國立交通大學

工學院聲音與音樂創意科技碩士學位學程

碩士論文

依據樂曲分析之演算作曲系統建構

Building an Algorithmic Compositional System Based on Music Analysis

研 究 生：謝仲其

指導教授：曾毓忠　教授

中華民國一〇二年一月

依據樂曲分析之演算作曲系統建構

Building an Algorithmic Compositional System Based on Music Analysis

研 究 生：謝仲其　　Student：Chung-Chi Hsieh

指導教授：曾毓忠　　Advisor：Yu-Chung Tseng

國 立 交 通 大 學

工 學 院 聲 音 與 音 樂 創 意 科 技 碩 士 學 位 學 程

碩 士 論 文

A Thesis

Submitted to Master Program of Sound and Music Innovative Technologies

College of Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Engineering

January 2012

Hsinchu, Taiwan, Republic of China

中 華 民 國 一 〇 二 年 一 月

依據樂曲分析之演算作曲系統建構

研 究 生：謝仲其　　指導教授：曾毓忠

國立交通大學工學院聲音與音樂創意科技碩士學位學程

摘要

演算作曲並非電腦科技的專利。從莫札特的骰子音樂到約翰．凱吉的機遇音樂，都可以說是運用演算規則進行作曲的實例。但是在二十世紀電腦科技出現之後，成為電腦音樂的一門重要研究方向。它使作曲家能夠專注在巨觀層次上去進行作曲，而不需要一個個設想、手動抄寫每個音符。同時，二十世紀許多現代樂曲樂風所講究的方法論，也與演算作曲相互呼應，本論文便透過演算作曲系統的實作來檢驗這點。

本論文透過將兩種現代音樂風格－－序列主義（serialism）及簡約主義（minimalism）的樂曲分析轉化為演算程式，對樂曲進行再創作來探討音樂風格與演算作曲的緊密關係。第一章首先探討過去既有的演算作曲，並且特別觀察互動作曲與純演算作曲的異同之處。第二章則檢視了過去幾種代表性的電腦演算作品及程式，了解各自的原理、操作方式、生成的樂風以及限制所在。第三章則是兩種音樂風格代表性樂曲《Structure Ia》及《Phrygian Gates》的基本結構分析，第四章則說明依據分析所建構而成的樂曲演算程式，整體的演算運作原理及操作方式。

依據樂曲分析所建構的演算作曲系統，除了可以使用原本樂曲的樂風進行再創作之外，還可以透過調變這些音樂特徵的相關參數產生新的樂風可能性。而本文所使用的 Max/MSP 物件導向程式語言，能夠快速地依據需要改變系統中各層級的結構。

關鍵字：演算作曲、自動作曲、互動作曲、序列主義、簡約主義、Max/MSP

Building an Algorithmic Compositional System Based on Music Analysis

Student：Chung-Chi Hsieh Advisor：Yu-Chung Tseng

Master Program of Sound and Music Innovative Technologies

National Chiao Tung University

ABSTRACT

From Mozart's dice music to John Cage's indeterminate music, many music works in history were composed using algorithmic techniques. However, with computer technology appeared in 20th century, the algorithmic composition becomes an important topic in computer music research. It lets composers focus on the macro-level of composition and do not need to think and compose note-by-note. At the same time, the many methodologies of contemporary music styles in 20th century are corresponding with algorithmic composition. This thesis wants to examine this fact by practically building an algorithmic composition system.

In this thesis, we transfer the music analyses of two contemporary music style, serialism and minimalism, into algorithmic programs, than examine the relationship between music style and algorithmic composition by re-composing. Chapter 1 discuss existing algorithmic compositions, especially on the similarities and differences between interactive composition and pure algorithmic composition. Chapter 2 is a survey of several classic works and programs of computational algorithmic composition, including their principles, operations, music styles and limites. Chapter 3 is the basic stractural analyses of two characteristic works of each styles, "Structure Ia" and "Phrygian Gates". Chapter 4 represent the algorithmic composition program we build according the analyses, showing its mechanism and methods.

Using this algorithmic compositional system based on analyses of musical works, we can not only re-compose music in existing styles, but also modulate these parameters of musical characteristics to create possibility of new music style. In this thesis, we using object-oriented program language Max/MSP, which can change the system structures in different level efficiently, based on what we need.

Key words: algorithmic composition, automatic composition, interactive composition, serialism, minimalism, Max/MSP

誌謝

在這個時代，聲響太容易成為視覺的附庸，甚至許多研究與應用均以娛樂與資本為準則。因此能夠有這樣傳承二十世紀前位音樂之傳統，將聲響作為音樂之擴充領域，不限於器樂地認真研究聲音藝術創作的學程，是非常難能可貴的事。能夠在此就讀，與具有相同志向的同學們彼此砥礪切磋，都是相當珍貴的經驗。

感謝這二年來老師們的教導與包容，感謝同學們的協助與合作。更要感謝父母無數的容忍與支持。當然，這一切都要感謝所有研究聲響、創作獨特作品的先鋒前輩們，因為有你們的努力，我們才能聽見這麼豐富精彩的聲響世界。

目錄

圖目錄

表目錄

第一章、緒論

1.1 演算作曲

電腦此一強大運算工具的出現，使得音樂的創作獲得前所未有的可能性。運用電腦來擴展音樂想法有兩種大方向，一種關注對於音色的創造變化，第二種則應用電腦的演算能力來生成音樂素材 [1]。後者稱為演算作曲（或準則作曲）。演算作曲不負責音色方面的深度調變，而專注在如何運用既有的音色進行樂曲創造。

這些演算法將作曲行為移升到高於個別音符的層次，其音樂特徵的演變是透過操控一定數量的參數來控制的 [2]。改變單一的參數，可能會同時影響作曲結果中大量音符的變化。相較於傳統音樂以音符為單位作的記譜作曲，透過演算法的建立，人們可以操作記譜作曲這個動作本身。因此，演算作曲以 Heinrich K. Taube 的話來說，就是「作曲的作曲」，或者「後設層次（metalevel）上的作曲」[8]。

雖然演算作曲因為電腦的出現而受到矚目，但是依據特定的形式化程序來創作樂曲的演算技法自古以來便存在著。中世紀的音樂理論家 Guido d'Arezzo 便提出了將詩文的母音分別對應到特定音高，使文章能自動生成旋律的演算作曲法 [10]。

而到了二十世紀，出現許多新式的作曲技法，更是利用新的演算規則，來打破人類既有的音樂慣性，產生新的音樂可能性。由 Arnold Schoenberg 開創的十二音列技法，以「八度中十二個半音音高類（pitch-class）在全部出現以前不重複」作為基礎規則，延伸出一連串樂曲演算的音列技術。十二音列的作品可以被視為以少數指定好的變形機制對一串音列進行持續的變奏 [9]。另外如極微派音樂則透過樂句不斷的反覆與相位（phase）、模式上的變奏，機動性地生成樂曲。

1956 年，依利諾大學教授 Lejaren Hiller 與 Leonard Issacson 創造了第一首電腦演算樂曲「Illiac Suite」（詳見後文），開啟了以電腦進行演算作曲的先例。另一方面，希臘作曲家 Iannis Xenakis 則先後以手工與電腦技術，持續運用各種數學界的理論與程序（包括最有名的、運用雲霧粒子運動機率參數的 stochastic 技法）來寫作樂曲 [11]。

就演算結果而言，演算作曲具有二大方向：創造獨特的準則來產生新的樂風、或藉由分析既有樂風來找尋生成演算法的發展可能性。Nick Collins 借用 Charles Ames 的詞彙對此進行了定義：

1. 主動樂風合成：意圖透過演算程序探索新奇的作曲結構。
2. 經驗風格建模：將演算程序用在歷史既有的樂風，進行樂理上的研究 [13]。

最早的電腦音樂系統由於設備不能負擔即時演出所需的大量數值計算，因此無法為互動而設計。這些系統需要花費好幾個小時編寫程式與等待，最後能聽到的可能只有一小段旋律[4]。 早期的電腦演算作曲在設定完演算程式後，使用者只能一次輸入數據、啟動，然後得到輸出的樂曲。雖然可以對輸出結果再進行必要的修改，對於演算過程使用者是完全沒有干涉的。因此作曲者必須對結果進行篩選程序，取得在美學上恰當的結果。而這意味著作曲者要耗費龐大的心力，將電腦能夠產生的大量輸出結果一一審查。

如何建構一個具多樣化、又能反應美學需求產生高水準作品的演算程式結構，也成為演算作曲的核心議題。同時利用電腦進行創造性的藝術活動，也使演算作曲與人工智慧領域產生關聯，許多演算作曲的研究會援引人工智慧研究的概念，研究人工智慧的經典程式語言 LISP 也被許多演算作曲家所運用 [8] [11]。

隨著電腦運算能力的日益強大，電腦程式使人們更容易對構思進行反覆的試作與檢驗，作曲者也開始運用電腦來快速生成、測試音樂構思，以探索的方式進行作曲，而不再需要電腦一次就將樂曲生成出來。電腦輔助作曲（computer-aided composition, CAC）的概念便因此而生 [5]。

依據使用者對於程式運作過程的參與程度不同，演算作曲也產生出不同的製作方向。常見的區分方式如 Eduardo Reck Miranda 將作曲軟體區分為「演算作曲」軟體及「電腦輔助作曲」軟體。前者在生成音樂時具有相當程度的軟體自主性，而後者則成為作曲者捕捉、組織音樂構想的協助工具 [9]。但是若要探討這兩種製作方向相互混合交雜的作曲程式現況，或許更應該採用 Otto Laske 的定義方式。他將電腦音樂作曲中人類參與的程度視為一整道光譜，光譜的兩個極端分別是「全手動作曲」：也就是人類作者直接對作曲過程中的每一個環節都親自作決定，以及「全自動作曲機器」，其程式生成樂曲的執行過程中間完全沒有人類介入 [14]。在這光譜之間，充滿著人類介入程度各有不同的電腦輔助作曲方式。

而人類要如何去介入電腦的生成過程、電腦要如何輔助人類的作曲結果，這就牽涉到「互動作曲」的議題了。

1.2 互動作曲

「互動作曲」一詞由 Joel Chadabe 提出，意指運用可即時演奏的電腦音樂系統來作曲與演奏。互動作曲包含兩個階段的程序：首先為建造互動作曲系統、其次為利用與此系統互動來作曲與演奏 [7]。互動作曲軟體通常集結了分析既有音樂而建構的各種模組，同時提供一套讓程式與作曲者溝通的介面，其外貌通常是符號或圖形物件 [11]。

由於即時的互動系統多半受到機器設備所限制，某些最尖端的科技只能在非即時系統下運作（雖然更快速的電腦處理器能彌補這點限制）。早期的即時電腦系統是 Max Mathews 與 F. Richard Moore 自 1968 年起開發的 GROOVE 系統。它可以讓人控制既成樂曲的速度與音符強弱，使操作者可以即時控制樂曲的最終合成演奏結果 [4] [12]。

自從 MIDI（樂器數位介面 Music Instrument Digital Interface）成為數位音樂的標準後，使音符訊號的傳輸處理能夠在相對低廉的個人電腦運算資源下進行，即時互動的電腦程式系統才得以普及開發。早期的商業化互動系統有 David Zicarelli 製作的 M 與 Jam Factory，而在同一時期，IRCAM 的 Miller Puckette 則創造了以串接數據處理物件的模組概念為特色的 Max 視覺化程式語言 [1] [4]。Max 在商業化之後成為電子音樂與多媒體領域的重要程式語言，本論文建構的程式實作也是基於 Max。

互動作曲與電腦的演算法息息相關。Todd Winkler 將作曲演算法定義為「作曲者物件」，其對於互動作曲來說有雙重意義：它們不但是對演奏者動作的回應機制，也能夠作為作曲者創作新曲的電腦輔助畫板 [4]。 互動作曲強調電腦直接產生出音樂結果，而作曲者可以聆聽音樂結果而直接、即時地改變既有的演算規則或輸入數值，再進一步影響接下來的音樂結果，透過這樣反覆的交互影響程序來形成最終的樂曲。人類的動作能夠影響與改變機器上的狀態及行為，而機器輸出之行為結果，也能反過來影響音樂家的思考與下一步動作的決定 [29]。除了生成音樂結果之外，電腦作為一個統整各種功能的機器，其互動的程度還具有繼續擴充的可能性：

. 機器分析：使機器不只是對作曲者的操作指令做出反應，更可以接收外界演奏或既有樂曲的 MIDI 音符訊號、即時做出分析，生成互動音樂結果。

. 機器聆聽：將聲音感知辨識能力進一步整合進來，使機器不需要外加音符訊號轉換介面，能夠直接聆聽現場音樂並進行分析而互動。

透過即時的互動，對於音樂作曲結果最重要的影響是在結構上。使用者輸入的每個參數變化，都直接影響到電腦輸出的音樂片段，使用者再聆聽這些片段做出判斷，繼續輸入參數變化。所有的音樂片段加總起來，才構成完整的樂曲內容。利用這種方式，作曲者可以將自

