臺灣二００七年國際科學展覽會

科　　別：電腦科學

作 品 名 稱：聽聽貝多芬作品的下一代：將碎形及基因演算法應用於數位音樂產生器

得 獎 獎 項：第一名
英特爾電腦科學獎：第一名
美國正選代表：美國第 58 屆國際科技展覽會

學校 / 作者：國立臺中第一高級中學　　林自均

作者簡介

我叫林自均，興趣是電腦、作曲(MIDI)、打鼓、看書...等等，對於程式設計(C 語言、BASIC)、動畫製作(Flash)、音樂編輯(Cakewalk)都很有興趣，也有一些基礎，從小就覺得電腦很好玩，而資訊技能應該對於我的未來應該很有幫助。由於父母反對填鴨式的教學方法，管教的方式開放，讓我有了更多自我發揮的空間和自己的想法，不會輕易的拘泥於形式。在小學、國中時擔任資訊股長，負責班上網頁與畢業光碟，進入高中後很幸運的參加進階的資訊班，對 C 語言也越來越有把握，這都得歸功於老師的指導有方。近幾年我經常參加各個資訊科學相關的競賽及活動，也加入資訊志工服務團隊，從不斷的資訊科技的參賽和研習活動中，我學習到許多珍貴的經驗與知識，程式設計、動畫製作與音樂編輯更是把我的生活點綴的多采多姿，這一切的一切實在讓我收穫太多了。

摘要

本研究整合了碎形圖形的迭代運算方法與基因交配觀念來達到音樂創新，並透過音樂和諧性判別機制來提高創新音樂的悅耳程度。利用基因觀念之交配的方法來解決長短的問題。這個方法是把原始音符輸入後，找出它們的中心點，以這個中心點為準，其他的音符按照一定比例向外延展，成為新的迭代點。再利用這些迭代點，迭代出新的音符。把製造好的音符染色體放置到交配池中，以隨機的方式在交配池中選取其中之一個染色體進行交配的動作，此二音符染色體會交換彼此的基因，產生下一代新的代表音符長短之染色體，隨後以「模仿母體判斷式」來判斷這新一代的音樂是否與母體音樂相似，藉此淘汰掉「不肖的」下一代，而若新一代與母體的相似程度高的話，它的悅耳性相信也會相對提高。最後把這些技術應用於數位音樂創作，以衍生新穎應用與創新的結果。

關鍵詞：碎形、基因交配、數位音樂

ABSTRACT

Fractals can be produced by IFS (Iterated Function Systems). By iterative computation of many times, we can obtain the similar graphics. In my research, the methods to generate the iterative algorithms were presented. In addition, I would discuss the regularity and the content as well as the properties of those digital patterns. At last, the advanced application of fractals to digital music pieces was presented. The program took a note of several measure of music as the beginning point, and made the IFS calculations for each new note in each measure. But there was no difference in beats if you just make the IFS iteration. So I changed the beats with genetic crossover method. In this research, the expression of the DNA to each beat of note was adopted. The same way, it took a note as a beginning point. And the system obtained the new DNA from the old notes for new ones randomly. After producing the new pieces of music, I want to know if it is good to listen. So I used the algorithm that checks the simulation to the shape of mother music. If its shape is similar to the mother music, the probability that the new music is pleasing may even increase. That would make a piece of brand new music. What I want to do in this research is improve the multiformity of music and find what the relationship is of ‘good music’ and mathematical algorithms.

Keywords: fractal, genetic crossover concepts, Digital music

一、前言（含研究動機、目的）

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。

想想看，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。

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]。

所謂碎形音樂，就是將樂曲組成自我相似性的音樂旋律，比如說將原來的某音樂曲找出兩段碎形結構，其一爲主旋律，另一爲副旋律，整個音樂的旋律係以

主旋律爲主，但在適當處接上副旋律，不斷重複、循環呈現。

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(Crossover)，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。

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。

二、研究方法或過程

2.1 研究方法

2.1.1 迭代運算法

碎形具有重複性與相似性，變化多端，在各個部位、各個比例下，看起來都很相似。碎形圖形可以利用迭代運算系統 IFS(Iterated Function Systems)碼來產生，代入迭代運算方程式後，經由多次的運算，可以得到重覆的圖形。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}。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’ (圖 1)：

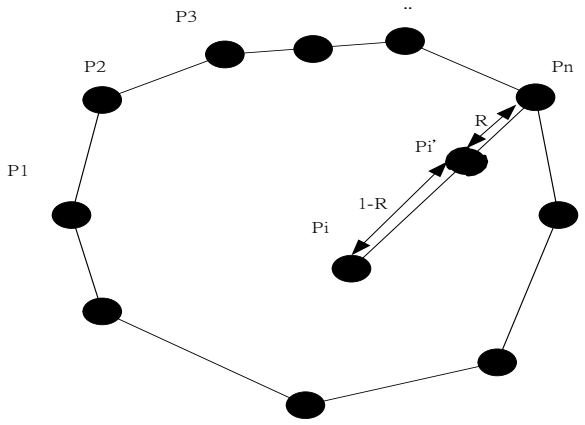


圖 1　反覆隨機迭代點 Pn 與新迭代點 Pi’

Pi’ = (1-R) Pn + R Pi

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。

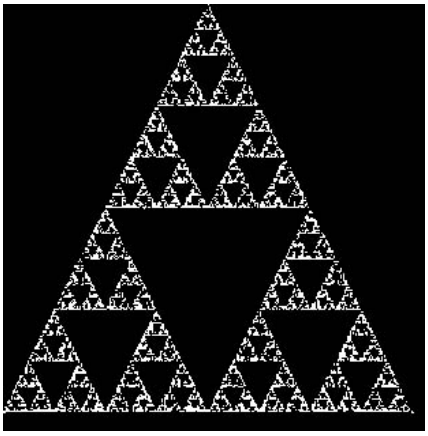
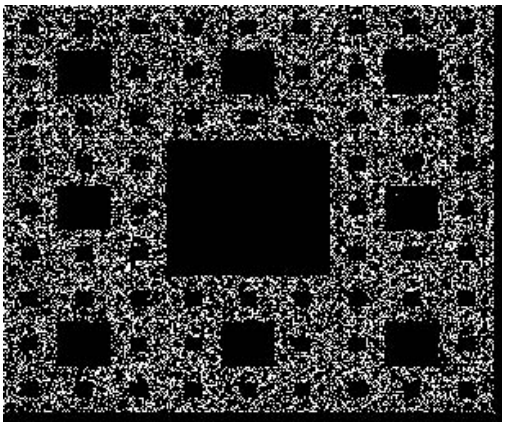


