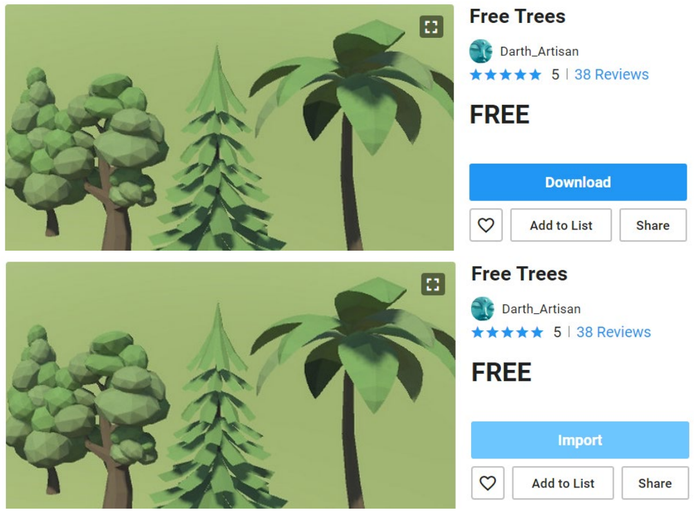
位置：

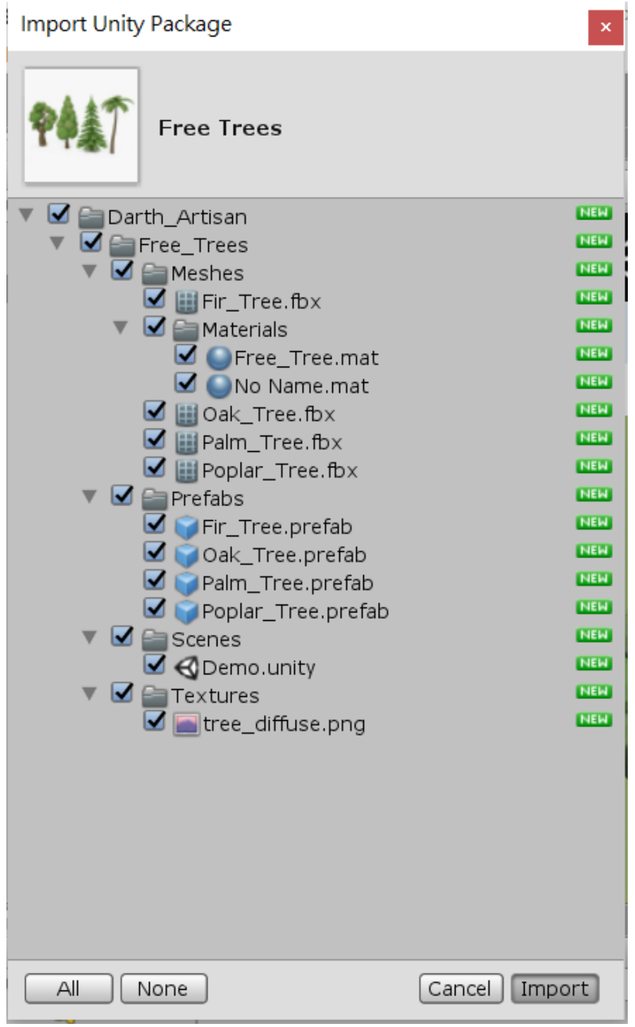


可以在Assets Store下載各種素材來使用

只需要下載一次，在每次開新專案Import就可使用

因為素材占空間量大，匯入簡單，取消匯入難，所以要謹慎思考匯入必要性。



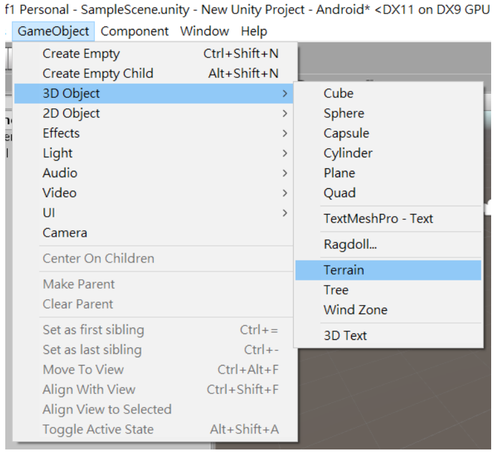


Week 02

【Terrain 地面】

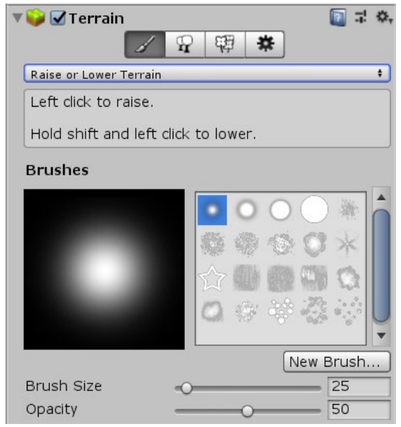
和Plane不同，但Plane可以加Component變成Terrain

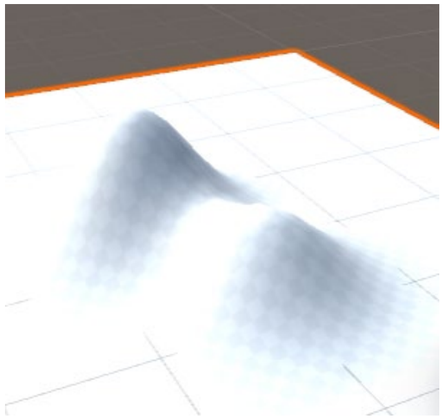
位置：



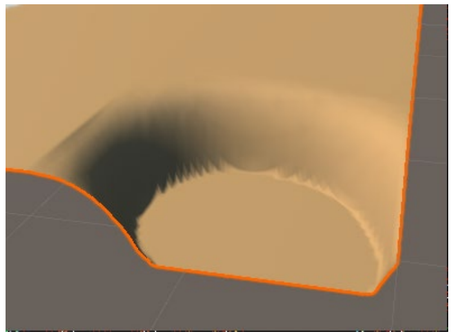
【用筆刷改變地形】

在Terrain中選擇筆刷，下拉式選單選擇Raise or Lower Terrain 升高與降低地形



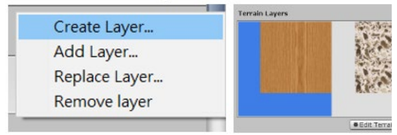


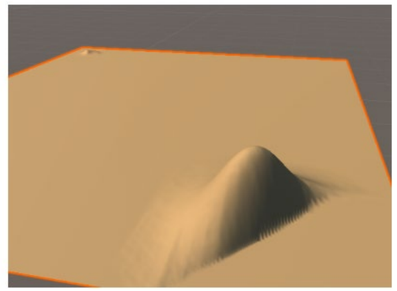
建立Terrain時的平面為最低面，若要製作凹地效果，就要在一開始將整個地面拉升或是用Set Height設定高度拉升地面後再使用



【Paint Texture 繪製材質】

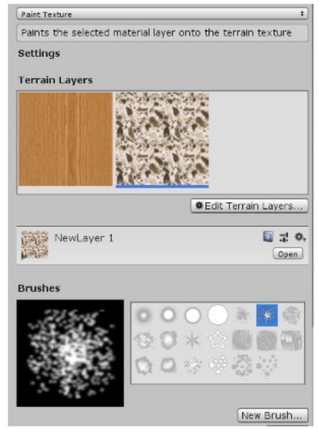
上網找無接縫材質，下載後放入Texture的資料夾接著就會出現在Unity中

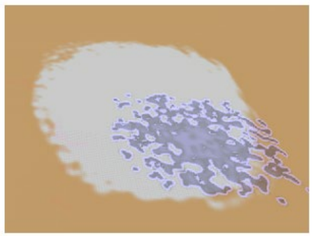




繪製材質

選擇要繪製的材質與筆刷直接畫在地面

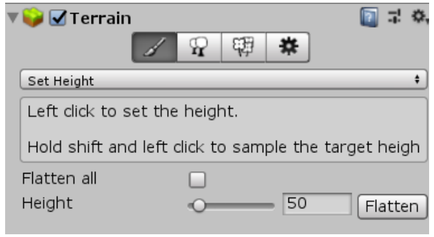




Week 02

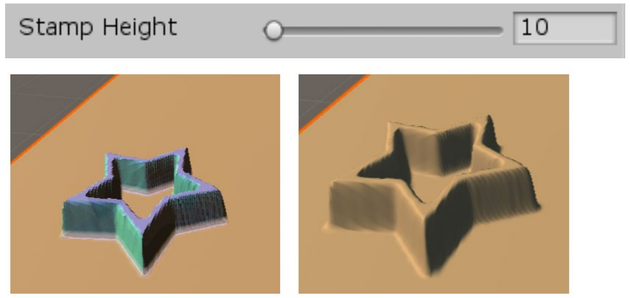
【Set Height 高度設定】

按Flatten會直接上升地面，通常用於確定低地很多時



【Stamp Terrain 蓋章】

可以直接蓋地形，記得設定高度

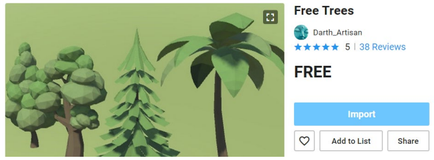


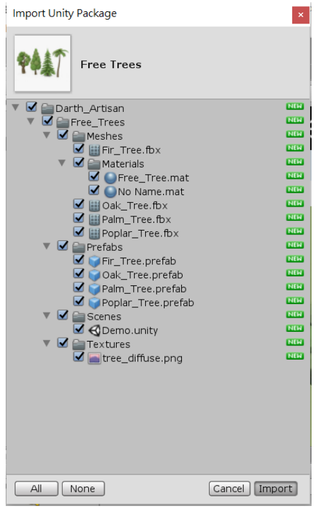
Opacity(強度)等於高低

【下載樹木素材】

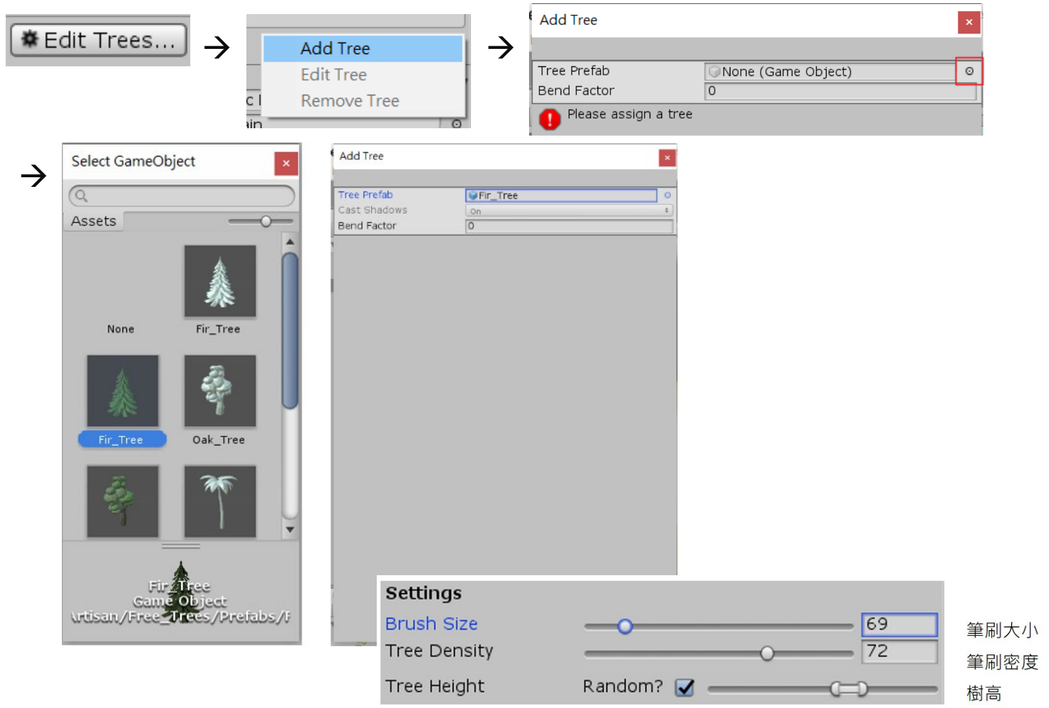
先到Assets Store下載免費樹木素材

下載後Import進專案

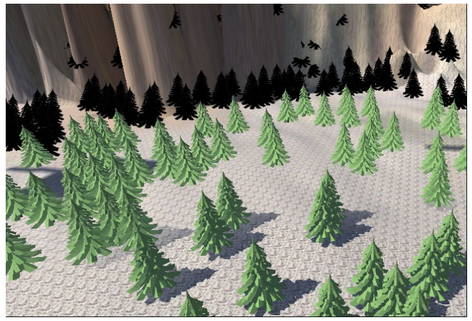




【增加樹木筆刷】



【完成圖】

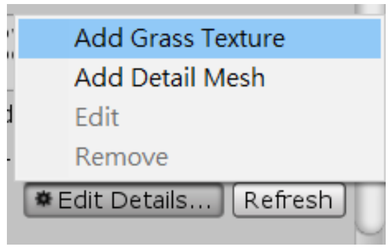


Week 02

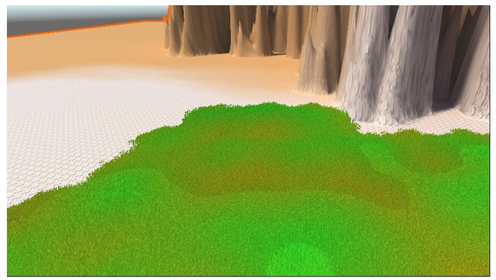
【Paint Detail 繪製細節】

- **Add Grass Texture**

  Edit Details-->Add Grass Texture (只要是去背的圖即可)