己的音樂美感判斷直接加入樂曲當中，而不需要如傳統演算作曲那樣，強制對音樂的每個環節都強加上難以控制、對美學上會造成不確定結果的演算成份，作者個人細緻的美感判斷無法靈活地影響作曲結果，而需要透過大量篩選等耗費心力的方式來彌補。

1.3 研究動機

本論文選用兩種現代音樂風格：「序列主義」以及「簡約主義」的代表性樂曲作為演算作曲程式實作的對象，其理由有二：首先，這兩種樂風均為二十世紀的重要現代樂風格，而它們均採用明顯的規則成為其樂風的特徵，因此我們從演算作曲的角度來理解它們的規則機制，能夠擴充演算作曲的發展脈絡，產生新的演繹可能。另外，這兩種樂風在音樂特徵及歷史發展上都是處於對極的狀態，我們將它們併置於同一個演算程式底下，從當中抽取出通用的參數使兩邊均可加以運用，則是企圖從其對極之處尋找出嶄新風格發展的可能性。

第二章、文獻探討

2.1 相關研究

2.1.1 Illiac Suite [12]

這首於 1956 年使用 ILLIAC 電腦所譜成的弦樂四重奏樂曲，是世界上第一首以電腦演算而成的音樂。這首四樂章的樂曲，每個樂章各名為一種「實驗」，分別採用了不同的實驗方式來生成樂曲。前三樂章主要是使用亂數隨機生成音符數值，再透過給定的各種音樂規則（對位法禁忌等等）篩去不合格的數值，最後一樂章則利用馬可夫鏈（Markov chain）對各音符出現的機率做出許多種不同的調整方式。

作為早期的電腦演算樂曲，Illiac Suite 的作曲系統自然沒有即時互動的成份。其最終的成品也是以樂譜形式完成，再交由真人樂手演奏，而不是由電腦產生音樂結果。但是當中的關鍵手法：隨機與馬可夫鏈，都是後來演算作曲經常用到的基本技術。

2.1.2 EMI（Experiments in Musical Intellegence） & Emily Howell

David Cope 的 Experiments in Musical Intellegence（簡稱 EMI）是知名的演算作曲系統，主要用於調性音樂創作，可以依據特定音樂風格生成樂曲。EMI 整合了 David Cope 歷年研究的眾多演算方式，其核心機制不是推論式的演算生成規則，而是對過去既有的大量樂曲進行分析，重組（recombination）[10]。在整體樂曲的結構分析中，Cope 引進了延伸自 Heinrich Schenker 的分析模型：「 SPEAC」：

Statement 敘述

Preparation 準備

Extension 延伸

Antecedent 前樂句

Consequent 後樂句

各元素的可能後繼元素被限定，主要以緊張（tension）/ 鬆弛（relaxation）的情緒機能作為分類依據，其實是來自和聲學的轉化 。除此之外，系統也透過樣式比對（pattern-matching）搜尋出一些在結構上具有明顯特色的素材，在重組過程中保持其結構完整，使樂曲能保持風格上的特徵 [10]。這套系統在生成過程中不做即時互動，其風格的生成仰賴資料庫，難以彈性調動規則，同時必須對生成結果進行大量篩選，可能要從幾千次輸出成果中選擇符合美學要求的樂曲 [17]。

近年來， Cope 不再使用 EMI，而以歸納式關聯（inductive-association）的概念為中心建構全新的程式，稱為 Emily Howell。使用者要運用口語問答的方式對程式解釋所需要的作品，並輸入一定的樂曲或樂句資料。程式再依據這些問答與資料，以其歸納式關聯網路創造出樂曲 [33]。雖然在演算前輸入樂曲需求的程序上具有互動的成份，但由於演算運行的過程中並無人類的參與，Emily Howell 並不算嚴格定義下的互動作曲。

2.1.3 Music Mouse

Music Mouse 是一款「智能樂器」（intelligent instrument）作曲程式，由曾在貝爾實驗室工作的美國作曲家 Laurie Spiegel 於 1985 年製作。使用者可以即時選擇好幾種可能的音階（半音音階、八聲音階、中東音階…等等）、速度、移調等參數控制。其介面是利用滑鼠在電腦螢幕上的晶格間移動，控制二維的音高範圍，其餘參數操作則利用電腦鍵盤輸入熱鍵，可即時生成符合和聲的多聲部旋律 [6]。

Music Mouse 的操作介面十分簡易，但是使用電腦鍵盤難以同時靈活變化多種參數。另外使用者對樂曲結構所能操作的變化只有在和聲的參數變化上，旋律變化上只有固定的幾組循環模式，難以彈性地改變生成規則。

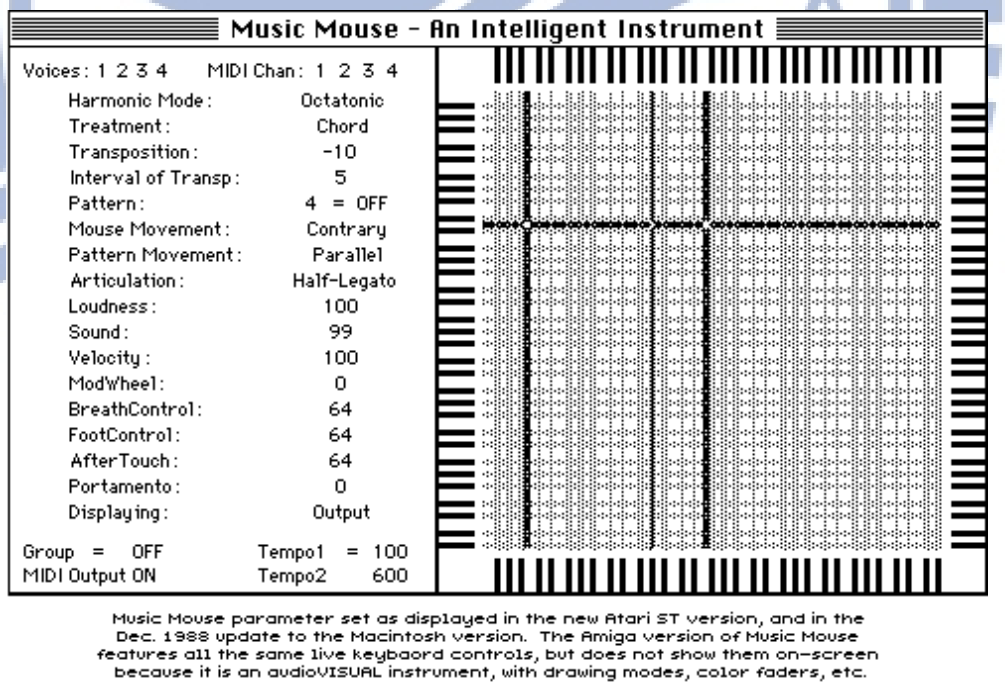

圖 1　Music Mouse 在 Apple Macintosh 電腦上的程式介面[38]

2.1.4 Pat-Proc [3][1]

這套互動作曲程式是由美國作曲家、曾經於交通大學擔任聲音與音樂創意科技碩士學程客座教授的 Phil Winsor 於 1991 年發表。正如其名，其原理主要運用了模式處理（pattern-process），主要用於極微音樂創作。程式結構分為兩個部份：第一部份生成音高素材，而第二部份則進行各種操作來控制這些素材。在第一部份音高素材的生成上，使用者可以

- 手動輸入個別音高
- 使用限制方式篩選音高
- 定義基本音程，再由程式依據這些音程產生音階

第二部份則以各種方式上述產生的音高結果來生成線性單音聲部。其提供的方法包括序列排序、亂數分佈、或者以音程的最小 / 最大值來對音高進行篩選。

Pat-poc 最終是以樂譜檔案形式輸出，因此它並非用來即時互動演出。但是它的演算特徵卻是演算生成法的範例。其提供不同互動自由度（從全手動到全自動）的概念，對於互動程式的製作也是很好的參考。

2.1.5 Lexicon-Sonate [13][15][16]

Lexikon-Sonate 是由奧地利作曲家 Karlheinz Essl 自 1992 年起持續研發的即時作曲程式。作者本人不將它視為可製作各種樂曲的作曲工具，而看作一首完整的作品 - - 一首具有無限可能、永遠也聽不完的鋼琴樂曲，當程式啟動時，我們聽到的就是樂曲的「片段」。其輸出樂曲可透過內建 MIDI 音源，或接上遙控鋼琴 Yamaha Disklavier 即時演奏。

本程式以 Max 寫成，Essl 引進了許多演算作曲的基礎模組（後來集結成為他的「即時作曲程式庫（Real Time Composition Library, RTC-lib）」），再依據對既有樂風的分析（其中包括當代作曲家 Karlheinz Stockhausen 與 Gottfried Michael Koenig 的演算作曲技法）建構出二十四種結構產生器。

在實際演奏時，演奏者可依據需求或亂數決定，即時呼叫出最多三種結構產生器來生成旋律，同時根據他們的權重（Essl 區分為前景、中景、背景）大小重新啟動運算，並依序改變權重。

Lexikon-Sonate 一開始是製作成全自動的作曲系統，後來逐漸被研發成一種樂器，當中的許多作曲參數可以用外接的 MIDI 滑桿控制器操作。這使得使用者完全不需要碰到琴鍵，就可以即興演奏出高度複雜的鋼琴音樂 [16]。

從現場演奏 [32] 可以看出，其樂曲的大結構是透過切換結構產生器來轉換段落，主要的音符結構生成是由電腦負責，而演奏者以滑桿控制器控制音符參數（多半是音高與音符強度），在某些段落中也透過電腦鍵盤來觸發樂句模組，有時則觸發固定的段落過門。

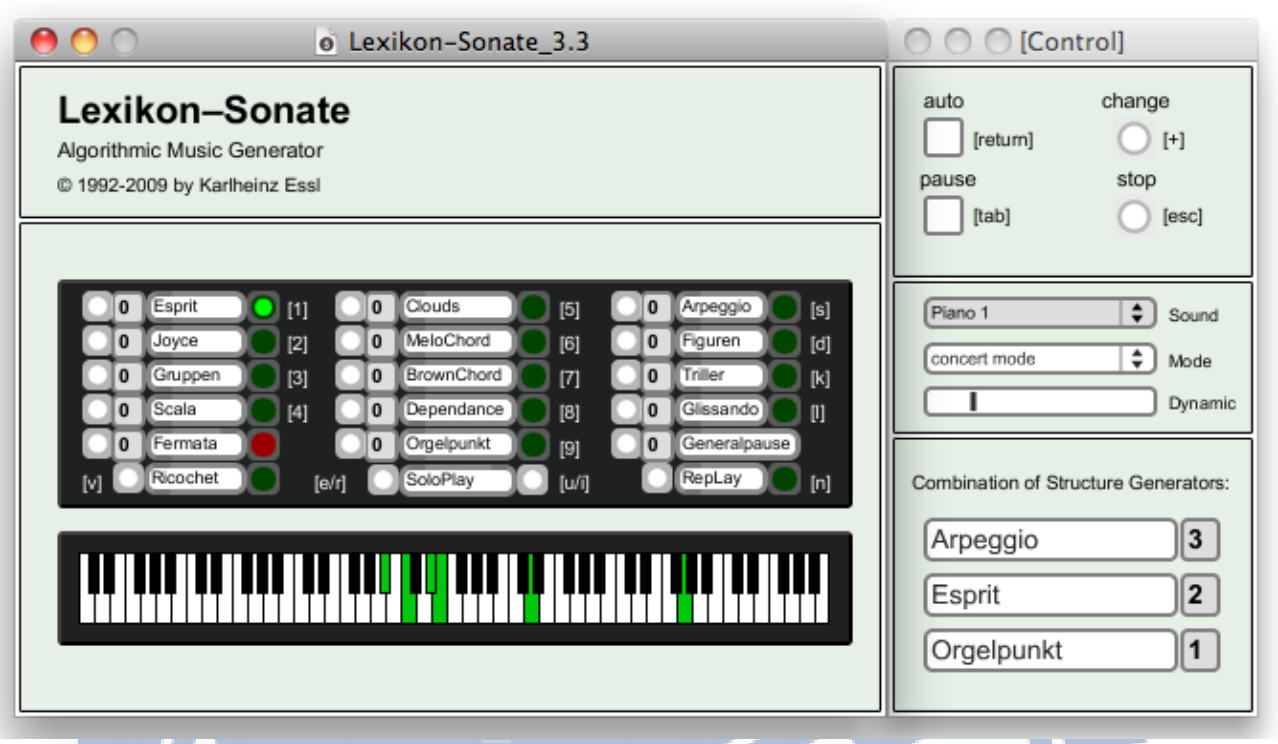

圖 2　Lexikon-Sonate 的操作介面[39]

