組合語言 - Final Project

The Music World

b96902039 黃信博

b96902091 歐志先

b96901134 楊上逸

* Motivation :

在因緣際會之下，在高一時組員接觸到了一款 NDS 遊戲「大合奏！バンドブラザーズ」，譯為「明星樂團大合奏」。這款是由任天堂製作團體群所開發出來的音樂遊戲，除了擁有明亮生動的介面做為號召外，最有別於其他音樂遊戲的特色為其大合奏的按鍵是確實的對應音階。不像普通跳舞機隨機出現，我們必須跟著譜表進行，然後在固定時機（明確來說為 TEMPO）按下對應按鍵。除此之外，更可以多人合奏一首曲子扮演好自己的角色，多方聯彈，相當的引人入勝。因此，當得知這款遊戲時，便決定要以此為主軸靈感，做為此次 Final Project 的核心想法。

* Our Game :

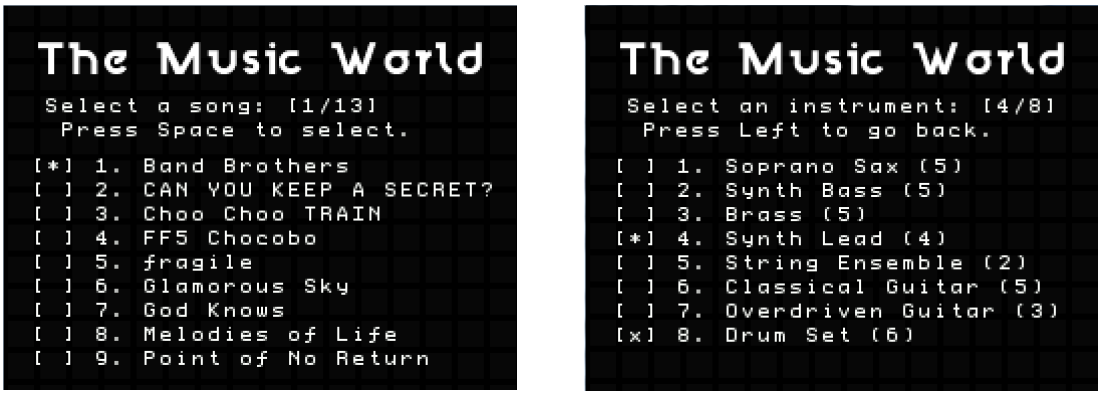
一個好的遊戲，需要良好的程式來支持整個系統；但如何能與玩家溝通以及提供更友善的使用者介面，也是我們的嘗試之一，亦是不可或缺的重要一環。

* Interface :

首先為使用者介面的部分，為了提供一個良好的介面，在下圖的主選單使用了 ArtWeaver 這個軟體。它所提供多樣性的功能，使得介面在視覺方面得到較貼近於一般市面遊戲的水平。

在遊戲的一開始，首先呈現於玩者前的是一張主選單，隨著使用者輸入上與下兩種指令，選單會隨著變動，並將目前所選的位置做特殊的處理，使得使用者明確能夠了解現在選單的情況；在主選單中，共有三個選項，分別是 Start Game、Thanks 與 End。使用者

