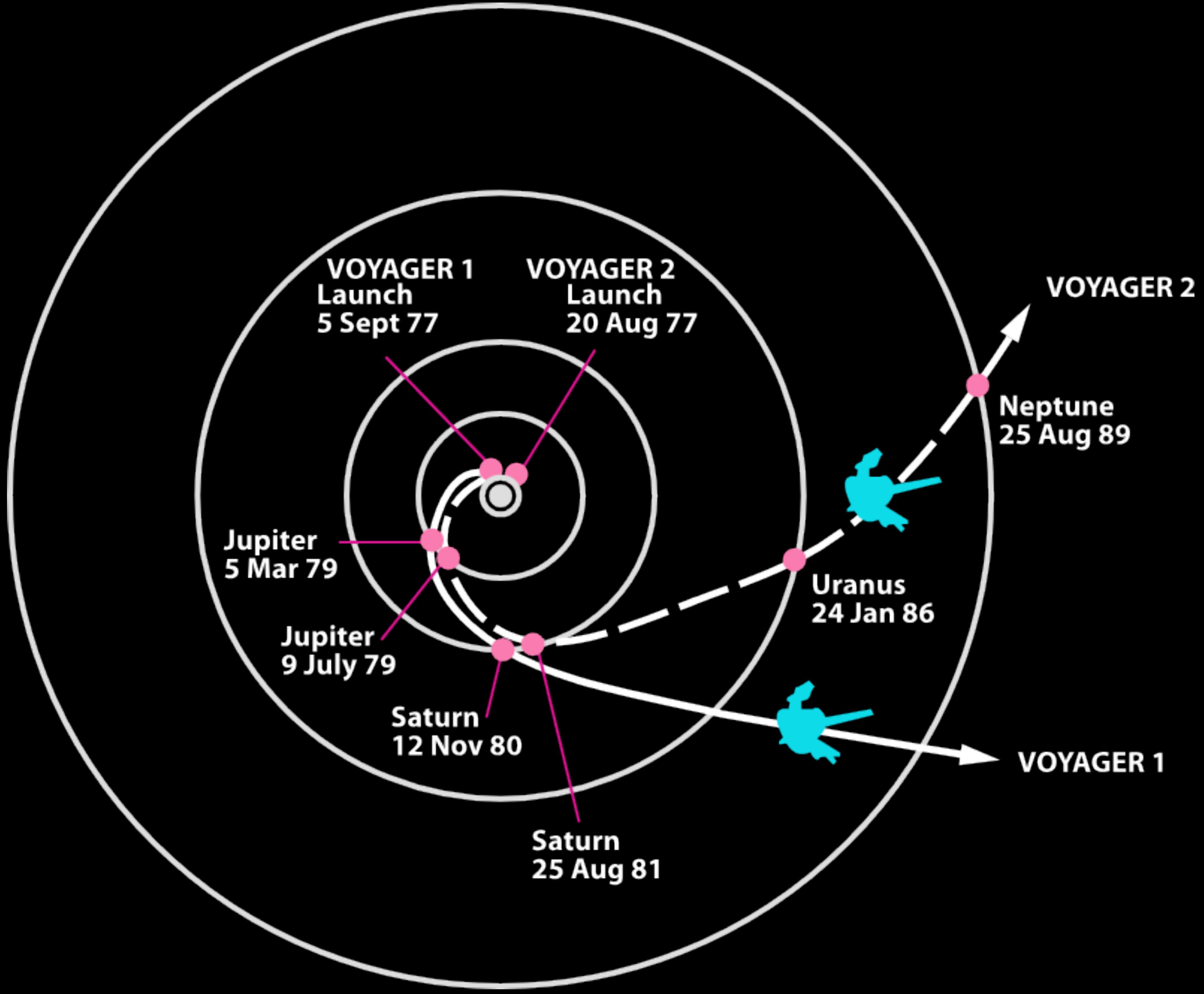
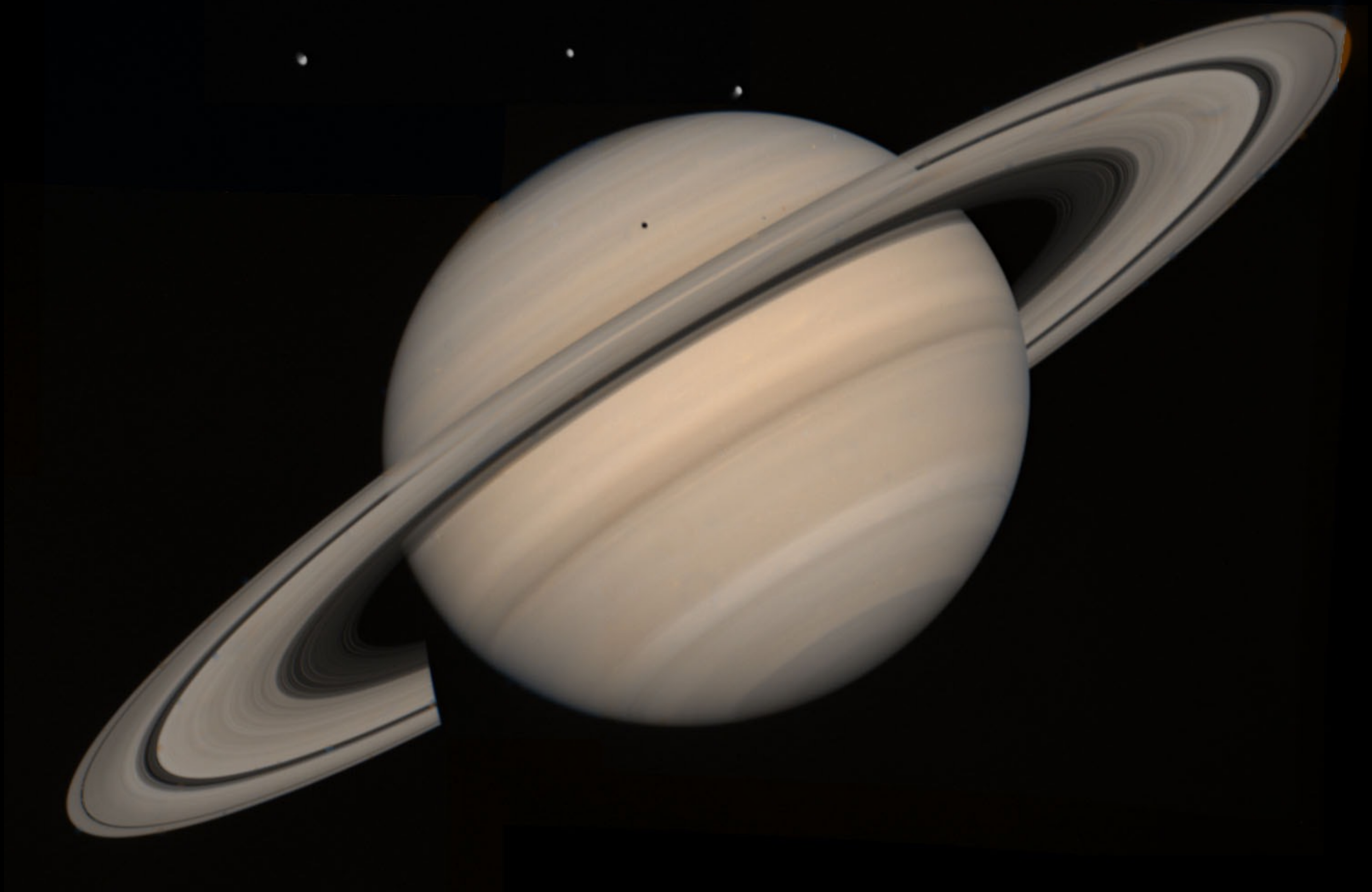
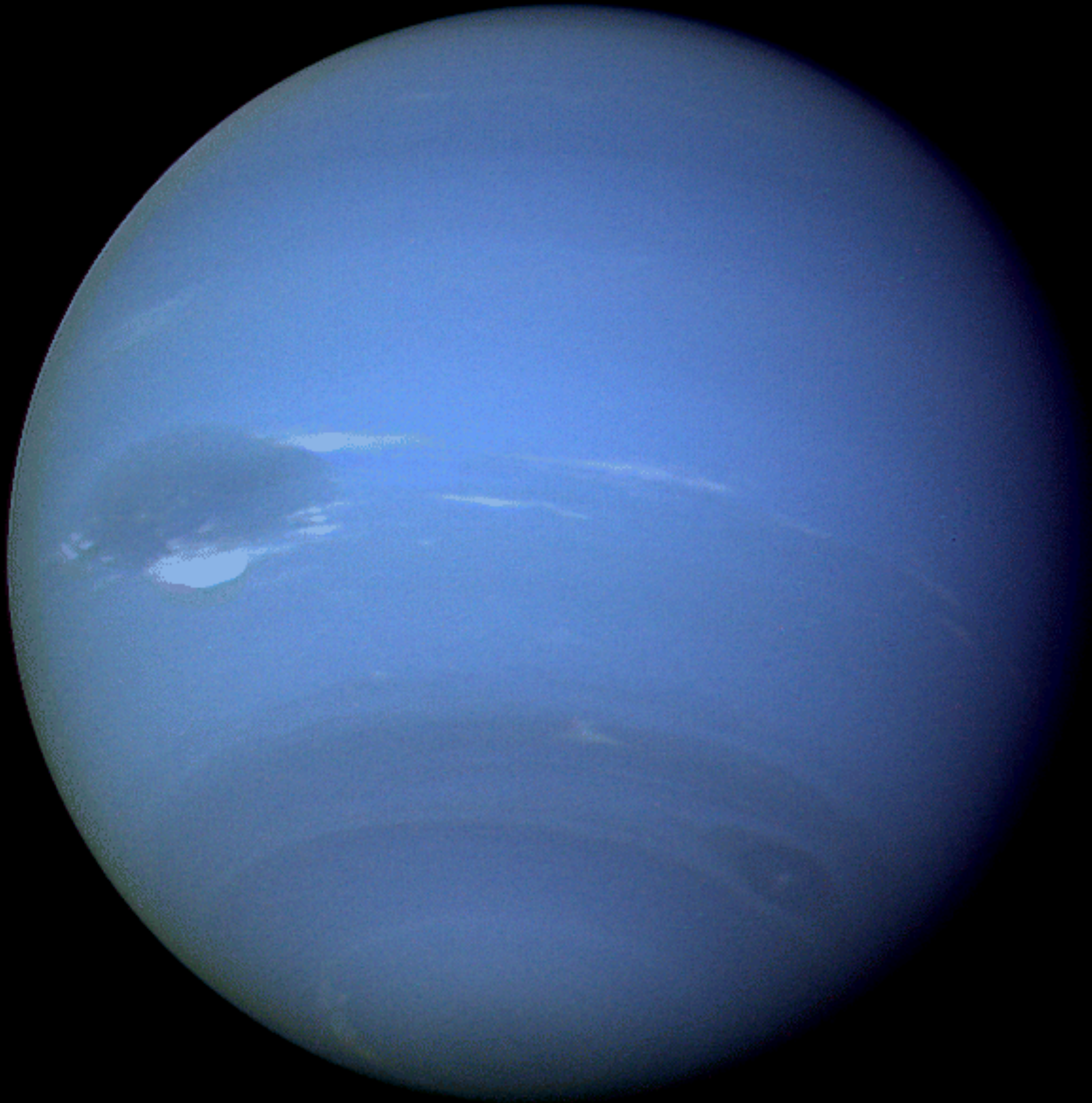
Rob Howard

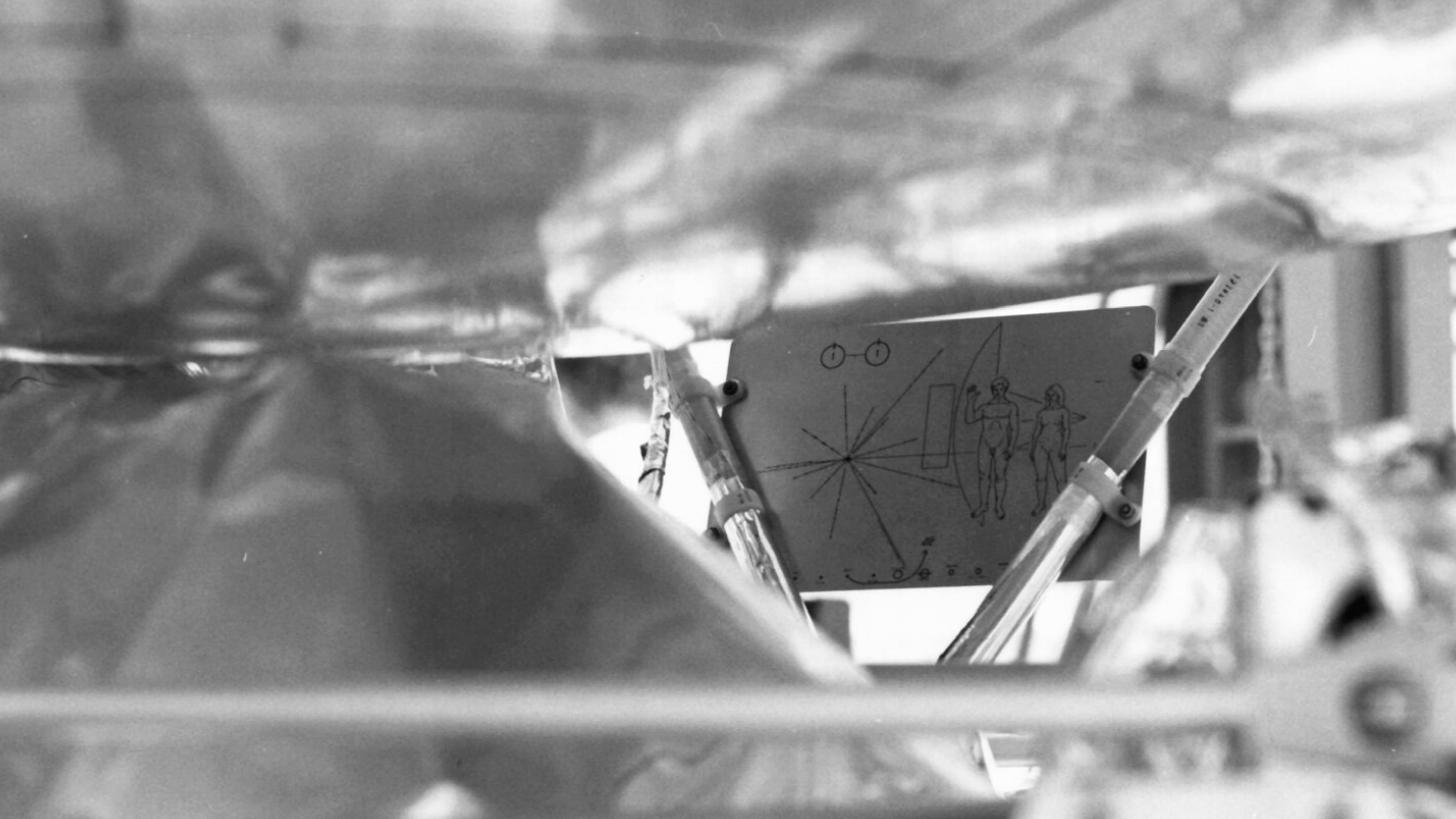
A Message to the Stars

RUBYCONF AU 2020

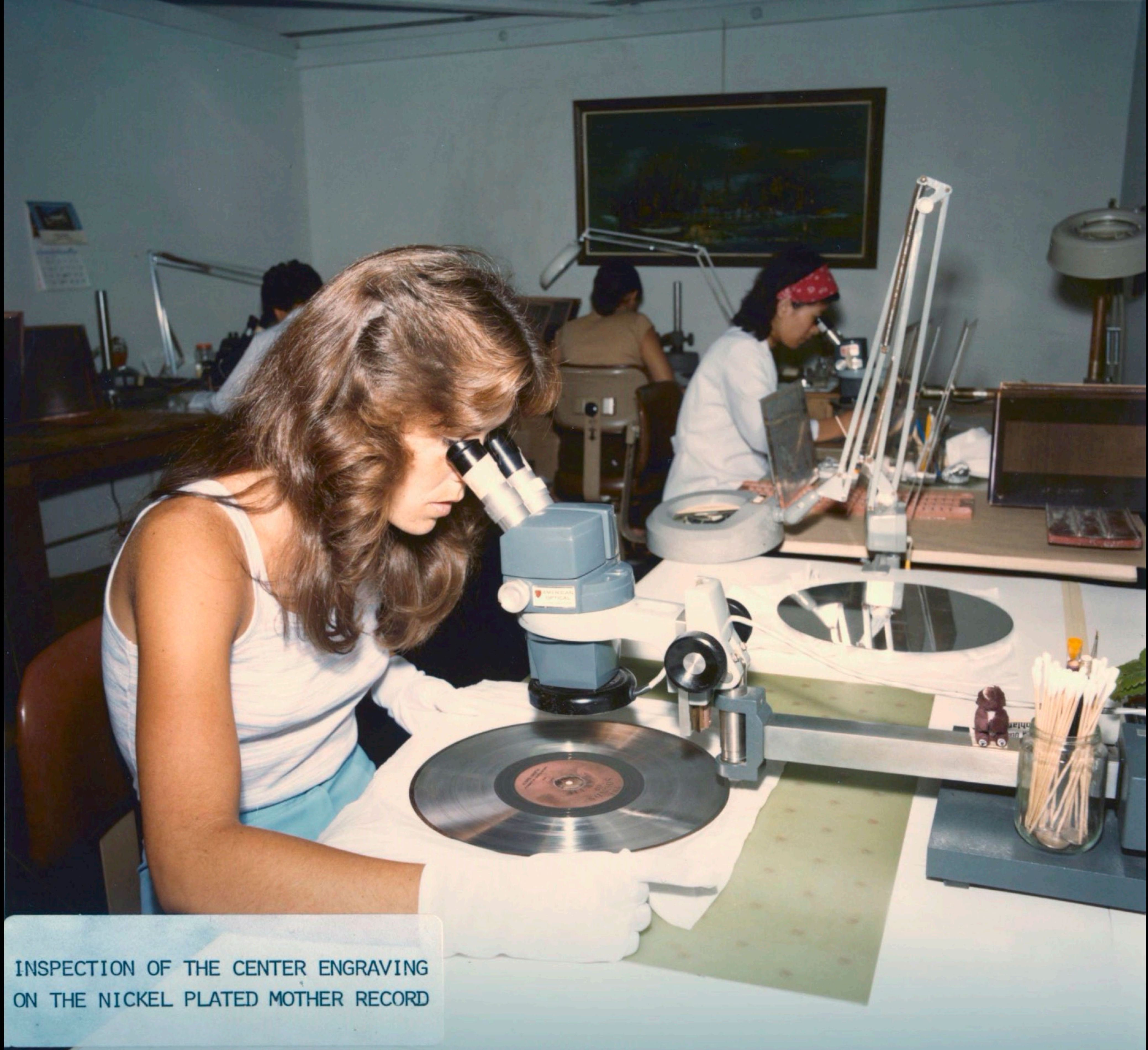

VOYAGER 1
Launch
5 Sept 77
VOYAGER 2
Launch
20 Aug 77
VOYAGER 2
Neptune
25 Aug 89
Jupiter
5 Mar 79
Uranus
24 Jan 86
Jupiter
9 July 79
Saturn
12 Nov 80
VOYAGER 1
Saturn
25 Aug 81



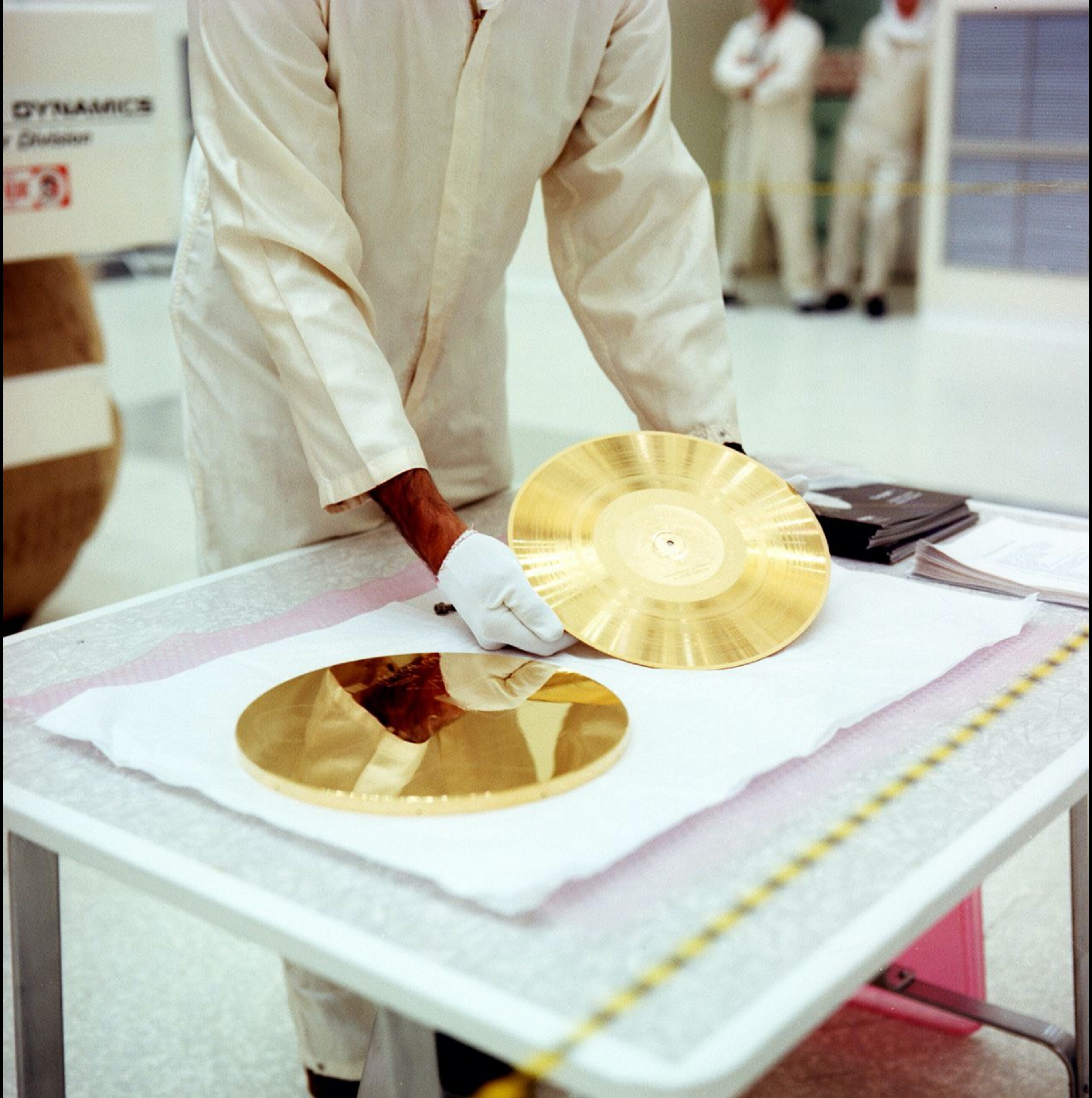


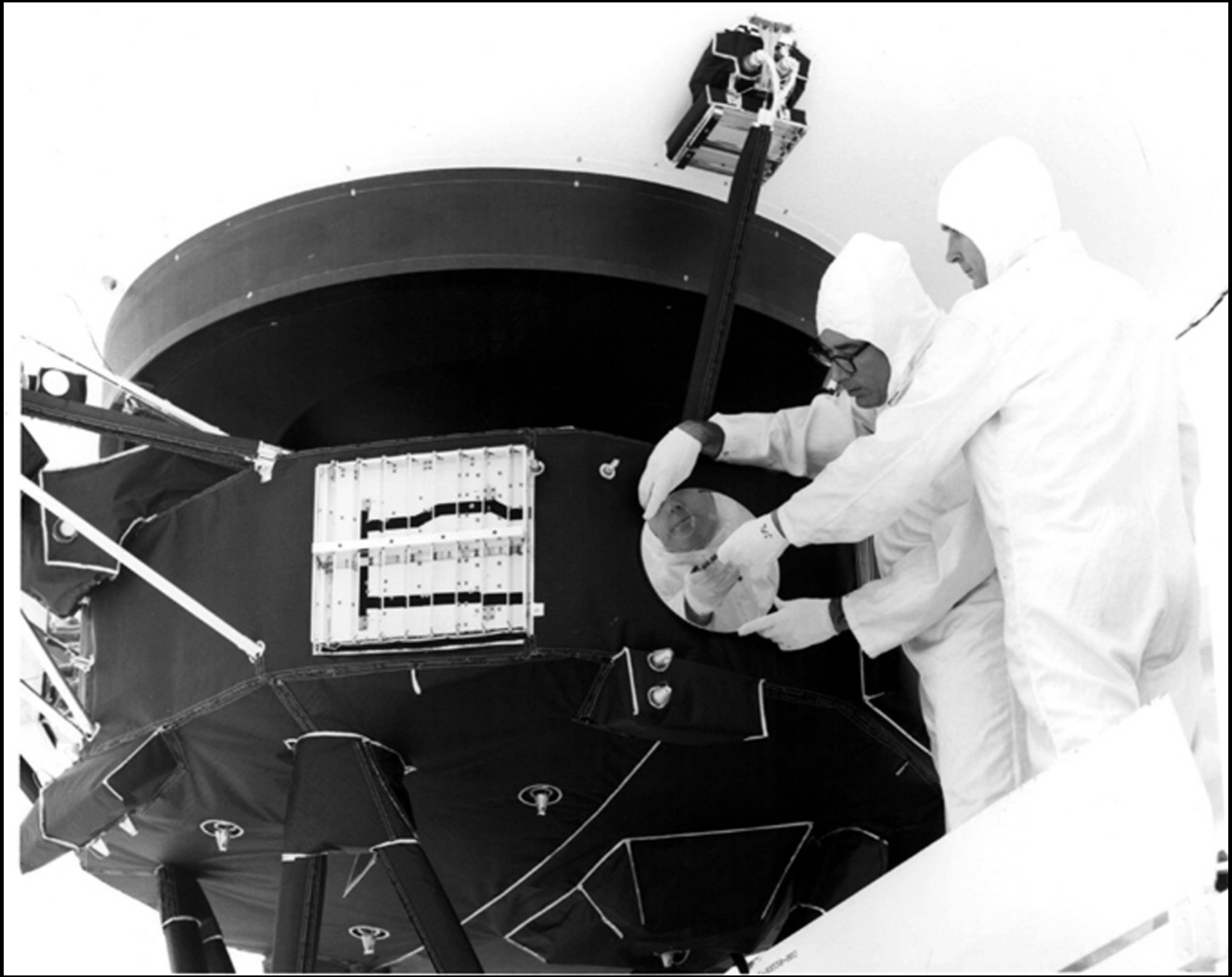


THE SOUNDS OF EARTH
Side 1
NASA
UNITED STATES OF AMERICA
PLANET EARTH
TO THE MAKERS OF MUSIC - ALL WORLDS, ALL TIMES
SIDE A

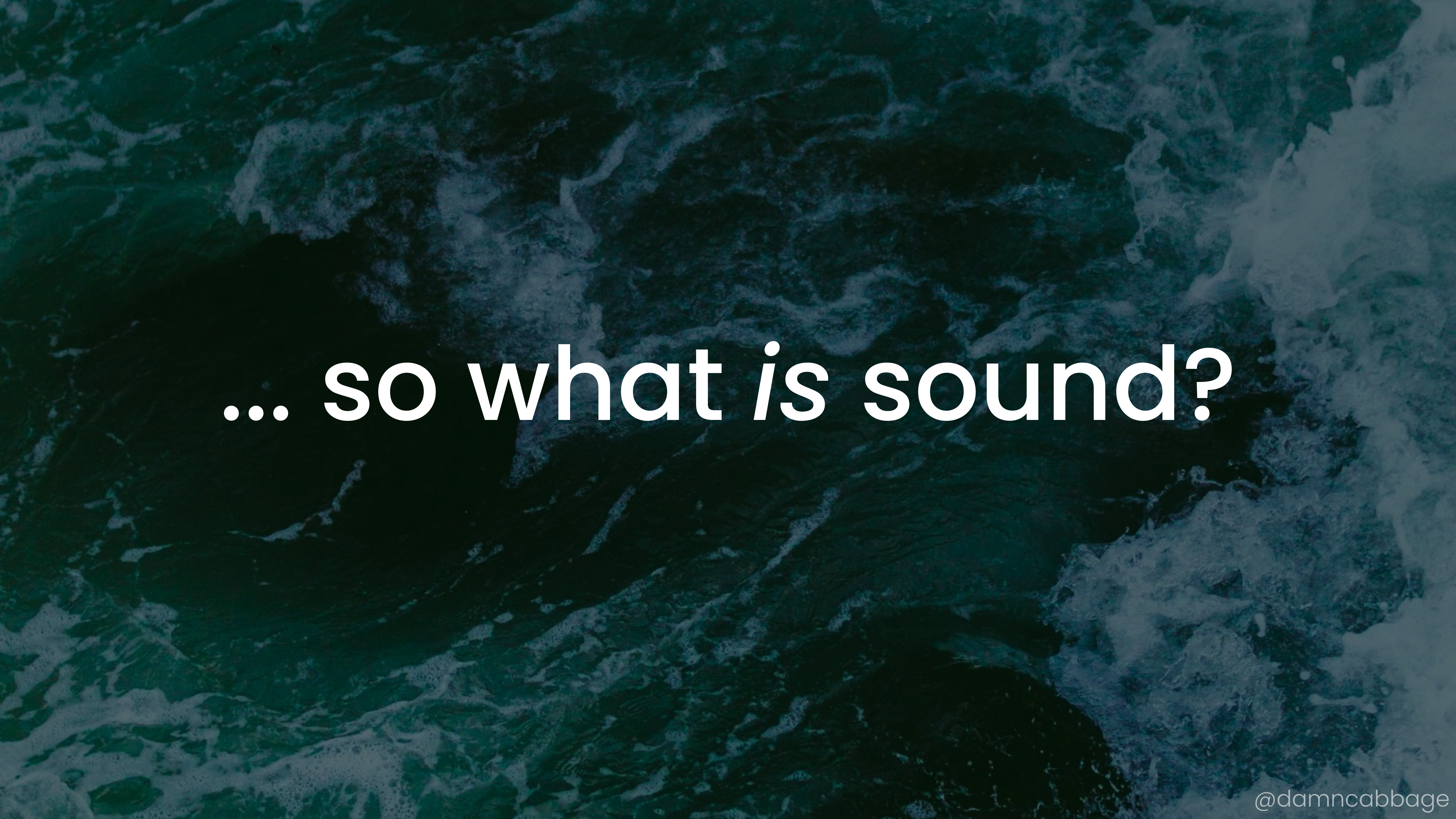


INSPECTION OF THE CENTER ENGRAVING
ON THE NICKEL PLATED MOTHER RECORD


DYNAMICS
Division



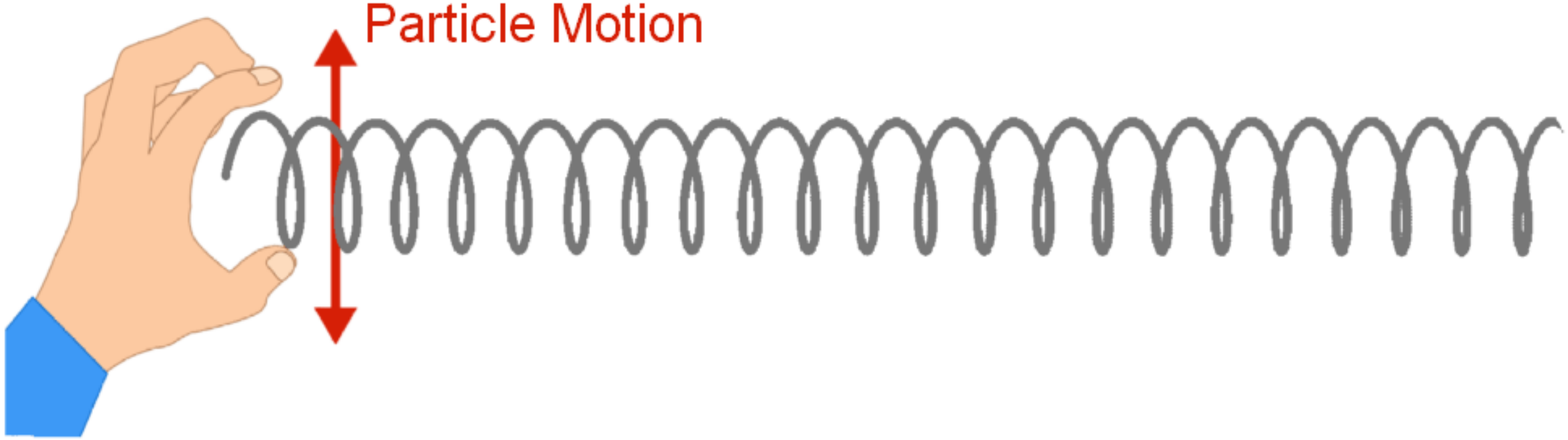
AUDIO

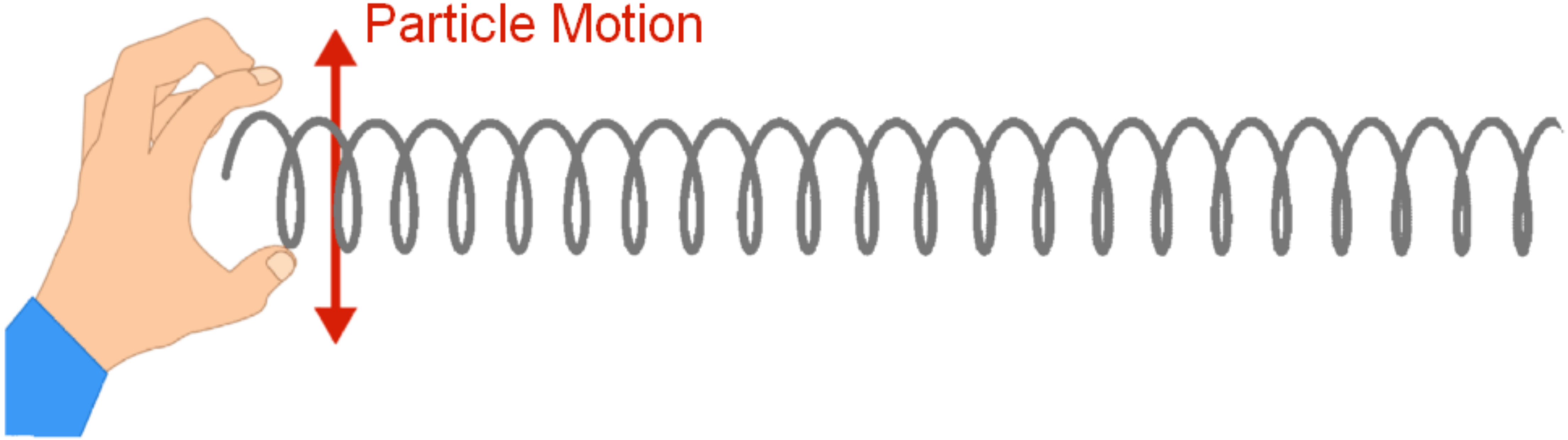


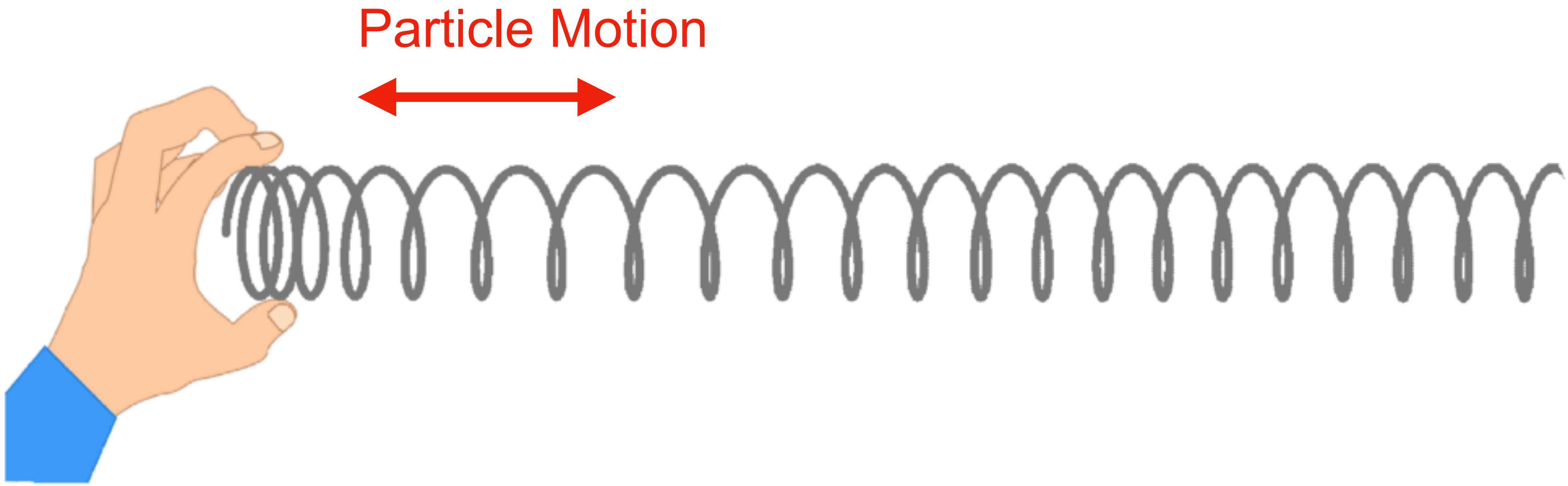
… so what *is* sound?

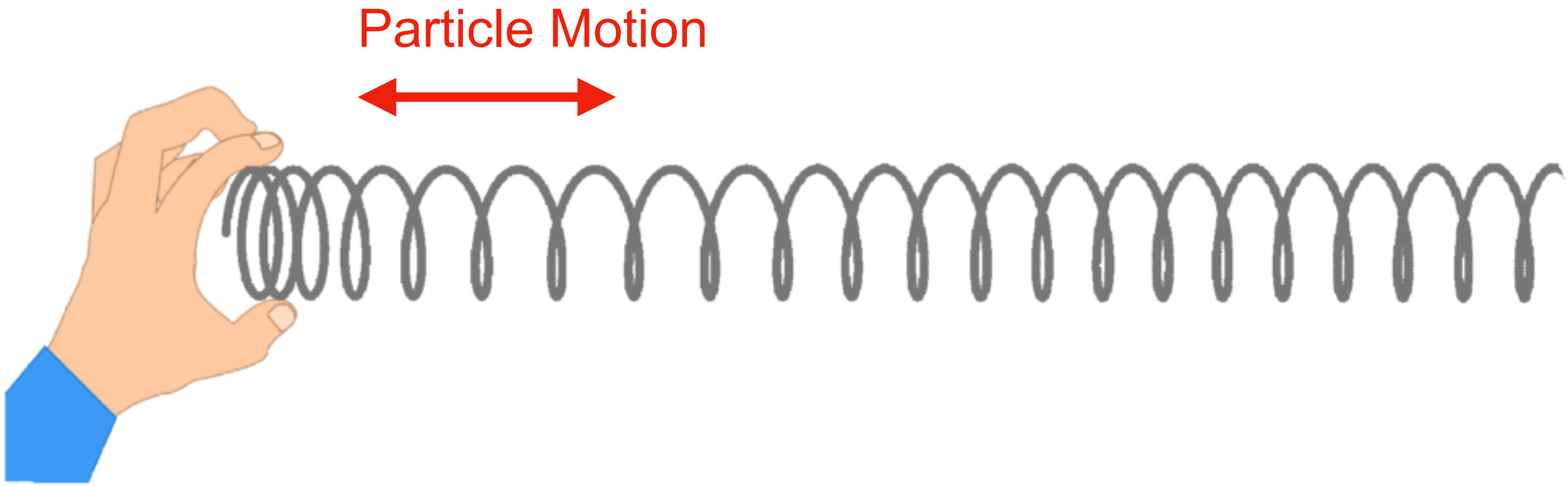
Let's talk about waves.

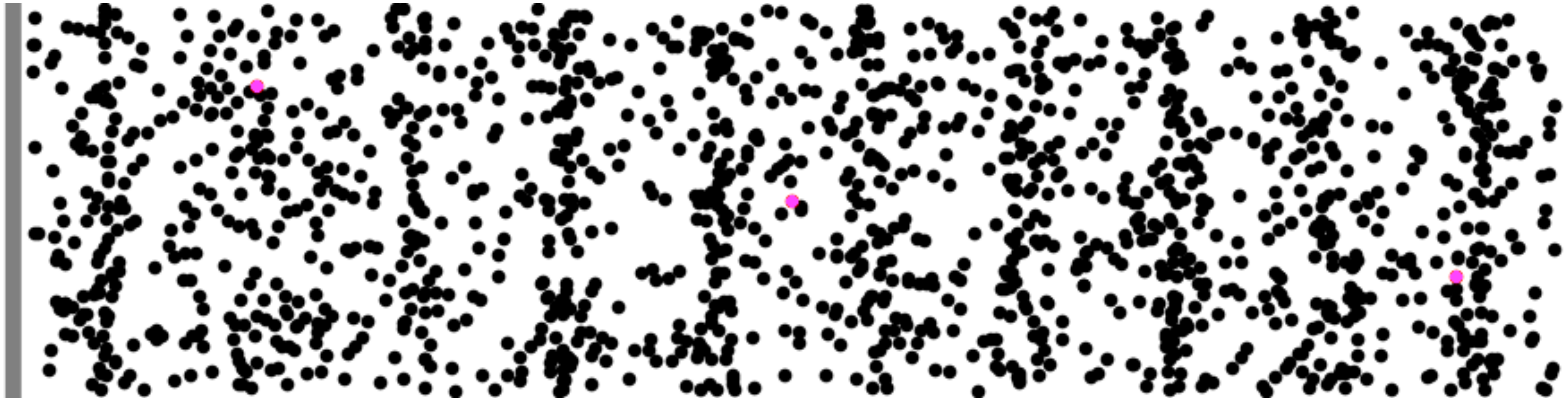
(but not that kind)

