Digital Signal Processing Lab

Demo 56 - Tkinter and Pyaudio (Slider)

Saad Zubairi
shz2020

October 21, 2025

Solution

In the original file, the clicking clearly happens because the amplitude changes instantaneously when the slider is moved, creating discontinuities. We need to smoothen the amplitude transition so it changes gradually accross blocks.
In the corrected version, we made the following code changes:

- Added a variable to store previous gain, because we need this to compute a smooth transition to the new gain value accross each audio block.

```
    A_prev = gain.get()

```

Snippet 1: Previous gain variable

- Introduced linear interpolation inside the block loop for a smoother transition from the old gain to the new one.

```
    A = A_prev + (A_target - A_prev) * (i / BLOCKLEN)

```

Snippet 2: Linear interpolation

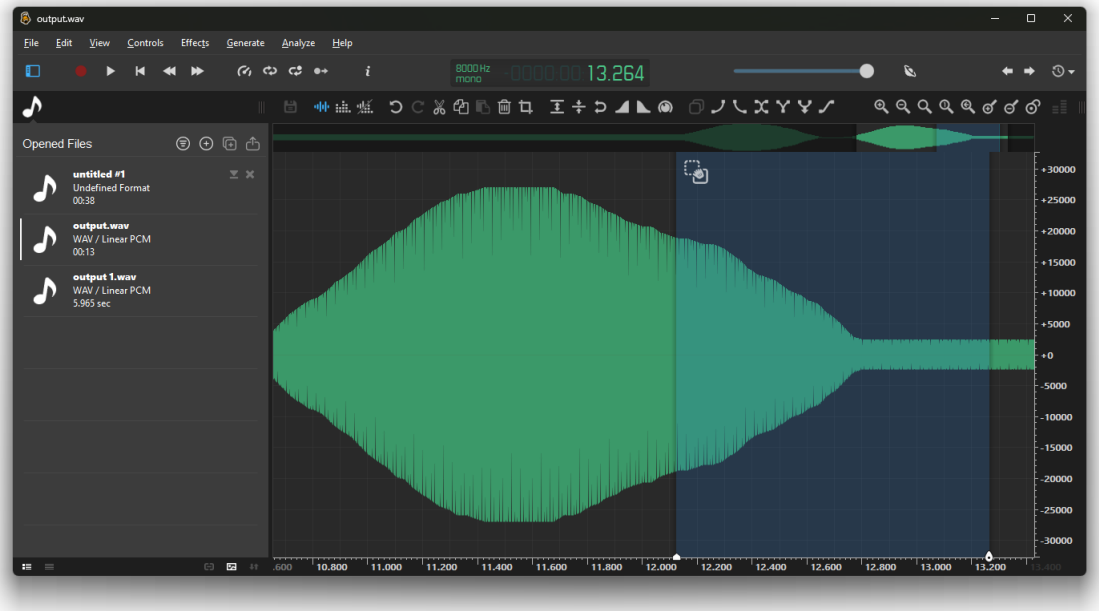
- Updated the gain after each block, i.e. storing the most recent gain so that the next block starts smoothly from the next gain.

```
    A = A_prev + (A_target - A_prev) * (i / BLOCKLEN)

```

Snippet 3: Gain effect updation

These changes ensure that the amplitude transitions continuously, as what can be seen from the waveform:



Addendum

Full implementation given here:

```
from math import cos, pi
import pyaudio, struct
import tkinter as Tk
import wave

RATE = 8000
gain = 0.2 * 2**15

# Create wave file
file_name = 'output.wav'
wf = wave.open(file_name, 'w')
wf.setnchannels(1)
wf.setsampwidth(2)
wf.setframerate(RATE)

def fun_quit():
  global CONTINUE
  print('Good bye')
  CONTINUE = False

# Define Tkinter root
root = Tk.Tk()

# Define Tk variables
f1 = Tk.DoubleVar()
gain = Tk.DoubleVar()

# Initialize Tk variables
f1.set(200)
gain.set(0.2 * 2**15)

# Define widgets
S_freq = Tk.Scale(root, label='Frequency', variable=f1, from_=100, to=400,
    tickinterval=100)
S_gain = Tk.Scale(root, label='Gain', variable=gain, from_=0, to=2**15-1)
B_quit = Tk.Button(root, text='Quit', command=fun_quit)

# Place widgets
B_quit.pack(side=Tk.BOTTOM, fill=Tk.X)
S_freq.pack(side=Tk.LEFT)
S_gain.pack(side=Tk.LEFT)

BLOCKLEN = 256

# Create Pyaudio object
p = pyaudio.PyAudio()
stream = p.open(
  format=pyaudio.paInt16,
  channels=1,
  rate=RATE,
  input=False,
  output=True,
  frames_per_buffer=BLOCKLEN)

output_block = [0] * BLOCKLEN
```

```
theta = 0
CONTINUE = True

# ---- Added: store previous gain for smooth transition ----
A_prev = gain.get()

print('* Start')
while CONTINUE:
  root.update()
  om1 = 2.0 * pi * f1.get() / RATE
  A_target = gain.get()

  # ---- Modified section: interpolate gain smoothly ----
  for i in range(0, BLOCKLEN):
    A = A_prev + (A_target - A_prev) * (i / BLOCKLEN)
    output_block[i] = int(A * cos(theta))
    theta = theta + om1
    if theta > pi:
      theta = theta - 2.0 * pi

  A_prev = A_target  # update previous gain for next block

  binary_data = struct.pack('h' * BLOCKLEN, *output_block)
  stream.write(binary_data)
  wf.writeframes(binary_data)

print('* Finished')

stream.stop_stream()
stream.close()
p.terminate()
```

Snippet 4: Complete implementation