第十六組 Bomber Man

會計四 B05702075 張文毓

會計四 B05702057 李季澄

會計四 B05702117 涂譽寶

農藝一 B08601050 謝佩臻

大氣二 B07605067 林庭暐

壹、 主題介紹

發想為大家耳熟能詳的爆爆王遊戲，並以較為簡化的模式呈現。遊戲主要以對戰模式進行，兩名玩家在開始時各有三條生命、一次最多可放置兩顆炸彈。炸彈強度為上下左右各一格。先用完三條生命者為輸家，而時間結束後以剩餘生命數較多的玩家取得勝利。玩家一以操作 WSAD 移動，以 R 鍵設置炸彈；玩家二則以↑ ↓← →移 動、以 M 鍵設置炸彈。

地圖中會隨機出現三種道具，背包的功能為增加一次最多可放置炸彈數;靴子功能為增加玩家移動速度，且效果可疊加;火藥功能為增加炸彈強度，吃到一次上下左右各加強一格。地圖中牆壁種類有鐵箱及木箱，鐵箱為不可炸破的牆壁;木箱則為可炸破。另外，道具有可能出現在木箱下，需炸破木箱後才出現。

貳、遊戲主程式與狀態轉換

為求遊戲 main function 的整潔，以及整體程式的易讀性、可擴充性等等，我們寫了一個 Game Class 讓主遊戲能夠在此 class 中執行。 以下式主遊戲大致上的運行邏輯:

```
while (Window is open)

{

    Get every event

    Update game (Game logic and application) Draw object

}
```

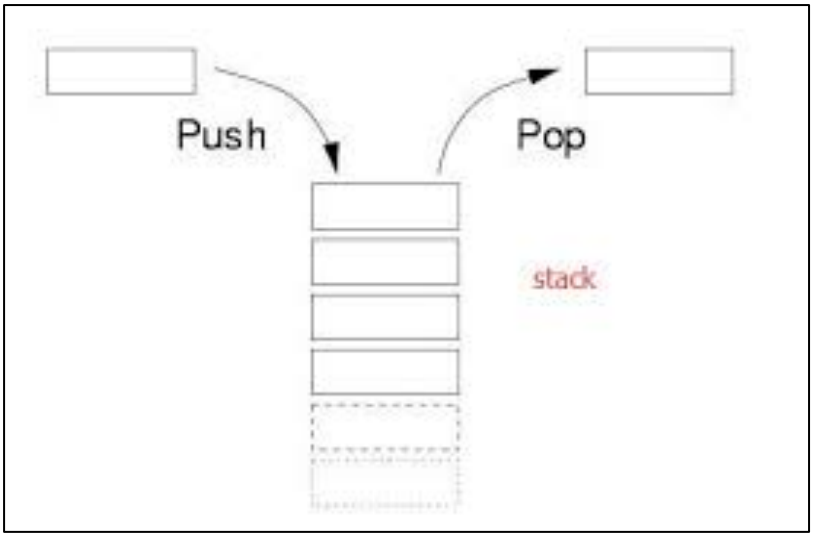
除此之外，為了讓遊戲能夠進行狀態的切換(在此程式碼中共有三個狀態，分別為 MainMenuStatre, GameState 和 EndState)，我們使用了 C++ STL 中 Stack 這個資料結構，利用 Stack LIFO(Last in First out)的特性，將下一個應該被 push 的遊戲狀態放在 stack 的最頂層，不需要時，便 delete&pop（如下圖)。 如此一來便能實現狀態的切換。另外，因為各個狀態本身的性質相似，我們使 用了 OOP 中繼承的概念讓每個不同的 State 能夠去繼承一個名叫 State Class 的各式變數及函式。

參、遊戲 class 類別介紹

遊戲物品 – 炸彈:

炸彈的設計，主要分成動畫及炸彈功能的設計。首先是動畫的部分，一般設計動畫的方式，不外乎製作出每隔動畫的細節 圖並且依照電腦系統預設的 fps 來播放圖片以此達到動畫的效果，然而，與遊 戲角色動畫不同的是，炸彈的動畫會隨著炸彈爆炸而產生更多圖片，使在創造 炸彈動畫時的複雜度增加許多。

