

1091_手機遊戲程式設計(B4CS030049A)

期末報告

主題：Volley—凌空抽射(單&多機板)

簡介：

- 「皮卡丘打排球」是一款大家熟悉的電腦遊戲之一，而這款遊戲是於 1997 年時由日本開發的 Windows 遊戲。
- 也於不同平台復刻，如：以 JavaScript 實作並於瀏覽器上遊玩
 - <https://gorisanson.github.io/pikachu-volleyball/zh/>
- 隨著智慧型手機的普及，手機遊戲(手遊)市場也逐漸蓬勃發展
- 本次課堂遊戲專題目標，為將皮卡丘打排球這款遊戲利用 **Unity 遊戲引擎**實作，並且設計為**手機端**也能遊玩的單人遊戲(日後能設計為雙人連線遊戲)

而本次的課堂專題目標，為將皮丘打排球這款遊戲利用 Unity 遊戲引擎實作，並且設計為手機也能遊玩的遊戲，並且本次實作出柳種不同版本，分別為

1. 單人手機互動版 2. 多人電腦連線版

(一)單人手機互動版

1. 遊戲規則&角色操控方法&關卡

在單機版遊戲中，本遊戲總共有四種畫面，分別是遊戲開始、開始遊玩、左方玩家勝利及右方玩家勝利的畫面，遊戲開始的畫面可以對應到圖 1，開始遊玩的畫面可以對應到圖 2，左方玩家勝利的畫面可以對應到圖 3，右方玩家勝利的畫面可以對應到圖 4。



圖 1、遊戲開始畫面

本遊戲的遊戲規則為當其中一方得到 25 分時，遊戲則結束(屆時螢幕會顯示圖 x 或是圖 x 的畫面)，得分方法為當球掉到其中一方的地板時，則另一位玩家會獲得分數，角色操控方式為利用圖 x 螢幕上的紅框部份操控角色的跳以及左右的移動。

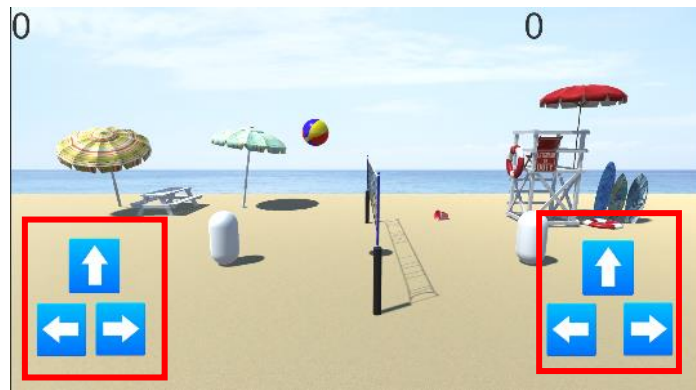


圖 2、遊戲遊玩畫面



圖 3、左方玩家獲勝畫面



圖 4、右方玩家獲勝畫面

2. 程式碼&遊戲物件

本遊戲在單機版遊戲的部份中總共需要三種程式碼的支撐，分別是 move_ball.cs、move.cs 及 move2.cs 三種程式碼，其中 move_ball.cs 的程式碼內包含玩家得分的統計及球移動方向、力道及球的移動軌跡，玩家得分的程式碼可以看到圖 5 的函式，在圖 x 中本遊戲是利用”tag”的方式判斷球落在哪一個玩家的地面上，可以看到圖 x 中本遊戲有設有兩種 tag，分別是 leftground 跟 rightground。因此當球撞到 leftground 時，表示的是右方玩家得分，所以球的重置點必須要在右方玩家的頭頂上，並將球固定讓玩家可以將球撞出去，反之當球撞到 rightground 時，則與上述敘述相反。

而在球的移動方式、力道與球的移動軌跡的判斷方式上，本遊戲是利用 Rigidbody 抓取玩家在移動角色時的方向與力道，並當玩家的角色碰到球時，玩家角色的速度與方向就會轉移至球上，球就會依據玩家的這些資訊去進行下一步的移動。

```
void OnCollisionEnter(Collision other)
{
    if (other.gameObject.tag == "leftground")
    {
        gameObject.transform.position = right_target_point.transform.position;
        rd.constraints = RigidbodyConstraints.FreezeAll;
        scoreRight++;
        right.text = scoreRight.ToString();
    }
    if (other.gameObject.tag == "rightground")
    {
        //move = new Vector3(move.x, move.y, move.z +0.1f );
        gameObject.transform.position = left_target_point.transform.position;
        rd.constraints = RigidbodyConstraints.FreezeAll;
        scoreLeft++;
        left.text = scoreLeft.ToString();
    }
}

if (scoreRight >= 25)
{
    Panelrightwin.SetActive(true);
}
if (scoreLeft >= 25)
{
    Panellleftwin.SetActive(true);
}
```

