Investigation of a Python Performance Problem

# Introduction

I’ve been working on a Python based tool to aid the beginning programmer who is blind. The tool provides the blind viewer a facility to aid in the visualization of simple graphics produced by simple python turtle programs. This tool creates tonal sounds, using the pysinewave module which, in turns, is uses python-sounddevice module.

From python-sounddevice, version 0.4.6:

This Python module provides bindings for the PortAudio library and a few convenience functions to play and record NumPy arrays containing audio signals.

My difficulty occurs when I need to create hundreds of stereo sounds. My problem is that the setup process, for 200 stereo sounds, takes minutes, which is very long for an interactive setup. I’m trying remedies, that for the most part minimize the number of tones created. I’m very interested in why so much time is taken, so that I might make the presentation more interactive.

Through the use of the Python profiling module cProfile, I’ve summarized (more details follow) the major time taking process as:

class SineWave \_\_init\_\_() at pysinewave\_master\sinewave.py:13

Calls

class OutputStream(RawOutputStream) \_\_init\_\_() at site-packages\sounddevice.py:1464

Calls

class \_StreamBase: \_\_init\_\_ at site-packages\sounddevice.py:725

During testing execution, StreamBase.\_\_init\_\_() calls take .959 seconds cumulative for the first 40 calls (20 tones x 2 🡪 stereo)

With steadily increasing usage, calls take 5.206 seconds for the final 40 calls (of the total 800).

**Major questions:**

* Why so much time to setup sound objects?
* Why do successive calls keep increasing in time?
* Is there anything we can do to reduce this time?

# Profiling

## Profiling code (profiling additions like this, summarized profiled code like this)

import math

**import time**

**import cProfile, pstats, io # profiling suppo**rt

…

**if self.profile\_running:**

**self.pr = cProfile.Profile()**

**self.pr.enable()**

**begin\_time = time.time()**

… setup code looping for n items

sinewave\_left = SineWave(pitch = pitch,

channels=2, channel\_side='l',

decibels=vol\_left,

pitch\_per\_second=10/self.cell\_time,

decibels\_per\_second=10/self.cell\_time)

sinewave\_right = SineWave(pitch=pitch,

channels=2, channel\_side='r',

decibels=vol\_right,

decibels\_per\_second=10/self.cell\_time)

sp\_ent.sinewave\_left = sinewave\_left

sp\_ent.sinewave\_right = sinewave\_right

end\_time = time.time()

dur\_time = end\_time-begin\_time

**if self.profile\_running:**

**self.pr.disable()**

**s = io.StringIO()**

**sortby = 'cumulative'**

**ps = pstats.Stats(self.pr, stream=s).sort\_stats(sortby)**

**ps.print\_stats()**

**SlTrace.lg(s.getvalue())**

**SlTrace.lg(f"{nitem} cells: time: {dur\_time:.3f} seconds")**

return next\_items

### Profiling Results

Several profile dumps (tops only)

#### 20230323\_161546 Add 20 new items at 20 with sound

20230323\_161547 3083 function calls in 0.963 seconds

Ordered by: cumulative time

ncalls tottime percall **cumtime** percall filename:lineno(function)

40 0.001 0.000 0.961 0.024 C:\Users\raysm\workspace\python\pysinewave\_master\sinewave.py:13(\_\_init\_\_)

40 0.001 0.000 0.960 0.024 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:1464(\_\_init\_\_)

40 0.955 0.024 0.959 0.024 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:725(\_\_init\_\_)

40 0.001 0.000 0.004 0.000 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:2642(\_get\_stream\_parameters)

20 0.001 0.000 0.002 0.000 C:\Users\raysm\workspace\python\resource\_lib\_proj\src\adw\_scanner.py:295(get\_vol)

60 0.000 0.000 0.001 0.000 C:\Users\raysm\workspace\python\resource\_lib\_proj\src\select\_trace.py:320(lg)

...

20230323\_161547 20 cells: time: 0.960 seconds

#### 20230323\_161556 Add 20 new items at 40 with sound

20230323\_161557 3083 function calls in 0.892 seconds

Ordered by: cumulative time

ncalls tottime percall cumtime percall filename:lineno(function)

40 0.000 0.000 0.890 0.022 C:\Users\raysm\workspace\python\pysinewave\_master\sinewave.py:13(\_\_init\_\_)

40 0.000 0.000 0.889 0.022 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:1464(\_\_init\_\_)

40 0.885 0.022 0.889 0.022 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:725(\_\_init\_\_)

40 0.001 0.000 0.004 0.000 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:2642(\_get\_stream\_parameters)

20 0.001 0.000 0.001 0.000 C:\Users\raysm\workspace\python\resource\_lib\_proj\src\adw\_scanner.py:295(get\_vol)

40 0.000 0.000 0.001 0.000 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\numpy\core\\_dtype.py:321(\_name\_get)

60 0.000 0.000 0.001 0.000 C:\Users\raysm\workspace\python\resource\_lib\_proj\src\select\_trace.py:320(lg)

...

