Sounder 1.0: Digitalized Sound Player for Windows 3.0

Sounder allows you to play digitalized sound files recorded in the popular Macintosh sound format on your PC. Better yet, no sound board is required, although a 10 MHz or faster machine is. This program was inspired by programs like REmac and Talk that operate on the principle of speeding up the system clock to a ridiculous speed, turning off everything else, and faking dynamic modulation with the imperfect dynamics of the PC speaker system and the human ear. For such a hack, the sound is actually pretty good, although the quality depends on the actual sound file and also may vary from system to system and speaker to speaker.

Sound files can be gotten via anonymous FTP from terminator.cc.umich.edu, sumex.stanford.edu (these are Mac files, so you'll need access to BinHex and Unstuffit as well as a Mac and something called Apple File Exchange), and other sites. Any Mac-format file should work. The newsgroup alt.tv.simpsons seems to generate a few sound bites from recent episodes now and then, too.

The format of Mac sound files is simply a string of bytes, each of which represents the volume at a given sampling time. Most Mac sound files are recorded at either 11 or 22 KHz, and Sounder can handle these as well as 7.33 and 5.5 KHz files. The de facto extension for these files is .SOU, which I see no reason to change, but...

Since Mac sound files by default carry no frequency or other information (at least in their PC incarnation), it is necessary to include this information on the command line when using Sounder. The Program Manager can handle this task very well, but to make life easier in the future, I propose an ad hoc file format, the .SND format, described later. When a .SND file is played by Sounder, frequency and such information internal to the file is used. It is relatively easy to convert a .SOU to a .SND file. Also, the internal .SND parameters can be overridden by passing command-line arguments. A DSOUND sound resource format is also described later.

Disk Contents

You should have the following files; if not, download a complete version of the archive:

SOUNDER.EXE	Program to play sounds
SOUNDER.WRI	This file
DSOUND.DLL	Digital sound driver
DSOUND.LIB	See DSOUND.DLL section below
DSOUND.H	Ditto.
BEAM.SOU	Sample sound file ("Beam me up, Scotty")
20TH_CEN.SND	Sample .SND file (20th Century Fox intro)

Using Sounder

To use Sounder, copy DSOUND.DLL into your Windows system directory (\WIN\SYSTEM, usually) and SOUNDER.EXE to somewhere on the PATH. I use a \SOUND directory for sound files and SOUNDER.EXE. If you're like me and most Mac users, you'll probably collect several megabytes of The Simpsons, Star Trek, Mission Impossible, and other oldies.

You should then associate all *.SND and *.SOU files with Sounder so that you can double-click on the sounds to play them. Do this by either hacking the WIN.INI file or by selecting such a file in the File Manager and choosing File Associate... to associate them with SOUNDER.EXE. Sound files can then be dragged into a Program Manager group where they'll assume the icon of Sounder (a bit stream going into a speaker, and a waveform coming out). In the Properties... dialog box, command line parameters specifyin the format of the sound file should be included, as described below.

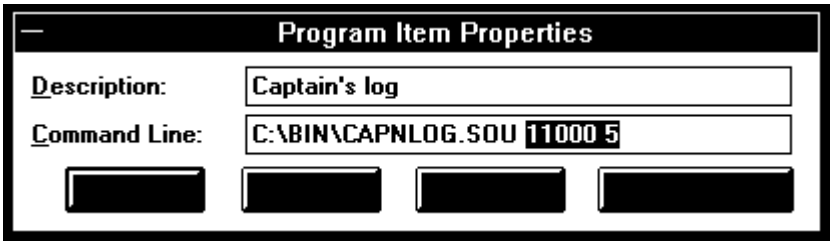
Sounder is invoked with the following command-line syntax:

SOUNDER *sound-file* [S] *frequency volume shift sample_type*

where *sound-file* is a .SOU or .SND file, S is an optional Save flag (described later),*frequency* is the desired playback frequency in Hertz (default: 22000), *volume* is the playback volume parameter (default: 10), *shift* is a value to be added to each sample value before being played (default: 4), and *sample_type* is the type of sample data (currently ignored, but should be 0 for future compatibility if specified). If a .SND file is specified and 'S' (or 's') is specified after the filename and before the other parameters, the settings specified will be saved in the file; no sound will play. Fewer than 4 parameters can be passed, but they'll be assigned in the order above. That is, if you assign a value to a parameter, all preceeding parameters must be passed as well to act as "place holders". *All* parameters should be passed if the S option is included, otherwise default values will be used.

Instead of spaces, commas and tabs are valid delimiters.

If you have placed SOUNDER on the PATH and associated .SOU and .SND files with it, you can omit SOUNDER from the command line. This will be the case with sound files dragged into the Program Manager. It's easiest to drag the sound files into a Sound group and simply add the frequency and volume parameters in the Command line field of the Properties... dialog box:



Sounder has no window, and the only non-error feedback is that the system pauses while the sound plays.

