Musical Partner - 你的練琴好夥伴

林昱睿 110022207
陳柏睿 109062105
黃驥軒 109591521

目錄

研究動機

本團隊有兩位隊員（林昱睿、黃驥軒）為電機系聲學與聽覺研究室的成員（由劉奕汶教授帶領指導），在過去一年的期間接觸到實驗室所開發的自動伴奏系統，而該系統的核心是以 Python 語言所開發。由於自動伴奏同時也是 MIR 領域所關切的研究項目之一，因此我們決定以此作為期末專題的切入點。

此自動伴奏系統主要以文字命令列介面（Command Line Interface, CLI）環境操作，因此在使用者體驗部分極不友善。於是我們決定替此系統打造一個友善的圖形使用者介面（Graphical User Interface, GUI），以改善使用者體驗不佳的部分。

研究方法

實作的部分我們選擇使用 Python 內建的 Tkinter 。

Tkinter 是用於創建 Python 圖形使用者界面（GUI）的標準函式庫。裡面提供了一組簡易的工具和小部件，讓 Developer 替自己的 project 創建互動式視窗、對話框、按鈕、選單等，包含各種 GUI 元素的程式。也因為 Tkinter 跨平台的特性，在 macOS、Linux、Windows 上都可以運行以其所開發的 GUI app。

我們原先想要嘗試以 PyQT（另一種 Python GUI tool）作為開發的工具，但是由於其上手難度較高，加上開發時遇到的一些問題，最終選擇較為簡易的 Tkinter 來完成。

程式介紹

此程式主要分為三個部分：Recorder、Midiplayer 和伴奏系統，在下方會分別做簡單的介紹。我們實作 GUI 的目的便是為了將這幾個部分做整合、連結，讓我們可以透過簡單的介面來實現資訊的即時顯示、錄音、播放預錄音檔和自動伴奏的功能。

其中，上述提到的 Recorder 與伴奏系統是實驗室已經做了一段時間的成果。在這次的 Final 中，我們完成的是 GUI 的整合以及 Midiplayer 的部分。

1. Recorder：

首先介紹 Recorder 。這部份是用於錄音，將捕捉到的 midi information 轉換成我們需要的形式並以 .txt 檔儲存，以便後續的使用。其中有一些實作上的細節，例如：在背景中持續播放固定節拍的高音作為節拍器來協助我們錄製音檔（輸出時忽略此音）、以 multi-threading 的方式來計算不包含在原始 midi information 中的 note 的彈奏時間並使 GUI 順利運行……等等。關於兩份 .txt 檔的介紹如下：

a. input.txt：儲存的是原始的 midi 訊息，我們可以使用 midiplayer 來播放其中收錄的資訊。其形式為：['Keyboard1', 128, 60, 122, 52.157103300094604]，分別代表：錄製的器材（Kb）、note on/off（按下/鬆開）、midi 音高、velocity（按鍵按下去的強度）以及 midi time。
b. output.txt：儲存的是伴奏系統需要的input形式，包含四個 information：[1 144 64 50]，分別代表：index（第幾個音）、note on/off（按下/鬆開）、midi 音高以及 midi time。

2. 伴奏系統

這部份細節較為繁瑣，因此這裡只介紹概念，程式細節的部份就不多做贅述。
伴奏系統的部份我們使用的是 Kuramoto Model。這個 model 可以用來形容多個震盪器之間的同步行為，如同我們報告時所說，給兩個人互相聽對方敲擊的聲音，最後這兩人會達到相同的敲擊頻率。藉由這個想法，我們想要使伴奏系統可以自己聽取我們彈奏的內容，並配合我們彈奏的速度演出。

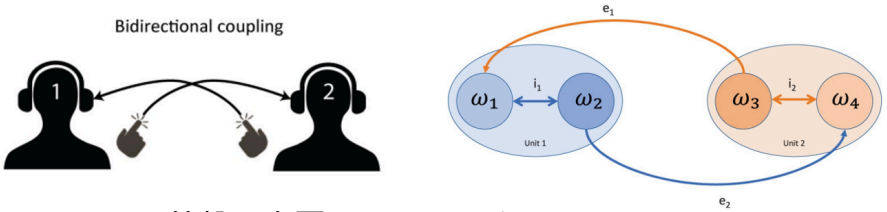


Fig1. 敲擊示意圖，w_1、w_4為耳朵，w_2、w_3是手

在程式的實作中，我們分別用三個 w 來表示這個伴奏系統的耳朵、大腦和手，如同把這個伴奏系統當成一個人。當耳朵聽到輸入的音樂時會將訊息告訴大腦，大腦再和手說要彈奏的內容，接著在手彈奏的過程中，大腦也會逐漸適應，最終讓大腦與手自我調適到至一平衡點。

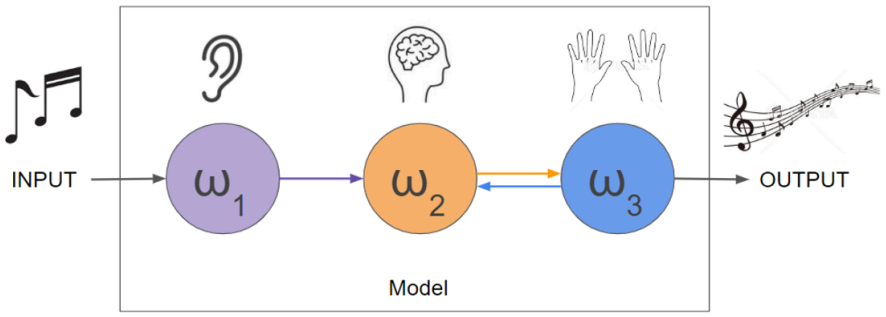


Fig2. 伴奏系統示意圖

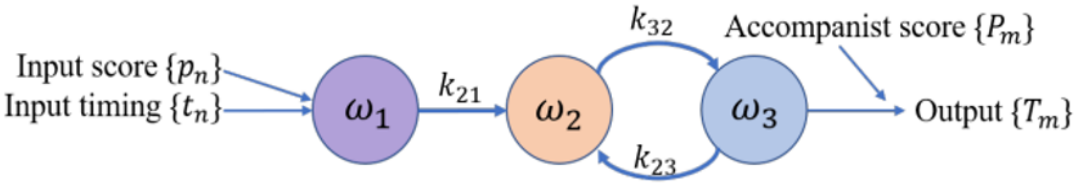


Fig3. 實際 input / output 的內容

上圖為整體的架構想法，在程式實作時還需要加入一些內容。我們使用3個 thread 來進行實作：第一個 thread 持續地接收輸入音並按照順序排好；第二個 thread 讓伴奏系統在收音的同時進行計算；第三個 thread 則把計算完成的音也依順序排好播出。