由於動畫的圖會在爆炸的瞬間，一次產生來自上下左右各方位的火焰，而此結果勢必會導致記憶體的負荷增加，為了盡可能壓低記憶體的負荷，我們選 擇使用 for loop 讓多出來的火焰圖片成為 for loop 中的 local variable，以此來縮 短記憶體負荷的時間。不過此舉的缺點便是，無法完全地將遊戲中 Update (決 定動畫每張圖應該被畫出的位置) 與 Draw 兩個概念完全分開(因為是宣告區域 變數 ，必須在變數壽命結束之前將其 Draw 出)。以下是炸彈動畫的演算法。




```
void ExplodeAnime()

{

    for (fire range)

    {

        set fire texture depends on the direction

        decide position

        draw()

    }

    void Update()
```

```
    {
        If (Explode)
        {
            ExplodeAnimeUP
            ExplodeAnimeDown
            ExplodeAnimeLeft
            ExplodeAnimeRight
        }
        else
            BombAnime
    }
}
```

接下來則是炸彈效果的演算法，也就是炸彈如何炸毀地圖上的物品及角色。在這裡主要用到的想法是 Check Collision 的方式。 藉由比較兩個物品之 間中心點的 x,y 軸的距離和個別的 0.5 倍長、寬 來判斷其是否重疊，如果重疊 再做是否炸毀的判斷。

而在這比較複雜的部份是，我們必須先去對炸彈的火焰範圍做判斷(因為會被地圖上木箱或鐵箱擋住導致火焰不能延伸)才能做角色是否會被炸死的判斷。在這裡我們的做法是，在製作火焰動畫時，同步去做與地圖上木箱與鐵箱的 Collision 判斷並回傳火焰實際的範圍，以此範圍判斷人物是否與火焰重疊。

以下是演算法 void ExplodeAnime():

```
void ExplodeAnime()

{

    for (fire range)

    {

        set fire texture depends on the direction

        decide position

        if collide with block save the fire range and break draw()
```

```
    }

}

void CheckDestroyPlayer

{

    If (player collide with fire)

        destroy player

}
```

遊戲物品 – 地圖與牆壁:

由於我們需要讓角色和炸彈可以與牆壁做碰撞測試，若僅以 SFML 內建的 TileMap 載入地圖，地圖就只會像背景圖片一樣，角色因而無法對此作碰撞測試。為了解決這個問題，建立一個名為 Wall 的 class，讓牆壁像 minecraft 的磚 塊一樣，一塊一塊的建立在地圖上。

首先，先將螢幕畫面鋪滿背景的草地，再將畫好的地圖轉成文字檔以一維 陣列的形式儲存在 TileMap 的 private variable 中。以 LoadWalls 讀取一維陣列 中的位置資訊，計算該片牆壁在地圖上應該出現的座標，儲存在 breakableWalls 和 unbreakableWalls 陣列中。

```
void LoadWalls()

{

    width = the numbers of tiles needed according to the window width height = the numbers of tiles needed according to the window height

    For (i from 0 to width)

    {

        x_axis = 16. // the x_axis of the first column

        x_axis += 32 * i

        For (j from 0 to height)

        {

            y_axis = 16 // the y_axis of the first row
```

```
                y_axis += 32 * i

            }

        }

        If (map[ i + (j * width) ] is equal to the selected tile)

            Append the tile to the wall array according to its xy coordinate

}
```

在遊戲每回 Update 中，角色會與畫面中的每一片牆做碰撞測試。而炸彈爆炸與牆壁的碰撞測試方式如上半部的炸彈部分說明，breakable 牆壁在確定與炸彈碰撞後，會變為 broken 的狀態，因此在 update 時就會被略過不被畫在螢幕上。

遊戲物品－計時器與計分版:

建立一個名為 GameClock 的 class，以倒數計時的方式計算遊戲時間。由於程式在每秒鐘會進行數次 update，因此需要建立一個 flag 來紀錄秒數變化，

當 flag 的秒數變化時，才將計時器的總秒數減一。將計時器的秒數除以 60 得到商 數與餘數，作為分鐘與秒數轉乘 sf::text 呈現在螢幕畫面中。玩家的生命條以愛心的畫面呈現在計分版上，每次遊戲 Update 時，會根據 玩家剩餘的生命決定要畫幾顆愛心在計分版上。

遊戲物品－道具:

道具的部分一共分為鞋子、背包及火藥三種，鞋子可以增加速度、背包能 增加炸彈放置數量，火藥則能擴大爆炸的範圍。程式部分，我們寫了一個道具 專用的 class，並在 class 中寫了一個函數，來改變 player 中的 private variable，藉此改變個角色的能力。