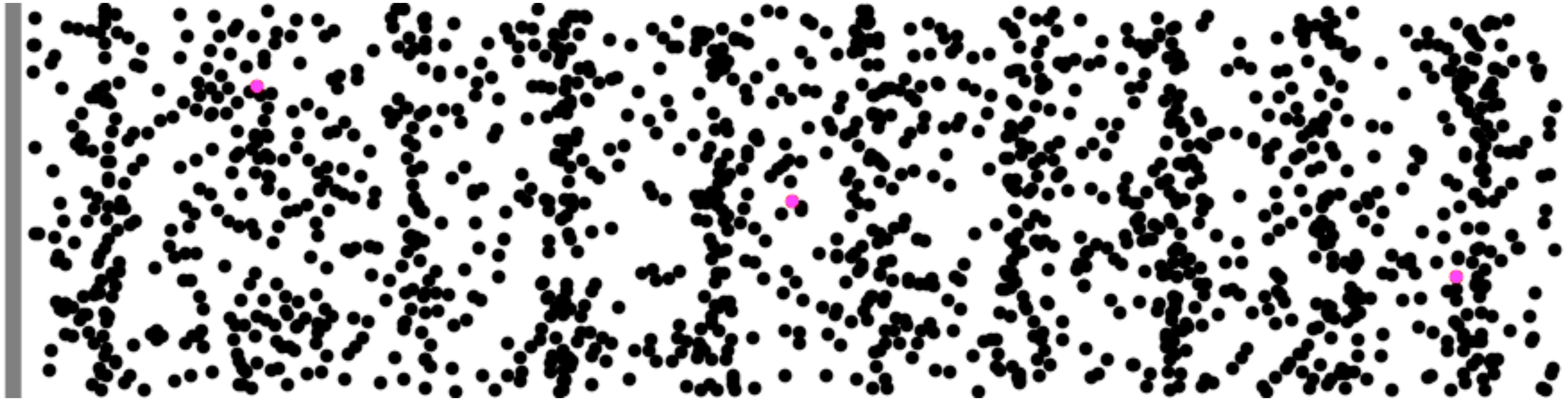

Particle Motion

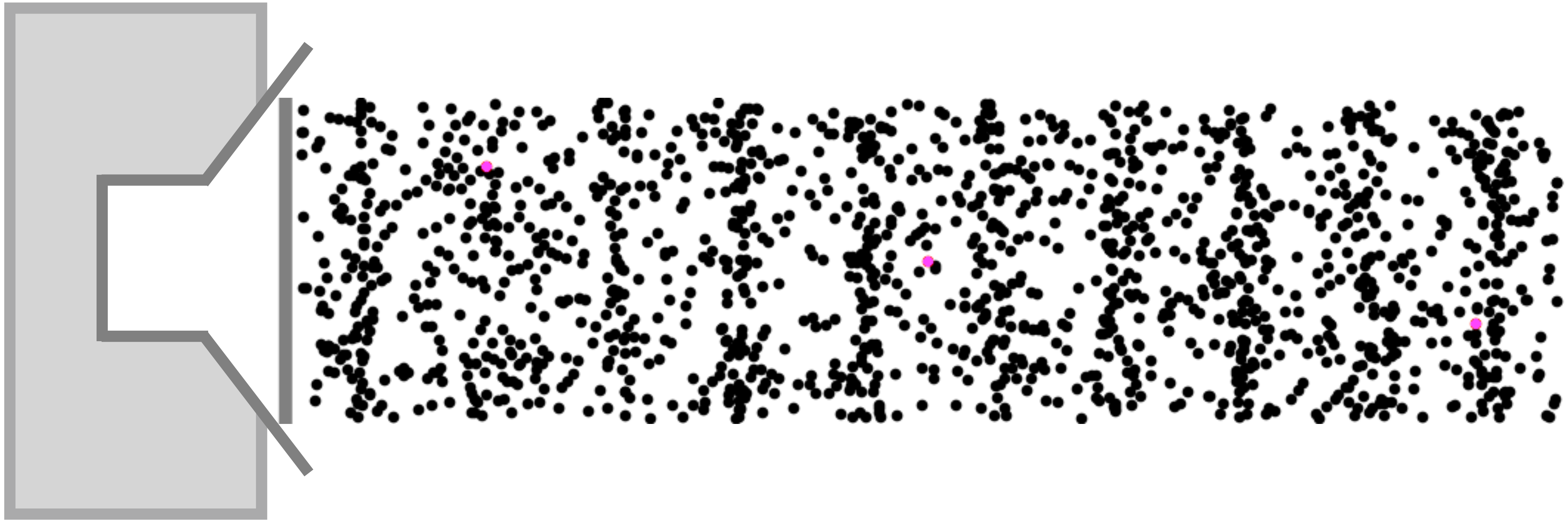

Particle Motion

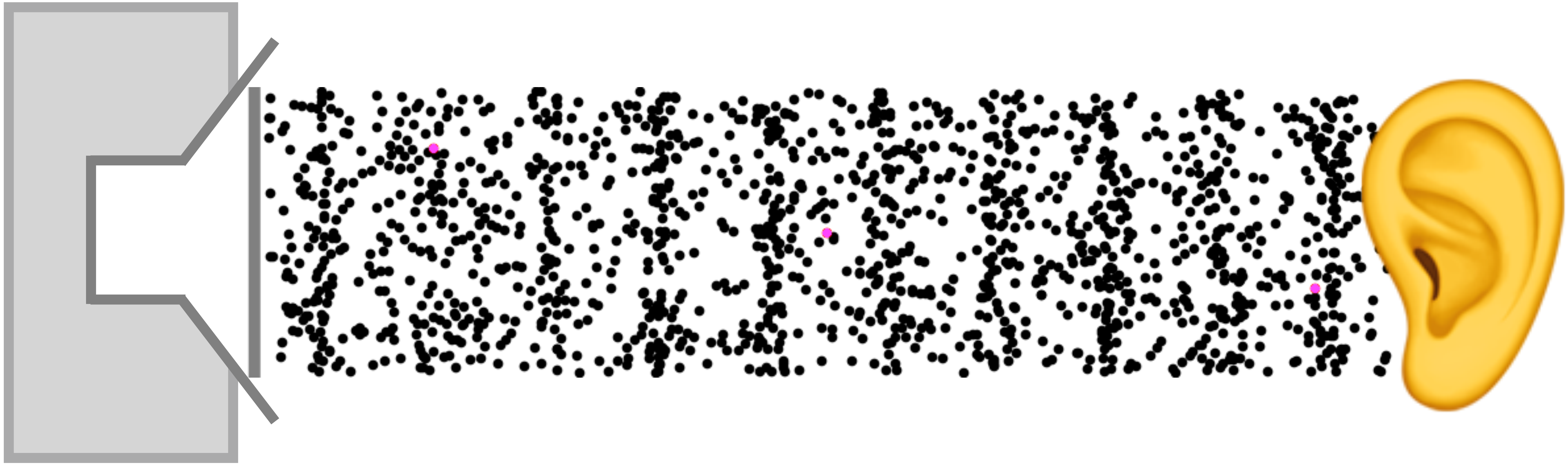

Particle Motion

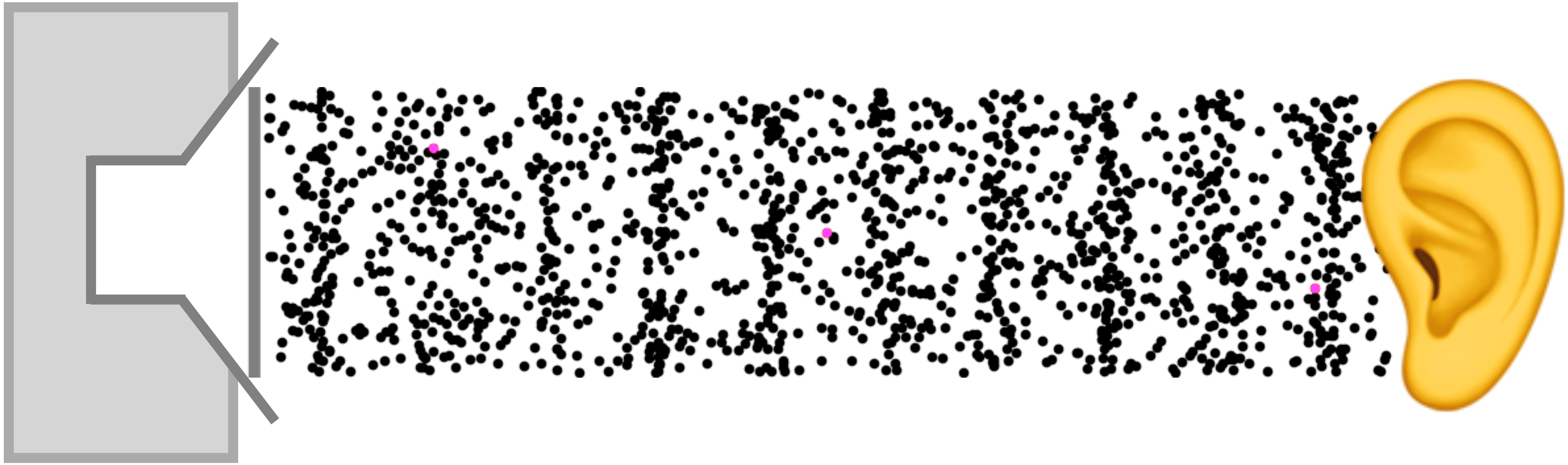

Particle Motion



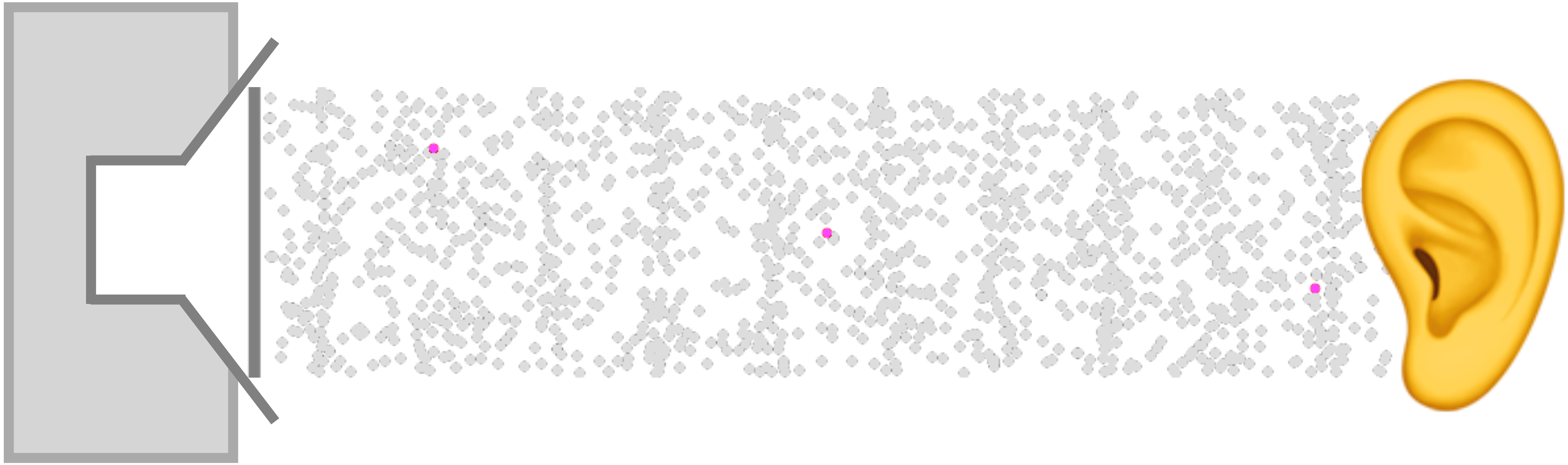






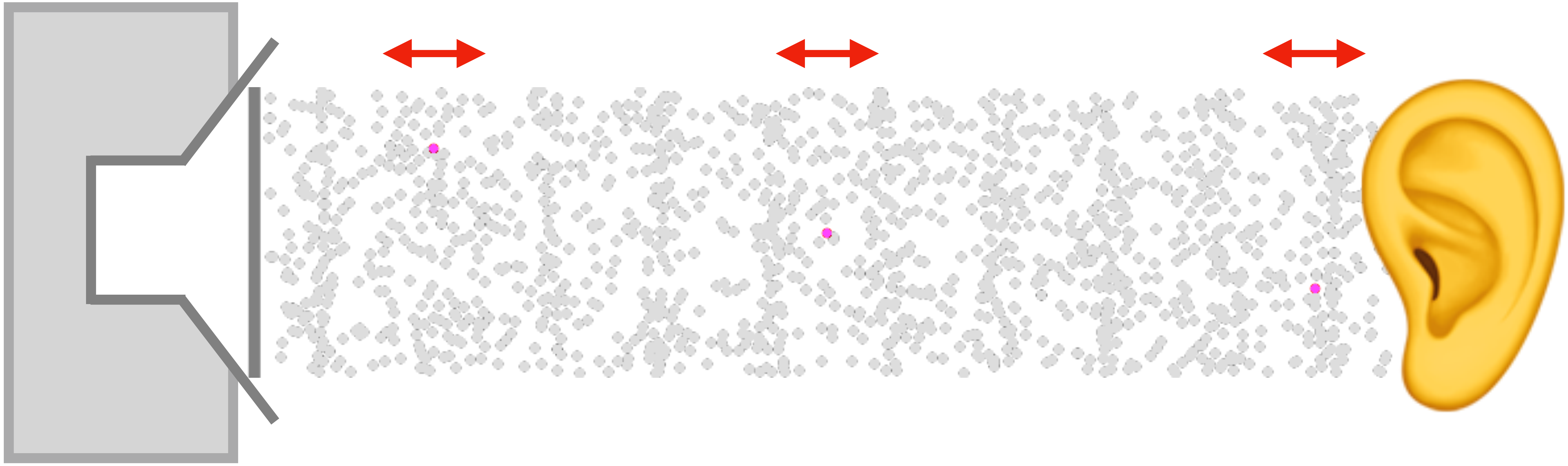


@damncabbage

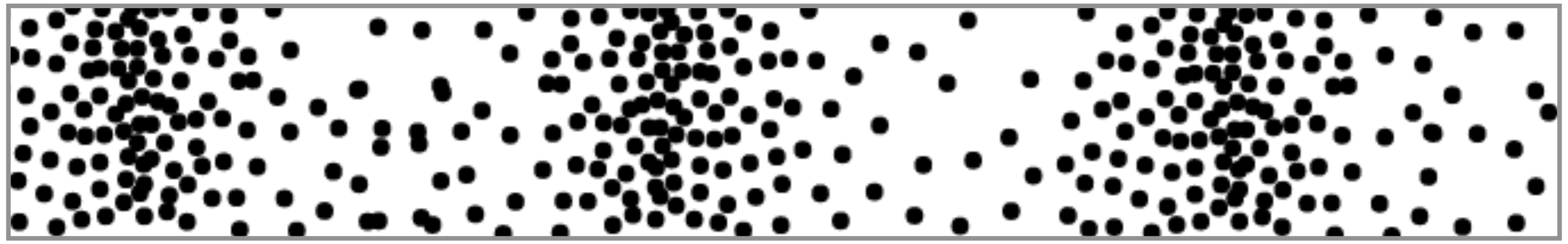


@damncabbage

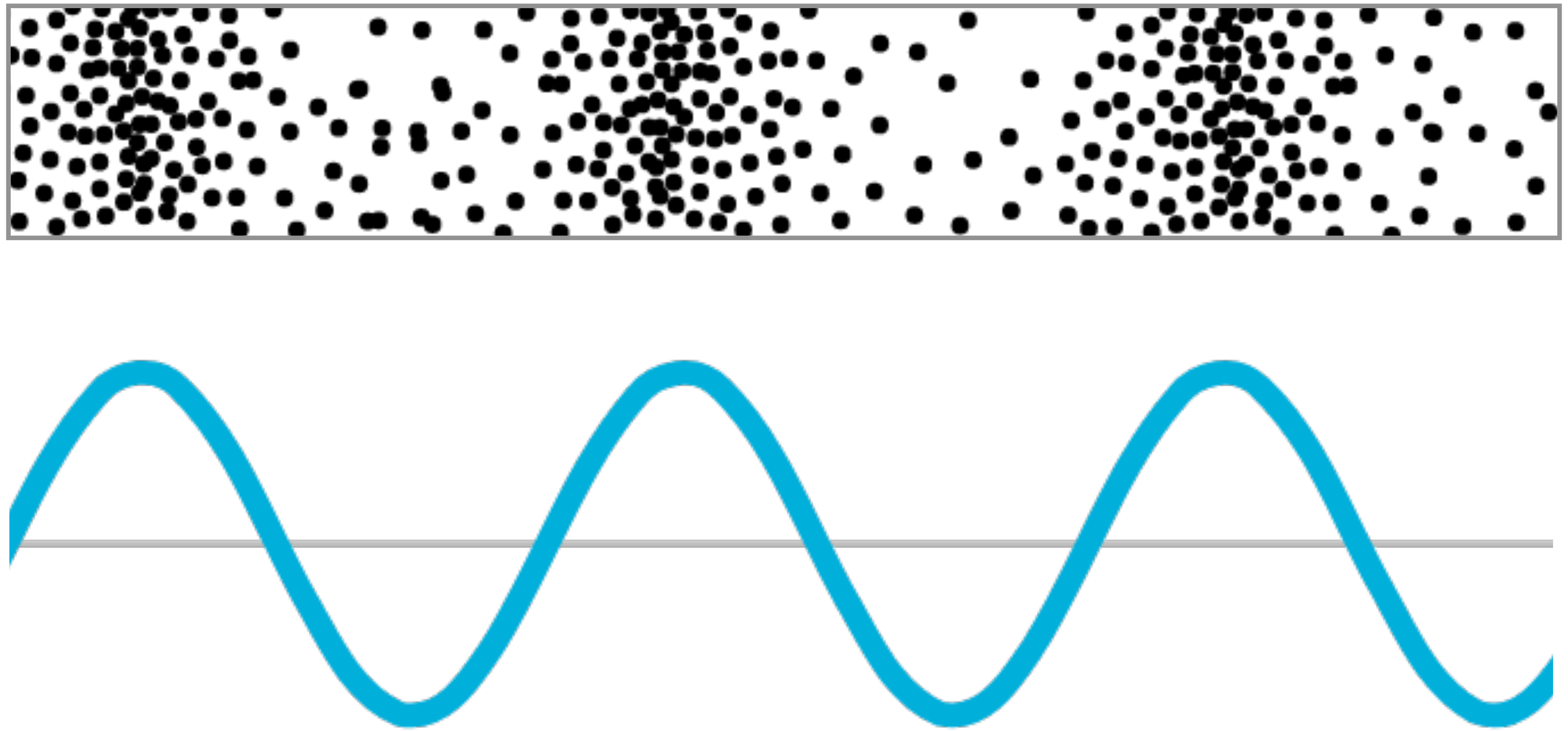
Particle Motion

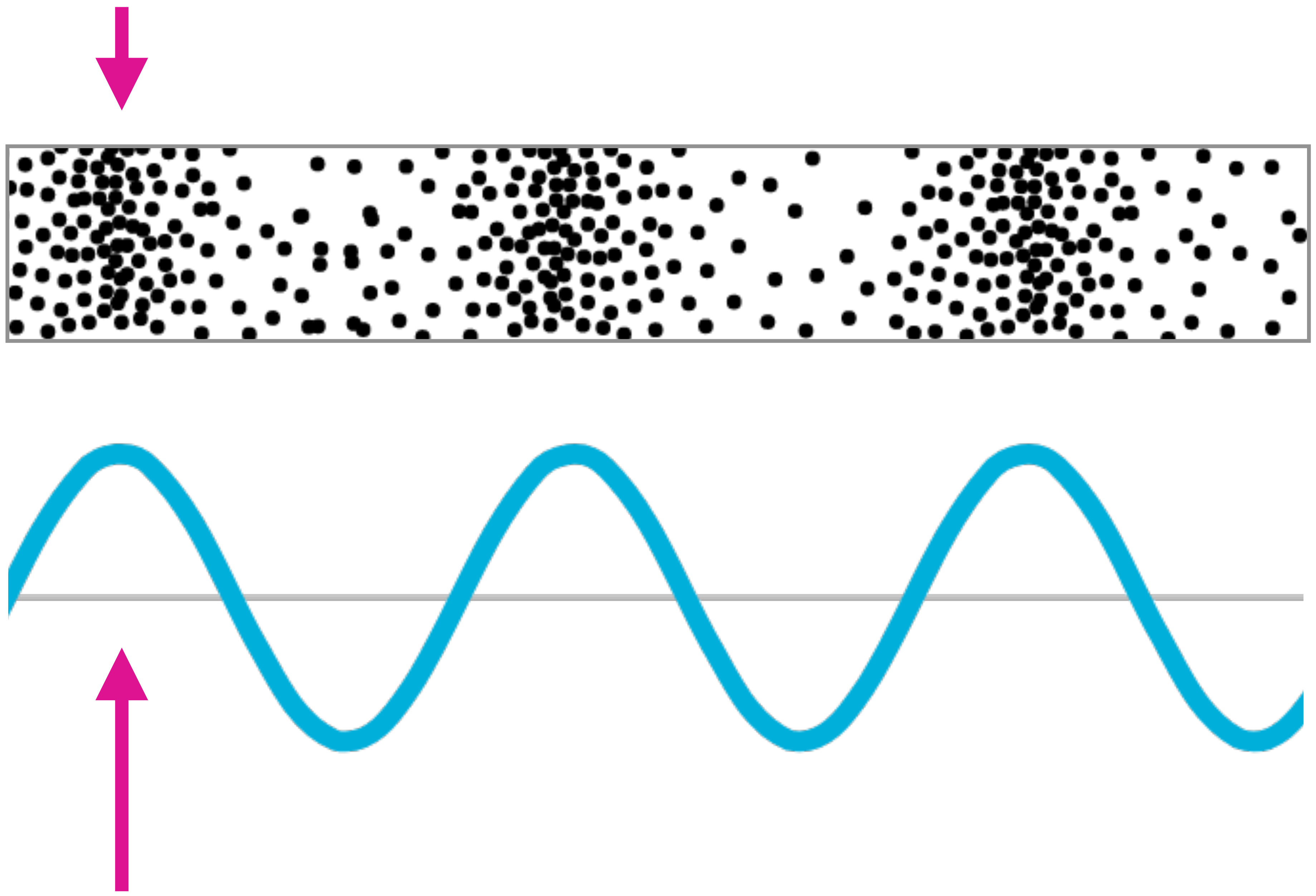


@damncabbage

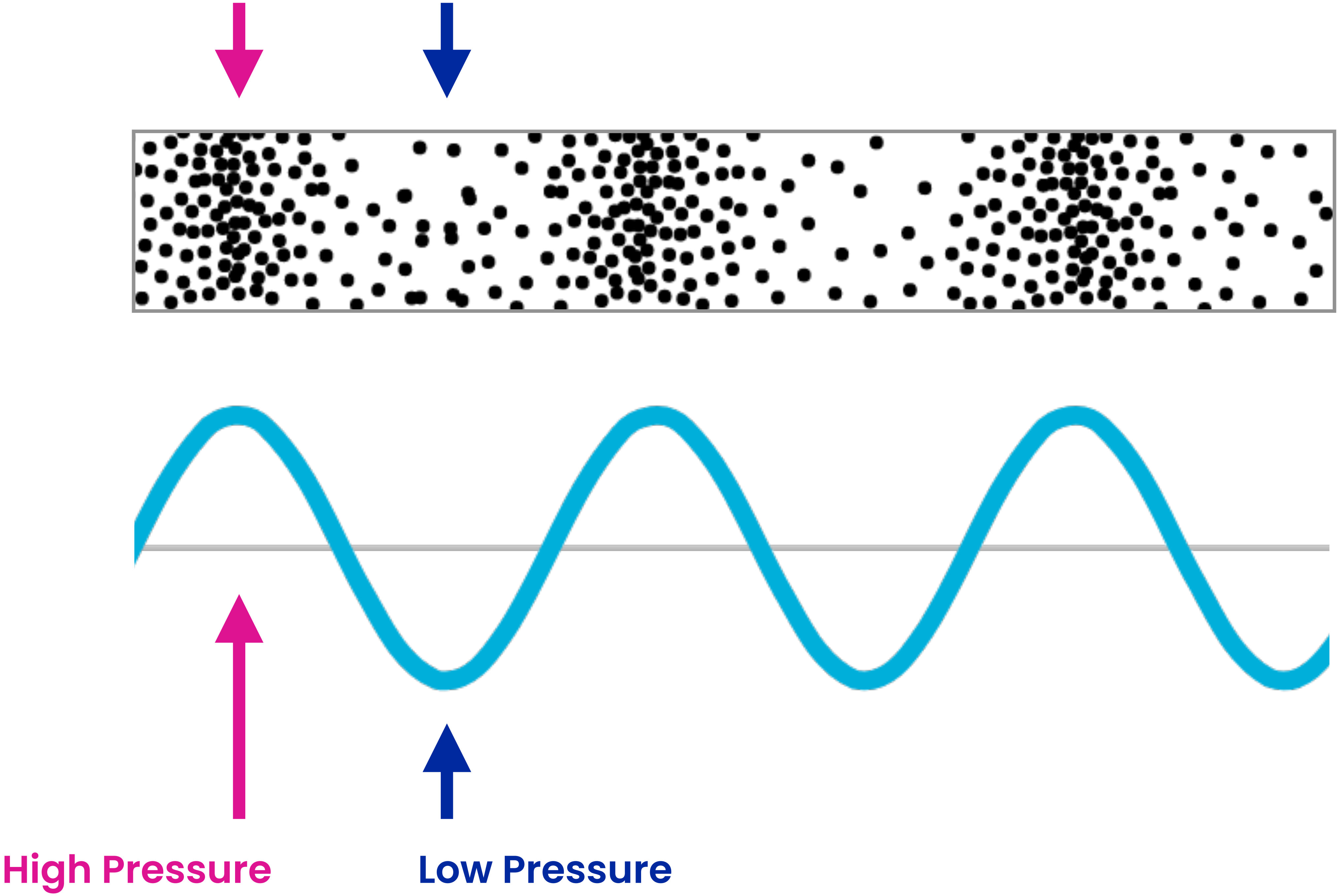


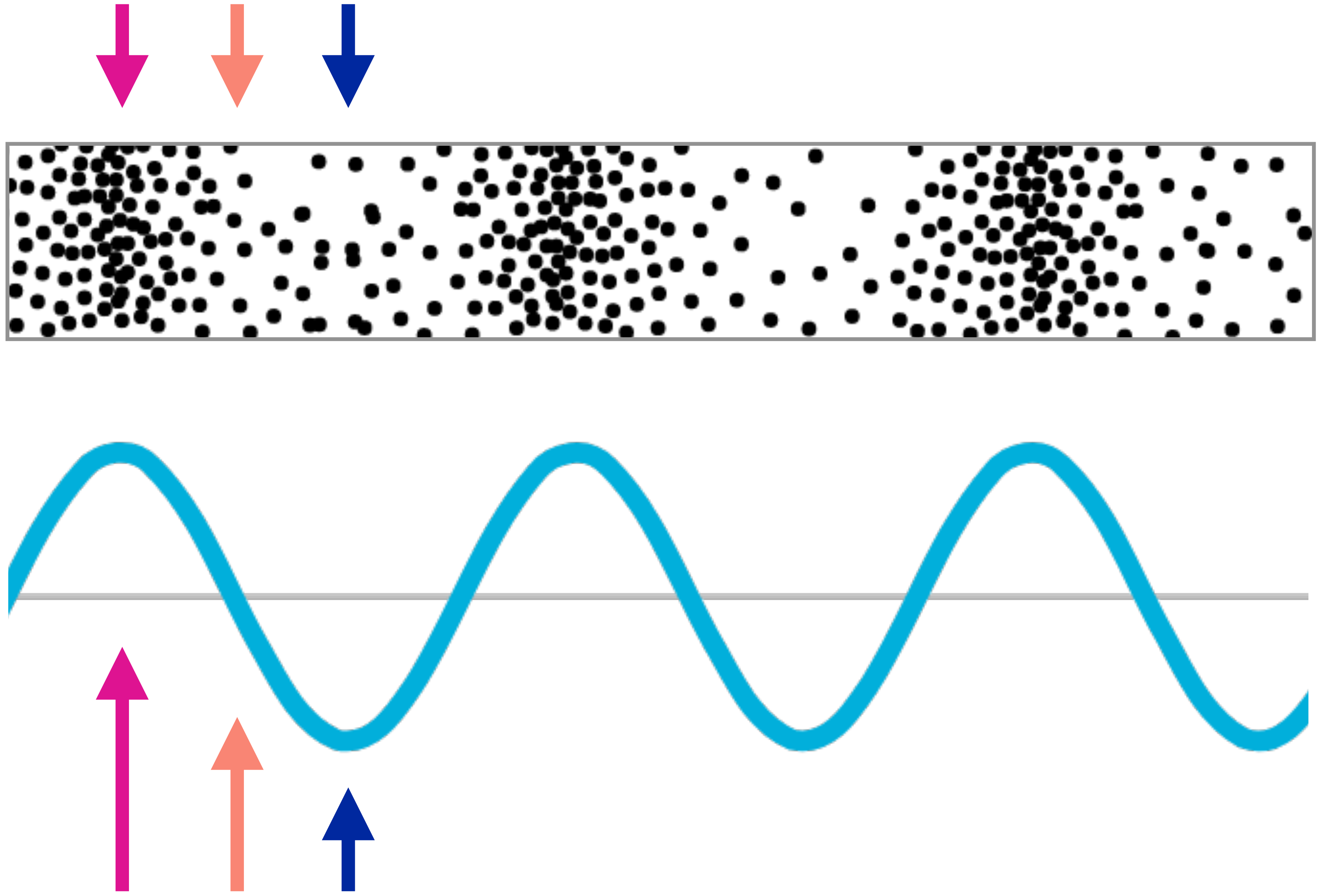
@damncabbage





High Pressure

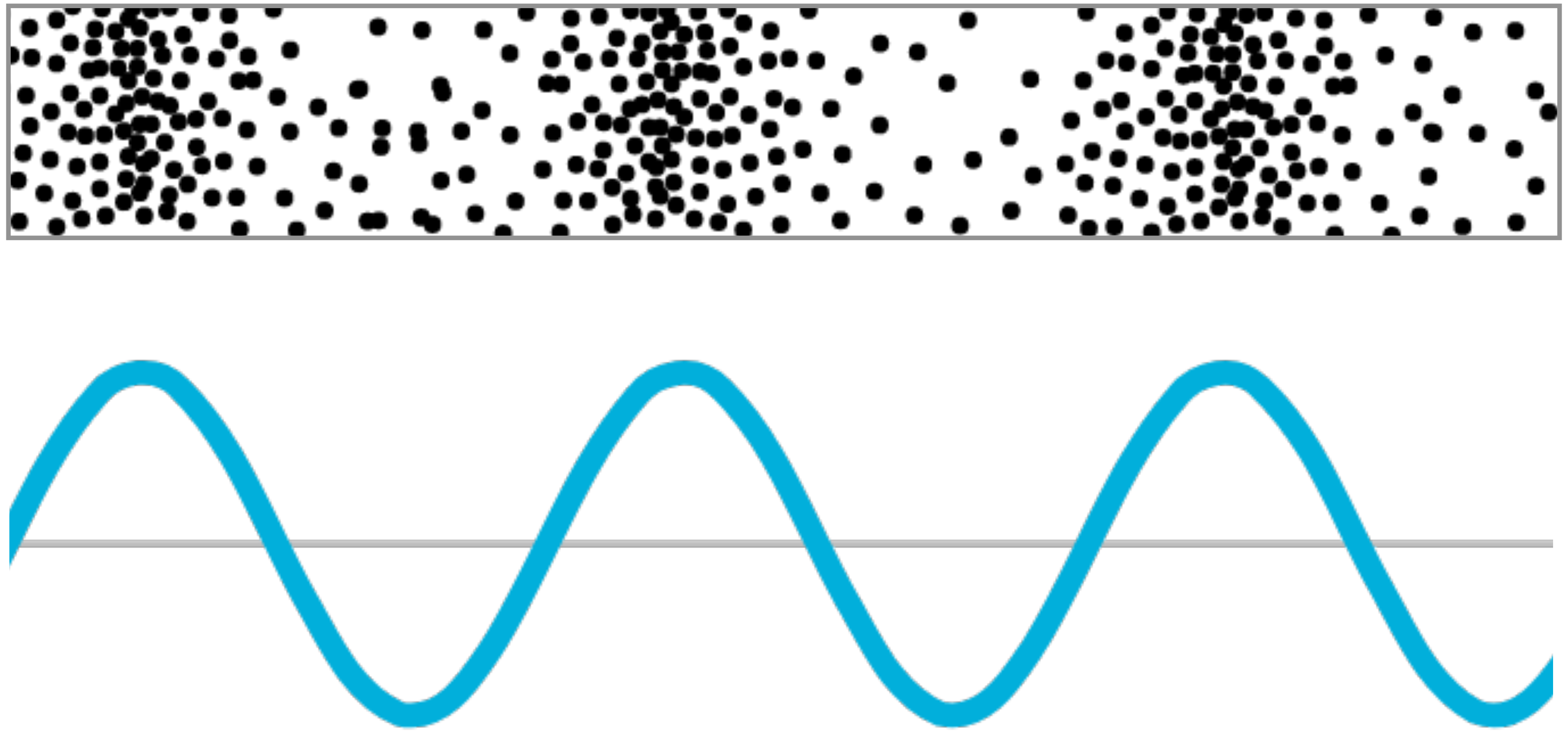

High Pressure
Low Pressure

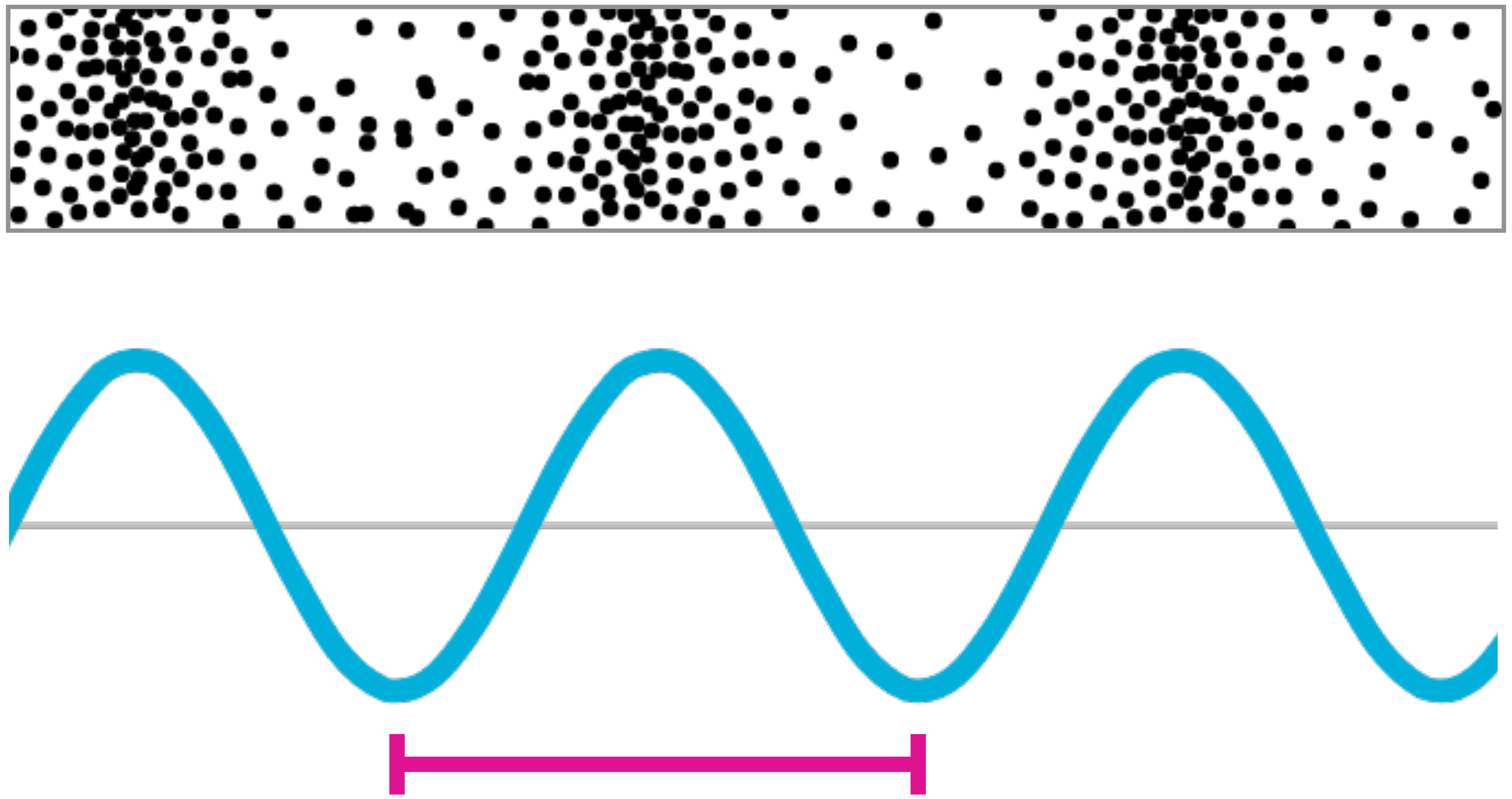


High Pressure

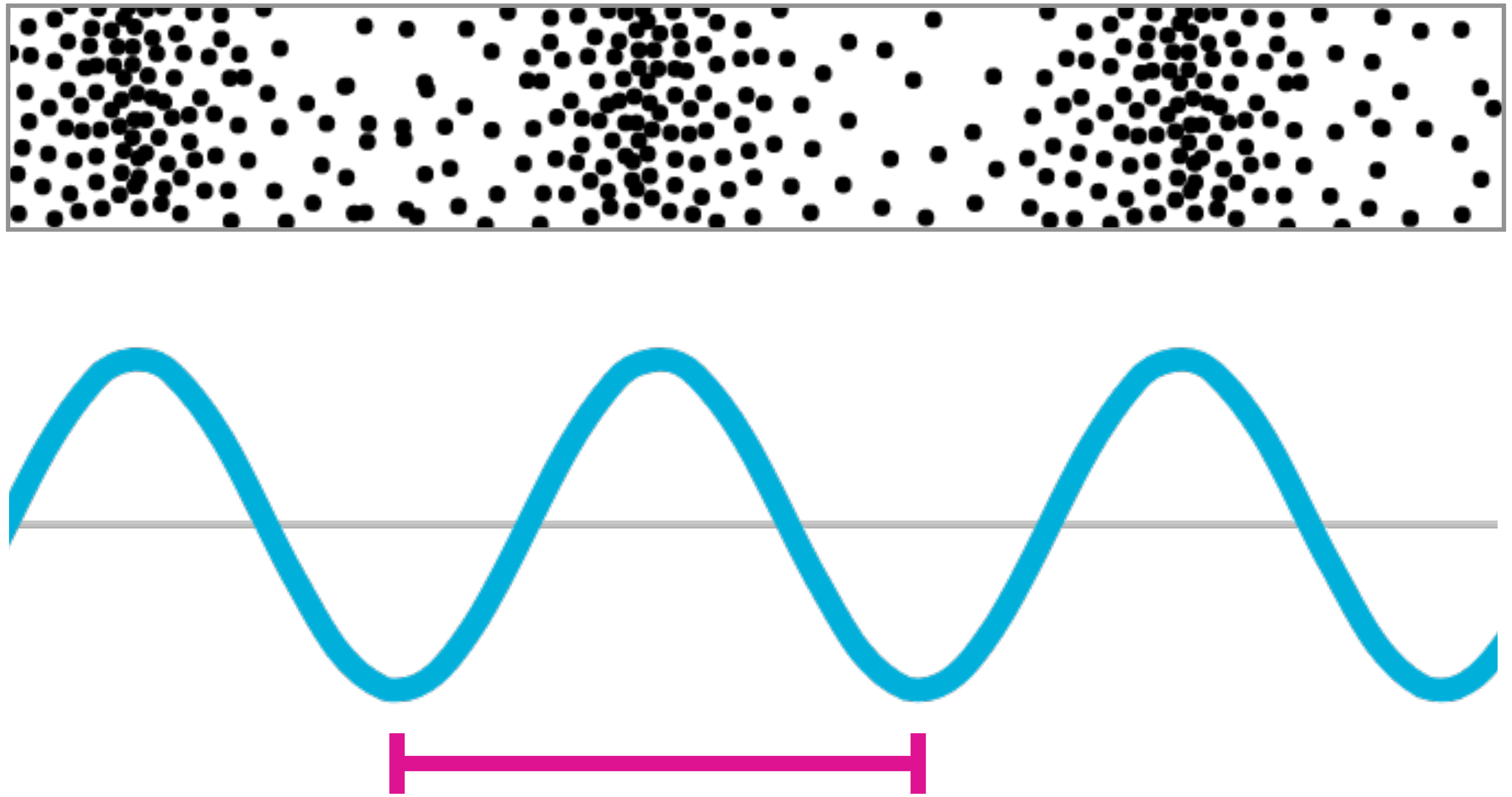
Low Pressure

Ambient Pressure

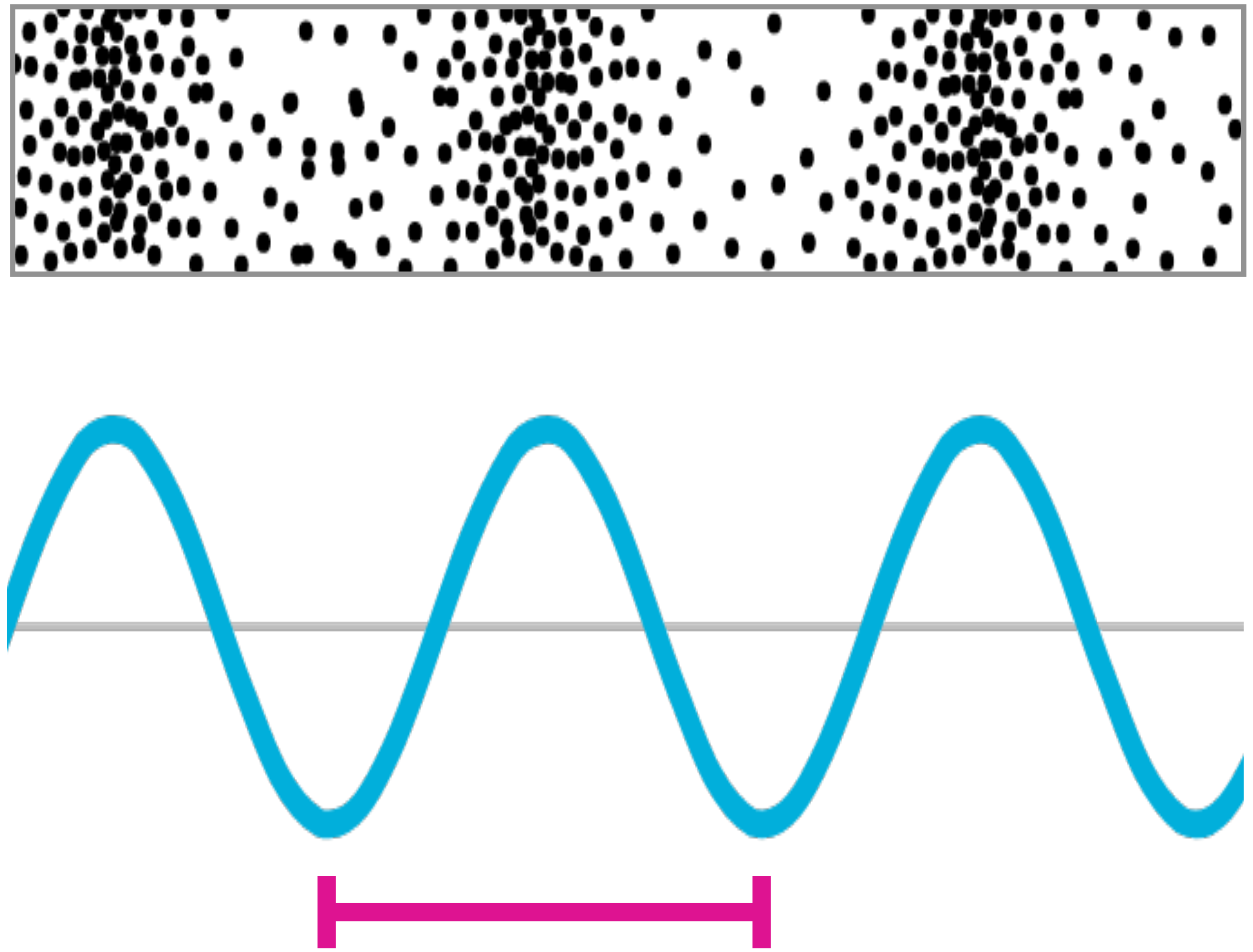




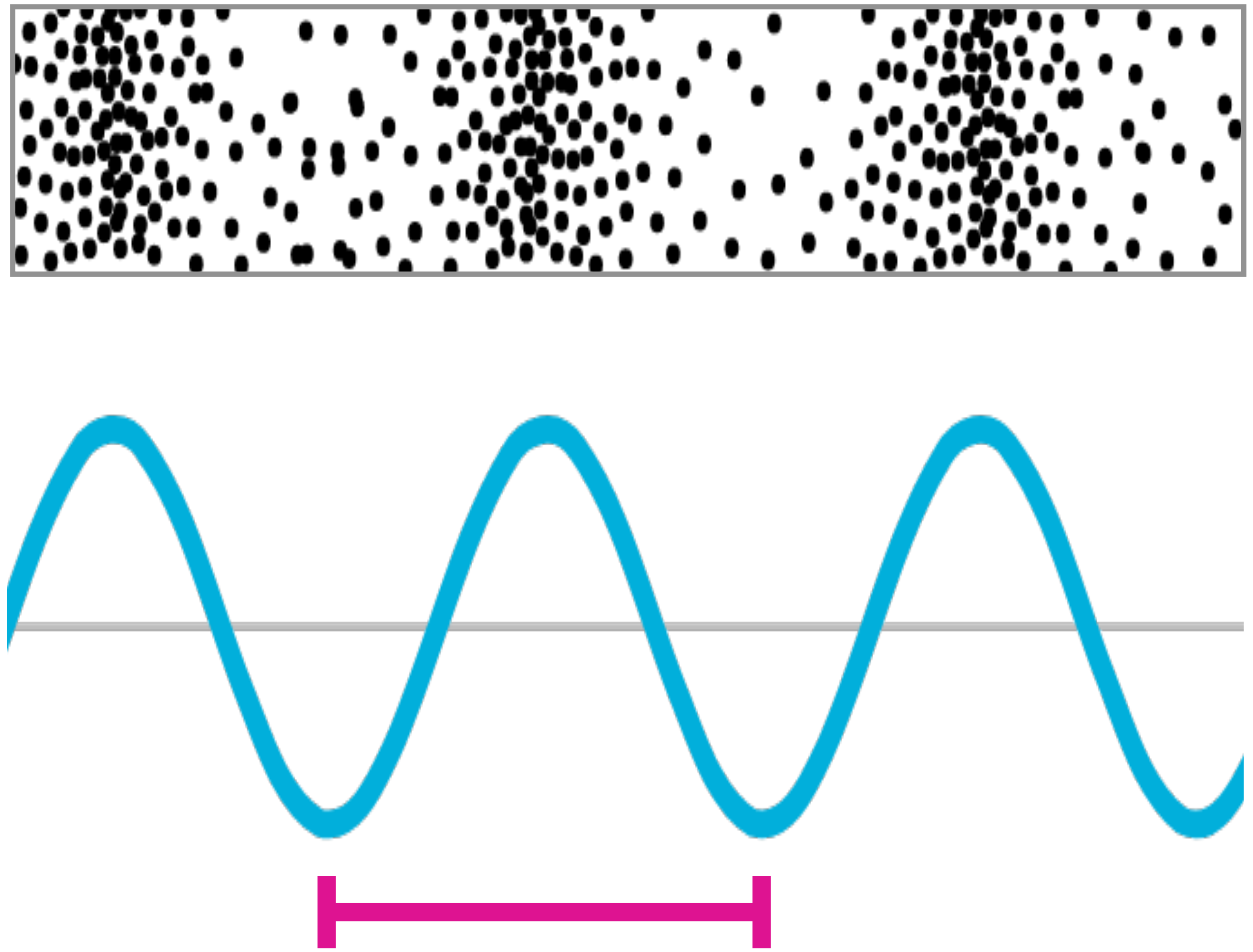
Wave-length



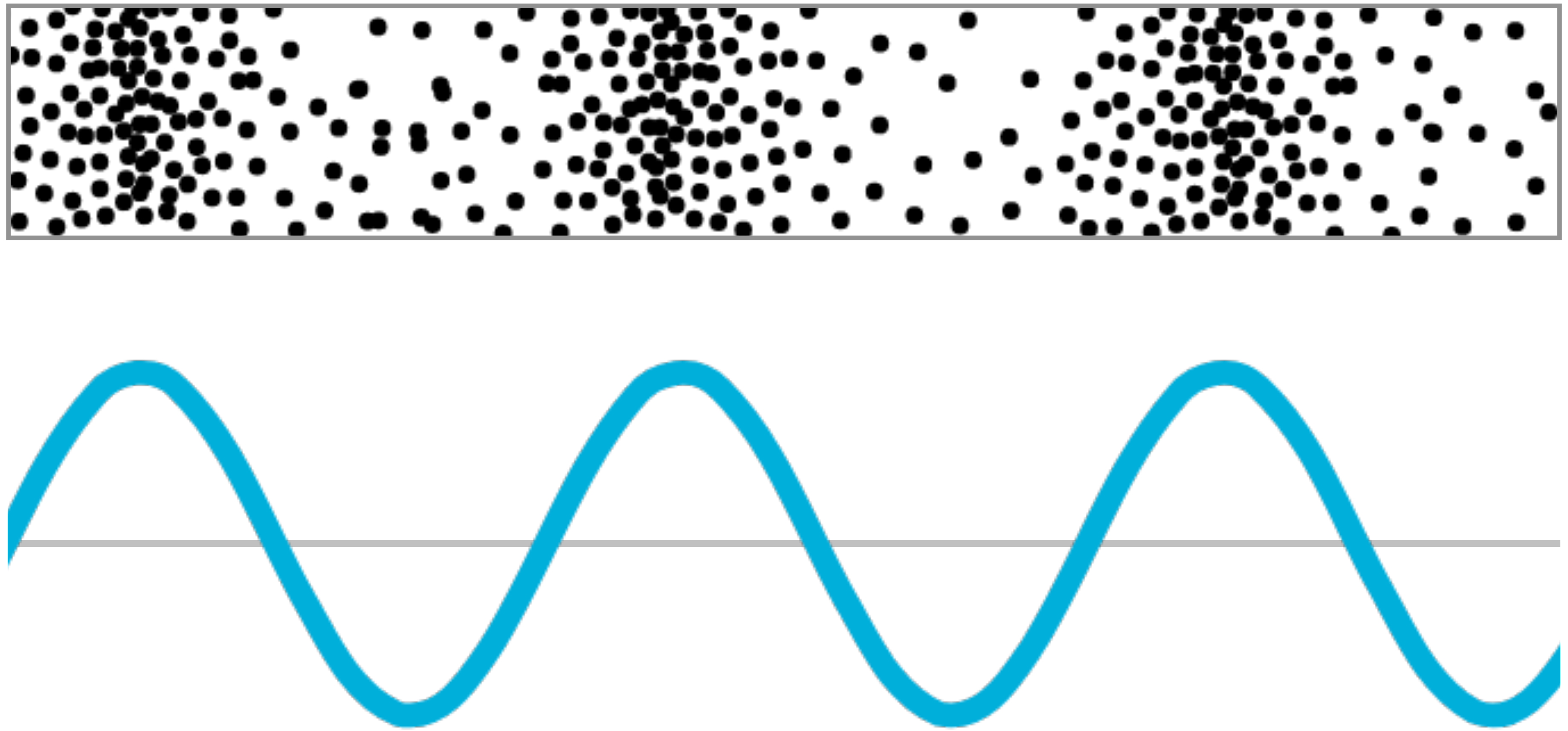
Wave-length

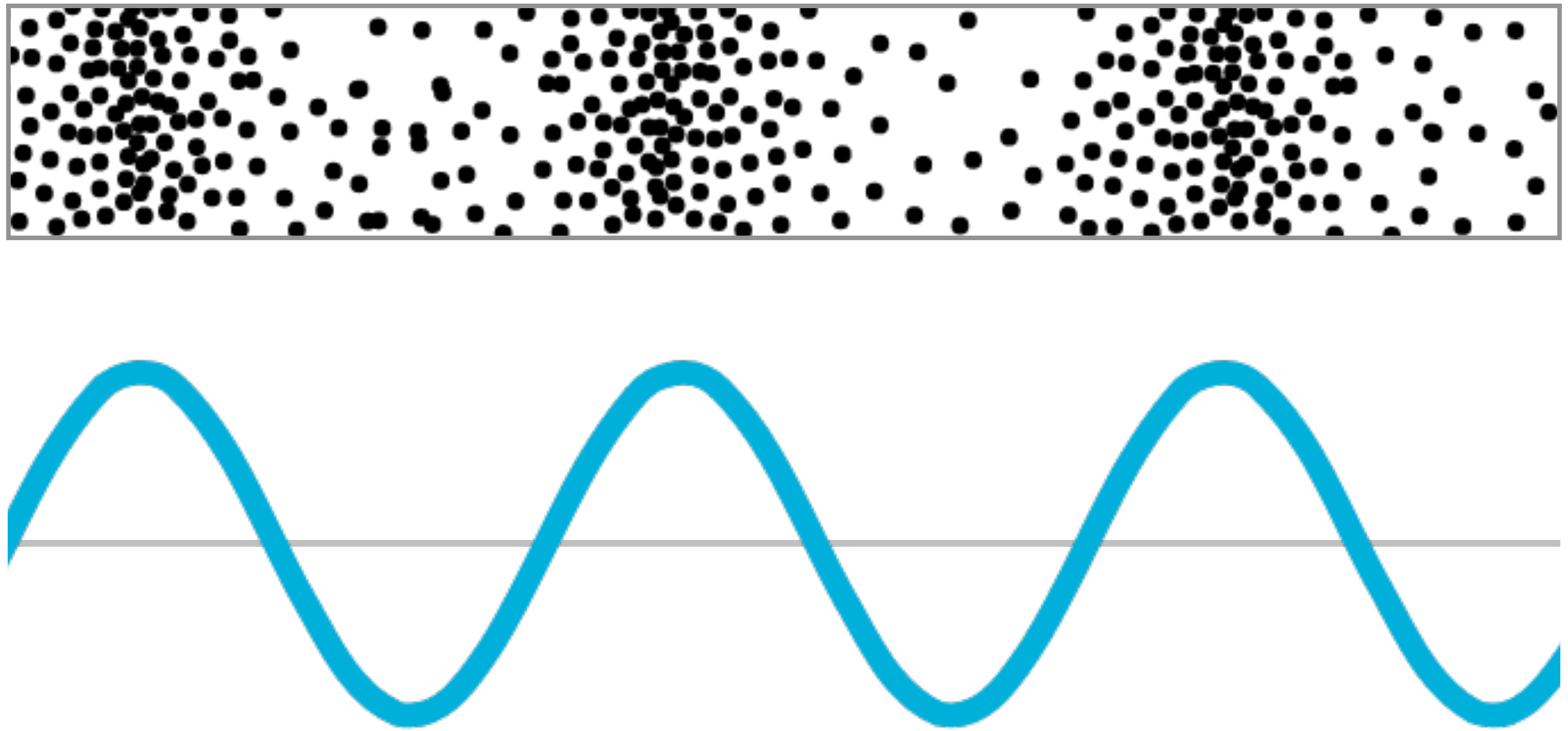


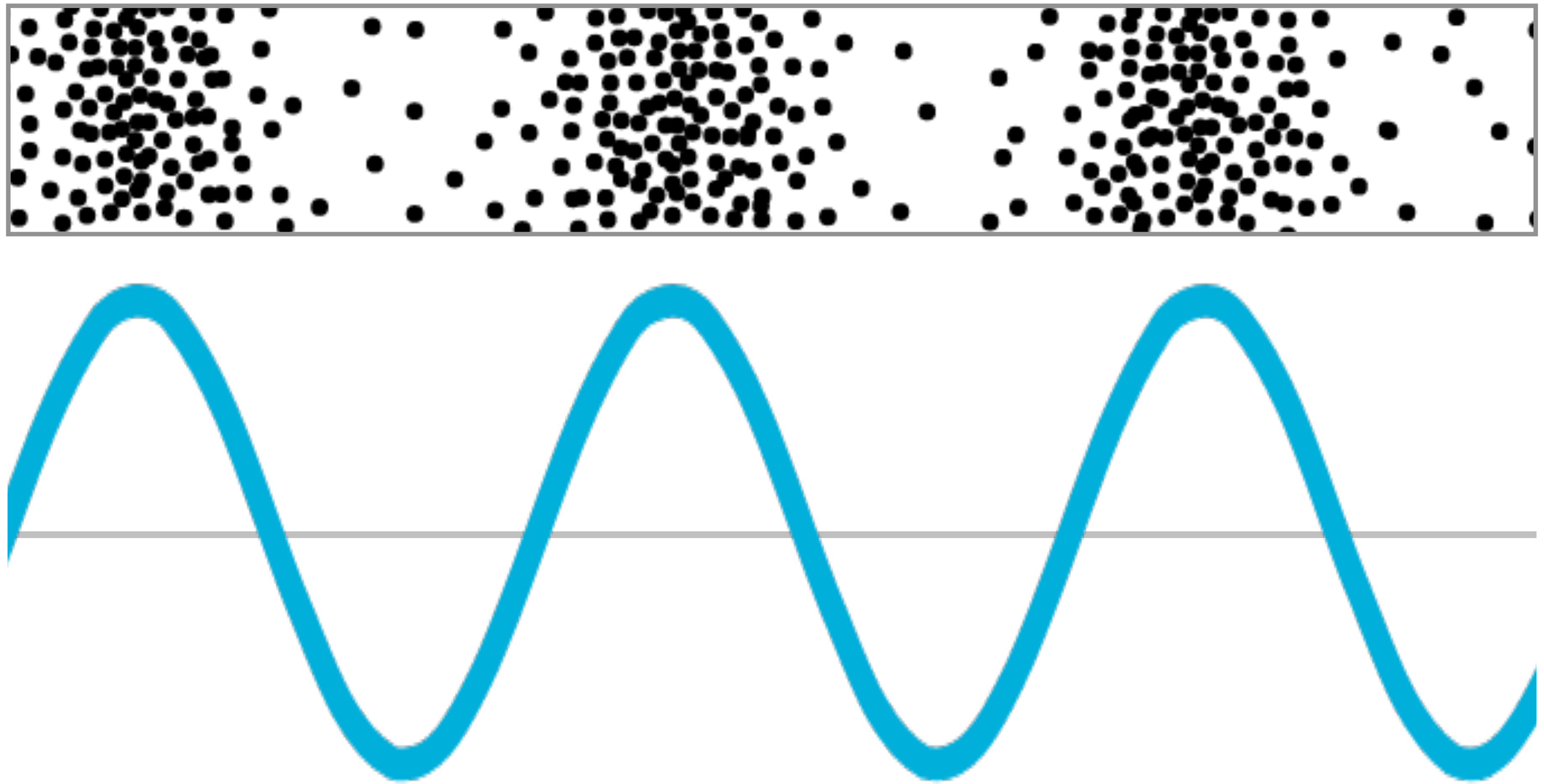
Wave-length



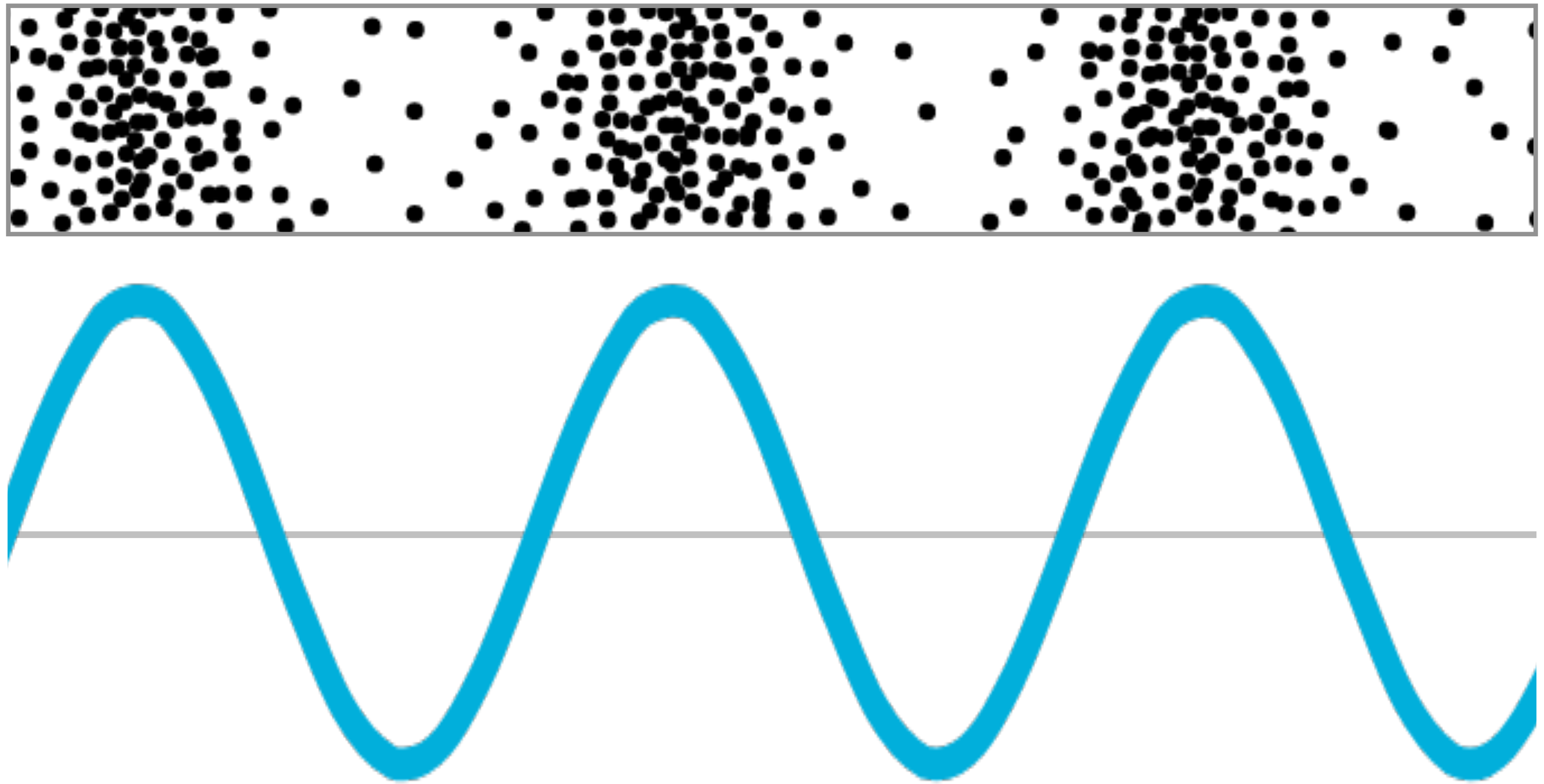
Wave-length



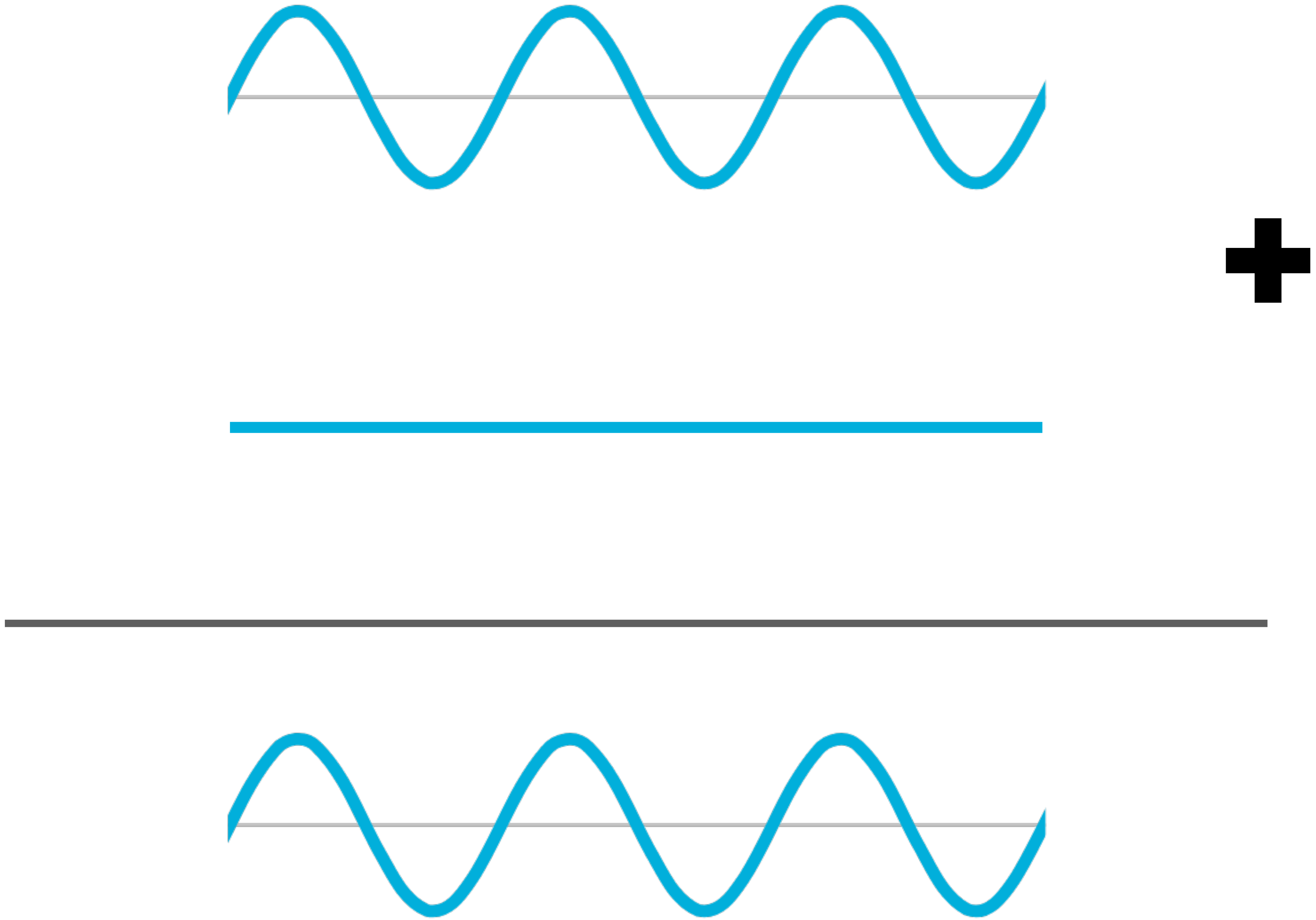


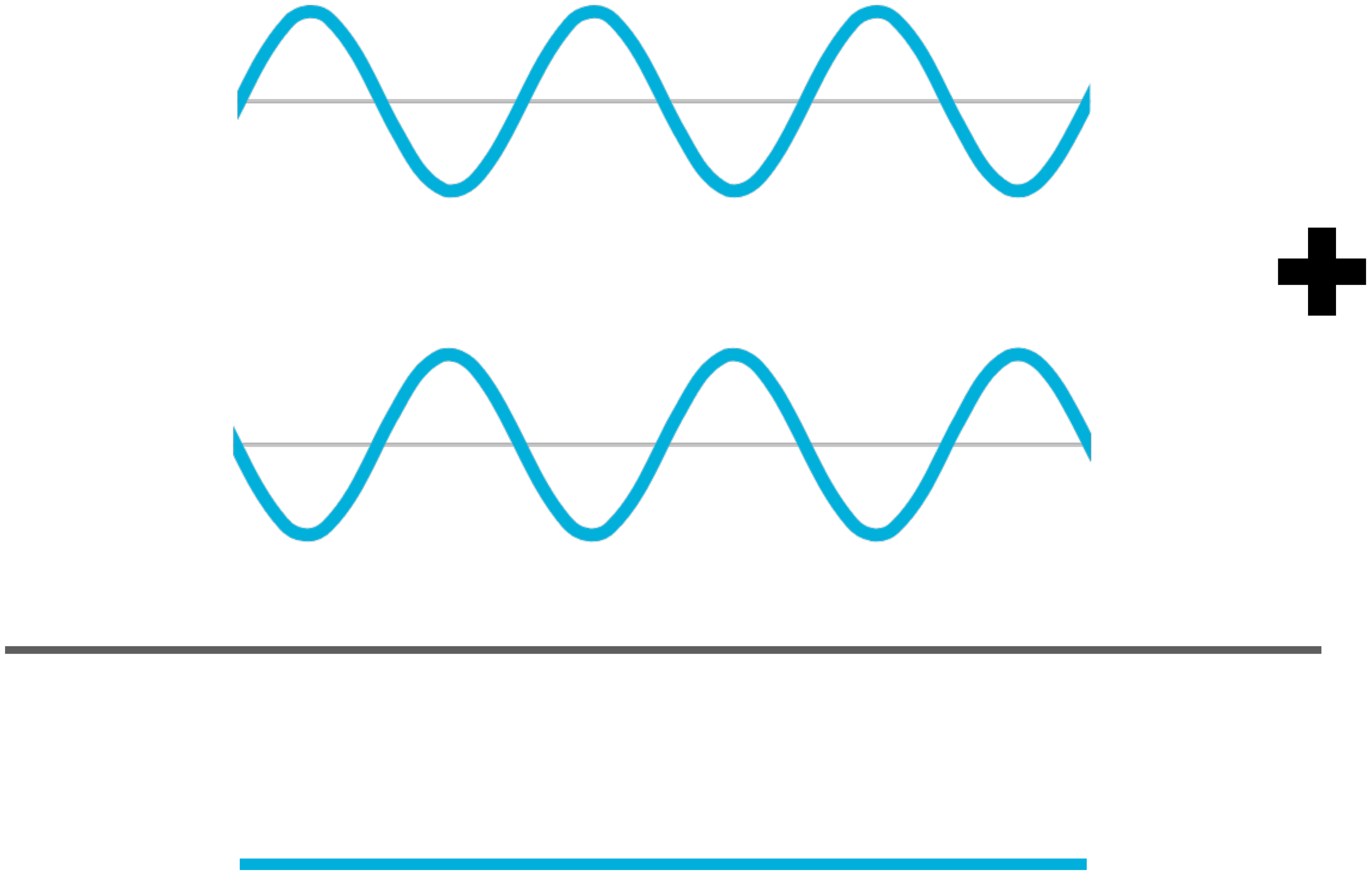


@damncabbage

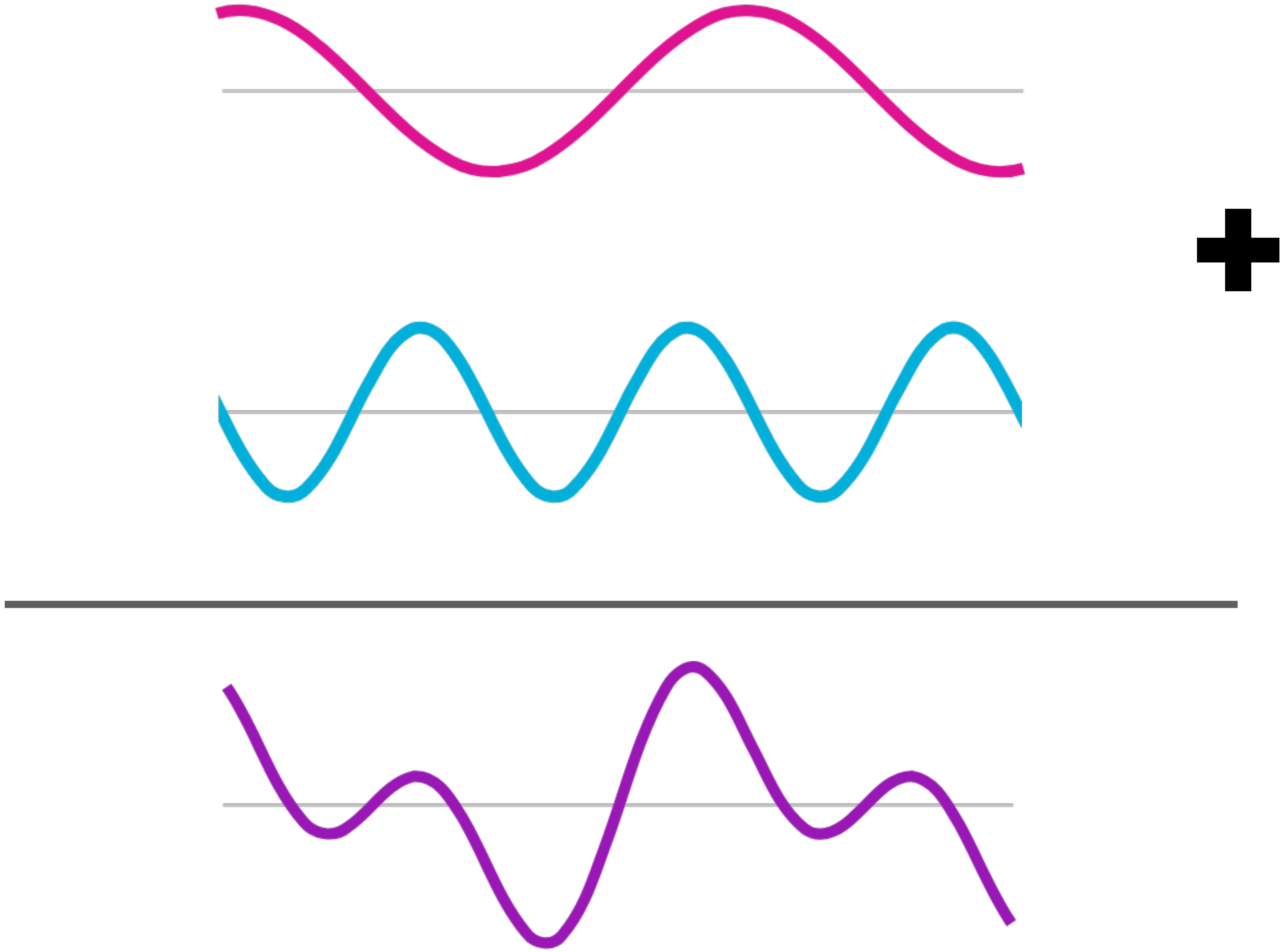


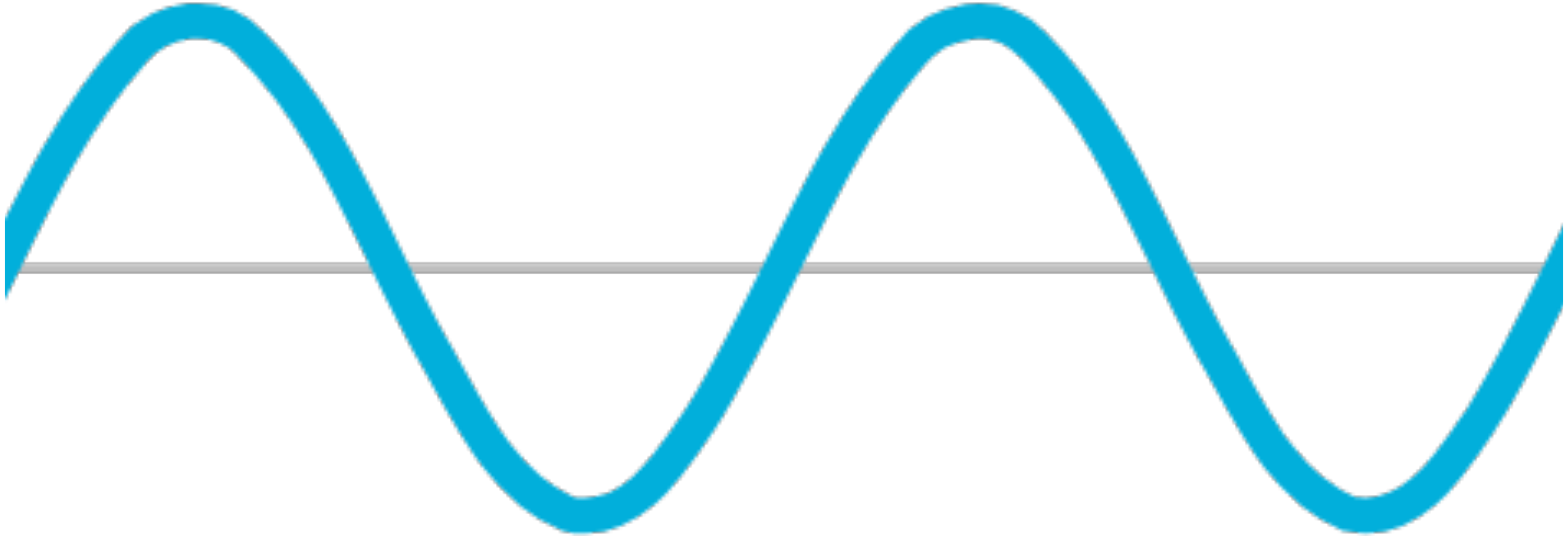
@damncabbage



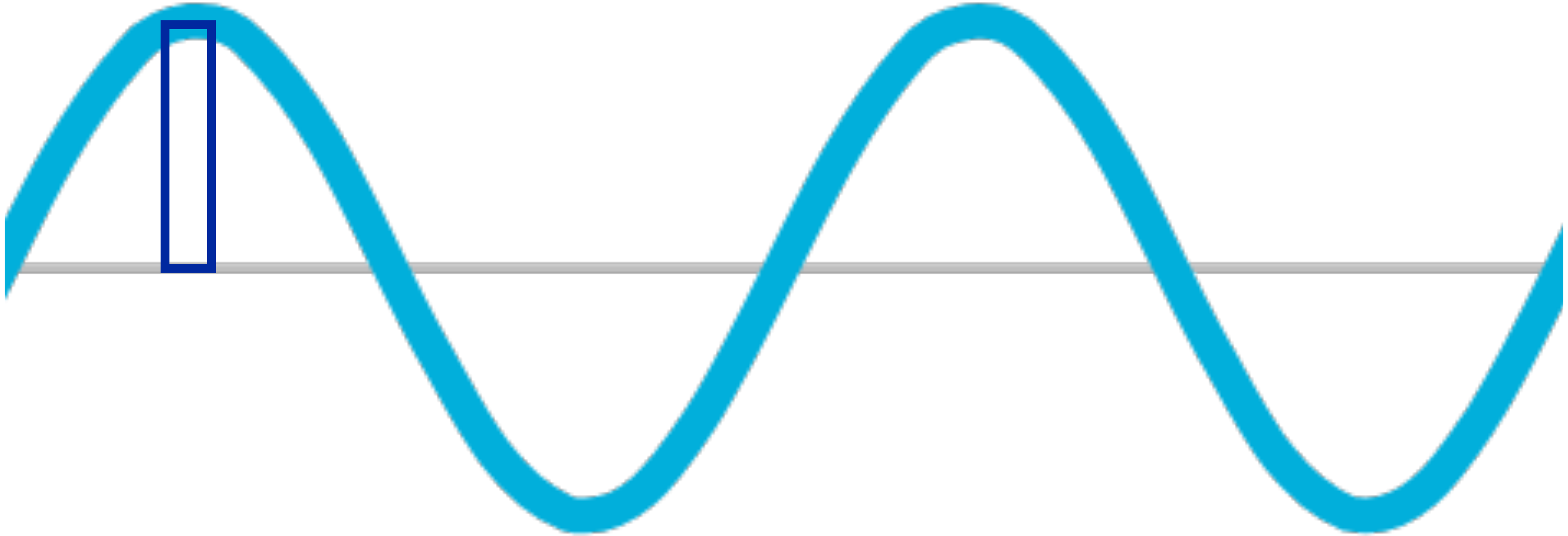


@damncabbage

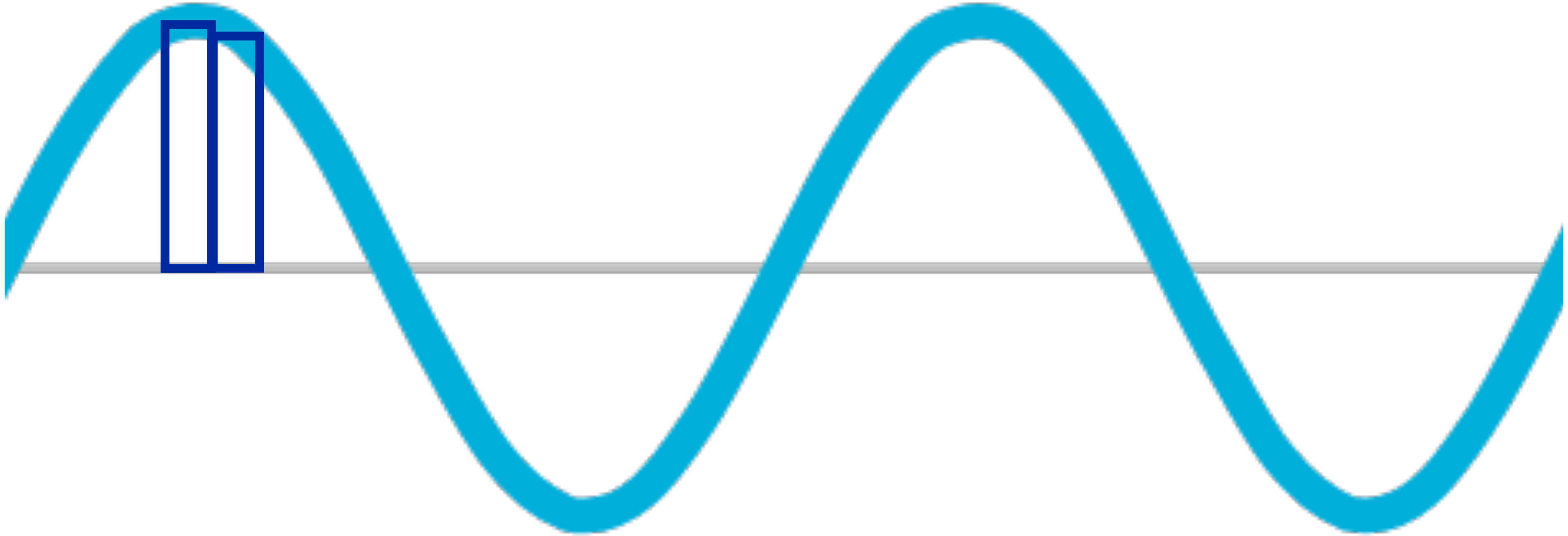


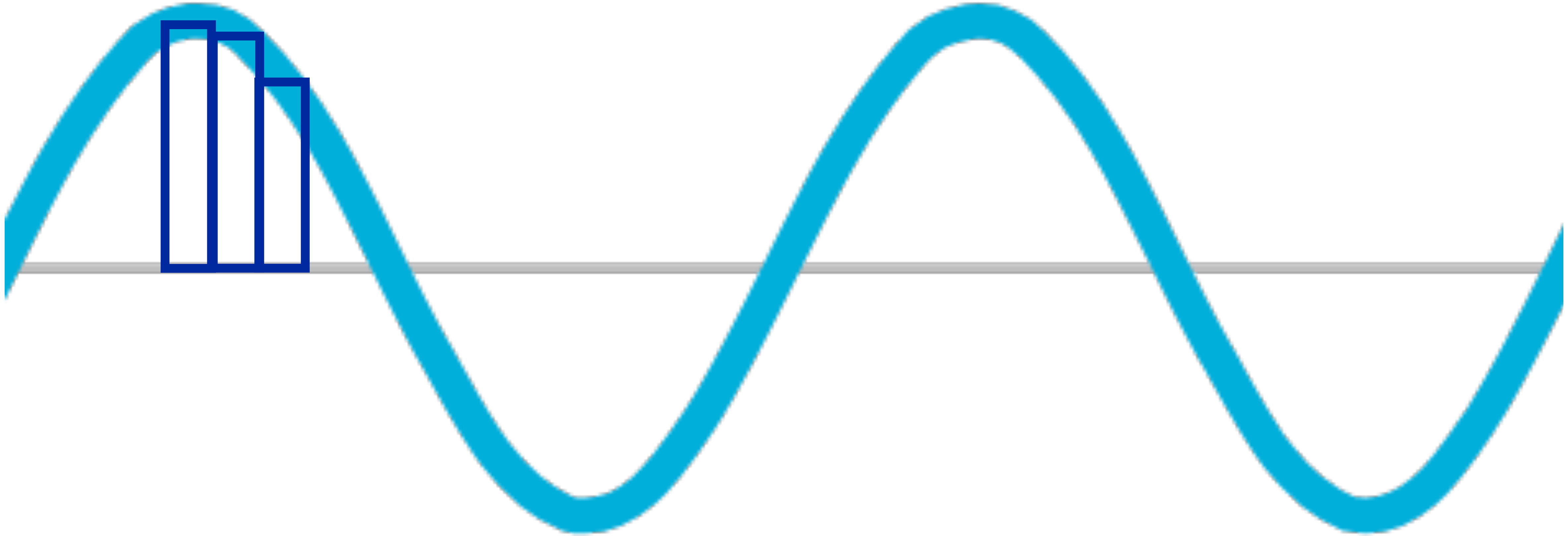


@damncabbage

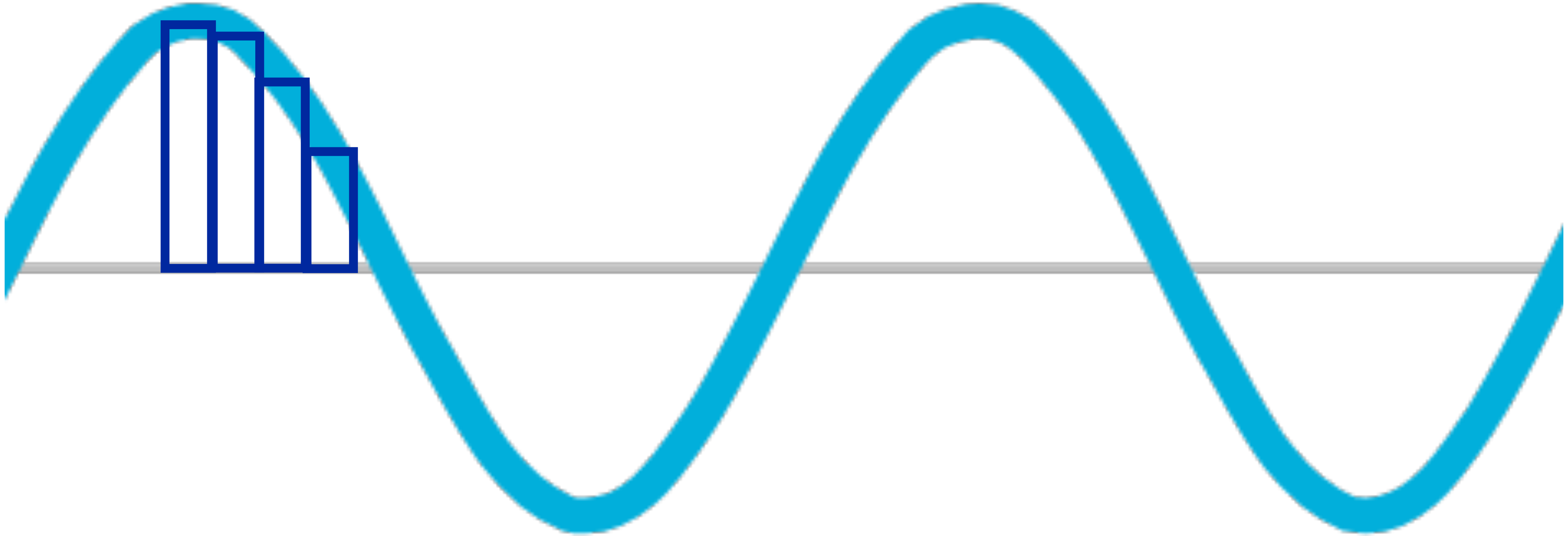


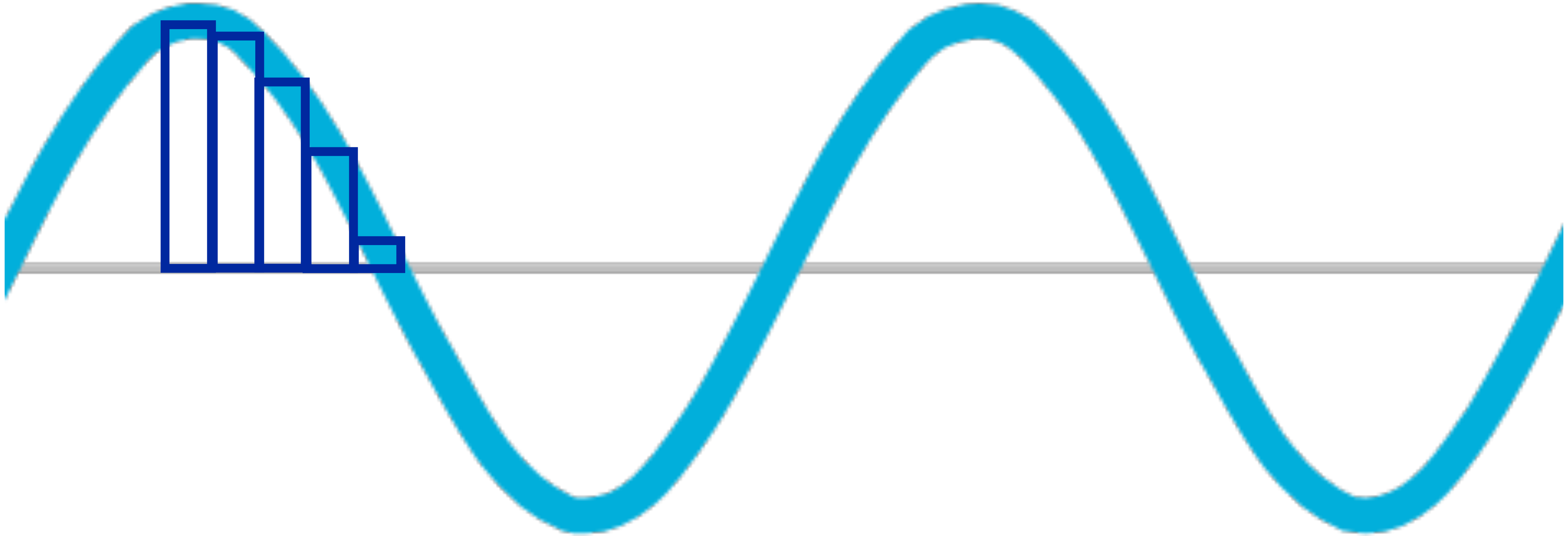
@damncabbage

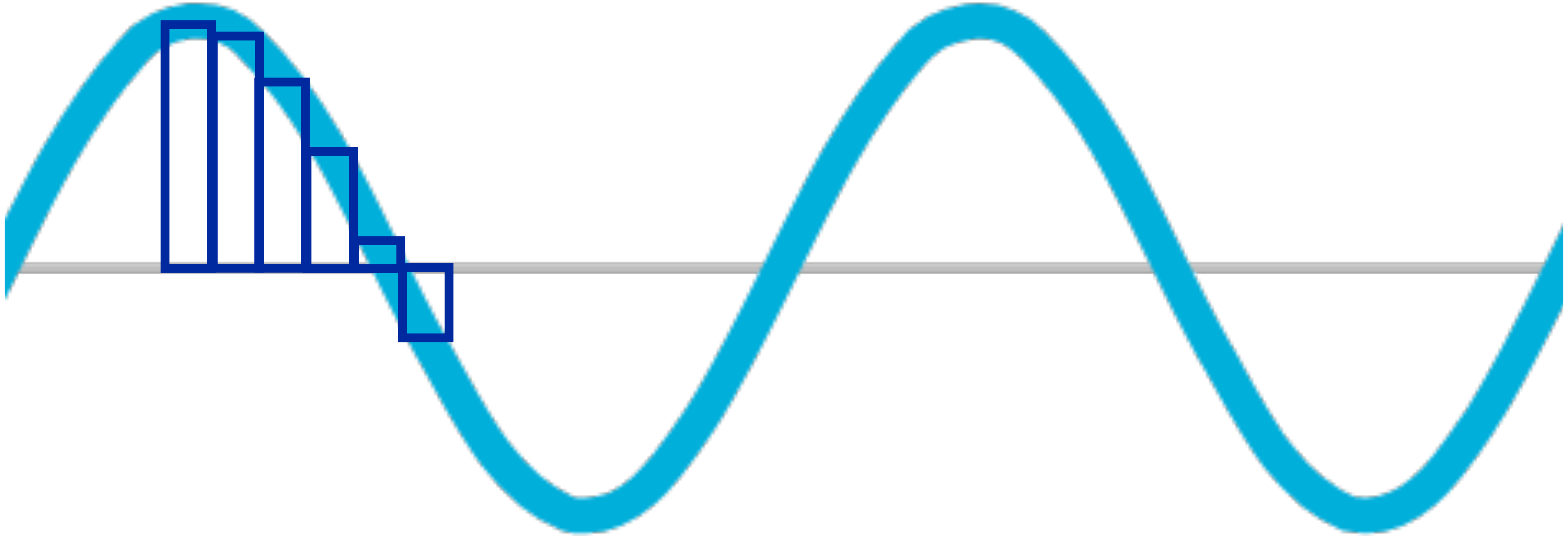




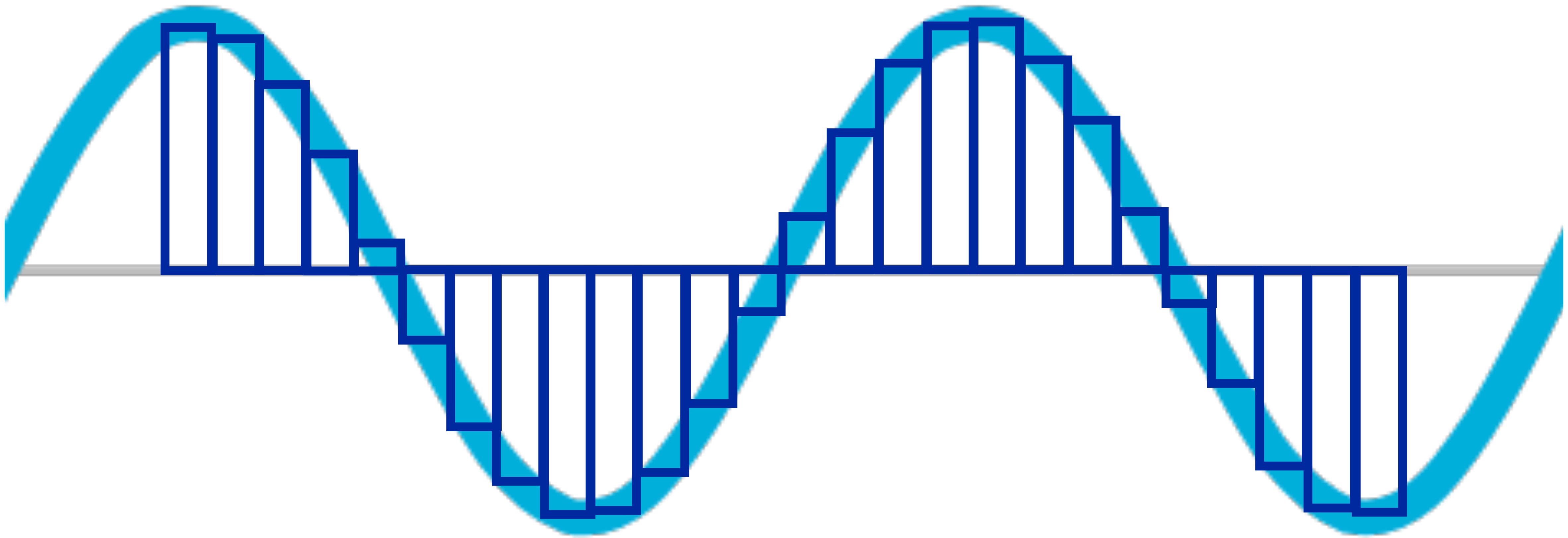
