

CPS621 Winter2022
Lab03 Report
Name: Tusaif Azmat, Student#: 500660278.

Work to Do:

3. Choose the sampling frequency $fs=44100$ and generate the necessary notes you need for your chosen melody. Then, generate the melody and save it in the following three file formats: (1) ".wav" with BitsPerSample=24; (2) ".wav" with BitsPerSample=8; (3) ".mp4" with default parameters.

Work done for task 3 below:

This script was used to generate the audio files for the melody for $fs = 44100$ and 24 bit and 8 bit depth in .wav and mp4 files.

```
#####
% CPS 621 Winter2022
% Lab03
% Name: Tusaif Azmat Student#: 500660278.
#####

% Work to do Section for task 3
fs = 44100;

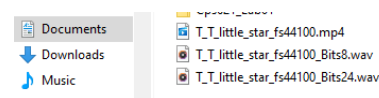
% eighth note as 0.5 sec
do = key(52,8,fs); %do 1
re = key(54,8,fs); %re 2
mi = key(56,8,fs); %mi 3
fa = key(57,8,fs); %fa 4
so = key(59,8,fs); %so 5
la = key(61,8,fs); %la 6
si = key(63,8,fs); %si 7

% quarter note as 1 sec
do_4 = key(52,4,fs); %do 1
re_4 = key(54,4,fs); %re 2
mi_4 = key(56,4,fs); %mi 3
fa_4 = key(57,4,fs); %fa 4
so_4 = key(59,4,fs); %so 5
la_4 = key(61,4,fs); %la 6
si_4 = key(63,4,fs); %si 7

% make a song
line1 = [ do do so so la la so_4];
line2 = [ fa fa mi mi re re do_4];
line3 = [ so so fa fa mi mi re_4];
line4 = [ so so fa fa mi mi re_4];
line5 = [ do do so so la la so_4];
line6 = [ fa fa mi mi re re do_4];
% complete song melody Twinkle Twinkle Little Star
song = [line1 line2 line3 line4 line5 line6];

%Playing and saving : (1) ".wav" with BitsPerSample=24;
sound(song,fs,24);
audiowrite('T_T_little_star_fs44100_Bits24.wav',song,fs,'BitsPerSample',24);
%Playing and saving : (2) ".wav" with BitsPerSample=8
sound(song,fs,8);
audiowrite('T_T_little_star_fs44100_Bits8.wav',song,fs,'BitsPerSample',8);
% saving : (3) ".mp4" with default parameters
audiowrite('T_T_little_star_fs44100.mp4',song,fs);

function wave = key(p, n, fs)
    t = 0:1/fs:4/n;
    idx = 440*2^((p-49)/12);
    mid = (t(1)+t(end))/2;
    tri = -(abs(t-mid)-mid);
    tri = tri./max(tri);
    wave = (sin(2*pi*idx*t)).*tri;
end
```



As you can see above the save three files as required.

4. Change the sampling frequency to $f_s = 8000$ and generate another set of three audio files as in Step 3.

Work done for task 4 below:

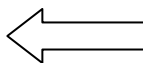
This script was used to generate the audio files for the melody for $f_s = 8000$ and 24 bit and 8 bit depth.

```
#####  
% CPS 621 Winter2022  
% Lab03  
% Name: Tusaif Azmat Student#: 500660278.  
#####  
  
% Work to do Section for task 4  
fs = 8000;  
  
% eighth note as 0.5 sec  
do = key(52,8,fs); %do 1  
re = key(54,8,fs); %re 2  
mi = key(56,8,fs); %mi 3  
fa = key(57,8,fs); %fa 4  
so = key(59,8,fs); %so 5  
la = key(61,8,fs); %la 6  
si = key(63,8,fs); %si 7  
  
% quarter note as 1 sec  
do_4 = key(52,4,fs); %do 1  
re_4 = key(54,4,fs); %re 2  
mi_4 = key(56,4,fs); %mi 3  
fa_4 = key(57,4,fs); %fa 4  
so_4 = key(59,4,fs); %so 5  
la_4 = key(61,4,fs); %la 6  
si_4 = key(63,4,fs); %si 7  
  
% make a song  
line1 = [ do do so so la la so_4];  
line2 = [ fa fa mi mi re re do_4];  
line3 = [ so so fa fa mi mi re_4];  
line4 = [ so so fa fa mi mi re_4];  
line5 = [ do do so so la la so_4];  
line6 = [ fa fa mi mi re re do_4];  
% complete song melody Twinkle Twinkle Little Star  
song = [line1 line2 line3 line4 line5 line6];  
  
%Playing and saving : (1) ".wav" with BitsPerSample=24;  
sound(song,fs,24);  
audiowrite('T_T_little_star_fs8000_Bits24.wav',song,fs,'BitsPerSample',24);  
%Playing and saving : (2) ".wav" with BitsPerSample=8  
sound(song,fs,8);  
audiowrite('T_T_little_star_fs8000_Bits8.wav',song,fs,'BitsPerSample',8);  
% Not Supported : (3) ".mp4" with default parameters  
%audiowrite('T_T_little_star_fs8000.mp4',song,fs);  
  
function wave = key(p, n, fs)  
    t = 0:1/fs:4/n;  
    idx = 440*2^((p-49)/12);  
    mid = (t(1)+t(end))/2;  
    tri = -(abs(t-mid)-mid);  
    tri = tri./max(tri);  
    wave = (sin(2*pi*idx*t)).*tri;  
end
```

