CONTROL OF VST PLUG-INS USING OSC

Michael Zbyszyński *Adrian Freed*
Center for New Music and Audio Technologies (CNMAT)
Department of Music
University of California, Berkeley
1750 Arch Street
Berkeley, CA, USA
{mzed, adrian}@cnmat.berkeley.edu

ABSTRACT

The basic control structure of VST audio plug-ins can limit their usefulness. Control can be improved through the use of Open Sound Control by developing a flexible name space that employs multiple, intuitive parameter names (and aliases), higher-level controls and range mapping, simplifying control for the user. We will demonstrate these ideas with Max/MSP patches that repackage VST plug-ins in a more usable way and also introduce the idea that plug-in interfaces themselves can be improved by building in a well-formed OSC name space. Such a name space would enhance the longevity and flexibility of finished musical works. We will also show that when the plug-in is controlled directly with OSC atomicity and queries, control could be further improved.

1. INTRODUCTION

While audio plug-ins are extremely useful, limitations of their control structure can make them unwieldy to use. Specifically, the name space of each VST plug-in [1] is flat and populated by parameter names that have been carefully chosen by the designers of the plug-in, but do not necessarily represent the terminology or language preferred by the user. Parameter names are mapped through a generic range (0. to 1.) without informing the user about the mapping range or the specific units that are employed inside the plug-in, and each message controls only one parameter.

Through the use of Open Sound Control (OSC) [2], a flexible name space can be developed that employs multiple, intuitive parameter names (and aliases), higher-level controls and range mapping, simplifying control for the user. We will demonstrate these ideas with Max/MSP patches that repackage VST plug-ins in a more usable way and also introduce the idea that plug-in interfaces themselves can be improved by building in a well-formed OSC name space. We will also suggest ways (e.g., atomicity and queries) that control could be further improved if the plug-in could be controlled directly with OSC.

In addition to creating a more useable control structure, careful use of the OSC abstractions proposed here will allow composers and performers to create more fully documented works than can be easily updated with changing technologies. A thoughtfully designed name space can separate the musical intention from the particular plug-in, allowing composers to adapt and repurpose pieces as plug-ins evolve.

2. Problems with VST Control Structure

2.1. Names Dictated by Plug-In Designers

2.1.1. Non-intuitive names

Many audio processing plug-ins fall into typical categories, such as dynamics processors or reverbs. Each specific user has expectations for the names of the parameters in an archetypical reverb, for instance, which are determined by that user's technical and linguistic background. Where one user might expect a parameter called "Wet Level," another might be more familiar with "Reverb Gain." Users must adapt to the naming scheme of the plug-in designer. This complicates the use of multiple plug-ins from disparate sources; the user must constantly change naming schemes to do something as simple as auditioning multiple reverbs with similar parameter settings.

2.1.2. Need for aliases

The use of a rich OSC name space provides the opportunity for multiple aliases to the same parameter. An intelligent naming scheme would direct both "Wet Level" and "Reverb Gain" to the same parameter, allowing the user to focus on the specifics of controlling the sound. If the user changes reverb plug-ins, they could continue with their preferred naming scheme for the generic parameters. They could concentrate on the sonic differences between reverb algorithms, rather then the naming idiosyncrasies of each design.

2.1.3. Simplified reconstruction

Another advantage to a carefully designed name space would come when documenting, preserving, or reconstructing a finished musical work. Because the musician's original sonic intent would be represented in a form that was not tied to a specific plug-in technology, it could be easily understood and adapted to suit the situation of future technologies. When a piece travels or is revisited, explicitly descriptive parameter names, such as "Reverb Gain," are more easily interpreted than opaque names, such as "Mix" or Mode."

2.2. Parameters mapped to a generic range

The next step in an intelligent plug-in control scheme would be to address parameters in established units. All VST parameters are currently mapped to the range of 0.0 to 1.0, effectively obscuring the values being controlled. Furthermore, a particular user might prefer setting crossover frequencies in Hertz or MIDI cents, and might prefer percentages to decibels as a unit of loudness. In addition to parameter name aliases, parameters could be addressed in specific units. For example "Wet Level 63%" or "Reverb Gain –6dB." By accommodating specific user's predilections, plug-ins would become more intuitively useable.

