VSTHost

A program to run VST-compatible PlugIns

Download Information

The latest version of VSTHost can be found at http://www.hermannseib.com/english/vsthost.htm

Contact Information

The author can be reached per email at office@hermannseib.com

Copyright Information

Table of Contents

Introduction 5
What is VST? 5
What is ASIO? 5
More Information on VST, ASIO and PlugIns 5
What is VSTHost? 5
Evolution 5
What does it cost? 6
Installation 7
Requirements 7
Configuration 7
Audio Configuration 7
ASIO Control Panel 9
ASIO Channel Selection 9
MIDI Configuration 10
MIDI Input Devices 10
MIDI Output Devices 10
MIDI Thru 11
Remote Control Port 11
Other Configuration Tasks 12
Operation 13
Command line parameters 13
Syntax 13
Main Window 13
Menu Entries 14
File Menu 14
Load Program 14
Save Program 14
Save Program As 15
Reload Program 15
New Effect 15
Exit 16
Effect Menu 16
Name 17
Close 17
Load Bank 17
Save Bank 17
Save Bank As 17
Assign Input Channels 18
Mute 18
MIDI Devices 18
MIDI Input Filters 19
Chain After 20
Unchain 21
Load Program 21
Save Program As 21
Next Program 21
Previous Program 21
Programs mm-nn 21
Engine Menu 21
Run 21
Restart 22
Priority 22
Input Assign 22

Output Assign 23
Recorder 23
Autorepeat 24
Recorder File 24
Configure Recorder 24
Devices Menu 25
View Menu 25
Auto Edit 25
Toolbar 25
Recorder Bar 27
Keyboard Bar 27
Configure Keyboard Bar 28
Status Bar 30
Status Bar Level Meter 30
Window Menu 30
Master 31
Help Menu 31
About VSTHost 31

Introduction

What is VST?

"Okay, what are VST PlugIns?", I hear some of you say… well, let's do a little history research. If you already know what it means, just skip to the next section.

The term **VST** was coined by Steinberg some years ago as an abbreviation for „**V**irtual **S**tudio **T**echnology". It is an interface definition that allows communication between a **VST Host** (originally, of course, Steinberg's Cubase sequencer, but many more programs have adopted the interface by now) and **virtual effects** and **instruments**. These effects and instruments are implemented as separate units and can be "plugged into" the VST Host wherever they're needed, thus they are commonly called "PlugIns". The VST Host sends audio data streams to the PlugIns in a special format and adds their output to its own audio processing.

Since V2 of the VST definition, there are two kinds of VST PlugIns: **effects** and **instruments**. The distinction is that effects process an incoming audio stream, while instruments *create* their own – they are triggered by MIDI events, just like an external synthesizer would be.

Steinberg's SDK provides a VST implementation for quite some operating systems; VSTHost, however, only works on Windows.

VSTHost can use the old-fashioned Windows Multimedia Extensions (MME) interface to exchange audio data with the sound card, or it can use an ASIO driver, if available.

What is ASIO?

"Okay, but what is ASIO?", I hear some of you say… probably the ones who didn't know "VST" ☺… well, let's do a little history research again. If you already know what it means, just skip to the next section.

The term **ASIO**, again, was coined by Steinberg some years ago as an abbreviation for "**A**udio **S**treaming **I**nput **O**utput". It defines a rather fast communication method between the audio hardware and an audio-processing program, such as a VST Host. An ASIO driver, if available, generally performs better than a MME driver, since it has considerably less overhead, allows multi-channel communication and so on.

More Information on VST, ASIO and PlugIns

The SDKs for Steinberg's VST and ASIO Software Development Kits can be found on the Internet. When I last checked, they could be found at http://www.steinberg.de/Steinberg/Developers8b99.html .

An exhaustive, searchable list of PlugIn descriptions can be found at http://www.kvraudio.com .

What is VSTHost?

VSTHost is a program to run VST PlugIns. In contrast to the big programs (Cubase, Nuendo, Logic come to mind), it is not a full-fledged giant sequencer package that needs many seconds just to come up to a point where it can say "Hello". It's small, and hopefully will stay so, and it *only* runs PlugIns. No sequencing, no elaborate recording facilities (although it does have a simple multi-track tape recorder built in). To make up for that, you can define very complex PlugIn setups and switch between them easily.

Evolution

The main goal for VSTHost, in the beginning, was simply to provide a little test bed for VST PlugIn development, and to understand how VST works "under the hood". By now, this goal has been

reached; VSTHost can load nearly every PlugIn (it even occurs in the "list of compatible hosts" for quite some PlugIns, which I find rather flattering). If you want to perform some in-depth debugging, the full source code for a reduced version of VSTHost is available for download on my web site (see page 2 for details).

VSTHost is still evolving, however; the goal for the next year is to turn it into a valuable tool for performing artists in a Live environment. Sort of a "super-synthesizer", if you want.

What does it cost?

Aaaah, this is the point where money comes into play; I was never good at that ☺… basically, it's free. The download version is not restricted in any way, doesn't even show a nag screen. Why encourage pirates to tinker with it?

In theory, it's "donationware", which means that you can download and use it; if you find it useful, it would be nice to register by sending a little bit of money to my PayPal account. There's a "Donate" button on VSTHost's web site for that. I don't insist on it, but it would be nice if you honored the countless hours I've invested into making this thing usable by donating a bit to the further development of VSTHost.

Installation

VSTHost is too simple (or too intelligently written? You decide ☺) to need an elaborate installation procedure. Simply copy the executable into a directory that suits you, eventually create a link to it in your start menu or on the desktop, and that's it.

Requirements

To run VSTHost, you need at least the following:

- a contemporary computer with a Pentium II (or better) or Athlon processor with least 500MHz; the more, the better;
- a fair amount of RAM; while 128MB should be sufficient for a minimalistic setup, 256MB are much better; for larger setups, 512MB or more are recommended;
- a sound card; while VSTHost works even with the measliest AC97 on-board chips, a modern card that can handle 24bit audio is recommended;
- Windows operating system; Windows 98, ME, NT4, 2000, and XP are supported.

If your sound card comes without an ASIO driver (obviously created with game players in mind instead of musicians), you might try to use Michael Tippach's ASIO4ALL driver, which can be found at, surprise, surprise, http://www.asio4all.com – it can work wonders compared to the MME drivers that are normally provided with the sound cards.

That's it – there isn't more to it. Simply start VSTHost for the next task:

Configuration

When you start VSTHost for the first time, it comes up with a minimal configuration. "Deaf, dumb and blind", as The Who would have called it. You'll see a window like this (the exact look, of course, is determined by your Windows setup):

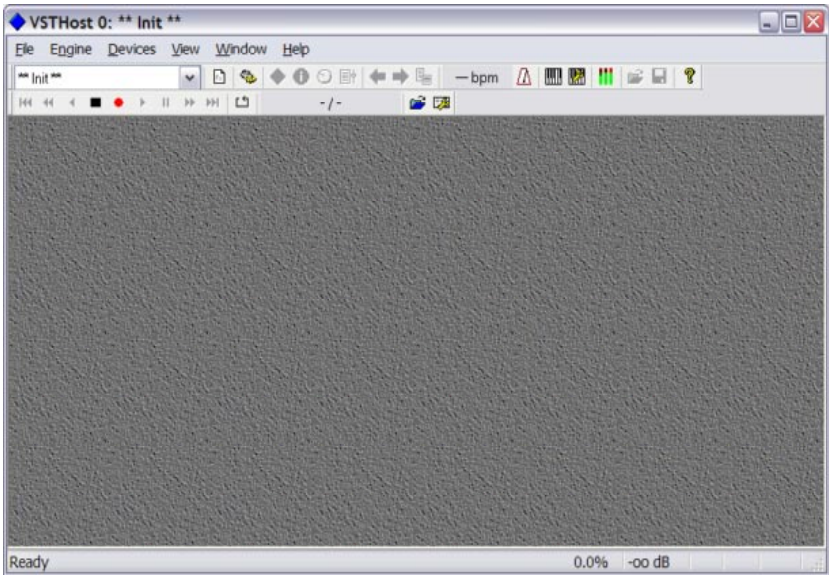


Figure 1: Initial VSTHost window

Hmmm. Well. OK. So what does that mean…? Let's start with the obvious first task:

Audio Configuration

When VSTHost comes up the first time, it doesn't know anything about the computer's configuration. Being rather conservative in its views, it doesn't preload any specific driver. In Windows systems, there's something called the **Wave Mapper** device. The user can predefine an audio device as the default device – the thing that is used whenever Windows needs to issue a "Ping!" or "Bleep!" or

"Whoop!" to indicate certain conditions. VSTHost preloads this standard device as its Audio output device, and that with a very large buffer size. This, however, is probably the worst possible solution – but one that's practically guaranteed to work.

