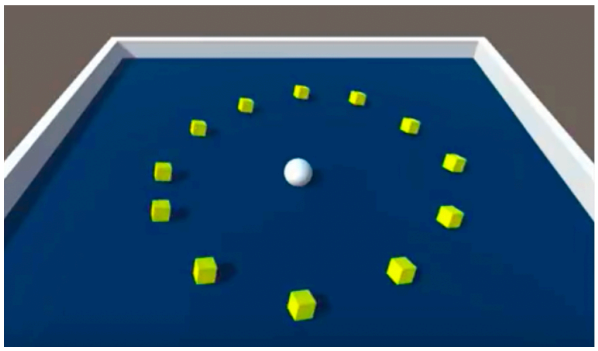
Introduction of Computer Networks Final Project

Rolling a Ball

第六組　　2020 年 6 月 11 日
B06901011　　電機三　呂承樺
B06901015　　電機三　徐芊祺
R08922195　資工所研一　夏睿陽

一、 Introduction 簡介

我們設計的遊戲是參考網路上的「ROLL-A-BALL」，玩家利用鍵盤上的「W」、「A」、「S」、「D」來移動遊戲畫面中的球，並利用撞擊黃色方塊來得到分數(如下圖一)。我們實作的版本才是將場景設成叢林，讓多個玩家可以同時在叢林中移動，除了用上述四個鍵盤指令來達到前後左右移動的效果，還加入了「SPACE」來達成玩家跳躍的動作，以便在凹凸不平的場景中移動。



圖一

二、 Game Framework 遊戲架構

為了實現多人連線的遊戲，我們將架構區分成 Server 端與 Client 端。Server 端則是模擬的遊戲場景，而 Client 端是真實玩家看到的遊戲場景，也可以說是根據 Server 端傳送的 packet 定義出來的世界。

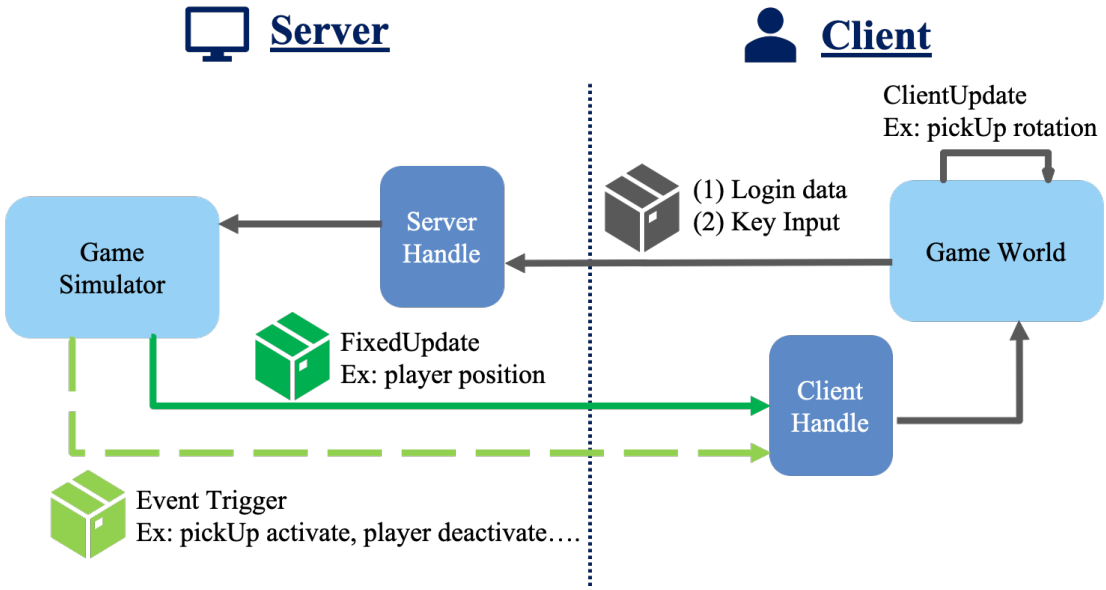
Packet 的部分簡單可以區分成兩種：由 Server 傳送出的以及由 Client 傳送出的。其中，Client 傳送出的 packet 只有兩種形式：一是當玩家登入時傳遞給 Server 的玩家資料，另一為玩家鍵盤輸入的「W」、「A」、「S」、「D」指令；

```
public enum ServerPackets
{
    welcome = 1,
    spawnPlayer,
    playerPosition,
    playerRotation,
    destroyPlayer,
    spawnPickup,
    deactivatePickUp,
    activatePickUp,
    playerScore
}
public enum ClientPackets
{
    welcomeReceived = 1,
    playerMovement
}
```

