附件一 專題研究

私立淡江大學

資訊工程學系

專題研究

以創新為主軸自主開發全新射擊遊戲之研究

【Ｇｏｄ顯神威】

專　題　生：董行咨

指導教授　：林志豪

中華民國一百一十一年九月

附件一　專題研究

專題簡介

專題名稱 — GOD顯神威

組員 — **董行咨** 陳裕坤 沈軍儒

類型 — 多人連線射擊遊戲

風格 — 創新、趣味

研究平台 — Unity物理引擎(2020.3.26版、VSCode)

發布平台 — PC

使用素材 — Unity Store、Photon Network

開發程式語言 — C#

資料來源 — Youtube、Udemy、國外論壇

摘要

近年來市面上遊戲只注重營收，愈發不重視玩家體驗，導致許多遊戲的生命週期嚴重限縮。遊戲廠商透過自媒體大力宣傳，但實際遊戲內容卻相當貧乏，與我們記憶中初次接觸遊戲所帶來的樂趣與感動背道而馳，基於我們多年的遊戲體驗而發想此項目。

技術問題與開發困境

由於Unity舊版的多人網路連線套件(Network Manager)過於老舊已被刪除，所以改使用Unity團隊開發的過渡產品PhotonNetwork免費版做為替代。

最新版本Unity存在導入資料失敗率高、部分遊戲機制設計改變、舊有套件刪減等問題，與當時從網路上蒐集到的資料相差甚大，故改用較早期之版本。

附件一　專題研究

內容描述

(1)元素融合:為了提升玩法多樣性，經組員商討決定，挑選幾樣較為經典的遊戲元素作為主要開發方向。

(2)創新玩法-推箱子&擊殺對手:遊戲開始後雙方需爭搶場地中央的方形衛生紙捲筒(圖一)，以射擊的方式推動捲筒，到達指定地點(圖二)即獲勝。遊戲途中亦可將對手擊殺使其被迫傳送至重生點，達到干擾拖延的目的。

(3)遊戲亮點:槍戰為許多早期遊戲的經典元素，大多是以打殺為主題。本團隊在保留「槍」一特點的前提下，將子彈(圖三)改成馬桶吸把的造型，並增設吸附效果(圖四)，使子彈能停留在對手、建築、牆壁上一段時間。

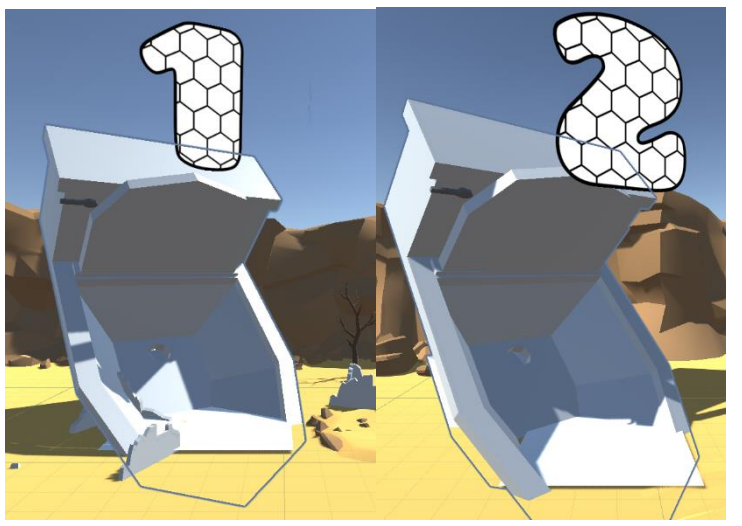
移動方式: 前(W)、後(S)、左(A)、右(D)

跳躍: 單擊空白鍵

(圖一)

(圖二)