To set up a better configuration, we have to enter the **Wave Device Settings** dialog by choosing its menu entry:

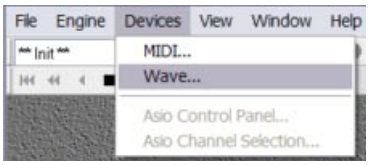


Figure 2: Wave Device Settings menu entry

… which opens the following dialog:

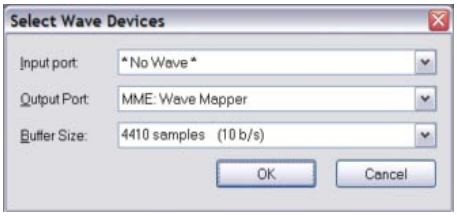


Figure 3: Wave Device Selection Dialog

Here, the wave devices used by VSTHost can be defined. The combo boxes contain the possible devices. As you can see, the Wave Mapper is preselected (and loaded), with a large buffer size. At 44.1kHz, which VSTHost uses with MME devices, this means that it processes audio at a rate of 10 buffers per second. This buffer size should work even on the slowest computers, but it doesn't allow real-time operation. So, we'd better redefine that setup to the best possible for the computer at hand.

The devices listed in the combo boxes all have a prefix, either **MME:** for Windows Multimedia Extensions drivers, or **ASIO:** for ASIO drivers.

The **Input port** combo box shows all available input devices. This box will never contain anything else than MME drivers; since ASIO drivers combine the operation of input and output drivers, they are only listed in the **Output Port** combo box. Whenever an ASIO driver is selected in the Output Port box, the Input Port box is grayed out to reflect the fact that ASIO doesn't need a separate Input port.

The **Output Port** combo box shows all available output devices. If it should be empty, your computer doesn't have (or doesn't think it has; Windows 98 sometimes works in mysterious ways ☺) any sound card. In this case, VSTHost can still be used to load and debug PlugIns, but you won't hear anything, which makes it a kind of useless intellectual exercise for most of us.

You should select the best possible combination for your computer; with Windows NT, there's not much choice, since there are not many ASIO drivers for NT floating around. ASIO4ALL doesn't work, since NT4 doesn't follow the WDM driver model, so your choice will probably be rather limited. In general, you should always take the drivers that are closest to the hardware; in audio processing applications, speed really counts, and avoidable overhead is bad. If an ASIO driver is available for your sound card, take it; it will provide the best performance in most cases… with a notable exception: if you have Cubase or Nuendo installed, these will install an ASIO emulation driver called "ASIO Multimedia Driver", which adds an additional layer *above* the Windows MME driver. Avoid this ASIO driver whenever possible, it gives a horrible performance.

The **Buffer Size** combo box lets you select between some buffer sizes; these are carefully selected for their property that their multiples exactly fit into one second at 44.1kHz. Now, that may be nice, but… first of all, 44.1kHz isn't mandatory for ASIO drivers, and some of these mandate other buffer sizes.

Apart from the predefined values, you can enter any (reasonable) buffer size you want into this combo box.
Here's how my main development computer's configuration looks like (this PC uses a Terratec DMX 6fire 24/96 card):

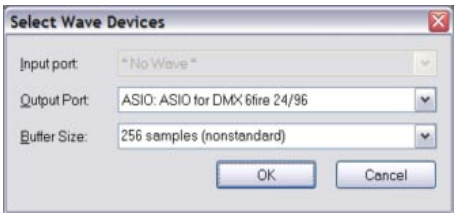


Figure 4: Setup for Terratec DMX 6fire 24/96

Experiment with the buffer size; if it is too large, you'll hear noticeable delays between the triggering of a note and its actual appearance at the speakers. If it is too small, the overhead introduced by the frequent buffer handling might become too much for your poor little computer; in that case, it starts to skip processing some buffers since it doesn't have enough *time* for it. This results in an occasionally audible crackling noise. In this case, increase the buffer size until it goes away.

Once you have configured an ASIO driver, the following two menu entries can be used:

ASIO Control Panel

Here, you can call up the selected ASIO Driver's configuration panel. This varies greatly between the various drivers and is not part of VSTHost.

ASIO Channel Selection

Normally, VSTHost operates with as many channels as the Wave device drivers permit. If you only need Stereo operation, however, you can use this to select a stereo pair to use. Note to self: does this really work any more? Haven't tried it for ages…

Selecting the menu entry brings up the following dialog:

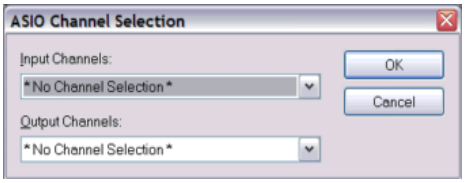


Figure 5: ASIO Channel Selection Dialog

Here, you can select a stereo input and output pair to use.

OK, we've set up our Wave devices… so now what?

MIDI Configuration

If you're only using VSTHost to load some VST Effects, you don't need this step. Most effects don't rely on MIDI communication. VST Instruments, however, need MIDI data. While VSTHost has a built-in keyboard bar (see "Keyboard Bar" on page 27 for details) that allows you to trigger MIDI notes with the mouse, this can't really be considered a good solution; an external keyboard (the one with the black and white keys, not your computer's ☺) is far superior. To tell VSTHost how to find this external keyboard, you have to define the MIDI devices it should use. So, open the **MIDI Device Configuration** dialog by choosing its menu entry:

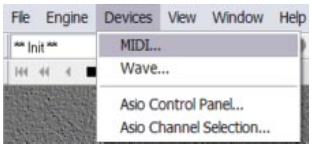


Figure 6: MIDI Device Settings menu entry

… which opens the following dialog:

Figure 7: MIDI Device Configuration Dialog

Now *this* is a more complex dialog. It has 4 tabs to define all necessary parameters. Let's start from left to right:

MIDI Input Devices

This tab is shown in the above figure, so have a look at it there. Here, you can select the MIDI Input Device(s) that VSTHost should use. You can select one of them by simply clicking on it; to select a range, click on the first and then shift-click on the last; to add or remove a specific selection, control-click on it.

MIDI Output Devices

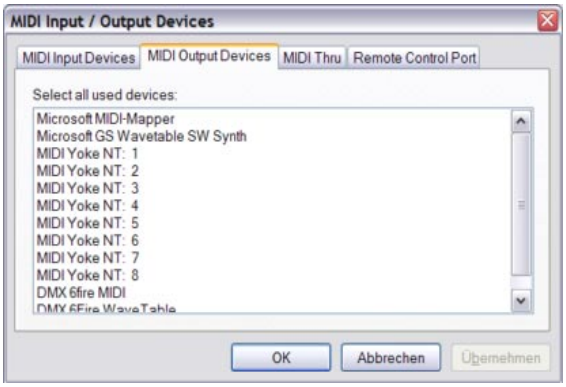


Figure 8: MIDI Output Device selection

Here, you can select the MIDI Output Device(s) that VSTHost should use. You can select one of them by simply clicking on it; to select a range, click on the first and then shift-click on the last; to add or remove a specific selection, control-click on it.

MIDI Thru

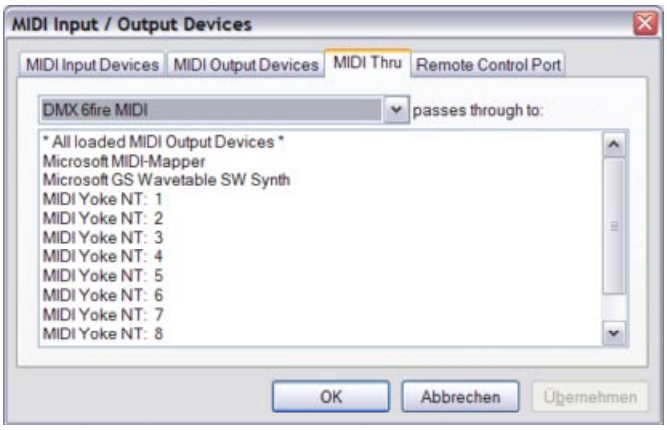


Figure 9: MIDI Thru Definitions

Here, you can define VSTHost's "Soft MIDI Thru" behaviour. Since PC sound cards normally don't have a MIDI Thru connector, the software has to provide it. Since VSTHost can load multiple MIDI input and output devices, a general MIDI Thru setting would be inappropriate; it might lead to MIDI feedback loops. Therefore, you can separately define the MIDI Output devices for each MIDI Input device that it forwards incoming MIDI messages to. You can select one of them by simply clicking on it; to select a range, click on the first and then shift-click on the last; to add or remove a specific selection, control-click on it.

Note: while you can define as many combinations as you like, VSTHost doesn't remember them all (spoilsport that it is). As soon as you close the dialog, it loads all requested devices and their MIDI Thru settings. All MIDI Thru settings for devices that are *not* loaded (either because they haven't been selected or because they could not be loaded for some reason) are lost.

