國立臺灣大學

計算機程式設計(宜大)作業二

獵殺黃方塊

Hunt the Yellow Cube

許哲榕

R07544033

授課教師：鄭士康 博士

August 2021

一、題目動機與發想

（一）題目動機

題目的動機是因為近日花費不少時間在觀看老師在台大開放式課程上的講課影片（http://ocw.aca.ntu.edu.tw/ntu-ocw/ocw/cou/106S201/13），尤其是夏季學院中因為時間因素未能涵蓋到的主題，其中單元 13・Unity 遊戲程式設計中的課程內容，特別吸引我，在觀看完老師的教學和 Unity 官方的教學影片後，發現或許可以嘗試自己寫寫看 Unity 的小遊戲，也便開始思考具體應該會有哪些內容涵蓋在一款有趣的小遊戲中，哪些需要模仿老師或官方教學影片中的實作，又有哪些可以自行去設計。

（二）內容發想

首先，私以為一款合格的有趣的小遊戲，一定要有可以主角和敵人互相追逐、獵殺的功能，因為被動地在遊戲中蒐集物品或獎勵，雖然也有趣，但相對少了刺激感；其次，應該要有方便使用的鍵盤快捷鍵可以快速上手，如此一來便能迅速建立起對遊戲使用的第一印象，也快快展開挑戰；第三，必須得有隨機的模式設定，才能夠在每一場遊戲內都感到刺激，因此遊戲中的敵人應該要能夠隨機走動並對主角產生威脅，而非仰賴既定的路線去攻擊玩家或固定在某些位置；四，若有適當的說明文字和清晰的配色，則能準確地讓玩家感受到設計者所欲玩家進行遊戲的方式，且投入遊戲的氛圍中；最後，背景音樂和各種音效的功能必不可少，少了音效，整個遊戲將全然仰賴觀看畫面的視覺，透過五感綜合起來享受遊戲勢必能較好的吸引玩家。綜上，對於自己想要在作業二中開發的獵殺黃方塊 Unity 遊戲程式，應該基本具備這幾項功能及內容。

二、構想解說

（一）詳說構想及功能

以下透過列表之方式，一一說明打算構築的各項程式功能：

1. 遊戲說明的顯示	告訴玩家如何操作主角之移動與攻擊（發射子彈）。
2. 大方塊（下稱敵人）血量條的顯示	透過血量條讓玩家知道還需要大約多少次才能打倒敵人。
3. 操縱主角的功能	需要偵測玩家在方向鍵上按下的時間和次數來反映到遊戲中主角不同方位的移動。
4. 操縱主角發射子彈的功能	需要生成子彈物件讓主角得以隨著玩家按下按鍵發射。
5. 主角吃掉小方塊的功能	當小方塊與主角接觸時，需要記錄下數目，並讓小方塊消失以模擬被吃掉的效果。
6. 子彈（主角）功能敵人的功能	子彈攻擊敵人後，應讓敵人的血量下降，除反映在血量條上，也要記錄下以確認是否滿足勝利條件之一。
7. 敵人攻擊主角的功能	敵人碰觸主角時，需要紀錄，並顯示出遊戲失敗字樣，告知玩家。且內部計算

	扣除既有的吃方塊數，使得勝利條件不得在失敗後再次達致。
8. 敵人靠近主角移動的功能	敵人的移動軌跡需要偵測玩家所操控之主角的現行位置，並逐漸靠近。
9. 吃掉小方塊的音效	透過音效讓玩家知悉成功吃下小方塊。
10. 子彈成功攻擊敵人的音樂	透過音效讓玩家知悉成功攻擊到敵人。
11. 遊戲背景音樂的音效	透過音效形塑遊玩情境的氛圍，同時背景音樂的撥放也是遊戲尚在進行中和遊戲勝負已判定兩種不同狀態的判定方法。
12. 計算敵人血量條的功能	每當子彈成功攻擊敵人，應記錄下數據，其也是勝利條件之一的判定，關乎到是否讓敵人消失（被打倒）。
13. 遊戲失敗的顯示	當被敵人碰觸到時，遊戲即失敗。
14. 遊戲成功的顯示	擊倒敵人並吃下所有小方塊，遊戲即獲勝。