@damncabbage



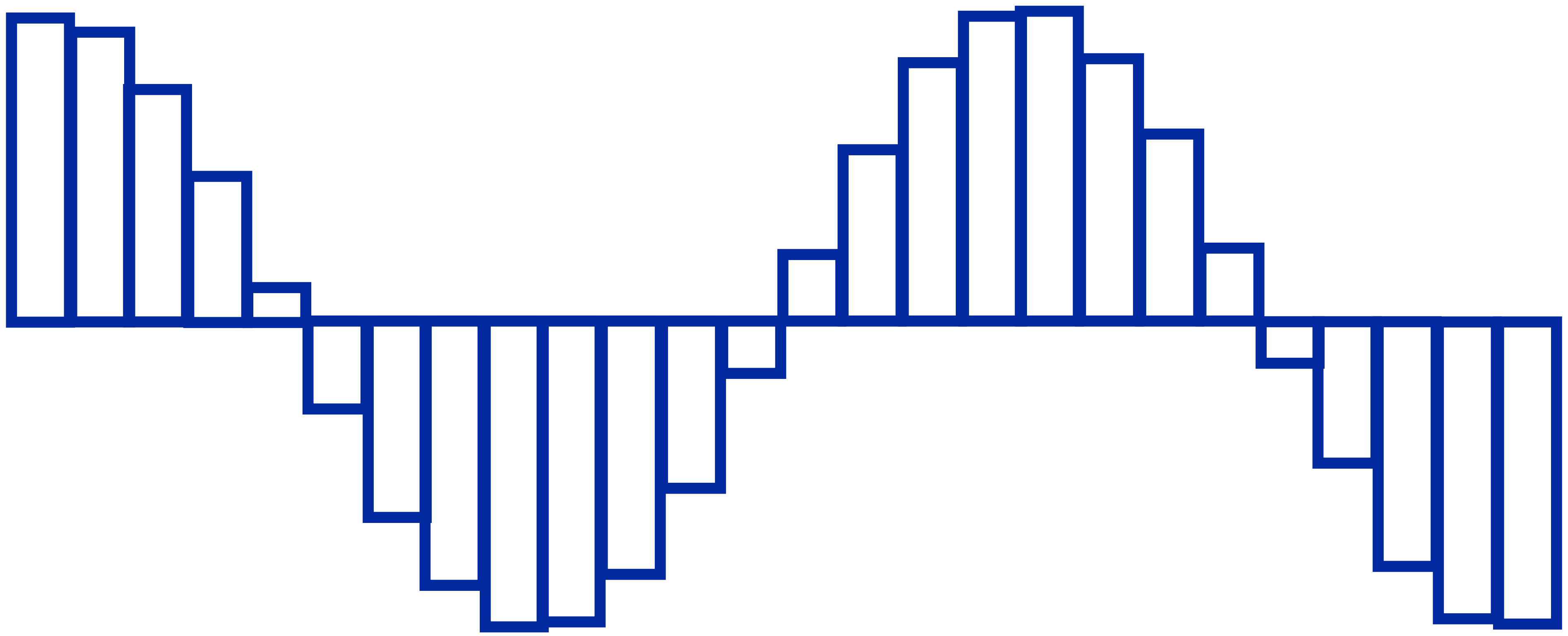




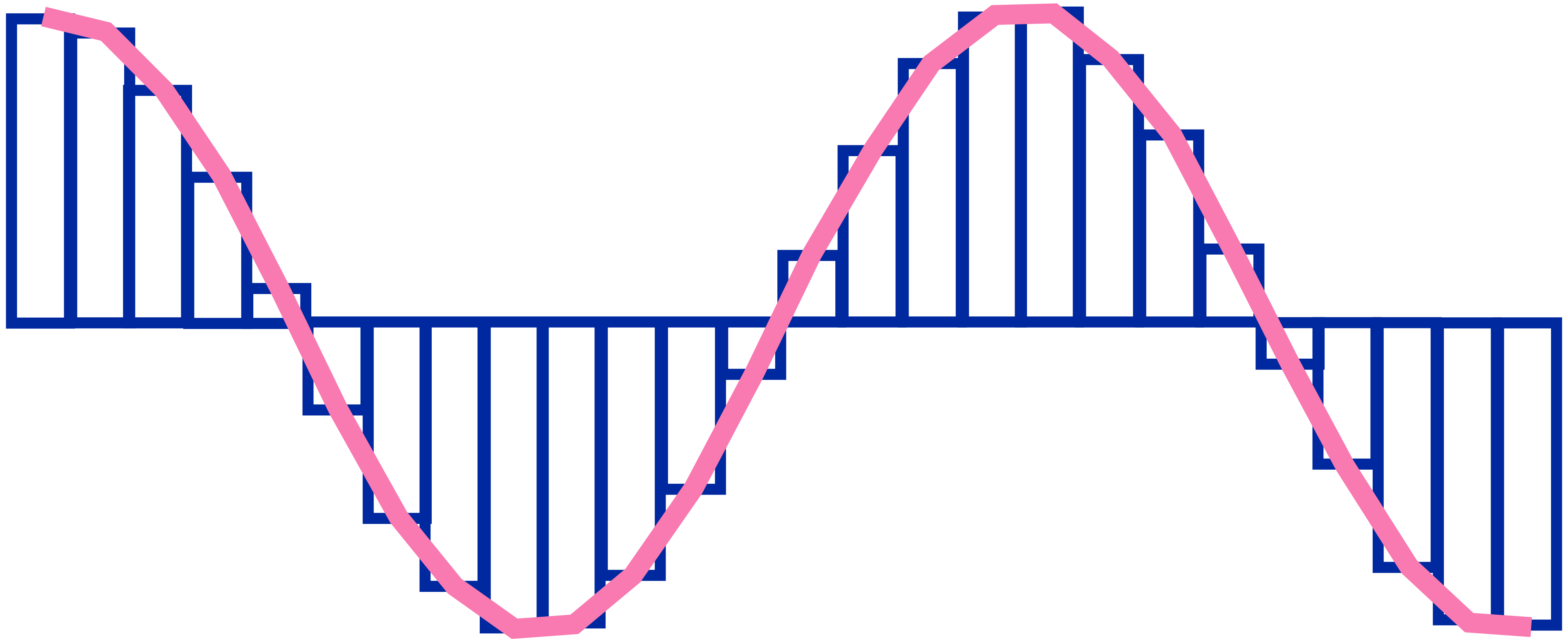
@damncabbage



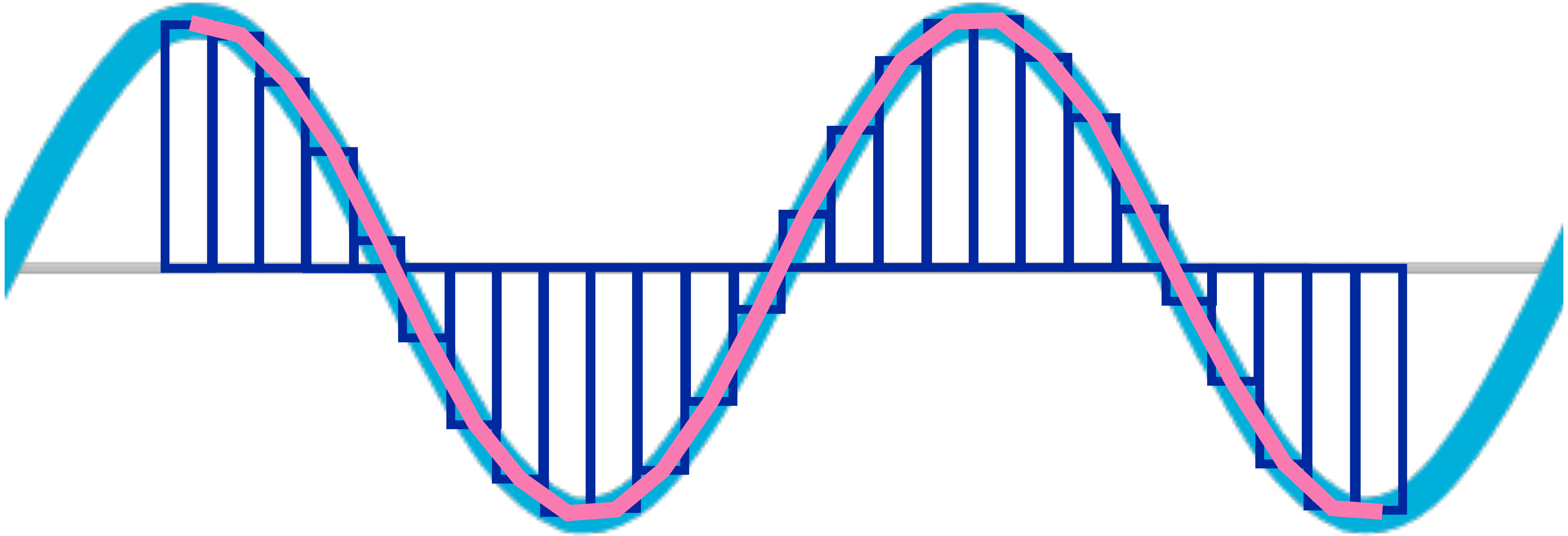
@damncabbage

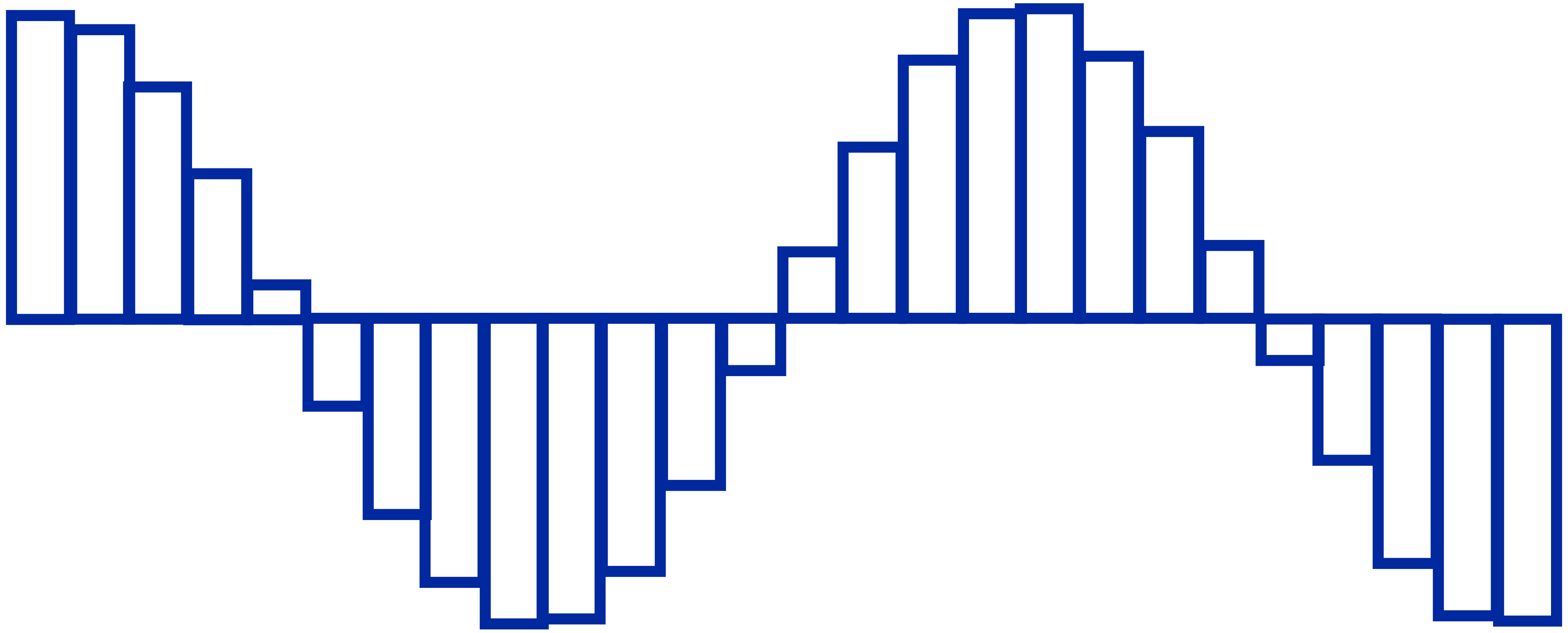


@damncabbage

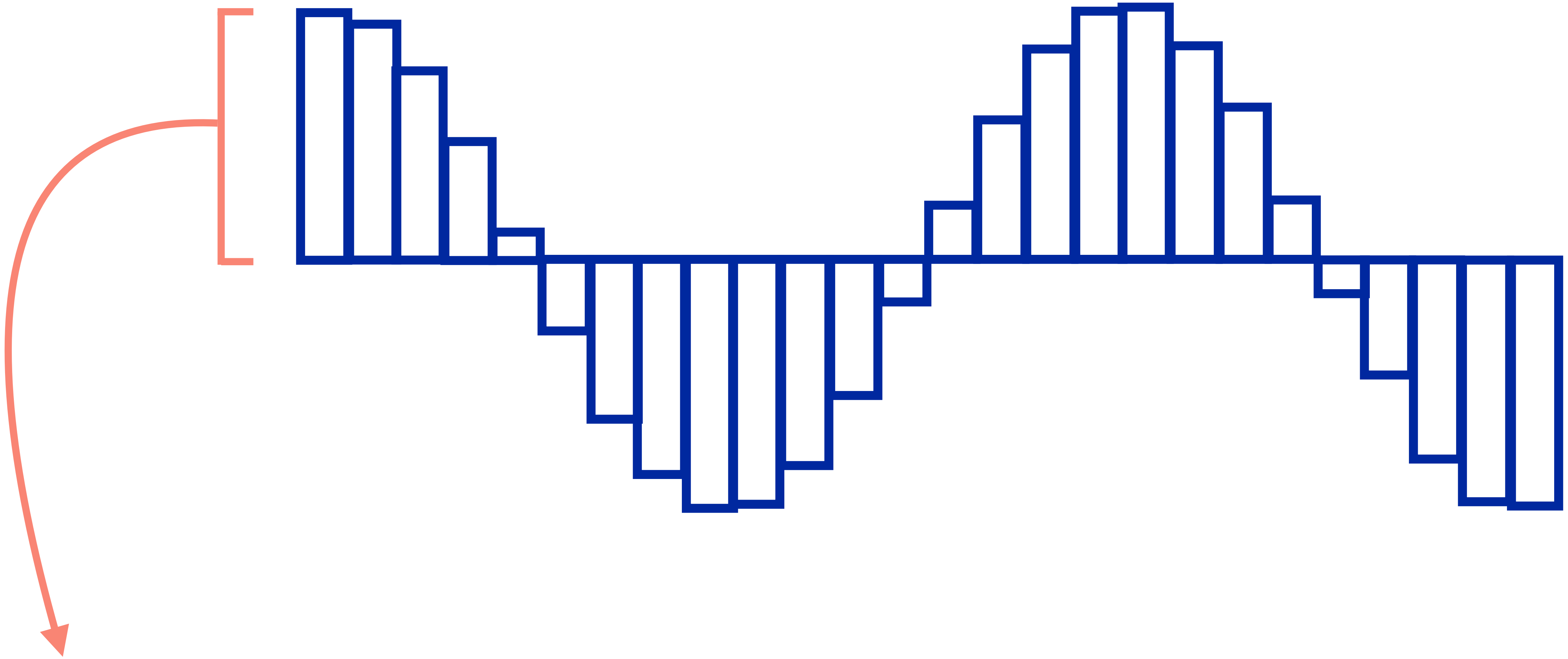


@damncabbage

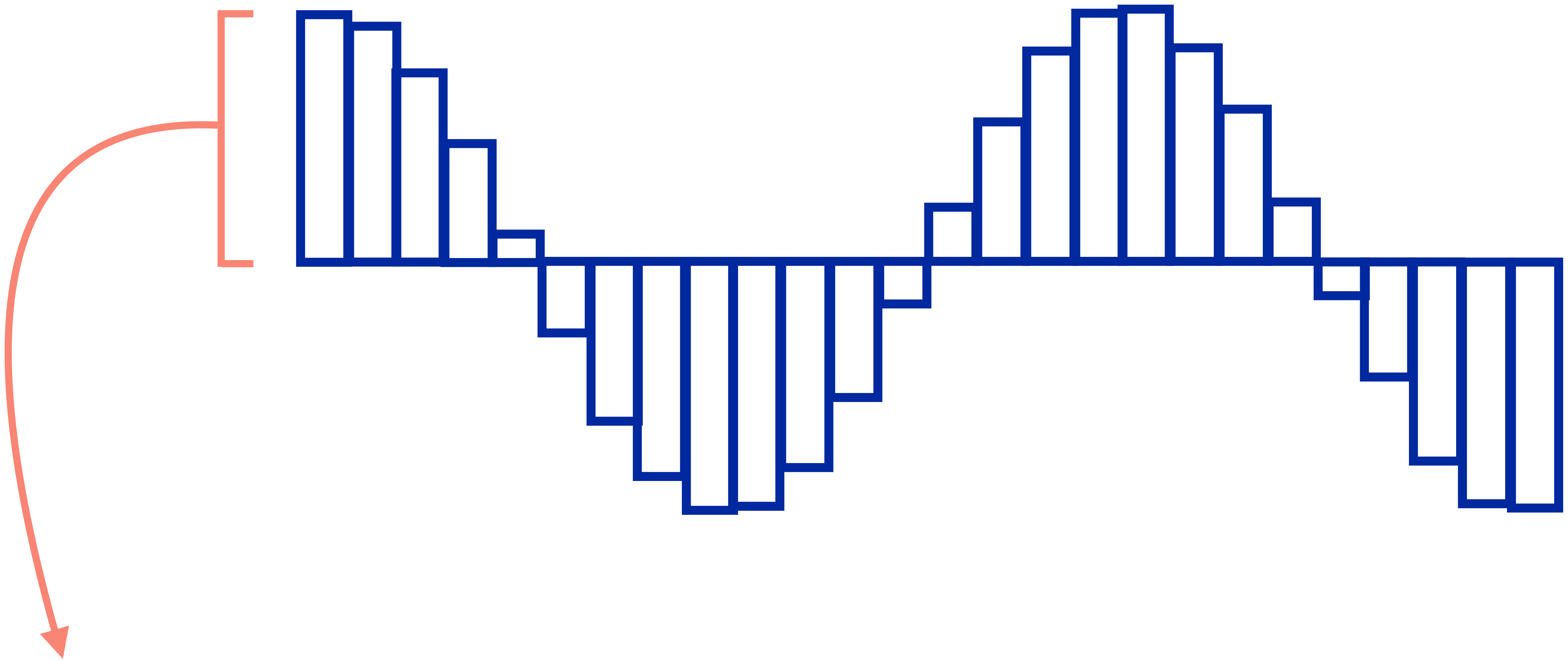




@damncabbage

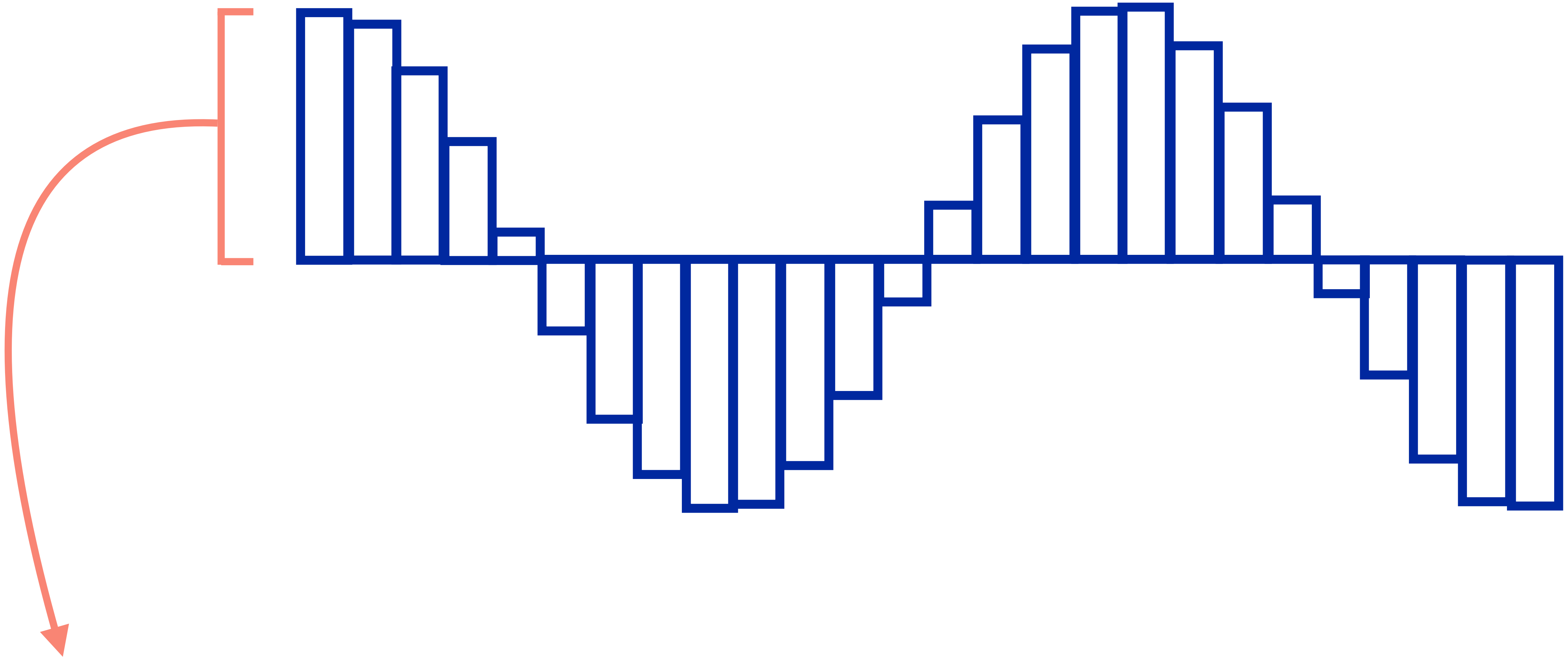


@damncabbage

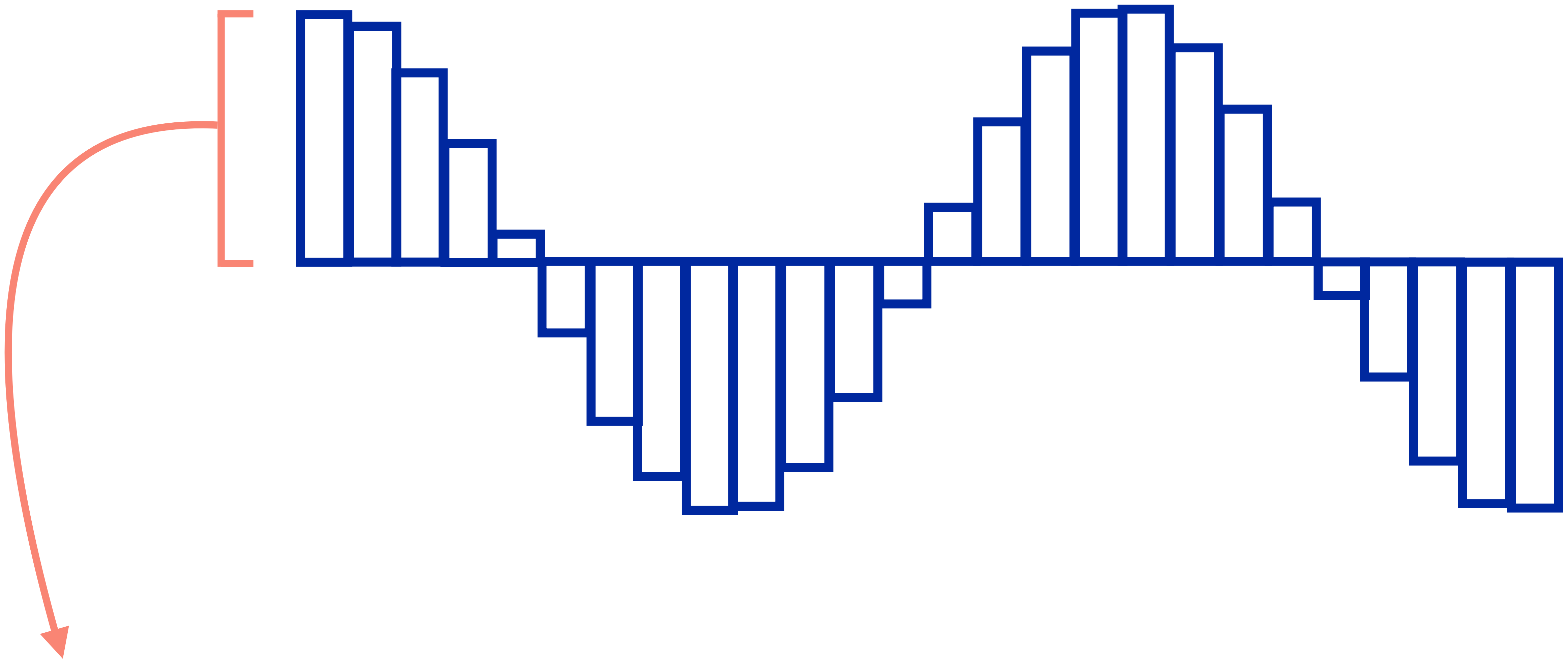


1,

@damncabbage

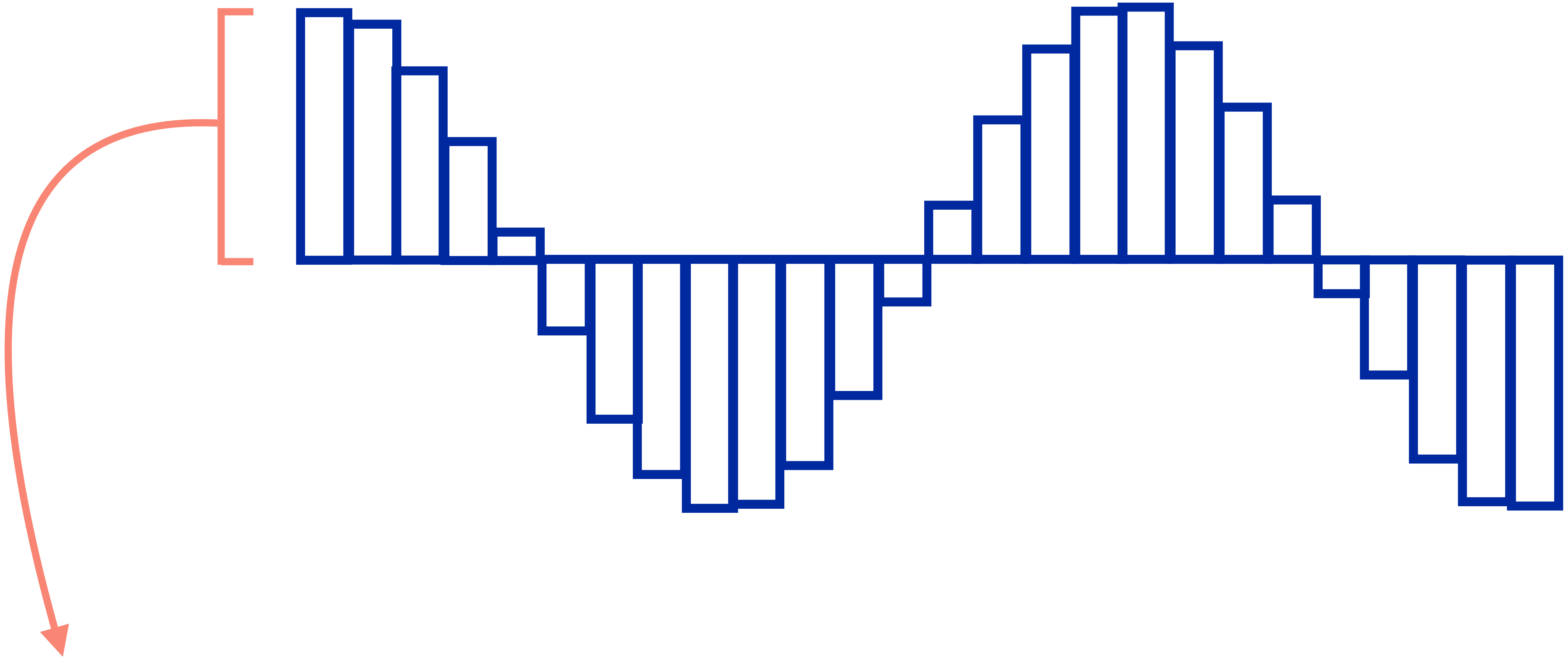


1, 0.97,



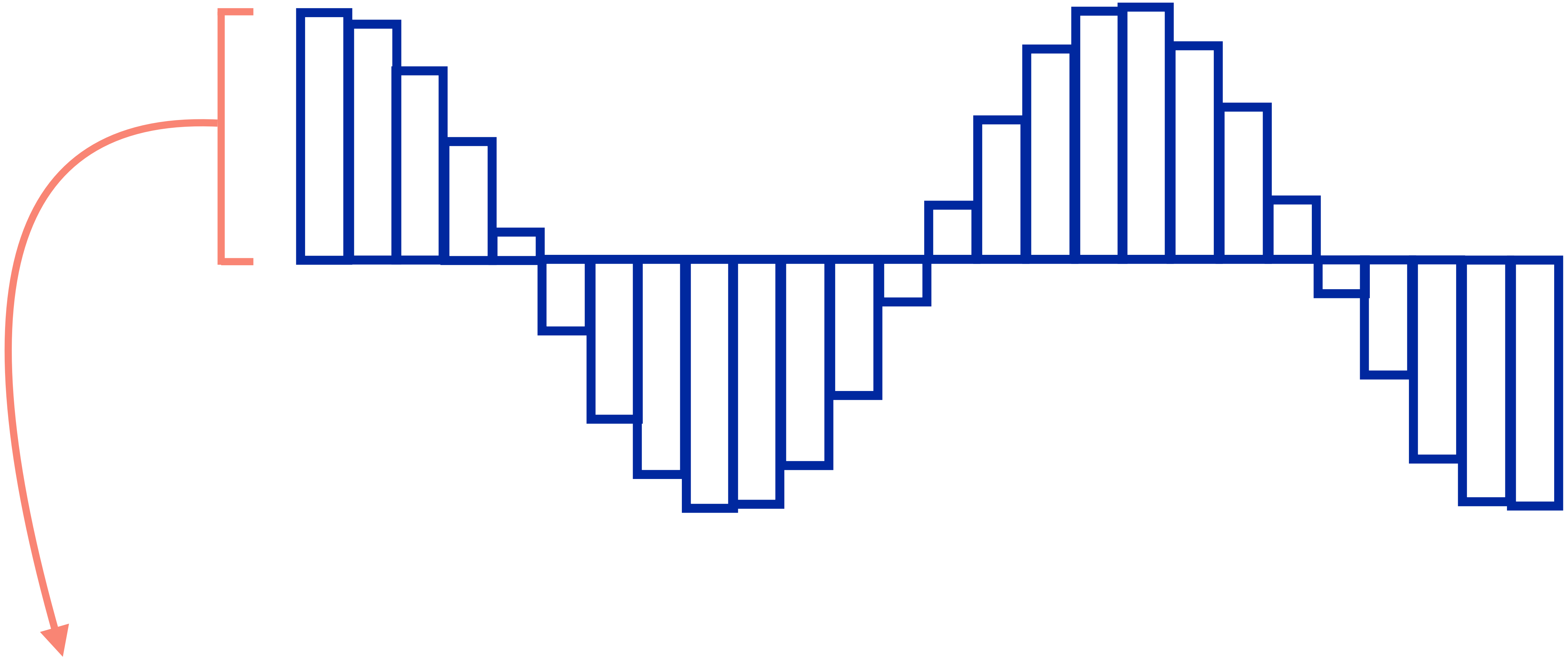
```
1, 0.97, 0.76,
```

@damncabbage



1, 0.97, 0.76, 0.5,

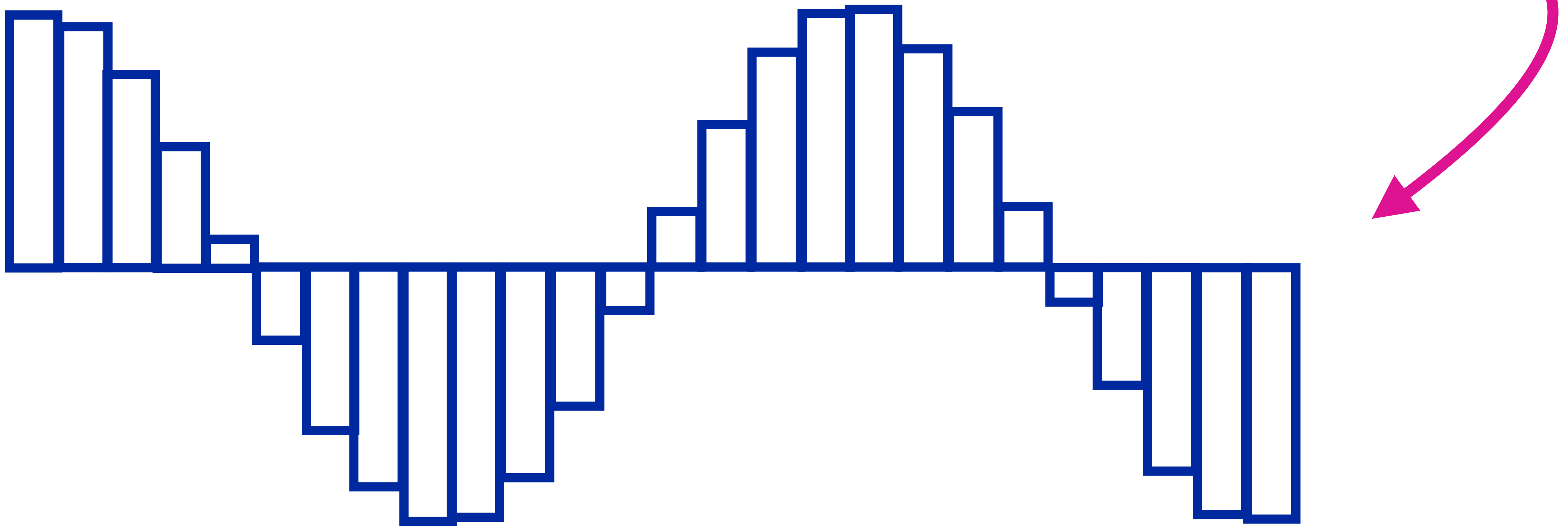
@damncabbage



1, 0.97, 0.76, 0.5, 0.2, -0.3, -0.7, -0.85, -1, -0.99, ...

THAT'S NUMBER-WAV

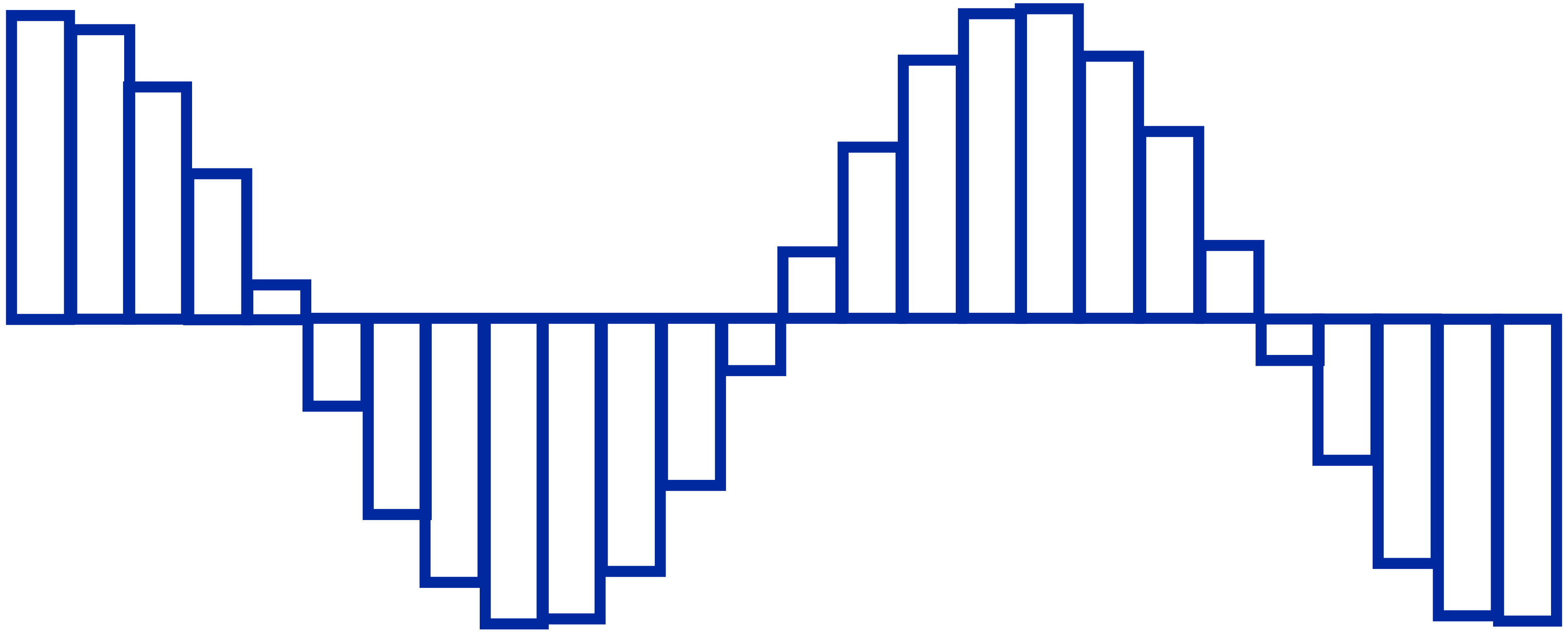
Sample rate: the number of these slices per second



1, 0.97, 0.76, 0.5, 0.2, -0.3, -0.7, -0.85, -1, -0.99, ...

@damncabbage

Sample rate: the number of these slices per second



1, 0.97, 0.76, 0.5, 0.2, -0.3, -0.7, -0.85, -1, -0.99, ...

Bit depth: the potential range (resolution) of these numbers

@damncabbage

Audacity

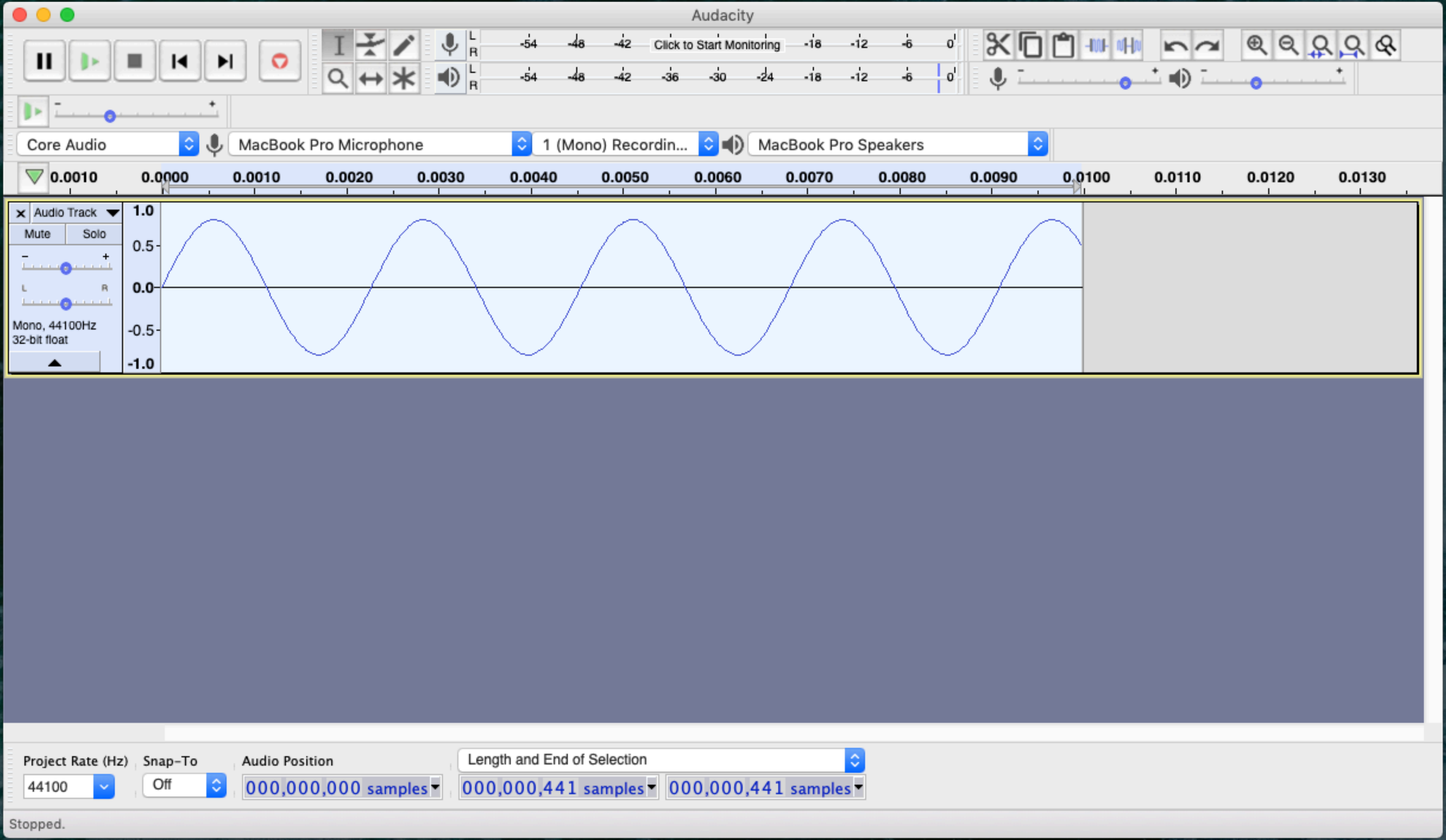
Audacity

or

Audacity

or

Rogue Amoeba's Fission



Core Audio

MacBook Pro Microphone

1 (Mono) Recordin...

MacBook Pro Speakers

Mono, 44100Hz
32-bit float

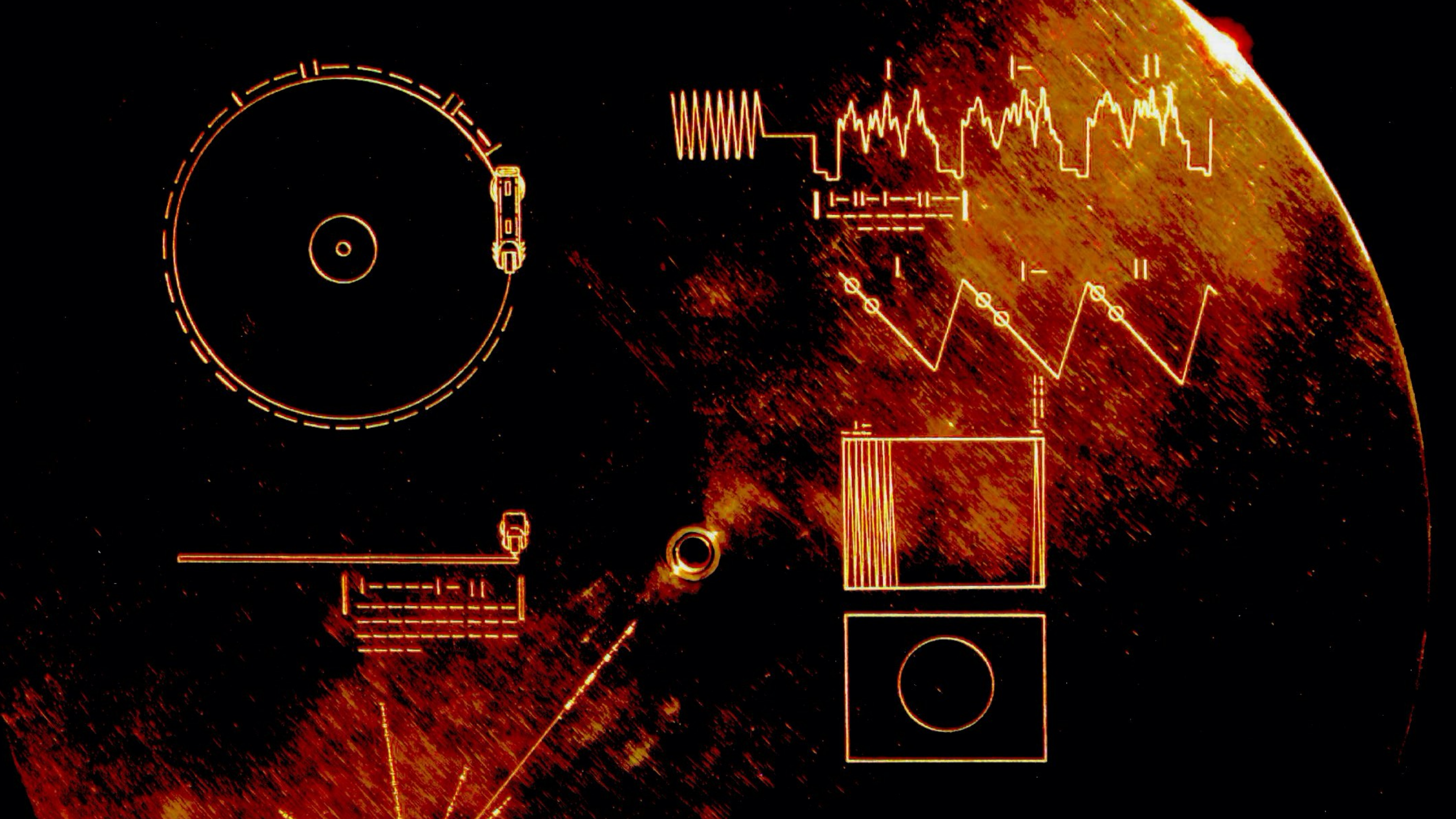
Project Rate (Hz): 44100

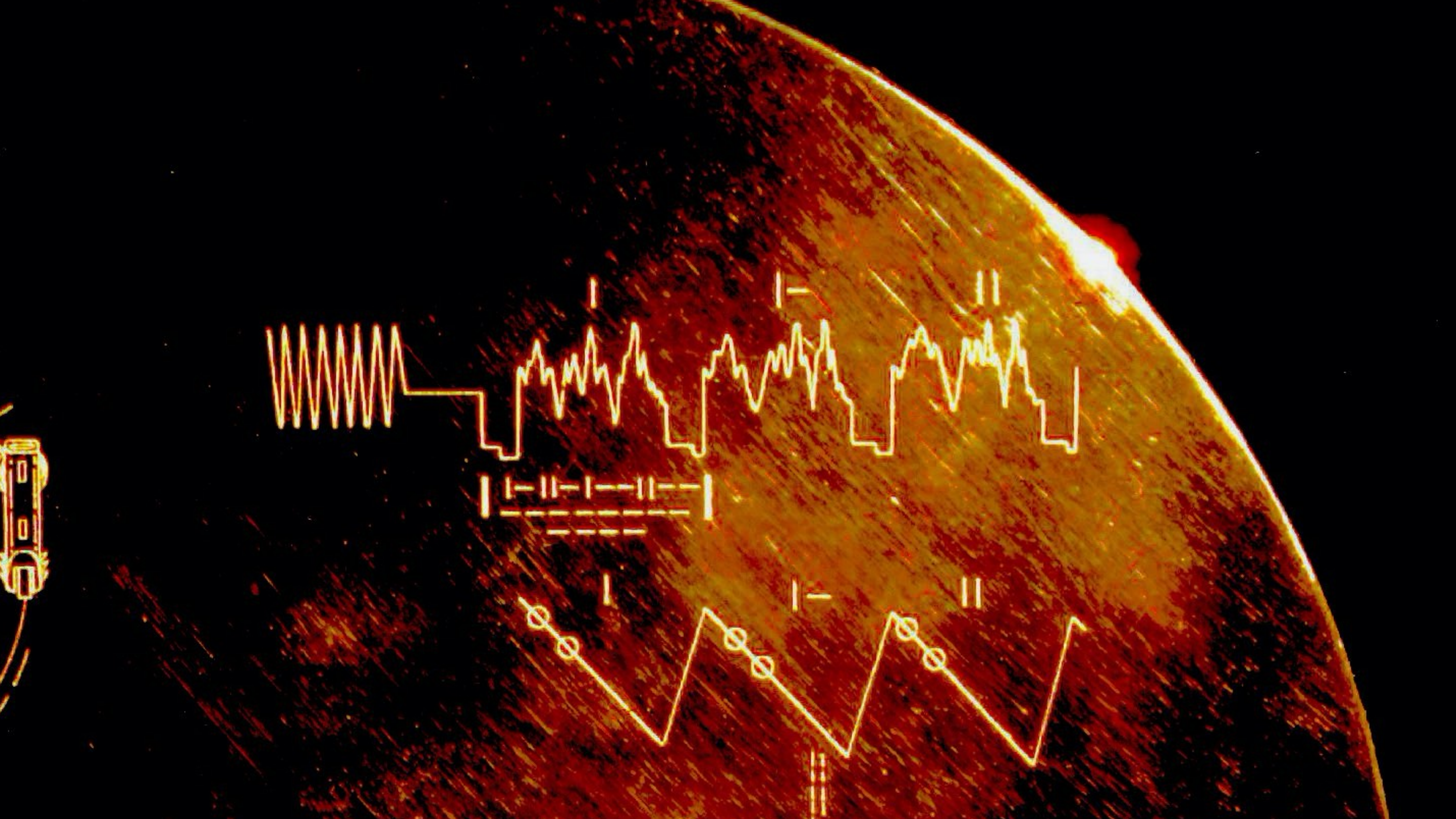
Snap-To: Off

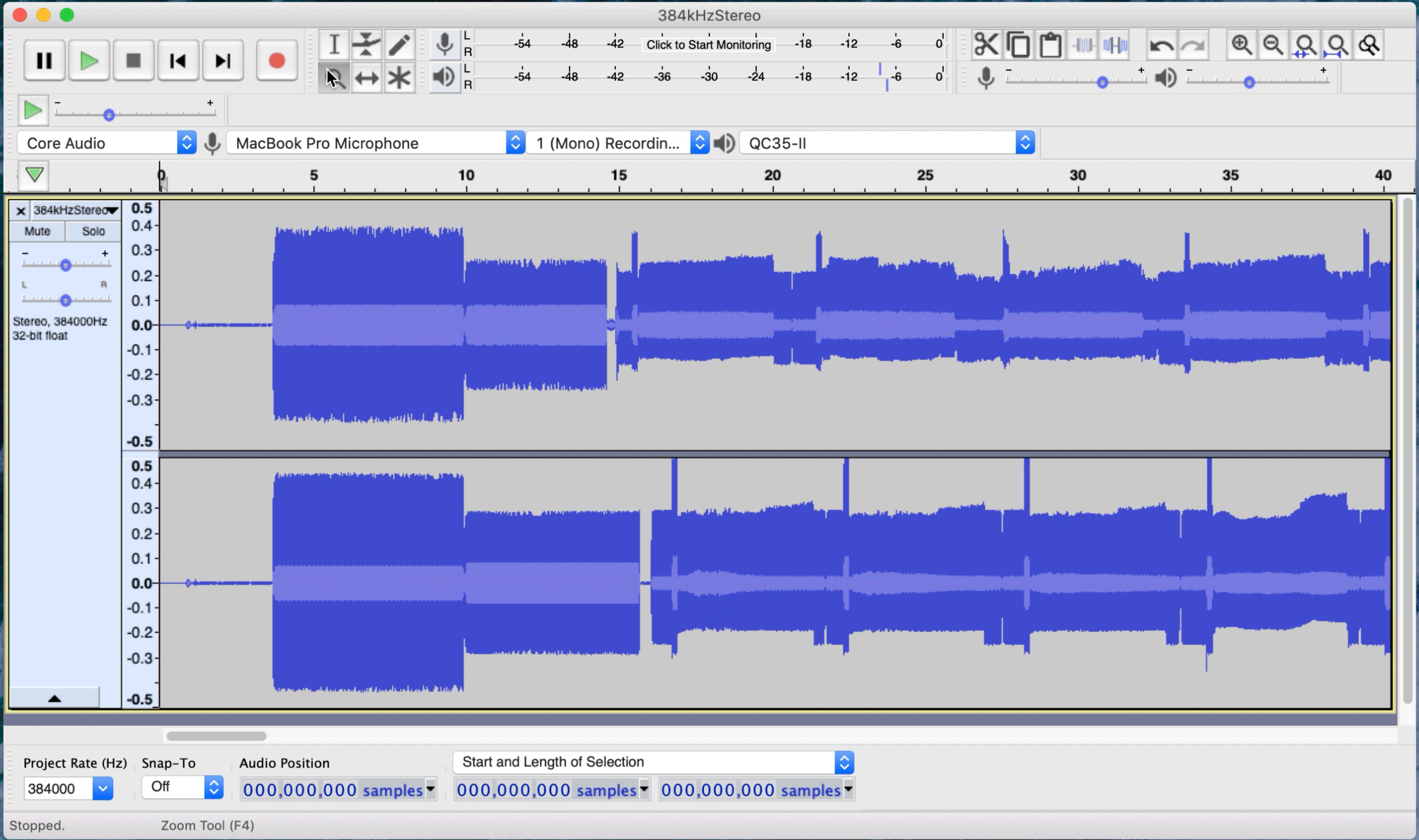
Audio Position: 000,000,000 samples

Length and End of Selection: 000,000,441 samples 000,000,441 samples

Stopped.







384kHzStereo

Core Audio

MacBook Pro Microphone

1 (Mono) Recordin...