Remote Control Port

Figure 10: Remote Control Port/Channel Definition

Here, you can define VSTHost's *Remote Control Port / Channel*. In addition to passing MIDI messages to the loaded PlugIns, VSTHost can be remote-controlled by MIDI messages, too. Here, you can define a MIDI Input Device (mental note to self: *finally* decide on "port" or "device" nomenclature!) that controls VSTHost. It has to be one of the devices selected on the **MIDI Input Devices** tab, otherwise VSTHost forgets this setting when you close the dialog. The default setting of "---" means that there's no Remote Control port.

The **Channel** combo box lets you define a specific channel; the default setting of "---" means that VSTHost should interpret *all* channels of the selected MIDI Input port as remote control channels. In this case, it completely swallows MIDI Input on this port. If you want to control VSTHost remotely, but still use 15 of the 16 possible channels for MIDI communication with loaded PlugIns, you can define a specific MIDI channel as Remote Channel. MIDI data on this channel is swallowed by VSTHost, data on all other channels can be received by PlugIns.

At the time being, the only thing that can be remotely controlled is the loaded VSTHost Program (see “Load Program” on page 14 for details).

Other Configuration Tasks

While there are quite some other things that can be configured in VSTHost, they are not installation-related, so they’ll be described later, when it is appropriate.

Operation

Command line parameters

Just to prevent mouse junkies from going "Eeeek!" – no, VSTHost is not a command line oriented text program, it is GUI-oriented; but you can give it some command line parameters for special occasions. So, let's describe them in a good old-fashioned style…

Syntax

```
VSTHost [/noload] [/nosave] [/noaudio] [/nomidi] [plugin]
```

The []'s mean that all parameters are optional. Here's their meaning:

/noload	Forces VSTHost to skip the initial loading of the previous (or default) setup when it comes up. This, together with the **/nosave** parameter, can be used to quickly debug a PlugIn.
/nosave	Forces VSTHost to skip the saving of the complete current setup upon program termination. Normally used together with the **/noload** parameter in a debugging situation.
/noaudio	Forces VSTHost to come up without any loaded audio driver. This can help to determine the cause if VSTHost inexplicably dies while initializing. At least in one case, VSTHost tried to load a (still installed) driver for a sound card that was *removed* from the computer – bang…
/nomidi	Forces VSTHost to come up without any loaded MIDI driver. Same reason as for **/noaudio**.
plugin	The complete pathname of a PlugIn to load. This is normally used in a debugging environment, but also if you drag a PlugIn onto VSTHost's icon.

Main Window

Now that we've finally started and parameterized VSTHost, let's return to our initial picture:

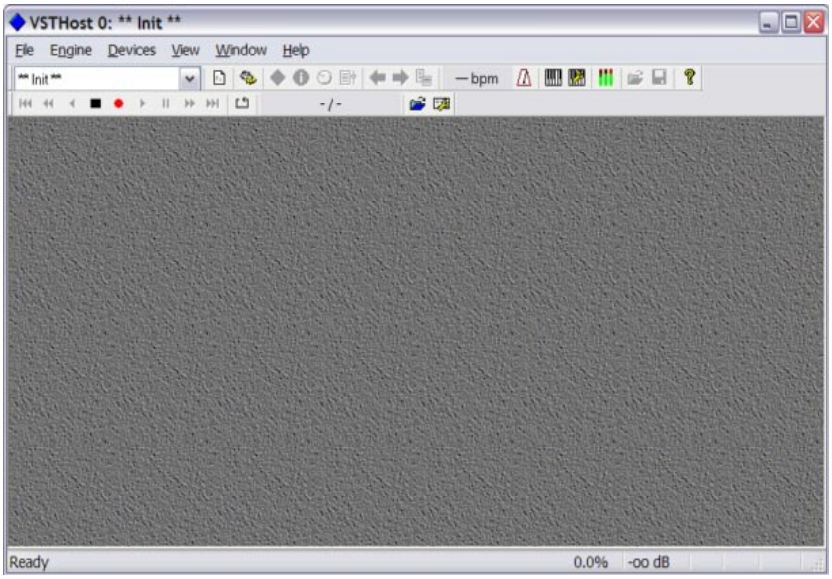


Figure 11: VSTHost Initial (again)

Still no sound… let's change that. The first thing that we need to do is to force VSTHost into operation; being a lazy program, it comes up with the host engine stopped. To start it, simply click on the nice little button on the toolbar. Don't worry, it will be described later. Voilà – VSTHost works.

Menu Entries

Figure 12: VSTHost's unimpressive Main Menu

Practically all of VSTHost's operations can be controlled from the menu, so let's examine that in detail.

File Menu

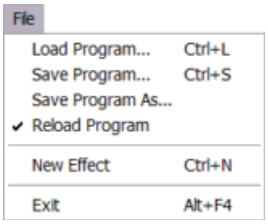


Figure 13: File Menu

Here, you can load and save VSTHost setups (called "programs") and load new PlugIns into the current program. Plus, of course, terminate VSTHost ☺.

Load Program

This menu entry opens the following dialog:

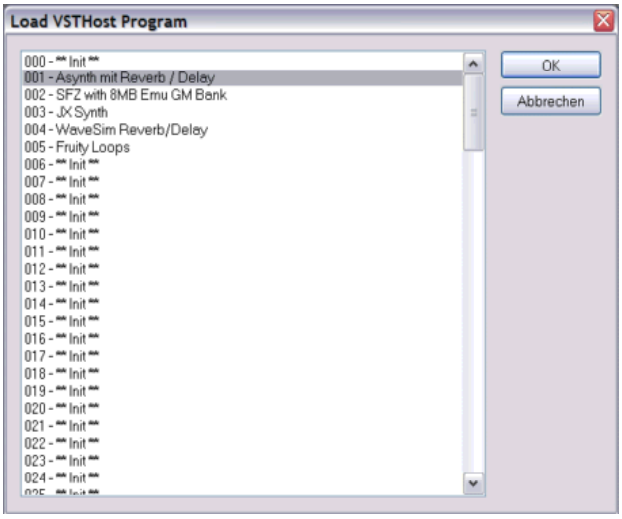


Figure 14: Load VSTHost Program Dialog

Here, you can select one of the 129 possible programs to be loaded.

This operation can also be performed by sending a **Program Change** MIDI message to the Remote Control Channel, with one exception: program **000** is a special program; this cannot be selected by remote operation. Unless **Reload Program** is checked on the File menu, VSTHost loads this program as its initial setup. This way, you can define a nice default environment.

A program's initial name is "** Init **" – not very inventive, I have to admit – and can be changed if you save a program with **Save As…** (described below).

Save Program

Selecting this menu entry saves the current program. Unless the **/nosave** parameter is used, VSTHost always saves the current program when it exits. When another program is loaded, the current program is saved in any case before loading the new one.

Save Program As

This menu entry opens the following dialog:

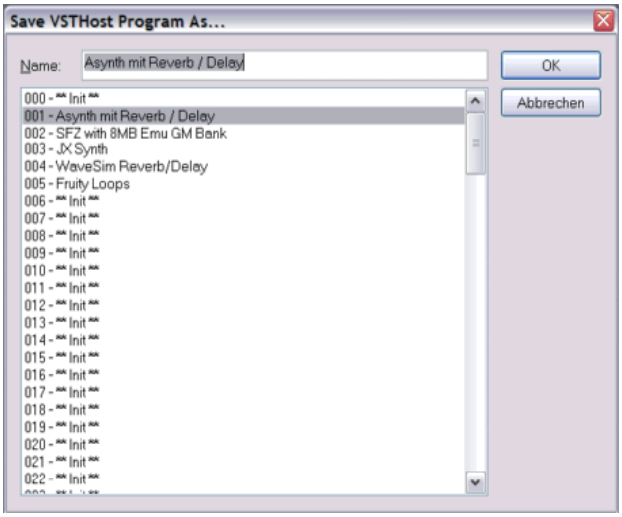


Figure 15: Save VSTHost Program As... Dialog

Here, you can save the current configuration to a(nother) program. After having selected the new position in the list box, you can give the program a descriptive name. Pressing OK saves the program to the new position and automatically uses it as the new current program.

Reload Program

This menu entry can be toggled; if it is checked, VSTHost reloads the last used program when it is started the next time (unless the **/noload** parameter is given, of course). If it is unchecked, VSTHost always loads the default program (Program **000**) when it starts.

New Effect

Now it gets interesting… if this menu entry is selected, VSTHost opens a file selection dialog box that allows you to locate a PlugIn that you want to add to the current program. Since VSTHost is a simple little program, it doesn't perform lengthy "Where are the PlugIns?" scans upon program start to present a nice, preformatted list of available PlugIns; it simply allows you to select the file containing the PlugIn. In Windows, PlugIns are normally simple DLLs, i.e., their names end with ".dll".

