WaveBeast

Jan Krutisch

COLLABORATORS			
	TITLE : WaveBeast		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Jan Krutisch	April 17, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 WaveBeast **1**

1.1 welcome to wavebeast . . . 1
1.2 news . . . 2
1.3 introduction . . . 3
1.4 credits . . . 3
1.5 requirements . . . 4
1.6 installation . . . 5
1.7 usage . . . 5
1.8 usage.subtractive.synthesis . . . 5
1.9 usage.settings . . . 6
1.10 usage.bottom . . . 7
1.11 usage.oscillators . . . 7
1.12 usage.filters . . . 9
1.13 usage.envelopes . . . 9
1.14 usage.lfos . . . 10
1.15 usage.effects . . . 10
1.16 usage.effects.reverb . . . 10
1.17 usage.effects.fuzz . . . 11
1.18 usage.effects.delay . . . 11
1.19 usage.sequencer . . . 12
1.20 usage.calculate . . . 13
1.21 usage.examples . . . 14
1.22 examples.base . . . 14
1.23 bugs . . . 14
1.24 future . . . 15
1.25 history . . . 15
1.26 acknowledgements . . . 17

Chapter 1

WaveBeast

1.1 welcome to wavebeast

```
                                          W·A·V·E·B·E·A·S·T

          idea.code.............................skyphos.artwork
          code.design.docs................halfbyte.pandemon!um
          version.....................................0.51.beta!

                             table.of.contents

READ!>>
                  news
                ..............................whats up ?

                  introduction
                ................what the hell is this ?

                  requirements
                ............what do i need to proceed ?

                  installation
                ...................what to copy where ?

                  usage
                ....................how to get a tone ?

                  bugs
                .........................are there any ?

                  future
                ...................what's coming next ?

                  history
                ............what's been there before ?

                  credits
                .........................who did what ?

                  acknowledgements
                ............am i greeted, and why not ?
```

NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE

!!! PLEASE RESAVE ALL YOUR PATCHES !!!
From this version on, WB will save a header with the Patches.
I will add a sanity check in one of the next versions, so please
load in all your patches and save them again. This might be a good
idea anyway, since you might need to adjust the Pitch Env Int Value
if you used the Pitch Envelope.

NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE NOTE

1.2 news

wave.news

read the latest development news and other necessities here, without
having to go through the whole guide.

20.9.97 - We got prefs.

Save your desired paths! Call Muiprefs from within. Smoke
some Weed! Be happy. I also fiddeled with skyphos' way
of normalizing samples. (Configurable Clip/Normailize/nuffin).
And I improved the pitch env handling. Stay tuned!

BTW: The website is open in full glory. No heavy gfx,
just plain support for the funkiest thing on your amiga !

http://wavebeast.home.pages.de

16.9.97 - Wooohaaa...16 SampleSave and more !

Now WaveBeast can save 16 bit AIFF samples. Needed a whole day
to figure out that EXTENDED IEEE stuff...I included some stuff
(c) Apple direct from the amisox sources. Works like a charm.
I also added the long awaited "Save/Load patch" stuff in
a very extensible manner. Wow, this gets on !

13.9.97 - Bugs killed, status line added, busy handling added !

As a first step in development, I killed the "Sequencer eats Mem
like pasta" bug. (Do we have to care for them allocated ? I guess
not, eh ?). I also added a status line that shows important news
(like "Calculating Sample" and stuff...). And I also added busy
handling, which means that WaveBeast is blocked while calculating!

Keep up to date. I will start to do a support web page soon !

(try http://wavebeast.home.pages.de)

13.9.97 - Jan "Halfbyte" Krutisch takes over development !

Since Skyphos doesn't seem to be active on the amy anymore (pc lamer,

he is), I took over the whole development. I can't guarantee for nuffin' tho, since I have enough projects on my back. AND I have to understand skyphos source first !