QC35-II

Stereo, 384000Hz
32-bit float

Project Rate (Hz): 384000

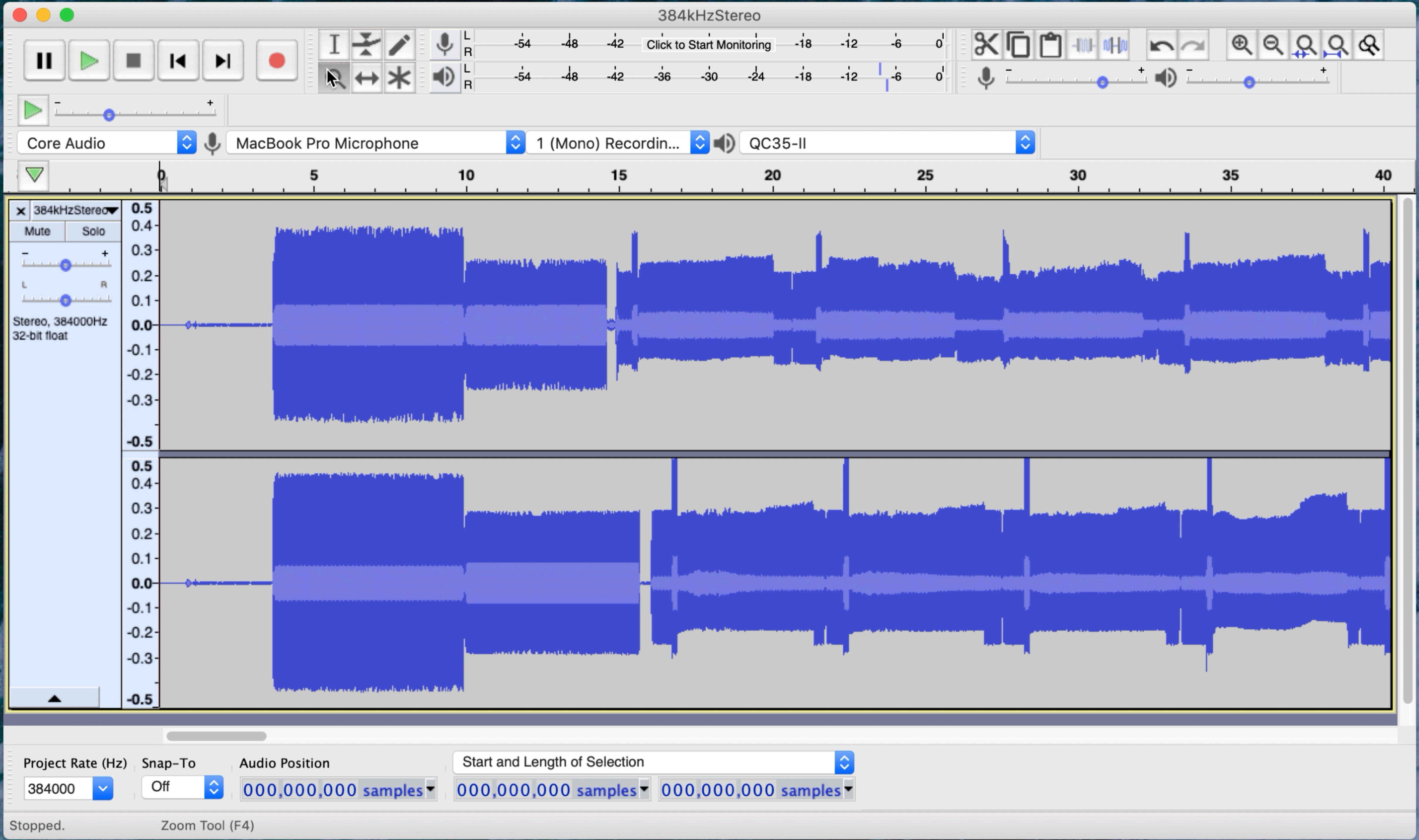
Snap-To: Off

Audio Position: 000,000,000 samples

Start and Length of Selection: 000,000,000 samples 000,000,000 samples

Stopped.

Zoom Tool (F4)

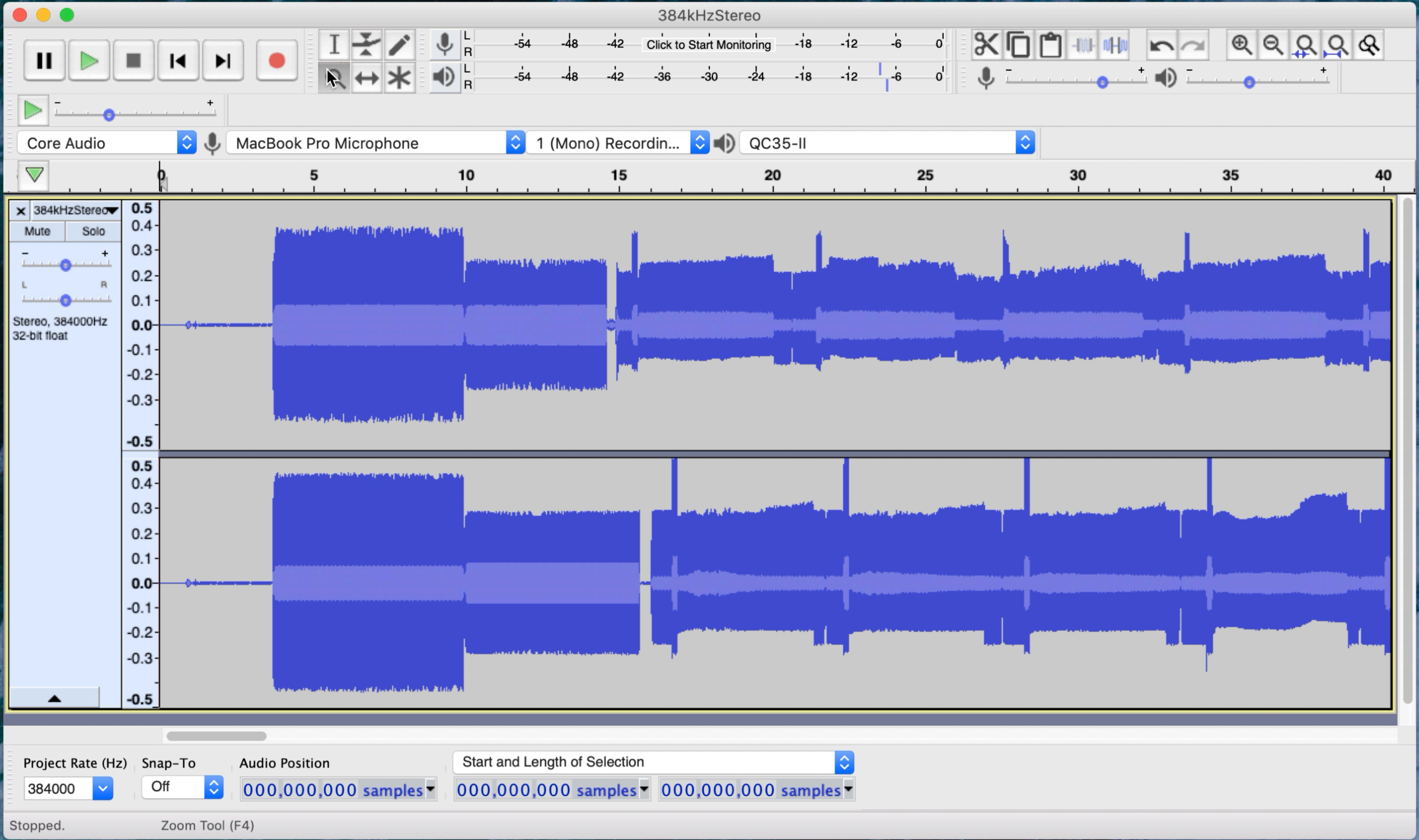


Core Audio
MacBook Pro Microphone
1 (Mono) Recordin...
QC35-II

384kHzStereo
Stereo, 384000Hz
32-bit float

Project Rate (Hz) 384000
Snap-To Off
Audio Position 000,000,000 samples
Start and Length of Selection 000,000,000 samples 000,000,000 samples

Stopped.
Zoom Tool (F4)



384kHzStereo

Core Audio | MacBook Pro Microphone | 1 (Mono) Recordin... | QC35-II

384kHzStereo

Stereo, 384000Hz
32-bit float

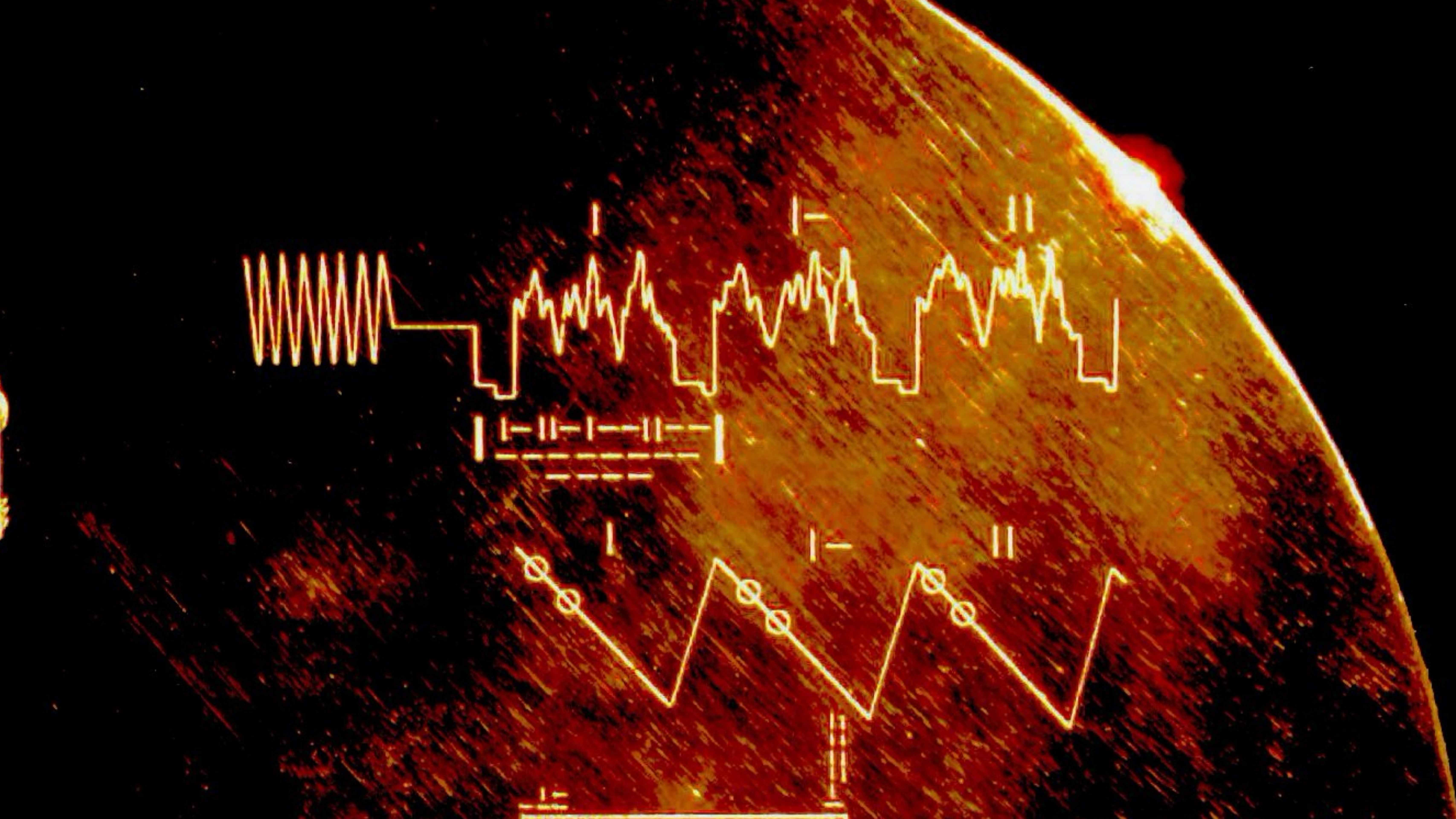
Project Rate (Hz): 384000

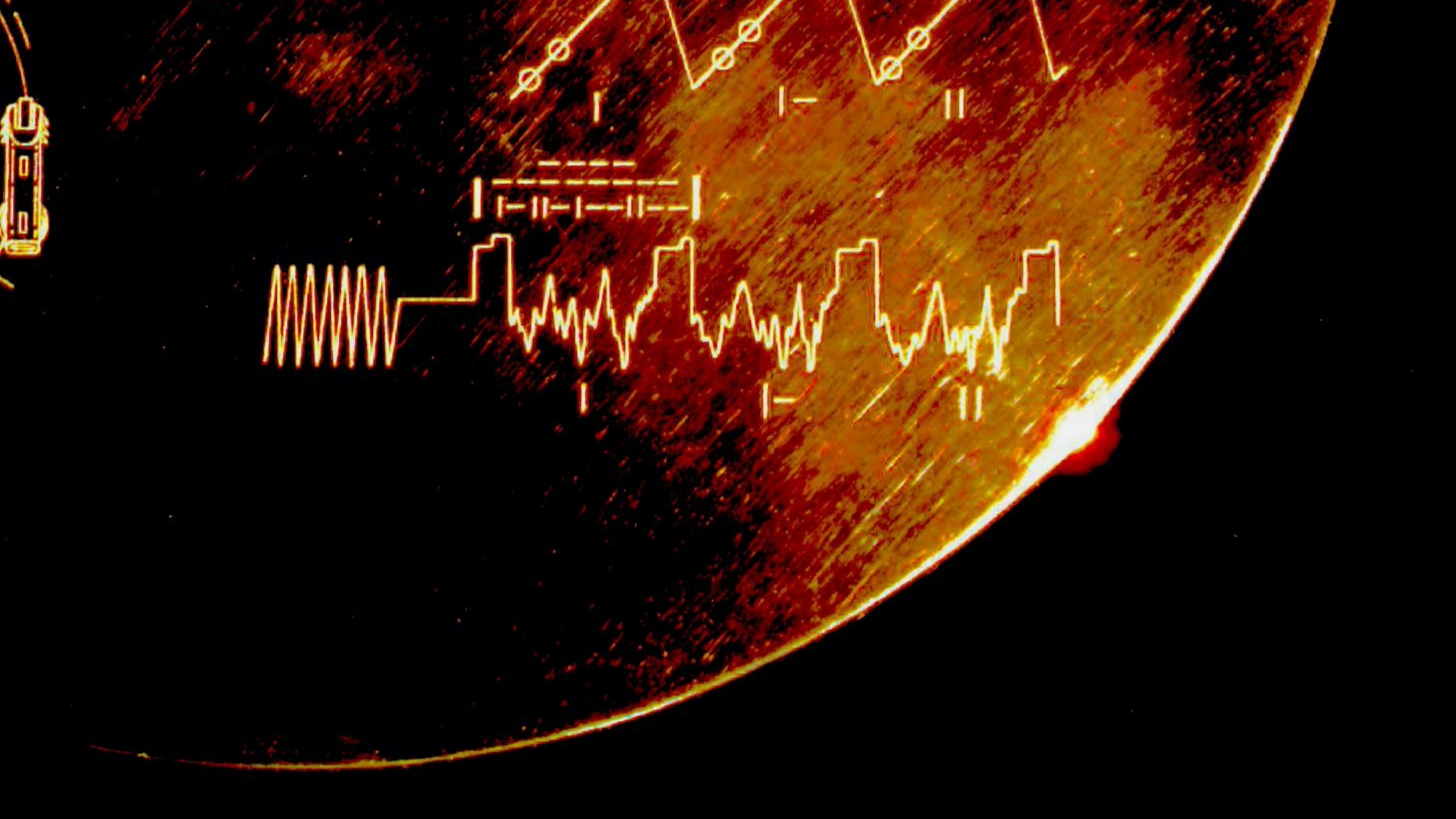
Snap-To: Off

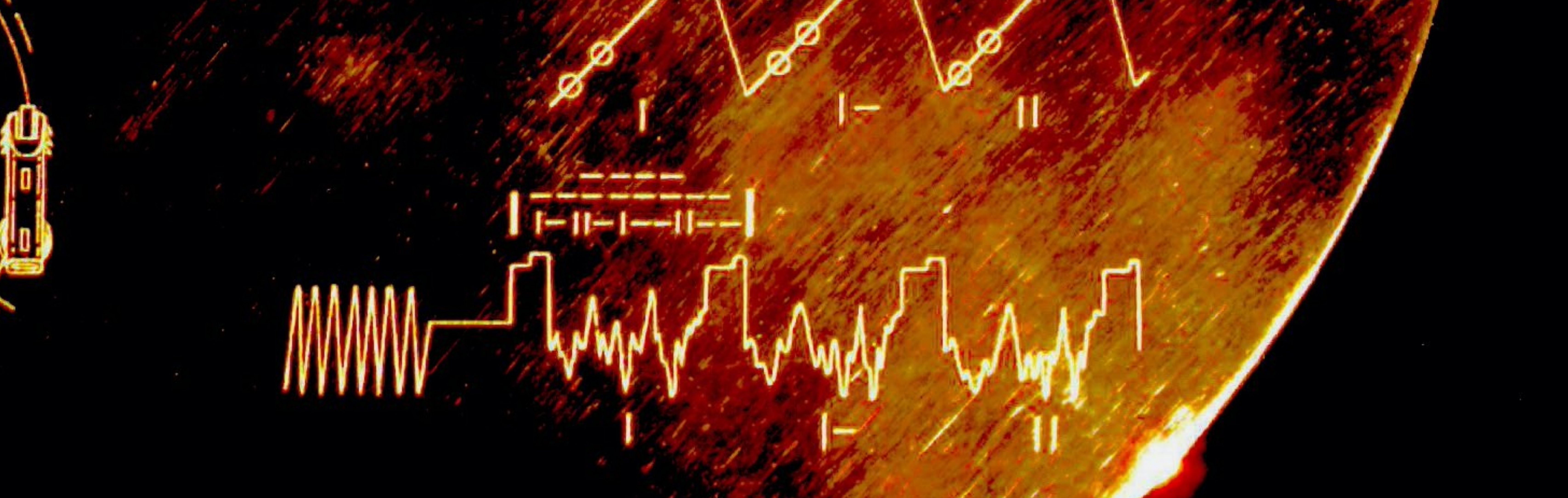
Audio Position: 000,000,000 samples

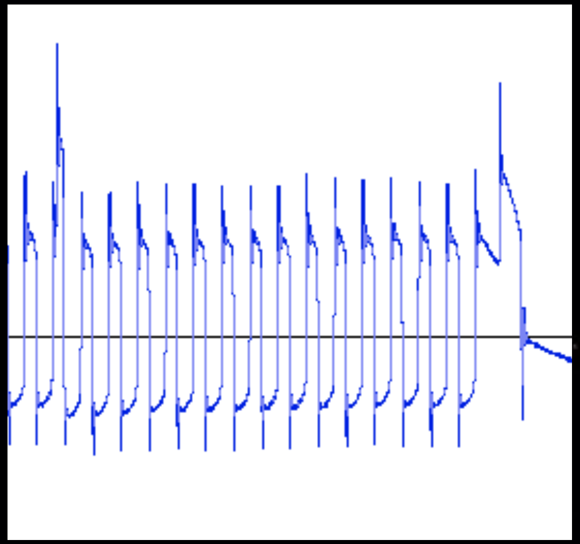
Start and Length of Selection: 000,000,000 samples 000,000,000 samples

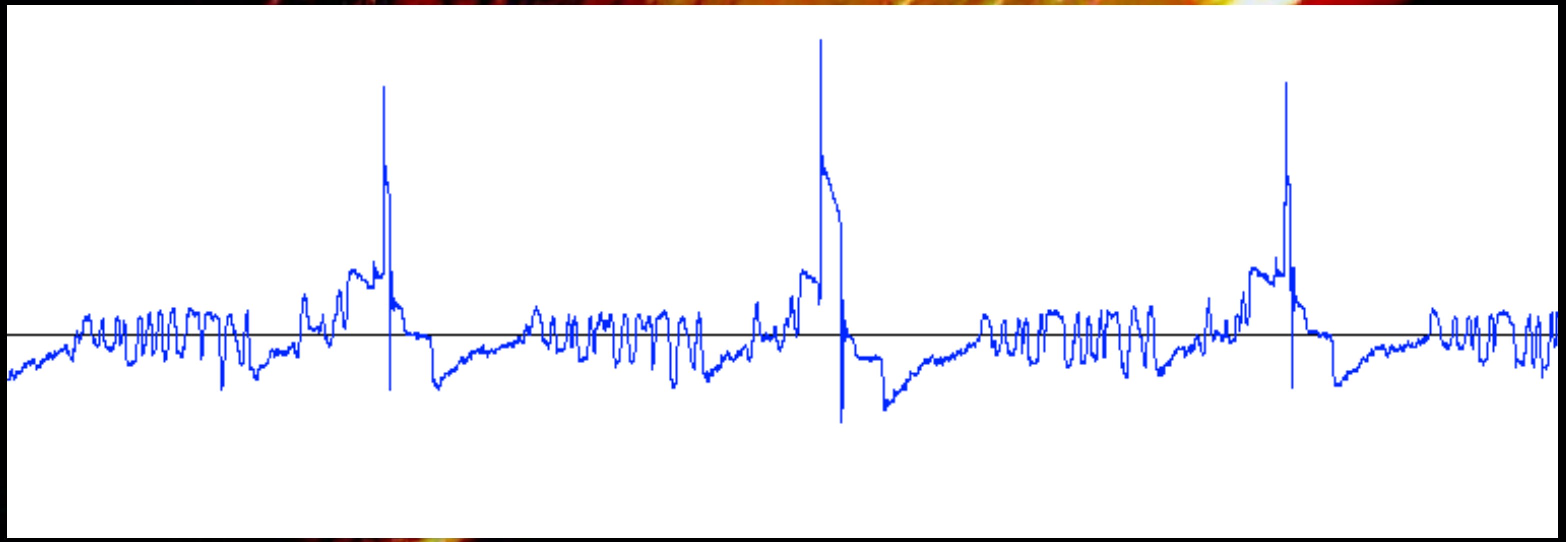
Stopped. Zoom Tool (F4)

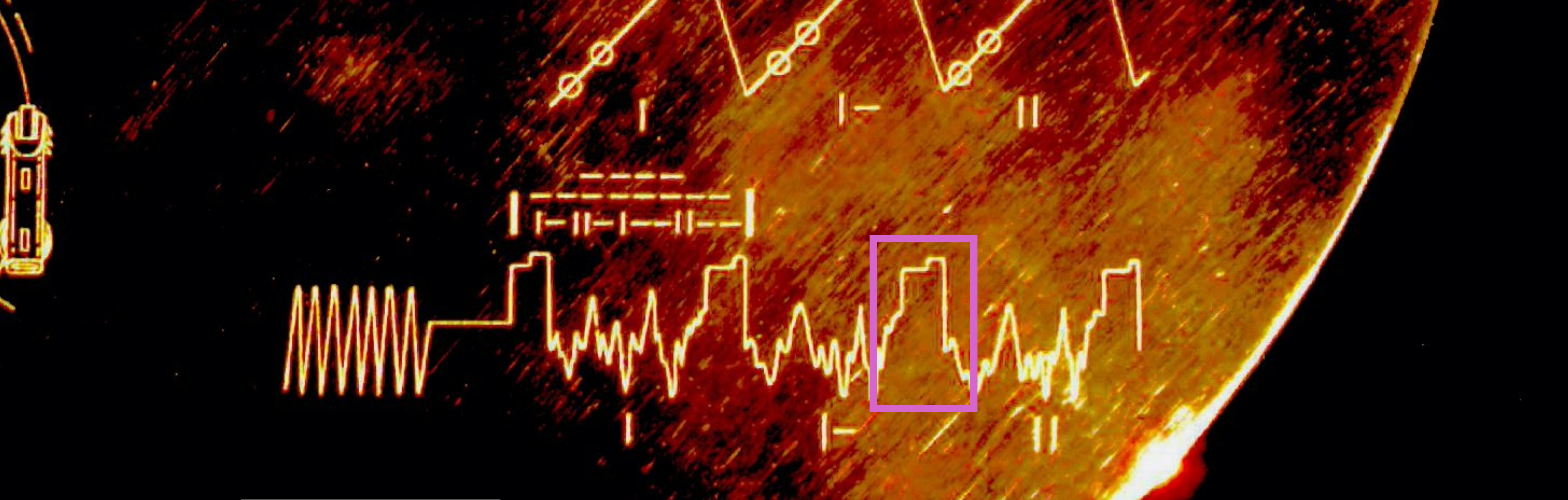




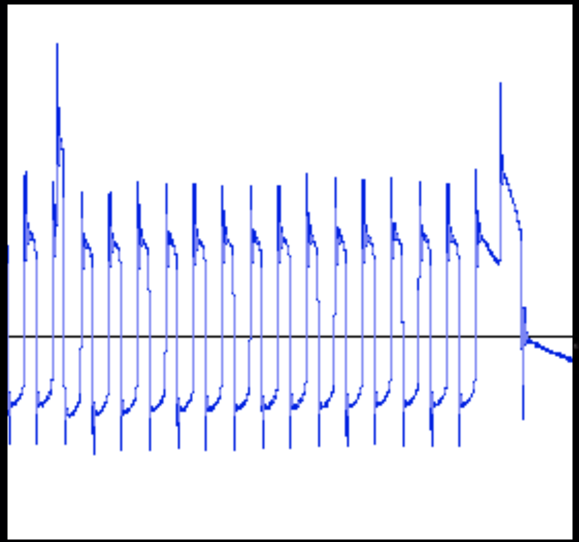


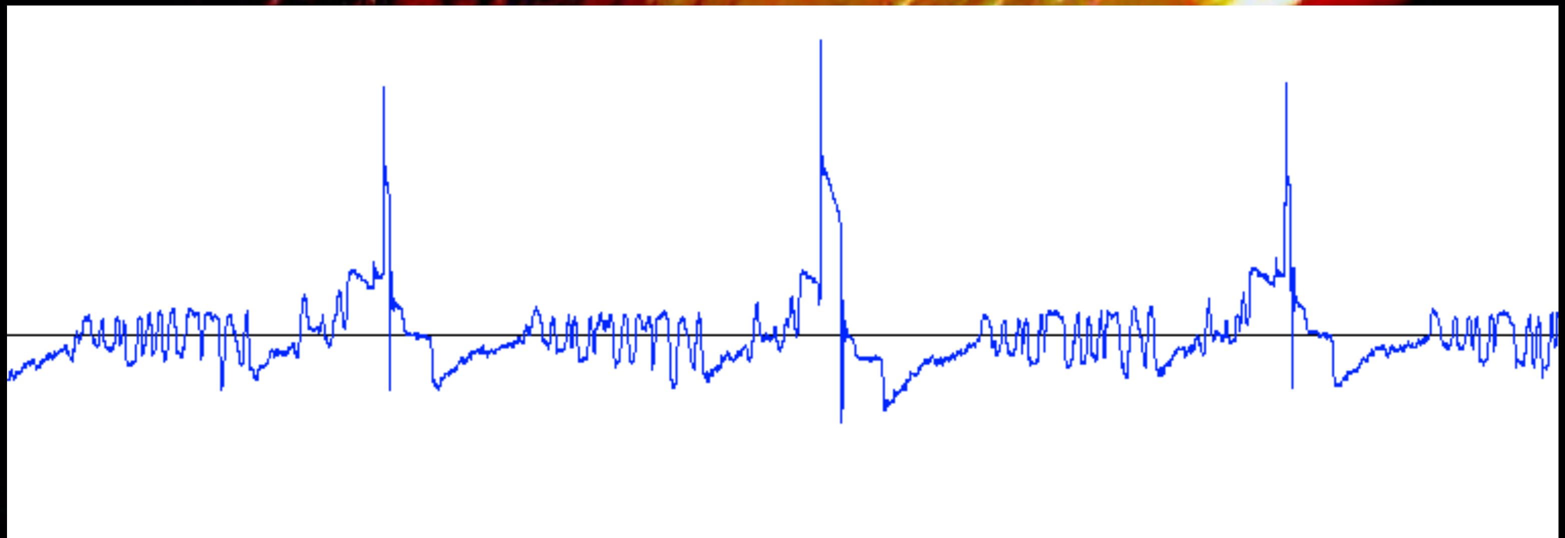


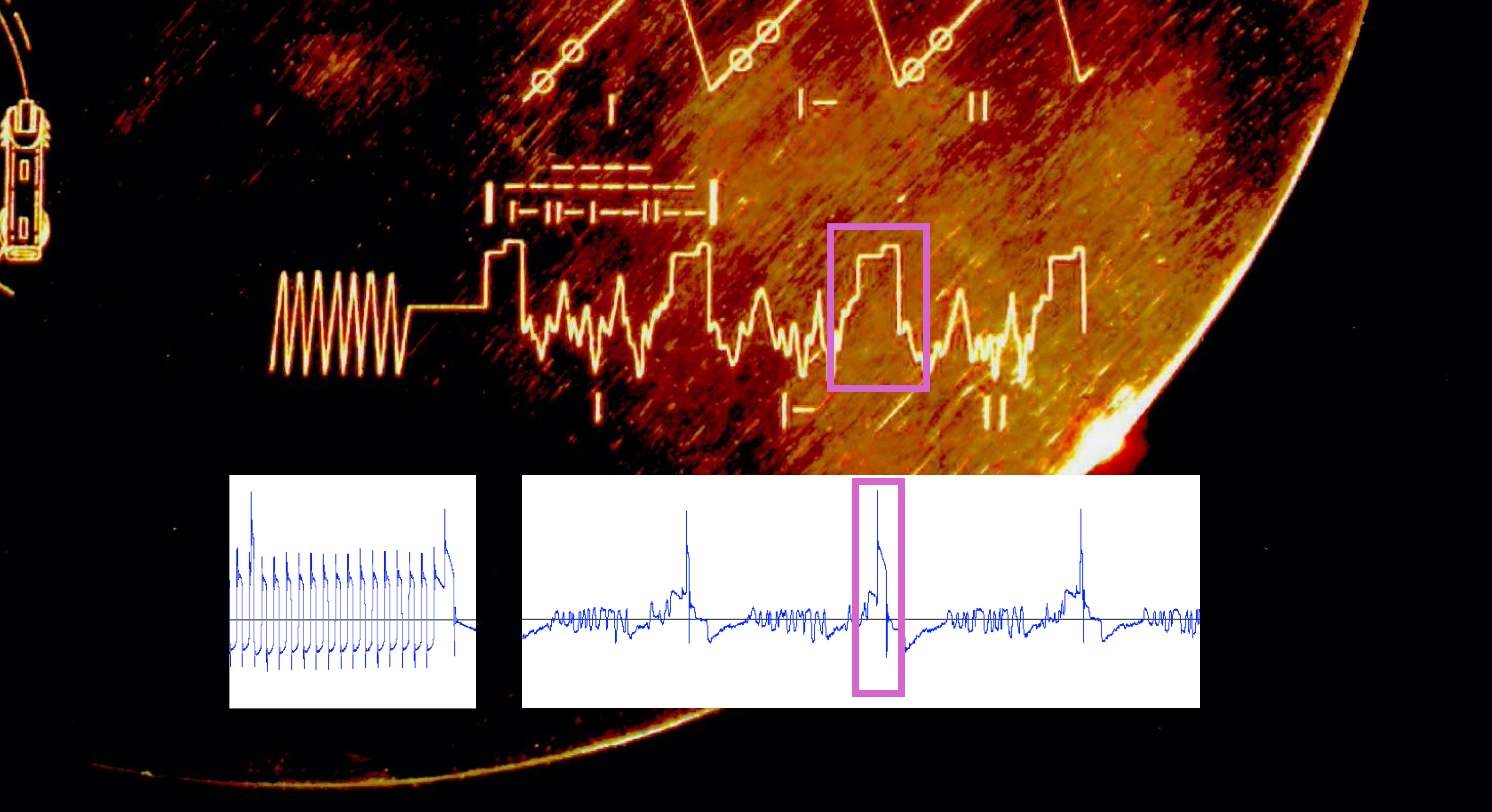


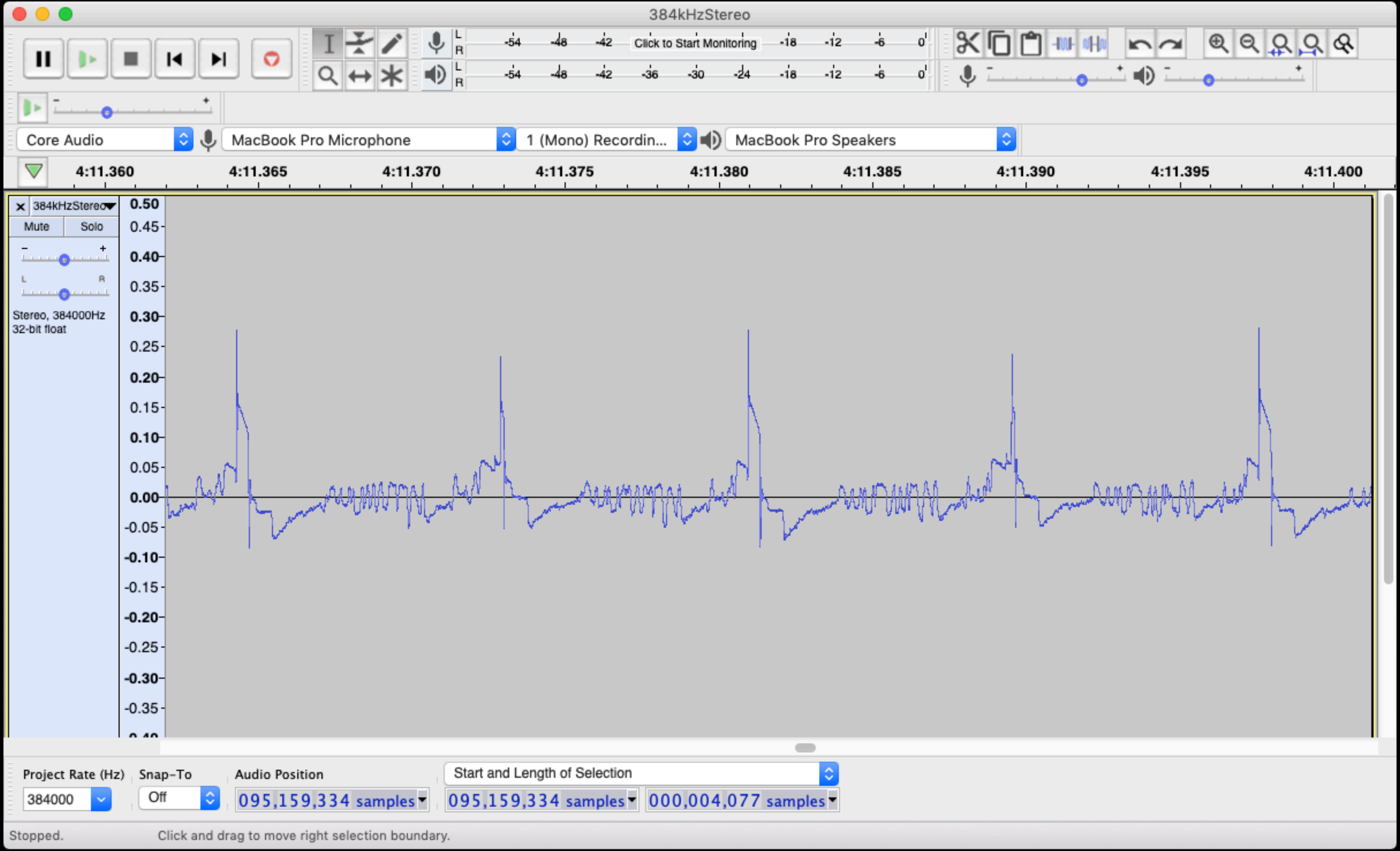


I
I–
II
I
I–
II









Core Audio

MacBook Pro Microphone

1 (Mono) Recordin...

MacBook Pro Speakers

Stereo, 384000Hz
32-bit float

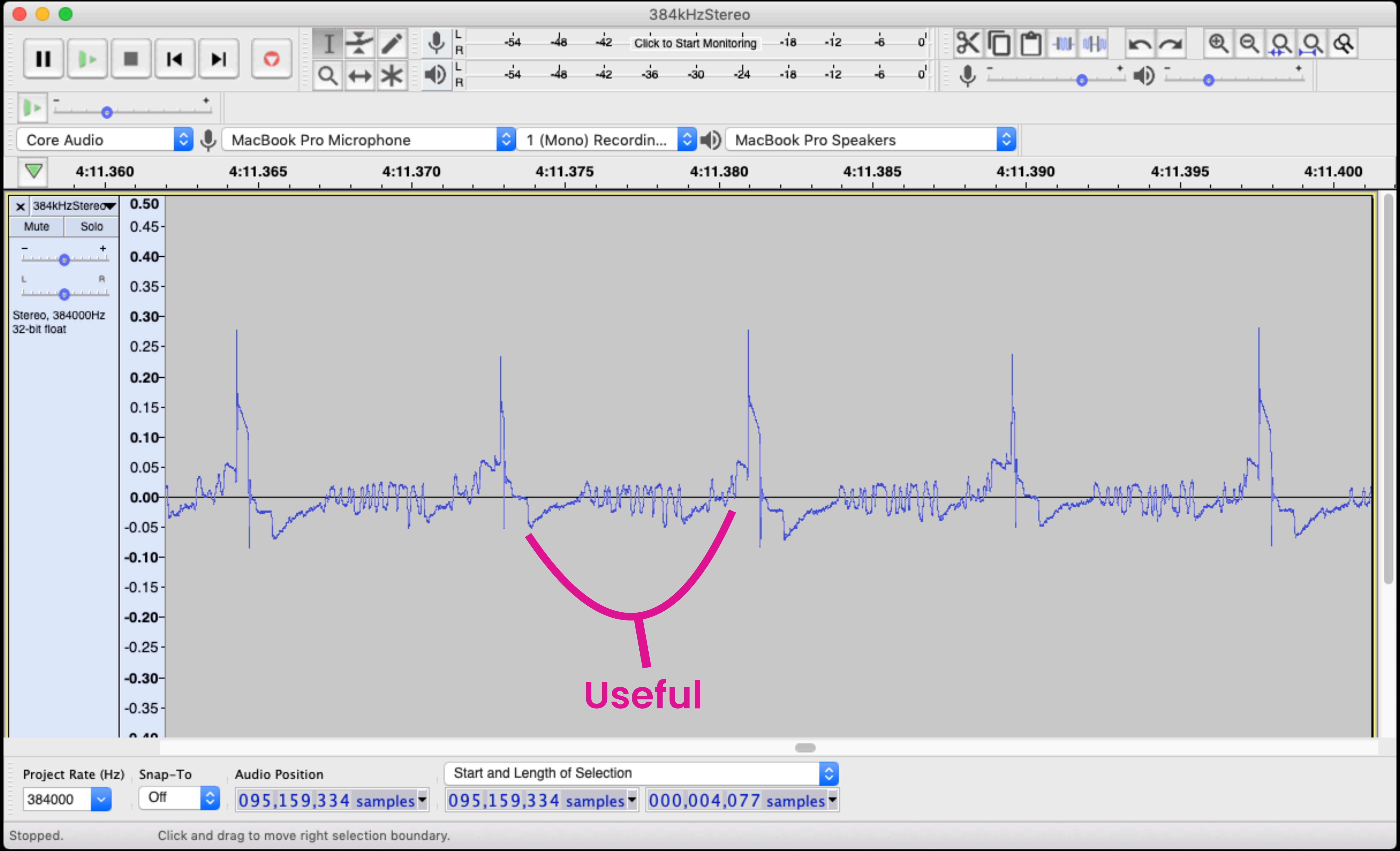
Project Rate (Hz): 384000

Snap-To: Off

Audio Position: 095,159,334 samples

Start and Length of Selection: 095,159,334 samples 000,004,077 samples

Stopped. Click and drag to move right selection boundary.



Stereo, 384000Hz
32-bit float

Useful

Project Rate (Hz): 384000

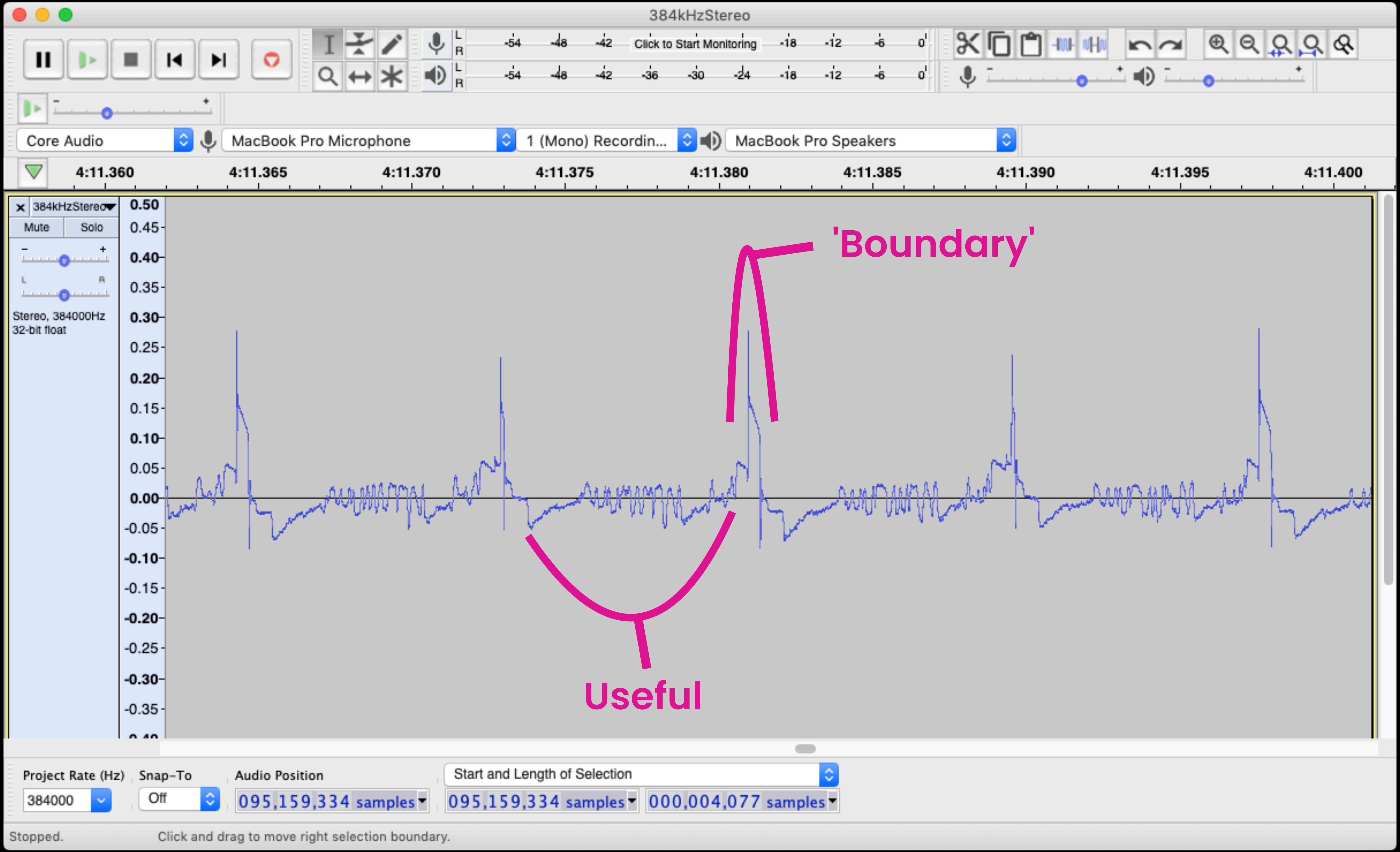
Snap-To: Off

Audio Position: 095,159,334 samples

Start and Length of Selection: 095,159,334 samples 000,004,077 samples

Stopped.

Click and drag to move right selection boundary.



'Boundary'

Useful

Core Audio

MacBook Pro Microphone

1 (Mono) Recordin...

MacBook Pro Speakers

Stereo, 384000Hz
32-bit float

Project Rate (Hz): 384000

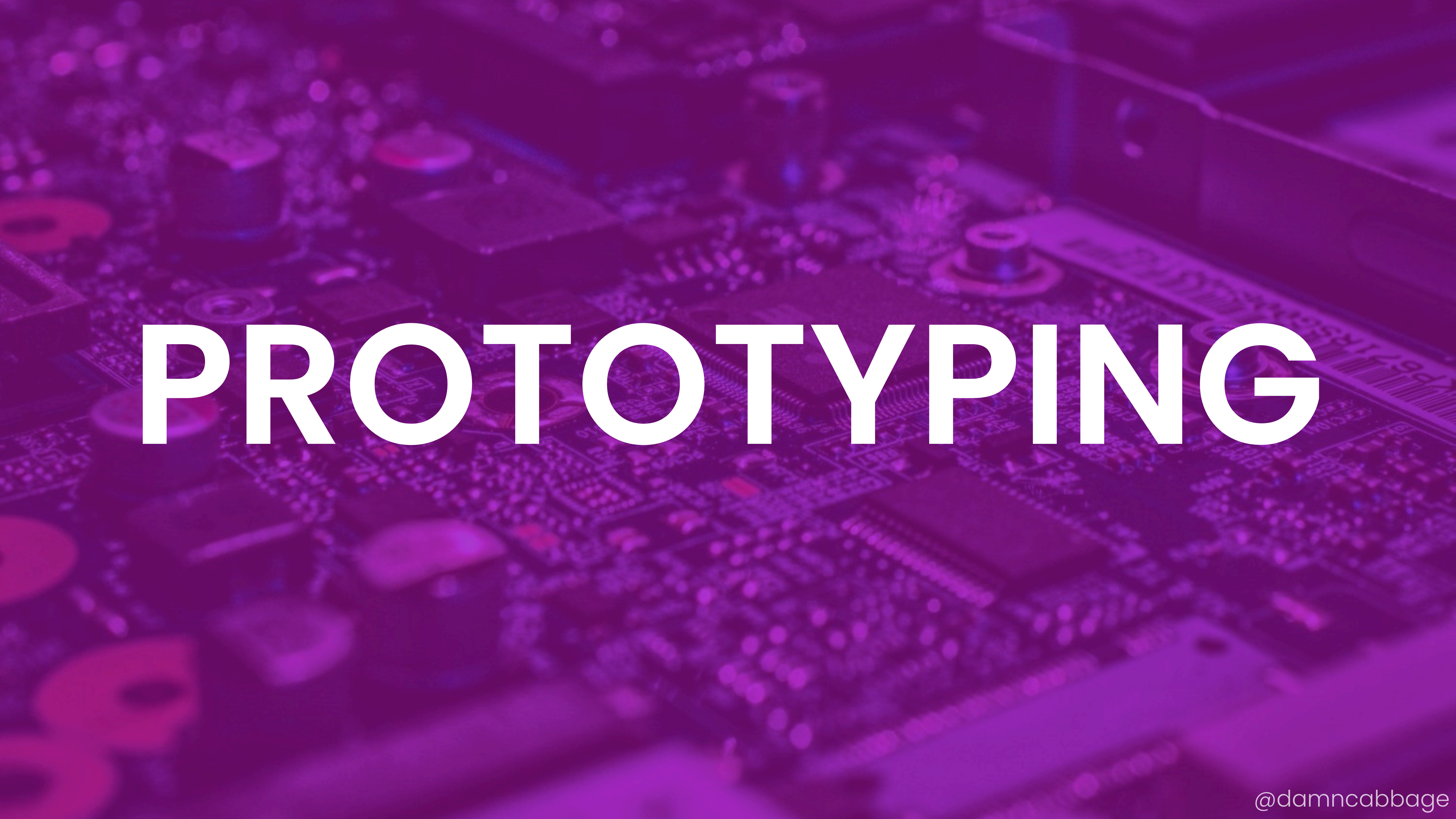
Snap-To: Off

Audio Position: 095,159,334 samples

Start and Length of Selection: 095,159,334 samples 000,004,077 samples

Stopped.

Click and drag to move right selection boundary.



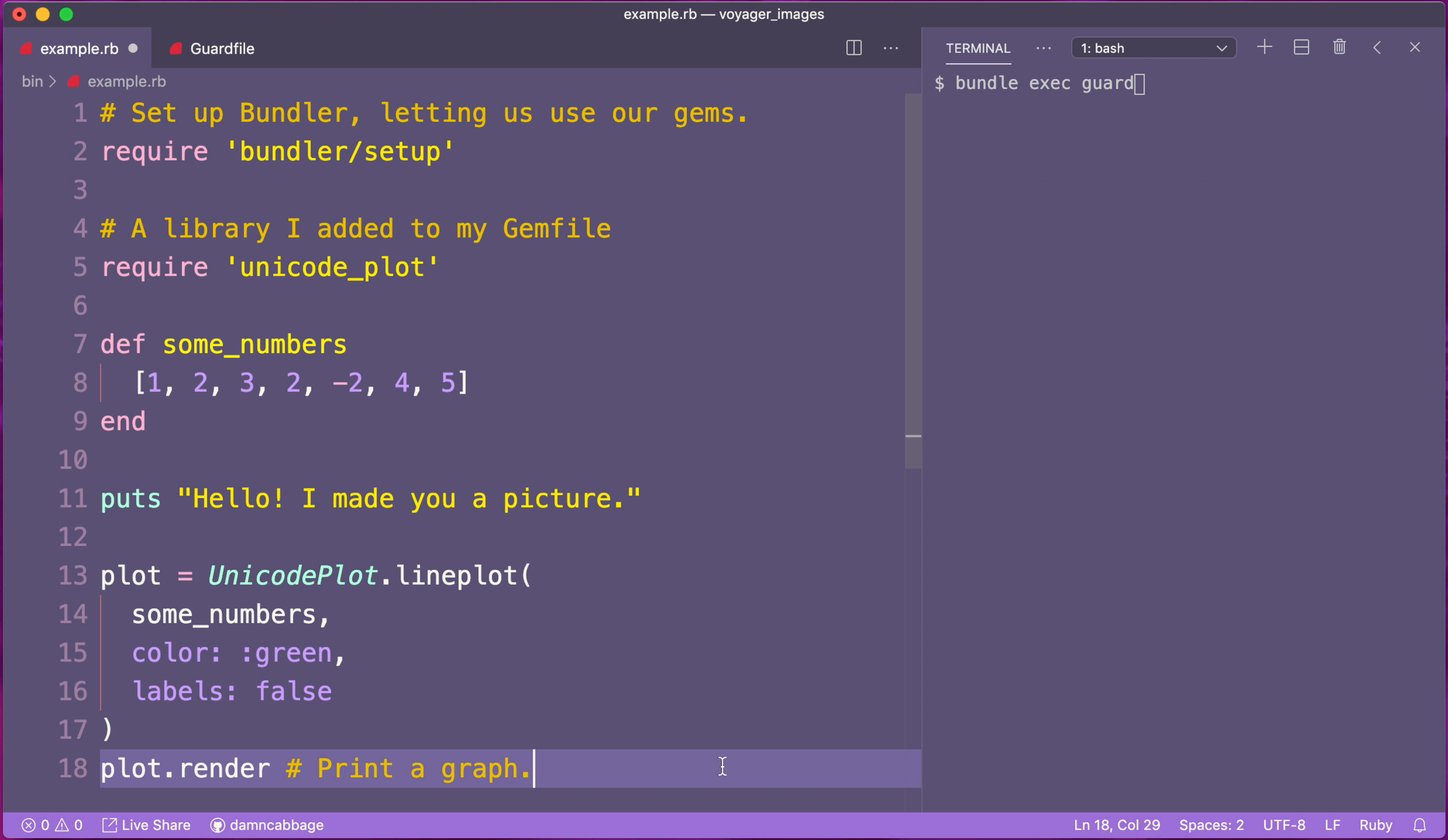
PROTOTYPING

JUST
DO
IT

JUST
DO
IT

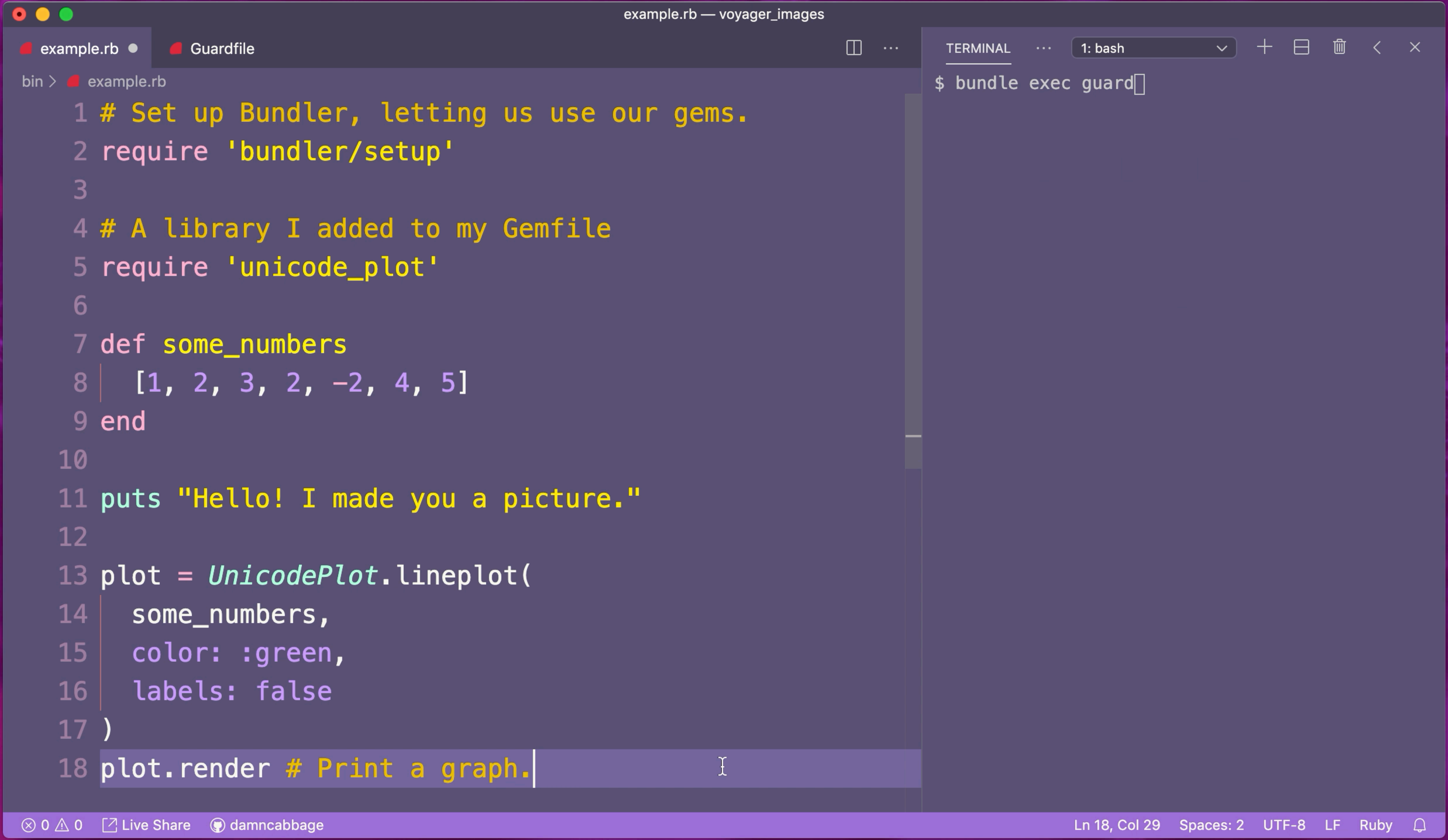
Many small attempts!

🔄



```ruby
# Set up Bundler, letting us use our gems.
require 'bundler/setup'

# A library I added to my Gemfile
require 'unicode_plot'

def some_numbers
  [1, 2, 3, 2, -2, 4, 5]
end

puts "Hello! I made you a picture."

plot = UnicodePlot.lineplot(
  some_numbers,
  color: :green,
  labels: false
)
plot.render # Print a graph.
```

```
$ bundle exec guard
```



```ruby
# Set up Bundler, letting us use our gems.
require 'bundler/setup'

# A library I added to my Gemfile
require 'unicode_plot'

def some_numbers
  [1, 2, 3, 2, -2, 4, 5]
end

puts "Hello! I made you a picture."

plot = UnicodePlot.lineplot(
  some_numbers,
  color: :green,
  labels: false
)
plot.render # Print a graph.
```

```
$ bundle exec guard
```

👍 Copy+paste!

👍 Copy+paste!

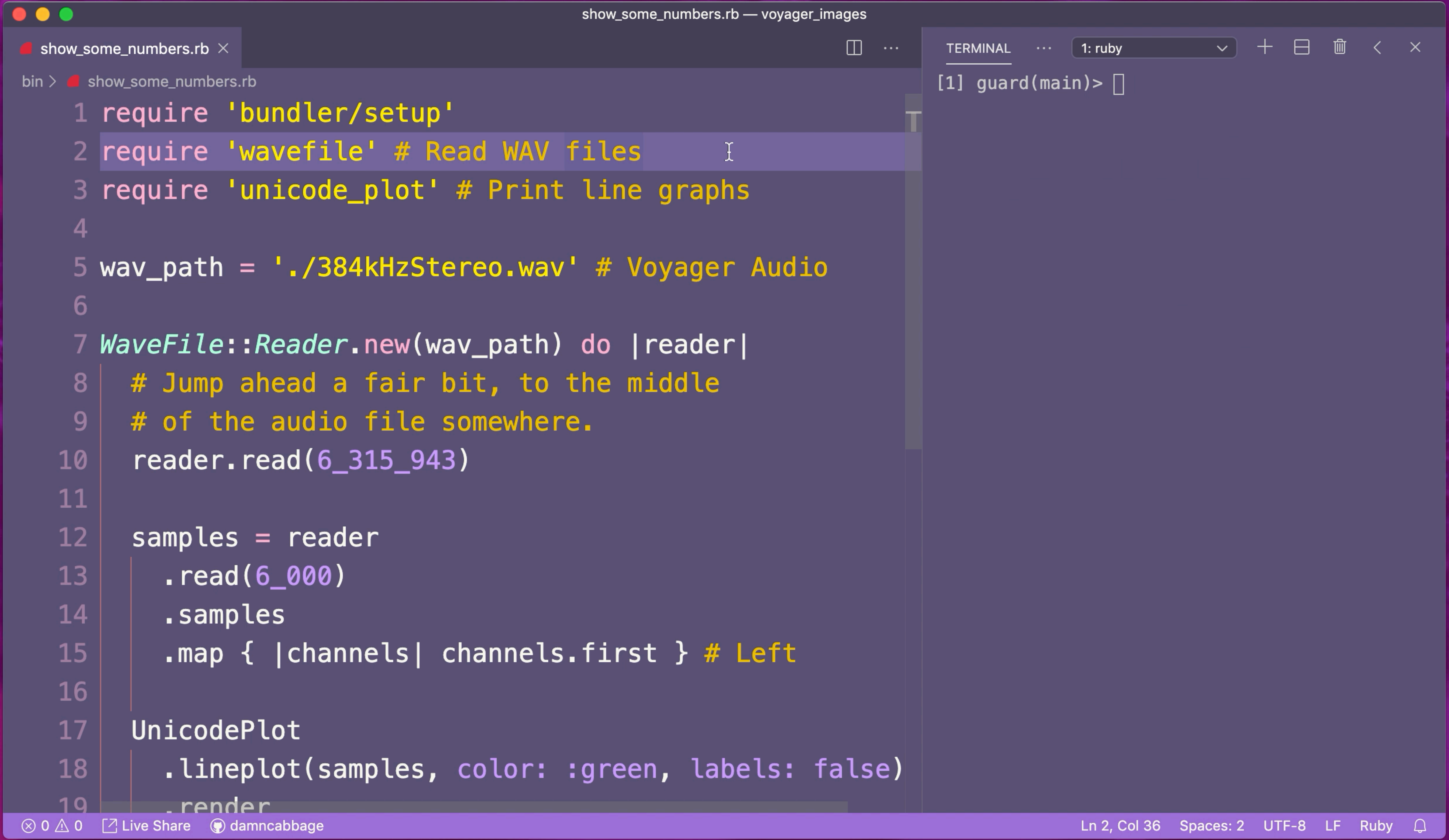
👍 One big file!

- 👍 Copy+paste!
- 👍 One big file!
- 👍 No tests!

- 👍 Copy+paste!
- 👍 One big file!
- 👍 No tests!
- 👍 `puts` everywhere!

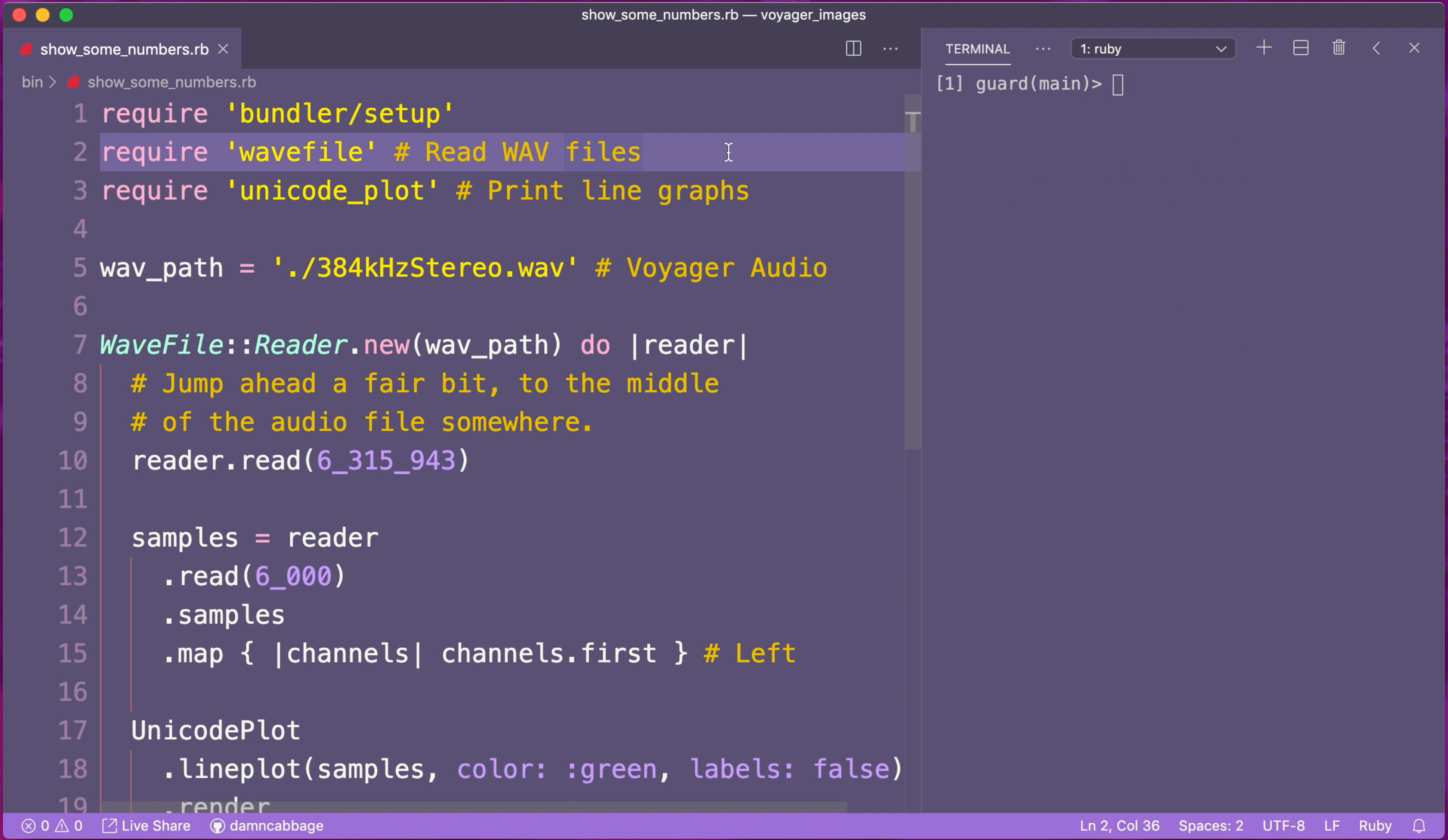
`byebug` if it helps!

