Playing around with real-time audio routing, analysis, and visualization in Python!

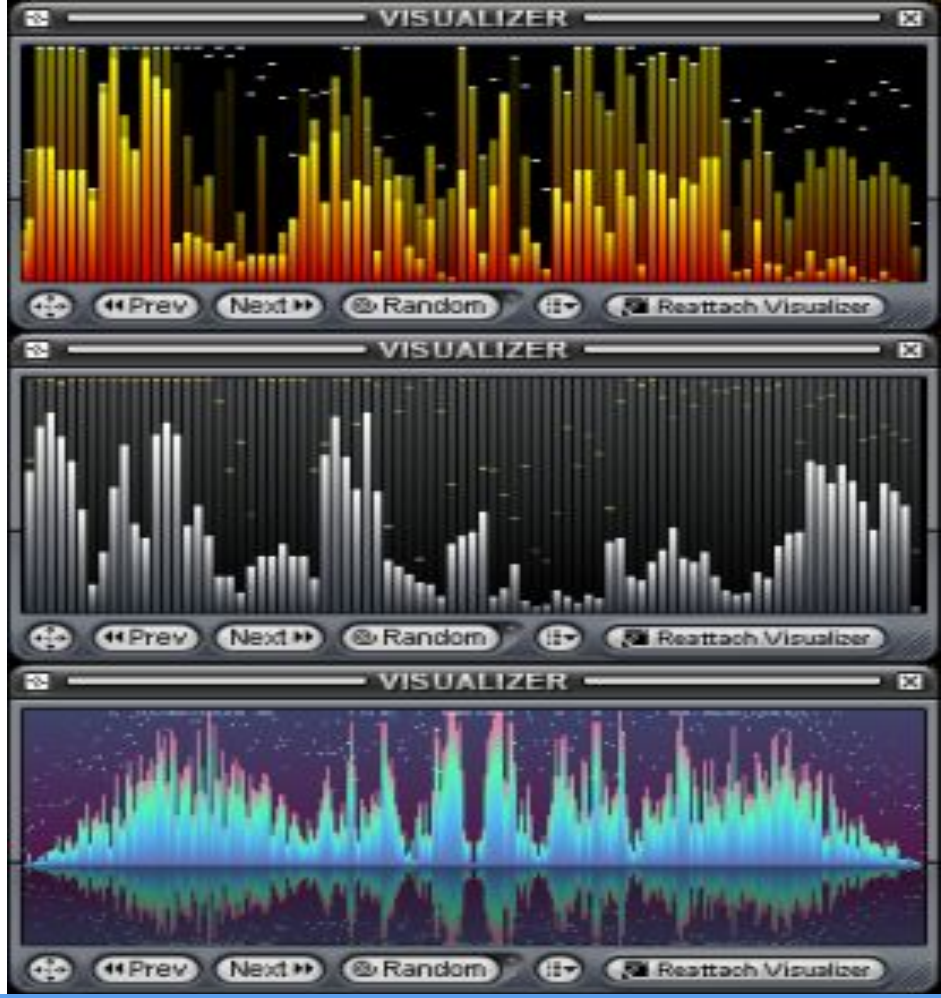
Brian Clee

11/30/18

AUTOMATED INSIGHTS®

The goal

- Real-time audio visualization off the soundcard
- a la winamp
- Captures the soundcard, not a specific file
- All python
- All open source



1st problem

How the heck do you capture audio in real-time off your soundcard?!

1st problem

How the heck do you capture audio in real-time off your soundcard?!

Mac locks down your audio out signal, but input signals are open

1st problem

How the heck do you capture audio in real-time off your soundcard?!

