[Year]

Politechnika warszawska

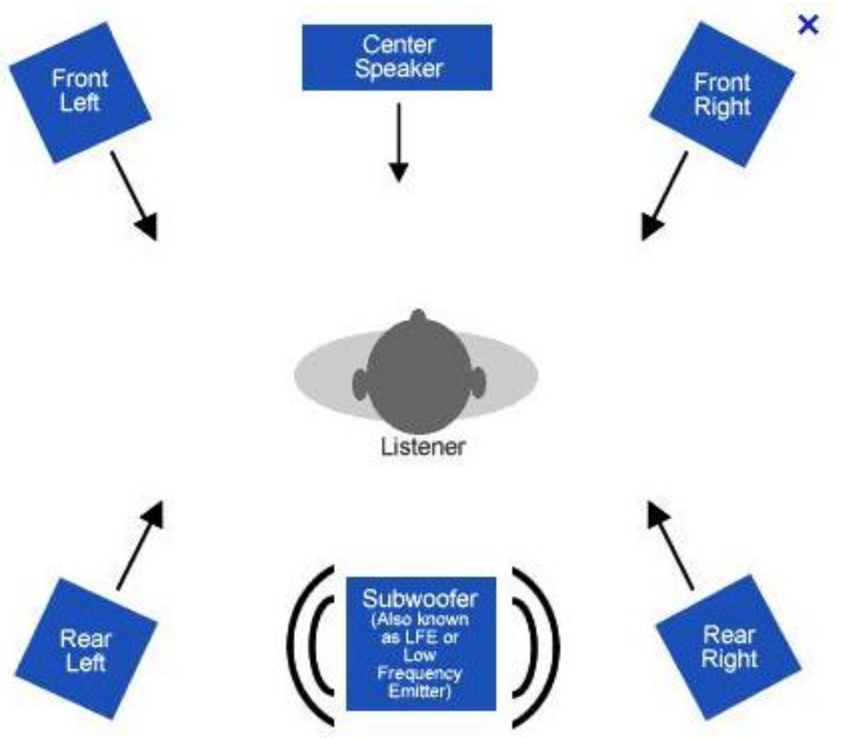
Andres Karabin
Selam GebreAnaneya

KOMUNIKACJA CZLOWIEK I KOMPUTER
GENERACJA I PRZETWARZANIE DZWIEKU

2. test a configuration using the programs "playDirect3DSound" and "Positioning sound" how human being locate sounds in space? how to make surround sound?

Surround Sound Systems and Human brain

The human ear's purpose in the area of hearing is to convert sound waves into nerve impulses. These impulses are then perceived and interpreted by the brain as sound. The human ear can perceive sounds in the range of 20 to 20,000 Hz. This section is broken down into a basic overview of the ear, a section of how sound is received by the ear, a section on how the ears communicate with the brain, and finally, a section of human factors. Understanding how the ear works is key to successfully implementing 3D sound in VR systems.

