SD5913 · WEEK 06

Sound as data

Noise, waveforms, speech, songs and interaction.

AI NEWS · 13 OCT 2026 · GOOGLE

Gemini 4 Argon

Complex coding and spatial reasoning; Google shows Blender / Three.js demos.

Released: paid API, Gemini Pro / Ultra; Antigravity access for Ultra.

Vendor demos and performance claims; not an independent class test.

What tests would prove the generated 3D scene works?

Announcement + 3D demos

Haiku 5.5: choose the job

Small model with adjustable effort for narrow, repeated agent tasks.

Released on Claude Platform, AWS, Google Cloud and Azure.

Larger Claude models remain stronger for complex coding in vendor tests.

Which task needs a bigger model, and which needs a clear test?

Release + evaluation conditions

An answer can be an interface

GPT-6 Intelligent UI creates interactive diagrams, forms and small tools.

Chat rollout: paid plans from 7 October; Free / Go from 8 October.

Enterprise access depends on admin settings. Check the generated behaviour.

Ask for frequency / amplitude sliders. How would you test them?

Release + availability

Motion you can edit

Claude Motion animates text, charts, shapes and images as editable code.

MP4 export; Team / Enterprise beta. Enterprise admins must enable it.

Code-based animation, not a video-generation model producing footage.

Which animation properties should remain explicit in code?

Product announcement + demo

Games become remix material

Windows launcher for mods and mashups of games you already own.

Give Claude Code or Codex its instructions to build and publish a mashup.

Free download; compatibility and safety claims need checking per mod.

Which behaviours would you test before sharing a game mod?

Melty: examples + requirements

If agents can inspect software…

REA connects agents to app / binary analysis through MCP and a CLI.

MIT-licensed tool; 6.4 adds Android inspection and analysis improvements.

Evidence can support reimplementation; inferred behaviour still needs tests.

What is the future of software companies if anyone can create and modify software?

REA: source, examples + limitations

What software would you make open source?

Today

01 Sound: vibration, samples and WAV files

02 Synthesis: frequency, amplitude and harmonics

03 Music: a grid and four Strudel layers

04 After the break: song API demo

05 Speech: check a transcript; hear a local reply

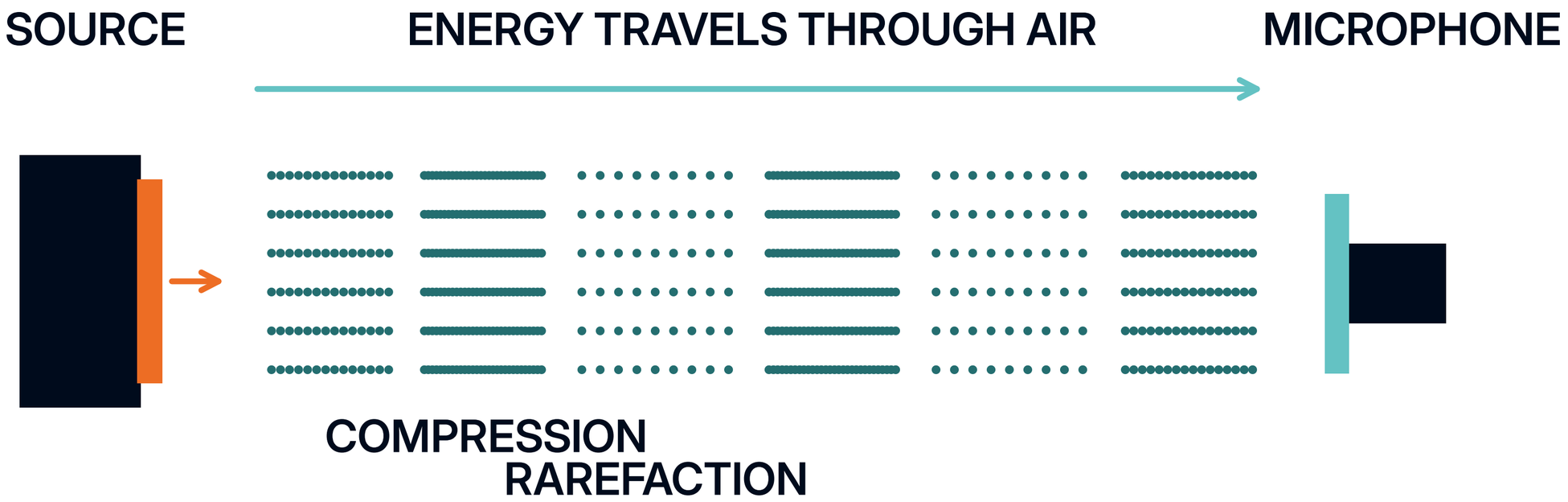
06 Interaction, Canvas mock and project brief

