臺灣二 OO 四年國際科學展覽會

科　　別：電腦科學科

作品名稱：音材施教--簡易音高辨識程式

得獎獎項：儲備作品

學　　校：臺北市立第一女子高級中學

作　　者：李雨霜、陳昀詩

作 者 簡 介

我是陳昀詩，目前就讀於台北市立第一女子高級中學數理實驗班，我的興趣很廣泛，不論是文學或是理工的領域都不排斥，目前是學校合唱團的一員。在小學的時候就曾經參與學校網站，以及各班畢業光碟的製作，我很喜歡電腦繪圖、電腦音樂以及製作網頁。上了高中後開始接觸 C 語言，也開始寫了不少程式，目前正在進行的專研是有關電腦音樂的研究，並且參與了崇有基金會的「中學生參與科學研究獎助計畫」，也得到許多新的體驗，獲益良多，感謝指導老師黃芳蘭，以及台大電機系所的鄭士康教授所給予我們的幫助，讓我學到了許多課本上讀不到的經驗和處世智慧。

我是李雨霜，就讀於台北市立第一女子高級中學高二數理實驗班。

自從國中擔任資訊股長接觸了網頁設計，便開始學習如 Flash 等的網頁媒體工具；等到上了高中進入了資訊專研，更是與程式語言結下了不解之緣。在用他人

已經寫好的程式外，設計些屬於自己的小程式，並且讓別人使用，是很愉悅的經驗。

在這過程中，要特別感謝台大電機所的鄭士康教授，黃芳蘭老師跟資訊專研的同學們。往後，我會更加努力。

摘要

中文摘要

我們製作了一個音準練習程式：使用者輸入聲音後，經由音頻辨識方法求出其頻譜中最高振幅之頻率，以之爲音高，再將其與目標音高相較，得到其誤差率及走音程度。此外還可發出對應的鋼琴及正弦波的聲音，方便使用者校音。

文中說明音頻辨識的方法，一些關於音樂的基本知識，微軟公司的 wave 檔格式，及此系統之應用。我們使用 FFT 辨識頻率，且將針對此部分演算法做簡單的說明，並探討如何達到所需之頻率準確度，及如何以較高效率辨識。目前誤差率已可達到 1Hz 以下，判斷時間也在 1 秒之內。

儘管國內外也有一些具備相似功能之音樂編曲軟體，但其功能十分繁複，使用者常需花費數週時間學習，且價格常高至數千元，非一般使用者所能負荷。而這個程式不但使用方便，功能簡單，容易上手，且不需任何費用。

Abstract

We have developed a singing-practicing program: after input the sound, we judge its pitch from the corresponding spectrum; then we compare it with the selected one, and output the deviation, so that the users can see if they had been out of tune. Also, we provide the sound of the piano and the sine wave sound of the chosen pitch, which can help users get the right pitch.

This report will briefly introduce the method of pitch recognition, some basics of music, and Microsoft's wave file format. In addition, we explain the application of this program. We use FFT to transform a sound wave into the spectrum, which are briefly explained in the article, too. Also we'll discuss how to improve the accuracy and efficiency of the transformation. So far the deviation is less than one Hertz, and the recognition takes less than one second.

Though there have been some commercial software with similar functions, they are often complicated to use, and cost a lot. This is not affordable to most users. On the contrary, our program is not only convenient and easy to use, but also has a simple user interface. What's more, it costs no money!

壹、緒論

一、研究動機

那時正當高一下學期之班級合唱比賽期間，練習過程中，時常有人走音，且走音者

往往無法發現自己走音，或即使發現，也無法自行校正。所以我們想到設計一個程式來幫助有心練習者增進音準。

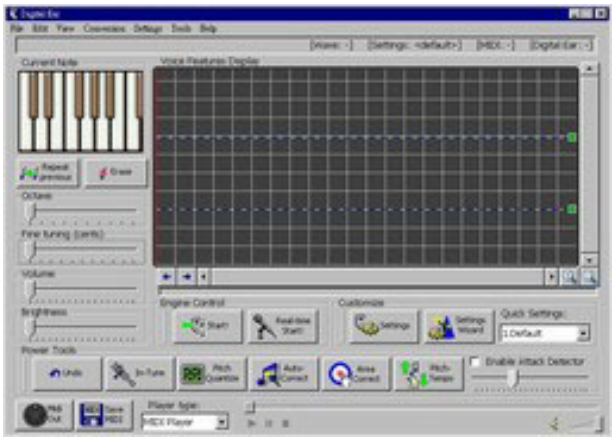
二、研究目的

我們希望設計一個程式，讓使用者輸入聲音之後，經由程式的轉換，可告知使用者所輸入聲音與所選定之音高有無偏差，若有則輸出偏差程度。除此之外，使用者也可以選擇不同之音高，並聽其對應的鋼琴或正弦波聲音，以方便校對音準。