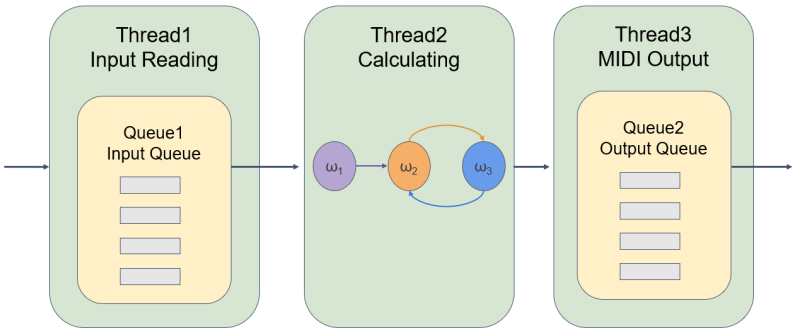


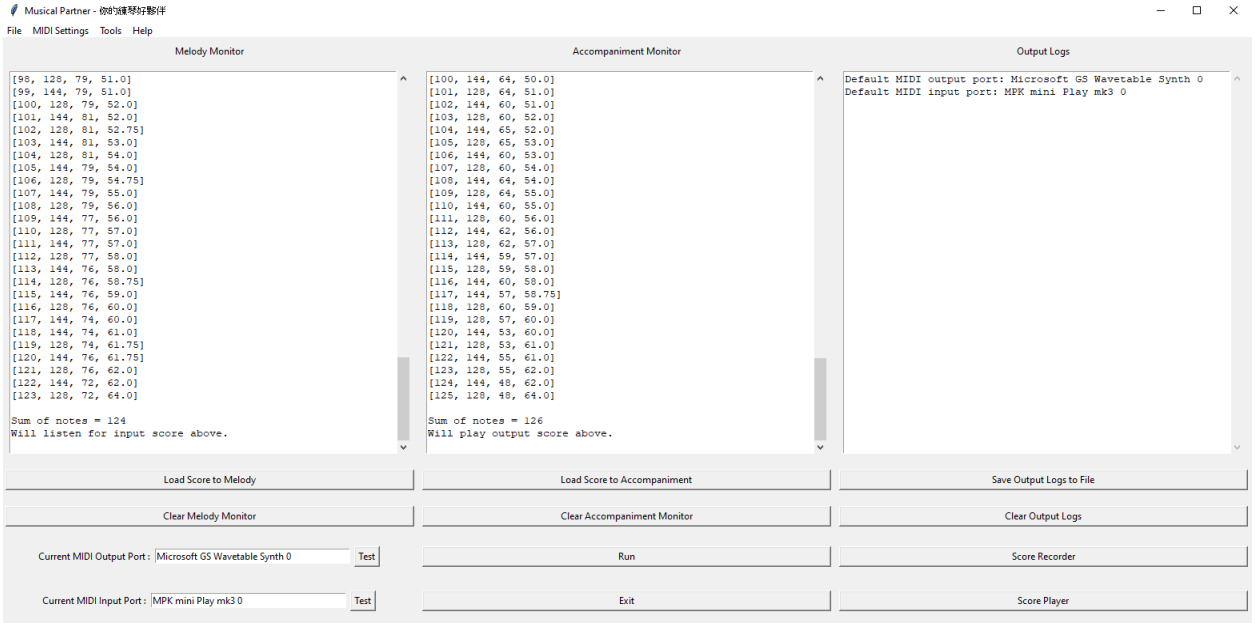
Fig4. 架構圖

其他更詳細的內容還有計算我們有在 reference 附上 paper，有興趣可以參考~

3.Midiplayer

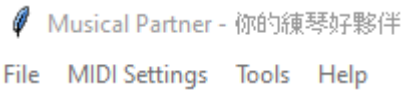
這部分的程式主要將 Recorder 程式所產生的 input.txt 純文字樂譜格式檔，轉換成標準的 MIDI 訊息，再轉換成標準的 MIDI 格式並播放。例如：['Keyboard1', 128, 60, 122, 52.157103300094604]，會被轉換成 [128, 60, 122] 的標準 MIDI 格式後，透過與系統所連接的 MIDI 輸出裝置播放此 MIDI 訊息。

4. GUI

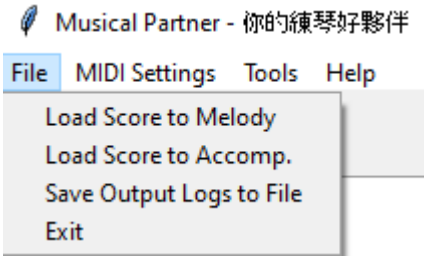


GUI 分別由下列部分組成：

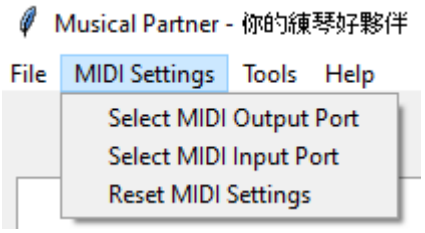
- 主功能表列 (Menu Bar) - File, MIDI Settings, Tools, Help



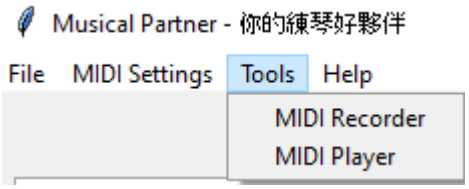
  - File - 檔案功能表
    - Load Score to Melody - 載入主旋律樂譜檔案至系統
    - Load Score to Accomp. - 載入電腦伴奏樂譜檔案至系統
    - Save Output Logs to File - 將系統輸出的訊息儲存成文字檔
    - Exit - 離開主程式



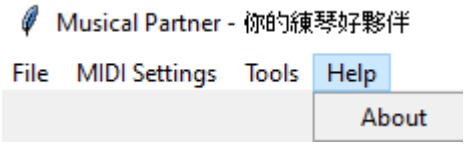
  - MIDI Settings - MIDI 環境設定與重置
    - Select MIDI Output Port - 選擇 MIDI 輸出裝置
    - Select MIDI Input Port - 選擇 MIDI 輸入裝置
    - Reset MIDI Settings - 重製 MIDI 環境設定



    - Tools - 開啟 MIDI 錄音器與撥放器
        - MIDI Recorder - 開啟 MIDI 錄製程式
        - MIDI Player - 開啟 MIDI 播放程式



