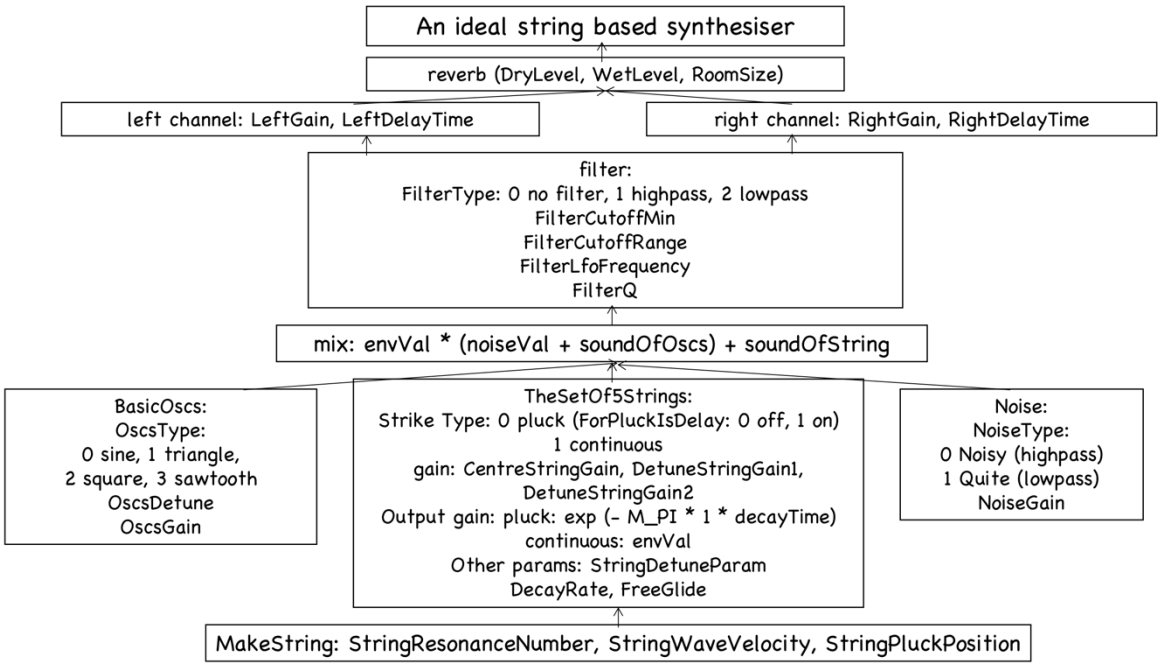
Audio Programming Assignment 3

Author: Yining Xie

An ideal string based synthesiser – in combination with natural sound

In this assignment I made a synthesiser based on physical modelling of ideal string, hence it is able to create a number of timbres by changing some parameters. Besides, white noise and background oscillators are added to make it possible to mimic sounds like wind or water. I was thinking like the situation when someone plays the flute in the wilderness, but the result is more surprising. It has a good balance of music and nature sound more than simply 1+1, and it's possible to produce more magical or cinematic sounds as well. It would be fun to explore what this combination can produce.

The structure is shown in the following graph:



Structure of my synthesiser

It's generally constructed by a combination of 3 classes of sounds:

– basic oscillators such as sine wave, triangle wave, square wave and sawtooth wave. **Users are able to choose which type and each has 2 oscillators of the same kind with a controllable detune;**

– the noise class has 2 types too, producing either **noisier (with high pass filter to simulate moving of leaves, sound of waterfalls, or electronic noise)** or **quieter (with lowpass filter, it sounds more like those of tide, wind in open air, etc.)** sound;

– A set of ***5*** physical modelled strings, with one's frequency exactly same with input midi frequency, and the other two pairs of 2 strings that can be **detuned** from either side of the original frequency. It also introduces 2 different strike types:

a) pluck, like guitar and many of the kind; you can also choose to have it **with or without delay** - in some ways with or without echo.

b) continuous - like violin though in this case it's just a rough simulation - way too completed to add up vibrations as it should be in reality

Then they are mixed and passed through another overall filter, which also includes **3 types for user to choose: no filter, low pass filter and high pass filter**. When filters are implemented, their **cut-off frequencies are able to change with a lfo**, or stay a constant if the lfo frequency is 0.

Then the mix is passed to both left and right channels with **each channel having independent gains and controllable delays. This helps to produce interesting stereo effects.**

Lastly **a reverb with some controllable parameters** is added to play with.