圖一：主選單。



圖二：選歌單及樂器選擇。

可藉由上下做任意的選擇，並按下 Space 鍵後繼續。

一、Start Game：

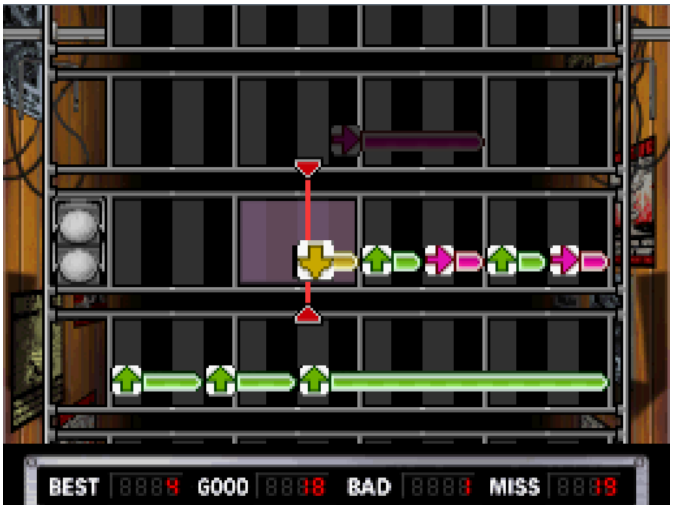
選擇 Start Game：與「大合奏!バンドブラザーズ」相同，我們提供了選歌單。在每首歌曲中都可因自己的愛好，自由選擇所想要演奏的樂器，並可以 Page Down 快速選歌。不同樂器所演奏出的音樂也都不同。所有音樂都是由樂團表演，在聽音樂的過程中，顯示出相當多元的風格，也正因為這種多樣性的演奏廣度，使得可以將單獨樂器抽離而不顯單調，玩家在依括號內的難度，任意選擇一項喜歡的樂器後，按下 Space 遊戲就開始了！

「精緻的畫面」也是「大合奏!バンドブラザーズ」當初令人耳目一新的地方，特別是它閃亮的排頭燈光以及動態的視角令我們印象深刻，因此，在遊戲畫面我們儘量追求較高的品質，如您所見，畫面中有閃亮的燈光搭配紅色的準線用以告知使用者進行的位置，使用者隨著螢幕的提示，鍵入上下左右，若有符合即可消掉音，按得越準分數越高！除了要按滿音符的長度，每個長音在不同的時機按下有不同的處理方式。(此時按下 Backspace 可回選歌單。)

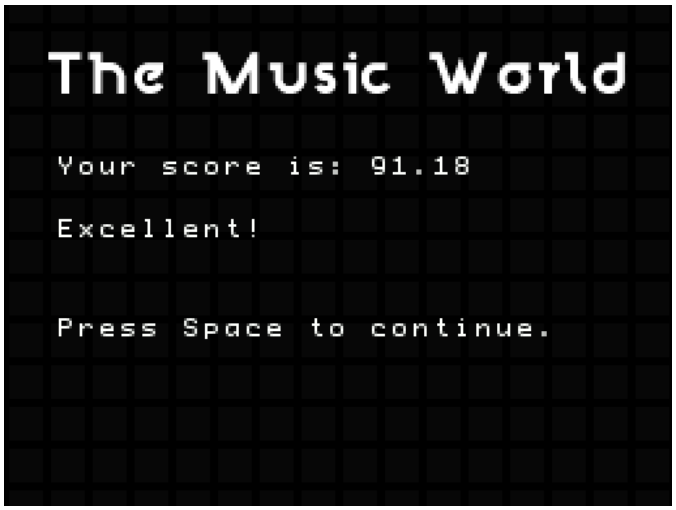
評分方式：

Best→節拍精確　Good→有按到　Bad→沒對到節拍但按對鍵

分數 = (100 x Best + 90 x Good + 60 x Bad-20 x Miss) / 總數



圖三：遊戲畫面。



圖四：分數總結。

二、Thanks：

這次的作業，我們一致認為要感謝祝教們以及老師的指導，以及為我們的作品留下紀念的表徵，因此有了 Thanks 這個選項，採用動畫的形式表現，配合翱翔於天際的海鷗，譜成一幅視覺的饗宴。

圖五：感謝狀。

三、End：退出遊戲。

Implementation：

在決定以遊戲作為主題，並決定內容後，我們便在 Intel 及 GBA 兩個架構上作抉擇，而我們想到可以使用的 Library 分別是 Allegro 及 HAM。

我們先上網查尋有關資料，我們發現網路上有許多網友將他們所寫的音樂檔 (*.bdx) 供人下載至 NDS 遊戲機裡。因此若我們能讀懂 bdx 的結構，將能輕易地以任何網友寫好的樂譜進行遊戲。

由於這兩套 Library 都是首次接觸，研究各自的功能後，我們選擇在 Intel 上，透過 Visual Studio 2008 以 Allegro Library 來編寫遊戲，並將可以以組合語言改寫的部分用組合語言改寫。原因是，稍微解讀 bdx 檔後，我們發現裡面儲存的是一個一個音符，而不是一般遊戲可以使用的 midi 檔。要完成這個遊戲，我們必須能直接控制音符的輸出，而 Allegro 正好有 **midi_out()**可供直接傳給 midi 指令給系統，在 HAM 裡則似乎缺少了相關函式。