圖 5、move_ball.cs 內的 OnCollisionEnter 的函式

另外 move.cs 及 move2.cs 的程式碼功能是為了控制左右兩隻角色而設置的。

而在遊戲配樂上，本遊戲採用的背景音樂是 youtube 的免費音樂，球的碰撞也是採用免費的音效，至於如何讓球碰撞產生聲音的程式碼是在分別在 move.cs 及 move2.cs 的程式碼中，用到的概念是 AudioClip 的物件，首先像圖 6 一樣在程式中宣告一個 AudioClip 的物件，方便使用者指定需要的音樂素材，再來是透過碰撞的函式將碰撞的音效加入其中，可以對應到圖 7 的紅色框框，判斷音效發出的標準如下，當角色與球進行碰撞之後，就會在碰撞的地點中發出碰撞聲。

```
public AudioClip ball_collision;
```

圖 6、宣告 AudioClip 的物件

```
private void OnCollisionEnter(Collision collision)
{
    if(collision.gameObject.tag == "ball")
    {
        rd.constraints = RigidbodyConstraints.None;
        rd.constraints = RigidbodyConstraints.FreezePositionX;
        Rigidbody player = gameObject.GetComponent<Rigidbody>();
        Vector3 move = new Vector3(Mathf.Abs(player.velocity.x), Mathf.Abs(player.velocity.y), Mathf.Abs(player.velocity.z));
        if (MoveType.HasFlag(MoveType.Left))
        {
            rd.AddForce(left_Ball_Force* ratio);
        }
        else if (MoveType.HasFlag(MoveType.Right))
        {
            rd.AddForce(right_Ball_Force* ratio);
        }
        //rd.AddForce(move * 150);
        Debug.Log(move);
    }
    if (ball_collision)
    {
        AudioSource.PlayClipAtPoint(ball_collision, collision.transform.position);
    }
}
```

圖 7、判斷碰撞音效發出的時機點

(二)多人電腦連線版

1. 遊戲規則&角色操控方法&關卡

多人電腦連線版本中，遊戲規則與單機版相同。而相異之處為，多人電腦連線版本使用 Joystick 虛擬搖桿控制角色左右移動，並且設計虛擬按鈕(右方)，其功能為跳躍(圖 8)。玩家必須操控角色，並成功將排球打至對方場域中，即可獲得分數。本遊戲為雙人遊戲。

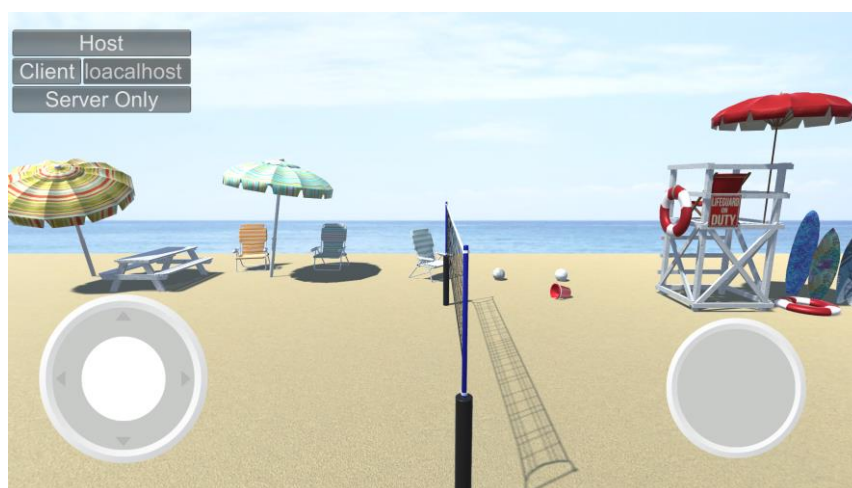


圖 8、多人電腦連線版遊戲畫面

遊戲開始，在同一網路環境下，如連接相同 WIFI。玩家一需按下 HOST 按鈕已建立房間，而玩家二則輸入區域網 LAN 玩家一主機 IP address 以進入玩家一房間內遊玩。

