The World's Longest Melody (Portable Version) Version 2.0

for David Feldman

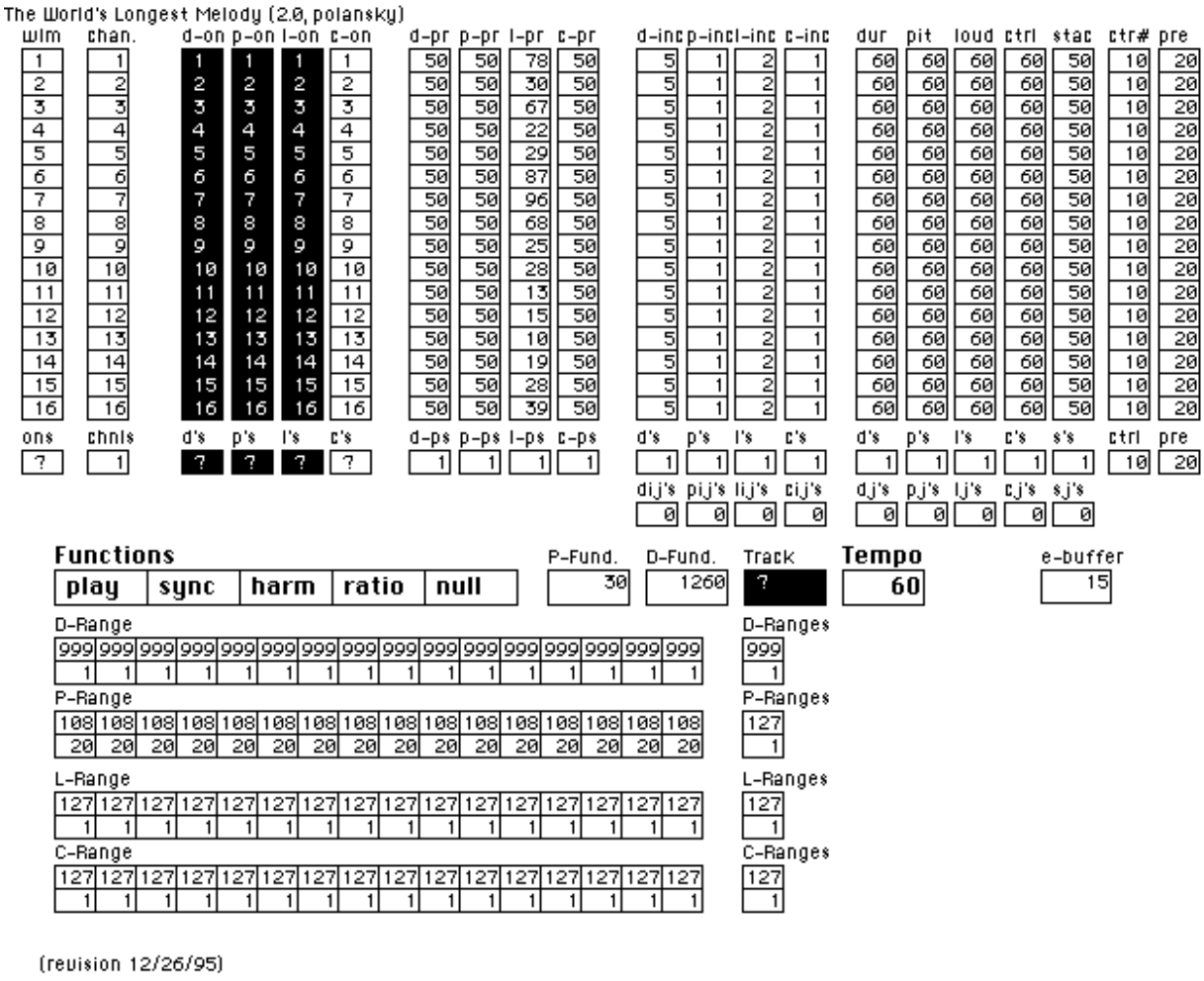
Larry Polansky

Documentation
January 5, 1996

The World's Longest Melody
(Portable Version)
Version 2.0
Larry Polansky

frog peak music (a composers' collective)
box 1052
lebanon nh 03766
ph/fax: 603-448-8837
email: frogpeak@sover.net
http://www.sover.net/~frogpeak/

larry.polansky@dartmouth.edu
http://music.dartmouth.edu/~larry/polansky.html
ph: 603-646-2139



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 turnkeyed, 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 $p = 1.0$ 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 $p = 0$, the melody will change direction at every step. In other words, the lower the value for p, the higher the probability of "ripple." Conversely, the higher 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 ($p_{pitch} = 1.0$) might be combined with a rapidly oscillating duration value ($p_{duration} = .2$).