01 · SOUND

Sound becomes data

VIBRATION, SAMPLES AND A FINITE WAV

A vibration travels through air

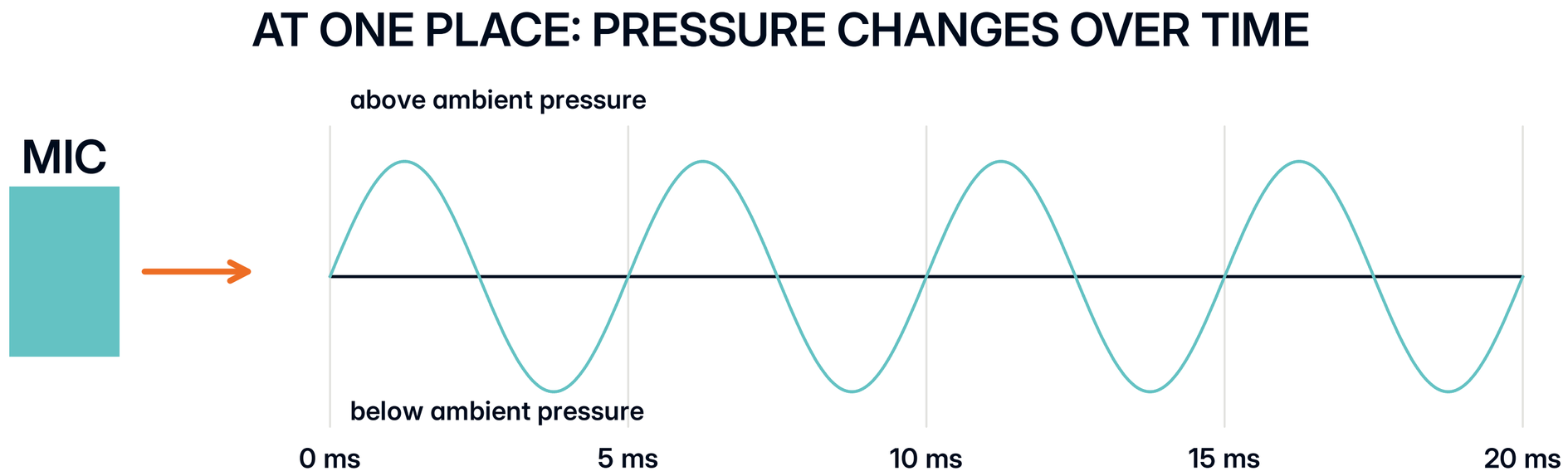


Air moves back and forth locally; it does not travel with the sound.

Schematic particle spacing, not a picture of individual molecules.

The source moves air locally; alternating compressions carry energy toward a microphone.

A microphone measures change at one place



Microphone > changing signal > ordered samples

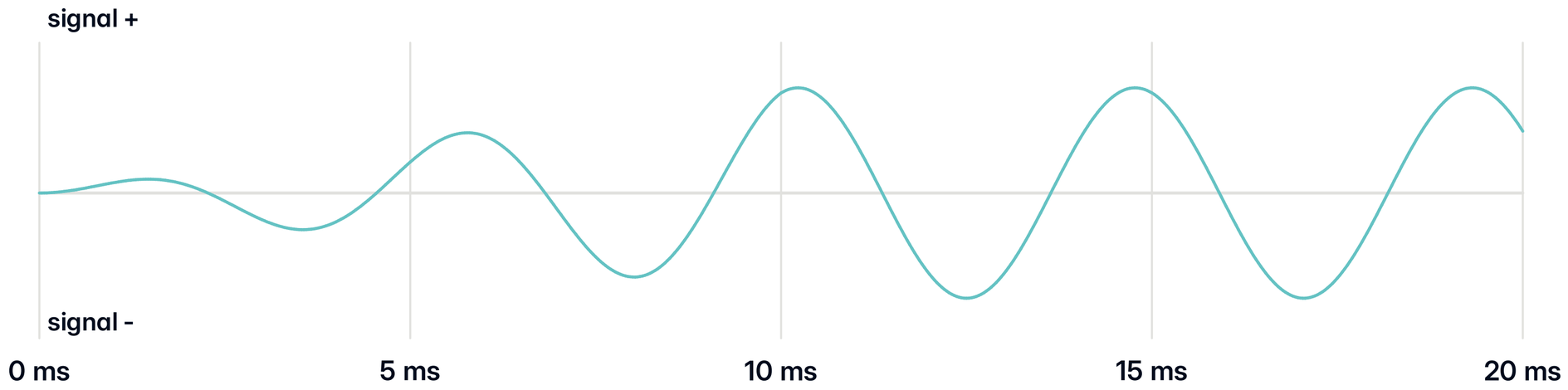
Illustrative 200 Hz pressure trace, not a microphone recording.