三、文獻探討

在進行這個研究之前，我們必須先了解目前相關的程式有哪些，以及這些程式的使用特性與著重的功能，如此才能避免與其重複及缺乏創新。以下是我們找到的一些相關的軟體：

1. Digital Ears：



[圖 1.1]Digital Ears 程式操作畫面

這個程式能讓使用者的電子音樂檔案就像是電腦中的樂譜一樣，可以使用 MIDI 檔案隨時編修裡面的各種資訊，例如變更樂曲的速度，或是變更某個演奏樂器等。Digital Ear 是一套可以將既有或現場的錄音轉換爲 MIDI 檔案的程式，能將所有的錄音都變成 MIDI 檔案供編輯之用。

2. Blaze Media Convert：

[圖 1.2]Blaze Media Convert 程式操作畫面

Blaze Media Convert 是個功能眾多且強大的多媒體轉檔工具，從影片、影像圖片到聲音共可以轉 70 種以上的檔案格式，並且支援多種互轉，幾乎所有的動作都是一個步驟就完成，並且支援 CD Recorder 燒錄功能，讓你的創意和作品完整的被備份下來。

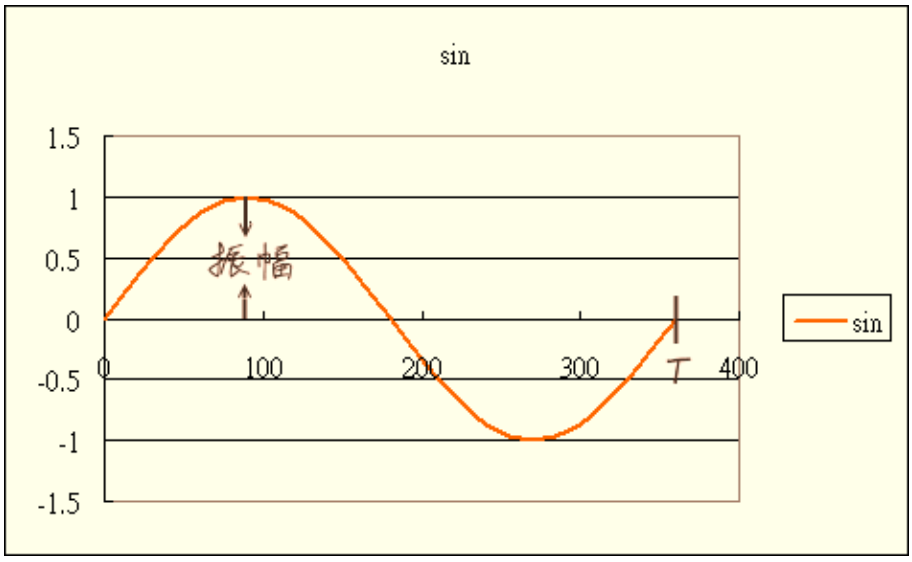
在比較過許多市面上的程式後，發覺其多功能繁複，而且大多是偏重在電腦編曲，樂曲製作方面，而較少偏重在音準的辨識上。

另外在這個程式中所用到的演算法，核心技術爲 Discrete Fourier Transform (DFT) 及 Fast Fourier Transform (FFT)，後者其實是前者的改良辦法，利用這兩個演算法作頻譜轉換，我們能找出一段聲音訊號的頻率，亦及這段聲音的音高。

貳、研究方法及過程

一、音高辨識所使用到的演算法

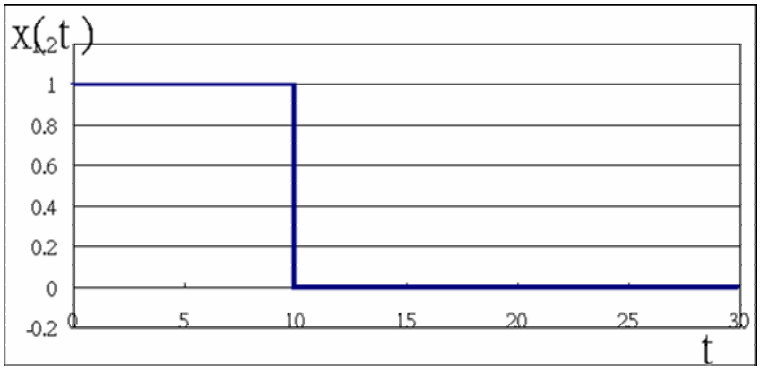
在進行研究之前，我們還需要樂曲中的許多要素以及其在電腦上的表達與操作方式，首先不可或缺的便是音高[註 1]：音高即聲音的高低，取決於物體振動的快慢。視振動頻率穩定與否而產生固定音高及不定音高，像鋼琴及各樂器的音即是固定音高；而打擊樂器大部分則產生不定音高。各種音高間的變化，便產生了旋律。另外與音高轉換息息相關的就是聲音訊號(Time domain)與頻譜(Frequency Spectrum)：聲音訊號聲音訊號是由許多正弦波組成，也具有正弦波的許多數學及物理特性，所以我們可以依其數理特性來對這訊號做分析與判斷。我們用以時間爲橫軸，能量爲縱軸的圖形來表示聲音的波形圖，在這樣的圖形中，波的高度爲其振幅，週期的倒數爲其頻率，波的形狀爲其音色。下圖[圖 2.1]爲一個基本正弦波的特性：