圖 2（A）西爾平斯基船帆圖形



（B）西爾平斯基掛毯圖形

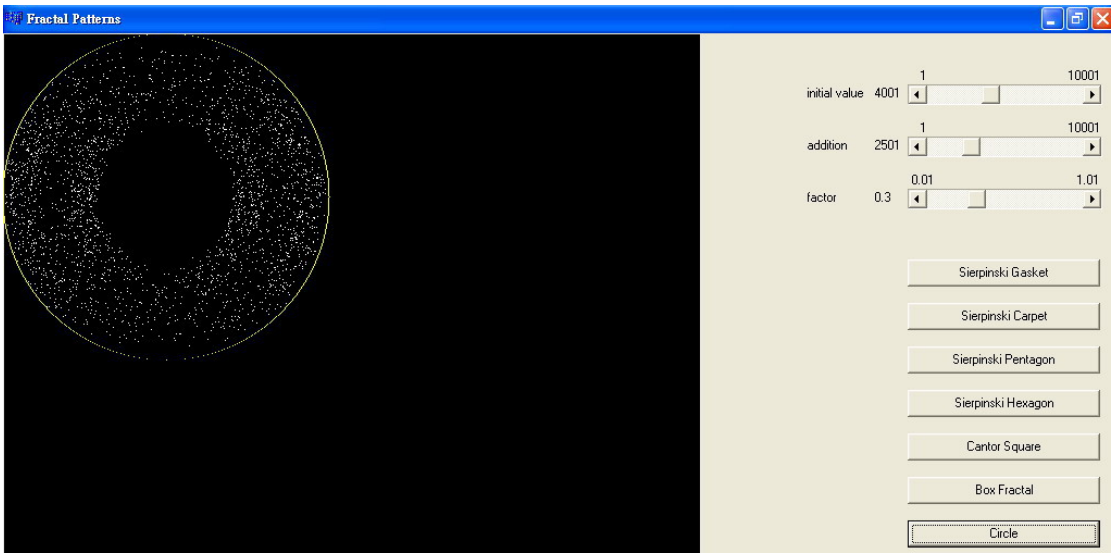


圖 2（C）迭代點排列成圓形之碎形圖案

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。

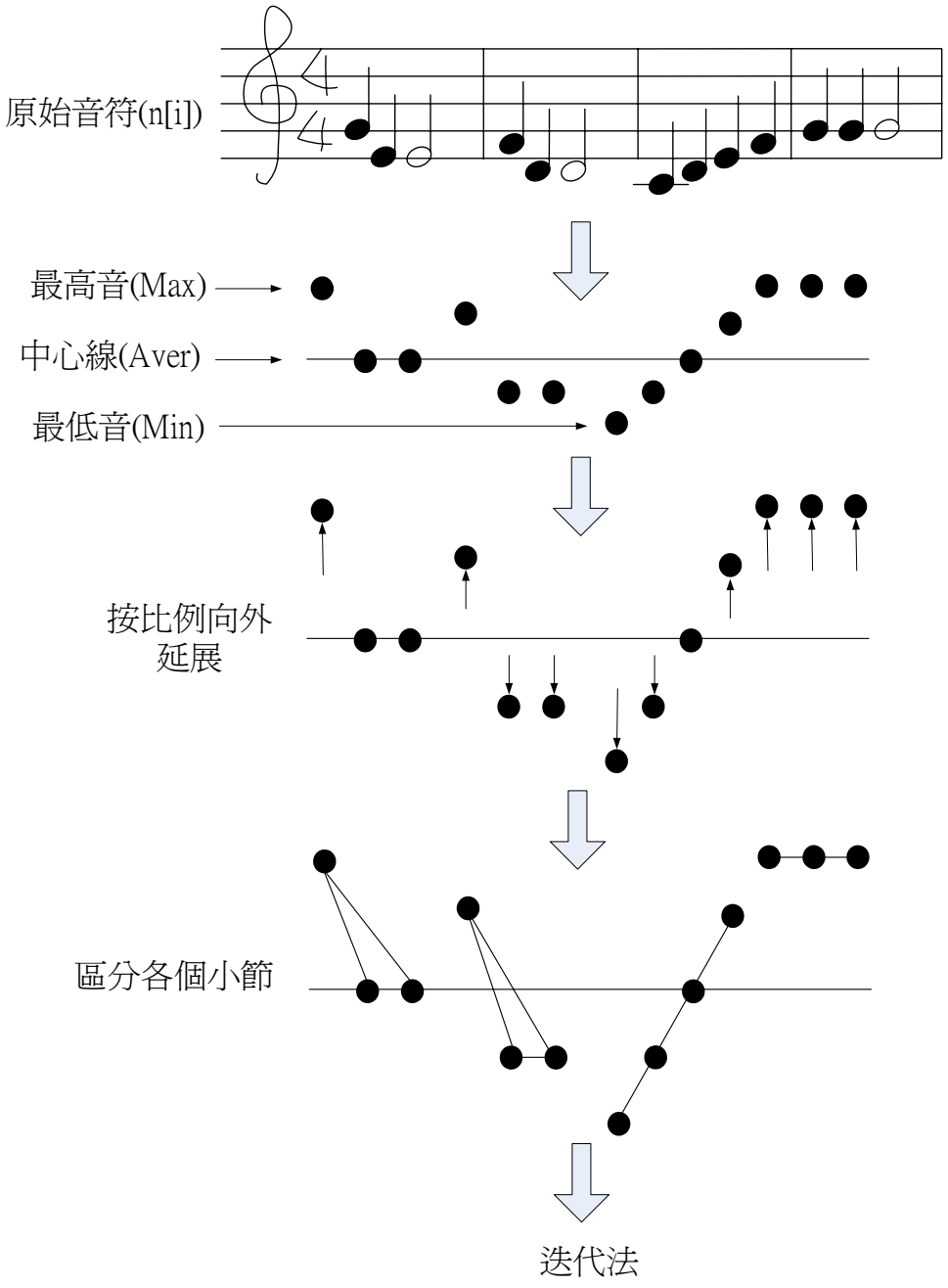


圖 3 找出中心音符將其他的音符按照一定比例向外延展

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
```

2.1.2 基因交配法

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）。

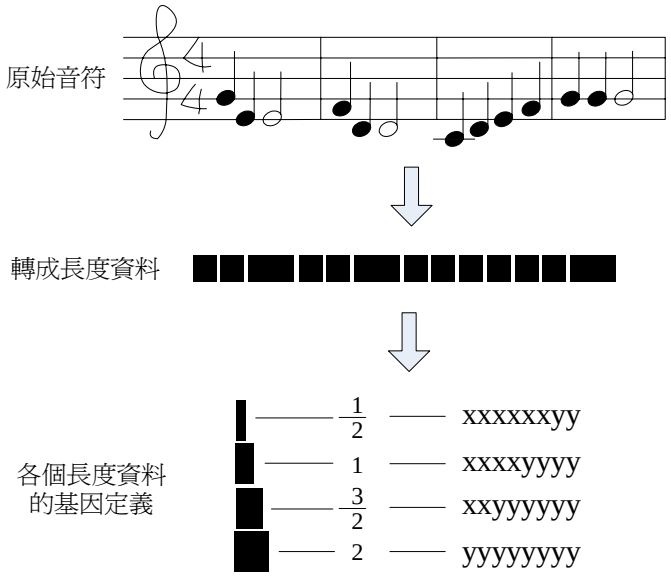


圖 4 每一個音符長短的基因之定義

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

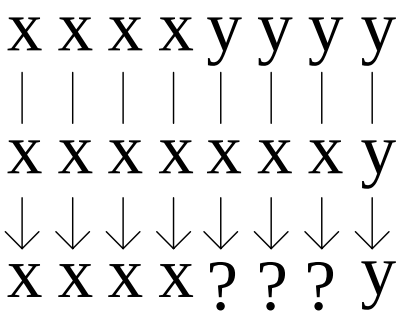


圖 5 把各個基因的 x 和 y 各自隨機取得

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 4 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]
```

2.1.3 模仿母體判斷式

當一首新的數位音樂產生之後，若要判斷該音樂是否悅耳，是可以用人耳來完成這件事；但面對幾百、幾千首的音樂，還以人力來判斷，不僅費時費力，而且人耳的敏銳度亦有限，久了不免感官疲乏。因此最好的方法便是以程式來執行這份任務。但是電腦無法直接以「好聽」或「不好聽」來做判斷式，所以我想：本研究找的母體音樂都是可以稱得上「悅耳」的，那如果以「是否與母體音樂性質相似」做爲判斷的話，程式產生的子代音樂便會與母體音樂之音樂性質相似（其中「音樂性質」主要是指各個音符相對音高的位置），藉此淘汰掉「不肖的」下一代，如此一來，相信可以產生更爲悅耳的音樂。

以下爲本研究「模仿母體判斷式」的定義：

A 爲原始音符音高序列之集合，A = { a_1, a_2, a_3 ..... a_n}

B 爲下一代音符音高序列之集合，B= { b_1, b_2, b_3 ..... b_n}

$Flag_i$ 爲第 i 個規則之判別布林常數

V_t 爲總投票數

V_i 爲各個規則所提供的票數

MAX(A) 爲序列之最大值，min(A)爲序列之最小值，Mid(A)爲序列之中間值

<Rule 1>下一代最高音高的位置順序與母體相同

If a_M = MAX(A)

And b_M = MAX(B)

Then $Flag_1$ = True

$V_t=V_t+V_1$

<Rule 2>下一代最低音高的位置順序與母體相同

If a_m = min(A)