Horizontal: time. Vertical: pressure change around ambient. Not a path through air.

Hear the rule. See its samples.

TONE: 220 CYCLES / SECOND

FIRST 20 ms (INCLUDES 10 ms FADE-IN)

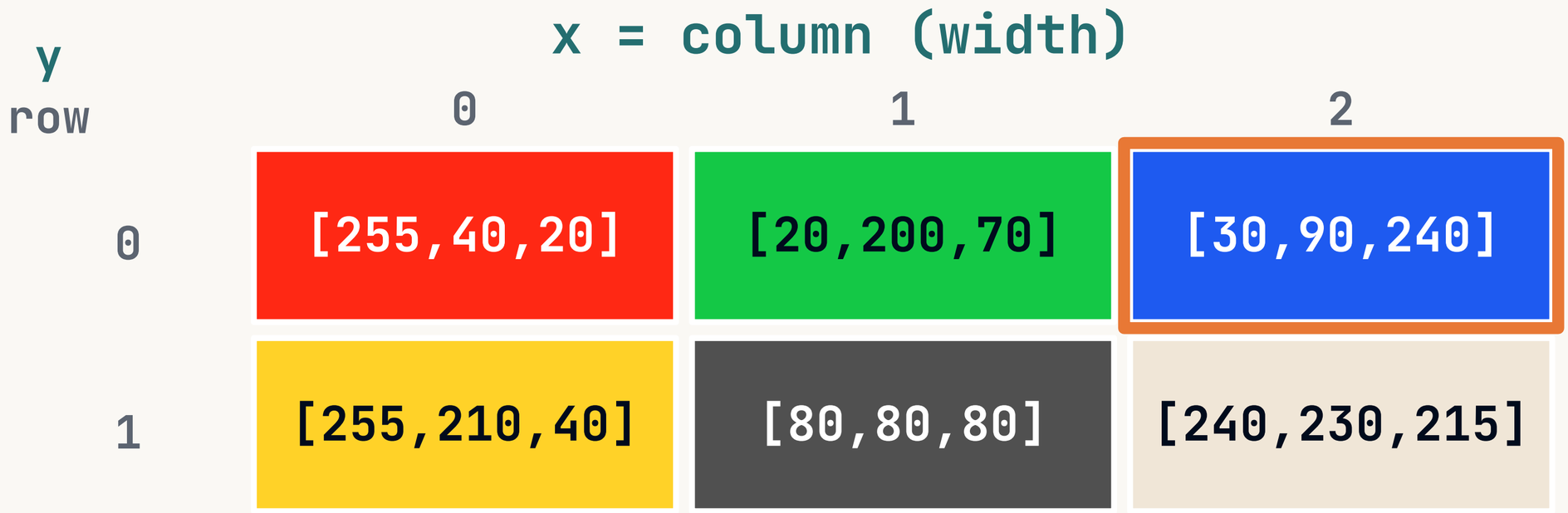


22,050 samples/s / 44,100 samples / 2 seconds

HTML: select tone or seeded noise, then PLAY 2 s or STOP.

HTML: click PLAY for a finite two-second tone or noise. No install, key or microphone.

An image gives values positions in space




```
pixel[y][x] = [red, green, blue]
```

A sound gives values positions in time

A mono sound can use a one-dimensional array.

Each number describes the signal at one instant.

The index tells us which sample comes next.

Zero is the centre of this signed signal.

```
samples = [
    0.00, 0.04, 0.08,
    0.04, 0.00, -0.04,
    -0.08, -0.04
]

print(samples[2])  # 0.08
```

Random values can become a signal

Week 5 used random RGB values for pixels.

Here each random value becomes a sample.