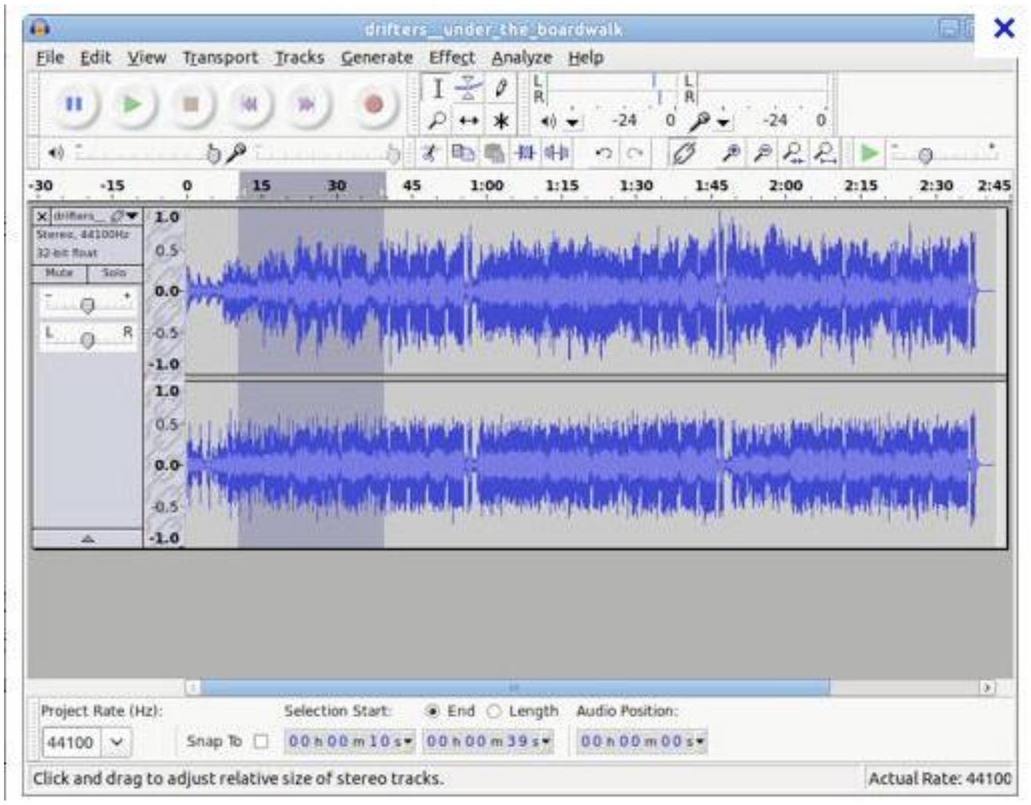


We have use a software to see how the surrounding sound system works .which demonstrate the activities of each speakers and in which way they work.

Our second activity was to create echo manually using software called audacity , **Audacity** is the name of sophisticated, open-source audio recording software, and the user can even edit the recorded audio.

In our laboratory we have performed an echo signal by adding the a signal to the original signal .this additional signal at first was the same as the original signal so in order to create an echo we add a delayed signal of the original signal and the magnitude of this deformed signal was also diminished .

The first step was to create a signal from the original signal which has 30% delays from the original one .then add this signal with the original and adjust the delay until we get the perfect echo.


drifters_under_the_boardwalk
File Edit View Transport Tracks Generate Effect Analyze Help
-30 -15 0 15 30 45 1:00 1:15 1:30 1:45 2:00 2:15 2:30 2:45
1.0 0.5 0.0 -0.5 -1.0

Project Rate (Hz): 44100
Snap To
Selection Start: 00 h 00 m 10 s
End: 00 h 00 m 39 s
Audio Position: 00 h 00 m 00 s
Click and drag to adjust relative size of stereo tracks.
Actual Rate: 44100

By performing this activity for longer time we went to a conclusion that the far the projection of the signal the less the echo sound and when we increase the time delay of the signal we can detect the echo clearly . when the strength of the sound(in DB) gets low then the echo can be differentiated .

4. Opracowac efekty echa i pogłosu (takie same jak w punkcie 3) z wykorzystaniem programu Matlab operując na próbkach zapisanych w plikach dźwiękowych.

The code starts reading a normal file (odliczanie.wav) and extracting from it the wave, the framerate of the sound and the bit encoding of the sound in different variables.

The variable *vector* works as a **delay** for our second wave (the echo sound wave) which we are gonna delay around 4% from the original sound wave. The purpose of *vector* is to be added at the begining of the copy wave in order to simulate the delay.

Wav3 is a variable that concatenates the original sound (now as a copy) and the delay of 4% (vector of zeros).

Dif is a variable that calculates de difference of lenght between the original sound and the echo sound (which should have a bigger lenght) the purpose of this variable is to be make both waves of the same lenght in order to make matrix operations over both.

Vector2 is a variable we will use to add zeros at the end of the original sound wave in order to make the lenght be the same of the echo wave sound.

Wav4 now is a concatenated original sound wave with zeros at the end (now Wav4 and Wav3 have the same length).

At the end *wav_final* is the summatory of *wav3* and *wav4* in order to join both, the original and the echo wave, notice that the wav3 is multiply by 0.5, this is the amount of volume, in a echo effect this represents something very important. **The lower the volume of the echo, the higher should be the delay (as the distance of the regresation wave in a real echo).**

To finish, we reproduce the final wave plus its echo filter using the function *sound*().

