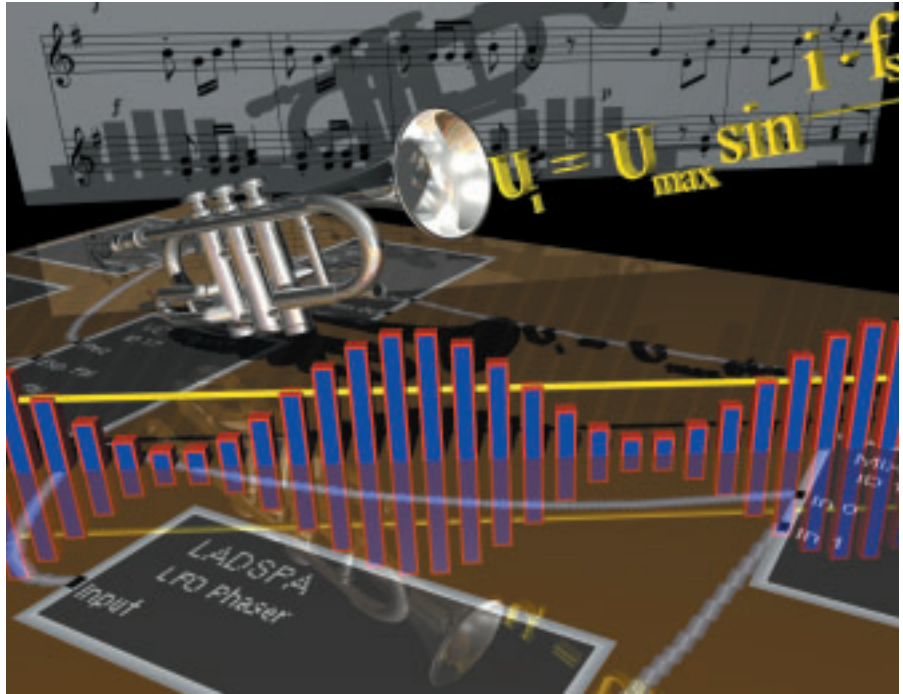


AlsaModularSynth, FluidSynth, and NoteEdit

Do-it-Yourself Instruments

Various synthesizer programs offer you a rich list of different instruments, or allow you to make your own. A musical score editor will allow you to play back your composition on the instrument of your choice. We guide you through the steps needed to make your first virtual instrument and show you how to make it talk to your sound card.

BY JÖRG ANDERS



Interested in composing your own scores and also playing them back on instruments of your own design? Two synthesizer programs, *FluidSynth* [8] and *AlsaModularSynth*, allow you to do this in conjunction with the musical score editor, *NoteEdit* [1].

NoteEdit

Currently, SuSE is the only well-known distribution that includes NoteEdit. For other distributions, you will have to compile the source code.

Recently, it has learned **Segno**, **Fine**, **Dal Segno**, **Dal Segno al Coda**, **Dal Segno al Fine** and **piano pedal markings**. Not only does it produce them, it also uses them when playing back a score. In addition to **Triplets**, **N tuples**, where N represents a number between 2

and 10, are now supported. Curly and square brackets for accolades are also available (see Figure 2).

Bar lines can be extended as needed over several notes. The programmer has also added several minor items, such as double bar lines, **Arpeggio** and the end bar.

Also, NoteEdit supports **Ritardando** and **Accelerando**, as well as chords and their corresponding guitar chord diagrams (see Figure 3).

Many users were previously upset by the fact that new notes were added by using the unconventional center mouse button. A left click now does this job. Also, the current version allows you to edit the individual notes of a chord, using the arrow keys to move them up and down.

Along with the export formats *MusiX-TeX*, *PMX* [2], *LilyPond* [3] and *MUP* [4], NoteEdit now supports the “ABC music” data format [5]. This is not only a popular exchange format for musical notation, it is also used by the *abcm2ps* score publishing system [6, 7].

