Implementing Modality in Algorithmic Composition

V. J. Manzo
Steinhardt School of Education
New York University
New York, NY 10003
vjm229@nyu.edu

ABSTRACT

Data-driven and rule-based algorithmic composition techniques can benefit from a global set of data constructs to work within a tonal framework. The Modal Object Library is a collection of algorithms to control and define modality in the Max/MSP/Jitter programming environment. A description of each data object is included as well as a description of the programming technique used toward its development. An explanation of the theoretical musical concepts related to the implementation and application of these objects is described in detail. Examples of algorithmic music composed with these objects are given. The object is available from the author's website at http://www.vjmanzo.com/mol.

Categories and Subject Descriptors
Programming Languages: Language Constructs and Features – *classes and objects, control structures.*

General Terms: Languages.

Keywords: Algorithmic Composition, Max/MSP/Jitter, Visual Programming Language, Multimedia, Digital Audio, Computer Music, Modality, Music Theory Models, Modal Object Library

1. INTRODUCTION

The Modal Object Library [1] is a collection of algorithms to define and control modality in the max/MSP/Jitter [2] programming environment. 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.

This library was created primarily to aid in my own compositional interests including algorithmic compositions and interactive music systems. Examples of my works that implement these objects can be heard and seen at www.vjmanzo.com/cv. This library was originally conceived to aid in teaching modal relationships to my theory students.

2. BACKGROUND

Musical modes are collections of notes in some ordered class with a structured distance between pitches. A wealth of literature exists concerning the Greek modes and the Church modes. For the purposes of this paper, we will focus on the 28 modes that are derived without chromatic alteration as they occur from the seven modes of the major scale, the seven modes of the harmonic minor scale, the seven modes of the melodic minor scale, and the seen modes of the harmonic major scale.

The sense of modality occurs by creating a sense of tonic relationship between a given sale degree and the other 6 notes of the collection. The major scale (also known as the *Ionian* mode), has a unique distance structure of whole steps (W) and half steps (H). This structure can be read (W W H W W W H) whereas the distance between the tonic and the next note is a whole step (**W** W H W W W H). Scale degree two is a whole step below scale degree three (W **W** H W W W H). Scale degree three is a half-step below scale degree four (W W H W W W H).

In order to have a computer interpret these distances, we simple calculate the semitones (half steps) as integers: whole step = 2 (two semitones), half step = 1 (one semitone). The major scale distance map can now be written as (2 2 1 2 2 2 1) instead of (W W H W W W H).

The harmonic minor scale and the melodic minor scale can also be reduced to interpretation by scale degree distance. The harmonic minor scale, however, contains the interval of an augmented second, a whole step plus one semitone. Its scale degree distance map is written (2 1 2 2 1 3 1). The melodic minor scale (ascending) is written as (2 1 2 2 2 2 1).

What I am calling the *Harmonic Major* scale is the seven note collection similar to the major scale except with a flatted sixth scale degree. Flatting the sixth scale degree makes way for the introduction of what are known as *borrowed chords*, chords "borrowed" from the parallel key. Triadic chord realization from the Harmonic major scale yields seven chords similar to that of the major scale with exception of the introduction of some triads that are borrowed from the parallel minor key.

Figure 1 shows the triads that occur naturally in the C Major scale compared to the triads that occur naturally in the C Harmonic Major scale.

In the harmonic major scale, both triads built on scale degrees two and four occur naturally in C minor. An augmented triad is also introduced on the flatted sixth scale degree. The harmonic major scale degree distance map can be written as (2 2 1 2 1 3 1).

C Major **Scale**	**I** C – E – G (C Major)	**ii** D – F – A (d minor)	**iii** E – G – B (e minor)	**IV** F – A – C (F Major)	**V** G – B – D (G Major)	**vi** A – C – E (a minor)	**vii**^{0} B – D – F (b^0)
C Harmonic **Major Scale**	**I** C – E – G (C Major)	**ii**^{0} D – F – Ab (d minor)	**iii** E – G –B (e minor)	**iv** F – Ab – C (F Minor)	**V** G – B – D (G Major)	**vi** Ab – C – E (Ab+)	**vii**^{0} B – D – F (b^0)

Figure 1. Triads as they occur in diatonic keys

In each of these four instances, the tonic of the aforementioned scale is the first scale degree. When a strong sense of tonal center is placed on another scale degree, we refer to the collection as a mode of the new tonal center; the same collection of pitches with a new sense of tonic.

Figure 2 shows the scale degree distance map for all 28 modes of the major, harmonic minor, melodic minor, and harmonic major scales. We will use the Greek names commonly used to refer to the modes of the major scale.

	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

Figure 2. Step patterns for all modes in diatonic keys

Ionian	*Dorian*	*Phrygian*	*Lydian*	*Mixolydian*	*Aeolian*	*Locrian*
Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7

Figure 3. Traditional Greek mode names used to label diatonic major modes

There are differing views held concerning the names of the modes of other seven-note collections. Here, I have named the modes according to their similarity to one of the common modes accounting for the least amount of chromatic inflections. For examples, Mode 4 of the melodic minor scale (2 2 2 1 2 1 2) most closely resembles the Lydian mode with a flatted seventh scale degree. Thus it is named Lydian b7.

