Hindemith and Algorithmic Harmonic Generation

Ken Paoli, DMA
College of DuPage
Glen Ellyn, IL USA
paolik@cod.edu

ABSTRACT

Hindemith proposed a chord analysis system based on his theoretical observations in the Craft of Musical Composition. His concept of chord classification based on the presence of a tritone in a vertical structure allowed him to develop a method for measuring the tension in a progression of chords called Harmonic Fluctuation. These notions are the basis of a harmonic generating algorithm that allows a user to choose the chords in a progression and plan the harmonic tension of the progression.

1. INTRODUCTION

Paul Hindemith (1895-1963) was a German composer, theorist, teacher and performer who migrated to the United States in 1940 where he taught at Yale University. He is usually associated with “Gebrauchsmusik” (utilitarian music) and Neo-classicism. However, like many German composers of the day, his output contains both Romantic and Expressionistic works. In the late 1930’s Hindemith wrote *Unterweisung im Tonsatz* (*The Craft of Musical Composition*) which was published in 1937. While the title might lead the reader to expect a composition “textbook,” Hindemith instead provides a detailed theoretical system which uses natural phenomenon (the overtone series and combination tones) to explain consonance dissonance and vertical structures. Like other theorists at this time (Persichetti and Hanson, notably), Hindemith is grappling with a system to explain the “extended tonal” techniques that were introduced in the late 19th- early 20th c. which ranged beyond tertian construction and the Roman numeral system of taxonomy. This system had been employed as a pedagogical tool since the early 19th-c. but was insufficient for non-tertian structures. Hindemith’s system attempts to explain common practice harmonic structures, as well as non-tertian structures. This desire for “backward compatibility” can be observed in the final chapter of the book where he provides analysis for works from Machaut to Hindemith via Bach, Wagner, Stravinsky and Schoenberg.

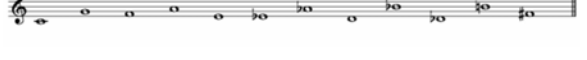
2. HINDEMITH’S THEORIES

2.1 Series One and Two

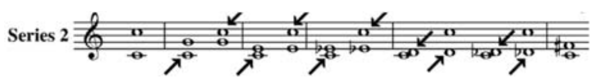
Using the overtone series and combination tones as the basis of his observations would qualify Hindemith as a natural theorist (as well as his discussion on the origin of major). His harmonic system relies on the evaluation of intervals and their tension as it relates to consonance and dissonance. Herein lies some of the criticism of Hindemith’s system. In using the overtone series, it has been argued that this system is applicable to just intonation since equal temperament does not use the natural ratios of partials and the lack of pure thirds clouds major and minor triads. A second criticism is that Hindemith makes no specific point of transition between consonance and dissonance [6].

Hindemith’s Chord Groups are based on two “series” derived from his examination of the overtone series and combination tones. Series 1, the “series of tones in relation to a progenitor tone from which they derive their tonal position” and Series 2, the twelve intervals based on “combination tones of the first and second order,” moving from consonance to dissonance to ambiguity (the tritone) [5].

Interestingly, Orlando Legname submitted Series 2 to a mathematical examination of the interval wave forms terming the results as “Density Degree.” Using this method to generate an interval series he places the tritone between the minor third and the minor sixth, instead of at the end of the series. He conjectures that Hindemith’s placement of the tritone at the end was because of the “harmonic meaning the tritone has held in Western music.” It may also be conjectured that Hindemith understood the ambiguous nature of the tritone as the only interval that inverts to itself and its resulting effect on structures that contain it. The mirror intervallic structure of the tritone makes it the dyadic equivalent of Chord Groups V and VI. The importance of the tritone’s effect on chordal structures is reinforced by Hindemith’s dividing of chords into two broad groups: those with tritones and those without. It should also be mentioned that the first edition of the “Craft” did not include the tritone in the Series One list.



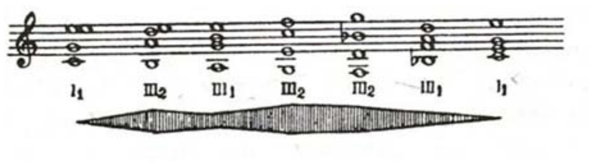
Example 1. Series One: Tonal Position from Tonic.



