RTSynth v1.9.0

http://www.linux-sound.org/rtsynth/

A real time software synthesizer

Introducing ................................................................. 2
News (not complete) .......................................................... 6
Midi Events ................................................................. 8
Main Panel ................................................................. 9
String Synth .............................................................. 12
Flute Synth ............................................................... 19
Drum Synth .............................................................. 20
Audio Effects ............................................................ 21
Index ........................................................................... 25

2nd draft version November 2002, Stefan Nitschke
Introducing

Welcome to RTSynth a midi event triggered real time software synthesizer entirely based on physics and mathematics. Goal of the software is to reproduce sounds of strings, organs, flutes and drums in real time. The sound generation is based on the physics of the instruments. That means all sounds created by this software are the result of a mathematical model based on some physics about how the sounds shape varies in time. You have access to most of the models parameters through a graphical interface. Further more most of the parameters can be controlled by MIDI controllers.

The intro of version 1.6.2 is some years old and CPU performance has increased a lot. RTSynth is now developed on an AMD Athlon XP1700+ based PC. Here are some benchmarks for the effects measured on my machine:

RTSynth v1.9.0 benchmarks on Athlon XP 1700+ with frame buffer size of 128:

| Effect         | Time  | NURT  
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>off</td>
<td>8.6 mS</td>
<td>3365</td>
</tr>
<tr>
<td>none</td>
<td>36 mS</td>
<td>792</td>
</tr>
<tr>
<td>dist (hard)</td>
<td>115 mS</td>
<td>252</td>
</tr>
<tr>
<td>filter (24dB)</td>
<td>192 mS</td>
<td>152</td>
</tr>
<tr>
<td>chorus</td>
<td>342 mS</td>
<td>85</td>
</tr>
<tr>
<td>pp delay</td>
<td>342 mS</td>
<td>84</td>
</tr>
<tr>
<td>Resonator (x5)</td>
<td>342 mS</td>
<td>84</td>
</tr>
<tr>
<td>dist (soft)</td>
<td>418 mS</td>
<td>69</td>
</tr>
<tr>
<td>PM resonator</td>
<td>544 mS</td>
<td>53</td>
</tr>
<tr>
<td>reverb8</td>
<td>814 mS</td>
<td>35</td>
</tr>
<tr>
<td>All together</td>
<td>2359 mS</td>
<td>12</td>
</tr>
</tbody>
</table>

NURT: number of units playable in realtime at 44100Hz, stereo.

And yes i tried the above number. My machine can play even more than the given 12 complete effect panels;) Anyway from the numbers above RTSynth should still be usable on a Pentium 200 like system.

Current versions of RTSynth need ALSA version 0.9 audio drivers and use internally 64 bit floating point numbers for best audio quality.

Introducing from old version 1.6.2

RTSynth supports a polyphonic string synth, a monophonic flute synth and a set of audio effects. It should run on any machine with a CPU/FPU performance comparable to a pentium 90 or higher. Any faster machine is welcome and will allow more complex sounds, audio effects and/or a higher number of independent voices. I hopefully did a good job in run time optimization of the code but anyway, as with all
real time applications, there is never a too fast machine. Internally all sound operations use 32 bit floating point numbers to give lowest distortion and highest possible dynamic range. RTSynth is currently developed on a Cyrix 6x86-166 (seems to be comparable to a pentium 90!) based machine running Linux v2.0.33, egcs 1.1 and glibc2.

To perform well, RTSynth should be set to userid root with:

```
chown root RTSynth   
chmod +s RTSynth
```

This will allow RTSynth to use the real time scheduler for best run time behaviour. After getting the scheduler RTSynth is setting back its root priority to the normal users one. To make the choice of the MIDI input source as flexible as possible, RTSynth can read midi events from a input stream. This can be a device, a pipe (stdin) or a named pipe (fifo).

Possible command line invocations:

```
RTSynth
```

for normal execution, reading midi events from /dev/midi00.

