第五屆旺宏科學獎

成果報告書

參賽編號：SA5-061

作品名稱：以碎形概念及基因交配法構成的數位音樂人機介面

姓名：林自均

關鍵字：碎形、基因交配法、數位音樂

一、研究題目

西爾平斯基船帆(Sierpinski Gasket)與西爾平斯基掛毯(Sierpinski Carpet)都屬於「碎形」(fractals)圖形的一種。「碎形幾何學」是由數學家曼德布洛(Mandelbrot)在 1960~70 年代發展出來的[1-3]，碎形會展現所謂的「自我相似性」，也就是說：不管把碎形圖形放大或縮小，它和原本的圖形都長得很相似，取一小部份來看，就像整體一樣複雜，而圖形就在越來越小的尺度裡不斷重複[4]。若代入迭代運算方程式，經由無限多次的運算，可以得到無限重覆的圖形[5]。

而基因演算法是由密西根大學的 John H. Holland 教授於 1975 年首先提出[6]，它基本上是模仿生物染色體在天擇時「物競天擇、適者生存」的概念，製造出一個環境，讓眾多染色體不斷地相互交配，再從中淘汰出最適合者，以製造出最符合需求的基因。

在這裡，我提出一些作法，找出碎形圖形其遞迴關係式，進而討論出其圖形之規律性及所涵蓋的內容與性質，著重在推廣西爾平斯基船帆與掛毯圖形的碎形概念：將碎形帶入至數位音樂。我的作法是將一段音樂曲取出，把它們看成反覆隨機迭代點，利用程式經由多次的插值運算，計算出各段音符。最後利用「基因交配法」來解決音符長短的問題，應用於碎形音樂創作，而衍生碎形概念加上基因交配的新穎應用與創新的結果。

二、研究動機

我國中畢業的時候，拿到了獎學金，就打算去書店買本書。我剛好看到一本剛出版的「一條線有多長？」的書，可能是有什麼神奇的力量或是機緣巧合，我拿起來翻了幾頁，就買下它了。

羅勃‧伊斯在這本書中說了很多個常人想不到的的數學問題[1]，例如說：「爲什麼有些聲音聽不到？／耳朵怎麼分辨出「難聽」與「悅耳」？／如何奏出好聽的組合音？／以噪音剋制噪音，真的有效？／和諧音的規則是用榔頭敲出來的？／十二音是怎麼來的？／史上最早的音階系統是什麼？／世上真有魔鬼音？／荒腔走板的歌聲也有可能是天籟美聲？」等與音樂相關的問題。

我買回家之後，看到以上這幾個問題，就被吸引住了。因爲以前我很喜歡研究關於音樂的知識和技巧，自己也曾經嘗試做過一些關於音樂的作品，像是動畫的配樂以及自己錄製的單曲等，知道做音樂的趣味性以及挑戰性，因此，我又想要進一步去探討這個主題：數學與音樂的關聯性。於是，這扇門就這樣的被打開了。

三、研究目的

目前電腦已被廣泛地運用到音樂創作上，從而增加了許多音效及使用的素材，呈現出多樣性、範圍廣、重視音響表現等面貌。本研究希望推廣基因演算法的概念，應用於數位音樂創作，而衍生新穎應用的結果。爲了避免音樂單調，在主旋律之外，又加入副旋律與伴奏，讓整體變奏音樂內涵豐富起來，而又不失整體的和諧性。

把某一段樂曲，譬如說莫扎特第 24 號奏鳴曲，將它輸入進一種演算法，結果輸出爲第 25 號奏鳴曲，這是有可能的嗎？或是，出現另一位作曲家——譬如說貝多芬的曲風？雖然聽起來有點匪夷所思，但是其中不可否認的，這是一個很有意思的想法。

音樂其實並不是很單純的東西，如果深入研究它，可以發現其中的多樣性。光是節奏的類型，它就可以分成 Blues、Swing、Waltz、Latin、Quick Steps、Rock、Disco、March 等等不同的節奏型態，而每一種型態又可以分成更細小的小塊。而在作曲方面的技巧又更多了，和聲、倚音、和絃音、裝飾音、低音聲部、弦樂聲部、打擊聲部……，簡直可以說是無遠弗屆。

但是，在這個廣闊的音樂界之中，「沒有靈感」這個惡夢還是普遍存在於作曲家的創作歷程中。作曲時腸思枯竭時，是很難過的一件事。故本研究之主要目的就是要幫助這些作曲者，引發他們的靈感，進而創造出更爲動聽的音樂，以造福更多的音樂愛好者。而且在未來，也許會因爲有了電腦音樂人機介面的輔助，會使作曲者更容易完成數位音樂創作。再更進一步，我希望未來能就此發現一條路，可以找出「好聽的音樂」與數學的直接關聯性。

四、研究過程

首先，要做迭代運算，以產生各個新的音高。迭代運算的意義爲：假設多邊形頂點、中心點或圓形等圖案特徵點爲各反覆隨機迭代點 Pm={ P1, P2, P3…..Pn}。

