# array

communications
of the
international
computer
music
association

vol. 12
number 3
summer 1992

TWLM
Note -
Polansky

contents

Contact:
Bob Willey
Center for Research in
Computing and the Arts
University of California, SD
0037 9500 Gilman Drive
La Jolla, CA 92093
USA

TEL (619) 534-4383,
270-0030
FAX (619) 534-7944

e-mail:
bobw@sdcarl.ucsd.edu

Max Mathews gave a workshop in the use of the digital baton with the Conductor program. His concert on June 24th demonstrated its use with Chopin's *Prelude in C minor, Three Chapters from "Book of Dreams"* by Richard Boulanger, and pieces which use the Pierce scale: *Eros ex Machine* by Jon Appleton, *I Know of No Geometry* and *Solemn Song for Evening* by Richard Boulanger, and *A Wild and Reckless Place* by Ami Rudunskaya.

Equipment for these residencies was provided in part by Apple Argentina with support from Softlider and the Macintosh Art Center.

# research notes

## The World's Longest Melody (Portable Version) Beta 1.1

*Larry Polansky*

### Introduction

*The World's Longest Melody* (*TWLM* ) is a "portable" piece for live performer and computer, and any MIDI synthesizer. It is written in HMSL and distributed as source code or as a turnkey, executable image.

### Musical Idea

The main idea of *TWLM* is the simple specification of a stepwise melody in terms of "the probability of the melody going in the same direction as it previously did." In each of the four parameters (pitch, duration, loudness and MIDI control), the melody, once started, proceeds by "step."

Whether that step is positive (up) or negative (down) is determined by the probability for that parameter ($p_{pitch}$, $p_{dur}$, $p_{loudness}$, $p_{control}$). For example, if $p = 0$, the step will be in the same direction as the previous one. If $p = .5$, there is an equal likelihood of the step changing direction or proceeding in the same direction. If $p = 1$, the melody will change direction at every step. In other words, the higher the value for $p$, the higher the probability of "ripple." Conversely, the lower the value for $p$, the longer the trajectory of the melody: once it starts going in some direction it will likely continue. Probabilities are independent in the four parameters. A continuously rising or descending pitch ($p_{pitch} = 0$) might be combined with a rapidly oscillating duration value ($p_{duration} = .8$).

This melodic idea is intended to be descriptively and algorithmically simple, yet melodically powerful. It is based entirely on a notion of binary contour (there is no possibility of no motion), and, like the process of delta modulation, only recognizes the existence of one stepsize at any given time. Attention is focussed on explainable melodic parameters, like trajectory, rate of change and so on.

### Technical Requirements

*TWLM* is distributed as a turnkey, stand-alone application, or as HMSL source code to be compiled and run. *TWLM* will run on on a Macintosh with at least a 13" monitor. Turnkey users do not need HMSL. Source code users need a reasonably current version of HMSL: it is written in HMSL 4.19 but will probably run on 4.16 or above. The performer should be using at least Macintosh System 6.07 and a current version of the MIDI Manager.

The piece is set up to use up to 16 independent MIDI channels, although it can use any number up to that. It could even run on one MIDI channel, by setting all of the *WLM*'s to the same channel (see the notes on the piece for the effects this will have on things like MIDI Control and MIDI Preset).

### Running TWLM on a small screen

If you only have a Mac Plus size screen, and still want to run it, you will need to alter the code a bit. The simplest solution might be to slightly rewrite the code so that it uses two screens instead of one. I'll be happy to advise you on the procedure for doing this. It's a fairly simple matter of repositioning all the control grids (in the file `WLM_SCREEN`) and is more of a graphic layout problem than a programming one. This is only an option for source code users.

There is a variable in the code for "number of channels" for those source code users who might want a less dense screen and don't need 16 MIDI channels. This will also help in porting the piece to a smaller screen.

**Compiling and starting the piece**

Turnkey version
With the executable (turnkey) version, click on the `HMSL_TWLM` icon. This will bring up the *TWLM* screen. If MIDI is connected properly, you're ready to go. When you're finished playing, click the close box.

