電腦網路導論 期末專題 第十一組

組員

- B06901089 蔡予謙
- B06901135 陳宣叡
- B06901168 黃其澤

遊戲介紹

EGGNOGG LITE

原作

這款遊戲主要參考來源為 EGGNOGG+ (https://madgarden.itch.io/eggnogg).
是一款拿著劍互相攻擊，阻擋對方前進的遊戲
原本的作品是單機雙人版遊戲，我們這次挑戰將其改成多人連線遊戲

多人版本簡介

我們嘗試去模仿原作的核心，也就是攻擊模式的判斷與相對應之動畫，我們將攻擊分為：
上攻擊、中間攻擊、下攻擊、移動上攻擊、移動中間攻擊、移動下攻擊等。
且與原作相同的是，不論玩家是否進行攻擊的動作，劍都具有殺傷力。
若防禦玩家沒有將劍放在相應的對位(上、中、下)，將被劍刺中而瞬間死亡，並於兩秒後復活。
我們與原作相同，在地圖中擺放刺以及炸彈增加遊戲的困難度，並放置寶箱增加玩家搜索地圖的興趣。且於多人遊戲中，我們引進了分數的機制，去評斷最後的贏家。
我們測試可以**連上15人以上**

玩家設計

動畫

如下圖，我們設了右邊這些parameter，讓我們去分辨現在應該在哪個state，透過PlayerController.cs去針對使用者的輸入，去對於這些parameter賦值，藉此，去判斷player現在的state，並在畫面上呈現相對應的動畫。
跳躍本身是拆成兩個動畫，分為Raising和Falling，跳躍上位置的變化則是靠著角色自身因為重力而有所下落。
死亡方面，我們是做兩秒後復活，我們的做法是透過宣告一個static IEnumerator，讓判定死亡後兩秒後再使其消失，並且於其他位置復活，因為所使用的函數為static，所以不會於死亡這兩秒內一直重複呼叫此函式，得以去避免不斷復活的狀況。

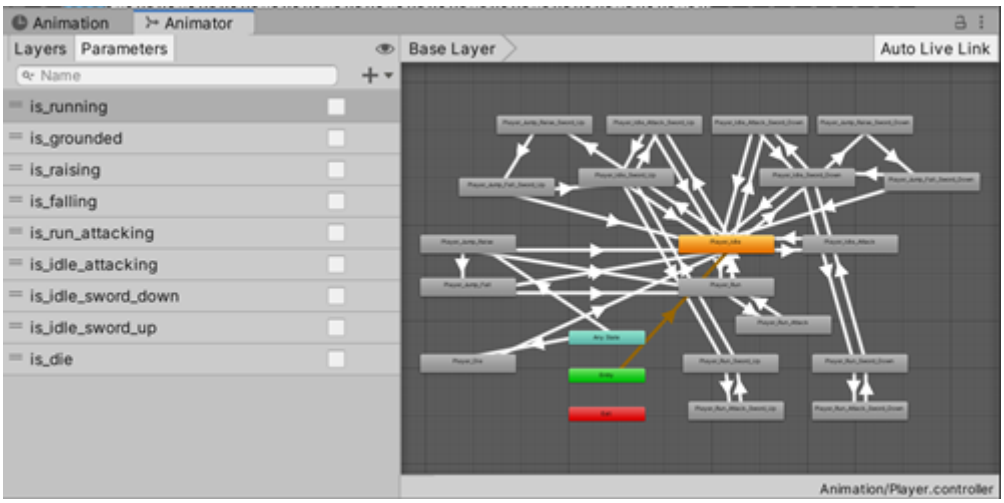


Fig1. state transition graph of our player.