    - Help - 其他功能
        - About - 顯示開發團隊與版本資料



- 3 個文字方塊 (Scrolledtexts)
    - Melody Monitor - 顯示已載入的主旋律樂譜檔案資訊
    - Accompaniment Monitor - 顯示已載入的電腦伴奏樂譜檔案資訊
    - Output Logs - 顯示系統即時狀態訊息

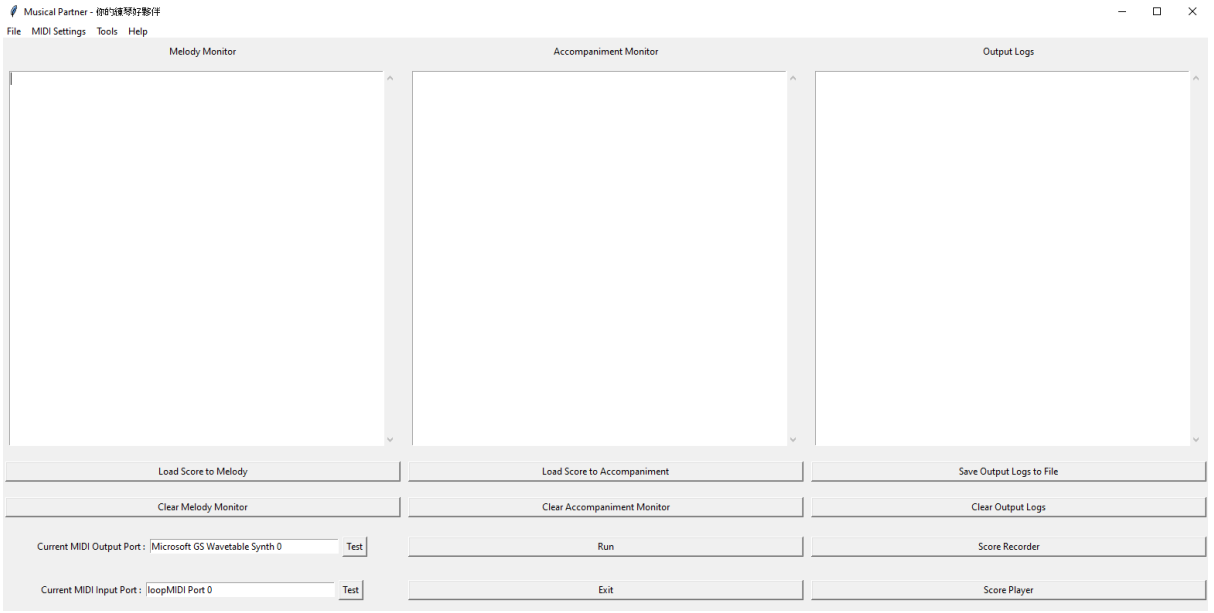
- 10 個按鈕 (Buttons)
    - Load Score to Melody - 載入主旋律樂譜檔案至系統
    - Load Score to Accompaniment - 載入電腦伴奏樂譜檔案至系統
    - Save Output Logs to File - 將系統輸出的訊息儲存成文字檔
    - Clear Melody Monitor - 清除主旋律樂譜檔案資訊
    - Clear Accompaniment Monitor - 清除電腦伴奏樂譜檔案資訊
    - Clear Output Logs - 清除系統即時狀態訊息
    - Run - 執行伴奏程式
    - Score Recorder - 執行 MIDI 錄製程式
    - Score Player - 執行 MIDI 播放程式
    - Exit - 結束主程式

- 2 個框架 (Frames)
    - 框架1 - 包含標籤(Label), 文字方塊(Entry), 按鈕(Buttons) 各 1 個
        - Label - Current MIDI Output Port :
        - Entry - 顯示目前系統的 MIDI 輸出裝置
        - Buttons - 測試 MIDI 輸出裝置
    - 框架2 - 包含標籤(Label), 文字方塊(Entry), 按鈕(Buttons) 各 1 個
        - Label - Current MIDI Input Port :
        - Entry - 顯示目前系統的 MIDI 輸入裝置
        - Buttons - 測試 MIDI 輸入裝置

