TONES KeyboardPitch Model, KeyboardNote, and KeyboardNoteGroup

anematode

January 3, 2019

1 Who is this intended for?

It’s intended for me. I might read this in a few months because I didn’t know what the hell I was doing. Thus, it’s written in a somewhat understandable way, but it won’t try to explain ideas that aren’t a product of this project.

2 KeyboardPitch

A keyboard pitch refers to a unique note “on the keyboard.” It does not refer to a specific pitch; this is important because then alternative/ microtonal tuning systems would be hard to implement and that would suck. Instead, it simply refers to something like “A4,” “F4,” etc.

We model a keyboard pitch as an integer from 0 to 144 inclusive, corresponding with the notes C-1 to C11. In the old TONES it was a special class, but this is such a simple egg that I don’t think it deserves another class.

When we convert a note to a ”name” like C1 or Db5, we can use the following formula:

noteToName(note) $:= octaves_{note \mod 12} + stringify(\lfloor note/12 \rfloor - 1)$

where octaves is [“C”’, “C#”’, “D”, “D#”, “E”, “F”, “F#”, “G”, “G#”, “A”, “A#”, “B”] and zero-indexed. For example, 69 gives us index 9 and $\lfloor 5.75 \rfloor - 1 = 4$, or A4.

The inverse of this function is the nameToNote function, which is a bit more complicated to deal with flats (b), double flats (bb), sharps (# or s), and double sharps (## or ss). We first apply the regex

$\hat{}([ABCDEFG])(\#|\#\#|B|BB|S|SS)?(-)?([0-9]+)\$$

to an uppercased version of the input. The first (zero-indexed) group (letter name) we reference Dict 1 to get a semitone offset from the octave (l). The second group (accidental) we reference Dict 2 to get a semitone offset of the accidental (a). The third group we reference to get whether the octave number is negative as a true or false (b). Finally, the fourth group is parsed to an integer as the octave number itself (o). Then the note’s value is

nameToNote(name) $:= l + a + (b\ ? -12 : 12) \cdot o + 12$.

The relevant dicts:

Dict 1: letter_nums = $\{$"C" : 0, "D" : 2, "E" : 4, "F" : 5, "G" : 7, "A" : 9, "B" : 11$\}$; Dict 2: accidental_offsets = $\{$"#" : 1, "##" : 2, "B" : -1, "BB" : -2, "b" : -1, "bb" : -2, "s" : 1, "ss" : 2, "S" : 1, "SS" : 2$\}$;

For utility and pitch estimation, there's the noteTo12TET function which converts a KeyboardPitch to its 12-TET frequency:

noteTo12TET(n, a4 = 440) := $a4 \cdot 2^{(n-69)/12}$.

There's also the concept of a KeyboardInterval, which is just a distance between two KeyboardPitches. Obviously, the KeyboardInterval between a and b is just $|a-b|$. For working with this there are the functions nameToInterval and intervalToName, but I don't expect them to be used and they are just an artifact from the old TONES, so I won't explain how they work here. Their algorithm is lengthy but straightforward. To get the value in cents of a KeyboardInterval **in 12 TET**, multiply by 100.