Example 2. Series Two intervals: consonant to ambiguous with intervallic "root" indicated.

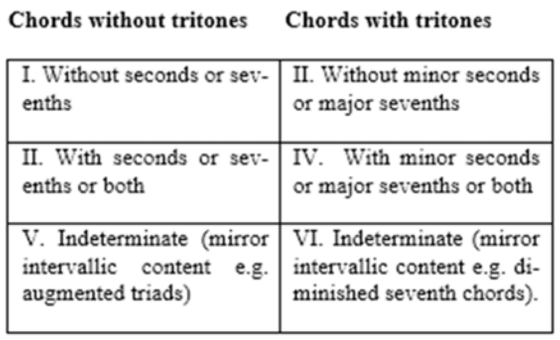
2.2 Hindemith Chord Groups

Hindemith proposed a method of chord analysis based on his system of chord classification. Like traditional theory and Schenkerian analysis, one aspect of this analytical system is the Two-Voice Framework: the bass and the most significant upper part (based on Series 2). The second aspect is the ability to chart the change of tension in a harmonic progression called Harmonic Fluctuation.



Example 2. Harmonic Fluctuation

As stated above, Hindemith presented the chord groups first making a division between chords that contain tritones and those that do not. He then establishes chord groups beginning with consonant chords, moving through dissonant chords to ambiguous structures referred to as "indeterminate." These indeterminate chords are rootless and contain equal-interval chord constructs (augmented triads, diminished sevenths and quartel chords). For the sake of organization Hindemith utilizes an outline format in the chord table which, unfortunately, makes use of Roman numerals. This can create confusion with the more common use of Roman numerals for traditional chord analysis. This possible confusion aside, what Hindemith provides is a system of chord classification and just as importantly a method to gauge the harmonic fluctuation of a chord progression. It was an appreciation of Hindemith's attempt to provide a taxonomy for the evolving language of the early 20th-c that made his system appealing as the basis for a harmonic-generating algorithm. The Chord Group table is included at the end of Hindemith's book (which has become a quick guide to Hindemith's rather detailed theoretical observations) [5]. An abbreviated version is included here.



Chords without tritones	Chords with tritones
I. Without seconds or sevenths	II. Without minor seconds or major sevenths
II. With seconds or sevenths or both	IV. With minor seconds or major sevenths or both
V. Indeterminate (mirror intervallic content e.g. augmented triads)	VI. Indeterminate (mirror intervallic content e.g. diminished seventh chords).

Table 1. Chord Classification Chart

While providing an appealing method to evaluate harmonic structures, the use of Hindemith's Chord Groups as the basis for a harmonic generating algorithm was not intended to recreate all aspects of his system. The notions of root as the best note of the best interval is not maintained, nor is the concept of the Guide Tone as the tritone member standing in best relation to the root of the chord. While future refining of the algorithm may include these aspects, the initial goal was to be able to create vertical sonorities and plot their progression using Hindemith's notions on Harmonic Fluctuation. At the same time the desire to include randomization to various parameters reflect the author's inclination towards stochasticism. In this iteration "stochastics" were deemed more important than "scholastics."

3. ALGORITHM

3.1 Xenakis and Sieves

The generation of material in many programs in computer assisted algorithmic composition (CAAC) is often linear. A method to generate a single-line may become the basis for the generation of several melodic lines producing a polyphonic texture. Obviously, generating chordal structures requires that several voices be output simultaneously. The initial notion for the organization of this algorithm was to incorporate logical sieves as proposed by Iannis Xenakis using moduli 3 and 4.

Sever Tipei has stated Sieves are…"logical filters expressed as Boolean operations on congruence modulo classes denoted by various indices of a single modulo [9]". Xenakis revisited Sieve Theory in several published documents between 1957 and 1990.

In *Formalized Music* (1971) sieves are discussed in Chapter VIII and in the 1990 edition Chapter XI is entitled "Sieves" and Chapter XII is a user guide for sieves. In his early work the calculations are completed "by hand" on graph paper and the sieves make use of union, intersection and complementation. In the first edition of *Formalized Music* the routines are in Fortran.