Source code version
With the source code version, the piece needs to be compiled. To do this, set up your HMSL `ASSIGNS` file so that there is a volume called

```
WLM:
```

For example, if your hard disk is named `DONALD_DUCK`, enter in your ASSIGNS file:

```
ASSIGN WLM: DONALD_DUCK:WLM
```

Boot HMSL, execute your assigns, and include `WLM_LOAD` from that directory. That compiles the piece. Then type:

```
DO.WLM <cr>
```

This will initialize the piece, and put up the *WLM Screen*. If MIDI is connected properly, you're all set. If you exit the piece, you can get back into it by just typing HMSL, and selecting the *WLM Screen*. Don't type `DO.WLM` again or you might crash your system (it'll try to instantiate a lot of data structures that already exist, etc.).

**TWLM Screen**
The screen for the piece consists of a set of HMSL control grids that allow for real-time control of up to 16 *WLM*'s, an HMSL data structure which is a subclass of jobs. Once started a *WLM* executes the melody algorithm described above independently in four parameters: duration, pitch, loudness and MIDI control. With this screen the performer can start, stop and shape these 16 melodic generators in flexible and interesting ways.

Grid functions are as follows, starting from the left hand side of the screen, going across, and down.

**WLM**
This column allows the performer to *start* and *stop* each of the 16 predefined *WLM*'s. When stopped, a *WLM* will turn off its last note.

**chan.**
This column allows the performer to set the *MIDI channel* for each *WLM*. A simple setup, with each *WLM* assigned to a separate MIDI channel and one MIDI voice per channel, will work fine. Each WLM turns off its previous note before turning on its next one, so, having more than one *WLM* on a channel will produce some interesting effects. If your synthesizer uses multiple voices on one channel, you could assign all the *WLM*'s to the same channel. When the *WLM*'s "cross" notes, they will turn notes off. The preset and control grids will no longer affect just their individual *WLM*'s since they are MIDI channel specific.

**d-on, p-on, l-on, c-on**
These four columns allow the performer to *enable or disable the WLM process itself* in the four parameters. In other words, if `L-on` is disabled (non-highlighted) for a specific *WLM*, then there will be no loudness evolution.

**d-prob, p-prob, l-prob, c-prob**
These four columns allow the performer to set the probability for the *WLM* process in the four parameters. The numbers are from 0-100 (fixed point representation of 0.0 -1.0). The higher the number, the more likely the parametric *WLM* process is to change direction. These are normal HMSL numeric grids: clicking on the top or bottom half of the cell raises or lowers the value by 1; holding the mouse button down and moving the mouse up or down (outside of the box), raises or lowers the value more quickly.

**d-inc, p-inc, l-inc, c-inc**
These four columns allow the performer to set the *step increment* for each of the four parameters. For example a step increment of 1 in pitch is a semitone, while 7 indicates that the *WLM* would move by perfect fifths. In the duration parameter, the increment specifies a ratio to the current duration. For example, if the current duration is 100, and the duration increment is 10, 1/10 * 100 (10) will be added or subtracted to the previous duration to get the new one (110, or 90). The next duration will be, in the negative case, 90 - (1/10 * 90) = 81.

**dur, pitch, loud, cont.**
These four columns allow the performer to force a value into a parameter. This can be useful when the *WLM* process is disabled in some parameter, and you want to move that parameter manually, or when you want to suddenly "kickstart" a *WLM* process into a new parametric range.

**ctr-#., pre-#**
These two columns allow the performer to set the preset (or MIDI program) value and controller number for a given *WLM*. The controller number specficies which MIDI controller will be used for that *WLM* (of course you need to assign that controller to some value, like pan, pitch bend, etc. on your synthesizer). Note that these numbers will affect the MIDI channel of the *WLM* with which they are associated. If two *WLM*'s are on the same MIDI channel, changing the value of controller-# or MIDI preset for one of them will affect that entire channel. You can have two *WLM*'s with different controller-#'assigned to the same MIDI channel , but changes in both controller values will affect all notes on that channel. MIDI preset is also channel specific: changing that value for one WLM will also change MIDI preset for all other *WLM*'s on that channel. Allowing changes in MIDI preset is something I added as a kind of afterthought. I originally envisioned *TWLM* as a kind of piano piece, an experiment purely in melody, and intended it for piano module, samples, or MIDI controlled piano. However, the performer is free to experiment with different timbres.