2.3. Flat Name Space

2.3.1. Hierarchical name space

The naming space for VST plug-ins is flat, simply a list of parameters that could be addressed. While this might be adequate for plug-ins with a few parameters, some plug-ins (e.g. Waves Parametric Convolution Reverb [3]) can have dozens of parameters. In such a case, it makes sense to organize parameters in a hierarchical name space. For instance, a multiband compressor will have many crossover frequencies. These could be addressed as "/crossover/low<->mid *n*" and "/crossover/mid<->high *n*" (or, depending on the user, "/crossover/middle<->hi *n*" or "/x-over/m<->h *n*"). Such a naming scheme would help keep the control structure clear and comprehensible.

The ability to reorganize parameters in a clear hierarchy would have special significance in to users working with assistive technologies. For example, blind users working with screen readers could customize their name space to quickly navigate to the most pertinent parameters, rather than waiting to read through a long list of all possibilities.

2.3.2. Pattern matching

A carefully designed hierarchical name space would allow the user leverage OSC's use of pattern matching. [4] Following the example of a multiband compressor, the compression thresholds might be named "/low/threshold *n*," /middle/threshold *n*," and "/high/threshold *n*." In the OSC Address Pattern, "*" matches any sequence of zero or more characters. This would allow the user to simultaneously adjust all thresholds by sending the message "/*/threshold *n*."

2.3.3. One message, many parameters

Another advantage of a careful created OSC name space would the use higher-level names that control multiple parameters. In the example the context of multi-band compression, the user might be given explicit control of the cut-off frequencies between the low, middle, and high bands. Alternatively, the user could prefer to control the center frequency and bandwidth of the middle band. With OSC, these high-level parameter names could be created to adjust the appropriate crossover frequencies accordingly.

3. OSC SOLUTION IN MAX/MSP

3.1. Two reverb example

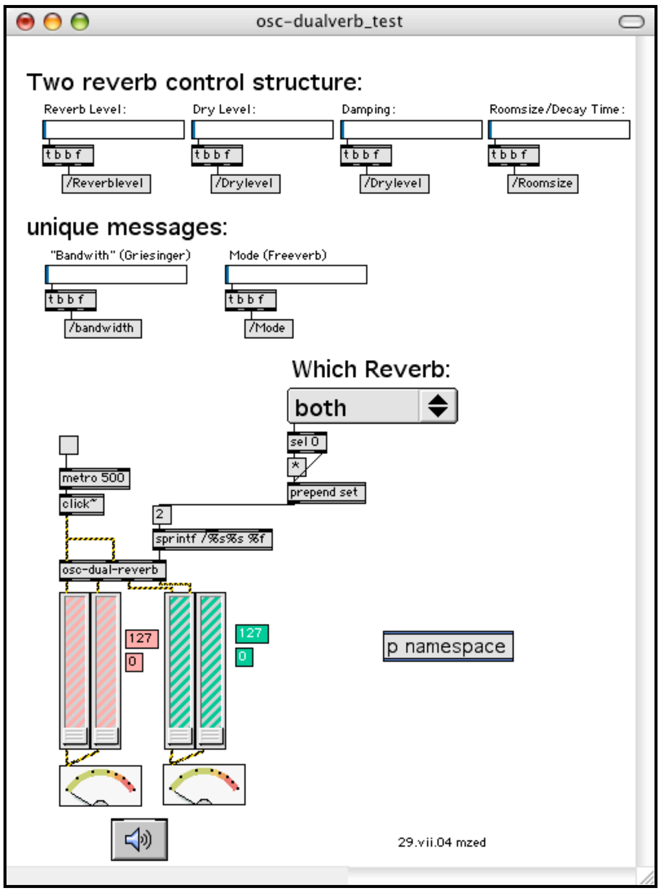

Figure 1. Top level of two reverb patch

The following examples were programmed in Max/MSP following the name space design guidelines suggest above. The first demonstrates a control structure for controlling two reverb plug-ins (mda Freeverb [5] and Griesinger 2.2 by Nathan Wolek [6]), each with different parameter names.

This name space was designed to enable the user to switch between the reverb plug-ins while maintaining a consistent control structure. Similar parameters in each plug-in are addressed with the same name. For instance, "Reverblevel" addresses "Wetlevel" in the Freeverb and "reverbgain" in the Griesinger reverb. The example also suggests the possibility of aliases and unconventional terminology; "slimeyness" is also mapped to "Wetlevel" and "reverbgain."