程式主要可以分為 5 大部份，分別為**遊戲控制**、**音樂讀取及輸出**、**繪圖**、及**組合語言實作**。(以下不依進度照列，因為彼此互相關連。)

A、**讀取音樂檔及控制音樂輸出** (Music.cpp, Music.h)

我們建立了 **class Music** 來掌管一切的 bdx 檔案讀取及 midi 音符輸出。這是一個龐大的 class，光是 public 可以存取的函式就有 17 個，並有兩個 array 存放音符的對應鍵及使用者輸入。

在讀檔方面，我們僅就有讀出的內容進行實作，因此有些許樂器找不出相應，以及變速等功能，都無法讀出。除此之外，為了以 midi 輸出音符，我們還研究了 midi command，如傳送 {0x90, 0x3C, 0x7F} 可以在 Channel 1 放出最大聲的 Do 音。

我們以每 10ms 就加一的 speed_counter 來調整遊戲速度，並可以依此算出目前的 tick，即進行到第幾個音符。以下列幾個重要函式的

功能。

函式名稱	功能
read_file	讀取 bdx 檔。
play	以目前的 tick 為準，播放該有的音。
play_metro	播放開始前的節拍器。
init_patch	設定好 midi 各 channel 的樂器、音量等。
note_on	播放某個音。
get_song_name	取得 bdx 檔內記錄的檔名。
get_tempo	取得 bdx 檔內記錄的音樂節奏。

除了將音符輸出外，我們還實作了左右聲道，以及選擇的樂器聲音會被放大等功能。而音色方面，則使用軟體模擬，以避開傳統 midi 的低品質聲音。

B、繪圖（遊戲 – Draw.cpp，選單 – Game.cpp）

我們先提選歌單及樂器選擇的畫面。我們想在 Allegro 上畫出好看的字，但卻一直無法成功，最後便求助於使用圖形來完成。因此我們寫了一個 draw_font 函式，能將字型好看地畫到螢幕上，加上會動的背景，也營造出高級的假像 XD。

遊戲畫面上，我們在 NDS 模擬器抓取原遊戲的圖形，並盡量做到盡善盡美。第一步先畫出會動的框架（frame），接著再畫出會移動的游標、音符（或是按鍵、key）、最後再加上頭燈以及會變色的 frame、最下方的分數列（分數跳同時會顯目標示）以及會變大的音符。

由於我們希望 miss 掉的音符（key）能夠慢慢變暗，我們另外寫了一個函式將圖形依程度變暗。這是我們首次使用 asm 的地方。另一個使用 asm 的地方是，我們捨棄不用幾個 Allegro 內建的重要繪圖函式，而以組合語言改寫，進而更了解真正的機器行為。主要有以下三個，就像第二次作業一樣，我們必須還判斷有沒有超出範圍。我們皆個各自進行過一點最佳化。

(a) `my_draw_sprite(BITMAP *bmp, BITMAP *sprite, int x, int y)`
將 sprite 的圖形，以 OxFFOOFF 作為 mask，畫到 bmp 的(x,y)上。

(b) `my_blit(BITMAP *source, BITMAP *dest, int source_x, int source_y, int dest_x, int dest_y, int width, int height)`
將 source 的(source_x, source_y)開始的矩形寬高(width, height)，畫到 deset 的(dest_x, dest_y)上。

(c) `my_masked_blit(BITMAP *source, BITMAP *dest, int source_x, int source_y, int dest_x, int dest_y, int width, int height)`
同上，但多了 mask 的功能。

C、遊戲控制 (Game.cpp, Main.cpp)

主要是進行按鍵偵測以及控制遊戲流程。其中 `dealKey_asm` 是我們翻譯的 asm 中最長的一段，並且由兩人合力才翻完。

* Difficulties：

第一個難點是，bdx 檔的解讀。除了網路上僅有少少一點網友解讀的英日文資料外（我們特地寫信給在澳洲的網友 Neobeo，請他分享他研究的成果），我們必須慢慢嘗試各種樂器在原始遊戲中及 midi 的對應，尤其是鼓組。在反覆的嘗試下，我們得出大部份的對應值，但仍有許多是嘗試不出來的。