Output options:

```
RTSynth --output=/dev/mydev
or
RTSynth -o /dev/mydev
```

use the given audio output device (must be a OSS /dev/dsp compatible device).

Input source:

```
RTSynth --input=/dev/midi
RTSynth -i /dev/midi
or
RTSynth </dev/mymidi
to read from a given midi device
```

```
RTSynth <fifo
or
cat foo | RTSynth
```

to read midi events form a fifo or pipe.

I have tried to use EsounD as an alternative audio output destination, but the current version of esound does not support real time audio. To be more detailed, the audio

2nd draft version November 2002, Stefan Nitschke
stream esound opens for RTSynth to write to does not offer a constant rate of samples per second like the /dev/dsp device. Maybe ALSA will become the future of audio and midi subsystems for Linux, but I did not take a closer look on it so far, maybe when upgrading to linux 2.2 ...

If you don’t have a midi keyboard (like I am) or any other external midi equipment, you can use clavier to put you computer keyboard into a midi one. The necessary command lines to connect clavier with RTSynth would look like:

```
mkfifo /tmp/midififo
RTSynth < /tmp/midififo
clavier -o /tmp/midififo
```

or you can use SoftWerk a cool software "analogue" sequencer to trigger RTSynth.
News (not complete)

Version 1.9.0
The blue edition reached another milestone on its way to version 2
-> version 1.9.0.

- In this version more physics can be applied to the sounds by the use of an improved physical modelling resonator.
- Eliminated stability problems. No more wild going flute sounds or drums that kick out their resonators.
- Improved PM-resonator effect. This multi resonator adds a body to your sounds and can also add some non linear effects of the body which depends, like with real instruments, on the playing loudness. Updated documentation about this effects as it is the most important (physical) one.
- The maximum number of voices has been increased to 32.
- Preset sounds have been updated. NOTE: The presets are for a sample rate of 44100Hz and may not sound as well on other rates.

Version 1.8.0
New release version 1.8.0 (code name "blue") is available for download. "Renovated" the quite old DSP core a little bit to fit to modern CPUs by removing look-up tables and replacing approximations by their exact formulas.

Benefits.
- The application is not bound to a fixed sampling rate anymore. As there is still some work to do in the effect section the supported range is for now limited to the range 44100-48000 Hz.
- Audible increase of overall tuning.
- The string synth has now the long missing pitch bend with a selectable range of +/- 0-4 half-tone steps. A step size of zero will switch back to the old behaviour. (pitch-wheel -> filter) NOTE: As this is a somewhat CPU intensive task you should switch pitch bend only to on when you really need it!

Disadvantages:
- Eats some more CPU cycles ;-)

2nd draft version November 2002, Stefan Nitschke
Version 1.6.8

- This version is for ALSA version 0.9.
- Added command line option to set the audio fragment size.
- Added small configuration dialog.
- Improved tracking of pitch-bender in string and flute synth.
- Added optional "auto pan" mode.
- Added support to control synth parameters by MIDI controllers.

Version 1.6.2

- BugFix: eliminated problems with reading from /dev/midi on linux 2.2.x based systems.
- Changed handling of midi NoteOff events -> eliminating possible multiple playing of the same note.
- Added new main-panel. You can now run several synth-/effect-modules and connect them together.
- Improved and scale-able reverb effect.
- Some minor bug fixes and runtime optimization.

Version 1.6.0

is the first glibc based version. Advantages over the older 1.5.x versions are:

- RTSynth now always runs in real time mode. The problems with reading midi events from a pipe are solved by using threads.
- RTSynth now works fine with the "SoftWerk" software sequencer program.
- New features are a peak load meter that monitors the CPU/system load from the view point of RTSynth.
- A auto switch off function on a system overload or audio underrun.
- The output filter frequency (string synth) and the embouchure size (flute synth) are connected to the midi pitch bend for real time control by a external device.
- Some build in preset sounds (instrument and effect).
- added command line option for audio output device.