Figure 4 shows all of the modes named according to the Greek names with the least amount of chromatic inflections.

	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

Figure 4. Chromaticized Greek labels applied to other scales

3. ALGORITHM LIBRARY OVERVIEW

The Modal Object Library is a collection of objects I have created for Cycling 74's graphical programming language Max. These objects allow programs to understand and interpret advanced knowledge of modality including analysis, common-tone modulation, and triad realization.

Modal Change

The modal_change object allows a user to specify a tonic and diatonic mode in its two inlets and get the pitch class value of each scale degree out its eight outlets. A user can send a pitch class number or a letter name message to its left inlet to set the tonic. A message box with a mode name such as major, minor, Phrygian, Lydian b7, can be sent to the right inlet to build up a scale from the given tonic. The object will output the scale degrees for any tonic within the modes of the major scale, the melodic minor scale, the harmonic minor scale, and the harmonic major scale (the major scale with flatted 6). Instead of using one of the mode names to build a scale, a user can also send a message with the number of whole steps and half steps desired to build their scale, and receive the scale degree pitch classes from its outlets.

Double clicking the object will open a display that allows the user to see what mode they're in and other information related to the mode including scale degree distances that make up the scale and the particular mode's context within the larger pitch collection.

The object can receive all of the organized pitch class data into a table or by using an internal table with the argument table1.

The object can also receive the organized pitch class data into a coll list or by using an internal coll list with the argument scale. The coll list also has an added feature: it will take any incoming pitch and filter it to the nearest note from the selected scale.

This allows you to set the tonic and mode, and filter all incoming pitch data so that whatever note is played, it will conform to the diatonic pitch collection you've selected.

Modal Triad

The modal_triad object allows a user to play tertian chords of any quality. It receives scale data from the modal_change object and, when a tonic and mode is selected, the object receives the numbers 1-8 in its leftmost inlet to output the notes of the chord function associated with that number. For example, in major keys, the numbers 1, 4 and 5 are always major chords, 2, 3, and 6 are minor, so, if C Major is selected, a 2 sent to the modal_triad object will yield the notes of a d minor chord.

For each selected chord, the notes of that chord are sent to the object's 7 outlets in the following order: root, third, fifth, seventh, ninth, eleventh, and thirteenth. Alterations like flat ninth or sharp eleventh are inferred by the chord function as it relates to the selected tonic and mode.

The second inlet of the object allows the chord tones, received as pitch classes, to be restricted to one octave.

The object also takes Roman numeral functions to yield chords. The standard capital Roman numerals for major, lower case Roman numerals for minor are used. A lower case Roman numeral iv in the key of C Major will yield an F minor chord regardless of the fact that chord has non-diatonic chord tones in it, the Ab.

A capital Roman numeral with a plus sign next to it will yield an augmented chord, and a lowercase Roman numeral with a zero next to it will yield a diminished chord.

In the same manner, a user can use letter names to build chords. A capital C will yield a C Major chord while a lower case e will yield an e minor chord. A capital C plus will yield an augmented chord and a lower case d zero will yield a d diminished chord.

This object also receives messages for tonicizations. A user can send the message Roman numeral V/5, to yield the 5 of 5 (a D Major chord in the key of C Major). The V Chord Tonicizations produce a Dominant 7th chord for each scale degree in the selected mode. That is, the root, 3rd, 5th, and 7th will form a Dominant 7th Chord exactly one perfect 5th above a given scale degree. The 9th, 11th, and 13th pitches of the chord are inferred according to the selected mode and NOT the mode from which the tonicizing chord prevails.

Similarly the object allows other types of tonicizations including leading tone tonications and minor four tonicizations. Augmented 6th chords and Neopolitan chords can also be implemented.

The object also has 7 switches that can be set to restrict notes from reaching the outlets.

Modal Prog

The modal_prog object takes a list of chords (as in a progression) in its right inlet and outputs each of those chords one at a time to the modal triad object when a bang is sent to the left inlet. The object integrates with the modal_triad object and will interpret any message that modal triad does.

By default, a new list of chords triggered when a list is currently being played will sound on the next bang received. With the optional argument '@immediate 0', a new list of chords triggered when a list is currently being played will sound as soon as each chord from the first list has been played.

Modal Shiftlist

The modal_shiftlist object receives a tonic and mode name in its inlets and populates a list of all related modes sharing 6 of 7 notes. It takes the pitches of the scale and moves each scale degree up or down one at a time to see if a new diatonic mode can be formed. This process will list 42 related modes for any of the major scale modes, 28 related modes for any of the melodic minor scale modes, 21 related modes for any of the harmonic minor scale modes, and 21 related modes for any of the harmonic major scale modes.

When one of these related modes is selected from the list, the object automatically repopulates the list with modes related to the new key.

Modal Shift

The modal_shift object is similar to the modal_shiftlist object, but it is optimized to randomly choose one of the related modes when a bang is sent to its left inlet.

Modal Mutation

The modal mutation object is identical to the modal_shift object, but it outputs only those related modes sharing a common tonic or a semitone inflection of the tonic.