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 (preferably 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 4.21 but the source code will probably run on 4.16 or above. The performer should be using at least Macintosh System 6.07 (for the source code version, for the turnkey, it shouldn't matter) and should not have OMS, FreeMidi, or any other MIDI extensions in the System Folder (see below).

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 MIDI Control and MIDI Preset).

IMPORTANT: DO NOT HAVE OMS or FREEMIDI IN YOUR SYSTEM FOLDER. HMSL/TWLM USES ITS OWN MIDI DRIVER. USE THE EXTENSIONS MANAGER TO TURN OFF ALL EXTENSIONS. USE THE SIMPLEST, DUMBEST MIDI INTERFACE SETTING YOU CAN (LIKE A STUDIO 4 OR 5 SET TO 1 MHz.).

Running TWLM on a small screen

If you only have a Mac+ size screen, and still want to run it, you will have to scroll to see the whole screen. Another alternative is to alter the code a bit so that the piece 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 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

Turnkeyed version

With the executable (turnkeyed) version, click on the 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 which 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. Below each column (16 cells, one for each voice) is a single cell which globally affects all the voices at once (either numerically, or turning them all on or off).

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. The ONS box below this column turns them all on or off.

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. However, 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. Each WLM turns off its previous note before turning on its next one. 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. The single cell below this column sets all voices to a single channel.

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, there will be no loudness evolution. The single cells below these columns set all voices on or off for the WLM process in that parameter. In other words, turning off all the duration WLM processes will keep the durations constant, but allow the other parameters to change.

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 lower 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. A value of 100 will produce long trajectories, a value of 0 will produce constant changing back and forth by the current increment size. In other words, the higher the value, the longer the trajectory in one direction will be. The single cells below these columns change all 16 voices at once to the specified probability.

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 specificies 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. The single cells below these columns change all 16 voices at once to the specified increment in that parameter.

dur, pitch, loud, cont., stac

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. The single cells below these columns change all 16 voices at once to the specified value in that parameter.. The only parameter that is not controlled by the WLM process is *stac*, which is staccato. It varies between 0-100, with 0 being the most staccato value (you probably won't hear anything for most MIDI voices) and 100 turning the note on for its full duation.

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.The single cells below these columns change all 16 voices at once to the specified value for either control # or preset #.

Functions (Play, Sync, Harm, Ratio)

This grid has five 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 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, HARM, sets the pitch of all of the *WLM*'s to be preset intervals to the value called

P-FUND (see below, it has its own numeric grid). Source code users may easily go into the code and change the intervals that it uses. Currently, they are set to be a "harmonic series," starting from the fundamental. From the fundamental, in terms of MIDI/tempered intervals, they are:

> — unison, octave, octave and a 5th, 2 octaves, 2 octaves and a major 3rd, 2 octaves and a 5th, 2 octaves and a minor 7th, 3 octaves, 3 octaves and a major 2nd, 3 octaves and a major 3rd, 3 octaves and a tritone, 3 octaves and a 5th, 3 octaves and a minor 6th, 3 octaves and a minor 7th, 3 octaves and a major 7th, 4 octaves.

These MIDI values are tempered approximations of the first 16 partials. Users with a tunable MIDI synthesizer can use this to acheive a quick harmonic series tuning (or any other "preset" scale or tuning if the code is altered). Also see the numeric grid P-FUND and the check grid TRACK for more on this feature.

The fourth cell, RATIO, sets the duration of all of the *WLM*'s to be preset divisions of the value called *D-FUND* (see below, it has its own numeric grid). Source code users may easily alter the code to change the duration intervals. The preset intervals are simply the duration fundamental divided by the *WLM* # (1-16). Note that there will be a kind of round-off error for the higher divisions because of the difficulty of getting a small enough duration value which is the LCD of the first 16 integers. Generally, this feature will have more or less the desired effect of "ranging" the duration values of the *WLM*'s. Also see the numeric grid P-FUND and the check grid TRACK for more on this feature.

