

**Final Music Project**  
**Due: May 2<sup>nd</sup>**

For your final music program problem I would like you to enhance the music player are currently developing. In order to complete this assignment you must first have a working music player. This assignment will be scored as bonus points added to your other music challenge scores. A simple enhancement will be worth 5 points and a more complex enhancement 10 points. You may add more than one enhancement and score up to 25 additional points. Please do come and see me if you need additional background on the music issues or on MATLAB issues.

What enhancements you add is up to you. Possible enhancements include:

- 1) Experimenting with adding overtones (harmonics). These are one of the elements that give various instruments their character. You can learn a bit about harmonics from an online paper at <http://amath.colorado.edu/pub/matlab/music/>. On this page are several links that may be helpful to this (and other) enhancements. Particularly check the link at the bottom of the page to the write up on Musical Harmonics. You will likely need to use a higher sampling frequency in order to successfully add harmonics to your music. For example `makesquare2.m` uses the odd harmonics to create a pseudo square wave. Your implementation should do something different.
- 2) Make you player able to change the key (how high or low) and tempo (how fast or slow) of a song independently (i.e., change the pitch without changing the tempo or change the tempo without changing the pitch). This can be done by mathematically increasing or decreasing the frequency (to change pitch) or the duration (to change tempo) of each note. From the frequency table you should be able to figure out how to manipulate the pitch up or down by notes or octaves. The web article mentioned in 1 above will also be of some help.
- 3) Develop a Graphical User Interface (GUI) for your player. This makes use of the MATLAB Graphical User Interface Development Environment (Guide). I can help you get started on this. (you can also take a look at the gui M3 that can be downloaded from the website)
- 4) Develop a player that can play more than one note at a time (that can play chords and harmonies). There are several ways to do this. You can simply use a constant “campfire harmony” where a second note a major third above the main note. A third is four half steps away from the base note. This is four positions apart on the note/frequency chart I handed out to you. Alternatively you could have the second note be part of your data file and combine them in your program. To play more than one note at a time you will need to add them together. You will likely need to use a higher sampling frequency to ensure that this is successful.
- 5) Set your player to play a song in stereo. You will need to decide what the difference between the two channels will be and how you will get that information into the program. The `sound` and `wavplay` commands can play in stereo if your wave form is a matrix. See the MATLAB help for the `sound` or `wavplay` function.

- 6) Develop a player that automatically plays random notes. Use MATLAB's random number generator and have it change the note, pitch and amplitude. Then play the resulting random song. The random number generator is covered in section 7.3 in your text.
- 7) In general high notes sound louder than low notes when played at the same amplitude. Develop a scheme for compensating for this effect by decreasing the volume of notes as the pitch increases. (See the web article mentioned in number 1 above.)
- 8) Automatically add emphasis on the first and third beat of a measure. This can be done by increasing the amplitude slightly on these beats. This is the emphasis for music in 4/4 tempo. You can also add the appropriate emphasis for other time signatures.
- 9) Add an amplitude envelope to your notes. This is a pattern of peak amplitudes over the time the note plays. When an instrument or voice sounds a note it is not at a single amplitude. An ADSR (Attack, Decay, Sustain, Release) envelope is a common approximation. The amplitude quickly increases to a maximum (Attack) and then decreases (Decay) toward a held amplitude (Sustain) and finally the note drops to zero amplitude (Release). A simple attack and decay pattern could be accomplished using an expression like the one in Chapter 5, problem 21. You would need to apply this expression twice, once with a positive  $b$  and then a slightly modified form for the decay. Other functions could also be used – see me if you for help with how to do this mathematically.
- 10) You could also develop a program that can simulate a wind chime with a given set of notes provided by the user. You would have to develop a random sounding of a limited number of notes selected by a user. Generally the user would choose five or so notes to simulate.
- 11) Propose your own enhancement (or alternative music program). This is a great option.
- 12) You may also propose your own programming project that is not related to music. The goals and point values must be negotiated with me before you begin your project.

Please do talk to me if you have any questions about these or other options. Please email me (1) a user introduction including a detailed description of your enhancement, clear user instructions and a summary of the program's logic, and (2) copy of your m-file(s).