Jose Maria Bernardez Galera

SYNTH CHALLENGE 2017

Task 1 : Change a song's instrument

The only thing I had to do was the synthesis of a single instrument, in my case it pretend to be a violin. I wrote the synthesis in 'synth.m' function, inside the case of *channel ~= 10* and *synthtype < 64*. I've written some explanations in the script below

```
% I synthesised the instrument (Violin)

if channel~=10; % Always working here
    if synthtype<64 % Always working here

        x_signal = sawtooth(2*pi*freq*t); % I've tried some, and this is the best x_signal
        F = [500,1500,3000,4000]; % Frequency parameters of violin
        BW = [300,200,700,1500]; % Bandwith parameters of violin

        % In the next loop I find the parameters of the filters
        for x = 1:length(F)
            theta = F(x)*2*pi/Fs;
            R = 1-BW(x)*pi/Fs;
            B(x) = 2*R*cos(theta);
            C(x) = -R^2;
        end

        % I filter the x_signal four times
        y = filter(1,[1,-B(1),-C(1)],x_signal);
        y = filter(1,[1,-B(2),-C(2)],y);
        y = filter(1,[1,-B(3),-C(3)],y);
        y_signal = filter(1,[1,-B(4),-C(4)],y);

        o2 = interp1([0 .3 .8 1],[0 1 .7 0], t/t(end)); % To make the sound softer (don't hear a 'click' at the end of each note)
        y = amp.*y_signal.*o2 ; % I multiply the result of the filters for the amplitude of each note and the interp. param.

    else
        y = amp.*square(2*pi*freq*t);
    end
end
```

Then I simply call the function 'synthchallenge.m' and the new song is generated.

Task 2 : Musical scale with the synthesized instrument

I create some parameters (a valid *sample frequency*, a random *amplitude* and a coherent *duration* for each note).

I also create the frequency vector that contains the frequency of every note that will sound. The reference frequency is usually A4 = 440 Hz, and if we notice that between each note the is a difference of 2^(1/12) with the note that is before, we can easy create the frequency vector.

Finally, I call 'synth.m' and normalize the result.

```
%% TASK 2: Scale (with the synthesized instrument)

clear all
Fs = 48000 ;
amp = 0.8 ;
dur = 0.6 ;
% Musical: C C# D D# E F F# G G# A A# B C
% Between each one: F(C#) = F(C)*2^(1/12)
% Reference frequency A4 = 440 Hz
% f = [C D E F G A B]
f = [440*2^(-9/12) 440*2^(-7/12) 440*2^(-5/12) 440*2^(-4/12) 440*2^(-2/12)...
    440 440*2^(2/12) 440*2^(3/12)] ;
result = zeros(1,10e3) ;
for i = 1:length(f)
    y = synth(f(i),dur,amp,Fs,1); % creation of the (violin) sound
    y = y-mean(y); y = y./max(abs(y)); % normalization (avoids signal saturation)
    result = [result y] ;
end

% Saving the scale into m4a file
cd result
audiowrite(['scale.m4a'],result,Fs);
cd ..
```

Task 3 : Arbitrary synthesis

I've created two synthesis, the first one is a basic gun repeated every 0.05 seconds so it appears to be a shotgun.

```
%% TASK 3: ARBITRARY SYNTHESIS
clear all

% WATERING PLANTS MACHINE
fs = 48000;
dur=2;
t = 0:1/fs:dur-1/fs;
x1=randn(1,dur*fs);
X=[0 .12 1];
Y=[1 0.07 0];
r=0.89;
f0=1200;

x=filter(1,[1 -2*r*cos(2*pi*f0/fs) r.^2],x1);
O=interp1(X,Y,t/t(end));
y=x1.*O;
for i=1:20
    soundsc(y,fs); pause(0.5);
end
```

The second one is the sound of a helicopter that is getting closer to the target. The input was to sawtooth signals with a very low frequency (but different). To make the helicopter getting closer I used the same technique that we used for the wind in lectures, but the Fmax is not high compared to Fmin in order to make credible the movement of the helicopter. Finally used a filter to obtain the output signal.

```
% HELICOPTER GETTING CLOSER
fs = 48e3;
dur = 10;
f1 = 20;
f2 = 30;
t = 0:1/fs:dur-1/fs;
x = 0.1*sawtooth(2*pi*f1*t)+0.1*sawtooth(2*pi*f2*t);

X = [0 .15 .3 .4 .5 .65 .7 .75 .8 .85 .9 .95 1];
Y = [0 .2  .1 .3 .05 .3  1 .8   1  .5  1 .7 .1];
Fmin = 100; Fmax = 400; % Fmax not very high so the approach of the vehicle is credible
y = (Fmax-Fmin)*Y+Fmin;

fr = interp1(X,y,t/t(end));

B = 200; % Bandwidth
R = 1-B*pi/fs; % Radius of the pole
b0 = (1-R)*sqrt(1-2*R*cos(2*2*pi*fr/fs)+R*R); % Normalization Coef
a = -2*R*cos(2*pi*fr/fs);

y = zeros(1,length(x));
for n = 3:length(x)
    y(n) = b0(n)*x(n)-a(n)*y(n-1)-(R.^2)*y(n-2);
end
```