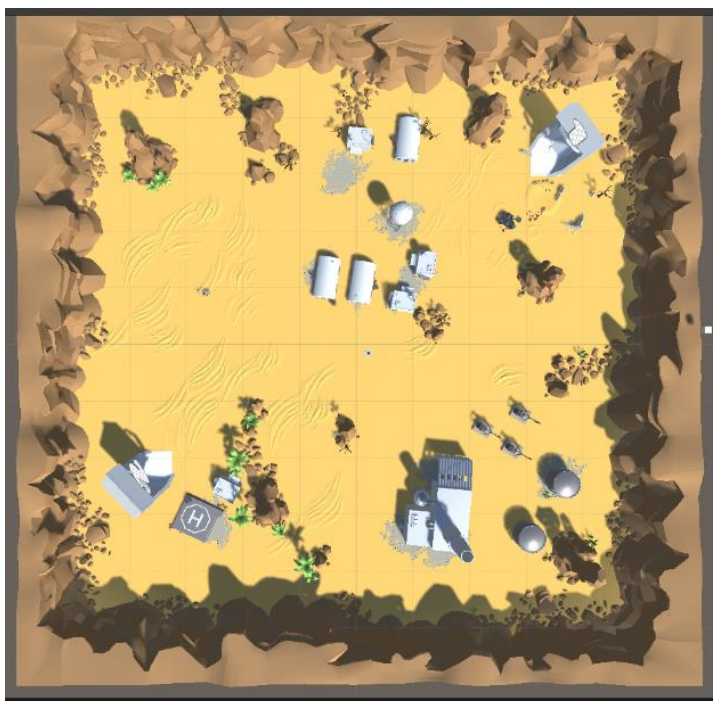
馬桶示意圖(球門)



(圖三)

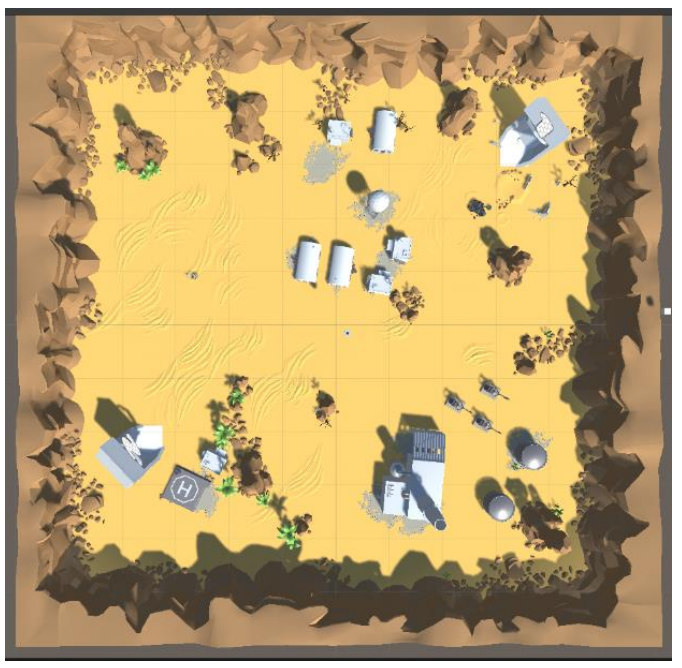
示意圖(圖四)

地圖(俯視角)



附件一　專題研究

技術運用與挑戰



(1)地圖建設:

因考量成本問題，此遊戲地圖由本團隊從網路上蒐集適合風格之模型，以手動添加物件的方式逐步製作。包含水塔、工廠、鑽油井...等。在不斷磨練添加地圖物件技巧的同時，也對此方面有了更深的了解。(細節會於*第19頁*的影片詳細介紹)

```
{

    //旋轉身體的視角
    float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
    float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

    xRotation -= mouseY;
    xRotation = Mathf.Clamp(xRotation, -90f, 90f); // 讓頭部旋轉在90度

    transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
    playerBody.Rotate(Vector3.up * mouseX);
}
```

(2)Camera控制:

偵測滑鼠之移動軌跡，控制玩家目視方向之Rotation，以及角色頭部模組的轉向，並藉由Photon View之IsMine屬性防止雙方取得對手的Camera控制權，達到更優良的遊戲體驗。

(3)角色移動:

通過偵測鍵盤按鍵獲取玩家欲移動之X、Y、Z軸方位，並給予角色模組相應速度的正向力。監控角色是否接觸地面，用以限制跳躍次數。

(Unity預設移動按鍵為WASD，跳躍鍵為空白鍵)

```
    Vector3 move = transform.right * x + transform.forward * z;


    controller.Move(move * speed * Time.deltaTime);

    if(Input.GetButtonDown("Jump") )
    {
        velocity.y = Mathf.Sqrt(jumpHeight * -2f * gravity);
    }

    velocity.y += gravity * Time.deltaTime;


    controller.Move(velocity * Time.deltaTime);

}
```