Mac locks down your audio out signal, but input signals are open

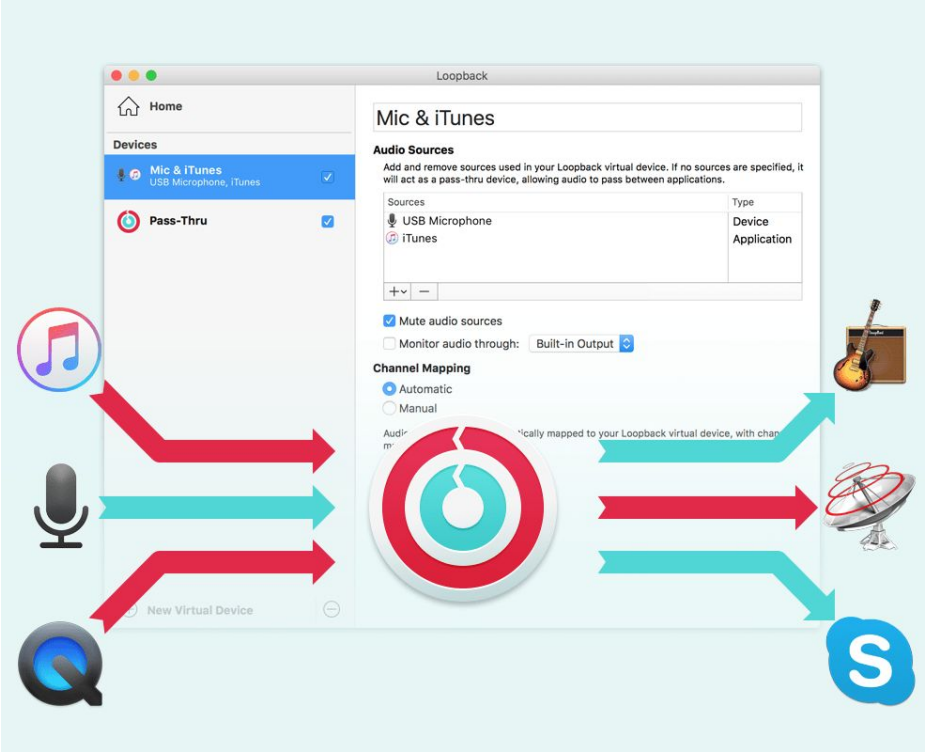
Good ol’ virtual audio routing

1st problem

How the heck do you capture audio in real-time off your soundcard?!