To play back your music, NoteEdit needs a working **MIDI instrument**. The menu entry *Settings/Configure.../Sound* lists the available devices. Pick one, and make sure it is working by using one of the test files found in the sub-directory *examples*. SuSE users will find these in `/usr/share/doc/packages/notedit/`.

FluidSynth

FluidSynth, a software synthesizer program, often produces better tone quality than a physical MIDI instrument.



Figure 1: NoteEdit offers a wide range of musical notations



Figure 2: Accolades and Arpeggio



Figure 3: Harmonies and guitar chord diagrams tables

Some instruments are so bad, that even amateurs will hear the difference. Box 1 describes the installation of FluidSynth.

This synthesizer uses *wavetable synthesis* to produce sound output. This method uses *soundfonts*, collections of *samples* for all the instruments in a virtual orchestra. From these collections the wavetable synthesizer then generates any other MIDI sounds that are required.

The *FluidR3* and *Unison* soundfonts can be downloaded at [10] and [11].

You should only use FluidR3 if you have at least 256MB RAM available, because FluidR3 loads 141MB of soundfonts into working memory. If there is too little RAM, this can impact the whole system. If this is the case, try using the smaller Unisono soundfont.

After unpacking the soundfonts, you can type

```
gzip -d FluidR3.sf2.gz
```

to open up the FluidSynth command line. The program will now keep on producing synthesized sound until you type *quit* to stop it.

```
FluidSynth FluidR3.sf2
```

Once you have set up the port of the instrument with *Settings / Configure... / Sound..Synth input port*, NoteEdit will use the software synthesizer for playback. The settings should be tested with a sample file to make sure everything is working as desired. If the volume is not satisfactory, it can be adjusted using the *gain n* command where *n* is a number between 1 and 5. If the score includes strings, then look at Box 2 which provides a workaround for this bug.



Figure 4: Several modules make an instrument

Box 1: Installing FluidSynth

To allow FluidSynth and NoteEdit to cooperate, ALSA [9] is required to communicate with the sound card. Unfortunately, SuSE is the only major distribution that uses ALSA by default.

You need to install the ALSA development package before compiling FluidSynth with ALSA support. SuSE users need to install the *alsa-devel* package, while Red Hat users need the *alsa-driver*. After doing this, type *tar -xzf fluidsynth-1.0.3.tar.gz* to unpack the file you have downloaded. To compile and install do the following:

```
cd fluidsynth-1.0.3 ./configure
make su
(at prompt enter root password)
make install exit
```

ALSA Modular Synthesizer

AlsaModularSynth [12] by Matthias Nagorni is also a software synthesizer. It works differently than FluidSynth, inserting pre-defined modules that represent instruments, instead of using the wave table method.

Users of SuSE 8.2 Professional Linux will have to install the *alsamodular* package using YaST2 from their distribution CDs.

The *ams* command launches *AlsaModularSynth*. The package also includes a few example instruments, that have the extension *.ams*. SuSE places them in */usr/share/doc/packages/alsamodular/*. You can load *example_full_midi.ams* using *File / Load* to access the graphical instrument description (see Figure 4). Please note that these example instruments will not accept any MIDI instructions, and they ignore the musical instructions from NoteEdit. They only play back predetermined or random melodies.

GLOSSARY

Software synthesizer: If you don't have a MIDI instrument that is supported by Linux, you can use a program to provide the extra functionality you need.

MIDI: The "Musical Instrument Digital Interface" allows different electronic instruments, MIDI keyboards, computers and individual applications to communicate with each other. MIDI data does not contain musical information, but instead comprises instructions on the instrument key to be used.

Segno, Fine, Dal Segno, Dal Segno al Coda, Dal Segno al Fine: notations that define the repetition of certain parts of a score.

Triplet: This special N tuple tells three notes to have the duration of two.

N-tuple: A musical notation that prescribes a new beat for a group of notes.

Ritardando, Accelerando: Continuous diminishment or increasing of the tempo.

Arpeggio: Italian for "harp like"; production of the tones of a chord in succession and not

simultaneously.

Piano pedal markings: notation for using the piano pedals.

Midi instruments: As MIDI files do not contain tone information, the sound output must be produced by a sound card or a software synthesizer.