If a "big player" (Cubase, Nuendo, etc.) has already been installed on your computer, it has set up a directory where VST PlugIns are to be installed; normally, this directory is stored in the registry under **HKEY_LOCAL_MACHINE\Software\VST** in the value **VSTPluginsPath**. If VSTHost finds this value, it starts the file selection dialog there; if not, it uses the current directory.

In any case, once a PlugIn has been selected and loaded, VSTHost remembers the directory where it took it from as its new start point.

Anyway, once you loaded a PlugIn, VSTHost presents it in a little window on its main client area, like this (the example uses ASynth, a very good freeware synth):

Figure 16: Example for a loaded PlugIn's Main window

There are some buttons on that window. All of them are just alternatives to entries of the Effect menu (see below) and the main toolbar (see "Toolbar" on page 25 for this). If you move the mouse over a button and leave it there, a little popup will be displayed that tells what the respective button can be used for. The Main window can be dragged around on the screen with the mouse to any location you

like. Since the links between PlugIns are displayed there, too, this can be used to arrange the PlugIn windows in a way that shows the relations between them, like this:

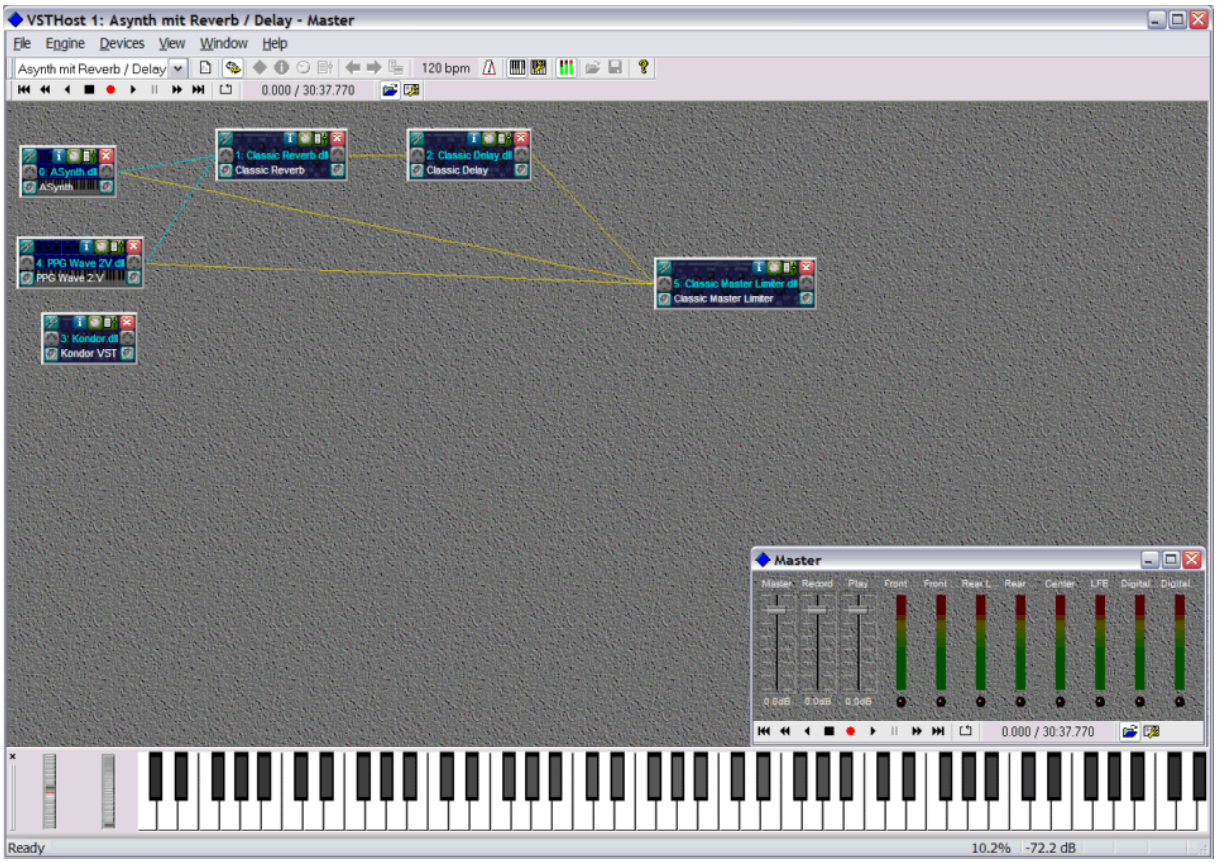


Figure 17: VSTHost Main Window with some PlugIns loaded

Exit

Does what it says and terminates VSTHost.

Effect Menu

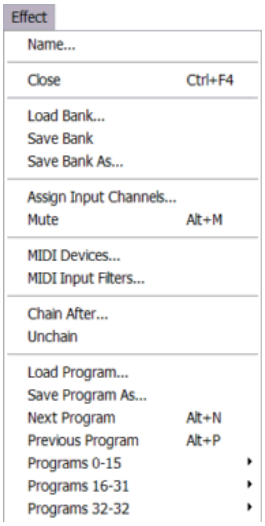


Figure 18: Effect Menu

This menu is only visible if an Effect window is currently selected (main window, edit window, parameter window, or info window); more on these later in the “Toolbar” section on page 25.

Effect, in this case, is used to designate any kind of PlugIn (effects and/or VSTis) here; the name is used only for historical reasons. I'll probably change it to the more generic "PlugIn" sooner or later.

Name

Selecting this menu entry opens a little dialog where you can enter a new name for the current program used in this PlugIn:

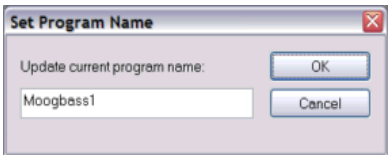


Figure 19: Set Program Name Dialog

Please note that while the change is visible at once after you have pressed OK, you need to save the PlugIn's Bank or Program to a file to really make it permanent.

Another ambiguity… **Program** is used with two different meanings in VSTHost. There are VSTHost "programs" (collections of PlugIns and their interconnections), and there are PlugIn "programs" (different configurations of that specific PlugIn). I'll presumably change the VSTHost "program" to "performance" sooner or later.

Close

Closes the currently selected PlugIn and all of its windows.

Load Bank

Selecting this menu entry opens a dialog where you can load a program bank (normally a file with extension ".fxb") into the current PlugIn. Once you specify one, VSTHost loads this bank into the PlugIn automatically whenever the program containing the PlugIn is loaded.

Save Bank

Saves the current PlugIn's program bank into the file. If there is no current program bank defined for the PlugIn (i.e., no **Load Bank** operation has been done before), it acts like **Save Bank As**.

Save Bank As

Selecting this menu entry opens a dialog where you can select a new file name (normally a file with extension ".fxb") to save the current PlugIn's program bank into.

Assign Input Channels

This menu entry is only selectable for PlugIns that have one or more audio input channels. Selecting this menu entry opens the following dialog:

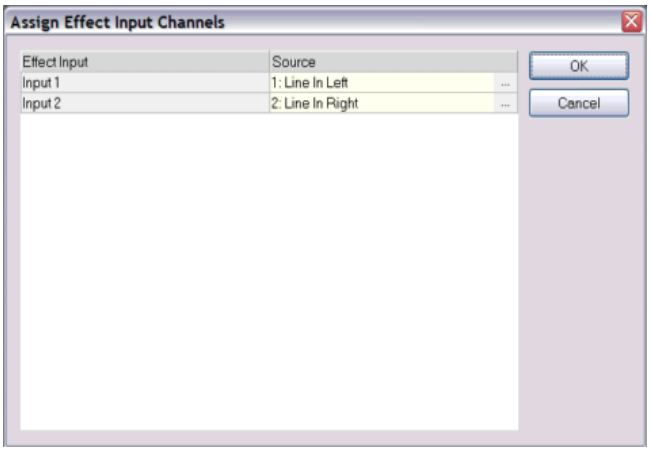


Figure 20: Assign Input Channels Dialog

Here, you can define the assignment between the audio channels provided by VSTHost and the PlugIn's input channels. Since it happens quite often that VSTHost offers more channels than needed by the PlugIn, it can be convenient to select which ones are really processed. On the left side, the PlugIn's input channels are listed; on the right side, you can select one of the channels provided by VSTHost for each of the input channels. Click on the "**…**" on the right side of the Source column to select from the available channels.

Note: this setting defines how the PlugIn operates when it's not part of a *PlugIn Chain* (for more on these see "Chain After" on page 20); if it receives its input from (an)other PlugIn(s), the assignment for *that* is defined on the "Chain Effect After…" dialog. Complicated? Yes, but this way there are far less restrictions on what goes where.

Mute

Aaaah, a very important menu entry. Selecting this (un)mutes the current effect.

