

"""

Supplementary Code 1

Reproduce the supplementary audio for:

"Sonification of 361 Standard Acupoints"

Input:

Supplementary_Data_1_361_acupoint_final_pitch_assignments.csv

Output:

Supplementary_Audio_1_All_361_acupoints.wav

Supplementary_Audio_2_Twelve_primary_meridians.wav

Supplementary_Audio_3_Primary_dysmenorrhea_core_combination.wav

Supplementary_Audio_4_All_361_acupoints_sequential.wav

"""

```
from pathlib import Path
```

```
import csv
```

```
import numpy as np
```

```
from scipy.io.wavfile import write
```

```
HERE = Path(__file__).resolve().parent
```

```
DATA_FILE = HERE /
```

```
"Supplementary_Data_1_361_acupoint_final_pitch_assignments.csv"
```

```
SR = 44100
```

```
DURATION = 5.2
```

```
FADE = 0.06
```

```
def midi_to_hz(midi):
```

```
    return 440.0 * 2.0 ** ((midi - 69.0) / 12.0)
```

```
def point_wave(midi, lateral, duration=DURATION, sr=SR):
```

```
    t = np.arange(int(duration * sr), dtype=float) / sr
```

```
    f = midi_to_hz(midi)
```

```
    x = max(0.0, min(1.0, lateral))
```

```
    # Fixed transverse-position → timbre rule.
```

```
    # Fundamental is unchanged; the 2nd–5th harmonics increase linearly.
```

```

amplitudes = [
    1.0,
    0.05 + 0.30*x,
    0.02 + 0.22*x,
    0.12*x,
    0.08*x,
]

y = sum(
    amp * np.sin(2*np.pi*(harmonic_number+1)*f*t)
    for harmonic_number, amp in enumerate(amplitudes)
)

nfade = max(1, int(FADE*sr))
env = np.ones_like(y)
env[:nfade] = np.linspace(0, 1, nfade)
env[-nfade:] = np.linspace(1, 0, nfade)
return y * env

def peak_normalize(y, peak=0.90):
    maximum = np.max(np.abs(y))
    return y if maximum == 0 else y * (peak/maximum)

def save_wav(path, y, sr=SR):
    y = np.clip(y, -1, 1)
    write(str(path), sr, np.int16(y*32767))

def read_data():
    rows = []
    with DATA_FILE.open(newline="", encoding="utf-8-sig") as f:
        reader = csv.DictReader(f)
        for r in reader:
            rows.append({
                "point": r["Acupoint code"],
                "meridian": r["Meridian code"],
                "midi": float(r["MIDI"]),
                "lateral": float(r["Transverse level"]),
            })

```

```
return rows
```

```
def main():
```

```
    data = read_data()
```

```
    # Supplementary Audio 1: all 361 standard acupoints simultaneously.
```

```
    y = np.zeros(int(DURATION*SR))
```

```
    for p in data:
```

```
        y += point_wave(p["midi"], p["lateral"])
```

```
    save_wav(HERE / "Supplementary_Audio_1_All_361_acupoints.wav",  
            peak_normalize(y))
```

```
    # Supplementary Audio 2: 12 primary meridians in circulation order.
```

```
    order = ["LU", "LI", "ST", "SP", "HT", "SI", "BL", "KI", "PC", "TE", "GB", "LR"]
```

```
    segment_duration = 3.0
```

```
    gap = np.zeros(int(0.45*SR))
```

```
    output = []
```

```
    for i, meridian in enumerate(order):
```

```
        points = [p for p in data if p["meridian"] == meridian]
```

```
        segment = np.zeros(int(segment_duration*SR))
```

```
        for p in points:
```

```
            segment += point_wave(p["midi"], p["lateral"],  
                                  duration=segment_duration)
```

```
        # Segment normalization is for listening audibility only.
```

```
        # Loudness is not treated as an analytic variable.
```

```
        output.append(peak_normalize(segment, 0.86))
```

```
        if i < len(order)-1:
```

```
            output.append(gap)
```

```
    save_wav(HERE / "Supplementary_Audio_2_Twelve_primary_meridians.wav",  
            np.concatenate(output))
```

```
    # Supplementary Audio 3: SP6 → CV4 → SP8 → all three simultaneously.
```

```
    lookup = {p["point"]: p for p in data}
```

```
    core = ["SP6", "CV4", "SP8"]
```

```
    segment_duration = 2.4
```

```
    output = []
```

```
    for code in core:
```

```
        p = lookup[code]
```

```

    output.append(peak_normalize(
        point_wave(p["midi"], p["lateral"], duration=segment_duration),
        0.72
    ))
    output.append(gap)

combination = np.zeros(int(segment_duration*SR))
for code in core:
    p = lookup[code]
    combination += point_wave(
        p["midi"], p["lateral"], duration=segment_duration
    )
output.append(peak_normalize(combination, 0.86))
save_wav(
    HERE /
"Supplementary_Audio_3_Primary_dysmenorrhea_core_combination.wav",
    np.concatenate(output)
)

```

```

# Supplementary Audio 4: all 361 standard acupoints, one by one,
# in the order listed in Supplementary Data 1.
point_duration = 0.55
point_gap = np.zeros(int(0.08*SR))
sequential = []
for i, p in enumerate(data):
    single = peak_normalize(
        point_wave(p["midi"], p["lateral"], duration=point_duration),
        0.82
    )
    sequential.append(single)
    if i < len(data)-1:
        sequential.append(point_gap)
save_wav(
    HERE / "Supplementary_Audio_4_All_361_acupoints_sequential.wav",
    np.concatenate(sequential)
)

```

```
if __name__ == "__main__":  
    main()
```