The Composer's Algorithmic Assistant:

Based on the Schillinger System of Musical Composition

by
Barry Jones

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

School of Computer Science and Information Systems

Pace University

September 2011

We hereby certify that this dissertation, submitted by XXX XXXX, satisfies the dissertation requirements for the degree of *Doctor of Professional Studies in Computing* and has been approved.

_____-_____

Dr. Fred Grossman Date
Chairperson of Dissertation Committee

_____-_____

Dr. Ronald Frank Date
Dissertation Committee Member

_____-_____

Dr. Charles Tappert Date
Dissertation Committee Member

_____-_____

Dr. Marc L. Smith Date
Dissertation Committee Member

School of Computer Science and Information Systems
Pace University 2011

Abstract

The Composer's Algorithmic Assistant: Based on the Schillinger System of Musical Composition

by
Barry Jones

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

September 2011

This work introduces a novel computer implementation of the elements of music composition that allows for maximum flexibility in creating music. The representation is derived from Joseph Schillinger's "The Schillinger System of Musical Composition", a largely forgotten methodology whose popularity peaked in the 1930's and 1940's.

The Schillinger system is built upon mathematical patterns which are then permuted in various ways to produce an extremely large set of possible musical works encompassing styles that span the range of human cultures across the globe, and the range of human civilization over time. The methods that Schillinger taught at Columbia University and the New School used a mathematical notation and a graphical notation using Cartesian coordinate graph paper. The Composer's Algorithmic Assistant (CAA) makes this method more accessible by letting the computer do the math allowing the composer to use the system while remaining in an emotive mode that is more conducive to being musically creative.

Joseph Schillinger contends that the patterns produced by his system are derived from universal natural patterns. These patterns can be found in music that has withstood the test of time in the sense that it remains powerful and emotive to those who listen to it. Great composers used these patterns even though they may not have been aware of the universal nature of the patterns they employed. This is quite similar to the patterns introduced by Christopher Alexander in the latter half of the twentieth century in the field of architecture. Where Christopher Alexander speaks of "Quality Without A Name" (QWAN), Schillinger speaks of "Universal Patterns" in the same context.

The CAA also uses several novel algorithms for matching a composer's partially completed work to the most appropriate Schillinger patterns so that a composer who is not familiar with the Schillinger techniques can still use the program. The CAA can therefore be used to teach the Schillinger System to music composition students, or simply to show composition students some possibilities that are natural consequences of their works-in-progress.

Acknowledgements

I would like to acknowledge the contributions of a number of people who have helped make this work possible.

I would like to thank my wife Barbara and son Sam for their patience over a number of years while I concentrated on this work. I am deeply indebted to them.

I would also like to thank my advisor and mentor Dr. Fred Grossman, who spent countless hours helping me crystallize my thoughts on the subject of algorithmic composition in general and the Schillinger system in particular.

I owe a debt of thanks to Dr. Marc Smith who has followed my intellectual journey is this research area for a number of years. His careful reading of this manuscript and many fine comments have helped improve the quality of the manuscript.

I also must thank a number of undergraduate research assistants who over the last three years have spent countless hours working with me, coding the ideas presented here and producing the CAA program. Thanks go out to Prairie Rose Goodwin, Anthony Parks, Gregory Katz, and Jesse Stuart.

Thanks also to Delia and Fernando Marx for providing support, advice, and a quiet place to work.

And finally, a special thanks to the Computer Science faculty at Vassar College who have supported my work in so many ways during this process.

Table of Contents

Abstract vi
Table of Figures vii
Chapter 1 Introduction 1
Chapter 2 Prior Work: A Short History of Algorithmic Music Composition 6
2.1 Pre-Computer History of Algorithmic Composition 6
2.2 History of Computers and Music 9
2.2.1 Sound Synthesis 11
2.2.2 Music Performance Analysis 13
2.2.3 Music Information Retrieval 13
2.2.4 Algorithmic Composition 13
Chapter 3 The Schillinger Method of Music Composition 21
3.1 Book I: The Theory of Rhythm 24
3.2 Book II: The Theory of Pitch-Scales 34
3.2.1 Evolution of Pitch Scale Styles 40
3.2.2 Melodic Modulation 47
3.3 Book III: Variations of Music by Means of Geometrical Projections 51
3.3.1 Geometric Inversions 51
3.3.2 Geometric Expansions 54
3.4 Book IV: The Theory of Melody 55
3.4.1 The Axes of Melody 56
3.4.2 Building a Sense of Resistance and Climax in Melody 63
3.4.3 The Superimposition of Pitch and Time 70
3.4.4 Forms of Trajectorial Motion 76
3.4.5 Composition of Melodic Continuity 78
3.4.6 Extending Our Representation 79
3.5 Final Comments on the Schillinger System 83