1.3 introduction

wave.introduction

wave is a very powerful software synthesizer based on subtractive synthesis. It has numerous features, listed here:

- .two oscillators doing sine.saw.square.pulse.noise.samples
- .pulsewidth modulation
- .additional noise generator
- .oscillator sync
- .ring modulation
- .two 2 pole filters with resonance 12db.lowpass.highpass.bandpass
- .four graphical envelopes for amp.pitch.filter1.filter2
- .five lfo's for pulsewidth.amp.pitch.filter
- .great dsp effects like fuzz.reverb.delay
- .step sequencer with 64steps.slide.accent.notelenght.transpose
- .variable output samplingrate.wordlength.note.volume
- .direct output via ahi.

the sounds are calculated non.realtime and can then be played via ahi or saved in the following formats:

- .8 bit raw
- .16 bit AIFF mono

1.4 credits

wave.credits

synthesis.code.idea.inital releases.....marco <skyphos> trush
code.gui.design.docs.website.....jan <halfbyte> krutisch

the used algorhythms of soundsynthesis are snatched together from various sources (mainly usenet) by skyphos.

the AIFF saveroutine contains a code fragment from the amisox sources. copyright below:

```
SNIP ------8<------8<------8<------8<------8<------8<------8<------8<

/*
 * C O N V E R T   F R O M   I E E E   E X T E N D E D
 */

/*
 * Copyright (C) 1988-1991 Apple Computer, Inc.
```

```
 * All rights reserved.
 *
 * Machine-independent I/O routines for IEEE floating-point numbers.
 *
 * NaN's and infinities are converted to HUGE_VAL or HUGE, which
 * happens to be infinity on IEEE machines.  Unfortunately, it is
 * impossible to preserve NaN's in a machine-independent way.
 * Infinities are, however, preserved on IEEE machines.
 *
 * These routines have been tested on the following machines:
 *    Apple Macintosh, MPW 3.1 C compiler
 *    Apple Macintosh, THINK C compiler
 *    Silicon Graphics IRIS, MIPS compiler
 *    Cray X/MP and Y/MP
 *    Digital Equipment VAX
 *
 *
 * Implemented by Malcolm Slaney and Ken Turkowski.
 *
 * Malcolm Slaney contributions during 1988-1990 include big- and little-
 * endian file I/O, conversion to and from Motorola's extended 80-bit
 * floating-point format, and conversions to and from IEEE single-
 * precision floating-point format.
 *
 * In 1991, Ken Turkowski implemented the conversions to and from
 * IEEE double-precision format, added more precision to the extended
 * conversions, and accommodated conversions involving +/- infinity,
 * NaN's, and denormalized numbers.
 */

SNIP ------8<------8<------8<------8<------8<------8<------8<------8<
```

1.5 requirements

```
                            wave.requirements


    absolutely.required

        .amiga.os3
        .mui 3.x
        .1 free mb of ram

    definitely.proposed

        .030 or 040 or 060 processor for shorter calculation times
        .fpu
        .lots of ram
        .harddisk
        .ahi for direct playing
        .creativity
```

1.6 installation

wave.installation

well, thats simple. Just copy the main program to where ever you like.
note that mui 3.x is absolutely required and ahi makes much sense.
please refer to the documentation of mui and ahi how to install these.

1.7 usage

wave.usage.introduction

the usage of wave is traightforward and very close to using an old
analogue synthesizer.
for all those that never used such a great machine, i will give a
short
intruduction on subtractive synthesis
. if you are familar with
that, you should skip this part.

table.of.contents

oscillators

filters

envelopes

lfo's

effects

sequencer

calculate

settings

bottom

1.8 usage.subtractive.synthesis

subtractive.synthesis

the path in subtractive synthesis is always the same: take a waveform
with a wide spectrum of frequencies (such as a saw wave or a
squarewave) and cut out the parts you like most. this is done using
so.called filters. these filters can have many characteristics, but
there are three of them that can be found in nearly all good
synthies.