A seed lets us repeat this procedure.

```
import random

rng = random.Random(7)
samples = [
    rng.uniform(-0.08, 0.08)
    for _ in range(12)
]

print(samples)
```

A sample rate gives the sequence a duration

22,050 samples per second.

Two seconds need 44,100 mono samples.

Duration = sample count / sample rate.

```
sample_rate = 22050
seconds = 2

count = sample_rate * seconds
print(count)  # 44100

# one sample per time step
time = index / sample_rate
```

Stretch time without changing the samples

Eight dots show eight fixed sample values. Run once at eight samples/s.

Change only rate to 4. Predict the duration before you run again.

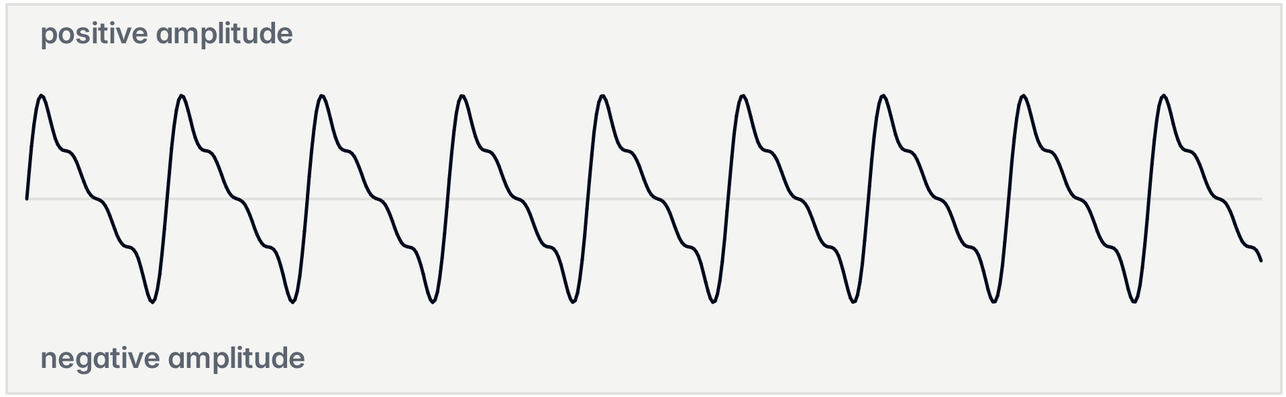
The values and their order stay fixed; only the time scale changes. No sound plays.

YOUR TURN

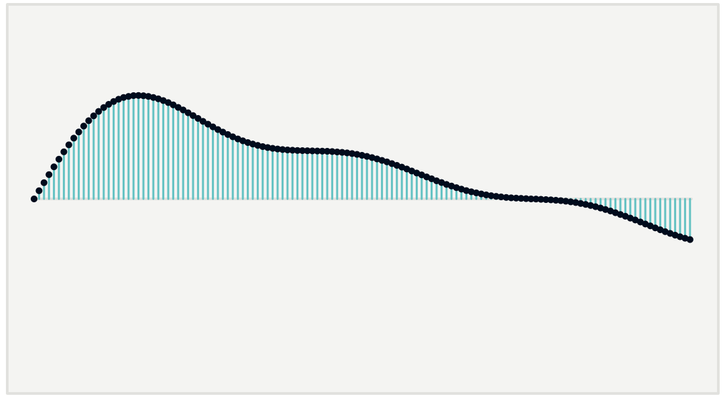
```
samples = [0, 1, 3, 2, 0, -1, -3, -2]
rate = 8
dots = [(int(i/rate*360)+20, 120-v*25)
        for i, v in enumerate(samples)]
print('<svg viewBox="0 0 800 220" width="800">')
for x, y in dots:
    print(f'<circle cx="{x}" cy="{y}" r="8"/>')
print(f'<text x="20" y="200">{len(dots)/rate:g}s</text>')
print('</svg>')
```

The waveform shows signal amplitude over time

Continuous waveform: 40 ms



Sampled waveform: 3 ms



133 samples at 44,100 samples/s

220 Hz fundamental + harmonics: about nine cycles in 40 ms.

Vertical axis = signal amplitude, not perceptual loudness.

This illustration uses 44,100 samples/s. The WAV demos use 22,050 samples/s.