Chapter 4 The Composer's Algorithmic Assistant 84
4.1 Requirements 84
4.2 Internal Representations 86
4.3 The Rhythm Analyzer 91
4.4 The Rhythm Generator 97
4.5 The Melody Analyzer 98
4.5.1 Finding the Pitch-Axis 99
4.5.2 Finding the Key and Mode 100
4.5.3 Building the Melody Data Structure 103
4.5.4 Populating the Melody Structure 105
4.6 Melody Generator 108
4.6.1 Inverting Trajectories 110
Chapter 5 Future Work 116
5.1 Extending the breadth of the program 116
5.2 Adding the Schillinger Theories of Harmony and Orchestration 116
5.3 Adding a Learning Component 117
5.4 Building a pattern language 117
5.5 Extending the kinds of patterns found by the Schillinger system 119
5.6 Extending the User Interface 120
5.6 Opening the Source Code 120
References 121

Table of Figures

Figure 1- A vertical line represents a point of attack. This is r(3:4). 25
Figure 2 - Music notation representing the same rhythm shown in Fig 1. 25
Figure 3 - Grouping the previous resultant, r(3:2), by the shorter pattern (3/4 time) 26
Figure 4 - Fractioned resultant f(4:3) in both graphical and musical notation 28
Figure 5 - Balanced resultant: f(4:3) + r(4:3) + 4(4-3) 28
Figure 6 - r(5:3:2) and its complement r'(15:10:6) 31
Figure 7 - The complement becomes a counter theme 31
Figure 8 - Constructing a melody from a two note scale. 37
Figure 9 - Modes created by circular permutation of a scale 41
Figure 10 - Transposing the modal intervals down to the original tonic (key note) 42
Figure 11 - Derived scales and keys changes created by transposing the intervals of a mode down to the original tonic or key note. 44
Figure 12 - An original melody and its inversions in pitch and time 52
Figure 13 - Coefficients of expansion and their respective step intervals sizes in frequency 54
Figure 14 - Secondary axes 58
Figure 15 - Polynomial pitch ratios given by the pattern r(5:4) 60
Figure 16 - Parallel correspondence of time and pitch ratios 61
Figure 17 - Contrary correspondence of time and pitch ratios 62
Figure 18 - Circumstantial Oblique Correspondence of time and pitch ratios 63
Figure 19 - Intentional Oblique Correspondence of time and pitch ratios 63
Figure 20 - Repetition as a form of resistance 65
Figure 21 - One Phase Rotation 66
Figure 22 - Incremental One Phase Rotation 66
Figure 23 - Full Periodic Rotation, Constant Amplitude 67
Figure 24 - Two Axis Combinations 68
Figure 25 - Three Axis Combinations 68
Figure 26 - Two Parallel Axis Combinations 69
Figure 27 - Resistance toward climax 69
Figure 28 - Resistance toward balance 69
Figure 29 - Superimposing a fractional group on a factorial group 73
Figure 30 - Ascribed Motion 74
Figure 31- Inscribed Motion 74
Figure 32 - Constraining the pitches to a scale 75
Figure 33 - Trajectory of an actual melody 75
Figure 34 - Upper arcs with constant amplitude 76
Figure 35 - Lower arcs with constant amplitude 76
Figure 36 - Combined motion with constant amplitude 76
Figure 37 - 3(Upper) + Lower + 2(Upper) + 2(Lower) + Upper + 3(Lower) with constant amplitude 77
Figure 38 - The previous example with variable amplitudes 77
Figure 39 - Trajectorial motion in Bach's Two-Part Invention, No. 8 77
Figure 40 - A plotted melody 81