And b_m = min(B)
Then $Flag_2$ = True
$V_t=V_t+V_2$

<Rule 3>下一代第二高之音高的位置順序與母體相同
If a_M = MAX(A) , a_S = MAX(A － { a_M })
And b_M = MAX(B) , b_S = MAX(B － {b_M})
Then $Flag_3$ = True
$V_t=V_t+V_3$

<Rule 4>下一代中間音高的位置順序與母體相同
If a_k = Mid(A)
And b_k = Mid(B)
Then $Flag_4$ = True
$V_t=V_t+V_4$

<Rule 5>下一代音高與母體的相對差值總和小於第一臨界值
If $\sum | a_i - b_i | < T_1$
Then $Flag_5$ = True
$V_t=V_t+V_5$
T_1 爲第一臨界值

<Rule 6>下一代音高相對起伏與母體音高相對起伏的差異值總和小於第二臨界值
Let
$c_t = a_t - a_{t-1}$
$d_t = b_t - b_{t-1}$
($2 \le t \le n$)
If $\sum | c_i - d_i | < T_2$
Then $Flag_6$ = True
$V_t=V_t+V_6$
T_2 爲第二臨界值

使用者可自行選擇要核取哪幾項規則來檢驗該段音樂。核取得越多，下一代音樂與原始音樂的模仿程度就越高，但所花費的計算時間也可能隨之大幅增加。使用者也可以選擇「設定票數」的功能：若使用者在欄位中填入 N，程式便會要求子代音樂要得到 N 票，才可通過檢驗。

2.2 研究過程

在本研究的軟硬體需求上，使用以下設備及軟體：

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。圖 6 爲針對整體的音樂和諧性以及基因交配的缺點加以改進，使數位音樂創作系統能達到整體的音樂和諧性最佳化的程式流程圖，而其中程式執行畫面如圖 7 所示。

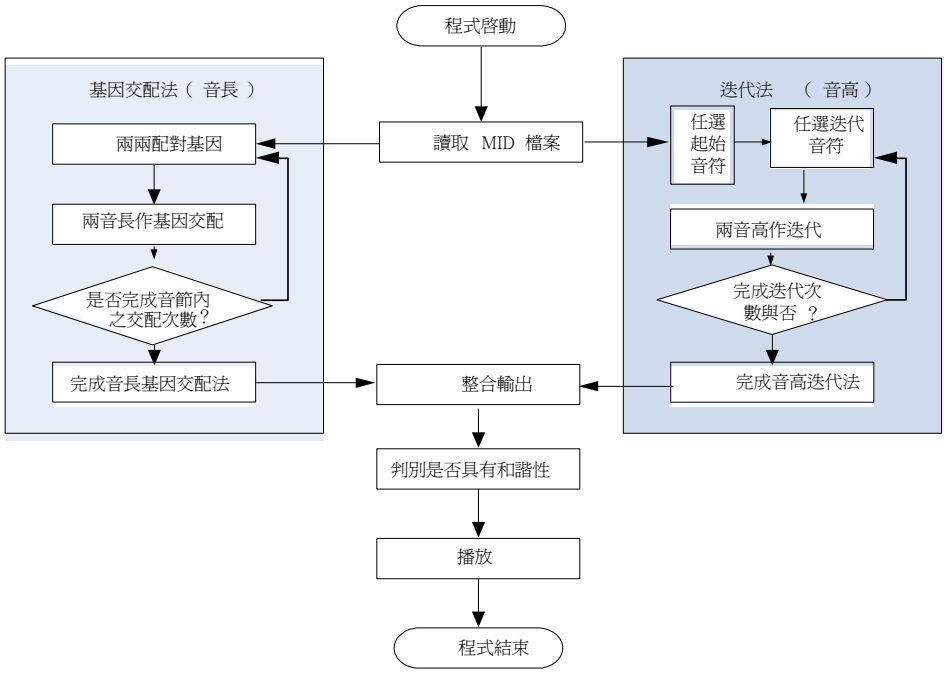


圖 6 碎形音樂產生程式流程

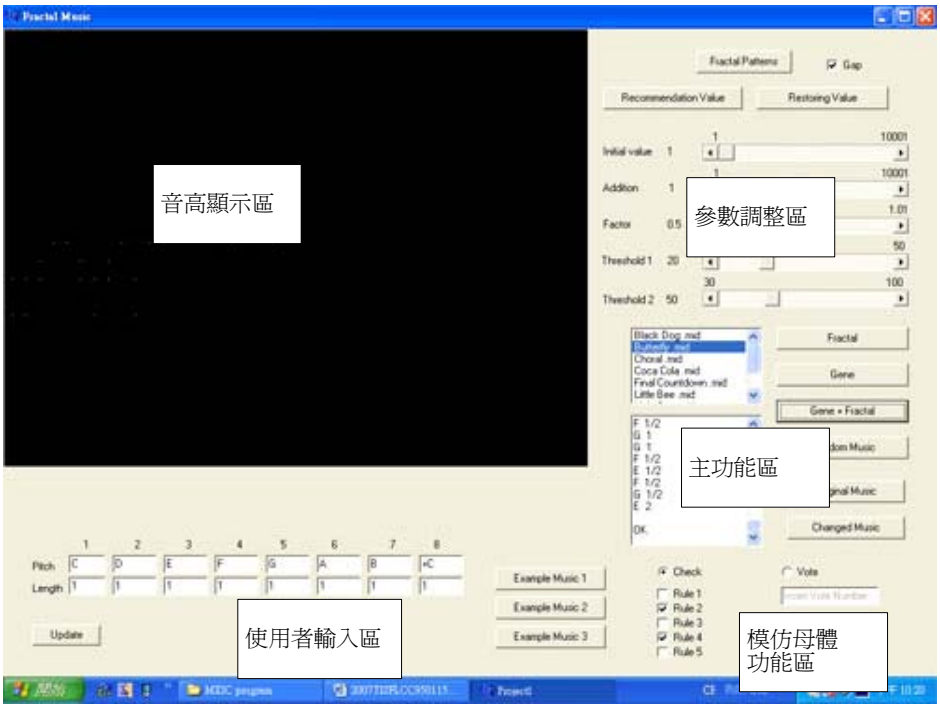


圖 7 碎形圖形及數位音樂產生程式之畫面

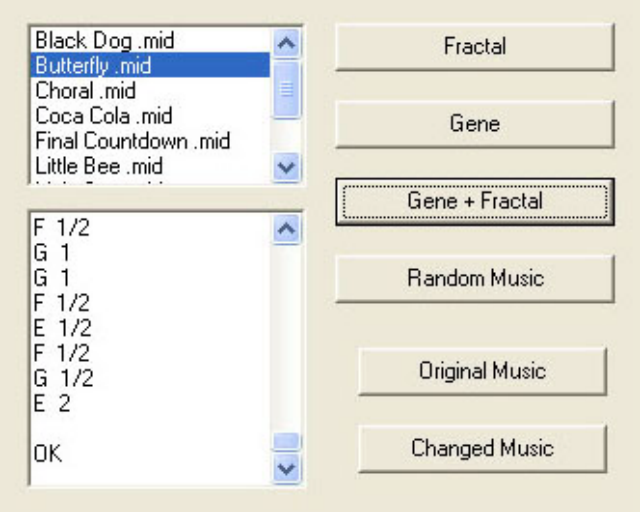


圖 8 音樂產生程式之主功能區塊

在程式中的主功能區塊（如圖 8），是由 (1)母代音樂選單 (2)碎形迭代、基因演算功能 (3)母代 MIDI 檔內容 (4)母代、子代音樂播放功能 (5)隨機音樂產生器所組成。在這個區塊，使用者可以挑選一段母代音樂，並決定要以此音樂進行碎形迭代法、或是基因演算法、或是兩者並行。按下按鈕之後，左方跳出的內容是母代 MIDI 檔之內容，裡面包括有各個音符的音長值和音高值。接下來使用者可以按下播放鈕以播放母代與子代音樂。而隨機音樂產生器可產生出完全以亂數出現的音高和音長，以與前述數位音樂比較之用。