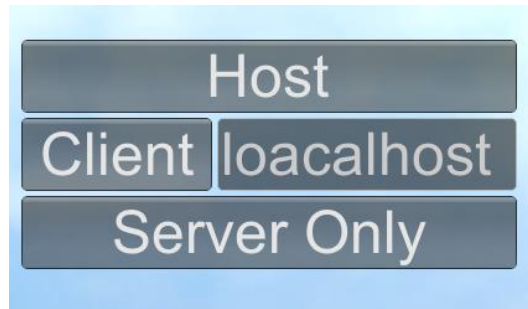


圖 x、連線相關設定按鈕

進入遊戲，若玩家一為房間建立者(HOST)則腳色為白色；反之，玩家二為訪客(Client)進入，則玩家二腳色為黑色。

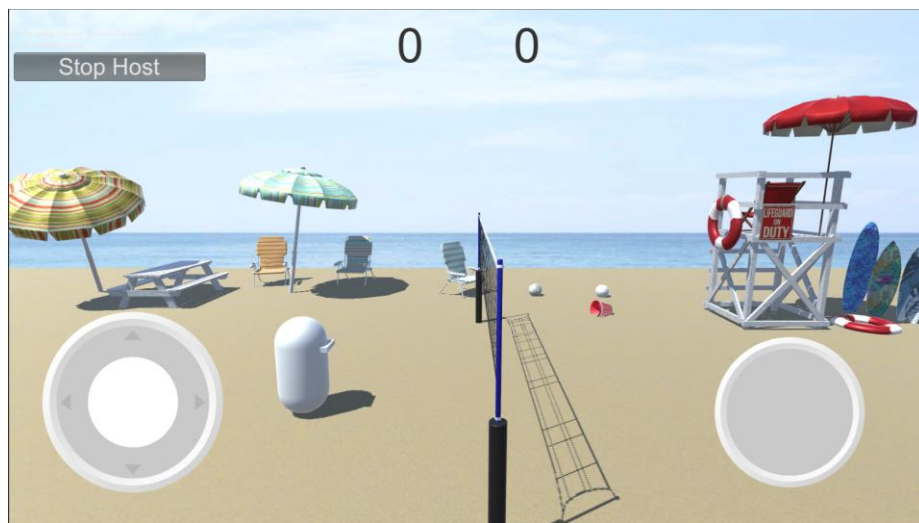


圖 x、 HOST 玩家遊戲進入初始畫面

當玩家二進入，遊戲才會正式開始，並且創建一顆排球。並且畫面會同步運行(些微的延遲誤差，分數也是同步的)。

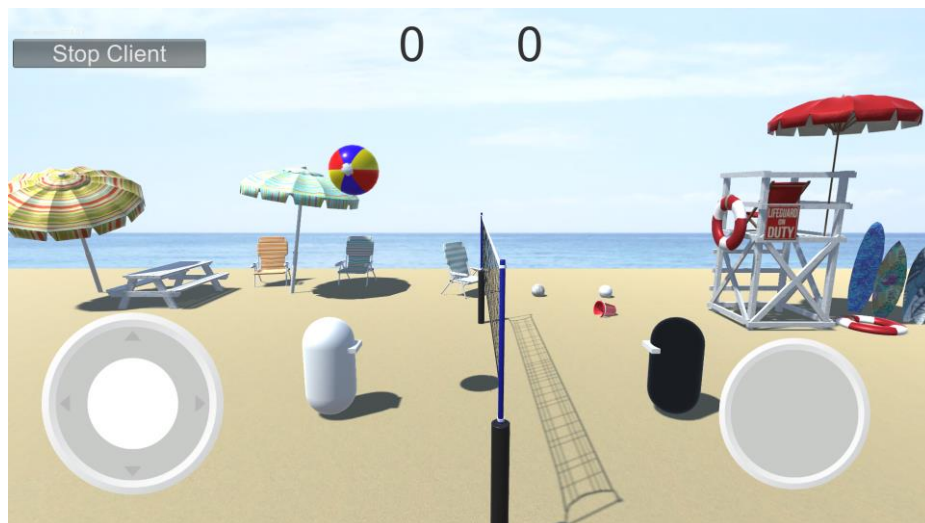


圖 x、Client 加入遊戲畫面

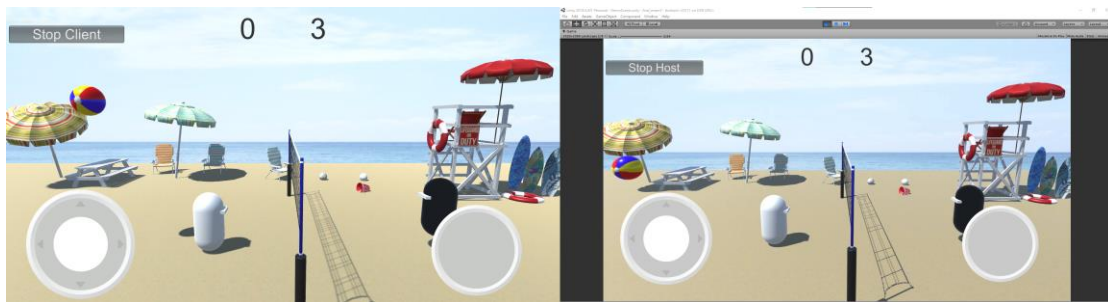