MIDI Devices

Selecting this menu entry opens the following dialog:

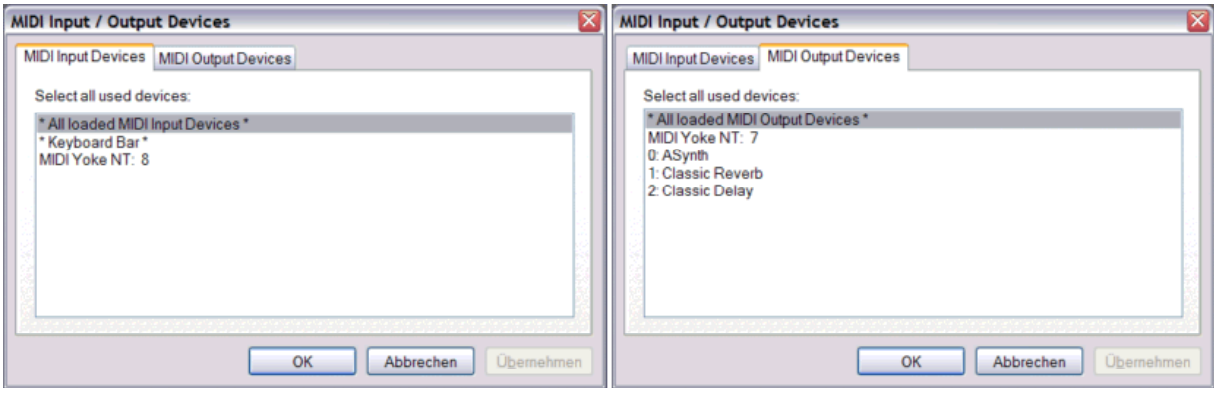


Figure 21: PlugIn's MIDI Input/Output Devices Dialog

Here, you can define the MIDI Devices for this specific PlugIn.

On the **MIDI Input Devices** tab, you can define the MIDI Input Devices that are used by this PlugIn; the default is to react on messages from all loaded devices. The Keyboard Bar is treated like a MIDI

Input device here. You can select one of them by simply clicking on it; to select a range, click on the first and then shift-click on the last; to add or remove a specific selection, control-click on it.

On the **MIDI Output Devices** tab, you can define the MIDI Output devices that this PlugIn sends MIDI messages to. Normally, it sends to all loaded MIDI Output Devices. If you look closely at the above figure, you'll see that there's another possibility hidden here: a PlugIn can send MIDI messages to any other PlugIn, too. This, however, has to be configured separately; that is, simply selecting "All loaded MIDI Output devices" does *not* send to the other plugins; if you want that, you have to select each of the connections *in addition* to the MIDI Output devices. You can select one of them by simply clicking on it; to select a range, click on the first and then shift-click on the last; to add or remove a specific selection, control-click on it.

MIDI Input Filters

Selecting this menu entry opens the following dialog:

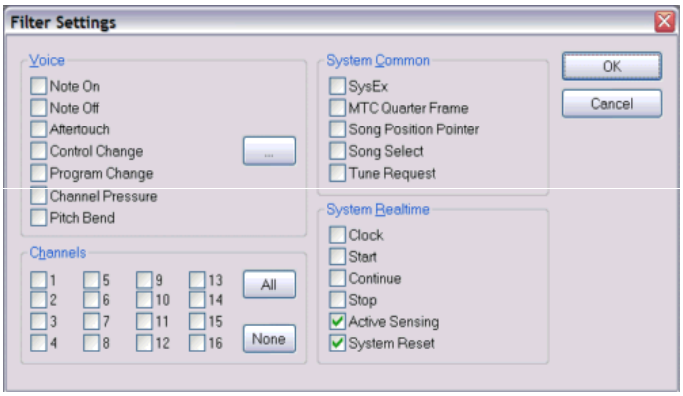


Figure 22: MIDI Input Filters Dialog

Here, you can select MIDI Messages that are *filtered*; i.e., they are removed from the MIDI stream that is sent to this specific PlugIn. Common usages include:

- Disallowing messages from specific MIDI channels; this can be used to separately control up to 16 PlugIns from a single MIDI Input Device
- Preventing Program Change messages to change the PlugIn's program
- Inhibiting certain controllers
- Preventing SysEx messages to reach the PlugIn

If you want to filter only some Control Change messages, you can define them by pressing the "**...**" Button, which opens the following subdialog:

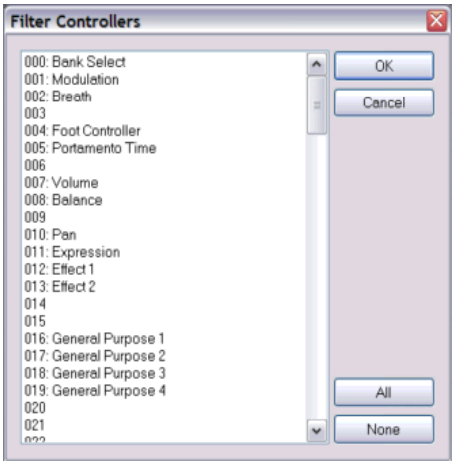


Figure 23: Filter Controllers Dialog

Here, any combination of Control Change messages can be filtered.

Chain After

Selecting this menu entry allows to define PlugIn *Chains*; a chain, in VSTHost, is a sequence of PlugIns that are linked together. The first PlugIn in the chain processes audio input from VSTHost (unless it's an Instrument), then its output is passed on to the next element(s) in the chain, and so on, until a PlugIn does not have a successor linked to it; this PlugIn's output contributes to the final audio output generated by VSTHost.

There are two kinds of links between PlugIns: **Insert** and **Send** links. These have been named after the roughly corresponding concept found in more traditional sequencer packages, which borrowed it from the mixing consoles that have been around for many years. An **Insert** link means that the output from the first PlugIn is only sent to the second PlugIn; it does not contribute to the final output. A **Send** link means that the output from the first PlugIn is sent to the second PlugIn, but it *also* contributes to the final output. That is, a **Send** link creates a "fork" – from here on, the first effect's output contributes to the final output in more than one way. There is, of course, an exception to the above rule… if a PlugIn is linked to more than one successor, *all* of the links have to be Send links. If there are one or more Insert links, this PlugIn does not contribute to the final output.

You can send up complicated setups this way. Each PlugIn can be linked to a multitude of other PlugIns.

Selecting the **Chain After** menu entry opens the following dialog:

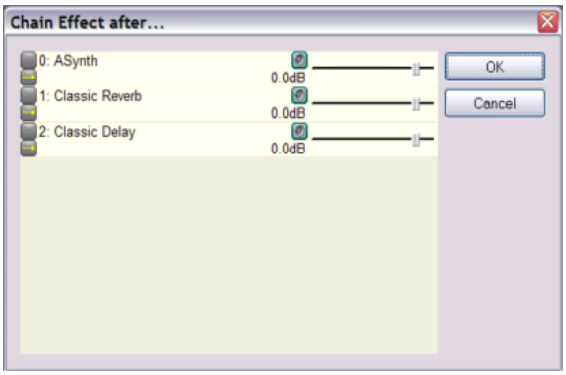


Figure 24: Chain Effect After Dialog

This dialog lists all PlugIns that the current PlugIn can be "chained after", or "linked to". It does not include PlugIns that include the current PlugIn in their predecessor chains, because that would lead to recursive setups, which would kill VSTHost.

There are some buttons available for each entry:

- this button is used to link the current PlugIn after the selected one. Selected PlugIns look like that: 

- the button has changed to , and the line is shown in a different color.

- this button is used to define the *type* of the link. If it shows one yellow arrow, like on the left, this link is an **Insert** link; if it shows a light yellow arrow and a blue arrow, like this: , the link is a **Send** link.

- this button is used to define the Input Channel assignment for the link. See "Assign Input Channels" on page 18 for more details.

Then, there's a little slider that allows to define the level passed through this link; if, for example, you want to chain a Reverb PlugIn after an Instrument PlugIn, you would define the link as a Send Link and reduce the level in this link to, say, -10dB. In this way, both the original instrument and the reverb are heard, but the reverb contributes much less to the overall audio output.

Unchain

This is a bit tricky… the **Chain After** dialog can be used to set up the PlugIn(s) that a PlugIn is linked to. Unselecting the link on that dialog removes it, breaking the chain if there's any other PlugIn linked to the current one.

Selecting the **Unchain** menu item, however, removes all links to and from the current PlugIn – but it leaves the rest of the chain intact and "glues" predecessors and successors together. That is, all predecessors and successors remain linked, just the current PlugIn is entirely removed from the chain.

Load Program

Selecting this menu entry opens a dialog where you can load a program (normally a file with extension ".fxp") into the current PlugIn.

Save Program As

Selecting this menu entry opens a dialog where you can save the current program of the current PlugIn into a file (normally a file with extension ".fxp")

Next Program

Selecting this menu item changes the PlugIn's current program to the next program in the list.

Previous Program

Selecting this menu item changes the PlugIn's current program to the previous program in the list.