圖二

而 Server 傳遞出的 packet 形式則相對較為複雜，以更新型態區分的話可以簡單分成「固定更新(FixedUpdate)」以及「事件觸發更新(Event Trigger)」兩種類型，前者例子為玩家位置、其他玩家位置等等；後者舉例來說像是玩家新增、玩家死亡、pickUp 出現與消失等等。(詳見上頁圖二)

Game World 的建構是來自 Server 端傳送的 packet，當 packet 傳送到 Client 端時會先經由 Client Handler 進行 packet 的解碼與認證，並在 Game World 中更新 packet 中所指定的項目。而 Client 端在經過 Login 的過程後僅會傳遞玩家鍵盤中輸入的指令給 Server，在傳遞到 Server 後會先經由 Server Handler 進行 Client packet 的解碼與辨識，並在 Server 端進行各項數值的更新與計算，最後再將更新後的數據經由不同種類的 packet 傳遞給 Client，來完成一個完整的循環。(見下圖三)



圖三

三、 Details Specification 詳細說明

我們用來傳輸 packet 的方式分成 TCP 以及 UDP 兩種，由於 TCP 要為耗費網路資源，因此重要度較低的資訊變改用 UDP 傳送，而將網路資源留給最需要被 Server 與 Client 接收到的資料使用。舉例來說：當一個 Client 登入進遊戲，他的玩家資料「必須」被 Server 端得知並建檔，因此 Client 端傳送的 Login Data 就會利用 TCP 傳送給 Server 來確保不會有 packet loss 與 corrupt 的問題；相比而言，每一個 time frame Server 都會傳遞玩家現在的位置以及面朝角度，但是這個資料若在一個 time frame 中出現 packet loss 或

是 corrupt 的問題並不會對遊戲的進行造成顯著的影響，因此，經過評估後這樣固定時間傳輸的 packet 變使用 UDP 傳送以節省網路資源。下表為整理後用 TCP 與 UDP 傳送的 packet。

TCP	UDP
● Client Packet - Login Data ● Server Packet - Event Trigger	● Client Packet - Key Input ● Server Packet - FixedUpdate

另外，在傳送資料時，不管是利用 TCP 還是 UDP 傳輸 packet，我們都將傳送對象區分成三種形式：只傳送給特定 client、傳送給所有 client、傳送給除了特定 client 之外的每個玩家(見下圖四)。這樣的設計是希望確保資料傳輸的唯一性與準確度，同時還能節省傳輸資訊，避免不必要資源的浪費。例如：在 Server 端計算出某個玩家因為撞到 pickUp 而分數增加時，這樣的資訊只需要傳遞給該玩家，讓該玩家的螢幕上顯示分數更新，而不需要昭告所有的玩家；相反的，若有玩家在遊戲期間離開或死亡，這樣的資訊就必須告知所有仍然在線的 client，如此一來才能確保所有玩家的遊戲狀態同步且正確。

