
Larry Polansky, Dartmouth College, Music Dept., Hanover, NH 03755
e-mail: larry.polansky@mac.dartmouth.edu
Nick Didkovsky, 171 E. 99th St #20, NYC NY 10029

Abstract: This paper describes several pieces written and performed by the authors, individually and together, in the computer music language HMSL between 1984-1991. These pieces explore different aspects of performer interaction with computer software, aspects of human-machine improvisation, and aesthetic issues in live computer music.

Introduction
This article documents works for live interactive computer and performers written in the computer music language HMSL between the years 1984 and 1991. The works are a small sample of the experiments, compositions and performances done in HMSL. For more descriptions of HMSL see Polansky, Burd, and Rosenboom [1987, 1990]; Polansky and Rosenboom [1985]; Riddle [1989]; Scholz [1990, 1988, 1988a], Didkovsky (1990b).

Aspects of our work in HMSL
Interaction: Each piece explores interactivity in a different way. The performer(s) have a specific, new and unusual task, or form of interaction with the making of music, in communicating with the machine and other performers.

Intelligence: Each piece approaches composition and creativity in a slightly different way, exploring notions of determinacy and indeterminacy, the co-involvement of the performer and the machine with the compositional process, and in many cases, the deliberate abdication of certain compositional decisions and determinations of musical parameters to the machine.

Development of HMSL and a community of composition: One of our interests was to develop code and ideas that would be useful to and interactive with the work of other composers. In most cases, the code for these pieces was made available to the large community of HMSL users through BBS’s, networks, and through copying disks.

Low overhead: All of the pieces involve a minimal, inexpensive and portable computer music system, in which the success and musical intent of the piece is not determined by the expense of the hardware. The pieces needed to travel (most of them have been performed internationally). We were interested in deemphasizing the relationship between consumerism and music, a natural result of MIDI; the interest of a piece shouldn’t be related to the economic means of a composer (something Ron Keulvila has jokingly and aptly termed “vitriolic consumerism”). The challenge was to make interesting pieces with inexpensive, flexible, and sometimes rather humble-sounding equipment.

Pieces: by Larry Polansky

\( \text{Cantillation Studies 1} \) (In the beginning... ) (Cantillation Study #1) (1984; revised 1987, 1989). This was one of the first pieces written in HMSL by me or anyone. It was written for Jody Diamond, with whom I had studied cantillation of the Torah, and is the first of a set called the Cantillation Studies, based on computer-aided morphological transformations of the 11th and 12th century Masoretic cantillation melodies (tropes). Each piece is based on successive 17-verse sections of the Torah, and named, in traditional manner, after the first few words. The Torah tropes are used as a basis for melodic transformation by the computer.

In B’rey Sheer, the tropes are sung, unadorned. The pitch of Jody’s voice is captured by HMSL by a Pitch-To-MIDI converter, using a significant amount of software de-bouncing of incoming pitches. Jody sings in a just intoned scale I selected for the highly modal trope.

The text is the first section of the Torah — describing creation, in which order is brought to the various cosmological parameters. With each of the 17 verses of the Torah section, the statistics of the computer’s musical response are constrained. In this way, the computer “listens” to the melodies and generates its own events based on what it hears, and on where it is in the piece. There is a predefined trajectory of “computer attention,” achieved by constraining the degree of randomness, which specifies that.
at the beginning of the work the computer mostly ignores the voice, but gradually and continuously increases its attention until the end, when follows the voice closely. The piece stands out in accompanimental gestural chaos in all of its parameters, and ends in unison with the voice. One variable controls the depth of all change in the piece, beginning high (17) and ending low (0). A second variable (usually me) changes the value of the variable (essentially saying where the computer is in the piece) through a simple HMSL graphic interface.

By anticipating this change in relation to the singer, the mood and macro-rhythm of the performance can be varied tremendously — Jody and I have found through many performances that even though both tasks are highly constrained (the singer sings the trope, I change a few variables more or less at specified times), the piece sounds as if the voice feels very different depending on our respective timings. Our growing sensitivity to these details make performing the piece enjoyable, exciting, and unpredictable.