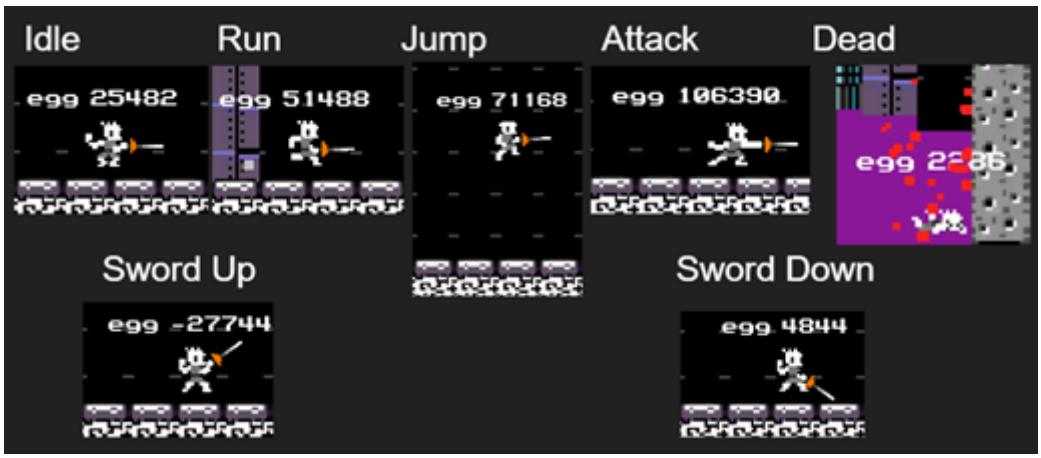
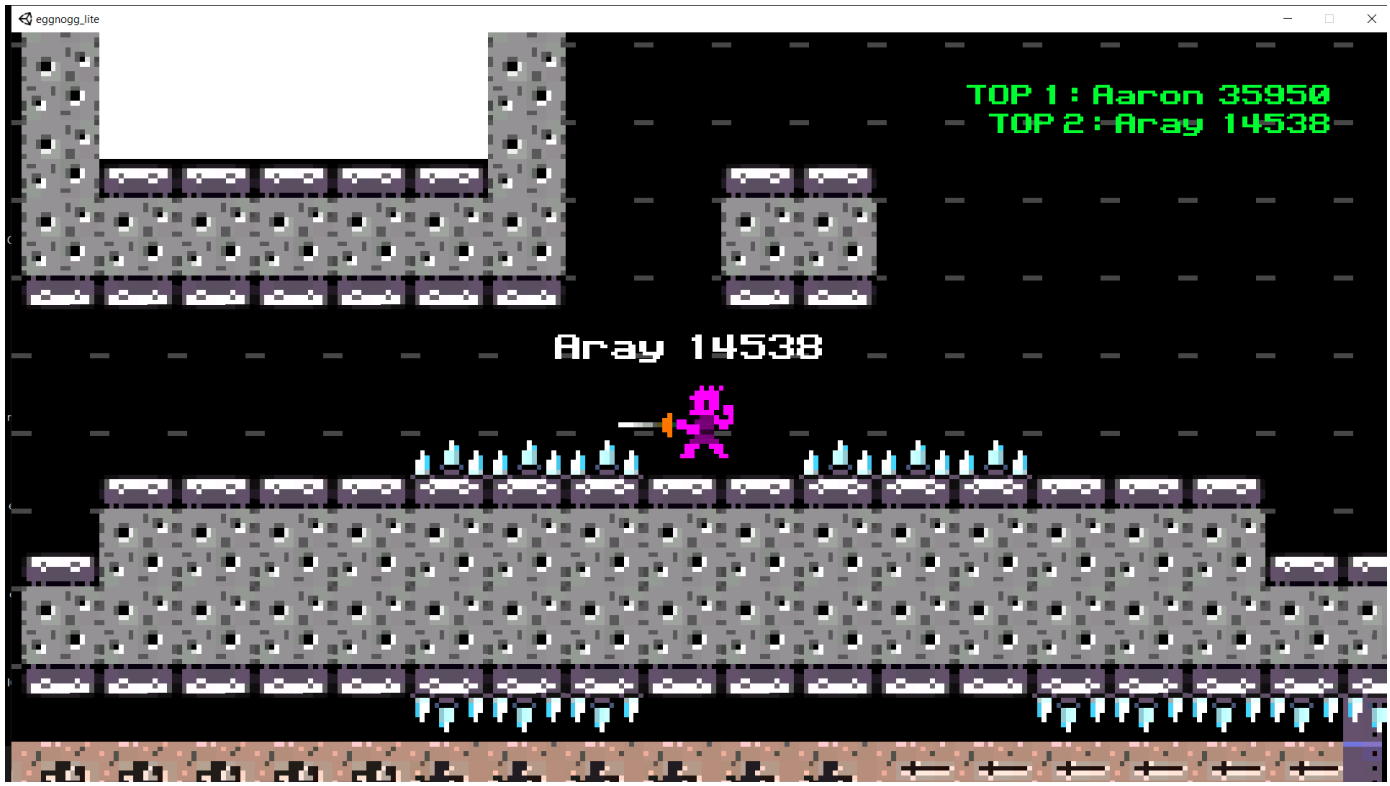


Fig2. animation of our player.