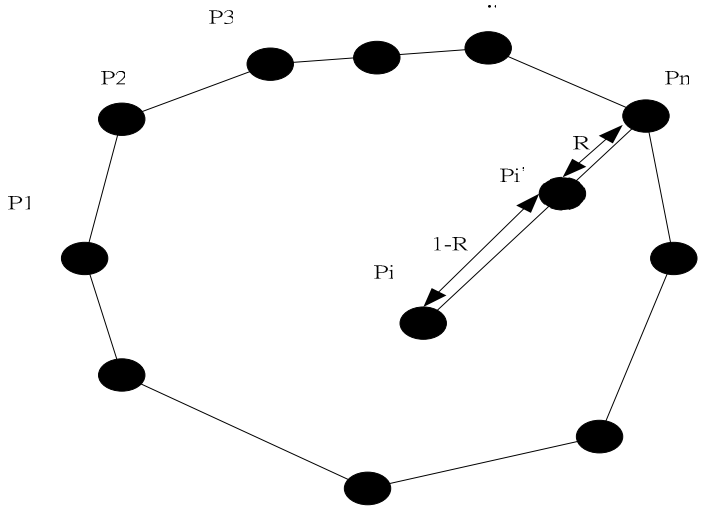


圖 1 反覆隨機迭代點 Pn 與新迭代點 Pi'

利用迭代運算系統，任取一點 Pi 出發，反覆隨機與各迭代點作插值運算，取適當的插值運算係數 R，得到新迭代點 Pi'（見圖 1）：

$$Pi' = R \cdot Pi + (1 - R) \cdot Pn$$

其中，R 爲新迭代點 Pi' 到各迭代點之距離與 Pi 到各迭代點之距離之比值。

在以上的假設及方程式爲基礎，我把它寫成 C++的程式，好讓我能了解在其中的 P 點、R 值與圖形最終的模樣，有何關聯性。如果把 P 點設爲三角形的三個頂點，R 值設爲 0.5，就可得到西爾平斯基船帆（圖 2(A)）；如果把 P 點設爲三角形的三個頂點，R 值設爲 0.5，就可得到西爾平斯基掛毯（圖 2(B)）。如果再加以其他不同的調整，就可得到其他各種不同的碎形圖形。推廣這樣的概念，我也可以將一段音樂中的音符視爲各迭代點，以迭代出新的迭代點（Pi' ），也就是新的音符。

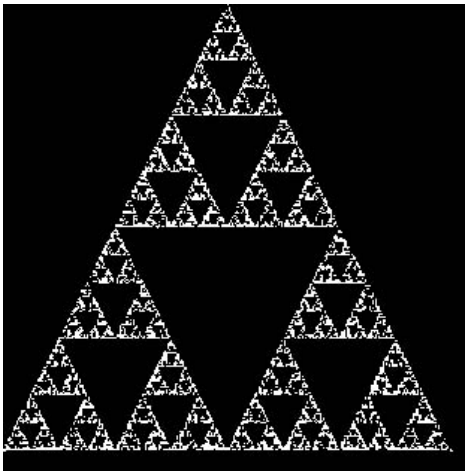
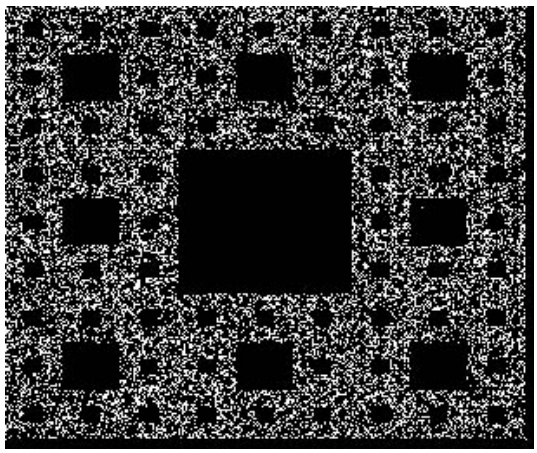