2.1.6　WebernUhrWerk [34]

這是 Karlheinz Essl 於 2005 年為紀念新維也納樂派音樂家 Anton Webern 所做的生成音樂程式。本程式利用了 Webern 未完成作品的音列，除了隨機對音列做出置換變化之外，並設定四種節奏模式，除了 MIDI 樂器的選擇外，使用者只有兩個模式可以使用，一個是讓音樂固定在每個整點當中隔十五分鐘觸發生成一次，或者全自動地持續變換生成。節奏模式的關鍵演算為 RTC-lib 中的「super-rhythm」物件，此物件能夠產生隨機而彈性的節奏，並隨設定的機率與音符數產生和弦。

本論文所實作的程式將用來製作互動程式的 RTC-lib 加上自動觸發機遇選擇，使程式可以完全自動執行，可以說是互動演算法能夠與自動演算法互換的證明。

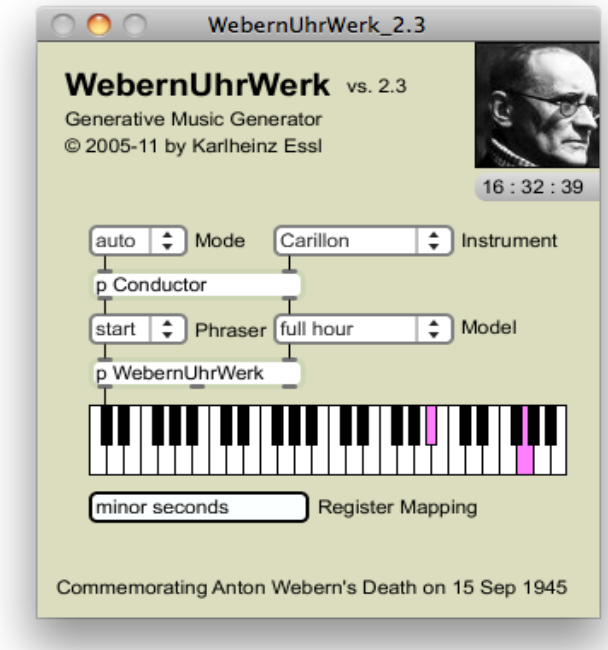

圖 3　WebernUhrWerk 的操作介面[34]

2.2 音樂風格

2.2.1　序列主義

序列主義（serialism，又稱全序列主義 total serialism）繼承著新維也納樂派建立的十二音列技法，並進一步擴充至音高以外的其他音符參數。這些參數包括節奏（長度）、速度、強度（響度）、奏法（觸鍵法）、密度（聲部數量）與空間安排（八度分佈）等等。原始的音高數列會依據某些制定好的方法轉換成一連串數值，這些數值則可以被用在其他參數上面[23]。 作曲者首先要決定要引進的元素、序列與操作程序，然後執行預先決定好的序列操作程序。第三階段則要決定其他沒有經過預先程序的音樂參數並做調整，使其落在允許的參數範圍內 [23]。

Boulez[1] 於 1952 年寫作的雙鋼琴樂曲《Structure I》是序列主義的代表樂曲之一。這首曲子所使用的音列來自 Olivier Messiaen 的鋼琴音樂作品《Mode de valeurs et d'intensités》[23]。曲子中所有音符的音高類、長度、強度、觸鍵法（articulation）等，均是透過指定的序列及其矩陣所生成出來的。當中的序列處理程序幾乎可以全自動地生成這首作品：除了音

1Pierre Boulez（1925 年 3 月 26 日 - ），法國作曲家、指揮家、音樂理論家。作為指揮家，他指揮過很多有名的交響樂團，自 50 年代中起，Boulez 就和 Karlheinz Stockhausen，Luigi Nono 並稱為先鋒派三大代表，特別是在序列音樂方面。

高序列以外，Boulez 也據此定義了音長序列、強度序列與觸鍵法序列。序列的處理程序決定了樂曲的整體結構，以組合的方式將這些序列統整起來 [9]。由於遵照音長序列的關係，兩台鋼琴在每個段落中的演奏時間長應當要完全相等。由於其複雜的規則，這首樂曲也以高難度彈奏著稱。

Boulez 說明本作品的概念在於：「我希望抹除我樂曲語彙當中一切傳統的痕跡，無論是在音型、樂句、結構發展或者在曲式上面；我希望一步步、一個元素接著一個元素地奪回曲式當中的各種不同階段，以顯露出一種完美的新合成體。 […] 另外，我追求著一種思維，要將這些至今仍然處於衝突狀態的音樂語言相貌加以標準化 [18]。」

2.2.2 簡約主義

簡約主義（minimalism）一詞原用於美術領域，轉借至音樂領域用以形容以幾位作曲家：La Monte Young, Terry Riley, Steve Reich, Philip Glass 等人為中心發展起來的音樂風格 [22]。

簡約主義音樂（簡稱極微音樂）在創作樂曲時，往往採用最小程度的基本原始素材（音符等等） [21]。 這些素材透過一定程度的法則（再加上作曲者機動性的增補修改），可產生出綿長不斷的音樂結果出來。

雖然同以簡約主義命名之，但各作曲家的風格與技術其實有很大的差異。 La Monte Young 運用調式系統，並依賴產生長音（drone）及建構在這長音上的和聲音頻（泛音）變化，配器上融合東西方器樂、甚至使用合成器產生正弦音波 。Terry Riley 則重視多重的反覆，而且重視反覆的自由變化。在合奏樂曲《In C》中，Riley 寫下幾組不同的旋律模組，演奏者可以自由組合這些模組的演奏順序，使得每一次的樂曲演奏都會有不同的結果。Philip Glass 則重視音樂的旋律性，反覆雖然作為樂曲的規則，但是對主題會不斷地疊加並複雜化 [24]。Steve Reich 則將音樂視為「一個漸進的程序」[24] [25]，在反覆的規則上著重相位的變化，以兩組演奏者從齊奏開始反覆演奏同一樂句，而其中一組演奏速度會微微調慢，使得二者的節拍會逐漸錯開來，而錯開的長度隨時間而逐漸拉大，中間的每個錯開的狀態都會造成和聲與總體樂句上不同的聆聽感受 [36]。

除去純粹概念性的作品，David Cope 將極微音樂定義為「相位與模式音樂」 [21]。 在極微音樂中，模式的反覆成為樂曲的核心關鍵。同時對模式的各種調變方式（變化拍子、增減音符、移調等等）則是造成樂曲實質變化、構成樂曲結構的技術所在。極微音樂被視為傳統音樂的反動：它的結構變化不遵循傳統音樂的張力結構，也不再進行傳統音樂所作的情緒表達。Wim Mertens 稱傳統形式的音樂為「辯證式音樂」：「在辯證式音樂中，實際的戲劇性奠基於形式與內容的對抗以及這種對抗的最終解決，但是當邏輯因果被移除時，聲音變得獨立自主，因此在進程式的作品中，沒有結構能先於聲響存在；它是在每個瞬間當中被創造出來的 [22]。」 雖然極微音樂的結構相對自由，但是要達成這種樂風的效果，就必須要進

行持續的反覆與變奏。

2.3 MIDI 規格

MIDI 的全名為「樂器數位介面（Musical Instrumental Digital Interface）」，是一種數位電子樂器的通訊規格。MIDI 主要用來轉換音樂演奏或控制的相關事件參數（如彈奏琴鍵、以滑桿控制聲音調變、甚至啟動舞台視覺特效等等）[26]。

MIDI 本身就是將音樂參數轉換成一連串的數字來傳遞，因此若對這些數字進行運算處理，就等於對聲音參數進行了演算，而這也就是對傳統器樂作電腦演算作曲的原理所在。以音高參數為例，MIDI 設定中央 C 的數值為 60，因此以中央 C 作為主音的半音音階就如下圖。

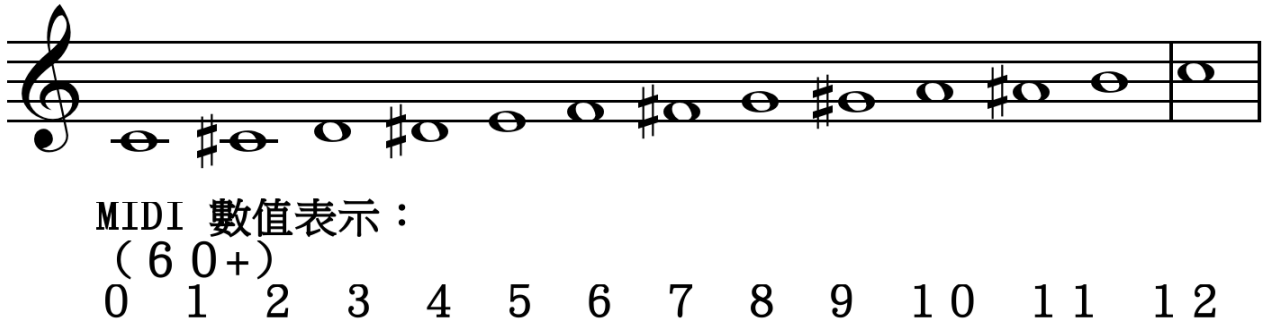

圖 4　以中央 C 作主音為例，半音音階與 MIDI 音高參數的代換

將各音階的實際音高數值減去中央 C 的數值，就可以得到半音音階的相對音高。

2.4 Max/MSP

Max/MSP 是利用物件銜接達成即時聲音處理的物件導向程式語言，其名稱取作「Max」是為了感念電腦音樂之父 Max Mathews 開創電腦聲響的貢獻 [30]。

1985 年 IRCAM 研究員 Miller Puckette 在 IRCAM 研發的數位聲音處理器 4X 上寫出 Max 的原始版本。1988 年被 Opcode Systems 買下商業發行權後，經過 David Zicarelli 發展而成為商業版本，之後不斷擴充改版、先後加入即時信號處理元件集 MSP 與影像處理元件集 jitter，發行在各種個人電腦作業系統上，成為十多年以來聲音藝術 / 電子實驗音樂最主流的程式之一 [30]。

Max/MSP 對比起以往的程式語言有許多特色，如「物件導向圖像式的撰寫方式」、「通過標準化的步驟，可以快速簡易地寫出程式」、「程式運作原理簡易、便於理解」、「能夠連續進行程式編輯與實行步驟，有利研發效率」、「具有充實的多媒體相關機能」…等等 [27]。其最特出的圖形編寫方式，讓使用者能夠有效地思考程式整體的運作原理及流程，

機動性地依據需要做出增補或刪改。

第三章、研究方法

3.1 研究方法

本論文使用 Max/MSP 物件導向程式語言，實作一套可生成鋼琴音樂作品的程式。我們對兩種特別受到演算法影響的現代音樂樂風的代表性樂曲：序列主義的《Structure Ia》與簡約主義的《Phrygian Gates》作生成機制上的分析，並將分析結果以演算程式重現，使其對原始樂曲做出一定程度的再現。接下來再加入互動或自動的參數變化機構，使整體生成能具有更高的作曲自由度。

3.2 樂曲分析

3.2.1 序列主義：Pierre Boulez《Structure Ia》

《Structure I》全曲分為三個部份，其中的 a 部份有完整的分析 [19]，常被人作為序列主義的教學範例。本論文所建構的程式也是依據這些分析所轉化而成的[2]。

全曲利用這種方式生成了四十八條（每台鋼琴各走完兩次矩陣序列）旋律線，Boulez 再將它們分為十四個段落，每一段落中，兩台鋼琴各擁有零到三條旋律線 [20]。「鋼琴１」與「鋼琴２」的總時值都是１～１２的總和，消耗音列材料的速度是一致的；也就是說，兩台鋼琴如果同時開始的話，必定也是同時結束 [31]。

本論文不對原樂曲的複雜規則作全盤的說明，只在介紹演算物件時解釋其所處理的規則與方法。同時在演算程式上也提供了序列主義的源頭－－十二音列技術的簡易演算模式（詳見 4.2.1.8）。