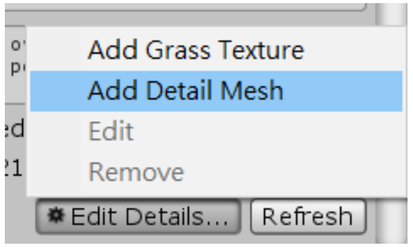


到Assets Store下載免費素材，Import→繪製



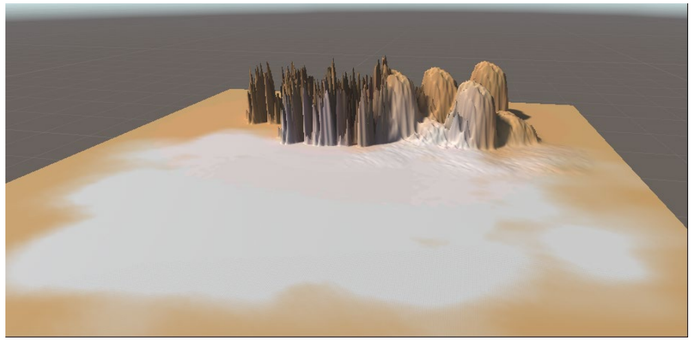
- Detail Mesh(盡量不使用，因為會影響效能)

  先繪製一個Sphere(縮小一點)
  Edit Details→Add Detail Mesh



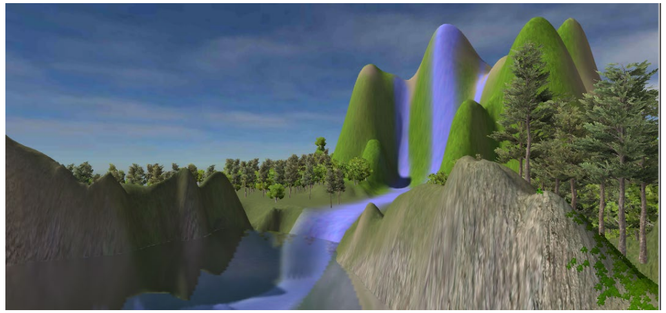
選擇3D物件(小圓點)-->選擇Sphere-->繪製

【課程成果】



【作業】

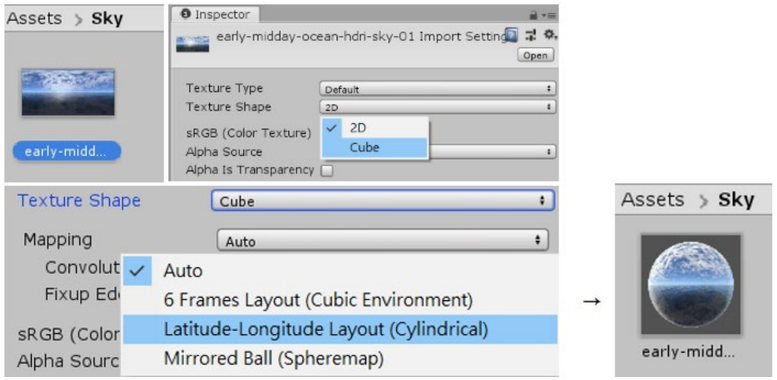
建置一個場景



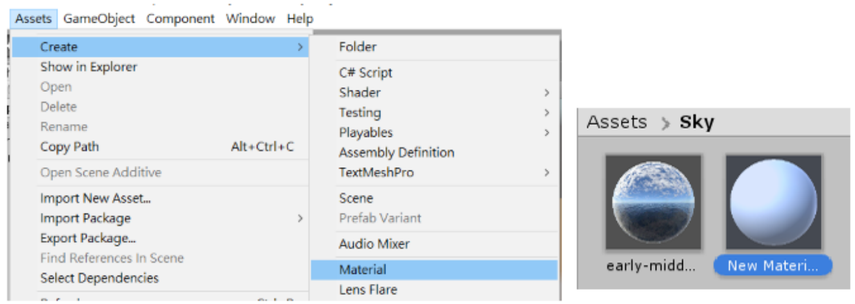
Week 02

【環景天空製作】

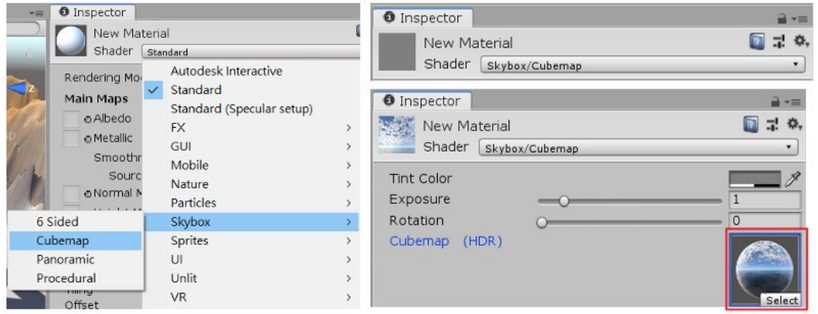
1. 上網搜尋HDRI存檔後，在專案的Assets裡，開一新資料夾，將HDRI的圖檔放進去
2. 在unity中選取HDRI圖片，修改Texture Shape的2D為Cube；Mapping選擇Latitude-Longitude Layout(Cylindrical)之後按Apply就會形成材質球



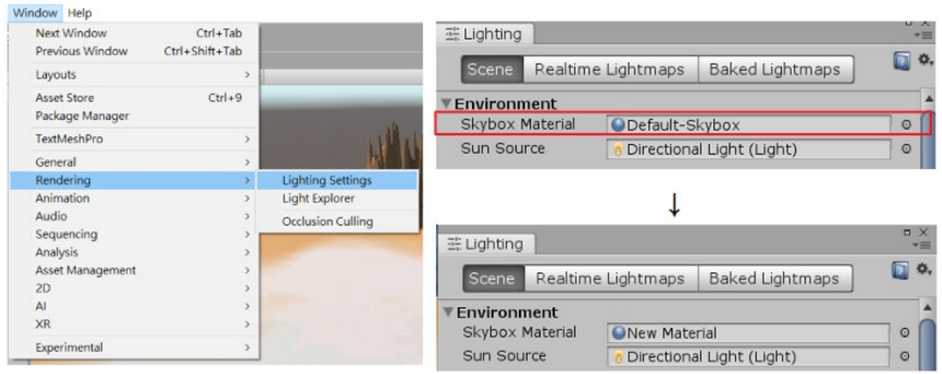
3. 新建一個Material



4. 將Material的Standard改為Skybox/CubeMap，並指定剛剛做好的材質球(紅框部分雙擊即可選取)



5. 在Window-->Rendering-->Lighting Settings，將Skybox Materia指定為剛才做的材質球完成



參考資料

http://w3.uch.edu.tw/ouyang/vr/Unity_02.html

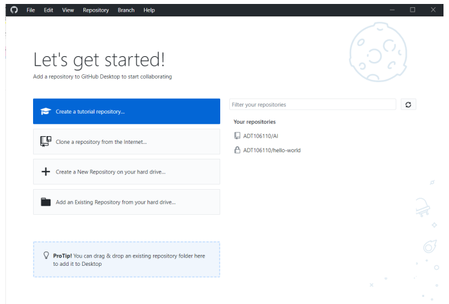
https://blog.csdn.net/AMalways/article/details/82322047