In the revised edition of *Formalized Music* Xenakis provides a number of changes [10]:

- Complementation is eliminated without comment.
- Two main procedures are described:
    - The generation of a sieve from a logical formula.
    - The generation of a logical formula from a sequence.
- The computer routines are in C
    - There are errors in the programs (Ariza)

Christopher Ariza has included corrected sieve routines (for pitch, rhythm and parameter generation) in his computer program athenaCL, which is written in Python [1].

In his 2007 dissertation entitled "Xenakis and Sieve Theory", Dimitri Exarchos presents a matrix for moduli 4 and

3. Since complementation was removed from sieve computation Exarchos rightly surmises that accessing pitch material within an octave is easily accomplished with a matrix of mod 3 and mod 4 values [3]. This matrix contains all the residual classes of mod 12. This seemed an efficient way to access the necessary pitch material and replaced sieves as the programming approach. This matrix is displayed in Table 2.

	(4,0)	(4,1)	(4,2)	(4,3)
(3,0) ⟶	0	9	6	3
(3,1) ⟶	4	1	10	7
(3,2) ⟶	8	5	2	11

Table 2. Mod 3/Mod 4 Matrix.

The intention was to utilize moduli 3 and 4 to locate chord pitches in the matrix. This would require the use of two-dimensional arrays and while many programming environments do not facilitate these types of arrays it can still be accomplished. A series of one-dimensional arrays can be used instead since a two-dimensional array is primarily an array of pointers; each pointer can refer to a one-dimensional array. The three arrays of four elements (mod 4) were to be addressed by three arrays (mod 3) using as many arguments as needed to retrieve the desired pitches. To generate a quartel chord the mod 3 elements would point to the mod 4 "pitches." Those pitches would be found at the following addresses:

(3,0,) (4,0) = 0 (C)   (3,1) (4,2) = 10 (Bb)   (3,2) (4,1) = 5 (F)

To reduce the number of arrays needed to supply the matrix "pitches" a single array (found at (3,0)) can be used to generate the remaining arrays by merely adding 4 and 8 to the original array values.

		(4,0)	(4,1)	(4,2)	(4,3)
(3,0)		0	9	6	3
(3,1)	+ 4 =	4	1	10	7
(3,2)	+ 8 =	8	5	2	11

Table 3. Matrix with addition.

3.2 Chord Structure Data

One problem became immediately apparent: many chordal structures require two pitches from the same mod 3 row. For instance, a major triad is found at (3,0) (4,0), (3,1) (4,0) and (3,1) (4,3). This would require two pitches from the (3,1) row. The solution is to add additional arrays so at least two pitches from the same mod 3 row can be addressed simultaneously. This led to a six-voice approach and provided a partial solution but still left problems with the generation of major, minor and diminished chords which will be detailed later.

A six-voice chordal texture would require six directly addressable arrays containing the values in the top row of the matrix: (0, 9, 6, 3). A single array was used to send this information to the remaining five arrays. Likewise, an additional six directly addressable arrays were loaded with the mod 3 column information: (0,1,2,3). The mod 3 arrays would "point" to the values in the mod 4 columns.

To access the values of the second mod 3 row the value of 4 was added to the array output and to access values in the third row the value of 8 was added to the array output.

In order to send the correct matrix address, fourteen "Chord Group" arrays were populated with structures from the Hindemith Chord Chart. For example, to address a quartel chord the pitch array would contain (0, 2, 1, 0, 2,1) which would be sent to populate the mod 3 arrays with the correct "pointer" addresses which, would in turn, pass that information to the mod 4 arrays.

