```
% HyungSeok Yoon
% Creates sound for each types of synthesis

function [soundSample]=generate_sound(instrument,notes, constants)
note_num = zeros(1,length(notes));
tone_num = zeros(1,length(notes));

for i = 1:length(notes)
    if length(notes) ~= 1
        temp = notes{i};
    else
        temp = notes;
    end
    note_num(i) = temp.note(1)-65; % A is ascii 65
    if note_num(i) > 0 && note_num(i) <3
        note_num(i) = note_num(i)+1;
    elseif note_num(i) == 3
        note_num(i) = note_num(i) + 2;
    elseif note_num(i) >3 && note_num(i)<6
        note_num(i) = note_num(i) + 3;
    elseif note_num(i) == 6
        note_num(i) = note_num(i) + 4;
    else
        note_num(i) = note_num(i);
    end
    flatsharp = temp.note(2);
    if flatsharp == '#'
        note_num(i) = note_num(i)+1;
    elseif flatsharp == 'b'
        note_num(i) = note_num(i)-1;
    end
    tone_num(i) = temp.note(end);
end

switch instrument.temperament
    case {'just','Just'}
        load('my_frequency.mat');
        FreqArray = FrequencyLookUp;
        freqslot = 2;
        for i = 1:length(notes)
            frequency(i) = FreqArray(1+(note_num(i)-3),freqslot) * 2^(tone_num(i)-52);
        end

    case {'equal','Equal'}
        for i = 1:length(notes)
            frequency(i) = 440/2^(9/12)*2^(tone_num(i)-52)*2^((note_num(i)-3)/12);
        end

    otherwise
        error('Improper temperament specified')
end

noteduration = constants.durationChord*constants.fs;

switch instrument.sound
    case {'Additive'} % Bell from figure 4.28
        Amplitude = [1,0.67,1,1.8,2.67,1.67,1.46,1.33,1.33,1,1.33];
        Durations_mult = [1,0.9,0.65,0.55,0.325,0.35,0.25,0.2,0.15,0.1,0.075];
        Frequencies_mult = [0.56,0.56,0.92,0.92,1.19,1.7,2,2.74,3,3.76,4.07];
```

```
        Frequencies_add = [0,1,0,1.7,0,0,0,0,0,0,0];
        envelope = zeros(length(Amplitude),noteduration);

        for i = 1:length(Amplitude)
            envelope(i,1:floor(noteduration*Durations_mult(i))) = Amplitude(i)*logspace(0,
-10/log2(10),floor(noteduration*Durations_mult(i)));
        end
        Frequencies = Frequencies_mult.'*frequency + repmat(Frequencies_add.',1,length(fre
quency));
        sinewave = zeros(length(Amplitude),length(frequency),noteduration);
        t = 0:1/constants.fs:constants.durationChord-1/constants.fs;
        for i = 1:length(frequency)
            sinewave(:,i,:) = sin(2*pi*Frequencies(:,i)*t);
        end
        final = sinewave.*permute(repmat(envelope,1,1,length(frequency)),[1,3,2]);
        final = squeeze(sum(final,1));
        if length(frequency)>1
            final = sum(final,1);
        end
        soundSample = final;

    case{'Subtractive'} %implements from the website
        N = 1000; % number of filter changes
        M = 140; %fast change region
        G = N-M;
        midpoint = 0.7;
        fact = [linspace(-1,midpoint,M),linspace(midpoint,1,G)];
        r = 0.85; % given in the website
        a1 = -2*r*fact;
        a2 = r^2;
        num_period = constants.durationChord*frequency;
        wave = zeros(length(frequency),noteduration);
        for i = 1: length(frequency)
            t = linspace(0,2*pi*num_period(i),noteduration);
            wave(i,:) = square(t);
        end
        out = zeros(length(frequency),noteduration);
        out(:,1) = wave(:,1);
        out(:,2) = wave(:,2) - a1(1)*out(:,1);
        filter_step = noteduration/N;
        for i = 3:noteduration-1
            out(:,i) = wave(:,i) - a1(1+floor(i/filter_step))*out(:,i-1) - a2*out(:,i-2);
        end
        out(:,end) = wave(:,end) - a1(N)*out(:,end-1) - a2*out(:,end-2);
        soundSample = sum(out,1);

    case{'FM'} %implementation of bell using FM
        envelope = logspace(0,-10/log2(10),floor(noteduration));
        fm = 200*7/5;
        IMAX = 7; % modified to 7
        F1 = IMAX*envelope;
        % Textbook suggest IMAX*fm but it somehow gives out error saying
        % matrix doesn't match
        % So I changed it to *envelope and it matches
        t = 0:1/constants.fs:constants.durationChord-1/constants.fs;
        fc = repmat(F1,length(frequency),1).*sin(2*pi*fm'*t);
        freqfin = fc + 200;
        final = repmat(envelope,length(frequency),1).*sin(2*pi*freqfin.*repmat(t,length(fr
equency),1));
        soundSample = sum(final,1);
```

```
    case{'Waveshaper'}
        envelope = ones(1,noteduration);
        amplitude_waveshape = 255;
        rise_time = .085;
        decay_time = .64;
        envelope(1:floor(rise_time*noteduration)) = linspace(0,1,floor(rise_time*noteduration));
        envelope(end-floor(decay_time*noteduration)+1:end) = linspace(1,0,floor(decay_time*noteduration));
        envelope = envelope*amplitude_waveshape;
        t = 0:1/constants.fs:constants.durationChord-1/constants.fs;
        wave= repmat(envelope,length(frequency),1).*sin(2*pi*frequency'*t);
        wave = wave + 256;
        waveshaped = wave_shaping(wave);
        soundSample = sum(waveshaped,1);
end
```

```
Not enough input arguments.

Error in generate_sound (line 5)
note_num = zeros(1,length(notes));
```

---

*Published with MATLAB® R2018a*