[圖 2.1] 正弦波特性與音高特性解說

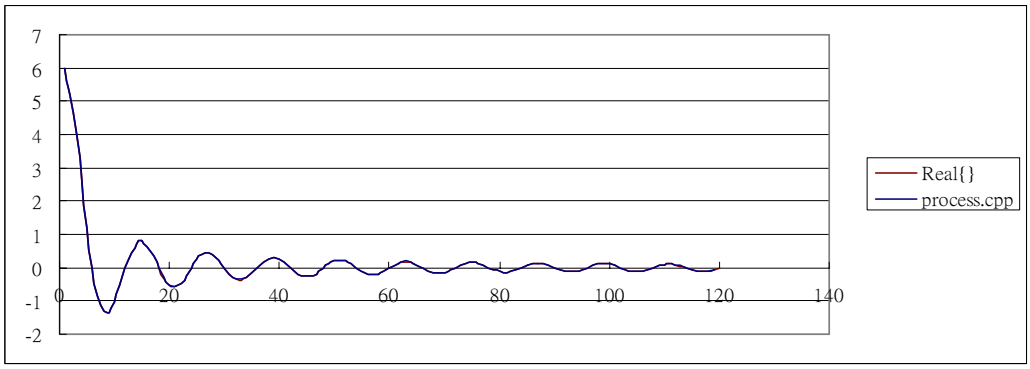
音訊號經過傅利葉轉換之後，得到以頻率爲橫軸，振幅爲縱軸的圖形，而在許多情況下，最高振幅所對應的頻率即爲基頻[註 2]。以下是一個方形波訊號經過頻譜轉換後所得到的結果範例：