ALSA: The "Advanced Linux Sound Architecture", a sound system for Linux, will replace the OSS/Lite system in the 2.4 kernel as of kernel version 2.6.

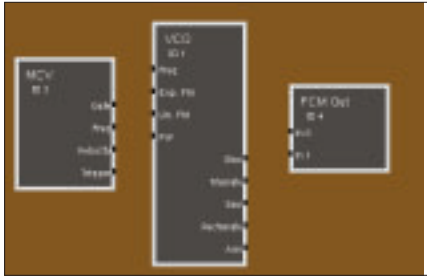


Figure 5: The first three instrument modules used to make an instrument

If you launched NoteEdit before *AlsaModularSynth*, close NoteEditor, and restart to make sure that the device is shown under *Settings/Configure.../Sound ams*. Pick your device and test it. *AlsaModularSynth* only plays single voices. Even though you can stipulate the *-pN* option, which tells the synthesizer to use more voices, (where *N* stands for the number of voices), more than one voice is too much even for the latest processors.

A right click on the *KCF* box drops down the context menu. The *Frequency* slider changes the pitch.

Do-it-Yourself

The basics ingredients for making your own instrument are provided by the **MCV**, **VCO** and **PCM-Out** modules, as shown in Figure 5. You can select *Module / New* to create a new module. Hold down the mouse button to drag and drop the module. A right click drops down a context menu that allows you to edit your module.

Now connect the ports by left clicking on the *Rectangle* output of the VCO module and the "In 0" input of the PCM-Out modules. Do the same to connect the "In 1" input of PCM-Out (see Figure 6). The geometric symbols on the VCO outputs

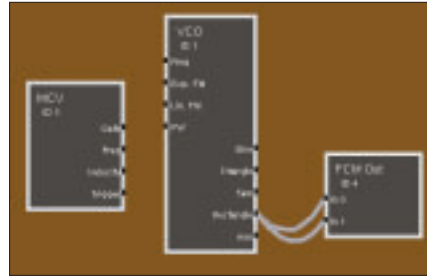


Figure 6: PCM-Out is the output on the sound card

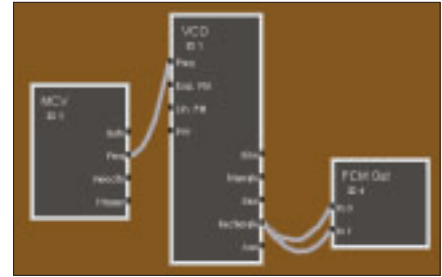


Figure 7: Connections bring the virtual instrument to life

describe the form of the oscillation, *Rectangle* is for a square pulse. The *Tune* regulator in the context menu of the VCO module changes the applied voltage manually, and should be set to 0 when working in conjunction with *FluidSynth*. The connection between VCO and PCM-Out is responsible for the oscillation on the loud speakers when playing back.

Box 2: Strings Errors in FluidSynth

Due to a programming error, *FluidSynth* synthesizes the *Strings* instrument incorrectly. You can work around this bug using *Choir-A*. First using the *Pointer* tool, the icon with the black arrow, select the line you wish to change. Now using the *Lines/Properties* dialog box select the 52. *Choir Aahs* instrument in the *Voices* window.

Synthesis / Stop breaks off the composition at any time. While *Synthesis / Start* will obviously start it up again. Right clicking will disconnect a connection you mistakenly hitched up to an input port.

The MCV module ensures care that the voltage reflects the pitch the MIDI data created by NoteEdit requires. Therefore, connect the *Freq* output with the *Freq*

input of the VCO module as shown in Figure 7.

Your instrument will now be used when NoteEdit plays a file with *ams*. If the tone is too high, then adjust its setting, *Note Offset* in the MCV module, from 24 to 36. This sets the pitch 12 half tones – one octave – lower.

As each tone has a constant volume until the next note is played, the output does not sound much like a real instrument. On a real piano, a note is at its loudest when the key is first struck and then diminishes in volume as shown in Figure 8.