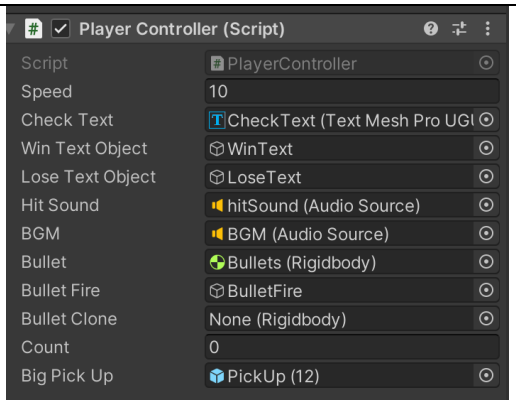
15. 場景中各物件的配置	

三、程式測試規劃

以下同樣透過列表之方式，一一說明打算測試規劃的各點：

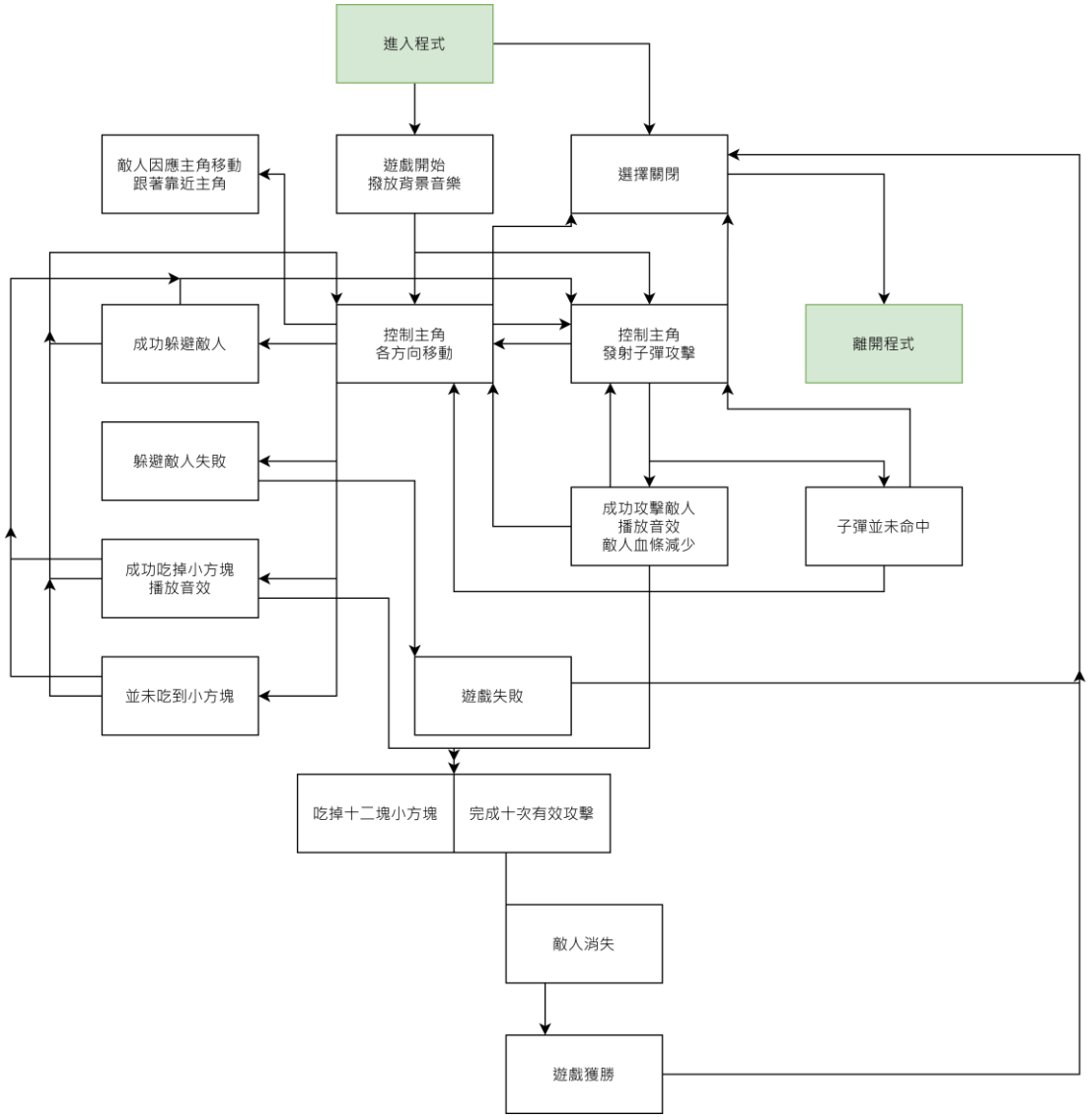
1. 遊戲說明的顯示	注意視線上不要遮掩到主要遊玩的區域。
2. 敵人血量條的顯示	注意是否能正確反映敵人的血量(相對