2對於書中的所有分析，筆者均有對照樂譜計算驗證過。因此發現書中兩台鋼琴的觸鍵法矩陣是相互顛倒的。

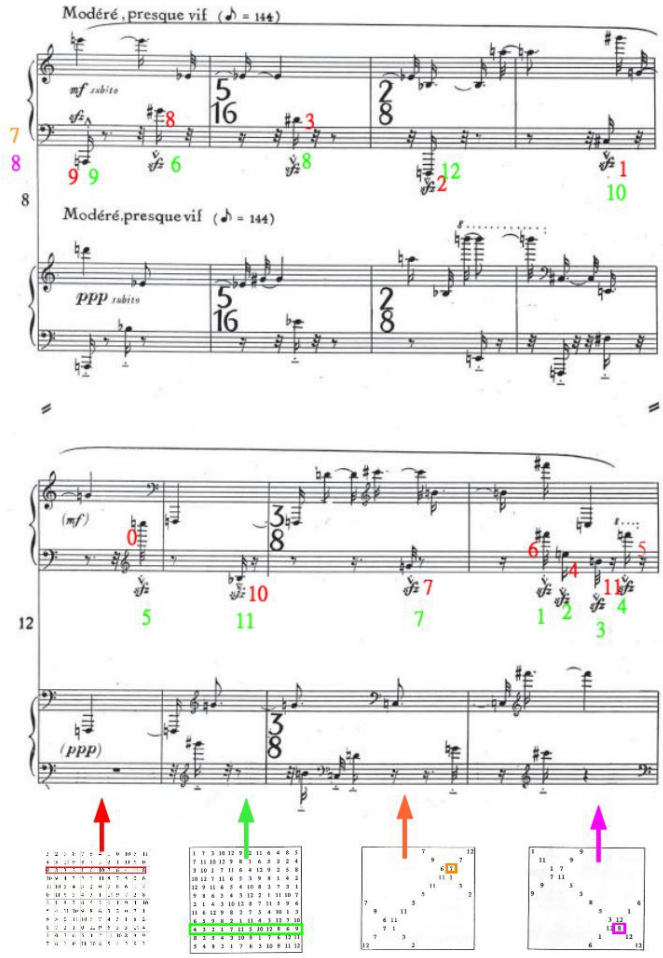

圖 5　《Structure I》的譜例，以樂曲中第三條生成音列為例。紅色、綠色、色、紫色分別代表音符的音高、長度、強度與觸鍵法。這幾個參數分別根據不同的規則、從不同的序列矩陣當中彼此依序合成起來，形成一整條旋律線。

3.2.2　簡約主義：John Adams《Phrygian Gates》

鋼琴音樂作品《Phrygian Gates》是 John Adams[3] 於 1977 到 1978 年所作。正如其名，樂曲以 phrygian 調式（可視為將音高類 4 當作開頭的大調音階）作為結構中心，設想一個在兩種調式之間切換的正弦波，曲名中的「gate」來自於電子電路的「閘」概念，表示樂曲在調式變換上就如通過閘門般直接變化而不經調變 [37]。在五度圈上變換主音（A→E→B…），在每一個主音上又交替變換著 phrygian 與 lydian 調式，而且隨著樂曲進展， lydian 的段落會越縮越短，而 phrygian 段落會越來越長 [37]。 除了大段落的音階調式變換外，持續的兩條

3　John Coolidge Adams（1947 年 2 月 15 日 - ），美國當代古典音樂作曲家，簡約主義音樂的代表人物之一。亞當斯的音樂結合了簡約主義音樂和浪漫主義的部分風格，其歌劇《尼克森在中國》是 20 世紀下半葉最著名的歌劇作品之一。

旋律線也不斷變化著其反覆的迴圈模組，每一次改變就造成樂曲的狀態變化 [37]。

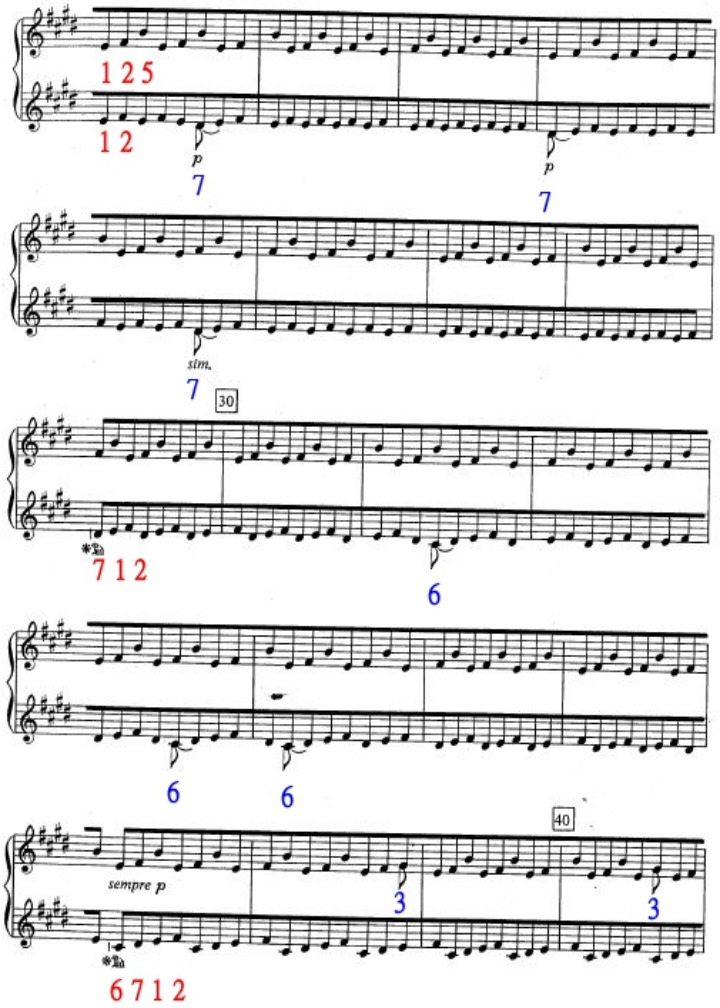

圖 6　《Phrygian Gates》的譜例，其中紅色為會出現多次反覆的旋律模組，藍色則為間歇插入的變化音符。（音符編號方式以大調音高類 C 為 1）

第四章、程式建構

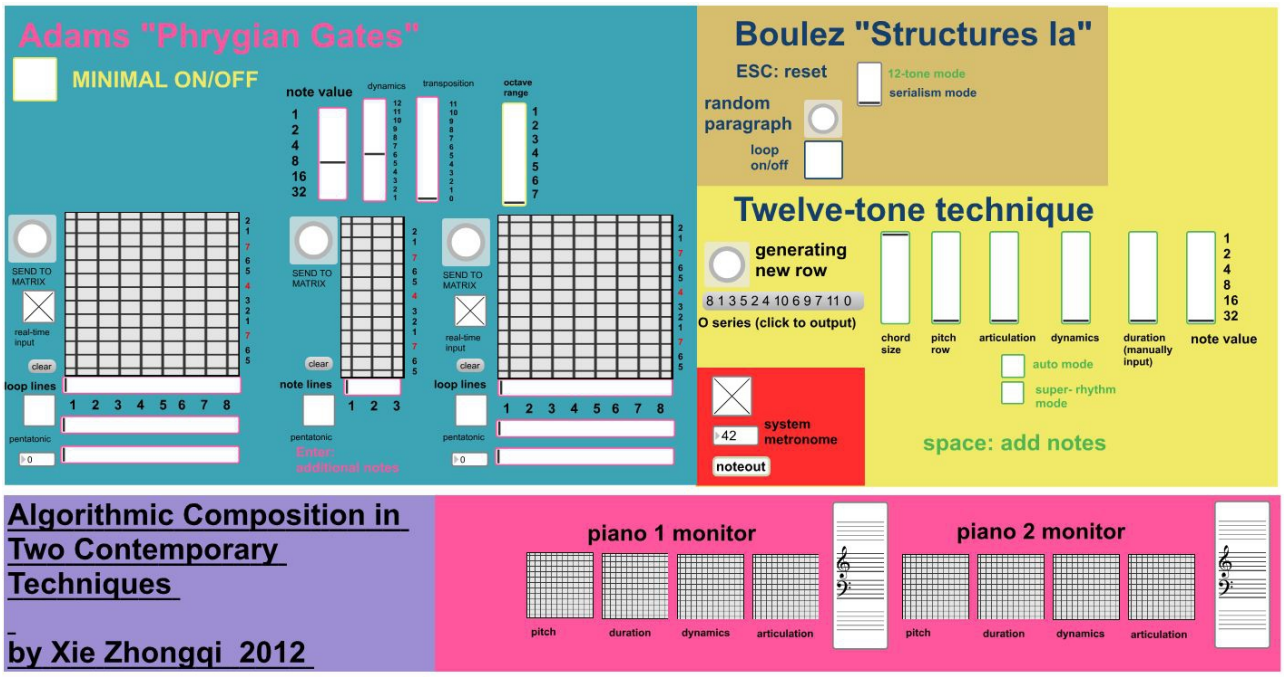

圖 7　本程式的操作介面

4.1 系統架構

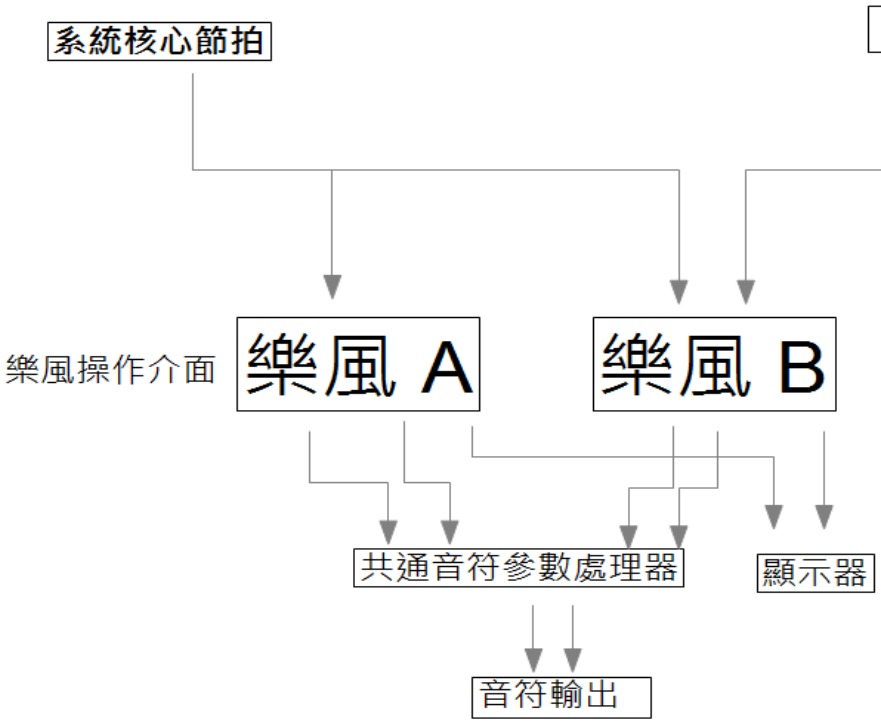

圖 8　程式系統架構示意圖

4.1.1 系統核心節拍

整個系統使用共通的節拍單位（為音符內容可能到達的最小時間長，在本程式中為 32 分音符），如此能使整個系統擁有共通的節奏而不致拍子錯位。同時節拍單位的實際時間長度可以機動調整數值，使整體樂曲速度能即時調整快慢。

4.1.2 樂譜

並非傳統意義下的音符樂譜，而是記載樂曲基本演算法則、各段落中的基本參數，與段落間切換的指令集合。當啟動樂譜時，程式將根據演算法與紀錄的參數，自動生成出原本的樂曲。在本程式當中，樂譜內容為《Structure Ia》的自動執行指令與參數。

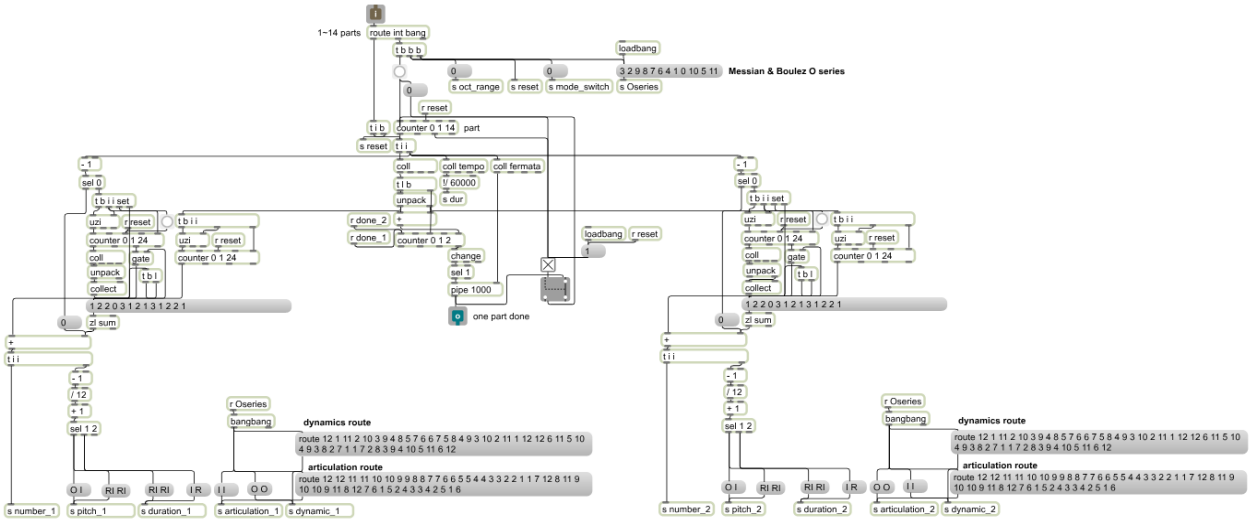

圖 9　本程式當中的《Structure Ia》「樂譜」

4.1.3 各樂風演算與操作介面

我們將每一種樂風視為一種演算系統，它們可接受共同的輸入參數、輸出相同的音符參數集合，同時各參數也保留給使用者即時操作的選項。彼此之間也有相互銜接以產生更複雜的演算結果的可能。在本程式中，我們採用了三種樂風：十二音列、序列主義與簡約主義。

