SoundHack

user's manual

version 0.74/0.80

Tom Erbe

School of Music
CalArts

SoundHack 0.74/0.80

© 1994 Tom Erbe
School of Music
California Institute of the Arts
24700 McBean Parkway
Valencia, CA 91355
tre@music.calarts.edu
http://music.calarts.edu/~tre

SoundHack performs various soundfile manipulations that have been previously unavailable on the Macintosh. SoundHack includes soundfile type conversion, spectral mutation, spectral dynamics processing, a varispeed/sample rate converter, soundfile convolution, ring modulation, the phase vocoder, a binaural filter and an amplitude analysis and gain change module.

SoundHack can now read and write the following soundfile formats: Sound Designer II, Audio IFC, Audio IFF, IRCAM, DSP Designer, Microsoft WAVE (RIFF), NeXT .snd (or Sun .au) and TEXT. It can read (but not write) raw data files. It can read and write soundfiles with 4-bit ADPCM, 8-bit μ Law, 8-bit linear, 8-bit unsigned, 32-bit floating point and 16-bit linear data encoding.

Important: SoundHack works only on Mac II and above machines with floating point! That is: you need a 68020 + 68881, a 68030 + 68882 or a 68040. Note that some versions of the 68040 have no floating point and will not work with SoundHack. If you have no floating point, use SoundHackNF. It is slow, but it works.

As of version 0.74 and 0.80, the program is split into two versions. Version 0.80 is a non-shareware version, which differs from version 0.74 in the following ways:
• There is a native Power Macintosh binary.
• The spectral mutation functions are included.

SoundHack 0.80 is available from Frog Peak Music, PO Box 1052, Lebanon, NH 03766, 603-448-8837. It costs $50 to unregistered users and $15 to those who are registered users (those who have sent me music or a shareware fee or have bought previous versions from Frog Peak).

Thanks to Dr. Durand Begault of NASA-Ames for letting me use his binaural filter coefficients, Dan Ellis of MIT Media Lab for helping me with the Csound analysis feature, Larry Polansky for his mutation functions and for helping me edit this document, Phil Burk, Curtis Roads, Richard Boulanger and many others for their many comments and encouragement, the Bregman Electro-Acoustic Music Studio at Dartmouth College for sponsoring my initial PowerPC developement and the Center for Contemporary

Music at Mills College and the CalArts School of Music for sponsoring all of the rest of my activities.

Please write me if you have any problems or suggestions!

Tom Erbe

• FILE MENU

Open...

Opens soundfiles with the following file types:

```
Sound Designer II (Sd2f)            Audiomedia (Sd2f)
DSP Designer (DSPs)           Audio Interchange File Format (AIFF)
MacMix (MSND)                 Audio Interchange File Format - C (AIFC)
IRCAM/BICSF (IRCM)            Microsoft WAVE (RIFF)
NeXT/Sun (NxTs)
```

If the file doesn't have a Macintosh 4 character file type, it will still appear in the open file dialog box provided it has one of the following name extensions:

```
.aifc .aiff .au   .irc  .sf   .snd  .wav  .WAV
```


Once the file is open the Soundfile Information dialog box appears. This dialog box gives the name, sample rate, length in seconds, number of channels, type and numeric format of the soundfile. This file is used as the input soundfile to the processes under the Hack menu. Only 1 input soundfile may be open at a time.

Open Any...

Opens any file as a soundfile.

This is used to open headerless and text soundfiles. Text soundfiles should be formatted so that each line is a fixed point sample. Here is an example of how the text should look:

```
-0.054688
-0.015625
-0.007812
0.015625
0.078125
0.093750
0.117188
0.093750
0.015625
0.000000
-0.117188
```

Headerless files (both text and raw) must be saved to another format before being processed. You should set the proper sample rate and data format with the Header Change... dialog before making this conversion.

Close

Closes the soundfile.

Save A Copy...

Saves a copy of the opened soundfile in any soundfile format. *This is the main command for those of you who are just copying one soundfile format to another.*

lisTen to AIFF file

This will play the open AIFF or AIFC file through the Mac's speaker. You will need SoundManager 3.0 or an AV machine for this to work properly.

Import SND resource...

This will allow you to convert an Apple sound resource to a soundfile. This does not work with compressed ‘snd ‘ resources.

Export SND resource...

This will allow you to write part of a soundfile into an Apple sound resource. The length of the sound resource is limited to the amount of memory allocated to SoundHack.