the first one (also the most used) is the lowpass filter. it
cuts off all frequencies above the cutoff frequency. this makes the
sound considerably "smoother".
the second is the highpass filter. as you may guess, it cuts out all
frequencies below the cutoff frequency. this makes
the sound kind of "hollow".
The third in the row is the bandpass. it just allows a band of
frequencies around the cutoff frequency to pass. it's sound heavily
relies on the cutoff frequency.

after the filter we normally have a so called "amplifier" that
controls the volume of the sound.

of course thats not too great until now, having basic tones without
any movement. while aphex twin could make whole songs out of a few
static tones, we want something more lively. to reach that goal,
we can "modulate" the basic parameters of a sound. these include the
oscillators' pitch (giving that "vibrato" sound), the cutoff
frequency of the filter (giving movement to the sound's "color") and
the amplifier (allows for a slow attack and.or a slow release). this
modulation is done by two basic mechanisms (referred as control
sources), envelopes and lfo's (low frequency oscillators). the
envelopes are a possibility to control a sound parameter over the
whole time you trigger that sound. just take a look at wave's
envelopes, they are quite straightforward. you can use env's (short
form) for punchy bass sounds, by opening and closing the filter
very fast at the start. or you create a long sweeping sound, by
letting the envelope start with a very long attack time.
the lfo's are different. a lfo is an oscillator of it's own right,
but oscillating with a fairy low frequency. if we use this low
frequency to gently modulate (for example) the pitch, we get a decent
vibrato, or, if we raise the amplitude of that lfo, a cool siren.
if we modulate the filter, we get a kind of "wah-wah" sound.
modulating the amplifier results in a kind of "tremolo" effect.

using both of the control sources, we can generate very lively,
moving sounds, that never sound static or something.
Just experiment!

this should be enough knowlegde for the first experiments.
now read on to get into the tricky world of using wave.

1.9 usage.settings

wave.usage.settings

popasl.patch.path
popasl.seqences.path
popasl.samples.path

here you can set your favourite paths to your patches, sequences
and samples. (samples not working yet !)

button.open.MUI.prefs

by pressing this button you can open the MUIprefs for WaveBeast,
where you can eg. alter the screen that WaveBeast opens on.

button.save.prefs

this saves your current prefs for wavebeast as "wavebeast.cfg" in
the same directory as the wavebeast executable lies.

button.load.prefs

this loads the last saved prefs. use this if you totally messed
up the preferences.

1.10 usage.bottom

wave.usage.bottom

button.load

load a patch. (all synth data)

button.save

save a patch. (all synth data)

button.quit

gently quits wavelab. but what have you expected ?

action.calculate

start the calculating the current patch.

action.play

once calculated, you can play the sample via AHI output.

1.11 usage.oscillators

wave.usage.oscillators

wave features two independent oscillators with exactly the same
parameter set, plus an independent noise generator.

parameter.waveform

choose on of the following waveforms:

.sine.....no harmonics in there. use it for the bass down low, or
for great effects with the fuzz.
.triangle.....nearly as flat as the sine, this one is great for jungle
basses.
.up.saw.....a cool wave for extensive filtering due to the great
harmonic content. try detuning for instant rave.
.down saw.....as we germans would say: the same in green.
.pulse.....this one is also full of harmonics. you can alter the
harmonic content with the PW knob. this knob alters
the pulsewidth (the ratio bewteen high and low signals).
.square.....this is a pulse with equal high and low signals.
.sharp square..huh, dunno!
.noise.....white noise is great for sweeping effects.

parameter.volume

this parameter regulates the output volume of the corresponding
oscillator. note that the volume is set behind the filters.

parameter.coarse

use this parameter to detune the oscillator in halftones (relative to
the original pitch). a detune of an octave on the second oscillator
always sounds great.

parameter.fine

with this parameter you can fine detune the oscillator (measured in
cents). detuning both oscillators just a bit (one up, one down) gives
a great chorusing effect.

