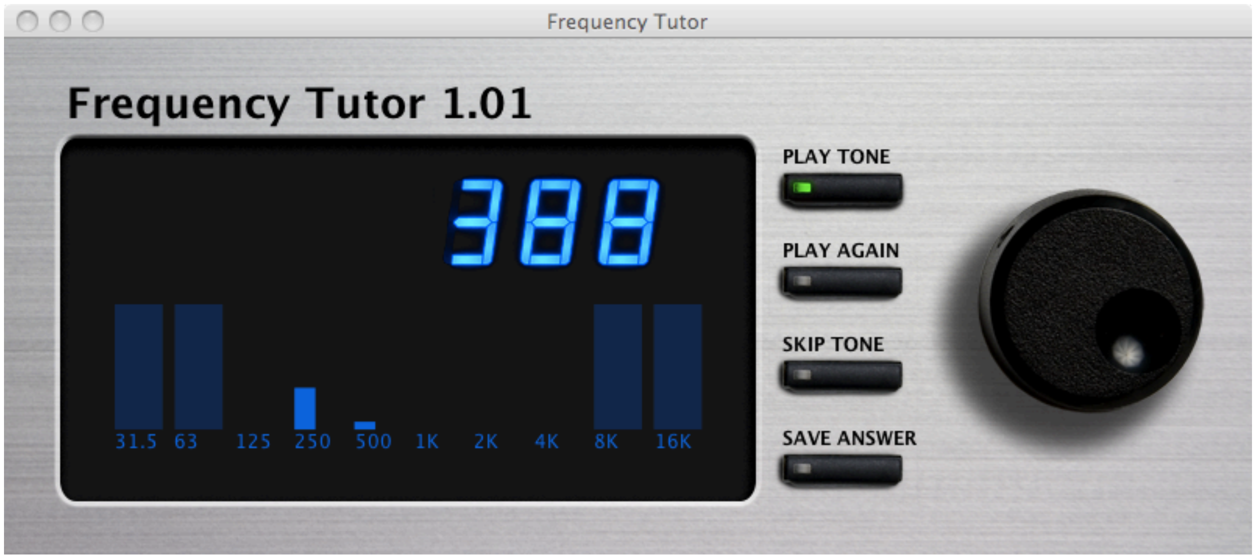
Frequency Tutor Users Guide



Introduction

Frequency Tutor is a standalone Java application designed to train an audio engineer to identify the frequency of a pure tone. This skill is particularly useful to a live sound engineer who must prevent a system from feeding back at certain frequencies; if the engineer could identify those feedback frequencies precisely, the system could be adjusted to reduce its gain only at the problem frequencies, without causing other undesirable effects.

Operation

Frequency Tutor is distributed as a Jar archive. It can be started on most systems by double-clicking on the Jar archive, named `frequency-tutor.jar`. If your system cannot start a Java program in this way, use the following command:

```
java -jar frequency-tutor.jar
```

Once Frequency Tutor has started, the main window is displayed, as shown in the image above. The four buttons displayed control the operation of the program. To the right of the buttons is a large black knob, the answer wheel, used to select your answer among the choices presented. It can be rotated by clicking on it and dragging in a circular fashion.

When Frequency Tutor starts, a new trial is created and placed on the answer wheel. The wheel stores a number of possible answers, one of which is the correct frequency of the tone played. If the Play Tone button is pressed, the trial's correct frequency is played for a second or so. You can then turn the wheel until the correct frequency of that tone is displayed, and press the Save Answer button to record your answer. The display will briefly change to show you the correct frequency of the tone, and your score will be added to the performance graph just under the frequency readout.

The performance graph shows your accuracy in identifying frequencies in each of the ten octave frequency bands. Your error is displayed as the number of octaves that your answer differed from the correct answer, with a full height bar equal to two octaves error.

After the results are displayed, a new trial is calculated and placed on the answer wheel. The program will choose a new frequency based on the results stored in the performance graph, according to this heuristic: a random band is chosen, and if you have never had a tone played from that band, it will be chosen for the new trial. If that band has been used in a trial, your accuracy in that band is noted, and another few random bands are selected. The band that you performed worst in will be used for the next trial.

The performance graph can also be used to control which frequency bands will be used for new trials. If you click on the data bar for a particular band, its performance data is zeroed and a gray bar is drawn over it to indicate that this band is disabled. This will not affect the trial already loaded onto the answer wheel, but it will prevent any new trials from using frequencies located in that octave band. You can click on the band again to enable it.

The process continues, with new trials presented, and new results displayed. Initially, the answer wheel will contain choices spaced one octave apart. If you get the right answer most of the time, the octave spacing between the choices will decrease, making it a little tougher to get the right answer, but also decreasing the error of a wrong answer. If you start to make a series of incorrect answers, the octave spacing between choices will be increased, making it easier to get the right answer.

There are two additional buttons, not yet described, that are largely self-explanatory. The Play Again button will simply play the tone already played, without affecting the scoring. The Skip Tone button will ignore the current trial, and calculate a new trial, again without affecting the scoring.

Frequency Accuracy

The synthesizer algorithm used in Frequency Tutor is extremely accurate due to its use of double precision floating point calculations. It is also capable of generating any frequency within the Nyquist limit of the audio playback hardware, and not just frequencies whose period is an integer multiple of the sample period.

The principal limitation of the accuracy of the synthesizer is the accuracy of the system sample rate. If, for some reason, the sample rate used by the system is not accurately determined, or differs from the value reported to Java, the frequencies generated will be in error.

