Mission Sidequest 10.2: The Magical Tone Matrix

Start date: 16 September 2017
Due: 24 September 2017, 23:59

Readings:

- Textbook Section 2.5

Background Information

The Pentatonic Scale

A pentatonic scale, in music, is a musical scale or mode with *five* notes per octave, in contrast to a heptatonic scale (introduced in Mission 14) which has seven notes. Pentatonic scales are extremely prevalent all over the world: they occur in Celtic folk music, German folk music, American music, and musics of many other cultures. They are at the heart of Chinese music (in which the Chinese people call them *gong shang jue zhi yu*), and they are also exactly the musical scale used by the Trebbls for their magical Tone Matrix. Music that uses only pentatonic scales tends to sound pleasant with any combination of the pitches in the scale.

In the magical Tone Matrix of the Trebbls, the five notes used in an octave are C, D, E, G, A. With twelve-tone equal temperament, the pentatonic scale consists of the note numbers 0, 2, 4, 7 and 9, if you number the twelve semitones from 0 to 11.

The Tone Matrix

A Tone Matrix consists of a 16 by 16 grid, in which the rows represent sounds of different pitches produced by a particular musical instrument. Each column represents a time at which the sound should be played. The player can keep any of the fields full or empty. The matrix starts playing the left-most column, and the sounds corresponding to the full fields of the respective instruments in that column are played concurrently. After a given `distance` seconds, the matrix plays the second column, after another `distance` seconds, it plays the third column, and so on. When it reaches the last column, its starts again at the beginning.

To get a feeling for the tone matrix, go to http://tonematrix.audiotool.com and click around — you will find magic.

This mission has **two** tasks.

Task 1:

To build the Tone Matrix, the first step is to experiment with the harmonic core of the Tone Matrix—the pentatonic scale.

Write a function `pentatonic_scale` that takes two arguments, a base MIDI note `b` and the number `n` of notes to be generated, and returns a list of `n` consecutive ascending notes of the pentatonic scale, starting with `b`.

Thus, the function will have the following type:

$$\texttt{pentatonic\_scale} : (Number, Number) \rightarrow List(Number)$$

For example, `pentatonic_scale(60, 10)` returns

```
list(60, 62, 64, 67, 69, 72, 74, 76, 79, 81)
```

You can also test using

```
play(consecutively(map(function (note) {
return trombone(note, 0.5);
}, pentatonic_scale(60, 10))));
```

Task 2:

The Source Academy provides a tone matrix that can be accessed any time using the function `get_matrix()`. It returns a list of lists of boolean values, such that

```
list_ref(list_ref(get_matrix(), my_row), my_column);
```

evaluates to `true` if the matrix currently is filled in row `my_row` and column `my_column`, and `false` otherwise. You count the rows starting from the bottom of the matrix, and you count the columns from the left. Write a function `play_matrix(distance, list_of_sounds)` which continuously plays the tone matrix. The sounds in subsequent columns start playing `distance` seconds from each other. The rows use the sounds given in `list_of_sounds`, such that the first sound is assigned to the bottom-most row, the second sound is assigned to the second row counting from the bottom, so on and so forth. You can assume that there are at least 16 sounds in `list_of_sounds`.

You may find the following HTML5 function handy, which is available in your environment.

`setTimeout(f, t)`: Calls the function f with no arguments after waiting for t milliseconds.

Try calling the function in the console as follows

```
setTimeout(function() { alert("Sorry I'm late!"); }, 1000);
```

and observe what happens.

A rather limiting feature of our `play` function is that only one sound is played at a time. For this task, we provide a function `play_concurrently`, which can play new sounds while other sounds are playing already.

To aptly conclude the sound missions, provide a function `stop_matrix`, which when called with no arguments, should stop the matrix. It's ok to play on for a few seconds.

For this last task, you may find the following given function useful.

`clearAllTimeout()`: Cancels all previously scheduled but not started setTimeout jobs.

You may find the following behaviors of `setTimeout` useful:

- The second argument does not always imply the actual delay, e.g

```
function f() { display('Test'); }
setTimeout(f, 100);
someComputationThatTake3s();
// Only 3 seconds later Test is displayed
```

- The function passed is evaluated asynchronously , i.e

```
setTimeout(f, 0);
display('Hey');
// 'Test' will be displayed AFTER 'Hey'
```

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!