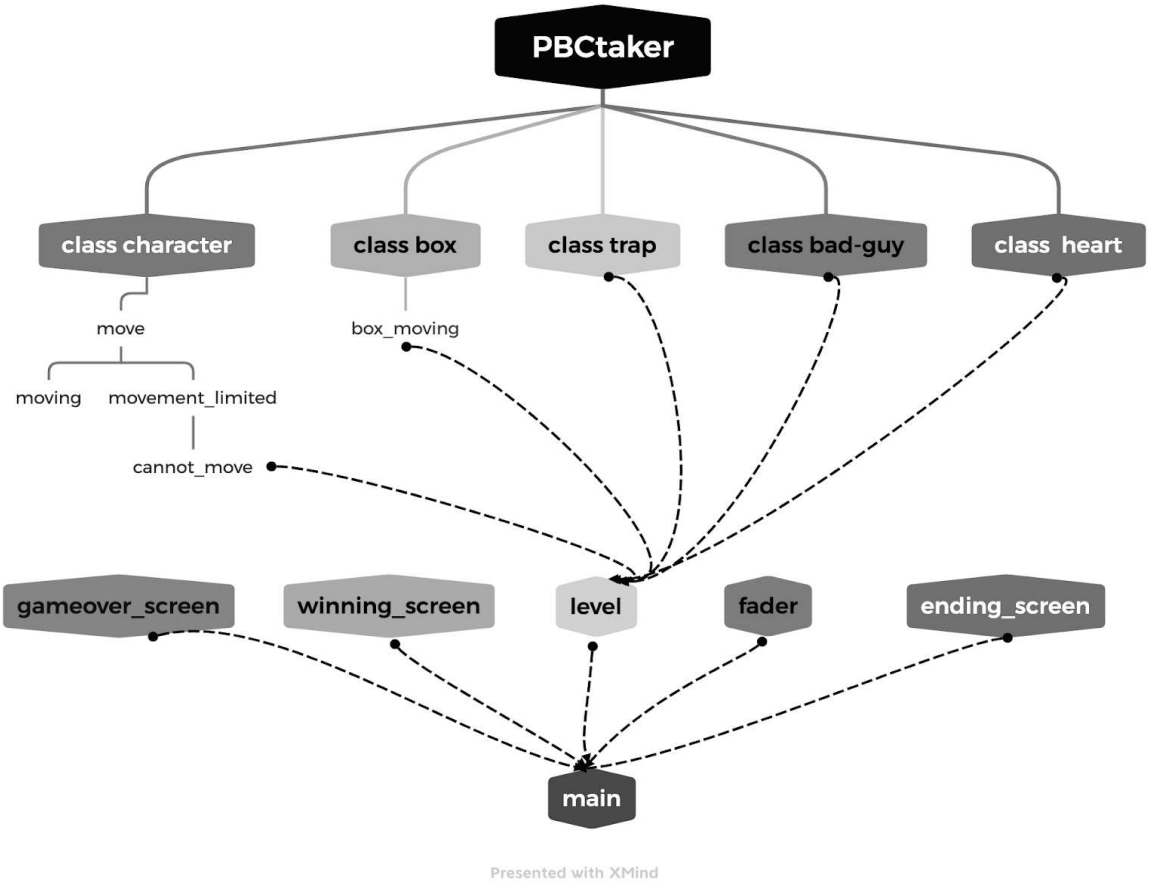
商管程式設計 110-2 期末專案第六組書面報告

- 組長：黃少凱B10702053
- 組別：6
- 組員：林柏諭B10610010
  郭柏亨B10702106
  孫嘉蔚B10702097
  蔡丞中B10702120

壹、主題 —— PBC taker

本次專題主題名為PBCtaker，參考steam上遊戲helltaker，並加入python相關知識。該程式設計目的為檢視同學學習python的最終成果，以確保每個同學對於python的基礎知識有足夠的了解。其內容一共有七關小遊戲，每個關卡要以限定的步數，從起點走到終點，其中會有許多小考驗，有時會需要推開障礙物；有時會需要毆打敵人；有時路上甚至會有陷阱妨礙玩家到達終點。而當玩家抵達每關的終點時，關主會問玩家一個關於python的問題，例如：請問17%4 =?，等到七關結束後，依照玩家的能力與python精熟度給予最終成績。下圖為整個程式的大致架構：