另外，為了讓遊戲的變異性提高，每場比賽中，道具位置的不同也是必須的。因此我們排除了所有鐵箱的位置座標，並在每場遊戲開始前產生 12 組亂數 座標，藉此提高遊戲的趣味性。

遊戲物品 – 音樂:

音樂的部分分為遊戲背景音樂以及遊戲音效兩種，皆透過 SFML 內部的套件完成。背景音樂的部分分為主選單音樂以及主遊戲背景音樂。音效部分則有主選單按鈕滑動聲、點擊聲，以及主遊戲中獲取道具與炸彈爆炸之聲效。

背景音樂部分，需在開啟遊戲時播放主選單音樂，並在點擊進入遊戲後，停止播放主選單音樂、開始播放主遊戲音樂，最後在遊戲結束時，停止播放主遊戲音樂即可。

而音效部分，需要注意的是，由於主要程式由一個 while 迴圈組成，若不另外設定多個條件，便會造成音效在短時間內連續播放的情形。舉炸彈爆炸聲 效為例，若我們只設定炸彈爆炸時播放音效，音效便會被連續播放好幾十次。這是因為，雖然爆炸的時間不長，但在不斷循環的迴圈中，程式不斷判斷炸彈 是否處於爆炸狀態，因此，音效也不斷地被播放。

另外，因為此遊戲有許多音樂會同時播放，也需要許多剪輯工作，因此我們使用 logic pro 同時處理所有音源，並調整整體遊戲音樂的 balance。

肆、心得

組員 A:

這次作業讓我對於 Class 的應用有了更深刻的了解，並深刻的學習到程式模組化與溝通的重要性。在每次討論中，除了跟大家更新自己的工作進度外，也需要跟組員溝通自己需要對方 Class 中配合的項目，才能有效率地讓工作順 利進行。對於呈現的結果當然還有許多可以更精進的部分，但就整體而言，我 認為我們的成果算是相當的完整。不管未來是否還有機會接觸類似的內容，我 相信這次的經驗已經讓我學到不少可以應用於未來的能力。

組員 B:

這次的報告讓我對 C++物件導向的概念有了更深的一層了解，也深深的明白要寫好 OOP 真的是需要花上非常多的時間，在物件、函式的設計上，都需要做到非常周到的思考，才不會在未來不管是與組員、同事合作上出現效率不 彰的問題。這次的團體合作也讓我了解到，程式設計上的合作比一般報告要來 得更難，必須在事前做到非常全面的溝通，不管是在函數設計、變數設計等 等。雖然這次的結果，只是一個非常陽春、功能不算非常完整的遊戲，

但是仍然讓我學到很多也從中得到很多成就感，也希望自己在未來可以寫出一個更成熟的程式。

組員 C:

這次的專案作業讓我更進一步了解 class 的運用，也因為選擇 sfml 這個函式庫而有非常多自己研究、找影片來看的學習經驗，除此之外，也學習到程式設計的合作真的非常不容易，除了需要先建構出完整、良好的基本架構，也要與組員有良好的溝通，讓自己寫的部分可以讓人輕易了解使用，或是了解組員寫的項目是如何運作才能讓他為自己所用。很感謝這次的經驗，雖然只是一個簡單的遊戲，卻讓我學習到很多東西，也很感謝大家的努力才能做出這樣的結果，也期待自己未來能做出更完整、有趣的程式。

組員 D:

這次專案我們為了做出遊戲而學習了 SFML，還有很多上課沒有教的東西，讓我找資料、自主學習的能力提升了些，雖然覺得很困難，但希望之後能慢慢上手。也讓我體會到平常帶給人快樂的遊戲需要耗費大量的時間精力。

組員 E:

雖然去年有修過商管程式設計，不過這是我第一次寫遊戲。起初，要寫出像爆爆王這樣的遊戲對我們來說是很難想像的。不過，幸好我們在 Youtube 上找到了一個十分完整的教學影片，跟著那位 Youtuber 一步一步做後，我們也漸漸地熟悉了 SFML 這個套件。另外，過去在學習 python 時我對 class 的概念是 很不熟悉的，但在這次與同學分工的過程中，我們一開始就必須把要寫的 class 及 variable 講好，這樣各自寫完後，才能順利地將程式組裝在一起。

總結來說，我十分慶幸有修習這門課程，並與同學合力完成了這份專案。 除了對於我的自學能力有幫助之外，也讓我對於 class 的概念有了更深一層的了解。日後我也會繼續努力鑽研程式設計，並與同學合作開發更多專案。