	應的子彈成功攻擊次數）。
3. 操縱主角的功能	確認玩家可以各方向移動，且不超出遊玩區域。
4. 操縱主角發射子彈的功能	確定子彈是否正確生成、發射。尤其是子彈發射的方向（向量）和動能是否正確得以有效攻擊到敵人。
5. 主角吃掉小方塊的功能	檢查能否讓小方塊被吃後消失，且正確記錄下讓後續勝利條件可以判定。
6. 子彈（主角）功能敵人的功能	確認子彈攻擊到敵人後，能否減少敵人的血量。
7. 敵人攻擊主角的功能	確認主角被敵人碰到後有沒有正確觸發失敗的條件。
8. 敵人靠近主角移動的功能	注意敵人能否正確靠近玩家操控的主角，且小心有沒有移動到遊玩區域外。
9. 吃掉小方塊的音效	注意音效是否正確出現、音量是否適中。
10. 子彈成功攻擊敵人的音樂	注意音效是否正確出現、音量是否適中。
11. 遊戲背景音樂的音效	注意音效是否正確出現、音量是否適中。且在完成勝負條件後有沒有正確反映。
12. 計算敵人血量條的功能	確保血量計算符合攻擊成功之數目。

13. 遊戲失敗的顯示	確保完成失敗條件後，不會再次達致成功的條件。
14. 遊戲成功的顯示	確保完成成功條件後，不會再次達致失敗的條件。
15. 場景中各物件的配置	 以 Player 這個 Game Object 為例，要注意各物件是否正確拖拉所需參考的物件至不同的 Script 所需的參考物件表格，若有缺失會讓 Script 的 C#程式出現錯誤。