The computer's sonic material is limited to four sine waves — a limitation accepted from my decision to use only Amiga sound. The Amiga has four b-bit D/A sound channels, which although of relatively low fidelity, are highly flexible in terms of timbre and timbre. Since a separate processor is responsible for updating the DAC outputs from a specified memory location, high-level software can change the output waveform transparently and quickly without interrupting the sound. In Brey's piece, I was interested in dynamic, point-by-point time domain waveform modifications that were "orthogonal" to the kinds of higher level morphological transforms that HMSL was designed for. The sines tables are treated as long melodies, and the waveable modulations have the following parameters: amount to wave (A), level to be modulated, degree, possibility and type of modulation. The modulation techniques are derived from the methods of the shape class in HMSL: simple ways of editing sine of musical data. For example, in Brey's piece, particular points in the sine table can be replaced with other points (a kind of spacial deformation), or portions of the table can be regraduated, inverted, scrambled, randomized, and so on. The degree of modulation is determined by location in the piece. Typically, at the beginning of the piece, all four sine tables are modulated at a rate, as is the type of modulation itself. This modulation calms down both in rate and degree as the piece progresses.

One of the most interesting aspects of Brey's piece is that the computer "sings" "on the fly," making use of a simple interactional trajectory which guides the tuning decisions. This trajectory begins in a complex 17-limit tuning space, and ends in a kind of simple 3-limit one (or Pythagorean tuning). The current vector number serves as the limit for the tuning space, and random tuning values whose prime factor less than that number are chosen for numerators and denominators of just intervals to the just intervals of the melody itself. For example, in the first verse (17-limit), the computer might use 5/150 (a small major second to the 8/7 using prime factors 17, 5, 3, and 2) to harmonize the septimal major second in the melody (8/7), resulting in an absolute "minor-third" interval of 204/175 (approx. 206.5 cents). This particular pitch might only last a fraction of a second, since in the beginning of the piece everything is changing quite rapidly. The tuning process happens simultaneously in all four voices. The computer is always in harmony with the voice, but the nature of the harmonic space is highly stochastic, and has no real functionality.

Specif (tor) the most interesting aspect of what I have called parametric tuning (Polansky 1987b,c) — all of the tuning is done in real-time, in response to the input. No concept of scale, or range, is ever invoked. The pitches used are not chosen from a set of pitches, but generated by the machine in real-time, simulating the ad libi ad vocum or stringing quartet would dynamically wrap itself over the course of a piece, except that in Brey's piece the rules are extremely primitive! A natural extension to the algorithms would impose some sort of voice leading or functionality rules upon this tuning system, but this was not part of my aesthetic intent in this piece.

Two Portable Pieces

17 Simple Melodies of the Same Length

17 Simple Melodies... (Polansky 1990b, 1988b) was written for composer/performer Daniel Goode in 1987. It was intended to be a truly portable computer music piece which was more than simply a sequence of note information — offering a live, interactive, and flexible piece for any perform/ T_Performer who could input MIDI pitch data. This was motivated to some extent by the fact that when I wrote it, many more extraordinary and experimental performers (like Daniel Goode, John Oswald, Ann LaBerge, George Brooks, and others who have performed this piece) were starting to work with MIDI equipment in sophisticated ways, programming their own sounds, using various input devices and so on. From a software standpoint, what tended to be available to these artists was, in general, commercially oriented, offering limited
experimental resources for composition and performance. I intended 17 Simple Melodies... as a kind of example of what could be done in designing unusual, interactive, and intelligent pieces for computer and performers of conventional instruments. By only supplying a musical form for the piece as a kind of "black box" software engine, leaving the timbres of the MIDI synthesizers as well as all the melodic material completely up to the performer, I was inviting what I hoped was an evolutionary and expansive collaboration.

From its inception, 17 Simple Melodies... was distributed to the performers as a disk with instructions. The disk contained a fully executable version of HMSL with the code for the piece compiled as well. In this way, the score for the piece was the code itself. Source code is also supplied with the piece, and performers are encouraged to modify it, or use it as a model for their own pieces, which would be some sort of "collaboration" with me.

The form of 17 Simple Melodies... is a kind of good-natured parody of "classic" artificial intelligence: data is gathered by a kind of perceptron (17 melodies of 17 notes each), the data is sorted (the intelligence), and finally, the data is re-played as a kind of simulation. In Section 1, the performer plays 17 melodies, signalling the computer through the ASCII keyboard when she is ready to play the next melody (in one performance, John Oswald put the Macintosh keyboard on the floor and made these signals with his foot!). The computer informs the performer when it thinks it has heard 17 notes. Glissandi, rapid arpeggios, multiphonics, noises, and so on will confuse it, usually in an interesting way.