Seven arrays were populated to reflect the following structures:

Clusters array numbers (0,0,1,3,1,2)

Whole Tone array numbers (0,0,0,2,2,2)

Augmented triads (Hindemith Chord Group V) array numbers (0,0,0,0,0,0)

Quartel chord (Hindemith Chord Group V) array numbers (0,2,1,0,2,1)

With additional programming:

Major triads: (Hindemith Chord Group I A) array numbers (0,0,0,0,3,3)

Minor triads: (Hindemith Chord Group 1 A) array numbers (0,3,0,3,3,0)

Diminished 7ths (Hindemith Chord Group VI) array numbers (0,0,0,2,2,2)

Seven additional arrays were populated with data yielding the following Hindemith Chord Group structures:

IIb1 (Two arrays) array numbers (0,2,2,2,2,2) (0,3,1,3,3,2)

III 1 (One arrays) array numbers (0,3,1,1,3,2)

III 2 (One array) array numbers (0,1,0,3,2,1)

IV 1 (One array) array numbers (0,3,0,0,1,2)

IV 2 (Two arrays) array numbers (0,0,3,2,0,3) (0,3,2,2,3,2)

A second method of populating the Chord Group arrays was also investigated. It was decided to use the interval distance from one chord member to the next (expressed numerically) to generate the six-voice chords. A major chord would yield 4 half steps, 3 half steps superimposed on the first value and 5 half steps superimposed on the second value. The array values, counting from zero, would be

(3,2,4). This required a null array with the value of zero acting as the first pitch of every structure. Using this approach makes "C" the root of every structure. This is a common approach that is used in commercially available algorithmic programs like Band-In-The-Box.

The fourteen directly addressable read-only arrays were populated with numerical data in this fashion representing the chord groups listed above. This eliminated the problem of two "pitches" on the same mod 3 row, but the level of data manipulation necessary after the arrays were filled was much greater. The single array approach was chosen as the more programmatically elegant solution.

3.3 Program Detail

This algorithm was designed in Music Wonk, a programming environment that is MIDI based, object oriented for Win-tel computers. The program may be ported via a virtual MIDI-patch "cable" to any virtual instrument or DAW program.

3.4 Data Generation

The fourteen populated Chord Group Arrays were input to a switch that outputs the array values to the six mod 3 arrays. These "pointers" then output to the six mod 4 arrays. Voices one and four supply the mod 3,0 value. With the addition of the constant 4, voices two and five supply the mod 3,1 values and with the addition of the constant value 8, voices three and six provide the mod 3,2 values. The value of each voice is increased by some multiple of 12 to raise it into the desired MIDI note range. These values are output to six MIDI Voice Modules which act as virtual synthesizers.

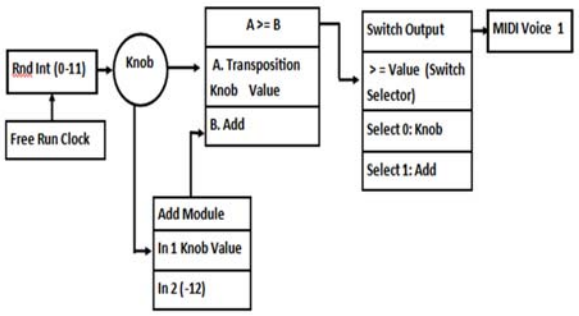
Three methods for the user to monitor the array output are provided. On the control panel each voice has a Mod 3 Knob which indicates the present value and there is a MIDI note readout from the MIDI Modules. In the program a Multi-tap Array uses the output of the Array Switcher to show the six values of the mod 3 arrays for quick data verification.

The pitch material needs to be musically spaced to reflect "open chord" voicing. This was accomplished with a set of user-controlled knobs that allows for octave changes in a range of three octaves. This control is available for all arrays except the Voice One array which is meant to supply the chord root and has a fixed value (This value can be changed at the program level if necessary.). Once the desired octave selections have been made, the generated data is sent as a MIDI note number to the MIDI Voice Modules.

3.5 Transposition

Progression implies changes of harmony and in order to facilitate root movement a transposition control was included. A random integer was input to a Transposition Knob which was triggered by a random clock generator. Transposition was set in a range of + or – 6 half steps. While transposition supplied the desired harmonic movement, the six-voice chords generated in this fashion would move only in parallel motion. This was severely limiting and would reduce this algorithm to generating examples of chordal planning or rock "power chords."