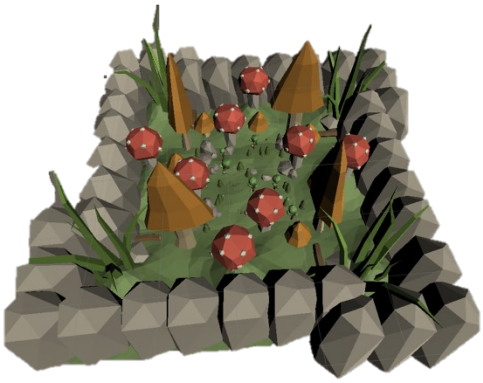
```
#region TCP basic method
4 references
private static void SendTCPData(int _toClient, Packet _packet){
    _packet.WriteLength();
    Server.clients[_toClient].tcp.SendData(_packet);
}

4 references
private static void SendTCPDataToAll(Packet _packet){
    _packet.WriteLength();
    foreach (Client _client in Server.clients.Values){
        if (_client.tcp.isConnected()){
            Server.clients[_client.id].tcp.SendData(_packet);
        }
    }
}

0 references
private static void SendTCPDataToAll(int _exceptClient, Packet _packet){
    _packet.WriteLength();
    foreach (Client _client in Server.clients.Values){
        if (_client.tcp.isConnected()){
            if (_client.id != _exceptClient){
                Server.clients[_client.id].tcp.SendData(_packet);
            }
        }
    }
}
```

```
#region Server UDP send method
0 references
private static void SendUDPData(int _toClient, Packet _packet){
    _packet.WriteLength();
    Server.clients[_toClient].udp.SendData(_packet);
}

2 references
private static void SendUDPDataToAll(Packet _packet){
    _packet.WriteLength();
    for (int i = 1; i <= Server.MaxPlayers; i++){
        Server.clients[i].udp.SendData(_packet);
    }
}

1 reference
private static void SendUDPDataToAll(int _exceptClient, Packet _packet){
    _packet.WriteLength();
    for (int i = 1; i <= Server.MaxPlayers; i++){
        if (i == _exceptClient) continue;
        Server.clients[i].udp.SendData(_packet);
    }
}
```

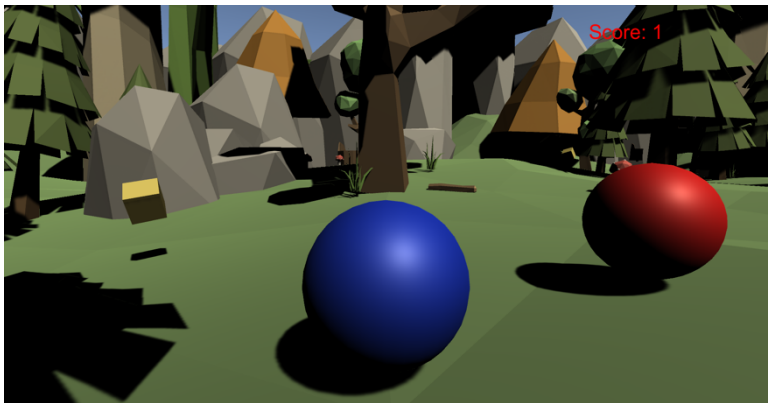
圖四(左圖為 TCP，右圖為 UDP)

四、 Results 實作成果

下圖五是我們實做出來的遊戲世界，內有大小不同的石塊、蘑菇、樹木來阻擋玩家視線，讓遊戲更生動有趣。圖六顯示玩家為藍色球，而在畫面中所見其他玩家則皆為紅色球，可以撞擊加分的則為在半空中旋轉的黃色方塊。

