CMSC 143: Introduction to Object-Oriented Programming with Robots

Lab 2: Scribbler Music

Due September 13, 2010

This lab gives you more practice writing expressions and functions and exploring the mathematics behind making music with the scribbler. As we discussed in class, and in the textbook, a musical note has a particular frequency. For example, the sound of 261Hz corresponds to Middle-C. You can increase a note by one octave by doubling its frequency (Tenor-C at 523Hz). Likewise, you can decrease a note by one octave by halving its frequency (Low-C at 131Hz). It's hard to remember the frequencies for all the different notes, but luckily there are clear mathematical relationships between all the notes. In this lab, we will explore this correspondence in more depth.

Learning Objectives

◦ Write functions that return values. ◦ Utilize parameter passing. ◦ Explore the mathematics of music.

Lab Report

Submit an electronic copy of your program on moodle. It should be named `cmsc143_lab2_NAME.py`. Your program should have your name, email, and the date at the top of the file as a comment.

Scribbler Piano

Each key on the piano corresponds to a frequency. Middle-C (the 40th key on a standard 88-key piano) is 261.63Hz, and A4 (the 49th key) is 440Hz. This formula can be used to find the frequency of a key:

$$freq(key) = 440 \times 2^{\frac{key-49}{12}}$$

Write a python function `pianoToScrib(key)` that returns the correct frequency for that particular piano key. Also, write a `playPiano(time, key)` function that plays the piano key using your `pianoToScrib()` function and myro's `beep()` function. Test your functions using the following C-major scale:

```
def testPiano():
    beep(0.5, pianoToScrib(40)) #C
    playPiano(0.5, 42) #D
    playPiano(0.5, 44) #E
    playPiano(0.5, 45) #F
    playPiano(0.5, 47) #G
    playPiano(0.5, 49) #A
    playPiano(0.5, 51) #B
```

Scribbler Guitar

The scribbler can only play two notes at the same time using the beep function: `beep(time, freq1, freq2)`. This means the scribbler can't really play proper 3-note chords. However, we can play 2-note "power" chords used heavily in rock music. A power chord is the root note and its fifth. A root and its fifth are related by a simple 3:2 ratio: $p5(f) = 1.5 \times f$.

Write a function `power(freq)` that returns the fifth for the root note `freq`. Also, write a function `playPower(time, freq)` that plays the power chord using your new function and myro's `beep()` function. The following code should play "You Really Got Me" by the Kinks.

```
def testGuitar():
    F = pianoToScrib(33)
    G = pianoToScrib(35)
    playPower(.15, F)
    for i in range(5):
        playPower(.15, G)
        playPower(.15, G)
        playPower(.15, F)
        playPower(.15, G)
        wait(.6)
        playPower(.5, F)
```

MIDI Music

MIDI (Musical Instrument Digital Interface) is an industry standard for communication between instruments and computers. MIDI represents notes in terms of pitches rather than raw frequencies. Using this integer notation we convert the frequency into a hole number: C as 0, C# as 1, D as 2, etc. The two following equations convert a frequency to a MIDI pitch and vice-versa:

$$pitch(f) = 69 + 12\log_2(f/440)$$

$$freq(p) = 440 \times 2^{\frac{p-69}{12}}$$

pitch class	note
0	C
1	C#
2	D
3	D#
4	E
5	F
6	F#
7	G
8	G#
9	A
10	A#
11	B

Write two python functions `freqToPitch(freq)` and `pitchToFreq(p)` that implement the above equations[1]. Using the pitch notation, we can find major scales in any key pretty easily. A major scale starts with the key and proceeds according to the progression 2, 2, 1, 2, 2, 2, 1. For example the major-C scale looks like: (0, 2, 4, 5, 7, 9, 11, 12). Write a function `playMajorScale(keyAsPitch)` that uses your `pitchToFreq()` and `beep()` to play major scales (you choose the time interval).

EXTRA: Pitches that are exactly 12 apart (or some multiple of 12) are the same note at different octaves. Using this fact, we can define **pitch classes**. For example, the 0th pitch class (the note C) represents C in all octaves. Write a function `pitchClass(freq)` that returns the pitch class of some particular frequency. The pitch class should be a number between 0-11.

[1] the `math` module has a `log(value, base)` function