Documents

Downloads

T_T_little_star_fs8000_Bits8.wav
T_T_little_star_fs8000_Bits24.wav



As you can see above the save two files as required but .pm4 is not supported for windows.

5. Check the length of your melody and compute the theoretical file sizes corresponding to the two sampling frequencies (44100 and 8000) and the two bit depths (24 and 8) you used in Steps 3 and 4. Then, check if they are equal to the real file sizes you generated.

Work done for task 5 below:

The Source code is checking the real file sizes for audio file generated in part 3 and 4.

```
#####
% CPS 621 Winter2022
% Lab03
% Name: Tusaif Azmat Student#: 500660278.
#####

%Part 5 Check the length of your melody and compute the theoretical file sizes
Melody_44100KHz_24bits = audioread('T_T_little_star_fs44100_Bits24.wav');
%get the .wav file information
fprintf('Melody_44100KHz_24bits, SampleRate: %d, Duration(s): %7.4f, Bits per Sample: %d\n', ...
    Melody_44100KHz_24bits.SampleRate, Melody_44100KHz_24bits.Duration, Melody_44100KHz_24bits.BitsPerSample);
Melody_44100KHz_8bits = audioread('T_T_little_star_fs44100_Bits8.wav');
%get the .wav file information
fprintf('Melody_44100KHz_8bits, SampleRate: %d, Duration(s): %7.4f, Bits per Sample: %d\n', ...
    Melody_44100KHz_8bits.SampleRate, Melody_44100KHz_8bits.Duration, Melody_44100KHz_8bits.BitsPerSample);
Melody_44100KHz_mp4 = audioread('T_T_little_star_fs44100.mp4');
%get the .wav file information
fprintf('Melody_44100KHz_mp4, SampleRate: %d, Duration(s): %7.4f, BitRate: %d\n', ...
    Melody_44100KHz_mp4.SampleRate, Melody_44100KHz_mp4.Duration, Melody_44100KHz_mp4.BitRate);
Melody_8000KHz_24bits = audioread('T_T_little_star_fs8000_Bits24.wav');
%get the .wav file information
fprintf('Melody_8000KHz_24bits, SampleRate: %d, Duration(s): %7.4f, Bits per Sample: %d\n', ...
    Melody_8000KHz_24bits.SampleRate, Melody_8000KHz_24bits.Duration, Melody_8000KHz_24bits.BitsPerSample);
Melody_8000KHz_8bits = audioread('T_T_little_star_fs8000_Bits8.wav');
%get the .wav file information
fprintf('Melody_8000KHz_8bits, SampleRate: %d, Duration(s): %7.4f, Bits per Sample: %d\n', ...
    Melody_8000KHz_8bits.SampleRate, Melody_8000KHz_8bits.Duration, Melody_8000KHz_8bits.BitsPerSample);

>> Lab03
Melody_44100KHz_24bits, SampleRate: 44100, Duration(s): 24.0010, Bits per Sample: 24
Melody_44100KHz_8bits, SampleRate: 44100, Duration(s): 24.0010, Bits per Sample: 8
Melody_44100KHz_mp4, SampleRate: 44100, Duration(s): 24.0094, BitRate: 128
Melody_8000KHz_24bits, SampleRate: 8000, Duration(s): 24.0053, Bits per Sample: 24
Melody_8000KHz_8bits, SampleRate: 8000, Duration(s): 24.0053, Bits per Sample: 8
fx >>
```

I will use the above information to calculate the theoretical sizes of the audio files as required in part 5.

Calculations:

1. For Melody audio file ('T_T_little_star_fs44100_Bits24.wav') with

fs = 44100 and BitsperSample = 24 bits


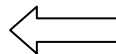
Duration(s) = 24.0010 seconds

Audio is Mono (Single Channel)

Bit Depth: 24 bits

Sampling Frequency = 44100

Theoretical file size = $1 \times 24.0010 \times 44100 \times 24 / 8 / 1024 / 1024 = 3.028232861 \text{ MB} \sim 3.03 \text{ MB}$

 T_T_little_star_fs44100_Bits24.wav	Size: 3.02 MB (3,175,370 bytes)	
	Size on disk: 3.03 MB (3,178,496 bytes)	

*File size verified by windows file properties.

