組語專題:

Google小恐龍(改)

組員:

資工2A-109502535-湯騏蔚

資工2A-109502536-沈富堅

分工狀況

1. 湯騏蔚(50%)
   * 開始、暫停、結束頁面(使用讀檔輸出)
   * 排行榜系統及畫面
   * 角色跳躍機制
   * debug
2. 沈富堅(50%)
   * 基本程式運行
   * 各種物體產生機制
   * 遊戲平衡性調整
   * PPT&書面報告製作
   * debug
3. 分工特色-使用github進行管理
   * https://github.com/jason01180118/ASMFinalProject
   * 有助於合作時運行的效率，在製作過程中可以時時更新狀態
   * 避免修改到隊友的程式碼導致不可預期的錯誤
4. 分工圓餅圖

遊戲玩法&功能介紹

1. 有開始、暫停、結束頁面(使用讀檔的方式)
   * 開始畫面有玩法說明並且可以查看排行榜
   * 結束畫面可重新開始或查看排行榜
2. 按下空白鍵進行跳躍(跳躍期間無法再次跳躍)
3. 不同方塊的意思
   * H(玩家操縱的主角)
   * X(障礙物，撞到就結束遊戲)
   * Z(彈簧，直接高速向上跳躍)
   * C(加速板，瞬間加速並且無視所有方塊)
   * O(金幣，可以讓分數增加)
4. 機率及延遲調整機制
   * 分數計算
   * 利用分數對速度進行調整(影響delay)
   * 利用分數對障礙物生成進行調整(機率控制)
5. 排行榜機制
   * 在遊戲結束的時候顯示分數兩秒然後請使用者輸入名稱並記錄進入排行榜
   * 記錄從始至今的前5名(用讀寫檔案的方式保留排行紀錄)
   * 可在開始畫面跟排行榜畫面切換