圖 x、雙人連線遊玩畫面

訪客退出遊戲後，房間主持人的畫面會發現訪客玩家已經退出並且房間尚未撤銷並等待訪客玩家進入。若房間主持人退出遊戲，則訪客畫面將會回到遊戲一開始介面。

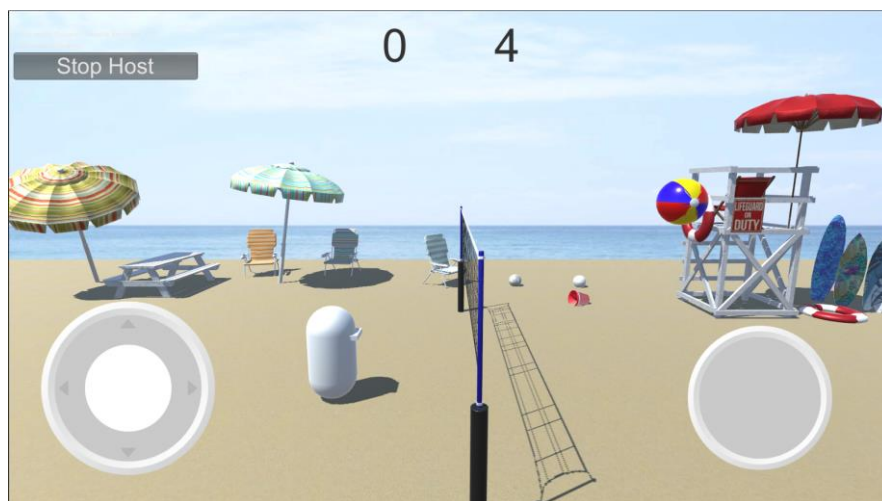


圖 x、Client 端玩家退出，HOST 玩家遊戲畫面



圖 x、HOST 玩家退出，Client 端玩家遊戲畫面

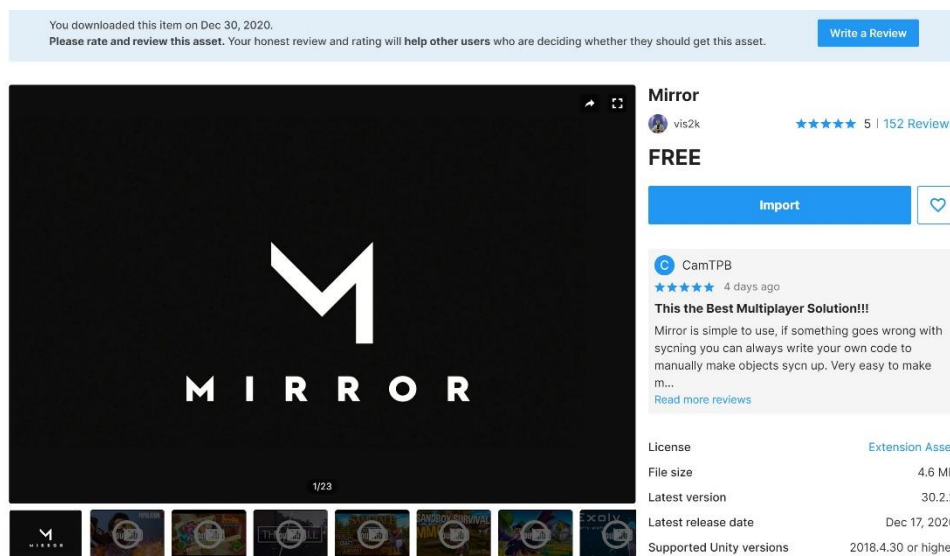


圖 x、 本遊戲 ICON 圖示

2. 程式碼&遊戲物件

本遊戲多人電腦連線版本使用 Unity Assest 中所提供的兩個遊戲套件，分別為

1. Mirror：



