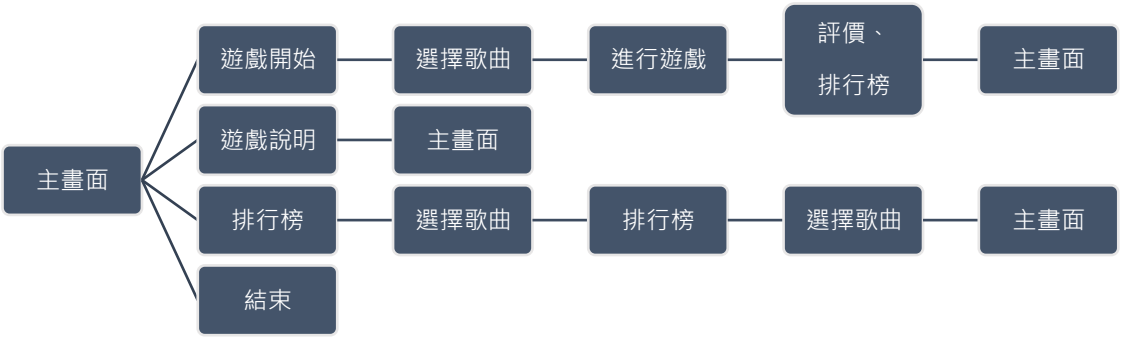
Final Project Report

(一)主題說明

1. 主題：音樂節奏遊戲
2. 遊戲說明：有四個按鍵(D, F, J, K)、五個按鍵(D, F, SPACE, J, K)兩種模式，玩家需在正確的節奏點輸入對應按鍵，程式會依據按下的精準與否給予不同分數，最後計算總和，給予相符評價。

(二)組員與分工

(三)遊戲架構



▲上圖為遊戲介面的架構

大致上的架構為：main function 內為主畫面，其餘畫面皆為函式，按下空白鍵會呼叫對應函式。幾乎所有輸出是利用 gotoxy()這個函式完成，它讓游標移至(x, y)座標處，能在特定位置輸出。

1. 主畫面：

共四個選項，按下空白鍵後跳到相對頁面。主體為一個無窮 while，在這個迴圈內會利用 GetKeyState()不斷檢查是否按下上下鍵或空白鍵。而每個選項皆對應一個數字，這裡有一變數 status 儲存這個數字，表示當下選擇的狀態。ex. 若當時的 status 為 1，則按下空白鍵時呼叫遊戲開始的函式。