Section 2 is silent, lasting a few moments while the computer sorts the 17 melodies into three independent lists, each in regard to some metric to the first melody played. All of the melodies are measured with respect to their similarity to the first melody by three different morphological metrics [Polansky, 1987a], or distance functions on melodies. The basic metric from which the three variations are derived is a simple version of what I have called the Ordered Combinatorial Direction metric:

\[
d(N,M) = \frac{1}{L_m} \sum_{p=1}^{L-1} \sum_{i=1}^{L_j} \text{diff} \left( \text{sgn} \left( N_{p,i} - N_{p,i+1} \right), \text{sgn} \left( M_{p,i} - M_{p,i+1} \right) \right)
\]

where: \( N, M \) are two morphologies, or melodies (ordered lists of numbers); \( N_{p,i} \) are the \( i \)-th elements of morphologies \( N \) and \( M \); \( \text{sgn} \) is a contour function: it returns a -1, 0, or 1 depending on whether or not first element is bigger, equal to, or smaller than the second; \( \text{diff} \) is a binary comparison: if the values are equal a zero is returned, if unequal, a 1; \( L_m \) is the binary coefficient of the length of the melodies (in this piece always 17), or the number of pairwise relationships. If the melody is \( L \) notes long:

\[
L_m = \frac{L^2 - L}{2}
\]

(or in this case, 136). \( L_m \) describes a "half-matrix minus the diagonal."

More simply put, this metric compares the combinatorial contours of two melodies by summing the difference of corresponding cells in the two \( L_m \) contour matrices generated by \( N \) and \( M \). Each cell contains a -1, -1, 0, or 0, depending on the combinatorial contour of the melody itself. (Polansky and Bassein (1990), Marvin and LaPrude (1987), Friedmann (1985, 1987).) If all cells of the matrix generated by \( N \) are equal to those generated by \( M \), the metric would be 0, and these melodies would be considered to be the same in terms of their combinatorial contour. A value of 1 would indicate that the two melodies were as far apart as "they could be" in terms of their contour.

Three lists of the 17 input melodies are made according to this metric in the pitch dimension, the duration dimension, and an equally-weighted average of pitch and duration dimension. There is no necessary correlation between the three lists: they could all be identical, or completely different depending on the correlation between the similarities in the duration and pitch dimensions to the first melody.

Section 3 simply plays back the three lists, simultaneously, on three MIDI channels. The performer does nothing, creating a nice symmetry: in the first part only the performer plays, in the second nobody plays, in the third only the computer plays. Several variables are left to the performer, which are specified at performance time, including: the average number of repeats for each melody, the probability of changing a MIDI preset at any given time in the piece, and the list of MIDI presets available to each channel. By
working with these parameters, and customizing the sounds and melodies, the performer has tremendous degree of control over the piece. The duration of the piece is greatly affected by the average number of repeats for each melody. The performer's specification of MIDI preset usage adds a distinctive individual voice to the performance. In this respect, the way Daniel Goode used the trees was a great achievement.

17 Simple Melodies was a philosophical and a technological experiment. There is a conceivable, if extreme description of music which says that hearing it is to some extremes, more, perhaps, avoid the decision about what it should sound like. This is of course odd with the prevailing notion that music is, at its most fundamental definition, in some way inescapably tied to sound (I don't believe this to be necessarily true), and also against the music fashionable idea that if a composer needs to explain a piece, it can't be very good. 17 Simple Melodies, as heard, but it is also in some sense, pure explanation, both from the composer to the machine and the performer, and from the performer and machine to the audience.

I was not so interested in removing sound from music as in refocusing the performer's and audience's attention: the particular timbres might be irrelevant (there is no real skill or compositional intelligence required to choose a MIDI preset). The very straightforward structure of the work should emerge as the salient feature. This has proved to be difficult for some listeners more accostomed to reacting to electronic music almost exclusively on the basis of its timbre. In 17 Simple Melodies this is like reacting to the color of the chair you're sitting in; it's an important part of your experience, but not one that I wish to have any part in determining. As a result, this work has confused, slandered (and, I hope, fascinated) many who have heard it.

