Tutankhaten – a variable composition by a JMSL generative algorithm

Allen Fogelsanger[1]

December 2011, revised July 2012

Abstract: *Tutankhaten* is a variable composition created in JMSL (Java Music Specification Language). On each run of *Tutankhaten* the computer creates and plays a new piece from randomly generated musical materials consisting primarily of short melodic fragments and chords. The music, inspired by Morton Feldman's composition for solo piano *Palais de Mari* (1986), consists of thirteen sections in each of which a single melodic fragment, chord, or chord pair is repeatedly presented with a small variation from the preceding occurrence.

Over the last ten years of his life Morton Feldman composed a number of pieces for solo piano[2] and for small chamber groups including piano.[3] These works mostly last an hour or more and are to be played largely at a dynamic level of piano or softer; they are characterized by sequences of repetitions of short cells of music laid together in a mosaic-like structure.[4] *Tutankhaten*[5] is an attempt to capture some of Feldman's compositional techniques in a generative algorithm. It is programmed in JMSL (Java Music Specification Language)[6] and built within an applet framework written by Nick Didkovsky. *Tutankhaten* references in particular *Palais de Mari* (1986)[7] for solo piano, the title of which was taken from a photograph of ancient ruins in southeast Asia that Feldman saw in the Louvre. The title *Tutankhaten* (a name of the Egyptian pharaoh whose nearly intact tomb was discovered in 1922) was chosen (1) as a reflection of the ruin and ancientness accessed by Feldman's title; (2) as a reference to the Egyptian desert, which seems an apt image to associate with the starkness of the generated music; (3) as a play on the "Toot" in the name of Didkovsky's original applet, called

[1] New York University, Department of Music and Performing Arts Professions, allen.fogelsanger@gmail.com, http://www.armadillodanceproject.com/AF/.
[2] *Piano* (1977), *Triadic Memories* (1981), *For Bunita Marcus* (1985), *Palais de Mari* (1986).
[3] *Why Patterns?* (1978) for flute, piano and percussion; *Trio* (1980) for violin, cello and piano; *Patterns in a Chromatic Field* (1981) for cello and piano; *For John Cage* (1982) for violin and piano; *Crippled Symmetry* (1983) for flute, piano and percussion; *For Philip Guston* (1984) for flute, piano and percussion; *Piano and String Quartet* (1985) for piano and string quartet; *For Christian Wolff* (1986) for flute and piano; *Piano, Violin, Viola, Cello* (1987) for piano, violin, viola and cello.
[4] Analyses include Ames (1996), Johnson (2002), York (1996).
[5] A working version is online at http://files.nyu.edu/alf431/public/JavaMusicSystems/TutankhatenPageRevision.htm.
[6] http://www.algomusic.com/jmsl/
[7] The 1986 edition of the score (Feldman, 1986) is a facsimile of the handwritten score. The 1995 printed edition is much easier to read. Commercial recordings include Feinberg (1996), Ginsburgh (2001) and Schroeder (1990).

"JScoreTootJavaSoundMIDI";[8] and (4) as a play on the "toot" in "tutorial," the applet's function.

Tutankhaten generates a new composition each time it is loaded. It randomly (with some guidance) chooses six sets of notes: **x**, **y**, **z**, **A**, **B** and **C**. The lowercase-labeled sets are called *linear cells* and are usually presented note-by-note; the uppercase-labeled sets are called *chordal cells* and are usually presented as chords. The linear sets are made up of two to four notes with an order and a rhythm, while the chords have two to six notes. Notes are chosen to minimize the appearance of octaves and unisons, major and minor thirds, and major and minor sixths; while two-note pairs will not have thirds or sixths, it is possible these intervals will appear between non-successive notes in longer linear cells. Likewise the original chords will not have thirds or sixths as intervals between adjacent notes, but if the chord is varied through the octave transposition of one of its notes then a third or sixth may appear. A third type of organizational set is the *chordal pair*, which consists simply of two chords played in an order.