Freeverb:	Griesinger:
/[Freeverb, 1]/Mode [0.-1.]	n/a
n/a	/[Griesinger, 2]/bandwidth [0.-1.]
/[Freeverb, 1]/Roomsize [0.-1.]	/[Griesinger, 2]/Roomsize [0.-1.]
/[Freeverb, 1]/Width [0.-1.]	/[Griesinger, 2]/Damping [0.-1.]
n/a	/[Griesinger, 2]/pre-delay [0.-1.]
/[Freeverb, 1]/Reverblevel [0.-1.]	/[Griesinger, 2]/Reverblevel [0.-1.]
/[Freeverb, 1]/Drylevel [0.-1.]	/[Griesinger, 2]/Drylevel [0.-1.]
/[Freeverb, 1]/slimeyness [0.-1.]	/[Griesinger, 2]/slimeyness [0.-1.]

Table 2. Name space for two reverb patch

3.2. Multiband compressor example

The namespace for controlling a multiband compressor (mda MultiBand [7]) is much more complicated, especially in its use of aliases. It also includes higer level parameters that influence more than one of the plug-in's built in parameters.

Specific bands:
/[Low, low, L, l]/[Comp, Compression, comp, compression] [0.-1.]
/[Low, low, L, l]/[dbcomp, dBcomp] [0.-30.]
/[Low, low, L, l]/[Out, Output, out, output] [0.-1.]
/[Low, low, L, l]/[dboutput, dBoutput] [-20.- 20.]
/[Mid, mid, M, m/[Comp, Compression, comp, compression] [0.-1.]
/[Mid, mid, M, m]/[Out, Output, out, output] [0.-1.]
/[Mid, mid, M, m/[dbcomp, dBcomp] [0.-30.]
/[Mid, mid, M, m]/[dboutput, dBoutput] [-20.- 20.]
/[Hi, hi, H, high]/[Comp, Compression, comp, compression] [0.-1.]
/[Hi, hi, H, high]/[Out, Output, out, output] [0.-1.]
/[Hi, hi, H, high]/[dbcomp, dBcomp] [0.-30.]
/[Hi, hi, H, high]/[dboutput, dBoutput] [-20.- 20.]
Higher-level parameters:
/Mid-CF [value in HZ]
/Mid-BW [value in HZ]
General Parameters:
/L<>M [0.-1.]
/M<>H [0.-1.]
/LM-Hz [87.-1020.]
/MH-HZ [111.-19606.]
/Attack [0.-1.]
/Release [0.-1.]
/Stereo [0.-1.]
/Process [0.-1.]

Table 2. Name space for multiband compressor patch

The first group of names address single plug-in parameters, and demonstrate multiple naming possibilities for the same parameter. Different users might prefer "Hi" to "high," or simply "H." This space could be expanded to include other aliases for language localization. Additionally, the parameters "/*/[dbcomp, dBcomp]" and "/*/[dboutput, dBoutput]" employ real units, decibels instead of the generic range of 0. to 1. They are mapped accordingly in the max patch before being sent to the plug-in. Other units of loudness or intensity could be added, each with their own mapping.

The second pair of names are higher-level parameters. They allow the user to directly control the center frequency and bandwidth of the middle compression band, implicitly influence all three bands. The OSC parameter value is translated into the proper crossover frequencies for both the low to middle and the middle to high crossover, and forwarded to the plug-ins built in parameters.

The final group of generalized parameters includes another example of real units, setting the crossover frequencies in Hertz.

The patch, below, also uses OSC's pattern matching abilities. To adjust all of the output gains simultaneously and in decibles, the user sends the message: "/?/dboutput *n*." Variants of this message using other aliases are also available.