The fifth cell does nothing (reserved).

P-fund

This sets the value of the pitch fundamental (in MIDI note number) used by HARM. If tracking is on, then all of the *WLM*'s will reset their pitches whenever this is changed. If tracking is off, then this can be changed without effect, but when HARM is clicked, all *WLM*'s will change their values to correspond to this new fundamental (redrawing the screen for every change in value).

D-fund

This sets the value of the duration fundamental used by RATIO. If tracking is on, then all of the *WLM*'s will reset their durations whenever this is changed. If tracking is off, then this can be changed without effect, but when RATIO is clicked, all *WLM*'s will change their values to be divisions of this new fundamental.

Track

This determines whether changing the values of P-FUND and D-FUND will immediately change all of the *WLM* pitch and/or duration values (on), or whether fundamentals can be changed without affecting the *WLM* values (off). If tracking is off, a new fundamental can be set and clicking on HARM or RATIO will compute the new values. Note that if tracking is on, the screen is redrawn every time the numbers for P-FUND or D-FUND change, which will slow the system down tremendously.

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 number.

e-buffer

This cell changes the value of the HMSL event buffer, allowing the system more or less time to pre-compute MIDI events. If the system is sluggish, or not performing accurately (may happen on slow Macintoshes) raise this value.

TWLM uses the standard HMSL MIDI event buffer. Certain actions (like changing MIDI preset, or a number of parameters altered on the screen), will have a delay associated with them (from when you change them until they happen). 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 RTC.RATE = 60) If you want a more immediate response, change the value on the screen, or (source code users) set a different TIME-ADVANCE before you boot *TWLM* by inserting something in the source code like:

```
10 TIME-ADVANCE !
```

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.

.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 processes to interact, cross, be independently perceived, etc.). When a *WLM* reaches the top or bottom of its range, it will clip (stay there). The single cells to the right of each row are global range changers, changing the lower and upper limits of all 16 voices for the specified parameter.

Miscellaneous Extensions and Ideas [source code users]

Default values and the WLM.USER.INIT function

The last file that is compiled in the loading sequence for TWLM is called WLM_USER_INIT, which contains a DEFERred word (see the HMSL manual for more on using DEFER) called WLM.USER.INIT. By placing one's own routines in this word, various "presets" can be created for the piece which will be executed when the piece is initialized. The word WLM.USER.INIT is executed as part of the main initialization word, DO.WLM and is done before the screen itself is initialized. One of the most obvious uses for this word would be, for example, to set the default preset, or tempo for the piece when it begins. These kinds of things can be done by using the WLM-LIST object, which contains all the predefined WLM's. The file itself can be consulted for example code.

The default values for the various parameters and *WLM* instance variables can also 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. I have left this as an excercise for the programmer.

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 [source code users]

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*. Recordings of several of *The World's Longest Melody Piano Studies* appear on the *CD entitled Hallways: 11 Musicians and HMSL*, available from Frog Peak Music. An extended live performance improvisation version of the piece has been performed several times, premiered at the Here Gallery in NYC (1994). A version for computer and rock band of the piece (*The World's Longest*

Melody (Ensemble, score available from Frog Peak) was premiered by the Nick Didkovsky/Larry Polansky Ensemble Boston, 1992.

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. 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 freely on the worldwide web and also by Frog Peak Music (A Composers' Collective), Box 1052, Lebanon, 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 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*. Please feel free to contact me with bugs and suggestions. *The World's Longest Melody* is shareware. Please send CD's, videos, tapes, LP's, software, comix, stamps, or whatever else you think is cool.

Acknowledegments

The World's Longest Melody is registered with BMI. Thanks to Jody Diamond for editorial assistance with these notes, and to Phil Burk, for excellent suggestions regarding the software.

Larry Polansky
Dept. of Music
Bregman Electronic Music Studio
Dartmouth College
Hanover, New Hampshire 03755
w: 603-646-2139 h: 603-448-1902

email: larry.polansky@mac.dartmouth.edu
http://music.dartmouth.edu/~larry/polansky.html

Things to Be Added, Known Bugs

- Silence function (probablity of a silence in a given WLM)
- Ability to store and retrieve screen configuration
- Ability to activate on a number of WLM's without turning them on, and then turning them on with a button at the same time.
- Remove menus from application