Pieces: by Nick Didkovsky

DrNerve.html

Doctor Nerve is a seven-piece band based in New York City that plays hard-edged, experimental atonal music for which I have been composing for over eight years. In February of 1991 I began work on a program called DrNerve.html, which generates musical compositions orchestrated specifically for Doctor Nerve's seven instruments: soprano sax, trumpet, bass clarinet, vibraphone, electric guitar, electric bass, and drums. I wrote DrNerve.html to give myself a compositional shock. I suspected that the resultant music would have an innate freshness, unpredictability, and curious mixture of clumsiness and precision, as the program would consist of only a limited framework of instructions, devoid almost entirely of world knowledge. I suspected that it would generate music that my prejudices overlooked, and I was interested in the result. One of the resultant works (DrNerve.html) has been developed into a complete live performance.

The realization of a composition in DrNerve.html takes four stages. First, the program is run, taking about thirty seconds of computer time, producing a piece of music from 1 to 3 minutes in length. Second, if a particular piece comes to a moment worth developing, it is stored to disk. Third, the file is loaded into Deluxe Music Construction Set (DMCS), a commercial notation program, where I can rearrange, edit, and develop the ideas originally generated by DrNerve.html. (On one occasion, DrNerve.html generated a piece which I felt required absolutely no rearrangement or editing whatsoever, and is currently in the band's repertoire.) Fourth, when I feel that a composition is complete, it is touched up in DMCS, and saved to disk. I wrote a program called Cogist Companion which translates a DMCS score file into one readable by Dr. T's The Cogist, which generates high-quality scores for the musicians. Finally, I make cassette tapes of the piece for each of Doctor Nerve's musicians. Since DrNerve.html typically generates music which is almost completely intractable to technical difficulty, these tapes speed the musicians' learning process.

Description of the program

DrNerve.html is neither an "expert system" nor "rule-based" software; it can generate music very quickly—it knows very little. To generate a measure of music, it first chooses a subset of a pitch row and a subset of a duration row, which may fixed for the entire composition. Using these subsets, a bass line is generated for one measure by splitting the measures in some number of phrases, and cycling through the subsets of pitch and duration. Once the bass line is composed, it generates a derivative drum part for key drum, snare drum, hihat, and toms. Doctor Nerve's drummer, Leo Cesa helped develop a drum part algorithm which creates an accompaniment that is beautifully tied to the bass. The results of this algorithm are shown in the score excerpt below. DrNerve.html has contributed a tremendous vocabulary to any composer's palette. I was repeatedly presented with ideas that should have been obvious to me, but which I'd never realized. For examples of the music see Didkovsky [1991, 1990a].
### Lottery

Lottery is based on a social model which promotes responsible resource sharing. Performers are temporary members of an evolving microsociety, the behavior of which is correlated to audible change. Lottery's goals are: 1) that its participants recognize their joint situation, and address this interdependence rationally 2) that its participants evolve their microsociety into responsible resource-sharing behavior, and 3) to produce a music which closely models the development, successes, and failures of the first two goals.

The piece is derived from the work of evolutionary biologist Garrett Hardin and computer scientist/theorist Douglas R. Hofstadter (1983). It was programmed by myself, with the assistance of Phil Burk, Larry Polansky, and Robert Marasy. The piece was first performed by Burk, Polansky, Marasy, and myself at the Mills College CCM on 4/1/90.

Lottery models a society whose participants vie for control of a common resource. "The social arrangements that produce responsibility are arrangements that create a resource." (Hardin). A shared musical environment constitutes this resource, and Lottery employs a cooperative device to oversee its allocation called a Luring Lottery.

A Luring Lottery introduces a contradiction between the desire to win, and the desire to gain. The awarded prize is a quantity \(X/N\) where \(X\) is some maximum prize value (like a large sum of money), and \(N\) represents the total number of ballots submitted to the lottery. Participants may enter as many ballots as desired, but as the number of ballots \(N\) increases, the awarded prize \(X/N\) decreases! An overly zealous participant might enter millions of ballots, assuring a victory but the awarded prize would be negligible. This simple and beautiful inverse relationship has tremendous power, and is inherently resistant to the hoarding and over-exploitation of common resources.