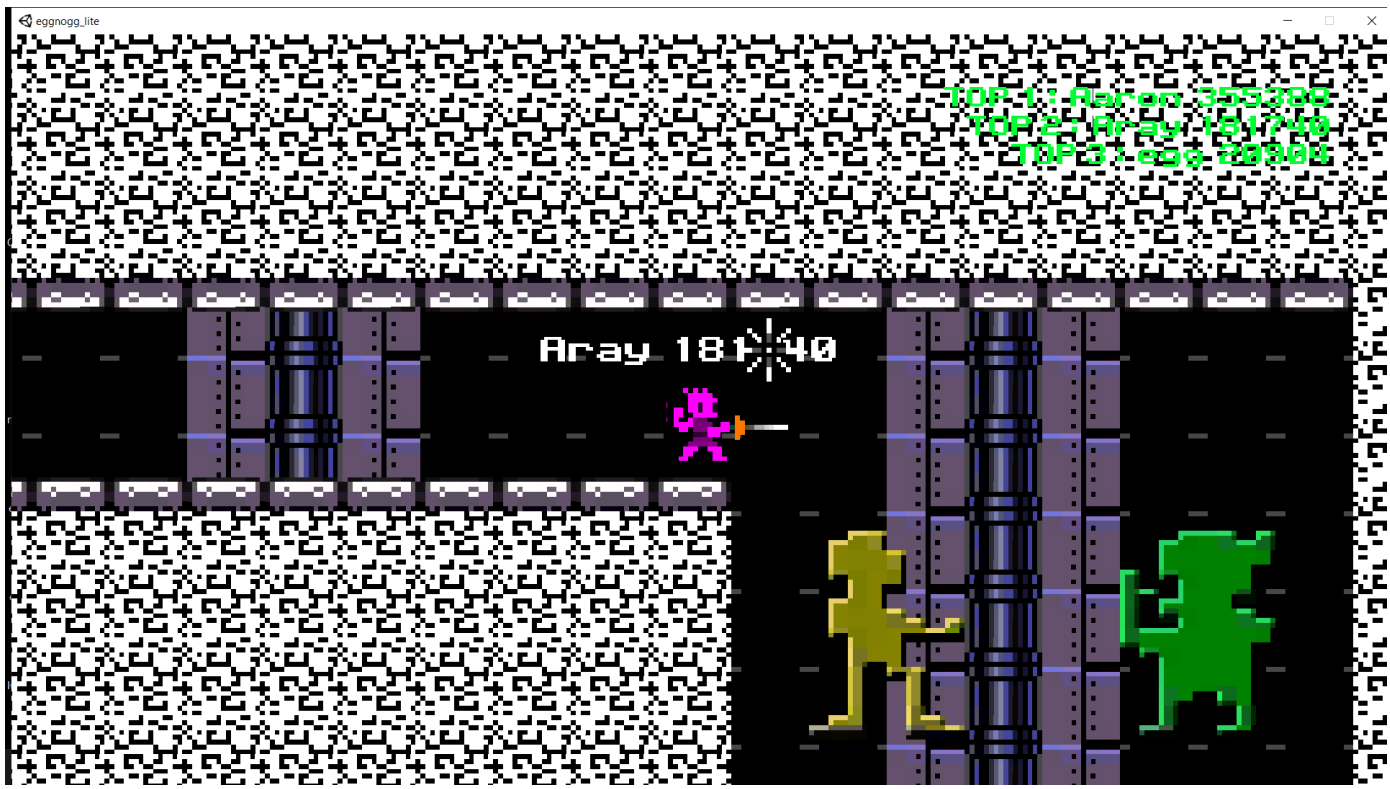
背景設置

- **刺**
  我們於FallerDetector.cs中設定如果刺感應到物體的tag為player，且y方向的速度超過我們所設定的閾值，則代表player跳躍到刺上，則我們判定player死亡。(因為我們的player於

idle時，也有很小的y方向速度，所以我們必須設定閾值，使得不會連走路經過都會觸發)
邊界:我們邊界也有放置detector，只要掉到邊界上也會引發死亡。

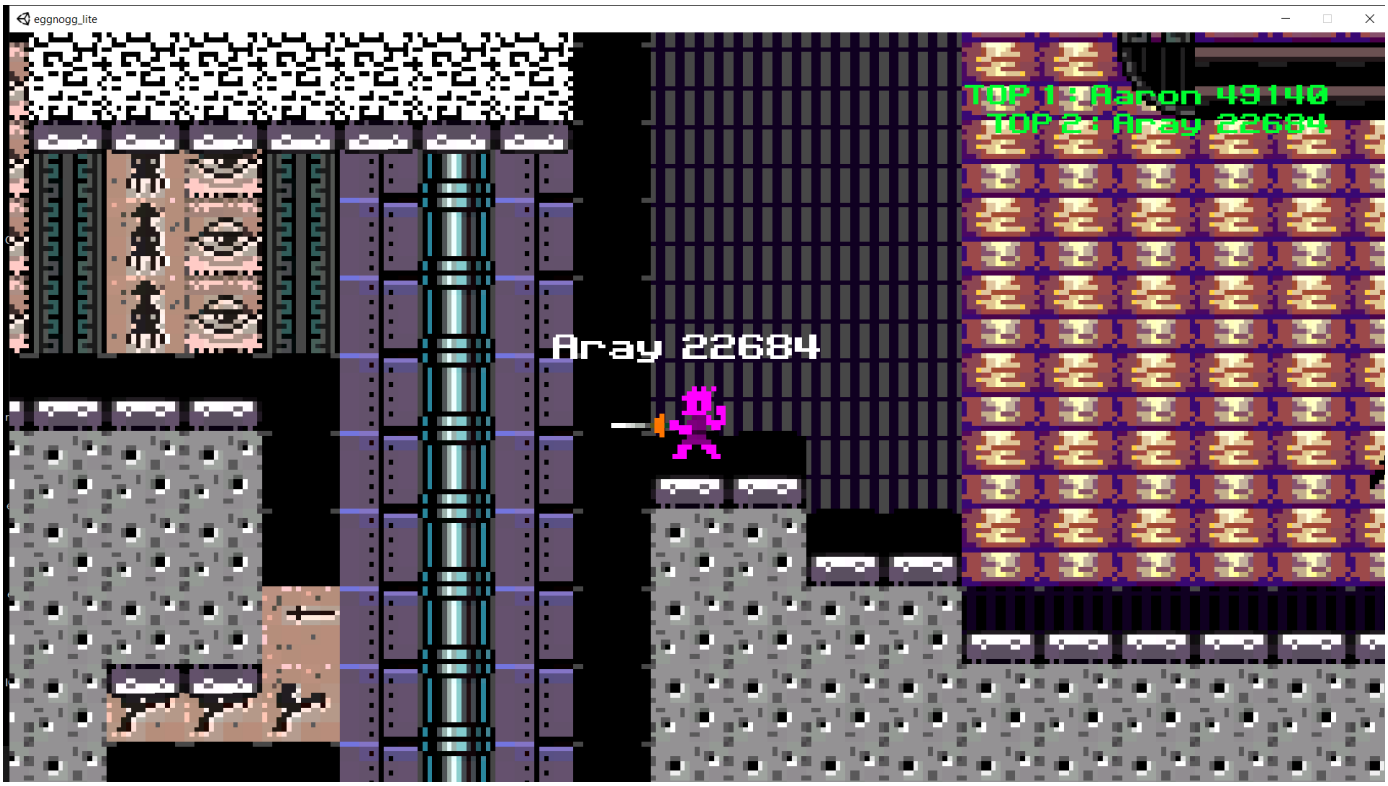


- **炸彈**
  我們於BombDetector.cs中設定如果炸彈感應到物體的tag為player，則代表player碰到炸彈，則我們判定player死亡。

