ECS512U Sound Design • 2014

Lab Assignment 2: Stick-Slip Friction

Handed out: Friday, 28 February 2014

Due: Tuesday, 11 March 2014

In this assignment you will first build a model of stick-slip friction, which can be used to create the sound of a creaking door. You will then use an 'analysis and synthesis' approach to extend the model to account for an effect or object attribute you have identified in a sound example.

Required resources:

• Pd-extended, available free for all platforms from http://puredata.info/downloads

• The pd patches and sound examples contained in the *assignment2.zip* file, available on QM+

PART 1

Step 1: *Use a bank of band-pass filters to recreate the frequency response of a door.*

Like many sound models that we have explored in the lectures, a door acts as a filter on any excitation signals that are passed through it (such as knocking or creaking). The door that you will use in this model resonates at the following eight frequencies:

62.5	125	250	395	560	790	1030	1375

a) Create an abstraction that passes an excitation signal through eight individual band-pass filters (`[bp~]`) set at the specified resonant frequencies. Set the Q-factors to 1, 1, 2, 2, 3, 3, 4, 4, respectively. Also use a `[*~ 0.2]` object to pass a small amount of the original signal to the abstraction's outlet. Make sure the patch has a signal inlet (`[inlet~]`) and a signal outlet (`[outlet~]`) and save the abstraction as *filters.pd*.

b) Create a new empty patch and save it inside the same folder as your resonator abstraction. Create an instance of the abstraction and use a noise burst to make sure it is affecting the sound correctly (don't forget to connect the abstraction to `[dac~]` object).

Step 2: *Extend the resonator model using delay-based resonators.*

a) Band-pass filters alone are not enough to simulate a convincing door. Delay-based resonators can be used to create a more lively sound. Copy all the patches from the *assignment2.zip* file into your working directory. Now you should be able to load the abstraction `[dres~]`. The resonator takes a *delay time* (in milliseconds) as a first argument and a *feedback value* (ranging between 0 and 1) as a second argument. These can be reset later using the second and third inlets respectively. Create a new abstraction saved as *delays.pd*. Inside this abstraction create eight instances of the delay-based resonators. Set all the feedback values to 0.1, and set the delay times to values that correspond to the resonant frequencies of the bandpass filters. Let the incoming signal pass through each of the resonators and pass the output of each resonator into a signal outlet.

b) Save your abstraction as *delays.pd* and add it to your main patch. Pass the output of the `[filters]` abstraction through the new `[delays]` abstraction. If you pass white noise multiplied by a value of 0.05 through your abstractions you should hear a similar sound to *freqResponse.wav*, contained in the same zip archive.

Step 3: *Use a pulse generator to recreate the sound of stick-slip friction*

a) Stick-slip friction is the result of complex dynamic behaviour between a moving object and a surface arising from two types of forces known as stiction and friction. The regular bursts of energy occurring as an object repeatedly 'sticks' against a surface and 'slips' along it can be broadly simulated using a pulse-train generator. In lab 3 you saw two different approaches to generating pulse-trains in PureData. The abstraction `[pulse_oversampled]` generates pulses using one of these approaches. Create an instance of it in your patch and pass the output through your resonator chain. The left inlet controls the *frequency* of the pulse-train and the right inlet sets the *index* controlling the width of each pulse. *(If you don't know what this is, go through the Lab 3 worksheet again.)*

b) Try adjusting the frequency to create the sound of a creaking or squeaking (a `[hslider]` with scaled output might be useful here) while using different index settings.

Question: Why does no single index value make the output sound quite like a squeaking door while shifting the frequency from low to high?

c) The abstraction `[calculateIndex]` helps calculate an optimal index. Create an instance of it and pass the current frequency (i.e. the same number going into the left inlet of `[pulse_oversampled]`) into the object's inlet, and the object's output into the the pulse generator's right inlet.

Question: Explain what the calculateIndex abstraction is doing, referring to the resulting shape and frequency of the pulse waveform. (Hint: double-click any abstraction to see or edit its contents.)

d) Save a copy of your patch in its current state before starting the next part of the assignment. This should be included in your submission.

PART 2

Before starting the second part of the assignment, create a copy of your working folder as you will need to submit both versions of the sound model.

In its current state the model is capable of recreating a very basic sound of a creaking or squeaking door. It can be extended and modified to create many different sounds caused by stick-slip friction, including different types of door squeaks. In this task you will be using the design approach of ‘analysis and synthesis’ to extend the model.

a) Listen to the example sounds located in the *examples* folder. Identify aspects of the sounds that the model is not capable of producing in its current state. List at least three of these in your report.

b) Think about how each of these aspects that you have identified could be implemented into the model. Which part of the signal chain is affected? Is the effect controlled by a *dynamic* or a *fixed* parameter?

c) Choose one effect and implement it in your stick-slip friction model, adding all necessary parameters to control it.

d) Explain your approach in as much detail as you can (feel free to use illustrations and/or further sound examples if necessary), describing the parameter(s) and how they affect the signal chain. Also explain your motivations behind the implementation - was your approach based on an understanding of a physical process or on a particular aspect of the sound’s timbre?

When implementing your effect don’t just think about adding new components to the signal chain but also about modifying it. This may entail editing the contents of the abstractions used in the model.

What to submit:

- A zip file containing:
    - A report (2-4 pages), including responses to questions from both parts of the assignment
    - A folder containing the patches (including abstractions) you have created in the **first part** of the assignment
    - A folder containing your final patches (including abstractions) for the **second part** of the assignment
    - Any supplementary material (e.g. sound examples) that you have referred to in your report