Figure 41 - Schillinger's representation for the continuity (time extension) of the melody of Fig 40 along with its corresponding musical notation. 82
Figure 42 - Overall structure of the Composer's Algorithmic Assistant 86
Figure 43 - A simple melody in music notation 89
Figure 44 – A rhythmic resultant r(4:3) in binary and decimal forms 90
Figure 45 - The Fuzzy-matcher – One pattern slides along another pattern, recording the degree of match at each offset 95
Figure 46 - A portion of a sample hit-list – Each entry contains the degree of matching, the offset, and the pattern index in the database. 97
Figure 47 - A simple four bar melody 104
Figure 48 - The same simple melody in trajectorial format 105
Figure 49 - The Melody Structure for the simple melody 105
Figure 50 - Inverted trajectories for the sample melody 112
Figure 51 - Output of the Composer's Algorithmic Assistant using a phrase from Chopin's Mazurka No. 68. Four possible extensions are shown. 114

Chapter 1

Introduction

This work introduces a novel computer implementation of the elements of music composition that allows for maximum flexibility in creating music. The representation is derived from Joseph Schillinger's "The Schillinger System of Musical Composition,"[35][36] a largely forgotten methodology whose popularity peaked in the 1930's and 1940's.

The Schillinger method has value in that it can be used to create music in many different styles across multiple cultures and across the millennia of human existence. One of its advantages is that it does so without drawing from a large database of previously composed music in a particular style or from a set of composition rules tailored to a particular style, as is the case for many other style based algorithmic composition programs. It has value as a computer based representation because it is mathematically based and music can be easily manipulated using mathematical techniques. We shall see that it has value in terms of musical intelligence because it does not simply recombine bits of music previously composed in a particular style or composed by an individual composer, rather it works at a more abstract level by finding patterns in a musical work and altering the patterns to find extensions to that work that remain within the genre and

style of the work in progress. The Composer's Algorithmic Assistant (CAA) also exhibits intelligence in the sense that it employs a Fuzzy-Matcher to find the closest match between a composer's musical phrase and a Schillinger pattern, to be used to extend the musical phrase. This Fuzzy-Matcher operates in a realm of incomplete knowledge, as does the key and mode finder. A learning component is also proposed in Chapter 5 as future work.

The most important part of this work is in laying out the basic representations and algorithms upon which future work can be built. We have built a proof of concept, a prototype which can be used to create several different kinds of programs. The CAA is a software prototype designed to work with a composer offering suggestions when requested and staying out of the way during times that the composer is creating music. As an assistant it is designed to help a composer, not replace her. The same underlying algorithms with a different interface can be used to create an educational device for teaching composition in general, or for teaching the Schillinger method in particular. One could also build an application that analyzes a composer's style, and may be used to recognize the style of a particular piece and attribute its authorship to a particular composer.

It is important to note that Joseph Schillinger makes some assertions about his methods that have made him and his system controversial during his lifetime and since his death in 1943. One assertion is that his patterns have a universal quality that makes music created with these patterns likely to withstand the test of time. He also asserts that the great

classical composers used these patterns whether they were aware of them or not. We now find a connection between Schillinger patterns and both the lower level patterns and the more abstract design patterns of Christopher Alexander[1][2], an architect who introduced the idea of design patterns and universal quality in the architectural field in the latter half of the twentieth century. Alexander asserts that his patterns, derived from architectures spanning time and culture have a "Quality Without A Name" (QWAN), which maps beautifully to Schillinger's concept of "Universal Quality."

In chapter 2 the reader will see that algorithmic music composition is not a new concept. Composers have been using various composition algorithms for many centuries. In that tradition the CAA is not designed as a replacement for a composer, rather it is an enhancement that serves to spark the composer's creativity and offer some possibilities that may not be obvious to the composer. In this chapter we also investigate a selection of previous work in computer based algorithmic composition programs and show that the CAA is in fact a novel approach that offers the advantages of these systems, bypasses some of their disadvantages, and offers advantages of its own to the user.

To understand the representation and algorithms in the CAA one must first understand the Schillinger system. Chapter 3 covers the parts of the Schillinger system that we have implemented thus far, namely the Theory of Rhythm and the Theory of Melody. It is worthwhile to spend some time and effort reading through this material before moving on to the description of the CAA.