In order to emulate the varying intensity on the *AlsaModularSynth*, add the *ENV* and "Lin. VCA" modules to your instrument. Connect the *Gate* input of ENV with the corresponding output of the MCV module. Also connect the *out* port to the *VCA Gain* port, as shown in Figure 9. Then connect the VCA output to both PCM-out input ports. The *Rectangle* output from the VCA should be connected to *In 0* on the VCA module, instead of PCM-Out.

An *Envelope* curve with the 5 tone phases *Attack*, *Hold*, *Decay*, *Sustain* and *Release* (see Figure 8) appears when you right click the ENV module. You can use the controls to adjust this curve to your needs. MCV uses the gate connection to tell the ENV module about incoming

GLOSSARY

MCV: "MIDI Controlled Voltage Supply" converts predetermined MIDI pitches to voltage values. VCO uses these values to generate sounds.

VCO: The "Voltage Controlled Oscillator" produces voltage induced oscillations.

PCM: "Pulse Code Modulation" converts digital signals to analog ones and vice versa. PCM-Out is the interface to the speakers.

VCA: The "Voltage Controlled Amplifier" amplifies signals according to the applied voltage.

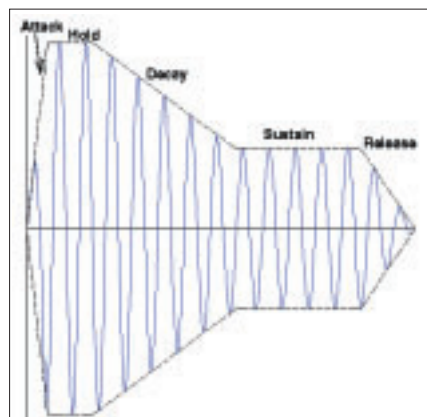


Figure 8: The volume of a natural note diminishes



Figure 9: Growing an instrument

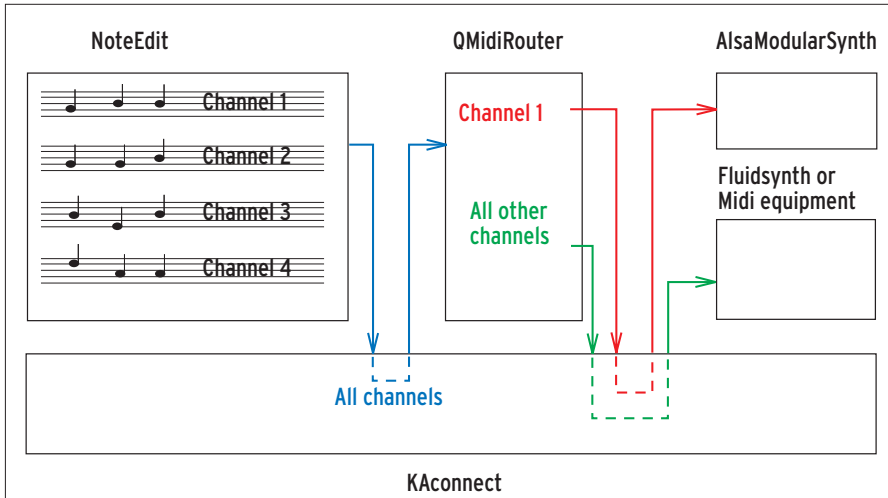


Figure 10: Assign the MIDI channels to different devices

notes. ENV then generates a voltage sequence envelope curve.

The VCA module serves as an amplifier and passes the signal on to the PCM-Out. If the adjusted envelope curve looks like the one in Figure 8, the output from NoteEdit will sound something like a natural instrument.

KAcconnect and QMidiRouter

If the first line of a NoteEdit file contains the melody, the other lines usually represent the accompaniment. With the help of *KAcconnect* and *QMidiRouter* [13] you can use different devices on different lines for the playback. For example, you can use *AlsaModularSynth* and your do-it-yourself instrument to play the melody, while with *FluidSynth* or a MIDI chip on your sound card takes over the accompaniment.

SuSE users will first need to install the *kalsatools* package.