此套件是以 UnityNetworking 為基礎，所架構出的 API 應用模塊。藉由創建 NetworkManager 建立基本遊戲的網路環境，與連線方式等。

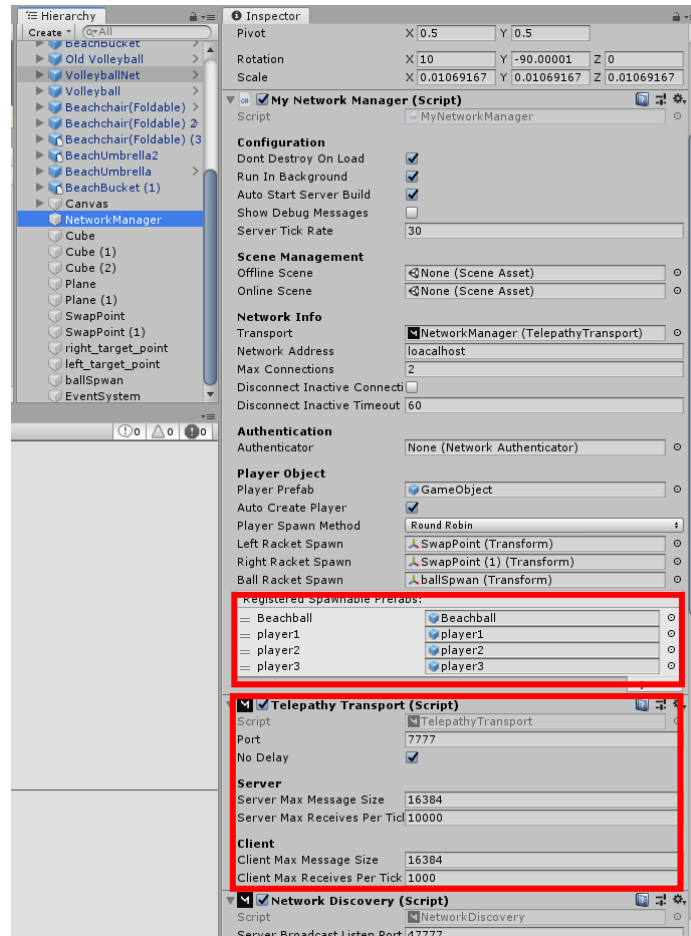


圖 x、 NetworkManager 參數設定介面，
並設定相關生成 Prefab 與連線封包傳輸

引入 Network Manager HUD 來對遊戲建立 GUI 網路操控介面。

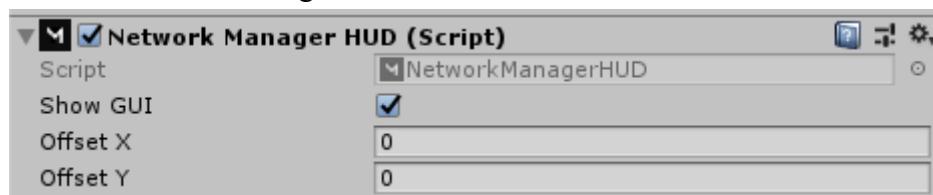


圖 x、 連線設定 GUI 物件

建立玩家出生點，因為是網路環境下連線，因此都需要加入 Network Identity