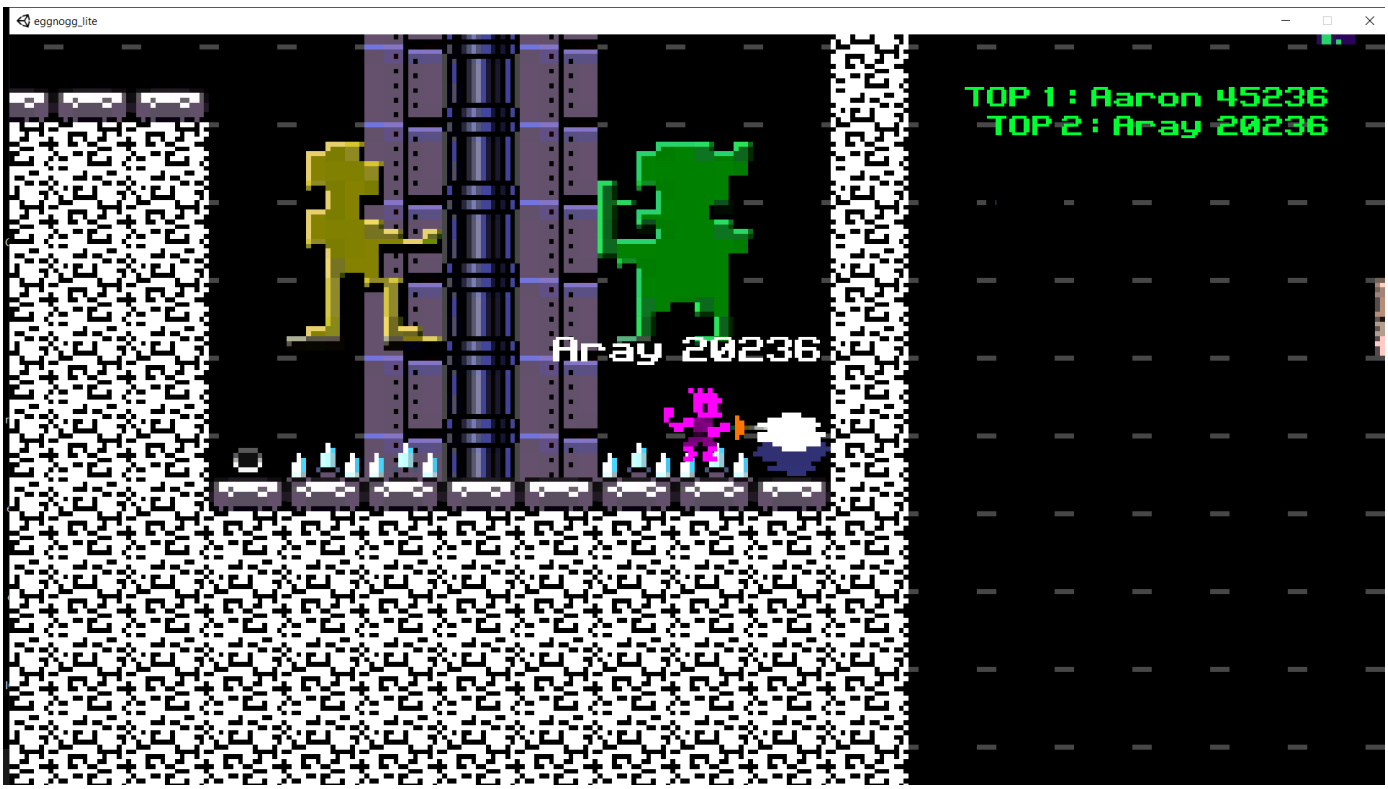


- **記錄點**
  我們於地圖中設置很多CheckPoint，並設定玩家死亡後會於上一個經過的CheckPoint復

活，藉此可以讓玩家不用重頭開始，並且可以改善玩家一開始復活就又被其他玩家殺死的窘境。

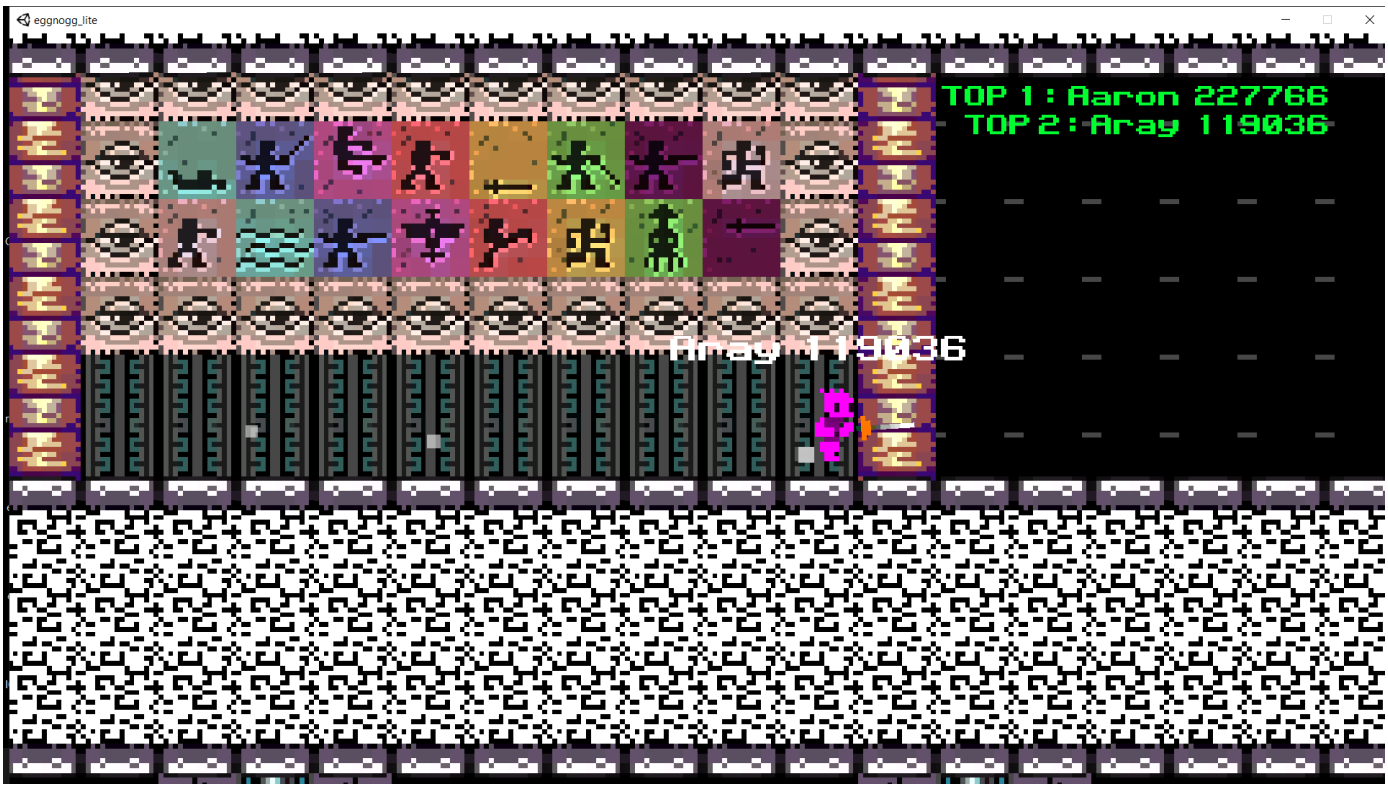


- **寶箱**
  我們於地圖的角落有放置寶箱，一旦玩家觸碰則會加分，寶箱經過冷卻時間後會重回地圖上，使同一位置的寶箱可以重複去觸碰。

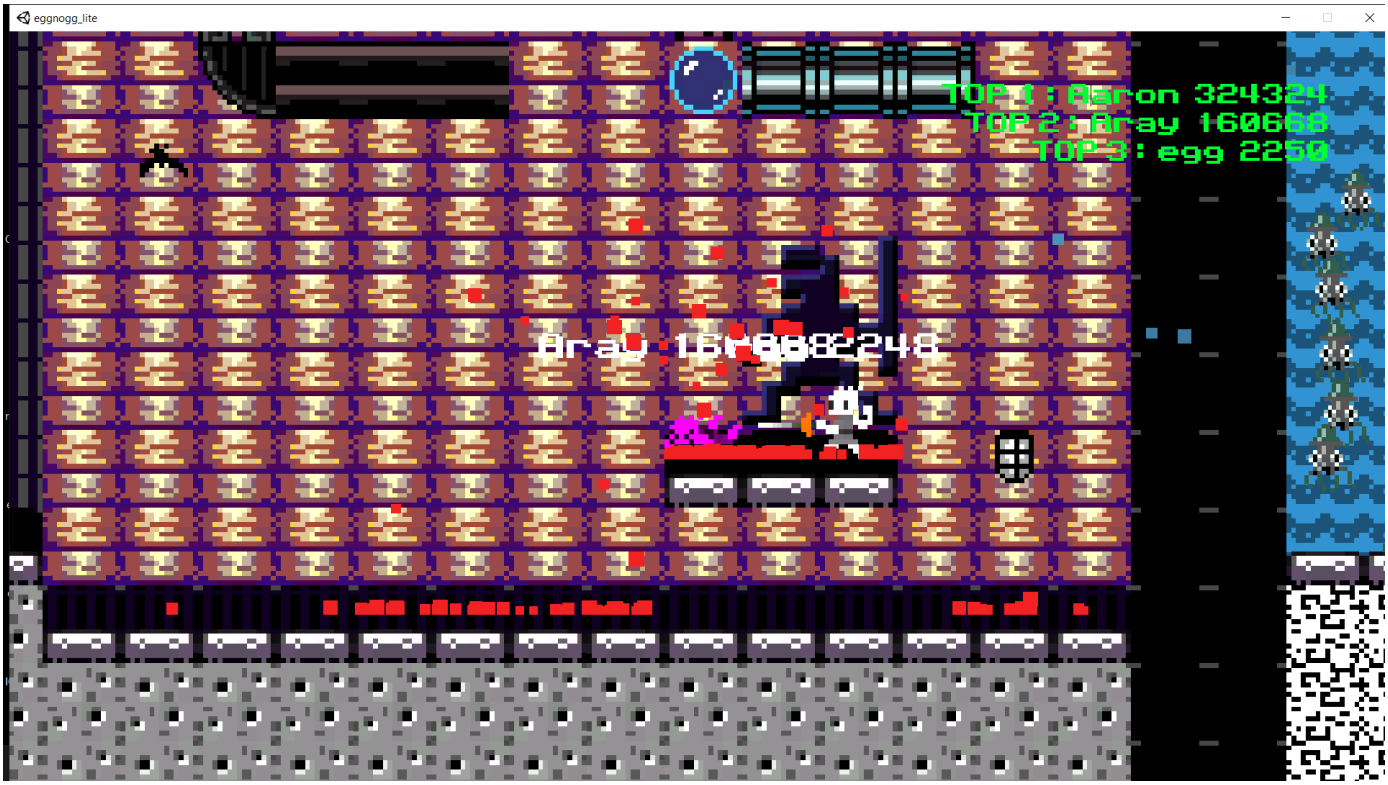


- **瀑布**
  為了增加遊戲的視覺體驗，我們在遊戲中加了許多動畫，例如以下的瀑布利用unity的particle system，做出瀑布的真實感覺。

- **跑步塵土**
  為了增加遊戲的真實感，在玩家跑步時會生成particle，讓跑步的動畫更加逼真。

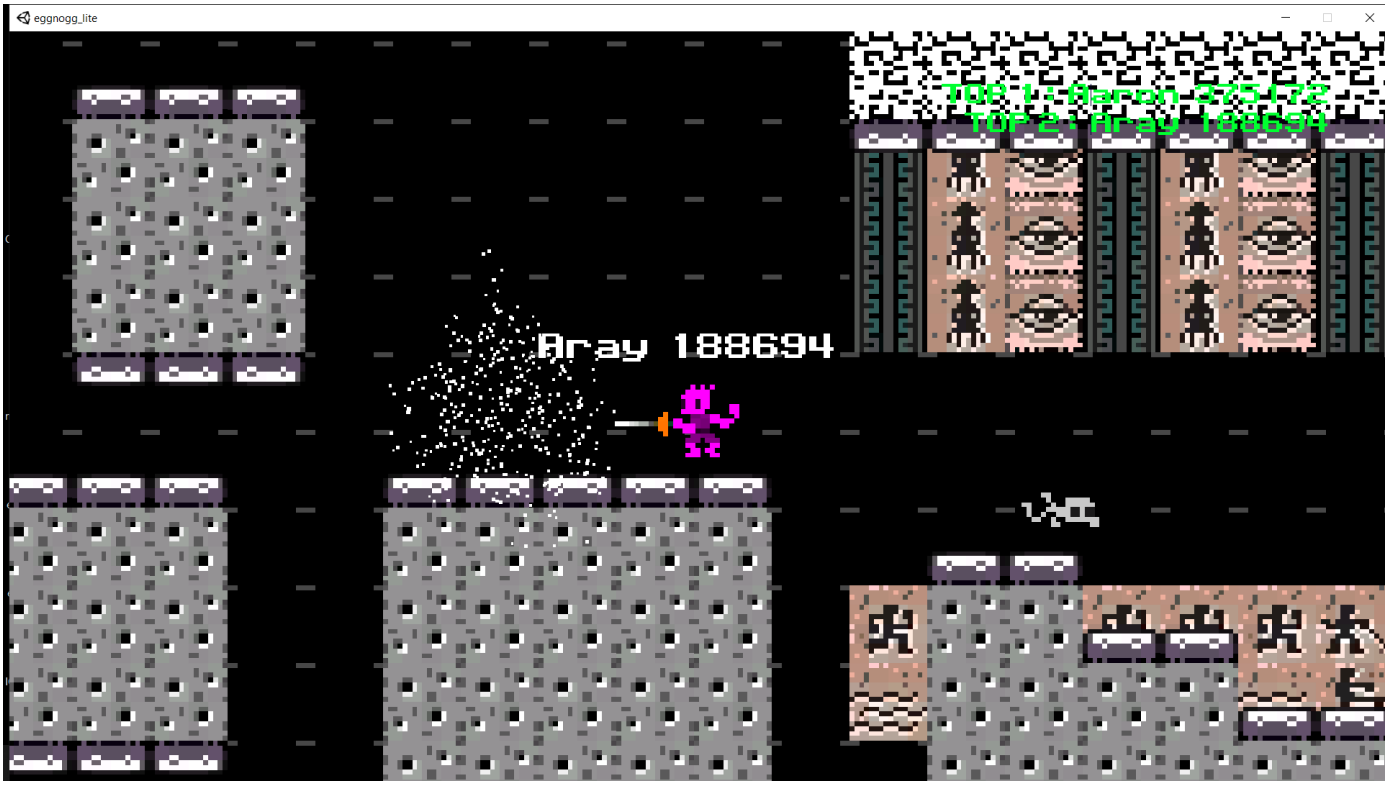


- **死亡噴寫畫面**
  為了增加視覺享受，在自己或其他玩家死亡的時候，會產生大量血跡。