It is worth noting that the Schillinger lecture notes [35][36] are known to be a very dense difficult read. Chapter 3 attempts to make the concepts as clear and understandable as possible. This chapter will be useful to anyone who is interested in studying the Schillinger method. It is a gentle introduction to a difficult subject. Reading the original lecture notes after absorbing Chapter 3 will make the experience much less painful for future researchers who wish to work in this area.

Creating the representations for Schillinger's views of rhythm and melody, building the foundations of the software system, and implementing the theories of rhythm and melody have proven to take a considerable effort. This may in fact be the reason why this is still an open area for research. Chapter 4 presents in detail the data representations and algorithms that constitute the design of the CAA. Some of the novel ideas include an inexact pattern matcher called the Fuzzy-Matcher and an algorithm for finding an expanded set of keys and modes under the condition of incomplete information.

The Fuzzy-Matcher allows a composer to enter a musical phrase which may or may not have been created using Schillinger patterns, and map the incoming phrase to the Schillinger pattern that is the closest match to the input. This allows the program to be used by composers who do not necessarily understand the Schillinger system, but would like to see how their works can be extended through the use of the Schillinger techniques.

We leave the theories of harmony, counterpoint, and orchestration to future work, which will be described more fully in Chapter 5 along with other ideas for the enhancement and extension of this work.

We hope that others will contribute to this work. There is much to be done in implementing more of Schillinger's methods, and in developing a pattern language for the upper level design patterns that Schillinger writes about in his lecture notes.

Chapter 2

Prior Work: A Short History of Algorithmic Music Composition

2.1 Pre-Computer History of Algorithmic Composition

One could say that compositional constraints are the basis of style in music. As we shall see there are composition methods for each style of European music from the fourteenth century to modern times. In general we have seen a relaxing of the rules (constraints) of composition over time. Constraints of almost any kind require algorithmic solutions, whether dictated by composer or by style. Therefore we can say that composers have utilized algorithms for composing music for centuries. Implementing these algorithms as computer programs is not a fundamental change in the way we approach composition. The fact that our algorithms are implemented as a computer program does not affect its validity. What follows is a short history of Western music algorithmic composition methods covering a period between the fourteenth century and the twentieth century.

In the fourteenth century we begin to see isorhythmic compositions – a set of melodic intervals and a rhythmic pattern of different lengths, repeated until they coincide and begin to repeat. Consider a repeated rhythm consisting of 9 attacks and a melody consisting of 8 notes. If we superimpose one upon the other we find the result is a

continually changing melodic phrase which will repeat after 9 x 8 = 72 notes. Philippe de Vitry (1291-1361) and Buillayume de Mauchaut (1300-1377) [8] are two composers who have used this technique. It can be found in the works of later composers as well, for example in the repeated one octave scale in George Gerswhin's "Rhapsody in Blue." It is interesting to note that Schillinger's theories of rhythm and melody are built upon the idea that the superposition of rhythmic patterns and melodic patterns can be used to create an enormous variety of music.

In 1660 Giovanni Andrea Bontempi created his "New Method of Composing Four Voices by means of which one thoroughly ignorant of the art of music can begin to compose [8][18]." His method employs a rota, or wheel, consisting of several concentric tiers populated with numbers representing scale degrees (notes of a scale). Following the rules for traversing the wheel yielded music in four part harmony, with a good sense of voice leading.

The idea of Musikalisches Wurfelspiele [8][26], or musical dice games has been used by a number of well-known classical composers dating back to the seventeenth century. Although the concept is generally associated with Mozart, composers such Haydn, CPE Bach, and Johann Philipp Kirnberger [14] have also used the idea. Musical dice games consist of a number of short musical phrases that are combined and recombined randomly according to dice throws.

We see the idea of recombinancy return in the computer era in David Cope's programs EMI and SARA [11], and in a composition method called Granular Synthesis. Granular Synthesis uses small samples of sound, typically 1 to 50 msec. in duration, to produce soundscapes. The concept was introduced by Dennis Gabor [13], an electrical engineer who is better known for inventing holography. However, granular synthesis was first utilized by Iannis Xenakis [43]. Xenakis also used set theory, stochastic processes, and game theory in his composition methods. Steve Reich has also claimed to use granular synthesis in works such as "Three Tales" and "City Life."