技術運用與挑戰

(4)音效建置:

一款優質的遊戲必須有合適的音效做襯托。為創造更加生動的遊戲環境，在角色移動的過程中適時播放腳步聲是很關鍵的一環。此處除了運用偵測玩家是否進行水平與垂直移動的技巧，還須使用isGround屬性確認玩家碰觸地面，並考量播放的音量。當使用滑鼠左鍵開火時，也會放出提前設置好的槍聲音效。

```
if(isGrounded && velocity.y<0)
{
    velocity.y = -2f;

}
float x = Input.GetAxis("Horizontal");//input水平
float z = Input.GetAxis("Vertical");//input垂直
if((x != 0 || z != 0))
{


    if(footPlayer.isPlaying == false)
    {

        footPlayer.Play();
    }

}else{
    footPlayer.Stop();
```

```
void ProcessInputs()
    {
        // 按下發射鈕
        if (Input.GetButtonDown("Fire1"))
        {
            if (!IsFiring)
            {
                IsFiring = true;
            }
        }
        // 放開發射鈕
        if (Input.GetButtonUp("Fire1"))
        {
            if (IsFiring)
            {
                IsFiring = false;
            }
        }
    }

[PunRPC]
public void RpcShoot()
{//重製計時器
        timer = 0;
        //產生火焰
        Instantiate(FirePre,FirePoint.position,FirePoint.rotation);



        //更新UI
        // BulletText.text = bulletCount + "";
        //子彈數減少)
        bulletCount--;


        //槍聲
        GunPlayer.PlayOneShot(clip);


        // 射擊特效
        // Debug.Log("你射出一發子彈");

        if(bulletCount == 0)
        {
          Invoke("Reload",1.5f);

        }
      if (BulletPoint.transform != null)
      {

   GameObject bullet = Instantiate(BulletPre,BulletPoint.transform.position,BulletPoint.transform.rotation);
   //產生子彈
   Rigidbody rBody = bullet.GetComponent<Rigidbody>();
   rBody.AddRelativeForce(Vector3.forward * 1000 * Time.deltaTime,ForceMode.Impulse);
      }
}
```

(5)射擊與特效:

相較於單機遊戲，多人連線遊戲的射擊機制更為複雜，除了需要對開火特效、開火點、方向、發射力度、音效做大量的調整外，需以呼叫服務器的方式開火射擊，通過廣播的形式告知場上玩家開火的資訊。

此處最大的困難在於，Photon Network為過渡用之測試階段產品，對於如何達成上述效果的教學與資料甚少，並缺乏完整性。在進行該處研究時，雖花費了不少時間成本，但也從中學習到了三種不同的實作手法。除了克服困難的成就感外，收獲也相當豐富!

附件一　專題研究

技術運用與挑戰

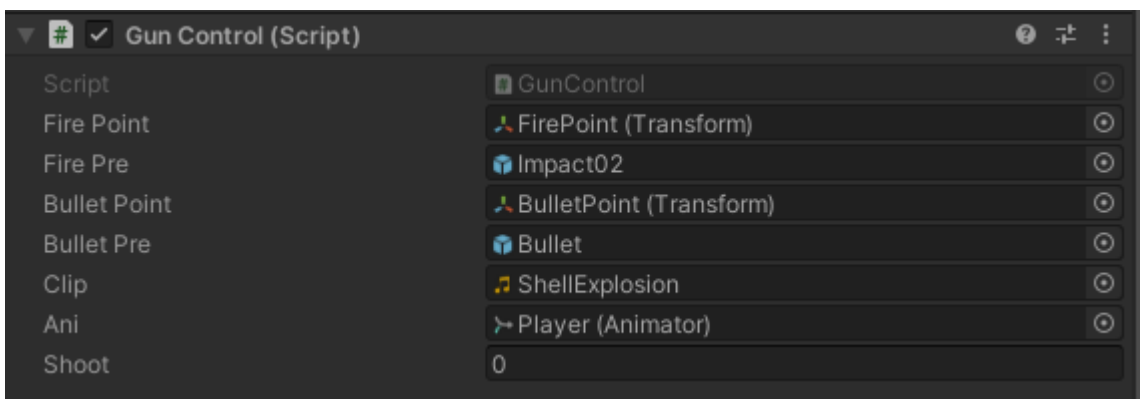
(6)物件刪除:

因子彈為馬桶吸把造型的緣故，在射出後會佔有一定體積，隨著遊戲期間發射數量的上升，定會造成物間堆積的情況，故必須設計一機制用以刪除多餘的子彈。

此處運用計時器的方式，在子彈射出後的四秒鐘呼叫Destory()對該物件進行刪除。

```
65
66      // Update is called once per frame
67      void Update()
68      {
69
70          age += Time.deltaTime;
71          if(age > maxage)
72          {
73              Destroy(gameObject);
74          }
75
76      }
77
78
79  }
80
```

```
void Start()
{
    PV = GetComponent<PhotonView>();

    age = 0.0f;

    GetComponent<Rigidbody>().AddForce(transform.forward*300);


}


private void OnCollisionEnter(Collision collision)
{
    Debug.Log(collision.collider.tag);
    if(collision.collider.tag == "Target")
  {
    rBody.isKinematic = true;
  }
}
```



(7)子彈的發射與停止:

此圖為子彈控制器的部分code。上方為子彈於發射點生成時，賦予該物件物理特性，並向提供前方向的加速度用以推出馬桶吸把。

下方則是第11頁遊戲亮點之圖四所提到，使子彈吸附於物體上的效果。運用OnCollisionEnter()搭配物件Tag之碰裝偵測，在子彈接觸物體後取消其運動特性。

在將火花製造點、子彈發射點、槍聲、子彈等機制與模型調適好後，分別放入預先寫好的欄位中使其正常運作。

附件一　專題研究

技術運用與挑戰

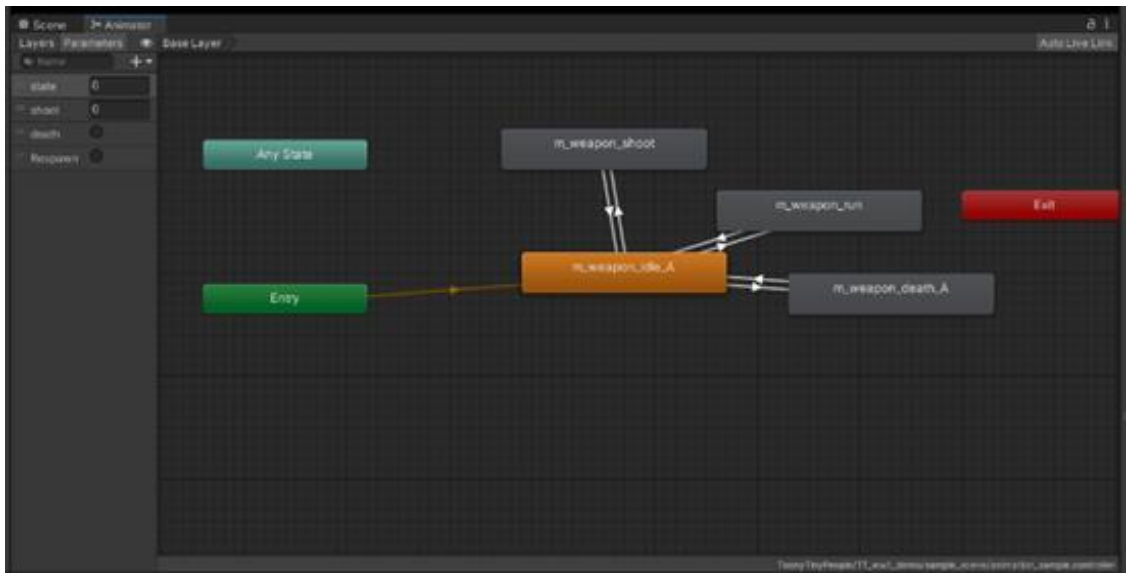
(8)動畫控制:

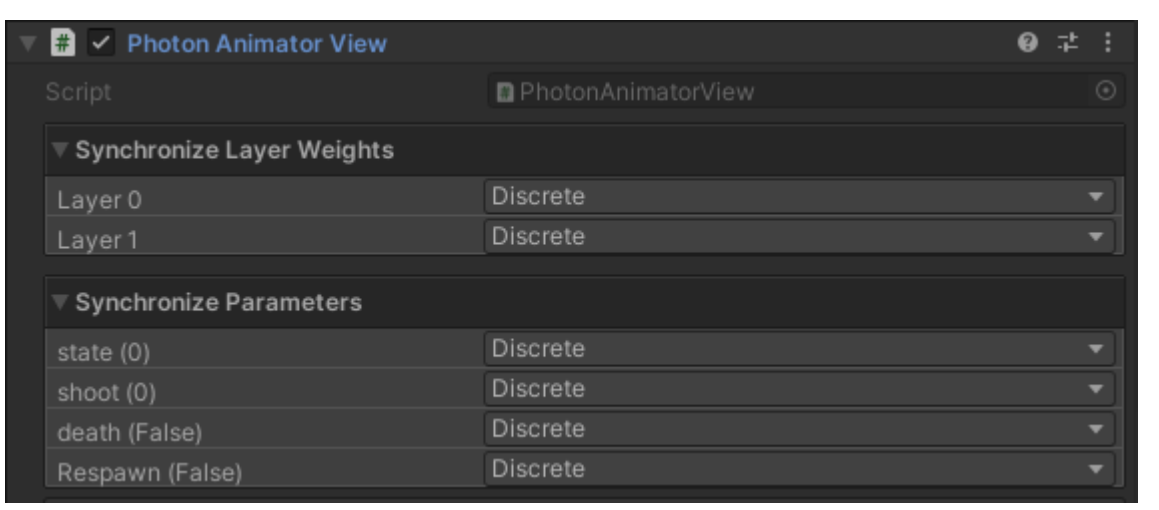
由於Unity store提供的免費人物模組非常稀少，且許多素材不提供已設計好的動畫流程控制器，只得後續自主研發。

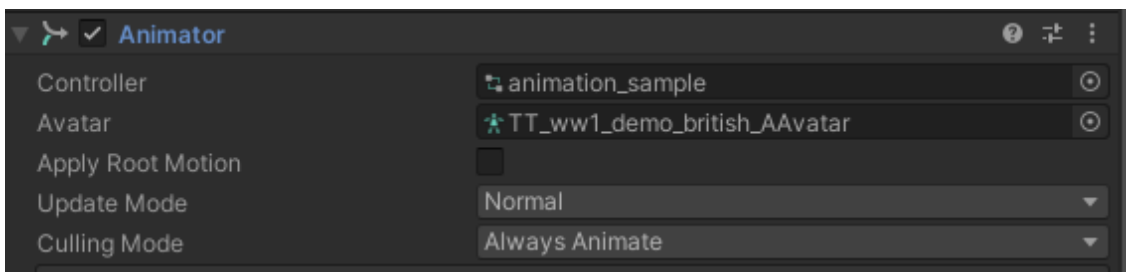
此處需要精準監控玩家的角色情況，故採用GetKey()獲取操作者所使用的按鍵，以此改變控制器的流程狀態，在行走與站立動畫之間切換。(1為ture 0為false)。

此模組還有提供開槍時因後座力影響的槍枝抖動特效，藉由GetMouseButton()獲取滑鼠點擊與鬆開的訊號，以改變Shoot動畫狀態。將設計好的animation_sample動畫控制器放入Animator元件中，最後通過Photon提供之套件Photon Animator View將各動畫狀態同步給場上玩家。

```
if(PV.IsMine)
{
if(Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.D))
{
    state=1;

}else{
    state=0;
}
}else{
    return;
}

ani.SetInteger("state",state);

if(Input.GetMouseButton(0))
{
    shoot=1;
}else{
    shoot=0;
    }
     ani.SetInteger("shoot",shoot);
```