Note: Because of the way Windows parses the run= statements in WIN.INI, a statement like

run=20th_cen.sou 11000

causes the sound to be played at the default 22000 Hz and a file named "11000" to be run (with an error, of course). This is probably not what you want. I suggest creating and using .SND files instead:

run=20th_cen.snd

This will allow you to have a start-up sound.

Errors:

Sounder will complain if:

- The wrong version of DSOUND.DLL is detected
- It can't find the sound file specified, or the file cannot be opened
- There was no sound file specified
- There is not enough memory to load the sound file (all of it at once, unfontunately for you real-moders out there)
- Any other file-related errors it catches spring up.

Getting the Paramters Right:

If the sound file seems to be playing too slowly, increase the frequency; likewise, if it's played too quickly, decrease the frequency. Although any frequency can be passed, currently sounds are played only at 22000, 11000, 7333 and 5500 Hz. The closest of these to the specified frequency is used.

If there is a lot of noise, "static," or a high-pitched whine, try adjusting the shift value. 4 is default, but small variations do make a difference on some recordings.

The playback volume can be changed with the volume parameter, but there is an upper limit to how loud a sound file can be played before the quality of the sound decays (a limitation of the PC speaker). This depends on the sound file and could be from 30 to 100. There is no lower limit, except that at 0 nothing will be heard except background noise.

Converting a .SOU (parameterless sound) file to a .SND (embedded parameter) file

The 'S' (Save) option described above may be used to create a .SND file. Simply rename or copy a .SOU file to have the .SND extension, then run Sounder with the 'S' options and the desired frequency, volume, shift, and sample-size. This can be done with the File Run... option in the Program manager; for example:

could be used as the Command line to save the settings in BEAM.SND after renaming BEAM.SOU to BEAM.SND, assuming *.SND files are associated with Sounder as they should be.

A very small amount of sound information will be overwritten using this approach, but it'll be too slight to notice. I plan on having a better .SOU to .SND conversion utility later, if need be...

Bad News:

Sounder won't work under 386 Enhanced mode. I have a 386sx, so this isn't jealousy. Nor do I work for Harris or AMD. Sounder needs to twiddle with the hardware of your machine, and Enhanced mode doesn't let it get close enough and operate fast enough. Windows also switches stacks at very bad times, which isn't too nice. Sounder does, though, work fine in Standard and Real mode, at least as far as I know. If someone can tell me what *really* happens with an INT 8h interrupt callback in Enhanced mode, or send me some info on DPMI if this is what I need, I'll try to fix this limitation. Actually, I've come to like Standard mode a lot--my hard disk seems more content, and I run very few DOS apps.

P.S. With that, the *adventurous* may wish to try Sounder on a recently-backed-up Enhanced mode setup and let me know if you find a configuration that works. Warning: Unrecoverable errors are *very* unrecoverable when all interrupts are turned off!

Ugly News:

Sounder stops everything dead in its tracks while the sound plays. No mouse, no communication downloads, no nothing. The floppy light stays on. The wait cursor stays on because Windows *can't* turn it off after loading the file. 22 KHz is fast, too fast to let things like these happen. Sounder needs to time things very precisely, and other interrupts would result in clicks and clucks in the speaker. Note that the system time also freezes; I could fix this if it were really a problem.

Future Attractions:

What's up with DSOUND.DLL? I'd love to see something like SoundMaster for the Mac under Windows. I may even write it myself. But don't hold your breath; school's about to restart and WinCrystal isn't done yet (although now I can have neat sounds in it). DSOUND.DLL has all the playback code in it, so if anyone wants to write a shareware/PD Windows sound program, feel free to use DSOUND.DLL. If you want to create a DSOUND.DLL that works under 386 Enhanced mode, let me know--I've got the source and can say what doesn't work.

DSOUND.DLL (For Windows Programmers Mainly!)

Don't you always hate it when a neat app comes out that has a widget you'd like to use, only you can't. Me, too. So I isolated Sounder's playback code into a .DLL file. Applications that you write can access DSOUND.DLL. The main function in DSOUND.DLL as prototyped in DSOUND.H is:

void PlaySound(*lpSound, dSize, uFrequency, uSampleSize, uVolume, uShift*)

lpSound is a long pointer to a chunk of memory with the samples in it, one byte each. This should be locked but need NOT be in real address space. For future compatibility, page-locking the memory may also be needed under Enhanced mode operation, if I ever get it working! GlobalAlloc with GMEM_MOVEABLE works fine for now in Real and Standard mode.

dSize is an unsigned long that tells how many samples there are in lpSound. Currently, this should be the size of the sound data in bytes. Be accurate; Windows complains (dies) if DSOUND.DLL accesses memory you don't own.

uFrequency is the desired playback frequency. Currently 22 KHz, 11KHz, 7.33 KHz, and 5.5 KHz are supported. However, to be open-ended about this all, all frequencies are "valid," and the nearest one is used. That is, 9 KHz will play at 11 KHz, and 44 KHz will play at 22 KHz. I'll probably add 44 KHz functionality if I can optimize the play loop enough.