圖 9 模仿母體功能區塊

而模仿母體功能區塊（如圖 9）則是與上述主功能區塊並用，以提高子代音樂之母體相似度，進而增加此音樂之可聽性。該區塊左側欄位可供使用者勾選所想要的規則項目，右側欄位則可供使用者填入想要的票數；而程式將會要求子代音樂符合正確的規則項目，或是具有足夠的票數。

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。而其執行畫面如圖 10。

圖 10 使用者自行輸入音符之程式執行畫面

三、研究結果與討論

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 11。

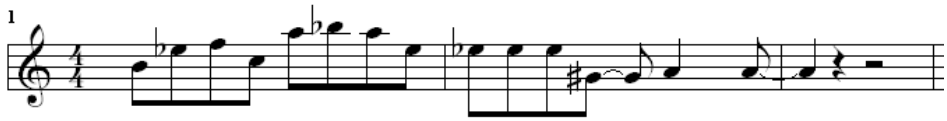


圖 11　推廣西爾平斯基船帆與掛毯圖形的概念而衍生的碎形音樂

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 12)。

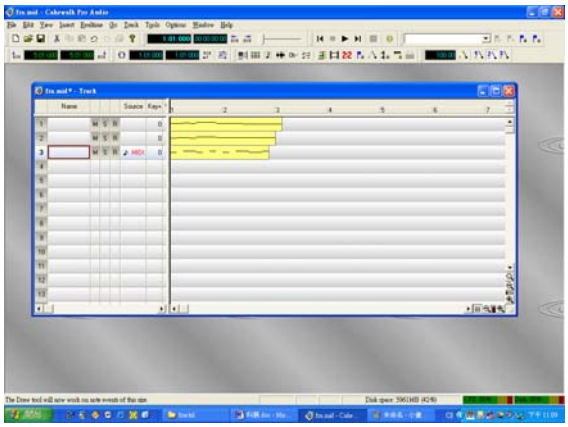


圖 12　在主旋律(Track 1)之外，又加入副旋律(Track 2)與伴奏(Track 3)

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 13。

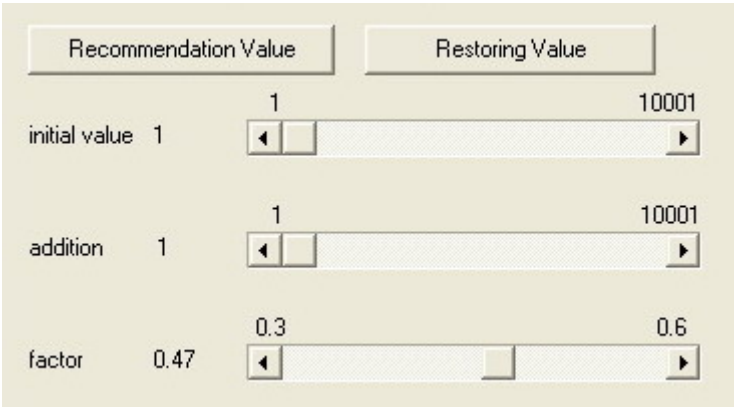


圖 13 本程式提供建議值之執行畫面

表 1 本研究之數位音樂的試聽結果

試聽結果	難聽	好聽
旋律 1	43.9%	56.1%
旋律 2	27.2%	72.8%
旋律 3	36.7%	63.3%

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，經由一百八十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。

表 2 爲隨機亂數產生之旋律與本研究方式產生不悅耳旋律之機率之比較。由表可知，本研究方式產生不悅耳旋律之機率大幅降低。

表 2 各種方式產生不悅耳旋律之機率

採用方法	不悅耳的機率
隨機亂數產生之旋律	92.2%
單純以碎形迭代法與基因演算法結合	61.8%
碎形迭代法、基因演算法再配合模仿母體之方法	14.6%

總而言之，經過超過 1000 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。

四、結論與應用

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「小節間的交配」、「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情況，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、能夠延伸「模仿母體」的判別機制，即先由第一個母體音樂透過基因交配及碎形迭代法產生之子代音樂，再模仿另外的第二個母體音樂，使其帶有另一種音樂之風格。如此一來，就有可能輸入莫扎特第 24 號奏鳴曲，結果輸出爲第 25 號奏鳴曲，並出現另一位作曲家——譬如貝多芬的曲風。
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。

五、參考文獻

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

Listen to the next generation of Beethoven's compositions:

Digital music generator using fractal and genetic crossover concepts

Chi-Chin Lin

ABSTRACT

In this project, we design and implement a novel music generator platform named Composer Imitator (CI) to create pleasuring music automatically. In the CI platform, we propose the Enhanced Iterated Function System and apply Genetic Crossover Method to create new music. The Imitated Parent Judgment (IPJ) function is defined to evaluate the *pleasure* of the music. Initially, the CI randomly selects an initial music note that will be used as the basis to obtain several new musical notes by applying the Enhanced Iterated Function System. The Genetic Crossover Method determines the length of these new notes. We set rules in the IPJ function, applying the rules, examine the *music similarity* (that is the relative note pitch) of the generated musical notes, and then filter out the *bad* musical notes. In this project, we run test experiments to study the *good music ratio* (that is the probability that the generated music is scored as the *good* music by the listeners). Our study shows that the CI platform has good *good music ratio* performance.

Keywords: Fractals, Genetic crossover concepts, Digital music

1. Introduction

Think of it! If we enter a piece of music, for example Mozart's No. 24 Concerto into a type of function system, will it become possible for us to generate the output of No. 25 Concerto or might we end up having another music style such as that of Beethoven? Although this may sound weird, it is undeniably, a very interesting thought.

Music is not a simple thing. One immediately discovers the diversity of music when conducting an in-depth study. Tempo alone is categorized into different types: Blues, Swing, Waltz, Latin, Quick Steps, Rock, Disco, Marching and so on, and each type has its own smaller sub-types. Composition skills are even more complicated and include: harmony, appoggiatura, chord, ornaments, bass, strings, percussion and many others.

Sierpinski Gasket and Sierpinski Carpet are two types of fractals. "Fractal Geometry" was developed by the mathematician Mandelbrot in the 1960~70s [1-3]. Fractals present so called "self-similarity;" that is, amplified or compressed fractals all look like the original form. Each small portion is as complicated as the whole, and fractals repeated are shown in ever smaller dimensions [4]. When we use Enhanced Iterated Function System within indefinite function systems, we are able to acquire indefinitely repeated fractals [5].

Fractal music refers to music composed of self-similar melodies; for instance, two pieces of fractal structure are major and minor melody. The presentation mainly focuses on major melody and transits with the minor melody to repeat the music.

Gene Function systems was proposed by John H. Holland, a professor from the University of Michigan, in 1975 [6], and based on the concept of "the survival of the fittest," he created an environment that allows chromosomes to crossover and then to produce the genes of the fittest recombinations.

Here we suggest some methods to solve the iterated relationship of fractals and then discuss its regular pattern and contained contents and properties with the focus on the fractal concepts of Sierpinski Gasket and Sierpinski Carpet: the introduction of fractals to digital music. The application makes use of a piece of music as repeated random points and with Enhanced Iterated Function System , after many times we are able to acquire musical notes. In the end, "Genetic Function systems" determines the

note length in the creation of digital music to present the result of a new application and creation based on the fractal concept and gene crossover.

Yet, in the unlimited field of music creation, “no inspiration” is a nightmare that strikes many music composers during their creation attempts. It is a great sorrow to them when no new ideas arise. As a result, the purpose of this study is to help and inspire these music composers to create greater pieces and give more benefits to music lovers. In the future, it is quite possible that with the assistance of computer aided interface, composers can more easily finish their digital music. Furthermore, it is also the hope of this study that we are able to determine the direct relationship between “great music” and mathematics.

2. Digital music generator with new methods

2.1 The novel concepts in this research

2.1.1 Enhanced Iterated Function System

Fractals have the properties of repetition and similarity with diversified changes; they all look alike in any part and portion. Fractals are produced by the code of Enhanced Iterated Function System (EIFS) and with multiple function systems are able to acquire repeated pattern. Let us assume the characteristics of the vertex of the polygon, the center, and the circle are the random points Pm={ P1, P2, P3…..Pan}. With the derived Enhanced Iterated Function System, we are able to start from any repeated random point Pi and obtain new music point Pi' (Fig. 1).