In the eighteenth century Johann Joseph Fux [8] created a composition method for applying contrapuntal techniques to tonal music. We find in Fux the concepts of consonance, dissonance, oblique and contrary motion that have become a standard part of music theory. Griesenger [14] notes, "Haydn took infinite pains to assimilate the theory of Fux [8]." Many other composers such as Beethoven, Cherubini, and Hindemuth [8][16] reference Fux in their letters, writings, or diaries.

Heinrich Schenker [8][30] created an analytic method for describing middle and background motions in his definitions of the Urlinie, or fundamental line of a musical work. "This work of the early twentieth century is a precursor to the idea that one can view a musical work as having larger general patterns which are composed of smaller patterns [8]." Joseph Schillinger uses this idea, and expands it to say that the larger external patterns and the smaller internal patterns should derive from the same basic set of patterns. The concept of a pattern recursively embedded within itself would most

likely be described today as a fractal pattern, and is hinted at in Schillinger's work [35], when he mentions, "The shore-line of North America, for example, may be measured in astronomical, or in topographical, or in microscopic values . . ."

Contemporary twelve-tone or serial composition introduced by Arnold Shoenberg, [37] is a twentieth century technique that follows even stricter algorithms. The concept is to free the listener from having to detect a tonal center by insisting that all twelve notes be used in the melody before reusing any notes. This produces an 'atonal' composition. Mathematical techniques were used to expand these compositions. One would start with a twelve-tone set and apply various manipulations to the set such as inversion and retrograde to produce a full musical piece. We find that Schillinger also uses the idea of inversion and retrograde along with rotations and permutations as a fundamental way of expanding a musical phrase or theme.

2.2 History of Computers and Music

Algorithmic composition has been a part of the computer age since its inception. Augusta Ada King, Countess of Lovelace, considered by many to be the first computer programmer, worked with Charles Babbage on his Analytical Engine, which is considered to be the first computer. She writes in her Notes in 1843 [39],

> "Again, [the Analytical Engine] might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine . .

> . Supposing, for instance, that the fundamental relations of pitched sound in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."

In the modern computer era, we should certainly mention Max Mathews [19], considered the father of computer music. In 1957, while working at Bell Labs, Dr. Mathews wrote the first program for a computer to synthesize and play back music. He continued to be a leader in the field for many decades until his death at the age of 84 in 2011. He collaborated with composers Edgar Varese and John Cage, and in the 1970s worked with Pierre Boulez in founding the Institut de Recherche et Coordination Acoustique/Musique (IRCAM) in Paris, one of the premiere centers for the research and development of computer based music. Dr. Mathews created a series of programs that have enabled many composers to create new and interesting musical works. The programming language Music V was used for creating computer based music and became the foundation for a widely used modern tool for computer music composition called Max/MSP. Many composers working with computers have used the Max/MSP program/environment to define algorithms that have created a wide variety of musical works.

John Chowning, [29] the inventor of the FM music synthesis technique used in Yamaha synthesizers, founded the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford. Max Mathews left Bell Labs and joined CCRMA in 1985, continuing his research there as a professor of music. CCRMA is another premiere center for research in computer music.

A third research center, the MIT Media Center, has been a locus of creativity in the computer music field since its inception in 1985. The Music, Mind and Machine project headed by Barry L. Vercoe, [29] deals with the construction of intelligent systems out of interacting audio-processing agents.

Although IRCAM, CCRMA, and the MIT Media Center are major research centers for computer music there are certainly many more organizations devoted to the study of computers and music.

We can classify current computer music research into four general areas which include sound synthesis, music performance, music information retrieval, and algorithmic composition. While this work concentrates on algorithmic composition, the next four sections aim to offer the reader an overview of each of these research areas to provide a context for the work done here.

2.2.1 Sound Synthesis

This research area involves the synthesis and production of sounds.

One of the earliest methods of electronic sound synthesis is Additive Synthesis [28] used by the Hammond Organ, where a fundamental frequency and a selection of harmonics or overtones are added by pulling out drawbars to determine which harmonics, and at what amplitudes the various harmonics should be combined.

The era of Additive Synthesis was succeeded by the era of Subtractive Synthesis [28], where one specifies a timbre by specifying the use of one or more waveforms, including square, triangle, sawtooth, and sine waves, each of which consists of the fundamental frequency and a set of harmonics. Each of these waveforms can then be fed through filters to alter the relative amplitudes of each of the waveforms' harmonics.