2. 歌曲選擇頁：

和主畫面寫法相似，按下空白鍵後會呼叫遊戲進行的函式。

3. 遊戲進行：

比較特別的是讀入音譜檔案、節奏掉落、判斷分數的部分，分為以下幾項說明：

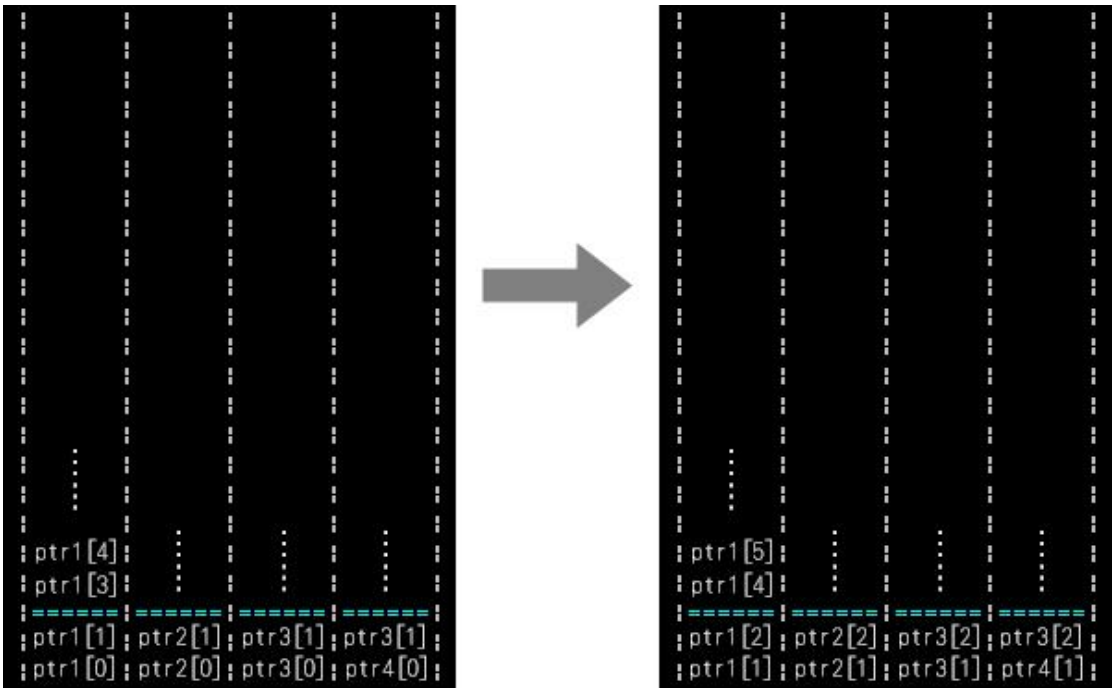
- 資料儲存
  - class Tempo：儲存節奏的資料。內有四個變數，分別為 x, y, graph, status。(x, y)紀錄當時落下的(x, y)座標，graph 為

c++ string，可能為空白（　　）或節奏圖形（------），status 用來判斷此節奏是否已按過，未按過為 0。另外還有輸出 graph、判斷 perfect, good, miss 的 member function。

- 五個指標陣列(ptr1-ptr5)：指向 Tempo 物件的陣列。每一個指標陣列都對應其中一個軌道的節奏(歌譜)。ex.ptr1 陣列儲存第一個軌道應掉落的所有的節奏。

◆ 節奏掉落方式

主體為 for 迴圈，每一次迴圈輸出整個遊戲畫面(上述的指標陣列)，並設置一變數 delay，等待數毫秒，進入下一次迴圈輸出，以製造掉落的感覺。如下圖所示：



由圖可見，第一次輸出畫面時最下面陣列位置為[0]，等待 delay 毫秒後，第二次輸出最下面的陣列位置為[1]，之後的輸出依此類推。這也是為什麼 Tempo 物件內的 graph 有可能是空白(　　)，因為要蓋掉上一次輸出的節奏圖形(------)，所以即使那個時間點沒有節奏，仍要輸出空白。

另外，為了讓掉落(輸出)的時間盡量維持同樣速度，在迴圈起始、最後利用 clock()紀錄當下時間，用兩者的差計算這個迴圈跑了多久，並在最後使用 while、clock()，直到補足 delay 時間後才進入下一圈，進行下一次輸出（然而這邊有些問題，詳見(四）待解決問題)。