Modal Mediant

The modal_mediant object receives a tonic and mode name in its inlets and populates a list of modes in a chromatic median relationship with the initial mode. When one of these related modes is selected from the list, the object automatically repopulates the list with modes related to the new key.

Modal Messiaen Objects

The Modal Messiaen objects operate similarly to the modal_change object, but output the pitch classes of Messiaen's Modes of Limited Transposition. You can send it messages like C

Whole Tone or E octatonic to receive the pitch classes of that mode. These objects are also useful for creating modes with 6, 8, 9, or 10 unique scale degrees as it also accepts scale degree distances.

Modal Analysis Objects

The modal_analysis object takes incoming notes in its left inlet and determines in what mode and tonic you're playing when a bang is sent to its right inlet. The object attempts to filter out repetitions and organize notes to infer a mode. Double clicking the object will reveal a window similar to that of the modal change object which shows the mode as well as the scale degree distances that make up the scale and the particular mode's context within the larger pitch collection.

The ordered scale degrees are output as a list from the objects left outlet and the scale degree distances are output from its second outlet.

The modal_analysis+ object does everything modal_analysis does, but is also set to integrate with the modal change object to trigger a new mode change when a mode is analyzed. A user could conceivably play a scale, have it analyzed and then generate chords from that scale in real-time.

Modal PC Match

The modal_pc_match object takes an incoming note in its left inlet and compares it against the diatonic pitch classes of any scale as defined by the modal_change object. If the incoming pitch matches one of the pitch classes of the scale, the object outputs a bang from one of its first seven outlets.

The object also defines the chromatic notes between diatonic scale degrees. If an incoming pitch matches a chromatic scale degree, the object outputs a bang from one of the next 14 outlets. For example, a C# played in the key of C Major is between scale degrees 1 & 2 - C & D - a whole step. An incoming C# in any octave will send a bang out of the outlet marked "Match Scale Degree #1" [read Sharp One].

The incoming note may also match a chromatic scale degree between a step and a half (3 semitones). In this case, two chromatic notes are next to each other separated by a half step. The lower of the two chromatic notes is referred to as the "#1" (assuming that the step and half interval is located between scale degrees 1 and 2 as is the case in the sixth mode of the harmonic minor scale, Lydian #2). The other chromatic note is closer to the higher scale degree and would be referred to as "b2", thus the object would output the message "Match Scale Degree b2". For example, imagine a G played in the key of A Harmonic Minor (between scale degrees 6 & 7 - F & G#). An F# is interpreted as "#6" and the G is interpreted as "b7". Note: only the harmonic minor and harmonic major scales and their modes have two scale degrees separated by a step and a half.

In addition to matching chromatic pitches, the modal_pc_match object also outputs the chromatic pitch classes out of its last 14 outlets. Note that this means some notes will be redundant. For example, scale degree_b2 will be the same pitch as scale degree_#1 in Major keys. Once again, this will not be the case in the modes of harmonic minor and harmonic major where two pitch classes are separated by 3 semitones.

Modal Fuzzharm

The modal_fuzzharm object outputs a fuzzy logic chord harmonization based on incoming pitch classes. The object integrates with several objects in the modal_object library including the modal_pc_match object to determine if the incoming note played is diatonic or chromatic and that notes relationship in the context of the specified tonic and mode. For example, is the incoming note scale degree 1? Is it chromatic scale degree #4? If so, how do we want to harmonize that note when we receive it?

Double clicking the modal_fuzzharm object allows a user to see a table of chord symbols that the modal_triad object can interpret. The user can increase the probability weight to the table by clicking on one of the cells in the column for the desired chord/function listed in the top row. By default, all probabilities are set to zero. For example, if the incoming note matches scale degree one, you'd probably want to harmonize that note with the I chord, the IV chord and the vi chord since that scale degree one is present in all of these chords. Other chords may be used to harmonize that note as well, but you'd probably want the object to choose some chords more frequently than others, so we give them a higher table weight by clicking further down on the cells.

A bang sent to the modal_fuzzharm object will choose one of the chords to harmonize that note with based on the weightings you've specified. The table can be opened and presets can be saved. In the help file for this object, the seven diatonic scale degrees all have modal_fuzzharm objects connected so that when one of these scale degrees is played, the note will be harmonized in any way the user specifies. A specified table file has been loaded for each modal_fuzzharm object when the help opens which illustrates some default probability settings that harmonize these notes with diatonic chord functions.

Modal Filter

Modal_filter is a simple abstraction that utilizes modal_change's built in coll list feature. The abstraction shows an incoming pitch separated into octave class and pitch class. A chromatic pitch class is then filtered to a diatonic one and merged with its octave class. Thus, any non-diatonic notes will become diatonic ones.

Modal Strictmod

Modal_strictmod is a simple abstraction that allows the modal change object to cause a strict modulation from one key to another. It takes the initial tonic in its right inlet and the new tonic in its left inlet and outputs the index of transposition number through its outlet.

Modal Netsend & Modal Netreceive

The modal_netsend and modal_netreceive objects are simple abstractions designed to optimize sending modal_change messages over a network. It uses normal UDP network features for port and IP address specification.