4.1.4 共通音符參數處理器

上述樂風演算產生的參數組合，要經過共通的參數處理器（在本程式當中，參數組合元素來自於《Structure Ia》的演算法，包括音高、時長、強度、觸鍵法與休止模式，參見後文），進行轉換之後才變為音符訊號。不直接用樂風演算法產生音符、而要多經過這樣一個步驟，是為了保持樂風之間的參數交換的可能性。在本程式中，就將序列主義定義的音符參數採用在簡約主義樂風上面，使其演算可以相互對應。

4.1.5　音符輸出

輸出 MIDI 音符訊號，除了直接利用 MIDI 音源器播放以外，亦可接上外加的音源系統，甚至透過電路控制實體樂器。

4.1.6　顯示器

各樂風的的演算元件會同步送出訊息顯示在顯示器元件上，在本程式為顯示序列行進路徑的矩陣圖及輸出的音高大小（以傳統樂譜格式顯示）。顯示器讓使用者能即時看到關鍵性的結構變化，進一步在操作上能作準備。同時利用顯示器也能夠檢驗演算元件是否運行正確。

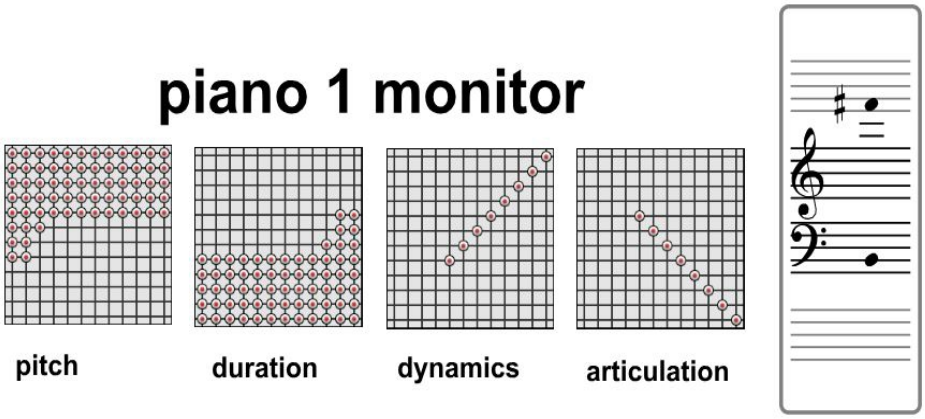

圖 10　本程式的矩陣顯示器

4.2 演算系統

4.2.1 序列主義

《Structure Ia》使用的數列如下：

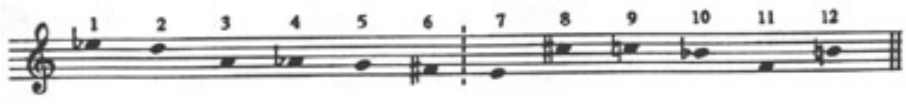

圖 11 《Structure Ia》使用的數列 [19]

我們將其音高類化為數列則可寫作：3 2 9 8 7 6 4 1 0 10 5 11。整首樂曲的主要參數皆是以這個數列及其所形成的矩陣所演算生成的。但是這裡的矩陣並非如傳統十二音列音樂一般，僅使用由原型 - 反行作為 x - y 軸所構成的音列矩陣，它還使用了原型 - 原型、反行 - 反行所構成的矩陣，同時在這種矩陣中，各參數的大小是對應到原型音列的音高次序（即上圖的阿拉伯數字）而非音高大小。

'O' matrix

1	2	3	4	5	6	7	8	9	10	11	12
2	8	4	5	6	11	1	9	12	3	7	10
3	4	1	2	8	9	10	5	6	7	12	11
4	5	2	8	9	12	3	6	11	1	10	7
5	6	8	9	12	10	4	11	7	2	3	1
6	11	9	12	10	3	5	7	1	8	4	2
7	1	10	3	4	5	11	2	8	12	6	9
8	9	5	6	11	7	2	12	10	4	1	3
9	12	6	11	7	1	8	10	3	5	2	4
10	3	7	1	2	8	12	4	5	11	9	6
11	7	12	10	3	4	6	1	2	9	5	8
12	10	11	7	1	2	9	3	4	6	8	5

'I' matrix

1	7	3	10	12	9	2	11	6	4	8	5
7	11	10	12	9	8	1	6	5	3	2	4
3	10	1	7	11	6	4	12	9	2	5	8
10	12	7	11	6	5	3	9	8	1	4	2
12	9	11	6	5	4	10	8	2	7	3	1
9	8	6	5	4	3	12	2	1	11	10	7
2	1	4	3	10	12	8	7	11	5	9	6
11	6	12	9	8	2	7	5	4	10	1	3
6	5	9	8	2	1	11	4	3	12	7	10
4	3	2	1	7	11	5	10	12	8	6	9
8	2	5	4	3	10	9	1	7	6	12	11
5	4	8	2	1	7	6	3	10	9	11	12

圖 12　《Structure Ia》所使用的原型及反行矩陣，數字取自原始音列的音高次序 [19]

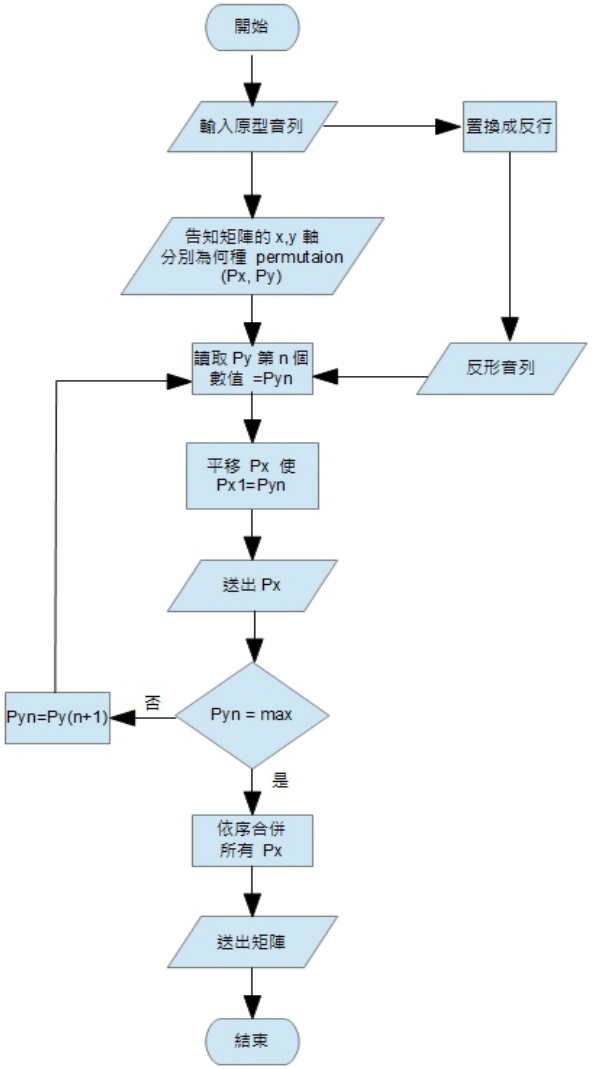

圖 13　矩陣生成的演算流程圖

為了能統一運算各種參數不同的矩陣演算方式，我製作了專門處理矩陣路徑用的子程式：「bot」。

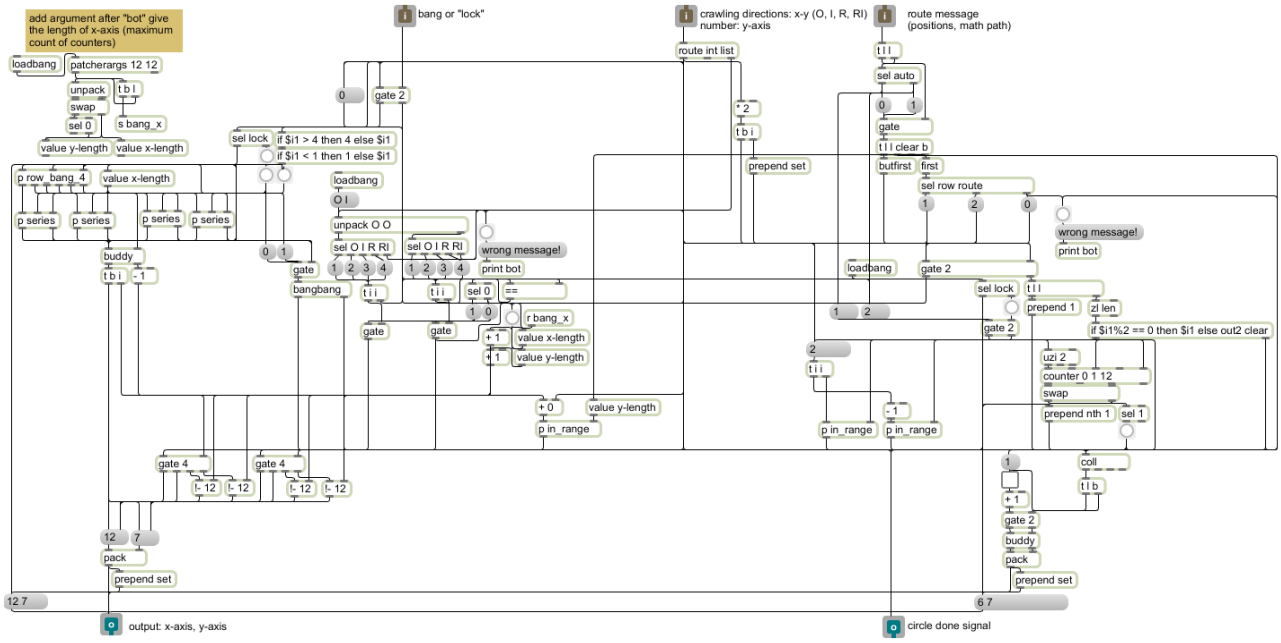

圖 14　子程式「bot」的 Max/MSP 結構圖

「bot」可以設定音符啟動訊號將在矩陣上面以什麼樣的路徑前進。它可以用原型、反行、逆行、逆反行方式，或者以特定的路徑行進方式（以 list 形式輸入）前進。當設定好路徑後，當它收到觸發訊息時，就會依路徑順序輸出矩陣座標，這些矩陣座標套用在相對應的矩陣上，就可以得到一個參數輸出值。以下幾個使用矩陣演算的參數都會利用到這個子物件。

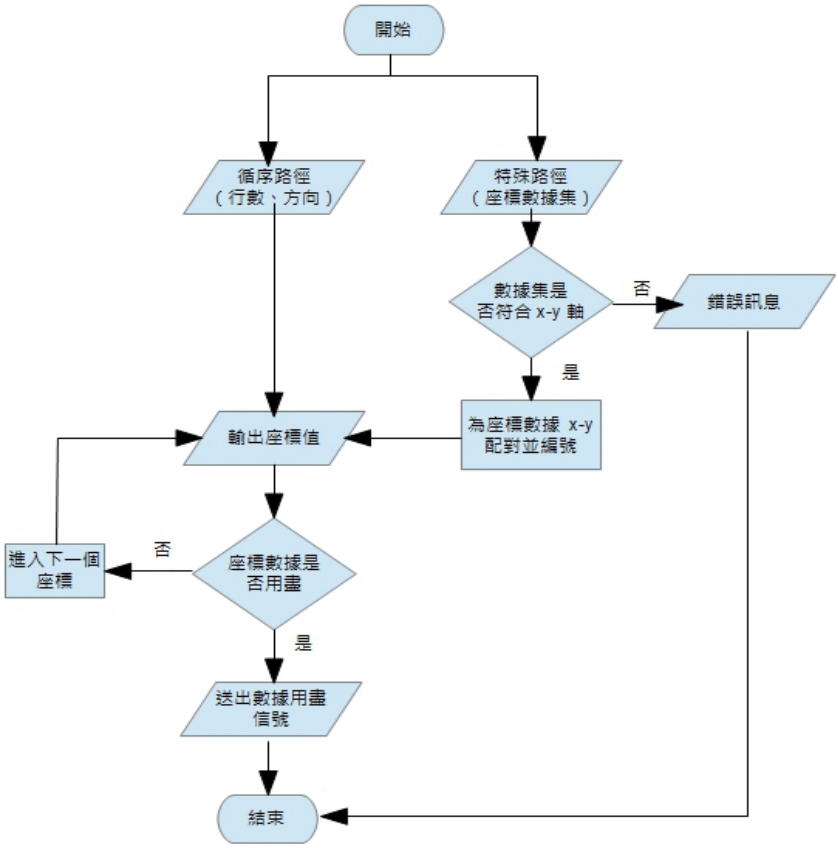

圖 15　子程式「bot」的演算流程圖

4.2.1.1　音高 / 長度

這兩種參數分別以四種排列置換（permutation）的數列交替構成不同的矩陣，每一種參數皆會跑完兩次矩陣。（以第一鋼琴的音高來說，會先後跑完 x-y 軸分別為 O-I、RI-RI 的兩個矩陣。）因此程式會先輸入矩陣 x-y 軸的排列置換，再讓「bot」依序經過這些矩陣。從對照表可看出，長度與序列參數的關係剛好是以３２分音符為單位去乘上參數值（比如說，參數值若為６，此音符長度就為六個３２分音符＝８分附點音符）。其中要注意，音高參數是根據原始音列的音高大小，長度參數卻則是根據原始音列的音高次序。因此長度參數輸出後，還要經過一次轉換程序。