Week 03

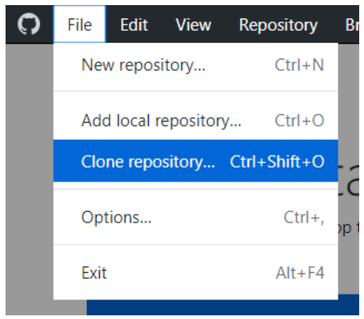
【視訊測試】

[Github匯入] 作業b1

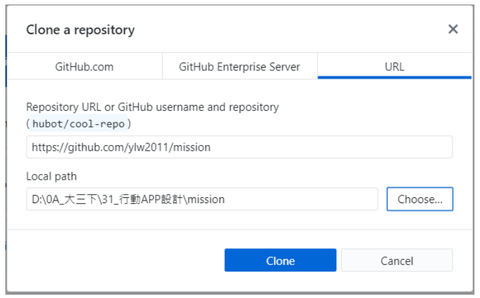
1. 開啟Github



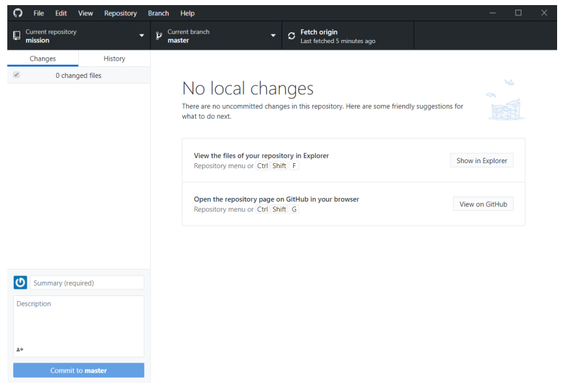
選擇File-->Clone



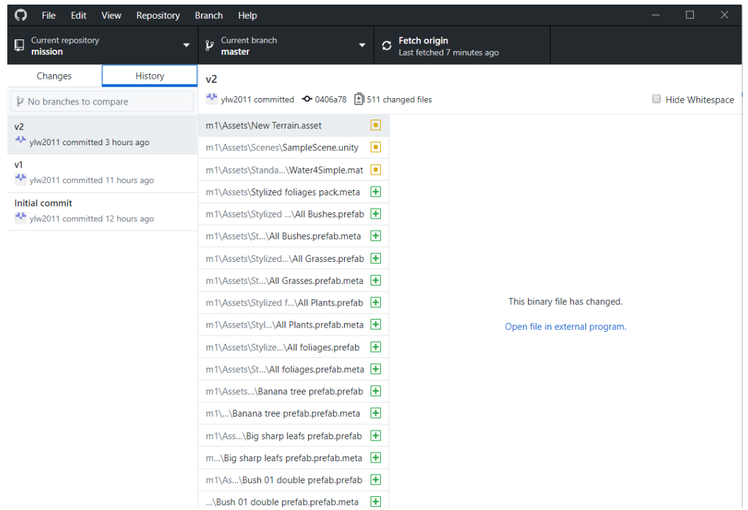
2. 選到URL，貼上FB社團的網址並選擇要匯入的資料夾，點選Clone



3. 等進度條跑完後會出現



4. 點選History截圖

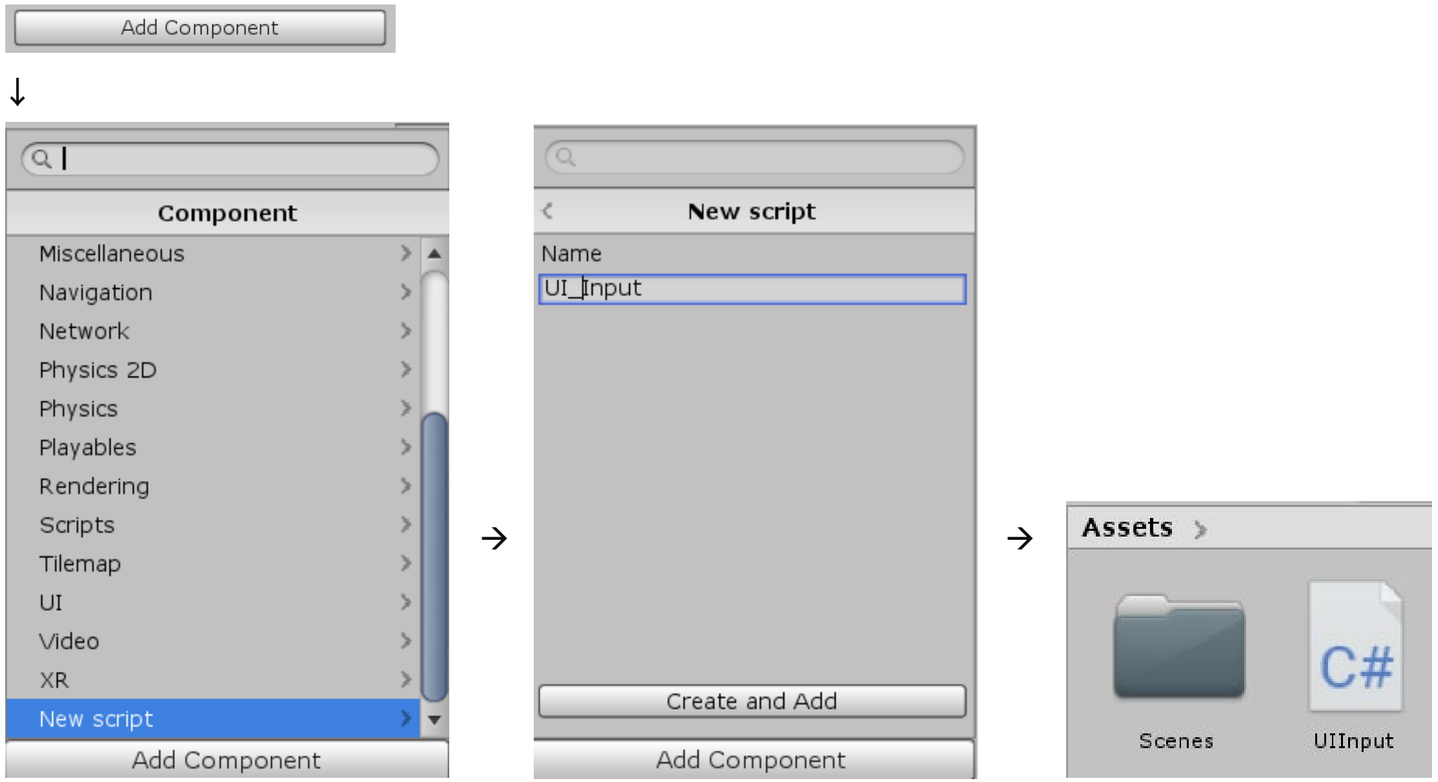


【程式碼】

程式碼名稱盡量不要更改，會影響很多地方

程式碼名稱相當於一個標籤，例如int

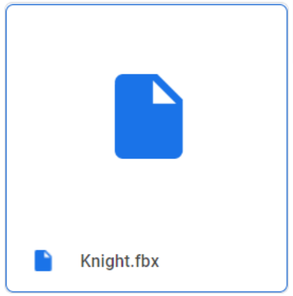
【建立程式碼】



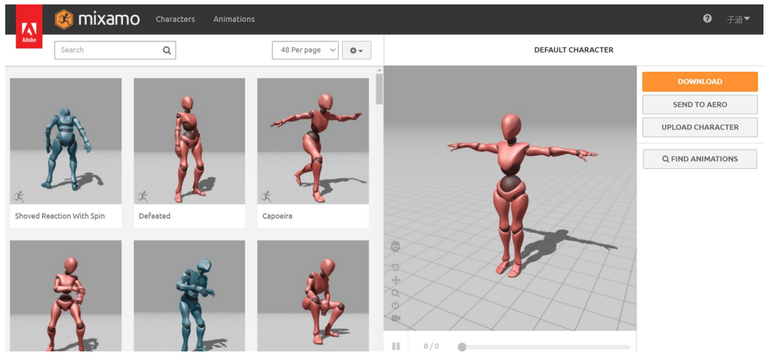
Week 08

【Mixamo】

先下載課程雲端中的



登入mixamo



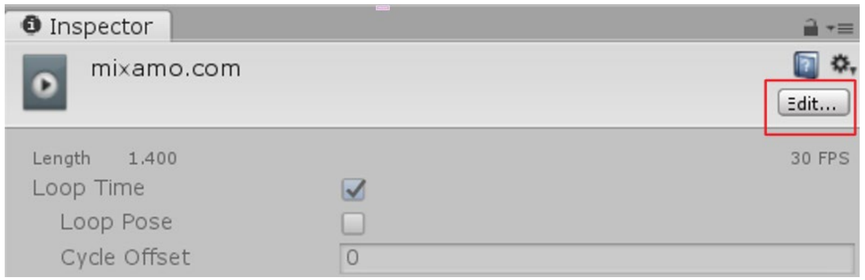
【設定循環】

1. 選擇要修改的動作區塊

2. 在右方Inspector欄中找到

3. 就能連結到assets裡的動畫的物件

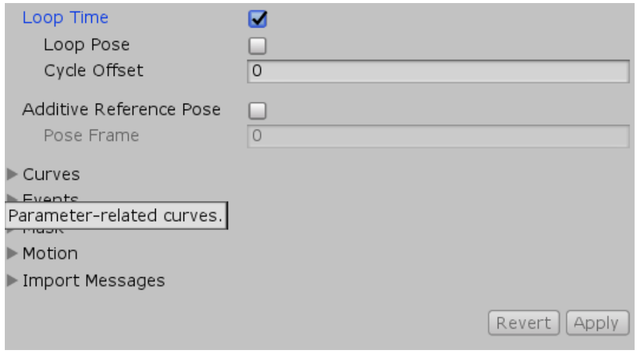
4. 在右方Inspector欄中選擇edit…



5. 選擇animation

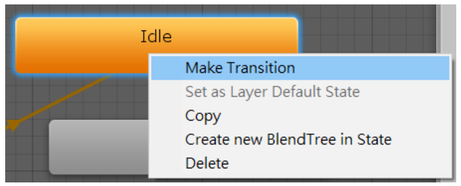
6. 下方有個Loop Time

勾選後記得Apply

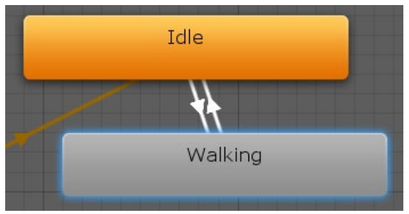


【Bool建立與條件增設】

◆ 增加動作連結

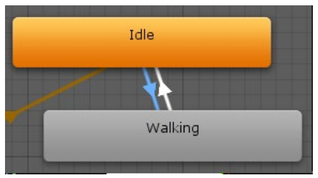


在idle區塊右鍵，選擇make transition，連結idle與walk(雙向)

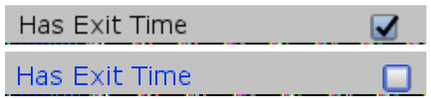


◆ 消延遲

消延遲：讓待機不用播完就能執行走路動作
點擊idle→walk的箭頭

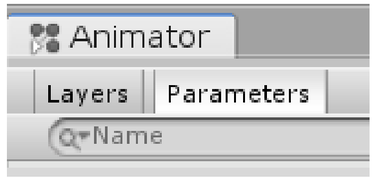


在右方Inspector欄中找到下方敘述，拿掉勾

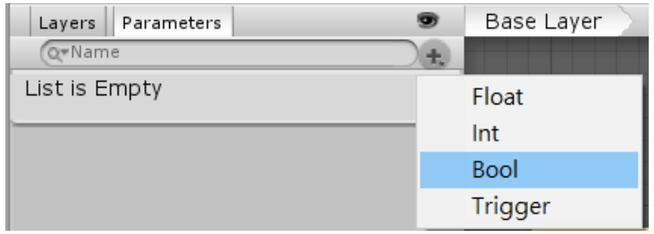


◆ 增加bool

在animator下方，切到parameters



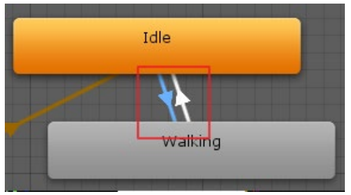
在搜尋框右方有個+號，點擊後選擇bool



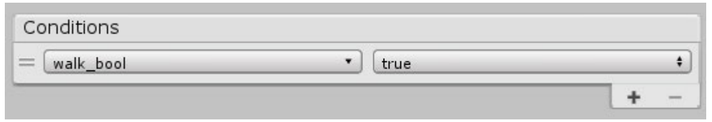
命名自己看得懂的名稱

◆ 增加動作行使條件

點擊箭頭



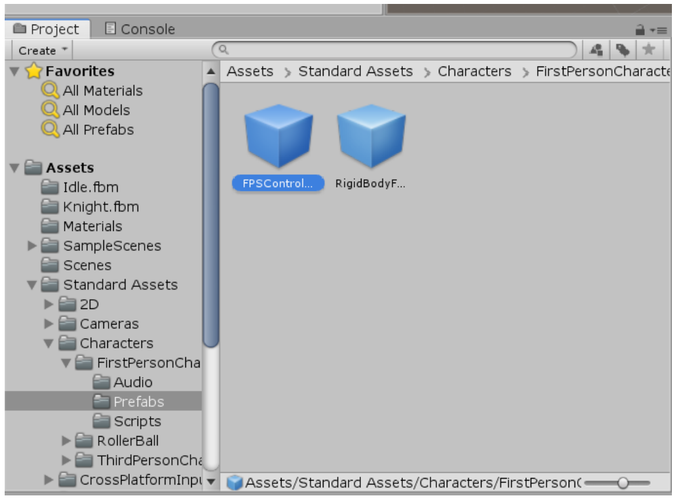
在右方Inspector欄中找到conditions，按+號就會增設一個條件

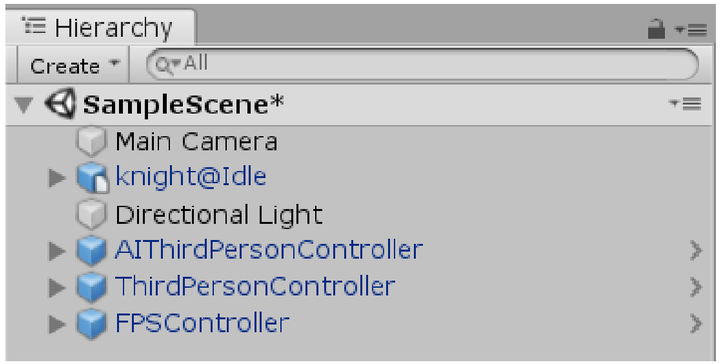


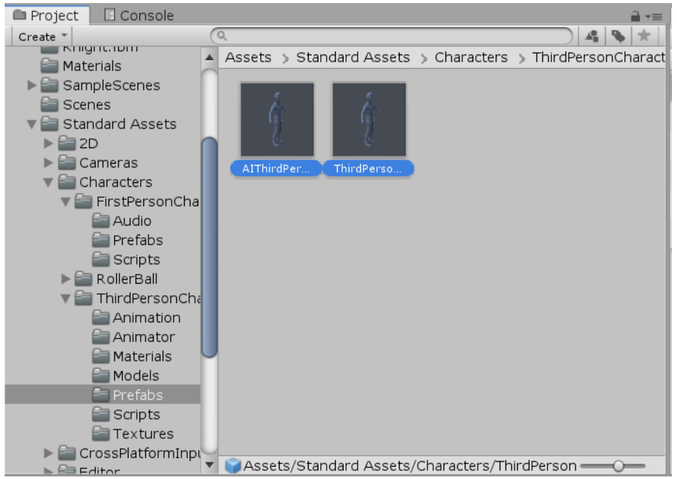
*左方箭頭設定walk→true；右方箭頭設定walk→false

【匯入standard assets基礎人物模型】

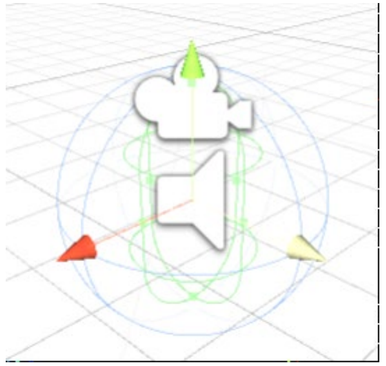
從standard assets中匯入以下三項物件







【FPSControler】

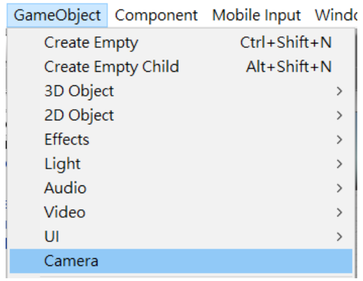


1. 綠色部分為身體，會有實際物理碰撞
2. 喇叭能固定發出聲音，例如移動時的腳步聲
3. 可隨意調整大小
4. 若無法正常運行，例如人物飄移，那是因為沒有放Terrain，放了就正常了
5. 差異：FPSControler：沒有人物模；
   ThirdPersonController：有人物模

Week 09

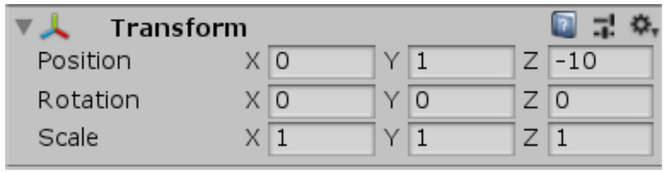
【製作追蹤人的攝影機】

1. 使用Main Camera或是新增一Camera

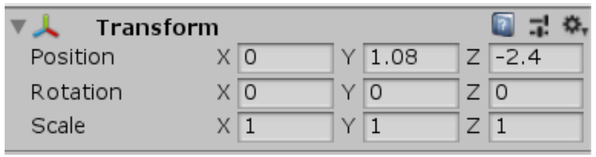


2. 將攝影機拖動到要追蹤的物件上即可(即依附)

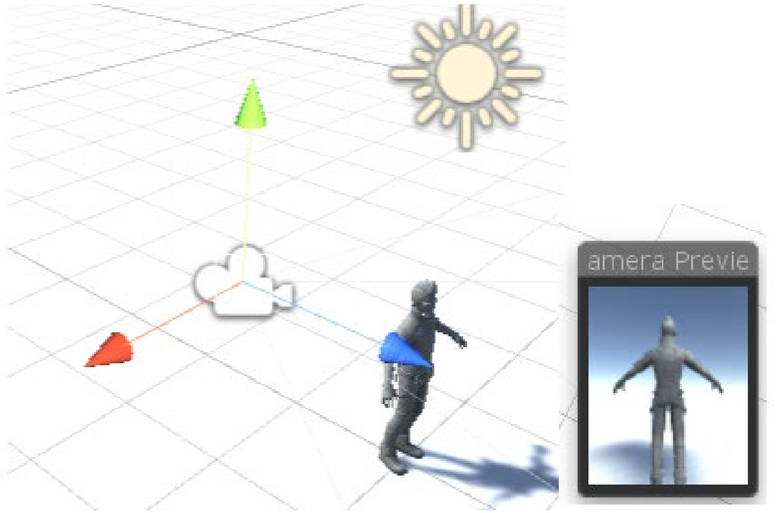
3. 原本的Camera座標為絕對座標，但依附後座標會改變為依附物的座標，此時將座標改為(0,0,0)後，再做微調至拍攝最佳視野



4. 調整：



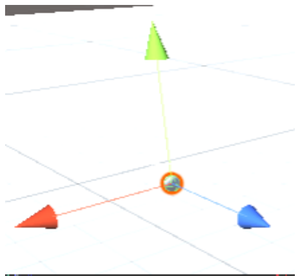
【Play操作】
*若Play時，無法正常運行
那是因為FPSControler的攝影機干擾，關掉FPSControler即可