1．以方形波爲例



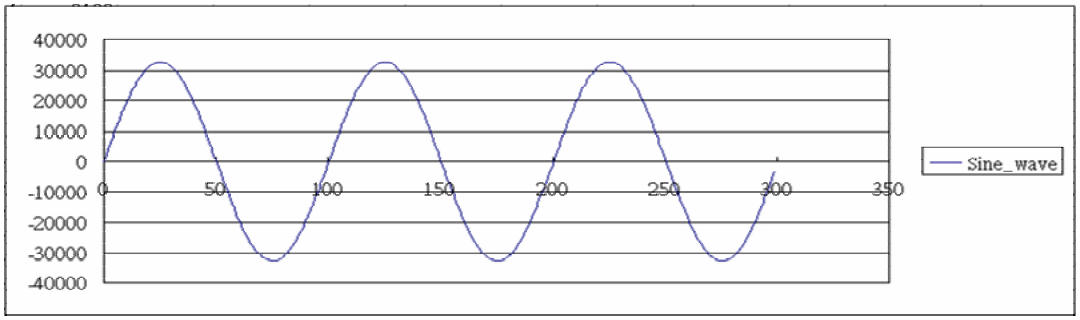
[圖 2.2] 方形波聲音訊號圖

↓Fourier Transform



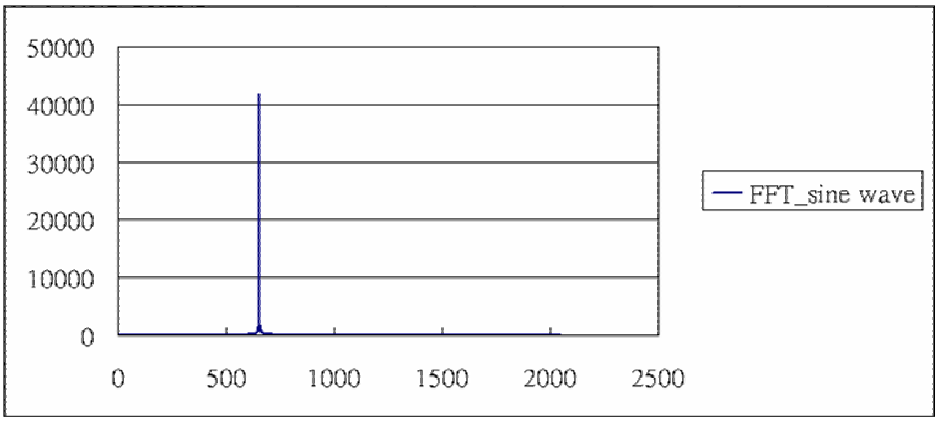
[圖 2.3] 方形波之頻譜圖

2．以正弦波爲例



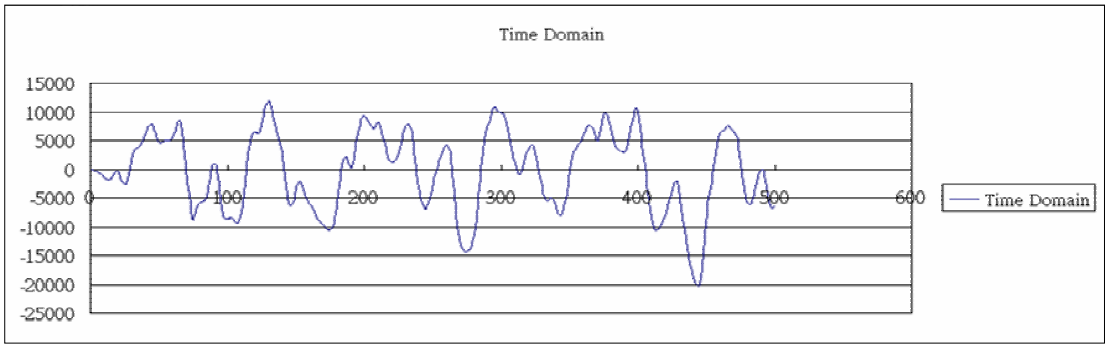
[圖 2.4] 正弦波聲音訊號圖

↓Fourier Transform



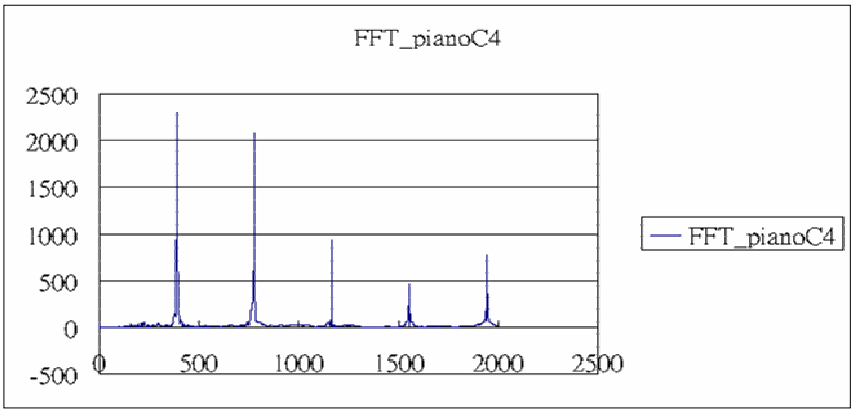
[圖 2.5] 正弦波頻譜圖

3.以鋼琴音爲例



[圖 2.6] 鋼琴聲音訊號圖

↓Fourier Transform



[圖 2.5] 鋼琴音頻譜圖

傅利葉轉換〈Fourier Transform〉

最初的聲音訊號以波形圖紀錄，亦即 X 軸爲時間，Y 軸爲信號強度，也稱爲音壓。而音高辨識則必須對數位訊號作分析，因此經由**傅利葉轉換**爲頻譜(spectrum)，亦即 X 軸爲頻率，Y 軸爲信號強度的圖形。聲音訊號轉換成頻譜後，在某些頻率位置會有高峰，代表此處能量較

爲集中；而其中的最高峰所對應的頻率通常便是基頻，也就是該音的頻率。

一開始我們使用**離散傅利葉轉換**(Discrete Fourier Transform；DFT)，它的公式如下：

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{kn} \quad \text{其中 } k=0,1,\ldots,N-1;\ n=0,1,\ldots,N-1$$

亦即

$$X[k]=\sum_{n=0}^{N-1}(R_e(x[n]))\,\mathrm{Re}(W_N^{kn})-\mu_m(x[n])\,\mu_m(W_N^{kn})$$
$$+j(R_e(x[n])\,\mu_m(W_N^{kN})+\mu_m(x[n]R_e(W_N^{kN})$$

$$k=0,1,\ldots,N-1$$

其逆轉換之定義則爲

$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X[k]W_N^{-kn}$$

（ 而由於某些數學特性，離散傅利葉轉換在計算上可以有取巧的地方，這也就是所謂的快速傅利葉轉換(FFT, Fast Fourier Transform) ，此點稍後再做討論。 ）

離散傅利葉轉換可以將信號由時間關係訊號轉換到頻率關係圖，即是將波形圖轉換到頻譜(Spectrum)；而離散傅利葉反轉換則可以將信號由頻譜轉換爲時域圖，這一點可從傅利葉轉換的積分式來看，在積分式中時間的數值及單位，由於代入積分的上下值而消去，因此即變成了頻率單位，而傳利葉反轉換則反是。

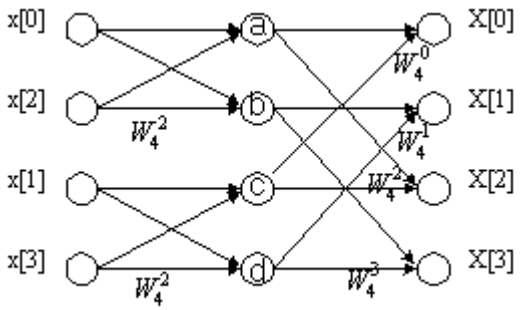
離散傅利葉轉換直接計算之演算法如下：

```
Define Frequency = 44100
FOR M=0 to Frequency {
    SUM = 0
    FOR N = 0 to Frequency {
        ARG = 2*PI*N*M/Frequency
        C = Complex (cos(arg),-sin(arg));
        SUM + = ARRAY[N]*C
//ARRAY[] 爲一個紀錄時間訊號資料的陣列
    }
        SUM*= DELTA_T
```

```
// DELTA_T 爲 PI/Frequency
        OUTPUT SUM.real() , SUM.imag()
}
```