**Functions (Play, Sync)**
This grid has three cells. The first, `PLAY`, starts a routine running which randomly and rather slowly turns on and off the 16 predefined *WLM*'s. It's a simple "automatic" performance routine that may or may not be useful in performing the piece. When you turn it off (de-highlight it), it shows the performer (via the *WLM* grid, the leftmost column) which *WLM*'s are left on.

The second cell, `SYNC`, starts all *WLM*'s currently turned on, at the same time. This is very useful for generating polyrhythms. For example, by turning off the duration function for two *WLM*'s, setting their durations to 15 and 45, starting them, and then hitting the `SYNC`cell, you will hear a 3:1 rhythm, with whatever pitch, loudness and control process you have running. Using SYNC, you can set up extremely complicated 16 voice integer polyrhythms quite quickly, with evolving pitch and loudness structures.

The third cell does nothing (reserved).

**Tempo**
This cell changes the value of HMSL's `RTC.RATE`, effectively speeding up or slowing down the entire piece. All duration values are scaled by this value.

**MIDI.kill**
This cell simply executes a MIDI.KILALL when it is hit. It's useful for silencing the piece. The piece itself executes this word when you quit.

**d-range, p-range, l-range, c-range**
These four double rows allow the performer to *set the upper and lower limits* for each of the four parameters. The upper row is the lower limit, the bottom is the higher. This allows you to split up the *WLM*'s, for example, into different pitch and duration ranges, and can be a very useful feature in performing the piece (allowing them to interact, cross, be independently perceived, etc.). When a *WLM* reaches the top or bottom of its range, it will clip (stay there).

## Miscellaneous Extensions and Ideas

Use of TIME-ADVANCE

Note that since *TWLM* uses the standard HMSL MIDI event buffer, certain actions (like changing MIDI preset), will have a delay associated with them. This delay is the value of TIME-ADVANCE, the HMSL variable for the length of event buffering. It's set by default to be 60 (1 second with the current RTC.RATE) If you want a more immediate response, change the value of TIME-ADVANCE before you boot *TWLM*, or insert something like:

```
10 TIME-ADVANCE !
```

somewhere in the source code for the piece. However, note that changing the TEMPO value on the *WLM* screen will also alter TIME-ADVANCE. This is because TIME-ADVANCE is measured in ticks, and RTC.RATE sets the number of ticks per second.

Default values

One obvious change the user might wish to make to the source code is in the default values for the various parameters and *WLM* instance variables. This may be done in either the INIT: method of the *WLM* class itself, or in the dynamic instantiation of the 16 *WLM*'s in the file WLM_LIST.

WLM "histories"

John Puterbaugh, a graduate student at Dartmouth, suggested a nice musical addition: the possibility of adding a longer *history* to the melodic probabilities. I think that this would make for some interesting extensions to the piece. For example, each parameter could have a separate variable, called PITCH-HISTORY-LENGTH, DURATION-HISTORY-LENGTH, and so on, and the *WLM* play function for each could test the change in direction probability (the current $p_{dur}$, $p_{pitch}$, $p_{loud}$, $p_{control}$) against some kind of weighted average of previous directions over the history. This would be an extremely simple (and powerful) feature to add. All that would need to be added to the code, in the file, WLM_OBJ, is a computation of the running averages of direction in the four parameters as the melody progresses, stored in (the currently defined) instance variables for last direction. By changing history length, the performer could greatly affect the behavior of the piece. It would probably be useful to chronologically weight that running average, so that more recently occurring directions had more weight.