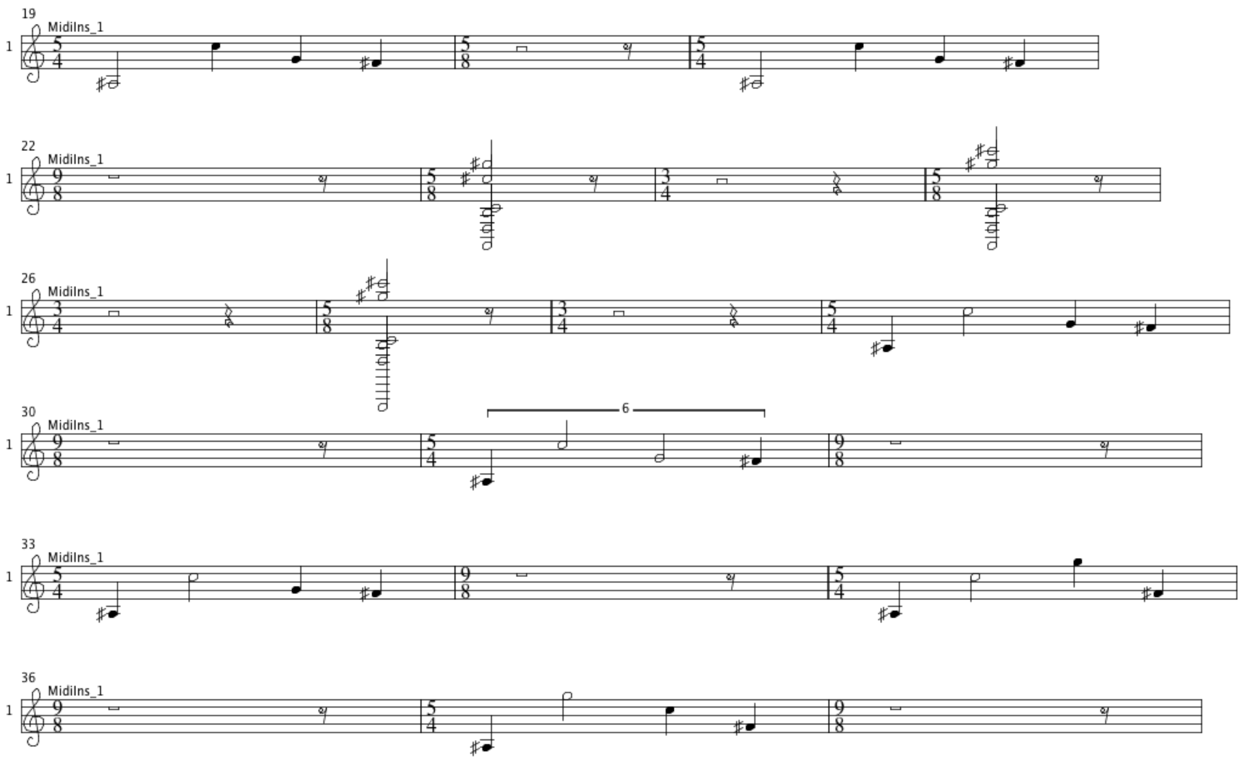
The note sets are presented in 13 sections as follows:

Section	**Material**	**Type of Cells**	**Variation**
0	**x**	linear	varied 11-24 times
1	**A**	chordal	varied 2-3 times
2	**x** or **y**	linear	varied 4-9 times
3	**A**	chordal	varied 2-3 times
4	**x** or **y**, with **B**	chordal pair	varied 11-24 times
5	**x** or **y** or **z**	linear	varied 4-9 times
6	**A** or **C**	chordal	varied 2-3 times
7	**z**, with **C** or **A**	chordal pair	varied 6-10 times
8	**z**	linear	varied 11-24 times
9	**A** or **C**	chordal	varied 1-3 times
10	**B** or **y** or **z**	linear	varied 5-10 times
11	**x**	linear	varied 1-3 times
12	**A** with **x**	chordal pair	varied 11-24 times

The choice of material in sections 2, 4, 5, 6, 7, 9 and 10 is made randomly, as is the number of variations within the given range. Because of the return of **A** and **x** at the end of a performance, there is a vaguely ternary structure. *Tutankhaten* generates

[8] http://www.algomusic.com/jmsl/tutorial/JScoreTootJavaSoundMIDI.txt

approximately two hundred measures with time signatures chosen from 5/8, 3/4, 7/8, 4/4, 9/8 and 5/4 to be played at 60 bpm. Playing times tend to be from ten to fifteen minutes. Scores are realized by a general MIDI piano with sustain pedal on throughout and all notes played at the same quiet volume.