傳統的 DFT 做轉換時，處理速度過慢，但是若使用快速傅利葉轉換 (FFT)便可節省許多時間。由於不斷有各種新的快速傅利葉轉換的計算方式被開發出來，目前快速傅利葉轉換的樣本個數已不限只是 2^n 個點，例如在 MATLAB 中，其快速傅利葉轉換是求出最接近樣本個數的 2^n 個做快速傅利葉轉換，其他則套入別種計算方式分別求出再與前者合併，而若樣本個數是 2^n 個點的話，花費的計算時間將節省許多。

經過推導發現：離散傅立葉轉換的複數運算中有不少一再重複的運算項次，因此可利用一種網狀的流程圖來運算，這也就是所謂的蝴蝶圖 (butterfly)，例如一個有四個元素的信號，其快速傅利葉轉換之蝴蝶圖如下：



有四個元素信號的快速傅利葉轉換之蝴蝶圖

系統設計（ System design ）

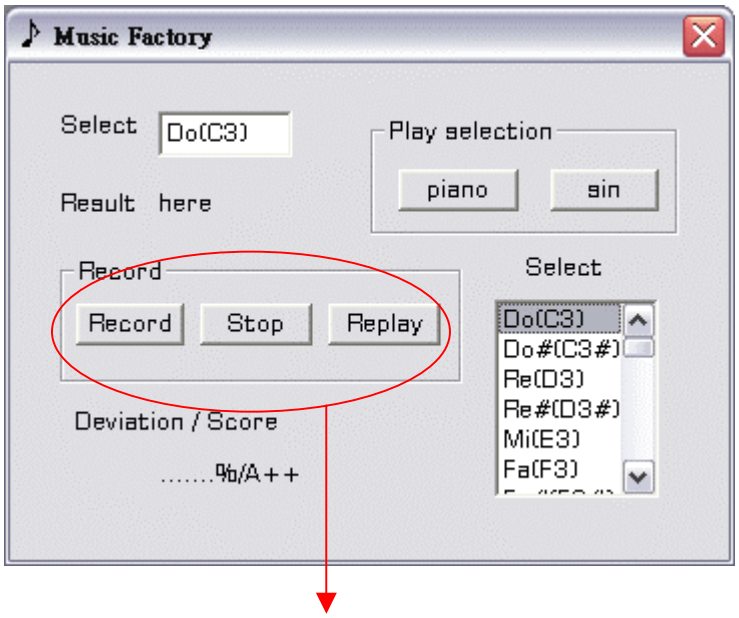
整個程式的使用及運作流程如下所示：

按下 Record 按鈕 → 開始錄音 →（系統內部動作：存檔並**辨識音高**）

→ 按下 Stop 按鈕 → 輸出結果

專案可分成三個部分：錄音、辨識以及比對；辨識的演算法在上一章已經提過，我們使用了**快速傅立葉轉換**來判別音高。至於錄音與辨識，我們則嘗試了多種方法來實做：

錄音程式



錄音程式必須盡量做到原音重現，也就是不失真。於是我們採用了與 CD 相同的取樣頻率，也就是每秒取樣 44100 個聲音訊號，而經過計算〔註三〕後，我們知道轉換的誤差將**低於 1 Hz**。

在決定了取樣頻率後，我們查詢 Wave 格式表〔註四〕來做出檔頭，之後將儲存聲音的陣列寫入.wav 檔，便完成了錄音的動作。

將聲音存入陣列，我們引入了 winmm.lib 函式庫，並運用其中的 WaveIn()函數部分來進行實做： 以 waveInOpen 來引入麥克風所輸入的聲音資料，並用 SetBuffer()將資料存入陣列中，而當使用者按下 stop 按鈕時，waveInClose()函數執行，停止資料的輸入。

而當聲音已經被存入陣列後，我們依照需求產生檔頭，在此我們嘗試了兩個方法，第一個是自己撰寫結構來定義檔頭，這個方法只需要單純的開讀檔，程式相當簡單而容易改造；但實際上 MFC 中已經有了相關的結構來幫助我們撰寫 Wave 檔。

下面便是我們自己定義的檔頭，LENGTH 代表檔案的長度，由於我們的取樣頻率是 44100，所以當檔案長 1 秒鐘時，LENGTH 的值便爲 44100。

在 Console 版的時候，我們便是運用這種方式來輸出檔頭，但等到我們將程式視窗化後，我們決定使用 MFC 的函式。

```
// 檔頭結構定義
wavHead.sRIFF = 0x46464952; // “RIFF”
wavHead.fld1    = LENGTH + 36;
wavHead.sWAVE = 0x45564157; // "WAVE"
wavHead.sFMT    = 0x20746d66; // "fmt "
wavHead.fld2    = 0x10;
wavHead.fld3    = 0x01;
wavHead.fld4    = 0x01; // mono
wavHead.fld5    = 44100;
wavHead.fld6    = 88200;
wavHead.fld7    = 0x02; // 16 bit mono
wavHead.fld8    = 0x10;
wavHead.sDATA = 0x61746164; // "data"
wavHead.fld9    = LENGTH;
```