Modal Line

The modal_line object operates similarly to the line object in terms of "ramping" from one number to the next. However, the ramp in modal_line only includes diatonic pitches of a specified mode.

Modal Chord Analysis

The the modal_chord_analysis object identifies a list of notes as tertian chords. The root, chord quality, harmonic function, inversion, and bass note are given as output. For added notes beyond the triad, M is used to indicate major intervals, m for minor intervals, P for perfect intervals. # and b are used to indicate alterations in perfect intervals.

4. EARLY LIBRARY DEVELOPMENT

This object library development began in Max version 4 with a single standalone application I wrote to help my theory students name the modes by manually entering the notes of a seven-note collection and asking the computer to name the mode. To achieve this, I began writing a Max patch that 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 5).

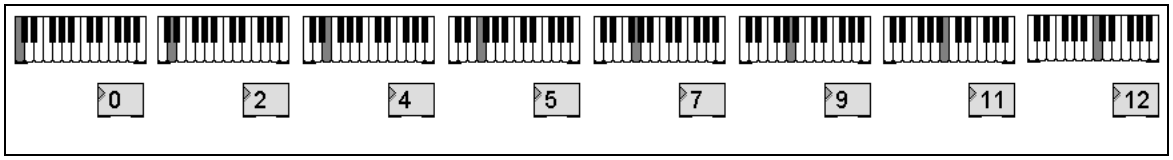


Figure 5. Pitch classes for the C Major scale

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 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.

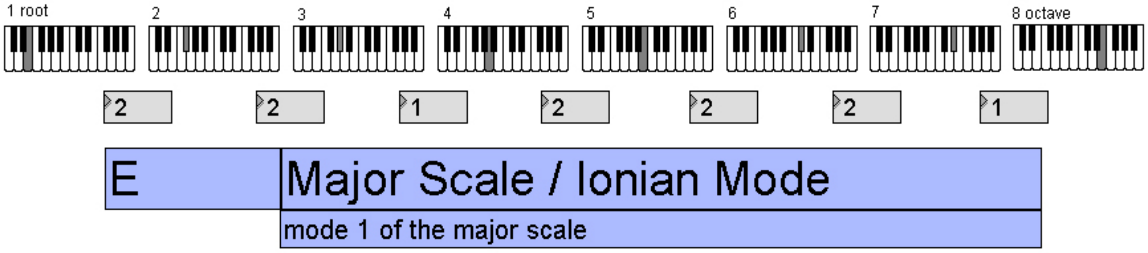


Figure 6. Pitch classes for the E Major scale showing modal context

By creating 28 match objects for each of the 28 modes defined earlier, we can map all of the scale degree distance maps (see Figure 2) to the appropriate mode names (see Figure 4).

Storing the Pitch Data in Tables

In order to implement the concepts of the mode naming software in software that will utilize the pitch class data in some useful way, the data must be stored in such a way that it is easy for programmers to incorporate into their existing work. The most obvious way of make this object more useable is to simply output all 7 pitch classes into number boxes.

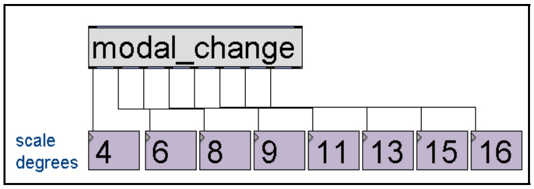


Figure 7. Single object outputting scale data

To take the functionality of this object on step further, when a new mode is entered, the data is stored in a table. The table is Figure 8 shows each of the pitch classes of the E Major at the appropriate table address. As you can see, the first four pitch classes (0-3) are not in use and are at zero.

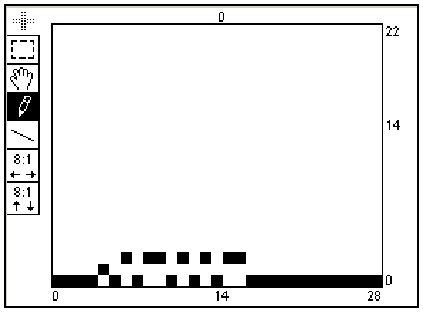


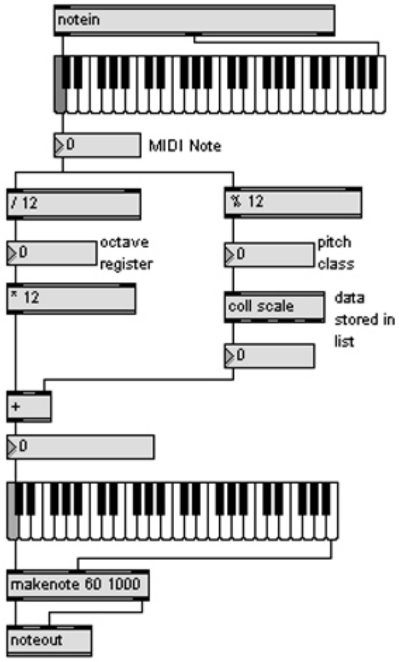
Figure 8. Scale data formatted into a table

In Max, another common way of storing data is in coll lists. By sending the message [send scale0 0] to the table containing the pitch classes, the value of the table at the address 0 will be sent to the receiving object named scale0. Sending similar messages to the table for all of the addresses which might be used to contain pitch class information (0 – 23), we will soon have the information from our table in a list. Table addresses where there is a usable pitch class will return a non-zero integer whereas all addresses where there is not a pitch being used will yield a zero.