It is customary to assign individual notes to different MIDI channels. The channel for a chosen line is shown in the *Note lines/Properties* dialog window in the *Channel* field.

Using the *QMidiRouter* tool, you can assign each of the channels so to allow *AlsaModularSynth* to

play channel 1, while *FluidSynth* plays the rest (see Figure 10).

After starting *qmidiroute* you select *Sort by* and *Channel* to sort by channel. To split on channel 1, set the *Split point* slider to 1 (see Figure 11).

After a re-launching, NoteEdit now shows the *QMidiRouter_in_0* device below *Settings/Configure.../Sound*. Choose this device in order to insert *QMidiRouter* between the output from NoteEdit and your MIDI devices.

Finally, we use *KAcconnect* to join the different inputs and outputs of the devices. Type *kaconnect* to launch the tool and display the ports it has

discovered. Among other things, *Readable Ports* displays a pair of *QMidiRouter* output ports. The upper port represents channel 1. Select it and *AlsaModularSynth* under *Writable Ports*, by left clicking with your mouse, and connect the two by clicking on the *Connect* but-

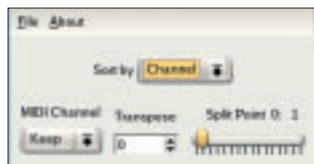


Figure 11: Split channels with QMidiRouter

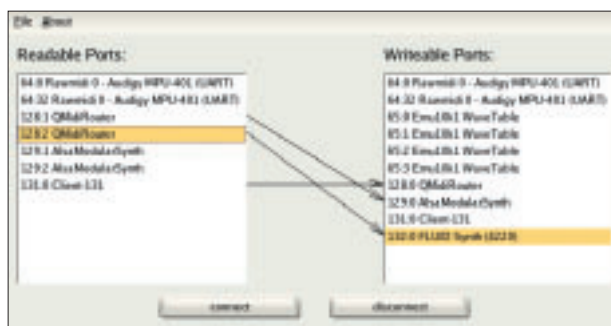


Figure 12: KAcconnect connects the input and output ports of MIDI instruments

ton (see Figure 12). Of course, *AlsaModularSynth* must be running for this to work.

Similarly, connect the two output ports of the *QMidiRouter*, which will produce all the other notes, with the input port of *FluidSynth* or another MIDI instrument of your choice. *disconnect* will disconnect the ports.

Clicking on *Play* in the NoteEdit window will now play the file on the appropriate MIDI instrument. If the volumes of the melody and the accompaniment are not well-balanced, you can adjust them in the appropriate synthesizer program. This is *gain* in *FluidSynth*, or the slider for the PCM-Out module in *AlsaModularSynth*.

What next?

This initial attempt at a do-it-yourself instrument may be a disappointment for some users, as it hardly sounds like the real thing. But, practice makes perfect, so keep trying for better results.

INFO

- [1] NoteEdit: <http://rnvs.informatik.tu-chemnitz.de/~jan/notedit/notedit.html>
- [2] PMX: <http://icking-music-archive.org/>
- [3] LilyPond: <http://lilypond.org/web/>
- [4] MUP: <http://www.arkkra.com/>
- [5] ABC: <http://www.gre.ac.uk/~c.walshaw/abc/>
- [6] abc2ps: <http://abcplus.sourceforge.net/>
- [7] ABC info: <http://moinejf.free.fr/>
- [8] FluidSynth: <http://www.fluidsynth.org/>
- [9] ALSA: <http://www.alsa-project.org/>
- [10] SoundFonts: <http://www.hammersound.net/>
- [11] SoundFonts for the SoundBlaster Live: <http://www.personalcopy.com/>
- [12] ALSA Modular Synthesizer: <http://alsamodular.sourceforge.net/>
- [13] QMidiRoute and KAcconnect: <ftp://ftp.suse.com/pub/people/mana/>

THE AUTHOR

Jörg Anders is the developer of NoteEdit. He is an assistant at the Computer Science Faculty of the Technical University in Chemnitz, Germany. Jörg uses Linux, plays guitar and a little piano.