parameter.sync

by selecting this switch you sync oscillator two hard to oscillator one.
this means that everytime osc one has finished one period, oscillator
two is restarted. when detuning osc two, this creates cool creeping
sounds, caused by the disturbance of osc two.

parameter.ringmodulation

this switch switches on ring modulation. just experiment with it (try
to detune and pitch modulate osc two), it’s too hard to explain. :)

parameter.pw

this parameter sets the pulsewidth of all oscillators set to the pulse
waveform. see above for deeper explanation.

parameter.noise

--out of function--

1.12 usage.filters

wave.usage.filters

wave featurs two identical filters. the parameters go as follows:

parameter.cutoff

the cutoff frequency of the filter.

parameter.resonance

the resonance of the filter. resonance is done by feeding a bit of the filter output back into the input, resulting in a ..um.. resonance in the frequencies around the cutoff frequency. if resonance is set to maximum, the filter starts to make an own sine sound in exactly the cutoff frequency. nice for effects. set resonance reasonably high to make sweeps interesting. a filter without resonace often sounds boring (hello, korg people).

parameter.env.int

sets the amount of influence that the envelope has to the filter.

parameter.type

choose between low-, high and bandpass. if you don't know what this is about, please read my
introduction to subtractive sythesis
.

please note that the cutoff frequency is also modulated by an envelope and an lfo.

1.13 usage.envelopes

wave.usage.envelopes

the envelopes are used to give a sound a kind of contour, by modulating the oscillator pitch, the amplifier volume and the filter cutoff.

the usage of the envelopes is straight forward. simply drag the small rectangles around and look how the envelope is formed. to fine-tune the values, use the numeric buttons.

every envelope consists of 8 (7 for the amp) parameters. the parameters labelled T1-4 are representing the times the envelope needs to get to the corresponding levels (marked L1-4).

note that the pitch envelope has also negative levels (to slide down below the basic pitch), while amp and filter just have positive

```
levels.
```

1.14 usage.lfos

```
                            wave.usage.lfos

wave features five independent lfo's. you can modulate:

    .pulsewidth (resulting in a chorusing sound)
    .oscillator pitch, called vco (resulting in a vibrato)
    .amplifier volume, called vca (resulting in a tremolo)
    .filter one cutoff, called vcf 1 (resulting in a kind of wah-wah)
    .filter two cutoff, called vcf 2

they all have identical parameters:

                            parameter.depth

depth controls how deep the destination is modulated by the lfo.

                            parameter.rate

rate specifies the frequency of the lfo.
```

1.15 usage.effects

```
                                        wave.usage.effects

until now, wave supports three effects that can all be used at the
same time:

.
            reverb
            (room, hall)
.
            fuzz
            (overdrive, distortion)
.
            delay
            (echo)
```

1.16 usage.effects.reverb

wave.usage.effects.reverb

the reverb section has two parameters:

parameter.feedback

by adjusting feedback, you have virtual control of the size of the room you want to simulate. be aware that large feedback values can turn your sound into a big mess without recognisable sounds.

parameter.damp

with this parameter you can specify the acoustic characteristics of the room you want to simulate. when setting to small values, you simulate a room with very reflective walls, resulting in a long, bright reverb. when setting to large values, the room is damped (eg. with soft walls), resulting in a shorter, muffy sound.

1.17 usage.effects.fuzz

wave.usage.effects.fuzz

the fuzz effect tries to simulate the sound of distortion and overdrive, useful to create this screaming 303 lines. the parameter set is rather simple:

parameter.fuzz

simply sets the grade of distortion. note that this algorhythm is very rude, so turn down the volume to a decent level before testing your fuzzed sounds, otherwise you could blow your tweeters.