Mac locks down your audio out signal, but input signals are open

Good ol' virtual audio routing

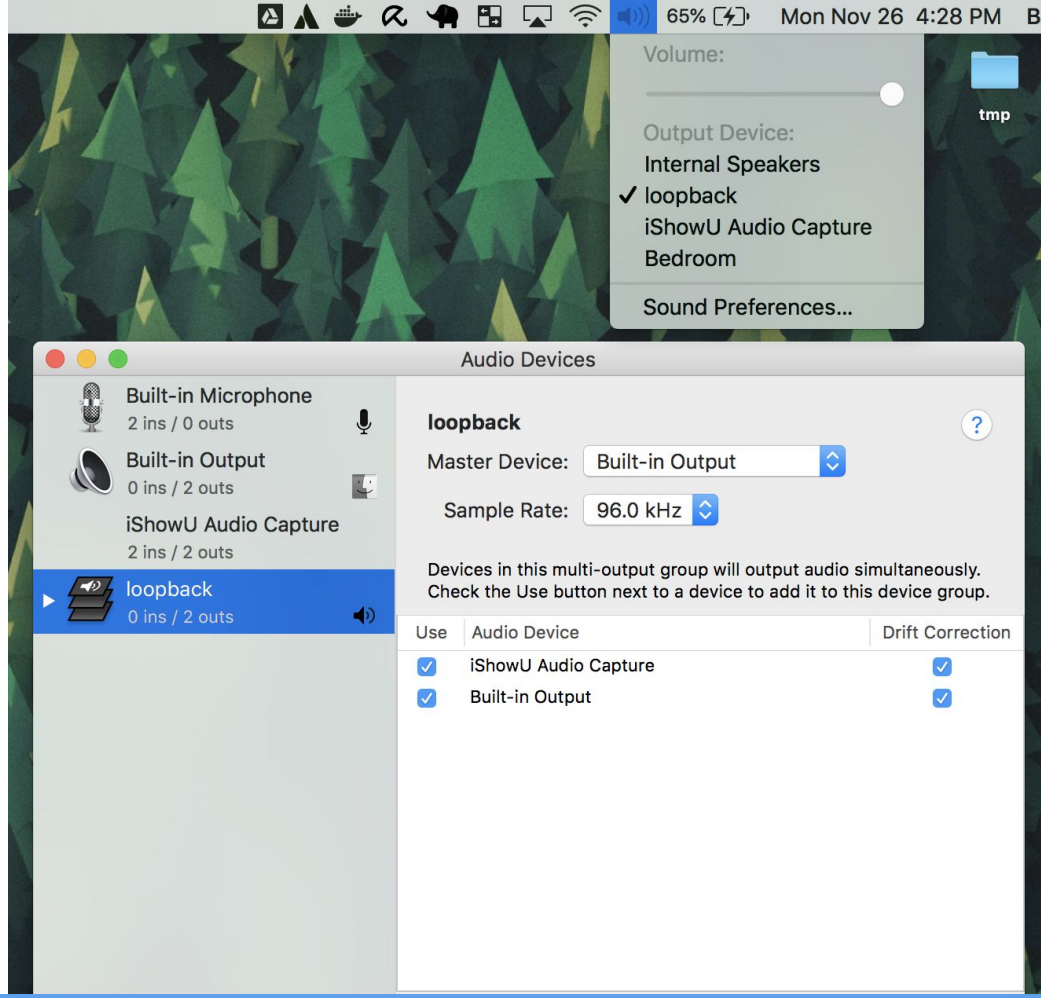


1st problem

How the heck do you capture audio in real-time off your soundcard?!

Mac locks down your audio out signal, but input signals are open

Good ol' virtual audio routing



2nd problem

Soundcard output is being looped back as an input device, how to I capture the audio stream and analyze it?

2nd problem

Soundcard output is being looped back as an input device, how to I capture the audio stream and analyze it?

PyAudio does the trick:

- Python wrapper for the PortAudio library

2nd problem

Soundcard output is being looped back as an input device, how to I capture the audio stream and analyze it?

PyAudio does the trick:

- Python wrapper for the PortAudio library

```
def run(self):
    pa = PyAudio()

    self.stream = pa.open(format=paFloat32,
                          channels=CHANNELS,
                          rate=SAMPLE_RATE,
                          output=False,
                          input=True,
                          stream_callback=self.callback,
                          input_device_index=2,
                          frames_per_buffer=CHUNK)

    self.stream.start_stream()
```

2nd problem

Soundcard output is being looped back as an input device, how to I capture the audio stream and analyze it?

PyAudio does the trick:

- Python wrapper for the PortAudio library

```python
def run(self):
    pa = PyAudio()

    self.stream = pa.open(format=paFloat32,
                          channels=CHANNELS,
                          rate=SAMPLE_RATE,
                          output=False,
                          input=True,
                          stream_callback=self.callback,
                          input_device_index=2,
                          frames_per_buffer=CHUNK)

    self.stream.start_stream()
```

```python
def callback(self, in_data, frame_count, time_info, flag):
    y = np.fromstring(in_data, dtype=np.float32)
    self._analyser.process_data(y)

    return in_data, paContinue
```

3rd problem

We have a numpy array representing the last chunk of our audio stream capture, how do we visualize it?

3rd problem

We have a numpy array representing the last chunk of our audio stream capture, how do we visualize it?

Can use **PyGame** for both windowing and drawing graphics to the screen:

- Different rendering options like **OpenGL**, DirectX, Windib, X11, **linux frame buffer**, etc.

3rd problem

We have a numpy array representing the last chunk of our audio stream capture, how do we visualize it?

Can use **PyGame** for both windowing and drawing graphics to the screen:

- Different rendering options like **OpenGL**, DirectX, Windib, X11, **linux frame buffer**, etc.

```
def _draw_spectrum(self, spectrum):
    # lower right = (0, SCREEN_HEIGHT)
    spectrum_length = int(len(spectrum))
    chunk_size = int(spectrum_length / SCREEN_WIDTH)
    normal = 0 if spectrum.max() == 0 else SCREEN_HEIGHT/spectrum.max()

    index = -1
    for i in range(0, spectrum_length, chunk_size):
        index += 1
        chunk_arrays = [spectrum[i]*normal]
        for j in range(i+1, i+chunk_size):
            if j == spectrum_length:
                break
            chunk_arrays.append(spectrum[j]*normal)
        mean_array = np.mean(chunk_arrays, axis=0)

        x = index
        y = SCREEN_HEIGHT - int(mean_array)
        width = 1
        height = SCREEN_HEIGHT
        pygame.draw.rect(self.screen, GREEN, (x, y, width, height), 0)

    self.update()

def update(self):
    fps = self.font.render(str(int(self.clock.get_fps())), True, WHITE)
    self.screen.blit(fps, (SCREEN_WIDTH - 50, 25))

    pygame.display.flip()

    self.clock.tick(24)
    self.screen.fill(BLACK)
```

live demo

Next steps

www.github.com/cleebp/rta

Next steps

www.github.com/cleebp/rta

MVP is finished:

- Routing ✅
- Stream Capture ✅
- Basic analysis ✅
- Basic visualization ✅

Next steps

www.github.com/cleebp/rta

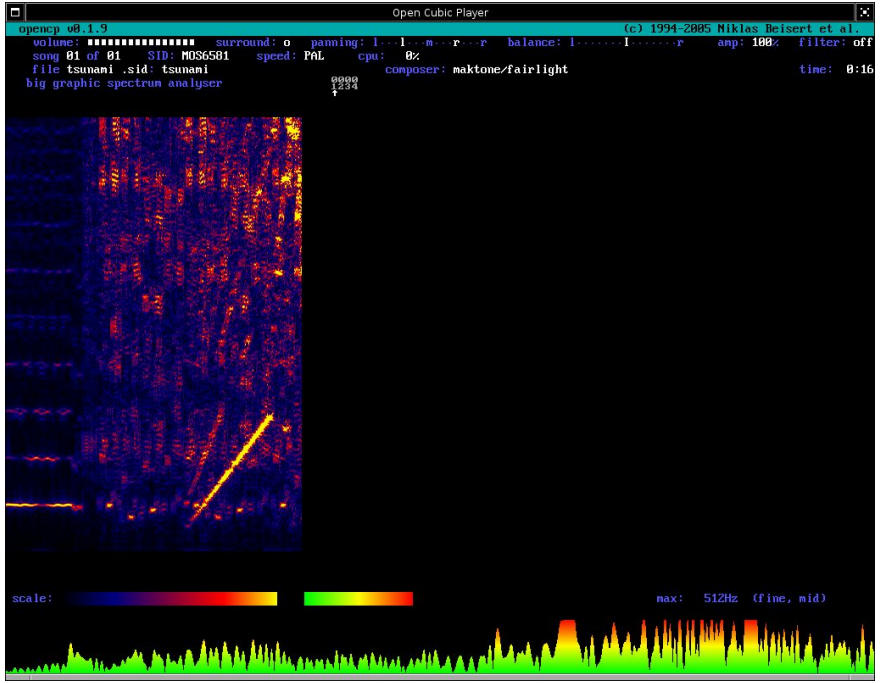
MVP is finished:

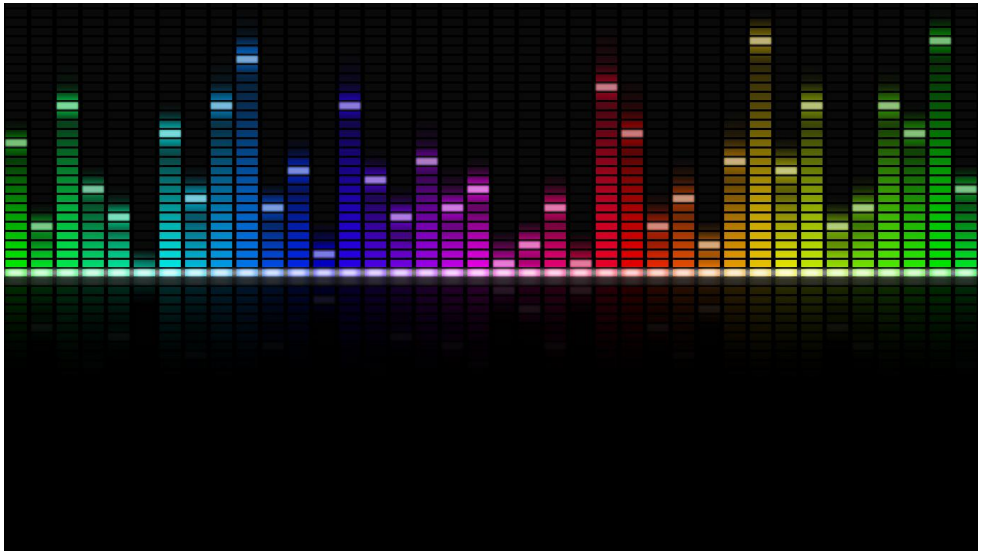
- Routing ✅
- Stream Capture ✅
- Basic analysis ✅
- Basic visualization ✅

