

Harmonic & Structural Design in Algorithmic & Electro-acoustic Composition

by Vincent Joseph Manzo
September 1, 2007

Table of Contents

Preface (pg. 3)

Early Algorithmic Compositions (pg. 4)

- Invention 16 (pg. 6)
 - Design
- Modal Change (pg. 8)
 - Design

Progressing

- Evolutio (pg. 14)
 - Software design
 - Harmonic design
- Ravelian Fuse (pg. 23)
 - Software design
 - Harmonic design

Appendix (pg. 28)

Preface

Algorithmic composition is not a new trend. For hundreds of years, composers have been using algorithmic devices to compose music. In this paper, I describe how I use a computer and my own algorithms to compose music.

Algorithmic compositions with a high level of sophistication (in the vein of David Cope or Robert Rowe) is more than just a computer arbitrarily spitting back notes when it is told to do so. It is more than a computer program creating musical gestures based solely on a composer's preferences. An algorithmic composer must account for musical syntax.

In my foundational research, I began working toward the creation of several objects for Cycling 74's Max software, a powerful graphical dataflow programming environment that would aid in my development of musical grammars. The result of my research was the Modal Object Library.

The Modal Object Library is a collection of algorithms to control modality. The heart of the Modal Object Library is the *Modal Change* object which interprets all of the scale data for major, harmonic minor, harmonic major and melodic minor and each of the 7 modes for each of these categories (28 modes all together). It outputs all of the pitch classes in order and stores the data in a table as well. It also puts the data into a coll list file which can easily filter incoming notes that are not from the user selected mode to those from the mode. For instance, if you play a D in Db Major, the list file will bump the note down to Db.

Additional objects in the library are *Modal Shift* (finds modes related to the one you select sharing 6 of the 7 notes), *Modal Mutation* (same as *Modal Shift* but excludes

the list of related modes to those sharing a common tonic), the Messiaen Objects (utilizes the Messiaen Modes of Limited Transposition), *Modal Chord* and *Modal Triad* (for chords and triads), *Modal Prog*, for creating 4, 8, 12 or 16 bar progressions based on each mode.

The Modal change object and the Messiaen objects all have the ability for you to input your own scale degree distance map (in steps. For instance major scale is 2 2 1 2 2 2 1). If the user entered mode is unidentified it will say that it doesn't have a name for it, but will still output all pitches from the scale in order and input the data into a table.

The library also includes networking objects optimized for controlling objects in this library over a network, and an analysis object that takes all notes played within a specified time frame and identifies the mode.

More information on this library (including download links) can be found at vincemanzo.com. In-depth documentation about this library can be read on the same site in my 2007 paper entitled “Implementing Modality in Algorithmic Composition”.

The objects in this library as well as several objects found in Max will be referenced throughout this paper. Several of my objects also list Peter Elsea'sⁱ *banger* object, part of the RTC Library.

In *Interactive Music Systems*, Robert Rowe (1993) defines three methods of algorithmic composition:

Generative Methods use sets of rules to produce musical output from the stored fundamental material.

Sequenced Techniques use prerecorded music fragments in response to some real-time input. Some aspects of these fragments may be varied in performance, such as tempo playback, dynamic shape, slight rhythmic variations, etc.

Transformative Methods take some existing musical material and apply transformation to it to produce variants. According to the technique, these variants may or may not be recognizably related to the original. For transformative algorithms, the source material is complete musical input.

Early Algorithmic Compositions

I have been composing music since I began playing music at age 13. During my undergraduate years at Kean University, my compositional interests shifted from popular music, progressive rock and musical theatre output to traditional art music (e.g. chamber music, solo repertoire). I had been using computers to engineer and record a lot of my music, but it was not until graduate school at New York University that I began to write software in which the computer would actually compose music.

Invention 16

My first computer/software-based algorithmic composition was *Invention 16* (2006). This transformative algorithmic composition for two pianos is an atonal work in 15 short movements. Not a true invention, this piece gets its name from its design: the program creates a histogram of every note and rhythm in each of Bach's 15 two-part inventions and then composes a new piece based on the data from the histogram.

Description

Figure 1.0 shows the data flow for this program. I began by creating a single MIDI file of all of Bach's 15 two-part inventions. My intention was to send the MIDI file through this software and have each MIDI note become part of a histogram. Once the file was done playing, the same file would play back using Bach's rhythms but with different

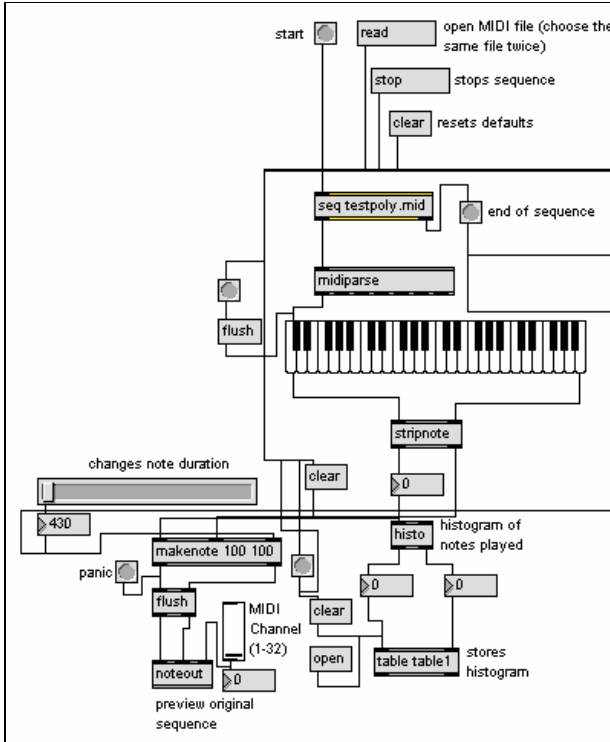


Figure 1.1 A close-up look at the first half of the Dislocator software. A MIDI file is read into the file where its MIDI data is stored in a table (table1).