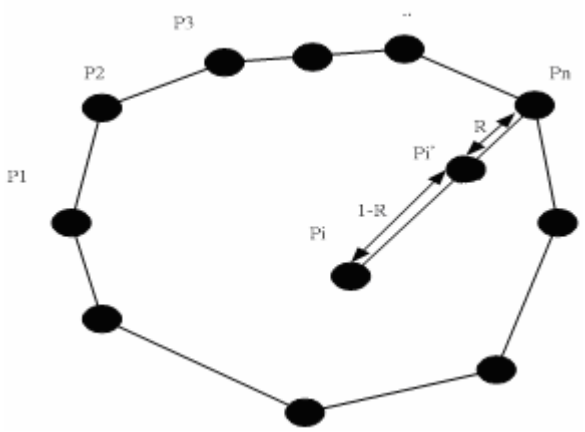


Fig. 1: Repeated random point Pn and new iterated point, Pi'

Pi'= (1-R) Pn + R Pi

Where R is the ratio of the distance from new iterated point, Pi, to each iterated point to that between each iterated point.

Based on the above assumption and function, we propose a C++ formula to discuss the relationship between the final structure of P point, R value, and fractals. If we assume P as the three vertexes of a triangle and R as 0.5, we then will be able to acquire

Sierpinski Gasket (shown in Fig. 2(A)). If P point is assumed as three vertexes of a triangle and R as 0.5, then we will have Sierpinski Carpet (shown in Fig. 2(B)). With different adjustments, other fractals are acquired. If we apply this concept to musical notes and use them as iterated points, then we can derive new iterated points, Pi' new musical notes.

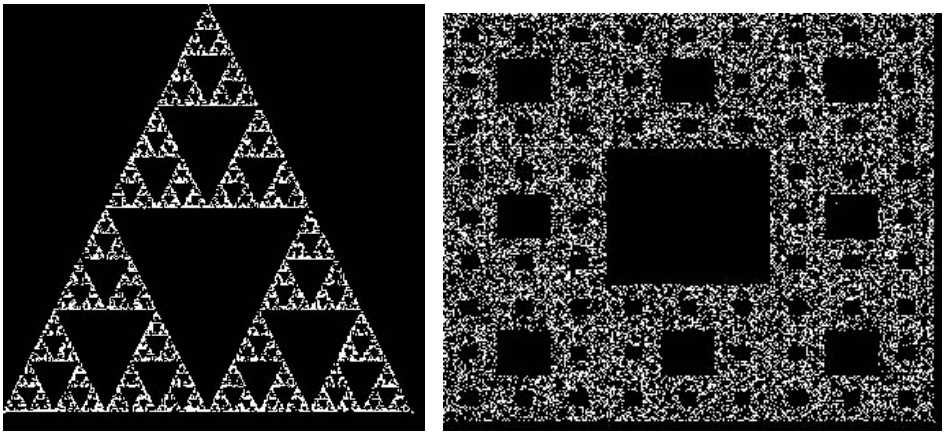


Fig. 2（A）: Sierpinski Gasket （B）: Sierpinski Carpet

Fig. 2（C）: Circular Fractal formed by iterated points

Before implementing integrated function systems of each musical note, we need to consider the problem that due to its invisibility, it is difficult for us to determine the iterated regularity and sequence. Thus, we need to adjust the function systems results of multiple iterated points, initial value, iterated frequency and Enhanced Iterated Function System coefficient R through fractals. For example, Fig. 2（C）shows a circular fractal formed by iterated points with the initial value of 4001 and iterated times of 2501. Enhanced Iterated Function System coefficient R is a fractal of 0.3. With this, we are able to begin the Enhanced Iterated Function System of musical notes.

Experiments, however, showed that if we only use initial musical notes to iterate, the result is proven to be very insignificant, with small changes. Therefore, this study suggests a solution to define a central point after the input of initial musical notes and with this central point, other musical notes are extended according to proportion to form new iterated points and new musical notes. These iterated new musical notes are better developed and improved. The development procedure is shown in Fig. 3 and the steps are:

1. Identify the highest and lowest note (as Max and Min) and determine their average value (Aver).

2. Determine the distance of each note (n[i]) and the average distance (n[i]-Aver).

3. Enlarge the distance to (n[i]-Aver)/R, and R is the coefficient of Enhanced Iterated Function System.

4. The vertex of each new iterated point is signified as (n[i]-Aver)/R + Aver.

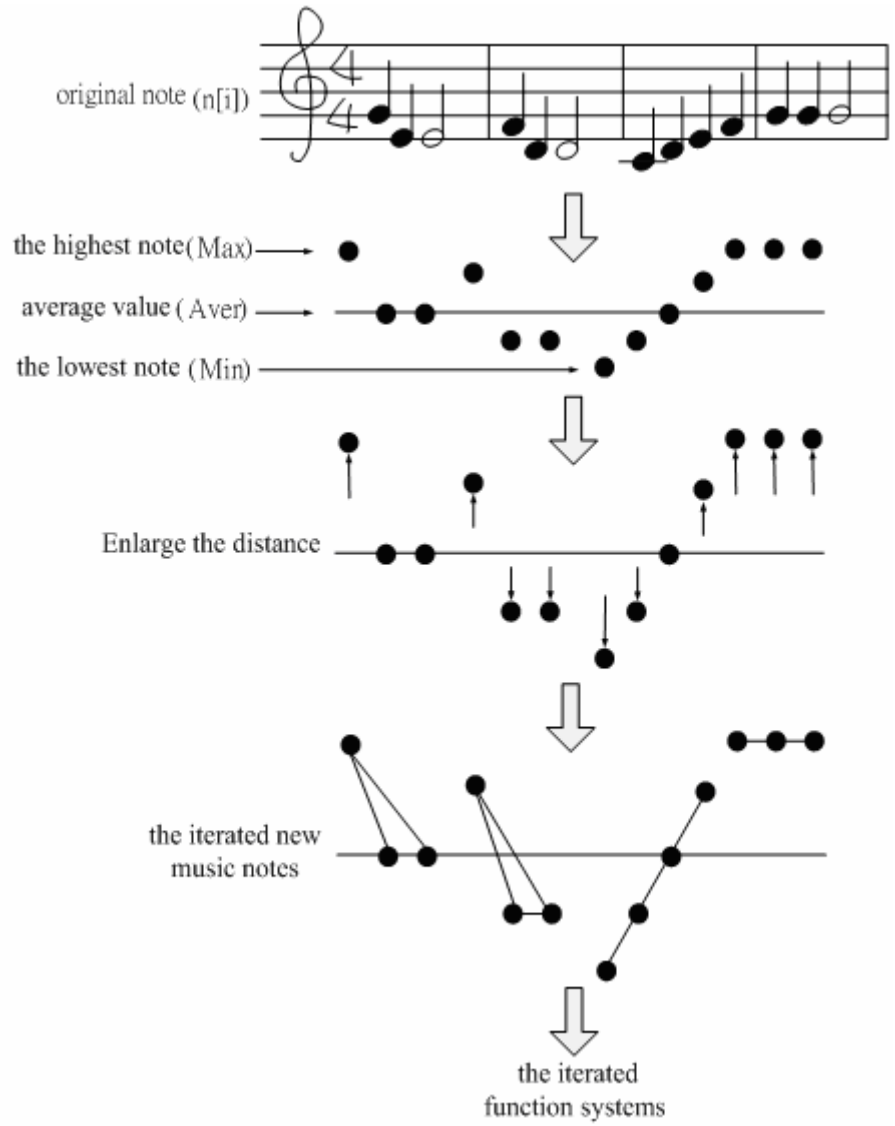


Fig. 3: With this central point, other musical notes are extended according to proportion.

Following is an example of changing note length:

from「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」

into「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」

Where, C, D, E, F, G, A, and B represent Do, Re, Mi, Fa, Sol, La, and Si respectively; numbers after letters indicate beat (note length), for example, "#" for sharp and "- " for flat.