表 1　長度與序列的關係表 [19]

4.2.1.2　強度 / 觸鍵法

這兩種參數使用了原型及反行矩陣，並且以特殊的方式前進。

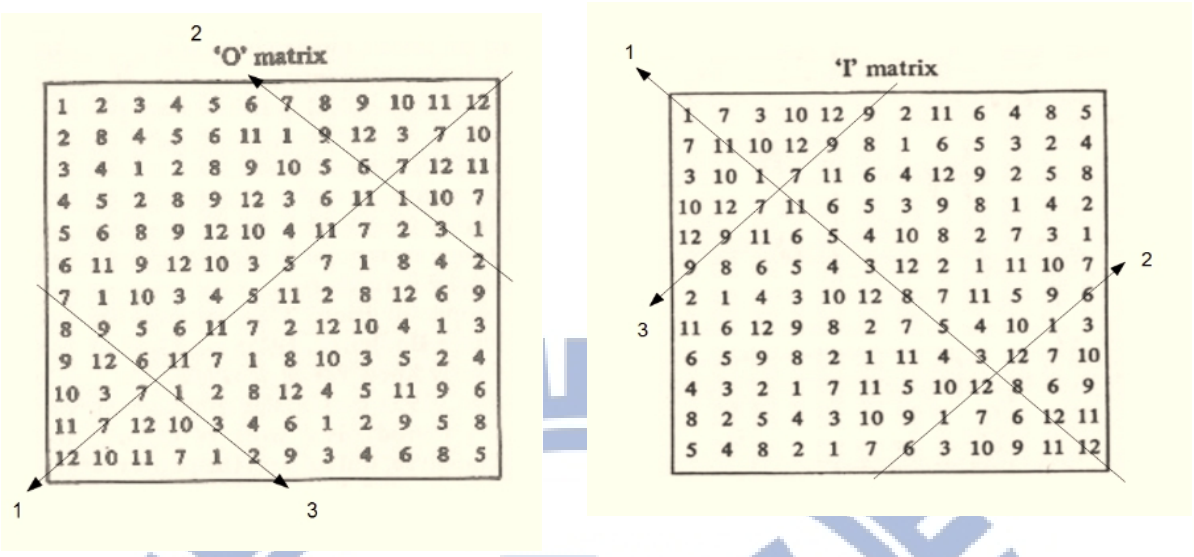

圖 16　強度與觸鍵法的行進路徑（以鋼琴 I 為例）

為此，程式預先將「bot」輸入如此的特定路徑，再依序送出。注意這兩種參數的大小也是根據原始音列的音高次序。因此長度參數輸出後，還要經過一次轉換程序。

這兩種參數所對應的都是記譜符號不是完全線性的音符參數，還需要進一步根據樂理及 MIDI 訊號規則加以轉換才能變成正式的 MIDI 訊號。

強度對應數列編號的大小，從最弱（pppp）排到最強（ffff）。而其與 MIDI 訊號大小的對照表如下：

Dynamic's note velocity

Dynamic	Velocity*	Voice
ppp	16	Whispering
pp	33	Almost at a whisper
p	49	Softer than speaking voice
mp	64	Speaking voice
mf	80	
f	96	Louder than speaking
ff	112	Speaking loud
fff	126	Yelling

Decrescendo (diminuendo)　Crescendo　Accent

*Note velocity adopted from Logic Pro

表 2　強度符號與 MIDI 訊號大小的對照表 [35]

而對照表上沒有的 pppp 我們定義為 10、ffff 定義為 127（MIDI 訊號的強度最大值）。

觸鍵法與序列的關係如下：

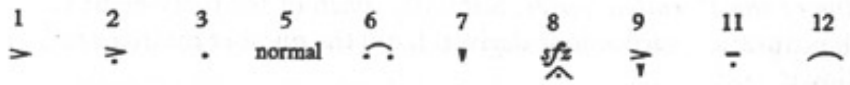

表 3　觸鍵法與序列的關係表 [19]

可注意到，關係表上沒有４與１０，這是因為依照樂曲所制定的觸鍵法矩陣路徑，並不會出現這兩個數字。

觸鍵法是對樂手彈奏音符的方式的指示。在音樂內容上，它實際影響到的是各音符的實際長度與強度。各種觸鍵法的參數一般而言是由樂手自行拿捏，但其實它還是有標準的對應方式，例如斷音（Staccato）表示演奏的音符長度約為原音符的一半。我們參照了樂理書籍的解釋 [28]，得到如下的代換子程式：

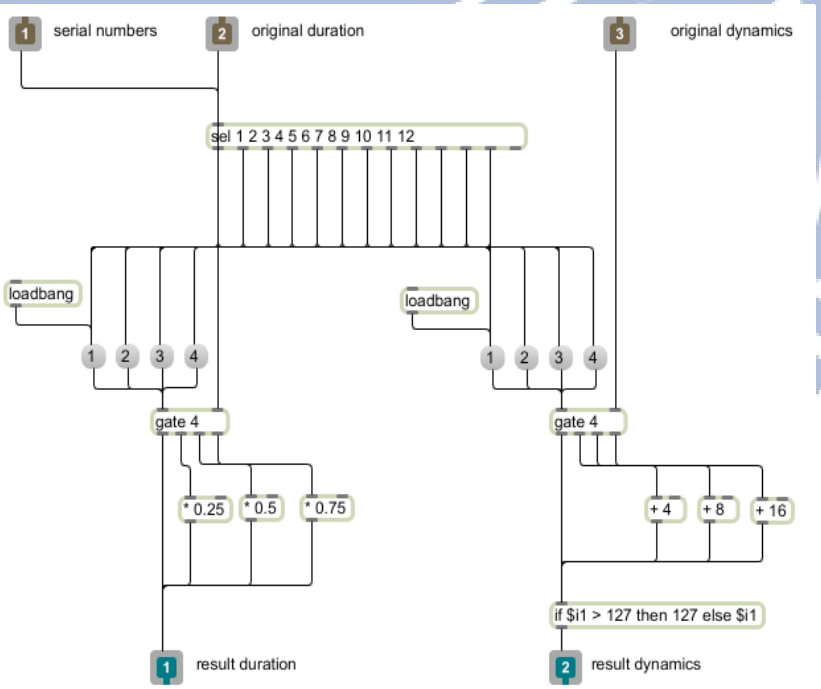

圖 17　觸鍵法的代換子程式

4.2.1.3　八度區域

《Structure Ia》各音符所處的八度區域並沒有嚴格的法則，但是它們基本上會交錯開來，各聲部前後兩音符的八度區域不會重疊。因此我利用下列的子程式，將輸入的音高類參數放入

不重複的隨機八度區域。子程式內的八度區域可以涵蓋鋼琴的 88 個琴鍵，也就是原始樂曲的所有音高範圍。如果將原始樂曲中各音符的八度區域確實記錄下來，依序加上輸入的音高類參數，則我們可以得到與原始樂曲完全相同的樂曲內容。

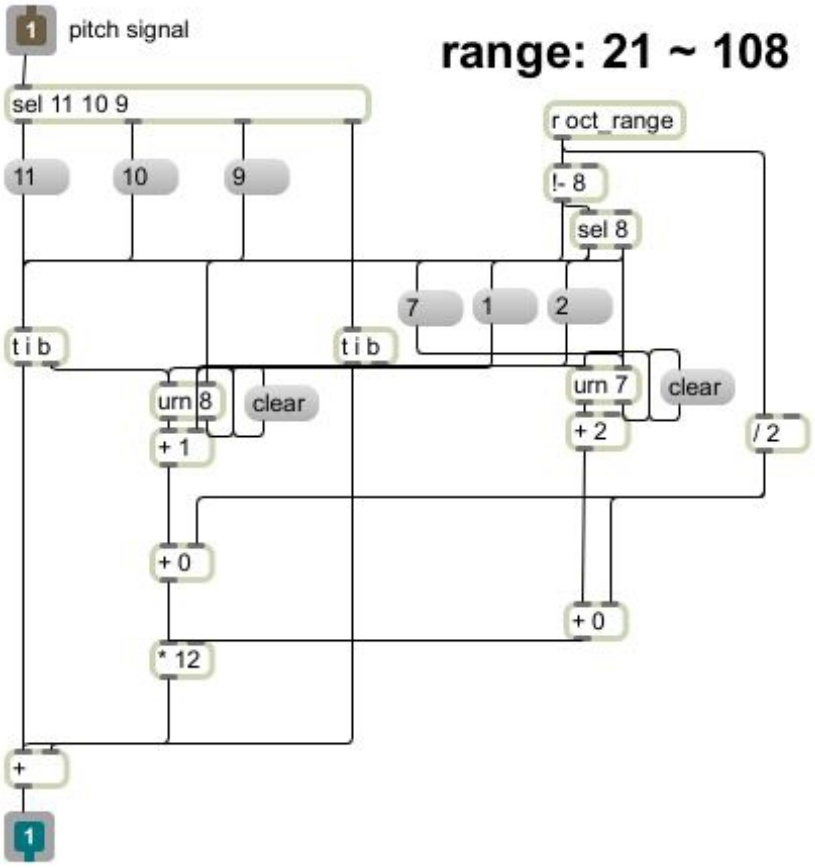

圖 18　八度區域的處理子程式

4.2.1.4　休止方式

在本樂曲中，每次十二個序列的實際音符長度有兩種計算方式，一種是直接帶入先前所演算出的長度序列大小，另一種則序列中音符長度各只有二到三個３２分音符，多餘的時間則全部為休止。兩種休止方式的出現順序並無特定安排，因此本程式直接將原樂曲每個序列中的休止方式記錄下來，當音符長度參數經過子程式時，會根據相應的段落而改變音符長度、加入休止符。

4.2.1.5　速度

《Structure Ia》中的十四個段落有三種不同的速度：lent（１６分音符 = 120）、 très modéré（８分音符♪ = 120）、modéré, presque vif（８分音符♪ = 144）。要從音樂記譜的速度

表示法轉換成單位音符（３２分音符）的毫秒數，公式就為「（一分鐘有 60 秒×每秒有 1000 毫秒）÷（節拍音符單位為３２分音符的幾倍×每分鐘有幾拍）」。以 lent 為例，就是 (60×1000)÷((32÷16)×120)=250 毫秒。換算過來，各單位音符的毫秒數分別為「250、125、104」，各段落採用哪一種速度並沒有明確的演算規則。本程式也是將其全部紀錄下來，依據段落切換而輸入系統核心節拍。

4.2.1.6　段落間延音記號

本樂曲各段落結束後有時會出現延音記號，表示段落之間的小休止，本程式也是將其全部紀錄下來，在段落結束之後，根據段落輸出，使啟動下一段落的信號稍作延遲。

4.2.1.7　全序列與十二音列的切換

為了使程式能夠囊括規則較為寬鬆的十二音列樂曲風格，我加入了只啟動音高參數序列演算的十二音列模式。利用右上角的滑桿，可以在兩種模式間切換。

圖 19　切換全序列與十二音列模式的滑桿

4.2.1.8　十二音列樂曲的控制

由於傳統十二音列技法只對音高類進行嚴格控制，其他參數則擁有完全的自由，可根據作者需要作安排。若能將音高序列的規則進一步搭配各作者的需求、符合作者美學的演算程式，就可產生出各種各樣完全不同風格的十二音列樂曲來。在此，本程式只內建了兩個示範性的演算模式：亂數自動生成（auto mode）模式與 super-rhythm 模式，搭配手動的即時觸發功能。