圖 2（A）西爾平斯基船帆圖形



（B）西爾平斯基掛毯圖形

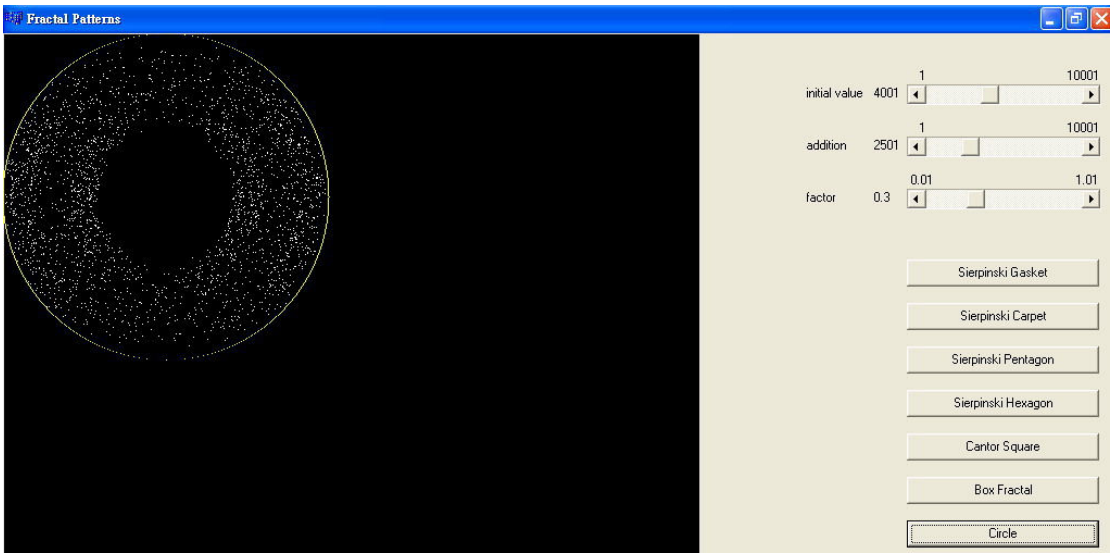


圖 2（C）迭代點排列成圓形之碎形圖案

在執行各個音符迭代運算之前，由於音樂是看不見的，較難掌握其迭代的次序與規律性。所以，我們也經由碎形圖案來了解調整多個迭代點、起始值、迭代次數與插值運算係數 R 時，所運算出來的結果。例如圖 2（C）即是迭代點排列成圓形，且起始值爲 4001，迭代次數爲 2501，插值運算係數 R 爲 0.3 之碎形圖案。有了這樣的準備工作以後，接下來我們就進行音符的迭代運算。

但是，如果光以原始的音符去迭代的話，我經由實驗發現，這樣迭代出來的新音符將會變得平淡無奇，也就是說，會變得音符起伏很小，可以說是幾乎沒有改變。所以我想出了一個解決的辦法，即把原始音符輸入後，找出它們的中心點，以這個中心點爲準，其他的音符按照一定比例向外延展，成爲新的迭代點。再利用這些迭代點，迭代出新的音符。如此迭代出的新音符就會有比較大的發展空間，於是它的可聽性也就跟著提高了。其發展流程如圖 3 所示。具體的作法爲：