Quit

Obvious.

• EDIT MENU

The usual stuff, not too useful in this program though.

• HACK MENU

This is where all the soundfile processing is. Most of the things in this menu involve a lot of calculations, and take time! SoundHack takes over your Mac to do these, so if it appears that your Mac has frozen up, it probably hasn't. The Phase Vocoder is the slowest of these processes, so have patience.

Header Change...

Allows you to change the sample rate, number or channels, and data format of the open file. If you open a headerless file, you should use this dialog to set things properly before saving a copy.

Loops & Markers...

This menu allows you to change loop pointers and marker locations for soundfiles in AIFF, AIFC and Sound Designer II formats. It also allows you to change AIFF/AIFC specific information.

Binaural Filter...

This process allows you process a monaural soundfile (creating a stereo file). The result is a signal at a simulated position around the head. This is done by using a HRTF (head related transfer function) as a filter, a function for each position around the head.

To use the binaural filter, enter the desired position in the "Angle" dialog box (in degrees) or click the appropriate radio button. This processing module has filter data for 12 positions. If you enter an angle between 2 positions you will get a filter which is the mix of the 2 filters around it. Check the "Normalize" box if you want the output to be normalized (brought to the loudest possible level) after computation.

The “Moving Angle” box will allow you to do moving spatialization. Unfortunately, due to the limits of these binaural filters, the moving binaural effect is less convincing than the stationary effect. I will be working on correcting this in future revisions. These binaural filters are sample rate conversions (for 44100 samples per second) of those used in the dissertation "Control of Auditory Distance" by Durand Begault, which are approximations of the averaged monaural transfer functions shown in Blauert's *Spatial Hearing*.

Convolution..

This process takes 2 soundfiles: an input and an impulse response file. It multiplies the spectra of the 2 files together, producing a new soundfile. The effect is a type of cross-synthesis, in which common frequencies are reinforced. In this implementation of convolution the sound is processed block by block, with each block as large as the impulse response. The "Length Used" window allows you to designate how much of the impulse response file to use. The "kilobytes Needed To Process File" number is an estimate of the application memory size that needs to be set for processing. If you want to use large impulse responses, that is, ones which cause this number to go over 900 (the default size), you will have to quit SoundHack and reset the application memory size.

SoundHack attempts to automatically scale the amplitude of filter gain, but this value is impossible to predict. The "Filter Gain" button allows you additional control over this. Set it to "High" for most cases, but if the input and impulse response have similar spectra, extreme resonance will occur and the button should be set to "Med" or "Low" to avoid clipping If a clipped output still seems unavoidable, save the output in NeXT floating point format, then use the Gain... module to normalize it back to an integer format.

Checking the "Ring Modulate" box allows for ring modulation (or convolution in frequency) between 2 soundfiles. Checking the "Impulse Window" box will cause SoundHack to apply a selected envelope onto the impulse before convolution, resulting in a smoother convolution. A smoothing window is desirable when performing a moving impulse response convolution (described below), since the impulse response will be changing for each block of samples processed. This is also true for moving ring modulation. The triangular window is probably the best for smoothing, a rectangular window is the same as no window at all. Check the "Normalize" box if you want the output to be normalized after computation.

The "Moving" box allows you to perform a moving impulse response convolution. In this process a window moves through the impulse response file, selecting a new impulse response after every block of processing. The window size is set in the "Length Used" field. This window moves through the impulse response file at a rate which insures that the ends of the impulse response file and the input file are reached at the same time. For example, if your input file is 10 seconds long and your impulse response file is 5 seconds long and you have set "Length Used" to 1.0 for 1 second impulse response windows, the process would look something like this:

The first 1.0 second frame of the input file (A) will be convolved with the first 1.0 second frame of the impulse response file (ab). Then the window on the input file is moved 1.0 second forward to B, but the window on the impulse response file is moved only 0.5 seconds to bc. This is so both files will finish at the same time. (Actually, the impulse response file reaches the end first and the last impulse response is zero-padded). In the other case, when the impulse response file is longer than the soundfile, sections of the impulse response file will be skipped over. It is a good idea to set gain to "Low" if using the moving impulse (or save things in a floating point format), as the scaling is fairly unpredictable.

Gain Change...