Two seconds of noise

```
import random
from wav_support import fade_edges, save_wav
rate, seconds, amplitude = 22050, 2, 0.08
rng = random.Random(7)
samples = [rng.uniform(-amplitude, amplitude)
           for _ in range(rate * seconds)]
samples = fade_edges(samples, rate)
save_wav("out/noise.wav", samples, rate)
```

Download from lesson resources, then run: python noise.py

Samples inside a WAV file

```
pcm = b"".join(struct.pack("<h", round(x * 32767))
               for x in samples)
with wave.open("out/noise.wav", "wb") as audio:
    audio.setnchannels(1)   # mono
    audio.setsampwidth(2)  # 2 bytes = 16 bits
    audio.setframerate(22050)
    audio.writeframes(pcm)
```

The complete helper imports struct and wave, and checks the values first.

02 · SYNTHESIS

Shape a sound

FREQUENCY, AMPLITUDE AND HARMONICS

A sine wave repeats smoothly

Frequency sets cycles per second.

Amplitude scales the signal values.

The sample rate sets how often we take a value.

```
import math

time = i / sample_rate
value = amplitude * math.sin(
    2 * math.pi * frequency * time
)
```

A two-second tone

```
import math
from wav_support import fade_edges, save_wav
rate, seconds, amplitude, frequency = 22050, 2, 0.08, 220
samples = [amplitude * math.sin(2 * math.pi * frequency * i / rate)
           for i in range(rate * seconds)]
samples = fade_edges(samples, rate)
save_wav("out/tone.wav", samples, rate)
```

Download from lesson resources, then run: python tone.py

Frequency and sample rate

Keep SAMPLE_RATE at 22050.

Change FREQUENCY from 220 to 440.

Predict what changes and what stays the same.

```
# sample clock
SAMPLE_RATE = 22050

# repeating waveform
FREQUENCY = 440

# unchanged duration
SECONDS = 2
```

Frequency 220 to 440 Hz: what stays the same?

- **A** Duration and sample count
- **B** Number of cycles
- **C** Pitch
- **D** Waveform spacing

Play a small keyboard piano

PLAY ONE NOTE. NAME ITS FREQUENCY.

Click a key or press 1 2 3 4 5 6 7 8 in the HTML slide.

C4	D4	E4	F4	G4	A4	B4	C5
1	2	3	4	5	6	7	8

C4 = 261.63 Hz / C5 = 523.25 Hz: twice the frequency.

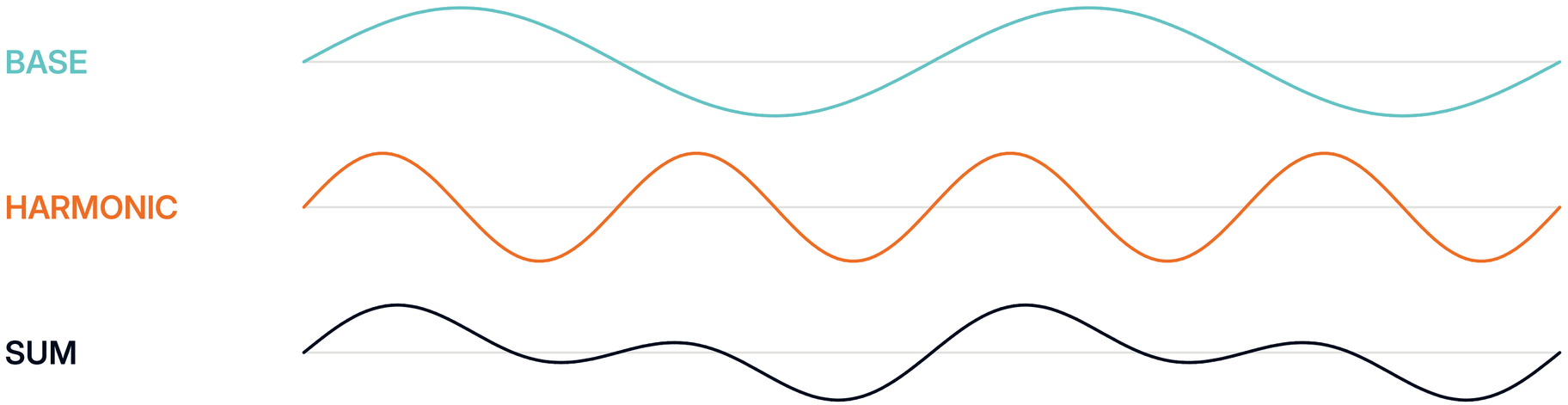
HTML: press a key to hear a short note. Escape or STOP silences it.

Try C4 then C5. Can you hear and see the frequency double?