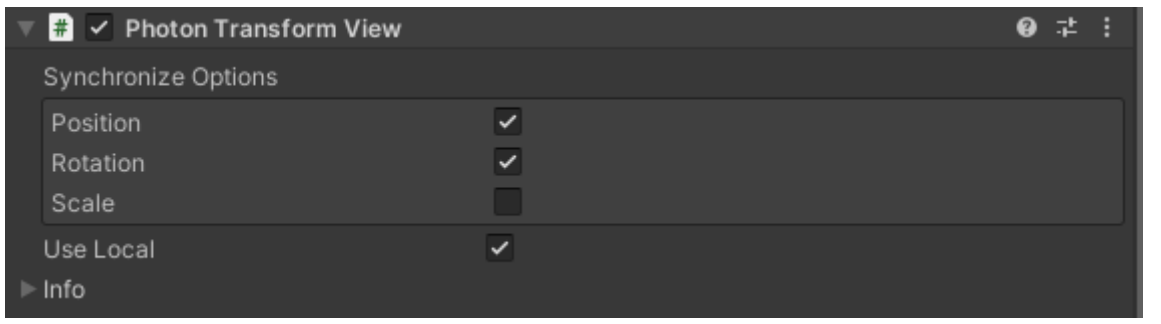


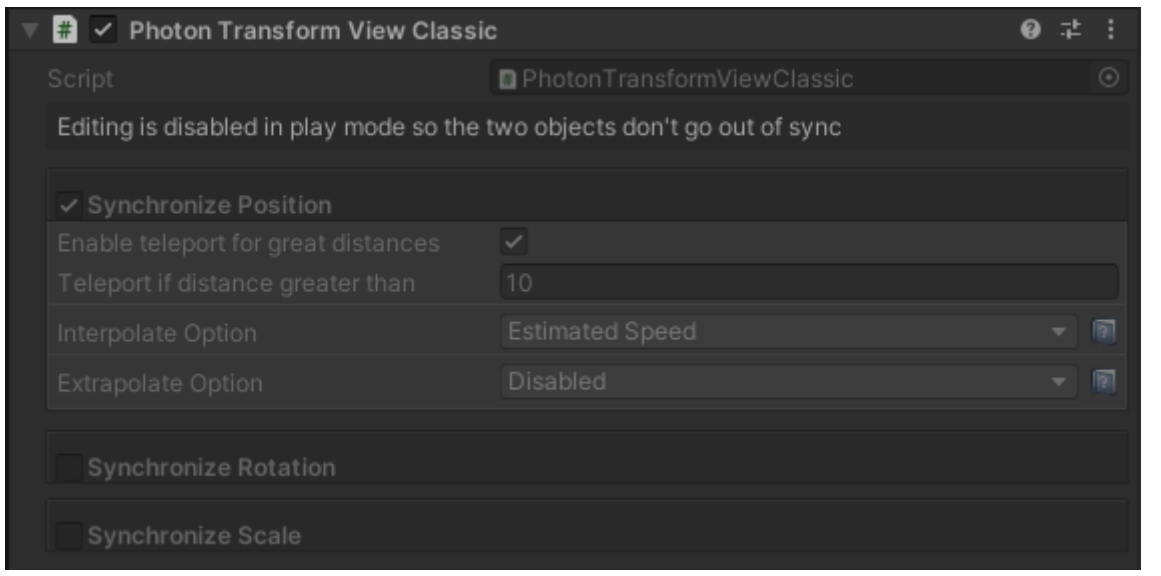
附件一　專題研究

技術運用與挑戰

(9) 同步角色與物件位置:

此處採用Photon Transform View 和 Photon Transform View Classic套件來同步場上物件、角色的Position and Rotation，後者更是可以設定單位時間之同步次數，達到更為精準的校正效果，但由於Photon服務器距離台灣較遠，延遲與玩家人數上限都相對不理想，對設備本身的效能要求也較高。






```
footPlayer = GetComponent<AudioSource>();
controller = GetComponent<CharacterController>();
ani = GetComponent<Animator>();
PV = GetComponent<PhotonView>();
if(!PV.IsMine)
{
    Destroy(GetComponentInChildren<Camera>().gameObject);
    Destroy(gameObject.GetComponent<CharacterController>());
    Destroy(this);

    if(PV.ViewID%2 == 0)
    {
        Destroy(gameObject);

    }

}
```

(10)刪除多餘角色:

由於創建房間與生成角色皆透過Photon服務器，當雙方玩家進入遊戲後即出現不可預期的Bug，角色生成數會從正常的2人變為生成4人，導致遊戲出現諸多不合理性。

此處利用Photon提供之套件Photon View中的View ID角色編號，搭配%運算刪除第2及第4順位生成之角色(前兩位控制權會被1號玩家取得，後兩位反之)，使角色數回歸正常。

附件一　專題研究

技術運用與挑戰

(11)血條:

作為槍戰遊戲最重要的元素之一，血條能夠清楚的顯示敵方剩餘血量，幫助玩家更好的進行遊戲。

此處使用OnTriggerEnter做作物件Tag碰撞偵測，若碰觸Tag為Bullet之模型則改變紅色血條。當血量為0時通過SetTrigger觸發死亡動畫。

```
private void OnTriggerEnter(Collider other ){

    if(other.tag == "Hp" )
    {
        hp.fillAmount = 1.0f;
    }

 if(other.tag == "Bullet" )
{
    if(!PV.IsMine){


    hp.fillAmount -= 0.1f;


     }


    if(hp.fillAmount <= 0)
    {
        GetComponent<Animator>().SetTrigger("death");


        Die();
    }
```

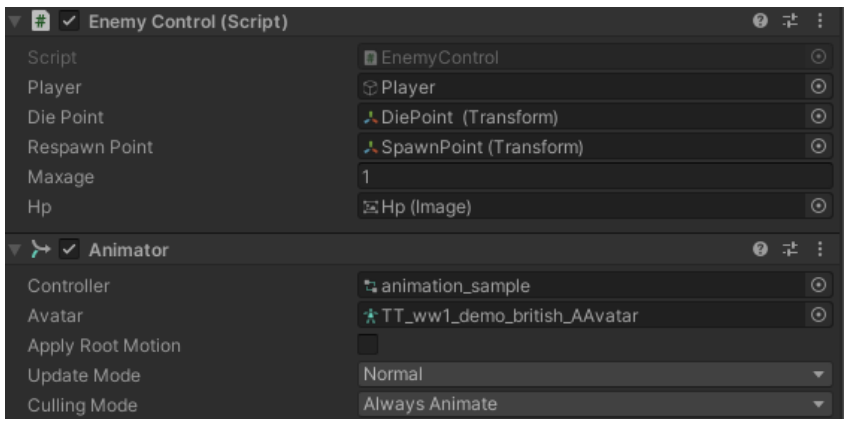
```
void Die()
{
    transform.position = DiePoint.position;

    if(player.transform.position == RespawnPoint.position)
    {
        hp.fillAmount = 1.0f;
        Debug.Log("血量恢復");
    }
    // hp.fillAmount = 1.0f;
    Debug.Log("死亡");
}
```

```
void Update()
{
    if(player.transform.position.y <= - spawnValue)
    {
        Debug.Log("復活");
        GetComponent<Animator>().SetTrigger("Respawn");
        RespawnPoint();
    }

}

void RespawnPoint()
{
    transform.position = SpawnPoint.position;
    // hp.fillAmount = 1.0f;
    // Debug.Log("血量恢復");
```

(12)死亡與重生:

當玩家血條為0時，觸發Die()改變角色之Position至地圖下方死亡點，並重新賦予生命值。

玩家於重生點向地圖下方墜落，當下降高度達到一定值，後再次被傳送回地面兩端預設之復活點，敵方可妥善利用此段時間改變戰局走向。



附件一　專題研究

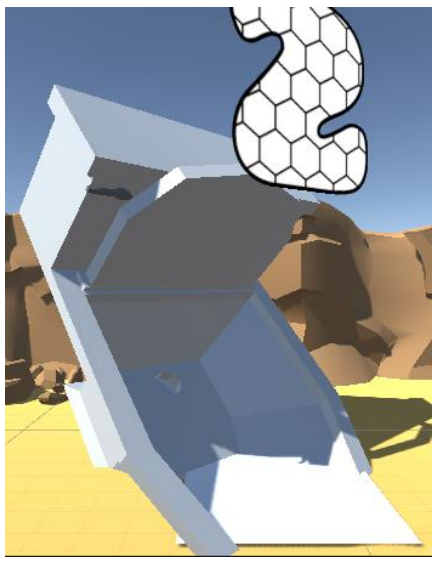
技術運用與挑戰

(13)勝利判定:

遊戲進行至最後階段，勢必會決定一方成為勝利者使其終止。本遊戲的獲勝條件為 ，某一方將衛生紙捲筒推入對方的馬桶(球門)即獲勝。當捲筒接觸到馬桶前方之白色地板(圖一)時，利用OnTriggerEnter偵測物件Tag是否為Box，觸發勝利的UI(圖二)通知雙方玩家遊戲結束(圖三)。通過SutUp()重新編輯UI顯示之內容(圖四)，達到預期效果。

