**Summary of the Project**

(with code snippets for easier comprehension)

The Karplus-Strong algorithm is an example of digital waveguide synthesis. An instrument is physically modelled and simulated. In this case, the random samples crudely represent the initial pluck: each part of the string is in a random position moving at a random velocity. The delay and feedback cause the waveform to repeat itself, oscillating as a string would. If we just had y[n]=y[n−N], we would have a waveform that repeats with frequency fs/N Instead, taking the average of two consecutive samples acts as a one-zero low-pass filter, mimicking dampening effects of a real string as it vibrates. Higher frequency oscillations lose energy quicker than lower frequency oscillations.

**Play a Note on an Open String**

When a guitar string is plucked or strummed, it produces a sound wave with peaks in the frequency domain that are equally spaced. These are called the harmonics and they give each note a full sound. We can generate sound waves with these harmonics with discrete-time filter objects. Determine the feedback delay based on the first harmonic frequency. Generate an IIR filter whose poles approximate the harmonics of the A string. The zeros are added for subtle frequency domain shaping. To generate a 4 second synthetic note first we create a vector of states with random numbers. Then we filter zeros using these initial states. This forces the random states to exit the filter shaped into the harmonics and normalizes the sound for the audioplayer.

***Code Snippet :***

**delay = round(Fs/A);**

**b = firls(42, [0 1/delay 2/delay 1], [0 0 1 1]);**

**a = [1 zeros(1, delay) -0.5 -0.5];**

**[H,W] = freqz(b, a, F, Fs);**

**plot(W, 20\*log10(abs(H)));**

**title('Harmonics of an open A string');**

**xlabel('Frequency (Hz)');**

**ylabel('Magnitude (dB)');**

**zi = rand(max(length(b),length(a))-1,1);**

**note = filter(b, a, x, zi);**

**note = note-mean(note);**

**note = note/max(abs(note));**

**hplayer = audioplayer(note, Fs); play(hplayer)**

**Play a Note on a Fretted String**

Each fret along a guitar's neck allows the player to play a half tone higher, or a note whose first harmonic is 21/12 higher. Create a 4 second note. Normalize the sound for the audioplayer.

***Code Snippet :***

**fret = 4;**

**delay = round(Fs/(A\*2^(fret/12)));**

**b = firls(42, [0 1/delay 2/delay 1], [0 0 1 1]);**

**a = [1 zeros(1, delay) -0.5 -0.5];**

**[H,W] = freqz(b, a, F, Fs);**

**hold on**

**plot(W, 20\*log10(abs(H)));**

**title('Harmonics of the A string');**

**legend('Open A string', 'A string on the 4th fret');**

**zi = rand(max(length(b),length(a))-1,1);**

**note = filter(b, a, x, zi);**

**note = note-mean(note);**

**note = note/max(note);**

**hplayer = audioplayer(note, Fs); play(hplayer)**

**Play a Chord**

A chord is a group of notes played together whose harmonics enforce each other. This happens when there is a small integer ratio between the two notes, e.g. a ratio of 2/3 would mean that the first notes third harmonic would align with the second notes second harmonic. Define the frets for a G major chord. Get the delays for each note based on the frets and the string offsets. Display the magnitude for all the notes in the chord. Combine the notes and normalize them.

***Code Snippet :***

**fret = [3 2 0 0 0 3];**

**delay = [round(Fs/(A\*2^((fret(1)+Eoffset)/12))), ...**

**round(Fs/(A\*2^(fret(2)/12))), ...**

**round(Fs/(A\*2^((fret(3)+Doffset)/12))), ...**

**round(Fs/(A\*2^((fret(4)+Goffset)/12))), ...**

**round(Fs/(A\*2^((fret(5)+Boffset)/12))), ...**

**round(Fs/(A\*2^((fret(6)+E2offset)/12)))];**

**b = cell(length(delay),1);**

**a = cell(length(delay),1);**

**H = zeros(length(delay),4096);**

**note = zeros(length(x),length(delay));**

**for indx = 1:length(delay)**

**% Build a cell array of numerator and denominator coefficients.**

**b{indx} = firls(42, [0 1/delay(indx) 2/delay(indx) 1], [0 0 1 1]).';**

**a{indx} = [1 zeros(1, delay(indx)) -0.5 -0.5].';**

**% Populate the states with random numbers and filter the input zeros.**

**zi = rand(max(length(b{indx}),length(a{indx}))-1,1);**

**note(:, indx) = filter(b{indx}, a{indx}, x, zi);**

**% Make sure that each note is centered on zero.**

**note(:, indx) = note(:, indx)-mean(note(:, indx));**

**[H(indx,:),W] = freqz(b{indx}, a{indx}, F, Fs);**

**end**

**hline = plot(W,20\*log10(abs(H.')));**

**title('Harmonics of a G major chord');**

**xlabel('Frequency (Hz)');**

**ylabel('Magnitude (dB)');**

**legend(hline,'G','B','D','G','B','G2');**

**combinedNote = sum(note,2);**

**combinedNote = combinedNote/max(abs(combinedNote));**

**hplayer = audioplayer(combinedNote, Fs); play(hplayer)**

**Add a Strumming Effect**

To add a strumming effect we simply offset each previously created note. Define the offset between strings as 50 milliseconds. Add 50 milliseconds between each note by prepending zeros.

***Code Snippet :***

**offset = 50;**

**offset = ceil(offset\*Fs/1000);**

**for indx = 1:size(note, 2)**

**note(:, indx) = [zeros(offset\*(indx-1),1); ...**

**note((1:end-offset\*(indx-1)), indx)];**

**end**

**combinedNote = sum(note,2);**

**combinedNote = combinedNote/max(abs(combinedNote));**

**hplayer = audioplayer(combinedNote, Fs); play(hplayer)**