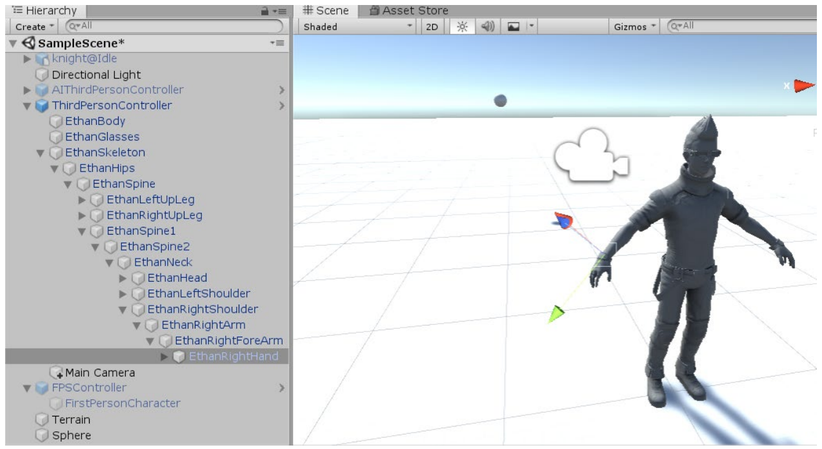
Week 09

【置入骨架物件】

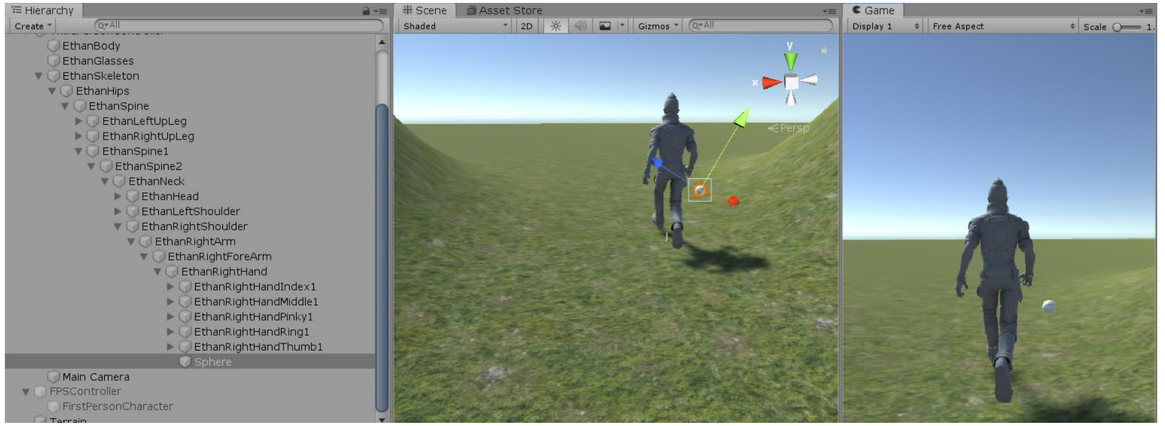
1. 先建立一個要拿的物件



2. 找到人物右手的骨架

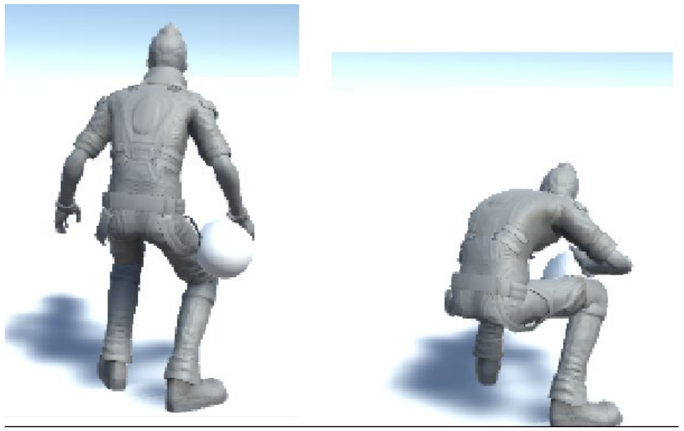


3. 將物件依附在右手的骨架底下，並微調位置後測試



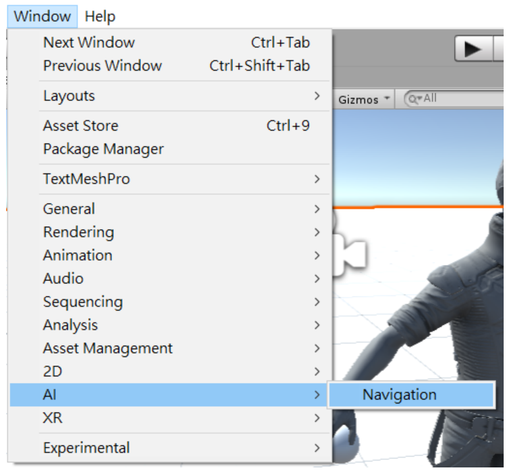
★若人物在Play時會不斷蹲下，就是物件導致碰撞造成

目前自己的解法是將物件遠離人物模型

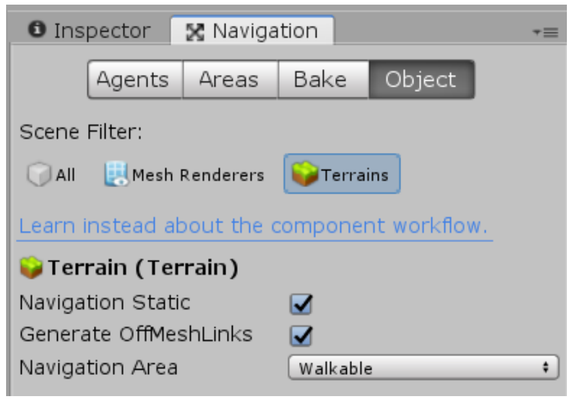


【Navigation 判斷地形】

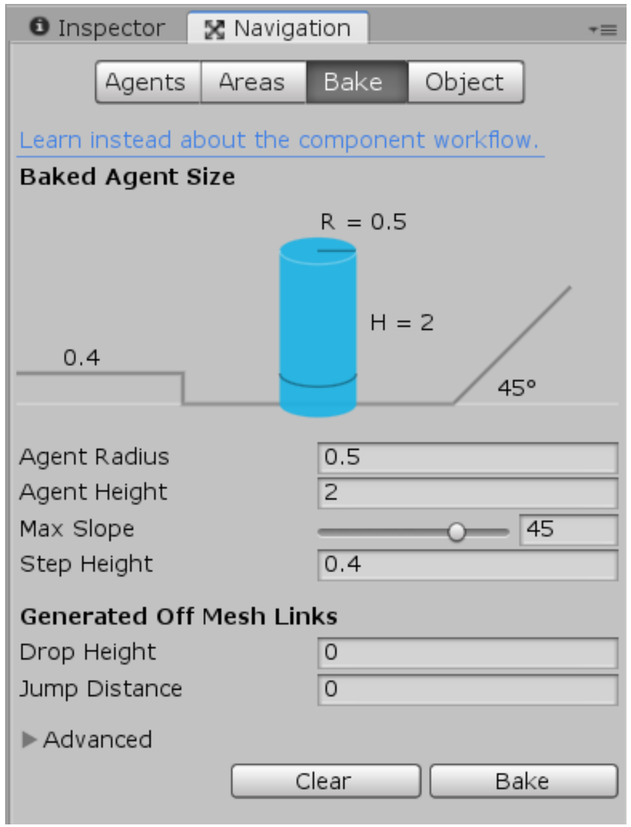
1. 開啟Navigation，位置：



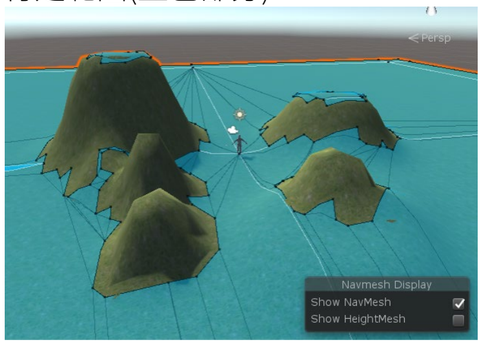
2. 預設出現在I欄中，在Object欄中選擇Terrain



3. 再到Bake，他可以控制人物走動地面的範圍角度影響人物爬坡的程度



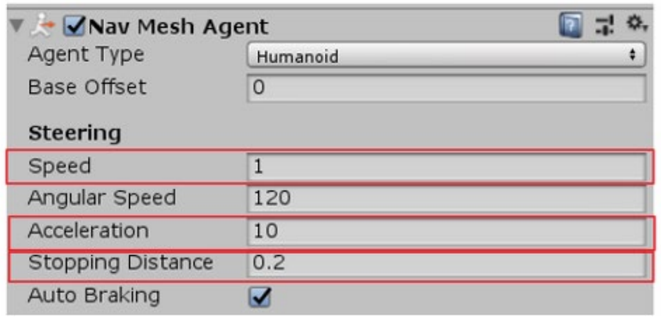
4. 決定好數值後按Bake即可出現可行走範圍(藍色部分)



【AIThirdPersonController 第三人稱視角AI】

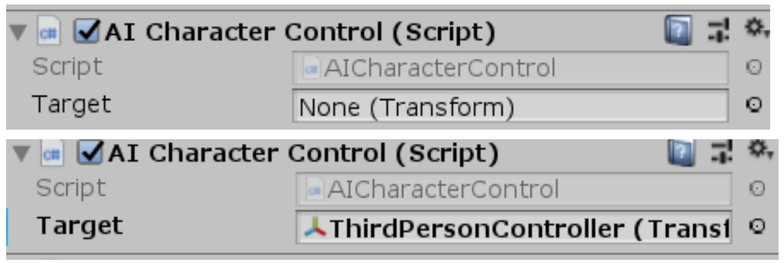
預設會追著設定的物件

◆ 基礎設定

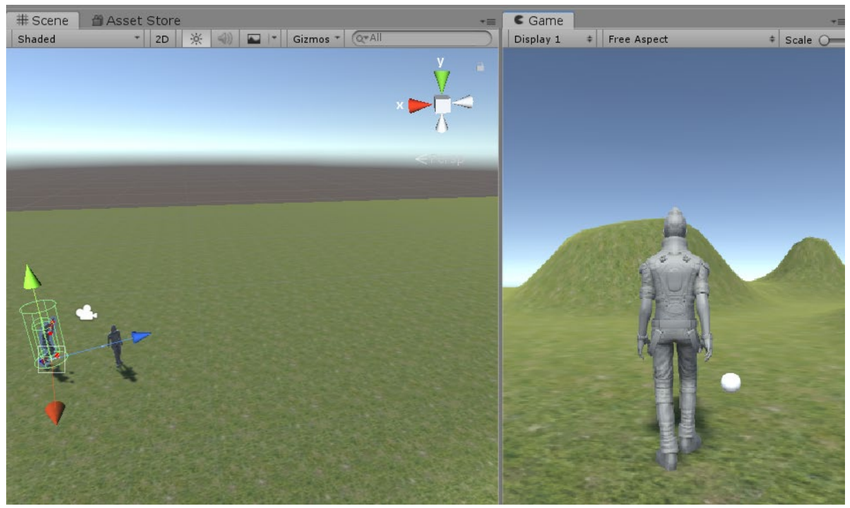


停止距離：當AI離人物一定距離時停下追趕的動作
設定停止距離的意義：
追到後做什麼，例如做砍擊的動作或是射擊的動作

◆ 設定追趕目標

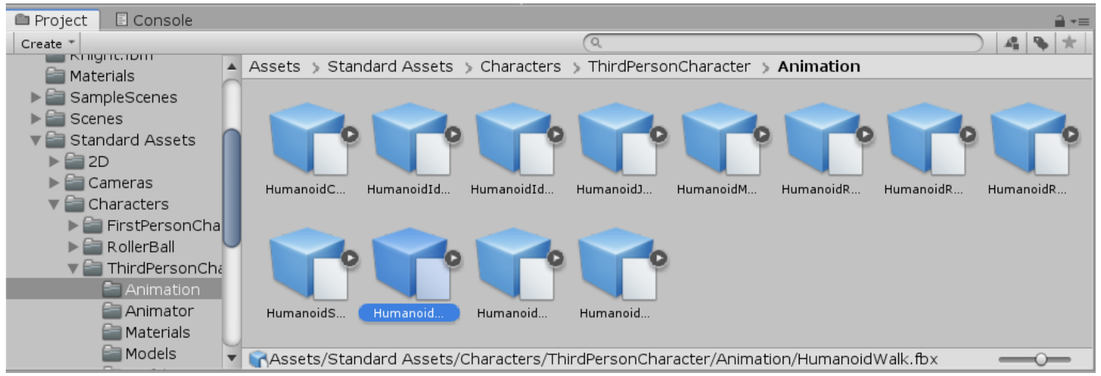


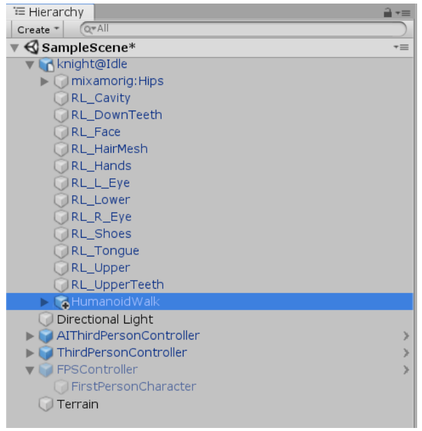
★若AI沒有做追趕動作，可能是沒有Bake好



【增加人物行走動畫】

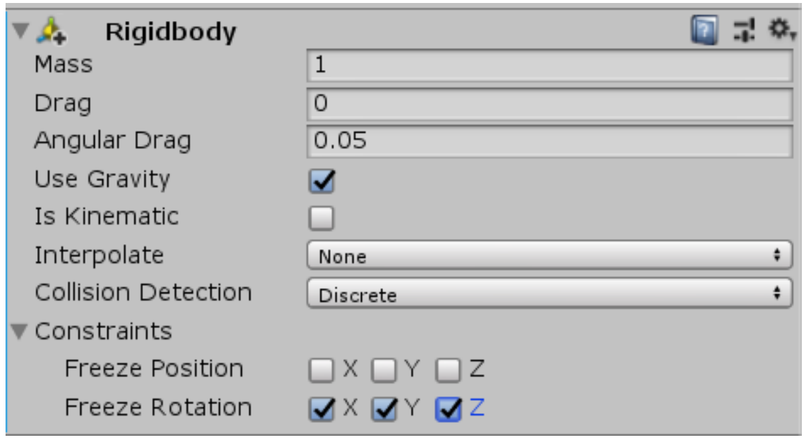
在ThirdPersonCharacter找到行走動畫，拉到騎士上



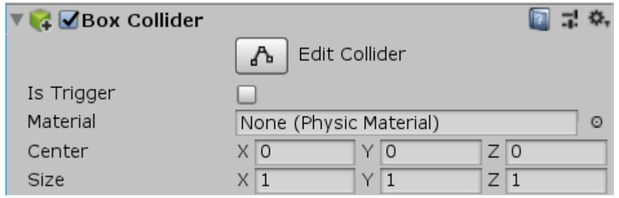


◆ **賦予物理特性**

增加條件：Rigibody重力(往下拉)
。勾選Freeze Rotation-→不會跌倒



Box Colider(盒子)：有很多種 Colider都可選。要調整盒子的Y到人物底部(適當即可)