Programs mm-nn

A PlugIn can define how many programs it supports. Some have no program, some have one, some have 10, some have 128… and so on. Since a single list of potentially thousands of programs would be rather awkward to use, VSTHost splits it into manageable parts and displays a list of submenus for these. Selecting one of the items on the submenu loads the selected program into the PlugIn.

The above three menu items change the current program for a PlugIn; VSTHost remembers this program number, together with an eventually loaded Program Bank. VSTHost loads this program into the PlugIn automatically whenever the VSTHost program containing the PlugIn is loaded.

Engine Menu

This menu contains entries that control the overall operation of the VST Audio Engine.

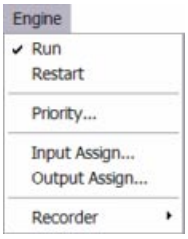


Figure 25: Engine Menu

Run

Selecting this menu entry toggles the engine's run mode. When VSTHost is started the first time, the engine is turned off; selecting this menu entry, or clicking on the corresponding toolbar button, turns it on.

This menu entry is primarily used for debugging purposes; sometimes, if you just want to debug various display aspects of a PlugIn, it's not necessary to keep the full audio engine running in the background, consuming tons of precious CPU cycles.

Restart

Stops and restarts the audio engine. Mainly useful if something goes wrong (buffer size too small, for example, leading to synchronization errors between VSTHost and the ASIO driver).

Priority

Selecting this menu entry opens the following dialog:

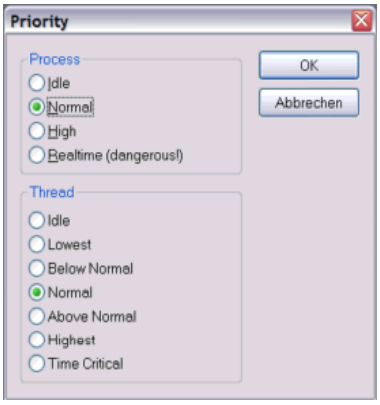


Figure 26: Priority Dialog

Normally, this can be left untouched; VSTHost automatically sets the priority to "very high" before loading an ASIO driver and then reverts to the configured setting. If the default settings don't work satisfactorily with your specific configuration, you can play with VSTHost's settings until you find a solution that works. Be careful, however – setting the process priority to **Realtime** and the thread priority to **Time Critical** can lead to situations where VSTHost consumes all of the computer's CPU time and doesn't let anyone else work – the computer freezes. Especially dangerous on slower machines. You've been warned – "Don't press this button!" ☺

Input Assign

Selecting this menu entry opens the following dialog:

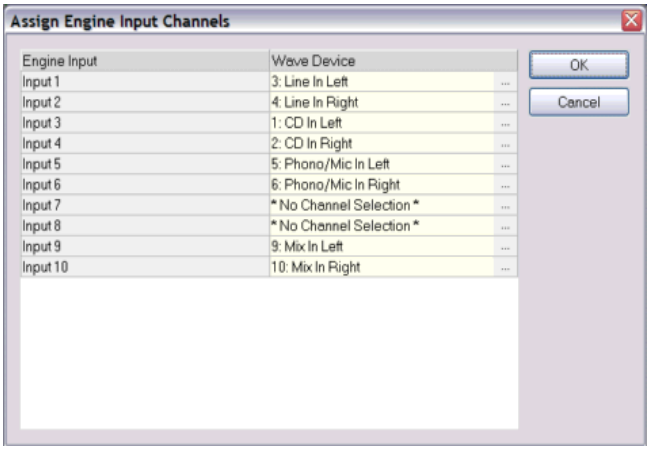


Figure 27: Assign Engine Input Channels Dialog

Just as a PlugIn can have its inputs reassigned (see "Assign Input Channels" on page 18), so can VSTHosts Audio Engine.

On the left side, the channels used by VSTHost's VST audio engine are listed; on the right side, you can select one of the channels provided by the loaded Wave Input device. Click on the "**…**" on the right side of the Source column to select from the available channels.

This allows a reassignment, or removal of certain unwanted channels. See the above figure – the DMX 6fire 24/96 ASIO driver presents the **CD In Left** and **Right** channels in positions 1 and 2; I prefer to have the Line Inputs there so that loaded Effect PlugIns automatically use these if their input assignment is not specifically set.

Output Assign

Selecting this menu entry opens up the following, already familiar, dialog:

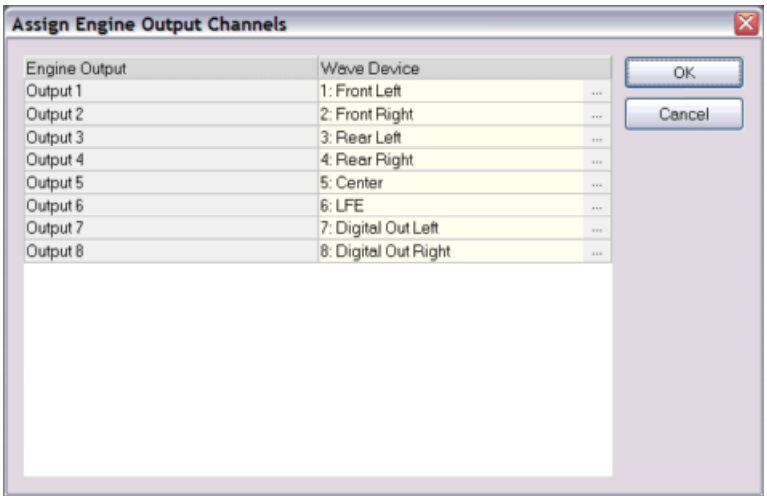


Figure 28: Assign Engine Output Channels Dialog

This works just like **Input Assign** above, but in the other direction. Here, you can assign an output channel provided by the Wave Output device to each of the VSTHost audio engine's internal channels.

Recorder

This is a submenu to configure and run VSTHost's built-in Wave Player and Recorder.

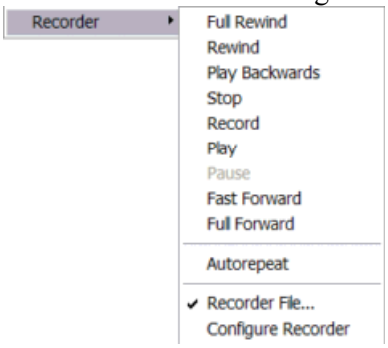


Figure 29: Recorder Submenu

VSTHost contains two separate entities for that: a **Wave Player** and a **Wave Recorder**. Both of these are inserted into the audio engine just before the sound data are sent to the output device. The Wave Player's output is added to the Plugins' output; the accumulated output can then be recorded by the Wave Recorder, allowing ping-pong style recordings.

The same menu is also available inform of a toolbar, which also offers a display of current and total time:

Figure 30: Recorder Toolbar

I admit freely that this is not a very elegant solution, but there are lots of sophisticated sequencer packages on the market. This little one-man spare time project called VSTHost doesn't, and can't

compete with them in their own arena. If you need nifty audio recording features, VSTHost isn't the right program. The Wave Recorder and Player are just little additions that allow capturing a jam session, for example, or allow the playback of a prerecorded track while you play some VSTi PlugIns to it.

The Wave Player mimics a tape recorder, just like the ones you all might know since early childhood, with some little add-ons – you can, for example, play the loaded file backwards, and you can record and playback at the same time.

Whenever the **Stop** button or menu entry is clicked while recording, VSTHost allows you to determine whether and under which name the recording should be saved.

Some of the menu items, however, do require a little explanation:

Autorepeat

This menu entry can be used to toggle between normal playback mode, in which the Player stops when it reaches the end of the input file, and auto-repeat mode, where it automatically restarts playback.

Recorder File

Here, you can define the file that the Wave Player uses. Most of the transport menu entries / toolbar buttons will be grayed out until a valid Wave file has been loaded. Note to self: rename that to "Player File" for greater clarity…

Configure Recorder

Selecting this menu item brings up the following dialog:

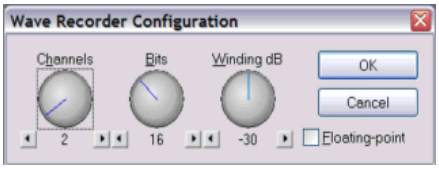


Figure 31: Wave Recorder Configuration Dialog

Here, you can define the format of files that are *recorded* by VSTHost. The default setting is rather conservative – 2 channels, 16 bit integer. This format has the big advantage that the generated files are comparatively small and can be read by virtually every wave file editor package in existence. It isn't the best format, however…

Most modern sound cards for musicians can do much better. They can deliver accurate sampling and playback of 24 bit audio streams, at potentially dozens of channels – far more than VSTHost's CD-compatible default format permits.

The **Channels** setting should not exceed the available number of output channels in the VSTHost audio engine – while you can set it to up to 32 channels, most of them would simply contain silence. A rather costly silence, since each little sample of silence occupies at least a byte on your hard disk, more likely two to four bytes.