貳、系統設計與演算法

本次專題引用pygame模組，其中主要架構為先設立以下函數：圖片導入、鍵盤滑鼠輸入、角色與障礙物移動、血量計算、轉場畫面、開啟與暫停畫面等前置作業，再將函數寫進關卡函數中，最後再將關卡函數帶入主循環。其中，重要函式如下：

一、移動與血量相關函數

移動函式分為角色移動與箱子移動，其中角色移動最為複雜，必須考慮到什麼時候能移動，什麼時候被卡住沒辦法移動，詳細如下：

- movement limited 函數

由於pygame的視窗是設定好特定禎數後重複跑迴圈所建立出的視窗效果，因此在按鈕被瞬間按下之後如果想做連續的、帶有動畫效果的移動，不能直接在移動函數內將角色座標瞬間改變要移動的距離，而是該設立相關的全域變數，將「偵測是否要移動的函數」以及「正在移動的過程函數」分開。其中，movement limited 函數旨在完成前者。

欲判斷角色是否該移動，主要分成三個條件：按下方向鍵、沒有移動到不該移動的位置，以及本次移動跟前次時間間隔大於一秒。其中，不該移動到的位置，在下面的cannot move 函數中會做清晰的判斷。至於為何兩次移動時間需間隔一秒以上，這是因為想營造出「讓玩家謹慎思考每次移動」的氛圍，然而，也有組員反映一秒與我們參考的遊戲——Helltaker相比實在過長，能夠調成0.5秒的話會更加適合。

在以上條件皆滿足時，四個全域變數——rr、ll、uu、dd中的其中一個將由FALSE變成TRUE，從而完成「是否讓角色移動」的判斷。另一個叫做target_loc 的變數，是在幫助下面函數導引角色移動的目標；而pressed 這個列表儲存當下時間，也是在幫助下面函數釐清從方向鍵被按下後經過多久。

- moving 函數

moving 函數即為「正在移動的過程函數」主體。函數偵測rr、ll、uu、dd四個全域變數，有其中一個經由上個函數變成TRUE之後，計算在特定時間點角色應該移動幾個單位。

由於最大圈的while迴圈每秒大概執行60次，dis這個變數用來算出在這次迴圈的當下離按鈕被按下究竟有多少毫秒。另外一個叫a 的變數，則將dis除以16，因為函數執行一次約花費16毫秒(1000//60)，將單位由秒換成次數，這是我們在計算pygame程式內時間單位的小巧思。後來發現可以利用acc_fps每經過一次迴圈就加一的概念，直接得出次數，因此這個巧思也變的無用武之地。

我們想讓角色移動的速度先慢到快再慢、最後停止，以此創造動畫效果，因此想利用將a這個變數帶入二次函數算出距離的概念。由於我們設定每按一次方向鍵角色移動一格，每格50單位，要如何讓角色總移動量為50呢？這個問題，等同於一個經過原點以及 (16, 0) 的二次函數，與x軸間的距離為50。經過簡單的積分計算，我們得出(-(300)/(30**3)*a*(a - 30))這個函數最為適合。因此當rr、ll、uu、dd四個全域變數的燈號亮起，就將角色座標加或減掉a代入的距離。

然而經由這個函數代出的總距離會是50加上數個小數點，為了在移動角色完後微調一下位置，我們假設由a所代出的距離不在是正數，在a會後退之前的那一刻，也就是總距離的最大值，將rr、ll、uu、dd四個全域變數重新改為FALSE，並且將角色的座標轉成target_loc所提示的位置後，將target_loc歸零。以此完成moving的過程。

- cannot move 函數

本函數分成四個方向，分別輔助movement limited 函數判斷是否有不能移動的狀況。以cannot move left 為例，角色如果左邊是邊界或者牆壁，則角色不能向左移動。稍微複雜點的情況，還有角色左邊有個箱子，更左邊有另一個箱子，或者更左邊是牆壁、陷阱、壞人或邊界，因為角色不能把箱子推到重疊、也不能讓箱子與牆壁、陷阱、壞人或邊界重疊。因此我們設立了雙層for 迴圈來確認是否角色的x座標減去50單位是箱子，而角色減去100單位則是上述這些東西。然而稍加思考過後，我們決定容許角色把箱子推到跟女生重疊，理由是因為這是某個關卡地圖的唯一通關方法。本函數經過判斷之後，回傳TRUE或FALSE回去給movement limited 函數。

- box_moving 函數

箱子移動函數則是當箱子受到角色撞擊後，箱子所相對應的移動方向，必須考慮角色從何方撞擊、是否有空間可以供箱子移動，以及箱子是否移動到正確的位置上等多種情況。詳細如下：