1.18 usage.effects.delay

wave.usage.effects.delay

another must for creating cool acid lines, the delay gives you a kind of echo, like the one you are experiencing in the mountains, when calling "hello, echo!" into the landscape, with the sound coming back to you a few moments later. the parameter set:

parameter.time

specifies the delay time in milliseconds. setting this to large values gives cool echo effects, while short times give you an instant "we are the rrrrobots" feeling.

parameter.feedback

by feeding the output back into the input, you can create echos that are coming back more than once. setting this to high values

creates interesting effects with infinite echos. but, as with delay,
the sound tends to hide behind the feedback, if misused.

1.19 usage.sequencer

wave.usage.sequencer

the sequencer is activated by hitting the checkmark "sequencer" in
the "calculate" group on the first page. a new window is opened,
containg all elements of a full fledged harware sequencer.
you can load and save the whole sequencer setup, using the "load"
and "save" buttons down there.
lets take a look at the other global parameters:

parameter.bpm

sets the tempo of the sequencer in beats per minute.

parameter.speed

sets the "resolution" of the sequencer. 6 means 4 steps/bar,
3 means 8 steps/bar.

parameter.steps

sets the total number of steps in the sequencer. up to 64 steps
are allowed.

parameter.transpose

sets the transposition of the sequencer. allows for easy tuning.

parameter.slide.time

sets the time needed to slide from one note to another (if the
slide flag is set).

---+---

all other buttons are referring to the actual step. just select a
step in the listview and you have access to the following parameters:

parameter.note

guess what. select a note, ranging from C-1 to C-11.

parameter.velocity

--out of function--

parameter.notelenght

this sets the notelenght, relative to the lenght of one step. if

set to 128, the note will be hold.

parameter.contoller

--out of function--

parameter.value

--out of function--

parameter.accent

--out of function--

parameter.slide

this checkmark activates the noteslide. the pitch will continously slide from the last note to the actual one.

parameter.portamento

when portamento is set, the note will not be triggered, but instead only the pitch will be set, so the last note will "ported" to the new pitch. should be obvious to protracker users (command 3xx). if slide is set, too, the pitch will slided with slide speed. if not set, it will be set instantly to the new value.

1.20 usage.calculate

wave.usage.calculate

in this section you can specify all parameters needed for the calculation and output. the parameters go as follows:

parameter.sequencer

open and close the sequencer window with this button. if activated, wave will calculate the whole sequence, otherwise it will just calculate one note.

parameter.note

if you calculate a single note, you can specify the note here.

parameter.volume

set the output volume of the calculated sample.

parameter.sampling.frequency

set the sampling frequency the calculated sample will have.

parameter.samplelenght

```
when calculating a single note, you can specify the samplelenght
here.

                          parameter.filename

set the filename for the calculated sample here.

                         action.save.calculate

saves the sample in 8 bit raw mode.
```

1.21 usage.examples

```
                                              wave.usage.examples

to give the subtractivly unexperienced user something to start with,
we collected some examples.

btw: we just give you the parameters that differ from the default.

the base in the switch:
            a 909 like kickdrum
```

1.22 examples.base

```
                     wave.usage.examples.basedrum

vco1.waveform:  sine
vco1.level:     127

vcf1.cutoff:    around 50 (experiment!)
vcf1.resonance: 255

vca.envelope:   T1:0    T2:16   T3:0    T4:0
                L1:255  L2:0    L2:0
pitch.envelope: T1:0    T2:4    T3:0    T4:0
                L1:127  L2:0    L3:0    L4:0
vcf1.envelope:  T1:0    T2:12   T3:0    T4:0
                L1:255  L2:0    L3:0    L4:0

off you go!
```

1.23 bugs

```
                                 wave.bugs

Although no programmer likes to admit that his programs have bugs,
```

most of them have to do it. since wave is still in a very early
beta phase, it is most likely that it is riddled with bugs. the
following things are known to bugger us still:

.wave chrashes randomly on any machine for a totally unkown
reason. no chrash is reproducable though, which makes
debugging even harder.

.DO NOT DO THE FOLLOWING: calculate a single sample. switch
on the sequencer. play the sample. bang!