Filtering Data from Atonality to Modality

The benefit of using the data in a list is that it can be used to filter route incoming notes that are not from a specified mode, to those from a specified mode. Figure 9 shows a simple Max patch in which the incoming MIDI notes are separated into pitch class and octave register (by using the modulo & divide operators). The pitch classes are sent through the list before being combined with the octave register.

The list the pitch classes are sent through contains information about routing incoming pitches to a new pitch class from the selected mode. The list in Figure 9 shows each pitch class within one octave being routed to the nearest pitch of the E Major scale.



```
0, 1;
1, 1;
2, 1;
3, 3;
4, 4;
5, 4;
6, 6;
7, 8;
8, 8;
9, 9;
10, 11;
11, 11;
```

Figure 9. Modulo operation performed on incoming notes according to filter map (at right)

The pitch C (pitch class 0) becomes C# (pitch class 1) from the E Major scale. In Figure 9, any pitch played will be filtered to one of the nearest notes in the E Major scale. This is accomplished by mapping all address in the table (0-23) and mapping the pitch class to 1 of 24 placeholders. For now, it does not matter that pitch class numbers exceed 11 (one octave). Twelve will be subtracted from all pitch classes above eleven later on in the algorithm. I have named the placeholders “SD#” for scale degree and each of the seven possible scale degree positions.

Figure 10 shows the values of the table being assessed [receive scale4]. If there is activity at a given table address, a 4 will be sent to the list to indicate that E (pitch class 4) is a usable pitch class in the selected mode. If there is no activity of that scale degree in the table a zero will result.

Figure 10 shows all zeros (pitch classes not from the selected mode) being compared to its neighboring scale degrees. A series of “if, then, else” expressions will analyze what the values of the neighboring pitch classes are and choose the one closest to become the resulting pitch class whenever the unusable, non-modal, pitch is encountered.

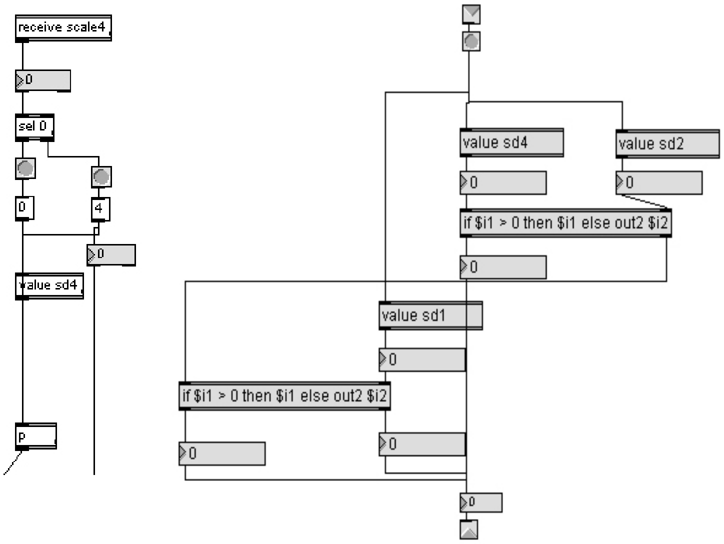


Figure 10. Scale degrees mapped with if then else statements

Both the table and the list update in real-time as soon as a mode is named, making it useful for real-time performance and composition.

Matters of Interfacing

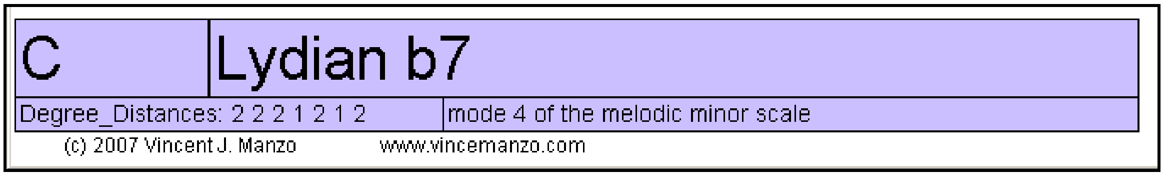
The basic concept of this object and the way it routes data was useful, but was lacking in terms of interface design. How would one access a particular mode quickly; certainly not by manually inputting each of the seven scale degrees.

The second version of the modal change object takes a MIDI pitch class in the left inlet (a pitch letter name message to the inlet also routes to the appropriate pitch class) and determines the tonic of the mode. The right inlet takes mode name messages which trigger the appropriate scale degree distance map to be applied to the tonic scale degree (the fundamental of the calculation).

Version 2.0 of the Modal Change object implemented a slightly new design for the display interface by showing the degree distance patterns for a given mode (Figure 11). The object also takes degree distance patterns in its right inlet allowing you to use any custom 7-note scale as you would the 28 preconfigured modes.