2nd draft version November 2002, Stefan Nitschke
Midi Events

Midi events recognized by RTSynth so far:

- NoteOn (includes velocity)
- NoteOff
- AllNotesOff
- AllSoundOff
- AllCtrlOff
- PanPosition
- HoldPedal - only available when the string synth is set to than 20 voices.
- Volume
- PitchBend - The filter/embouchure value given by the interface slider is multiplied by two on max pitch bend and zero on min pitch bend. A centred pitch bend leaves the slider value unchanged.¹
- Support for midi "running status" used by some sequencer programs like CuBase.

NOTE: For a full list of supported MIDI controllers please see the online help within the synth panels.

¹Since version 1.8.0 MIDI pitch has been implemented.
Main Panel

From the main panel you have access to all elements of the program:

- Create, control and connect audio modules (synths, effects).
- Change the audio settings from the Options menu.
- Save/Load patches and the machine state.
- Switch the machine to On/Off state.
- Toggle between mono/stereo mode.
- Control the master volume of the application.
Every Synth- and Effect-module is represented by a tile

You can set the midi channel to receive events from and switch the synth/effect module on/off directly from the tile by using the control elements shown in the picture.

- Pressing the left mouse button on the audio output symbol and dropping it over the target tile will build a connection. Doing the same process between already connected modules removes the connection.
- Any module with a output that is not explicitly connected to another one is connected to the main output by default.
- Circular connections between modules are not allowed and will be rejected by the program.
- Left/right mouse button over the midi channel field decrements/increments the channel value.
- Pressing the right mouse button opens a context menu.
Options Panel

From the option panel you can set:

- the audio device.
- change the sampling rate.
- select the audio frame size and number of fragments.
- set the path to your patch files.

You should have a low latency patch applied to your Linux kernel to get optimal audio output quality even under high system load.
String Synth

As the name suggests this synth is mainly designed to produce string like sounds. But it is not limited to that. You can also use the string synth for example to produce a sinus wave sound or together with the resonance able low pass filter some cool synthetic sounds. It is a polyphonic synthesizer with up to 32 independent voices based on the so called ‘physical modelling synthesis’. Because the base algorithm does not have much to do with the physics of a real string instrument (the interaction between the string, the instrument corpus and the players fingers are much too complex to be calculated on a PC in real time), I call it the ‘energy dissipation model’. The maximum number of playable voices of course depends on the speed of your machine, synthesizer settings like filters and the number and type of activated audio effects.

To become familiar with the synthesizer settings you may try out some of the synth voices included in the archive.

2nd draft version November 2002, Stefan Nitschke
Simple Example

A small example starting with a synth producing a sine wave, as mentioned above, and then modifying it to went into a "SynthBass" sound is given here:

1. We want a clean sinus wave -> Wave ampl. = 1, Wave type = sin.

2. The sine wave should have a constant amplitude. This is meet by setting all damping parameters in the Energy dissipation panel to "no damping" -> frequency = 1, freq max = 1, filter follows note = no, feedback = 1, filter attack = 0, off-delay = 0. It is a little bit hard to set the feedback to a value of exact 1.00, soft damp. can help in do it for you by setting it to a value greater zero.

3. Normally a sinus sound needs no output filter but as you can hear on some notes there is some noticeable distortion. To reduce it we use the Output filter with filter follows note = lin, frequency = 1 (filter any frequency higher than the base octave), bessel, 12 dB. You may also use a small resonance value (0.1) to get a even sharper filtering.

4. Now, we want to apply a conventional amplitude envelope to the sinus wave to get a short percussive sound. The Output filter envelope can be used for that -> env mode = normal, env ampl. = 0 (we don’t want a filter sweep), env speed = 1(fastest attack), env sustain = 0 (no sustain at all), env release = 0.2 (short sound). And last but not least in the Output and note control panel the ampl-env must be activated.