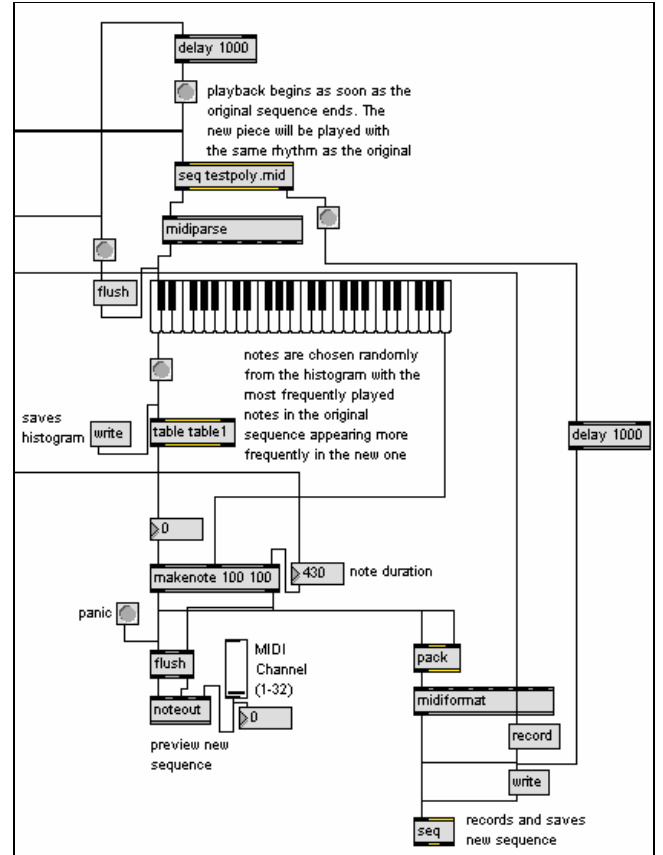


Figure 1.2 A close-up look at the second half of the Dislocator software. This is the part of the program that uses the MIDI file's rhythm to trigger playback using notes as they appear in the histogram.

Modal Change

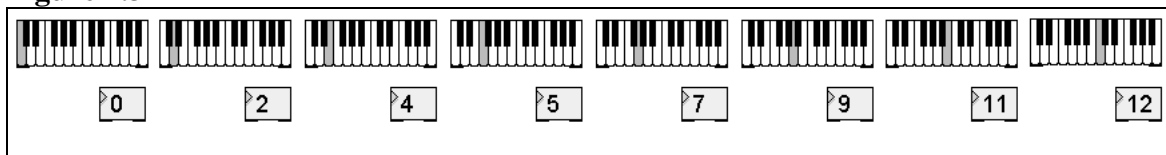
While I was still enjoying the personal success of *Invention 16* (though it was very simplistic in design), I had already begun working on what would soon be version 1.0 of the Modal Change object, the first object in what became the Modal Object Library.

My efforts became this piece, *Modal Change (2007)*, a generative algorithmic composition for string orchestra, harp, and piano composed in real-time.

Description

Initially, my program contained seven kslider objects which display seven two-octave keyboards. This allows a user to manually select each note of any mode starting on C (pitch class 0) through to B (pitch class 11). All notes entered send an appropriate pitch class number to seven number boxes (see figure 1.3).

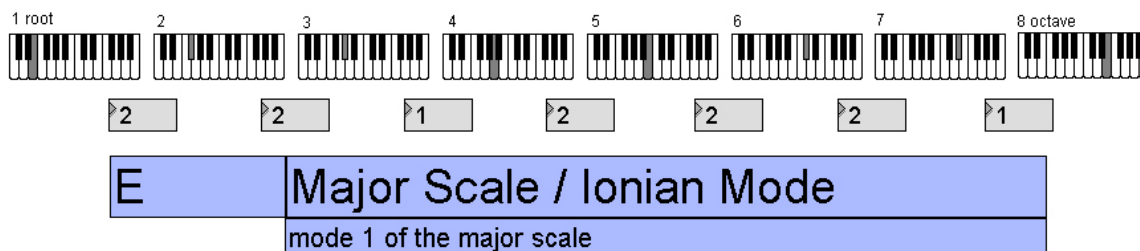
Figure 1.3



Once we have a pitch class number for all seven degrees, we are able to map the distance between any two degrees by subtracting the first scale degree's pitch class number from the second scale degree's pitch class number. In Figure 1.5, we see that the first scale degree is 0 and the second is 2. When we subtract scale degree 1 (0) from scale degree 2 (2), we will have the number of semitones between them (2). The difference of 2 represents two semitones, or one whole step.

When we repeat this process for all 8 scale degrees (including the octave) we will get the distance map for the manually entered notes: (2 2 1 2 2 2 1). We will pack these numbers into a list using the pack object and then, using the match object, we will cause any occurrence of this pattern to display the first scale degree (as a letter name, not a pitch class number) followed by the phrase "Major Scale/Ionian Mode" & "Mode 1 of the Major Scale". The distances between the scale degrees of C Major are identical for all major scales, so entering the notes of E Major will yield the same distance map and, thus, display.

Figure 1.4



By creating 28 match objects for each of the 28 modes defined earlier, we can map all of the scale degree distance maps (see below) to the appropriate mode names.

	Major	Harmonic Major	Harmonic Minor	Melodic Minor
Mode 1	2 2 1 2 2 2 1	2 2 1 2 1 3 1	2 1 2 2 1 3 1	2 1 2 2 2 2 1
Mode 2	2 1 2 2 2 1 2	2 1 2 1 3 1 2	1 2 2 1 3 1 2	1 2 2 2 2 1 2
Mode 3	1 2 2 2 1 2 2	1 2 1 3 1 2 2	2 2 1 3 1 2 1	2 2 2 2 1 2 1
Mode 4	2 2 2 1 2 2 1	2 1 3 1 2 2 1	2 1 3 1 2 1 2	2 2 2 1 2 1 2
Mode 5	2 2 1 2 2 1 2	1 3 1 2 2 1 2	1 3 1 2 1 2 2	2 2 1 2 1 2 2
Mode 6	2 1 2 2 1 2 2	3 1 2 2 1 2 1	3 1 2 1 2 2 1	2 1 2 1 2 2 2
Mode 7	1 2 2 1 2 2 2	1 2 2 1 2 1 3	1 2 1 2 2 1 3	1 2 1 2 2 2 2

	Major	Harmonic Major	Harmonic Minor	Melodic Minor
Mode 1	Ionian	harmonic major	harmonic minor	melodic minor
Mode 2	Dorian	Dorian b5	Locrian #6	Dorian b2
Mode 3	Phrygian	Phrygian b4	Ionian #5	Lydian #5
Mode 4	Lydian	Lydian b3	Dorian #4	Lydian b7
Mode 5	Mixolydian	Mixolydian b2	Phrygian #3	Mixolydian b6
Mode 6	Aeolian	Aeolian b1	Lydian #2	Locrian #2
Mode 7	Locrian	Locrian b7	Mixolydian #1	Locrian b4

These processes, in essence, became what is now the Modal Change object; an object that takes MIDI pitch class names or numbers in its right inlet, and mode names or scale degree distance maps in its right inlet and outputs discrete scale degrees. The pitch data is stored in a table. Several subpatches I call “pitch engines” are responsible for “banging” the table to produce one of the notes from selected mode. Using the tempo object, I have some pitch engines banging the table on whole notes, half notes, etc. The resulting pitch classes that are sent from the table are multiplied by a factor 12 which assigns them to a particular octave designation (See Figure 1.5).