1.先將最高和最低的音符(分別設爲 Max 及 Min)找出來，再找出它們的平均值(設爲 Aver)。

2.把各個音符 (n[i]) 和平均值的距離 (n[i]-Aver) 找出來。

3.將差距加大爲 (n[i]-Aver)/R，其中 R 爲迭代運算係數。

4.即可得到各個新的迭代頂點爲 (n[i]-Aver)/R + Aver。

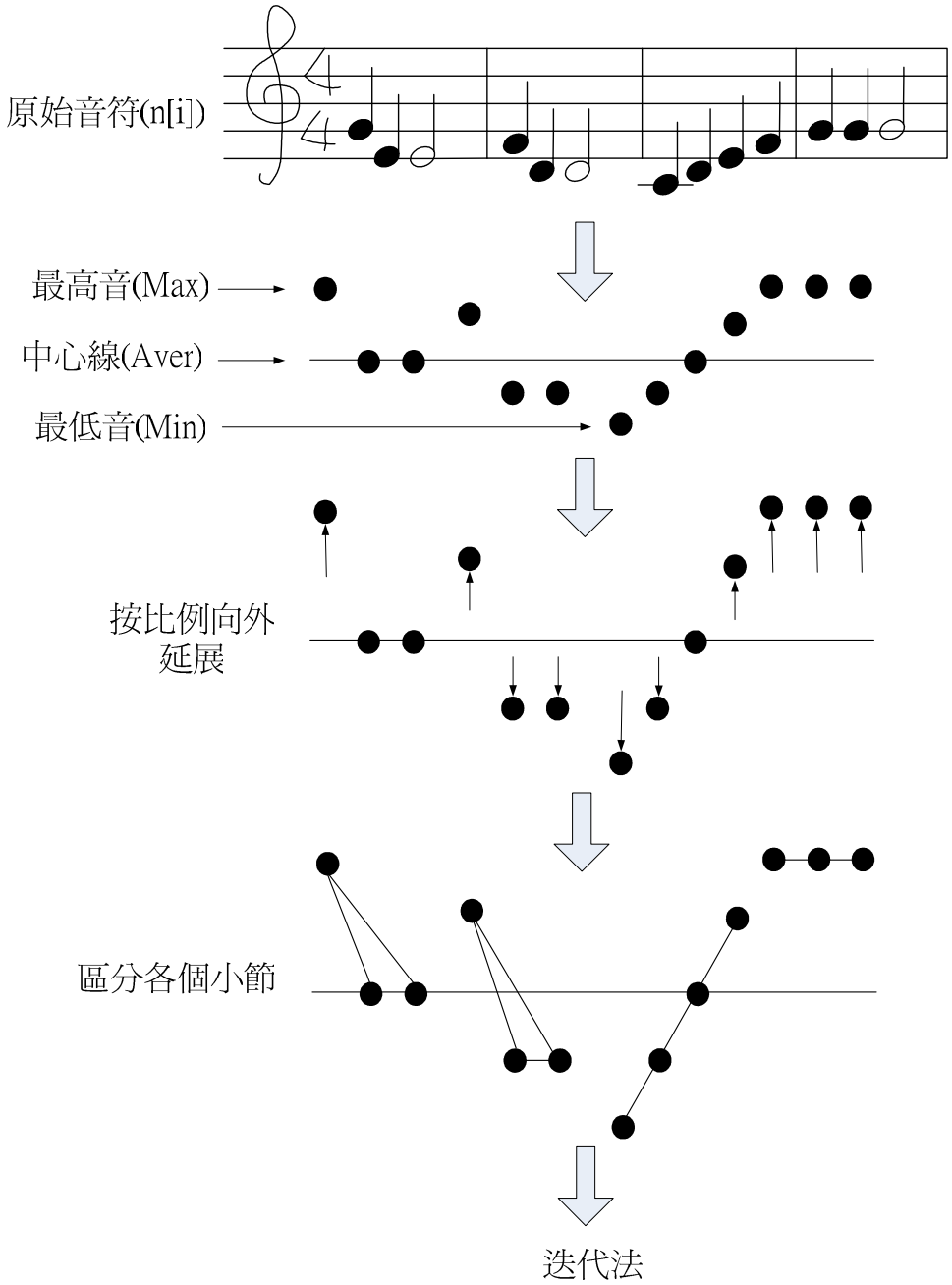


圖 3 找出中心音符將其他的音符按照一定比例向外延展

以下是一段改變音高的範例：

由「C1 C1/2 D1/2 E1 E1 D1/2 C1/2 D1/2 E1/2 C2 E1 E1/2 F1/2 G1 G1 F1/2 E1/2

F1/2 G1/2 E2」

變成「C#1 C1/2 C#1/2 -B1 C1 -A#1/2 -B1/2 -B1/2 -B1/2 C2 E1 F#1/2 G1/2 G#1 G#1 E1/2 F#1/2 F#1/2 F1/2 F2」

其中，C、D、E、F、G、A、B 各代表 Do、Re、Mi、Fa、Sol、La、Si，而其後附加的數字爲它的拍數（長度），「#」記號表示升半音，「-」記號則表示降八度。

以下爲本研究所發展出的數位音樂的迭代演算法則：

1. Notation
   R: the ratio of Iterated Function Systems
   nt: the complete note array of the piece of music
   bt: the complete beat array of the piece of music
   k: the number of notes
   p: the iterated point of each note
   bg: the gene of each beat
   sum, kk: the tools to help us to know that how many notes are there in each measure
2. Initialization
   k = the number of complete note array; sum = 0;
3. Recursion
   repeat for all
   read the source file for nt[i] and bt[i]
   if bt[i] == 60 k=k-1;
   if bt[i] == 90 k=k-2;
   if bt[i] == 120 k=k-3;
4. Recursion
   repeat for all
   find the highest and lowest note in the measure
   calculate the average value aver of highest and lowest note;
   get the value for R
   repeat for all
   if nt[i] > aver
   nt[i] = aver + (nt[i] - aver) / R;
   if nt[i] < aver
   nt[i] = aver - (aver - nt[i]) / R;

   repeat for all
   sum=sum+bt[i];
   if sum is divisible by 240
   kk[j]=i+1;

   repeat for all
   write the new beats into the new midi file

   repeat for all
   if i == 0

```
            get the random value that is from 0 to (kk[i]-1) for r
    else
            get the random value that's from kk[i-1] to (kk[i]-1) for r
    choose a random note (nt[r]) for beginning point (p)

repeat for all
        if i == 0
                get the random value that is from 0 to (kk[i]-1) for r
        else
                get the random value that's from kk[i-1] to (kk[i]-1) for r
            p = p * R + nt[r] * (1 - R);

    if i == 0
            repeat for all
                get the random value that is from 0 to (kk[i]-1) for r

                p = p * R + nt[r] * (1 - R);
                to write p into the new midi file
    else
            repeat for all
                r=kk[e-1] + rand()%(kk[e]-kk[e-1]);

                p=p*R + nt[r]*(1-R);
                to write p into the new midi file
```

但是，以上的方法所改變的音符與原曲比較並沒有節奏上的不同，所以我們要做基因交配[7]的動作。首先要製造每一個音符長短的基因，我的定義是：把一拍分成八等份，利用 8 個 x 或 y 的組合來表示，而以 y 的數量多寡來代表音符的長短（如圖 4 所示）。

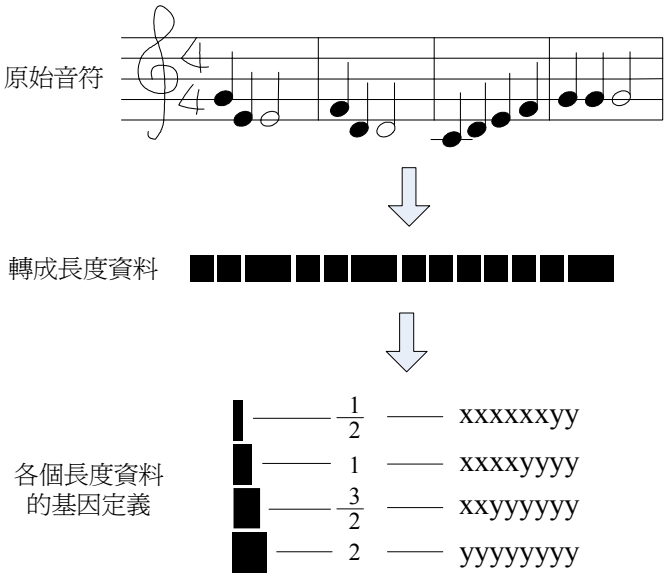


圖 4 每一個音符長短的基因之定義

例如說：

1/8 的音符，表示成 xxxxxxxy

1/2 的音符，表示成 xxxxyyyy

第二個步驟，要把交配池的隨機取兩個基因來交配，交配的方法是把各個基因的染色體(x 和 y)各自隨機取得(見圖 5)，成爲一個新的基因。

把以上這兩個例子隨機取出新的染色體，可能是：

1/8 的音符，表示成 xxxxxxxy

1/4 的音符，表示成 xxxxxxyy

3/8 的音符，表示成 xxxxxyyy

1/2 的音符，表示成 xxxxyyyy

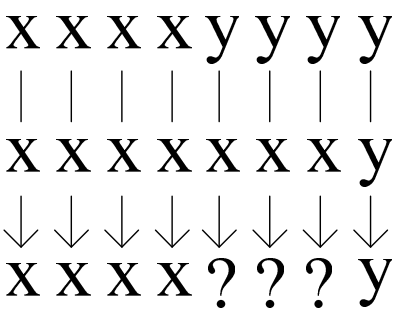


圖 5 把各個基因的 x 和 y 各自隨機取得

也許會有人說:「如果我不那麼麻煩，只是單純的在它兩個基因的範圍之間，隨便挑一個基因，那它的效果不是和交配法一樣嗎？」看似一樣，實際上是不一樣的。以機率來說，交配後的音符，它的長短介於它的父母之間的機率，會大於其他的機率。

