Mission 9: POTS and Pans

Start date: 10 September 2017
Due: 16 September 2017, 23:59

Readings:

- Textbook Section 2.5

Technical Handbook for Dual-Tone Multi-Frequency Signaling

The technical handbook on the DTMF specification recovered by the technical crew goes like this:

Dual-tone multi-frequency signals consist of dial tones, which are two sine waves of different frequencies playing
concurrently with short pauses in between as symbol delimiters.
There are twelve pairs of such sine waves to represent the ten Arabic numerals and two special symbols. (Table 1)

Symbol	1209Hz	1366Hz	1477Hz
697Hz	1	2	3
770Hz	4	5	6
852Hz	7	8	9
941Hz	*	0	#

Table 1. Supported symbol and corresponding frequency pairs

To 'dial' the 'phone number', the sender's communicator needs to vary the pair of sine waves in time.
Your mission is to dial a phone number or a series of phone numbers according to the DTMF protocol.

You are given the following two utility functions in addition to the list processing library:

- `simultaneously(list_of_sound)` takes a list of `sound`s as argument, and returns a `sound` consisting of those sounds played
  simultaneously.
- `consecutively(list_of_sound)` takes a list of `sound`s as argument, and returns a `sound` consisting of
  those sounds played consecutively.

You are also given the following sound generating functions:

- `silence(duration)` takes in a duration (in seconds), and return a sound that is silent when played.

- `sine_sound(frequency, duration)` takes in a frequency (in hertz) and a duration (in seconds).

This mission has **four** tasks.

Task 1:

You will need to generate many DTMF signals to dial phone numbers, so it is best to prepare suitable data structures and functions for that.

You need to represent the frequencies for each numeral and the symbol `#` by a sequence whose `n`th component (`n` starting at 0) is the pair of corresponding frequencies. Write a function `get_dtmf_frequencies` that takes an integer from 0 to 9 or the number 10 for '#' as argument and returns a pair of `frequencies` of the corresponding two sine waves. In your function, use the sequence of pairs that you have constructed.

Task 2:

Now you have a utility function to retrieve the proper pair of frequencies.
Your next task is to construct sine waves with these frequencies that are half a second long.
Write a function `create_dtmf_tone` that takes a pair of frequencies as argument and returns a `sound` that consisting of the simultaneous overlay of the two sine waves.

Task 3:

You are very close to the goal of dialing numbers. Write a function `dial` that takes a list of digits (0–9) as argument and returns a `sound` conforming to the DTMF protocol, with `0.1s silence` between each tone as delimiter.

Task 4:

You now have everything you need to make a call. Unfortunately, when you tried calling some numbers, you discovered that many of them do not work. You need to figure out which ones of the phone numbers in the list is able to connect.

Write a function `dial_all` that takes a list of phone numbers (each of which is a list of digits 0–9) as argument
and returns a `sound` that dials all phone numbers.
Note that this time each phone number must end with `#` to attempt the POTS connection to the receiver's communicator.
Remember that your `get_dtmf_frequencies` also gives you the frequencies for `#`.

But watch out: Under no circumstances should you ever dial Darth Vader's number, which is of course 18005211980.

However, if a number includes Darth Vader's number as a sub-sequence, you should dial that number. For example, you
should dial the numbers 4518005211980 and 180052119800, but not the number 18005211980 itself.
You are required to use `map`, `accumulate` and `filter` for this task.

Submission

To submit your work to the Source Academy, place your program in the "Source" tab of the online editor within the mission page, save the program by clicking the "Save" button, and click the "Submit" button. Please ensure the required function from each Task is included in your submission. Note that submission is final and that any mistakes in submission requires extra effort from a tutor or the lecturer himself to fix.

IMPORTANT: Make sure you've saved the latest version of your work by clicking the "Save" button before finalizing your submission!