MEGA

Project number: IST-1999-20410

GEM EyesWeb Audio Tools

software contributions to the MEGA SE

SoundSynthesis Library, v. 2.0

http://www.generalmusic.com

Contact:
Amalia de Götzen
Carlo Drioli

corvo@dei.unipd.it
drioli@dei.unipd.it

SoundSynthesis Library

The SoundSynthesis library is intended to add audio synthesis features to the EyesWeb platform. The library is contained in the EywSoundSynthesis.dll file. Some of the features provided are control signal generators (ADSR), general purpose audio filters, and other utilities. The blocks provided by the SoundSynthesis library are contained in the Sound.Synthesis folder in the EyesWeb block browser.

1. Control signal generators

The control of sound generators and processors requires a set of control-rate signal generators which allow the modulation of slowly-varying parameters, e.g. the pitch and the amplitude of a sinusoid. Examples of the most common control curves are pitch and amplitude ADSR envelopes, sinusoidal modulation for vibrato and tremolo effects, etc. The SoundSynthesis library contains blocks for the generation of ADSR envelopes from MIDI input (**MIDI_Adsr**) and a conversion block to convert from audio-signal to control-signal. This block allows one to use audio-rate oscillators (sinusoidal, square wave, random noise, etc.) to generate control-rate signals.

1.1. MIDI_Adsr (monophonic MIDI-to-Adsr generator)

Let say that a MIDI musical keyboard is to be used to generate a frame-rate curve to control some audio parameter. This is the case, for example, of the pitch or amplitude envelopes to drive a sinusoidal oscillator. Let us describe what the **MIDI_Adsr** block does: as MIDI messages comes in, they are parsed in order to detect Note_ON/Note_Off messages and checked for time and note coherence (a Note_Off should come after a Note_On, his key number should correspond to an already switched-on note, and so on). In the monophonic version, only one active note at the time is allowed, and some decision has to be taken if a new Note_On arrives while another note is still active: either the new note can be discarded, or the old one can be changed in the new one by possibly generate a legato between the two notes (the second solution has been implemented in the block). As a Note_On message is detected, a classic Attack-Decay-Sustain-Release (ADSR) envelope is generated (see Figure 1).

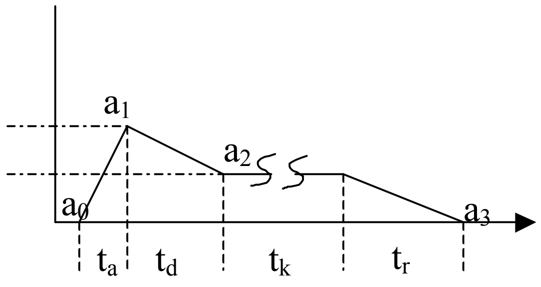

Figure 1: the ADSR envelope.

Since pitch envelopes and amplitude envelopes have different range and can be different in shape, a combo box parameter let the user select what type of control needed. The value of a1 is automatically computed from the key-number (byte 2 of the MIDI message) in the case of a frequency envelope, or from the key-velocity (byte 3 of the MIDI message) in case of an amplitude envelope. The tables below resume the policy for the computation of the a_i and t_i parameters in the two cases (a_j ([0 1]): relative to a_k means that the actual value of the parameter will be $a_j \cdot a_k$).

MIDI_Adsr (Freq. Option):

a0 ([0.5 1.5]): relative to a1
a1 ([0.5 1.5]): relative to MIDI note pitch
a2 ([0.5 1.5]): relative to a1
a3 ([0.5 1.5]): relative to a1

ta ([0 Inf]): msec
td ([0 Inf]): msec
tk : determined by how long the key is held down
tr ([0 Inf]): msec

MIDI_Adsr (Ampl. Option):

a0 ([0 1]): relative to a1
a1 ([0 1]): relative to MIDI key velocity
a2 ([0 1]): relative to a1
a3 ([0 1]): relative to a1

ta ([0 Inf]): msec
td ([0 Inf]): msec
tk : determined by how long the key is held down
tr ([0 Inf]): msec

1.2. MIDI_PolyAdsr (polyphonic MIDI-to-Adsr generator)

The block **MIDI_PolyAdsr** is the polyphonic version of the MIDI-to-ADSR generator. A note queuing algorithm integrated in the block provides the policies to decide whether a new Note_On event should enter the queue and to which voice will be allocated, which voice has to bee freed to let a new Note_On event enter the queue. The algorithm takes also care of the retrieval and elimination of a note from the queue when a Note_Off message is received. The user can specify the maximum length of the queue (the number of available voices), say *N*. The output of the block will then be an *Nx1* matrix in which the *i*th row contains the *i*th ADSR generator instantaneous value (the ADSR envelopes are generated as described in the monophonic case). If *N* scalar envelopes are needed, the **GetEntry** block in the Math.Matrix folder can be used.