同樣地，我們先含入 winmm.lib 來幫助我們從事這個工作。

在這之中，定義了以下的結構：

```
typedef struct {
    WORD  wFormatTag;
    DWORD nSamplesPerSec;
    DWORD nAvgBytesPerSec;
    WORD  nBlockAlign;
    WORD  wBitsPerSample;
    WORD  cbSize;
} WAVEFORMATEX;
```

以及 BulidFormat()函數：

```
void CWave::BuildFormat(WORD nChannels,
                        DWORD nFrequency, WORD nBits)
{
    m_pcmWaveFormat.wFormatTag = WAVE_FORMAT_PCM;
    m_pcmWaveFormat.nChannels = nChannels;
    m_pcmWaveFormat.nSamplesPerSec = nFrequency;
    m_pcmWaveFormat.nAvgBytesPerSec = nFrequency *
nChannels * nBits / 8;
    m_pcmWaveFormat.nBlockAlign = nChannels * nBits / 8;
    m_pcmWaveFormat.wBitsPerSample = nBits;
    m_buffer.SetNumSamples(0L,
m_pcmWaveFormat.nBlockAlign);
}
```

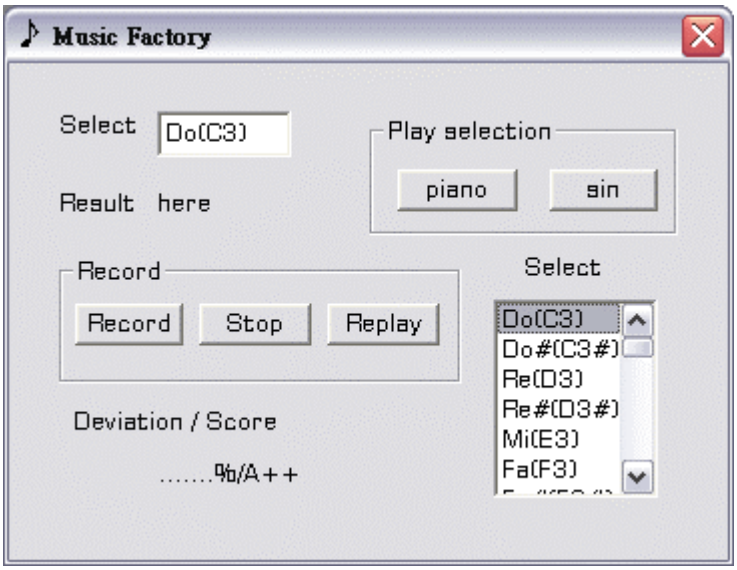
在準備完檔頭後，我們採取以下方法把 Wave 檔輸出：

```
Load length, Hz
Output header
For i = 1 to length
      WriteIn(data[i]);
Output data []
Output 00
// 00 是檔尾結束字元
```