DO IT



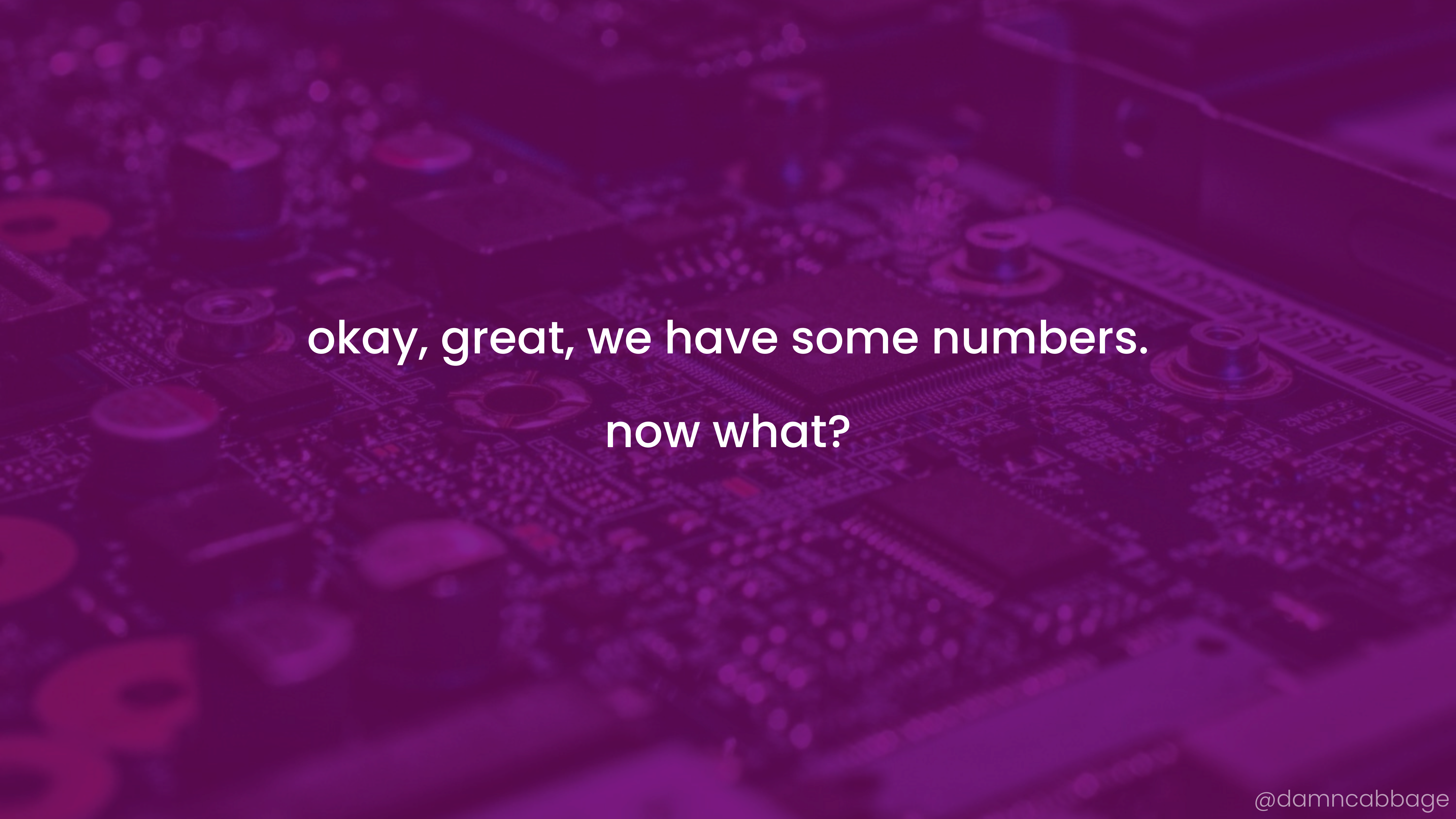
```ruby
require 'bundler/setup'
require 'wavefile' # Read WAV files
require 'unicode_plot' # Print line graphs

wav_path = './384kHzStereo.wav' # Voyager Audio

WaveFile::Reader.new(wav_path) do |reader|
  # Jump ahead a fair bit, to the middle
  # of the audio file somewhere.
  reader.read(6_315_943)

  samples = reader
    .read(6_000)
    .samples
    .map { |channels| channels.first } # Left

  UnicodePlot
    .lineplot(samples, color: :green, labels: false)
    .render
```

```
[1] guard(main)>
```



```ruby
require 'bundler/setup'
require 'wavefile' # Read WAV files
require 'unicode_plot' # Print line graphs

wav_path = './384kHzStereo.wav' # Voyager Audio

WaveFile::Reader.new(wav_path) do |reader|
  # Jump ahead a fair bit, to the middle
  # of the audio file somewhere.
  reader.read(6_315_943)

  samples = reader
    .read(6_000)
    .samples
    .map { |channels| channels.first } # Left

  UnicodePlot
    .lineplot(samples, color: :green, labels: false)
    .render
```

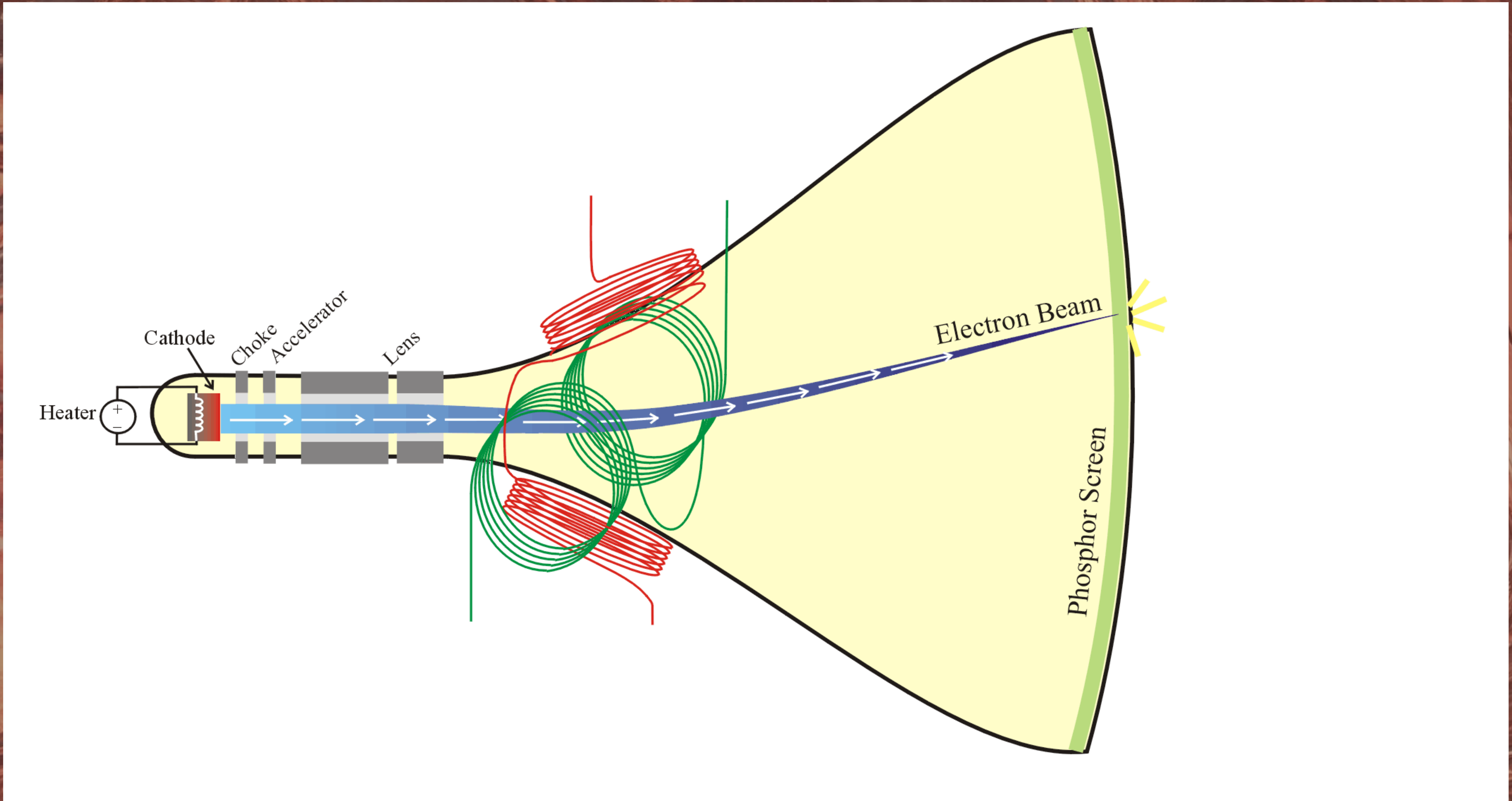
```
[1] guard(main)> 
```

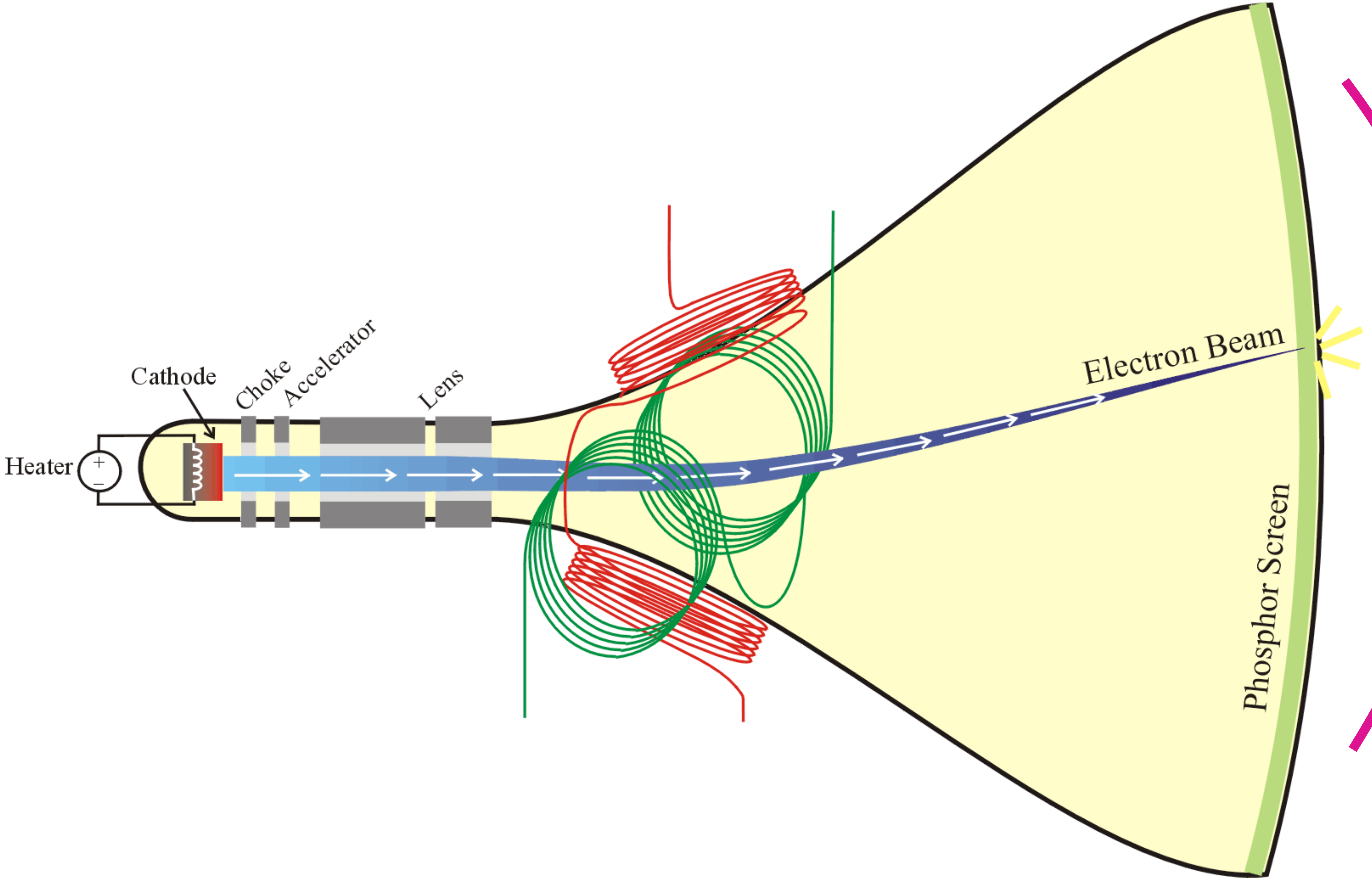


okay, great, we have some numbers.

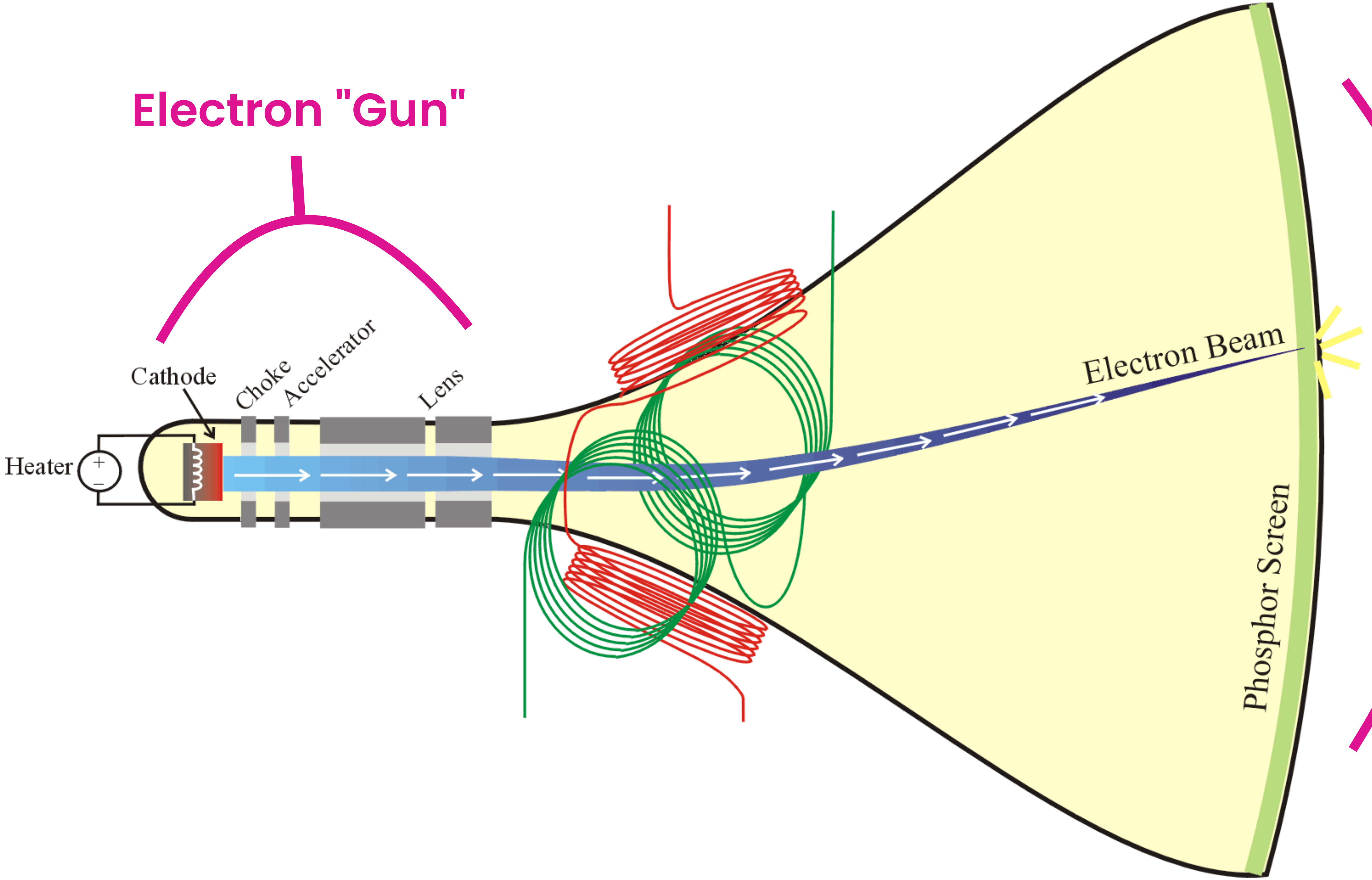
now what?

IMAGES

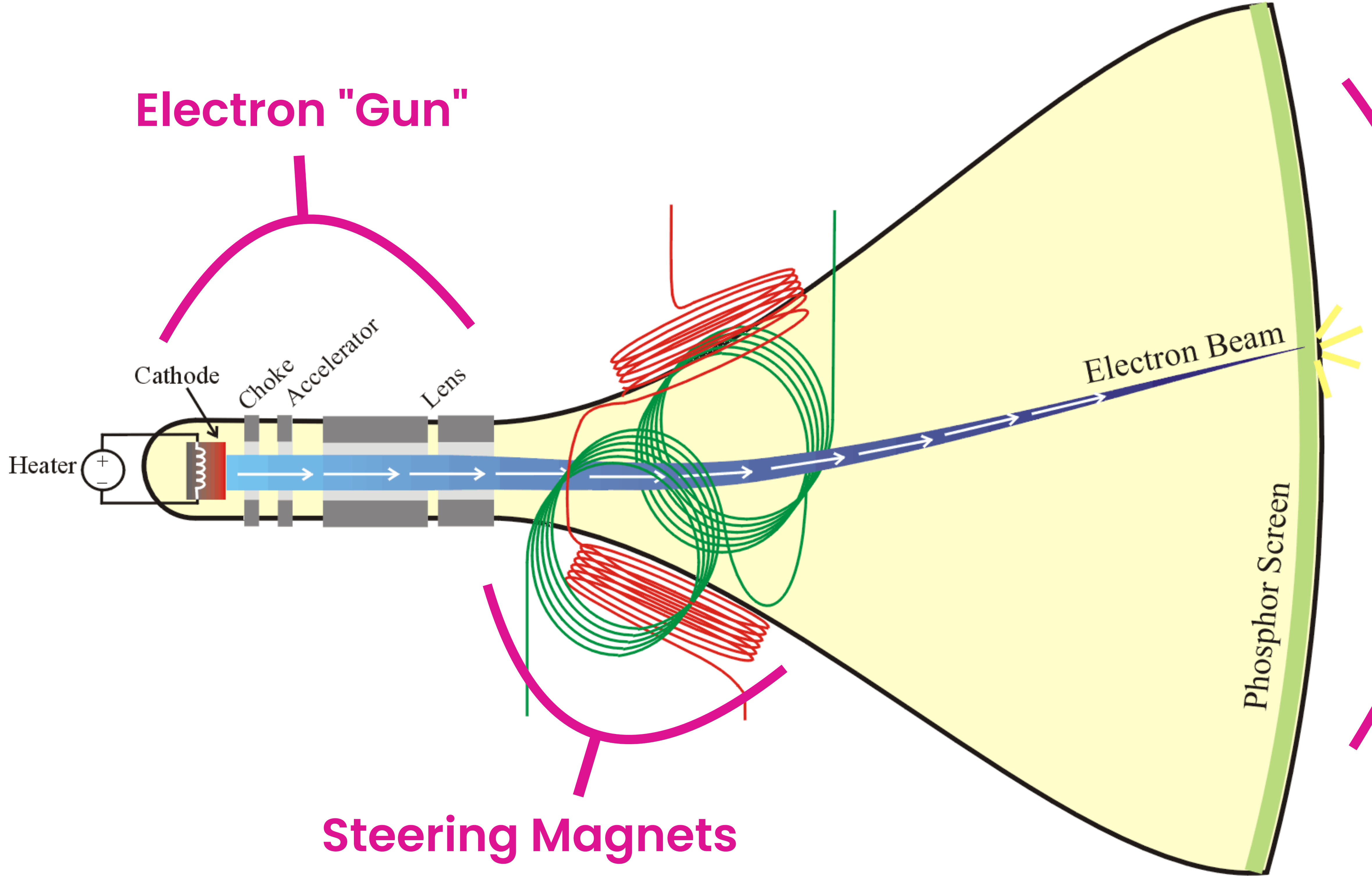

Cathode
Choke
Accelerator
Lens
Heater
+
–
Electron Beam
Phosphor Screen



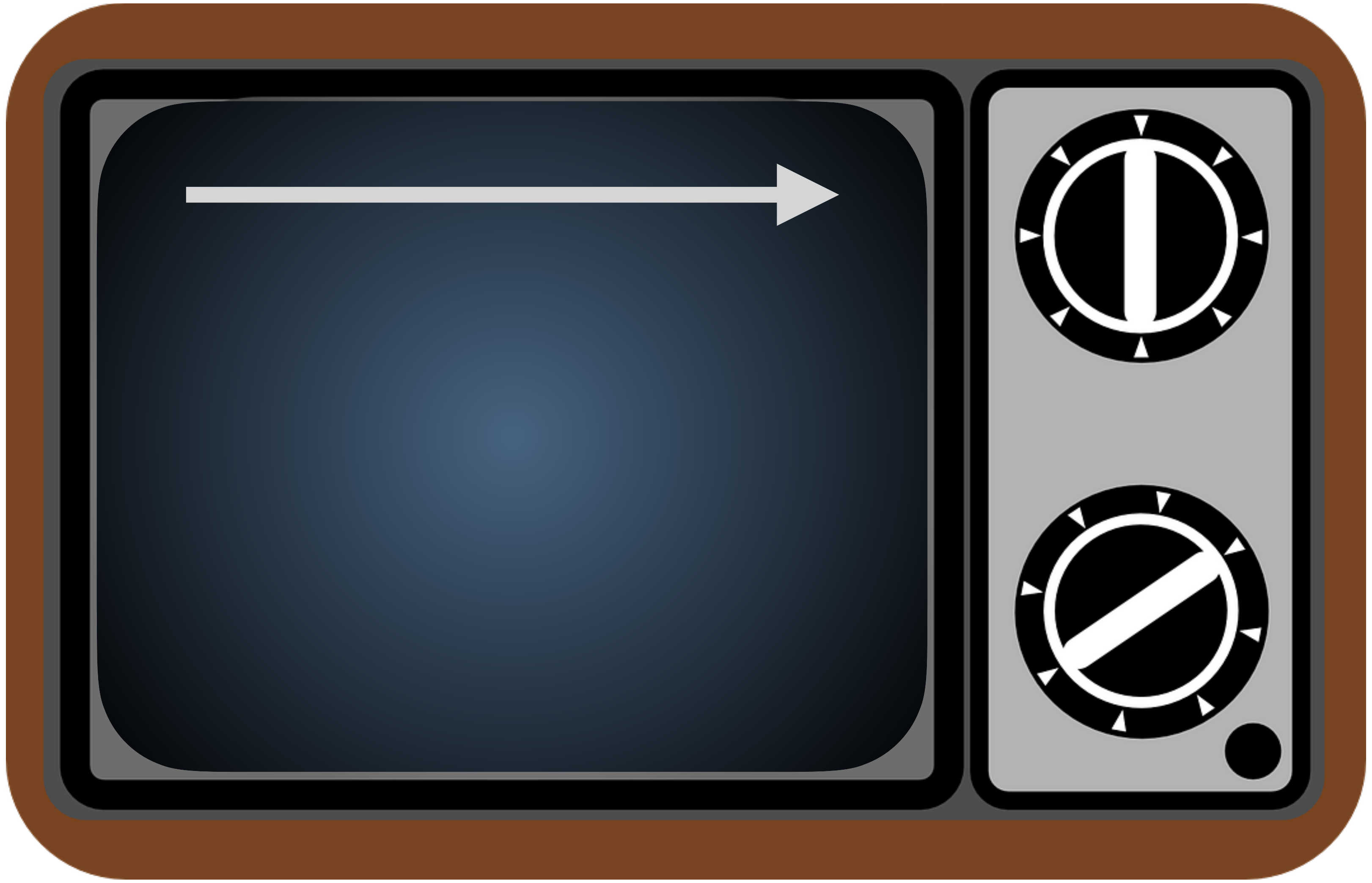
The TV bit you can actually see.

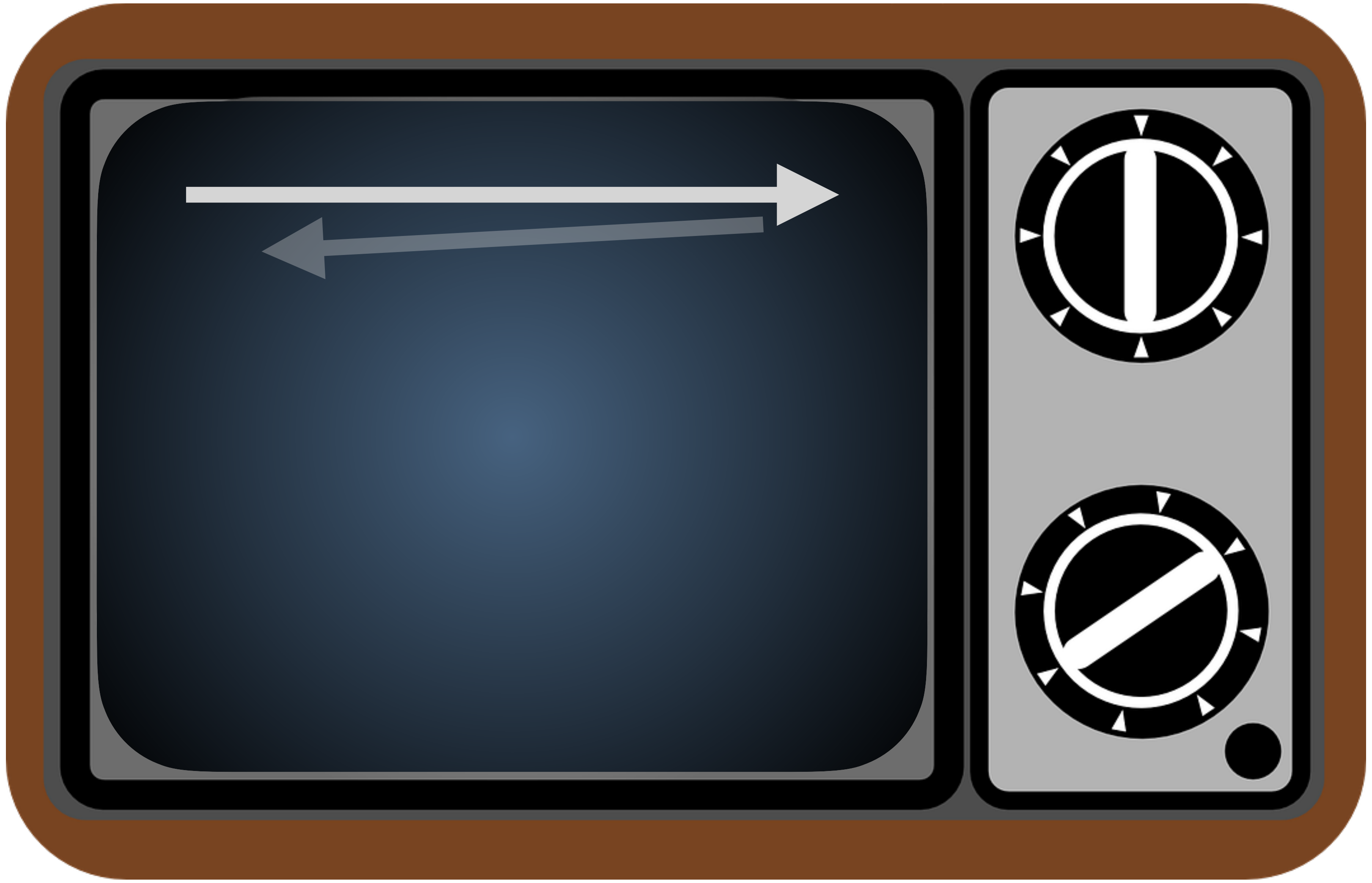


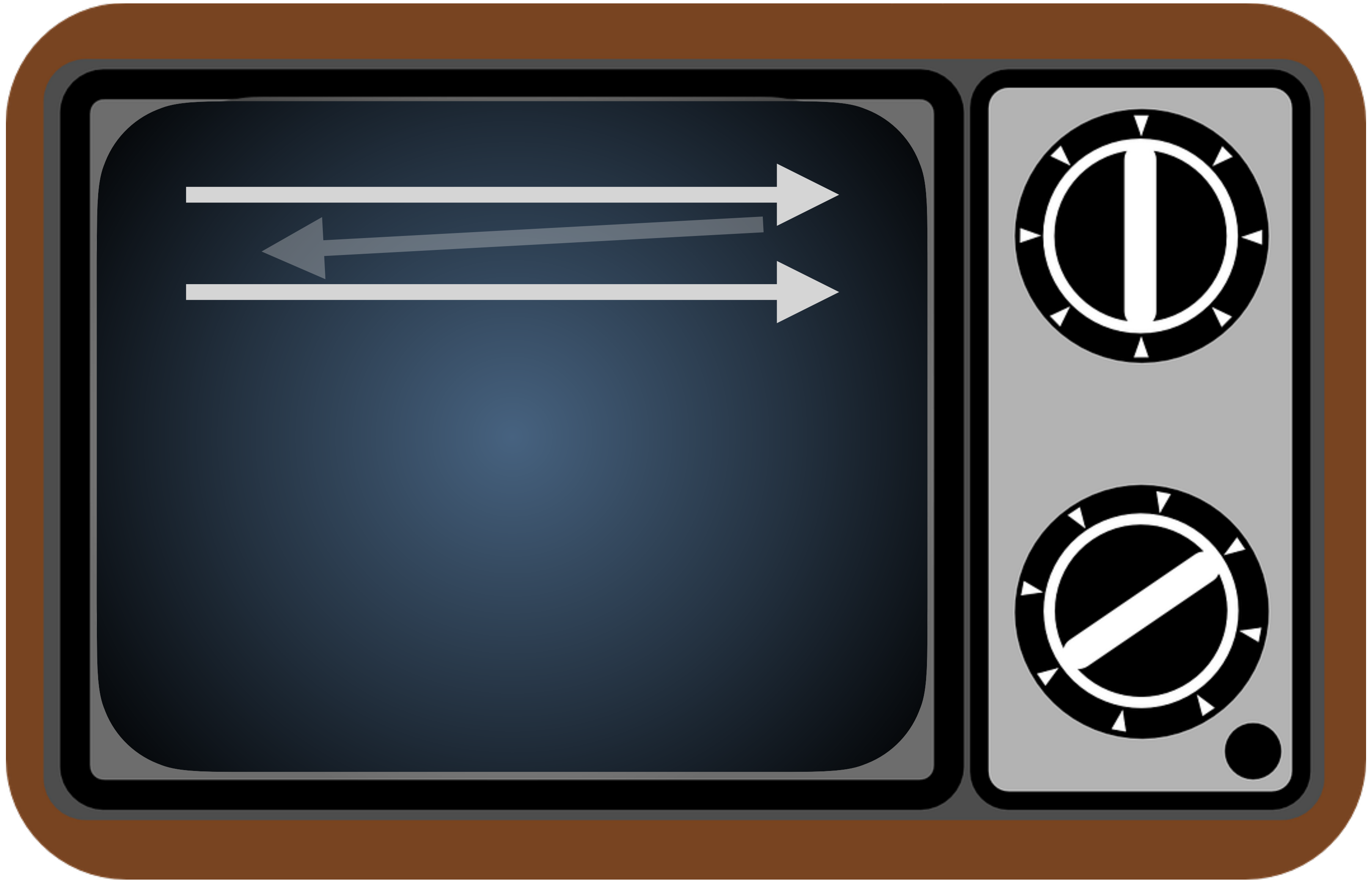
The TV bit you can actually see.

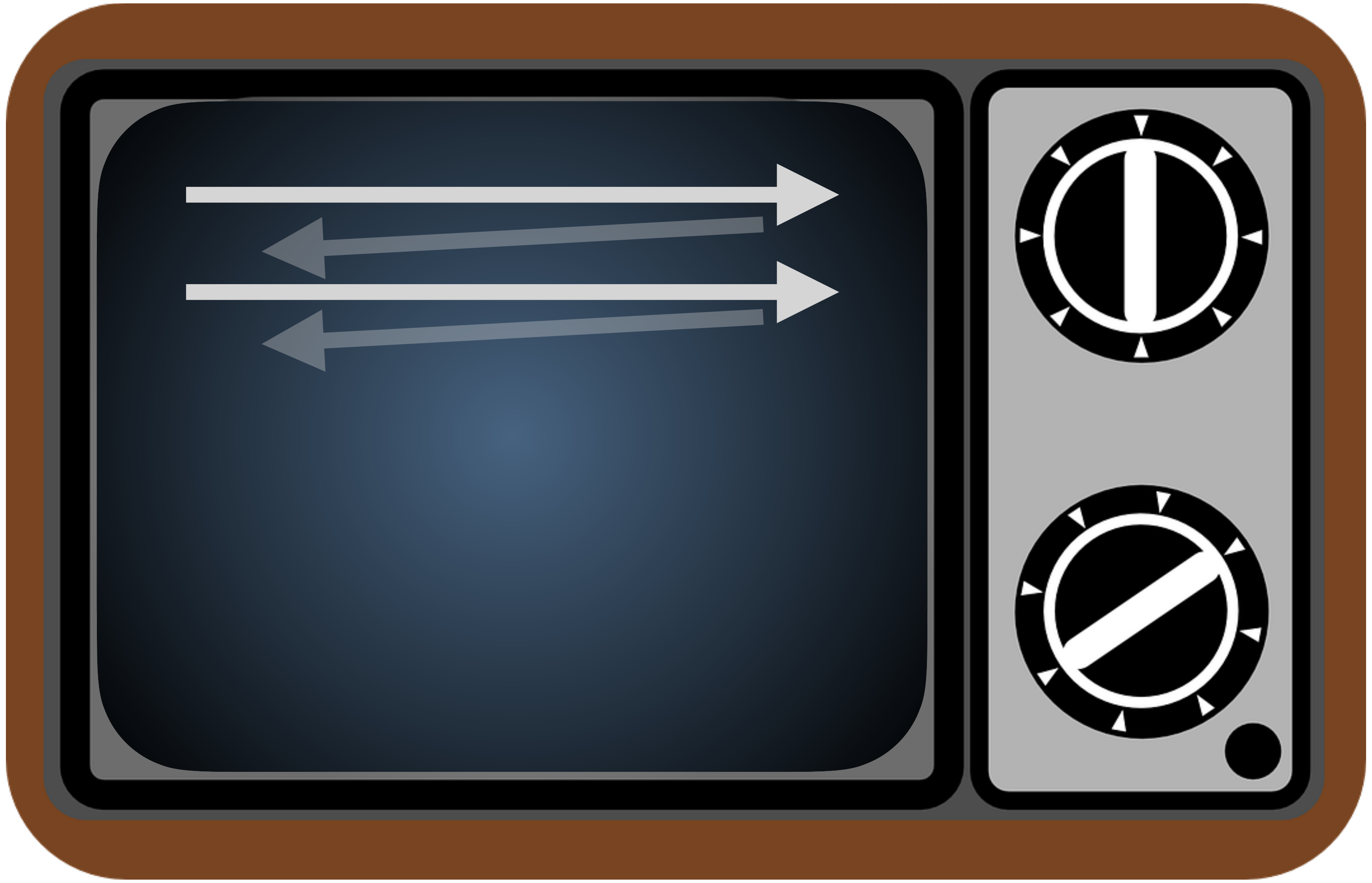


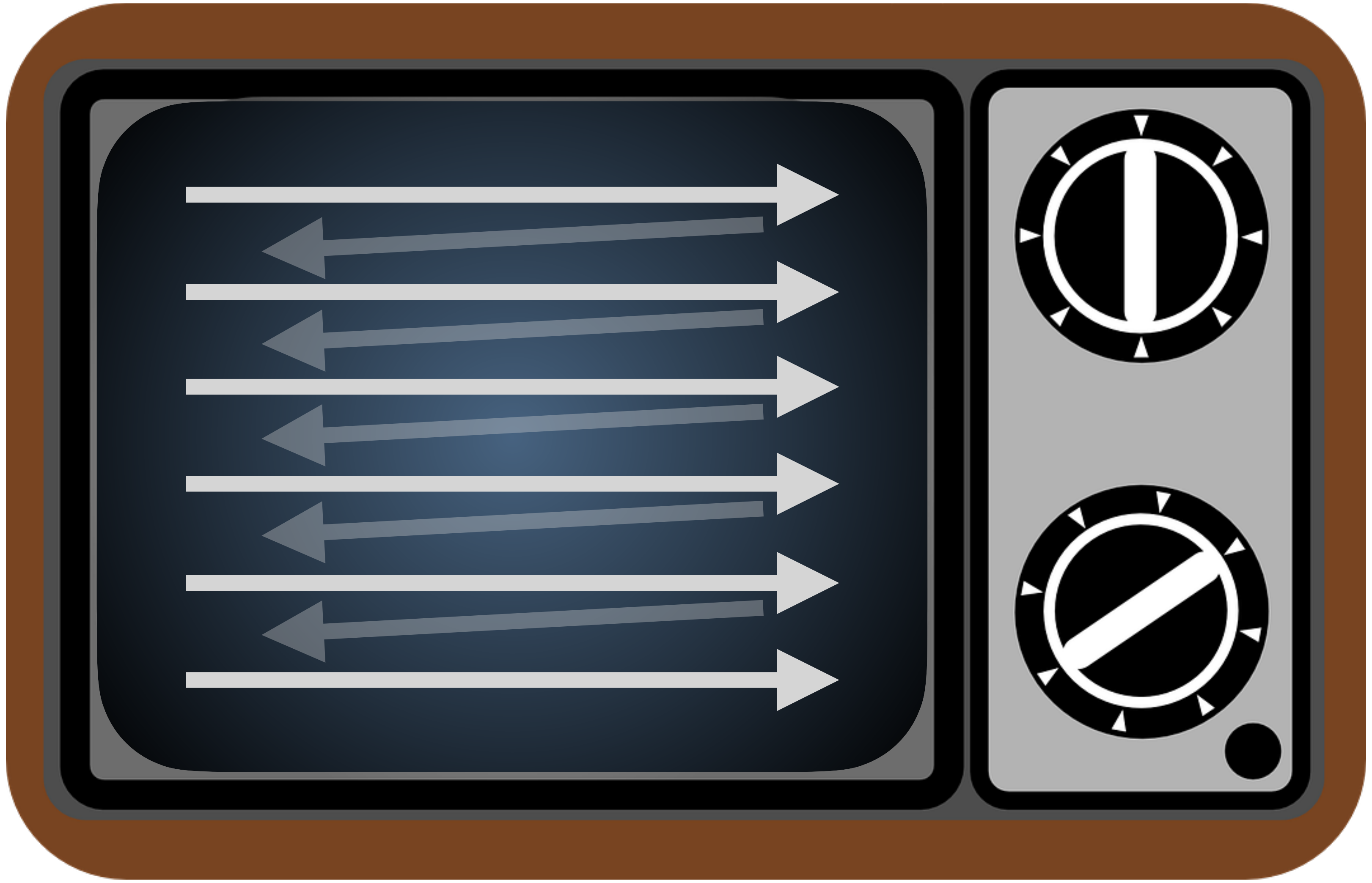
The TV bit you can actually see.

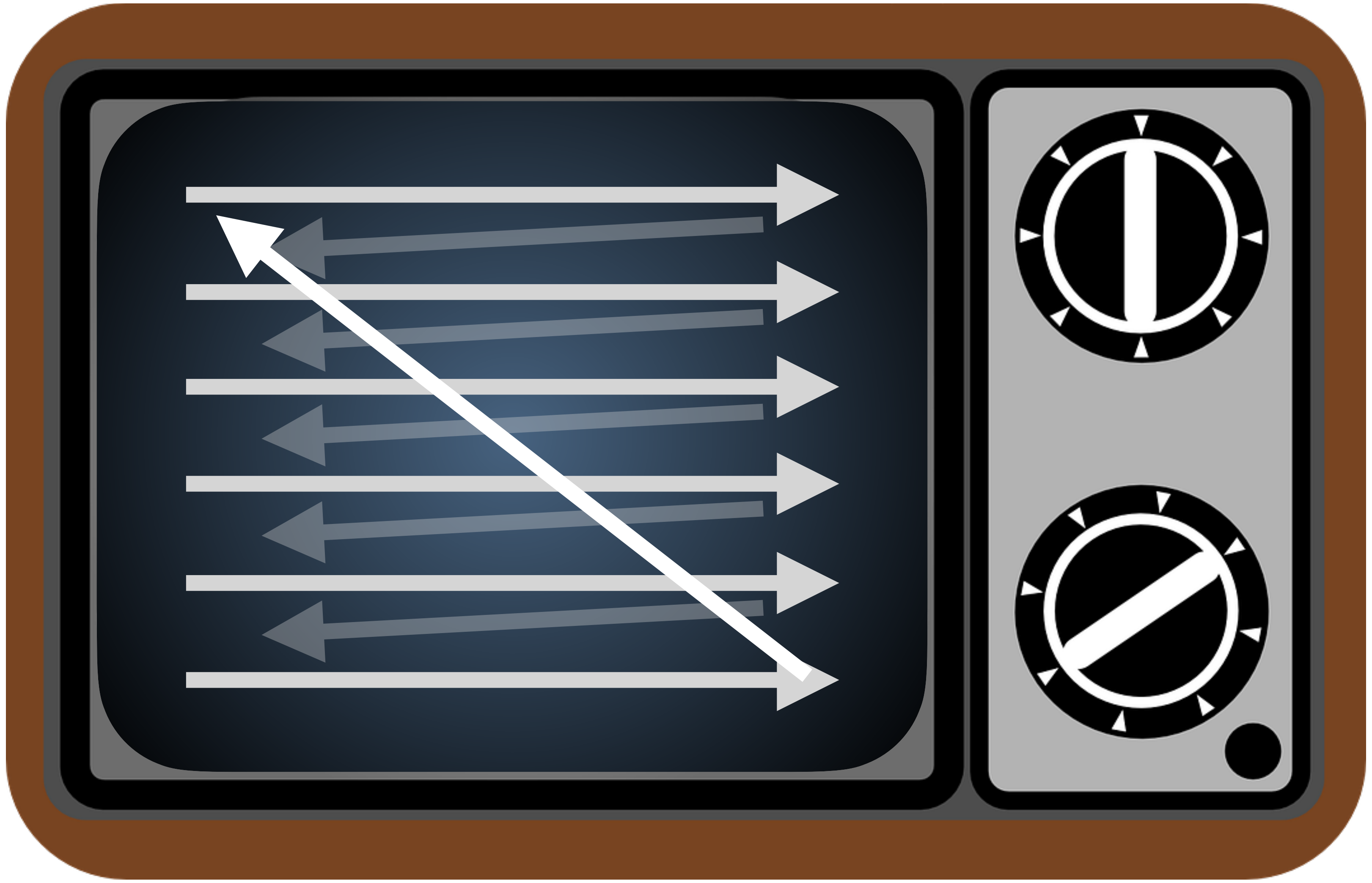












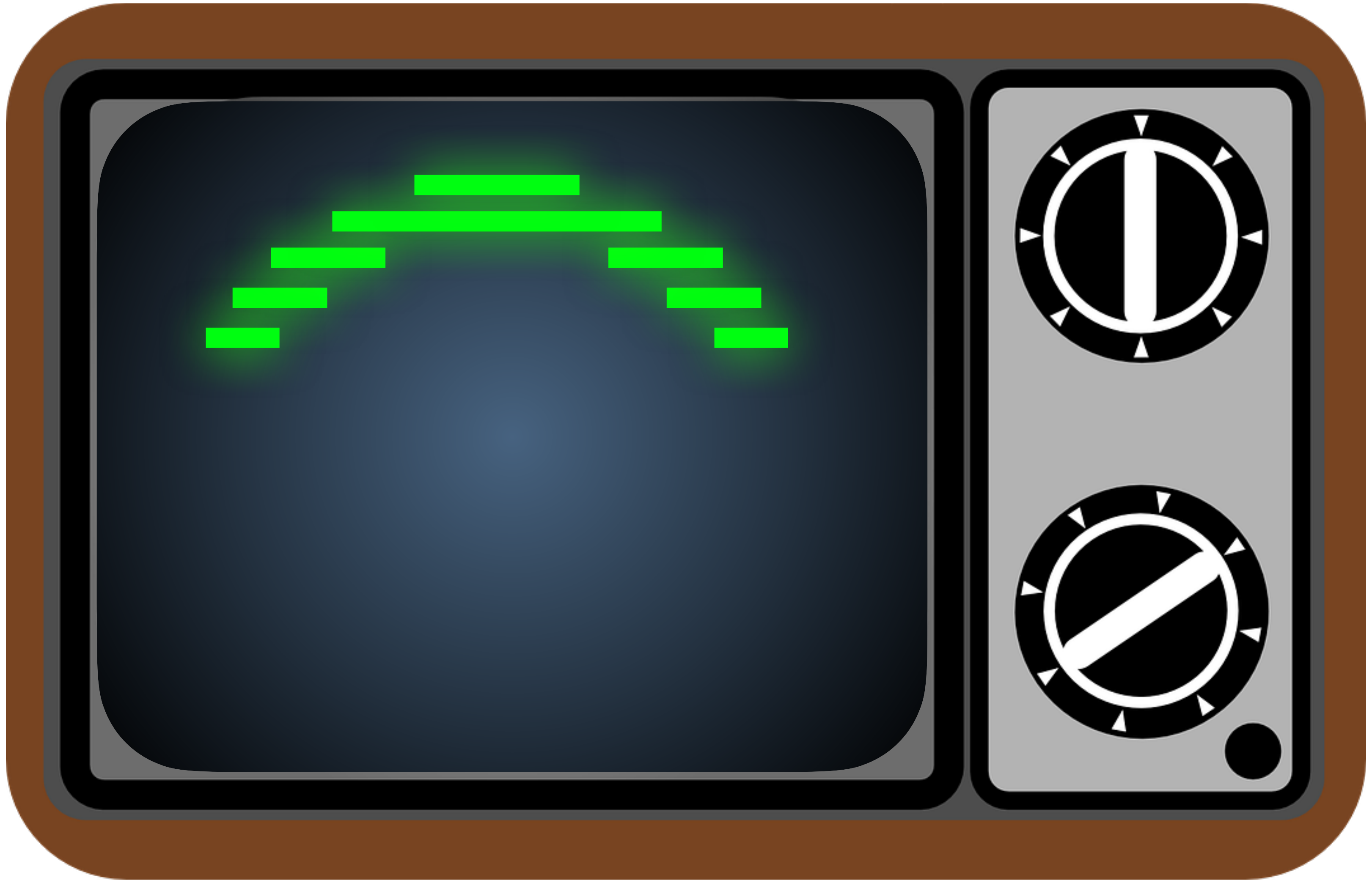
@damncabbage

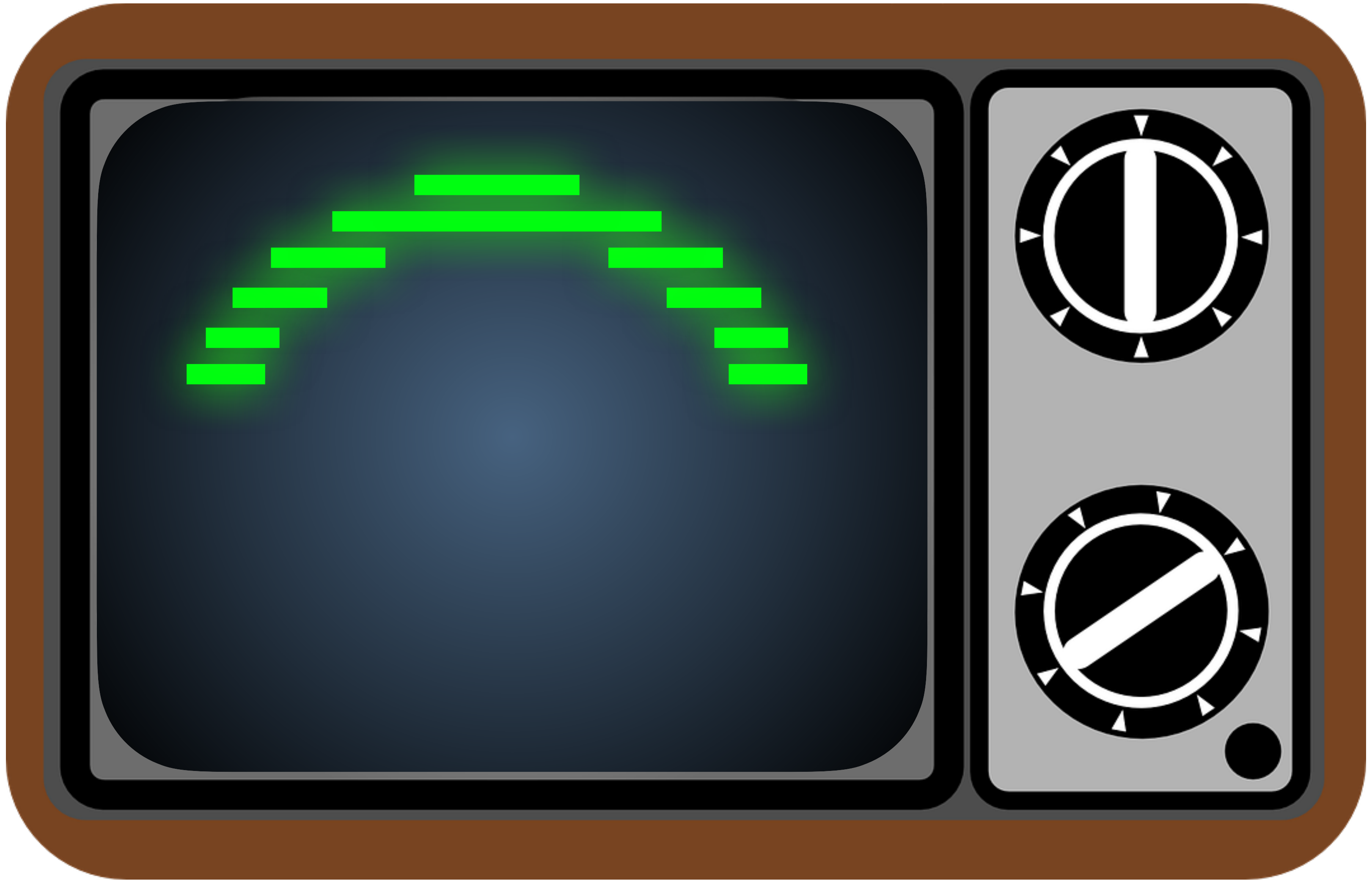
@damncabbage

@damncabbage

@damncabbage

@damncabbage



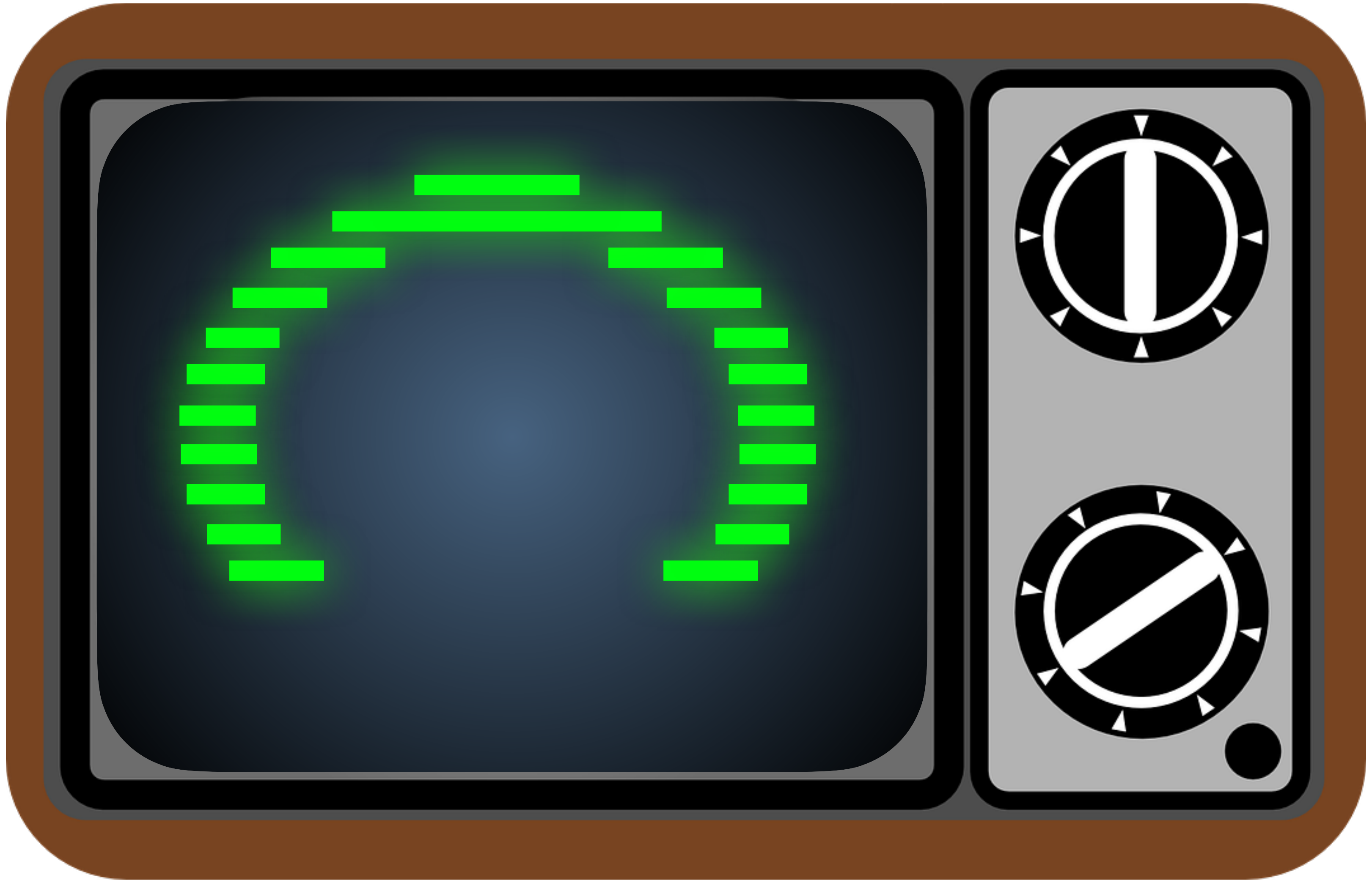


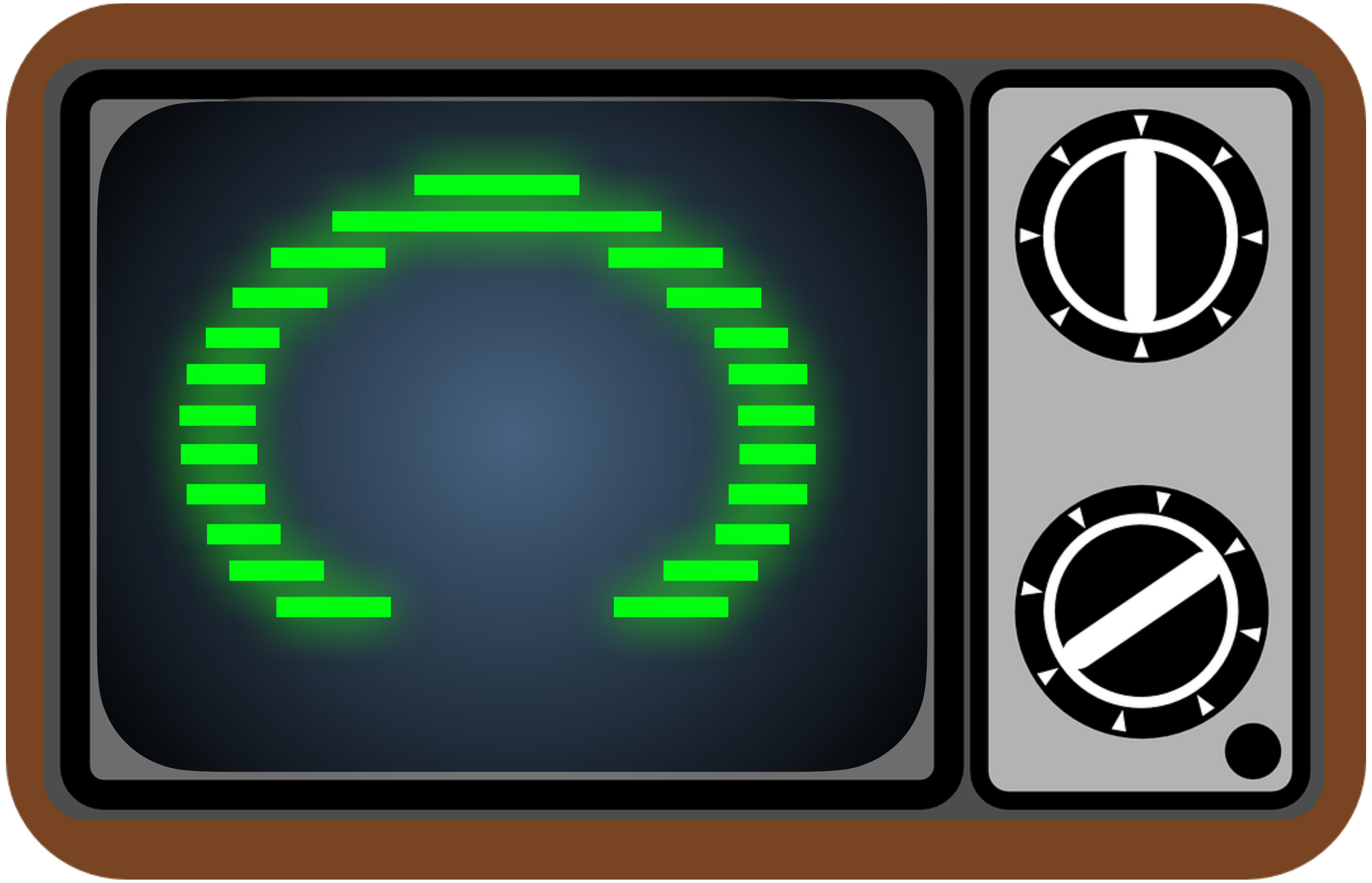
@damncabbage

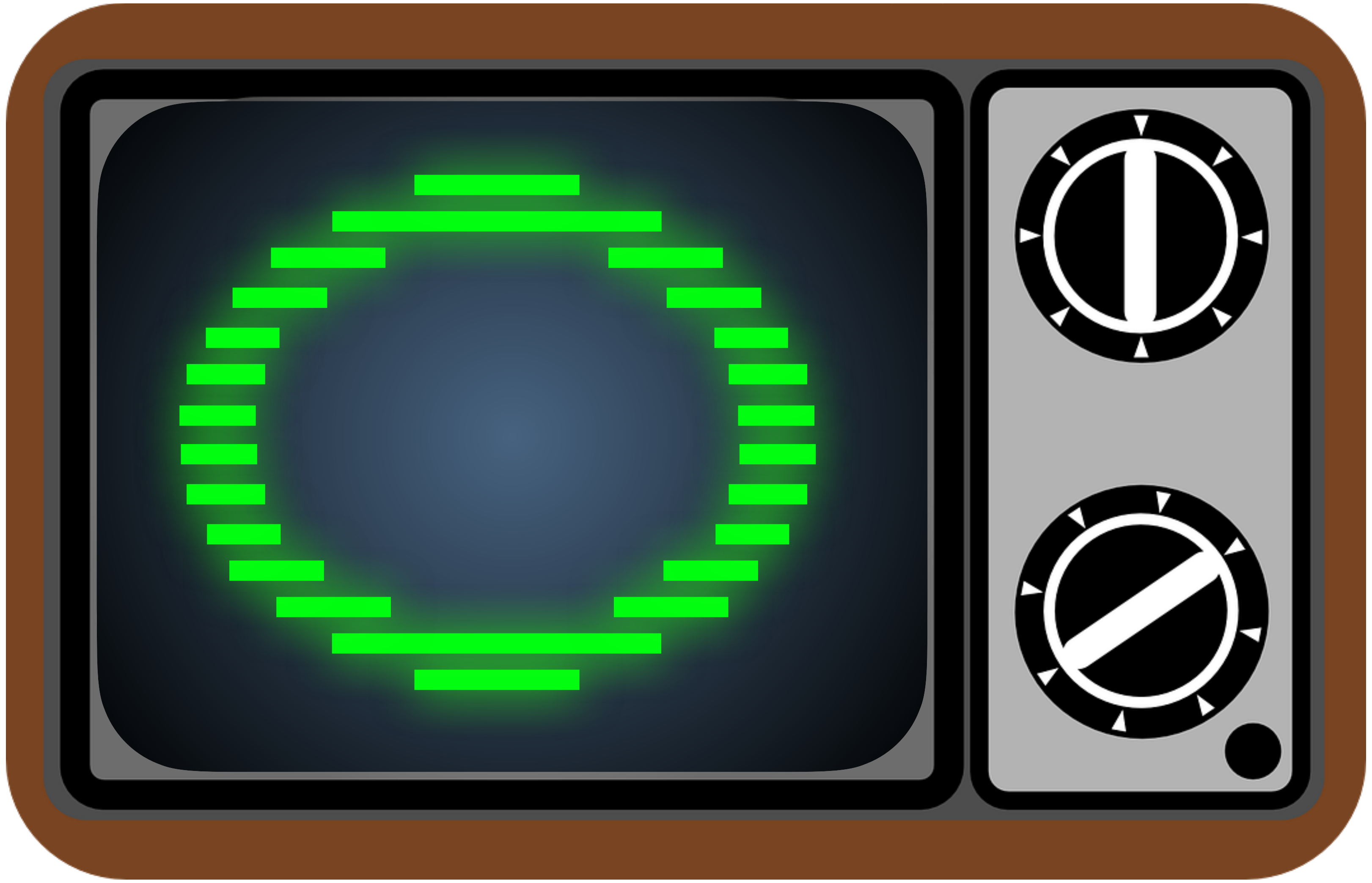
@damncabbage

@damncabbage

@damncabbage

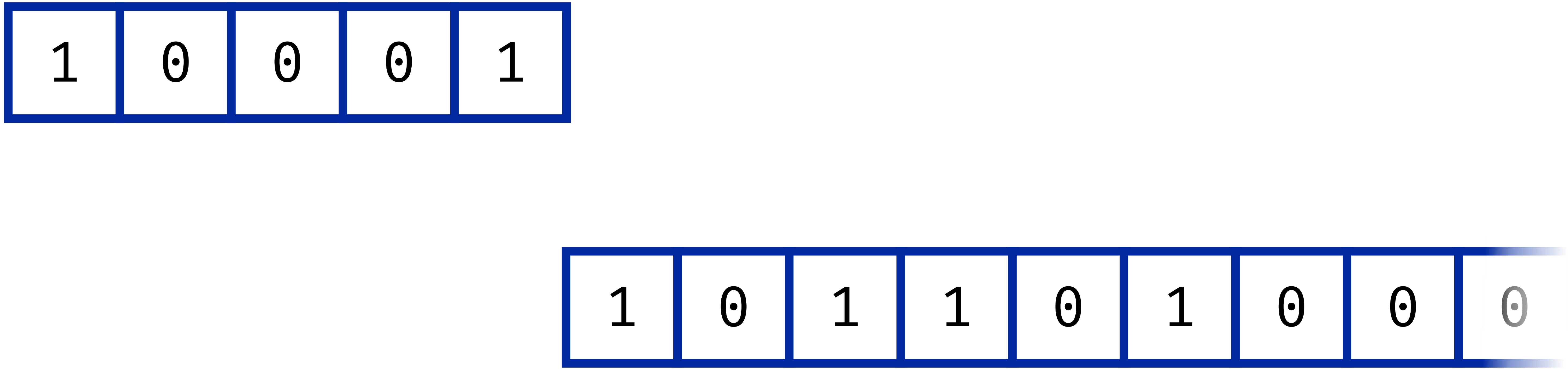






1	0	0	0	1	1	0	1	1	0	1	0	0	0

1	0	0	0	1	1	0	1	1	0	1	0	0	0



1 0 0 0 1

1 0 1 1 0 1 0 0 0

1	0	0	0	1
1	0	1	1	0

1	0	0	0

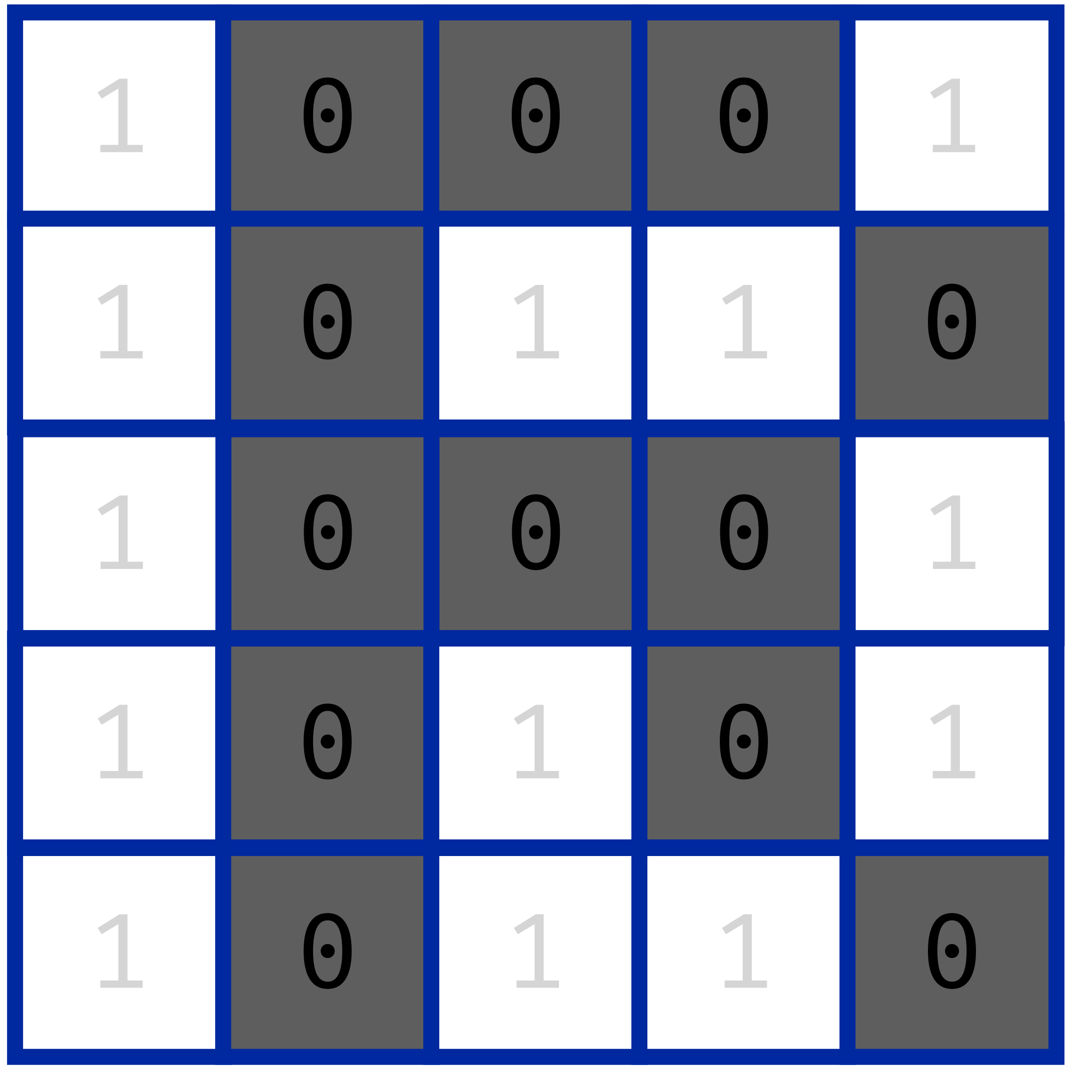
1	0	0	0	1
1	0	1	1	0
1	0	0	0	1

1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1

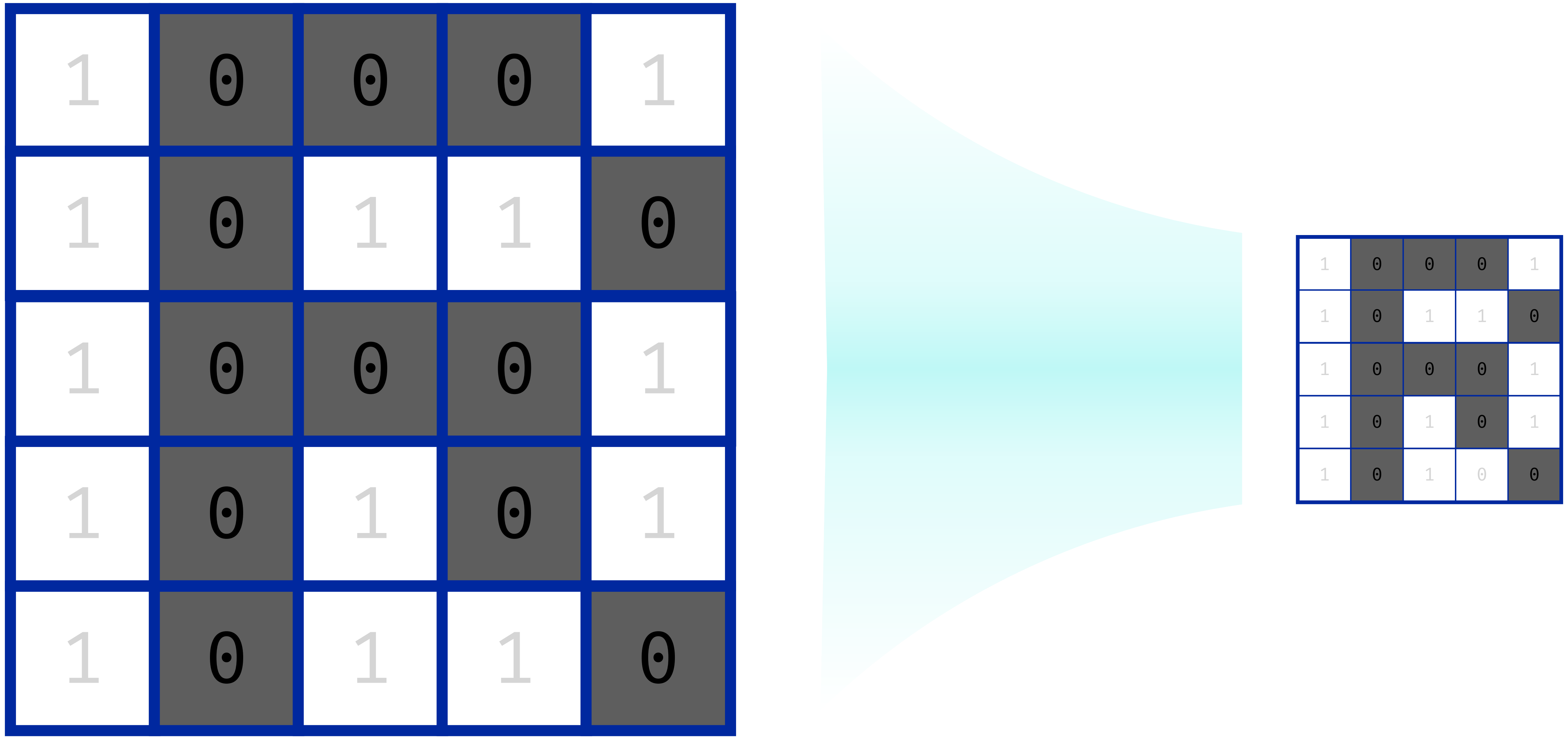
1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0