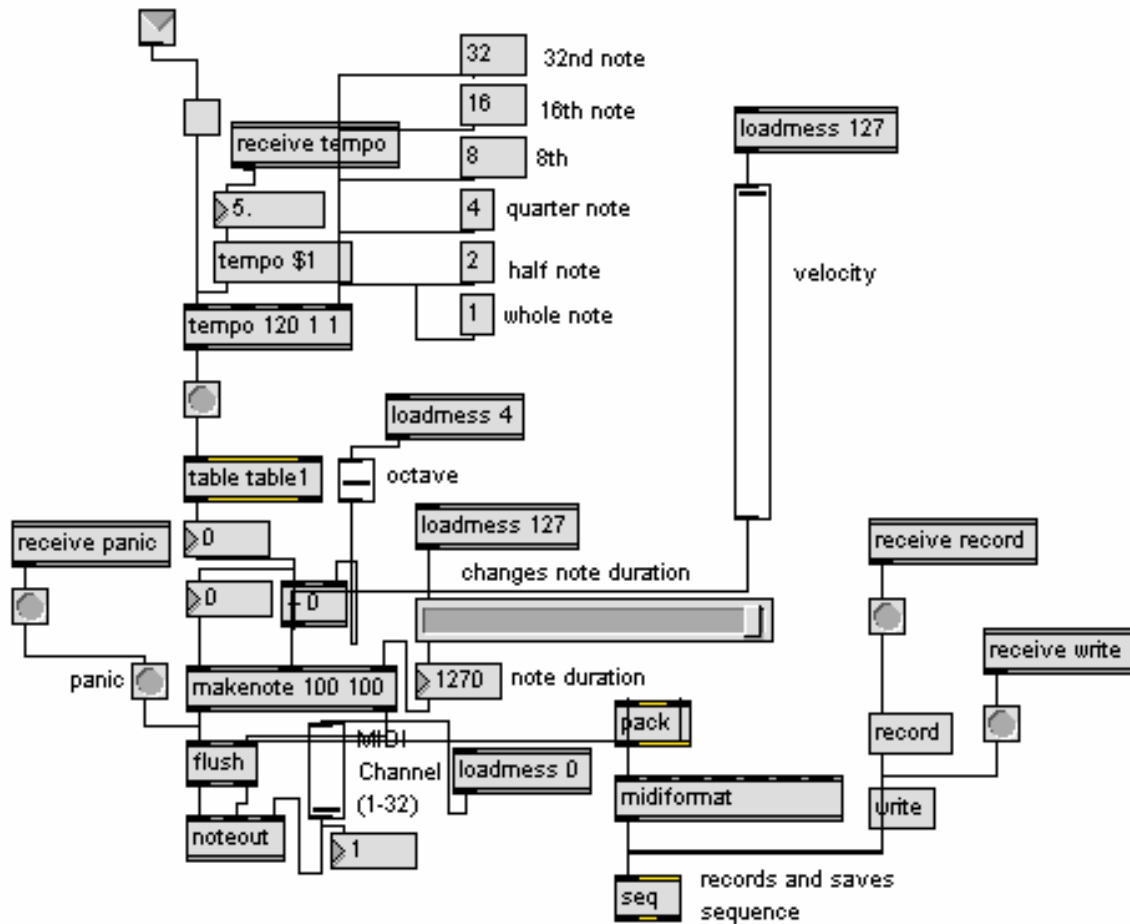


Figure 1.5 Pitch engine generates notes from the table.

Figure 1.7 *21 Related Modes of the Harmonic Major Scale*

Scale	Move Scale Degree 1	Move Scale Degree 2	Move Scale Degree 3	Move Scale Degree 4	Move Scale Degree 5	Move Scale Degree 6	Move Scale Degree 7
Mode 1 C Harmonic Major			down - (C) Harmonic Minor			up - (C) Ionian	down - (C) Mixolydian b6
Mode 2 D Dorian b5		down - (D) Locrian #6			up - (D) Dorian	down - (D) Locrian #2	
Mode 3 E Phrygian b4	down - Eb Ionian #5			up - (E) Phrygian	down - (E) Locrian b4		
Mode 4 F Lydian b3			up - (F) Lydian	down - (F) Melodic Minor			down - (F) Dorian #4
Mode 5 G Mixolydian b2		up - (G) Mixolydian	down - (G) Dorian b2			down - (G) Phrygian #3	
Mode 6 Ab Aeolian b1	up - A Aeolian	down - (Ab) Lydian #5			down - (Ab) Lydian #2		
Mode 7 B Locrian b7	down - Bb Lydian b7			down - (B) Mixolydian #1			up - (B) Locrian

any of the Harmonic Major Scale modes (in blue) can become one of these modes (in red) by changing just one note up or down one semitone

Figure 1.8 *21 Related Modes of the Harmonic Minor Scale*

Scale	Move Scale Degree 1	Move Scale Degree 2	Move Scale Degree 3	Move Scale Degree 4	Move Scale Degree 5	Move Scale Degree 6	Move Scale Degree 7
Mode 1 C Harmonic Minor			up - (C) Harmonic Major			up - (C) Harmonic Minor	down - (C) Aeolian
Mode 2 D Locrian #6		up - (D) Dorian b5			up - (D) Dorian b2	down - (D) Locrian	
Mode 3 Eb Ionian #5	up - (Eb) Phrygian b4			up - (Eb) Lydian #5	down - (Eb) Ionian		
Mode 4 F Dorian #4			up - (F) Lydian b7	down - (F) Dorian			up - (F) Lydian b3
Mode 5 G Phrygian #3		up - (G) Mixolydian b6	down - (G) Phrygian			up - (G) Mixolydian b2	
Mode 6 Ab Lydian #2	up - A Locrian #2	down - (Ab) Lydian			up - (Ab) Aeolian b1		
Mode 7 B Mixolydian #1	down - Bb Mixolydian			up - (B) Locrian b7			up - (B) Locrian b4

any of the Harmonic Minor Scale modes (in blue) can become one of these modes (in red) by changing just one note