20230323\_161557 20 cells: time: 0.883 seconds

...many more till...

#### 20230323\_161911 Add 20 new items at 380 with sound

20230323\_161916 3083 function calls in 5.210 seconds

Ordered by: cumulative time

ncalls tottime percall cumtime percall filename:lineno(function)

40 0.000 0.000 5.208 0.130 C:\Users\raysm\workspace\python\pysinewave\_master\sinewave.py:13(\_\_init\_\_)

40 0.000 0.000 5.207 0.130 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:1464(\_\_init\_\_)

40 5.202 0.130 5.206 0.130 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:725(\_\_init\_\_)

40 0.001 0.000 0.004 0.000 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\sounddevice.py:2642(\_get\_stream\_parameters)

20 0.001 0.000 0.002 0.000 C:\Users\raysm\workspace\python\resource\_lib\_proj\src\adw\_scanner.py:295(get\_vol)

60 0.000 0.000 0.001 0.000 C:\Users\raysm\workspace\python\resource\_lib\_proj\src\select\_trace.py:320(lg)

40 0.000 0.000 0.001 0.000 C:\Users\raysm\AppData\Local\Programs\Python\Python39\lib\site-packages\numpy\core\\_dtype.py:321(\_name\_get)

60 0.000 0.000 0.001 0.000 C:\Users\raysm\workspace\python\resource\_lib\_proj\src\select\_trace.py:926(trace)

...

20230323\_161916 20 cells: time: 5.211 seconds

20230323\_162056 Testing Passed

20230323\_162056 Test End

#### Code for the top offenders

##### sinewave.py:13

class SineWave:

'''Generates and plays a continuous sinewave, with smooth transitions in frequency (pitch)

and amplitude (volume).'''

def \_\_init\_\_(self, pitch=0, pitch\_per\_second=12, decibels=0, decibels\_per\_second=1, channels=1, channel\_side="lr",

samplerate=utilities.DEFAULT\_SAMPLE\_RATE):

self.sinewave\_generator = sinewave\_generator.SineWaveGenerator(

pitch=pitch, pitch\_per\_second=pitch\_per\_second,

decibels = decibels, decibels\_per\_second=decibels\_per\_second,

samplerate=samplerate)

# Create the output stream

self.output\_stream = sd.OutputStream(channels=channels, callback= lambda \*args: self.\_callback(\*args),

samplerate=samplerate)

##### site-packages\sounddevice.py:1464

class OutputStream(RawOutputStream):

"""Stream for output only. See \_\_init\_\_() and Stream."""

def \_\_init\_\_(self, samplerate=None, blocksize=None,

device=None, channels=None, dtype=None, latency=None,

extra\_settings=None, callback=None, finished\_callback=None,

clip\_off=None, dither\_off=None, never\_drop\_input=None,

prime\_output\_buffers\_using\_stream\_callback=None):

"""PortAudio output stream (using NumPy).

This has the same methods and attributes as `Stream`, except

:meth:`~Stream.read` and `read\_available`.

Furthermore, the stream callback is expected to have a different

signature (see below).

##### site-packages\sounddevice.py:725

class \_StreamBase:

"""Direct or indirect base class for all stream classes."""

def \_\_init\_\_(self, kind, samplerate=None, blocksize=None, device=None,

channels=None, dtype=None, latency=None, extra\_settings=None,

callback=None, finished\_callback=None, clip\_off=None,

dither\_off=None, never\_drop\_input=None,

prime\_output\_buffers\_using\_stream\_callback=None,

userdata=None, wrap\_callback=None):

"""Base class for PortAudio streams.

# Code

## **test code main file**

GitHub raysmith619/resource\_lib/src/tb\_test\_scanning\_timing.py

## Support files

### Local Python source files

GitHub raysmith619/resource\_lib/src/

### GitHub <https://github.com/daviddavini/pysinewave> master

## Setup

### Create BASE\_DIR

### Copy GitHub raysmith619/resource\_lib/src/ to BASE\_DIR/resource\_lib/src

### Copy https://github.com/daviddavini/pysinewave master to BASE\_DIR/pysiewave\_master

## Running

### Change to BASE\_DIR/src

### python tb\_test\_scanning\_timing.py

## Test Documentation (this file)

GitHub raysmith619/resource\_lib/Docs/