Remarks:

The blocks **MIDI_Adsr** and **MIDI_PolyAdsr** should be preferably given a high priority. A typical problem that can arise when then priority is too low is that an incoming MIDI Note_Off message can be lost and a note will continue to play until the key is pressed again.

1.3. MIDI_PolyAdsr_FA (polyphonic MIDI-to-Adsr generator)

The block MIDI_PolyAdsr is the polyphonic version of the MIDI-to-ADSR generator, as described in the MIDI_PolyAdsr, but in this block there are simultaneously both type of control: pitch envelopes and amplitude envelopes are the outputs.

2. General purpose audio filters

The SoundSynthesis library provides some filtering tools useful for audio filtering and for control signal generation. **IIR_SOcell** is a second-order cell IIR filter implementing a band-pass filtering. The band center frequency and width can be controlled in real time. This block can be used both for the processing of audio and for the generation of control signals. In this case, however, a conversion from audio buffer (at sample rate) to scalar value (at control-rate) is necessary. This conversion is provided by the block **Audio2double.** The patch in Figure 2 is an example of how the **Audio2double** conversion block can be used to generate vibrato and microvariations control-rate curves using audio generators and filters.

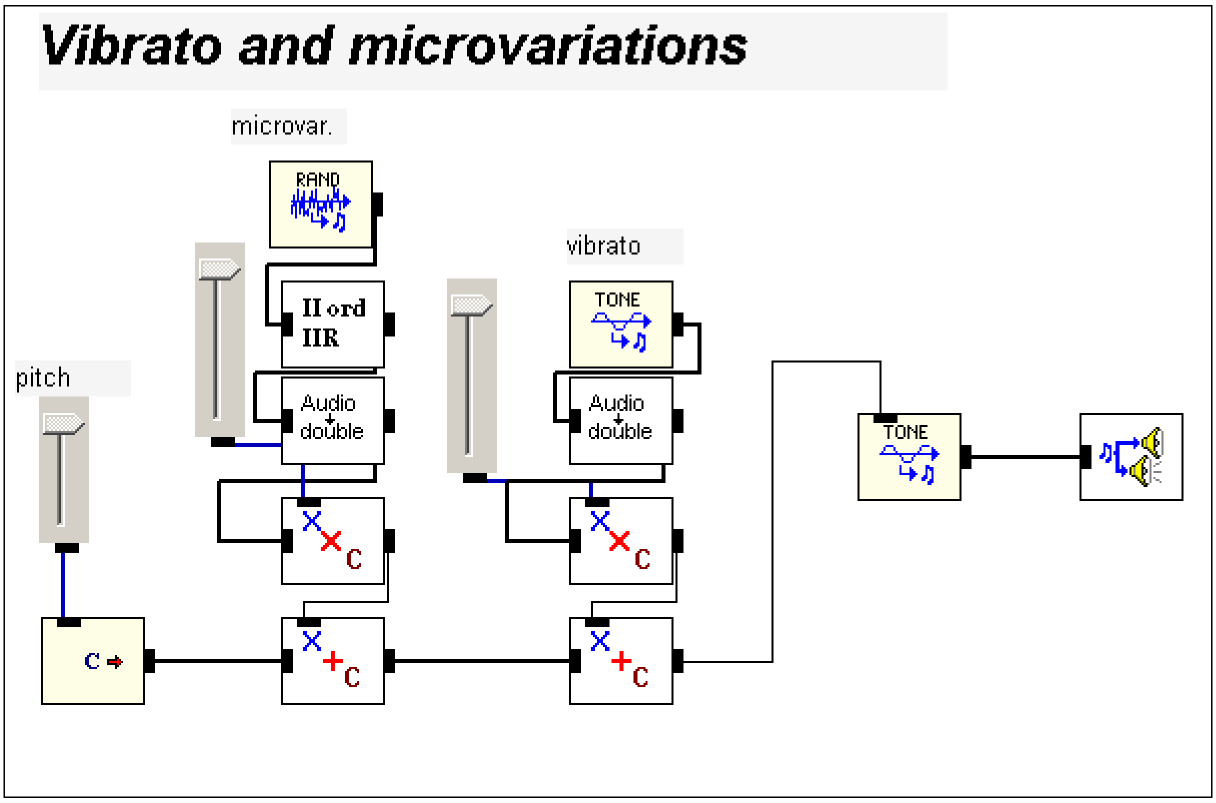

Figure 2: generation of control signals for frequency microvariations and vibrato

3. DevGen (Vibrato and microvariation generator)

The SoundSynthesis library provides also a block that performes microvariations and vibrato using the same concept as in the above patche. A combo box parameter let the user select what type of control needed. The vibration control have two kinds of parameters: VibRate and VibDepth. The patch in Figure 3 is an example of how the **DevGen** block can be used to generate vibrato and microvariations control-rate curves.

