21M.385 Lecture Notes

Lecture 1

Sound

- Sound = vibrations of the air
- Vibrations can be modeled as physical systems – mass/spring or pendulums
- Most vibrations are sinusoidal and can be modeled with sine waves.
- Example of tuning fork.
- Sine waves, like pressure changes in the air, are added together.
- Microphones: convert physical movement (pressure change) of air into an electrical signal.
- Speakers: convert an electrical signal back into pressure changes in the air

Audio Representation

- ADC (Analog to Digital Converter): turns voltage/current into discrete numbers
- DAC: turns numbers back to an analog voltage/current
- In the digital / computer space, values are discrete, in both time and quanta precision.
  - Time – sampled 44,100 times per second (sometimes 48,000 or higher).
  - Precision/quantization – 16bits per value (~65k discrete values) is enough
  - "44k/16" is CD-quality audio
- Why choose 44100?
  - Nyquest sampling rate: undersampling causes aliasing
  - Threshold of human hearing: ~20,000 Hz
- Mono vs Stereo

Python

- Handy quick language ref: http://rgruet.free.fr/PQR27/PQR2.7.html
- pyAudio – interface for sending/receiving audio data. Note parameters:
  - format: float32 (instead of int16)
  - number of channels: 1 (mono) or 2(stereo)
  - frames per buffer
  - sampling rate
  - device ID
  - input / output
- pyAudio expects data as a python byte string.
- Two modes of operation:
  - Single-threaded & possibly blocking, with variable buffer size.
  - Multi-threaded callback, with fixed buffer size.
  - We will use single-threaded. It avoids threading complications at the risk of audio starvation.
- Frames Per Buffer: internal buffer size used by audio driver.
  - Smaller: more responsive / lower latency, with increased risk of buffer under-run.
  - Larger: less responsive / higher latency, with less risk of buffer under-run.

Examples

- Output noise (random values)
  - Change loudness with gain multiplier.
- Sine function to create a tone
- Adjust frequency of sine function to change pitch
- Fix sine popping problem by keeping track of current frame
- Switch to using numpy to do math quickly. The difference is profound (20-25x speedup).

Modeling Pitch

- $y(t) = \sin(\omega t) = \sin(2\pi f t)$, where f is the frequency in Hertz.
- For discrete time, n (sample number) must replace t.
- $t = nT. S_r = {}^{1}/_{T}$
- $y = \sin\,(2\pi f n / S_r)$, where $S_r = 44100$.
- By the way, concert A = 440Hz.

Kivy Framework

- Create a `BaseWidget` class to make a Kivy application. Note some features:
  - `on_key_down()` and `on_key_up()` to handle keyboard presses
  - Use `Label` to display text on screen (use the helper function `topleft_label()` to create a `Label` at the top-left corner of the screen)
  - Use `on_update()` to update stuff every screen frame (usually 60fps).
- Create a python audio class (`Audio`) that encapsulates `pyAudio.`
- Create a generator to feed data into audio.

Examples

- Use `Label` to display text on screen and modify it with `Label.text`
- Pressing different keys to change pitch.

More Kivy Application

- Use up/down arrow keys to change gain.
- Use left/right arrows to change frequency
  - Listen to aliasing when frequency goes too high.
- A note on `Audio` – choosing a good device index
  - Should (hopefully just work). Different on Mac vs PC. PC should use ASIO drivers.
  - Run `../common/audio.py` to see choices.

Generators

- Useful building block model for managing audio objects and software complexity
- Generators can be chained together into data flow graphs
- Mixer is a class that can take multiple generators and add them together.

Debug Audio

- `AudioWriter` can help you see the audio sent out the speaker. This is a great debugging tool!
- Audio data written to a .wav file
- View the wav file in Audacity.

Perception

- Pitch perception
  - Hearing range: 20Hz – 20,000Hz. But varies with age, gender, and experience.
  - Sweet spot is 30Hz-5,000Hz (like a piano! A0=27.5Hz, C8=4186Hz). Beyond that, we don't hear it as well-defined pitch.
- Loudness perception
  - Equal loudness curve. Sensitivity varies with frequency.
  - Sweet spot (most sensitive) is ~1,000Hz - 3,000Hz
- Both pitch and loudness perception are logarithmic – not linear!

Modeling Intervals

- The most “pure” interval – the octave. All cultures have octave equivalence.
- Pitch perception is not linear. It is exponential (or geometric). *F2* = 2 * *F1*.
- Western scale: 12 “equal” divisions per octave:
    - *F2 = d * d * d * d * d * d * d * d * d * d * d * d * F1*
    - $d = \sqrt[12]{2}$
    - This is the equal tempered scale.
- But, perfectly in-tune intervals are ratios of small numbers
    - Octave = 2
    - Fifth = 3/2
    - Fourth = 4/3
    - Major Third = 5/4
    - Minor Third = 6/5
- $\left(\frac{3}{2} = 1.5\right) \neq (d^7 = 1.498)$. Or worse: $\left(\frac{5}{4} = 1.25\right) \neq (d^4 = 1.260)$

Modeling Timbre

- Pressure waves add together linearly. We model many simultaneous sounds by adding them.
- Example of a plucked string vibrating.
- Modal vibrations: with fundamental frequency f (and $\omega = 2\pi f$), we can model a complex modal tone as: $y = a_1 \sin(\omega t) + a_2 \sin(2\omega t) + a_3 \sin(3\omega t) + \cdots$
- Note the range of $y \in [-1.0,1.0]$. Outside this range, we get clipping. Bad.
- Fourier series: with proper values for a, any repeating waveform can be created.
- Some geometric waveform examples:
    - Square wave: $a_1 = 1, a_2 = 0, a_3 = \frac{1}{3}, a_4 = 0, a_5 = \frac{1}{5}, a_6 = 0, a_7 = \frac{1}{7} \ldots$
    - Sawtooth wave: $a_1 = 1, a_2 = -\frac{1}{2}, a_3 = \frac{1}{3}, a_4 = -\frac{1}{4}, a_5 = \frac{1}{5}, a_6 = -\frac{1}{6}, a_7 = \frac{1}{7} \ldots$
    - Triangle wave: $a_1 = 1, a_2 = 0, a_3 = \frac{1}{9}, a_4 = 0, a_5 = \frac{1}{25}, a_5 = 1, a_6 = 0, a_7 = \frac{1}{49} \ldots$ Note that technically, a triangle wave is a sum of cosines. Our ears can’t hear the difference though.
- When creating these waveforms, save them and inspect what they look like in Audacity.

Modeling Envelopes

- Amplitudes a change over time.
- Overall tone changes over time.
- Basic amplitude envelope controls gain as a function of time.
- ADSR – (Attack, Decay, Sustain, Release) is a common envelope modeling synthesis technique.
- For now, a simplified form: Attack/Decay with a predefined duration.