The equivalent functions in Sound Designer II and Alchemy are faster, so in most cases you will want to use those. However, SoundHack will give you an RMS value for the file, and will allow a different gain factor for each channel. It will also work on floating point and μ Law files. Finally, it will allow you to correct for DC offset in your file. Click on "Analyze" and the peak amplitude, peak position (in samples), RMS values and DC offset will be calculated. The gain factors will be set to normalize both channels independently and the additional offsets (a number between 1.0 and -1.0) will be set to correct the file. "Change Gain" will create a new file adjusted by the gain factors set. If you are dealing with a monaural file, only the channel 1 information is applicable.

Mutation...

(This section of the manual is written by Larry Polansky.)
The seven different spectral mutation functions (*USIM*, *ISIM*, *IUIM*, *UUIM, LCM, LCM/IUIM*, *LCM/UUIM*) produce different types of timbral “cross-fades.” Each mutation takes 2 soundfiles: a *source* and a *target*, and returns a third soundfile, called the *mutant*. The mutation functions operate on the phase/amplitude pair of each frequency band of the source and target spectra. The output of the functions is a phase/amplitude pair for each frequency band in the mutant soundfile. Each phase/amplitude pair in the mutant is some “combination” of the phase/amplitude pairs of the source and target, for the corresponding frequency band. The mutations work on the sign (Contour) or the

magnitude of an interval, or both. They change completely a selected number of bands from the source to the target (Irregular) or partially change all frames (Uniform).

Type
The *Type* box allows you to select between seven quite different mutation functions: *USIM* (Uniform Signed Interval Mutation), *ISIM* (Irregular Signed), *IUIM* (Irregular Unsigned), *UUIM* (Uniform Unsigned), *LCM* (Linear Contour Mutation), and the concatenations *LCM/IUIM* and *LCM/UUIM*. (For specific definitions of the functions, see below. For more information see the two articles cited in the bibliography).

Try starting with the simplest ones, the *USIM* (a simple spectral crossfade) and the *ISIM* (a spectral replacement). These two mutations, unlike the *UUIM* and *LCM*, actually arrive at the source or target, depending on which direction you mutate (that is, you will actually hear the source or target with Ω = 0.0 or Ω = 1.0, respectively). The *IUIM*, *UUIM* and the *LCM* will, with Ω = 1, give an image of the target. These "incomplete mutations" mutate either the sign or the magnitude, but not both of the intervals between the amplitudes of successive spectral bands.

The concatenations (*LCM/IUIM*, *LCM/UUIM*) are "pipes": they apply the second mutation to the output of the first. The concatenations are complementary: for each frame the *LCM* mutates sign while the *IUIM* and *UUIM* mutate magnitude. Different frequency bins are used for each "stage" of the concatenation, so the mutation trajectory can be rather unpredictable.

Mutation Index (Ω)
Each of the mutation functions uses an *index*, called *Ω* (omega), or an Ω –function. This determines the amount of spectral mix, from 0 to 1, between the source and target resulting in the mutant. Ω = 0 results in all source file, Ω = 1 all target. Ω may vary over the course of the mutation. A constant index will result in a sound which is a spectral mix of the source and target. More dynamic sounds are produced with an index function, which changes Ω over time.

Absolute Interval
There are two methods used by the mutation functions to compute intervals between frequency bands: *Absolute* (the default) and *Relative*. You may check or uncheck the Absolute Interval box to get these two methods. If Absolute intervals are used, you may specify an absolute amplitude value between 0.0 and 1.0 for the source and target (Source Abs. Value, Target Abs. Value), from which all intervals will be taken. The choice of values can produce interesting effects, often "centering" the frequencies in which the mutation happens, making the mutations themselves less extreme. If two different values are used, amplitudes will be "transposed" from the source to the target. The use of Absolute intervals rather than Relative will be most noticeable for the *LCM*, *IUIM* and *UUIM*, as well as the concatenated mutations. Low values (around .1 — .2) are a good place to start (note that .1 means 1/10th of the total amplitude of the soundfile's spectra).

If Absolute is unchecked (Relative intervals), each mutation function uses amplitude intervals between successive frames of the spectra, multiplied by Ω , to create the

corresponding frame of the mutant sound. Relative interval mutations will tend to “drift,” often in extreme ways, and the *ISIM* and concatenated mutations may never “arrive.” However, some very interesting sonic results may be produced in this way.