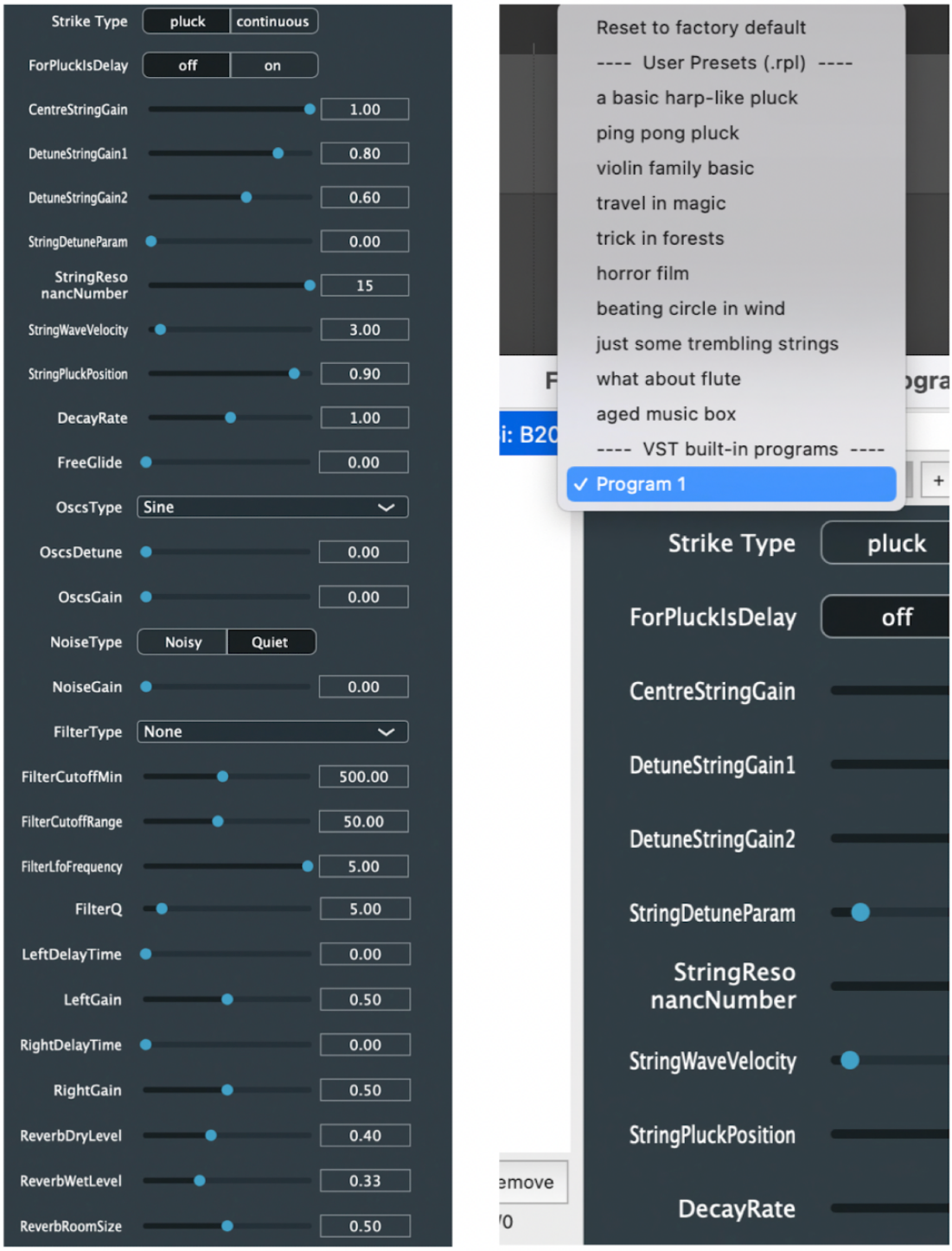
I'd also like to explain the physical modelling of string, which is what the 5 string class is based on. As we haven't learnt the numerical method, I wrote the code **by myself with purely acoustical formulas**. Speaking of ideal strings, loss is not considered, but one special point that might be worthy of noticing is that my modelling **considers the position of pluck** – which often is not something people pay special attention to but does change the timbre as well. Basically it is a summation of vibration modes with time variables that gives displacement hence makes sound. The parameters that change timbre are **number of resonance frequencies, wave velocity and pluck position.**

***** *Note that as far as it is physical modelling, parameters except frequency are not able to change smoothly during the playing of a certain note, this is due to the physical function structures - and when playing an instrument in reality you are not able to change its physical parameters either. Nonetheless they can still be changed between two notes. Another thing to pay attention to is that 'ForPluckIsDelay' only works for the pluck case of the string, as I can't see much meaning to do so in the continuous case.*

The final plugin UI is shown in the following picture, as well as some presets I have discovered. I think the names have already clearly show what they are like. As to the recording demonstration, it is just a sequence of different effects/presets made in limited time. And note that the second mini part is made deliberately with faked noise – using 'aged music box' preset. So as later ones – all distorted sounds are intended.

One parameter that cannot been demonstrated in presets and not used in the recording either is the **FreeGlide. It's a rough mimic of glide** by making the string's frequency changeable even when it's already set by midi input – hence it helps when you want to

mimic a glide with parameter automation in DAW, simply start the value high and drag it down to 0, then it's the midi frequency.



Plugin UI and presets

Those I'd like to highlight are in summary:

- The physical modelling of strings as talked of earlier. In some ways this part is in tribute to GarageBand – *1) work with one or multiple strings, 2) make either discrete or continuous sound, 3) make delays. 4) Of course it can sound polyphonically too.* Differences are that: *5) though physical parameters for 5 strings are the same here, they are controllable by users to change the timbre – there can be a lot, 6) this considers pluck position 7) it's possible to include detunes and detune is design to be symmetric about the middle string 8) delay is more carefully designed to have a losing volume hence in closer to echo;*

- The free glide effect as explained above, just came up with it as I did use such tricks for musical performance of a folk instrument;

- Filtered noise that can produce different kinds of sound from nature – in DAW you can also automize the gain to get more – for example I did once added AM to make it come and go to mimic tide. AM is not included in this plugin as it can be done by changing gain as well;

- A wide range of stereo effect that can be produced by delay in channels. For example see the 'ping pong pluck' in preset, or with shorter delay time you can make sound coming from one ear and fading to the other.

- In coding, I managed to make the pluck case such that it is only related to the moment the key is pressed – it doesn't fade when you stop pressing nor continue to sound if you don't stop. It doesn't affect the next key either. This is more like the real situation and I did take a lot of time to figure this out.
  It was also challenging to put all functions of the MakeString class in the right position of MySynthesiser.h to make it sound normally. There were trade off considerations.

Others are just for you to explore – there are presets beyond my expectation and hope you too! :)