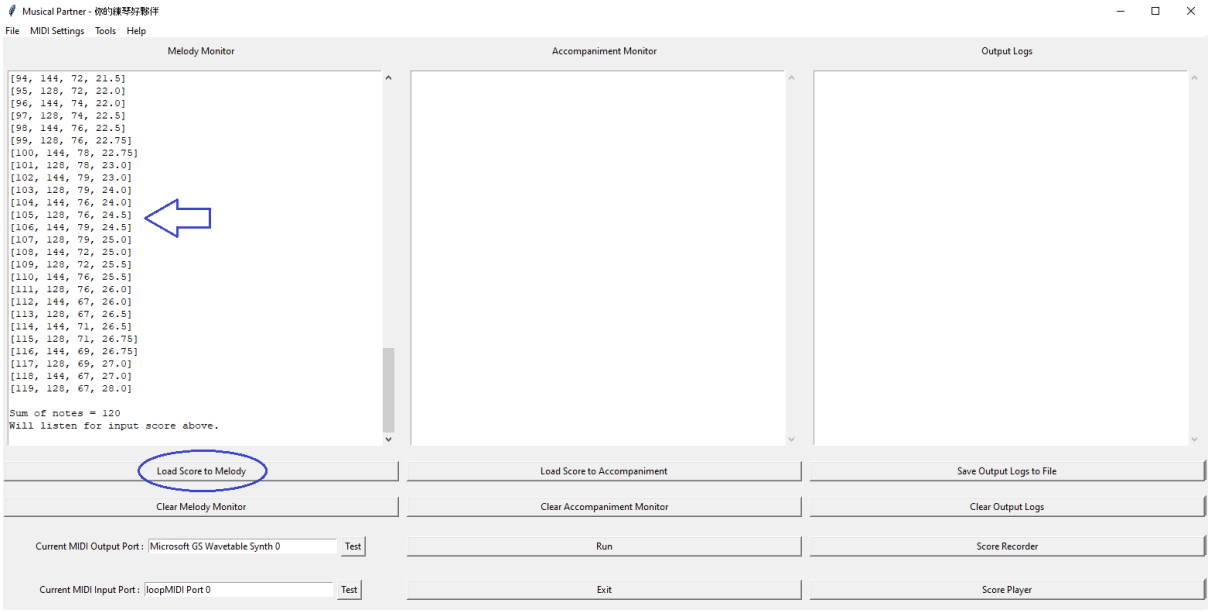
操作說明

1. 將帶有音源的電鋼琴或 MIDI 鍵盤透過 USB 介面連接至電腦

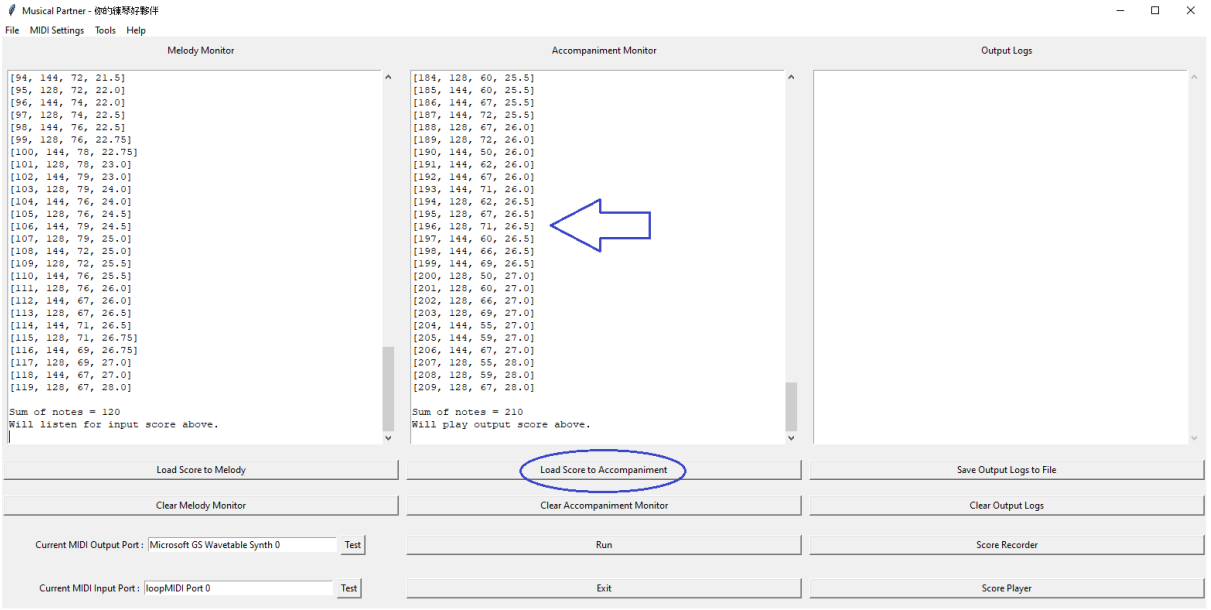
2. 打開主程式



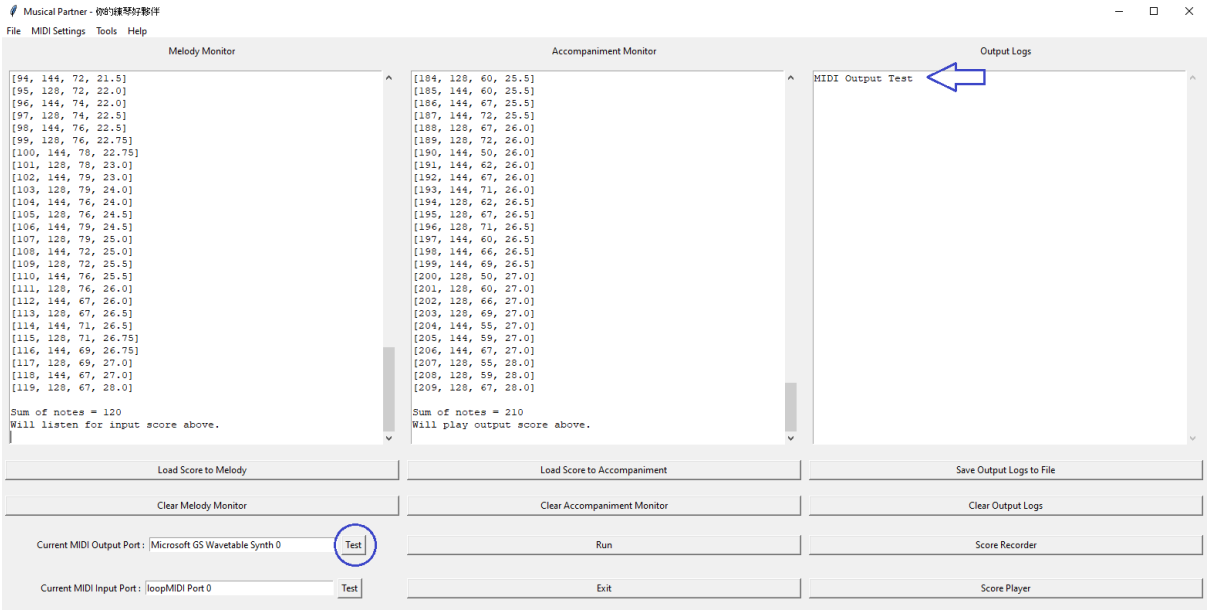
3. 點選「Load Score to Melody」按鈕，將主旋律樂譜檔案載入



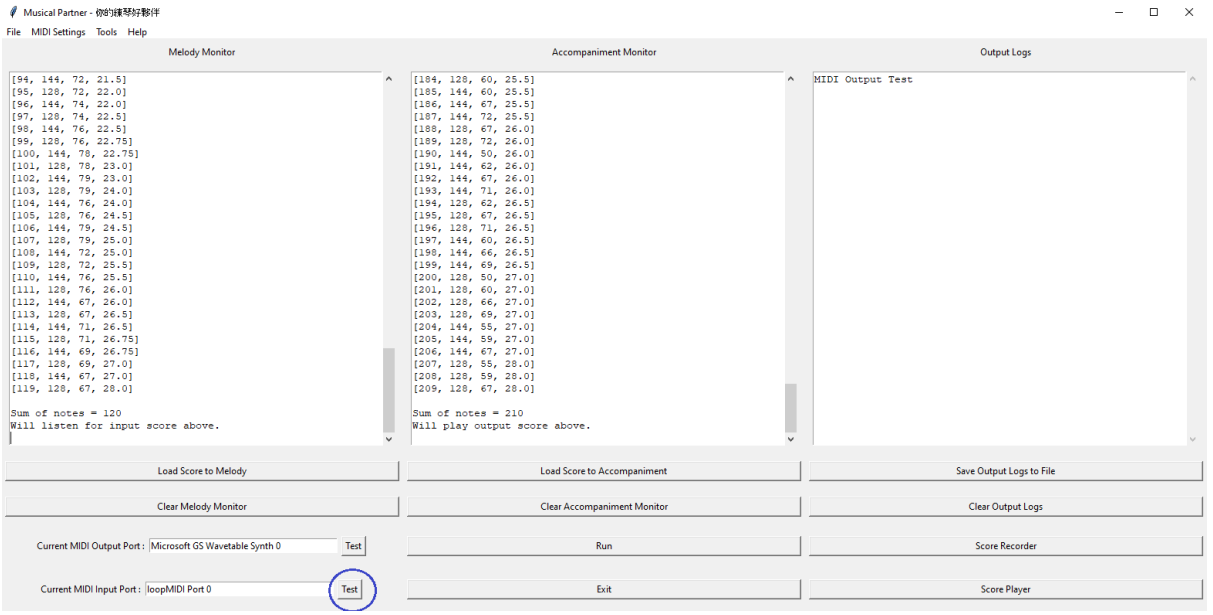
4. 點選「Load Score to Accomp.y」按鈕，將電腦伴奏樂譜檔案至系統載入