uSampleSize is a code for the sample size. Currently only 8-bit linear is supported, which has the code 0. This parameter is currently ignored, but future versions may support 16-bit sample sizes, log scales, and such.

uVolume adjusts the dynamics of the sound before playing. Basically, the amplitude of the sound centered around a middlepoint is expanded/compressed by the factor (uVolume/10). Thus, a volume of 5 is about half as loud, and 20 is twice as loud. In theory, that is--the PC speaker is not all that dynamic.

uShift is an amount added to each sample before being processed (but after the volume has been accounted for). This wouldn't be useful on a real sound chip, but it allows for compensating different speaker hardware in this hack. 3-5 seem to work best; 4 is the default Sounder uses.

Also defined in DSOUND.DLL is:

int GetDSoundVersion()

which returns the version of DSOUND.DLL as an integer whose upper byte is the major revision and whose lower byte is the minor revision. Currently this is 1.00, which supports the features described above in Real and Standard mode under Windows 3.0.

To use DSOUND.DLL by linking implicitly, DSOUND.H should be included in your source files that access it and DSOUND.LIB should be linked with your files. DSOUND.H prototypes all exported functions and defines a few useful constants. PlaySound should be IMPORTed from DSOUND.DLL; it's ordinal number is 1. GetDSoundVersion has the ordinal value of 2 and should be called to verify that the version of DSOUND installed supports any needed features. Your .DEF file should have:

```
IMPORTS
        PlaySound=dsound.1
        GetDSoundVersion=dsound.2
```

There are other ways of dynamically linking a DLL that vary from environment to environment; see the appropriate documentation for details (i.e. Actor).

.SND Files and DSOUND Resources

I propose the following format for sound files and resources of type DSOUND, and plan on incorporating a few useful functions in DSOUND to make accessing DSOUND resources easier:

.SND files and DSOUND resources are assumed to have a 32-byte header with the following fields defined:

word 0 Sample size code (see uSampleSize above)
word 1 Frequency to play back at (see uFrequency above)
word 2 Volume to play at (see uVolume above)
word 3 Shift (see uShift above)
long 1 Size of sound, in samples
long 2 Starting sample (0th sample follows header immediately)

All other fields are undefined but reserved. DSOUND.H defines these offsets and the header size.

The size of sound and starting sample fields are not currently used by Sounder, but would allow a partial .SND file or DSOUND resource to be specified. The size field is needed in resources to get an accurate size since the SizeOfResource call rounds to memory allocation units.

SOUNDER determines the size of a sound file by the file's length, not by the size field. This will probably change when/if the .SND format becomes real enough to matter.

Terms:

While the prospect of making a few bucks off this is tempting, reality says that no one pays shareware fees. The license to use it is $1, but I'm flexible. If you think this is too steep let me know. Checks can be sent to my P.O. Box, and as far as I'm concerned, you need only pay once for Sounder and any possible upgrades.

If you want to distribute DSOUND.DLL with a program that needs it, here are the terms:

o Public Domain and non-crippled or nag-ware shareware: use it for free.

o Commercial software, crippled shareware, nag-ware, others: contact me.

Oh, and if you do distribute this, don't modify it. It's hard enough fixing my own bugs...

Acknowledgements

I loved REmac, only it didn't work under Windows. I'm not exactly a DOS-type person anymore. A snoop or two with Debug suggested it could be done under Windows after extensive rewriting to make it protected-mode friendly and all. A few other programs (TALK/PLAY) showed how to get around a few other problems. I'm indebted to the authors of these programs for "showing that it could be done."

Compatibility

Sounder and DSOUND have been tested on a IBM PS/2 50 and 80, Zenith 386/33, a C + T-based 386sx, and a C + T-based 286-12. It does mingle with a lot of hardware-specific stuff, but everything it does is documented and otherwise kosher.

Trivia

On Micro Channel machines, the timer interrupt condition must be cleared with a write to port 61H with bit 7 set to prevent repeated interrupts. Thing is, this doesn't have to be done in real mode, at least on the PS/2 50. I've noticed that a lot of programs (i.e. REMac) don't clear the interrupt condition at all. Be warned, and be sure to test on nonstandard architectures like Micro Channel.

Source

Let me know if you're interested in the source to DSOUND.DLL, especially to make it run in 386 Enhanced mode. (Sounder is a rather boring hack, although it may be one of the few Windows programs that never touches a message, proving that a command-line oriented Windows program with a horrible interface *can* be written). If you come out with a non-freeware product that does digitalized sound a few weeks later that has a sound driver with the same algorithms or non-essential operations I put in DSOUND, I'll be real upset...

Aaron Wallace
P.O. Box 13012
Stanford, CA 94309-3020

aaron@jessica.stanford.edu