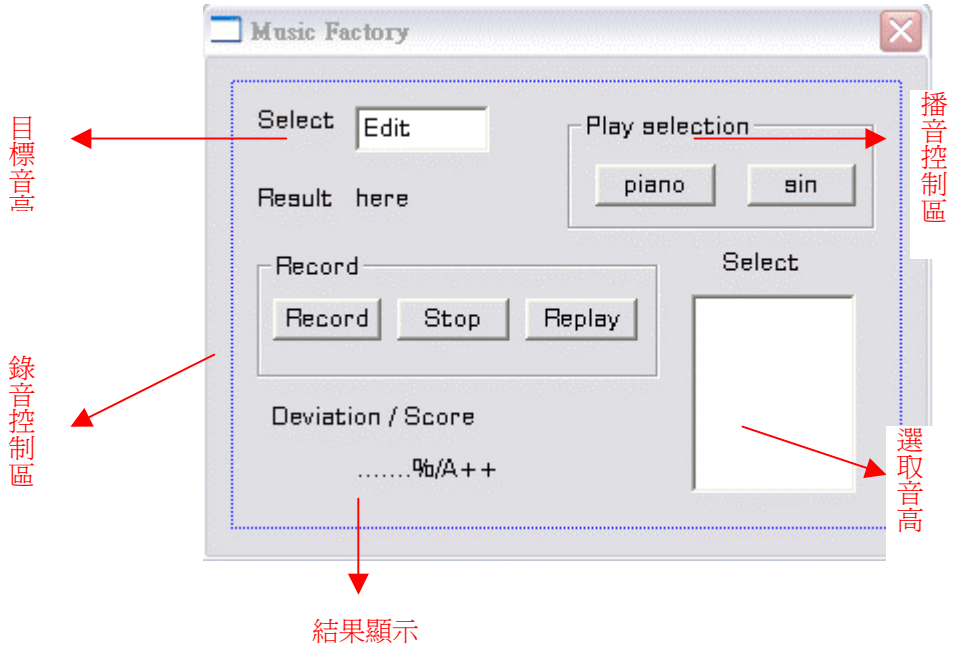
此時並同時彈出一視窗提示使用者錄音檔案的存檔名稱以及存檔成功的訊息。

♬ GUI Design

以下是我們程式的執行畫面：



接下來看看每個按鈕的功能(這是在 Visual C++ 6.0 中的樣子)：



使用流程：

首先，使用者從列式盒中選取所想要唱的音高，接著此音高將會顯示在左上方的〝目標音高〞此欄位中。而播音控制區，則是播放目標音高的聲音，piano 按鈕播放鋼琴音，而 sin 按鈕則播放正弦波的聲音。另外左下方的錄音控制區，則是讓使用者輸入自己的聲音，首先按 record 按鈕開始錄音，直到按下 stop 爲止，接著下方的結果顯示中將會出現使用者聲音跟目標音高的差距，而使用者也可以經由 replay 來聽聽自己當初唱的聲音，並可用播音控制區中的按鈕來校正自己的音高。

按鈕功能一覽：

列式盒(select)：選取音高

錄音控制區

Record：開始錄音
Stop：停止錄音(聲音輸入完成)
Replay：聽剛剛自己唱的聲音

播音控制區

piano：播放目標音高之鋼琴音
sin：播放目標音高之正弦波音

顯示資訊一覽

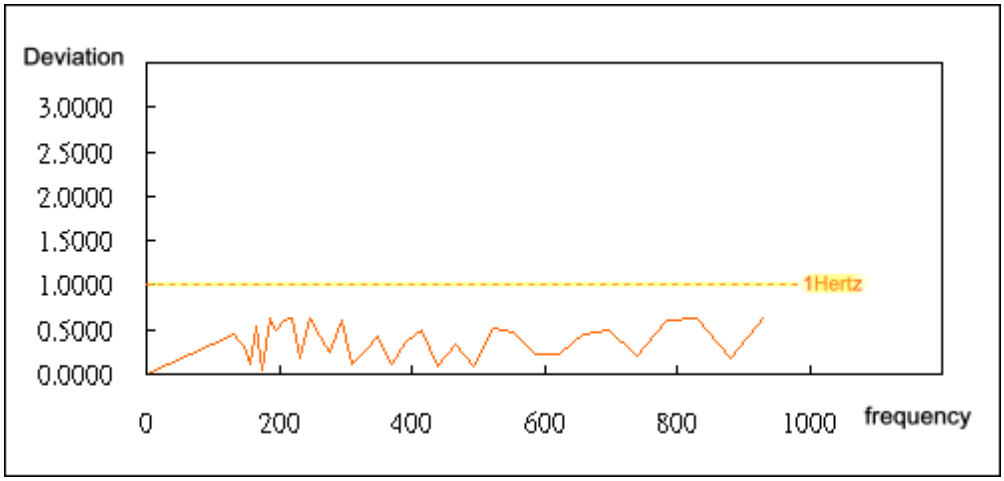
左上角的 Select：目標音高
左下方的 Deviation：使用者聲音跟目標音高的差距

參、研究結果與討論

一、音高辨識實驗

爲了了解我們的程式做音高辨識的準確度，我們必須對各個音高做辨識測試，並紀錄其誤差值。下表[表 3.1]爲比較各對頻率做 FFT 後所得之結果與原因頻之誤差：

The Following [chart] is a chart of Testing on the pitch of form



[表 3.1] 誤差比較表

照理論來說，越高音的音高辨識越能準確，低頻率的音高辨識則反之，這是因爲高頻率的聲音訊號其能量較高，較容易被偵測到其變化，而低頻率的音高辨識其聲音訊號得變化值很小，容易被忽略。由上圖可看出，誤差範圍都小於 1Hertz，所以整個轉換的過程是很準確的。

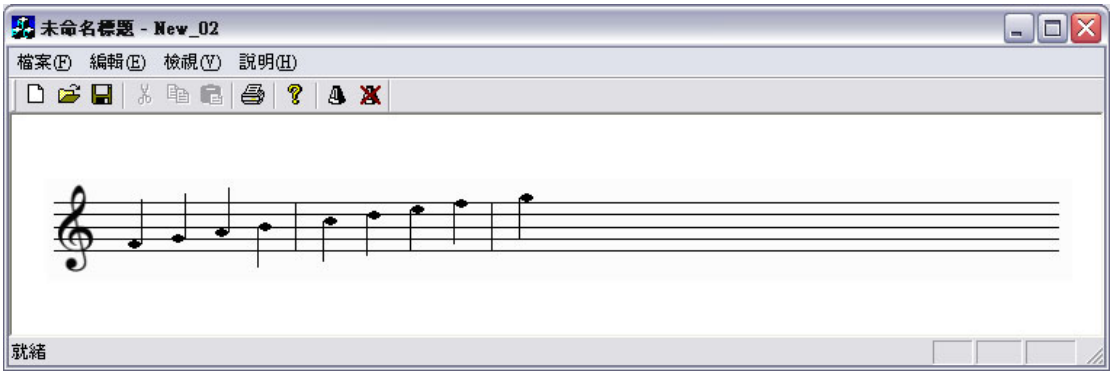
肆、結論與應用

一、結論

目前爲止，我們所能做到的功能有：錄音、播放該音之標準音(鋼琴及正弦波)、判斷該音之頻率並比對、輸出誤差率。這對於一般要尋求起音的人而言已是相當便利，而我們整個程式經由包裝後，僅佔 124KB，若再包含鋼琴音的資料，整個程式也不到 1MB。故而，的確有做到我們一開始所標榜的輕薄短小且方便使用。但是目前的功能仍不足以做到我們的研究目的，也就是讓人學習歌曲的歌唱，因此到一月之前，我們將會繼續整個計畫，並預計在一月中旬完成下一步(詳見未來展望)的動作。