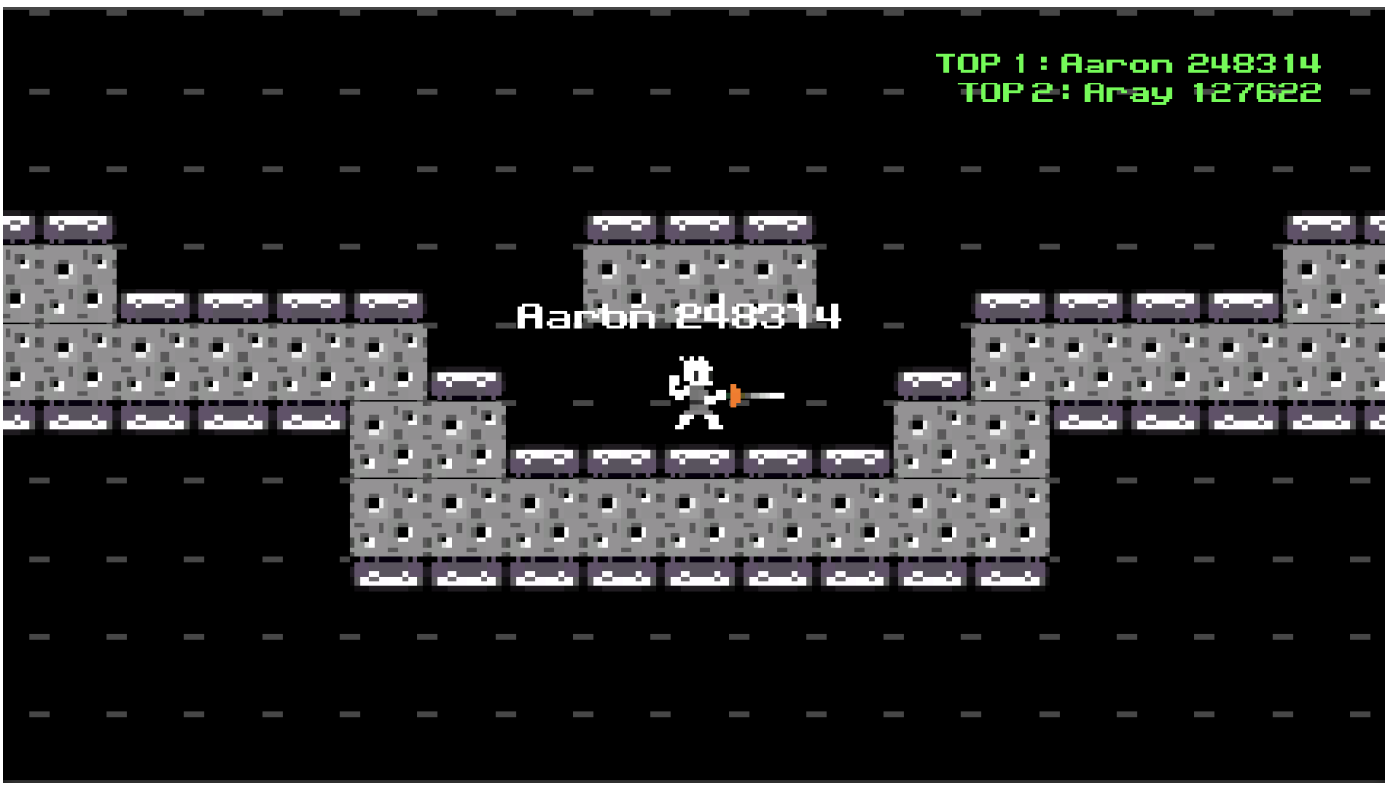


- **玩家屍體**
  為了留下戰鬥的痕跡，玩家在死亡之後會在死亡地點留下一個屍體。

- **玩家斷線消失畫面**
  為了讓其他玩家短線消失不會顯得太突兀，玩家斷線時會變成一團塵土而後消失。



- **彩蛋**
  我們於地圖中也設計了彩蛋的機制，是一些極難抵達的地方或是具分數限制的特效，為遊戲增加驚喜與挑戰。



評分機制

- 活著分數會慢慢增加。
- 死亡時會給予懲罰，扣玩家的分數，死越多次，扣分則越重。
- 觸碰到地圖中的寶藏，則會有相對應的加分，碰越多次，加分則越多。

提升QoS

基本設定

- 為了避免packet loss的情形，在玩家嘗試連線的時候使用tcp進行溝通，當server確認收到玩家後會回傳一個player id，當client收到server回覆後就可以開始進行遊戲。
- 因為這個遊戲需要使用者輸入ip位址，使用者難免會打錯，因此我們貼心的設計了timeout機制，當client一直未收到server回應時，會提醒使用者檢查ip設定。

- 傳送玩家位置以及animation等等資訊時，為了增進效能，我們使用udp進行傳輸

同步所有玩家的位置

基本上的作法和助教提供的sample code相同加上幾點優化。第一，在有新玩家連線的時候，不會只傳一次新玩家的資訊，而是在每次玩家更新的時候檢查此玩家是否已經在遊戲內，可以避免生產玩家時如果發生packet loss，造成server和client不同步的現象。第二點優化是，當一個玩家斷線的時候，會有timeout的機制，在一定的時間內如果沒有重新聯線的話就會踢出遊戲，改善玩家斷線後在遊戲中停止的現象。

```
        if ((_id != Client.instance.id))
        {
            Debug.Log($"UDP Message from server: {_id} and new position ({ _position.x}, {_position.y}, {_position.z})");
            try
            {
                // GameManager.players[_id].transform.position = _position;
                GameManager.players[_id].transform.position = new Vector2(_position.x, _position.y);
                GameManager.players[_id].update_animation(_animation_bools);
                GameManager.players[_id].update_score(_score);
            }
            catch (KeyNotFoundException e)
            {
                // do nothing, since we have not spawn the client
                GameManager.instance.SpawnPlayer(_id, _username, _position, _color);
                Debug.Log($"{e}");
            }
            GameManager.players[_id].idle_timer = 3f;
        }
    }
```