1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

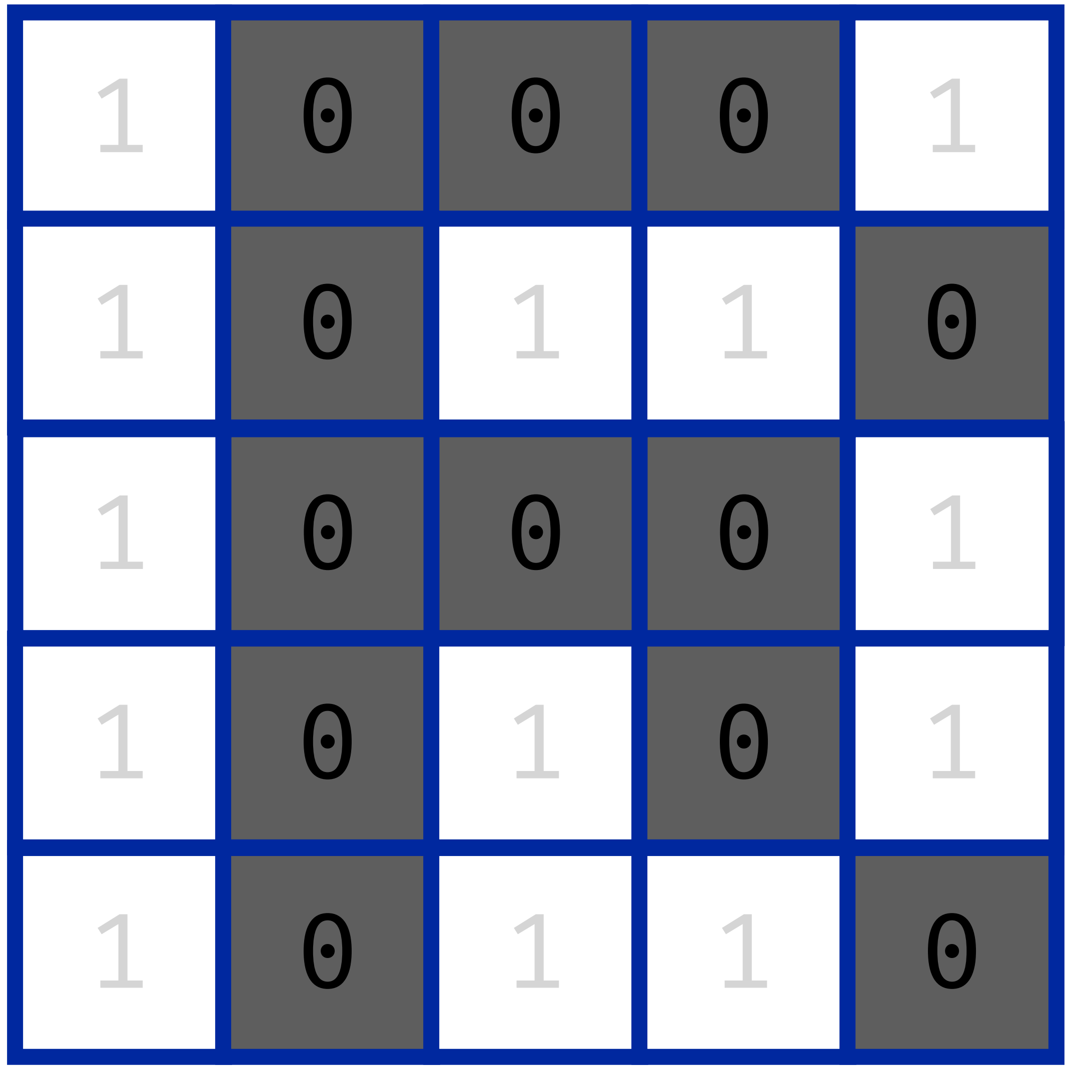


1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

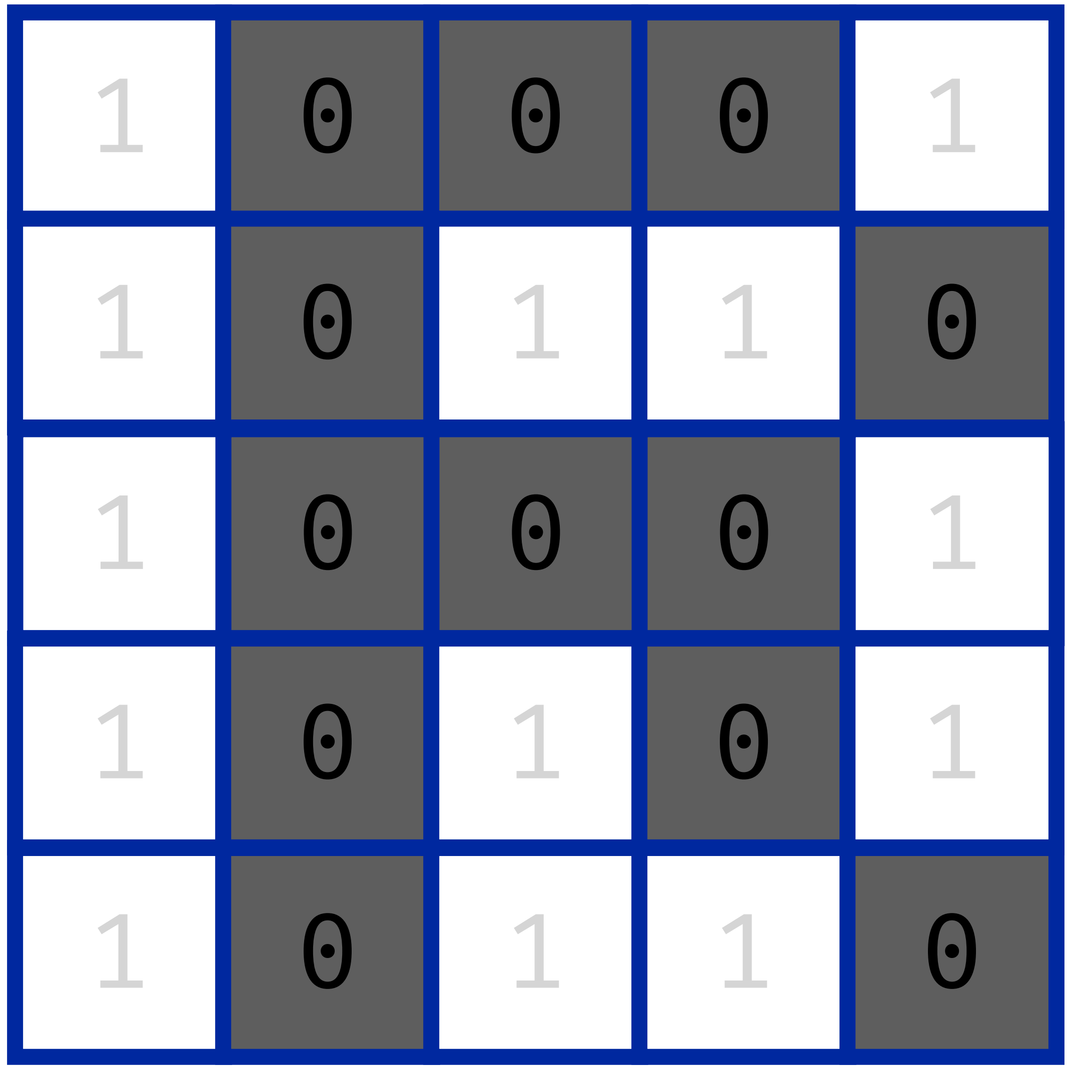
1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	0	0

1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

P2

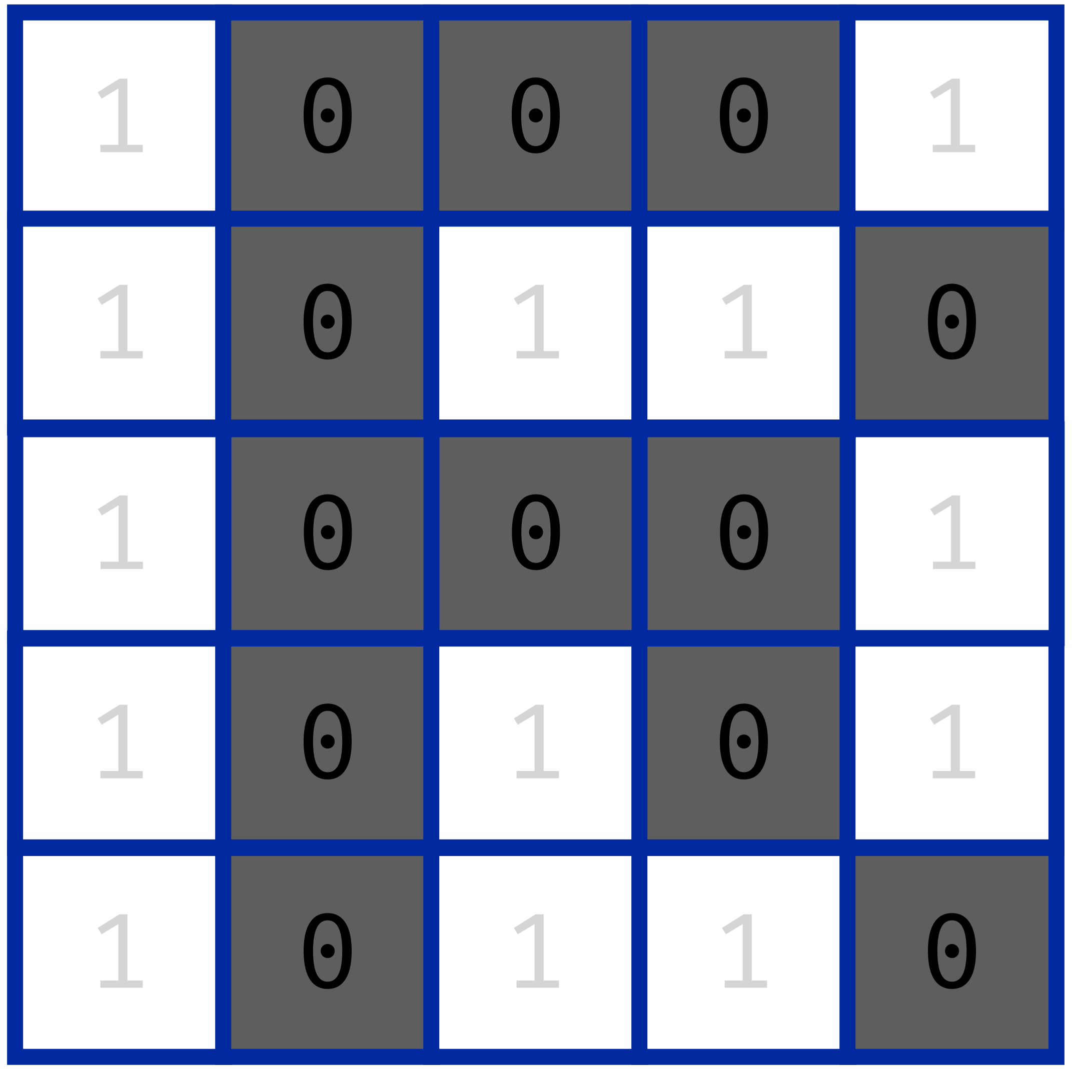


1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0



1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

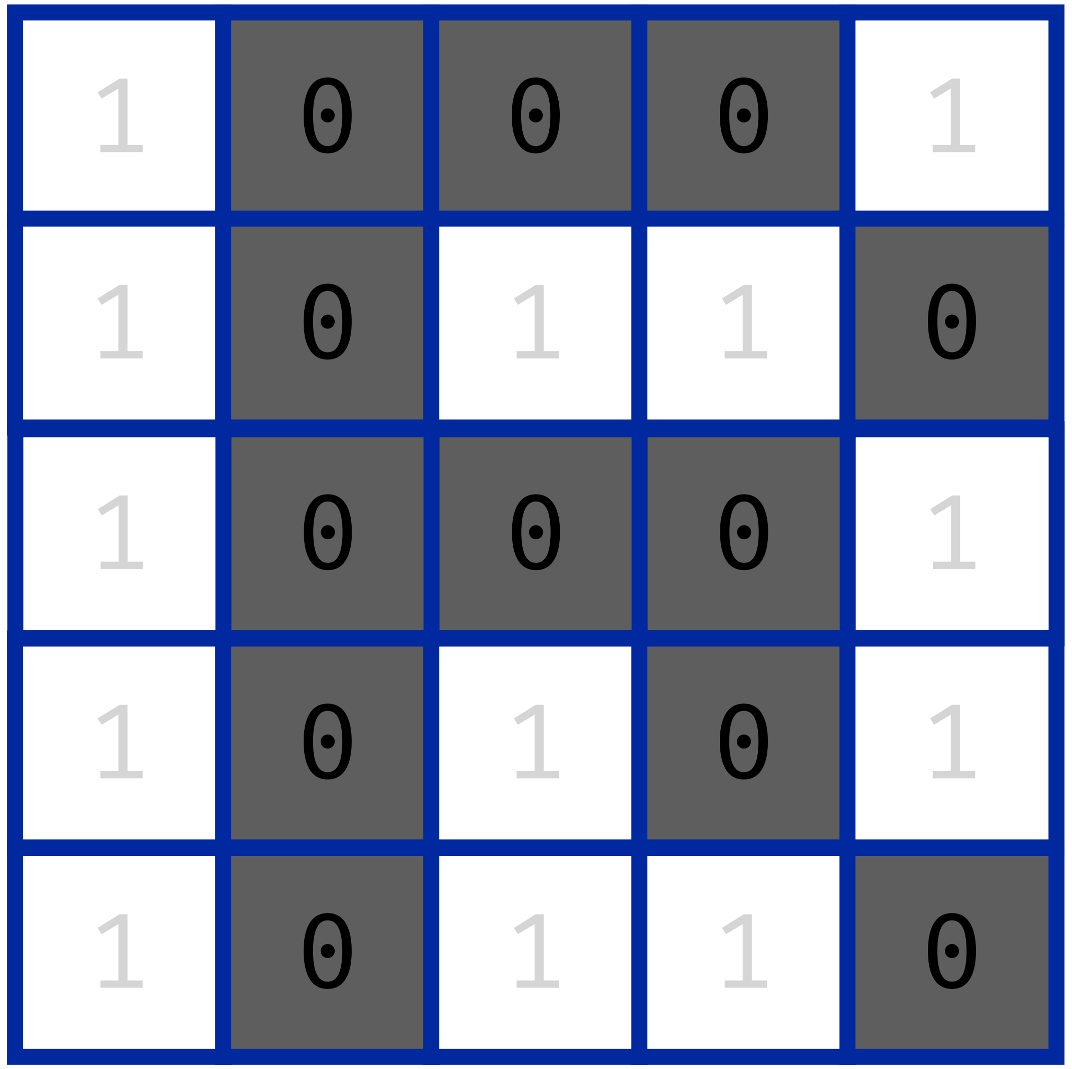
```
P2
5 5 # 5x5 grid
```



1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

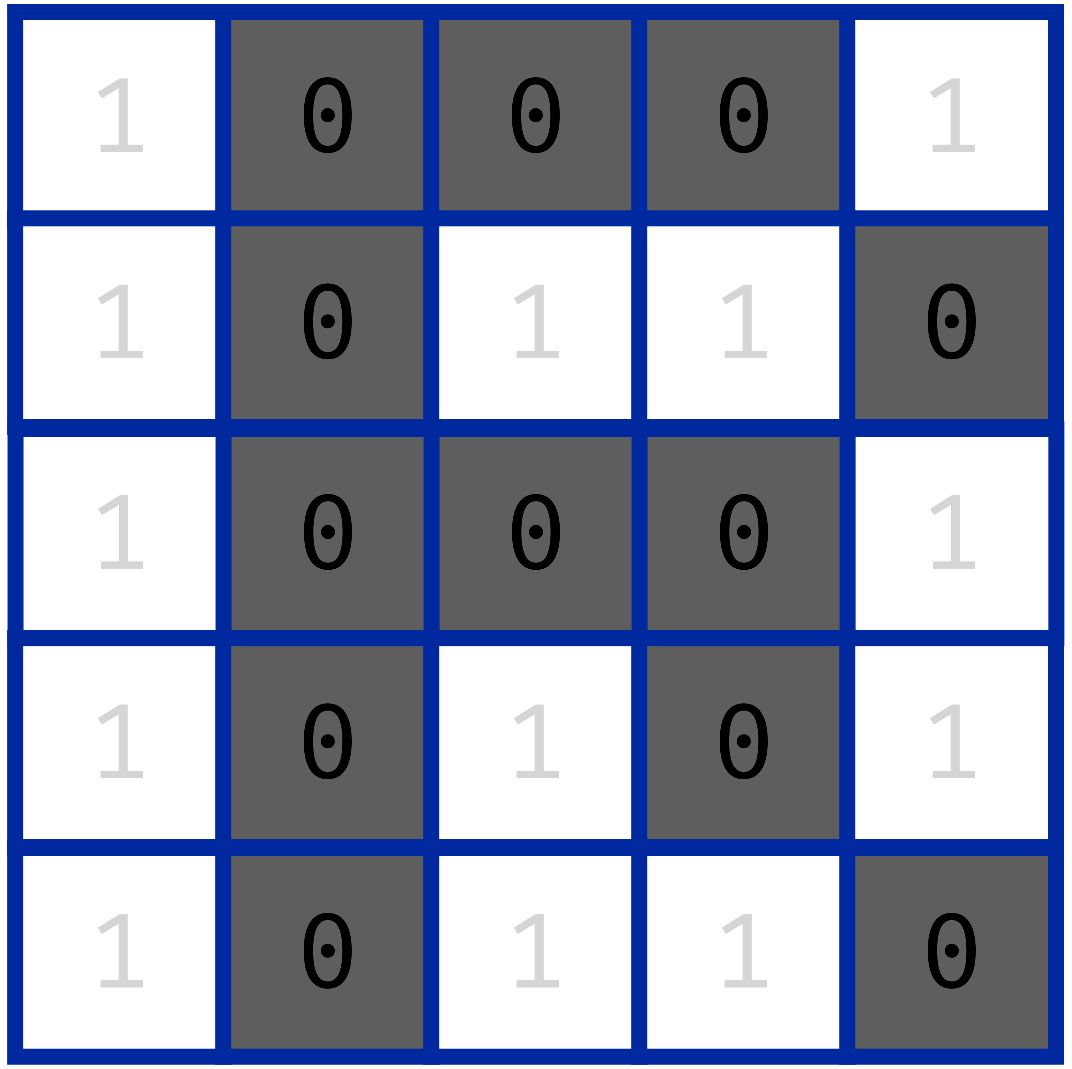
```
P2
5 5 # 5x5 grid
1 # Max pixel value
```

@damncabbage



1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

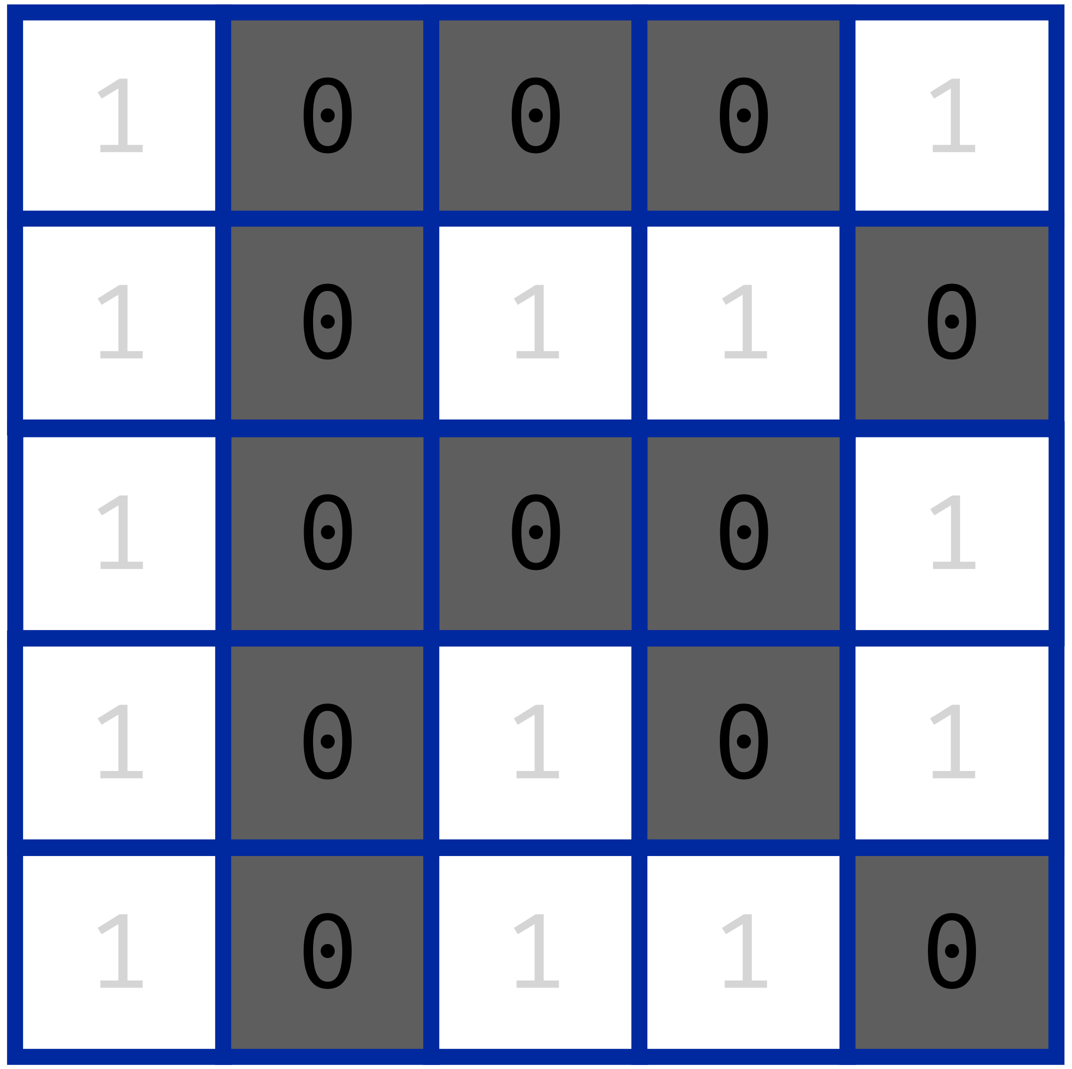
```
P2
5 5 # 5x5 grid
1 # Max pixel value

# Draw the picture:
```



1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

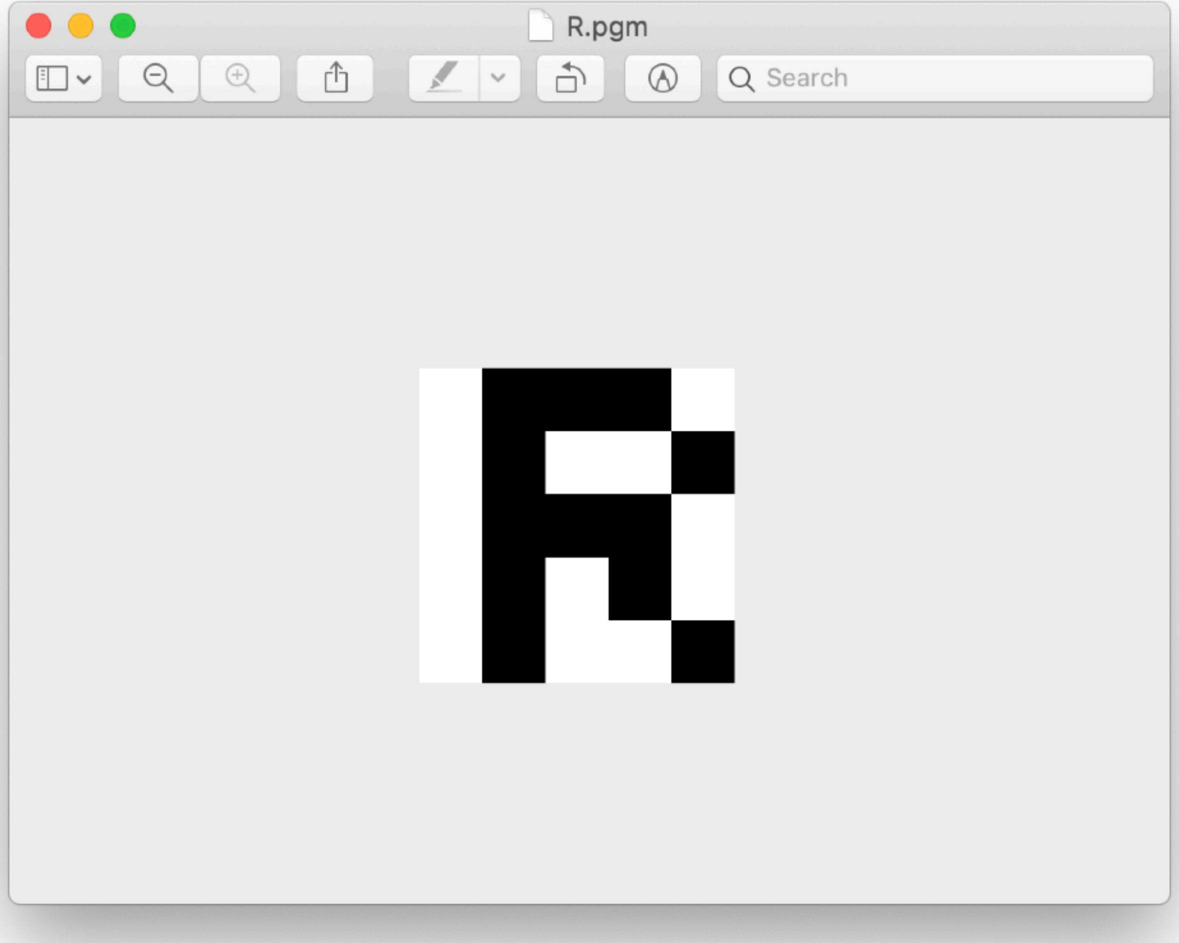
```
P2
5 5 # 5x5 grid
1 # Max pixel value

# Draw the picture:
1 0 0 0 1
1 0 1 1 0
1 0 0 0 1
1 0 1 0 1
1 0 1 1 0
```

@damncabbage



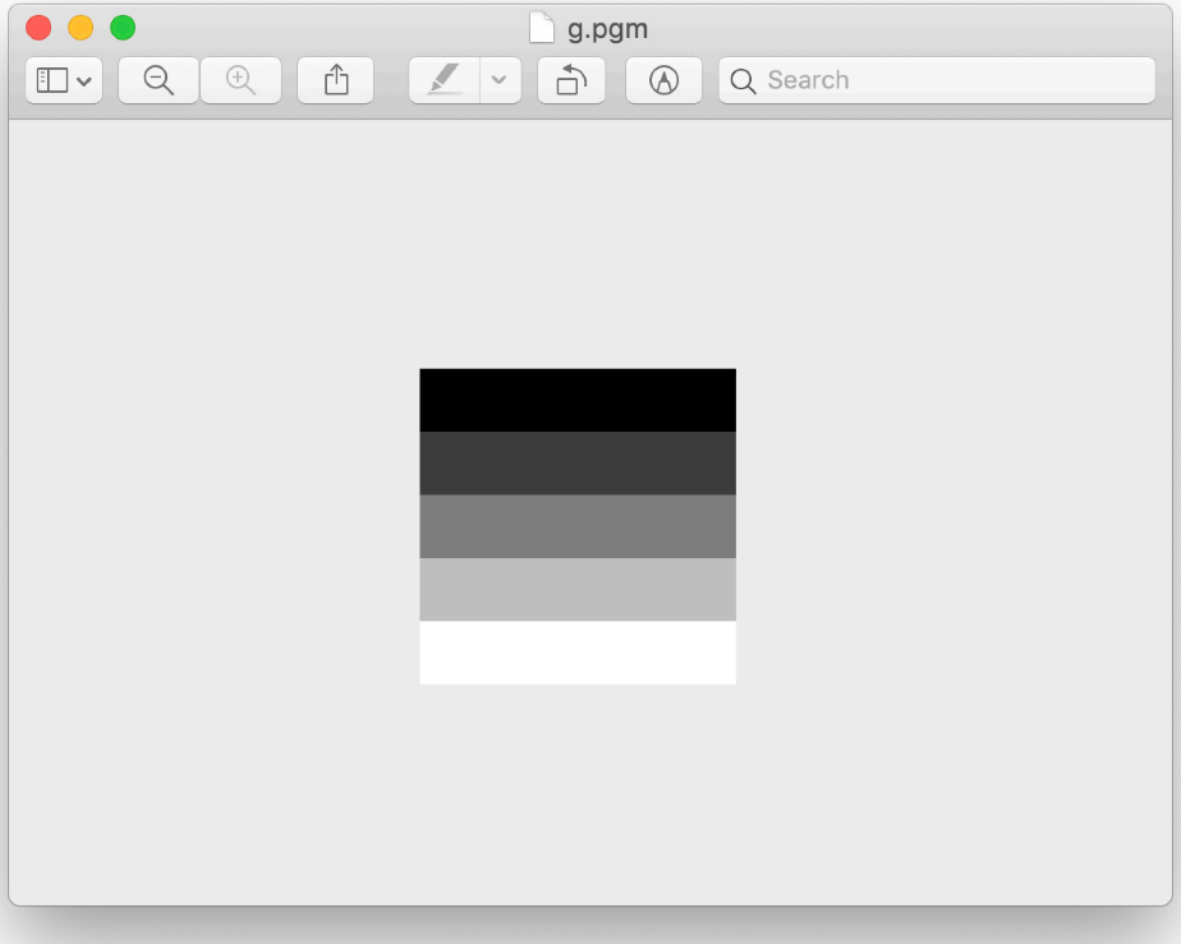
1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

```
P2
5 5 # 5x5 grid
1 # Max pixel value

# Draw the picture:
1 0 0 0 1
1 0 1 1 0
1 0 0 0 1
1 0 1 0 1
1 0 1 1 0

# ... and save as 'R.pgm'
```



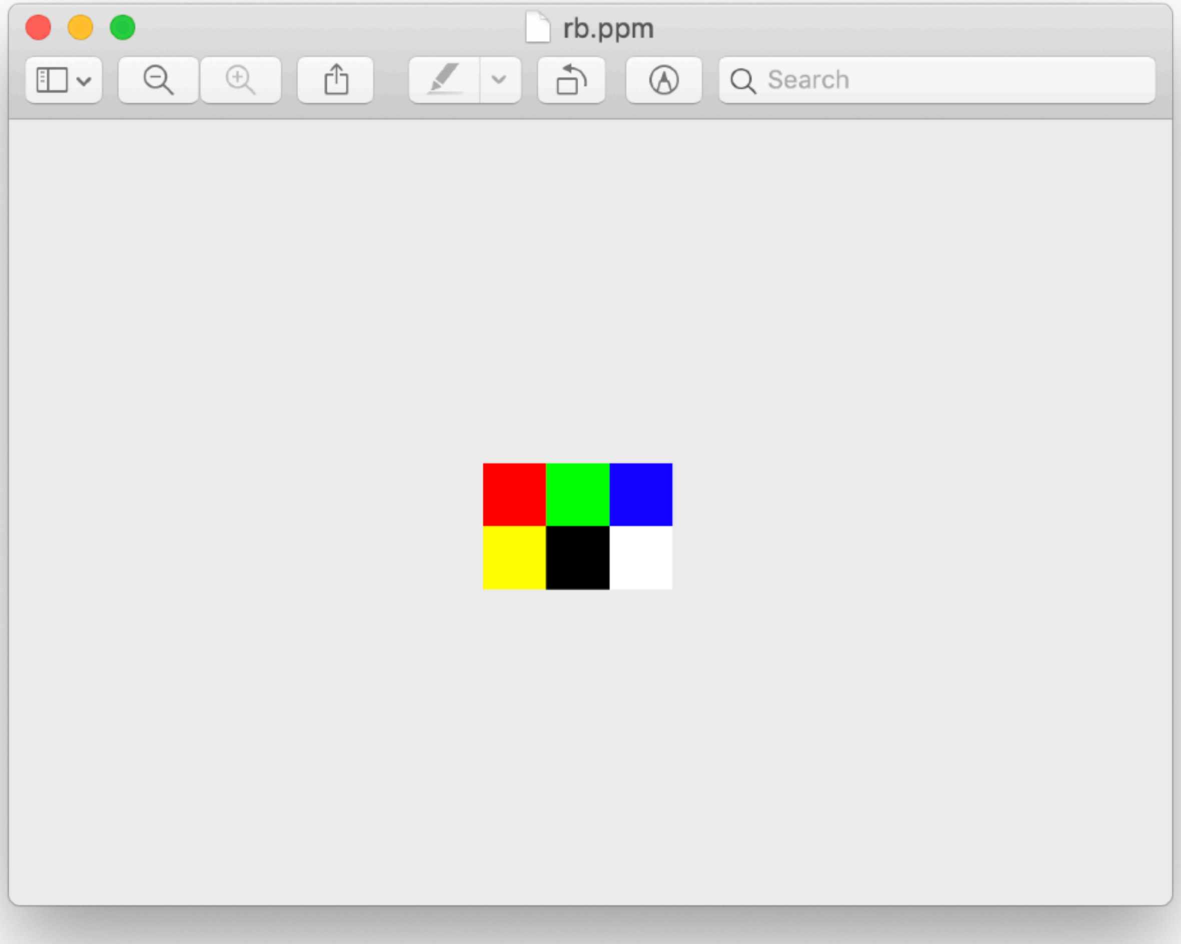
```
P2
5 5 # 5x5 grid
1 # Max pixel value

# Draw the picture:
1 0 0 0 1
1 0 1 1 0
1 0 0 0 1
1 0 1 0 1
1 0 1 1 0

# ... and save as 'R.pgm'
```

@damncabbage



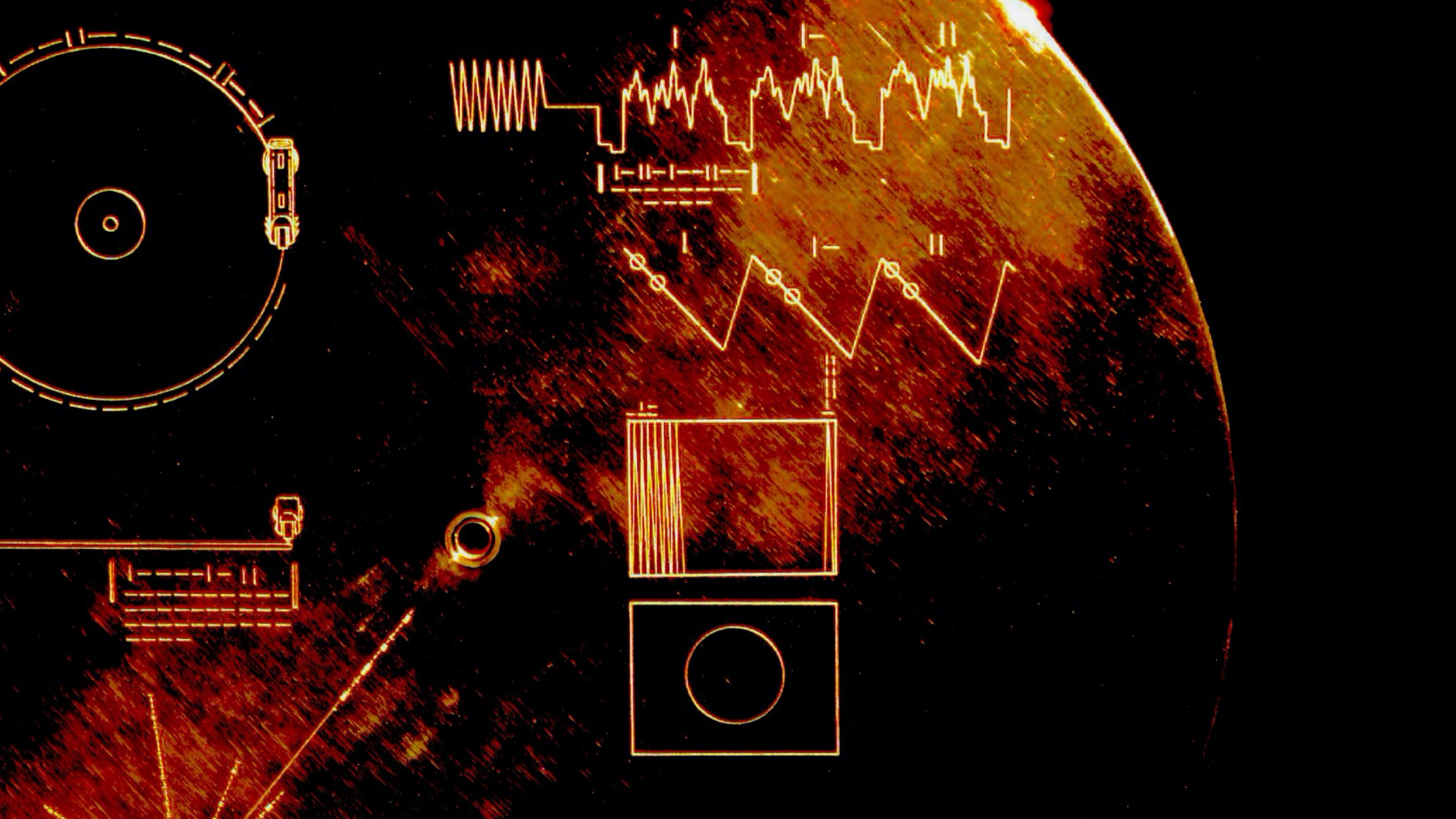
```
P2
5 5 # 5x5 grid
4 # Max pixel value

# Draw the picture:
0 0 0 0 0
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4

# ... and save as 'g.pgm'
```

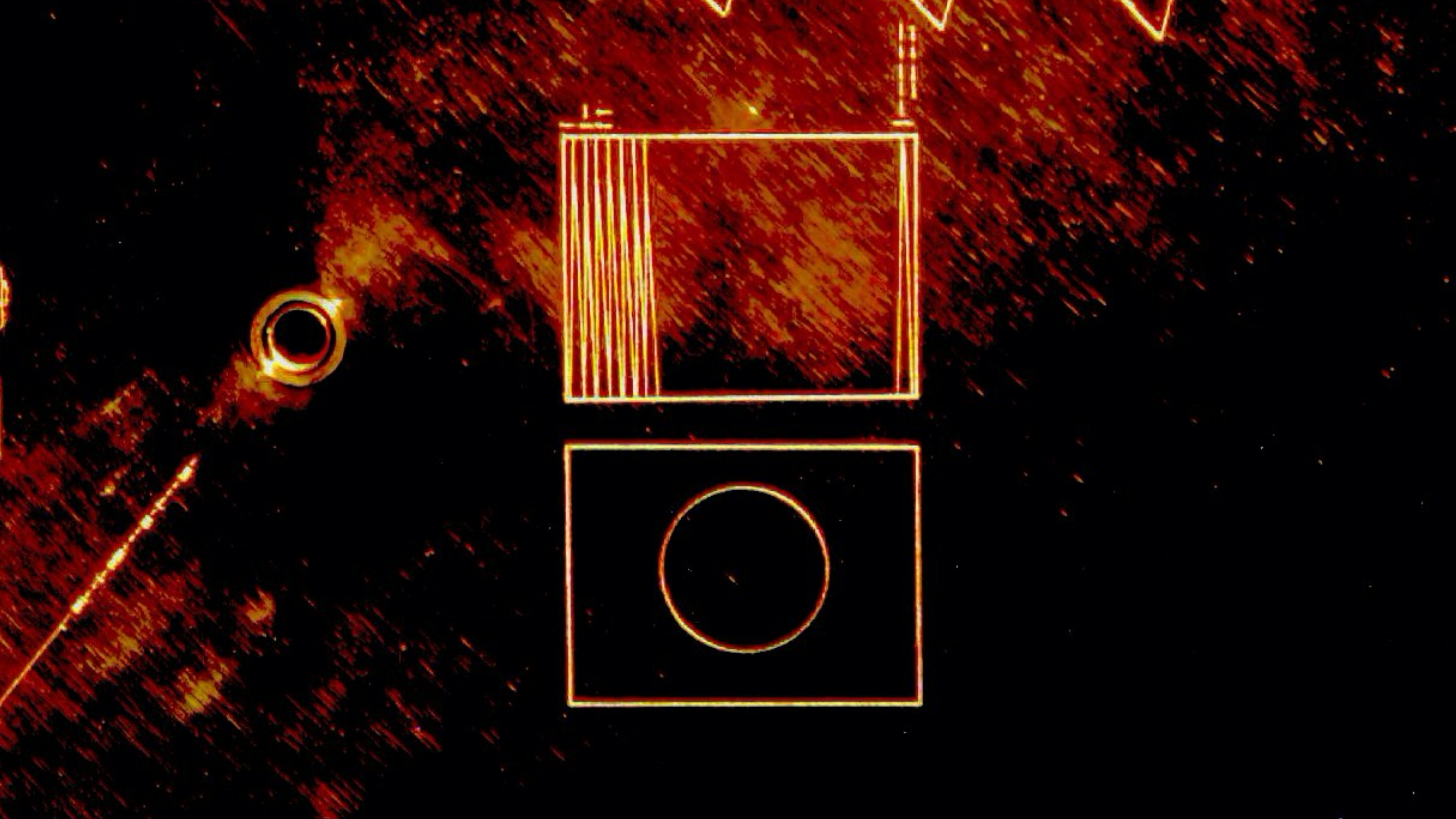
@damncabbage

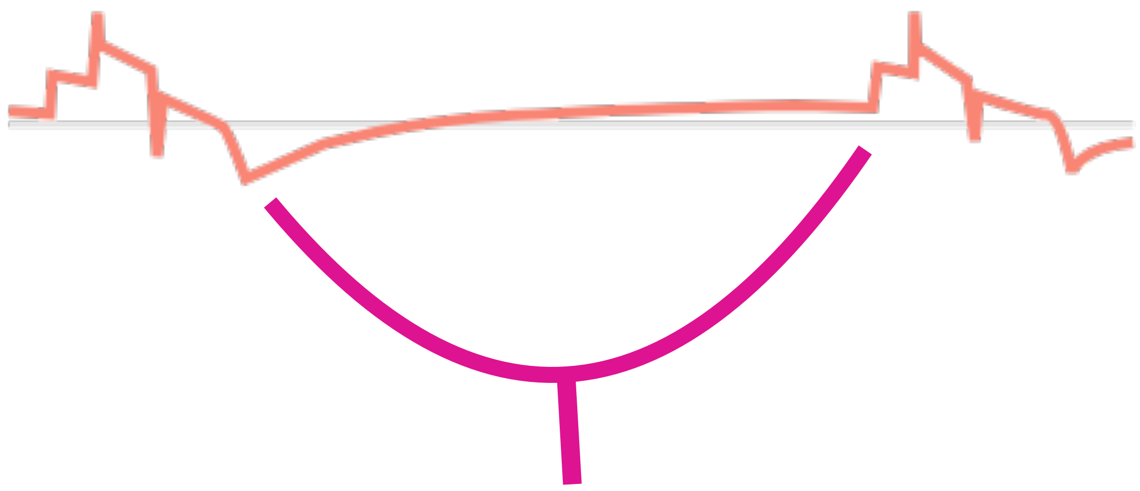


```
P3
3 2 # 3x2 grid
255 # Max pixel value for
    # each colour

# Draw the picture:
255   0   0  # red
  0 255   0  # green
  0   0 255  # blue
255 255   0  # yellow
  0   0   0  # black
255 255 255  # white
```