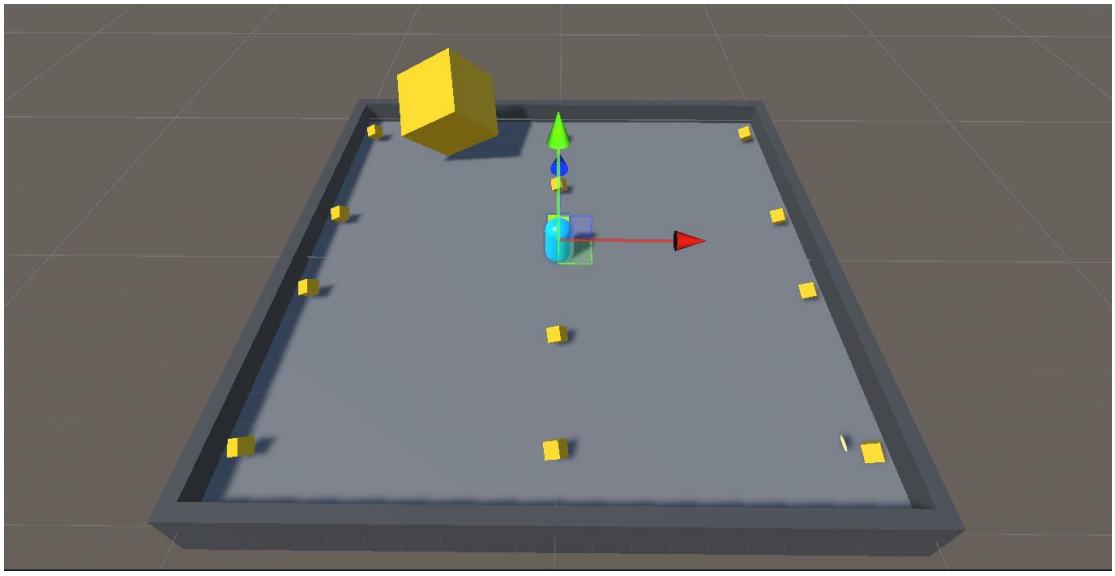
四、流程圖

以下透過圖片之方式，說明獵殺黃方塊程式之流程圖：



五、程式列表

以下主要針對 Scripts 資料夾中四個 Script 程式做說明：PlayerController.cs、BigPickUpMission.cs、CameraController.cs、Rotator.cs。程式之功能以圖片中的綠色註釋字句說明：



```
Unity 指令碼 (1 個資產參考)|0 個參考
public class PlayerController : MonoBehaviour
{
    public float speed = 0; //主角移動初速，Unity中會再設為10
    //上行參考Unity的Roll-a-Ball教學 https://youtu.be/_uO5B7bP9jo
    public TextMeshProUGUI checkText; //遊戲操作說明文字
    public GameObject winTextObject; //遊戲勝利文字
    //上行參考Unity的Roll-a-Ball教學
    public GameObject loseTextObject; //遊戲失敗文字
    public AudioSource hitSound; //吃到方塊的音效(採用無版權音效)
    public AudioSource BGM; //遊戲背景音效(採用無版權音效)
    public Rigidbody bullet; //主角發射的子彈
    public GameObject bulletFire; //主角發射的子彈生成的地方(大約在主角的眼睛處)
    public Rigidbody bulletClone = new Rigidbody(); //主角發射的子彈
    public int count; //吃到幾個方塊
    //上行參考Unity的Roll-a-Ball教學
    public GameObject bigPickUp; //敵人
```

```
private Rigidbody rb; //主角
//上行參考Unity的Roll-a-Ball教學
private float movementX; //預計要讓主角移動的數值
//上行參考Unity的Roll-a-Ball教學
private float movementY; //預計要讓主角移動的數值
//上行參考Unity的Roll-a-Ball教學
private float timer; //設置敵人移動的計時
```

```
// Start is called before the first frame update
Unity Message|0 個參考
void Start()
{
    rb = GetComponent<Rigidbody>(); //主角物件
    //上行參考Unity的Roll-a-Ball教學
    count = 0; //剛開始都沒吃到方塊
    //上行參考Unity的Roll-a-Ball教學
    checkText.ToString(); //顯示遊戲說明文字
    winTextObject.SetActive(false); //遊戲勝利文字隱藏
    //上行參考Unity的Roll-a-Ball教學
    loseTextObject.SetActive(false); //遊戲失敗文字隱藏
    BGM.Play(); //播放背景音樂的音效
}
```

```
// Update is called once per frame
Unity Message|0 個參考
void Update()
{
    SetText(); //每幀呼叫函數確認勝利條件

    if (Input.GetKeyDown(KeyCode.K)) { //按K時主角發射子彈
        bulletClone = Instantiate(bullet, bulletFire.transform.position, bulletFire.transform.rotation); //從Prefabs的子彈去實體化子彈物件
        bulletClone.AddForce(bulletClone.transform.forward.x * 8, bulletClone.transform.forward.y * 6, //配合主角和子彈生成的方向，給子彈一個力去往前飛
            bulletClone.transform.forward.z * 8);

    }
    timer += Time.deltaTime; //敵人移動的計時
    if (timer >= 0.5) { //每半秒一動一次
        BigPickUpMove(); //敵人移動
        timer = 0; //移動完重新再算下一次移動的計時
    }

}
```

```
0 個參考
private void OnMove(InputValue movementValue) //透過上下左右和WSAD按鍵，計算主角的移動
{
    Vector2 movementVector = movementValue.Get<Vector2>(); //設移動的值
    //上行參考Unity的Roll-a-Ball教學
    movementX = movementVector.x; //設移動的值
    //上行參考Unity的Roll-a-Ball教學
    movementY = movementVector.y; //設移動的值
    //上行參考Unity的Roll-a-Ball教學
}

1 個參考
void SetText() //確認勝利條件
{
    if (!bigPickUp.activeSelf && count >= 12) //確定敵人消失以及吃了全部十二塊方塊時
    {
        winTextObject.SetActive(true); //顯示勝利文字
        BGM.Pause(); //暫停背景音樂的音效
    }
}
```

```
Unity Message|0 個參考
private void FixedUpdate()
{
    Vector3 movement = new Vector3(movementX, 0.0f, movementY); //設主角移動的值
    //上行參考Unity的Roll-a-Ball教學
    rb.AddForce(movement * speed); //施力給主角，讓主角移動
    //上行參考Unity的Roll-a-Ball教學


}
1 個參考
private void BigPickUpMove() //計算敵人的移動，會讓敵人逐漸靠近主角當前位置
{
    Vector3 movement = bigPickUp.transform.position; //設移動的值
    Vector3 target = bulletFire.transform.position; //抓取主角(子彈生成處)現在位置
    movement.x = (6 * movement.x + target.x) / 7; //設移動的值
    movement.z = (6 * movement.z + target.z) / 7; //設移動的值