5. We modify the sound now a little bit. For the wave type we choose an unfiltered rectangular wave (rect) an apply an output filter envelope by setting env ampl. to 1.4 and to make the sound a bit more interesting we apply a value of 0.3 to the output filter resonance. To apply note velocity to the filter sweep amplitude we switch the env mode to accent.,to ampl. if we want the note volume to be independent from the note velocity, and to speed if our intention is to modulate the envelope speed by the note velocity. To have a sustained sound again we can switch the ampl-env off

6. to produce a more string like sound we switch the output filter to off, the ampl-env to off. Set the Wave ampl. to 0.5, the Random filter frequency to 0.53, its filter follows note mode to "cool" and let Energy dissipation do the decay stuff by setting: frequency = 0.68, freq max = 0.89, filter order = 1, filter follows note = "cool". To have a slowly decayed sound off delay = 30, off-feed. = 0.998 for example can be set.

7. now we switch the Output filter on again to have a some what different "SynthBass" sound than in Step 5.
The String Synth panels

The string synth panel is subdivided into smaller panels sorted by functionality.

**Energy dissipation**
This together with *Random Filter* and *Additional wave* is the "heart" of the synthesizer. Here you can set the parameters for the time dependent behaviour of the oscillator like frequency dependent damping, cut off frequency ...

**Random filter**
The random filter is used to filter the noise source output and may also be used to filter the output of the additional wave. The filtered random noise together with an additional wave are used to initialize the oscillators starting wave form. The usage of a random noise amount to produce a wave form makes the sounds generated by the string synth quite different form the static sounds generated by sample based synthesizers. Like for a real string instrument the produced sound will never be exact the same even when playing the same note.

**Additional wave**
In addition to the noise source the output of the additional wave unit can be used to build the starting wave form. This panel includes a mixer for the balance between the random noise and the additional wave amplitude.

**Output filter**
Here the generated wave can be filtered by a resonance able low pass filter.

**Output filter envelope**
Settings for an output filter and/or amplitude envelope can be done here. Each String Synth voice has only one envelope generator but it can be used for output filter and amplitude modulation.

**Output and note control**
Sound volume, amplitude attack, decay, amplitude envelope and more can be set here.
Energy dissipation panel

**frequency**
Cut off frequency used for frequency dependent energy dissipation (damping by successive low pass filtering). This one together with ‘feedback’ give you basic control over the time dependent character of the generated sound. (short, sustained, ...) 

**freq. max**
Highest cut off frequency. A value of 1 is equal to infinite.

**filter order**
Setting the damping rate of the energy dissipation low pass filter.

order: 1 <= 6 dB, 2 <= 12 dB, ...

**filter follows note**
- **no**
  fixed filter frequency.
- **yes**
  filter frequency follows note value. Higher note gives higher cut off frequency.
- **cool,usr1,usr2**
  filter follows note value non linear.

**feedback**
Frequency independent energy loss. (1 = no damping)

**soft damp**
Starting with the value given by ‘feedback’. Reduce damping until a feedback value of 1 isreached. If ‘off delay’ is in use the ‘off-feedb.’ value will be used after the ‘off delay’ time has been reached. A value of zero means, disable "soft damping"

**filter attack**
Attack time is given in number of periods. (‘fixed time’ off). A value of zero means disable ‘filter attack’. At the end of ‘filter attack’ time the filter frequency value given by ‘frequency’ is reached.

**fixed time**
Interpret the number given in ‘filter attack’ as a fixed length time, independent of the note
start freq

The frequency of the energy dissipation filter used by the ‘filter attack’ as the starting value.

off delay

Time in periods until the ‘feedback’ value is replaced by the ‘off-feed.’ value. A value of zero means, disable ‘off-feed.’.

off-feed.

Damping value used after the time given by ‘off delay’ has been exceeded.

Random filter / Additional wave panels

**frequency**
Cut off filter frequency used for the noise source and wave given by the Additional wave panel.

**filter follows note**
For a description take a look at Energy dissipation panel.

**Wave ampl**
Mixer for the amplitudes of the wave given by ‘Wave type’ and the filtered noise source. A value of zero means only noise and a value of 1 only wave as source.