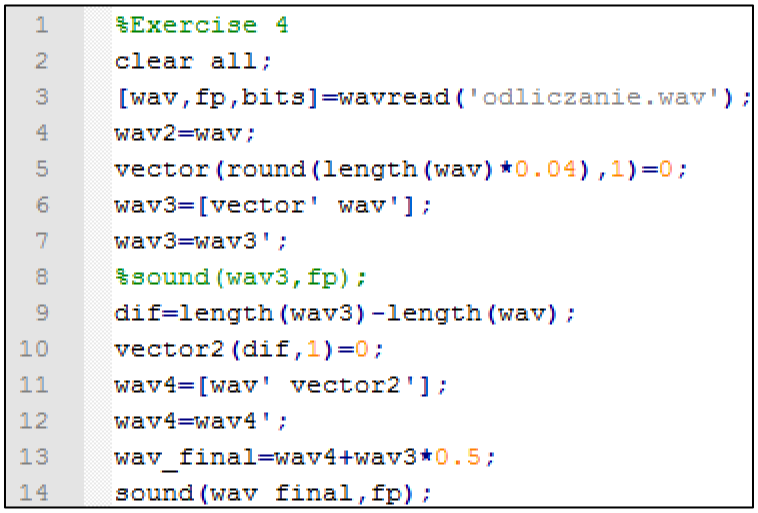

```
%Exercise 4
clear all;
[wav,fp,bits]=wavread('odliczanie.wav');
wav2=wav;
vector(round(length(wav)*0.04),1)=0;
wav3=[vector' wav'];
wav3=wav3';
%sound(wav3,fp);
dif=length(wav3)-length(wav);
vector2(dif,1)=0;
wav4=[wav' vector2'];
wav4=wav4';
wav_final=wav4+wav3*0.5;
sound(wav_final,fp);
```

Fig. 1. Echo filtering script

5. Opracować filtry cyfrowe w środowisku Matlab, które umożliwiłyby usunięcie zakłocenia (w postaci sygnału sinusoidalnego o określonej częstoliwości i amplitudzie) z plików. Najpierw projektujemy filtr dolnoprzepustowy plik, przy użyciu filtrów ellip() i remez(), pózniej projektujemy filtr dolnoprzepustowy również przy użyciu filtrów ellip() i remez().

Low-pass filter (using *ellip()* function):

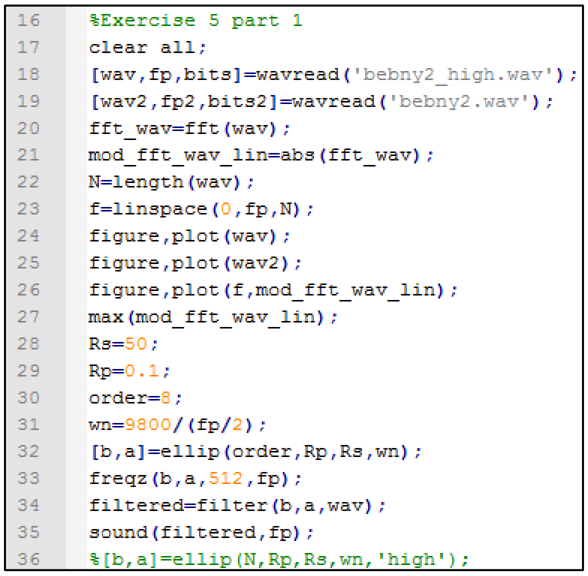
The aim of this exorcise is to eliminate non desired noise of a wave sound, for this purpose we will create a low pass filter which will focus in a high frequency sound added into the original sound wave.

As the previous exorcise, we will separate from a wave file different variables such as, the wave, the frame rate, and the bit rate from a given file *('bebny2_high.wav')* which have added noise. Then we applied a Fourier Transformation which will express our wave (in function of time) in a wave in the function of frequency, that result will be stored in the variable *fft_wav.*

We apply an absolute value to this result in order to express a graphic with positive picks that express the sound abnormalities in the original wave sound.

N is a variable which will extract the length of the original wave, then using the function *linspace* we can create the plane where we are going to plot the absolute value of the signal after the fast fourier transformation.

Notice that the function *max* is used to return the maximum value of the wave after the Fourier transformation, variables *Rs*, *Rp* and *order* are necessary to set up the low pass filter which will be applied to the wave sound. Those values can be calculated or in this case expressed by visual calculation over the graphic of the Fourier transformation, the result of this filter over the wave sound should be near of cleanness. The ***ellip()*** function designs an order n lowpass digital elliptic filter with normalized passband edge frequency.



```
%Exercise 5 part 1
clear all;
[wav,fp,bits]=wavread('bebny2_high.wav');
[wav2,fp2,bits2]=wavread('bebny2.wav');
fft_wav=fft(wav);
mod_fft_wav_lin=abs(fft_wav);
N=length(wav);
f=linspace(0,fp,N);
figure,plot(wav);
figure,plot(wav2);
figure,plot(f,mod_fft_wav_lin);
max(mod_fft_wav_lin);
Rs=50;
Rp=0.1;
order=8;
wn=9800/(fp/2);
[b,a]=ellip(order,Rp,Rs,wn);
freqz(b,a,512,fp);
filtered=filter(b,a,wav);
sound(filtered,fp);
%[b,a]=ellip(N,Rp,Rs,wn,'high');
```