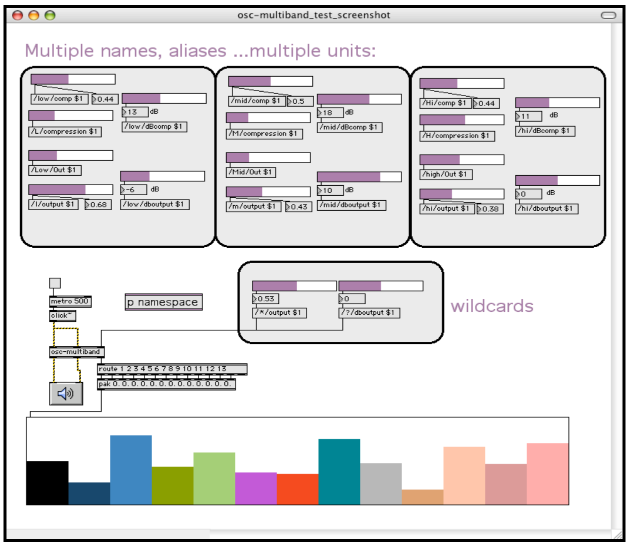

Figure 2. Top level of multiband compression patch

4. SUGGESTIONS FOR FURTHER IMPROVEMENT

All of the Max/MSP examples in this paper were hand built around specific plug-ins. [8] It is possible to query parameter names, but each plug-in had to be loaded and observed in action to determine the exact units and range of each parameter. Even the, the curve to which the parameters are mapped is not always obvious. OSC supports a full querying system. If OSC support was integrated into VST plug-ins, much of this namespace could be built automatically, or built in advance by the plug-in designer. Plug-in vendors have met this suggestion with enthusiastic support. In the Open Sound World environment, some of this handwork would be simplified because all objects inherit a hierarchical OSC name space, including the VST plug-in loader. Products by Native Instruments [9], Reaktor and Intakt, already support OSC in their standalone applications, but it is not possible to address their plug-ins directly using OSC.

Another important feature of OSC is atomicity. Messages that should be executed simultaneously can be sent together as an indivisible bundle; they will be either all executed in one scheduler tick, or none will be. While this behavior is desirable in many circumstances, it could be critical in cases such as updating filter coefficients. Coefficients received serially could lead to an unstable filter state. Atomicity is not implemented in the current VST specification. It is possible to atomically set plug-in parameters in the OSW wrapper by connecting them up to synchronous source, such as an OSC bundle.

This paper has dealt exclusively with control of VST audio plug-ins, but OSC would be similarly helpful in managing control of VST Instruments. Also, the control structure recommendations are applicable to other formats of plug-ins, such as AU, TDM, or RTAS.

5. CONCLUSIONS

Currently, the control structure for VST plug-ins is disadvantaged by a limited, flat name space. When using multiple plug-ins, users must remember idiomatic names for generic parameters, and navigate through values that have all been mapped to the same range. The usefulness of plug-ins in general could greatly improved by implementing a thoughtfully designed control structure in OSC that includes:

- an hierarchical name space designed to take advantage of pattern matching
- a set of naming conventions that exhibits clarity and portability
- flexibility in naming to accommodate users with diverse abilities, backgrounds, and intentions
- values addressable in real units
- high-level parameters that allow the user to control multiple values or multiple plug-ins with single messages
- atomicity in message handling

Such a control structure would improve control during the creation of musical works, as well as simplify the preservation and reconstruction of works in the future.

6. ACKNOWLEDGEMENTS

Special thanks to the UC Discovery Grant Program for its generous support.

7. REFERENCES

[1] Steinberg Media Technologies “VST Plug-ins SDK 2.3.”http://www.steinberg.net/steinberg/ygrabit/vstsk/OnlineDoc/vstsdk2.3/index.html, 4 March 2005.

[2] Wright, Matthew, and Adrian Freed. "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers." Paper presented at the International Computer Music Conference, Thessaloniki, Hellas 1997.

[3] Waves Ltd. Digital Audio Processing, IR1 Parametric Convolution Reverb, http://www.waves.com/content.asp?id=1564, 4 March 2005.

[4] Torkington, Nathan. *Regular Expression Pocket Reference*. Sebastopol, Calif. ; Farnham: O'Reilly, 2003.

[5] Maxim Digital Audio "Freeverb" http://www.mda-vst.com/, 5 March 2005.

[6] Wolek, Nathan “Griesinger 2.2” http://www.nathanwolek.com/, 5 March 2005.

[7] Maxim Digital Audio "mda MultiBand" http://www. mda-vst.com/, 5 March 2005.

[8] Zbyszynski, Michael "OSC Control of VST Plug-ins." Poster presented at the Open Sound Control Conference, Berkeley, CA, USA, 2004.

[9] Native Instruments GmbH http://www.nativinstru-ments.de/, 10 March 2005.