**Wave type**
The wave type. The choices "crect", "csaw" and "ctri" are the same as their counterparts without the leading "c" but will not go through the Random filter. "piano", "piano2", "metal" and "sawres" are a little bit more complex wave types and functions.

**cont level**
Amount of continuous energy that flows into the system.

**cont flow**
Enable continuous energy flow.

**clean flow**
When enabled the amount of energy has the "clean" time shape of the selected ‘Wave type’.
**Output filter panel**

**frequency**
Cut off frequency of the low pass filter. This value can be altered in real time by the midi pitch bend\(^1\).

**resonance**
Amount of filter resonance.

**filter type**
critical - critical filter characteristics.
bessel - filter with Bessel characteristics.
butterw - "Butterworth"
tscheby - "Tschebyscheff"
tscheb3 - ditto with 3 dB cut off resonance.

NOTE: The filter types are only correct for filters without any resonance.

**filter order**
- **off** Filter off.
- **6, 12, 18, 24 dB** Low pass filter with the given damping.
- **+12, +18, +24 dB** Mix of filtered and unfiltered output.

**filter follows note**
- **no**
  fixed filter frequency.
- **lin**
  filter frequency follows note. A value of 1 means same octave as the played note, value = 2 one octave higher. NOTE: This values are only approximate.
- **usr**
  filter frequency follows note value non linear.

\(^1\) will be deactivated by a MIDI pitch-bend range value different from zero (string panel)

2nd draft version November 2002, Stefan Nitschke
Output filter envelope panel

**env ampl**
The amplitude of the envelope. Setting this value to zero will disable the filter envelope but has no influence to the amplitude envelope behaviour.

**env speed**
The attack speed of the envelope. A value of 1 means no attack phase.

**env sustain**
The sustain time of the envelope.

**env release**
The release speed of the envelope.

**env mode**
- **off**
o no filler/amplitude envelope. (saves lots of CPU time)
- **normal**
envelope speed and amplitude are independent from note velocity.
- **accent**
envelope amplitude depends on note velocity.
- **acc.inv**
ditto but with inverse dependency.
- **ampl.**
like accent but the note velocity only affects the envelope not the volume of the played note itself.
- **a. inv**
ditto but with inverse dependency.
- **speed**
like ampl. but now the envelope speed depends on the note velocity, note volume is constant.
- **s. inv**
ditto but with inverse dependency.
Output and note control panel

volume
The amplitude of the played sound.

mute
scales down the amplitude.

attack
The attack time of the amplitude envelope. A value of zero means no amplitude attack.

release
The release time of the sound after receiving the ‘midi note off’ event.

silence
The number of "silence" played periods of the sound. Will normally give a more soft attack sound.

LFO freq.
The frequency of the amplitude envelope LFO.

LFO ampl.
The amplitude of the LFO. A value of zero disables the LFO.

dc-filter
A filter to reduce the DC part of the generated sound. Normally not needed.

ampl-env
When set, the "Output filter envelope" can also be used as a input for the amplitude envelope.

modulation
Some kind of modulation applied to the sound.

bass boost
This will boost the amplitude of low frequency notes.

auto pan
Adds a stereo placement to the sounds.

lin velocity
Selects a linear MIDI velocity to amplitude mapping.
This is a monophonic synth. It implements the ‘base clarinet’ and the ‘base flute’ algorithms with some small extensions and modifications. Since version 1.8.0 the flute synth has become really usable. The tuning is now fine and instabilities as found in older versions have gone away ;)
Drum Synth

First and incomplete version of a drum synthesizer based on synthesis techniques as used by the early analog drum machines in combination with some physical modelling aspects.

The current version implements the following voices:

- 2 bass drums.
- 6 toms.
- 2 snare (incomplete),
- hihat closed/foot/open (incomplete).
Audio Effects

RTSynth has seven hard wired audio effect units.