Fig 2. Low pass filtering code, first version

Low-pass filter (using *firpm()* function):

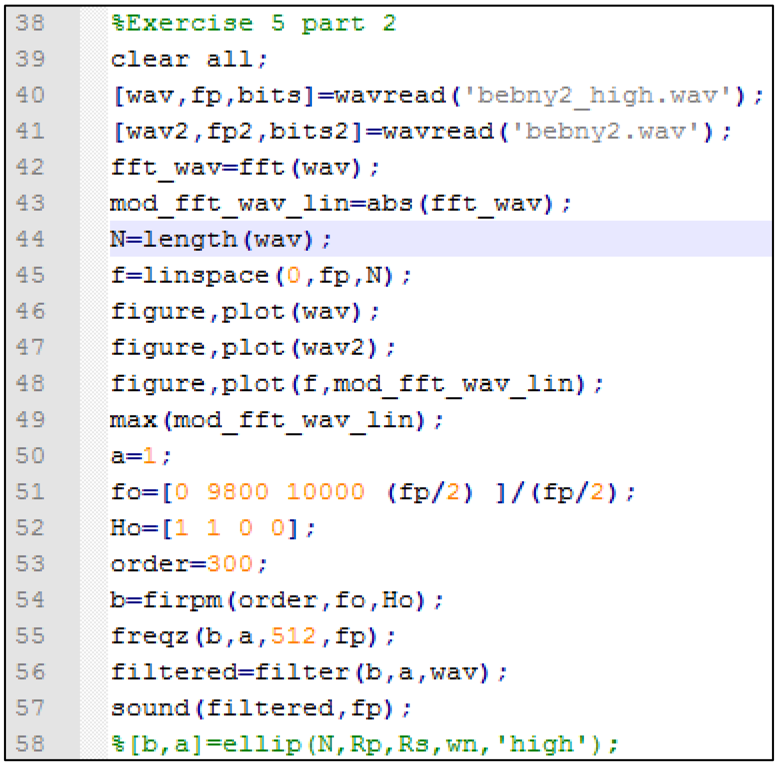
The purpose of this filter is similar to the previous script, we can establish some differences between the quality of both results applying different filtering methodologies.

Firpm() designs a linear-phase FIR filter using the Parks-McClellan algorithm. The Parks-McClellan algorithm uses the Remez exchange algorithm and Chebyshev approximation theory to design filters with an optimal fit between the desired and actual frequency responses.

The filters are optimal in the sense that the maximum error between the desired frequency response and the actual frequency response is minimized.

fo is a vector of pairs of normalized frequency points, specified in the range between 0 and 1, where 1 corresponds to the Nyquist frequency. The frequencies must be in increasing order.

H is a vector containing the desired amplitudes at the points specified in f.



```
%Exercise 5 part 2
clear all;
[wav,fp,bits]=wavread('bebny2_high.wav');
[wav2,fp2,bits2]=wavread('bebny2.wav');
fft_wav=fft(wav);
mod_fft_wav_lin=abs(fft_wav);
N=length(wav);
f=linspace(0,fp,N);
figure,plot(wav);
figure,plot(wav2);
figure,plot(f,mod_fft_wav_lin);
max(mod_fft_wav_lin);
a=1;
fo=[0 9800 10000 (fp/2) ]/(fp/2);
Ho=[1 1 0 0];
order=300;
b=firpm(order,fo,Ho);
freqz(b,a,512,fp);
filtered=filter(b,a,wav);
sound(filtered,fp);
%[b,a]=ellip(N,Rp,Rs,wn,'high');
```

Fig 2. Low pass filtering code, second version.

Conclusion:

Both filters are very efficient, although the *ellip()* filter precision is based on the given data (Rs,Rp,Order) which could be calculated or not affecting the resulting wave sound. In the other hand, the *firpm()* filter is based on the given data in the vector f0, both filters depends in the calculations and their efficiency is closely related to this aspect.