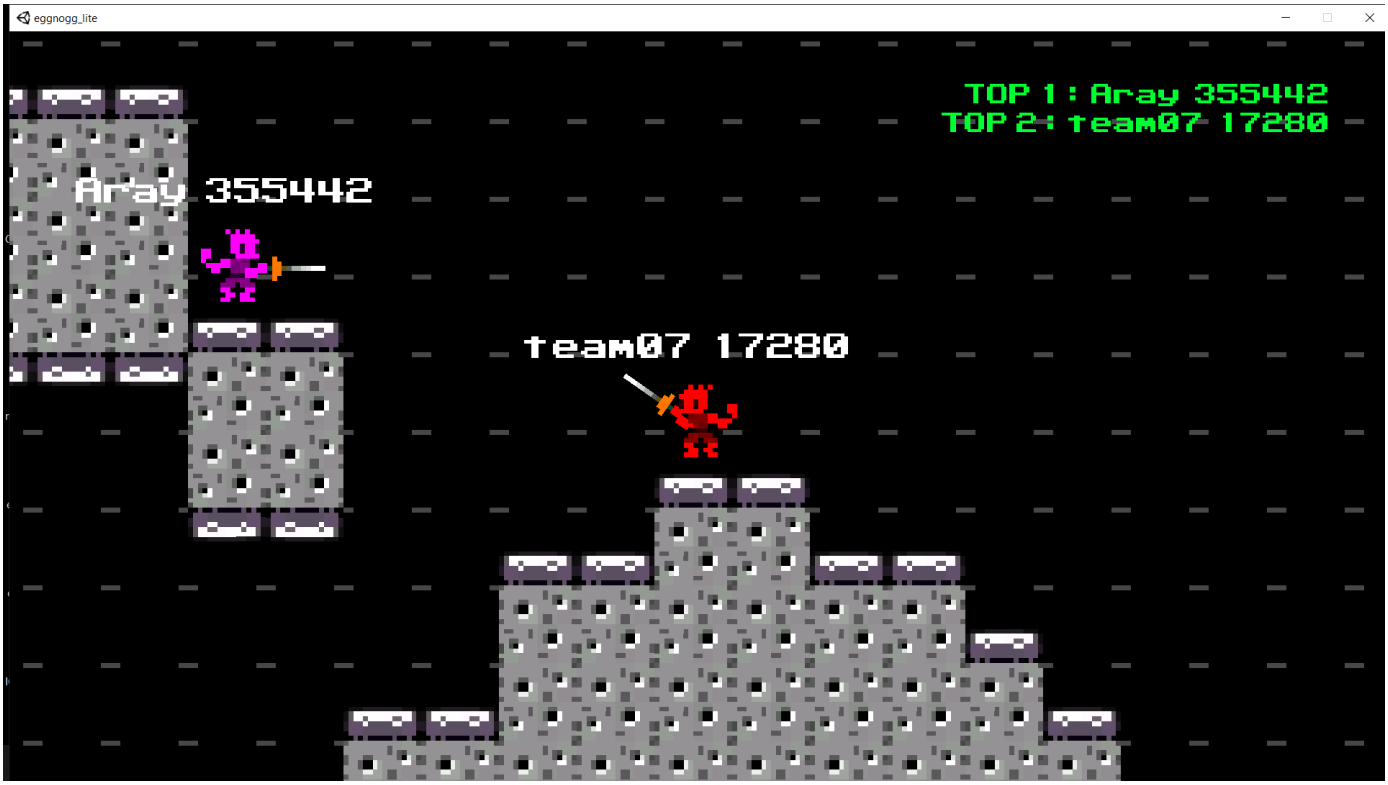
動畫優化

這個遊戲中不只要更新玩家的位置，還需要更新玩家的動畫，以讓遊戲更佳逼真。然而如果直接將每個玩家的動畫直接傳到server同步，資料量太大，遊戲表現不佳，因此我們的做法是將動畫拆解成只用十個bool來表示，在同步的時候只需要同步十個bite的資料，完整的動畫會在client端重建，大幅的改善了遊戲連線的動畫品質。

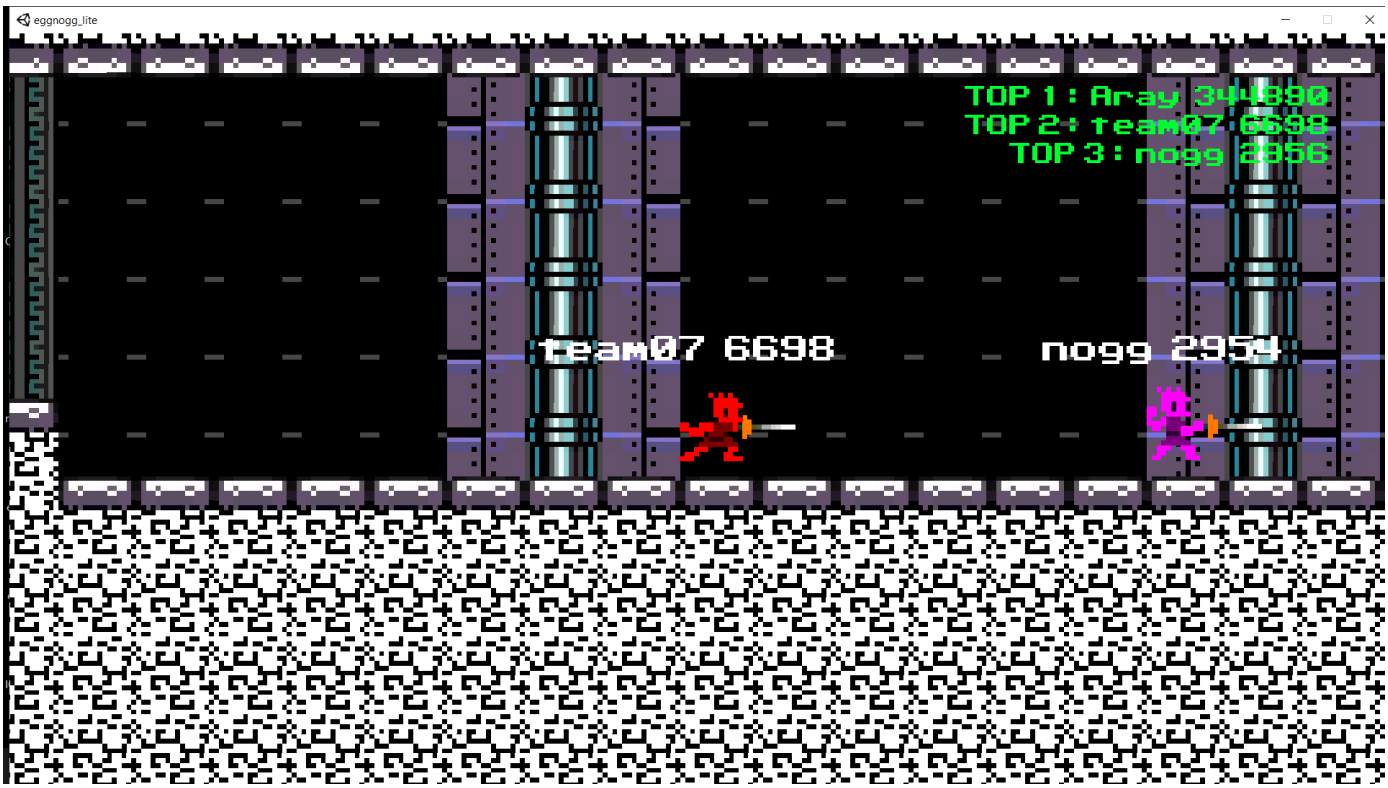
```
    private List<string> animation_keys = new List<string>() {
        "is_running",
        "is_grounded",
        "is_raising",
        "is_falling",
        "is_run_attacking",
        "is_idle_attacking",
        "is_idle_sword_down",
        "is_idle_sword_up",
        "is_die"
    };
```

額外網路相關功能

- 這個遊戲採計分制，計分的邏輯是在client端完成，同步時只會同步分數，以增加遊戲效能，同時各個玩家會同步所有玩家的分數，且在遊戲在遊戲中會即時顯示自己以及每個對手的名字及分數



- 遊戲右上角會顯示即時的排行榜，顯示目前前三名的玩家名稱以及分數，排行榜中的分數會即時同步更新。



遊戲介面

我們的介面可以更改名子、IP、player的顏色、Port，讓使用者更有彈性地去選擇自己想要的角色顏色和名子。
只要輸入完畢，按下Battle就能夠進去與其他玩家一起遊玩。

Fig. interface of our game user

謝謝!