如圖 5 的例子，出現 1/4、3/8 音符的機率爲 3/8，而出現 1/8、1/2 音符的機率只有 1/8，就像兒女的身高，介於他們父母的身高之間的機率，會比其他情況的機率來得高。因此我認爲在這個地方使用基因交配法是比較合理的一個作法。

以下是一段改變音長的範例：

由「C1 C1/2 D1/2 E1 E1 D1/2 C1/2 D1/2 E1/2 C2 E1 E1/2 F1/2 G1 G1 F1/2 E1/2 F1/2 G1/2 E2」

變成「C1/2 C1/2 D1 E1 E1 D1/2 C1/2 D1/2 E1/2 C1/2 E1 E1 F1 G1 G1 F1/2 E1/2 F1/2 G1/2 E1」

以下是一段音高、音長都改變的案例：

由「C1 C1/2 D1/2 E1 E1 D1/2 C1/2 D1/2 E1/2 C2 E1 E1/2 F1/2 G1 G1 F1/2 E1/2

F1/2 G1/2 E2」

變成「C#1/2 D1/2 D1/2 D1/2 -B1/2 -A2 -A#2 -A#2 -B2 -B2 G1/2 G#1/2 G1/2 F#1/2 F#1/2 G1/2」

以下爲本研究所發展出的數位音樂的基因交配演算法則：

1. Notation

nt: the complete note array of the piece of music
bt: the complete beat array of the piece of music
k: the number of notes
p: the iterated point of each note
bg[4]: the gene of each beat
sum, kk[4]: the tools to help us to know that how many notes are there in each measure
xy: to record the gene of each beats

2. Initialization

k = the number of complete note array; sum = 0;

3. Recursion

```
repeat for all
    sum=sum+bt[i];
    if sum is divisible by 240
        kk[j]=i+1;

repeat for all
    switch bt[i]
        case 1: to write in the gene of beat of half beat
        case 2: to write in the gene of beat of one beat
        case 3: to write in the gene of beat of one and half beat
        case 4: to write in the gene of beat of two beat

repeat for all
    if i == 0
        get the random value that is from 0 to (kk[i]-1) for r
    else
        get the random value that is from kk[i-1] to (kk[i]-1) for r
    repeat for all
        bg[j] = xy[r][j];
        repeat for all
        if i == 0
            get the random value that is from 0 to (kk[i]-1) for r
        else
            get the random value that's from kk[i-1] to (kk[i]-1) for r
        repeat for all
            get the random value that is 0 or 1 for r2
            if r2 == 0
                bg[m] = xy[r][m];

    if i==0
        repeat for all
```

```
                repeat for all
                        get the random value that is 0 or 1 for r2
                        if r2 == 0
                                bg[m] = xy[r][m];
                    bt[j]=0;
                    repeat for all
                            to count up that how many 'y' in each note and write
                           the new value into the bt[i]

        else
                repeat for all
                        repeat for all
                                get the random value that is 0 or 1 for r2
                                if r2 == 0
                                        bg[m] = xy[r][m];
                        bt[kk[i-1]+j]=0;
                        repeat for all
    to count up that how many 'y' in each note and write the new value into the bt[i]
```

另外，我在程式裡加裝了一份可以讓使用者來自行輸入音符的功能。只要選取其中任何一個音符，即可以按鈕來調整該音符的音高及音長，再按下「更新」鈕，便可完成一首自創的曲子。還可用這首曲子進行迭代運算和基因交配法，來製造出下一代的的數位音樂。由於只是進行初步的驗證工作，所以讓使用者來自行輸入的音符只限於八個音高及四種音長，未來可視需求再來擴充。

五、討論及應用

在本研究的設備需求上，使用以下軟、硬體：

1.個人電腦一台(CPU：PentiumⅢ 500，記憶體：128MB ，顯示卡：S3 savage 3D/M)，用來執行系統

2.圖像印出裝置

3.程式軟體：Borland C++ Builder 6.0

4.數位音樂創作軟體：Cakewalk 8.0

我利用 MIDI 編曲，並透過 Windows 的 MIDI 功能來播放音樂，在此討論 MIDI 檔格式如下：MIDI 它是一種電子樂器之間的通訊協定標準，是數位樂器介面（Musical Instrument Digital Interface）的縮寫[8]。MID 檔案是將音樂內的每個音符通通以文字記錄下來。就是因爲 MID 檔案只是記錄文字，而不是記錄某一時間的波形，所以 MID 檔案不會像 WAV 檔、WMA 檔、MP3 檔之類音樂檔案的那麼大。所以說如果只是純音樂的話，大家就比較常在網路上使用 MIDI。