Delta Emphasis
If the mutation uses Relative Intervals (Absolute Intervals unchecked), you can set a value for *Delta Emphasis* (DE). DE allows control over the degree to which successive mutation intervals are emphasized in the resulting mutant. DE values range from -1.0 to 1.0, with the default at 0.0 (no emphasis or de-emphasis). For positive DE values, the current frame’s intervalic characteristics will be emphasized more than the previous mutant frames. For negative values, the current frame will be “damped,” emphasizing the previous information. One way to think about this is as a way of “slowing down” the mutation: a negative DE value will keep the more chaotic mutations from getting “out of control.” A negative DE value will function as a lo-pass filter, averaging the previous spectral frames into the current output. Positive DE values will accentuate the often high-frequency activity of the mutations. Delta Emphasis can be useful in tailoring the relative interval mutations, especially the “incomplete” ones.

Band Persist (Irregular Mutations Only)
Band Persist pertains only to irregular mutations, the *LCM*, *ISIM*, *IUIM* and the concatenations (and not to uniform mutations, the *USIM* and *UUIM*). It will only appear on the screen when irregular mutation is selected. High values for Band Persist (towards 1.0) will produce more “stable” mutations. Low values (towards 0.0) will introduce a kind of frequency pumping at the frame rate. Try changing the number of bands for unusual results.

In irregular mutations, not every frequency band is mutated for each FFT frame. Ω determines the percentage of bands that are mutated for a given frame. If a band is mutated, it completely assumes the particular characteristic (interval sign or magnitude) of the target interval, and retains either the sign or the magnitude of the source interval. For example, the *LCM* takes the sign of the target interval, and “pastes” it onto the magnitude of the source. However, it only does that for (Ω * #-of-bands). The selection of which bands to mutate in irregular mutations is done stochastically, but setting Band Persist high will ensure that once a band is mutated, it will keep being mutated as long as possible. That is why a high value for Band Persist will stabilize these highly unusual mutations, making them a bit more “well-behaved.” A good experiment is to try an irregular mutation (*LCM*, *IUIM*, *ISIM*, the concatenations) with a fixed Ω , and two different values of Band Persist, one high and one low.

Time Scale Target
The form of the mutation functions used in SoundHack require that each soundfile (source, target, mutant) be of equal length. The default technique is to truncate the longer of the two files, producing a mutant which is the length of the shorter file. If Time Scale Target is checked, the target soundfile will be time-stretched or -compressed to be of the same length as the source. Other techniques are of course possible (including windowing and zero-padding), and we encourage other software developers to investigate these.

Theoretical Descriptions of the Mutation Functions
The mutation functions can be classified as follows:

	Uniform	Sign		Magnitude
USIM	yes	yes		yes
ISIM	no	yes	yes	
UUIM	yes	no		yes
IUIM	no	no	yes	
LCM	no	yes	no	

Note that the opposite of “irregular” is “uniform.” Uniform mutations do something to every frequency band. How much they do depends on Ω . Irregular mutations change a given frequency band completely from source to target (sign, magnitude or both), but the number of frequency bands they operate on depends on Ω . Note that there is no *UCM* given here (it wouldn’t make sense).

The *USIM* and the *ISIM* are the simplest mutations, a spectral cross-fade and spectral band-replacement, respectively The *UUIM* and the *IUIM* are two different ways of “pasting” the magnitude differences of the target spectra onto the sign of the source, resulting in a mutant which, when completely mutated, is still some combination of the source and target. The *LCM*, perhaps the most unusual sounding and difficult mutation to control, does the opposite, pasting the signs of the target onto the magnitudes of the source.

The functions are defined below. *S* and *T* are the source and target soundfiles. S_i, T_i are the amplitudes for a given frequency band of the FFT for the *ith* frame of the sound. S_j, T_j are either the amplitudes of the same band in the previous frame (Relative Interval) or some absolute amplitude (Absolute Interval). M_i is the new amplitude of the given frequency band of the current frame of the output sound, M_j is the amplitude for that band in the previous frame of the output sound (Relative Interval), or some absolute amplitude value between 0.0 and 1.0 (Absolute Interval). T_{int}, S_{int} and M_{int} are the signed magnitude intervals between the amplitude of the current frame for a given band, and the amplitude of that band in the previous frame (Relative) or to some fixed amplitude (Absolute) Each of the equations below applies to one frequency band of the source, target and mutant soundfile spectra. In other words, for all of these functions, S_i, T_i, and M_i run from 0 to the number of bands in the FFT.