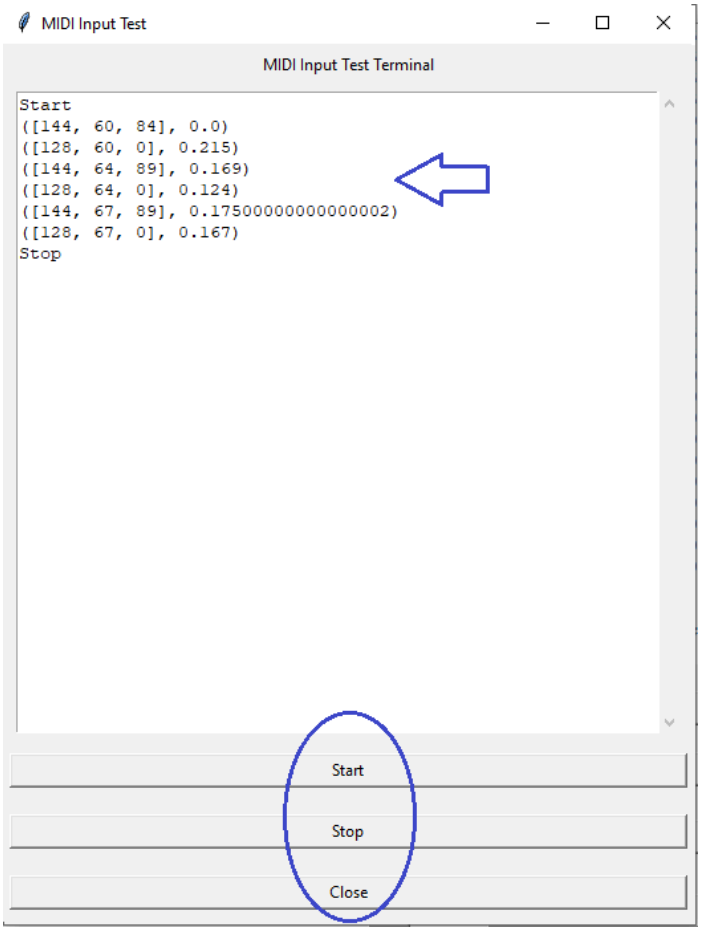
5. 測試 MIDI 輸出裝置，按下「Current MIDI Output Port」旁的「Test」按鈕，「Output Logs」會顯示「MIDI Output Test」，此時電腦所連接的 MIDI 輸出裝置會發出四個音，若有聽到聲音即代表 MIDI 輸出裝置連接正常



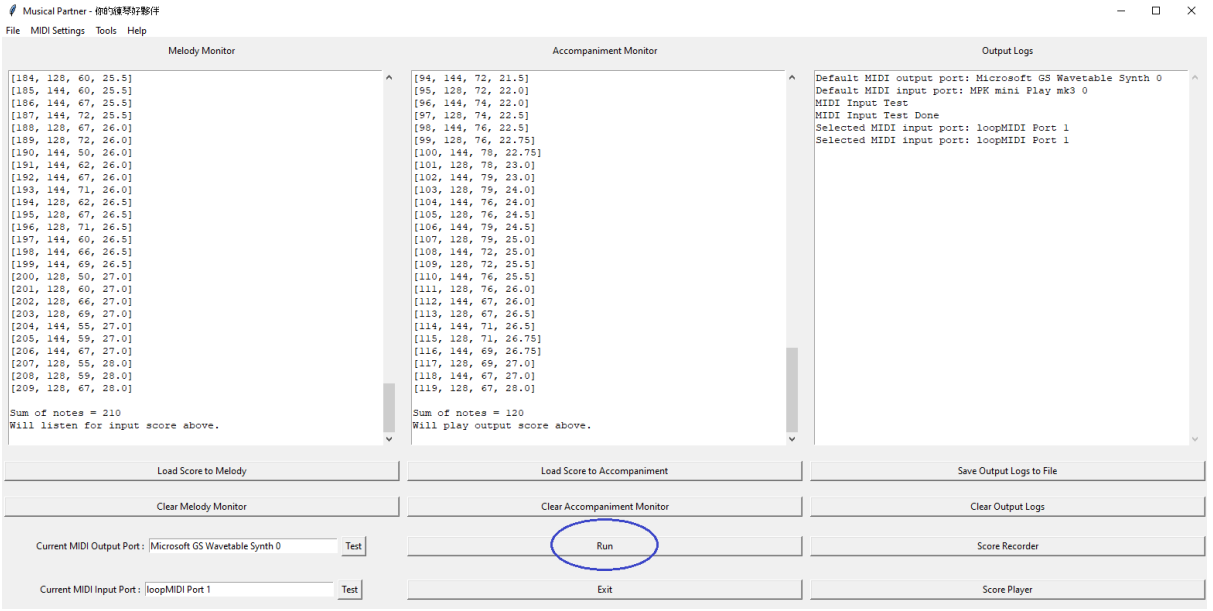
6. 測試 MIDI 輸入裝置，按下「Current MIDI Input Port」旁的「Test」按鈕，此時會出現 MIDI Input Test Terminal