Also note that the front panel frequency display can only display an integer, but the actual generated frequency is double precision floating point accurate, properly rounded to an integer.

System Requirements

Any modern device that has Java 1.6 installed should be able to run Frequency Tutor. The software was designed to be platform independent, so it should run properly on a wide variety of systems.

Your system's display must be large enough to show an 800 by 330 pixel image, plus any extra graphics that your operating system places around its windows. On many systems, the Frequency Tutor window can be resized, but it will not rescale itself to fit the new window dimensions.

Since Frequency Tutor generates a sine wave using the system's audio hardware, this hardware must be capable of playing audio with 16 or more bits of resolution and a sample rate high enough to allow playback of high frequencies. If a format with 16 or more bits is not available, Frequency Tutor will not start up.

The tones played back to the user are generated in real time. If the system is underpowered or heavily loaded, tiny clicks or glitches may be heard when a tone is played back. Future versions will provide a more advanced buffering scheme to prevent this problem, but the current version appears to be usable on a number of systems.

Source Code

Frequency Tutor is free software, licensed to you according to Version 3 of the GNU Public License, or any later version. As such, the full Java source code is distributed along with the compiled Java code, allowing you to modify the operation of the software, study it, repair any problems that you find, and so on.

Please read the details of the GNU Public License, a copy of which is included in the release in the file `COPYING.txt`, to determine your rights and responsibilities. If you make any modifications to this code, please send me any changes you have made so that these can be incorporated into the official release of Frequency Tutor, allowing others to also benefit from your work.

To extract a copy of the GNU Public License from this release, execute the following command from the same directory as the release Jar file:

```
jar xf frequency-tutor.jar COPYING.txt
```

Alternatively, you can download a copy of the GNU Public License from the Free Software Foundation at this URL: http://fsf.org/

The source code is packaged in the Jar file containing the executable, and it can be extracted using the Jar tool. If this interests you, please get the most current Java development kit in order to compile your changes. This program was developed using Eclipse, a very powerful integrated development environment, and I highly recommend it for developing Java code.

Graphics Files

Frequency Tutor uses a number of graphics files to create the user interface. These files are displayed together to form a composite image that is photo-realistic. These files use the PNG format with an RGBA encoding. This allows an image to be partially transparent, revealing portions of the image below. For example, the buttons have transparency around their labels, allowing the front panel brushed aluminum to show through, and the entry knob has a shadow, defined in its Alpha Channel, that is cast on the front panel image.

The graphics files are normally stored in the Jar file containing the release, and nothing special needs to be done to use them. If you would like to override the built in files, extract the default files from the Jar archive using the following command:

```
jar xf frequency-tutor.jar gfx
```

This command will create a directory called `gfx` next to the Jar file, containing all of the files needed to create the user interface. If you place a PNG file in this directory with the same name as a file that was unpacked, your version of that file will get used at run time. Make sure that the file you use as a replacement has the same size as the standard file that it replaces, so that the layout of the graphics elements is not altered.

The `gfx` directory that you create for your own modified files does not have to have all of the necessary graphics files; any file that is not found in the `gfx` directory will be retrieved from the Jar file, and any file present in the `gfx` directory will override the file stored in the Jar file. Thus, it makes sense to place only modified versions of the graphics files in your `gfx` directory.

Problem Solving and FAQ

Clicks in the Generated Audio

The audio generation algorithm uses buffering to try to produce and play back the tone smoothly in real-time. However, on some systems, especially underpowered or heavily loaded ones, you may hear clicks while the tone is playing. These clicks are random, and not an indication of a problem with your system or the audio hardware. A future version will employ a more complex buffering scheme to minimize the likelihood of these clicks.

Startup Problems

If Frequency Tutor does not start up properly, check the Java console for possible error messages. Currently, Frequency Tutor does not generate useful error messages on its own, so it might be difficult to determine the exact nature of a problem. This will be fixed in a future release, but for now, the Java console is the place to find error messages.

Here's an example of what you might see if the Java class loader cannot find the class containing the `main()` method:

```
6/13/11 Jun 13, 2:59:47 AM
     [0x0-0x8bc8bc].com.apple.JarLauncher[40821]  Exception in
thread "main" java.lang.NoClassDefFoundError: Frequency_Tutor
```

In this case, your Java installation might not be configured correctly. Check the `classpath` that Java uses to load executable code and make sure that '`.`' (dot) is in the path.

If Frequency Tutor cannot find one of the graphics files used for the user interface, or if it is somehow corrupt, it may fail with a null pointer exception. This is probably because you created a `gfx` directory to override the built-in graphics, but it contains a broken graphics file. In this case, delete the `gfx` directory located next to the file `frequency-tutor.jar` to force it to use the built-in graphics, and then resolve the issue you have with your custom graphics.

Other Issues and Feedback

My goal is to make sure that Frequency Tutor is useful and stable software, able to run on a wide variety of computers and other devices. I would appreciate hearing about any problems you have with the software, as well as any suggestions you may have for improvement.

While I can't guarantee a reply or a solution to your problem, please email any comments or suggestions to monte@alum.mit.edu and I will try to address the issue.

I am also quite interested in knowing whether using Frequency Tutor has helped you to identify frequencies more accurately, how you use the software, and any suggestions you have for improving the training process.

Have fun, and I hope you find Frequency Tutor to be useful!