ECS7012U Music and Audio Programming

**Lab Assignment: Pluckable String**

The purpose of this assignment is to build a realistic model of a plucked string using the Karplus-Strong algorithm. Your submission should consist of your source code, a standalone executable or VST instrument plug-in, recorded output audio for each of the steps, and a short report describing your work. answering the questions and depicting results using figures such as waveforms and spectrograms.

**Required Software:**

* Visual Studio, or other development environment supported by Juce. a free on-line text and source code editor
* JUCE
* Any tool to record your audio output, e.g. Reaper, if building a VST instrument, or the Recorder plugin
* **Sonic Visualiser, for producing waveform, spectrum and spectrogram plots from an audio file**

**Step 1:** *Get set up.* Create a new JUCE project, KarplusStrong, as a standalone plugin (we can rebuild it as a VST3 plugin later). Open this in your IDE

**Step 2:** *Pass a simple noise burst generator through a delay line*

a) Have a button on the interface, or just a bool parameter using the generic user interface, that when triggered will produce a burst of noise. Control the amplitude of the noise over time by having it multiplied by a gain parameter that decays to zero over 2 milliseconds.

b) Now connect this output to a delay. Delay the signal by 100 milliseconds. We will vary the delay time at the audio rate and need to avoid unwanted artifacts.

**Step 2:** *Introduce feedback into the delay line*, so the delayed noise burst is fed back into a new GainNode, whose output connects to the input of the delay node. Multiply the feedback signal by a gain between 0 and 1, and try setting delay time to a smaller number e.g. 5 milliseconds.

**Step 3:** When setting delay time to small numbers we start to hear audible frequencies resembling the sound of a metallic string. The decay of the ‘string’ can be set by varying the feedback gain. Use a slider to control the string’s decay.

**Step 4**When plucking a string, high frequencies tend to decay faster than low frequencies. Introduce a lowpass filter, just before or after the feedback gain, that filters out the high frequencies of the feedback signal. Attach a slider to add variable control of the centre frequency. Note that the built-in lowpass filter in the Web Audio API’s BiquadFilterNode gives a boost to the signal near the cut-off frequency, so set the filter’s Q parameter to -3.01 to avoid this.

The model resembles Karplus­Strong synthesis, named after the people who invented it in 1983. It produces a sound similar to that of a guitar string. Karplus­Strong synthesis works because the delay line simulates the way energy travels from one end of the string to the other.

**Step 5:** *Changing the burst signal will change the timbre of the note. Add in an html selector control in the interface to change between the burst being noise, a sinusoid, a triangle wave, a square wave and a sawtooth wave. You can set the frequency of each of those waveforms to 800 Hz, and remember to start the Oscillator.*

Your final model should be controllable using the following four parameters:

● Delay

● Decay (the gain in the feedback loop)

● Frequency preservation (i.e. centre frequency of low pass filter)

● Width (the duration of the short burst of sound that is input into the feedback loop)

● Choice of burst signal

Source code has been provided for the interface of the final implementation, but without any generating sound using the Web Audio API.

**Your submission should consist of the following:**

• Your final javascript and html code

• Audio recordings of the output of your code for each step

• Short report (PDF, 1­2 sides of A4) describing your procedure and results, and briefly addressing the following questions.

* Step 1 a: How could you change the amplitude of the noise burst without using a gain node?
* Step 1 b: You should only be able to hear the sound 100 milliseconds after generating the noise burst. What would you need to do to hear both original and delayed signal?
* Step 2: Why does the feedback signal need to be multiplied by a number smaller than 1? What would happen if we were to use a number greater than 1?
* Step 3: The delay node has a limitation. <https://webaudio.github.io/web-audio-api/#DelayNode> states ‘If DelayNode is part of a cycle, then the value of the delayTime attribute is clamped to a minimum of one render quantum’. Experiment with low values of the delay, and analyse the output sounds in Sonic Visualiser, to see what the actual pitch of the resulting sound is for given delay settings, and how that relates to the delay, the sample rate and the render quantum (number of samples processed in a block by an audio node).
* Step 4: Why do we hear a gradual decay of high frequencies rather than only hearing sounds below the specified centre frequency?
* Step 5: How could one combine burst signals to get the timbral characteristics of different kinds of sources?

Extra credit: These are both optional, and they are quite challenging

1. Using one delay node is like assuming energy only travels in one direction along the string. To create a more accurate simulation we need two delays: one for each end of the string. Extend your model to enable bidirectional movement. Use a slider to vary the plucking position. See the section ‘Simulation of a Moving Pick’ in <http://musicweb.ucsd.edu/~trsmyth/papers/KSExtensions.pdf> for ideas of how to do this.

2. Write your own feedback delay as an audio worklet, so that you can implement shorter delays, and hence higher frequency notes, without the limitation mentioned above.

• Make a .zip file of your code, audio samples and report ,and submit it via QMPlus.