@damncabbage

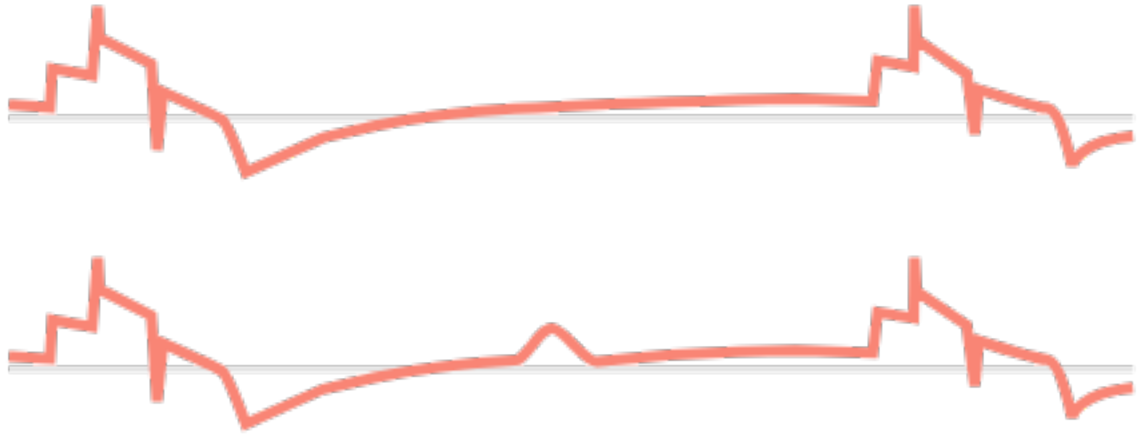


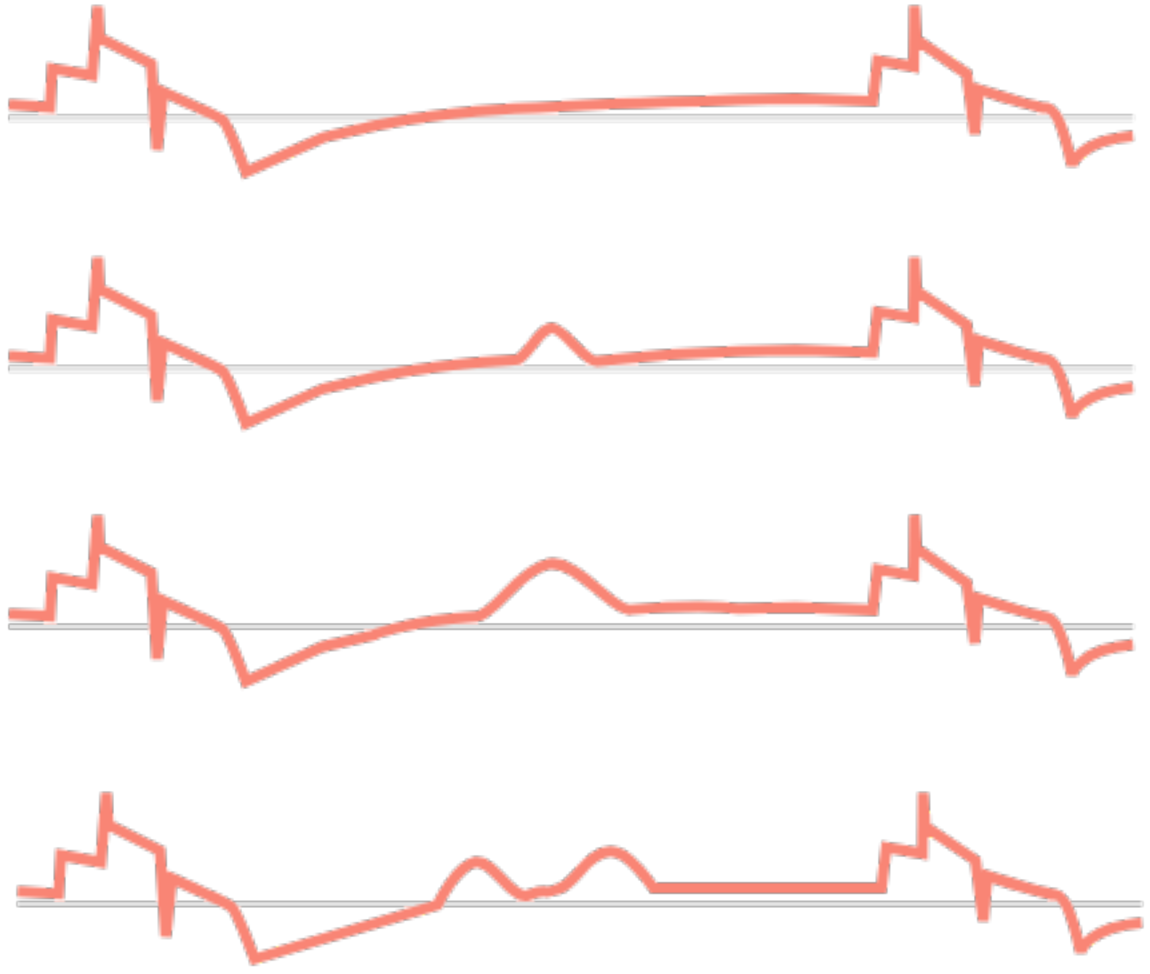


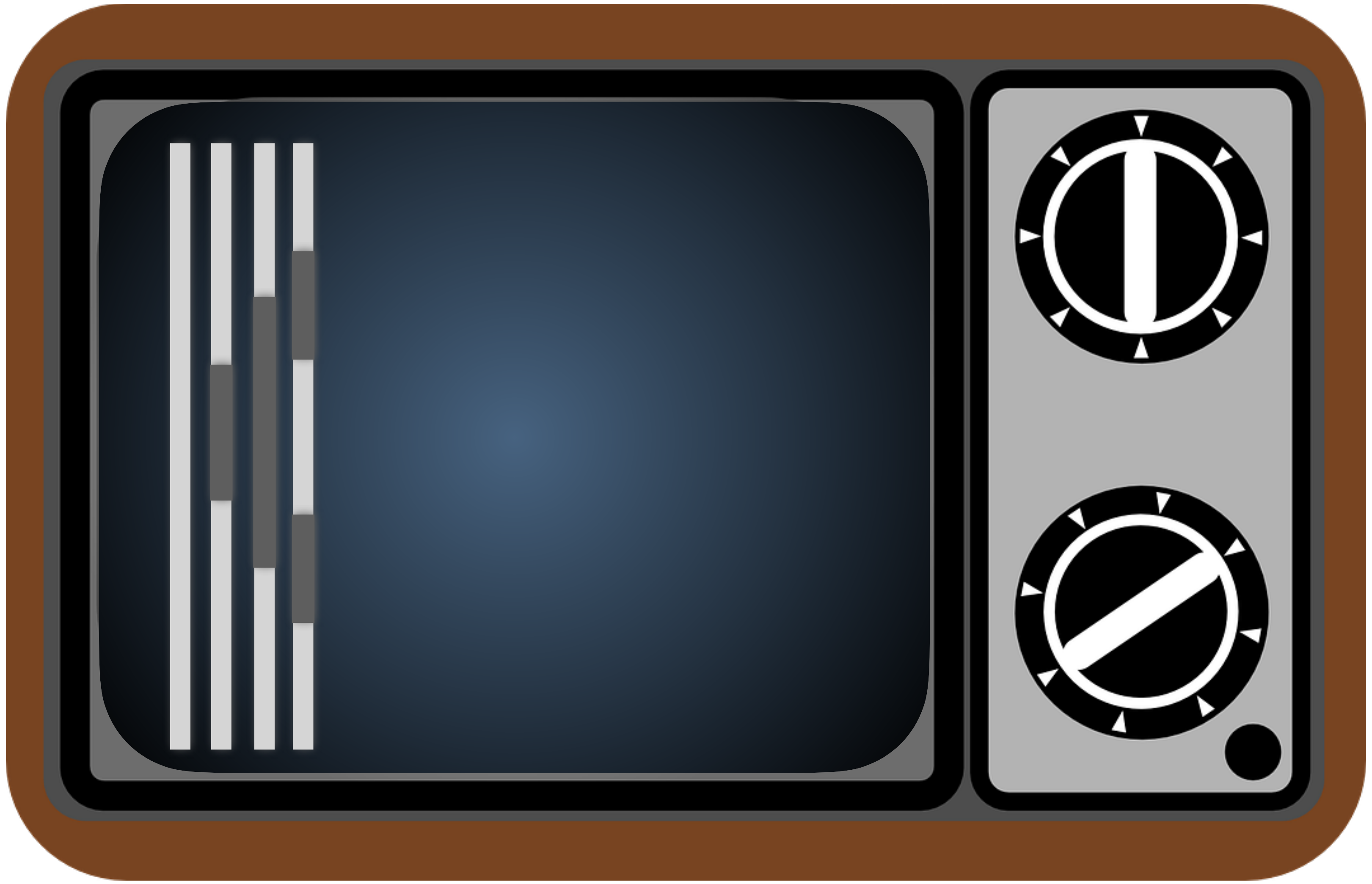


'Blank' line

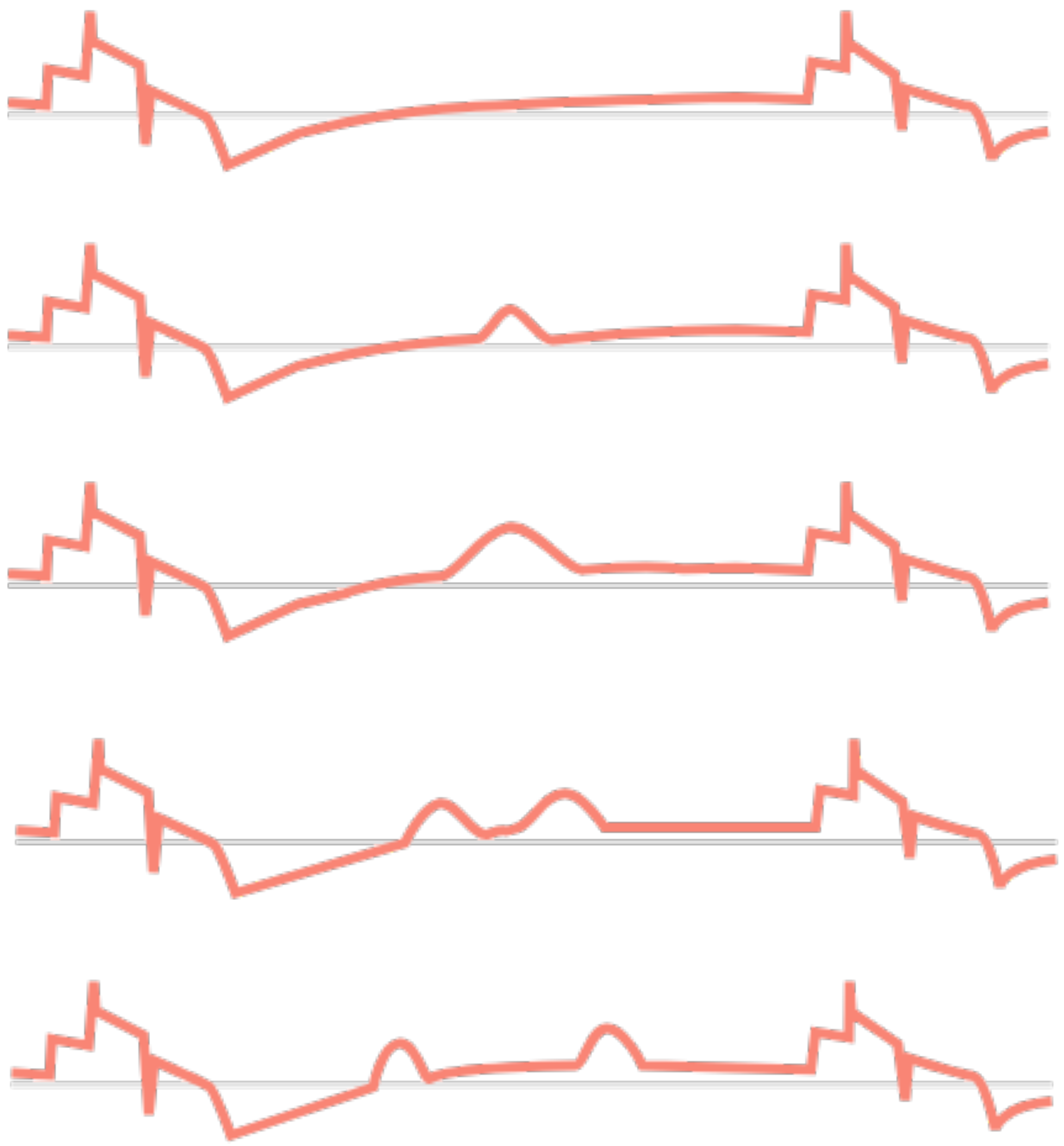
@damncabbage

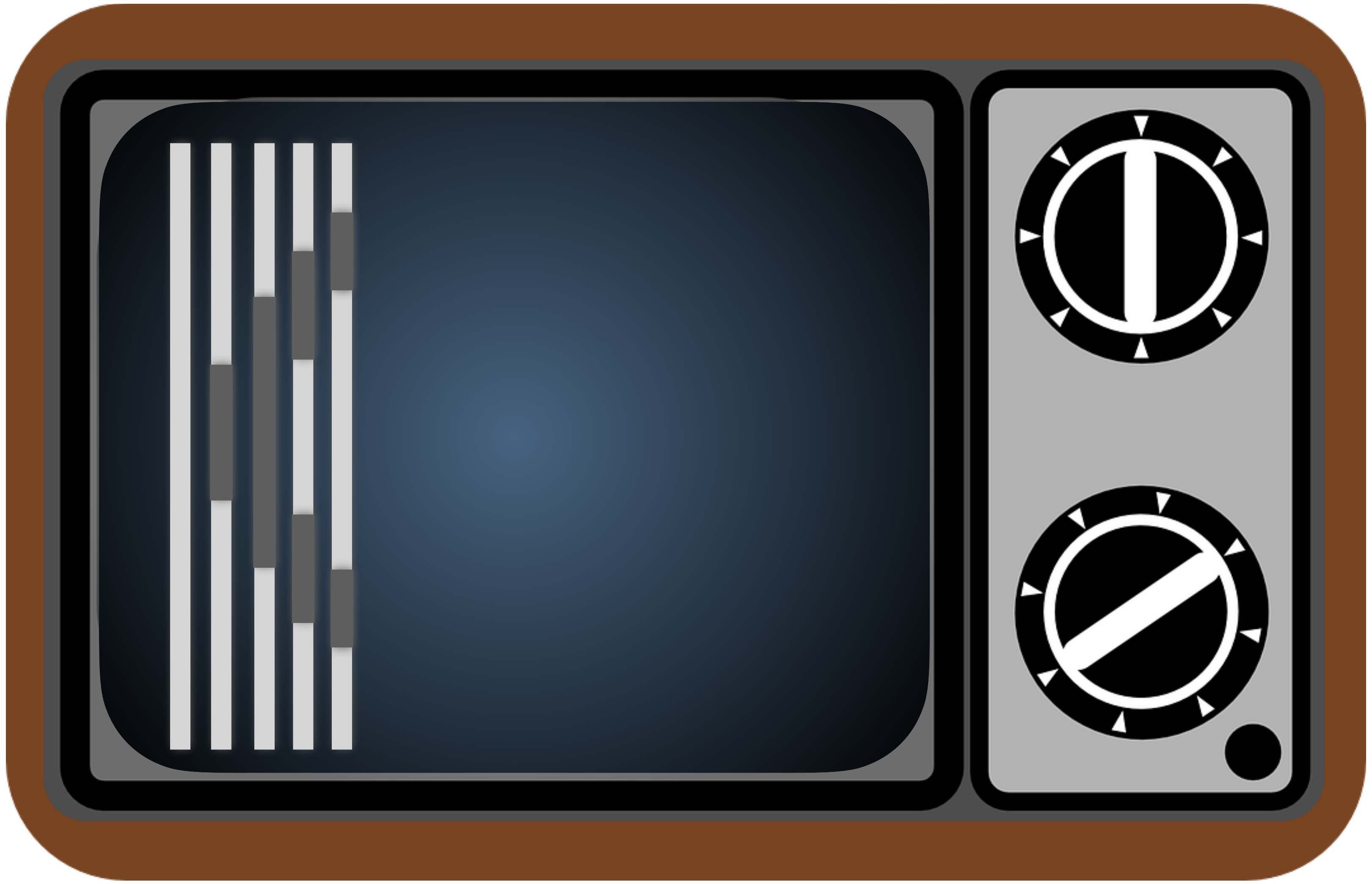


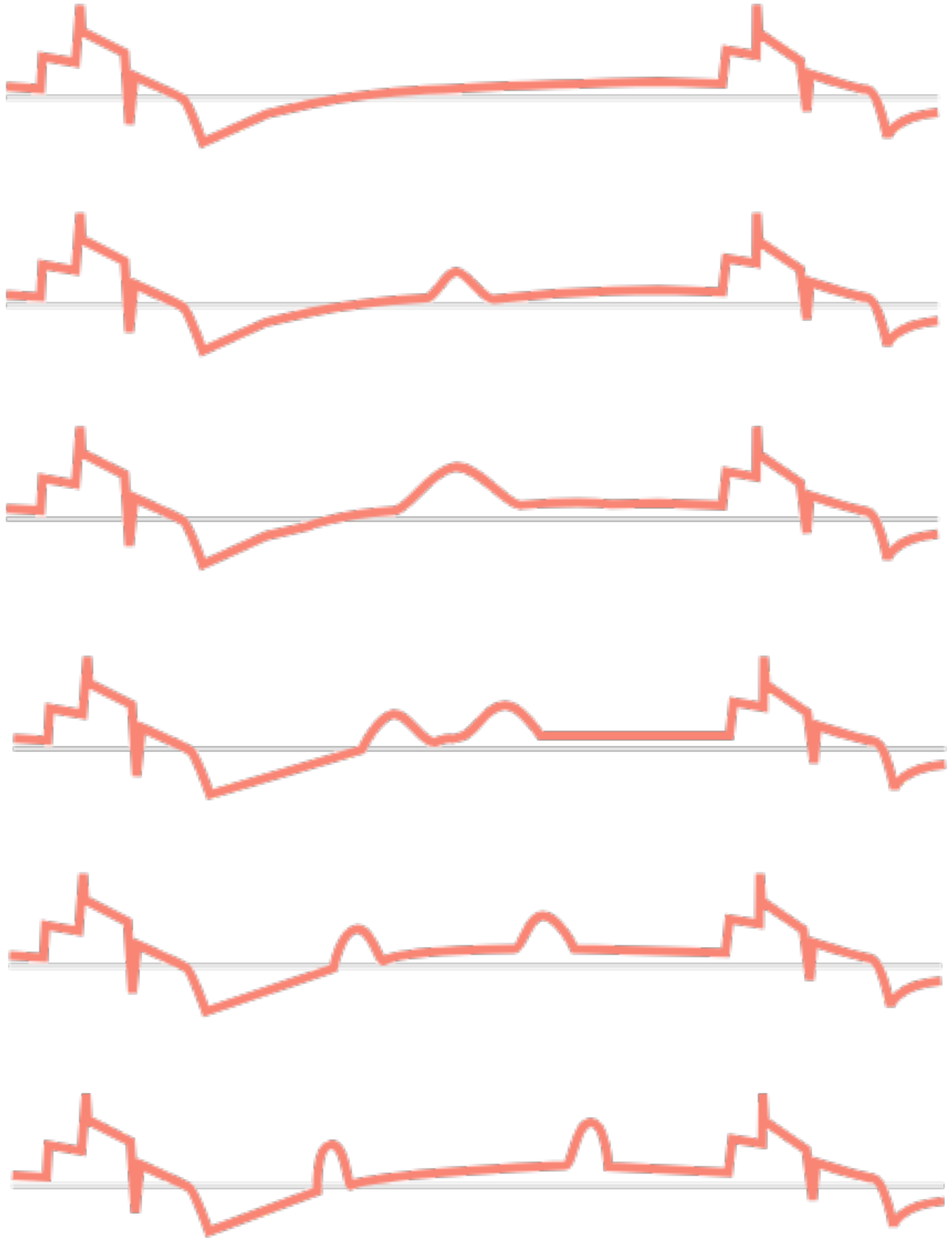


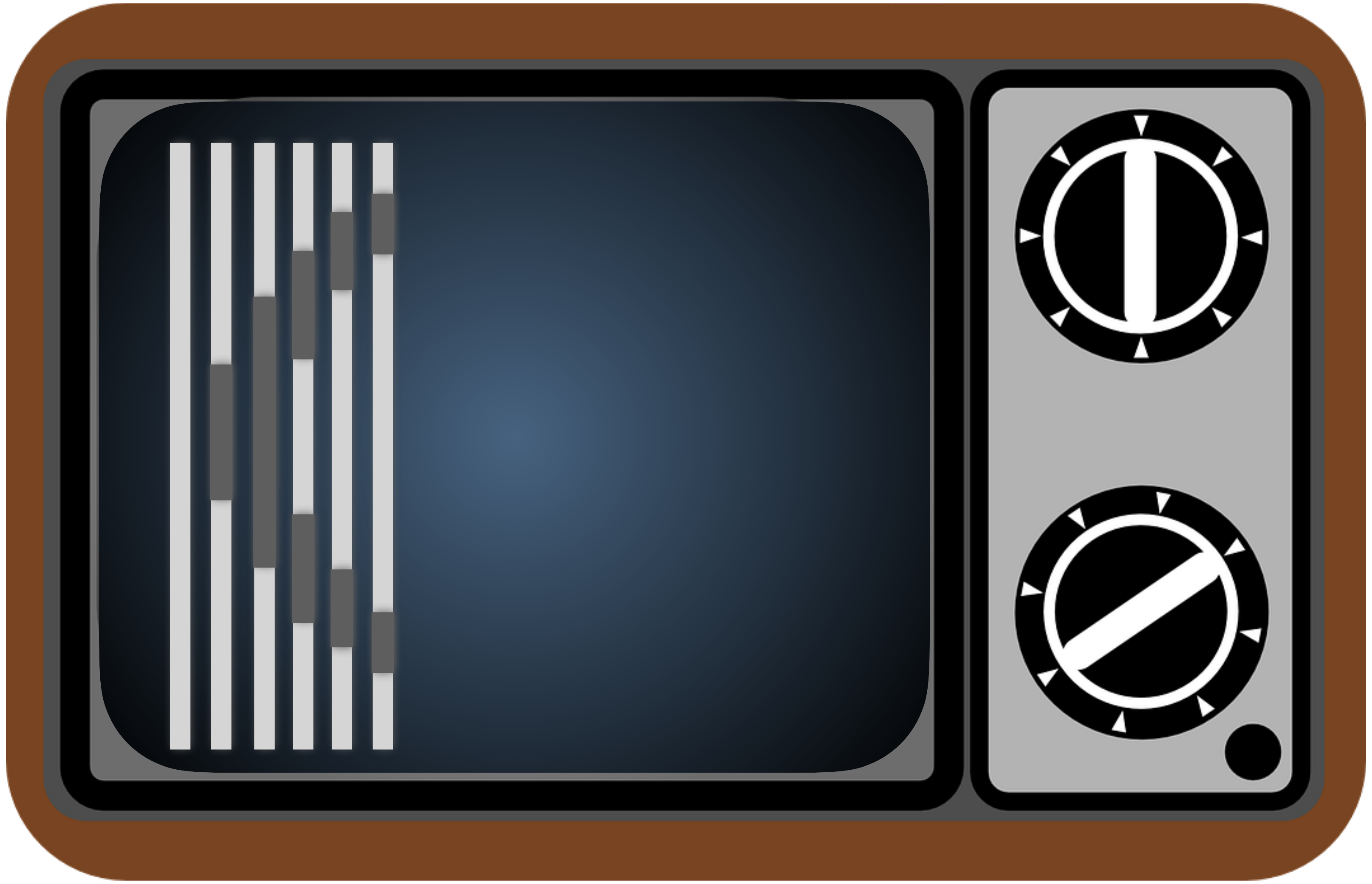


@damncabbage

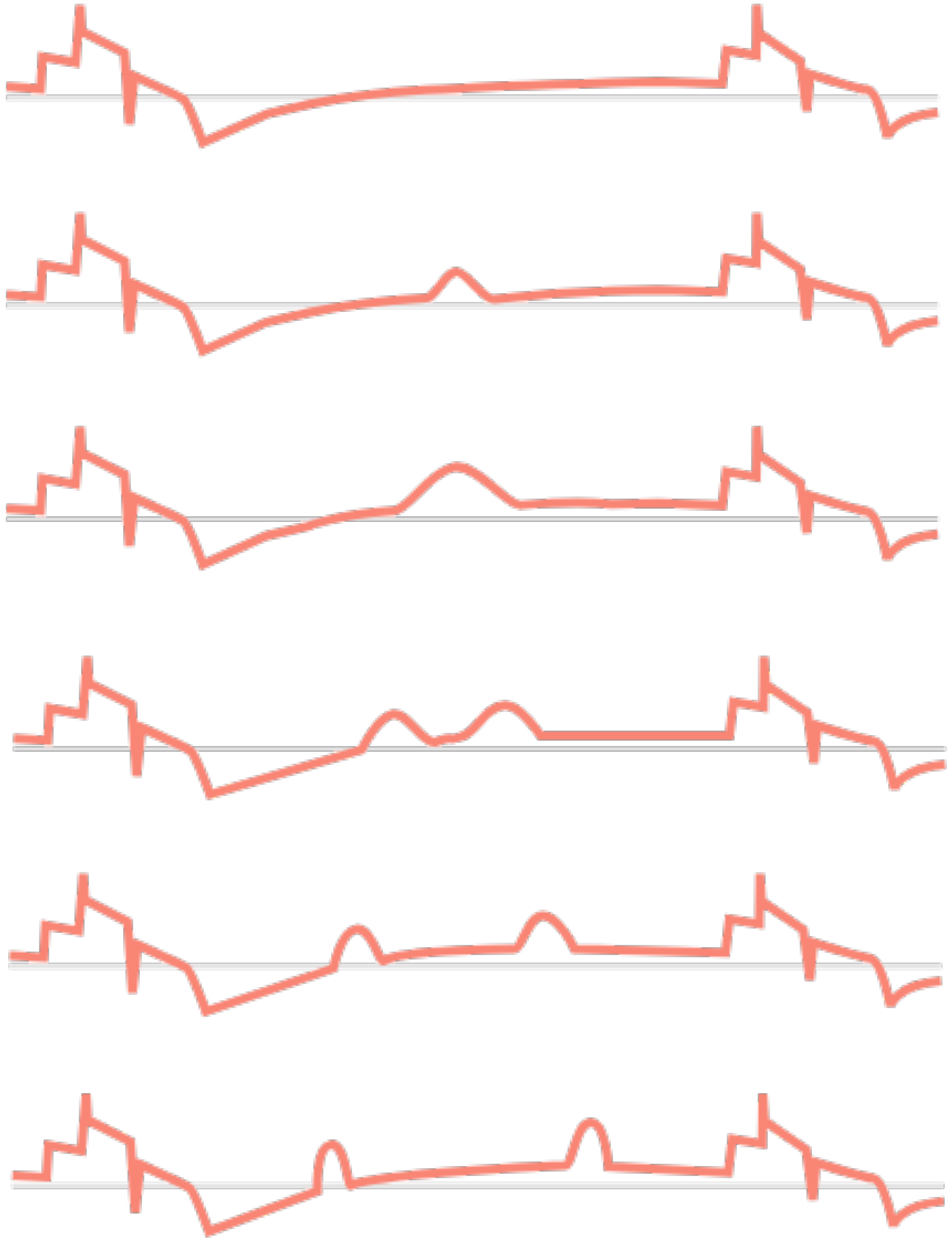


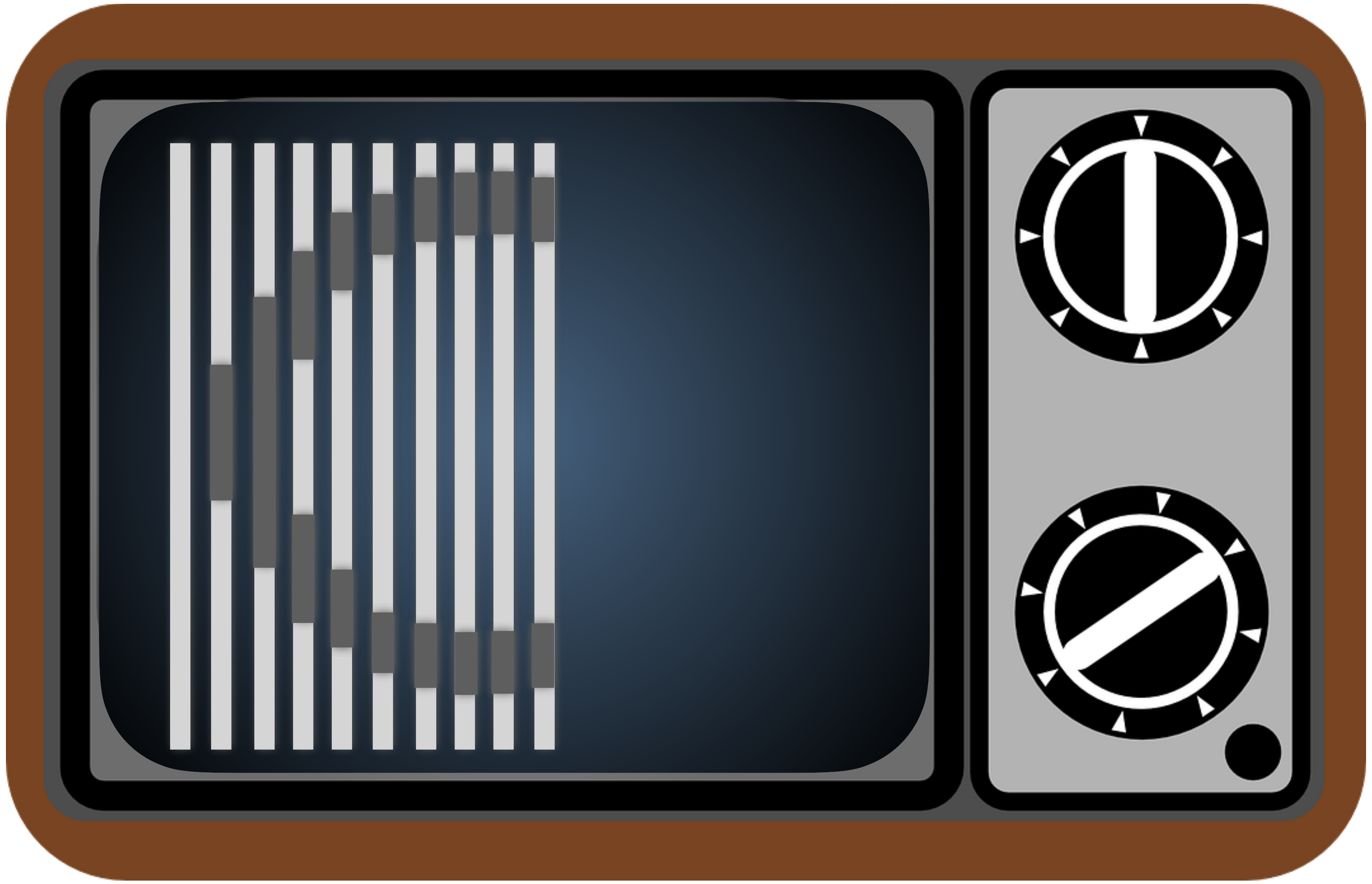


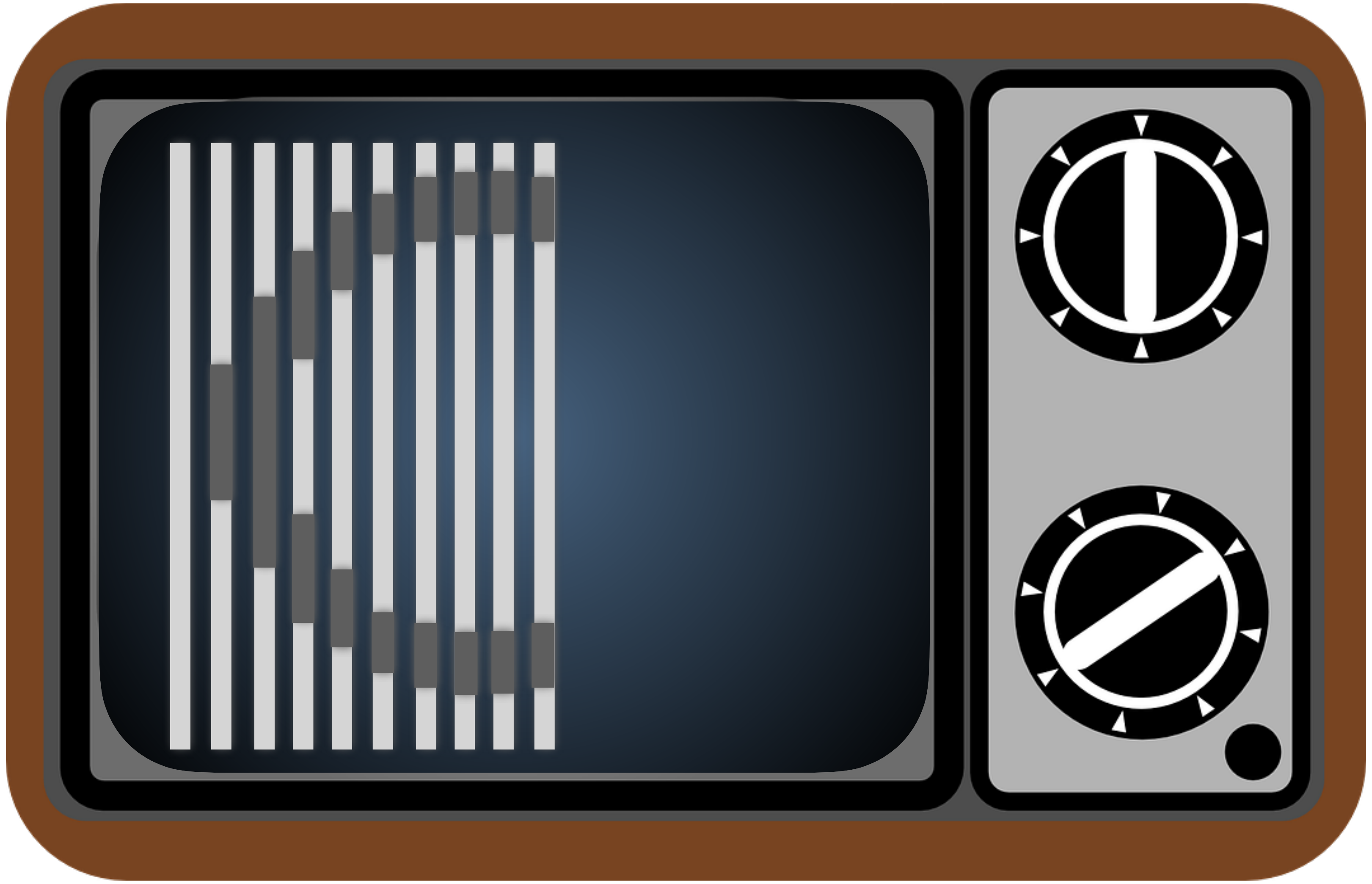


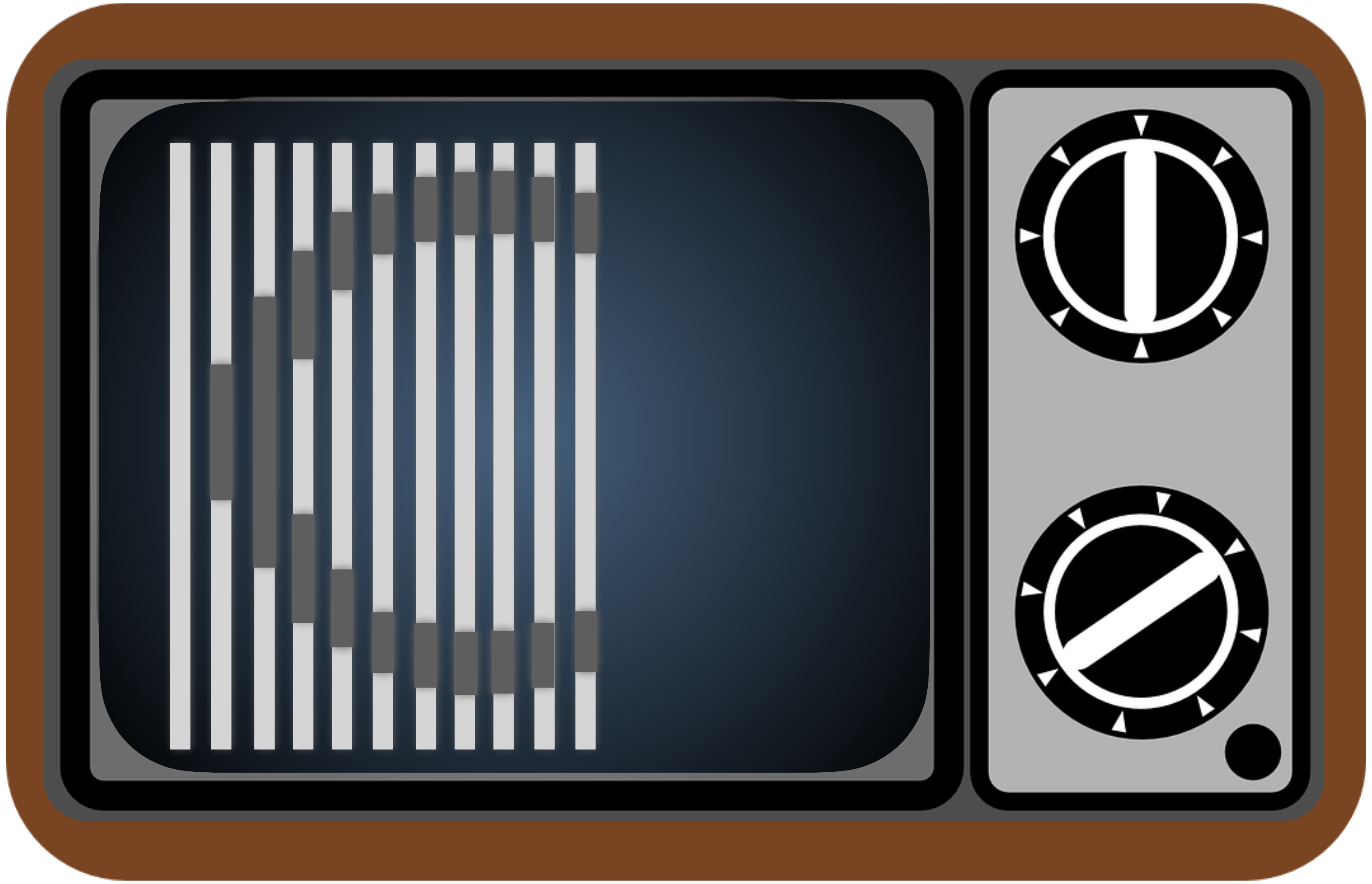


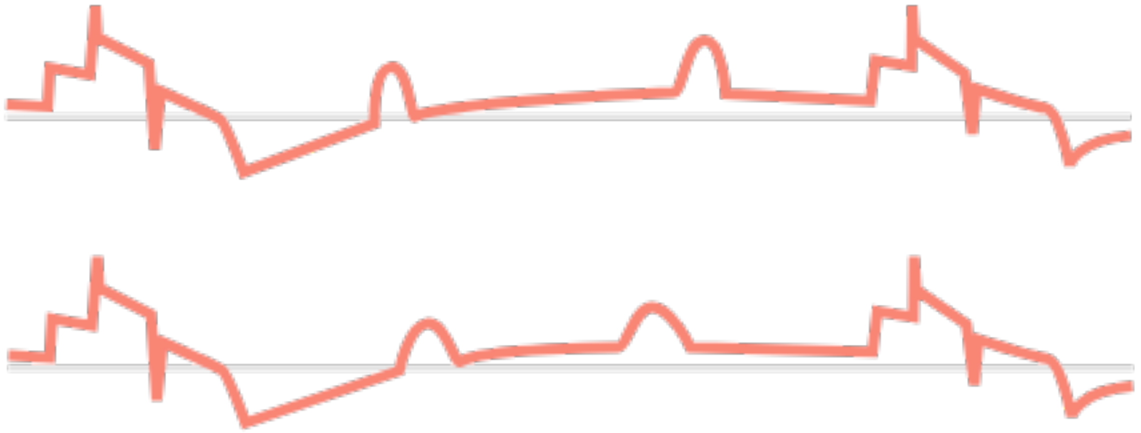
@damncabbage

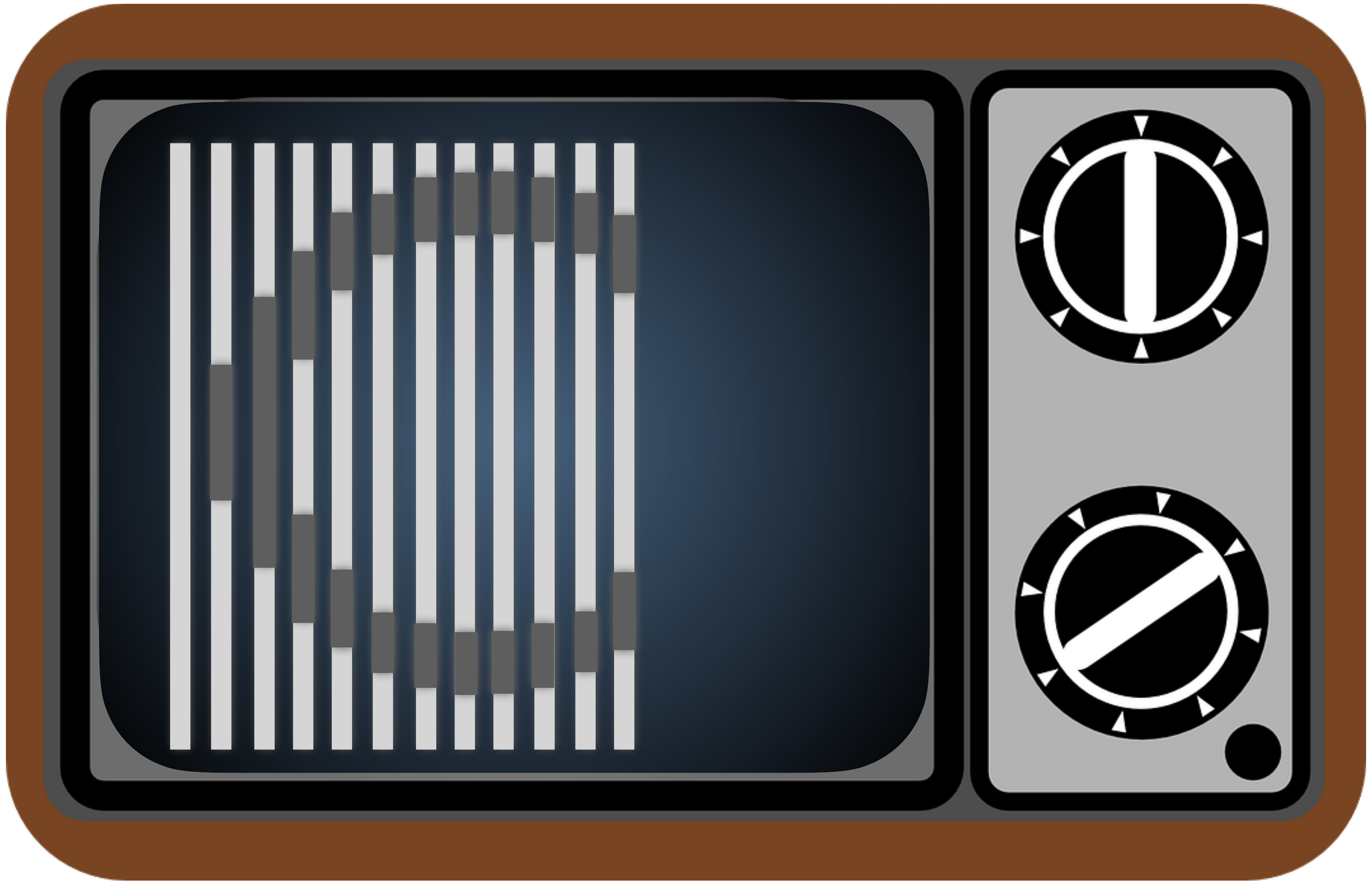


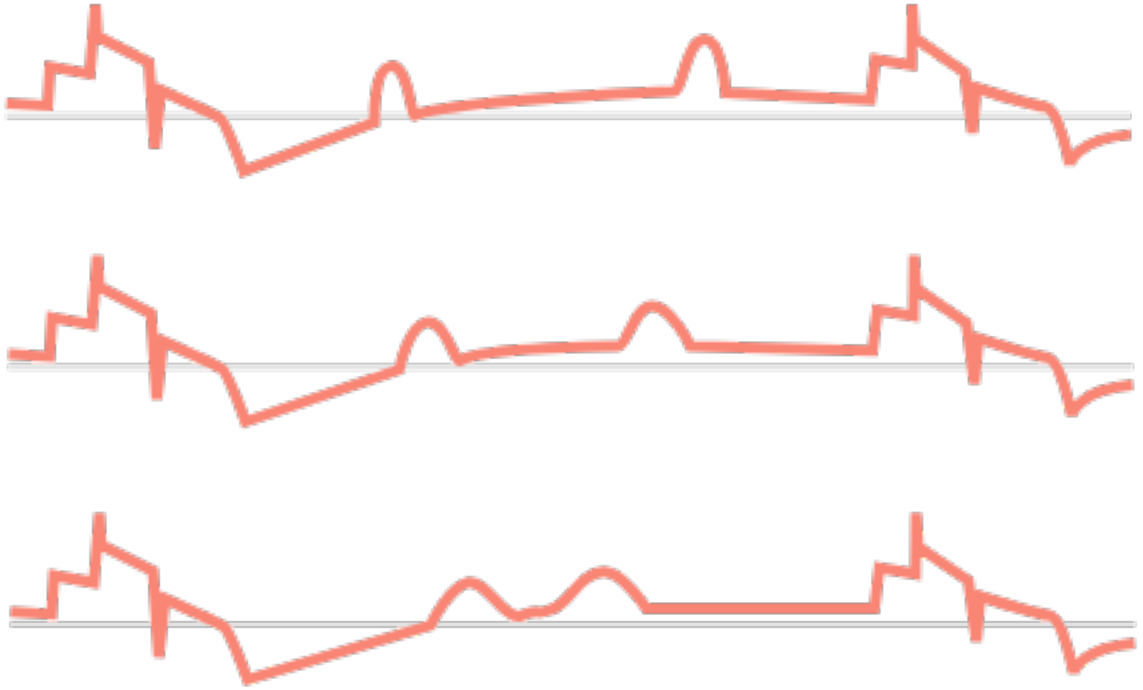


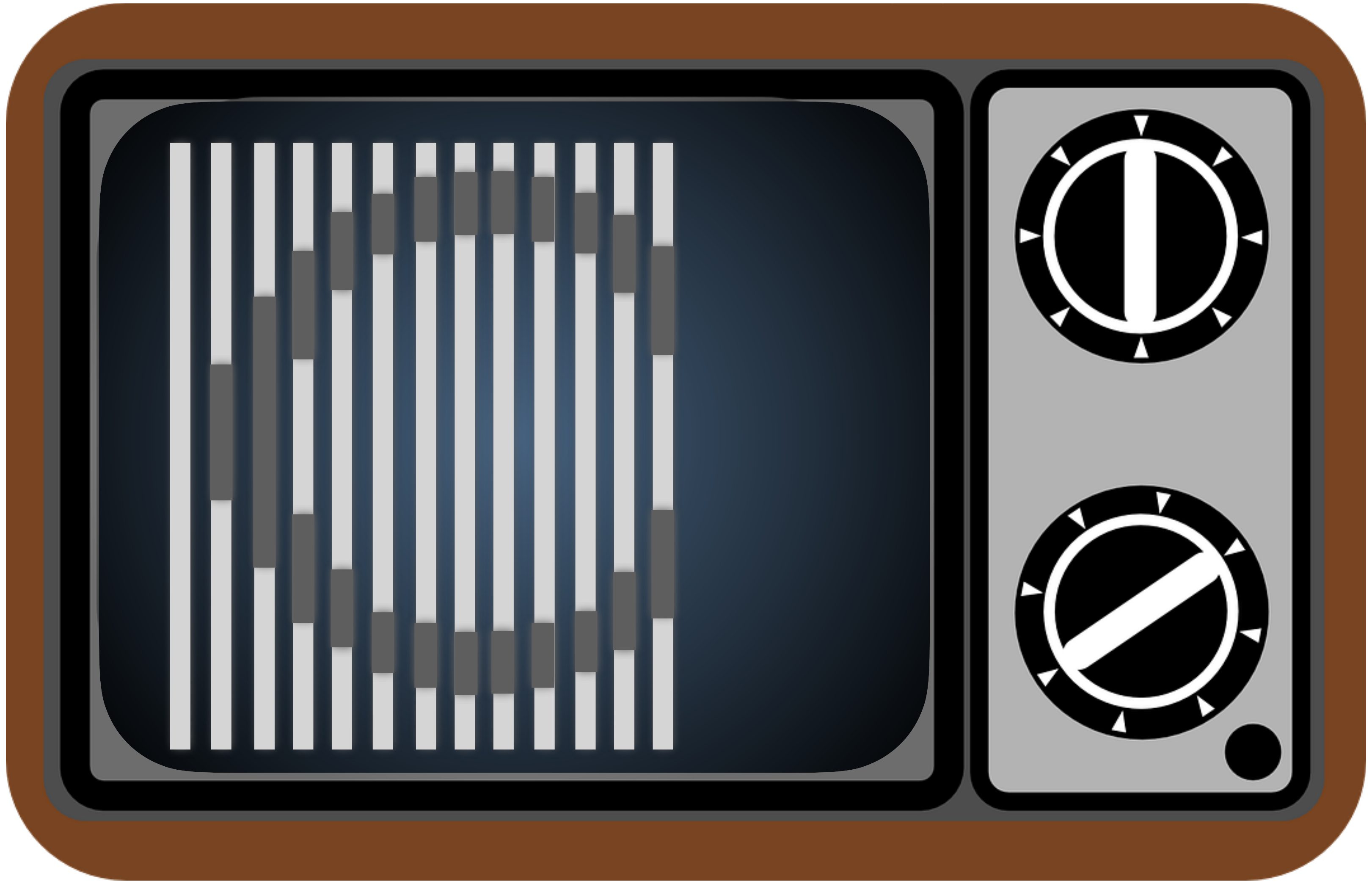


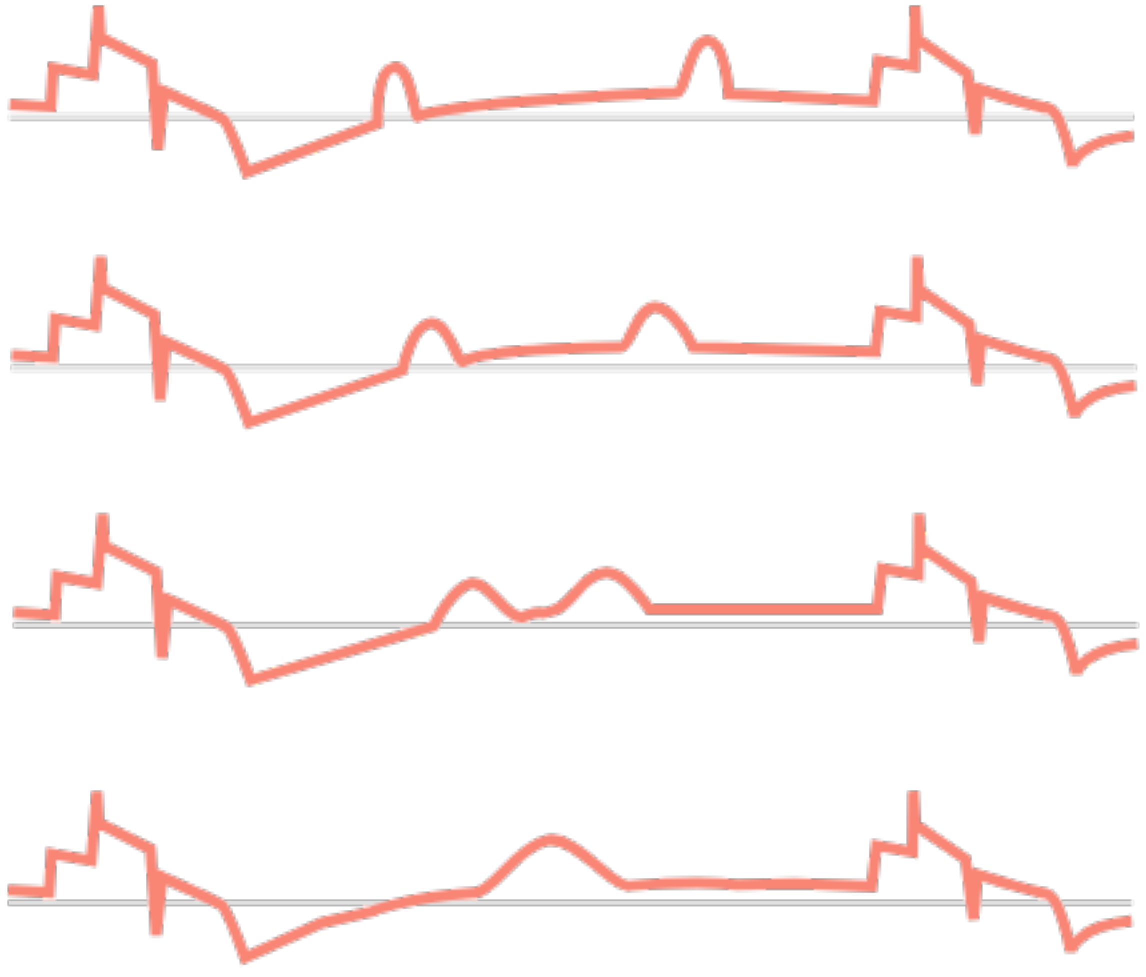


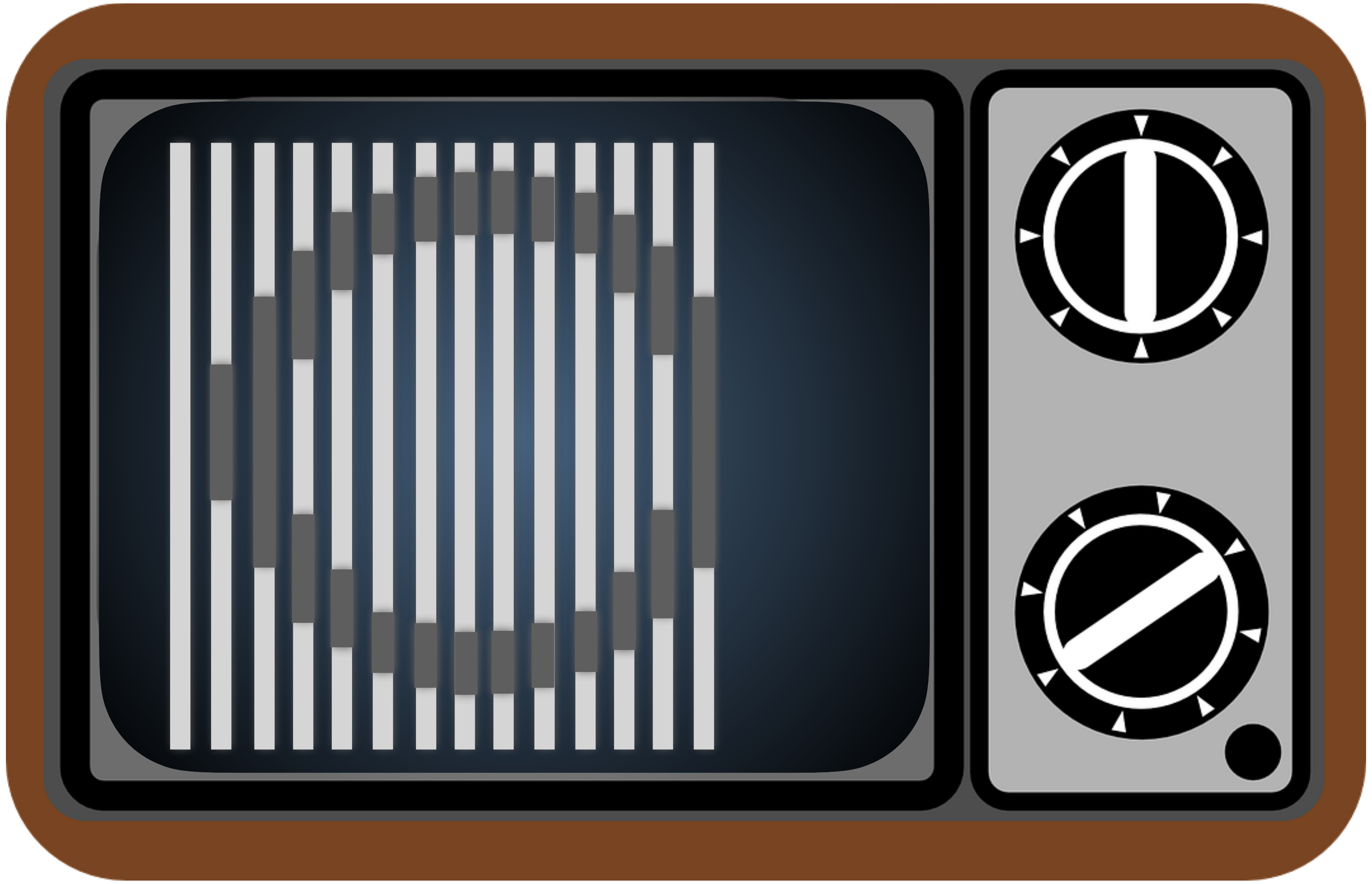


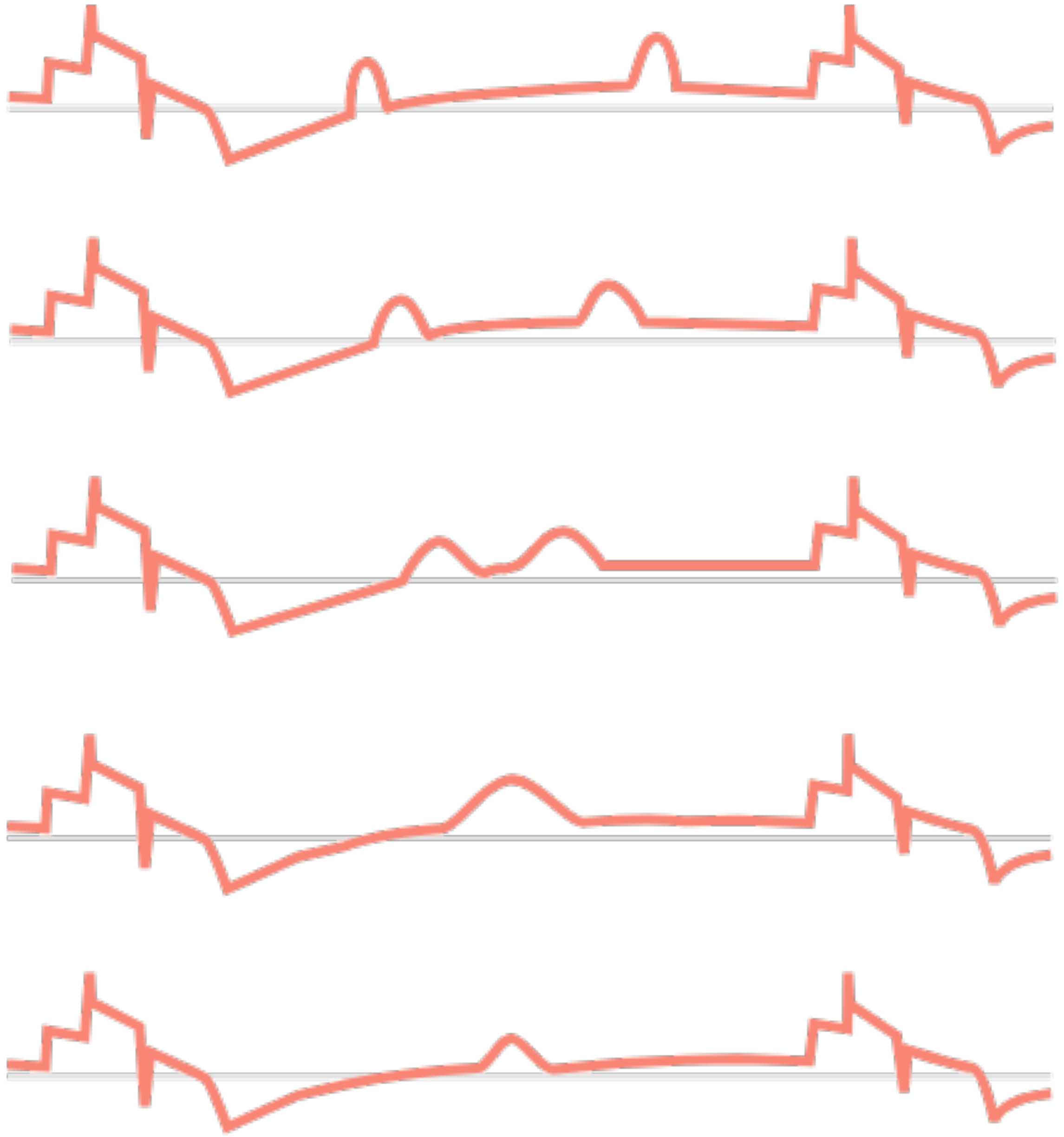


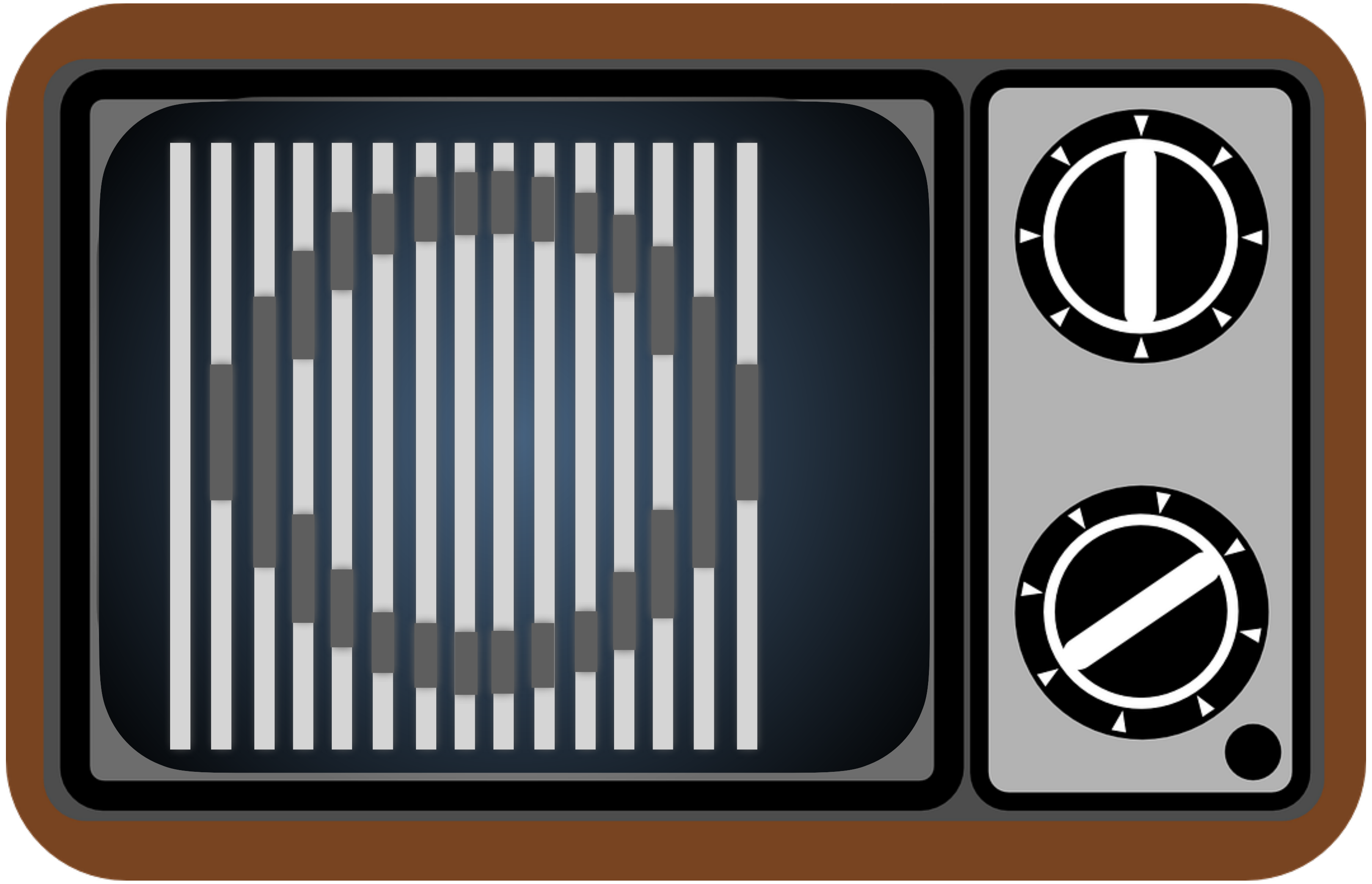


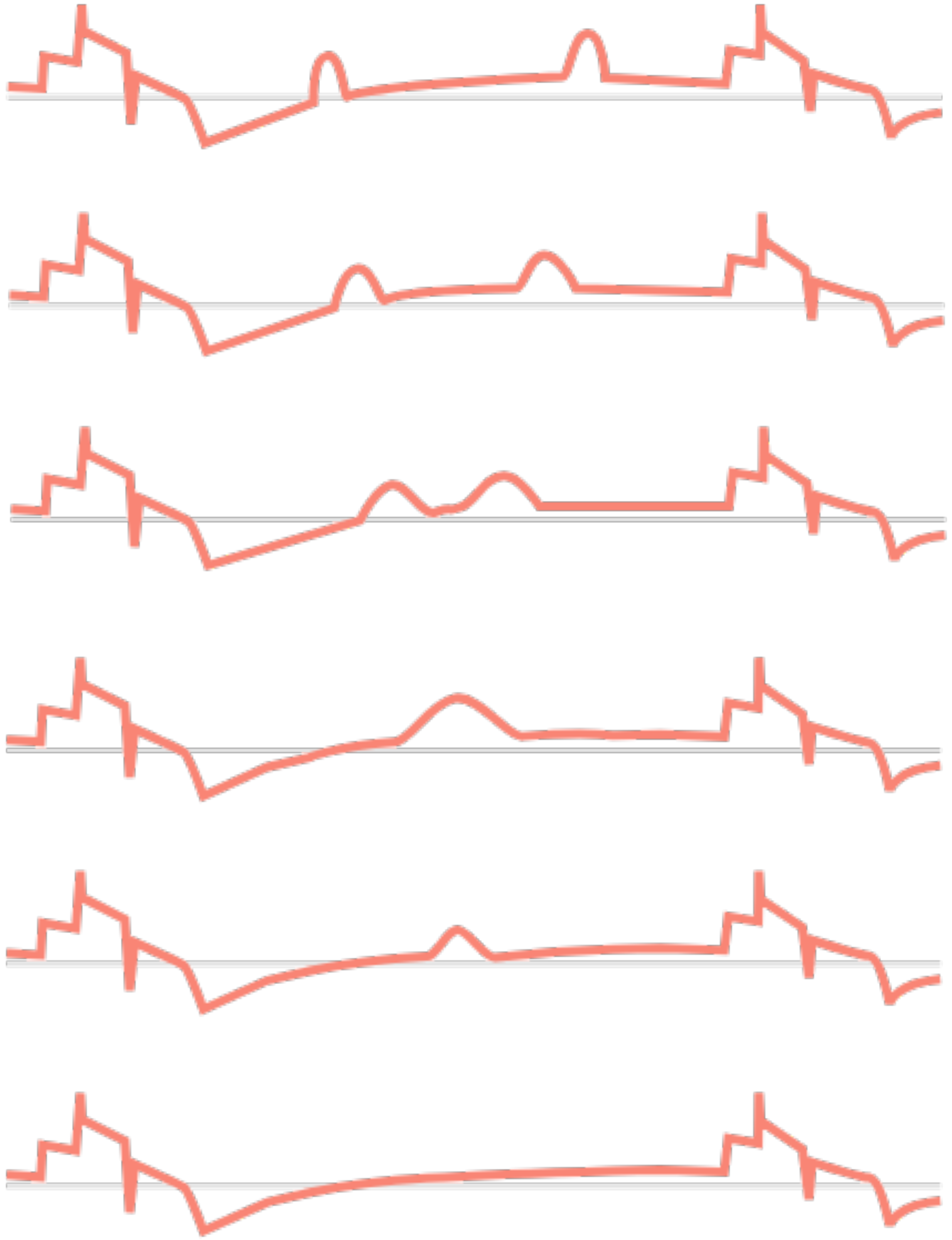


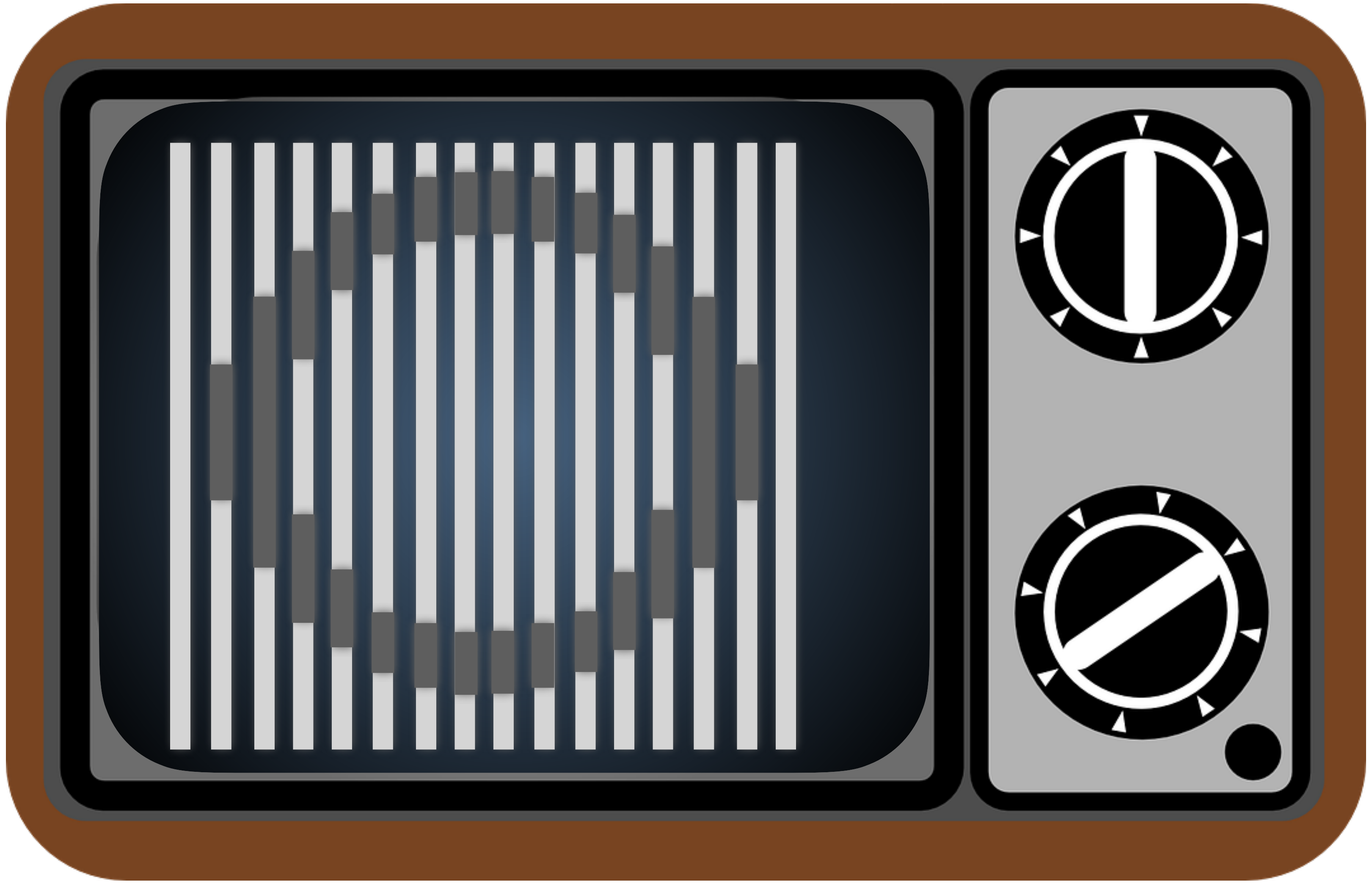


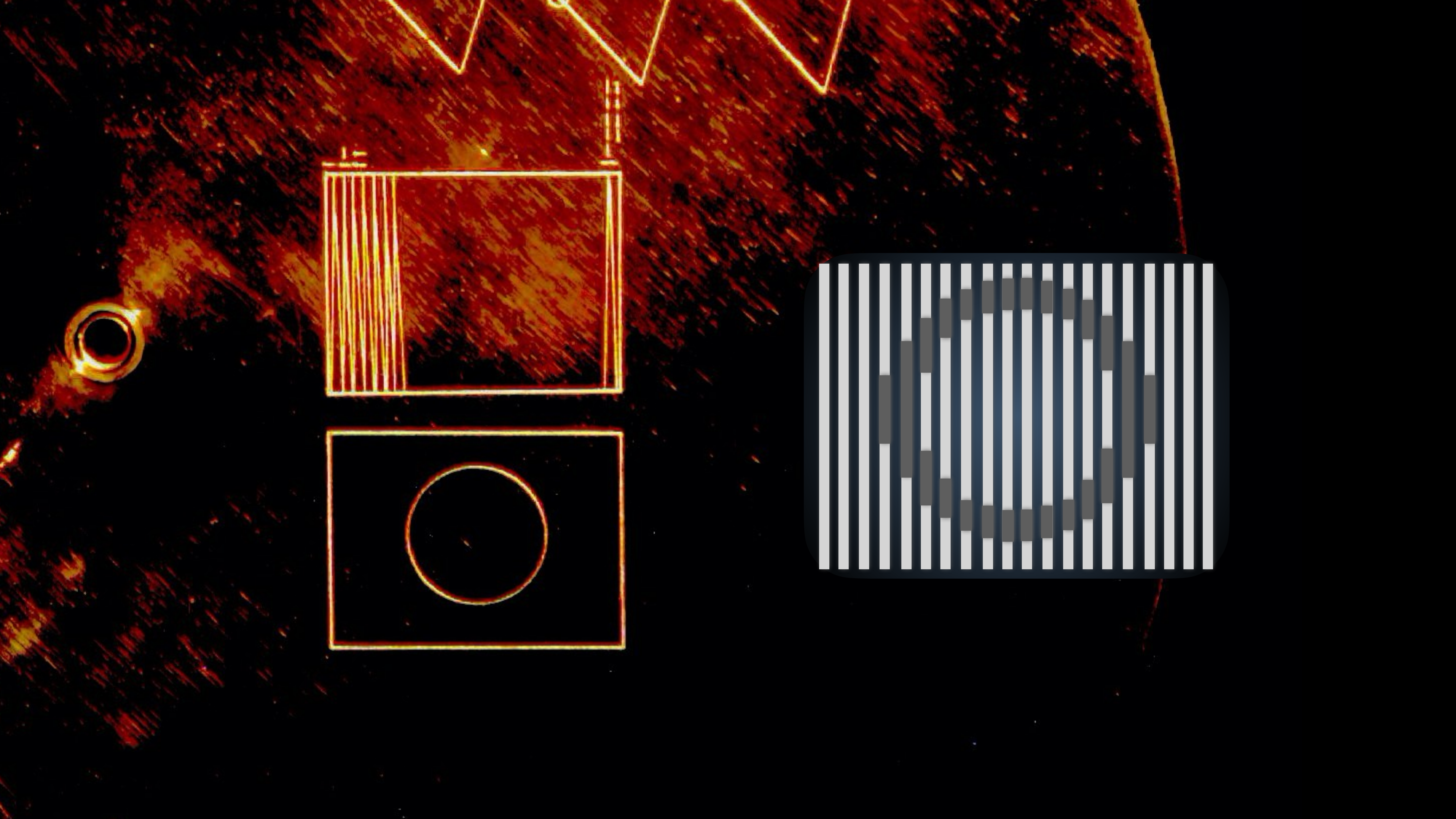






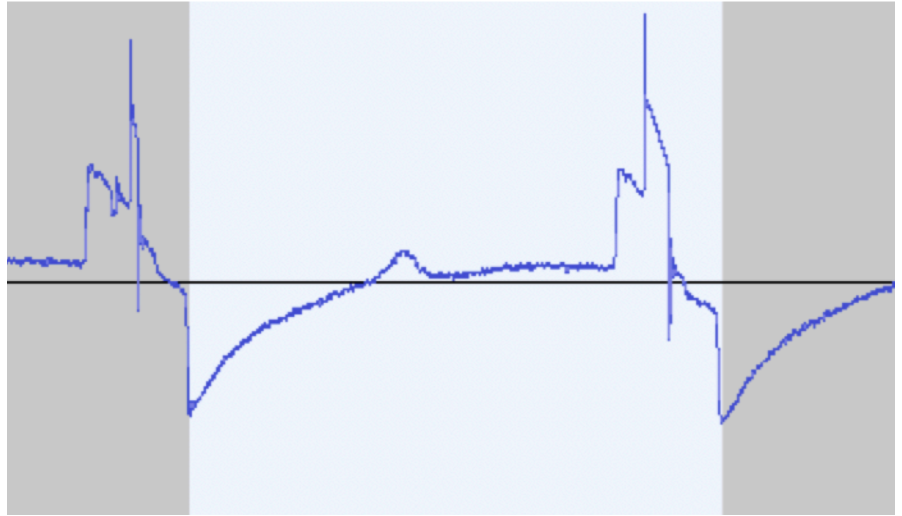






Read chunk of WAV as numbers

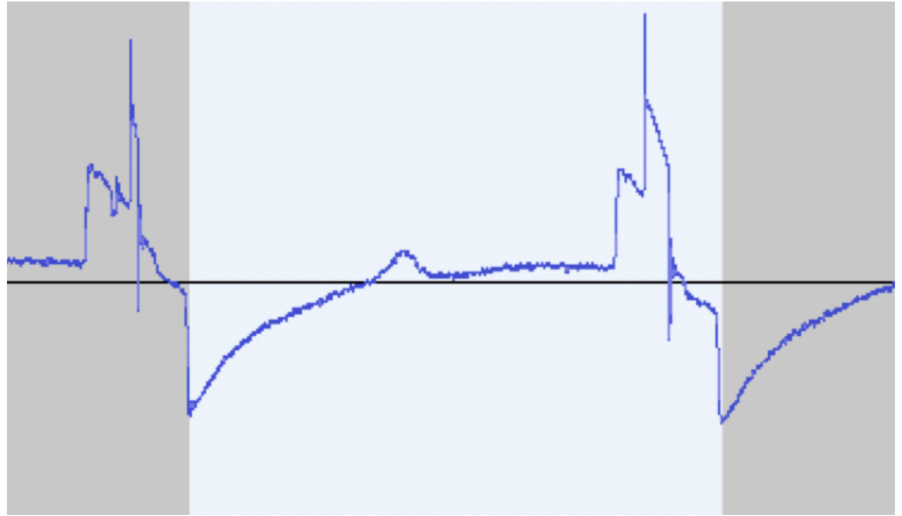
Read chunk of WAV as numbers



Read chunk of WAV as numbers

↓

Re-scale WAV numbers (-1 to 1) to image numbers (0 to 255)