2. For Melody audio file ('T_T_little_star_fs44100_Bits8.wav') with

fs = 44100 and BitsperSample = 8 bits

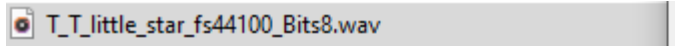
Duration(s) = 24.0010 seconds

Audio is Mono (Single Channel)

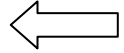
Bit Depth: 8 bits

Sampling Frequency = 44100

Theoretical file size = $1 \times 24.0010 \times 44100 \times 8 / 8 / 1024 / 1024 = 1.009410954 \text{ MB} \sim 1.01 \text{ MB}$



Size: 1.00 MB (1,058,486 bytes)



*File size verified by windows file properties.

3. For Melody audio file ('T_T_little_star_fs8000_Bits24.wav') with

fs = 8000 and BitsperSample = 24 bits

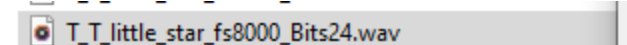
Duration(s) = 24.0053 seconds

Audio is Mono (Single Channel)

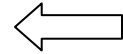
Bit Depth: 24 bits

Sampling Frequency = 8000

Theoretical file size = $1 \times 24.0053 \times 8000 \times 24 / 8 / 1024 / 1024 = 0.54943771362 \text{ MB} \sim 0.55 \text{ MB}$ or $\sim 550 \text{ KB}$



Size: 562 KB (576,170 bytes)



*File size verified by windows file properties.

4. For Melody audio file ('T_T_little_star_fs8000_Bits8.wav') with

fs = 8000 and BitsperSample = 8 bits

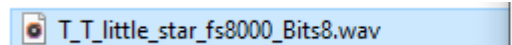
Duration(s) = 24.0053 seconds

Audio is Mono (Single Channel)

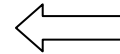
Bit Depth: 8 bits

Sampling Frequency = 8000

Theoretical file size = $1 \times 24.0053 \times 8000 \times 8 / 8 / 1024 / 1024 = 0.18314590454 \text{ MB} \sim 0.18 \text{ MB}$ or $\sim 180 \text{ KB}$



Size: 187 KB (192,086 bytes)



*File size verified by windows file properties.

Note: The file size differences in 3 and 4 points above are due to the rounding error otherwise file created and calculated are same.

Report: Q&A

1. A brief review of Nyquist Theorem and pulse code modulation (PCM) in 1-2 paragraphs.

Answer: The Nyquist Theorem also known as the sampling theorem is useful in Digital to Analog conversion. The theorem states that one must sample a signal twice the rate of the highest significant frequency found in the sample as to avoid any loss of information. So, a sample containing a highest frequency of 1 KHz must be sampled at 2 KHz as to not lose any information.

PCM (pulse code modulation) is a technique that is used to digitally represent a sampled analog signal. The Analog signal is quantized into various discrete levels. The PCM converts an analog signal into digital pulse signal.

2. A brief summary of your chosen melody and how you generate and save the melody (i.e., the key Matlab scripts).

Answer: I choose the same melody (Twinkle Twinkle Little Star) provided in the lab instructions link and used the same pre-created notes as 0.5 seconds and 1 seconds to create the melody. I thought it would be easy to use and work with the provided melody example in lab instructions.

The melody has 6 lines; the melody array is made up of those 6 lines. As can be seen above in the scripts used to create the melody for (frequency) $f_s = 44100$ and 8000 and 24 bit and 8 bit depth. I used the `audiowrite` function of matlab to save the melody into `.wav` and `.mp4` files. I used the `sound` function to play the melody and test them later manually calculated the size of each generated file.

3. The computation details and results of the file sizes in Step 5, the real file sizes you saved in Steps 3 and 4, and brief analysis of why your computation results are equal or not equal to the real file sizes?

Answer: I found out in my computation that the calculated file sizes match the real file sizes. You could see above in the task five that the file size calculations were for $f_s = 44100$, bit depth = 24 bits, the calculate file size was 3.03 MB and the real file size was also 3.03 MB. For 8 bit depth, the calculated file size was 1.01 MB; the real file size was 1.01 MB. For $f_s = 8000$, bit depth of 24 bits, the calculated file size was 0.55 MB and the real file size was 0.564 MB, the results somewhat match this could be due to rounding errors. For 8 bit depth, the calculated file size was 0.18 MB, and the real file size was 0.188 MB. There was some discrepancy in results that I believe could be due to rounding errors.

4. After listening to all the files you generated, can you tell the difference between them? Why?

Answer: Yes, I could tell the difference between them as all the melody sounded somewhat different. The 24 bit depth audio files sound much better than 8 bit depth. The 8 bit depth audio had some noise and speed issues, which are not present on the 24 bit depth. The $f_s = 44100$ with 24 bit depth audio sounds the best. I believe the reason why I could tell the difference between them is probably have something to do with sampling frequency and bit depth of each melody sound.

5. All the source code can be found above in task 3, 4 and 5.