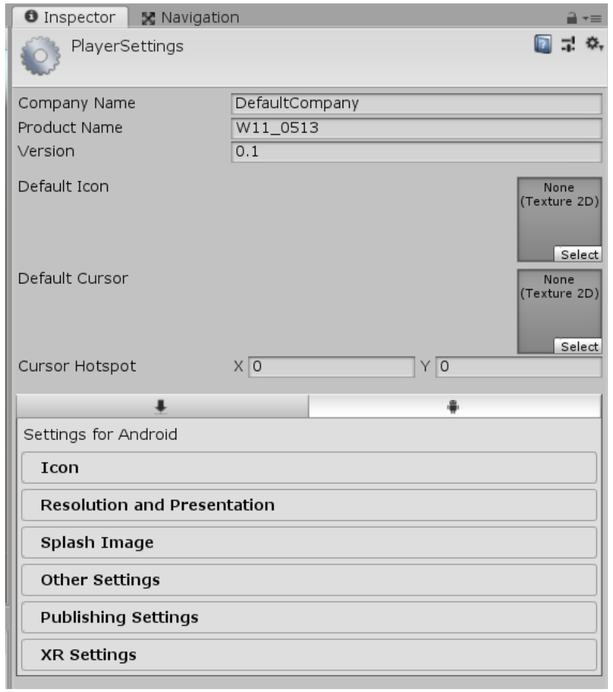


Week 11

【開啟HelloVR】

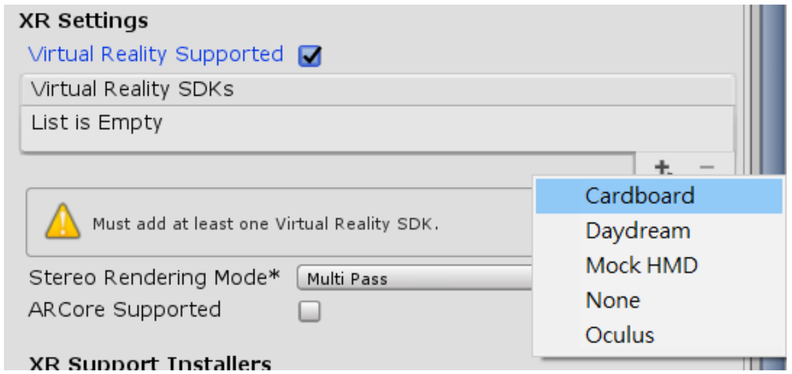
1. 開啟Build Setting轉換為Android平台
2. 點擊左下角的Player Setting 

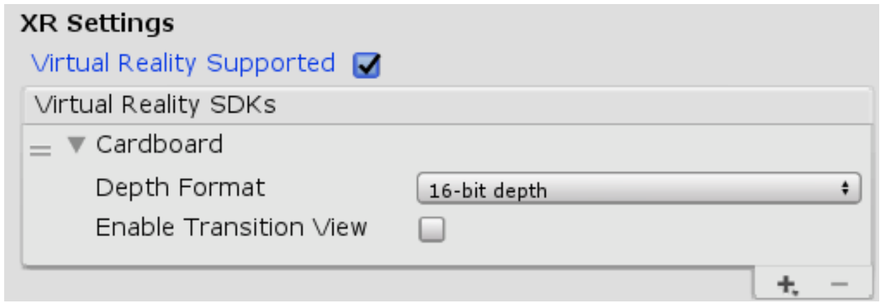
開啟此面板



3. 點開XR Setting，勾選Virtual Reality Supported

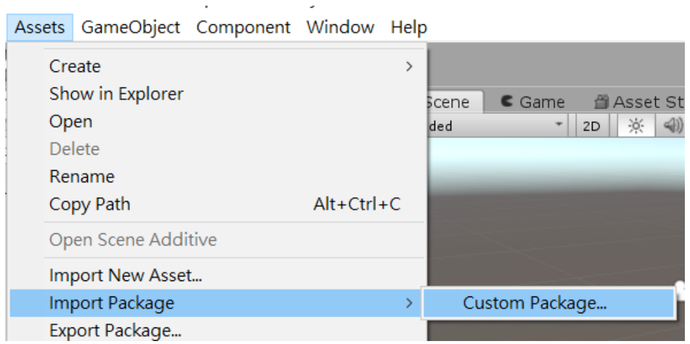
4. 勾選後在Virtual Reality SDKs的框中點擊+號，選擇Cardboard



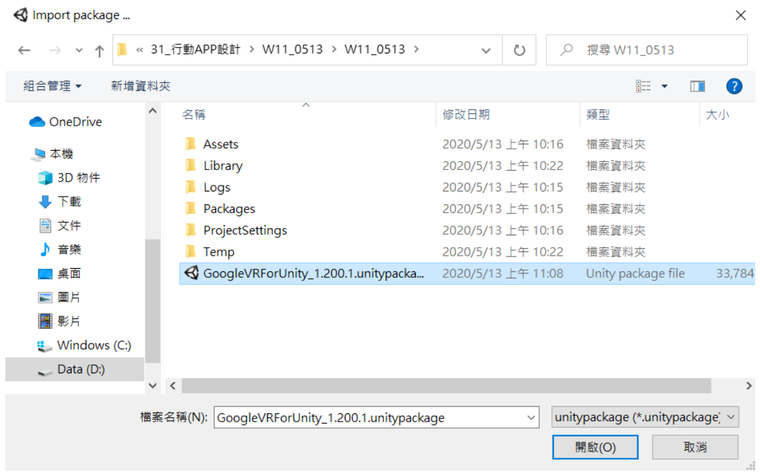


【開啟HelloVR】

1. 在Assets-->Import Package-->Custom Package

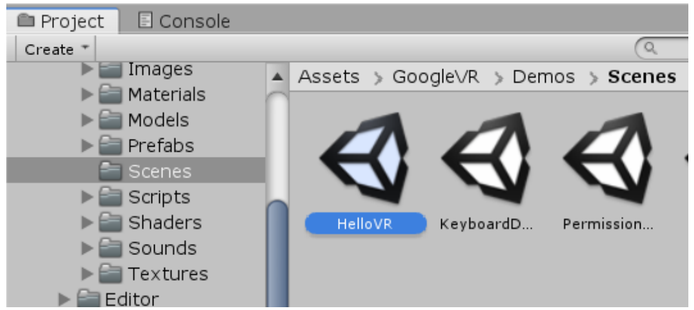


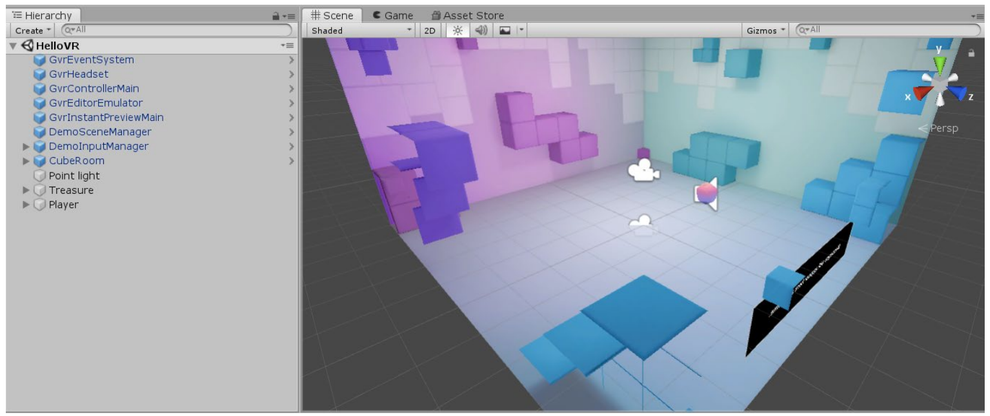
2. 選擇GoogleVRForUnity的檔案



*GoogleVRForUnity　可在課程雲端下載

Import後，在Project找到HelloVR，雙擊開啟範例完成

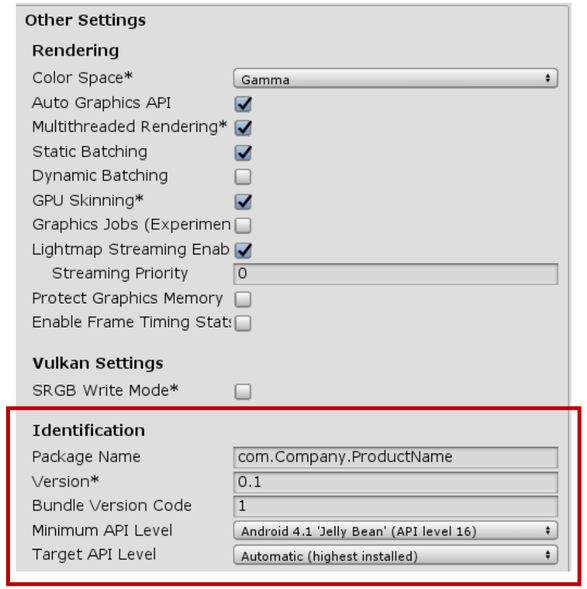




Week 11

【Player Setting設定修改】

點開Other Settings



找到Identification

1. Package Name是自己的APP的名稱，要修改，並有一套自己的命名規則，此名稱在發布APP後盡量不要修改，因為一旦修改將會重製所有東西，例如人氣等等
2. APILevel要19以上，越高會造成可執行的手機越少
3. Bundle Version Code：版本，初始值為1，當每次有修改並要進行覆蓋時都要加1才能進行覆蓋

【手機部分】

手機要開啟USB偵錯

USB設定要開啟傳輸檔案的選項

【Event Trigger】

Pointer Enter ：進入VR時

Pointer Exit：離開VR時

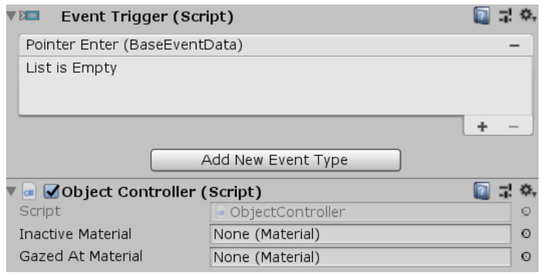
Pointer Click ：

點擊時。為手機上點擊而非VR空間中點擊

Week 11

【做到當看到物件時，物件會消失】

1. 隨意新增一物件後，增加條件：Event Trigger、Object Controler
2. Event Trigger底下再新增



3. 雙擊Object Controler開啟程式碼
4. 找到並複製public void TeleportRandomly(BaseEventData eventData

```
#endif  // !UNITY_EDITOR
        }

        /// <summary>Teleport this instance randomly when triggered by a point
        /// <param name="eventData">The pointer click event which triggered th
        public void TeleportRandomly(BaseEventData eventData)
        {
```

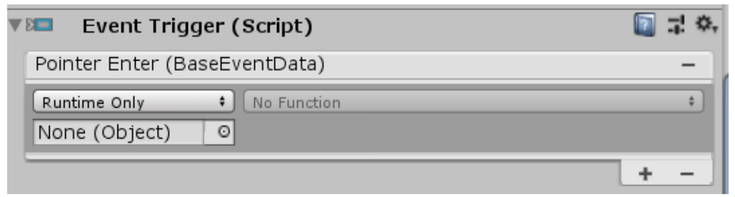
5. 在其上方貼上，修改 TeleportRandomly的部分為Attack

程式碼部分打上Destroy(this.gameObject);

```
    public void Attack(BaseEventData eventData)
    {
        Destroy(this.gameObject);
    }

    public void TeleportRandomly(BaseEventData eventData)
    {
```

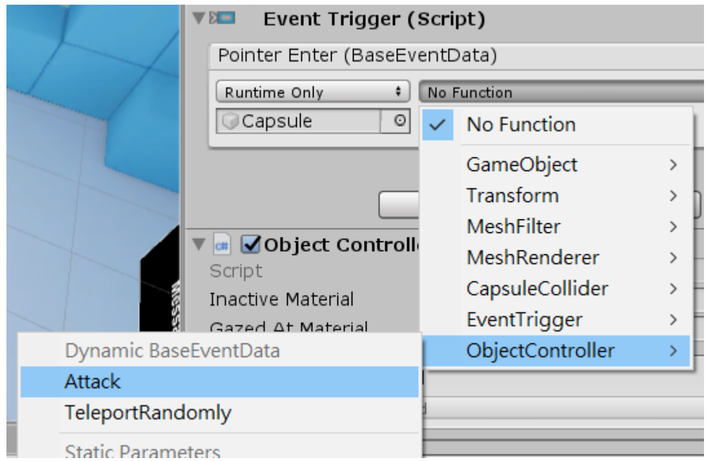
6. 回到Unity，點擊欲消失的那個物件，找到Event Trigger

點擊+號新增一Event Type



7. 將物件拖到None(Object)的部分

Function選擇ObjectControler裡的Attack



Week 11

【簡單做出360或720環景】

1. 先關掉CubeRoom和Treasure

2. 放入一顆Sphere，並將攝影機藏在裡面
3. 找一張360或720的環景圖匯入Unity
4. 直接將圖片拉到Sphere就能貼上去为

   (拉上去時會發現當畫面進入球體內時，就沒有貼圖了)
5. 所以要做以下動作：

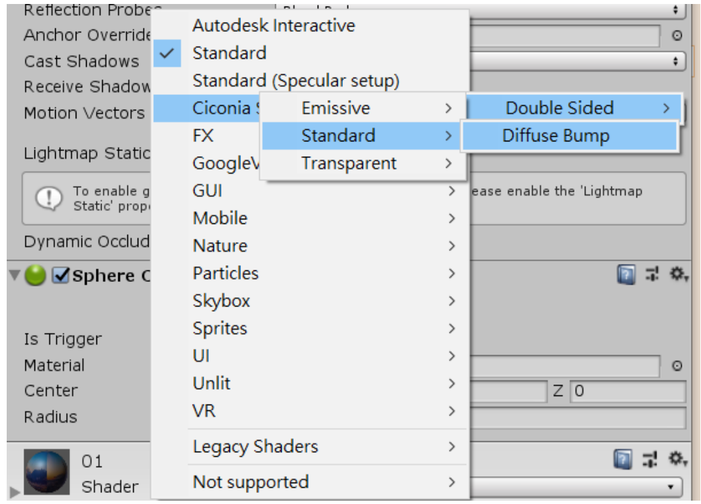
到Store下載Double SidedcShader後Import

CICONIA STUDIO

Double Sided Shaders

★★★★★ (239)

