Subject: Paris-Senderella Beta 1.0 RELEASED!
Posted by drfrankencopter on Sun, 02 May 2010 00:33:19 GMT
View Forum Message <> Reply to Message

Not sure how hotly anticipated this has been, but I've been working away on it for the past couple of months and feel is ready for Beta release. It doesn't have all the features I'm looking for yet, but I think it's a significant enough improvement over the original Senderella that its worthy of Beta release now.

Here's the web version of the manual:
http://web.ncf.ca/fk824/paris/ParisSenderella.htm

And here's the download zip with the .dll itself.
http://web.ncf.ca/fk824/paris/Paris-Senderella.zip

Here's just a little blurb from the 'manual' to clear up what the changes are between Paris-Senderella and ModuLR's Senderella

What's the difference between Paris-Senderella beta 1.0 vand ModuLR Senderella 1.08?

Senderella is a routing/bussing plugin that allows the user to instantiate it as a SEND, or a RECEIVE. Sends are summed together, and individual send volume is controlled via a parameter.
The Paris-Senderella has the following visible differences:
1) 16 Channels as opposed to 64
2) Send & Receive volume controlled in dB as opposed to %
3) Pan control that approximates the PARIS pan law
4) Delay compensation for parallel processing (drum buss squishing anyone?)

In addition, Paris-Senderella has the following 'invisible' differences:
5) VST 2.3 spec compliant
6) Tested with the FXPansion wrapper
7) Reduced memory footprint
Future development path by a PARIS user
9) Available source code

These differences are further described below:

1. 16 Channels as opposed to 64
At the outset less channels available seems like a bad idea, right? Well, unfortunately the only way to get this plugin to work properly in PARIS is to wrap it to Direct-X, which for most means using the FXPansion wrapper and its default GUI. This doesn't limit the number of channels, but it does make the plugin GUI somewhat cramped to work with, and I found that 64 channels made it a little 'touchy' to get the exact channel #'s right. Besides, with the pan parameter implementation added to Paris-Senderella you effectively have 16 stereo channels, which would have required 32 channels under the ModuLR version of Senderella. If users find this to be a limitation, it's easy enough to change the number of channels in the code.

2. Send and Receive volume controlled in dB instead of %

ModuLR's Senderella presented volume control to the user via percent %, which worked, but was not practical since we hear sounds in a logarithmic (decibel) sense. Paris-Senderella allows the volume to be controlled in dB's (from -30 to +10dB). This 40dB range was chosen to align with PARIS 40dB Aux gain range (though PARIS goes -20dB to +20dB). I've also allowed for a Inf setting which turns off the send, but leaves the signal active for delay compensation (more on that later)

3. Pan Control that mirrors the PARIS pan law

In PARIS, in order to implement a stereo send/return with ModuLR Senderella it was necessary to use 2 instances per track (a Left channel send, and a right channel send), which made it a real pain to make quick pan adjustments. Paris-Senderella features a pan control that matches the PARIS pan control quite well. If you want the sends to match your PARIS mixer pan positions, just slide the Paris-Senderella controls to the same value (also scaled -100 to +100 just like PARIS).

4. Delay compensation for parallel processing

Senderlla opens up possibilities for doing parallel processing, such as 'New York Style' drum compression. The only problem is that many plugins also introduce some latency which must be compensated for in order for this trick to work. Vertex Faderworks can help out here, but I've coded up some functionality into Paris-Senderella that handles this latency compensation for the simpler parallel processing tasks (within a single submix only, at this time).

5. VST 2.3 Spec Compliant

Not that this really does anything for PARIS considering it predates VST 2.3...

6. Tested with FXPansion DX-VST Wrapper

The PARIS VST implementation is somewhat wonky (that's an understatement), and as such in order to get Paris-Senderella, or ModuLR Senderella to work it needs to be wrapped to DX (or possibly other hosts like XLUtop Chainer might work....let me know). Most PARIS users are using FXPansion 3.X for wrapping their plugins, and this is what I've tested with.

The FXPansion wrapper introduces some limitations in the plugin interface, and some specific code changes had to be made to support parameter saving under this wrapper. These are not too cumbersome, and will be explained later.

7. Reduced Memory Footprint

Not that this is a big deal with today's system, but who couldn't use a bit more free memory? ModuLR Senderella assigned a big buffer of each possible instance of the plugin (64 instances * 64 channels * stereo * 32 bit = 128 MB!). Paris-Senderella only uses big buffers for the receive instances (16), and uses very small buffers for the sends (approx 1MB of memory use). Actual memory use in the Beta version may be higher than this since it's a 'debug' release and has extra symbols loaded into memory. Once user reports come in I will release a 'release' version without the debug symbols.

Software geeks will like to know that the difference here is that ModuLR Senderella stored each send in a separate big buffer, and then summed these individually in the receive instances. Paris-Senderella uses a separate buffer for the 'receives,' and each 'send' writes to them via accumulating (+=) operations. This also has a side benefit in that its less processor intensive for multiple receives on a given channel.

8. Future development path by a PARIS user
Obvious advantages here for the PARIS community.

9. Source code available
The source for ModuLR's Senderella version 1.03 was made available on KVR's forum, and that's what I started from. The source for Paris-Senderlla is available by request. PM me on the Paris forum if you're interested in developing further additions.

Enjoy, and please provide feedback in this thread!

Cheers,

Kris