Lottery’s common resource is musical control over Sound State. A Luring Lottery is called periodically, which awards temporary control of the current Sound State to the winner. The magnitude of control given to the winner is divided by the total number of ballots submitted. Participants are in a simultaneously cooperative and competitive environment, where they must balance the desire to control the piece, and assuring that such control will not be diluted to a meaningless level. This behavior is explored in a highly quantified manner, where the evolution of the performers' microsociety is readily analyzed.

Lottery is for any number of performers, each with an Amiga. All Amigas are linked together via MIDI, used purely as a networking protocol. The sound generation of Lottery is handled by the Amigas. The common resource is a Sound State of audio waveforms, built from the sum of nine waves in natural harmonic ratios. The number of waveforms is four times the number of Amigas in the piece. Each new Sound State, realized every thirty seconds as the result of a new Luring Lottery, is a weighted mix of the piece's previous Sound State and the winning Sound State. During the thirty second interval, each performer may conduct a potential Sound State by editing the magnitudes of the partials in any of all waveforms. Each performer may also submit any number of ballots to the lottery during this interval. The host machine selects a winner at random, and implements the winner's Sound State. As required by the Luring Lottery, the extent of this implementation is scaled down by the total number of ballots. If participants in a performance submit a huge amount of ballots, the piece will not change noticeably from its current Sound State.
The first rehearsal of Lottery ought to have been public — the performers’ behavior was the most radically evolutionary and self-aware. The "microcosmic" evolved most painfully during this rehearsal, as we witnessed group behavior ranging from "arms escalation" (all participants begin to subtract higher and higher numbers of balloons), "de-escalation" (participants see the futility of their joint behavior and begin dropping out of the ballooning process, or submitting very low counts), "peace" (a group behavior of low ballooning), and "de-stabilization" (one participant gets greedy and starts to submit significantly more balloons than the rest of the group; starts to win disproportionately often; others follow suit). By the time we performed the piece, the group as a cooperative organism had already been established, and the performance was rather tame. But what it lacks in social dominance it gained in a cooperative musical making. The group never established a definition of responsible behavior, that is, defined by Hardin, which is displaced by all and which benefits each member maximally. Our behavior developed intuitively into low ballooning. Recently, I have derived more formal approaches which maximize each participant’s personal gain. These models require access to a random number generator which determines whether or not a participant enters a given lottery, and how many balloons she should omit. The next version of Lottery will have this tool available to its participants, who can pursue a more quantified notion of cooperation.

Collaboration: Slippers of Steel
There Is More Headroom, But One’s Feet Are Forced Into Slippers Of Steel is a live performance for two electric guitarists, two computers linked via MIDI, and electronic sound-generating hardware. The title is an excerpt from Melody Sumner’s booklength poem, The Time Is Now, Slippers... was composed and programmed by Nick Nikolov and Larry Polansky in the spring of 1991, and premiered on 5/22/91 at Dartmouth College. It addresses human and computer anxieties, pushing the limits of the capabilities of live and inorganic performers to perceive and react to a rapidly changing performance environment. The piece alternates six times between two performance behaviors: Layers and Hits. The decision to change from one to the other can be made by either computer, which informs the other machine (via MIDI) and the live performers (via text on the screen) of the change. The machines continuously sample each other’s behaviors via MIDI, keeping all aspects of the performance in sync.

Layers consist of parameterically defined streams of sound, each lasting from 4 to 20 seconds. During a Layer section, any number of sound streams can occur, separated by 2 to 13 seconds of silence. As both computers and performers play during layers, four sets of such sound streams are audible, overlapping in various combinations. The computers use the following parameters to generate their sound streams: Pitch Mean and Range, Loudness Mean and Range, Horizontal Density Mean and Range, Vertical Density Mean and Range, and Legato Mean and Range. Each computer-performed sound stream is played on MIDI devices. Each stream may be performed in one of three modes: 1) independently of the other computer, 2) imitating the other computer, or 3) opposing the other computer.

In independent mode, random sets of starting and ending parameters are created. The starting values evolve to the ending values either linearly or by following a half cosine curve (Tesney, 1987) over the course of the 4-20 second stream. A stream might go from low high pitches, while its dynamics fall from fff to mf, while evolving from monophony to polyphony, while changing from a very fast dense cloud of sound to a very sparse one, and from a great degree of overlap to a very short staccato treatment of pitch events. Each of these parameters may have their own trajectories resulting in unusual and quite beautiful phrasing. We found that the common clichés such as fast-slow and sparse-quiet were shattered by this approach to phrase generation, which deliberately avoids such artificial correlations.