6. 在面板修改



7. 最後要長這樣

【發布到手機】

需先設定以下兩個部分

1. 增加VR場景

   打開Build Setting

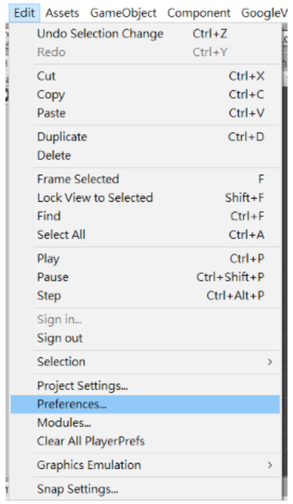
   點選Add Open Senes，並將空的頁刪除

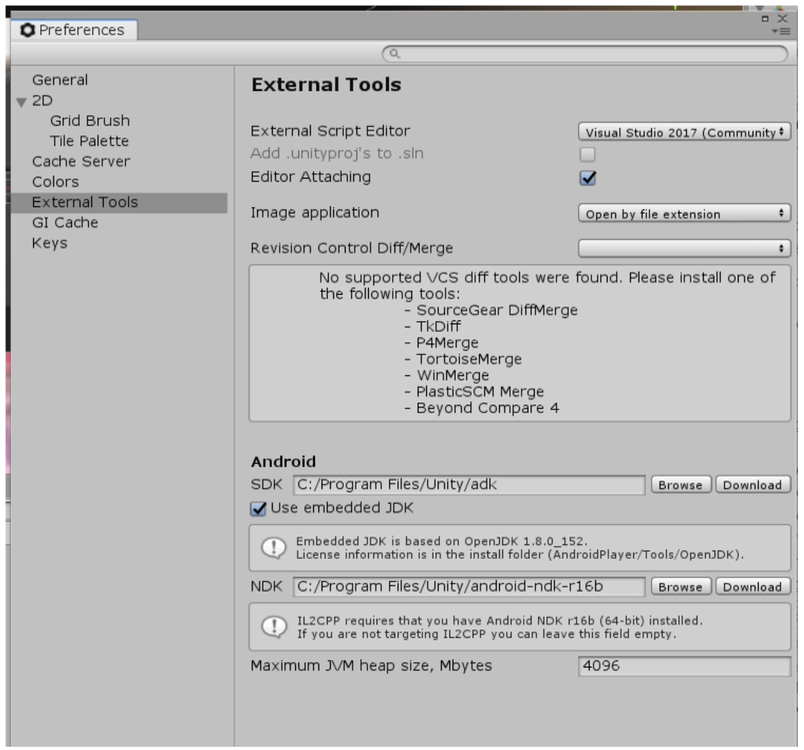
2. 設定External Tools

   在Edit-->Preference-->External Tools

   指定好Android大標中的SDK及NDK的位置

   *兩個檔案都能在課程雲端中下載





Week 12

【複製物件】

*注意不能一開始就關閉複製物的顯示

一、複製物件

新增空物件，並加上空Script，再開啟

將以下程式碼放在start

target = GameObject.Find(gameObject.name);

target.SetActive(false); //先將複製標的物停止

將以下程式碼放在update

GameObject newobj = Instantiate(target); //複製標的物

newobj.SetActive(true); //啟動標的物

宣告target：GameObjecy target;

[程式碼]

```
public class enemy : MonoBehaviour
{
    GameObject target;

    // Start is called before the first frame update
    void Start()
    {
        target = GameObject.Find(gameObject.name);

        target.SetActive(false); //先將複製標的物停止


    }

    // Update is called once per frame
    void Update()
    {
        GameObject newobj = Instantiate(target); //複製標的物

        newobj.SetActive(true); //啟動標的物
    }
}
```

*宣告的存在只在{}裡，所以宣告要放在上層

Find("欲複製的物件名字")

【複製物件】

二、控制物件以一定頻率生成

在 GameObject target; 底下宣告deley

int deley = 0;

在Update中加入

deley++;

if(deley % 20 == 0)

{

}

*%求餘數，數字越大頻率越低

並將以下程式碼放入{}

GameObject newobj = Instantiate(target); //複製標的物

newobj.SetActive(true); //啟動標的物

[程式碼]

```
public class enemy : MonoBehaviour
{
    GameObject target;
    int deley = 0;


    // Start is called before the first frame update
    void Start()
    {
        target = GameObject.Find("Foes");
        target.SetActive(false); //先將複製標的物停止



    }

    // Update is called once per frame
    void Update()
    {
        deley++;
        if(deley % 20 == 0)
        {
            GameObject newobj = Instantiate(target); //複製標的物
            newobj.SetActive(true); //啟動標的物

        }
```

[運行畫面]

【複製物件】

三、控制生成數量上限

int enemynum = 5;

控制物件生成頻率為隨機

if(deley % 20 == 0)-->

if(deley % Random.Range(10,30) == 0 && enemynum>0)

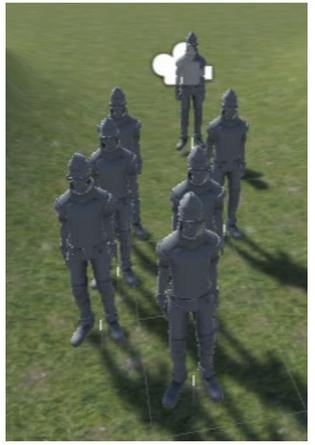
newobj.SetActive(true); //啟動標的物

enemynum--;

[程式碼]

```
public class enemy : MonoBehaviour
{
    GameObject target;
    int deley = 0;
    int enemynum = 5;

    // Start is called before the first frame update
    void Start()
    {
        target = GameObject.Find("Foes");
        target.SetActive(false); //先將複製標的物停止



    }

    // Update is called once per frame
    void Update()
    {
        deley++;
        if(deley % Random.R[完成圖],30) == 0 && enemynum>0)
        {
            GameObject newobj = Instantiate(target); //複製標的物
            newobj.SetActive(true); //啟動標的物
            enemynum--;
        }
```

[運行畫面]



Week 12

【複製物件】

四、設定物件生成範圍

Vector3 nv = new Vector3(Random.Range(5, 15), 0, Random.Range(10,20));

*範圍程式碼Vector3(Random.Range(x-,x+), 0, Random.Range(z-, z+));

[程式圖]

```
// Update is called once per frame
void Update()
{
    deley++;
    if(deley % Random.Range(10,30) == 0 && enemynum>0)
    {
        GameObject newobj = Instantiate(target); //複製標的物
        Vector3 nv=new Vector3(Random.Range(5, 15), 0, Random.Range(10, 20)); //給
        newobj.SetActive(true); //啟動標的物
        enemynum--;
    }
```

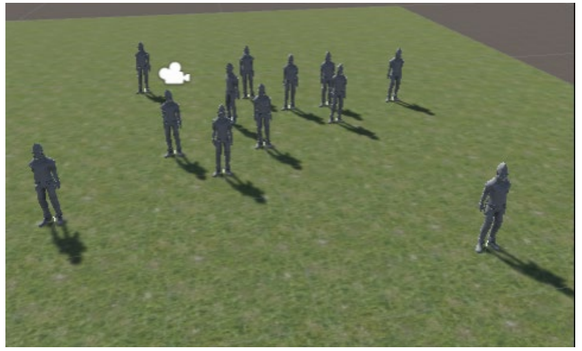
【更換物件座標】

newobj.transform.position = nv;

[程式圖]

```
void Update()
{
    deley++;
    if(deley % Random.Range(10,30) == 0 && enemynum>0)
    {
        GameObject newobj = Instantiate(target); //複製
        Vector3 nv=new Vector3(Random.Range(5, 15), 0,
        newobj.transform.position = nv; //儲存新位置

        newobj.SetActive(true); //啟動標的物
        enemynum--;
    }
```

[運行畫面]



【碰撞】

[Collision模式]

創建一個Cube

增加TouchMe的Script

[程式碼]

*關於碰撞

分為Collision和Trigger

兩者只能存在一種

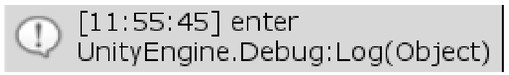
Collision：不可穿透

Trigger：可穿透

```
void OnCollisionEnter(Collision other)
  {
     Debug.Log("enter");
  }
  void OnCollisionStay(Collision other)
  {
     Debug.Log("stay");
  }
  void OnCollisionExit(Collision other)
  {
     Debug.Log("exit");
  }
```

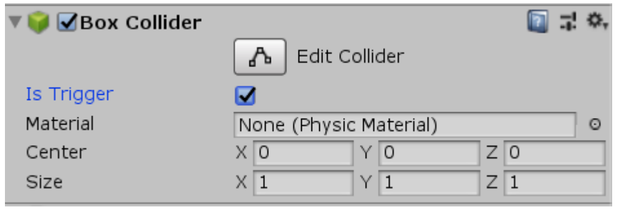
```
void OnCollisionEnter(Collision other)
{
    Debug.Log("enter");
}
void OnCollisionStay(Collision other)
{
    Debug.Log("stay");
}
void OnCollisionExit(Collision other)
{
    Debug.Log("exit");
}
```

成功會顯示



[Trigger模式]

要先將Box Collider裡的Is Trigger打勾



[程式碼]

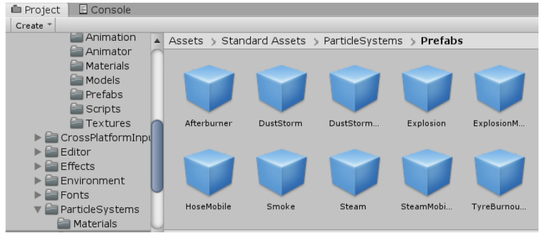
```
void OnTriggerEnter(Collider other)
  {
     Debug.Log("enter");
  }
  void OnTriggerStay(Collider other)
  {
     Debug.Log("stay");
  }
  void OnTriggerExit(Collider other)
  {
     Debug.Log("exit");
```

```
void OnTriggerEnter(Collider other)
{
    Debug.Log("enter");
}
void OnTriggerStay(Collider other)
{
    Debug.Log("stay");
}
void OnTriggerExit(Collider other)
{
    Debug.Log("exit");
}
```

Week 14

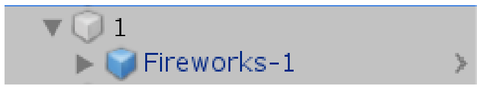
【粒子系統 (Particle System) 】

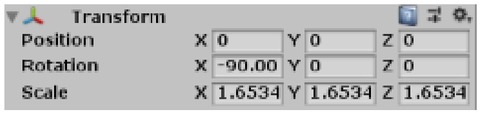
在Standard assets裡直接拖拉進去即可



【發射控制】

將煙火特效附屬於物件





將位置調成(0,0,0) **未調成000可能會導致下面的錯誤

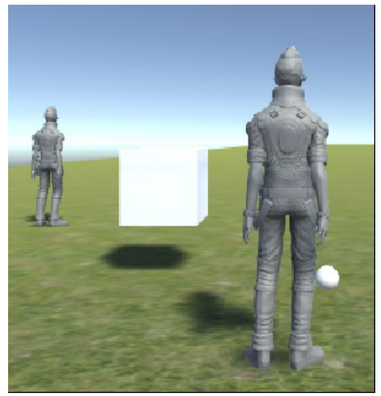
NullReferenceException: Object reference not set to an

```
public class TouchMe : MonoBehaviour
{
  GameObject p1;  //定義一個變數來儲存找到的煙火
}
void Start()
  {
    p1 = GameObject.Find("Fireworks");  //尋找煙火，並用p1儲存
    p1.SetActive(false);  //設定一開始時是關閉的
  }
```

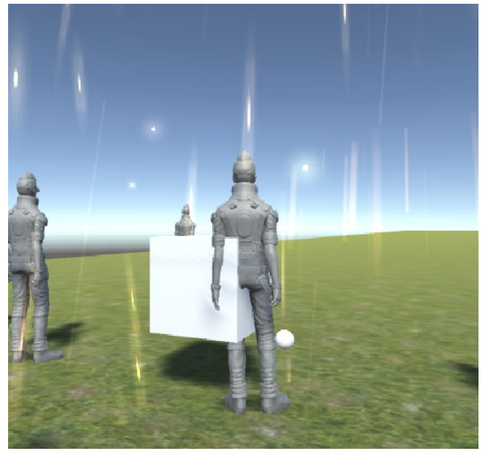
```
public class TouchMe : MonoBehaviour
{
    GameObject p1;
    // Start is called before the first frame update
    void Start()
    {
        p1 = GameObject.Find("Fireworks");
        p1.SetActive(false);
    }
```

[運行畫面]

程式開始時無動作



當人物碰到物體時發射煙火



Week 14

【發射控制多個】

當要在同一個程式碼裡控制多個物件時

可使用特效名稱+this.name來做到控制

```
void Start()
{
    p1 = GameObject.Find("Fireworks-"+this.name);
    p1.SetActive(false);
}
```

命名時 - 的含意

```
void Start()
{
    p1 = GameObject.Find("Fireworks"+this.name);
    p1.SetActive(false);
}
```

V.S

```
void Start()
{
    p1 = GameObject.Find("Fireworks-"+this.name);
    p1.SetActive(false);
}
```

比較上下兩個，-就是之的意思，1-1、1-2…和程式碼的+-無關

【讓立方體六面不同圖】

建立一個Cube，並增加一個Script，命名CustomUVS

[程式碼]

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

[ExecuteInEditMode]
public class CustomUVS : MonoBehaviour
{
    public Vector2 topPoint;
    public Vector2 bottomPoint;
    public Vector2 leftPoint;
    public Vector2 rightPoint;
    public Vector2 frontPoint;
    public Vector2 backPoint;

    private Mesh m_mesh;
    public enum CubeFaceType
    {
        Top,
        Bottom,
        Left,
        Right,
        Front,
        Back
    };
```

```
 // Use this for initialization
    void Start()
    {
        MeshFilter meshFilter = GetComponent<MeshFilter>();
        if (meshFilter == null)
        {
            Debug.LogError("Script needs MeshFilter component");
            return;
        }

#if UNITY_EDITOR
        Mesh meshCopy = Mesh.Instantiate(meshFilter.sharedMesh) as Mesh;    // Make a
deep copy
        meshCopy.name = "Cube";
        m_mesh = meshFilter.mesh = meshCopy;                            // Assign the copy to the
meshes
#else
        m_mesh = meshFilter.mesh;
#endif
        if (m_mesh == null || m_mesh.uv.Length != 24)
        {
            Debug.LogError("Script needs to be attached to built-in cube");
            return;
        }