Build harmonics from a base frequency

BASE 220 Hz x 2 = 440 Hz

Change the multiplier; hear each wave, then hear the sum.



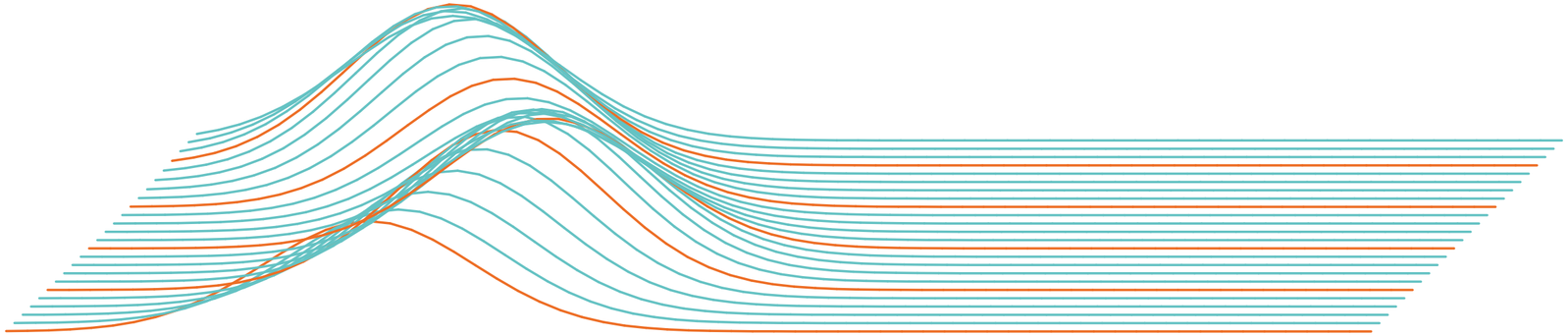
HTML: adjust base and multiple. The sample clock stays fixed.

Keep 220 Hz fixed; try 2× then 3×. Predict the new frequency and shape.

Speak and see a flowing spectrum

MICROPHONE > FREQUENCIES > FLOWING SURFACE

A synthetic preview. The HTML asks permission before hearing a voice.



frequency across

time flows away

height = energy

HTML: click MIC or try DEMO. Nothing is recorded or uploaded.

Across: frequency. Into the scene: time. Height: relative energy. Nothing is recorded.

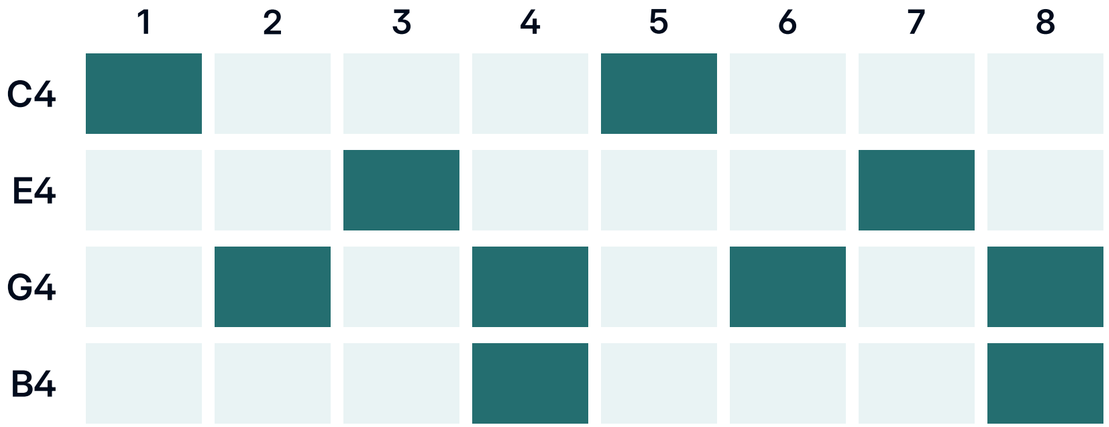
03 · MUSIC

Make a musical pattern

A BROWSER GRID AND FOUR STRUDEL LAYERS

A click changes the score

A written score: four sine voices, eight steps



100 BPM: two steps per beat; one step = 0.3 s
Eight steps = 2.4 s (four beats); read, then repeat.

Predict a cell edit, test the next loop, then show your partner how to Stop.

03 · MUSIC · LIVE CODING

Hear your code

HI-HAT → BASS → CHORDS → MELODY

Begin with a working example

Open strudel.cc

Play the existing starter or an example selected by the instructor.