亂數自動生成模式會以完全隨機方式產生各音樂參數的數值而觸發音符，不斷產生出符合十二音列規則的旋律來。而 super-rhythm 模式則是由 Karlheinz Essl 撰寫的 RTC 物件（見 2.1.5），能夠產生在一定程度固定而又具有柔軟變化的觸發時值 [40]，使得生成的音符更有樂句的感覺。這兩種模式均控制著音高以外的音符參數，使生成的音樂結果能在遵循十二音列的規則的同時，產生許多的變化。

同時在十二音列模式中還設置了手動觸發機能，讓操作者可以按壓空白鍵來觸發音符，同時各種參數（包括和聲，也就是同時發出的音符數之可能性）均可透過滑桿控制。在此，由於音高的可能順序已經完全由 bot 子程式控制，因此無論是自動生成、手動觸發或者兩種

模式並用，都絕對不會背離十二音列的規則。

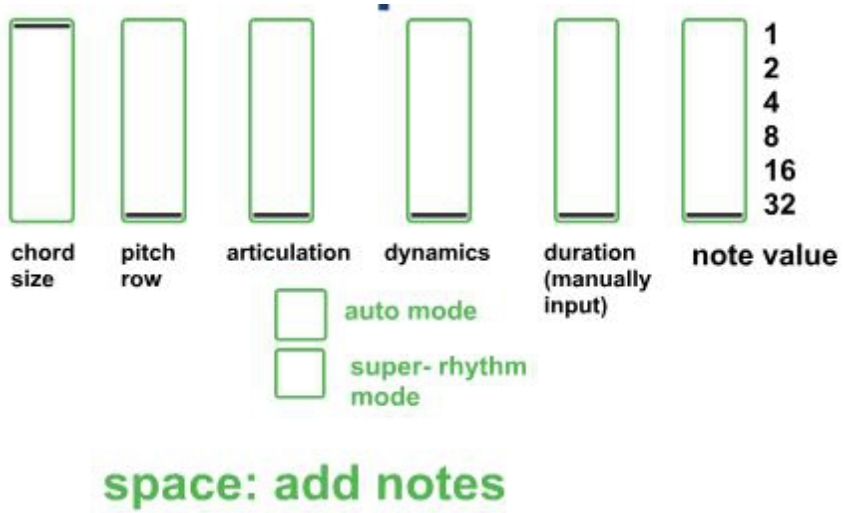

圖 20　可自由隨機控制十二音列音符參數的滑桿組

4.2.2 簡約主義

《Phrygian Gates》的結構變化主要來自反覆模式的改變及不定時（但非隨機）穿插進來的音符。因此本程式的簡約主義演算元件分成兩部份：

4.2.2.1　模式控制介面

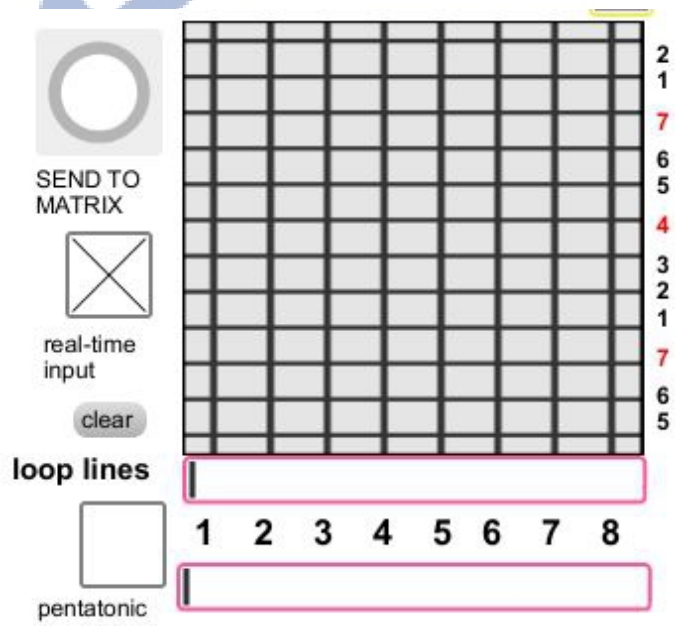

圖 21　簡約主義的模式控制介面（單一聲部）

介面的中心為一組 12 乘 8 的矩陣物件，橫軸（由左至右）為時序、縱軸為相對音高

（預設數值為大調音階的相對音高，從 G_{n-1} 到 D_{n+1}，n 為整數）。使用者可以在矩陣物件上開啟或關閉任意座標，當一矩陣座標顯示為開啟（紅點）時代表會送出音符。矩陣物件底下的滑桿可以控制反覆模式會送出的橫軸時序，當滑桿數值為 1 時，程式會不斷反覆矩陣最左一行的音符；若滑桿數值為 8，則程式的反覆模式就會是從最左行一直到最右行。

當演算元件啟動時，模式控制介面在預設上會將矩陣上的任何改變即時輸入音符產生模組。但是為了使模式保有瞬間變化的可能性，即時輸入功能設定為可開關。當關閉時，音樂的反覆模式對於矩陣物件上的任何改變都不會即時反應，而要按下左上角的觸發鍵才會送出新的反覆模式。

為了確保操作模式介面時的和聲和諧程度，介面左下方另有一個五聲音階的開關。當打開時，矩陣上面音階次序為 4 與 7 的音符（分別為大調音階的 F 與 B，不存在於五聲音階當中）都會被過濾掉，使輸出的模式必定落在五聲音階上。

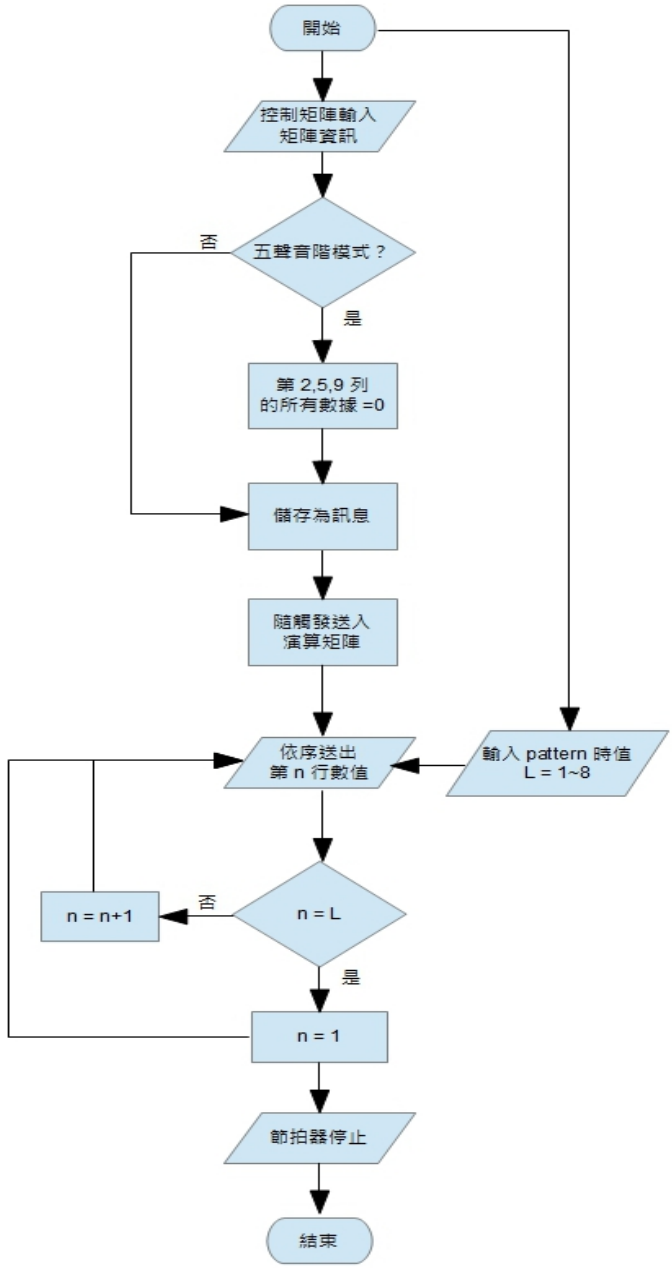

圖 22　簡約主義的模式演算流程圖

控制模式演算的子程式構造如下圖。介面上的模式會先依序呼叫出模式上各行列中，各音高數值是否存在（為 1 或 0），連帶矩陣位置的參數（x-y 軸）一同輸入右下方的訊息物件中。當受到[bang]觸發時，訊息物件會輸入左側的矩陣中，再由左側的矩陣接收依序輸出每一行音高數值的訊息，形成音高。

當五聲音階開關打開時，每次的[bang]會同時觸發最右邊的演算程序，將每一列音高類為 4 與 7 的音符存在值皆變為 0，使得最終輸出之音高訊息不會出現這些音符。

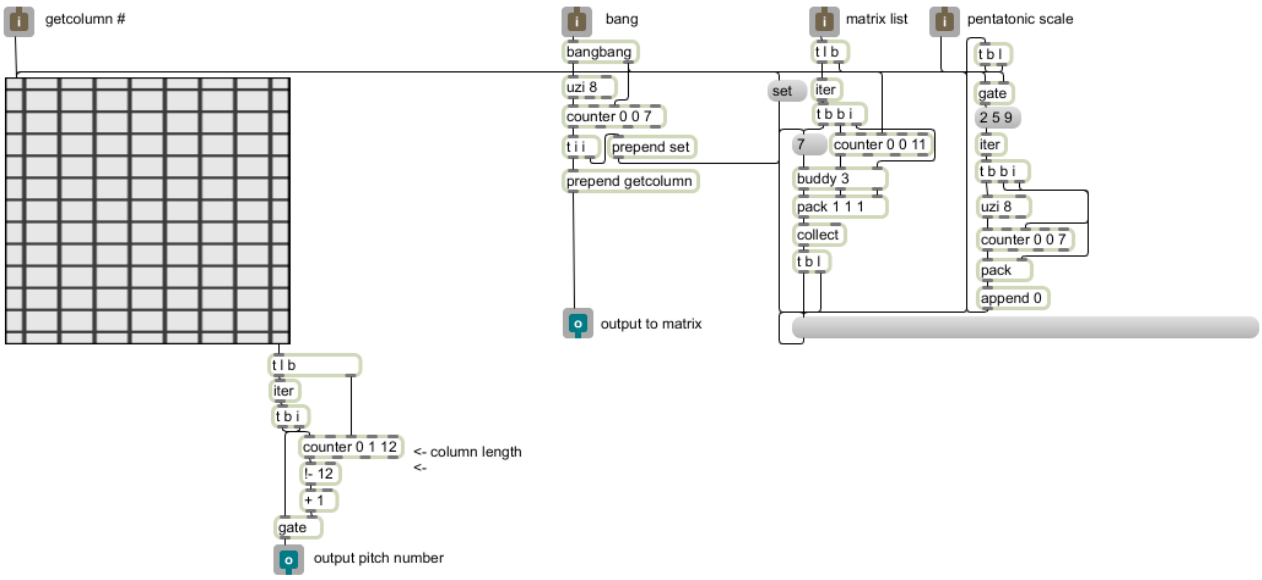

圖 23　簡約主義的模式演算子程式（單一聲部）

4.2.2.2　穿插音符

這部份則是簡單地將反覆模式中的輸出音符訊號作出調換。穿插音符的控制介面與模式的控制介面類似，但是時程上只有最高三個音符單位的長度。當使用者按下 enter 鍵時，從演算模式輸出的音符觸發訊號會改通往穿插音符的子程式當中，依序觸發穿插音符後再回到模組子程式上。這樣設計的用意在於確保穿插的單音能自動服貼在反覆模式的拍子上，而沒有任何拍子錯落的風險。

由於穿插音符是輸入後再視時機啟動，因此沒有即時輸入產生音符的開關。

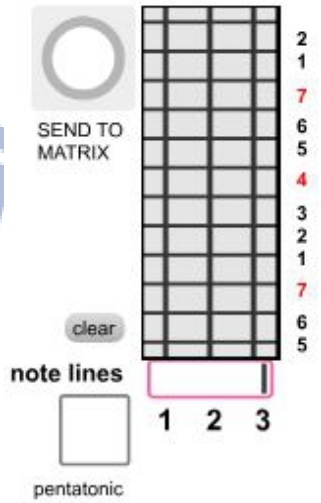

圖 24　簡約主義的穿插音符控制介面

4.2.3 樂風共通操作物件

序列主義樂曲中含有大量八度區域的跳躍，相反地簡約主義絕大多數時候只在一個八度區域內運作。為了使兩種樂風能有彼此過渡，本程式加上了八度區域隨機範圍的控制滑桿。當滑桿移至最底部（輸出數值為 7）時，八度區域的隨機範圍將遍佈整個鋼琴鍵盤音高範圍；相反地當滑桿移至最頂部（輸出數值為 1）時，若不另作移調，音符只會固定在中央 C 以上的八度區域內移動。