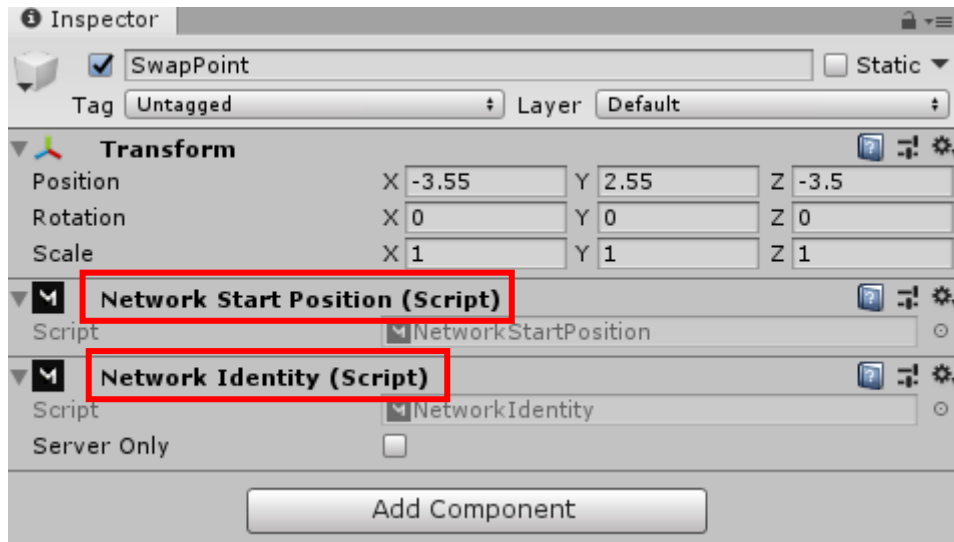


圖 x、網路遊戲玩家生成點與網路狀態下的物件身分

玩家的產生都是一 Prefab 方式運行。

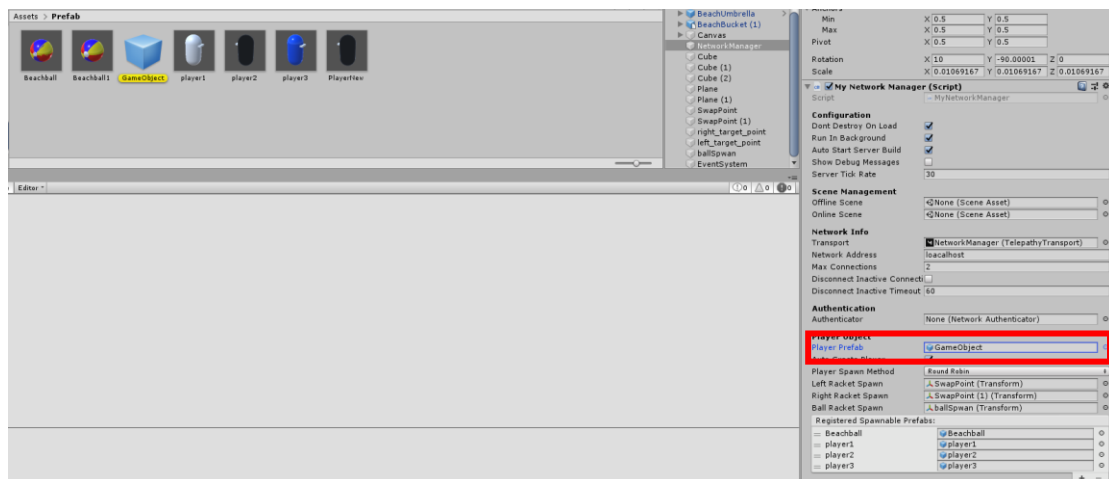


圖 x、Prefab

遊戲角色塑模，建立角色 Prefab，並給予網路環境下的身分。

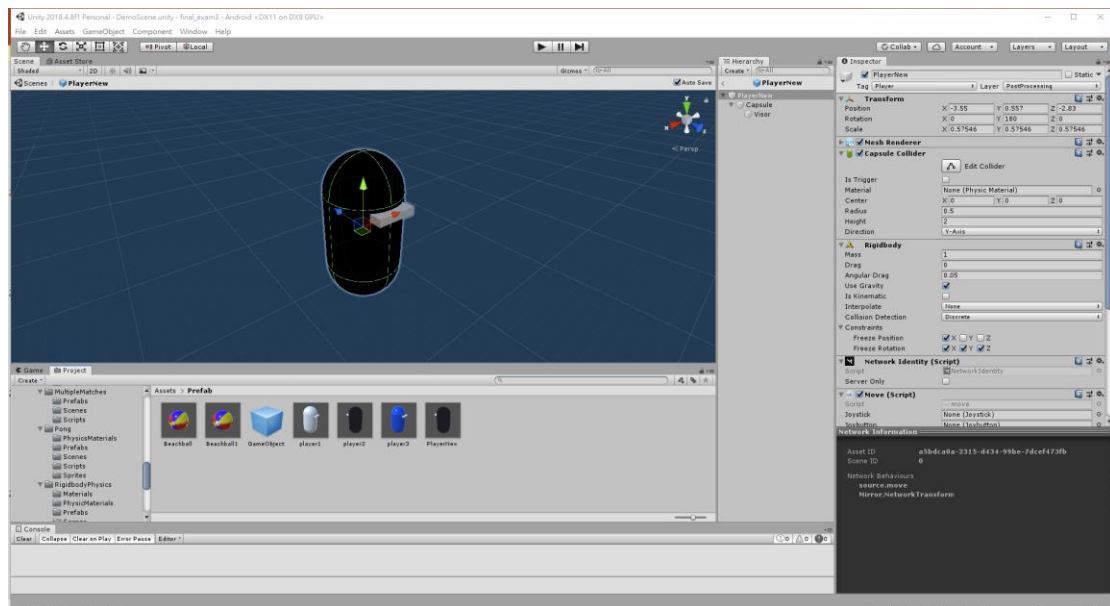


圖 x、角色塑模

繼承 NetworkManager.cs 的 MyNetworkManager.cs 建立判斷玩家人數進入，並產生排球物件與相關連線狀態。並且檔案都是使用 **NetworkBehaviour**。