Immediate tasks:

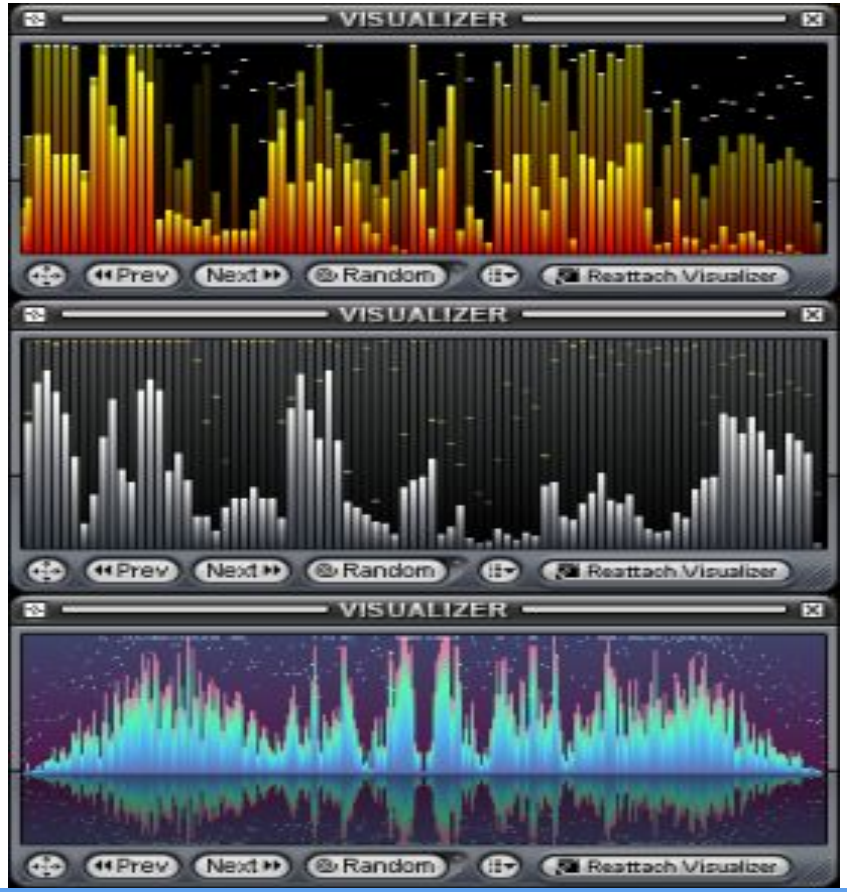
- Improve pygame rendering (24fps)
- Introduce a ring buffer to hold previous captured frames
  - Smoother frame transitions
  - Better normal calculation
- Detect BPM
- Detect key of sample
- Improve visualization...

Viz inspirations





I miss winamp, but I really like Spotify...

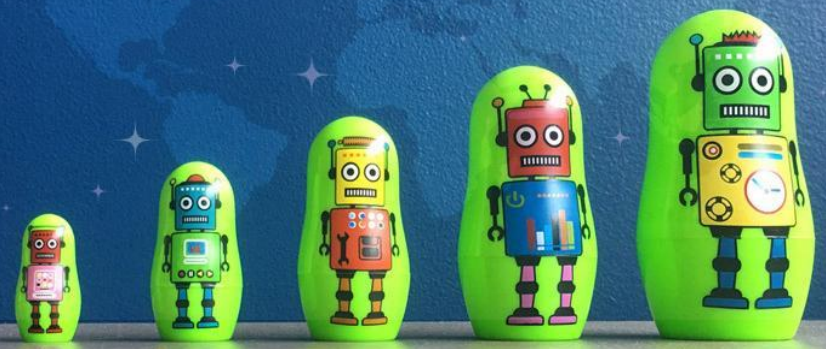


Thank you.

AUTOMATED INSIGHTS®

Brain Clee

www.github.com/cleebp/rta



This project is brought to you by seasons 8&9 of Law and Order: SVU

Executive Producer

DICK WOLF