        UpdateMeshUVS();
    }

    // Update is called once per frame
    void Update()
    {
#if UNITY_EDITOR
        UpdateMeshUVS();
#endif
    }
```

```
void UpdateMeshUVS()
    {
        Vector2[] uvs = m_mesh.uv;
        // Front
        SetFaceTexture(CubeFaceType.Front, uvs);
        // Top
        SetFaceTexture(CubeFaceType.Top, uvs);
        // Back
        SetFaceTexture(CubeFaceType.Back, uvs);
        // Bottom
        SetFaceTexture(CubeFaceType.Bottom, uvs);
        // Left
        SetFaceTexture(CubeFaceType.Left, uvs);
        // Right
        SetFaceTexture(CubeFaceType.Right, uvs);
        m_mesh.uv = uvs;
    }

    Vector2[] GetUVS(float originX, float originY)
    {
        Vector2[] uvs = new Vector2[4];
        uvs[0] = new Vector2(originX / 3.0f, originY / 3.0f);
        uvs[1] = new Vector2((originX + 1) / 3.0f, originY / 3.0f);
        uvs[2] = new Vector2(originX / 3.0f, (originY + 1) / 3.0f);
        uvs[3] = new Vector2((originX + 1) / 3.0f, (originY + 1) / 3.0f);
        return uvs;
    }
```

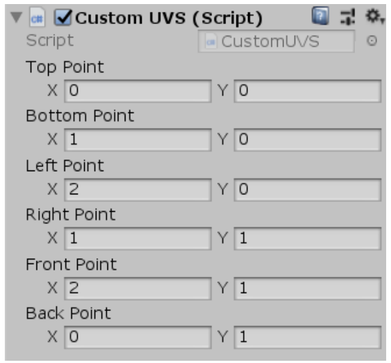
[程式碼]

```
void SetFaceTexture(CubeFaceType faceType, Vector2[] uvs)
  {
     if (faceType == CubeFaceType.Front)
     {
        Vector2[] newUVS = GetUVS(frontPoint.x, frontPoint.y);
        uvs[0] = newUVS[0];
        uvs[1] = newUVS[1];
        uvs[2] = newUVS[2];
        uvs[3] = newUVS[3];
     }
     else if (faceType == CubeFaceType.Back)
     {
        Vector2[] newUVS = GetUVS(backPoint.x, backPoint.y);
        uvs[10] = newUVS[0];
        uvs[11] = newUVS[1];
        uvs[6] = newUVS[2];
        uvs[7] = newUVS[3];
     }
     else if (faceType == CubeFaceType.Top)
     {
        Vector2[] newUVS = GetUVS(topPoint.x, topPoint.y);
        uvs[8] = newUVS[0];
        uvs[9] = newUVS[1];
        uvs[4] = newUVS[2];
        uvs[5] = newUVS[3];
     }
```

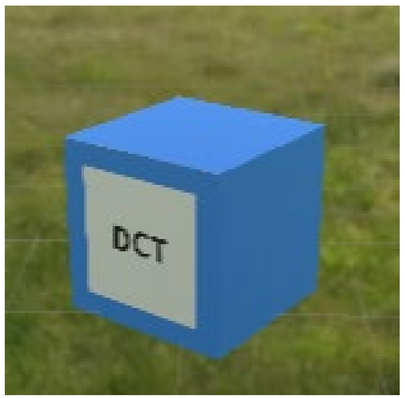
[程式碼]

```
     else if (faceType == CubeFaceType.Bottom)
     {
        Vector2[] newUVS = GetUVS(bottomPoint.x, bottomPoint.y);
        uvs[12] = newUVS[0];
        uvs[14] = newUVS[1];
        uvs[15] = newUVS[2];
        uvs[13] = newUVS[3];
     }
else if (faceType == CubeFaceType.Left)
     {
        Vector2[] newUVS = GetUVS(leftPoint.x, leftPoint.y);
        uvs[16] = newUVS[0];
        uvs[18] = newUVS[1];
        uvs[19] = newUVS[2];
        uvs[17] = newUVS[3];
     }
     else if (faceType == CubeFaceType.Right)
     {
        Vector2[] newUVS = GetUVS(rightPoint.x, rightPoint.y);
        uvs[20] = newUVS[0];
        uvs[22] = newUVS[1];
        uvs[23] = newUVS[2];
        uvs[21] = newUVS[3];
     }
   }
}
```

貼完儲存後回到Unity，在Cube的Inspector介面中應該會多出



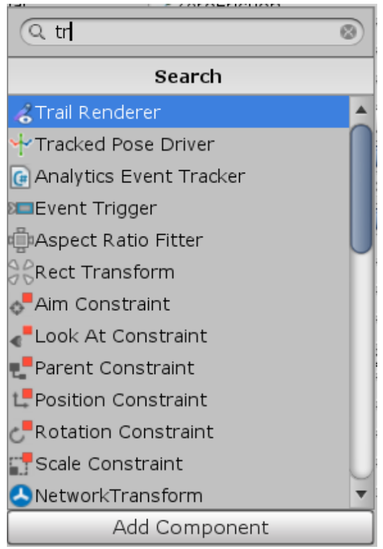
修改數字就是更換面的座標點，依據需求可自由換面



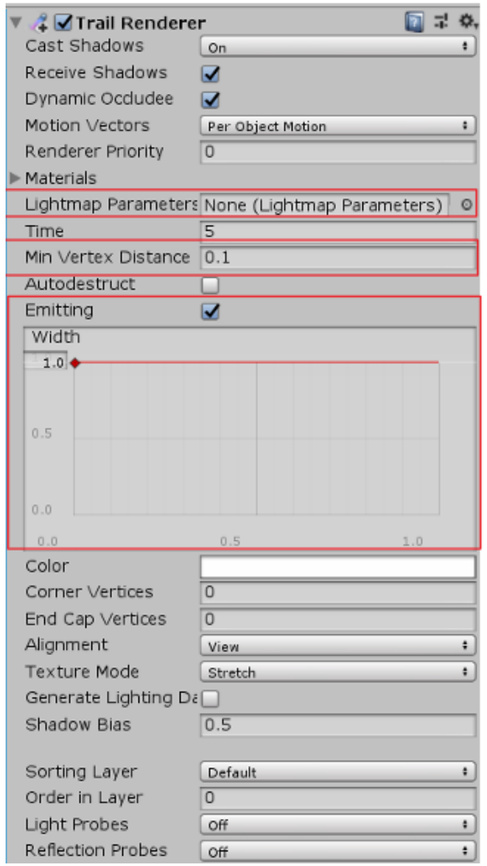
Week 14

【Trail Renderer 光跡】

Add Component



功能列表



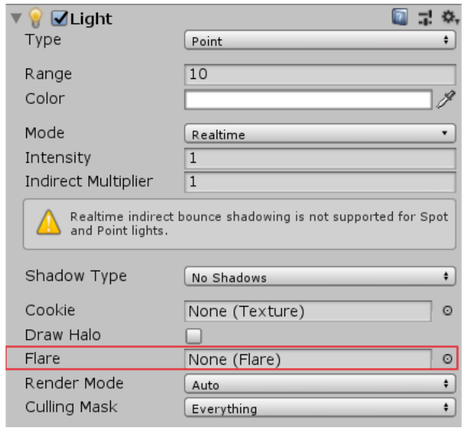
貼圖

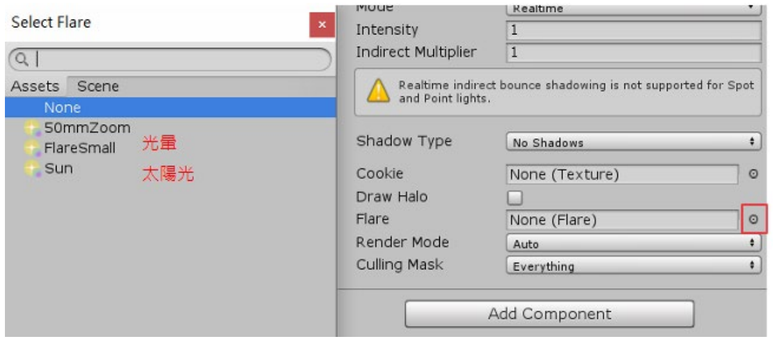
延遲時間

變化曲線

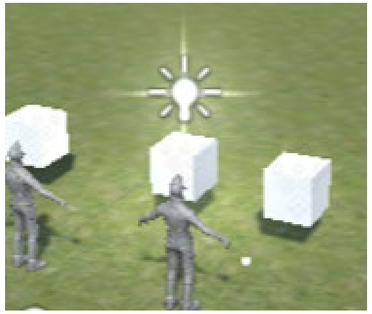
【FlareSmall光暈】

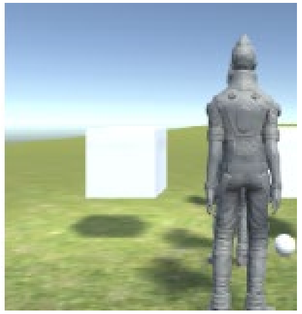
Point Light → Flare → FlareSmall光暈





[運行畫面]





*若在Play後光暈沒有出現，那要在Main Camera加上Flare Layer攝影機辨識到光暈

【UI_程式碼控制Text文字】

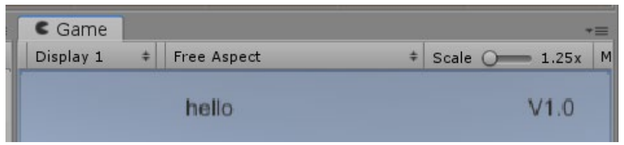
建立分數與版本

在玩家的物件上加上Script：UserUI

[程式碼]

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //4. 因為Text有3D和UI兩種，所以要增加這個讓系統辨識到UI的Text

public class UserUI : MonoBehaviour
{
    GameObject uui; //1. 首先宣告一個變數uui
    Text[] ts; //3. 增加一個儲存文字的陣列

    // Start is called before the first frame update
    void Start()
    {
        uui = GameObject.Find("Canvas"); //2. 先找到Canvas
        ts = uui.GetComponentsInChildren<Text>();
        //5. 取得Component中的Text(因為有很多Component所以要加s)並讓ts儲存，且整體為函數
        ts[0].text = "hello"; //6. 測試抓取Text並讓框中文字變成hello
    }
```

[運行畫面]



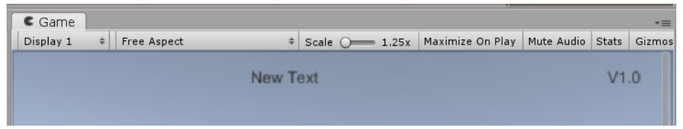
【UI_原點調整】

紅框處為物件的「原點」

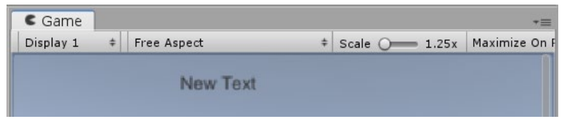
用於當手機解析度大小不同時，控制物件對於原點的相對位置

----------例如----------

當手機螢幕細長時，V1.0會存在

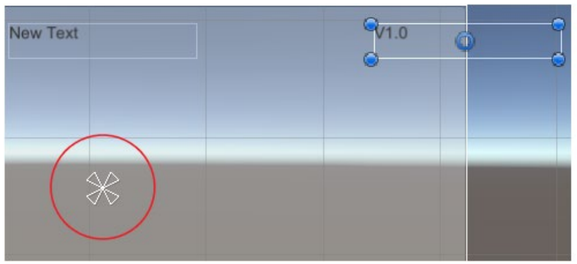


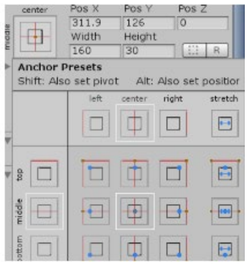
但當手機螢幕較短時，V1.0則消失



這是因為原點在螢幕正中央，因此要調整原點的位置，讓V1.0可以隨

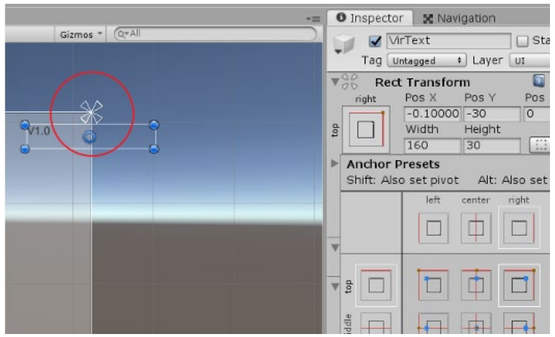
螢幕解析度變化而改變位置



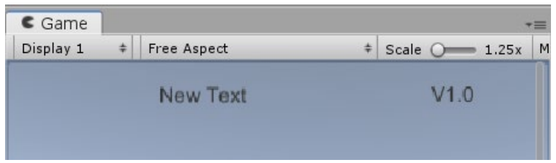


【UI_原點調整】

將原點改成右上角



就會跟著解析度改變位置了



【UI_改變Text其中的數字】

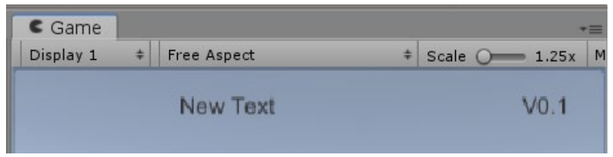
續用上個程式碼

將 ts[0].text = "hello";

改成

```
ts[1].text = "V" + Application.version;
//6. 測試抓取Text並讓框中文字變成版本數
```

[運行畫面]



【UI_分數改變】

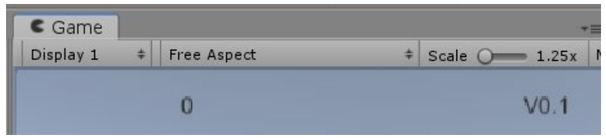
在Text[] ts;底下加上

```
int score = 0; //7. 定義score
```

在Update部份加上

```
ts[0].text = score.ToString(); //8. 抓取分數框並將文字轉為字串
```

[運行畫面]