Users familiar with the Max programming environment will understand that mode changes can be easily mapped to user input such as MIDI messages. Even though all of the objects contained in the *Modal Object Library* are intended for use inside of the Max environment, users who wish to create similar algorithms in other programming languages can do so using similar concepts of scale degree distance mapping and matching

Figure 11. Modal_change 2.0 object interface



C | Lydian b7

Degree_Distances: 2 2 2 1 2 1 2 | mode 4 of the melodic minor scale

(c) 2007 Vincent J. Manzo www.vincemanzo.com

Messiaen Modes

Modal Change 2.0 allows a user to add their own degree distance pattern, which effectively enables a user to implement any seven note collection into their compositions. So why stop at seven-note collections? A major contribution to the Modal Object Library followed the release of Modal Change 2.0: 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 12).

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

Figure 12. Modes of Limited Transposition step patterns

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.

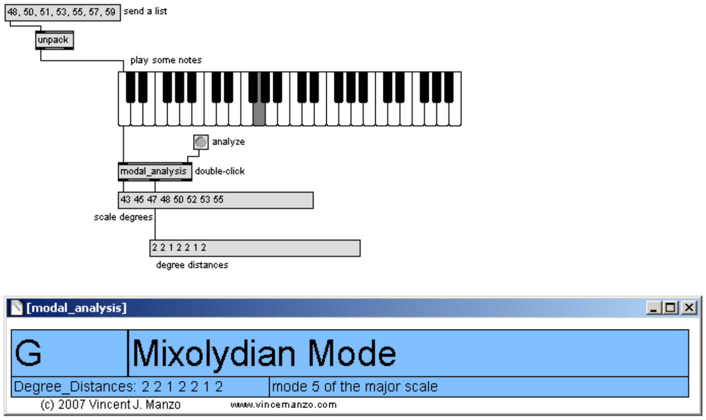
The Modal Messiaen objects also have the same table feature as the Modal Change object which allows their data to be stored in a table in addition to supplying each scale degree of the mode.

Modal Analysis

The Modal Object Library has several objects for Modal Analysis that allow the user to send pitches and have the mode identified. When the analyze button is pressed, the scale degrees are analyzed and a degree distance pattern is matched against a database where the pattern is identified.

The *Modal Analysis* object performs the same actions with some added features: it can take lists or MIDI pitches (not just pitch classes), it can analyze notes in any specified time threshold while filtering out repetitions. This makes it useful for real-time analysis.

Modal Analysis has three outlets which output the pitches of each scale degree in order regardless of the order they were received, the scale degree distance pattern (which can be used to trigger the modal change object), and a bang when the analysis takes place. By using this object with the modal change object, a performer can play something, have it analyzed and have something else played back in response in the same mode. Figure 13 shows the *Modal Analysis* object and the display window.



48, 50, 51, 53, 55, 57, 59

43 45 47 48 50 52 53 55

2 2 1 2 2 1 2

G | Mixolydian Mode

Degree_Distances: 2 2 1 2 2 1 2 | mode 5 of the major scale

(c) 2007 Vincent J. Manzo www.vincemanzo.com

Figure 13. Analysis object

As is the nature with simple analysis algorithms like this one, Modal Analysis has some shortcomings. It takes the lowest pitch as the scale tonic, even though the lowest scale degree played in a time frame does not explicitly mean that it is the tonic of the scale. For example, if you play the pitches C, D, E, F, G, A, B, C, Modal Analysis will call it C Major, but if you play E, F, G, A, B, C, D, E, it will call it E Phrygian. While both of these analyses are correct in identifying the appropriate mode, Modal Analysis 1.0's analyses are contextually independent.

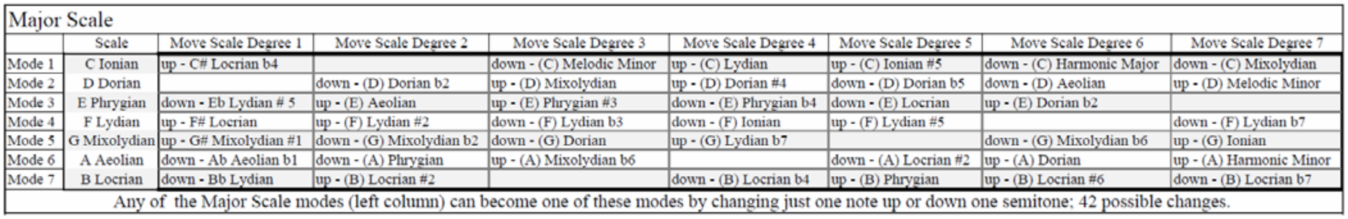
Algorithms for Modal Relationships

The Modal Change object allows us to easily recall the pitch classes of the 28 modes in Figure 4. An understanding of modal relationships, that is, the similarities and differences between two modes, provides a mechanism for implementing the Modal Change object in composition and performance.

Modal Shift

The Modal Shift object takes the specified mode of the Modal Change object and looks at all of the possible modes that are related to the present mode by changing just one note of the 7 note collection. By retaining 6 of the 7 notes of a given mode and creating an tonic emphasis on a given scale degree, the Modal Shift object populates a list of all related modes and modulates to this mode when it receives input in its far left inlet.