The primary focus moment-to-moment is on the ongoing variation that occurs with each repetition of a cell. While a linear cell may be transformed into a chordal cell and vice versa from one section to another (for instance the linear set **z** appears as a chord in a chordal pair in section 8, and the chordal set **B** may appear as a linear cell in section 10), within a section linear cells, chordal cells, and chordal pairs remain linear cells, chordal cells, and chordal pairs respectively. The variation in successive instances is relatively subtle and requires close listening. It can consist of changing the octave of a single pitch or making a tiny adjustment to a duration. Linear cells may have pitches switched, rhythms altered, or speed adjusted; successive notes may change order. The time between successive cells changes. The variation is recursive so that cells, especially linear cells, can eventually be considerably transformed, though they retain the same pitch-class set. If a cell returns in a later section it sometimes re-initializes in original form and sometimes in the form in which it was last heard.

Users may reload the algorithm to generate another realization of *Tutankhaten*. The method of choosing pitches, the method of variation, and the structure of the composition remain the same and guarantee a certain quality of sound and pacing, but the unique courses followed by the scores lead to a variety of nuances in character and feeling.

Technical Description

The following description refers to the source code for *Tutankhaten*, consisting of the files Tutankhaten.java, MariLinearCell.java and MariChordalCell.java.[9]

Linear cells and chordal cells are instances of the classes MariLinearCell and MariChordalCell respectively (their names referring to *Palais de Mari*). Each class includes fields for an array of MIDI pitch values, an initial measure length and a blank measure length. The two measure lengths are in eighth beats, the first indicating the time signature of the measure in which the notes are presented and the second indicating the time signature of an empty measure following the initial measure. The MariLinearCell class additionally includes a field for an array of duration values indicating the rhythm of the cell; the MariChordalCell class instead has a field for a flag showing whether or not its blank measure is added to the score.

The three linear cells and three chordal cells comprising the seeds of the piece are each placed in two buffers: one which is unchanged throughout the composing process and one which always holds the most recent variation of the original material. The original linear cells are generated randomly as follows. Both the initial measure length and the blank measure length may be from 5 to 10 eighth beats[10] and each is chosen randomly (uniform distribution) from among the six options. Cells **x** and **z** are given four notes but cell **y** is randomly set to have either two or three notes. The notes are ordered and to begin with the pitch value of the first note of a cell is set at 0. Subsequent notes are chosen based on their interval mod 12 with the preceding note.

[9] Online respectively at http://files.nyu.edu/alf431/public/JavaMusicSystems/Tutankhaten.java, http://files.nyu.edu/alf431/public/JavaMusicSystems/MariLinearCell.java and http://files.nyu.edu/alf431/public/JavaMusicSystems/MariChordalCell.java.
[10] Even-beat measures will be notated in quarter-beat time.

The first interval mod 12 is randomly picked from [1,2,5,6,7,10,11] and the second note is randomly chosen from the two or three appropriate notes that lie at most a major ninth up or down from the first note. For example, if the first interval mod 12 is 1, given that the first note is 0 the second note may be -11, 1, or 13. If there are more than two notes the interval choices are limited so that no unisons or octaves occur between the notes. A typical four-note linear cell melody at this stage might be [0,-11,-9,2]. It is then transposed to lie at the bottom of the piano MIDI range [21,108] (in this case by adding 32 to move it to [32,21,23,34]) and then further transposed by adding an integer *i* generated by the method centralChoose(span,3) that tends to place the cell near the middle of the keyboard. The method centralChoose(*n*,*k*) randomly chooses *k* times from [0,*n*) and averages the results; in this case *n* (the "span") is the number of positions which the melody could occupy on the keyboard and *k* is set to 3 for a moderate centralizing tendency courtesy of the Central Limit Theorem. The example cell could fit on the piano anywhere from [21,34] to [95,108], thus having 75 positions it could occupy; the method centralChoose(75,3) is employed to choose a number tending toward the center of the range [0,75) that is then added to the linear cell melody. If the method generates say *i* = 37, then the melody is transposed to [69,58,60,71].

Rhythms are represented as arrays of positive integers whose sum (the *rhythm sum*) is randomly chosen from the interval [5,10]. The integers are conceived of as standing for eighth note values, so that the rhythm [1,3,4,2] would indicate an eighth note, dotted quarter note, and half note, followed by quarter note.[11] Given the rhythm sum *n* and the length *k* of the rhythm array, the method randomCompose(*n*,*k*) randomly (uniform distribution) chooses among the (*n*-1 choose *k*-1) *compositions* of the positive integer *n* that have *k parts*[12] to generate an initial rhythm. It is important to note that the

[11] Rhythm durations of more than 4 are notated as an even duration note with a 1 or 2 duration rest, e.g. a rhythm duration of 5 is notated as a note of duration 4 and a rest of duration 1, and a rhythm duration of 6 is notated as a note of duration 4 and a rest of duration 2. This is done to avoid using tied notes, which are unnecessary with the sustain pedal down.