Following are the proposed Enhanced Iterated Function System of digital music of this study:

```
1. Notation
  R: the ratio of Enhanced Iterated Function System
  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;
  derive 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
```

2.1.2 Genetic Crossover Method

But in comparison, we found that changed notes do not show difference in tempo with the initial notes, so we needed to use a Genetic Crossover Method [7]. First, we needed to produce genes in different note length in eight portions of different combinations of x and y. The y number indicates note lengths (shown in Fig. 4).

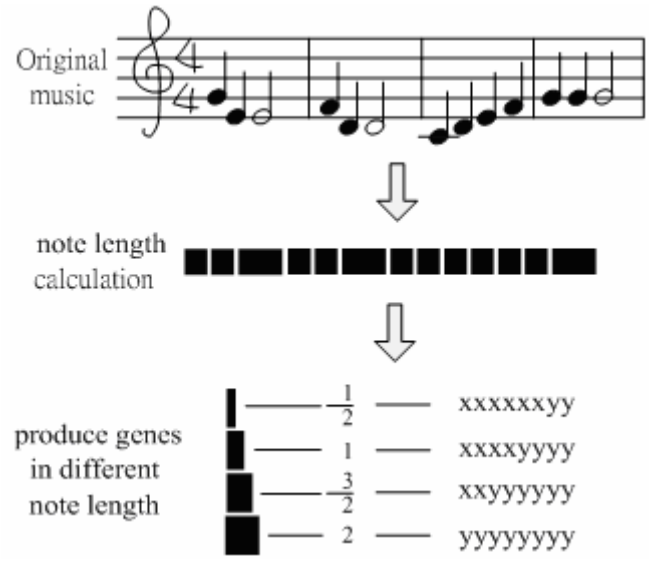


Fig. 4: Definition of genes with different note lengths

For instance,

1/8 note is shown as xxxxxxxy;

1/2 note is shown as xxxxyyyy.

The second step is to select two genes randomly from the crossover pool; the method is to pick chromosomes of each gene (x and y) randomly (shown in Fig. 5) to produce a new gene.

We randomly picked new chromosomes and the possible combinations were:

1/8 note, xxxxxxxy;

1/4 note, xxxxxxyy;

3/8 note, xxxxxyyy; and

1/2 note, xxxxyyyy.

Fig. 5: Randomly picking x and y from each gene

Some may wonder: “if I simply pick a gene between the range, then will this show a similar result?” It may look so, but in reality, the result is different. In terms of ratio, notes after crossover are more likely to have their length in the range of the length of the parent generation.

Take Fig. 4 as an example: the ratio to have 1/4 and 3/8 notes is 3/8, and the ratio of the occurrence of 1/8 and 1/2 notes is only 1/8. This is similar with the pitch of children that have a higher probability ratio of falling in the pitch range of their parents. Consequently, this study concludes that Genetic Crossover Method is fair application.

The change of note length is shown in following example:

FromC1 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」

into「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」

This shows the changes of both note pitch and length:

from「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」

into「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」

Here is the digital music developed by a Genetic Crossover Method in this study:

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 know how many notes there are 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 how many 'y' there are in each note and write the new value into
                                    the bt[i]