Effects in order of their connection:

- **Distortion**
  The distortion effect has a soft mode, emulation of a tube like distortion, and a normal mode for a transistor like distortion.

- **Universal filter**
  This universal filter can be used as a LP, HP or BP filter. The filter order can be set from 6 dB to 24 dB. It also supports a simple filter sweep function.

- **PM resonator**
  The PM resonator gives you the possibility to add a "body" to your sounds. It is a multi resonator with a non linearity on higher input levels. As you can see from the "PM resonator" panel/table, this effect has a lot of settings to play with ;) Just as in real life you can tune the PM resonator in such a way that some notes will sound "dead", so be careful ... but usually the PM resonator is a good friend in making your sounds more reach and full. Please use one of the settings from the presets as an starting point to get somewhat familiar with this effect.

- **Resonator**
  This is an equalizer with a bank of five band pass filters. Setting a filter to a
value of zero disables this filter and therefore saves CPU time.

- **Chorus**
  The well known chorus effect.

- **Reverb**
  A simple reverb effect. In stereo mode two reverbs, one for each channel, can be mixed to give a more complex effect.

- **Delay**
  The delay effect can operate in stereo mode to give a ping-pong like effect.
PM resonator

**feed**
Amount of feed back. The resonator will start to oscillate when set to high values. (good values are around 0.5) High values may be useful in finding a resonance mode setting.

**feedfreq**
Controls decay of high frequencies and therefor the sound characteristics of the effect. Real physical resonators like a guitar body have a fast decay of high frequency. Anyway a very low value will suppress any resonance on higher notes.

**Intensity**
Here the amount of the continuous "energy flow" through the resonator, which is necessary to get some resonance, can be set. Good values are 0.6-0.8. Greater values may give higher non linear effects.

**Input gain**
The PM-resonator effect has a limit for the maximum input value. Before this limit is hit the effect becomes more and more non linear. The input gain control allows you to down/up scale the input value. Be careful at very low gain values the movement of other sliders can generate quite loud noises as in this case the output gain is set automatically to very high values.

**efx vol.**
The volume of the effect.

**efx only**
When switched on only the effects signal will pass through but not the original one. Properly interesting when cascading several resonators.

**reso. mode**
A value of (assuming feedfreq is set to 1.0):
1.0 will suppress the base frequency.
0.5 will result in a resonator which most likely will resonate at its base frequency but higher octaves are suppressed.
0.25 will give a resonance at the base frequency and its first octave. Higher octaves are suppressed.
0.125 ... guess you know what will happen here.

Smaller values will in general result in a resonator which will resonate at its base frequency and more and more octaves as the value decreases (higher octaves are preferred). But that’s not what we normally want to have. We can achieve to have a resonator with any resonance mode by choosing any other values. Good starting points are to move the mode value slightly out of tune :)

**mode fine**
Fine tune of current reso. mode value (+/- 15%).

---

2nd draft version November 2002, Stefan Nitschke
scale x20
Multiplies reso. mode by value of 20.

frequency
The base resonance frequency of the resonator in Hz.

d fr eq. fine
Fine tune base resonance frequency (+/- 10%).
# Index

## A
- Additional wave 13
- Audio Effects 21

## C
- Chorus 22

## D
- Delay 22
- Distortion 21
- Drum Synth 20

## E
- Energy dissipation panel 14
- Energy dissipation 13

## F
- Flute Synth 19

## I
- Introducing 2

## M
- Main Panel 8
- Midi Events 7

## N
- News 5

## O
- Options Panel 10
- Output and note control panel 18
- Output and note control 13
- Output filter envelope panel 17
- Output filter envelope 13
- Output filter panel 16

2nd draft version November 2002, Stefan Nitschke
Output filter

P
PM resonator 21, 23
Possible command line invocations 3

R
Random filter / Additional wave panels 15
Random filter 13
Resonator 21
Reverb 22

S
Simple Example 12
String Synth 11

T
The String Synth panels 13

U
Universal filter 21