【UI_無法存取Score】

每個Component如同一個國家，會將自己內部的程式碼們保護起來。因此當TouchMe想存取UserUI中的Score時，會因為保護機制而無法取得。

若想讓Score能被存取，只需要在 int Score前面加上public就可以了→public int Score。

但這並非最佳解，只是能最簡單解決問題而已

最佳解應是加一個Function，因為是Function所以可以加些程式邏輯，例如防破解→分數需小於5才是正常遊玩，

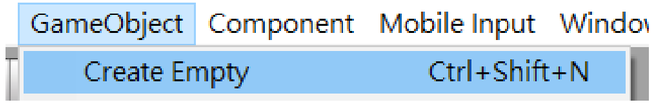
並將原本的Component uui; 改成 UserUI uui;

```
UserUI uui; //找到元件uui
```

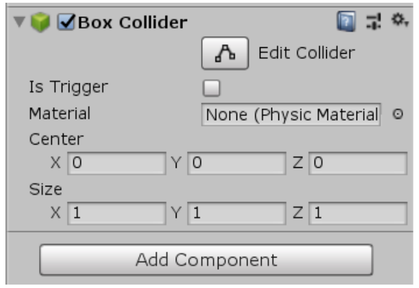
Week 15

【空氣牆製作】

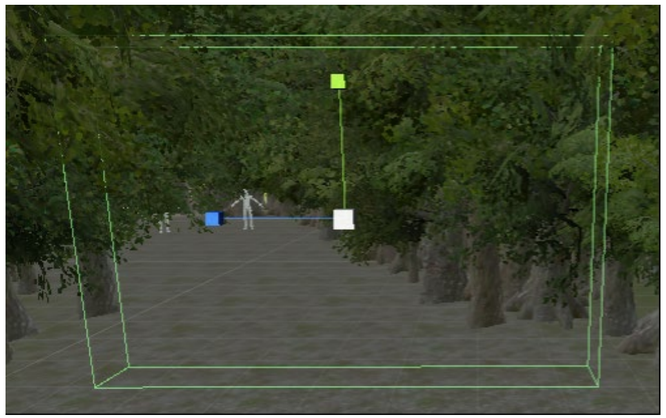
建立空物件



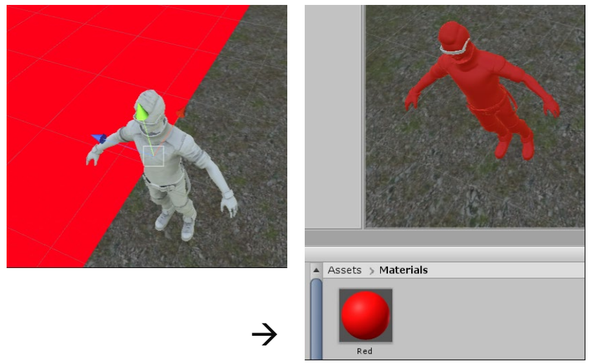
增加Box Collider的Component



調整大小、方向與距離

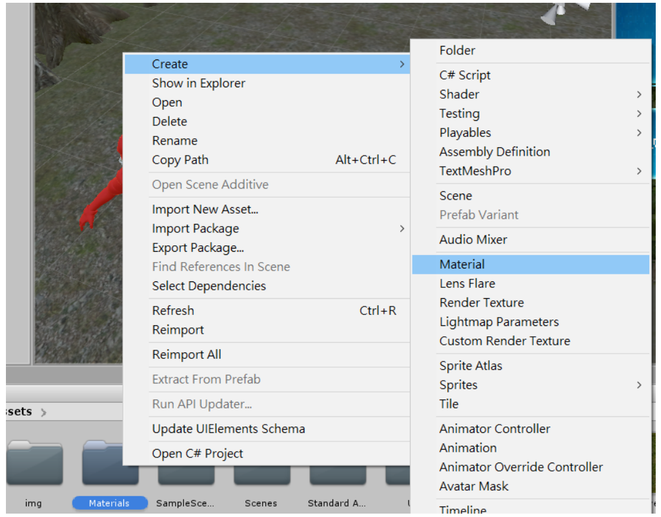


【為Standard Assets的素模上色】



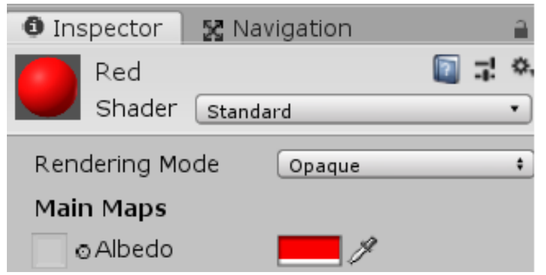
----------步驟----------

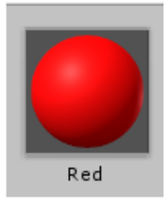
1. 建立一個Material的資料夾
2. 在資料夾上右鍵 → Create → Materia



3. 接著會出現一空白的材質球，並命名

4. 點擊材質球，在Inspector修改成其顏色





5. 直接拖拉到模型上即可上色

【常數chr】

在GameObject p1;下新增一常數chr：GameObject chr;

```
GameObject p1;
GameObject chr; //定義一常數chr
```

讓chr去找到RigiBodyFPSController

```
void Start()
{
    p1 = GameObject.Find("Fireworks-"+this.name);  //找到Fireworks-"+this.name
    p1.SetActive(false); //設定p1一開始為停止

    chr = GameObject.Find("RigiBodyFPSController");  //找到RigiBodyFPSController
}
```

找到uui並讓uui為chr去得到元件從UserUI

```
GameObject p1;
GameObject chr; //定義一常數chr
Component uui; //找到元件uui
```

```
chr = GameObject.Find("RigiBodyFPSController");  //找到RigiBodyFPSController
uui = chr.GetComponent<UserUI>();  //讓uui為chr去得到元件從UserUI
```

當箱子被碰到時，則觸發加分

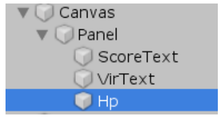
```
void OnTriggerEnter(Collider other)
{
    Debug.Log("enter");
    p1.SetActive(true);
    uui.score++;
}
```

Week 16

【UI生命值】

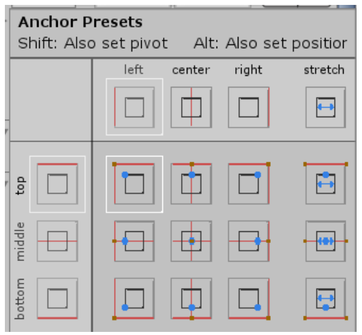
選一張圖片放入Unity，轉成Sprite

在Panel右鍵新增一image命名為Hp

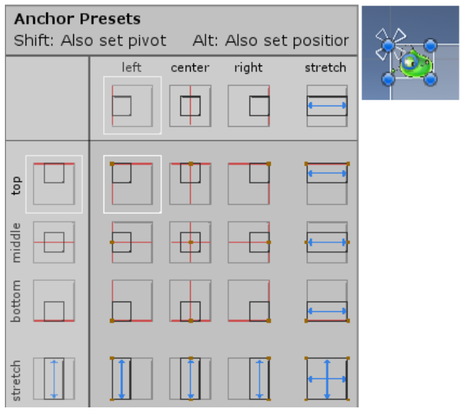


在image的Inspector指定Hp的圖片

將他的基準點改為左上角



按住Alt後點擊左上角可以將圖片瞬移到左上角



【生命值程式碼】

1. 在GameObject uui;底下加上GameObject hpi;與GameObject[] hps;

```
GameObject uui; //1. 首先宣告一個變數uui
GameObject hpi;
GameObject[] hps;
```

2. 在int score = 0;底下加上int life = 3;與int maxlife = 10;

```
Text[] ts; //3. 增加一個儲存文字的陣列
int score = 0; //7. 定義score
int life = 3;  //0617_2.定義初始生命數量
int maxlife = 10;  //0617_2. 定義最大生命數量
```

3. 在ts[1].text = "V" + Application.version;加上hps、hpi…

```
ts[1].text = "V" + Application.version; //6. 測試抓取Text並讓框中文字變成版本數
hps = new GameObject[maxlife]; //0617_3. 創造新物件_陣列maxlife
hpi = GameObject.Find("Hp");  //0617_3. 找到Hp
hpi.SetActive(false); //0617_3. 設定一開始hpi為無動作(false)
```

*[maxlife]最大生命值，要在上面先定義maxlife這個變數，並賦予他值：int maxlife=10;

當要分辨常數與變數時，先問自己這個數字或者詞語是否含有意義，是否會改變，是則變數，否則常數(不變)

Week 16

【生命值程式碼】

在hpi.SetActive(false); 底下加入以下程式碼

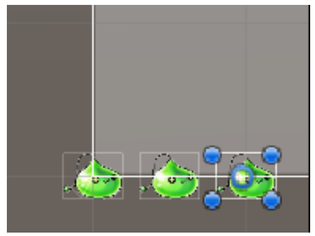
```
for (int i = 0; i < maxlife; i++) //迴圈i從0開始，到maxlife-1停止
{
    hps[i] = Instantiate(hpi); //Instantiate意為要複製的物件
    hps[i].name = "Hp" + (i + 1).ToString(); //定義複製品的命名規則
    hps[i].transform.SetParent(uui.transform); //讓複製品在複製物的上層
    if (i < life) hps[i].SetActive(true); //當i小於life時，讓複製行為持續
    Vector3 nv = new Vector3(i * 50, 313, 0); //定義複製品的位置座標
    hps[i].transform.position = nv; //讓複製品到指定座標
}//0617.End
```

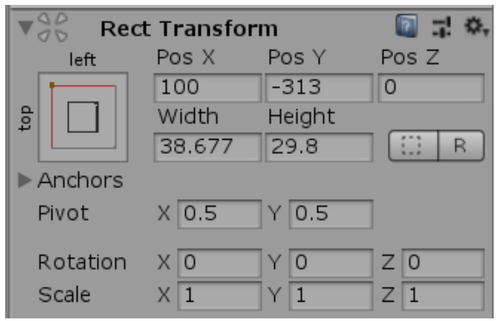
1. "Hp" + (i + 1).ToString();：因為前面是字串後面是數字，沒辦法接在一起，所以要將數字轉成字串
2. "Hp" + (i + 1)，為何要i+1，而非i？

   若為i則Hp的數字會從0起算，那+1則較符合從1起算的習慣
3. (i * 50, 363, 0);，為何是50及313？

   50為間隔，開始為(0,0,0)，接下來就是(50,0,0)….

   313則為Unity的Bug，會導致運行時生命值的Y變成-313導致沒出現在畫面中，因此此處用較笨的方法解決





Week 16

【生命值程式碼】

```
public void SetScore(int s) //0617_定義分數計算公式函數
{
    if (s < 5) score = score + s; //若s小於5，分數加s
    ts[0].text = score.ToString(); \
    //8. 抓取分數框並將文字轉為字串
    //0617_從Update移到此處，因為分數不須一直更新
}
```

*//0617_從Update移到此處，因為分數不須一直更新

分數變動原在Update中，讓分數會不斷做更新的動作

而Funtion函數如同守衛，程式在需要動到函數中變數時

都需重跑一次函數中的程式，因此讓分數變動的部分從

Update中移出，改進函數中，降低分數更新的次數，也就

是當函數被呼叫(使用到分數)時才更新。

```
public void SetLife(int s) //0617_//定義生命計算公式函數
{
    life += s; //生命值以加s增加
    if (life <= 0) //若生命值<=0
    {
        life = 0; //生命值為0
        //game over
        gameover.SetActive(true); //使GameOver為真(執行)
    }
    for (int i = 0; i < maxlife; i++) //0617
    {
        if (i < life) hps[i].SetActive(true); else hps[i].SetActive(false);
        //若i小於生命值，則讓其顯示，否則不顯示
    }
}
```

*Life = life + s 和 life += s 為同，只是後者為前者簡化

```
    gameover.SetActive(true); //使GameOver為真(執行)
}
for (int i = 0; i < maxlife; i++) //0617
{
    if (i < life) hps[i].SetActive(true); else hps[i].SetActive(false);
    //若i小於生命值，則讓其顯示，否則不顯示
}
```

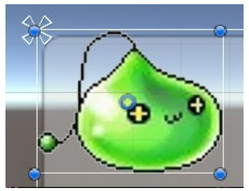
* if (i < life) hps[i].SetActive(true); else hps[i].SetActive(false);

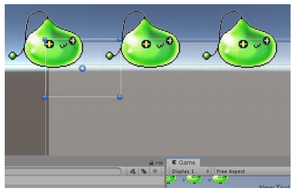
顯示時會對應life的數量：假設life(生命值)為3，則程式運行時會唱名1號、2號…，這些數字小於3所以會「被顯示」，而唱名到4號時，因為大於3了，所以不會「被顯示」。

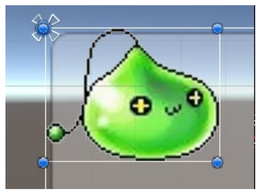
【生命值位置調整】

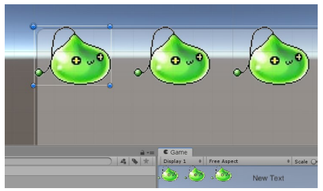
另外也要將藍色圈圈(圖片原點)移到左上角位置，生成時

圖片才會全都在畫面裡









Week 16

【敵人碰到自己時扣血】

在Foes建立一Script，命名EnermyAI

```
public class EnermyAI : MonoBehaviour
{
    GameObject chr;
    UserUI uui;

    // Start is called before the first frame update
    void Start()
    {
        chr = GameObject.Find("RigidBodyFPSController");
        uui = chr.GetComponent<UserUI>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Vector3.Distance(chr.transform.position, transform.position) < 1.0f) uui.SetLife(-1);
    }

}
```

* if (Vector3.Distance(chr.transform.position, transform.position) < 1.0f) uui.SetLife(-1);

[解析]

if (… < 1.0f)	當什麼小於1這個單位
uui.SetLife(-1);	執行生命-1
Vector3.Distance	取得位置
(chr.transform.position,transform.position)	玩家的位置和該Component所在的位置(Foes的位置) 當 transform.position前方無指定時，自動認定為該Component所在的位置