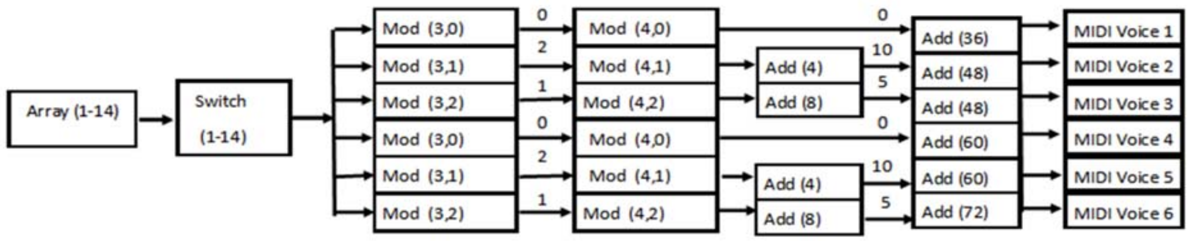
A method was needed that would change the bass direction a portion of the time. Lowering the generated data by an octave would reduce the amount of parallel motion. The output of the Transposition Pot was inputted to a "greater than or equal to" logic module (GrEq) with a with a comparison value of 1. Chord roots with a transposition number below 1 (0 to -6) would be reduced by an octave. The output of the GrEq module was routed to the Select input of a switch to choose between the value of the Transposition Knob or a constant with the value of -12. The Switch output is input to the transposition port of the MIDI Voice module for the Voice 1. The result is the lowering of the root voice by an octave and contrary motion to the upper voices. The Transposition Knob output is sent to the five remaining Midi Voice Modules with no data manipulation.



Example 4. Bass Movement flow chart

3.6 Harmonic Progression

In order for the user to control the Harmonic Fluctuation of the generated material a Number Matrix was utilized. A 14 X 3 grid was implemented yielding 42 chord selection slots. This graphic interface served two purposes. It allowed user input for the selection of chords from the Chord Group Arrays. This facilitated the planning of a harmonic progression (and thereby, the harmonic fluctuation) using

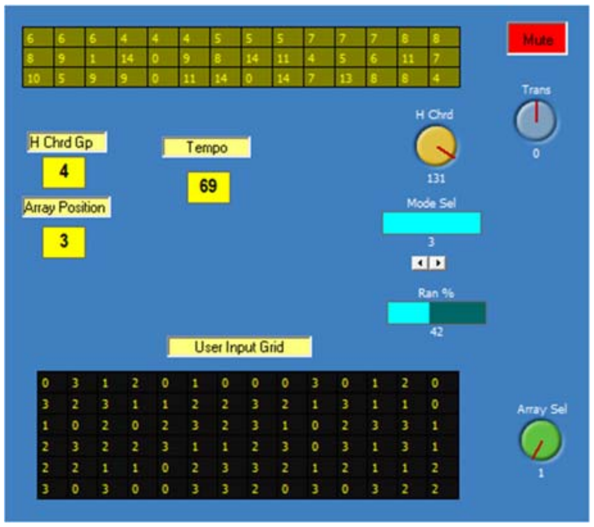


Example 3. Data generating flow chart (quartel harmony).

the preloaded Chord Groups. Secondarily, the Number Matrix allowed for the tracking of the generated chords. To make this easier for a user to follow, two readouts were included: one for the Chord Group Array and a second for the current position in the Number Grid. The output from the Number Grid selects the desired array from the switch containing the Chord Group arrays.

While the Chord Groups Arrays are "preloaded" with the array values stated earlier, a series of number grids are included that allows a user to change the content of any of the fourteen Chord Group Arrays and, of course, the resultant harmonic output. With this capability a user can explore other Hindemith chord structures or any other vertical material since the preloaded arrays are not meant to be exhaustive, but rather, representative. Fourteen number grids are included and a knob provides input to a switch that allows the user to choose between the Number Matrix or the User Grids.

At this point the generation of chords was limited to the ordering of either the Number Matrix or the User Grids. It was decided that additional modes of operation should be made available. A linear slide pot designated as "Hindemith Chord Group Selection" was included to allow for three modes of operation. A user can select between a random selection of chords for the arrays, a single array type (yielding one harmonic entity that is transposed) or the progression of chords through either the Number Matrix or User Grid. The random option uses a random integer generator triggered by a random clock. The single chord mode can be incremented/decremented to the desired Chord Group Array. This output of the Group Select slide pot is input to a Switch that selects between the three modes of operation. This portion of the Control Panel is Example 5.