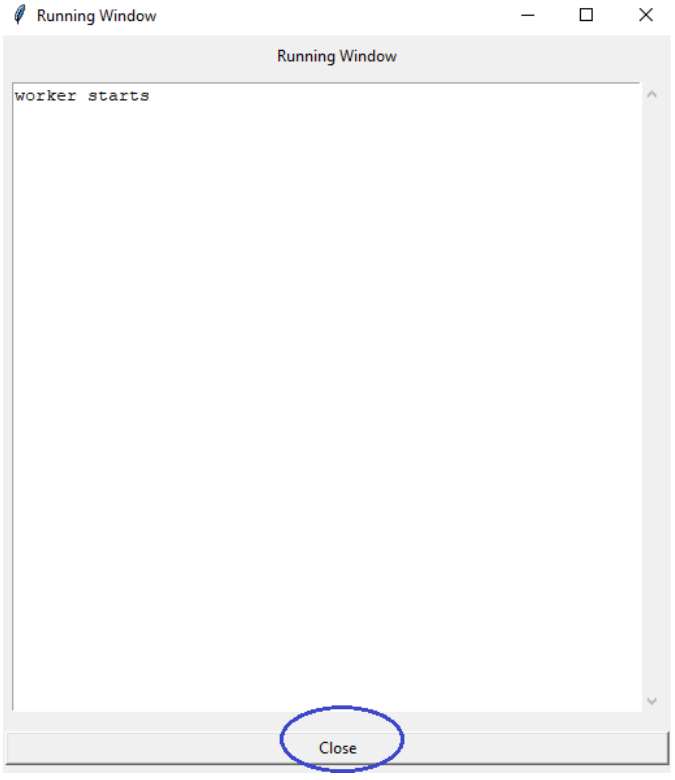
按下「Start」按鈕後開始彈奏所連接的 MIDI 輸入裝置，若有MIDI 訊息出現在視窗即代表 MIDI 輸入裝置連接正常，按下「Stop」按鈕結束 MIDI 輸入測試，最後再按下「Close」離開 MIDI Input Test Terminal



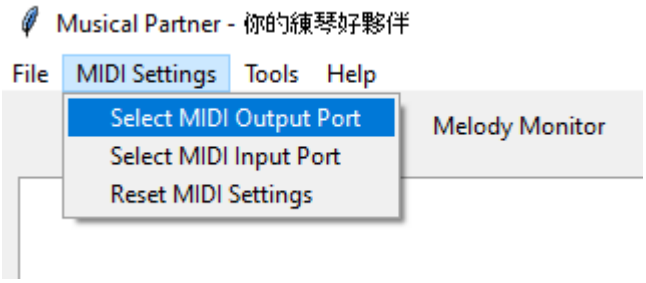
7. 按下「Run」，此時會出現「Running Window」，確認「worker starts」訊息出現後即可開始彈奏，此時會聽到電腦所彈奏的伴奏部分。



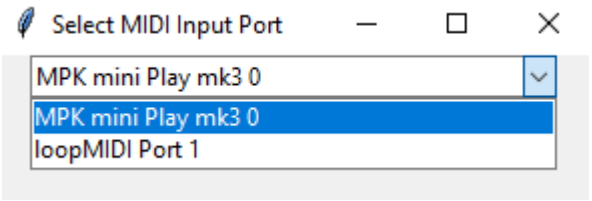
彈奏結束後再按「Close」按鈕離開「Running Window」



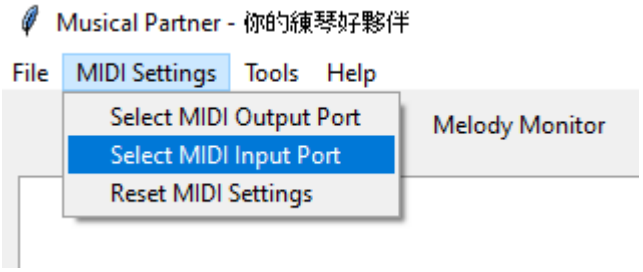
8. MIDI 裝置設定
   a. 更改 MIDI 輸出裝置步驟如下：
      主功能表 → MIDI Settings → Select MIDI Output Port



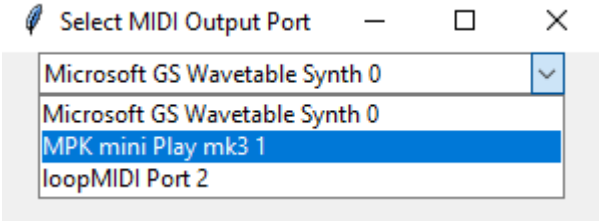
      出現 Select MIDI Output Port 視窗後，再從下拉式選單更改輸出裝置



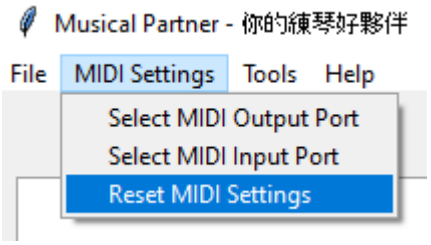
   b. 更改 MIDI 輸入裝置步驟如下：
      主功能表 → MIDI Settings → Select MIDI Input Port