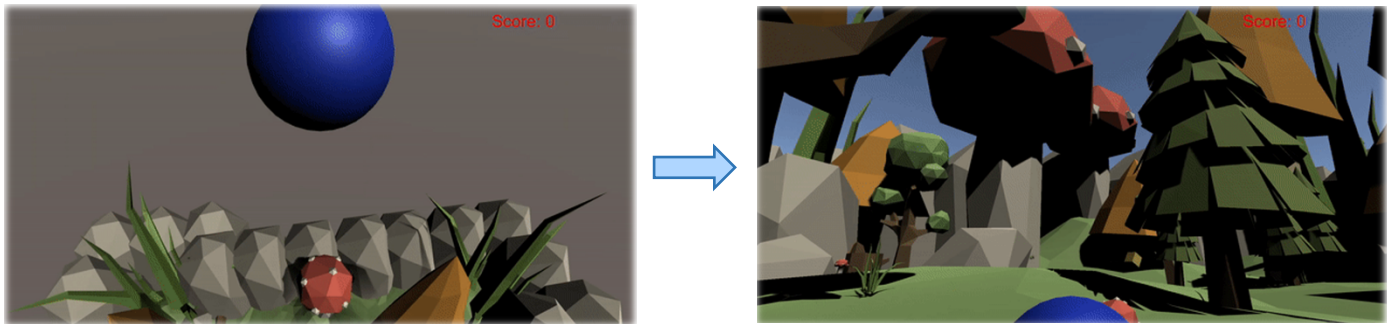


圖五

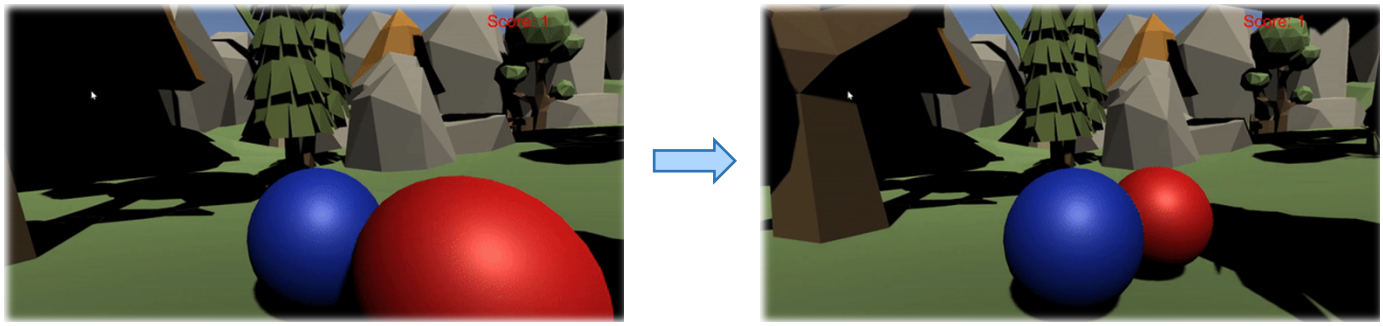


圖六

另外，我們在實作中利用重力讓玩家能貼平遊戲空間，這也是當玩家登入時會從空中墜落的原因，如下圖七。當玩家互相碰撞時，會因為感測到物體而避開，不會重疊穿過去，如下圖八。



圖七



圖八

五、 Problems and Discussion 問題與討論

為了達到多人連線遊戲的網路暢通，我們做了如下述幾項的努力：

1. 在 Client 傳輸 packet 時，僅傳送 x, y, z 座標，而非整個物件，用以節省網路資源。
2. 在 FixedUpdate 中所傳輸的 packet 均利用 UDP 傳輸，儘管有可能產生 loss 與 corrupt，但是因為更新時間極短，些許的 loss 不易被肉眼察覺。
3. 因為 pickUp 在我們設計的遊戲中皆在固定的位置，而旋轉角度並不會影響其他玩家的遊戲體驗，因此我們讓 Client 端自行計算 pickUp 旋轉的位置與角度，而非傳遞給 Server 計算，如此一來，便可以減少 Server 的工作量。

然而，在 Client 端運算的資料仍偏少，很多人會認為玩家的位置與旋轉角度可以在 Client 端計算後在輸出給 Server，但是我們認為若玩家的位置能夠在 Client 端計算，勢必就有機會偽造座標，而有作弊的可能性，因此綜合玩家的遊戲體驗品質與遊戲本身公平公正性，我們仍然堅持多數資料在 Server 端計算。

六、 Work Distribution Chart 分工表

呂承樺	徐芊祺	夏睿陽
● 玩家分數更新、傳遞與程式實作 ● PPT ● 紙本報告	● 遊戲規則設計 - 玩家碰撞 - pickUp 碰撞得分	● Server/Client 網路架構設計與程式實作

七、 References 文獻參考

1. https://reurl.cc/vDA4vo
2. https://docs.unity3d.com/Manual/index.html
3. https://reurl.cc/QdxG59