Change one value. Update. What did you hear?

Stop before returning to the slides.

Four layers share one cycle

Hi-hat marks the pulse. **Bass** gives a low foundation.

Chords put notes together. **Melody** moves through notes.

*8 repeats; ~ is a rest; commas put notes together.

setcpm(100/4) means 25 cycles/minute: four beats at 100 BPM.

Hi-hat

```
setcpm(100/4)
stack(
  s("hh*8").gain(.12),
)
```

Change hh*8 to hh*4. Predict the difference before Update.

Open this complete stage in Strudel

Add bass

```
setcpm(100/4)
stack(
  s("hh*8").gain(.12),
  note("c2 ~ g2 ~").s("triangle").gain(.15),
)
```

Keep the hats. Move c2 and g2 up to c3 and g3.

Open this complete stage in Strudel

Add chords

```
setcpm(100/4)
stack(
  s("hh*8").gain(.12),
  note("c2 ~ g2 ~").s("triangle").gain(.15),
  note("[c3,e3,g3]").s("sine").gain(.08),
)
```

Keep the first two layers. Change e3 to eb3.

Open this complete stage in Strudel

Add melody

```
setcpm(100/4)
stack(
  s("hh*8").gain(.12),
  note("c2 ~ g2 ~").s("triangle").gain(.15),
  note("[c3,e3,g3]").s("sine").gain(.08),
  note("c4 e4 g4 b4").s("sine").gain(.12),
)
```

Keep the other layers. Replace one melody note with ~.

Open this complete stage in Strudel

10 MINUTES

Break

STOP THE AUDIO. API DEMO NEXT.

From an image request to a song

Describe the sound you want.

Check the request, then follow its status.

Save the tracks and their settings.

```
# Week 5
POST /v1/images/generations
image response

# Week 6
POST /v1/audio/generations
attempt status, then tracks
```

A request describes the intended sound

```
{
  "model": "suno-music",
  "prompt": "Gentle ambient piano, warm strings, slow pulse",
  "title": "Evening", "make_instrumental": true,
  "duration_seconds": 30,
  "weirdness": 25, "style_influence": 70
}
```

Easel extension: POST /v1/audio/generations

Submitted is different from ready

Prepare the request.

Submit once; keep its attempt ID.

Check status, then follow the track IDs.

Download the ready tracks.

The default is an offline request preview

```
uv run --no-project request_song.py

# Preview a different idea:
uv run --no-project request_song.py
  --prompt "Bright synths, playful rhythm"
  --title "First steps" --instrumental
```

From PFAD week06/sound. Join the last three lines. Both commands preview JSON.

Which parts of the brief are precise?

Compare “30 seconds” with “a gentle sense of arrival”.

Which can you measure? Which needs a listener’s judgment?

Change one field. Predict its effect.

05 · SPEECH

Talk, check, reply

A TRANSCRIPT, A LOCAL MODEL AND A SPOKEN RESPONSE

Speech to text. Text to speech.

Speech-to-text proposes what was said.

Check and correct the words before responding.

Text-to-speech gives the reply a voice.