程式架構&困難之處

1. 創建畫面
   * 一開始本來想要用字串輸出成整個畫面，但後來發現會導致程式碼過於冗長，於是改用讀取檔案的方式，將畫面先在txt檔案裡設計好，再於遊戲運行時讀取檔案，同時也使得畫面排版設計更容易。
2. 讀寫檔案
   * 本以為讀寫檔案不會太困難，但實際上卻遇到了許多難處與bug。首先，因為讀取檔案的函式將檔案內容讀取進入buffer之後不會在最後面加上null，所以直接使用WriteString函式會導致輸出過多的字元亂碼，因此改用WriteConsole指定輸出字數。
   * 中途出現了一個bug發現若多次在暫停畫面與遊戲中切換，則暫停畫面可能出現亂碼，後來發現似乎是在LOCAL變數宣告的時候要求的記憶體太小，因此將buffer調整成4000而非原本的剛剛好3600字元。
   * 在製作排行榜系統時需要多次的開啟READ Handle與WRITE Handle，而且在讀取檔案的時候需要原有資料，在寫入檔案的時候則是要完全覆蓋原本資料，所以無法使用Irvine Library的讀寫檔函式，因此將所有的讀寫檔函式都改為使用Windows Library的CreateFile、ReadFile、WriteFile，同時因為參數不須放在暫存器中，也將程式碼稍微精簡了一些。
3. 排行榜運算
   * 一開始的想法是將使用者名稱與分數一同記錄進檔案，但後來發現並不是所有的編碼都有對應的字元，因此不能將整數型態的遊玩分數直接當作字串紀錄進檔案，改成使用雙層迴圈與條件判斷，把分數轉成字串型態再寫進檔案中，並且因為使用者名稱沒有固定長度，因此將名稱與分數分成不同檔案紀錄，而分數則是在前面補0直到滿10位數。
   * 在讀取榜上資料的時候，因為使用者名稱不像分數一樣可以用每10位為單位判斷是第幾個人，所以使用’|’字符當作每個名稱的分隔，並且在遊玩結束輸入名稱的時候加上判斷不能含有’|’字符，否則就要重新輸入。
   * 最終的排行榜函式分為幾個大部分，讀取使用者名稱，讀取舊有檔案內容，迴圈運算將舊有分數字串改成整數型態並與當前新分數比較，然後寫入進新scoreBuffer同時記錄當前新分數是第幾名，依照記錄好的當前新分數名次將新名稱插入進舊有的榜上名稱並記在新nameBuffer，然後分別將scoreBuffer與nameBuffer覆寫進檔案中。
4. 函式與變數
   * 我們為每個不同物體或功能都做了一套函式，雖然原理差不多但我們不希望他們共用一個變數或陣列，這樣的話如果想要有不同的生成機制，那在更新上將會相對困難。
   * 取名原則:我們採用lower camel case與有意義的名稱及註解增加可讀性
5. 角色跳躍
   * 我們使用了一個變數來偵測跳躍的狀態，並依據狀態和角色位置來判斷是否在跳躍期間或是落下期間，以此做出隨著時間上下移動的手法，並且一次跳躍高度為7格。
6. 字元讀取
   * 一開始我們使用的是實習課使用的readchar進行實作，但這使得在未讀取字元時沒辦法繼續執行，與我們所需要的功能不符，經過一陣子的查詢才終於找到readkey的功能來控制角色跳躍和暫停遊戲的功能
   * 而開始、暫停與結束頁面則需要玩家按下案件後才進行下一步，所以使用的是readchar的方式
7. 功能方塊
   * 共通：在每次進行迴圈過後，會將每個陣列的值往前挪動，以此來呈現角色移動的感覺。
   * 障礙物：我們使用一個陣列判斷是否有敵人在那格，並且記錄高度，利用與地板距離判斷印出對應高度的敵人，與主角座標相同時則會進入結束遊戲的函式中，並離開迴圈。
   * 彈簧：我們使用一個陣列判斷是否有彈簧在那格，並且生成在地上，與主角座標相同時則會進入開始彈跳的函式中，在5ms的延遲下連續向上7格(與跳躍高度相同)，期間內不改變其他物體位置，以此來模擬向上彈起的感覺。
   * 加速板：我們使用一個陣列判斷是否有加速板在那格，並且記錄高度，利用與地板距離判斷印出單獨的加速板(可懸空)，與主角座標相同時則會進入加速的函式中，在1ms的延遲下連續向前10格，期間內會呼叫與物體生成和移動的函式但不處理碰撞，除了金幣，達到類似傳送概念的加速。
   * 金幣：我們使用一個陣列判斷是否有金幣在那格，並且記錄高度，利用與地板距離判斷印出單獨的金幣(可懸空)，與主角座標相同時則會進入拾起金幣的函式中，金幣陣列中的紀錄會消去並將分數增加10分。
8. 遊戲運作機制
   * 使用delay的延遲與無窮迴圈來重複進行判定，那一開始設定的延遲是50ms/格，這是經過多次測試最適合進行此遊戲的速度。
   * 分數的計算就是依照玩家所走的格數和金幣計算，也就是(迴圈執行的次數+金幣數量\*10)。
   * 為了增加遊戲難度，我們用(原本的延遲-分數/64)來增快移動速度，使得分數上升到1000以上之後就會不那麼容易存活。
9. 機率產生及機制
   * 在一開始初始化先重置變數後，在每次移動一格後判定各種物體的生成，每個物體都有獨立的機率生成，但是當兩個物體同時生成則有優先度順序，分別是障礙物>彈簧>加速板>金幣。
   * 障礙物有隨著分數變動的機率，一開始是1/100，分數增加一分則增加1/100000的機率，也就是說，當分數到達一千分時會增加一倍的敵人數量
   * 其他物體的生成機率為，彈簧1/100，加速板1/50，金幣1/20

心得感想

由於我們組別demo時間較早且只有兩個人完成，且對於一些內建函式的內容不甚了解的情況下，在撰寫方面常常遇到瓶頸，即便兩個人一起討論常常也花上不少時間才能夠解決，尤其是readkey那個部分，是上網查詢不少資料才實現出來，算是前進一大步，另一方面，實習課開始教學與demo相關的東西時間有點晚，導致一開始不知道要怎麼下手，不論是handle的取得，或是其他種類的功能，許多函式因為網路上的組合語言太過多元，所以很難查詢，而在許多迴圈函式運算的部分，也是經過不斷的思考與修正，甚至不乏有整個函式重寫的狀況，