[12] The *compositions* of a positive integer *n* are the ordered sums of positive integers that equal *n*, with each positive integer called a *part* of the composition. For example the compositions of 4 are 1+1+1+1, 2+1+1, 1+2+1, 1+1+2, 2+2, 3+1, 1+3, and 4; and the composition 3+1 has 2 parts, namely 3 and 1. The wikipedia article on compositions of a sum, http://en.wikipedia.org/wiki/Composition_(number_theory), includes a very elegant proof that the number of compositions of *n* is 2^(*n*-1): each composition of *n*

rhythm sum will usually not be the same as the initial measure length; one of the ideas inspired by Feldman's notation is that of placing, say, a rhythm consisting of nine eighth notes (for instance [1,2,2,4]) in a 7/8 measure.

The initial chordal cells have their initial measure lengths and blank measure lengths[13] chosen in the same way as those of the initial linear cells, but their pitches are chosen somewhat differently. The chord's first (lowest) note is set to 0 and each succeeding note is stacked above its predecessor at an interval chosen randomly from [1,2,5,6,7,10,11,13,14]. Any octaves that result are then stripped from the chord by removing the lowest note(s) of the octave. For example a pitch sequence of [0,7,12,19,24] could be initially produced, but it is then reduced to simply [19,24]. This means thirds and sixths between adjacent tones will occasionally appear, for instance through [0,2,4,9,14] being reduced to [0,4,9,14]. The blank measure flags of the initial chordal cells are set to true.

Once the beginning linear and chordal cells are chosen, each section of the piece takes one or two of them through repeated recursive variation.[14] Which cells are varied depends on the section and will be covered below; here is discussed the methods of variation, of which there are three: variation of a linear cell, variation of a chordal cell, and variation of a chordal pair.

A linear cell with three or four notes is equally likely to be varied in any of six ways. The initial measure length may be changed up or down by a minimum of two eighths, e.g. 4/4 may become 5/4. The blank measure length may be changed likewise. One of the notes may be transposed up or down one octave. The order of two successive pitches may be reversed (with the rhythm of the cell remaining unchanged).

---

corresponds to a sequence of *n* ones interleaved with a sequence of *n*-1 commas and pluses, e.g. the composition 2+1+1 corresponds to 1+1,1,1. There are 2^(*n*-1) such sequences of pluses and commas; for a given sequence the number of parts equals the number of commas plus one. The method randomCompose(*n*,*k*) looks at all the binary representations of the integers 0 to 2^(*n*-1)-1 as if the zeros correspond to pluses and the ones correspond to commas. It finds the binary representations with *k* parts (i.e. with *k*-1 ones corresponding to *k*-1 commas), and randomly (uniform distribution) chooses among them.

[13] In 5/8, 7/8 and 9/8 measures chords are notated as lasting 2, 3 and 4 quarter notes respectively and are followed by an eighth rest. In 3/4, 4/4 and 5/4 measures they last 2, 3 and 4 quarter notes respectively and are followed by a quarter rest. This is done to avoid using tied notes, which are unnecessary with the sustain pedal down.

[14] In this respect *Tutankhaten* strays from Feldman's example. Feldman often uses immediate *exact* repetition—something *Tutankhaten* does not do.

One of the notes may be reduced in duration by one eighth in tandem with another note duration being increased by one eighth. Finally the rhythm sum may be increased or decreased by one eighth through increasing or decreasing the duration of one of the notes by one eighth.

A linear cell with two notes has an extra option and the chances of each of the seven options are not equal. There is a 20% chance the entire cell will be transposed up or down one octave, a 20% chance that just one of the notes will be transposed up or down one octave, and a 10% chance that the pitches will reverse in order (and trade durations so that the rhythm remains unchanged). The four other options (initial measure length variation, blank measure length variation, rhythm sum variation, rhythm variation with same sum) each have 1/8 chance of occurring.

A chordal cell can vary in fewer ways. There is a 49% chance one of its pitches will move up or down an octave. Because the sustain pedal is down the other three options (each equally likely) all simply change the time from the onset of the chord to the onset of the next chord or note: the blank measure flag may be reversed; the initial measure length may vary in the same way as for a linear cell; and the blank measure length, regardless of whether the blank measure flag is true or false, may be varied as for a linear cell—if the flag is false it will be set to true.