Figure 1.9 *18 Related Modes of the Melodic Minor Scale*

Scale	Move Scale Degree 1	Move Scale Degree 2	Move Scale Degree 3	Move Scale Degree 4	Move Scale Degree 5	Move Scale Degree 6	Move Scale Degree 7
Mode 1 C Melodic Minor			up - (C) Ionian	up - (C) Lydian b3		down - (C) Harmonic Minor	down - (C) Dorian
Mode 2 D Dorian b2		up - (D) Dorian	up - (D) Mixolydian b2		down - (D) Locrian #6	down - (D) Phrygian	
Mode 3 Eb Lydian #5	up - E Phrygian	up - (Eb) Aeolian b1		down - (Eb) Ionian #5	down - (Eb) Lydian		
Mode 4 F Lydian b7	up - F# Locrian b7		down - (F) Dorian #4	down - (F) Mixolydian			up - (F) Lydian
Mode 5 G Mixolydian b6		down - (G) Phrygian #3	down - (G) Aeolian			up - (G) Mixolydian	up - (G) Harmonic Major
Mode 6 A Locrian #2	down - Ab Lydian #2	down - (A) Locrian			up - (A) Aeolian	up - (A) Dorian b5	
Mode 7 B Locrian b4	down - Bb Ionian			up - (B) Locrian	up - (B) Phrygian b4		down - (B) Mixolydian #1

any of the Melodic Minor Scale modes (in blue) can become one of these modes (in red) by changing just one note

My task, then, was to make the program modulate to one of these modes. For *Modal Change*, I created a map of modes for the software to use by sending MIDI messages to trigger different modes. This research, however, became the basis of the second object in the Modal Object Library: Modal Shift (used in *Evolution* (2007)). Once the overall harmonic direction was realized, the piece was ready to be performed by the software.

Evolutio

This algorithmic composition for string quartet was composed in real-time with software based on the ideas and algorithms of my earlier work *Modal Change*. It differs in that the harmonic emphasis is focused more narrowly on tertian triadic harmony despite its four independent voices. It uses many objects in my Modal Object Library including modal change (to control modality), modal shift (for common-tone modulation), modal triad (to establish tertian harmony for each mode), modal prog (to control chord progressions).

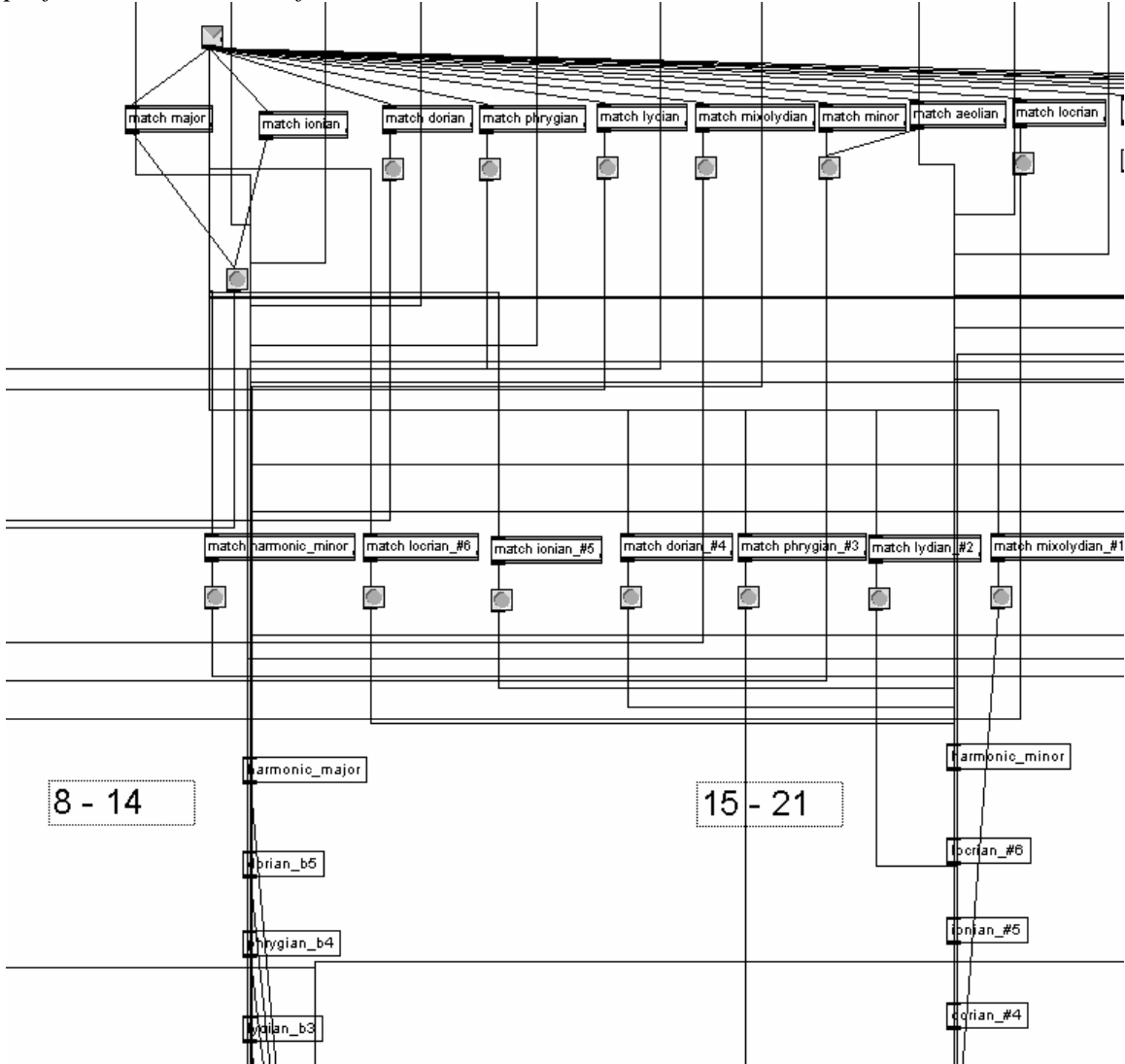
Description

The necessity for more objects in the Modal Object Library became apparent in the creation of the piece *Modal Change* (2007). The next object I began developing was the Modal Shift object, an object to do the type of common-tone modulation that can be seen in the aforementioned composition.