— and —

• Uniform Signed Interval Magnitude (*USIM*)

$M_i = M_j + (S_{int}) + \Omega \quad * (T_{int} - S_{int})$

• Uniform Unsigned Interval Magnitude (*UUIM*)

$M_i = M_j + S_{sgn} * (S_{mag} + \Omega \quad * |T_{mag} - S_{mag}|)$

— where S_{int} and T_{int} are $(S_{sgn} * S_{mag})$ and

$(T_{sgn} * T_{mag})$ respectively

• Linear Contour Mutation (*LCM*)
$M_i = M_j + T_{sgn} * S_{mag}$ (for mutated intervals)
$M_i = M_j + S_{sgn} * S_{mag}$ (general form for non-mutated intervals)

• Irregular Unsigned Interval Magnitude (*IUIM*)
$M_i = M_j + S_{sgn} * T_{mag}$
(for mutated intervals; non-mutated intervals same as *LCM* above)

• Irregular Signed Interval Magnitude (*ISIM*)
$M_i = M_j + T_{sgn} * T_{mag}$
(for mutated intervals; non-mutated intervals same as *LCM* above)

Notes: 1) In Absolute Interval mutations, M_j, the absolute amplitude to which the new interval is added, is interpolated between the absolute values for source and target, according to the value for Ω . 2) The Irregular mutations are in two forms: one for the case when that particular frequency band is chosen for mutation, one for when it is not.

More Information and Acknowledgements
For more on morphological mutations, including their use in other contexts, please contact Larry Polansky at Dartmouth College, Bregman Electro-Acoustic Music Studio, Music Dept., Hanover, NH 03755, email: larry.polansky@dartmouth.edu. Various students at Dartmouth, including Chris Langmead, Eric Smith, Steve Berkley and Martin McKinney have provided invaluable assistance over the past few years in helping me to formulate both the spectral mutation ideas and accompanying software techniques. Ken Overton, Sergei Kossenko, and Chris Langmead all contributed valuable suggestions to this documentation.

Phase Vocoder

This process allows one to change pitch without changing the length of the soundfile or to change length without changing pitch. It does this by extracting amplitude and phase information for 16-8192 frequency bands with a bank of filters. If time stretching is desired these phase and amplitude envelopes are lengthened (or shortened, for time compression), and then given to a bank of oscillators with corresponding frequencies to the filters. For pitch shifting, the envelopes are untouched, and given to a bank of oscillators with frequencies related by the pitch ratio.

To use the phase vocoder, set the number of "Bands" to the number of filter-oscillator pairs one would like to use. A large number of bands will give one better frequency resolution, a small number of bands will give one better time resolution. The "Filter Window" menu allows one to chose different pre-FFT windows for different filtering characteristics. Only the Hamming, Hanning and Kaiser will give good results (the

others are there only because I wanted to use a single menu throughout the program for all window selection). The "Overlap" setting adjusts the size of the filter window (relative to the number of filter bands) for analysis and synthesis and thus, the sharpness of the filter. A large setting (4x) will give the sharpest filter. A sharper filter will differentiate better between frequencies which are between bands, but responds to amplitude changes slower. Click the "Time Scale" button for time scaling, "Pitch Scale" for pitch scaling. Type the scale factor in the "Scale" box. Click on the word "Scale" (a pop-up menu) to specify time scaling by the length desired, or pitch scaling by equal tempered semitones. Click the "Analyze Only" box to produce Csound compatible pvoc analysis files. If one wants the time expansion factor or the pitch transposition factor to change during processing, click the "Scaling Function" box, and the "Draw Function..." button. This will bring up the Draw Function dialog, which is described below.

“Resynthesis Gating” performs a simple spectral gate which lets only some of the spectral data through. If a band is below the “Minimum Amplitude” it is not let through. “Threshold Under Max.” cuts off all bands which are lower than the threshold below the peak band in a given block of samples. So if the peak band has an amplitude of -7 dB and “Threshold Under Max.” is set to -40 dB, all bands below -47 dB will be cut off.

Spectral Dynamics...