二、未來展望

在接下來的研究中，我們主要會將重點放在多音辨識的部分，因爲多音辨識牽涉到節拍與換音處位置的一些較爲複雜的處理。之後我們也希望爲程式加上節拍器及樂譜顯示的功能。下圖[圖 4.1]是我們預計明年一月時的程式雛型；



[圖 4.1] 預計程式操作畫面

此程式的使用流程：

首先選擇一段樂譜，所選擇的樂譜會顯示在右邊的方框中。接著按下 “metronome”按鈕，打開節拍器，程式便會按照此曲的速度規範打拍子。按下 ” start” 按鈕，便開始錄音，使用者可聽著系統節拍器所發出之規律節奏，以合乎此曲的節拍唱入聲音。唱完後按下 “stop” 按鈕，結果顯示區便會顯示剛才輸入的聲音與該曲比對後之結果。最後可按下 ” replay” 聽聽剛才所唱入的歌聲，或者也可選擇別首曲調來練習。

按鈕功能一覽

下拉選單(combo)：選取所要練習之曲調。

Metronome：打開節拍器。

錄音控制區

Start：開始錄音
Stop：停止錄音(聲音輸入完成)
Replay：聽剛剛自己唱的聲音

顯示資訊一覽

Result：顯示比對結果。
右方圖片框：顯示樂譜。

伍、參考資料

1. 音高辨識 與 MIDI 及 Wave 之轉換與應用 /
李思毅 著 / 中山大學碩士論文(1996)
2. 離散時間訊號處理 (第二版) /
曾建誠.陳常侃 著 / 2000 年 6 月全華科技出版
3. http://www.ringthis.com/dev/wave_format.htm
4. http://www.sfu.ca/sonic-studio/handbook/index.html
5. http://cslin.auto.fcu.edu.tw/scteach/lego/mickya.htm
6. http://www.math.ncu.edu.tw/~guo/Games/menu2.html
7. http://wwwme.chit.edu.tw/~rthong/course

附註

[註 1] 音高(頻率)：聲音的頻率影響音高，在一個八度之間，依照十二平均率劃分爲 12 個音，十二平均率是由明朝的朱載育所提出，經由十二平均率的分割，每兩個半音之間的頻率差大約爲 2 的 1/12 次方，相當於 1.059463094 Hz，且每個音各有其對應的頻率 (如下圖)。

	1	2	3	4	5	6	7
C	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00
C#	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46
D	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32
D#	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02
E	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02
F	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83
F#	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96
G	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96
G#	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44
A	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00
A#	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31
B	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07

[註 2] 對於大多數的波形來說，其頻譜圖的最高振幅處所對應的頻率即是該音的音高，但有少數例外，例如號角的音色等，其最高振幅處所對應的便不是其基頻，而是其他泛音。

[註 3] Wave 檔案(Waveform Audio File)格式：

wave 檔分三區：控制訊息區、錄音資料區及文字說明區，每一區最開始有四個位元組記載著該區名稱：控制訊息區的名稱爲〝_fmt〞，錄音資料區的名稱爲〝data〞，文字說明區的名稱則爲〝LIST〞。接著是標記大小的另四個位元組，最後是此區的資料內容。

1. 檔頭訊息區塊(_fmt)

   大小固定爲 16 bytes。

   (1) 資料的格式形態(wFormatTag)：到目前爲止都是 1

   (2) 錄音軌數(nChannels) ：1 表示單音、2 爲立體音

   (3) 取樣頻率(nSamplePerSec)：每個樣本佔 1 至 4 個位元組，可爲 8 或 16 位元

   (4) 每秒平均取樣位元數(nAvgBytesPerSec)：取樣頻率 x 頻道數 x 取樣大小 / 8

   (5) 排列區間數(nBlockAligan)：頻道數 x 取樣大小 / 8

   (6) 每個樣本所佔位元(nBitsPerSample)：取樣大小，多爲 8 或 16 位元

2. 語音資料區塊(data)

   原始語音資料存放處〈立體音左聲道在前〉

   (1) 取樣大小 8 位元：所錄得的語音資料以不含正負號的數值表示

   (2) 取樣大小 16 位元：每個樣本以 2 bytes(單音) 或 4 bytes(立體音) 表示，以含正負號的數值表示。

3. 文字說明區塊(LIST)

   版權、製造日期…等，與檔案的播放無直接關係

評語及建議事項

作者著重之主題較不具創新性，此次作品完整，作者具科學精神，思考程序完整，若能將作品作延伸更佳作品，具應用價值，學術支持性質再加強，表達可再生動些。