Open the local speech exercise

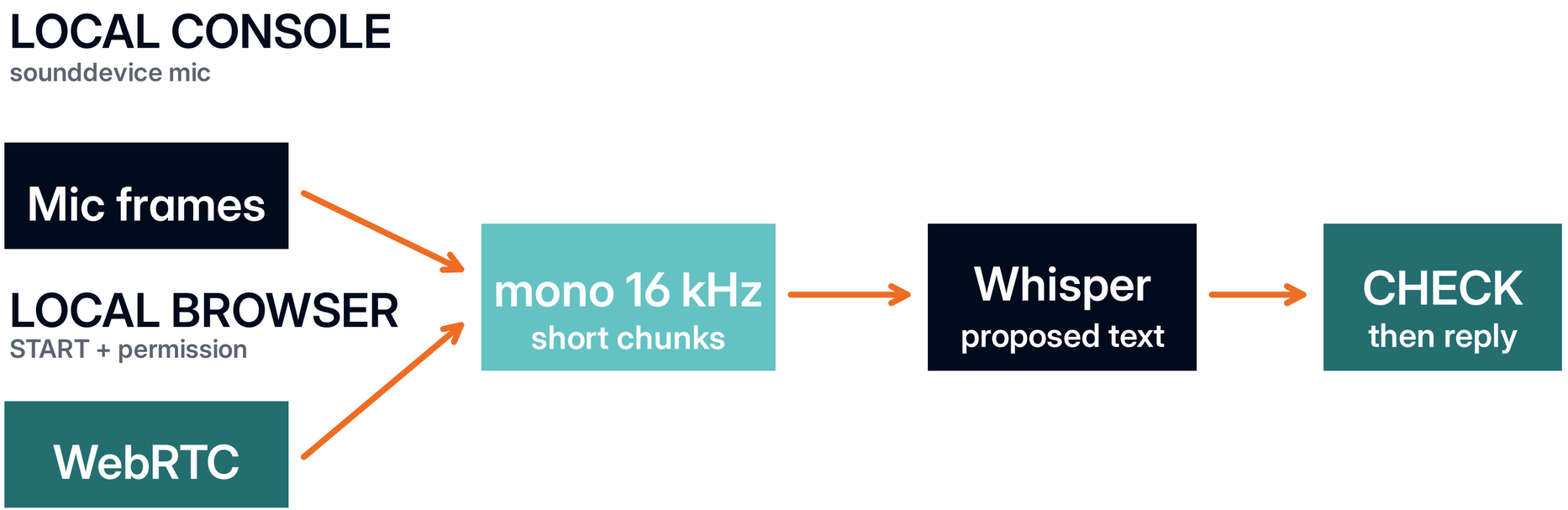
Speak into Python. Check each chunk.

```
# From PFAD week06/speech, after console setup:
uv run --no-project --python .venv/Scripts/python.exe transcribe.py
  --microphone --chunks 3 --seconds 5 --out ../out/mic.txt

# No device? Read a consented local WAV:
# --audio ../out/onnx-hello.wav --offline
# macOS/Linux: use .venv/bin/python instead.
```

Join the command into one line. Three 5-second chunks; Ctrl+C stops early.

One recognizer, two microphone paths



Capture moves to the browser. A person still corrects the transcript.

WebRTC transports browser audio; it is not a different speech recognizer.

The same transcript moves into the browser

```
# From PFAD week06/speech, after portable setup:
uv run --no-project --python .venv/Scripts/python.exe
  python -m streamlit run speech_webrtc.py
  --server.address 127.0.0.1 --browser.gatherUsageStats false

# macOS/Linux: use .venv/bin/python instead.
# Open localhost:8501; Start mic / Stop mic.
```

Join the command into one line. Full portable setup: speech.html.

A response begins with a checked transcript

Start mic: words appear in the chat box.

Stop mic, then edit any recognition errors.

Press Enter to send the message.

LM Studio replies; Kokoro ONNX speaks.

Kokoro saves a spoken cue

```
# From PFAD week06/speech, after portable setup:
uv run --no-project --python .venv/Scripts/python.exe
  synthesise_onnx.py --text "Your file is ready."
  --speed 1.0 --out ../out/cue.wav

# Keep the first file. Try a new speed:
# --speed 0.85 --out ../out/cue-slower.wav
# macOS/Linux: use .venv/bin/python instead.
```

Join the command into one line. Kokoro ONNX: CPU default, 24 kHz output.

06 · INTERACTION

Design the response

REFLECT, PRACTISE AND PLAN YOUR PROJECT

REFLECTION

Which choice can you explain?

Name one input and the data it changes.

Name one choice you left to a rule, an agent or a model.

What evidence would make you revise the sound?

PRACTICE · 10 MINUTES

Try the existing mock

Open the mock test on Canvas.

Try a few questions with your course notes beside you.

Trace one answer; explain why it works.

Choose one concept to revisit before 22 October.

LINKED RESOURCES

Keep the code, not just the sound

Open the lesson resources

Four cumulative Strudel scores and links to the editor.

Python WAV, local speech models and the song-request preview.

Mock test: on Canvas.

ASSESSMENT · 22 OCTOBER 2026

Quiz · Thursday 22 October

40 questions · 30 minutes.

Open-book, multiple choice.

Practise with the mock test on Canvas.

ASSIGNMENT 3 · INDIVIDUAL · 15%

An interactive experience

A user input and program response should form a loop.

A website, survey or command-line program can work.

Due Sunday 1 November, 23:59 Hong Kong time.

Submit your repository URL on Canvas; include README and PROCESS.

What should the listener notice?

A sound is data, a design choice and part of an interaction.

SD5913.GITHUB.IO/TEACHING