This process performs standard dynamics processing (gating, ducking, expansion, compression) on each spectral band individually. It has individual threshold detection for each band, so that one band could have the dynamics process active, while another is inactive. The process can be limited to affect only a specific frequency range. One can select whether to affect sounds which are above the threshold or sounds which are below the threshold. The threshold level can be set to one value for all bands, or it can be set to a different value for each band by reading in and analyzing a soundfile. This soundfile's amplitude spectrum is used for the thresholds for each band. This is especially useful if there is a sound that one wants to emphasize or deemphasize.

Most controls are self-explanatory. The first popup menu allows you to select the type of process to use; gating/ducking, expansion or compression. The second popup sets the number of filter bands to separate the sound into. 512 is a good compromise for the number of bands at a 44100 sample rate as each band is about 43 Hz apart and the filters used have a (512*2)/44100 or .023 second delay. In other words, a pretty good frequency resolution (provided no partials are closer than 43 Hz) and not too much time smearing.

The "Highest Band" and "Lowest Band" boxes allow one to limit the frequency range affected. The "Gain/Reduction" box allows you to set the amount of gain or reduction for the bands which are past the threshold. For compression and expansion this box becomes the ratio. The compressor and expander hold the highest level steady and affect lower levels (also known as "downward" expansion or compression). “Attack/Decay Time” allows one to set the speed that each triggered band opens or closes. The default is the minimum time for the number of bands used. "Threshold

Level" is where you set the threshold level!

Varispeed...

As a sample rate converter, this is slower and maybe less accurate than the Sound Designer II or Alchemy software, so this function may not be useful to those who own that software. However, SoundHack includes a variable sample rate conversion utility (varispeed). The "Varispeed" box enables this feature. The "Varispeed Function..." button will bring up the Draw Function... dialog, giving one control over a 10 octave varispeed. The "Quality" buttons give one control over the size of smoothing filter used, and the resultant quality of interpolation/decimation. The "Vary by Scale" and "Vary by Pitch" buttons allow one to draw a curve for either pitch or scaling factor.

Draw Function...

This dialog box allows for the creation of control functions. You can draw in the function window, set upper and lower limits for the function, clear, invert, reverse, smooth or shift the function, or use simple wave forms (up to 100 cycles) as a control function. If the function window is cleared (with a constant value through the center of the screen), the wave form icon buttons will draw the wave form as the new control function. However, if the function window is not cleared, the wave form icon buttons will modulate the existing control function. There is no facility for selecting, copy, paste or cut. One can read or write control functions as soundfiles. The "Time:" refers to the time in the input soundfile and the other legend ("Semitones:") is updated depending on how the control function is applied.

• CONTROL MENU

Show Output

This will bring up a window to show the sound whenever SoundHack writes sound to the output soundfile (except during file copying and normalization). This slows down processing somewhat.

Show Spectrum

This will bring up a window to show the spectral data in all spectral operations (most everything but varispeed, which I do in the time domain)

Pause Process

This allows you to pause during a long process in case you need to use the Mac, but you don't want to start the processing over. If you are running a convolution, it sometimes takes a while to pause (up to 3 minutes).

Continue Process

This will resume processing where you left off.

Stop Process

This will kill your process and close the output soundfile.

•SHAREWARE INFO

Version 0.74 of SoundHack is shareware/artware, if you use it and would like to become a registered user, send me either $30.00 or some music or art that you have created. My address is: Tom Erbe, 25908 Tournament Road #276, Valencia, CA 91355. As a registered user you are entitled to email help (tre@music.calarts.edu). Updates to SoundHack are available through anonymous ftp to music.calarts.edu (internet address: 198.182.157.104) in directory /pub/soundhack.

You can buy the commercial version of SoundHack (version 0.80) from Frog Peak Music, PO Box 5036, Hanover, NH 03755 for $50.00 ($15.00 for registered users). The commercial version comes with the mutation functions enabled and a native Power Macintosh binary.

•FUTURE PLANS

1.0 SoundHack is finished. All copies of the source code are destroyed. Tom and Betsy take a holiday.

.90 Batching is added.

.75/.85 RIFF loops are added, MPEG-2 audio is supported.

•BUG FIXES AND REVISIONS

.80 Split into shareware (0.74) and commercial (0.80) versions. Added Mutation functions. Ported to the Power Macintosh.

.74 Fixed early ending sample rate conversion problem. Fixed the spectral display. Added attack/decay time to spectral dynamics. A moving binaural effect is added. Allow raw data files to have text packmode and text files to have raw packmodes. Fixed various GUI complaints (0.743) Double-clicked soundfiles now open correctly in all versions.