The third unit available for variation, the chordal pair, is created by placing two chordal cells together in a chordal cell array and turning off their blank measure flags. The chordal pair may be varied in four equally likely ways: one pitch of the first chord may be transposed up or down one octave; one pitch in the second chord may be transposed up or down one octave; the measure length of the first chord may vary in the same way as for the initial measure length of a chordal cell; and the measure length of the second chord may vary likewise.

Each section of *Tutankhaten* consists of a number of instances of one type of unit going through variation. Section 0 adds MariLinearCell **x** to the score (an initial measure with notes and a blank measure with rests) and then randomly chooses to add from ten to twenty-three more instances, each a variation of its predecessor. Likewise Section 1 adds MariChordalCell **A** to the score (with both an initial and a blank measure) and randomly adds one or two variations, the last of which is forced to include its blank measure (which may result in an exact repetition).

Section 2 begins by randomly choosing to vary either MariLinearCell **x** or MariLinearCell **y**. In the first case the varying will pick up where it left off at the end of section 0, adding from four to nine variations recursively applied to the last variation of **x** in section 0. In the second case MariLinearCell **y** is added to the score followed by four to nine variations applied successively. Section 3 brings back MariChordalCell **A**, adding two or three successive variations of the last variation from Section 1, the last of which again is forced to include a blank measure.

Section 4 brings up a new technique, the conversion of a linear cell into a chordal one. The chordal version retains the same initial and blank measures and has a true blank measure flag like any new chordal cell. The ordered pitches of the linear cell may no longer be in original form—their order may have changed as well as their octave. The intervals between them are reduced by modulo 12 to within [1,12); then each interval of size 1 and 2 is either left unchanged or increased by 12. The first note is transposed whatever number of octaves is necessary to place it in [0,12) and the new intervals are added successively to it to produce a new set of increasing pitches for a chord. The method centralChoose(span,3) is used to transpose this new pitch material onto the piano, with some tendency to favor the middle of the instrument; the result is further transposed by the smallest interval possible to regain the pitch classes of the original linear cell.

Similarly there is a method to convert chordal cells to linear ones. Again the new version retains the same initial and blank measures. Only the first four pitches (or fewer if that is all the chord has) are used to form a melody—and these are not necessarily the lowest notes in the current chordal cell: they will instead be the first four notes originally assigned to the initial version of the chordal cell before it underwent any variation. The first pitch is transposed to within [0,12) and the intervals between successive pitches (in the chordal cell pitch array, not in the current chord itself) are transposed to within [0,12). Interval values of 1 are then replaced randomly by -11, 1 or 13; likewise a 2 is replaced randomly by -10, 2 or 14. Similarly intervals of 10 and 11 are transposed 0, 1 or 2 octaves lower. Interval values between 2 and 10 (including the rare 3, 4, 8 or 9 that may appear through octave removal) either remain unchanged or drop 12. These new intervals are added to the first pitch to create the basic melody, which is, like the melody converted to a chord, transposed to within the piano range using the method centralChoose(span,3) and then further transposed to the nearest piano position that matches its pitch classes with the original pitch classes of the chord. A rhythm sum is chosen randomly from [5,11) and a rhythm composition is randomly chosen from among the possibilities for the chosen sum.

Section 4 adds ten to twenty-three chordal pairs to the score. The first chord in the pair is a conversion of either the current variation of the **x** linear cell or the current variation of the **y** linear cell. The second chord is the original chordal cell **B**. In either case the first half of the chordal pair is added at the end, with its blank measure turned back on.

Section 5 adds four to nine variations of the current state of one of the original three linear cells (**x**, **y** or **z**). Section 6 adds two or three variations of the current state of one of the chordal cells **A** or **C**, with the blank measure forced on in the last variation. Section 7 pairs the chordal conversion of the original linear cell **z** with the original version of whichever of **A** or **C** was not used in section 6; the resulting chordal pair is added with variation four to nine times with an extra half pair at the end as in section 4. Section 8 continues the use of **z** material but now as a linear cell starting with the original version and repeating ten to twenty-three times. Section 9 brings back material from either section 6 or 7 by presenting two or three chordal cell variations of either material **A** or material **C**, forcing the last to include a blank measure.

Section 10 adds four to nine linear cells starting with either a linear conversion of the original chordal cell **B** (in which case the conversion precedes the variations) or with the current version of either linear cell **y** or linear cell **z**. Section 11 recapitulates the opening of *Tutankhaten* by bringing back the original version of linear cell **x** followed by zero to two variations. The work finishes with section 12 adding ten to twenty-three variations of the chordal pair formed by chordal cell **A** and the chordal conversion of linear cell **x** (with no half pair at the end). *Tutankhaten* thus finishes working on the same two pitch sets it started with, but in alternation.