In order to create this object, an understanding of the relationships must be established. By taking a seven note scale and shifting each scale degree *either* up or down a semitone yielded either a mode of the major, harmonic minor, melodic minor, or harmonic major scale, or none at all. Figures 14 - 17 shows the complete list of modes that are accessible by shifting just one note and placing tonal emphasis on a given scale degree.

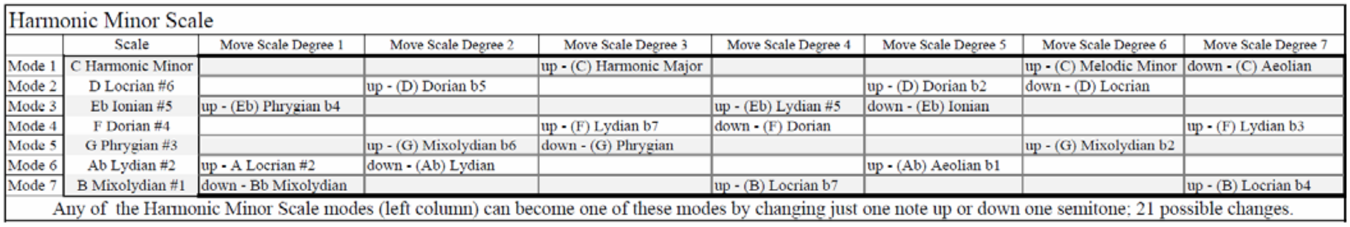


Major Scale

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

Any of the Major Scale modes (left column) can become one of these modes by changing just one note up or down one semitone; 42 possible changes.

Figure 14. Related Modes of the Major Scale; 42 in all

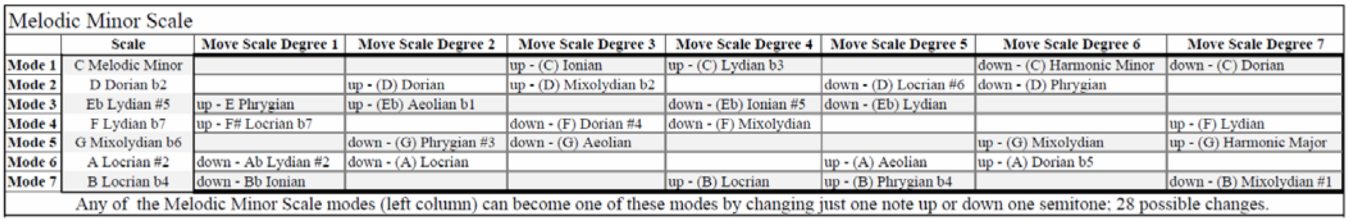


Harmonic Minor Scale

	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) Melodic 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 (left column) can become one of these modes by changing just one note up or down one semitone; 21 possible changes.

Figure 15. Related Modes of the Harmonic Minor Scale; 21 in all

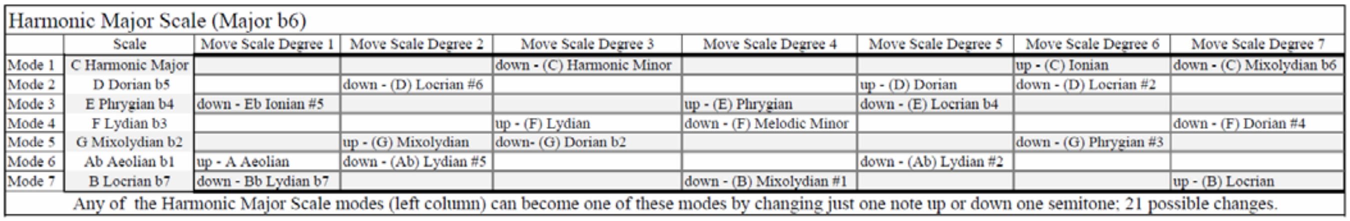


Melodic Minor Scale

	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 (left column) can become one of these modes by changing just one note up or down one semitone; 28 possible changes.

Figure 16. Related Modes of the Melodic Minor Scale; 28 in all



Harmonic Major Scale (Major b6)

	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 (left column) can become one of these modes by changing just one note up or down one semitone; 21 possible changes.

Figure 17. Related Modes of the Harmonic Major Scale (major scale b6); 21 in all

Once these figures were realized, an object was built that takes the mode tonic and mode type information from the Modal Change object and selects a related mode using a 1st Order Markov chain selecting the next mode based on the previous mode.

Switching from *C Ionian* to *C# Locrian b4* is as simple as creating a single operation that both raises the mode tonic one scale degree (by adding one to the tonic pitch class number) and sends a message to the Modal Change object to change to the mode *Locrian b4*. The Modal Shift object implements all of these operations allowing a user to send an input to the objects left inlet to cause a shift to a related mode.

Expansion to Tertian Triadic Harmony

Since we have a collection of objects that can produce scales, we can now form triads from how they occur naturally in modes. The *Modal Triad* object connects to the Modal Change object to allow a user to play tertian triads based on any mode. Modal Triad generates the third and fifth above any scale degree (becomes the root) to make triadic harmony.