.73 Fixed varispeed clicking for the third time. Fixed the output soundfile dialog. Ported the code from Think C to Codewarrior. Spectral display is now broken and needs to be redone (disabled for the present).

.72 Fixed 16 bit Microsoft RIFF/WAVE file bug.

.71 Fixed a minor bug which caused spectral dynamics crashing. Also fixed a typo (thank you Phil Burk).

.70 Added spectral dynamics process. Added AIFC and Microsoft WAVE file support. Added 4-bit ADPCM. Opens double-clicked documents. More types show up in "Open" menu item. Fixed stupid error about number of bands in the phase vocoder and the spectral dynamics processor. It used to refer to the number of FFT points used, it now correctly refers to the number of filter bands used.

.68 Added import and export of SND resources. Limited soundfile playback to AIFF and AIFC until Apple comes up with a stable sound manager. Sound Manager 3.0 is required for playback of 16-bit files.Added a limited spectral gate to the phase vocoder. Fixed old bug in pitch shifting.

.67 Added soundfile playback.

.66 Added windowing selection for both convolution and the phase vocoder. Added "Vary by Pitch" /"Vary by Scale" selection in varispeed.

.65 Fixed crashing at the end of long varispeed calculations, cleaned up memory allocation (especially for convolution), added ring modulation (spectral convolution), cleaned up dialog annoyances. Made output window resizable, and made it white on black (like a scope). Changed software to shareware, too few people sent me music when software was musicware. Added bibliography screen.

.64 Added varispeed processing, added many things to the function dialog.

.63 Added TEXT soundfile format. Added interpolation, sinc and number of cycles to function dialog. Added ramp enveloping of impulse to convolution. Added popup menus.

.62 Added function window for multirate phase vocoder.

.61 Sped up pitch transposition significantly, allowed pitch transposition by semitone and time warping by desired length.

.60 Added the phase vocoder, phase vocoder Csound analysis and show output.

Fixed problem with normalization after moving convolution.

.59 Added moving convolution.

.58 Fixed problems with 8-bit AIFF files.

.57 AIFF files used to read everything from SSND chunk to EOF, now only SSND chunk is read. The Binaural processor often destroyed AIFF headers, I think I have this fixed. DSP Designer files are now written as well as read (though I have no way of checking, please DSP Designer users, give me feedback).

.56 Filter files are now closed properly after convolution.

.55 Normalization feature disabled Binaural filtering, now fixed. Filter sensitivity added to Convolve.

.54 Normalize after processing feature added. Dialogs adjusted for small monitors.

•BIBLIOGRAPHY

Apple Computer, Inc., *Inside Macintosh, Volume 1-6,* Addison Wesley, Reading, Mass., 1985-91.

Begault, Durand R., *Control of Auditory Distance,* P.H.D. dissertation, University of California, San Diego, 1987.

Begault, Durand R., *3-D sound for virtual reality and multimedia,*Academic Press Professional, Cambridge, MA, 1994.

Blauert, J., *Spatial Hearing,* MIT Press, Cambridge, Mass., 1983.

Dolson, Mark, "The Phase Vocoder: A Tutorial", *Computer Music Journal 10:4,* 1986.

Ellis, Dan, "pvanal.c", part of the Csound distribution, MIT, 1991.

Gordon, J. W. and Strawn, J., "An Introduction to the Phase Vocoder", *Digital Audio Signal Processing: An Anthology,* editor J. Strawn, Kaufmann, Los Altos, Calif., 1985.

Mark, David and Reed, Cartwright, *Macintosh C Programming PRIMER, Volume I,* Addison Wesley, Reading, Mass., 1989.

Moore, F. Richard *Elements of Computer Music,* Prentice Hall, Englewood Cliffs, NJ, 1990.

Polansky, L. “More on Morphological Mutations: Recent Techniques and Developments,”*Proceedings of the ICMC.*, San Jose, 1992.

Polansky, L. and McKinney, M. "Morphological Mutation Functions: Applications to Motivic Transformations and to a New Class of Cross-Synthesis Techniques." *Proceedings of the ICMC.* Montreal, 1991.

Reed, C. E. and Passin, T. B., *Signal Processing in C,* John Wiley, New York, NY, 1992.

Vaseghi, S. and Frayling-Cork, R., "Restoration of Old Gramophone Recordings", *Journal of the AES,* 40:10, 1992.