Formant Synthesis Tutorial

Goals:

- Learn how to use the AudioLazy toolbox to do simple synthesis of vowels.
- Examine the impact of including various numbers of formants in your synthesis
- Learn how to emulate different speakers by shifting the frequency of vowels.
- Synthesize the entire vowel quadrilateral.

Step 1: Install AudioLazy toolbox and PyAudio in the command line.

If you're using Linux:

```
pip3 install AudioLazy
pip3 install pyaudio
```

If you're using a Mac:

```
Brew install AudioLazy
Brew install pyaudio
```

Step 2: Import modules and set up toolbox.

```
#Import modules
from __future__ import unicode_literals, print_function

from audiolazy import (sHz, maverage, rint, AudioIO, ControlStream,
                       CascadeFilter, resonator, saw_table, chunks)
from time import sleep
import sys

# Initialization
rate = 44100
s, Hz = sHz(rate)
inertia_dur = .5 * s
inertia_filter = maverage(rint(inertia_dur))

#Initialize api options
api = None
chunks.size = 16
```

Step 3: Create a dictionary of vowel formants and define the first two formants for /i/ (“ee” as in “beet”). Create a list of vowels currently just containing “i”.

```
# Initialize formant dictionary with /i/
vowels = [“i”]
formants = {“i”: [437, 2761]}
```

Step 4: Synthesize /i/ using the following code:

```
# Use the with statement to close the player when complete and prevent degradation in audio
# synthesis
with AudioIO(api) as player:
        # Initialize formant streams for synthesis using first vowel
        first_coeffs = formants["i"]
        f1 = ControlStream(first_coeffs[0] * Hz)
        f2 = ControlStream(first_coeffs[1] * Hz)

         # Apply gain to fade in smoothly
        gain = ControlStream(0)

        # Generate signal using cascade filter
        filt = CascadeFilter([
                resonator.z_exp(inertia_filter(f1).skip(inertia_dur), 400 * Hz),
                resonator.z_exp(inertia_filter(f2).skip(inertia_dur), 2000 * Hz),
        ])
        sig = filt((saw_table)(100 * Hz)) * inertia_filter(gain)

        # Begin player in the background
        th = player.play(sig)

        # Play current vowel
        coeffs = formants[“i”]
        print("Now playing: i”)
        f1.value = coeffs[0] * Hz
        f2.value = coeffs[1] * Hz

         # Apply gain to fade in smoothly
        gain.value = 1
        # Sleep to allow vowel to be synthesized by player
        sleep(2)
```

```
    # Fade out sound smoothly
    gain.value = 0
    sleep(inertia_dur / s + .2)
    # Divide by s because here it's already expecting a value in seconds, and we don't
    # want to give a value in a time-squared unit like s ** 2
```

Step 5: Add the first two formants of /u/ (“uh” as in “but”) to the formants dictionary. Add “u” to the vowels array.

```
# Add /u/ to the formant dictionary and to list of vowels
vowels.append(“u”)
formants[“u”] =  [459, 1105]
```

Step 6: Repeat step 4, but loop through both vowels.

Hint: Loop through the list not the dictionary keys to maintain the vowel order! Making this into a function will make your life easier later!

Can you tell the difference between the vowels?

Step 7: Redefine each vowel with the first three formants and repeat step 6.

```
# Add third formants to each vowel
formants[“i”] = [437, 2761, 3372]
formants[“u”] =  [459, 1105, 2735]
```

Hint: Be sure to define the f3 Control Stream, f3 resonator, f3 and f3.value! (Set f3 resonator to 3000 Hz.)

Do the vowels sound qualitatively any different with the addition of a formant?

Step 8: Redefine each vowel with only the first formant and repeat step 6.

```
# Remove formants from each vowel to leave only one
formants[“i”] = [437]
formants[“u”] =  [459]
```

Do the vowels sound qualitatively any different with the removal of a formant?