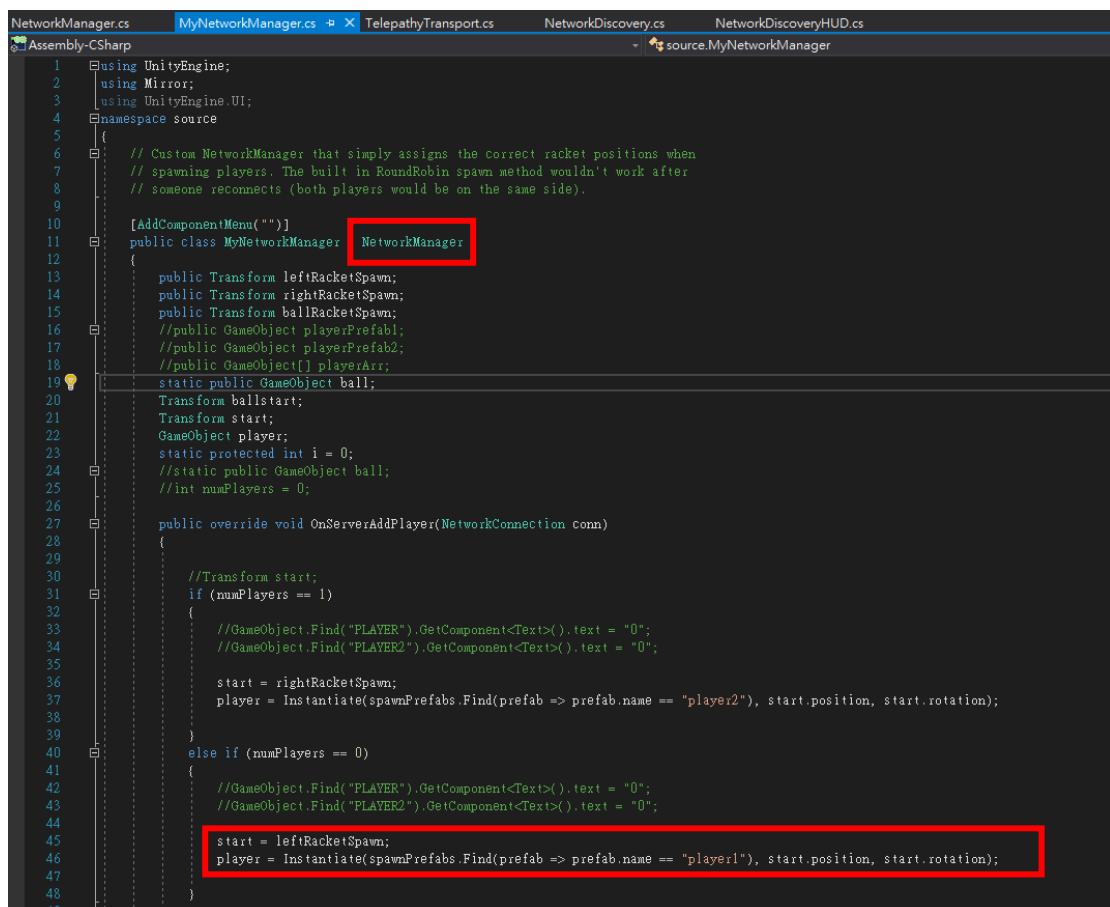
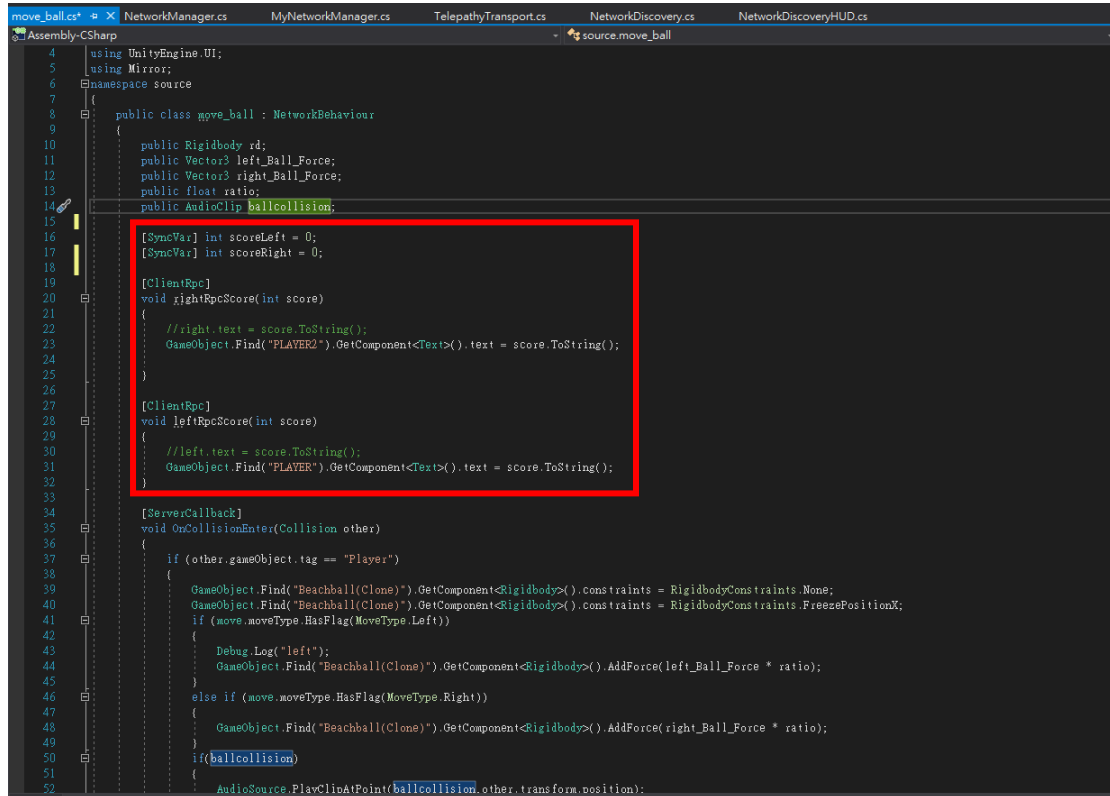