The Modal Change object can accept mode name messages in order to successfully change the mode. The new object I was developing would send out mode name messages in order to integrate with the Modal Change object.

First, there was the need for this object to accept mode name messages as well, so it would know what mode the user selected and calculate which modes were just one scale degree away. For this, I used the match object for both tonic and mode name (tonic also accepts MIDI pitch class numbers).

Figure 2.0 *The Modal Shift Object taking mode name messages as input in order to perform a calculation of related modes*



The next step was to cause the triggering of one mode to calculate all related modes. I labeled each mode with a number so that I could easily call up a mode with integers instead of messages. For example, the dorian mode could be called up by sending the number two to the output where it would be converted back into message to be received by the modal change object. The reason for this will become apparent in a moment.

In order to calculate related modes, I needed to generalize the way that one mode would become another with respect to the related modes list (see Modal Change). If I selected to C Ionian as my original mode, the program would tell me that C Lydian was just one note away. It knows that when an Ionian message is received (1), retain the tonic and select the Lydian mode (4). However, what happens to the other 41 modes related to C Ionian?

In order for C Ionian to become C# Locrian b4 (mode 7 of the melodic minor scale), the program takes the input (Ionian=1) and outputs the mode message for Locrian b4 (28). Simultaneously, the value of tonic (C=0) is calculated and has 1 added to it in order to change the tonic from C to C# (See Figure 2.1).

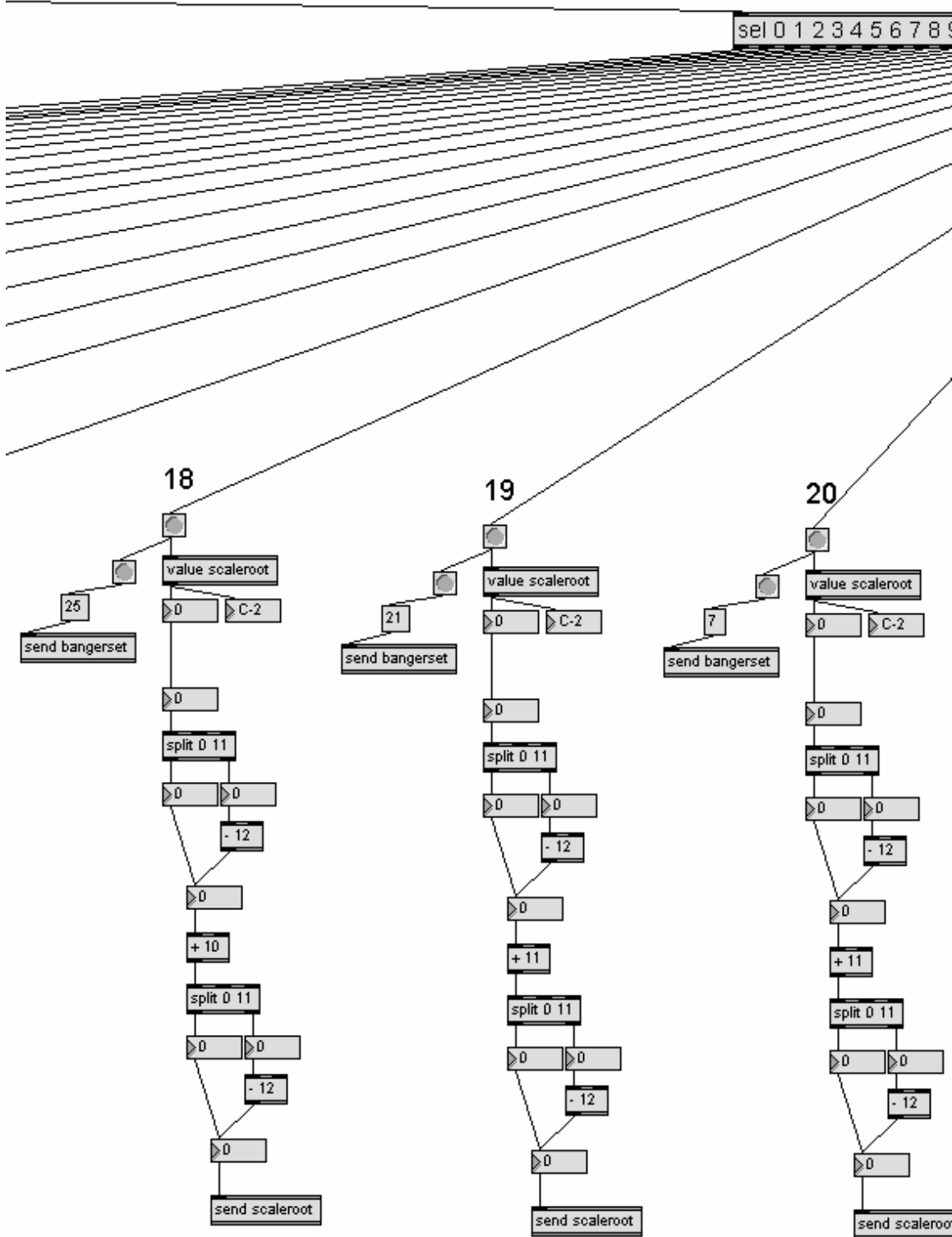


Figure 2.1 *The Modal Shift Object triggering new modes by evaluating the tonic and sending a new mode message*

For *Evolutio*, the object calculated related modes using a 1st order Markov chain. As we will see, future compositions, like *Ravelian Fuse* (2007), used the Modal Shift List object which populates the related modes into a list for the user to select.

With a device to switch easily between related modes, I decided to put my object into practice with a new composition. *Modal Change* had subtle minimalist gestures when performing this common-tone modulation. I decided to use a 4 voice texture for this piece with more rhythmic freedom.

In order to accomplish this freedom, I needed to build some rules to govern how each voice would play each note. I set up a tempo object to calculate the steady beat of the entire program. I also told the program that I'd be using 16 bar phrases with a cadence on two half notes at the end of every 16 bars. With the steady beat realized, I could give each of my voices the freedom to choose different rhythmic note values (whole notes, half notes, etc.) while still maintaining respect to the steady beat (if for no other beats, for at least the cadence). In order to achieve proper legato for each voice, I divided the tempo by the note value selected to achieve a MIDI note duration value.