6	6	6	4	4	4	5	5	5	7	7	7	8	8
8	9	1	14	0	9	8	14	11	4	5	6	11	7
10	5	9	9	0	11	14	0	14	7	13	8	8	4

H Chrd Gp 4

Array Position 3

Tempo 69

H Chrd 131

Trans 0

Mode Sel 3

Ran % 42

Array Sel 1

User Input Grid

0	3	1	2	0	1	0	0	0	3	0	1	2	0
3	2	3	1	1	2	2	3	2	1	3	1	1	0
1	0	2	0	2	3	2	3	1	0	2	3	3	1
2	3	2	2	3	1	1	2	3	0	3	1	3	1
2	2	1	1	0	2	3	3	2	1	2	1	1	2
3	0	3	0	0	3	3	2	0	3	0	3	2	2

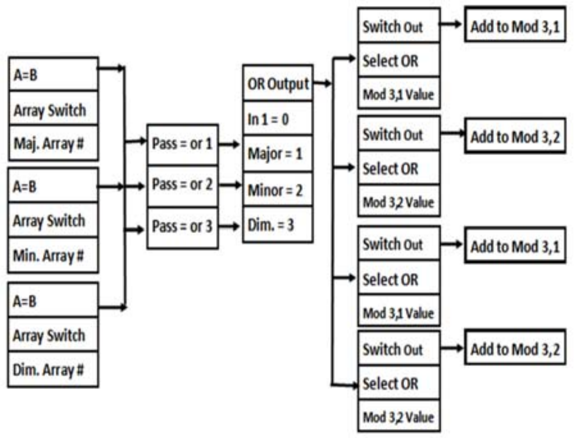
Example 5. Chord Group Grid and User Number Grid

3.6 Major Minor and Diminished

Three of the most common structures, major and minor triads and diminished sevenths, created a problem because the position of chord members in the matrix left one or two mod 3 rows unused. The major and minor triads had no information in the mod (3,2) rows. The diminished seventh had all of its pitch material generating from the mod (3,0) row of the matrix. This resulted in the playing of "wrong notes" from either or both of the mod (3,1) and (3,2) rows. The solution was to "undo" the addition of the constant values 4 and 8 that were added to the mod (3,1) and (3,2) rows.

This was accomplished by singling out the three arrays in question and reducing the values sent to the Add constants associated with the mod 4 array. Each of the three arrays were input to an "Equal to" logic module that would pass a true value (1) if the array matches the value output by the Chord Array switch. The major array passes a value of 1, the minor array a value of 2 and the diminished array a value of 3. An "Or" module passes these values to a series of switches with three outputs that allow the correct subtraction values to be sent to the mod 4 arrays thereby adjusting the "pitch" to the required chordal component. This adjustment only applies to the mod (3,1) and (3,2) rows since no addition process is applied to the mod (3,0) row.

A passing observation: logical operations in graphically based, high-level programming environments are less direct than their line-level counterparts. The results may be the same but the journey is more circuitous.



Example 6. Major, minor and diminished correction chart.

4. SUBROUTINES

The initial iteration of this algorithm yielded six-voice harmony with no rhythmic, textural or dynamic change. To increase the usefulness of the program output it was decided to include subroutines for rhythmic value selection, velocity change using range-based zones and the ability to vary the number of voices in the chord to create textural variety.

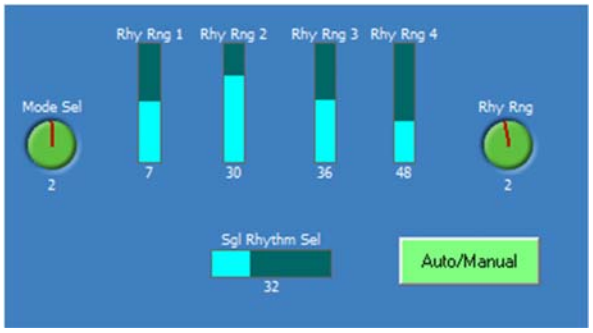
4.1 Rhythmic Subroutine

The Rhythm Subroutine allows for three modes of operation. Rhythm may be randomly selected in the MIDI range of 2-96 ppqn (pulses per quarter note), or it may be set to a single rhythmic value based on from 2-192 ppqn (24 ppqn= a quarter note). A third mode allows for four overlapping rhythmic ranges. Each available mode is user selectable.