    bigPickUp.transform.position = movement; //設移動的值


}
```

```
Unity Message|0 個參考
private void OnTriggerEnter(Collider other) //主角吃到方塊、或被敵人碰到時的反應
{
    if (other.gameObject.CompareTag("PickUp")) { //確認是不是吃到方塊
        //上行參考Unity的Roll-a-Ball教學
        other.gameObject.SetActive(false); //讓方塊消失
        //上行參考Unity的Roll-a-Ball教學
        count++; //吃方塊的計數加一
        //上行參考Unity的Roll-a-Ball教學
        hitSound.Play(); //播放吃到方塊的音效
    }
    if (other.gameObject.CompareTag("BigPickUp")) //確認是不是被敵人碰到
    {
        loseTextObject.SetActive(true); //碰到後，遊戲失敗文字顯示
        count -= 50; //吃到方塊數減五十，確保之後(失敗文字顯示下)即便繼續打倒敵人，或吃完方塊，也無法滿足勝利條件
        BGM.Pause(); //背景音樂的音效暫停
    }
}
```

```
Unity 指令碼 (1 個資產參考)|0 個參考
public class BigPickUpMission : MonoBehaviour
{
    public int bigPickUpLife=10; //敵人總血量
    public GameObject bigPickUp; //敵人物件
    public AudioSource hitBigPickUp; //打到敵人的音效(採用無版權音效)
    public GameObject blood; //血量條相關
    public RectTransform bloodbar; //血量條相關

    private float bloodCount; //血量條相關

    // Start is called before the first frame update
    Unity Message|0 個參考
    void Start()
    {
        bloodCount = blood.GetComponent<RectTransform>().rect.width; //這邊取值待會要調整敵人的血量條
        bloodbar = blood.GetComponent<RectTransform>(); //這邊取值待會要調整敵人的血量條
    }
```

```
// Update is called once per frame
Unity Message|0 個參考
void Update()
{

}
Unity Message|0 個參考
private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("Bullet")) //敵人被子彈打到時
    {
        bigPickUpLife--; //敵人血量減一
        hitBigPickUp.Play(); //播放打到敵人的音效
        bloodCount -= 30; //減少血量條寬度
        bloodbar.SetSizeWithCurrentAnchors(RectTransform.Axis.Horizontal, bloodCount); //減少血量條寬度
        bloodbar.anchoredPosition3D = new Vector3(0f, 236f, 0f); //重設血量條位置(但在RectTransform下效果不太好)
        if (bigPickUpLife <= 0) { //敵人血量歸零或以下時
            bigPickUp.SetActive(false); //敵人消失
        }

    }

}
```

```
Unity 指令碼 (1 個資產參考)|0 個參考
public class CameraController : MonoBehaviour
{
    public GameObject player;
    private Vector3 offset;

    // Start is called before the first frame update
    Unity Message|0 個參考
    void Start()
    {
        offset = transform.position - player.transform.position;
        //上行參考Unity的Roll-a-Ball教學 https://youtu.be/_uO5B7bP9jo
    }

    // Update is called once per frame
    Unity Message|0 個參考
    void LateUpdate()
    {
        transform.position = player.transform.position + offset; //隨著每幀變化(在其他東西計算後)視角，配合主角移動
        //上行參考Unity的Roll-a-Ball教學
    }
}
```

```
Unity 指令碼 (1 個資產參考)|0 個參考
public class Rotator : MonoBehaviour
{

    // Start is called before the first frame update
    Unity Message|0 個參考
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message|0 個參考
    void Update()
    {
        transform.Rotate(new Vector3(15, 30, 45)*Time.deltaTime); //隨著每幀變化讓黃方塊旋轉
        //上行參考Unity的Roll-a-Ball教學 https://youtu.be/_uO5B7bP9jo
    }
}
```

六、程式測試執行結果

程式運行十分順利，各種音效、操作、與遊戲的勝負都正常顯示。不過在血量條的顯示上因為對 GUI 工具的規範不熟悉，當血量減少時血量條會出現預期外的偏移，但大致上不影響玩家判讀敵人的血量條。當前遊戲各種功能與實作的測試都無出現錯誤，但相對的遊玩內容略顯簡陋，也就難以吸引玩家多次的遊玩和參與。綜合考量上述幾點，未來在期末報告中應改進，著重在遊戲內容的增加，無論是遊戲效果或遊戲場景，並且在使用者介面上做出更多的變化，讓使用者更清晰理解遊戲操作方法，也更加融入遊戲的故事當中。

參考文獻

鄭士康（2019），《計算機程式設計（台大開放式課程）》，

http://ocw.aca.ntu.edu.tw/ntu-ocw/ocw/cou/106S201/13。

Unity (2020), ‘Roll-a-Ball’ tutorial, https://youtu.be/_uO5B7bP9jo.