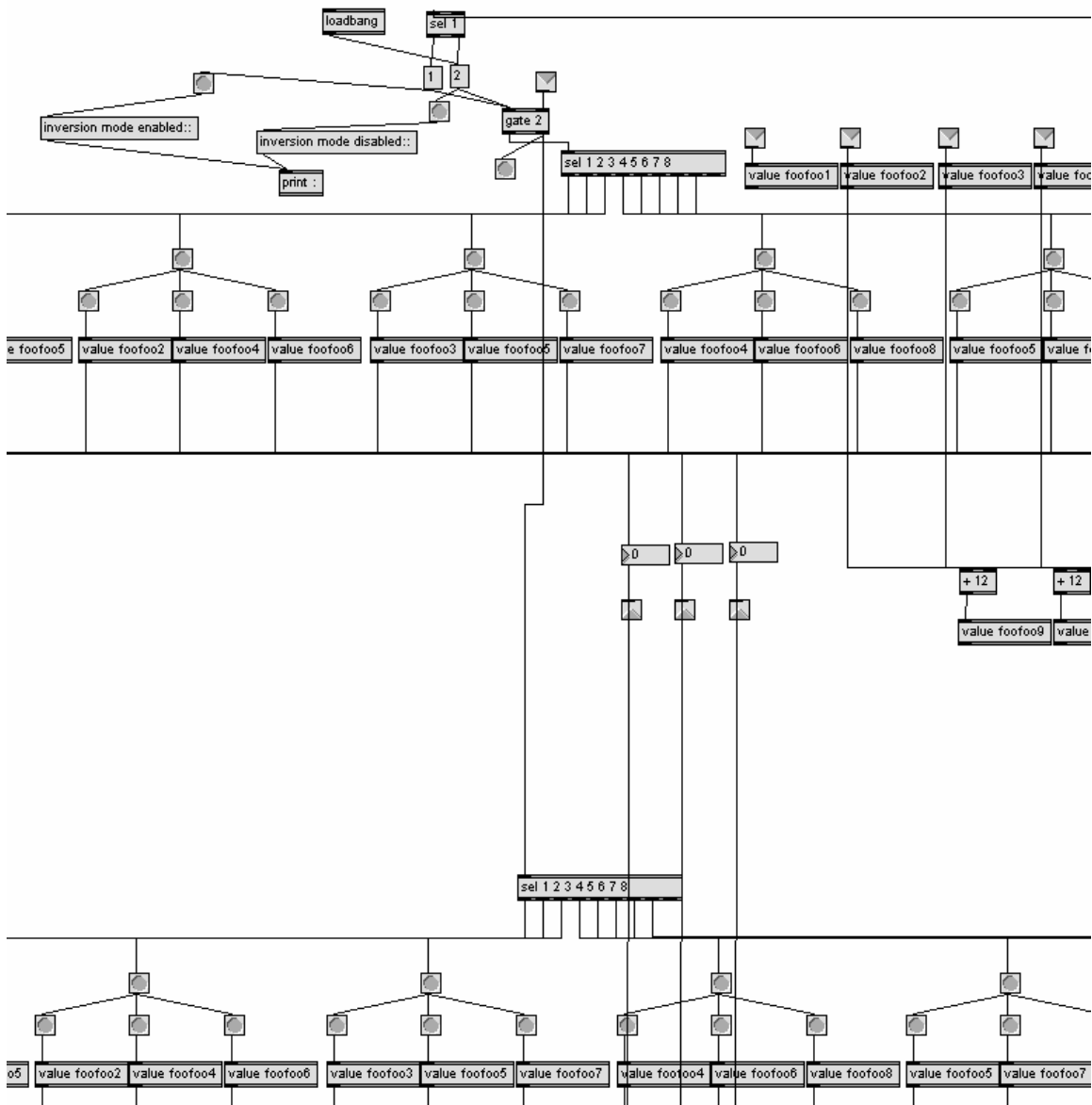


Figure 2.3 *The internal working of the Modal Triad object. Gives root position and inverted triads*

With the voice of each triad separated, I implemented this into my composition. As you can see, the possibilities are divided into three section: 1) notes from the triad, 2) the possibility for a note to be either from the triad or a non-chord tone using the notes from the mode as stored in the table (see Modal Change), or 3) cadence notes, a specific tone for each voice when the music cadences (every 16 bars).

With each voice generating pitch classes set to a different octave (multiples of 12), the next step was to make the music go somewhere within the context of a single tonal center (i.e. before the common-tone modulation). To accomplish this, I created an object that would simply give us progressions for a given mode.

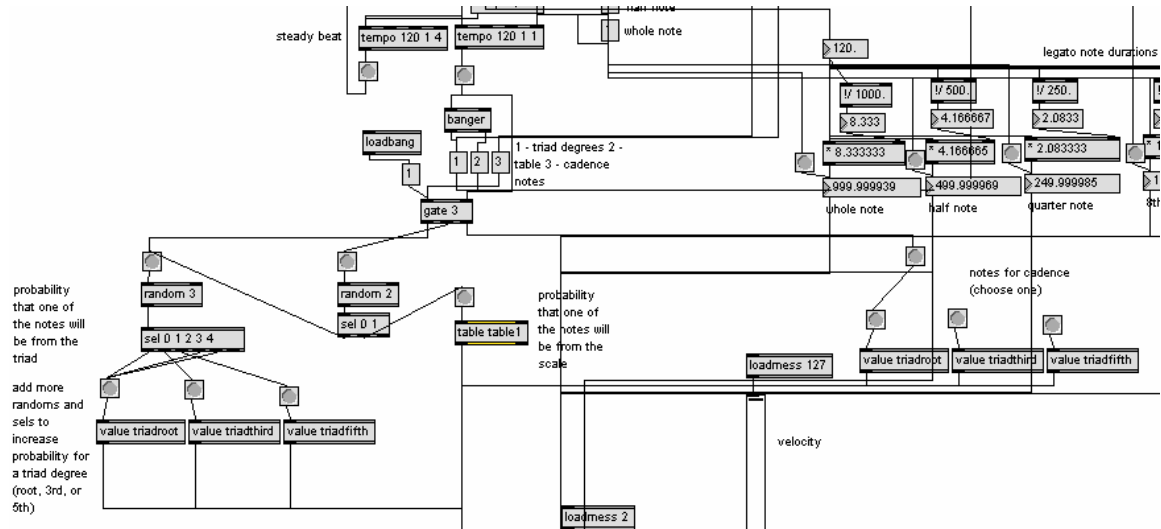


Figure 2.4 Pitches from the Modal Triad object being played using rhythm

The Modal Prog object is quite simple; it takes a list of numbers, which represent scale degrees, in a message box and plays the appropriate harmonic function for each scale degree as dictated by the mode. For instance, in the C Ionian Mode, 1 4 5 4 will play the chords 'C F G F'. A 1564 will play "C G a min F" since a minor naturally occurs as the triad built on scale degree six in C Major (Ionian).