當函數偵測到了rr、ll、uu、dd四個全域變數中，有其中一個變成TRUE之後，函數會判定這時箱子是否就在角色要移動到的下一個位置，若是是，則箱子會往相同方向做50的位移，但是箱子因為隨著角色移動的關係，加起來的總移動量未必剛好會是50，可能是50.1/50.幾，箱子最後的座標就會變成125.01之類的浮點數，因此為了修正，我們另外用125.01//25再*25等方法讓它變為整數，最後箱子才會移動到正確的位置。

- traps 類別以及bad_guys 類別

在本專案程式碼前面所規範的類別裡面，wall、trap、bad_guy 以及box 都有額外新增加上s的類別。這些加上s的類別主要透過locations的attribute，儲存含有wall、trap、bad_guy 以及box 單個物件的列表。透過類別的形式存為列表，還有一個好處：譬如traps 以及bad_guys類別，只要locations列表裡面有任何trap或者bad_guys的物件跟角色的座標重疊，不管是哪一個，都要讓步數扣一。因此在traps類別裡面的trapped函數，只要有角色跟任何一個trap的座標相同，即傳回TRUE；bad_guys類別

裡的update_status函數亦然。這兩個函數所傳回的bool值，都會被下面提到的heart類別引用。

值得一提的是，在bad_guys類別裡面新增一個status的attribute，用來紀錄locations裡面第i個bad_guy物件，在本次迴圈以及上次迴圈所回傳的、「是否與角色座標重疊」的bool值。為何要知道每個bad_guy是否有與角色座標重疊呢？因為bad_guys類別的fading函數當中，若該bad_guy物件有與角色座標重疊，便用調整透明度的方式，讓該bad_guy物件消失在地圖上。因此不同於traps類別只要知道「是否有其中一個trap物件與角色座標重疊」，bad_guys類別需要知道「每個bad_guy物件是否與角色座標重疊」。

- heart 類別

heart 類別所代表的，是角色每一關所被規定的布數的字體，這個類別等下會被一個名為HEALTH變數給引用。在heart 類別的建構式當中我們規範兩個attribute——number 跟 font，前者儲存角色剩餘的步數，後者則負責將前者變成一個白色、40單位大小的字體。稍後在類別裡的draw 函數當中，會及時把number的數字變成font，然後在WIN這個surface上面畫上去，以確保數字的更動被即時更新。

至於如何偵測角色步數的減少呢？會使角色步數減少的條件主要包括角色移動、採到陷阱或者與bad guy 接觸。為了在角色觸發這些事件的瞬間將number扣一，我們在deducting函數中規範一個叫做refer 的字典。這個字典每一個循環會更新上次循環的四個全域變數——ll、rr、uu、dd的bool值，以及透過引用Traps 類別的get_trapped函數傳回的、「是否踩到其中一個陷阱」的bool值，還有透過Bad Guys類別的update_status 及fading 函數更新「是否遇到一個壞人」的bool值。當for迴圈偵測到refer裡面，是否正在移動角色、是否踩到其中一個陷阱或者是否遇到一個壞人的bool值由FALSE轉為TRUE，deducting函數就會傳回TRUE，反之將傳回FALSE。在後方的heart_deducting函數中，當deducting 函數傳回TRUE，便將number扣一。

事後看到使用refer字典來反映角色瞬間狀態的寫法，有點像是數學裡面的「求斜率」，雖然僅能了解每個while迴圈地當下狀態，但是將本次與前次迴圈的當下狀態比對，便能透過角色的變化率偵測到是否有需扣分的情況。然而refer當中加入ll、rr、uu、dd四個key的寫法稍嫌繁瑣，未來也許能夠在character類別裡面的movement_limited函數，新增TRUE或FALSE的傳回值，引用在本類別的deducting函數裡面，便能直接在每個迴圈中得知是否按下方向鍵，省略refer當中ll、rr、uu、dd四個key。

二、轉場函數

- fader函數

在通過關卡、gameover、抑或是答對題目時，為了營造轉場效果以提升整個遊戲的流暢度，我們加了一個fader函數來調整兩個不同關卡的背景之間隨著時間經過而不同的透明度透明度關係，以做出淡入淡出的效果。