In 1967 John Chowning [29] discovered the concept of FM Synthesis, which takes the ideas of FM radio transmission and applies them to waveforms in the audio frequency range. This offered a new and very rich set of tones to work with. An alternative to FM synthesis was Phase Modulation (PM) Synthesis which distorted the underlying waveforms by stretching and compressing sections of the waveform to produce different sets of harmonics. Both FM and PM synthesis techniques offered a richer set of sounds than the previous generation of additive and subtractive synthesis.

In the 1970s we saw the emergence of Sampled Sound Synthesis which uses digital samples of acoustic instruments and other sounds to create pitches. Ray Kurzweil [6] created the ultimate Sampled Sound Synthesizer of the day by using high quality samples for each note of each timbre. This was a stretch given the current level of technology available. It made for a very expensive, but wonderful keyboard synthesizer.

With the rise of computer power and availability in the 80s we see a wide variety of sound production techniques, but each can be traced back to its roots in one of the earlier methods.

2.2.2 Music Performance Analysis

Research into computer aided music performance analysis involves the understanding and reproduction of performance parameters [15]. We note that the same piece of music can sound quite different when performed by different artists. This research area aims to capture those parameters, understand how these parameters affect the quality of a performance. The information is used to improve the performance of computer based music and to analyze and hopefully improve the quality of human performers. Also in this area are accompaniment programs that follow a soloist and either match the written accompaniment to the soloist, or create a jazz style accompaniment in an ad-hoc manner.

2.2.3 Music Information Retrieval

Research in the area of Music Information Retrieval [5] addresses issues involved in classifying audio files, finding an audio file based on similarities to other music files, finding an audio file by using a melody hummed in by a user, etc. Music Recommender Systems can be classified in this area as well and find use in many of today's Internet based music repositories.

2.2.4 Algorithmic Composition

This has been a rich area of research, and is the area in which this author is working. There are a number of algorithmic composition programs using a variety of methods. We can find composition programs that generate semi-random music using stochastic

methods, genetic algorithms, neural nets, Markov chains, fractal and chaotic algorithms, and various mathematic, connectionist, or pattern-matching algorithms. A sampling of these programs is presented below.

Common Music [40][41] is an open source project created at the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University. Common Music is a multi-platform, object-oriented programmer's environment for creating algorithmic music. It contains commands for applying inversion, retrograde, and rotation to a melody. It offers the programmer the ability to create an nth degree Markov chain, apply randomization, and create cellular automata. It converts MIDI to an internal string representation and vice versa. It can also handle sound in the frequency domain, varying the frequency spectrum of a note. Common Music is a good all-purpose environment for exploring a wide variety of algorithmic techniques.

Max/MSP, originally written by Miller Puckette [24][25], is a visual environment for creating computer based music and video. There is a programmer's interface to Max and a drag-and-drop visual interface which allows the user to add various modules to the workspace and interconnect them creating an executable workflow diagram. Max is a descendent of Max Mathew's MUSIC V programming language. MSP is a later addition that handles signal processing for handling digital audio files. Originally developed at IRCAM, it is now maintained and sold by Cycling '74 [44]. There is a free version available under the name of Pure Data [25]. A version of Max/MSP called Jitter [44] is available that extends its capabilities to the video realm. Max/MSP is the most widely

used computer music environment and offers an easy way to manipulate music files, or create algorithmic music compositions.

Symbolic Composer [42] is a commercial product for computer based music composition. It uses two different grammars to define sections and instruments. Sections refer to formal structures of a work that consist of any number of instruments. Instruments define timbres. Each instrument inherits musical properties called classes. The main classes include symbols, lengths, tonalities, zones, and velocities.

Symbolic Composer also includes modules with functions such as neurons, tonalities, processors, and generators. Processors include filters, shifters, transposers, mixers, and morphing tools. Generators generate fractal and chaos-based music, programmable neurons, Fibonacci series, autocatalysis, Fourier analysis/synthesis, and recursive structures.

Each of these programs can create new material, but the likelihood of the new material agreeing in style with the style of a composer's previous work is very small.

2.2.4.1 Style Creation and Style Imitation Programs:

A number of programs have been created to allow the user to create music within a particular style, or imitate a particular style.