.wave starts throwing around enforcer hits when playing
via ahi. I'm currently investigating if this is my
or Bloms fault.

1.24 future

wave.future

the future for wavebeast is not that bright. since sky stopped
development, i took over. but my time is limited. for the next few
releases (still beta...) i will do all that neccessary cosmetic
changes. only then i can go back to the roots and try to implement
something more trickier, like the following things:

.implementation of all unimplemented features such as:
.noise generator
.velocity, controllers and accent in the step editor
.better usability by exchanging the few string gadgets by knobs.
.intensity regulation for envelopes (partially done).
.more lfo waveforms.
.different sample outputs (16/8 bit, 8svx, aiff, riff, partially
done).
.groove option for the sequencer to allow instant shuffle.

1.25 history

wave.history

! this history starts at the dev.takeover by halfbyte !

symbolics

FIX - Bugfix
NEW - Added new feature.
CHG - Changed feature.

V0.53 beta

```
    FIX - Both Sync and FM Settings didn't appear in the savefile.

V0.52 beta

    NEW - Env Intensity for Pitch. (Do we need one for Amps ? Guess not)
    CHG - Made clipping configurable. Clipping/Normalize/Nuffin.
    CHG - Patchsaving now saves a header. Please Resave all patches NOW,
    since I want to add a sanity check (so that you can't load in
    your startup-sequence).


V0.51 beta

    NEW - Settings ! Save your most liked paths. Path handling for
    samples is NYI, though. But it works like a charm for the
    Sequences and Patches. You can even call the MUIPrefs from
    that page.
    FIX - I lost some bytes by not freeing temp. memory while parsing
    the patchfiles (and settings). what did I say ? Never use strdup()!

V0.49 beta

    FIX - There was a bug in the loadPatch routine, giving unlimited delay
    feedback if feedback>0.
    CHG - sky clipped to whole sample when it was too loud. I implemented
    a normalize function instead. I will make it configurable.


V0.47 beta (note: been a bit fast with the revisions...fixed)

    FIX - Mungwall hit while saving/loading Patch (double FreeAsl).
    NEW - Added CycleGadget for Sample-Save-Formats.
    NEW - Implemented AIFF16 Sample Save.
    FIX - Wrong way of calculating the SampleFreq for AIFF storage.

V0.46 beta

    NEW - patch storage ! i invented a simple and very reliable way of
    saveing and loading patches. it might not be extraordinarily fast,
    but who cares, anyway :-P. I hope it works on other machines, too !

V0.39 beta

    CHG - moved play and calc buttons to bottomline.
    NEW - added "eg intensity" knob to vcf.
    FIX - calc-busy window can be closed by ESC now.


V0.14 beta

    CHG - reconstructed the calculation process. now i open a small window
    with a cancel button so that you can break the calculation process.
    still, asynchronous calculation is the target !

    CHG - the play button is deactivated if no correct sample is available
    for playing.
```

```
    FIX - the filerequesters are now opened on the correct screen.


V0.2 beta

    FIX - the longstanding sequencer bug (eated mem like pasta) has
    been fixed using con- and destructor hooks for the listview. there's
    got to be a better way, tho.

    CHG - the whole application is blocked during sample calculation and
    playing. the idea is to do both things asynchronous, but i have to
    check a few things, before i can do it. also a neat window with
    progress bar and cancel button would be nice.

    NEW - added a status line to the main gui. now you always know whats
    up. i needed this thing. most important if calculating will be done
    asynchronous one day...
```

1.26 acknowledgements

```
                          wave.acknowledgements

    .a biiiiiig shoutout to all the music guys at #amigascne for
     heavy gamma/beta testing on wave. thanks for keeping the core
     and not roaming too much about the many chrashes :)

    .another one goes out to martin "leviticus" blom for his
     AHI --- "god bless" --- system. due to this, wave runs with
     nearly all known soundcards. grrreat.

    .finally we like to thank the whole amiga community for the
     endless support on a "dead can dance" system :)
```