MID 檔主要是由兩大部份節區資料所組合而成：表頭資料節區 Mthd、音軌資料節區 Mtrk 節區。在表頭資料 Mthd 節區的內容包括節區名稱、節區資料大小、MIDI 檔類型、音軌節區數目、時間基準。而在 Mtrk 音軌節區包含節區名稱、節區資料大小、節區資料。

而我在前面所提到的「將音符高低、長短改變」，在程式中就是讀取一個 MIDI FORMAT 0 檔，經過迭代法、基因演算法之後，再把它寫入另一個 MIDI 檔。因爲我要做的碎形音樂只是單音，所以在這裡要更動的東西不多，我只需要用到 Mtrk 音軌節區的節區資料的一小部份而已。

在 MIDI 檔中的 Mtrk 音軌節區，一個音符的樣子是長這樣：

ABCA

其中「A」是表示音高，「B」是表示音量，「C」是表示長度，它們都是以 ASCII 碼的數值運作的。例如一個高音 Do，音量 100，半拍的音符，它就表示成：

Hd < H

其中「H」的 ASCII 碼爲 72，代表高音 Do；「d」的 ASCII 碼爲 100，代表音量 100；「<」的 ASCII 碼爲 60，代表長度爲半拍。

本研究透過 MID 檔讀取四小節的樂曲，利用多次的插值運算與基因交配，藉此產生下一代新的碎形音樂，再寫入到新的 MID 音樂檔案中，並進行播放音樂。其中的程式流程圖以及程式執行畫面就如圖 6 以及圖 7 所示。