The Rhythmic Range mode uses a knob to select between four rhythmic ranges. Range 1 has a range of 2 pulses (thirty-second note) to 12 pulses (eighth note). Range 2 has a boundary from 12 to 24 pulses (eighth note to quarter note); Range 3 has values from 24 to 48 pulses (quarter note to half note); Range 4 includes a range of 36 to 72 pulses (a dotted quarter to a dotted half note).

A Slide Pot lets a user select the range value which is output to a Switch that chooses between the Slide Pot value and a Random Integer Generator that outputs a value between 2 and 72. A "Free Run" Clock Module provides a strobe for the Random Integer Generator. A second Free Run clock provides the strobe for the four Rhythmic Range Switches. An Auto Rhythm Button is used to send the desired data out of the switch. When the Switch is on Auto-Rhythm Mode is selected; Manual Mode is selected when the switch is off.

The output of the four Rhythmic Range Switches are input to a second Switch which is also receiving a Random Integer in a range from 0-3. A Random Clock triggers this Random Integer generator. The Rhythmic Range output is determined by which of the four ranges are selected by the Random Integer. Since the desired result is a chordal texture, all six voices share the same rhythm. By supplying the user with the three methods of selecting rhythm, a wide variety of rhythmic values may be utilized. Example 7 is shown below.



Example 7. Rhythm Control Panel.

4.2 Velocity Subroutine

In some circumstances it is desirable that the individual voices of a chord have the same velocity level. When using this algorithm with samples of orchestral sections, equal velocity provides a more homogeneous sound where individual notes do not "pop out" of the chordal texture. On the other hand, when using a single instrument for each chord voice, variation in velocity produces a more musical, less machined result.

It was decided to provide both approaches to velocity in this algorithm. Constant velocity is available with each MIDI Voice module receiving the same velocity. A rotary knob provides the option to set the velocity value anywhere in the MIDI range (0-127). A second option, provided by an On/Off Switch allows for the different velocities for each voice. This is accomplished with a Random Integer Generator that has six outputs. A random integer is generated in a range from 45-110, which is output to a bank of six Switches that, in turn, output velocity values to the MIDI Voice modules.

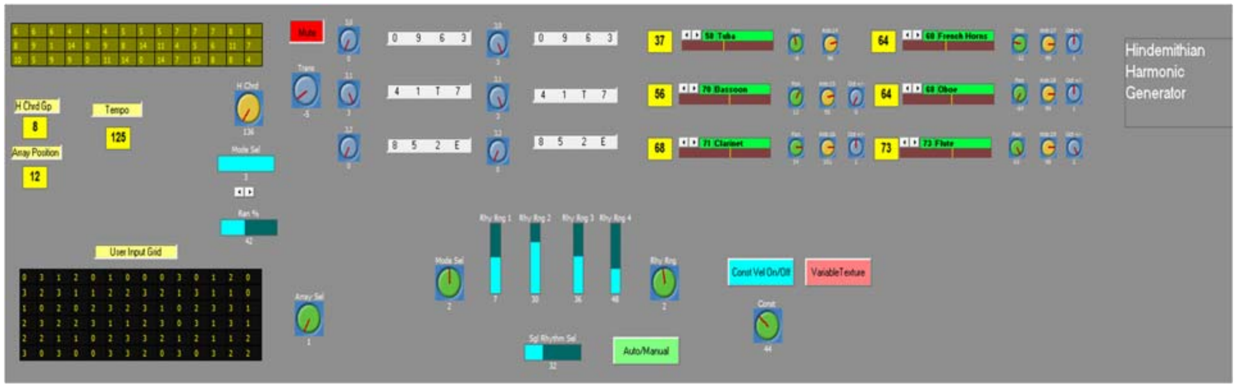
4.3 Texture Subroutine

The same approach was applied to the textural component of the algorithm. While it is possible to generate a constant six-voice texture, the musical reality is that this textual invariance becomes tiresome rather quickly. A method to create textural variance was devised using a Switch Module.