Sending the numbers 1-8 to the left inlet of Modal Triad will produce the chord function associated with that number. For example, sending the message “1” in the key of C Major will produce a C Major triad (the I chord in C Major). Sending the message “5” in A Harmonic Minor will produce an E Major triad (the V in A Harmonic Minor). An optional on/off argument toggles between triads in root position or triads in inversions by restricting pitch classes to a single octave. Figure 18 shows an implementation of the Modal Triad object to play back diatonic chords.

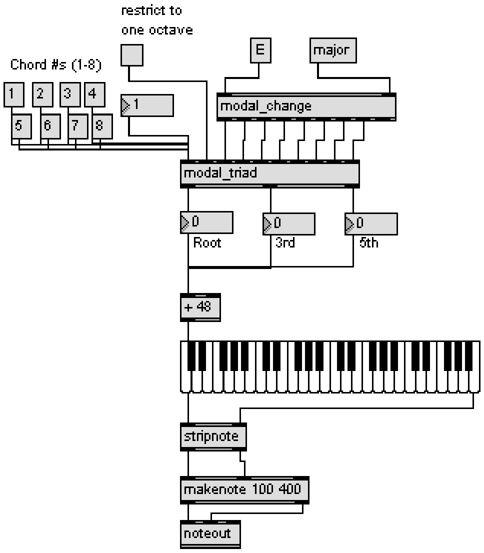


Figure 18. Messages 1 – 8 used to play chords in E Major

5. COMPOSITION EXAMPLES

An example of using the Modal Change object’s coll list feature can be heard in my 2007 work for guitar and software *Optical String*[i]. The software used is a program I wrote in 2006 called *Lazy Guy* which tracks the color of a laser pointer and maps it’s X Y orientation to produce pitch and velocity.

As is the case with many new electronic instruments for musical expression, implementing a sense of idiomatic traits, especially with regard to tonality, is often difficult and, more so, overlooked. An implementation of the Modal Change object allows *Lazy Guy* to easily conform to any key/mode in real-time. This creates a set of idiomatic performance gestures that conform comfortably to a user-defined note collection.

The first formal piece I composed using the Modal Change object was *Modal Change* (2007) for string orchestra, harp and piano. Using only the pitch table from the Modal Change object and some patches to create rhythm, I created a Max patch that assigned an octave and register to multiple “pitch engines” that created the score for each of the instruments using the table data.

I wrote a patch to control the overall textural design and confined the pitch engines to producing small, minimalist gestures in equal phrases. My harmonic intention was to have the patch begin with a single 7-note collection (C Major) and every 16 measures or so, have the piece choose another mode closely related to C Major by changing only one scale degree. For examples, the C major scale can easily become C Lydian by changing just one note. C Lydian can then become C Lydian b7.

The changes in mode (which I refer to as a *mode row)* continue in this way until all 28 modes are used. The Max patch then cadences and stops. The idea of related modes by changing just one scale degree (which easily allows common-tone modulation) became the basis of the second object in the Modal Object Library: Modal Shift

Monochrome (2007) is an interactive music system that generates pitch material with data received through a graphics tablet (such as a Wacom). A graphic artist begins drawing a picture on the tablet and pitch is generated based on the X, Y orientation of the pen. Velocity is determined by pen pressure against the graphics tablet.

In *Monochrome*, an initial mode of C Major is set at the beginning and all pitches are filtered to the notes of C Major using the Modal Change objects list filter using MIDI channel 1. The activity taking place on the canvas is mapped to 16 equal sections. When one of these sections becomes more than 50% filled, the MIDI data is sent out a subsequent MIDI channel in addition to the previous channel and a bang is sent to the modal shift object triggering a new, related mode.

Evolutio (2007) is an algorithmic composition for string quartet. It is similar in design to the piece *Modal Change* (2007), in that it uses modulates fairly regularly via the Modal Mutation object. It is, however, quite different: where *Modal Change* used minimalist gestures in a large ensemble, *Evolutio* functions more like a traditional string quartet.

With the Modal Prog object, a chord progression was assigned to the pitch engines (string quartet). Using the Modal Triad object, each voice in the string quartet played notes from the triad specified by the Modal Prog object. Each voice chose other notes from the assigned mode as non-chord tones between the beats and chord progressions. This allowed each voice to use passing tones on weak beats to get from note to note.

6. CONCLUSION

The Modal Object Library has been used in numerous compositions and interactive installations including EAMIR [5] and more. For more information and to download the library, visit www.vjmanzo.com/mol .

7. REFERENCES

[1] Modal Object Library, available at http://www.vjmanzo.com/mol.

[2] Max - Cycling 74, available at http://cycling74.com/products/maxmspjitter.

[3] Pure Data - Pd Community Site, available at http://puredata.info.

[4] OpenMusic, available at http://repmus.ircam.fr/openmusic/home.

[5] EAMIR, available at http://www.eamir.org

[6] R. L. Dubois. Applications of Generative String-Substitution Systems in Computer Music. Doctoral dissertation, Columbia University, New York, NY, 2003.