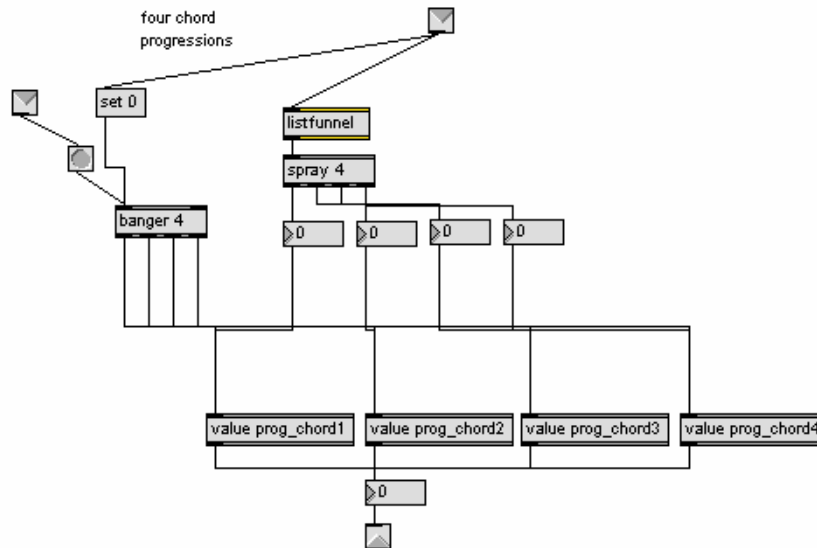


Figure 2.5 *The Modal Prog object (shown here with 4 possible chords per phrase)*

By setting the modal prog object (16 bar phrases) to the steady beat tempo object, I had a 16 bar progression that would last until the first common-tone modulation (Modal Shift). My voices were divided by register and able to create their own rhythm by choosing rhythmic values.

The resulting piece sounded much more like a late classical/romantic string quartet than the minimalist style of *Modal Change* and was, indeed, labeled as a string quartet.

Sometime later, I decided to use this software to perform 4 bar progressions in my Church at the closing of the service. In modifying this program, I created an engine that plays sustained triads on whole notes with a warm pad sound, and a melody improviser that emulates the minimalist style I've played and heard universally in this Church music using single notes with a piano sound. This program's (called *V-Bot* (2007)) performances was so well-received by the congregation in my home Church and many other Churches, that the demand came for CD's to be released.

Ravelian Fuse

After *Evolutio*, I created a new set of scale objects for the Modal Object Library: the Modal Messiaen objects. The Modal Messiaen objects are 4 objects that operate almost identically to the modal change object. Instead of giving the modes of the 7-note collections, they give the *Modes of Limited Transposition* as named by Olivier Messiaen. The same construction based on degree distances is used and a user may easily add custom degree distances patterns (degree distances are shown in Figure 2.6).

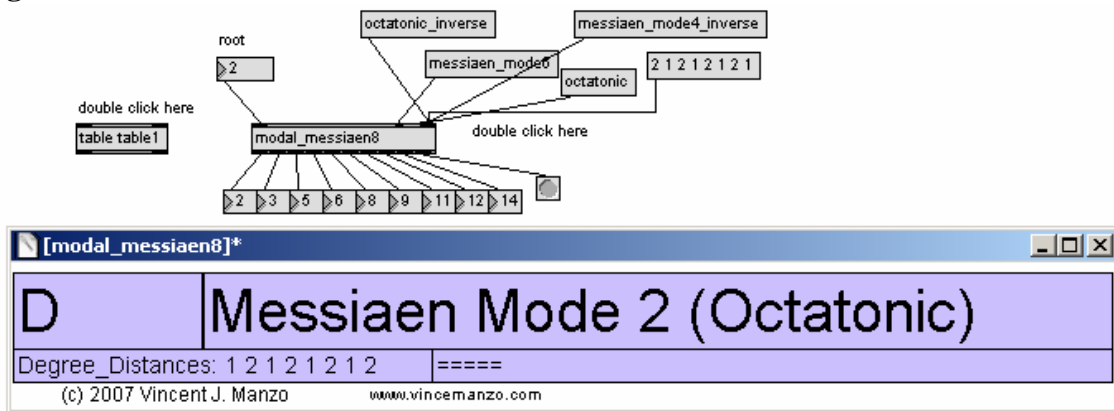
Figure 2.6

Olivier Messiaen's Modes of Limited Transposition	
Organized by number of scale degrees	
Messiaen 6 Note Collections	
○ 2 2 2 2 2 2	Whole-tone, Messiaen mode 1
○ 1 1 4 1 1 4	Messiaen mode 5
○ 4 1 1 4 1 1	Messiaen mode 5 inverse
Messiaen 8 Note Collections	
○ 1 2 1 2 1 2 1 2	Octatonic, Messiaen mode 2, Half-Whole step scale
○ 2 1 2 1 2 1 2 1	Octatonic, Messiaen mode 2 inverse, Whole-Half step scale
○ 1 1 1 3 1 1 1 3	Messiaen mode 4
○ 3 1 1 1 3 1 1 1	Messiaen mode 4 inverse
○ 1 1 2 2 1 1 2 2	Messiaen mode 6
○ 2 2 1 1 2 2 1 1	Messiaen mode 6 inverse
Messiaen 9 Note Collections	
○ 1 1 2 1 1 2 1 1 2	Messiaen mode 3
○ 2 1 1 2 1 1 2 1 1	Messiaen mode 3
Messiaen 10 Note Collections	
○ 1 1 1 1 2 1 1 1 1 2	Messiaen mode 7
○ 2 1 1 1 1 2 1 1 1 1	Messiaen mode 7 inverse

The objects *Modal Messiaen 6*, *Modal Messiaen 8*, *Modal Messiaen 9*, and *Modal Messiaen 10* each generate the number of scale degrees specified by the argument (following ‘*Modal Messiaen*’). In addition to generating the appropriate note collection when requested by sending a message to the right inlet, these objects will take any scale degree pattern in the right inlet as well allowing the user to add their own custom modes for 6, 8, 9 and 10 note collections in addition to the custom 7-note modes available through the Modal Change object.