在此函數中，我們設立alpha變數來調整畫面的透明度，先把黑色畫面的alpha上升到一定的數值，蓋掉上一個關卡的背景，再把下個畫面的背景的透明度alpha2上升，以把黑色畫面蓋掉，每個迴圈都調整一次，每次變數+1，就能夠很好的營造轉場效果。

三、關卡函數

- draw_level 函數

為了讓整個程式碼看起來更有效率，執行起來更有條理，我們在進入主循環前先設立一個關卡l函數，在關卡內放入所有前置作業準備好的內容，包括移動函數、地圖、角色函數、背景、血量、暫停函數等。

其中，地圖以location的方法把wall、trap等不同圖片置入到螢幕中，再把他們掛上移動相關函數，即可進行操作。

將1~7關的函數分別進行與上述相同的操作進行函數整合，畫出1~7關函數後再將各level函數放入主循環。

四、主循環

- main 函數

main 函數是將所有規範畫面的函數統合起來的函數，在本函數裡面需要考慮畫面之間的轉換以及過場。

首先，我們希望在一個關卡尚未挑戰完畢之前，不能進入到下個關卡。因此我們將每個level函數、fader函數、winning_screen函數以及gameover_screen函數均劃入一個迴圈，共七個迴圈。這些while迴圈均在變數not_passed為TRUE的前提下運行。唯有在winning_screen中按下了正確的選項，才能破除這個關卡的迴圈。

其次，為了規範在何狀態下要跑出gameover的畫面，我們又規範另一個變數losing。在迴圈一開始losing預設為FALSE，但是若在運行level函數並且角色的步數小於零，代表輸了，losing便會變成TRUE；另外若運行level函數並且成功通關，但在winning_screen函數中點了錯誤的選項，那losing會變成TRUE。我們分別在關卡地圖的level函數後跟問題選項的winning_screen函數後面各設一個if，用來確認是否losing變為TRUE，是的話便直接跳到gameover的畫面。因此會在main函數中看到7個這樣寫的while迴圈，並在每個關卡之前都重置not_passed與losing這兩個變數。

運行完七個迴圈之後，會跳到ending_screen的通關畫面，顯示「你的商管程成績為：」後加一個數字。這個數字由每一關結束後剩餘的HEALTH.number，或稱角色剩餘的步數加總，若總步數超過八，則顯示100分；若超過五，則顯示85分；若超過三，則顯示70分；總步數三步以內顯示60分；其餘為不及格。

參、分工方式與心得

（一）藝術總監——孫嘉蔚：

- 工作內容：找或畫出其他組員需要的圖片，例如：背景、按鈕、物體，並對部分圖片做去背、修剪等後製處理。

- 心得：學到了去背軟體的使用，以及各種圖片的繪製，也了解到如何與其他組員配合好，以增加效率。使用電腦繪圖其實不如我想像中的容易，要畫出整齊、符合遊戲設定的圖，是需要不斷修改、重畫的耐心的。

（二）技術總監——黃少凱：

- 工作內容：負責轉場、開啟程式結束程式、各個畫面的流轉之類的，還有其他組員的協調。

- 心得：學到在建構大型專案時，按照專案內容設定各種class與定義函數，以及程式撰寫有困難時查找資料的重要性。在撰寫程式之前如果能先釐清思緒與專案目的，能夠大幅提升效率。

（三）技術工人一——郭柏亨：

- 工作內容：負責主角的移動、以及跟其他角色的互動等一切與主角相關的工作。

- 心得：以為只是個簡單的遊戲，沒想到要花如此大的功夫才能完成。原本以為開發遊戲不是個很難的工作，現在完全改觀，只能對他們獻上最大的敬意。

（四）技術工人二——蔡丞中：

- 工作內容：負責障礙物、陷阱等外在物與主角互動後所造成後果，例如：箱子後的移動，踩陷阱後扣血。

- 心得：沒想到看起來簡單的闖關遊戲，背後的程式碼就如此複雜。在市面上發行的各種遊戲他們的程式想必都是製作人嘔心瀝血之作。經過了這次的期末專案，我更加了解程式設計師的工作，以及他們有多了不起。

（五）技術工人三——林柏諭：

- 工作內容：負責主畫面，包含start、quit 等按鈕，以及主角生命、關卡名稱、地圖等會掛在關卡背景之下的東西。

- 心得：本次大型專案十分困難，不單學習到了python技巧、遊戲製作、有困難時尋找資料的方法與其重要性，更學習到了組員之間的溝通與互助，期望未來有任何分組相關活動時運用這次專案所學，更加精進自己。