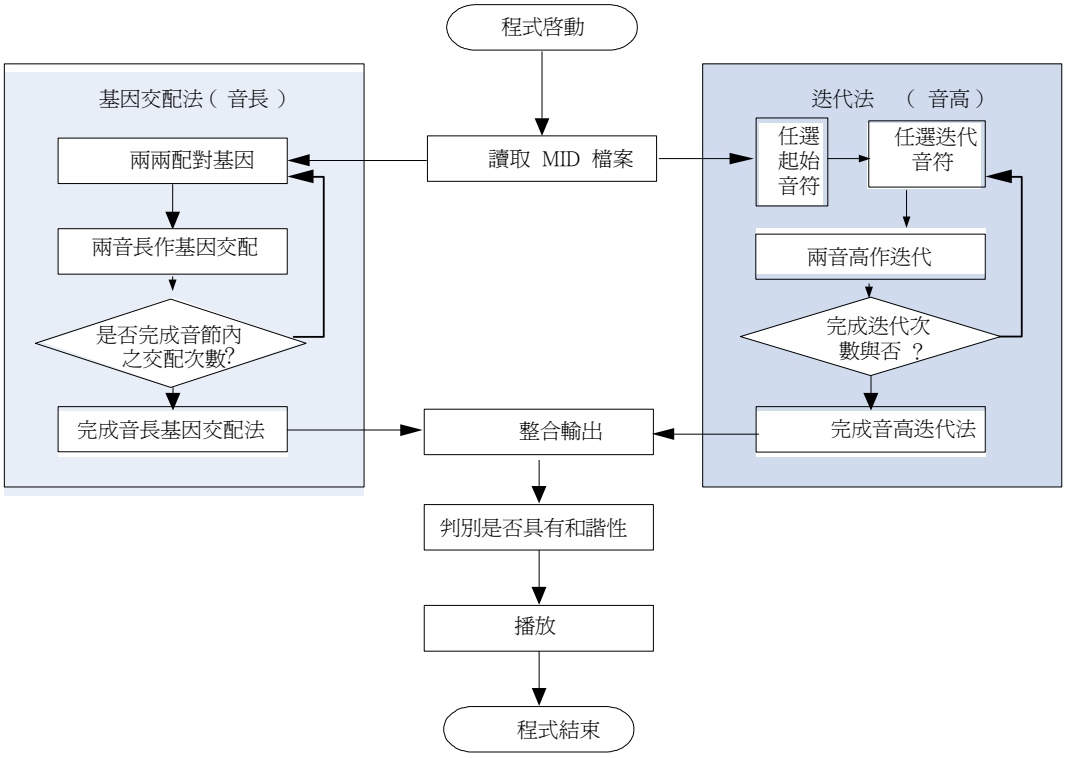


圖 6 碎形音樂產生程式流程

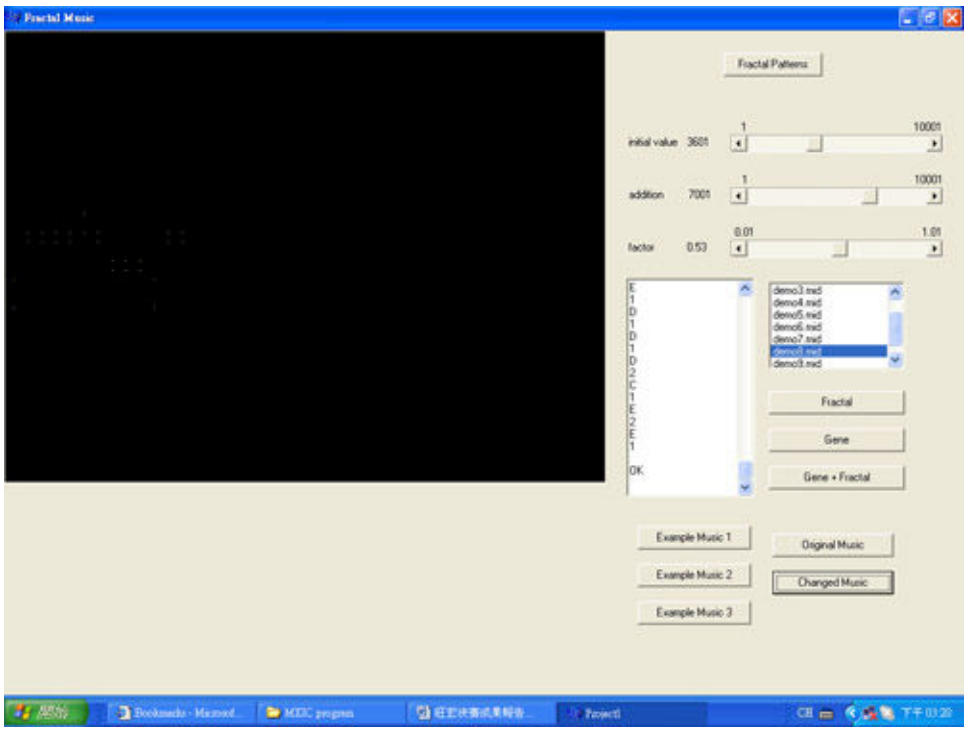


圖 7 碎形圖形及碎形音樂產生程式之畫面

利用以上的技巧產生碎形音樂，再利用數位音樂創作軟體 Cakewalk 4.0 進行編曲，如圖 8。

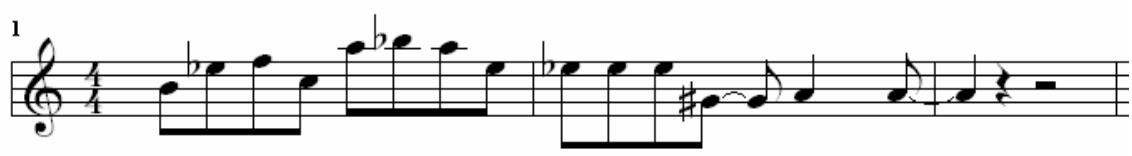


圖 8　推廣西爾平斯基船帆與掛毯圖形的概念而衍生的碎形音樂

爲了避免音樂單調，在主旋律之外，又加入副旋律與伴奏，讓整體變奏音樂內涵豐富起來(見圖 9)。

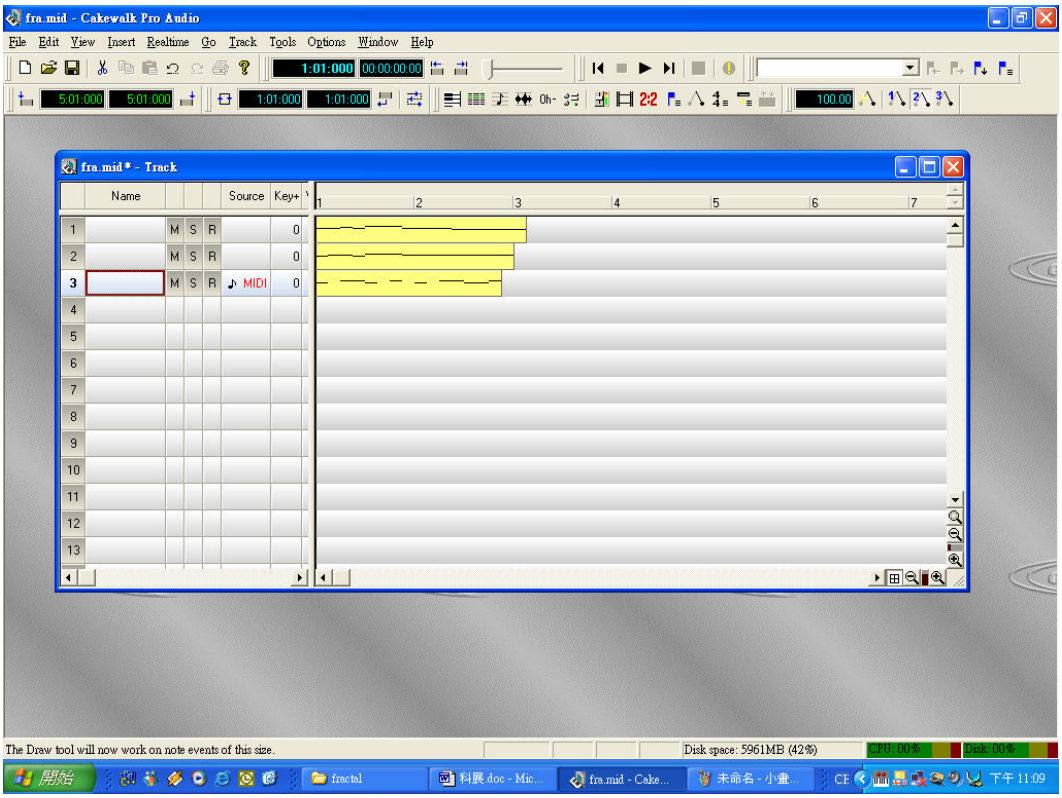


圖 9　在主旋律(Track 1)之外，又加入副旋律(Track 2)與伴奏(Track 3)