Discussion

The most successful aspect of *Tutankhaten* is its varying of material bit by bit. The tiny changes have the same effect as those in Feldman’s music, focusing listening on the slightest variation in rhythm and pitch. With regard to rhythmic variation *Tutankhaten* is arguably more extensive than any of Feldman’s pieces because it presents more subtle changes more often. The independence of the rhythm sum from the time signature contributes to a sense that the linear cells “float” and “slide” over the constant 120 eighth beats per minute tempo. The variation in the length of the blank measure is also important, as it makes one never sure exactly when the next cell will begin. The chordal cell variation is not as interesting as the linear cell variation, simply because there is no rhythm to vary; the implementation of Feldman’s chordal pair technique makes up for this, because in pairs the durations of the two chords play off of each other to form an irregular gait that drunkenly walks from one iteration to the next.

The pitch variation of both melodies and chord pairs also works very well, enough so that it was unnecessary to include in this version of *Tutankhaten* methods that replace one of a cell’s old pitch classes with a fresh one. For the linear cells simple single pitch octave transposition is enough to keep one’s attention, and a particular fascinating effect occurs when pitches range far enough away from each other that they cease being heard as a single melody but by gestalt processes are heard instead as two lines in counterpoint. On the other hand, with chords octave displacement results in minute timbral alterations that require a focus on the music as concentrated as that needed to follow the rhythmic changes.

While the variation methods work very well, the overall structuring of the piece will be the main reconsideration for future versions. Section changes are always rather abrupt, and options that allow for more gradual transitions would be useful. Here is where a note-by-note replacement method, applied to consecutive (or perhaps not-quite-consecutive) variations, might prove especially useful. Another problem is the lack of variety (though there is a certain satisfying austerity to the present algorithm); additional units besides linear cells, chordal cells and chordal pairs would profitably expand the rather limited vocabulary of *Tutankhaten*. Triples of objects, combinations of linear and chordal cells, and other sorts of objects altogether would helpfully enlarge *Tutankhaten*'s palette.

While a near term goal is to add to the flexibility of *Tutankhaten* in showing how a single solo sound without dynamic change of any kind can yet prove compelling, a longer term goal is to address the challenge of using the same techniques to write for different sounds, for combinations of sounds, and for dynamically changing sounds. A further question to be explored is whether *Tutankhaten*'s methods of variation and compositional structuring can successfully work with other styles of music, especially ones less "environmental" in character and not so near to disappearing into the genre of background music.

The Java/JMSL/JScore package is perfect for implementing Feldman's measure-based cell structure, for retaining original and current versions of cells, for making choices among various options randomly, and simply for composing a piece rather than building a process. In these ways it has advantages over Max/MSP/Jitter[15], which, although it provides a visual graphic interface that is perhaps more inspiring than a code-writing environment, does not encourage the realization of score-based algorithms. Of course the fact that Java is platform independent makes its functionality ubiquitous, so that patches created in it seem to be more easily distributable than those made in Max. Finally Java's online user interactivity makes the idea of adding user-chosen options to *Tutankhaten* extremely attractive.

[15] http://cycling74.com/whatismax/.

Bibliography

Ames, Paula Kopstick. 1996. Piano. In Thomas DeLio (ed.) *The Music of Morton Feldman*. New York: Excelsior Music Publishing Company, 99-143.

Feinberg, Alan. 1996. *Wuorinen: Third Piano Sonata; Bagatelle; Capriccio. Feldman: Palais de Mari.* Koch International Classics 3-7308-2.

Feldman, Morton. 1986. *Palais de Mari*. London: Universal Edition. Reprinted 1995 (Universal Edition ue 30 238).

Ginsburgh, Stephanie. 2001. *Last Pieces*. Sub Rosa SR189.

Johnson, Steven. 2002. Jasper Johns and Morton Feldman: What Patterns? In Steven Johnson (ed.) *The New York Schools of Music and Visual Arts: John Cage, Morton Feldman, Edgard Varèse, Willem de Kooning, Jasper Johns, Robert Rauschenberg*. New York: Routledge, 217-247.

Schroeder, Marianne. 1990. *Piano*. hat ART CD 6035.

York, Wes. 1996. For John Cage. In Thomas DeLio (ed.) *The Music of Morton Feldman*. New York: Excelsior Music Publishing Company, 147-195.