      出現 Select MIDI Output Port 視窗後，再從下拉式選單更改輸入裝置

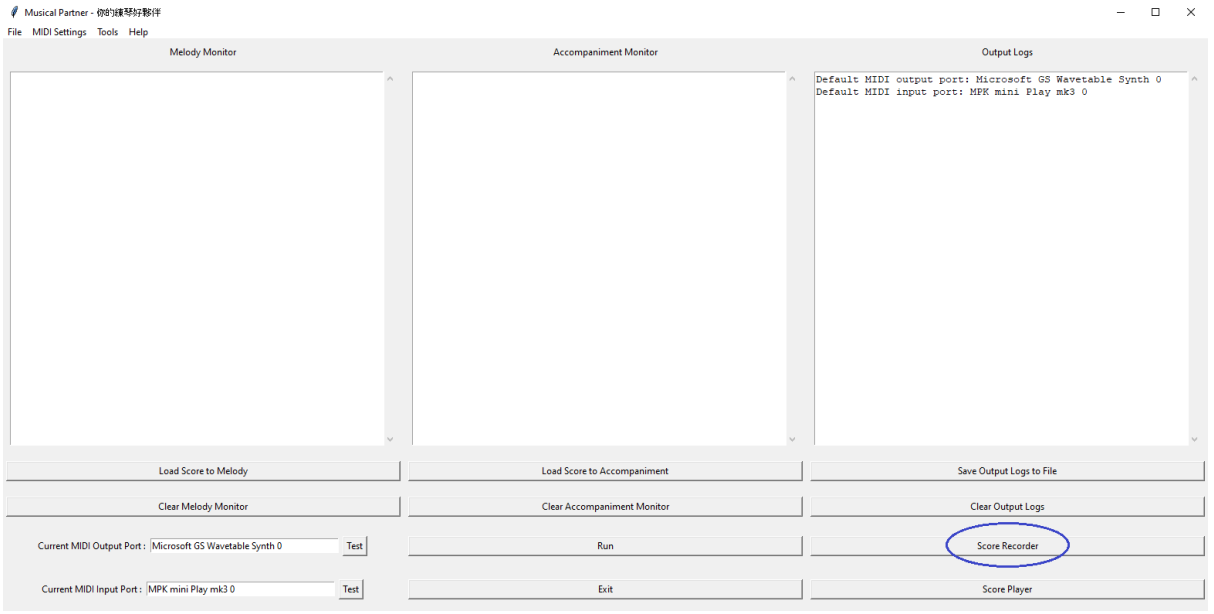


   c. 重置 MIDI 裝置步驟如下：
      主功能表 → MIDI Settings → Reset MIDI Settings



9. 使用 Score Recorder

點選「Score Recorder」按鈕後，會出現「Score Recorder Terminal」



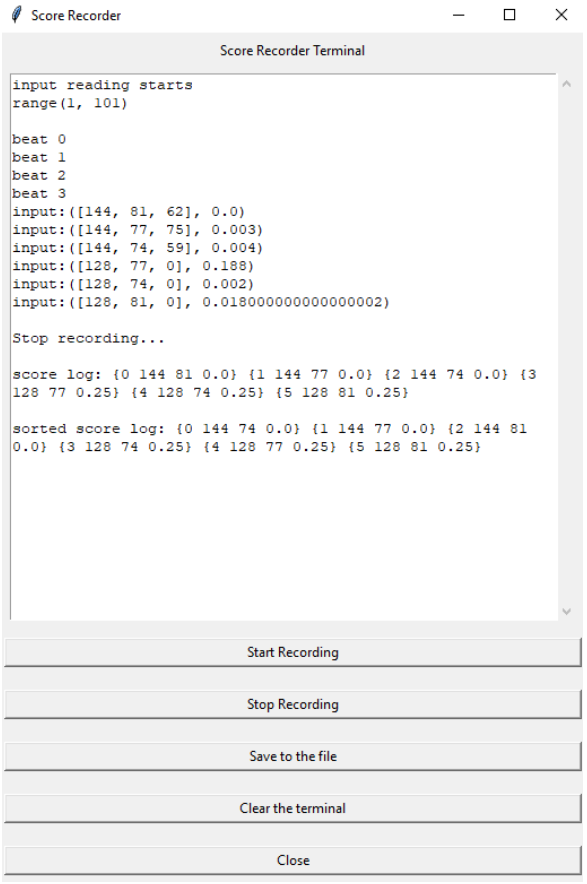
點選「Start Recording」按鈕，聽見節拍器聲音後可開始彈奏錄製 MIDI 資訊

點選「Stop Recording」按鈕可停止錄製

點選「Save to the file」可將錄製的 MIDI 訊息存成純文字的樂譜格式檔

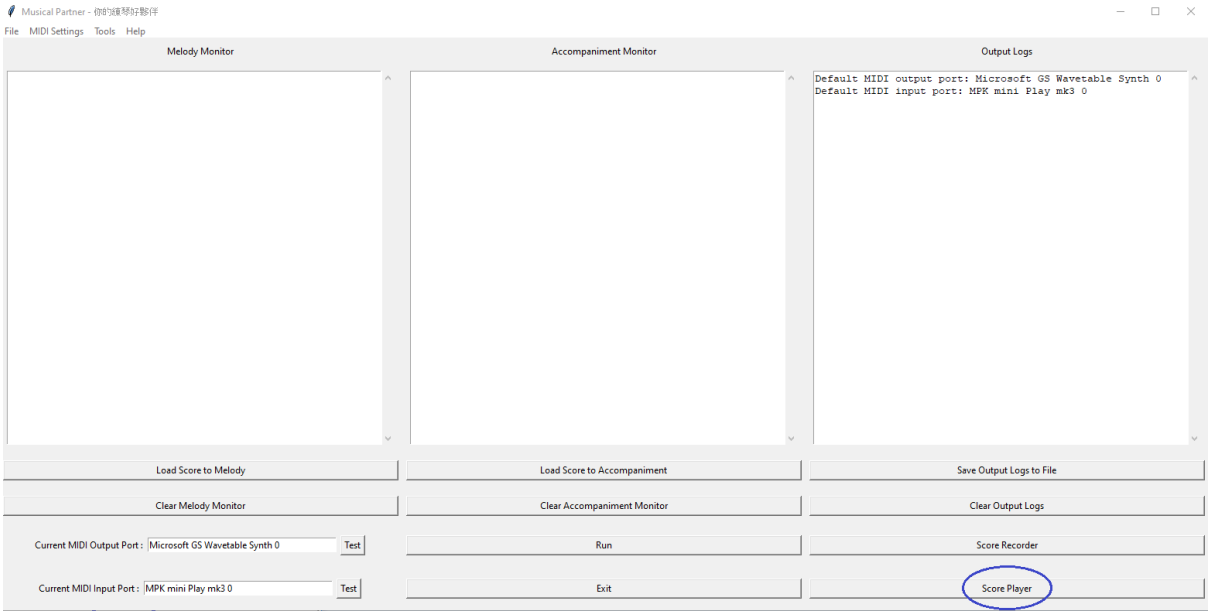
點選「Clear the terminal」可將視窗清空並重新錄製 MIDI 資訊

點選「Close」可關閉 Score Recorder



10. 使用 Score Player

點選「Score Player」按鈕後，會出現「Terminal 1」與「Terminal 2」



點選「Open File to Terminal 1」按鈕，將待播放的樂譜載入至「Terminal 1」

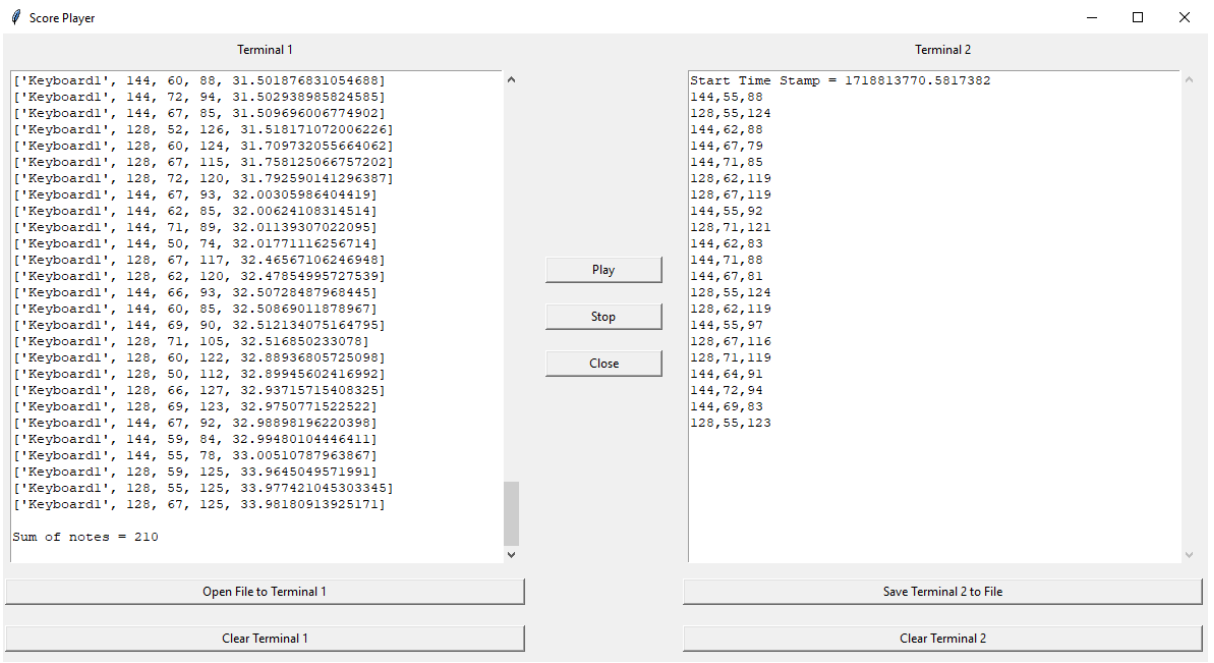
點選「Clear Terminal 1」按鈕，可將「Terminal 1」資訊清除並重新載入樂譜