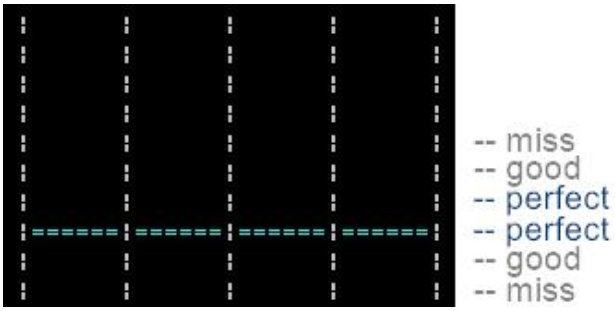
◆ 音樂播放

使用 PlaySound 函式同步播放音樂，需引入 libwinmm.a 這個檔案。

- 歌譜製作與讀入
  - 歌譜製作：先在 Osu!上製作歌譜，紀錄從開始播放音樂，每個節奏出現的秒數，再將 BPM 換算成程式中的 delay（詳見(五)補充），並測試每個陣列位置掉落至 perfect 線(淺藍色線)的秒數，對照兩者後，紀錄成 txt 檔。
  - 讀入 txt 檔：每一首歌都有 txt 檔，用以記下歌譜。txt 中第一行為所需指標陣列大小、delay 時間。再來的每一行分別記錄軌道、節奏出現的陣列位置。ex.0 10 即代表 ptr1 的第 10 個陣列位置是有節奏的，Tempo 物件的 graph 為(------)。
    讀入檔案之後，指標陣列中物件的建立就完成了，接下來就按照上述節奏掉落的方法進行遊戲。

- 分數判斷

在節奏掉落的 for 迴圈裡同時進行判斷，判斷的函式寫在 class Tempo 內的 member function。最下面幾行皆為判定區，會在一行一行輸出指標陣列圖形的同時，檢查這之中是否有節奏，即 graph 為(------)的物件。若有，則利用 GetKeyState()檢查是否按下對應按鍵，有按下按鍵則加上對應分數(perfect, good)，並將物件的 status 設為 1(已按過)；如果提早按下會判定為 miss，並將 combo 歸零，或到最後一行時，Tempo 物件的 status 仍為 0，也判定為 miss。



▲判定區域示意圖

4. 評價畫面

遊戲進行的同時會記錄分數(score)、最大 combo 數、perfect, good, miss 的個數。遊戲結束後輸出評價畫面，會將這幾個變數輸出，並計算準度(accuracy)，給予 S, A, B, C, D 等對應評價(Rank)。在這個頁面會要求使用者輸入名稱，更新排行榜。按下空白鍵跳至遊玩歌曲的排行榜，再按一次空白鍵則跳回主畫面。

5. 遊戲說明頁

輸出遊戲時的畫面，說明玩法。按下空白鍵可返回主畫面。

6. 排行榜

每次遊戲結束後，會讀入排行榜的 txt 檔，逐一比對分數，查看是否更新紀錄；若有，則更改排名，輸出檔案。每一首歌都有排行榜，從主畫面進入時，會先出現選擇頁，寫法與主畫面類似；選擇後讀入對應的 txt 檔，輸出畫面。按下空白鍵可返回選擇頁。

(四)待解決問題

前面已說明如何調整落下速度到一致，即利用 while、clock()調整每次迴圈的速度；這個方法理想上可行，但實測時落下速度仍然不一。推測速度不一致的情形可能是以下原因造成：

1. 跑迴圈的速度已超過 delay 時間
2. clock()的準度問題

我們曾測試過各自電腦跑迴圈的時間，在 delay 為 50 毫秒的情況下，有的電腦是很完美的 50 毫秒，然而有的是 60 毫秒左右。原本以為用更準確的計時 QueryPerformanceCounter()可以解決這個問題，但後來發現，有時候是 clock()準確，QueryPerformanceCounter()反而會使掉落速度增快。以下是幾台電腦跑這個遊戲的情形：

A: 完全準確

B: 有時中途會變速

C: 有時準確，有時會整體變慢

D: 不按按鍵時準確，開始按按鍵會變慢

目前還沒辦法解決這個在不同電腦、不同時間，掉落速度不一的問題，不知道這樣的情況是否也和硬體配備、系統等有關係。畢竟在音樂節奏遊戲中，節拍和音樂的配合相當重要，若掉落時間不一致，遊玩的樂趣必然會下降。

(五)補充

- BPM 換算成 delay 的方法

  為了考量程式運算速度，使用較折衷的換算方法：找 Offset 跟每一拍距離的公倍數，再除以一個數字。

  ex.

BPM: 180.960 每一拍距離：0.331 秒或 0.332 秒 譜面最短距離：0.165 秒 Offset：0.281 秒 落下長度：20 格	0.331*0.281 = 0.093011 0.093011/2 = 0.0465055 取 delay = 46.5055 毫秒 (最大誤差為 0.1)