The **Bits** setting is a bit misleading. You can set the number of bits per sample, but internally VSTHost treats them the following way:

- **8** generates 8-bit unsigned samples; like in the old SoundBlaster 1 days ☺
- **9-16** generates 16-bit signed samples
- **17-24** generates 24-bit signed samples
- **25-32** generates 32-bit samples

Floating-point, if selected, overrides the **Bits** setting. In this case, the 32-bit floating point data that are used internally by VSTHost are directly recorded. This provides the most accurate recording that also deals very gracefully with clipping, but produces rather large recordings.

Winding dB is actually a setting for the Wave Player. While rewinding or fast forwarding in the wave file, you can listen to the data currently being under the "virtual head" of this simulated tape recorder. With this setting, you can define the playback level which is used while winding. Setting this to **-60** dB disables the feature, thereby saving some CPU cycles.

Devices Menu

All entries of this menu have already been discussed in the Configuration section (see "Configuration" on page 7 for details).

View Menu

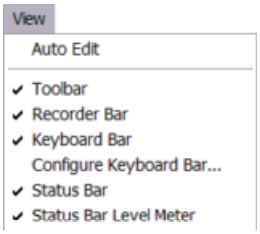


Figure 32: View Menu

This menu can be used to configure VSTHost's general layout.

Auto Edit

This menu entry defines how new PlugIns are treated; when it is checked, and you load a new PlugIn that allows to open an Editor window (see "Toolbar" below), this editor window is opened automatically.

Toolbar

Toggles the main toolbar display. This is the main toolbar:

Figure 33: Main Toolbar

First, obviously, there's a combo box that allows direct selection of a VSTHost program. This works just like the **Load Program** menu entry in the **File** menu (see "Load Program" on page 14 for details).


acts like the **New Effect** menu entry (see „New Effect" on page 15 for details).


acts like the **Run** menu entry (see „Run" on page 21 for details).


activates the main window of a PlugIn, if any of the others (editor, parameter, information) is currently active.


Opens or activates the currently selected PlugIn's **Information Window**; this window contains valuable(?) information about the PlugIn and its properties, like this:

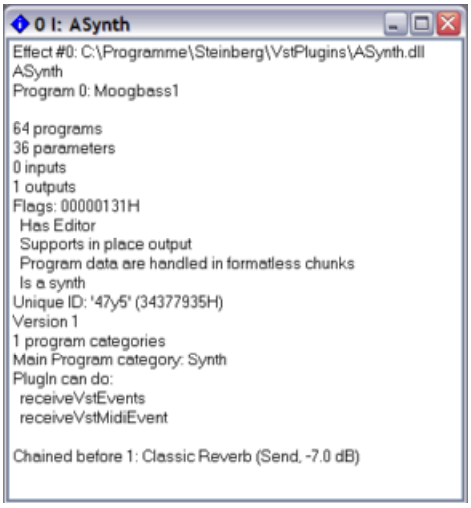


Figure 34: PlugIn Information Window

Opens or activates the currently selected PlugIn's **Editor Window**; this window is provided by the PlugIn for configuration purposes. Since it's provided by the PlugIn, the layout of this window can vary. Greatly. And since the window behavior is controlled by the PlugIn, this can vary greatly, too… anyway, Here's an example:

Figure 35: Example PlugIn Editor Window

Opens or activates the currently selected PlugIn's **Parameter Window**; since not all PlugIns provide an Editor window, this can be the only possibility to set the PlugIn's parameters. Some PlugIns can only be configured on the Parameter window; others refuse that completely and only accept input from the Editor window. Anyway, you can open both in VSTHost.

Here's an example:

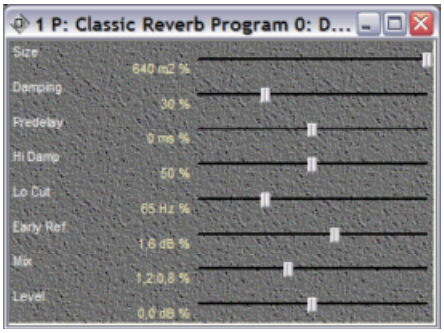


Figure 36: Example PlugIn Parameter Window

The displayed parameters and their values, of course, vary from PlugIn to PlugIn; while the layout is done by VSTHost, the parameters, their value range and display are provided by the PlugIn. If the PlugIn has more than 100 parameters, the window contains a menu at the top that allows to select a parameter range to be displayed.

VSTHost remembers the opened windows and their positions for each PlugIn.

These buttons correspond to the **Previous Program** and **Next Program** menu entries (see "Next Program" and "Previous Program" on page 21 for details).

This button opens a popup menu for the current PlugIn that allows selection of a specific

program. It corresponds to the **Program mm-nn** menu entries in the **Effect** menu, but is displayed over the active PlugIn's main window.

120 bpm This field shows the currently defined speed. The default used by VSTHost is 120 BPM. This value is reported to loaded PlugIns. Clicking on the speed puts the value into parentheses (like "**(120 bpm)**"). When parenthesized, VSTHost stops and resets the transport information. Clicking on the speed again removes the parentheses, and VSTHost starts to send transport information to the PlugIns again.

This button, in case you didn't find that out by yourself, resembles a metronome. With this, you can define the speed that VSTHost reports to the loaded PlugIns. Pressing it opens the following dialog:

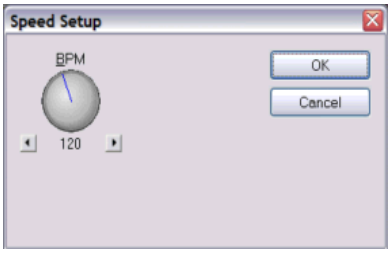


Figure 37: Speed Setup Dialog

Pretty empty, huh..? Well, since VSTHost isn't a really big sequencer, it doesn't need to configure more. Not yet, that is.

This button shows or hides VSTHost's **Keyboard Bar**. See "Keyboard Bar" on page 27 for details.

This button is used to configure the Keyboard Bar. See "Configure Keyboard Bar" on page 28 for details.

This button opens the **Master** window. See "Master" on page 31 for details.

This button is used to load a program bank into the currently selected PlugIn. See "Load Bank" on page 17 for details.

This button is used to save the current program bank of the currently selected PlugIn to disk. See "Save Bank" on page 17 for details.

The most important button of them all. See "About VSTHost" on page 31 for details.

Recorder Bar

Selecting this menu item hides or shows the recorder toolbar display. The Recorder toolbar has already been discussed (see "Recorder" on page 23 for details).

Keyboard Bar

Selecting this menu item hides or shows the Keyboard bar. The keyboard bar, when opened for the first time, looks like this:

Figure 38: Keyboard Bar

It comes up with a Pitch Wheel, a Mod Wheel, and a keyboard with 61 keys, covering the bottom area of VSTHost's window. This bar can be configured to a very high degree (see "Configure Keyboard Bar" below for details), and you can grab it with the mouse and fix it on top of the window area or on the bottom (default), or anywhere else on the screen if you want to. VSTHost remembers the position.

The Keyboard bar can be used as a "poor man's MIDI keyboard"; you can enter MIDI messages by mouse or (computer) keyboard with it. You can configure for each PlugIn whether it reacts on messages from the keyboard Bar (see "MIDI Devices" on page 18 for details).

The following keys on the PC keyboard can be used to trigger MIDI Notes, if the Keyboard Bar is active (i.e., activated by clicking on it with the mouse):

Figure 39: PC keyboard keys used for MIDI Note generation

In addition, the following keys can be used:

Left shift, Right shift	transposes the PC keyboard's range two octaves down/up
Ins, Del	increment/decrement pitch wheel data
Home, End	increment/decrement modulation wheel
PgUp, PgDn	increment/decrement key velocity
Left, Right	decrement/increment upper keyboard octave
Down, Up	decrement/increment lower keyboard octave

Configure Keyboard Bar

Selecting this menu entry opens the following dialog:

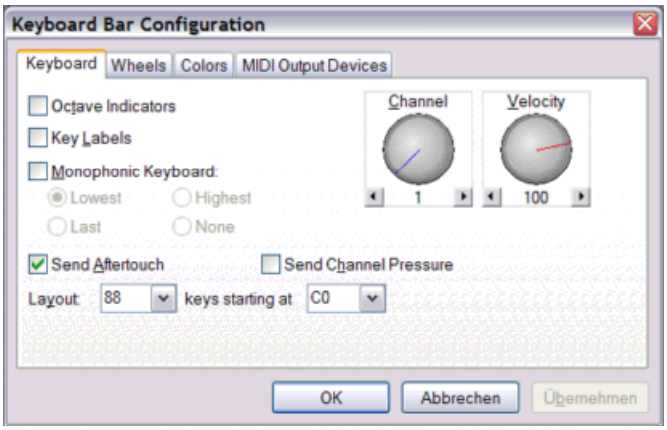


Figure 40: Keyboard Bar Configuration Dialog, Keyboard Tab

Huh. Yet another of this tabbed configuration dialogs. Isn't it depressing to see how many options there are in such a simple program? ☺

Anyway, this is the **Keyboard** tab where you can define the general layout of the Keyboard Bar.