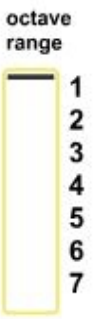

圖 25　控制八度區域範圍隨機程度的滑桿

4.3 操作模式

4.3.1　既有樂曲的生成與引用

本程式當中建構了完整的《Structure Ia》曲式演算法，只要按下 b 鍵，就可以直接生成整首曲子。而簡約主義的《Phrygian Gates》則是提供了樂曲的速度、音符長度等參數，只要按下 a 鍵，就可以自動輸入這些參數。

4.3.2　自動隨機生成

《Structure Ia》的 14 個段落可以不照順序隨機生成。按壓隨機選取段落的紐，就會隨機生成 14 個段落當中的任一段落。若將反覆開關打開，則可以不斷地隨機選取段落生成，產生出永不停止的《Structure Ia》。

十二音列模式也有自動模式（詳見 4.2.1）。

4.3.3　互動模式

序列主義模式的基礎序列，預設值為《Structure Ia》使用的音列。但是使用者可以按壓序列產生按鈕隨機產生新的序列，再選擇想要的序列按壓輸入整個系統。

圖 26　隨機產生新序列的按鈕

在十二音列模式中，使用者可以按壓空白鍵來觸發音符，同時利用各滑桿來操控音高之外的音符參數。

簡約主義模式則含有大量互動成份，使用者必須自己對矩陣物件輸入反覆模組型態、控制反覆時序，同時可操作滑桿來控制音符長度、強度與移調。

第五章、結論與未來展望

5.1 結論

在本論文中，我們透過分析樂曲，建構了可以根據演算法生成兩種現代音樂風格的程式。其生成控制方式包括了自動與互動。本程式可以透過演算生成《Structure Ia》的樂曲，達到除了八度區域外完全再現的程度。同時還可以依據序列主義、十二音列及簡約主義風格參數的變化法則，進行自動或互動的樂曲生成程序，產生全新的樂曲。

我們也發現，《Structure Ia》中定義的各種音樂參數對應法，同樣可以應用在極微音樂的生成上面。二者在音符處理生成的基本概念上具有共通之處，只是相較於序列音樂不斷地變化音符的各種參數數值，極微音樂的數值往往沒有那麼劇烈的變化而傾向穩定。

利用 Max/MSP 進行演算作曲的好處在於，我們可以依據需要機動性的改變演算法，同時程式也便於拆解為許多演算子程式的組合。當有需要修改或增加功能時，只需要針對個別的演算子程式進行修改，而不需每次都影響到整體架構。以此延伸，將樂曲內的各種音符、結構參數分開，以各自的演算子程式進行計算，最後再組合起來，這樣的程式才容易以最小的改動達到最有效的樂曲創作可能性。

在程式的建構過程當中，每個處理步驟、數據流程的先後次序非常重要。兩個鄰近的步驟，如果先後次序相互對調，其處理結果可能完全不同。而在 Max 寫作當中，光是依靠「由上而下、由右至左」的程式基本規則，有時不容易確認、控制各子程序的次序。因此我們往往需要靠[trigger]、[buddy]、[swap]等物件來確保或機動對調這些次序。

5.2 未來展望

5.2.1 搭配傳統樂譜擴充程式

Max 主要的功能在於將 MIDI 信號作為數值上的轉換處理，因此所預設搭載的傳統樂譜只有很簡單的以音符顯示音高功能而已。若能搭配樂譜處理用的外掛程式如 Note for Max、MaxScore，相信可以作到更細緻的樂曲編寫。

5.2.2 多樂器編曲

目前的演算方式都是以單一種樂器來進行，若要結合多種樂器的組合進行演算，必須將各樂器本身與相互影響的特性均轉化為演算法則。如此才能使演算程式駕馭更多種音樂曲式、創造更複雜的音樂結果。相對來說，多樂器的樂曲可視為單一樂器樂曲的擴充，因此只要能掌握單一樂器樂曲的原則，其餘樂器間的演算法則可以搭建在這上面。

5.2.3 自動跟譜

自動跟譜（score following）技術能使電腦程式即時聆聽現場樂手的演出、判斷演奏到樂譜的什麼位置，同步做出相應的聲音處理。傳統的自動跟譜採用一套音符完整寫定的既定樂譜，而電腦正如字面所示，一個音符一個音符地去跟隨樂手的演奏 [4]。若是加上適當的感應與分析介面及程式，則演算系統也可以搭配樂手的演出即時運算合適的伴奏等音樂內容。

5.2.4 結合其他程式語言

Max 雖然屬於圖形介面的程式語言，但它一直保有能使用其他文字程式語言（如 Java、JavaScript 或 perl，甚至 C 語言）結合運用的空間。這些程式語言對於 Max 來說是各有長才，尤其在數學演算的自動化上面，可以作到比 Max 更加簡潔的程式。若能適當地結合它們，必定能更有效率地編寫演算規則。

5.2.5 畫面美學更直覺、美觀

本程式的視覺介面僅使用 Max 內建的物件外型，並沒有作很複雜的介面設計。若要進一步改良視覺介面，除了加上圖像設計（物件顏色大小配置、加掛介面圖片檔等等）之外，操作的介面也可以作得更直覺。例如將操作用的矩陣元件與顯示用的矩陣結合起來，使矩陣顯示輸出音樂參數的同時，使用者也可以在其上面直接操作音樂內容進行互動。

參考文獻

外文：

[1] Robert Rowe. Interactive Music Systems. MIT Press 1993.
[2] Robert Rowe. Machine Musicianship. MIT Press 2001.
[3] Phil Winsor. "PAT-PORC: An Interactive, Pattern-Process, Algorithmic Composition Program." Proceedings of the International Computer Music Conference. San Francisco: International Computer Music Association, pp. 114-117, 1991.
[4] Todd Winkler. Composing Interactive Music: Techniques and Ideas Using Max. MIT Press 1998.
[5] Eduardo Reck Miranda (2009). “Preface: Aesthetic Decisions in Computer-Aided Composition.” Contemporary Music Review, 28:2, 129-132.
[6] Thom Holmes. Electronic and experimental Music. Routledge; 3rd ed., 278-279, 2008.
[7] Joel Chadabe. “Interactive Composing: An Overview.” Computer Music Journal. Vol. 8, No. 1. CA: People's Computer Company: 22-27.
[8] Heinrich Taube. Notes from the Metalevel: An Introduction to Computer Composition. Routledge, 2004.
[9] Eduardo Reck Miranda. Composing Music with Computers. Focal Press, 2001.
[10] Gerhard Nierhaus. Algorithmic Composition Paradigms: Paradigms of Automated Music Generation. Springer Verlag, 2009.
[11] David Cope. New Directions in Music. Waveland Pr Inc; 7th ed., 2000.
[12] Charles Ames. "Automated Composition in Retrospect." Leonardo, MIT Press, 1987.
[13] Nick Collins. Introduction to Computer Music. Wiley, 2009.
[14] Otto Laske. “Composition Theory in Koenig's Project One and Project Two”, in The Music Machine. MIT Press 1989.
[15] Karlheinz Essl. "Lexikon-Sonate. An Interactive Realtime Composition for Computer-Controlled Piano." Proceedings of the II Brazilian Symposium on Computer Music, 1997.
[16] Karlheinz Essl. "Algotithmic Composition." The Cambridge Companion to Electronic Music, 2007.
[17] Keith Musutt. "Composing with Algorithms: An Interview with David Cope". Computer Music Journal, MIT Press 2007.
[18] Markus Bandur. Aesthetics of Total Serialism: Contemporary Research from Music to Architecture. Birkhäuser Basel, 2001.
[19] Reginald Smith Brindle. The New Music: The Avant-garde since 1945. Oxford University Press, USA; 2 edition, 1987.
[20] Lynden DeYoung. 1978. "Pitch Order and Duration Order in Boulez Structure Ia", Perspectives of New Music 16, no. 2:27–34.
[21] David Cope. Techniques of the Contemporary Composer. Schirmer, 1997.
[22] Wim Mertens. American Minimal Music: LA Monte Young, Terry Riley, Steve Reich, Philip Glass. Pro Am Music Resources, 1988.
[23] Leon Dallin. Techniques of Twentieth Century Composition: A Guide to the Materials of Modern Music. Wm. C. Brown Company Publishers; 3 ed., 1974.
[24] Michael Nyman: Experimental Music. Cage and Beyond. Cambridge University Press,1999.
[25] Steve Reich. “Music as a Gradual Process” in Audio Culture: Readings in Modern Music. Continuum, 2004.
[26] David Miles Huber. The MIDI Manual: A Practical Guide to MIDI in the Project Studio. Focal Press; 7th ed., 2007.
[27] 赤松正行 + 左近田展康。トランス Max エクスプレス。リットーミュージック, 2011。

中文：

[28] 張錦鴻。基礎樂理。大陸書局, 1991。

[29] 曾毓忠。「電子/電腦互動音樂初探」。中華民國電腦音樂學會會刊; 復刊 3 民 92.04; 頁 6-13。

[30] 謝仲其。「全球聲學與音樂研究中心發展簡介」。數位藝述, 第貳號; 2012.11; 頁 151-152。

[31] 鄭英烈。序列音樂寫作教程。2 版, 上海音樂出版社, 2007。

網路：

[32] Lexikon-Sonate: laptop performance by Karlheinz Essl. Retrieved 2013-01-30, from http://www.youtube.com/watch?v=WxtAGzlikNM

[33] Ryan Blitstein. Triumph of the Cyborg Composer. Retrieved 2013-01-30, from http://www.psmag.com/culture/triumph-of-the-cyborg-composer-8507

[34] Karlheinz Essl: WebernUhrWerk (2005-2011.) Retrieved 2013-01-30, from http://www.essl.at/works/webernuhrwerk.html

[35] Wikipedia, Dynamics (music.) Retrieved 2013-01-30, from http://en.wikipedia.org/wiki/Dynamics_%28music%29

[36] Matsumoto Akihiko. “Max/Msp Tutorial: Phasing (Steve Reich).” Retrieved 2013-01-30, from http://akihikomatsumoto.com/maxmsp/pianophase.html

[37] John Adams. “John Adams on Phrygian Gates and China Gates.” Retrieved 2013-01-30, from http://www.earbox.com/W-phrygiangates.html

[38] Computer Software by Laurie Spiegel. Retrieved 2013-01-30, from http://retiary.org/ls/programs.html

[39] Karlheinz Essl: Lexikon-Sonate - algorithmic music generator. Retrieved 2013-01-30, from http://www.essl.at/works/Lexikon-Sonate.html

[40] Max Objects Database: super-rhythm. Retrieved 2013-01-30, from http://www.maxobjects.com/index.php?v=objects&id_objet=1561

附錄一、程式操作熱鍵

本程式除了面板上的操作介面之外，亦制定了某些熱鍵來操作。以下列出各熱鍵與機能作為參考。

b	將 Structure Ia 整個演奏一次
space	即時觸發音符，與 auto mode/super-rhythm mode 可並行使用
a	設好 Phrygian Gates 的音符參數，實際音高則由下面左右的 matrix 輸入
Enter	即時觸發添加的音符，實際音高則由下面中間只有三行的 matrix 輸入
p	關閉程式的核心節拍器，使樂曲暫停

附錄二、Max/MSP 常用物件簡介

Max/MSP 內建許多功能便利的物件，以下簡介一些程式寫作時常用到的物件以作參考。若希望瞭解完整的物件清單與功能，可閱讀 Max/MSP 內建的說明文件。

loadbang	此物件在程式開啟時會自動送出能啟動物件機能的「bang」訊息，可用來控制程式的各種預設值。
select	可簡寫為 [sel]，其能夠篩選輸入的數據是否與物件所設參數一致。
gate	閘門物件，能將輸入的物件送入不同的流通路徑。
send / recieve	可簡寫為[s] / [r]，當這兩個一組的物件參數相同時，[s]所接收的數據就會從[r]送出。
urn	不重複地隨機送出給定範圍內的數值。
buddy	接收複數個數據，當所有輸入端均接收到數據時才會一同送出。
uzi	同時送出給定數量的「bang」訊息。
pack	接收複數個數據，結合成單一個 list 訊息。
counter	能夠依序計數，除了給定數數範圍外，也可以指定是正數還是倒數。
iter	將輸入的單一 list 訊息拆解為個別數據依序送出。
trigger	可簡寫為[t]，其能夠指定一連串的訊息類別（「bang」、整數、浮點數…等等），當收到數據時，會依照順序送出這些訊息類別。
swap	兩個輸入口接收到數據時，會左右顛倒過來再輸出。
bangbang	可簡寫為 [b]，其具有複數個輸出口，當收到一個數據時，會從各輸出口依序送出「bang」訊息。
coll	能夠接收複數個相同或不同類型的訊息，依序儲存起來，亦可進行編輯。