The Modal Messiaen objects also have the same table feature as the Modal Change object which allows their data to be stored in a table in addition to supplying each scale degree of the mode. Figure 2.7 shows the *Modal Messiaen 8* object and the display window.

Figure 2.7



After creating these objects, I began composing a new composition that implemented them in a tonal context. I decided to use the Messiaen Modes in the way that they appear is a lot of the work of Ravel, Debussy and other impressionist composers.

In the Ravel & Debussy, the 6 tone ‘Whole Tone’ scale (Messiaen Mode 1) can be easily modulated to from the melodic minor scale. In A Melodic Minor:

(a b c d e f# g# a) = degree distances (2 1 2 2 2 1)

the degree distances: closely resembles that of the whole tone scale (2 2 2 2 2). If scale degrees 1 and 2 of A Melodic Minor: (a b c d e f# g#) are averaged, we get a six note collection beginning on Bb with the scale degree distances (2 2 2 2 2): Bb C D E F# G#.

The algorithms in *Ravelian Fuse* integrates with the Modal Shift object to introduce this type of modulation to the whole tone scale as the 29th common-tone mode of melodic minor that can be reached by changing just one scale degree.

A similar modulation occurs in this piece from the melodic minor scale to the Octatonic scale (2 1 2 1 2 1 2 1) (Messiaen Mode 2 Inv.). If scale degree 5 of A Melodic Minor:(a b c d e f# g#) is split into two scale degrees (one half step above and below), we get an 8 note Octatonic collection with the scale degree distances (2 1 2 1 2 1 2 1): a b c d eb f f# g# a.

With those differences stated, the way pitches are produced is an amalgam of the techniques used in *Invention 16* (2006), *Modal Change* (2007) and *Evolutio* (2007). The same concept of triadic harmony being chosen by independent voices (e.g. *Evolutio*) with a optional non-chord tones via a central table with all pitch classes present (e.g. *Modal Change*) is present in *Ravelian Fuse*. The piece borrows transformative rhythm techniques from *Invention 16*, using another MIDI file as a rhythmic impetus for pitch creation.

In this piece, the MIDI file used was my string quartet *Sonatine* (2006). Each of the 4 voices from *Sonatine* were sent to a different pitch engine for *Ravelian Fuse*. Once the MIDI file had run its course through the software, the files were quantized in order to remove EDUs below 256. This made the piece much less intense rhythmically.

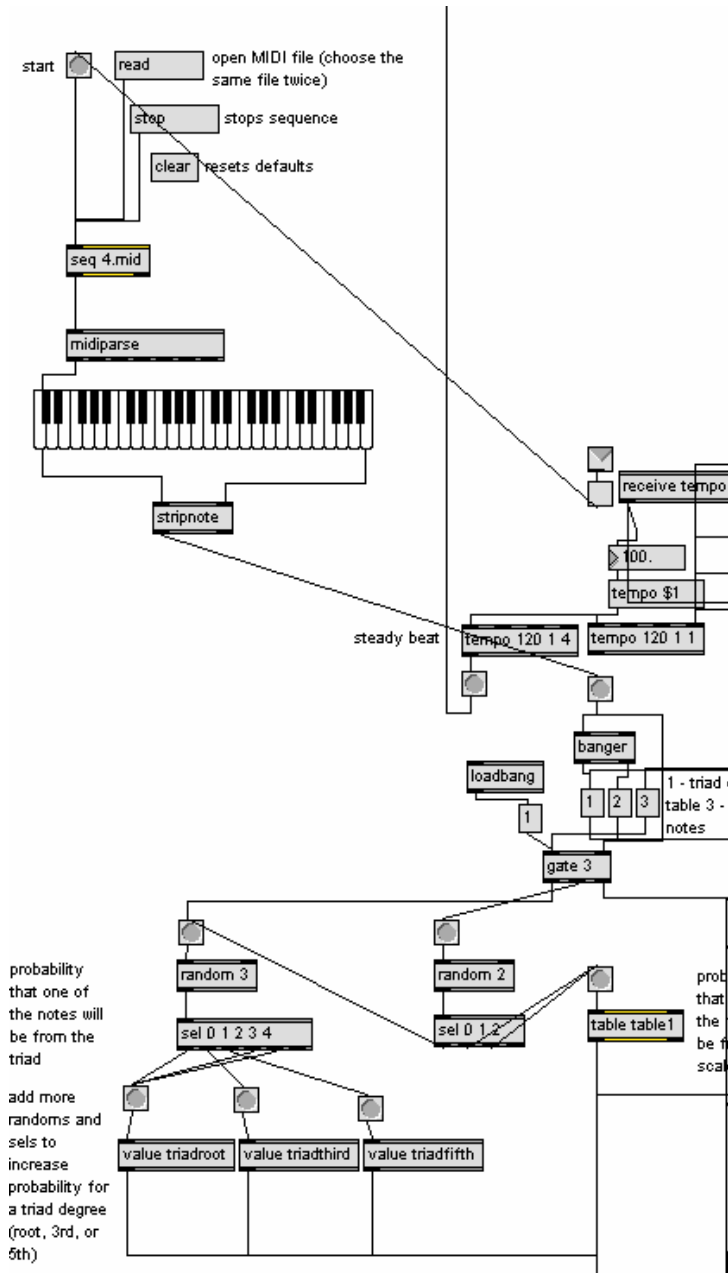


Figure 2.8 *Ravelian Fuse* pitch engine triggered by MIDI

The piece is arranged for two pianos and cello due to its somewhat non-idiomatic piano writing.

The Modal Object Library continues to expand with new algorithms and features for algorithmic composition. As it expands, new compositions and interactive music systems will be created. The Modal Object Library can be obtained through vincemanzo.com.

Appendix

ⁱElsea, Peter, *LObjects* University of California, Santa Cruz

Rowe, R., 1993. *Interactive Music Systems*. Cambridge: MIT Press.