A sound stream in imitative mode continuously samples the parameters of the other computer, and plays them. Each computer is responsible for programming his own "paranetic player" which interprets these values musically. The skeleton of the overall program runs identically on both computers, but provides for customized interpretation. Thus, even in imitative mode, the performance of each machine differs enough to be musically-vital, yet follows the parameters closely enough to make the notion of imitation obvious to the listener. For example, one computer’s imitative response might be to play dyads in perfect fourths with the pitch values it sampled.

A sound stream in opposition mode also continuously samples the other computer, fetches its current parameters, then "flips" these values with a composer-written custom opposition function. Again, each computer provides his own opposition functions, some of which are simple, some more elaborate.

While performing their layers electronically, the composers also give performance instructions to the two guitarists, which are identical to the modes of the machines: 1) "Play independently" 2) "Imitate the..."
other player" 3) "Oppose the other player." An additional "Stop Playing" command provides for the silence between live sound streams. The guitarists apply their own intuitive notions of opposition and imitation to the other's performance. When independent, each guitarist may improvise freely, bearing in mind that his playing must evolve from a starting state to a goal state. This scheme creates fascinating interactions.

Performance modes can combine in the six following ways ("unit" refers to either guitarist or computer):

<table>
<thead>
<tr>
<th>Units</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>independent</td>
<td>independent</td>
<td>independent</td>
<td>imitate</td>
<td>imitate</td>
<td>imitate</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Behaviors chosen by (1), (2), and (3) are self-evident. (4) - (6) are more interesting. (We will use "fast" and "slow" as generic descriptors of high and low parameter values respectively). (4) causes both units to converge immediately on the same unchanging parameter values. If I plays "fast," so does II (imitation). As each unit poll the other, confirmation of "fast" is fetched, and imitated, thus never changing. This behavior is static and identical. (5) generates rapidly oscillating behavior. Following the "fast/slow" terminology, if I fetches "fast" from II, it will imitate, and play "fast." An instant later, II fetches "fast" from I, and opposes with "slow." An instant later, I fetched "slow" from II and imitates, causing 2 to swing back to "fast" by opposition. On the average, these oscillations reound at the rate at which each machine reads the other's data. For human performers, this speed of oscillation is directly proportional to the performer's ability to perceive the other's activity and react decisively. One of our goals is to reduce the live "sampling rate" as much as possible. (6) results in static, opposite behavior. If I fetches "fast" from II, it will oppose it with "slow." When II fetches this "slow" value from I, it will oppose with "fast." This exchange settles almost immediately into static opposites.

At any time during Layers, one of the computers may decide stochastically to switch the whole piece to Hits, a sequence of reaction exercises, separated by unpredictable silences. Performers wait in silence until one of the computers initiates a "hit" by emitting a single sound event, at which time the other computer and the two guitarists must react as quickly as possible by responding with their own single sound event. After this flurry, all wait in silence again for the next "hit."

The idea is to minimize the duration between the original sound event and the reaction events. The computers' limitations are easily quantified (the MIDI data rate and the efficiency of the software response). It is more difficult to quantify the response time for live performers, whose ability to react quickly is a complex summation of personal concentration, speed at which sound reaches the performer from the PA system, and the rate at which sensory signals travel through the human body. We aim to minimize this human interval as much as possible. During Hits, any computer may give the signal to switch back to Layers, which the ensemble follows.

Performance of Slippers of Steel vary tremendously. The use of stochastic elements in structuring the changes from Layers to Hits makes it difficult to predict the overall length of the piece. We have rehearsed 5 minute versions and 20 minute versions! The use of chance elements in the selection of performance parameters and performance modes also ensures that the middle level elements of the piece are unlikely to be repeated. As the piece addresses the very elemental musical disciplines of reaction time and fast decision making, it is a raw and exposed performance experience, exhilarating and exhausting to play.

A Selected List of Other HMSC Works by the authors

Larry Polansky
Four Voice Canon #6 (1990a, 1988a, 1988c)
Horn (for Chris Bobrowski) 1990
3 Studies. 1: Rhythm II. Melody III, Harmony 1990. For the Downtown Ensemble. (1990d)
Distance Music 1987, 1987d)

ICMC 43