      Otherwise,

          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 how many 'y' there are in each note and write the new value into the bt[i]
```

2.1.3 Imitated Parent Judgment Function

When a new piece of digital music is produced, we are able to use human ears to judge the presentation, yet it is quite time consuming to judge the music with human labor and at the same time, the accuracy of human ears may diminish after lengthy usage. Thus, the best way is to carry out the task via function. Computers may not be able to directly judge the music; however, this study selected parent music that is "pleasant to ears," so if "the music similarity with parent music" is used as the basis of judgment, offspring generation music similar with the characteristics of parent music, will be generated ("music characteristic" here refers to the relative note length position) to eliminate "different" offspring. It is thought that more pleasant music will then be composed.

Next formulates IPJ function used in this study:

A : the assembly of note pitch sequence of initial notes ; A = { a_1, a_2, a_3 ..... a_n}

B : the assembly of note pitch sequence of next generation; B= { b_1, b_2, b_3 ..... b_n}

$Flag_i$: the determinant Boolean constant of Rule i

VT : the total vote

V_i : the vote provided by each rule

Max(A) : the maximum value of the sequence

Min(A) : the minimum value of the sequence

Mid(A) : the average of the sequence

<Rule 1>:The position of the maximum note length of the immediate offspring

generation corresponds to that of the parent note.

If a_M = MAX(A)

And b_M = MAX(B)

Then $Flag_1$ = True

VT = VT +V_1

<Rule 2>: The position of the minimum note length of the immediate offspring generation corresponds to that of the parent note.

If a_m = min(A)

And b_m = min(B)

Then $Flag_2$ = True

VT = VT +V_2

<Rule 3>: The position of the second maximum note length of the immediate offspring generation corresponds to that of the parent note.

If a_M = MAX(A) , a_S = MAX(A – { a_M })

And b_M = MAX(B) , b_S = MAX(B – {b_M})

Then $Flag_3$ = True

VT = VT +V_3

<Rule 4>: The position of the medium note length of the immediate offspring generation corresponds to that of the parent note.

If a_k = Mid (A)

And b_k = Mid (B)

Then $Flag_4$ = True

VT = VT +V_4

<Rule 5>: The sum of the relative deviance value of the immediate offspring generation note and parent note is smaller than the first threshold value.

If $\sum | a_i - b_i | < T_1$

Then $Flag_5$ = True

VT = VT +V_5

T_1 is the first threshold value

<Rule 6>: The sum of deviance value of relative fluctuation of the immediate offspring generation note and the parent note is smaller than the second threshold value.

Let

$c_t = a_t - a_{t-1}$

$d_t = b_t - b_{t-1}$

($2 \le t \le n$)

If $\sum | c_i - d_i | < T_2$

Then $Flag_6$ = True

$VT = VT + V_6$

T_2 is the second threshold value.

Users are allowed to select evaluation rules to check the music presentation and the more rules selected, the more the next offspring music imitates the parent music; however, the calculation time also increases accordingly. Users can select the "vote set" function by keying in N in the field, then the function will request offspring music to receive N votes to pass the evaluation.

2.2 Research procedures

This study used software and hardware as follows:

1. One personal computer (CPU：Pentium Ⅲ 500, memories：128MB, display card: S3 savage 3D/M) is used to operate systems.
2. Printing equipment;
3. Software program：Borland C++ Builder 6.0
4. Digital music creation software：Cakewalk 8.0

This study used MIDI to compose music and music was played via MIDI of Windows. MIDI is a communications standard between and among digital music instruments; MIDI stands for Musical Instrument Digital Interface [8]. MIDI files record all the musical notes in characters, and because they record characters not waves within a certain period as other music files WAV, WMA, and MP3 do, the files are smaller. MIDI, thus, is widely used on the Internet.

MIDI files consist of two major data segments: header data segment, Mthd, and audio data segment, Mtrk. Mthd includes the name of the segment, the size of the segment, MIDI file format, the number of audio segments, and timeline; Mtrk includes the name of the segment, the size of the segment, and segment information.

The "changes of note pitch and length" mentioned earlier in this study involved reading a MIDI FORMAT 0 file, and after iterated function and Genetic Crossover Method, the file was transferred to another MIDI file. This study only required a single note of digital music; therefore, minor changes were achieved. We made use of a small

portion of Mtrk segment data.

A music note in Mtrk of MIDI reads like:

ABCA

"A" refers to note pitch, "B" refers to volume, and C is the note length; all are implemented with the value of the ASCII code. For example, a high note, Do, with volume 100 and half beat is shown as：

Hd < H

Where, "H" has the ASCII code of 72 presenting Do; the ASCII code of "d" is 100 presenting volume 100, and the ASCII code of "<" is 60 presenting the length of a half beat.

This study accessed four-note music via MIDI files and made use of Enhanced Iterated Function System and a Genetic Crossover Method to produce digital music of immediate offspring generation and transferred it to new MIDI files for music presentation. Fig. 6 shows the improvement of music harmony and the drawbacks of Genetic Crossover Method, to optimize a digital music creation system for the achievement of overall music harmony. The program implementation is shown in Fig. 7.

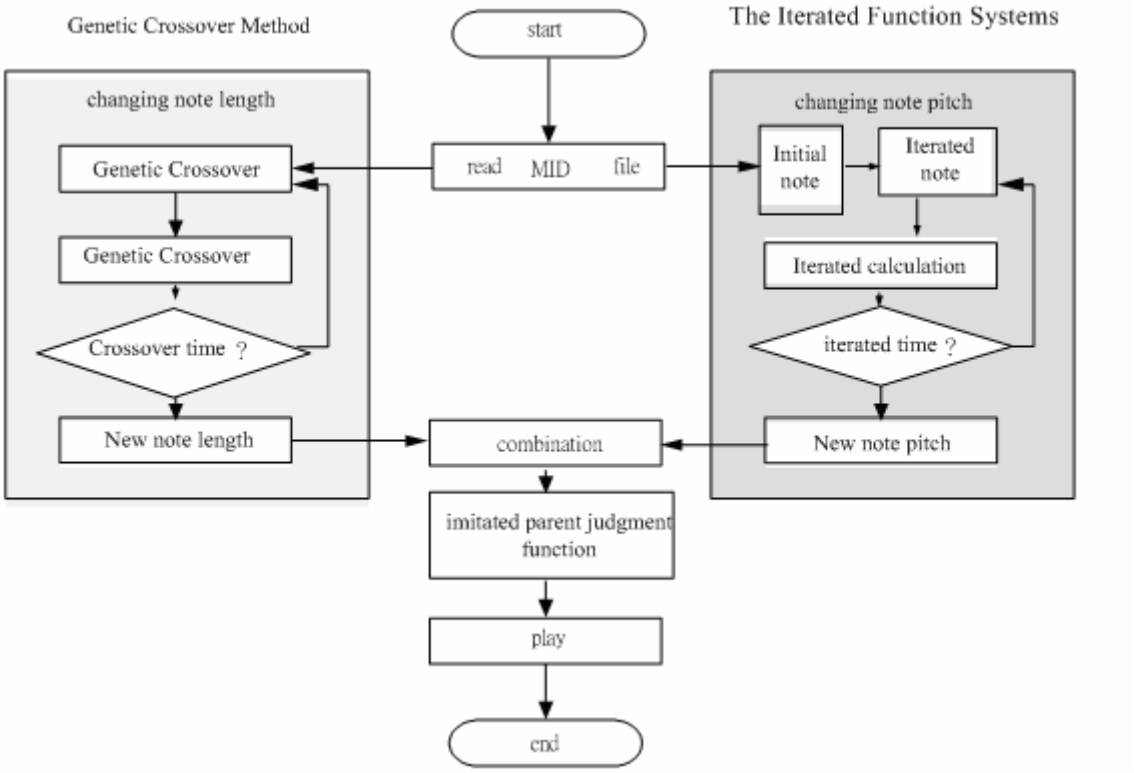


Fig. 6: Production procedure for the generation of digital music

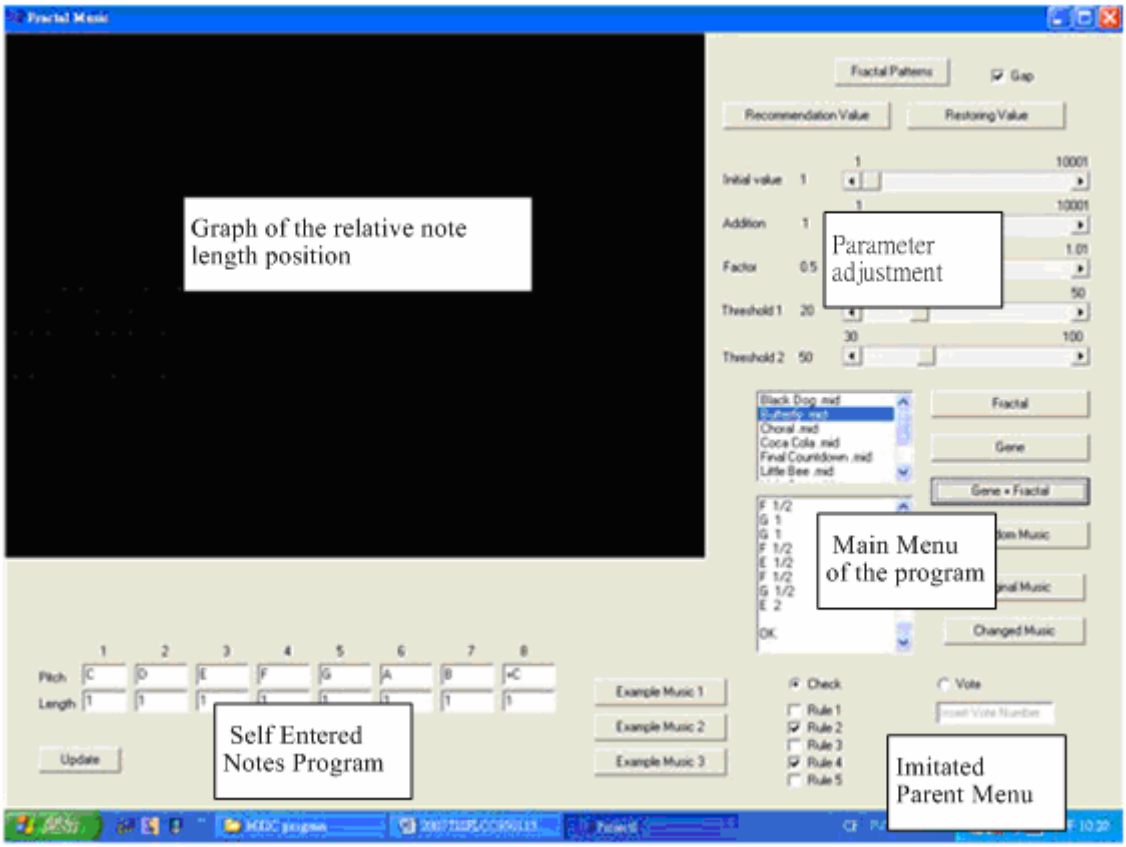


Fig. 7: Screen Window of fractals and program used for the creation of digital music

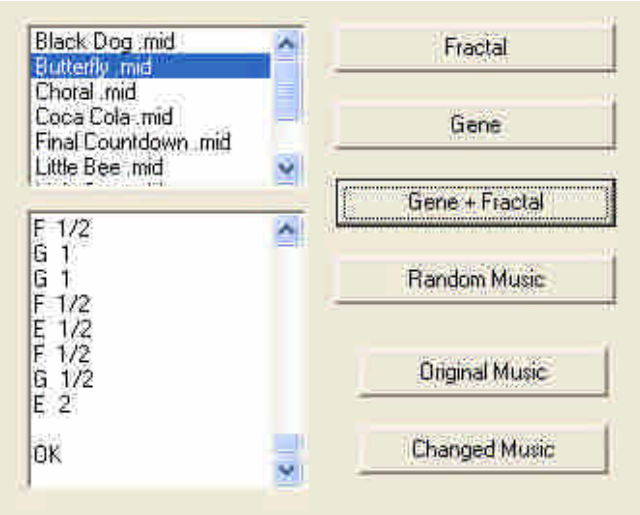


Fig. 8: Main Menu of Music Generation Program

The Main Menu of the program (shown in Fig. 8) consists of (1) Selection menu of parent music; (2) function systems of iterated fractals and genetic function systems; (3) MIDI file contents of parent generation; (4) music play function of parent and offspring generations; and (5) random music composer. In this menu, users are allowed to select a piece of parent music and then decide to use either fractal iterated or Genetic Crossover Methods or both. After clicking, the contents of parent MIDI file will pop up on the left, including note length and pitch values. Users then are able to click play button to listen

to parent and offspring music. Random music composer produces random note pitch and length for comparative purposes.

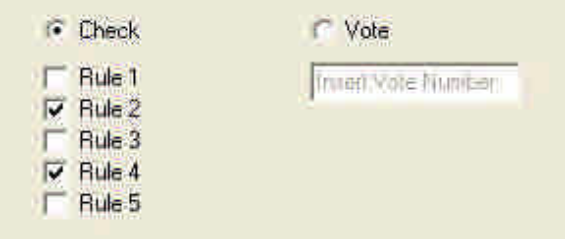


Fig. 9: Imitated Parent Menu

