Holy City Audio Forum

Former Home of SpinCAD Designer
Skip to content

Search... Search
Advanced search
[Moderator Control Panel]

Semitones and cents

Post a reply

Search this topic... Search

2 posts • Page **1** of **1**

- Edit post (./posting.php?mode=edit&f=42&p=2265)
- Delete post (./posting.php?mode=delete&f=42&p=2265)
- Report this post (./report.php?f=42&p=2265)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2265&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2265)

Semitones and cents (#p2265)

by **Digital Larry** » Thu Apr 09, 2015 6:30 am

During my recent deep dive into the pitch shift simulation, I learned some important things. Both about debugging strategy and pitch shifting itself!

Now of course I started by reading the Spin application note AN-001. Boy have I read that note. Over and over and over. To get to an actual pitch shifted signal you have to do a number of things. Basic pitch shifter takes 7 or 8 instructions. Trying to debug this by looking at the end result was a ridiculous waste of time! And yet, until a couple of weeks ago, that was more or less what I was doing. I realized I was really going to need to break it down to every individual step.

First you pitch shift using whichever Ramp LFO you want, starting at 0 of the buffer you're using. Since the Ramp LFO adds to the current position, depending on whether you are shifting up or down, the Ramp readout position is constantly wrapping around past the read position, going one way or the other. When you wrap around from 0 to 4096, that's when you get that nasty glitch. More on that later.

Then you do the inter-sample interpolation, just like you do when you do a sine wave LFO chorus. Well let me back up a bit here, because there's one thing I learned right at the end of the exercise, which I wished I'd been more aware of early on.

Since the coefficient of 16384 gives you an octave of pitch shift up, that means that when you use this coefficient, you're jumping ahead by a full sample each sample period. This implies that the minimum increment of the "high resolution LFO" is 1, or 1/16384 of a sample. So, I just made the high-res LFO using this assumption. That worked up until the point that I tried using:

```
RDAX POT0, 0.5
WRAX RMP0_RATE, 0
```

I read from the AN-001 that 0.5 in the RMP0_RATE register corresponds to an octave up, or 16384 in the WLDR instruction, e.g.:

```
WLDR 0, 16384, 4096.
```

Now I don't know about you, but even after two years of intense study on the FV-1 I have not internalized the relationship between floating point values and fixed point values. Most of the time you don't need to bother. But here's a case where you really do. As it turns out, which I discovered mostly by trial and lots of error, 0.5 corresponds to 16384 * 256. So, the resolution of the high res LFO has to be 16384 * 256. Keep in mind that the increments of this are still going to be 256 steps apart. There is no interpolation down to that level. I had been using 16384 as the high res LFO resolution up until the very end and it was working fine. So this factor of 256 is simply to scale it into the proper zone so that 0.5 loaded into RMP0_RATE gives you an octave up.

Moving right along, the sin LFO looks like it uses 8 fractional positioning bits which give you your inter-sample interpolation. However the Ramp LFO is obviously giving 14 bits (16384) and so I used 16384 steps for inter sample interpolation on Ramp instead of 256 as described for Sin LFO. I don't know what the FV-1 REALLY does here.

Maybe later I will post what the two halves of that inter-sample interpolation thing look like before you add them together. They look ridiculous! But then you add them together and they look beautiful.

So, you do that for the RMP and RPTR2 modes. Now you have two pitch shifted signals, offset by half the LFO buffer length from each other. So each one of them has a nice wraparound glitch at equally spaced points, but the nice thing now is that each one's glitch is in the middle of the other one's "good part". Now you gotta get on to the crossfade LFO generation.

I decided to put the glitch in the MIDDLE of the crossfade's flat part, though I read another article on pitch shifting where it showed putting the glitch at the edge. Putting it in the middle requires slicing the time into 8 slots, while at the edge, only 4 are needed. I implemented it with 8 and I don't know how the FV-1 really does it. I don't think it should matter, as long as you are really at zero when the glitch occurs. Maybe some other time when I have nothing important to do I will try to simplify it (ha ha ha).

I don't even want to get into how I dealt with the crossfades and compensating for different LFO buffer lengths and frequencies. But that was all necessary.

Last time I got into debugging this, I arranged to print out all sorts of numbers which I put into a spreadsheet and then made graphs of what I

imagined the internal waveforms were doing. This took a LONG time and didn't give me much insight into what was really going on. This time around, I took temporary audio outputs from each stage of the process, ran the simulator in "write to file" mode with a 500 Hz sine wave as the simulator input. Then I'd quickly read the resulting WAV file into GoldWave to see what was going on with the AUDIO signals. This was way more useful, because honestly I was guessing at a large number of points about how many left shifts to do, etc. and it was more obvious this way.

Top
- Edit post (./posting.php?mode=edit&f=42&p=2266)
- Delete post (./posting.php?mode=delete&f=42&p=2266)
- Report this post (./report.php?f=42&p=2266)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2266&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2266)

Re: Semitones and cents (#p2266)

by **Digital Larry** » Thu Apr 09, 2015 6:45 am

Yes, you're correct. I didn't talk about semitones and cents the first time around. So let me do so now.

To calculate the pitch shift coefficient for a given number of semitones requires you to use the 12th root of two. 16384 represents an octave, but if you divide that number by 12 and use that for equally spaced semitones, you will be playing gamelan music before you know it!

So now that you have semitones, how do you get down to cents? A cent is the division of a semitone into 100 equal "pitch steps". This would involve the 100th root of whatever the distance between adjacent semitones was. The coefficient for 1 semitone shift is about 900. The 100th root of 900 is 1.07 something. So it's fairly clear that with the LFO resolution we have, we're not going to be able to get down to cent resolution on pitch shifts.

But let's be realistic here. I've been spending more time pitch shifting recently than is normally considered healthy. And I've come down to what I think the useful pitch shifts with the FV-1 are.

#1 7 semitones up or down
#2 12 semitones up or down
#3 very slight pitch shifting mixed in with the dry signal for a detuning effect (maybe up to a maximum of 300 on the pitch coefficient).

So, I'm sure I'll revise the "Pitch Shift Adjustable" block because nobody really wants to deal with "what does 12724 mean?" What I think I will eventually do is to give a readout, which is semitones and cents, based on the current value of the coefficient.

Top

Display posts from previous: All posts Sort by Post time Ascending Go

Post a reply

2 posts • Page **1** of **1**

Return to Developer's Corner

Jump to: Developer's Corner Go

Quick-mod tools: Lock topic Go

Who is online

Users browsing this forum: Digital Larry and 0 guests

Powered by phpBB® Forum Software © phpBB Group

Administration Control Panel