接著，我們進行起始值、迭代次數與插值運算係數 R 的調整實驗，再根據試聽的結果，發現起始值與迭代次數的改變與所產生之音樂之悅耳程度並沒有直接的關聯性。並且，我們也得到插值運算係數 R 較能產生悅耳音樂的範圍，並由程式提供建議值，其執行畫面如圖 10。除了利用 MIDI 檔案作爲音樂旋律的輸出與輸入之外，使用者也可以自行輸入音符進行編曲，而提供使用者來自行輸入音樂介面的執行畫面如圖 11。

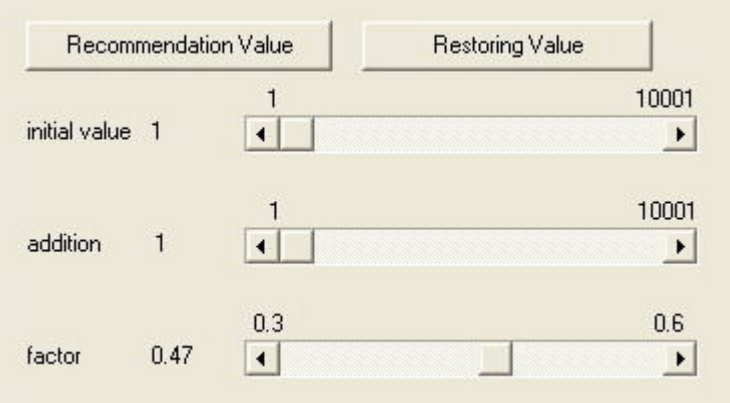


圖 10 本程式提供建議值之執行畫面

圖 11 使用者自行輸入音符之程式執行畫面

表 1 本研究之碎形音樂的試聽結果

試聽結果	難聽	普通	好聽
旋律 1	21%	40%	39%
旋律 2	12%	47%	41%
旋律 3	17%	32%	51%

初步利用設計出來的程式和處理流程，挑選三段旋律作爲實驗素材，產生碎形音樂，經由一百人次的實驗與試聽之後，結果所得數據如表 1 所示。在試聽過程中逐一地將每首碎形音樂進行是否悅耳的判斷，由表中的數據可以得知，對於本研究所實驗的旋律，利用我所設計出的碎形音樂技術重新編曲，結果「好聽」的程度皆大於「難聽」的程度。至於「旋律 1」的悅耳程度較低，根據分析，我認爲是所採用的旋律母體或是音符的長短變化量不足、或是音高缺乏變化，或是

音符數目又太少，導致迭代點不足，因而造成較爲難聽的結果。

總而言之，經過許多次的檢驗，本研究產生出的音樂之中，還是能出現較耐聽、較具有價值性的旋律，可以說本音樂產生器雖然祇是呈現「偶有佳作」的情況，但仍然可以提供激發作曲者靈感的素材，也就是說，它是真的具有實用性的。

六、結論

由於 MIDI 檔案格式相當複雜，無論是網路上或是書本上的現成程式庫，都只有提供 MIDI 檔案單純地讀取與播放的功能，並不能夠詳細讀入 MIDI 檔案內各音符的內容，也無法去更改各個音符的資料並覆寫。因此本研究花了相當多的心力於 MIDI 檔案的格式上的分析，目前所寫的程式只能夠對特定的 MIDI 格式進行操作，然而已經可以滿足原先所設定的迭代與基因運算功能。在未來，希望對任何的 MIDI 格式皆能處理，並自動產生第二、三道音軌來進行和聲、伴奏。

並且，本研究之基因演算法的部份，只做到了「產生基因」、「交配」、「挑選出最終基因」等步驟，尚有「淘汰」、「突變」等尚未完成。另外，在本研究中，程式是將原始音樂的各個音符分別迭代，再產生新一代的音符，組成新的音樂；但是想像在真實世界中，DNA 的交配並不是一個原子和另一個原子在交配，而是一段蛋白質和另一段蛋白質在做交配的動作，而這個部份對應到本研究中，即爲「一個小節和另一個小節交配」。若將來把這個部份完成後，也許應該會更爲貼近真實世界中的基因交配之實況，而產生出更爲貼近人類喜好的音樂。

綜合以上之討論，本研究未來須改善的部份可歸結出以下幾點：

一、可對任何格式之 MIDI 進行處理。
二、自動產生第二、三道等音軌來進行和聲、伴奏。
三、做出自動挑選出較悅耳音樂之「淘汰功能」。
四、於基因交配中加入「突變」的可能性。
五、交配方法加入「小節間的交配」的功能。
六、增加外接之輸入設備，例如以電子琴輸入音符、以麥克風輸入人聲……等，再加以運算，從而構成完整的人機界面。

七、參考資料

1. 羅勃・伊斯(譯者：蔡承志)，一條線有多長？生活中意想不到的 116 個數學謎題，三言社出版，2005
2. 吳文成，碎形，Fractal，http://alumni.nctu.edu.tw/~sinner/ complex/ fractals/index.html
3. Lawrence H. Riddle, Sierpinski Gasket, http://ecademy.agnesscott.edu/ ~lriddle/ ifs/siertri/siertri.htm
4. Stewart, Ian. Four Encounters with Sierpinski's Gasket, The Mathematical Intelligencer, 17, No. 1, 1995, p.52-64.
5. 廖思善，碎形的魅力，科學月刊 363，2000，p.222
6. 周鵬程，遺傳演算法原理與應用，全華出版，2001
7. Goldberg, D, E. , Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley , New York, 1989
8. 林宸生，數位信號--影像與聲音處理，全華書局，1997