Imitated Parent Menu (shown in Fig. 9) is used together with the above main menu to increase the similarity between offspring and parent music and improve music audibility. On the left field, users are able to select their desirable rules and on the right, they are able to key in ideal votes. The program will require that offspring music has correct rules and votes for evaluation.

In addition, this study added one more function for users to enter their desirable note requirement. With the selection of any note, users are able to adjust note pitch and length. Click “renew” button and a piece of self-composed music is completed. Users are able to use this piece of music to compose next generation digital music with Enhanced Iterated Function System and Genetic Crossover Methods. This is still only at the stage of initial evaluation; thus, users are able to enter notes with eight scales of pitch and four scales of length; future expansion is possible according to actual need. The implementation is shown in Fig. 10.

Fig. 10: Implementation of Self Entered Notes Program

3. Experimental results and discussions

The above mentioned skills are used to produce digital music and then the music is composed with music creation software Cakewalk 4.0 shown as Fig. 11.

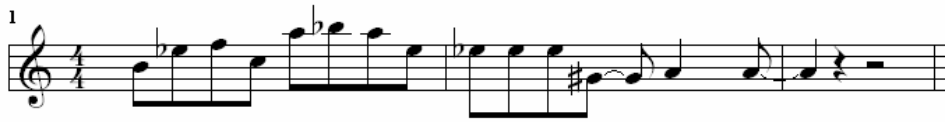


Fig. 11: Digital music derived from the concepts of Sierpinski Gasket and Sierpinski Carpet

To enrich music contents, minor melody and music accompaniment are added to major melody (shown in Fig. 12).

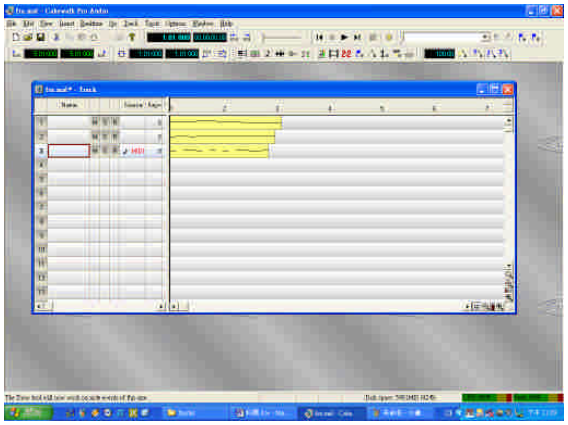


Fig. 12: In addition to major melody (Track 1) minor melody (Track 2) and music accompaniment (Track 3) are added.

An adjustment experiment of initial value, iterated time, and Enhanced Iterated Function System coefficient, R, was conducted and according to test listening, there is no direct relationship between the change of initial value and iterated time and the pleasure level of music presentation. Moreover, this study also found that Enhanced Iterated Function System coefficient, R, defines more of the pleasant music range. Based on the suggested value provided by the program, we implemented it as shown in Fig. 13.

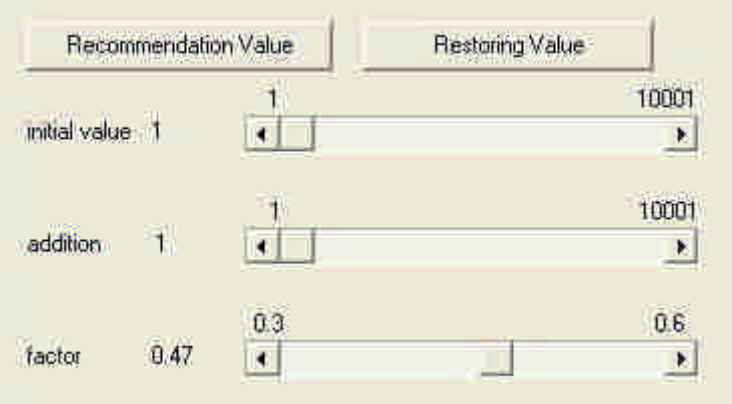


Fig. 13: Implementation of suggested value provided by the program

Table 1: Test listening result of digital music of this study

Results	Pleasant
Melody 1	56.1%
Melody 2	72.8%
Melody 3	63.3%

Initially, three pieces of melodies were selected via designed program and handling procedure to produce digital music, and the testing results of 180-listeners experiment are shown in Table 1. During the test listening process, we determined the pleasure level of each piece of digital music and from Table 1, there were more "good" responses of the re-designed and re-composed digital music of this study than "not good" response. In terms of poorer response to "Melody 1," this study attributed the reasons to shortage of iterated points caused by insufficient changes of parent melody, note length or note pitch or fewer numbers of notes.

Table 2 demonstrates the comparison of poor response to melodies randomly generated and those composed by this study. According to the table, the ratio of poor response in this study is greatly reduced.

Table 2: Ratio of Poor Response to Melodies Generated by Different Methods

Method	Poor Response Ratio
Random Selection	92.2%
Enhanced Iterated Function Systems and Genetic Crossover Method	61.8%
Enhanced Iterated Function Systems, Genetic Crossover Method, and Imitated Parent Judgment	14.6%

In summation, after 1025 times of testing, music generated by this study is more pleasant to ears and more valued. That is to say although this music composer may produce good music once in a while, yet it can serve the practical function of inspiration for human composers.

4. Conclusion

MIDI files have a complicated format and on the Internet or database available in books, both provide MIDI files with simple accessing and playing functions and the content of each note inside cannot be read, not to mention editing or re-writing. As a result, this study expended great efforts to analyze MIDI file formats and although the proposed program only can operate under certain MIDI formats, it satisfied presumed Enhanced Iterated Function System and genetic crossover function. In the future, it is hoped that all MIDI formats will be able to be processed to produce track 2 and track 3 for vocal harmony and music accompany.

In addition, this study only took the steps of "gene production," "crossover," and "selecting the final genes" but did not complete "crossover between beats" and "mutation." Programs in this study iterated each note to generate the note of the immediate offspring and compose new music. In reality, crossover of DNA , however, does not occur between two atoms but between proteins and it should be applied to this study in "crossover between beats." If this part can be completed, it will be more practical for real genetic crossover and for generating favored music.

To sum up, future improvement of this study will include:

A. Processing for any MIDI format;

B. Automatic generation of Track 2 and Track 3 for vocal harmony and music accompaniment;

C. Determinant mechanism extends "imitated parent;" that is, the first parent music produces offspring music via genetic crossover and fractal iterated offspring music, and then imitates another second parent music to compose another style of music. With this, if we enter Mozart's No. 24 Concerto into function systems, we will have the output of No. 25 Concerto or Beethoven's music style.

D. The possibility to add "mutation" in genetic crossover.

E. Adding "crossover between beats" in crossover method.

F. Connecting to external input equipment such as note input from keyboard and human voice input from microphone for deriving calculations required for constructing a comprehensive human-machine interface.

5. References

1. Rob Eastanway (Translated by Tsai Cheng-chi), How Long is a Piece of String?: More Hidden Mathematics of Everyday Life, Cite Publishing Ltd, 2005
2. Wu Wen-chen, 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. Liao Si-shan, The Charm of Fractal, Science Monthly 363, 2000, p.222
6. Chao Peng-chen, Theory and Application of Genetic Algorithm, CHWA Publishing, 2001
7. Goldberg, D, E. , Genetic Function systems in Search, Optimization, and Machine Learning, Addison-Wesley , New York, 1989
8. Lin Chern-sheng, Digital Signal-Image and Sound Processing, CHWA Publishing, 1997

評語

本作品利用碎形中的重複性與相似性，與音樂的樂曲結構類似，因此以電腦演算法可產生新的音樂。其次，電腦所產生的音樂，可以用基因演算法來應用，讓兩段音樂如染色體基因，產生交配(Crossover)，而得到下一代音樂。

最後，如何從下一代音樂中選擇優秀的子代?本作品用”與母體音樂相似度”來篩選。結果，有篩選比沒有篩選的，可讓”不悅耳”的音樂從 60%降至 20%以下。本結果有科學的方法與創意，又有實用性，故推薦之。