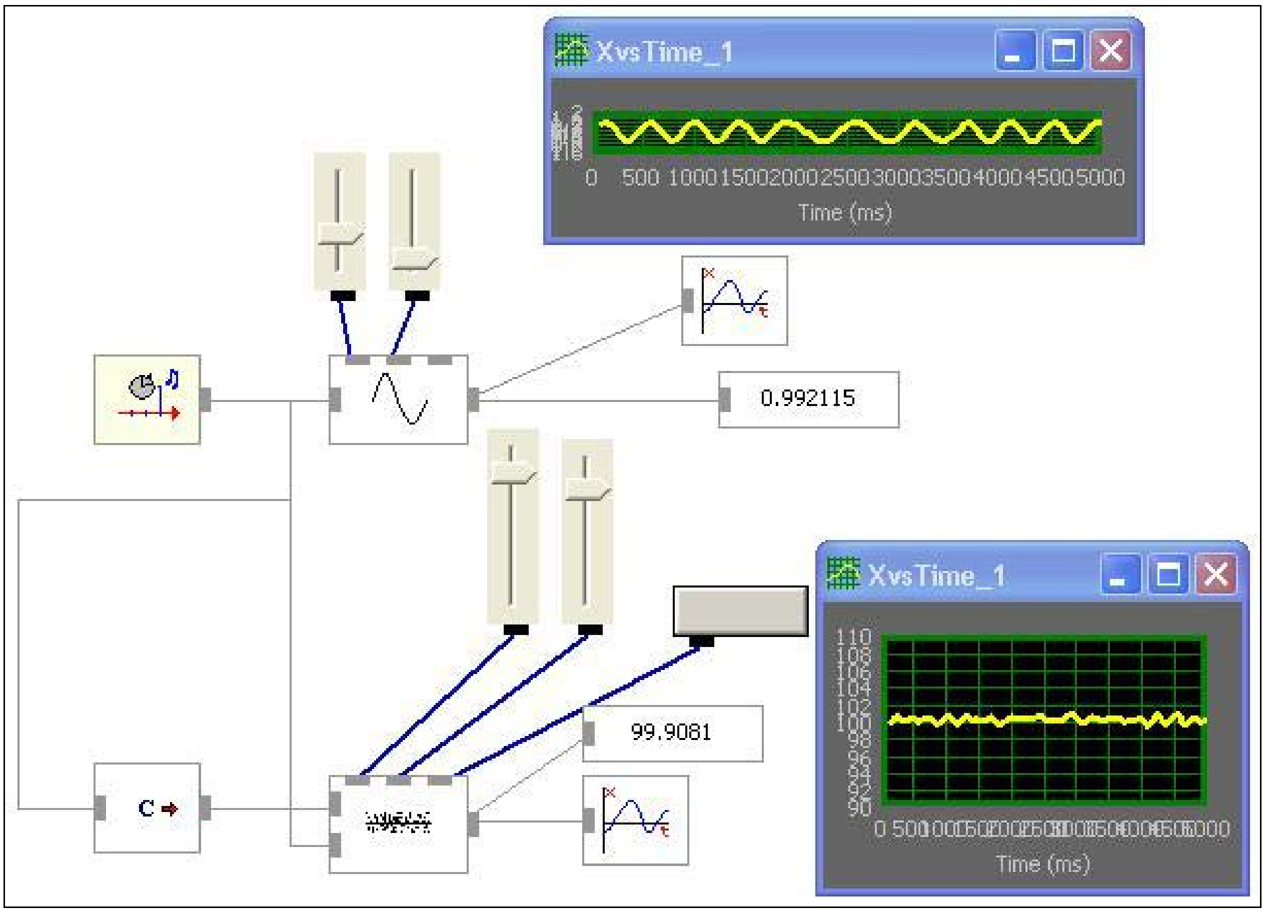

Figure 3: generation of control signals for frequency microvariations and vibrato