點選「Play」按鈕，即開始播放樂譜，已撥放的音符將會顯示至「Terminal 2」

點選「Stop」按鈕，即可停止播放

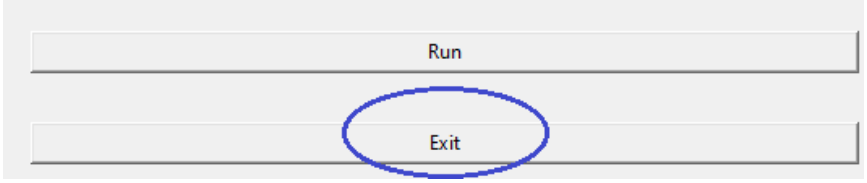
點選「Save Terminal 2 to File」按鈕，可將「Terminal 2」的資訊存成文字檔

點選「Close」按鈕，即可關閉 Score Player



11. 關閉程式

點選「Exit」按鈕，即可離開主程式



遇到的困難與未來展望

困難：

1. 因為我們的錄音與伴奏系統都是使用 thread 實作，在整合的時候需要特別注意 multi-threading 會衍生的各種問題（例如：deadlock、starvation等等）。

2. 我們原本是以獨立的 python 檔來寫 recorder 與伴奏系統，後來因為 thread 的運行順序問題，沒辦法達到我們預期的效果，所以我們嘗試過許多方法，像是subprocessing、將recorder寫成class的形式再import…等。雖然這樣做的確能以分開的檔案寫進 GUI 中，但是程式只能在功能完成後才一次把資訊都顯示在 GUI 的 window 中，無法即時顯示所有資訊（i.e. 使用 recorder 時，錄音過程中無法看到自己彈了甚麼）。這樣做並不是我們所期望的，最後我們只好把所有功能都寫進 GUI 的檔案中，但如此的程式碼非常不好維護與更新。

3. 目前使用的RtMidi函式庫只能在 Windows 平台運行，在 macOS 會發生找不到 MIDI port 的問題，從系統的錯誤訊息看到的是 RtMidi 在 macOS 無法初始化，此部分目前正在排除中。

未來展望：

1. 如同前面所提及，因為種種因素，我們選擇 Tkinter 作為開發的工具。但是其創造出來的介面風格有些單一（雖然可以透過其他 module 更動配色），之後可能可以嘗試不同的 python GUI 開發工具來增進美觀。

2. 未來也許可以結合機器學習 / 深度學習來實現電腦的自動（最佳化）伴奏，而非像現在仍然需要我們手動輸入音檔的資料。

3. 希望能將每個部份各自寫成獨立的檔案，再以import的方式整合進GUI內，方便之後的維護與更新。目前我們想到的方法是可能可以把 recorder、伴奏系統寫成 class 的形式，然後再 import 進 GUI 內，透過 inheritance override 的方式，將原先 print_to_terminal 的輸出位置改為 GUI 的 window，這樣或許就能夠以獨立檔案的方式來完成所有功能了。

4. 目前伴奏系統的即時性還有待加強，特別是在 Windows 平台運作的情況下，會發現 MIDI Output 延遲特別嚴重，這個部分我們預計將 MIDI Output 的 Thread 程式移植至一個單晶片嵌入式系統來實現，電腦只負責接收 MIDI 訊息和自動伴奏的計算，嵌入式系統透過和電腦同步時間的方式來輸出 MIDI 訊息，這樣MIDI Output 就不會被電腦在計算時使用大量的資源所延遲，此單晶片嵌入式系統已用 Arduino 打造雛形，目前正在系統整合與測試的階段。

程式自訂函數功能簡介

def print_to_terminal(message, terminal):
將文字訊息輸入至指定的文字方塊 (Scrolledtexts)

def clear_terminal(terminal):
清除指定的文字方塊 (Scrolledtexts)內容

def open_file_to_terminal1():
開啟純文字樂譜檔輸出到 Terminal1 (Melody Monitor):

def open_file_to_terminal2():
開啟純文字樂譜檔輸出到 Terminal2 (Accompaniment Monitor):

def save_terminal_output_to_file(terminal):
將指定的文字方塊 (Scrolledtexts) 內容存成檔案

def open_score_recorder():
開新視窗並啟動 Score Recorder（此部分程式由 NTHU AHG 實驗室所開發）

def open_score_player():
開新視窗並啟動 Score Player

def midi_output_select():
設定 MIDI 輸出裝置

def midi_input_select():
設定 MIDI 輸入裝置

def midi_output_test():
執行 MIDI 輸出裝置測試

def midi_input_test():
開新視窗並執行 MIDI 輸入裝置測試

def midi_reset():
MIDI 裝置重置

def midi_init():
MIDI 裝置初始化

def run_real_accomp():
執行自動伴奏程式（此部分程式由 NTHU AHG 實驗室所開發）

def show_about_dialog():
顯示程式名稱、程式版本與版權資料

參考文獻

[1] O. Heggli, J. Cabral, I. Konvalinka, P. Vuust, and M. Kringelbach, “A Kuramoto model of self-other integration across interpersonal synchronization strategies,” PLoS Comput. Biol., vol. 15, no. 10, e1007422, 2019.

[2] K Armstrong, JX Huang, TC Hung, JH Huang, YW Liu, “Real-time piano accompaniment using Kuramoto model for human-like synchronization”, 16th Int. Symp. Computer Music Multidisciplinary Research, 744-747

[3] Kit ARMSTRONG, Tzu-Ching HUNG, Ji-Xuan HUANG, and Yi-Wen LIU, “Real-time piano accompaniment model trained on and evaluated according to human ensemble characteristics,” accepted by Sound and Music Computing, Porto, Portugal, 2024.