未來若要繼續進行深入研究，此功能可有更深更廣的應用。

圖一



圖三

```
private void OnTriggerEnter(Collider other )
{
    if(other.tag == "Box" )
    {
      GameOver();
      // Debug.Log("一號玩家贏了");
      PrintTextWinner1();
    }


}




public void PrintTextWinner1(){
    print("Player1 is Winner");
}

public void GameOver(){
    GameOverScreen.SetUp();
}
```

圖二

```
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class GameOverScreen : MonoBehaviour
{
    public Text gameovertext;

    public void SetUp(){
        gameObject.SetActive(true);
        gameovertext.text = "1號玩家贏了";

    }
    public void SetUp2(){
        gameObject.SetActive(true);
        gameovertext.text = "2號玩家贏了";

    }

    public void RestartButton(){
        SceneManager.LoadScene("Main");
    }
}
```

圖四

附件一　專題研究

發表與宣傳製作

(14)成品發表與展示:

在投注了大量的時間與不斷地改進後，產出成品的宣傳自然相當重要。不僅要提綱挈領的展現遊戲內容，更要在其中添加富有趣味性的橋段，以吸引觀看者嘗試遊玩。

由於製作時間匱乏僅有一週左右，因個人過去有製作遊戲影片之興趣，故主動承擔此影片相關內容，包含橋段企劃、影片錄製、旁白文案、後製字幕、剪輯等工作。不但發揮了特有的專長，還幫助本組在發表內容時取得優秀的成果。這段經歷讓我認識到學習多元化的重要性，吸收不同面向的知識，對未來發展將有正面影響。

影片連結: https://youtu.be/Xq_GtKVlJ7M

附件一　專題研究

參考資料與素材

- 如何使射彈武器黏在敵人或目標上
- 創建多人FPS
- 區域觸發判定
- 產生勝利條件
- 如何在Unity中檢測某個區域的物體
- 參考玩家角色移動,操控,動作表現與血量的即時同步等細節
- 製作衛生紙捲筒(箱子)
- 玩家死亡後重生點
- 觸發判定輸贏條件

unity assets
免費模組與音效、特效

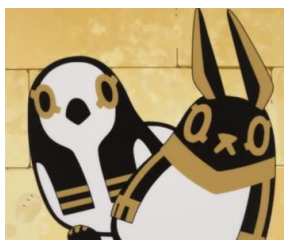
- ◆馬桶數字
- ◆衛生紙捲筒
- ◆馬桶與馬桶吸把
- ◆人物模組
- ◆第一次嘗試的士兵模組
- ◆槍口火花特效
- ◆早期測試AKM
- ◆腳步聲
- ◆免費版Photon2 Network
- ◆基礎FPS教學
- ◆多人連線

附件一　專題研究

總結

本研究以Unity作為平台，使用Photon與Unity store並結合

Udemy課程和網路搜索之大量文獻、教學，加入組員發想的創意與



個人多年涉足遊戲之體驗，成功製作出一款邏輯與完整性充分的全新遊戲。相比此成品，專題研究早期的規劃原先更為廣泛，有豐富多樣的技能點數天賦系統，來創造戰場上形形色色的個體差異。也有各式各樣的創新概念武器，想帶給玩家一種新奇又特殊的遊戲體驗。更有發想以埃及神話為原型，自主設計、建模的角色外觀，和多樣的遊戲模式。但由於計畫成員的中途退出，使我不得不在原有進度的基礎上，修改並刪減部分計畫內容，對整體方向重新規劃與審慎評估，確保在有限的時間內能兼顧計畫的全面性與進度穩定，並每週皆有產出成果報告，是我在這次歷程中學到的寶貴經驗。

在最初摸索場景設計、角色控制、音效處理、動畫原理的過程中，逐漸熟悉C#撰寫邏輯和Unity開發技巧。後續進入連線開發皆段時能更快速發現問題所在，並以正確的方向查詢資料度過難關，也讓我在最後整合階段成功培養合適的節奏與自信。

附件一　專題研究

若要對此研究做更深入的探討，可對UI使用面向做進一步製作。遊戲中給予玩家適當的提示能有效減少其在遊玩過程中的負擔，更快速的掌握遊戲機制與技巧，讓玩家對產品的體驗有所提升。