圖 x、MyNetworkManager.cs 程式運作邏輯

分數同步方式，利用[ClientRpc]與[SyncVar]與 HOST 同步。並利用[ServerCallback]判斷球的碰撞同步動作



```
using UnityEngine.UI;
using Mirror;
namespace source
{
    public class move_ball : NetworkBehaviour
    {
        public Rigidbody rd;
        public Vector3 left_Ball_Force;
        public Vector3 right_Ball_Force;
        public float ratio;
        public AudioClip ballcollision;

        [SyncVar] int scoreLeft = 0;
        [SyncVar] int scoreRight = 0;

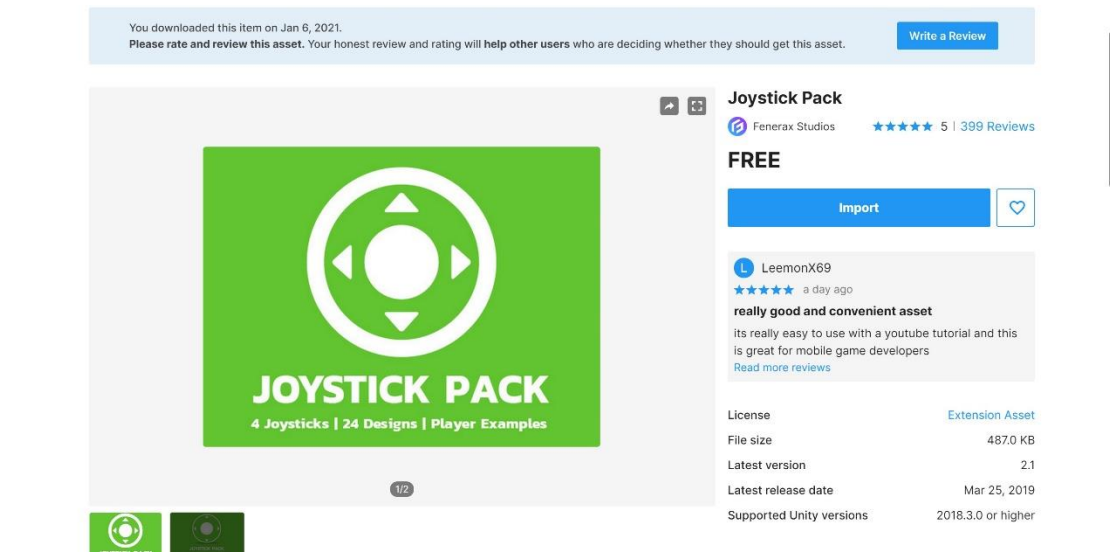
        [ClientRpc]
        void rightRpcScore(int score)
        {
            //right.text = score.ToString();
            GameObject.Find("PLAYER2").GetComponent<Text>().text = score.ToString();
        }

        [ClientRpc]
        void leftRpcScore(int score)
        {
            //left.text = score.ToString();
            GameObject.Find("PLAYER").GetComponent<Text>().text = score.ToString();
        }

        [ServerCallback]
        void OnCollisionEnter(Collision other)
        {
            if (other.gameObject.tag == "Player")
            {
                GameObject.Find("Beachball(Clone)").GetComponent<Rigidbody>().constraints = RigidbodyConstraints.None;
                GameObject.Find("Beachball(Clone)").GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezePositionX;
                if (move.moveType.HasFlag(MoveType.Left))
                {
                    Debug.Log("left");
                    GameObject.Find("Beachball(Clone)").GetComponent<Rigidbody>().AddForce(left_Ball_Force * ratio);
                }
                else if (move.moveType.HasFlag(MoveType.Right))
                {
                    GameObject.Find("Beachball(Clone)").GetComponent<Rigidbody>().AddForce(right_Ball_Force * ratio);
                }
                if(ballcollision)
                {
                    AudioSource.PlayClipAtPoint(ballcollision, other.transform.position);
                }
            }
        }
    }
}
```

圖 X、 分數同步方法

2. Joystick Pack :



You downloaded this item on Jan 6, 2021.
Please rate and review this asset. Your honest review and rating will help other users who are deciding whether they should get this asset. [Write a Review](#)

Joystick Pack
Fenerax Studios ★★★★★ 5 | 399 Reviews
FREE
[Import](#) [Heart](#)

LeemonX69
★★★★★ a day ago
really good and convenient asset
its really easy to use with a youtube tutorial and this is great for mobile game developers
[Read more reviews](#)

License: Extension Asset
File size: 487.0 KB
Latest version: 2.1
Latest release date: Mar 25, 2019
Supported Unity versions: 2018.3.0 or higher

利用此套件作為虛擬搖桿控制遊戲角色的移動與跳躍狀態。

創建 Joystick Pack 相關運作函示與速度等參數調整。

```
static public MoveType moveType;

public Joystick joystick;
public joybutton joybutton;
public bool jump;

void Start()
{
    joystick = FindObjectOfType<Joystick>();
    joybutton = FindObjectOfType<joybutton>();
}
```

圖 x、 遊戲開始時載入 Joystick 物件

```
var rd = GetComponent<Rigidbody>();
rd.velocity = new Vector3(joystick.Vertical * 10f, rd.velocity.y, joystick.Horizontal * 10f);
if(!jump && joybutton.Pressed)
{
    jump = true;
    rd.velocity += Vector3.up * 0.5f;
}
if (jump && joybutton.Pressed)
{
    jump = false;
}
```

圖 x、 Joystick 移動邏輯與速動參數設定

組員：

b0743021 資工三 余孟倫

b0743022 資工三 林聖博

b0643023 資工四 張皓雲