A Random Integer Generator outputs a value of 0 or 1. This output is sent to a Distribute Module. The Select input of the Distributor is sent another Random Integer in the range of 0-5. The Distribute Module has six outputs and they are sent to a bank of Switches that output 0 or 1 to the Play input of the MIDI Voice Module. The result is a variable texture ranging from one to six voices.

4.4 Additional Controls

Panning and Volume Controls are included in this algorithm so that the MIDI information concerning these two parameters are transmitted from the host program. When initially listening to generated material it is much clearer when panning is applied to the voices. Volume controls provides for the matching of volume levels for the individual and disparate levels of various sample instruments. Individual controls for panning position and MIDI volume are included on the Control Panel and are user controlled.



Example 8. Control Panel.

5. CONCLUSIONS

The algorithm presented here successfully generates harmonies of both tertian and non-tertian construction. The addition of user selectable controls allows for the planning of a harmonic progression and the inputting of any harmonic desired harmonic structure. The addition of subroutines for rhythmic generation, velocity and textural variance add to the musicality of the program output. The inclusion of randomization gives the user additional controls in the areas of progression (array choice), rhythmic values, velocity and texture.

Most importantly, using Hindemith's system of chord evaluation to plan the harmonic fluctuation of a progression allows a composer to preset and then easily evaluate the resultant output. The outputting of MIDI data, which can be routed to any DAW program, makes both recording and editing extremely easy. While the instrumental choices inside the program are General MIDI instruments, once the output is sent to a DAW, any available virtual instrument may be used.

While aspects of Hindemith's theories have been criticized, harshly at times, and may eventually be relegated to a footnote by theorists, his notions on harmonic structures are valid and accepted by a number of his detractors [7]. The importance of the tritone as an element of ambiguity and motion in common-practice tonality are universally accepted. The same can be said of the tritone when using "extended tonal" harmonic structures involving non-tertian construction. Hindemith provides a usable taxonomy that is inclusive of vertical structures regardless of their era of origin. This algorithm allows a composer to utilize Hindemith's chordal system to easily generate harmonic material with control over the harmonic fluctuation of a progression.

6. REFERENCES

[1] Ariza, Christopher. "An Object-Oriented Model of the Xenakis Sieve for Algorithmic Pitch, Rhythm and Parameter Generation." *Proceedings of the ICMC*. Miami, 2004.

[2] —. "The Xenakis Sieve as Object: A New Model and a Complete Implementation." *Computer Music Journal, 29:2* (2005): pp. 40-60.

[3] Exarchos, Dimitrios, Iannis Xenakis and Sieve Theory: An Analysis of the Late Music (1984-1993). 2007.Goldsmith College. PhD dissertation.

[4] — and Jones Daniel. "Sieve analysis and construction: Theory and implementation." *Proceedings of the Xenakis International Symposium*. London, 2011. 1-3.

[5] Hindemith, Paul. *Craft of Musical Composition*. Trans. Arthur Mendel. New York: Schott Music Corp., 1945.

[6.] Legname, Orlando." Degree Density of Intervals and Chords."*20th-Century Music 4.10 (1997):pp.8-14*

[7] Luttmann, Stephen. *Paul Hindemith: A Research and Information Guide*. 2nd. New York: Routledge, 2009

[8] Sorenson, Andrew C. and Brown, Andrew R. "A Computational Model for the Generation of Orchestral Music in the Germanic Tradition: A progress report." *Proceedings Sound :Space - The Australasian Computer Music Conference*. Sydney, 2008. pp.78-84.

[9] Tipei, Sever. "Composing with Sieves: Structure and Indeterminancy in Time." *Proceedings of the ICMC*. Perth, 2013.

[10] Xenakis, Iannis. *Formalized Music*. New York: Pendragon Press, 1992.