第二個是遊戲中按鍵的偵測。因為在很短的時間內，可能同時有很多個音，又 Allegro Library 只能偵測按鍵「是否是按著的」，而不能偵測「是否被按下」，再加上人的速度不可能太快，必須在一定誤差內有一定給分，因此連兩個音容易誤判，甚至有按一下消去兩個音的情形。

由於按鍵的偵測我們容易將他從 Allegro 中分離開來，這也成為我們決定翻成組語的函式，但也是組語最難翻的一段。過程中不斷有 bug 出現，以及對 instruction set 的不熟悉都成為我們的絆腳石。例如，我們忽略了 char 應以 `cmp byte ptr [eax + ebx], cl` 進行比對，而以 `cmp byte ptr [eax + ebx], cl` 比對，而導致許多錯誤。

第三點則是，由於程式是由 Visual Studio 寫成，在完成之際，才發現在其他電腦上幾乎都不能跑，經過幾個小時上網尋找答案，我們決定使用 g++再編譯一遍，然而，兩個編譯器的 inline assembly 語法截然不同，一直失敗。幸好我們最後有找到 AssyXlate，可將 Intel 的語法轉為 AT&T，接著再手動稍微修改一下程式碼，使兩個編譯器皆可編譯！我們將 Visual Studio 用的檔名取為 asm.cpp，而 g++用的則是 asm_gcc.cpp，而附上的執行檔則是以 g++編譯而成。

* How We Divide the Work :

以下列出我們的分工方式，報告則各自負責。

姓名	學號	工作
黃信博	B96902039	設計主畫面、測試、提供點子。
歐志先	B96902091	組語翻譯、測試。
楊上逸	B96901134	整體架構、音樂、畫面設計、debug、組語。

* What We Want to Say & What We've Learned :

B96902039　黃信博

在這次的期末專題報告中，學到了不同於以往的真實感，在之前的程式之中，不管是 c 語言或是 JAVA 都感覺離現實生活有一段距離，顯的較無實用性。透過這次機會，原來遊戲可以真的被實做出來！在透過繪圖、allegro 的 c 語言 library 以及組合語言中，慢慢地一步可以操作的遊戲，從積沙成塔以至成形，現實的象牙塔像似倒塌，隔閡不復存在了，這種感覺很美好。還要特別謝謝老師這學期心苦的教導、助教的用心，使得學生對於電腦基本架構、組合語言有更深入的了解及興趣。

B96902091　歐志先

對於關鍵 code 部分的加速，其實不是很容易。因為 c 語言本身就是個很有效率的語言，要在這種情況下再加速自然不是一件容易的

事情。其中，發現組語是很有技巧的，就是要利用各種奇怪的 tricks 去想辦法加速，而且跟 code 的本身也有相當大的關聯，必須越接近一些特定格式的比較能加速，萬一不能比 compiler 聰明的時候...，所以有的時候組語寫不好甚至會比 c 還慢。

然後，我覺得寫組語最難的莫過於抓 bug 了，相較於那些小技巧只要仔細想一下通常都有，可是其實不管哪種語言的 debug 一向就是大家最討厭的工作，而偏偏組語的 debug 又比其他語言要來的更不容易，不只 code 本身很難看懂，而那些奇奇怪怪的方法在 debug 的時候就是最痛苦的地方了，我覺得其實時間大多是花在這件事身上，debug 真的是很痛苦啊。

B96901134　楊上逸
雖然我不是本系生，但卻對電腦資訊方面很有興趣。除了這學期的課程讓我更了解電腦運作原理外，這次的 final project 更讓我複習、加深所學，而且也學習到了許多程式設計技巧。比如說，如何設計程式架構讓程式更容易維護等，尤其是學會了 Allegro 的使用，讓我在設計的層面上更廣。雖然要慢慢地嘗試 bds 檔的樂器對應，且目前程式仍有一些 bug，但能從無到有，在短短三天內，天天熬夜趕出屬於我們自己的作品，也很有成就感！

執行檔網址：
http://20h.3b8.cc