[mini-project] Vibraphone virtual musical instrument (VMI) in LabVIEW*

Ed Doering

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 2.0†

Abstract

The vibraphone percussion instrument can be well-modeled by a sinusoidal oscillator, an attack-decay envelope with a short attack and a long decay, and a low-frequency sinusoidal amplitude modulation. In this mini-project, develop code to model the vibraphone as a LabVIEW "virtual musical instrument" (VMI) that can be "played" by a MIDI music file.

	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide[1] module for tutorials and documentation that will help you:
	• Apply LabVIEW to Audio Signal Processing
	• Get started with LabVIEW
	• Obtain a fully-functional evaluation edition of LabVIEW

Table 1

1 Objective

The **vibraphone** percussion instrument can be well-modeled by a sinusoidal oscillator, an attack-decay envelope with a short attack and a long decay, and a low-frequency sinusoidal amplitude modulation. In this mini-project you will develop code to model the vibraphone as a LabVIEW **virtual musical instrument** (**VMI**) to be "played" by a MIDI file within **MIDI JamSession**.

2 Prerequisite Modules

If you have not done so already, please study the pre-requisite module Tremolo Effect[2]. If you are relatively new to LabVIEW, consider taking the course LabVIEW Techniques for Audio Signal Processing[3] which

*Version 1.2: Mar 17, 2008 9:28 pm -0500

†http://creativecommons.org/licenses/by/2.0/

[1] "NI LabVIEW Getting Started FAQ" <http://cnx.org/content/m15428/latest/>

[2] "Tremolo Effect" <http://cnx.org/content/m15497/latest/>

[3] *Musical Signal Processing with LabVIEW – Programming Techniques for Audio Signal Processing* <http://cnx.org/content/col10440/latest/>

provides the foundation you need to complete this mini-project activity, including working with arrays, creating subVIs, playing an array to the soundcard, and saving an array as a .wav sound file.

3 Deliverables

- All LabVIEW code that you develop (block diagrams and front panels)
- All generated sounds in .wav format
- Any plots or diagrams requested
- Summary write-up of your results

4 Part 1: Tremolo Envelope Generator

Create LabVIEW code to create a time-varying intensity envelope for the tremolo effect. Your code will require tremolo rate (in Hz), depth (in dB), and total number of samples and will produce a tremolo envelope with a sinusoidal shape as follows:

This is an unsupported media type. To view, please see
http://cnx.org/content/m15498/latest/behavior-tremolo-envelope.llb

The maximum intensity will be fixed at 0 dB, and the sinusoid's amplitude will be "depth." Once you develop your code, compare its behavior with that of the interactive front panel above. Note that the time range of the interactive front panel is fixed at 1 second, but your code should produce any number of required samples.

5 Part 2: Attack/Decay Envelope Generator

Create LabVIEW code to create a time-varying intensity envelope for the overall attack and decay of the note. Your code will require attack time and decay time (both in seconds), and will produce an envelope composed of two straight-line segments as follows:

This is an unsupported media type. To view, please see
http://cnx.org/content/m15498/latest/behavior-AD-envelope.llb

The maximum intensity will be fixed at 0 dB, and the minimum intensity will be -40 dB. Once you develop your code, compare its behavior with that of the interactive front panel above.

6 Part 3: Attenuator

Create LabVIEW code that will accept an "amplitude" parameter in the range 0 to 1, and will convert this parameter to an attenuation in the range -40 dB to 0 dB. The amplitude parameter will ultimately be supplied by MIDI_JamSession and represents the MIDI "note-on" velocity. Your code will map linear velocity onto a logarithmic intensity.

7 Part 4: Overall Amplitude Envelope

Combine the code fragments you developed in Parts 1 to 3 to create an overall intensity envelope. Remember that when you work with intensity values in decibels, you simply need to **add** them together. Next, "undo" the equation for decibels to convert the intensity envelope into an amplitude envelope (hint: you need a value of "20" someplace). Choose a representative set of parameter values and plot your overall intensity envelope and your overall amplitude envelope.

8 Part 5: Vibraphone VMI

In this part you will design a vibraphone **virtual musical instrument** (**VMI** for short) that can be played by "MIDI JamSession." If necessary, visit MIDI JamSession[4], download the application VI .zip file, and view the screencast video in that module to learn more about the application and how to create your own virtual musical instrument. Your VMI will accept parameters that specify frequency and amplitude of a single note, and will produce an array of audio samples corresponding to a single strike on the metal bar of a vibraphone instrument. Use a sinusoidal signal as the oscillator (tone generator), and apply the amplitude envelope you generated in Part 4. You may wish to keep your parameters as front-panel controls and add the "Play Waveform" ExpressVI to listen to your VMI during development. Adjust the parameters to obtain pleasing and realistic settings, then convert the front-panel controls to constants and remove "Play Waveform." Your finished VMI must not contain any front panel controls or indicators beyond what is provided in the prototype instrument.

> NOTE: The prototype VMI includes the "length" parameter to set the number of samples to be produced by your own design. The length is derived from the amount of elapsed time between "note on" and "note off" MIDI messages for a given note. To make a more realistic sound for the vibraphone, ignore this length value and create a fixed number of samples determined by your attack and decay times.

Finally, choose a suitable MIDI file and use MIDI JamSession to play your vibraphone VMI. MIDI files that contain a solo instrument, slower tempo, and distinct notes will likely produce better results. Create a .wav file of your finished work. The Figure 1 screencast video provides some coding tips.

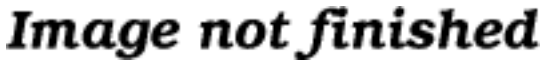

Figure 1: [video] Coding tips for Part 5

9 Optional: Modifications to Basic Vibraphone VMI

Following are some suggested modifications you could try for your basic vibraphone VMI:

- Make the decay time vary according to the "length" parameter provided by the prototype VMI. While a variable decay time may not necessarily be physically realistic, it may sound interesting.
- Use a fixed decay time, but use the "length" parameter to determine when to cut off (or damp) the tone. You will need to include a short envelope segment to taper the amplitude back to zero, because an abrupt cutoff will cause click noise.
- Make the tremolo depth (or rate, or both) vary according to the "amplitude" parameter provided by the prototype VMI. For example, a higher amplitude could be mapped to a faster rate or more depth.
- Remove the tremolo envelope from the vibraphone VMI, and use it as a single envelope for the entire piece (you would need to read the .wav file produced by MIDI JamSession and apply the tremolo envelope). On the real vibraphone, the rotating disks turn at the same rate for all of the resonators, so placing the tremolo on each individual note is not the best way to model the physical instrument.

[4] "[LabVIEW application] MIDI_JamSession" <http://cnx.org/content/m15053/latest/>