ADM vs. DM: WLM control of step-size

The size of the step in each parameter could also be subjected to the *WLM* process. Since I have used the analogy of delta modulation (DM) for the *WLM* process, this might be thought of as adaptive delta modulation (ADM). The musical affect will be a melody that not only has a direction, but a kind of directional acceleration. Steps will get bigger and bigger and smaller and smaller (it would probably be necessary to have an upper limit for step-size). For source code users, this is relatively simple to do: simply mimic the code used for applying the WLM process to direction. The biggest problem might be finding a place on the screen for eight more columns (enabling or disabling the *WLM* process for step-size, and changing its probability).

Using TWLM without the graphic screen

If you are an HMSL user, and have the source code for *TWLM*, you can use it without the graphics screen, simply by using the *WLM* class in a normal HMSL context. *WLM*'s are a subclass of the HMSL class `OB.JOB`. The source code itself should serve as a manual for how to use them, but the following is a simple example of how to create and start one:

```
OB.WLM MY-WLM       (instantiate a WLM)
SETUP: MY-WLM       (required, it places the "play" functions in the WLM)
HMSL.START          (this will start HMSL and bring up the Shape Editor)
START: MY-WLM       (type this in the Forth window)
```

That will start a *WLM* playing with its default values (see the source code, in the `INIT:` method for the *WLM* class in the file `WLM_OBJ`). There are many predefined methods for changing a *WLM*'s behavior. The file `WLM_OBJ` contains all of the methods for this class (for example, `PUT.PITCH-PROB:`, `PUT.CHANNEL:`, etc.), and the class itself is very simple to use. The file `WLM_LIST` just defines an Object List for 16 dynamically instantiated *WLM*'s, which are used in the file `WLM_SCREEN` for the graphic interface.

**History**

An early version of *The World's Longest Melody* was first written in the mid-1980's, and generated on an Amiga computer for a concert by Phil Stone, Phil Burk and myself as part of the Network Muse series of live interactive computer music concerts in San Francisco. The "score" for the piece was published in *Perspectives of New Music* (Volume 25, Nos. 1 and 2, Winter/Summer 1987), as number V of the set of pieces called *Distance Musics I-VI*.

This current version of the piece is an HMSL MIDI implementation, with a graphics interface for live performance. It is meant as a portable piece for other performers, and does not require any computer or HMSL expertise to perform. Heavily documented source code is provided so that the performer may rewrite, alter, and extend the piece as she wishes. I am happy to support this kind of work in any way I can, and performers are encouraged to call, write or email me with ideas and questions.

**Distribution**

*TWLM* is distributed by Frog Peak Music (A Composers' Collective), Box A36, Hanover, NH 03755. It is intended as "evolving shareware." Performers who are HMSL users can use the piece as is, or make changes in the musical ideas, implementation, and source code. If performers distribute their own changes, I would like the piece to retain its title, and that I be listed as the composer with extensions and revisions by the performer. I would also like to, if possible, receive notification, program notes, recordings and so on from any performances. Performers and other users may not sell their own versions of *TWLM*. Performers who only have the executable version may not distribute it, all turnkey versions of the system should be obtained directly from Frog Peak Music.

Contact:
Larry Polansky
Dept. of Music
Dartmouth College
Hanover, NH 03755
USA

TEL w: (603)646-2139
h: (603) 448-1902

email:
larry.polansky@dartmouth.edu

**Acknowledgements**

*The World's Longest Melody* is registered with BMI.
Thanks to Jody Diamond for editorial assistance with these notes.

# opinions and editorials

## Mental Structures

*Stephen Smoliar*

The Generalized Theory of Tonal Music (GTTM) hypothesis of Fred Lerdahl and Ray Jackendoff reduces the role of cognition in listening to music to one of parsing—the formation of "mental structures" which bear the same relation to music as parse trees do to sentences. This approach seems feasible if we identify music with the symbolic primitives which constitute its notation; but it is less viable if we consider music as epiphenomena of behavior (manipulation of our voices and physical devices known as instruments). Perception is concerned with musical events, rather than any notation