Step 9: Return to the first two formants. Define new formant dictionaries with higher-shifted and lower-shifted formants. Repeat step 6 and compare each vowel with medium, low and high formants. These shifted formats represent different speakers.

```
#Define low, middle, and high formants
formants[“i”] = [437, 2761]
formants[“u”] =  [459, 1105]
high_frequency = {“i”: [452*1.5, 3081*1.5], “u”: [494*1.5, 1345*1.5]}
low_frequency = {“i”: [342/1.5, 2322/1.5], “u”: [378/1.5, 997/1.5]}
```

Do the vowels sound qualitatively any different with these new formants?

Step 10: Sweep between the vowels by interpolating the “half-way formants.”

Hint: Try taking the mean of each formant!

Step 11: Sweep through the entire vowel quadrilateral as shown below using only the first two formants.

Hint: Start from /i/ and sweep between the vowels in a circular manner.

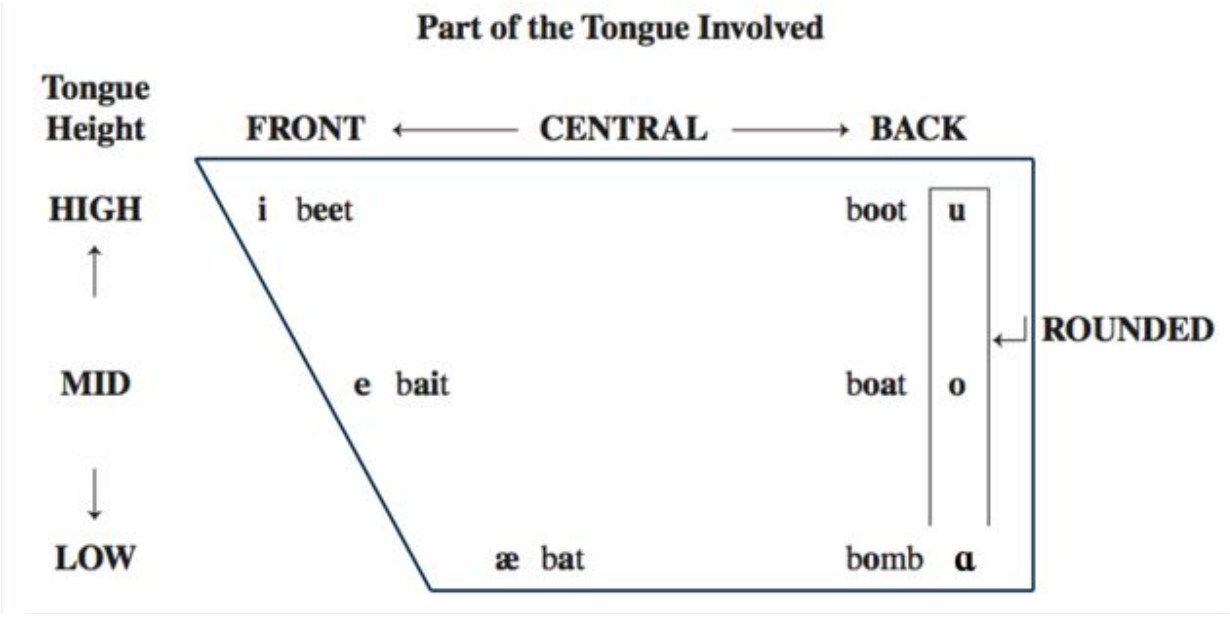

Figure 1: Vowel quadrilateral based on tongue height and position.
(Source: http://www.imagequiz.co.uk/quizzes/95279020)

Vowel	F1	F2
/i/	437	2761
/u/	459	1105
/o/	555	1035
/a/	936	1551
/ae/	669	2349
/e/	536	2530

Table 1: Formants for each vowel in vowel quadrilateral.
(Source: https://www.peterlang.com/view/9783034326179/Chapter02.xhtml)