Octave Indicators

If this item is checked, the Keyboard Bar shows 2 **octave indicators** below the keyboard. These are two (normally gray) bars that indicate the keys that can be played on your PC's keyboard. You can drag these bars around with the mouse to change the octaves used by the upper and lower key range on your PC keyboard (see "Keyboard Bar" above for the usable keys).

Key Labels

Gimmick for people that don't use keyboards very often. You can display the note names on the keys.

Monophonic Keyboard

This check box can be used to switch between polyphonic and monophonic keyboard mode.

In monophonic keyboard mode, every pressed key terminates the previous note. The radio buttons govern what happens when the current note is released, and there are still other notes held:

Lowest	the lowest currently pressed key determines the current note
Highest	the highest currently pressed key determines the current note
Last	the last note pressed before the terminated note determines the current note
None	still pressed keys are ignored

Channel

This knob defines the MIDI channel used for MIDI messages generated by the Keyboard Bar.

Velocity

This knob defines the velocity used if you trigger notes with the PC keyboard, which cannot send velocity information (unfortunately… I'd *love* a keyboard where the attack can be used to define whether a character is printed **bold** in a word processor, or where the pressure defines the autorepeat rate, or… ☺). Can be redefined by adding a **Velocity Wheel** to the keyboard, or by the PgUp/PgDn keys.

Send Aftertouch

This check box can be used to activate the sending of (Polyphonic) Key Aftertouch MIDI messages if you slide the mouse up and down a key while keeping one of the mouse buttons pressed. Note that not all PlugIns will be able to react on Key Aftertouch, since devices that offer it are relatively rare.

Send Channel Pressure

This check box can be used to activate the sending of Channel Pressure MIDI messages if you slide the mouse up and down a key while keeping one of the mouse buttons pressed.

Layout

Here, you can define how many keys starting at which key are displayed on the Keyboard Bar. If the predefined values in the combo boxes don't suit your needs, you can enter any reasonable value you like.

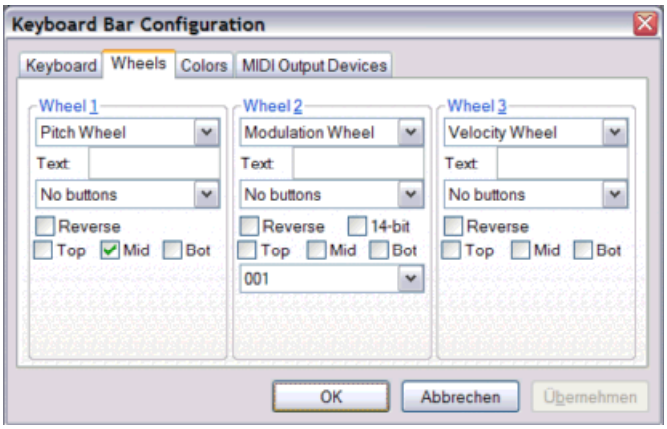


Figure 41: Keyboard Bar Configuration Dialog, Wheels Tab

On this tab, the wheels shown by the Keyboard Bar can be configured. There are three types of wheels:

Pitch Wheel	Can be used to send Pitch Wheel MIDI messages. Auto-centers when the wheel is released.
Modulation Wheel	Can be used to send a configured Continuous Controller MIDI message. By default, this is set to send **Mod Wheel** data (CC#1), but it can set to any other CC# you like (or need).
Velocity Wheel	Can be used to increase or decrease the velocity for MIDI messages generated with the PC keyboard.

You can set quite a lot of options for these wheels; have fun experimenting!

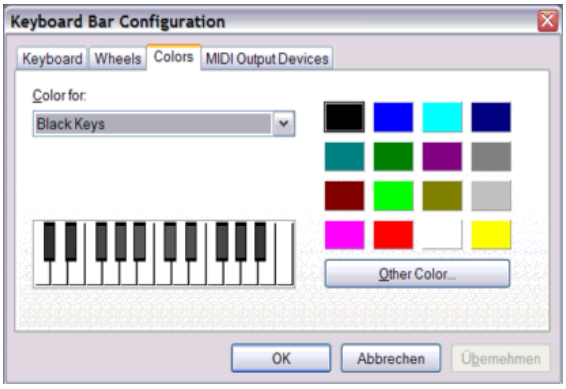


Figure 42: Keyboard Bar Configuration Dialog, Colors Tab

Here, you can redefine the color of the black and white keys to any color scheme you like, Again, have fun experimenting!

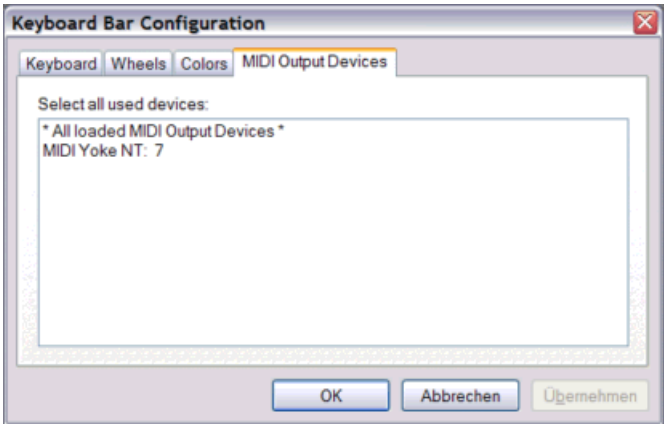


Figure 43: Keyboard Bar Configuration Dialog, MIDI Output Devices Tab

Normally, the Keyboard Bar is used to send MIDI messages to the loaded PlugIns. It can, however, be used to send MIDI messages to attached MIDI devices that are connected to one of the configured MIDI Output ports, too. Here, you can define these.

Status Bar

This menu entry can be used to toggle the Status Bar on the bottom of VSTHost's main window on or off.

Status Bar Level Meter

The status bar can be used to show a little level meter, if you don't use the **Master** window for that (see "Master" below). Since this can be a bit irritating, it can be turned on or off with this menu entry.

Window Menu

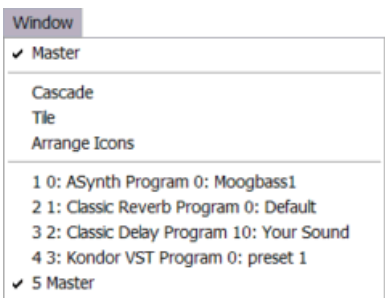


Figure 44: Window Menu

The contents of this menu vary with the number of loaded PlugIns and the various windows they can open. The first menu entry, however, is always there:

Master

Selecting this menu entry opens or activates the **Master** window:

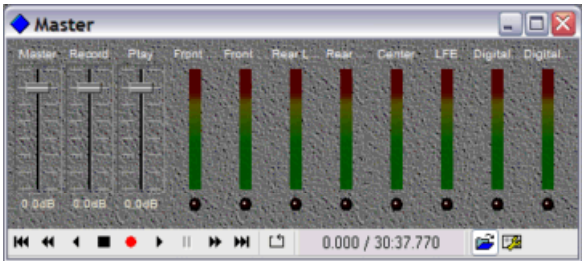


Figure 45: Master Window

The number of the level meters on the right varies with the Wave Output device; there is one meter for each audio channel. The faders on the left side are always there. On the bottom, the Master window has a copy of the Recorder Toolbar for convenient tape recorder operation; if you prefer the Master window, you can hide the Recorder Toolbar and vice versa.

The **Master** fader can be used to set the overall output level of VSTHost's audio engine.

The **Record** fader can be used to set the recording level of the Wave Recorder.

The **Play** fader can be used to set the Wave Player's level. This is independent of the setting of the Master fader.

Help Menu

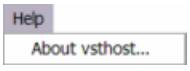
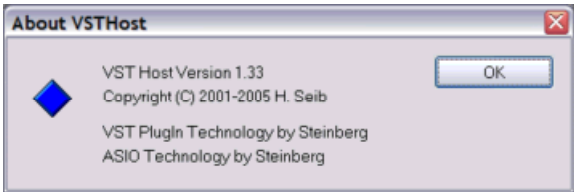


Figure 46: Help Menu in its entire glory

This menu is not extremely helpful at all; as this is a spare time project, I haven't found the time yet to create a help file, so it contains only one entry:

About VSTHost

Selecting this menu entry opens the most important dialog of the whole program, the thing you've all been waiting for:



… and with this extremely important information I'll end this document.

Have fun using VSTHost!

Hermann Seib
Vienna, 15 August 2005