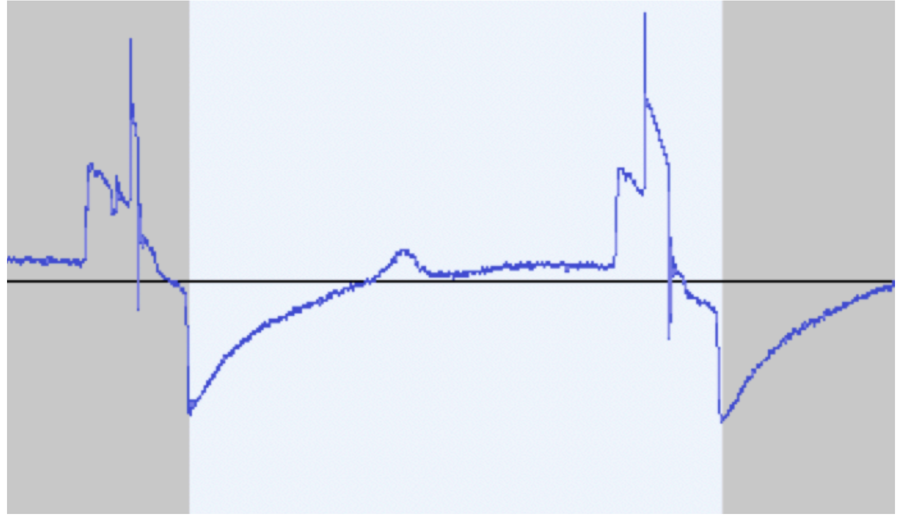
Read chunk of WAV as numbers

↓

Re-scale WAV numbers (-1 to 1)
to image numbers (0 to 255)

↓

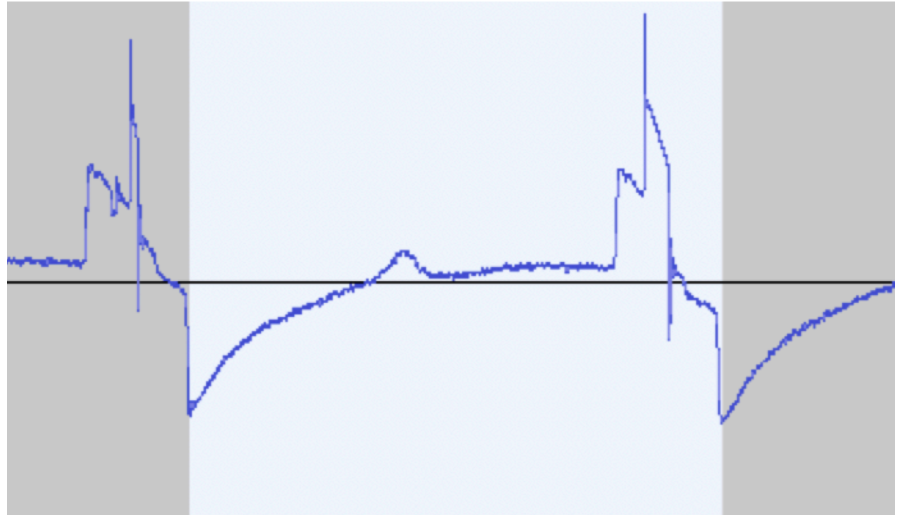
Add line to 'image' array

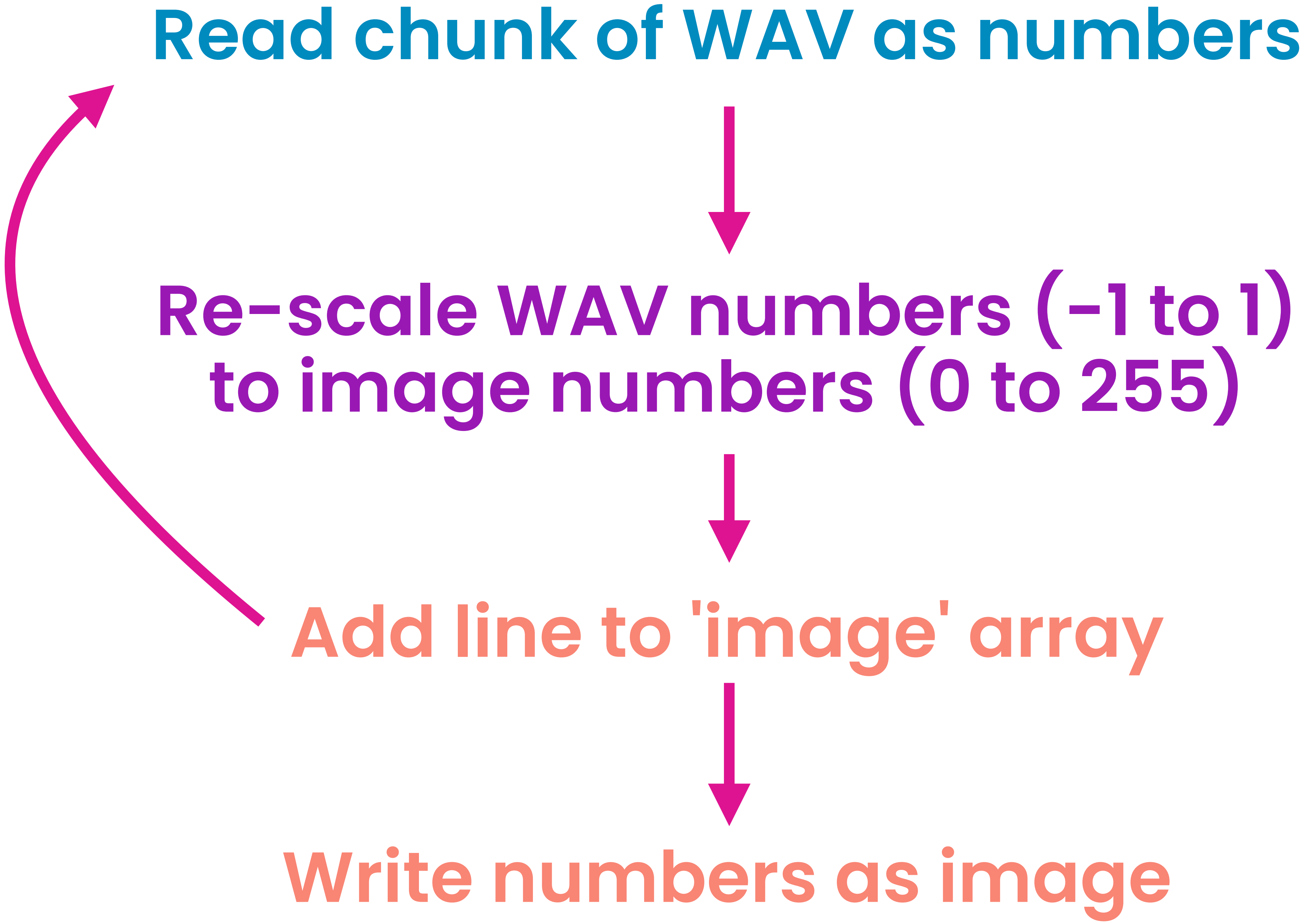


Read chunk of WAV as numbers

Re-scale WAV numbers (-1 to 1) to image numbers (0 to 255)

Add line to 'image' array



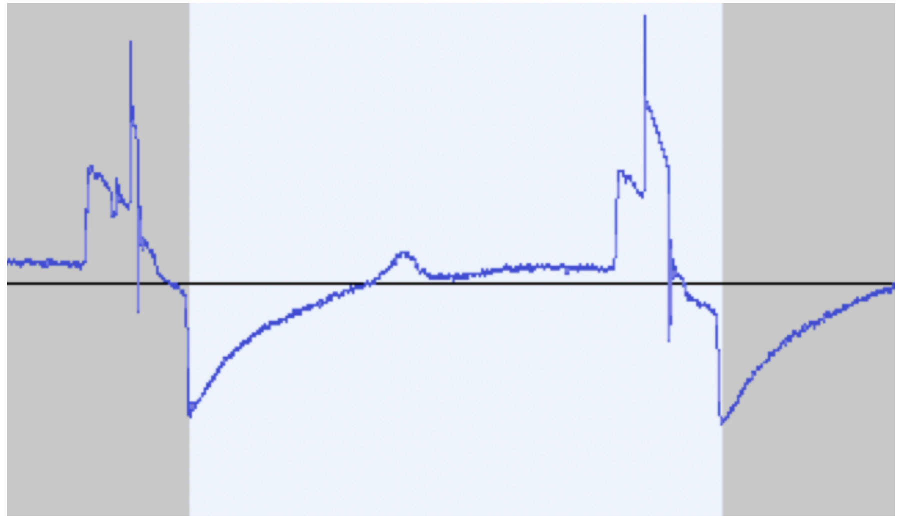


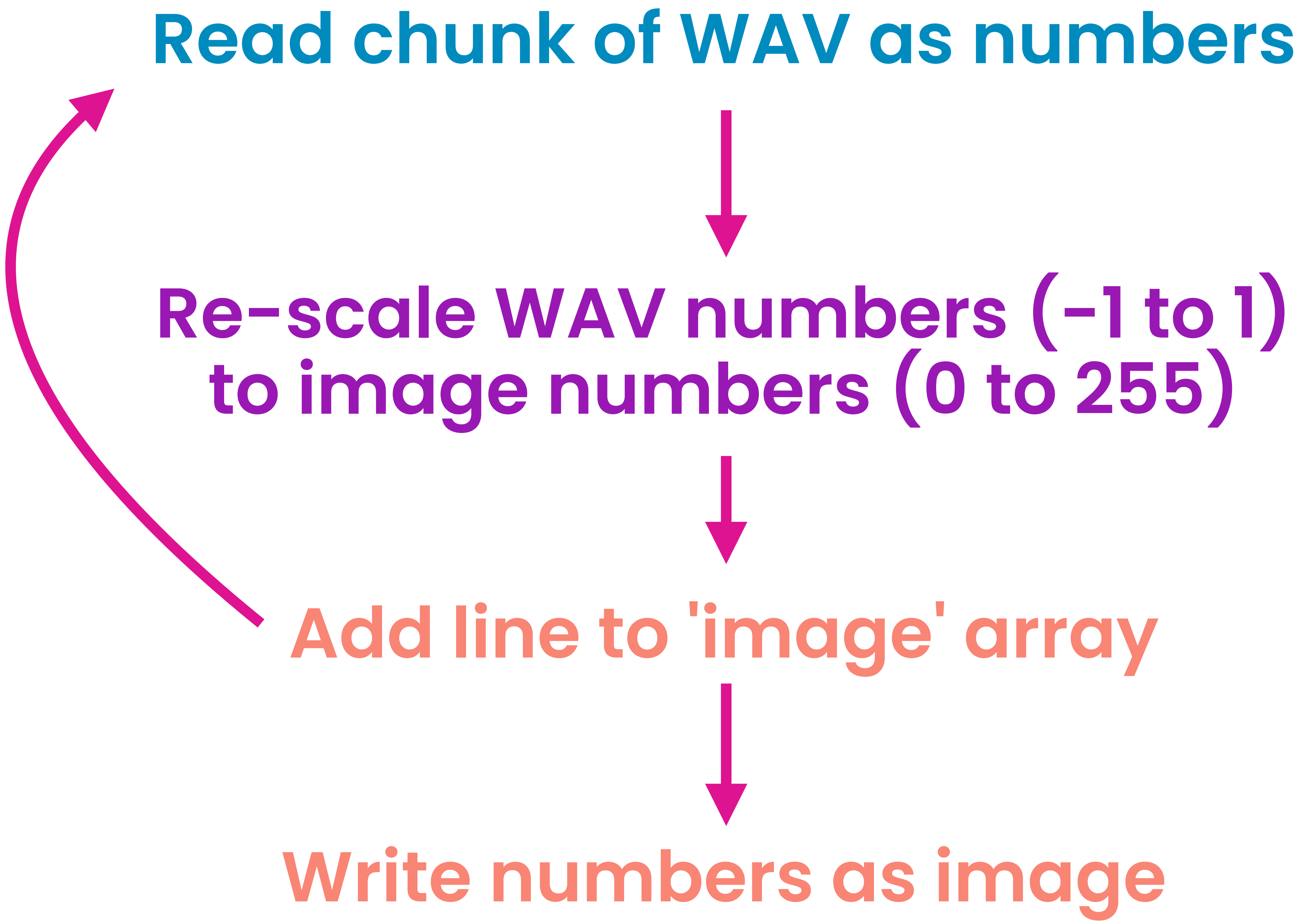
Read chunk of WAV as numbers

Re-scale WAV numbers (-1 to 1) to image numbers (0 to 255)

Add line to 'image' array

Write numbers as image



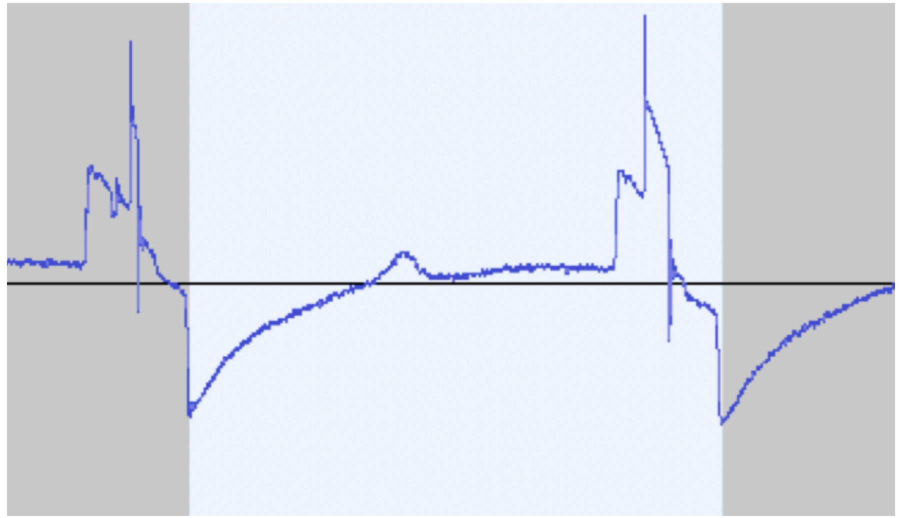


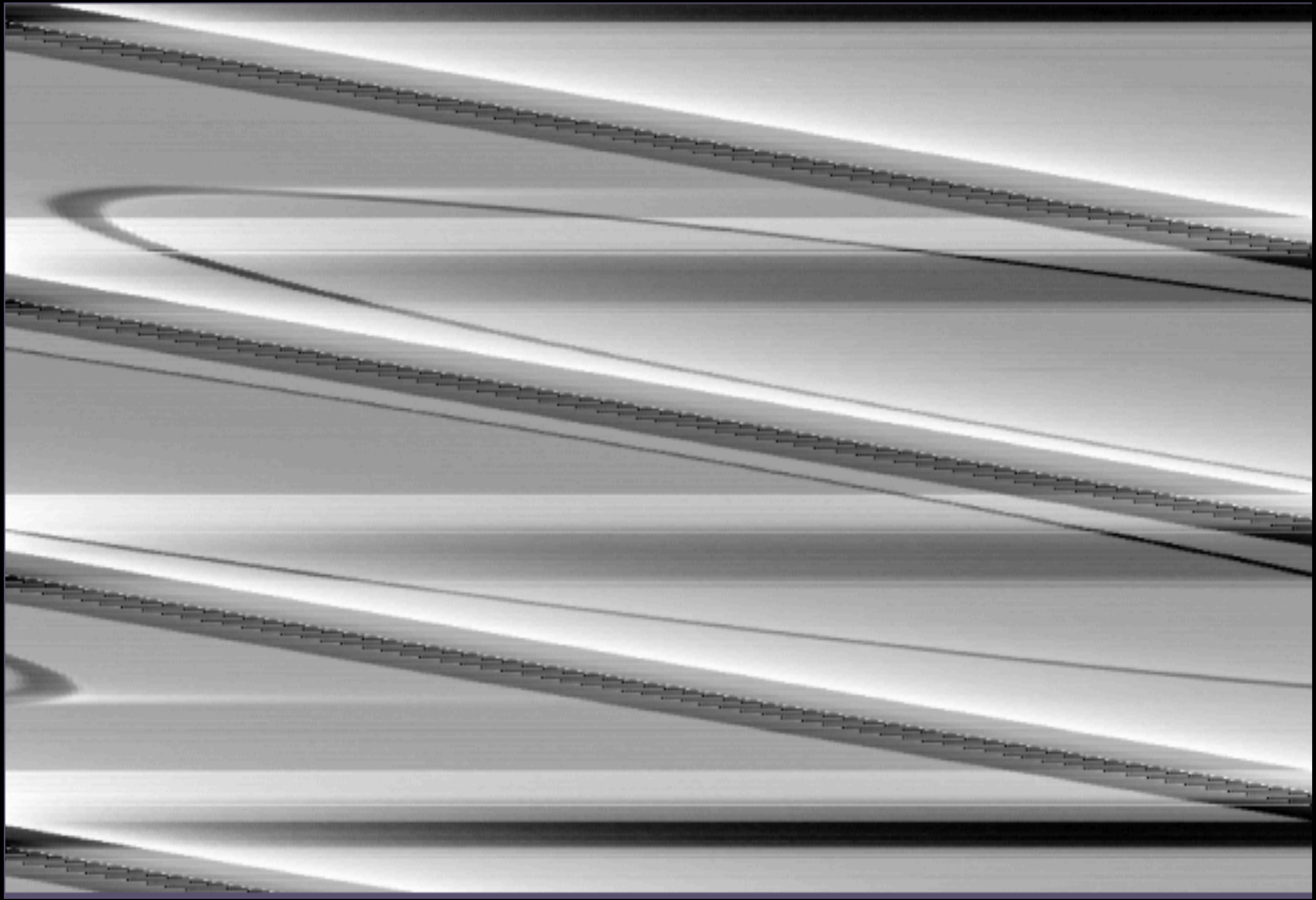
Read chunk of WAV as numbers

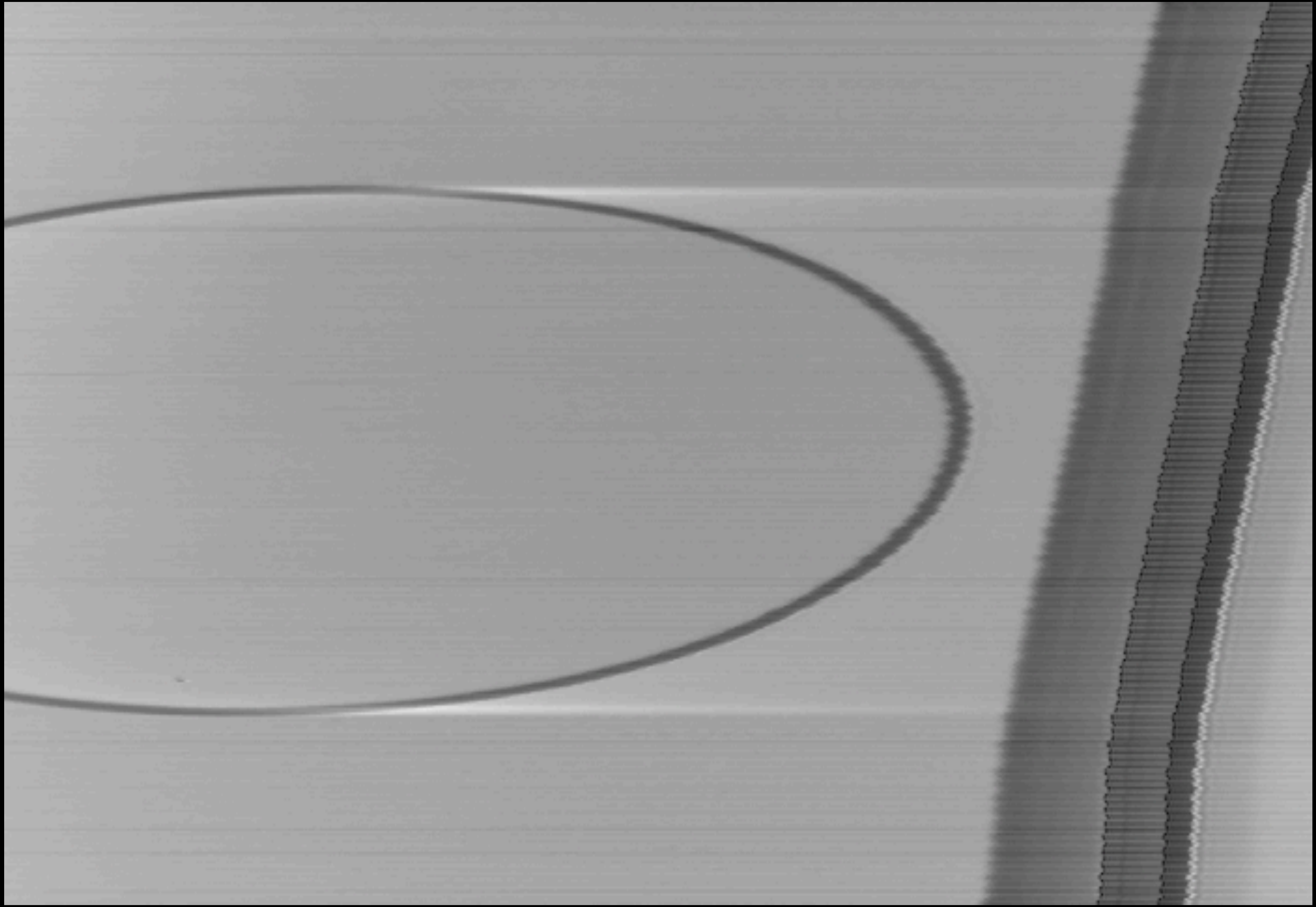
Re-scale WAV numbers (-1 to 1) to image numbers (0 to 255)

Add line to 'image' array

Write numbers as image



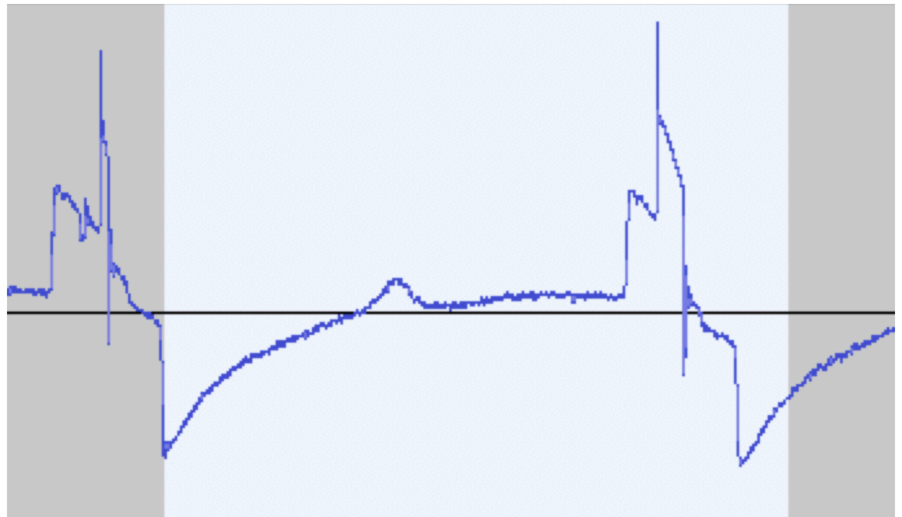




Read chunk of WAV as numbers

@damncabbage

Read chunk of WAV as numbers

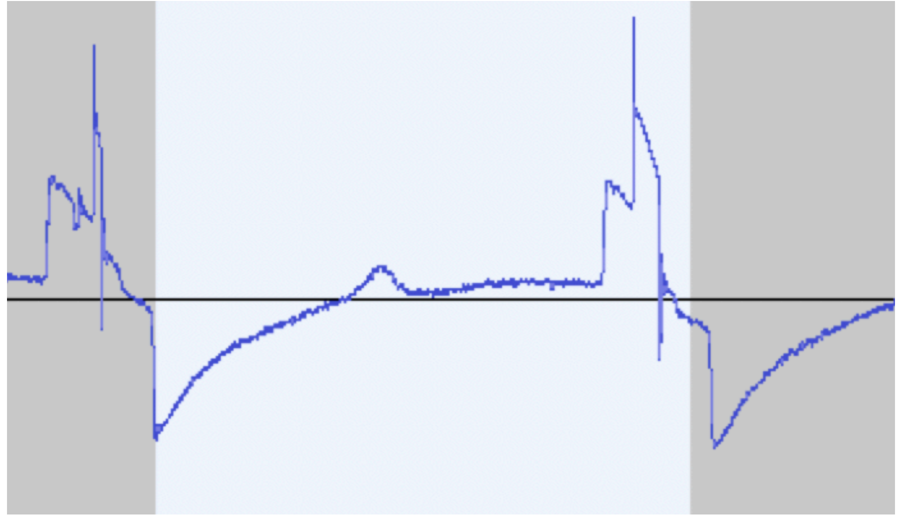


@damncabbage

Read chunk of WAV as numbers

↓

Assume that most of the chunk is fine.



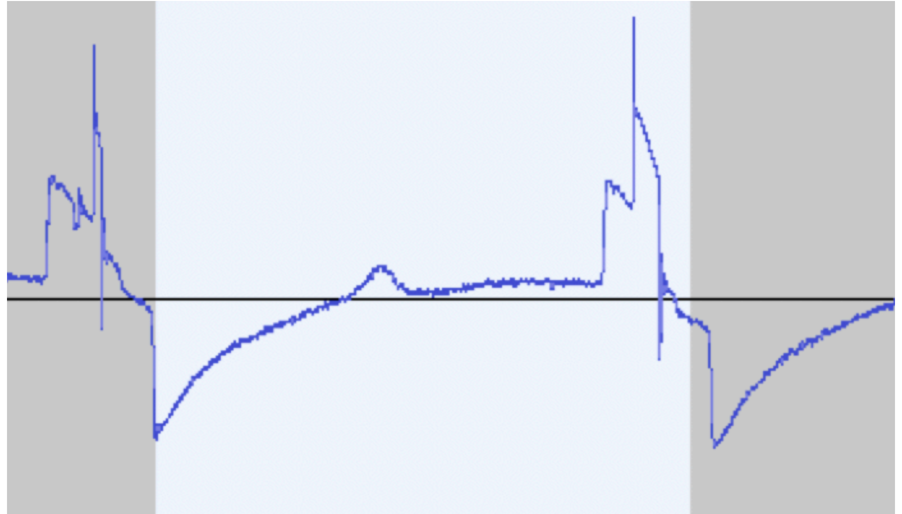
@damncabbage

Read chunk of WAV as numbers

↓

Assume that most of the chunk is fine.

↓



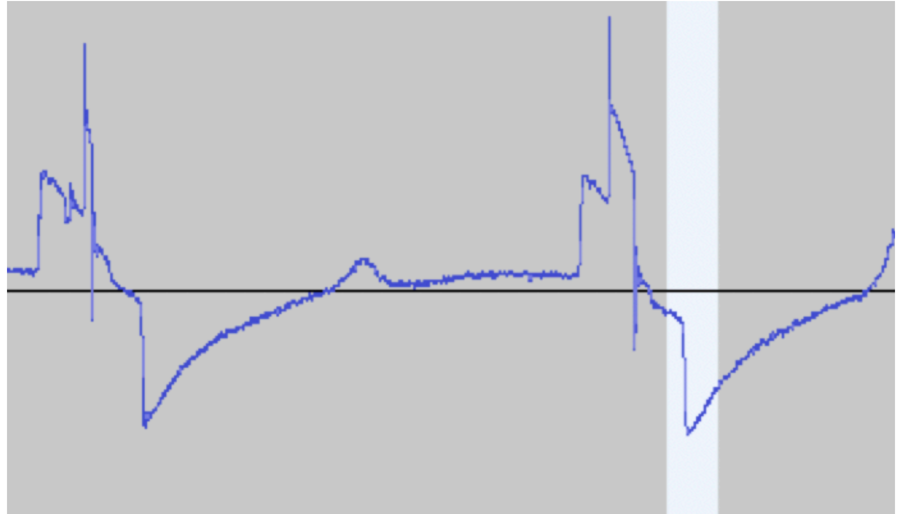
Read chunk of WAV as numbers

↓

Assume that most of the chunk is fine.

↓

Look for the lowest number in the rest; that's our 'boundary'.



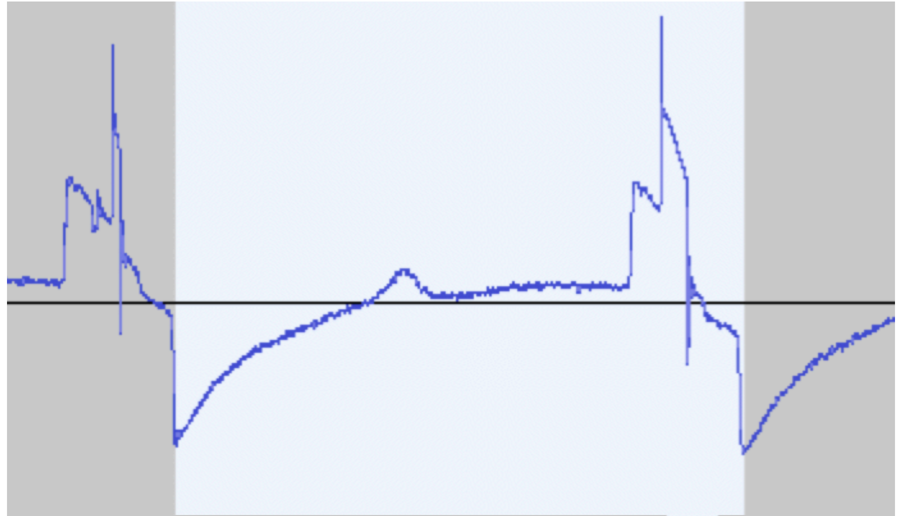
Read chunk of WAV as numbers

↓

Assume that most of the chunk is fine.

↓

Look for the lowest number in the rest;
that's our 'boundary'.



Read chunk of WAV as numbers

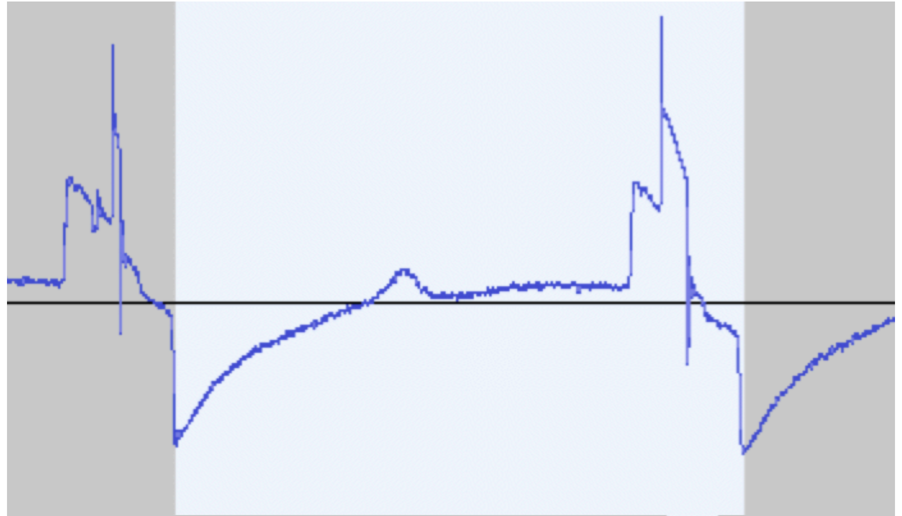
↓

Assume that most of the chunk is fine.

↓

Look for the lowest number in the rest;
that's our 'boundary'.

Keep everything left over for next frame.



@damncabbage

Read chunk of WAV as numbers

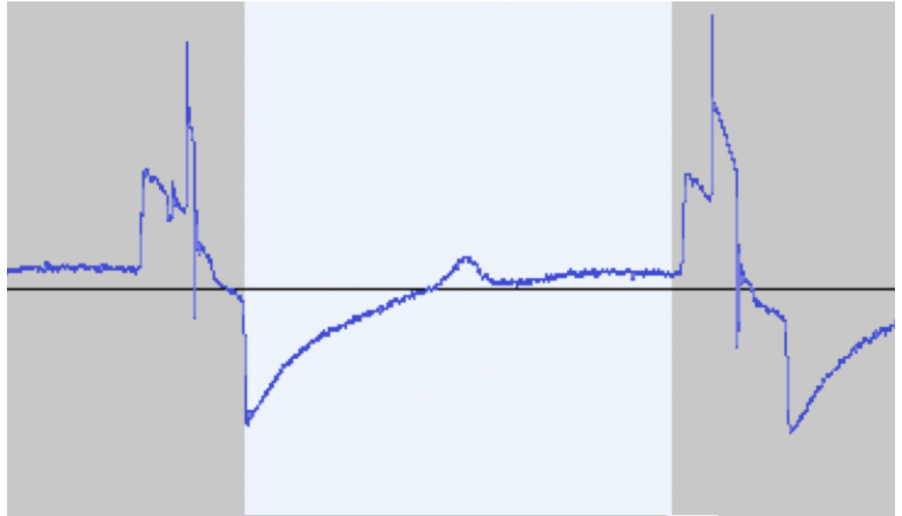
↓

Assume that most of the chunk is fine.

↓

Look for the lowest number in the rest; that's our 'boundary'.

Keep everything left over for next frame.
Chop the garbage off.



Read chunk of WAV as numbers

↓

Assume that most of the chunk is fine.

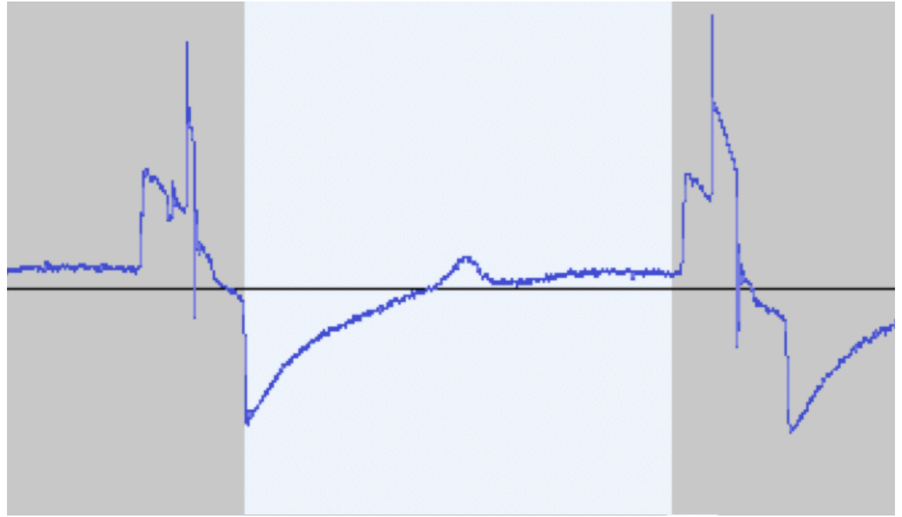
↓

Look for the lowest number in the rest; that's our 'boundary'.

Keep everything left over for next frame. Chop the garbage off.

↓

Re-scale WAV numbers (-1 to 1) to image numbers (0 to 255)



@damncabbage

Read chunk of WAV as numbers

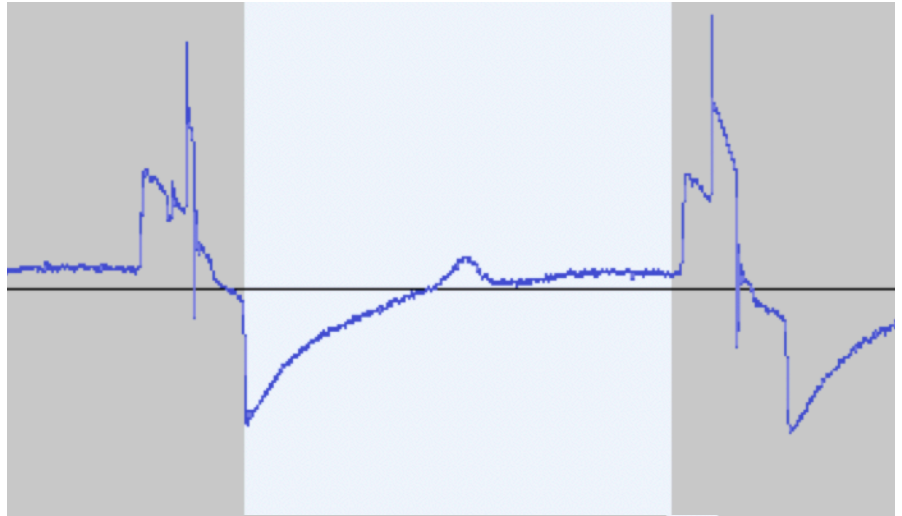
↓

Assume that most of the chunk is fine.

↓

Look for the lowest number in the rest; that's our 'boundary'.

Keep everything left over for next frame. Chop the garbage off.



↓

Re-scale WAV numbers (-1 to 1) to image numbers (0 to 255)

↓

Add line to 'image' array

Read chunk of WAV as numbers

↓

Assume that most of the chunk is fine.

↓

Look for the lowest number in the rest;
that's our 'boundary'.

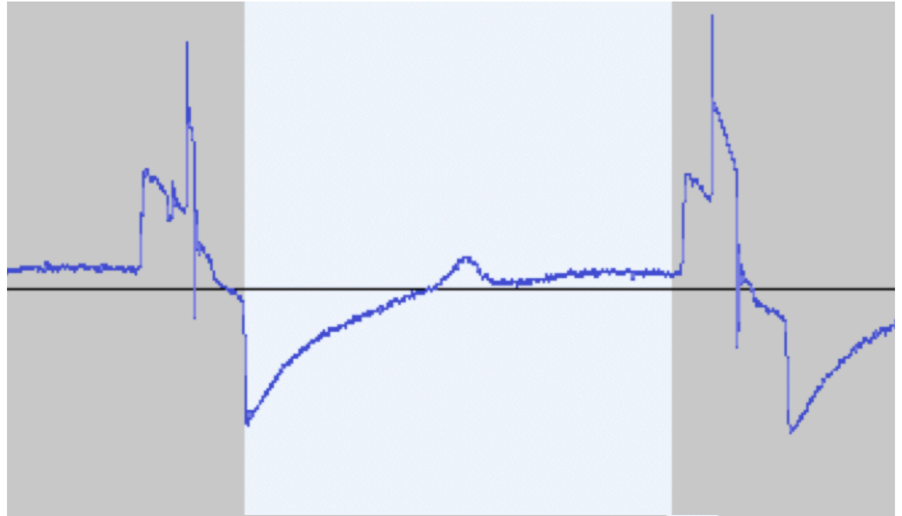
Keep everything left over for next frame.
Chop the garbage off.

↓

Re-scale WAV numbers (-1 to 1)
to image numbers (0 to 255)

↓

Add line to 'image' array

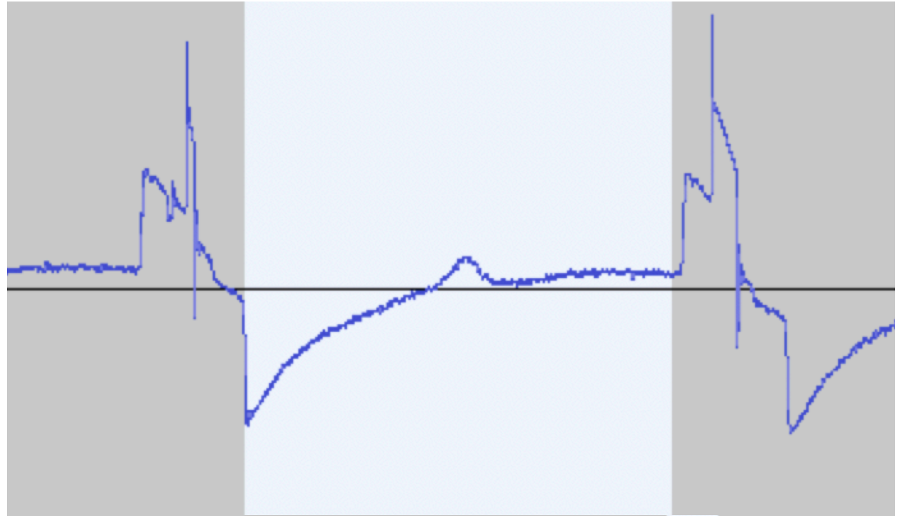


Read chunk of WAV as numbers

Assume that most of the chunk is fine.

Look for the lowest number in the rest;
that's our 'boundary'.

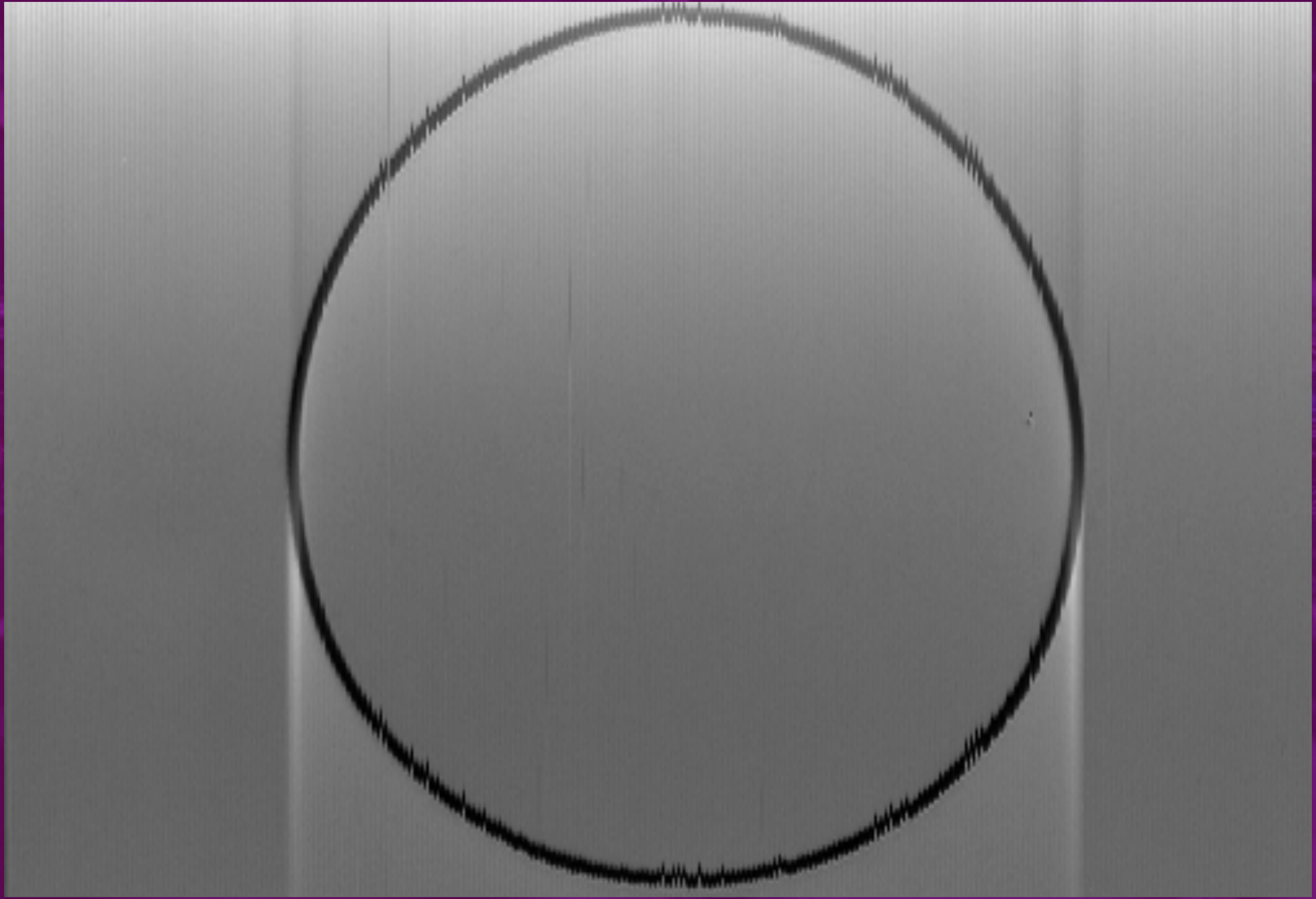
Keep everything left over for next frame.
Chop the garbage off.

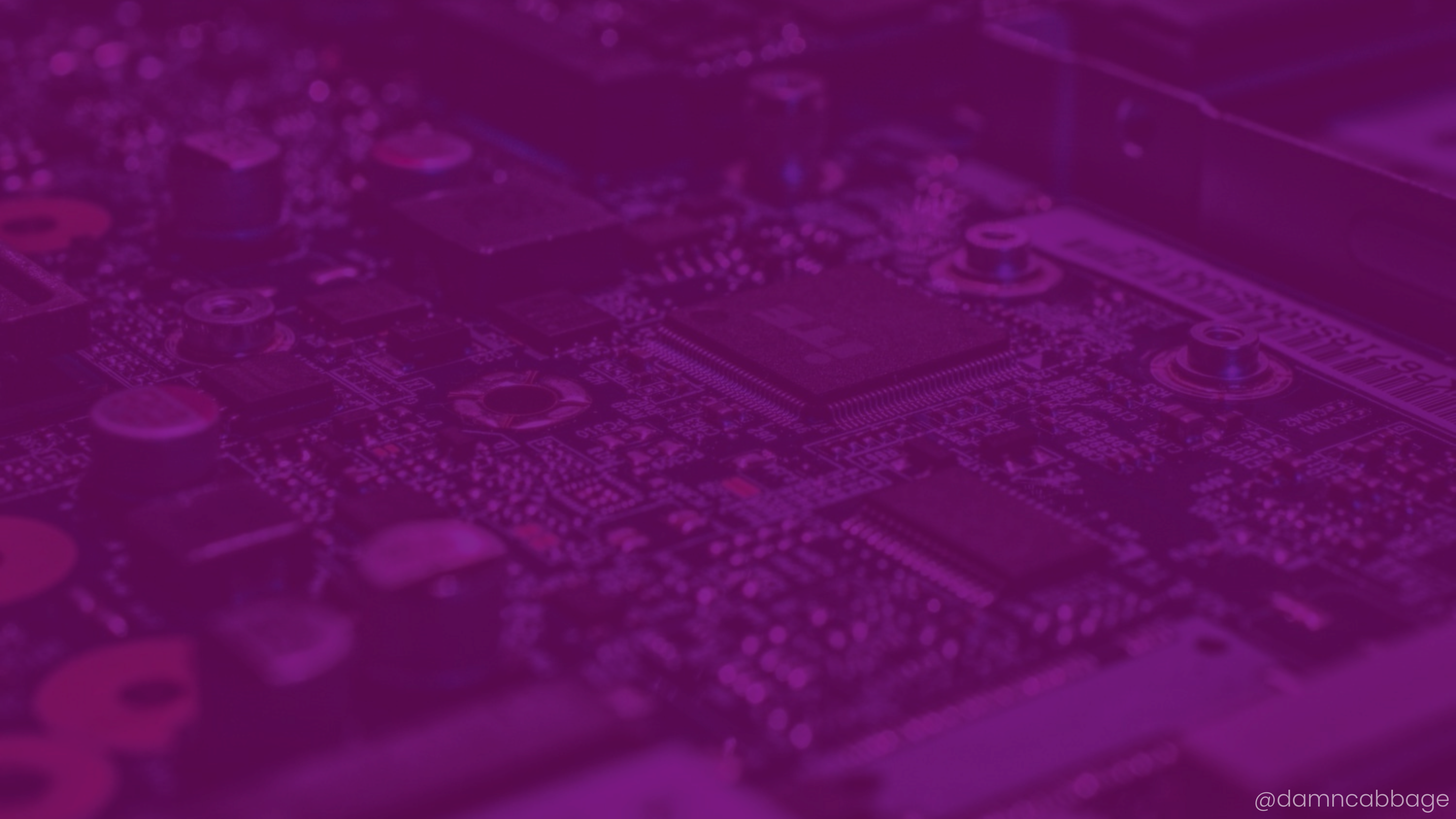


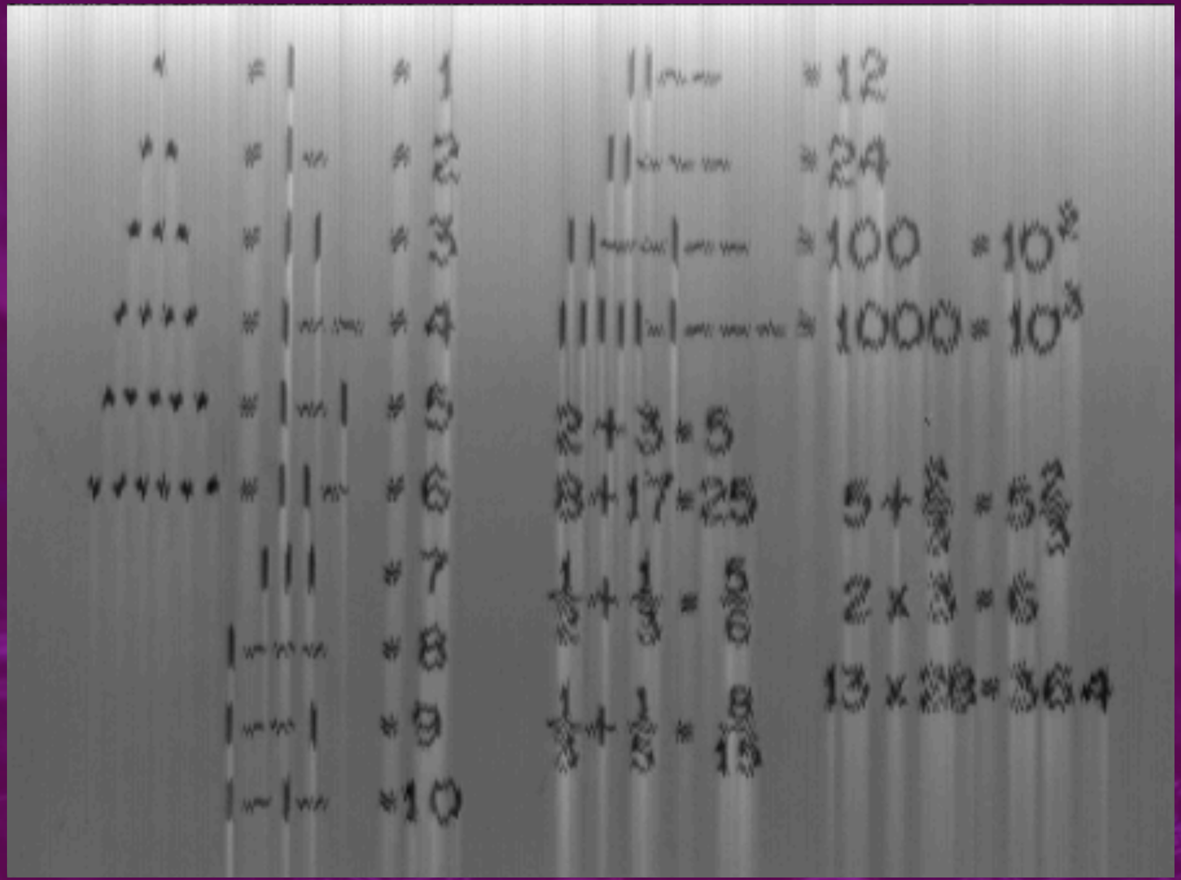
Re-scale WAV numbers (-1 to 1)
to image numbers (0 to 255)

Add line to 'image' array

Write numbers as image







• = | = 1

•• = |~ = 2

••• = || = 3

•••• = |~~ = 4

••••• = |~| = 5

•••••• = ||~ = 6

||| = 7

|~~~ = 8

|~~| = 9

|~|~ = 10

||~~ = 12

||~~~ = 24

||~~|~~ = 100 = 10^2

||||~|~~~~ = 1000 = 10^3

2 + 3 = 5

8 + 17 = 25

$\frac{1}{2}+\frac{1}{3}=\frac{5}{6}$

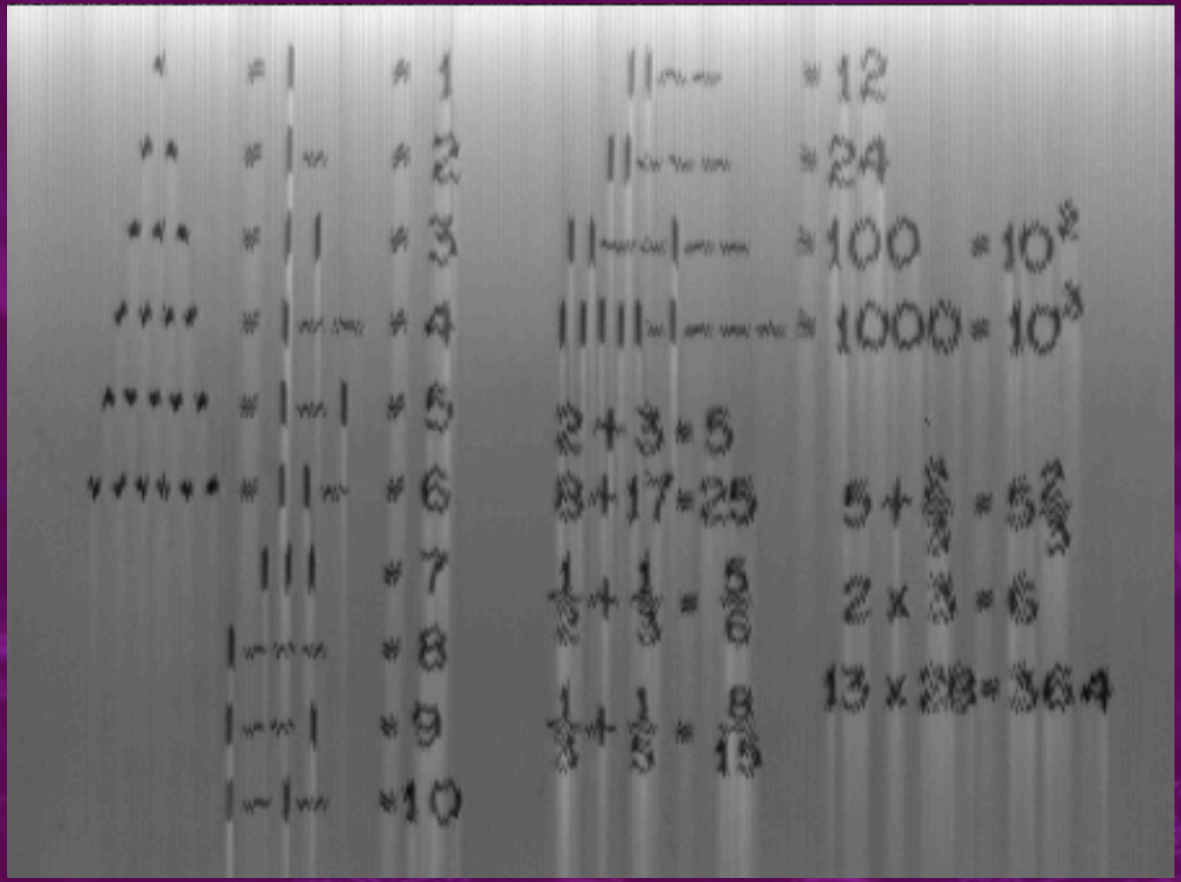
$\frac{1}{3}+\frac{1}{5}=\frac{8}{15}$

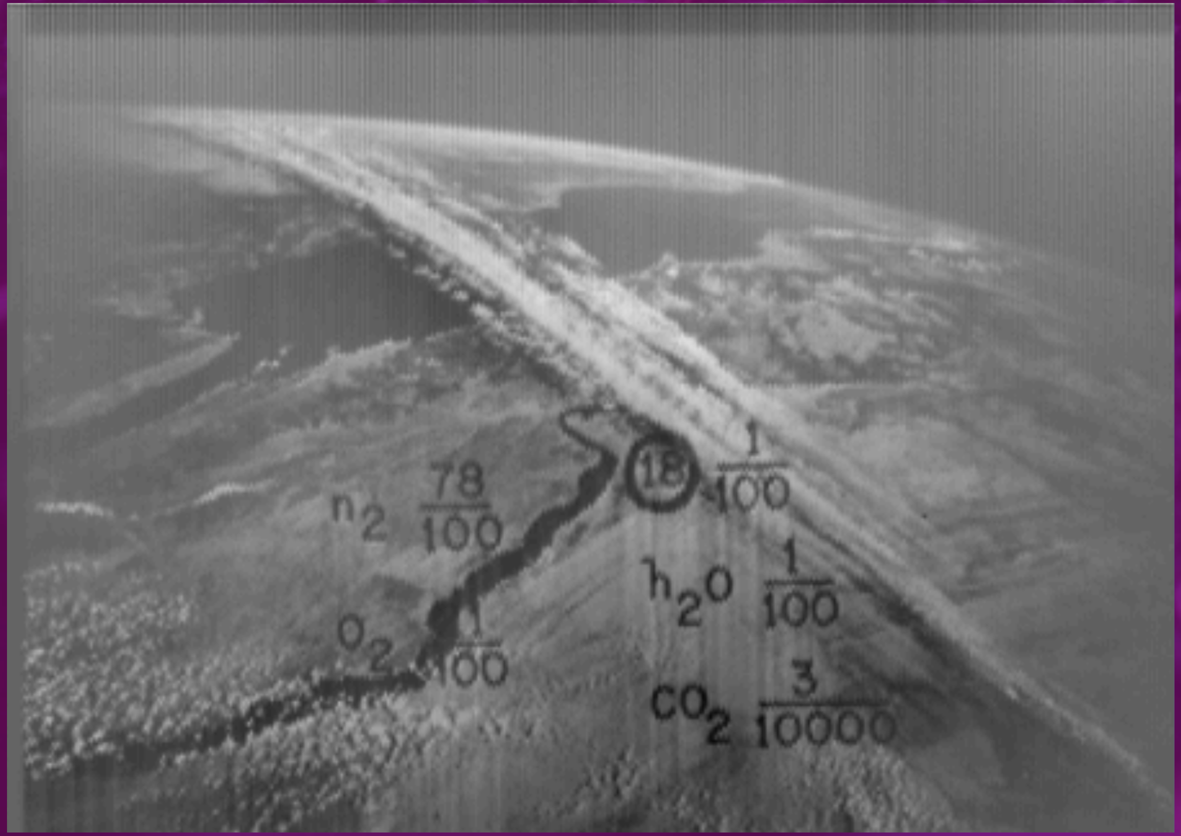
$5+\frac{2}{3}=5\frac{2}{3}$

2 × 3 = 6

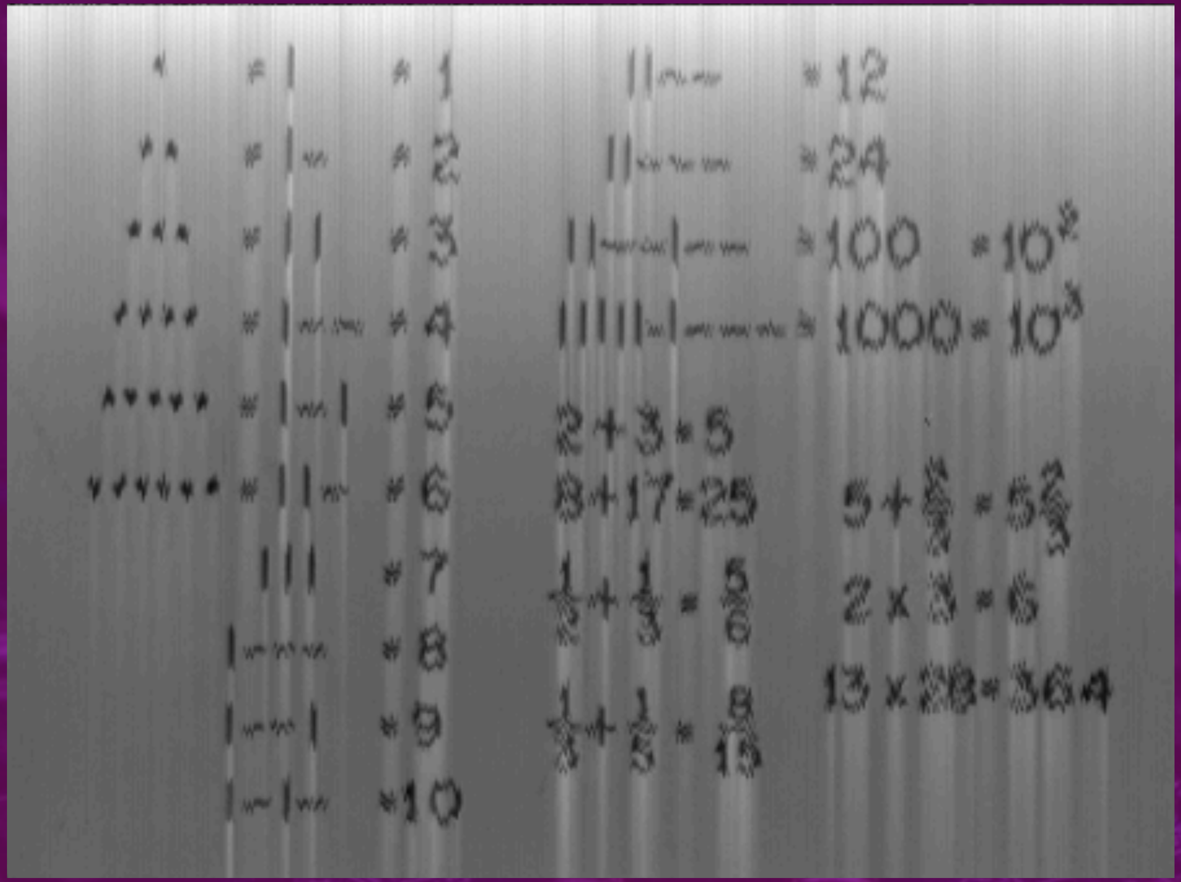
13 × 28 = 364

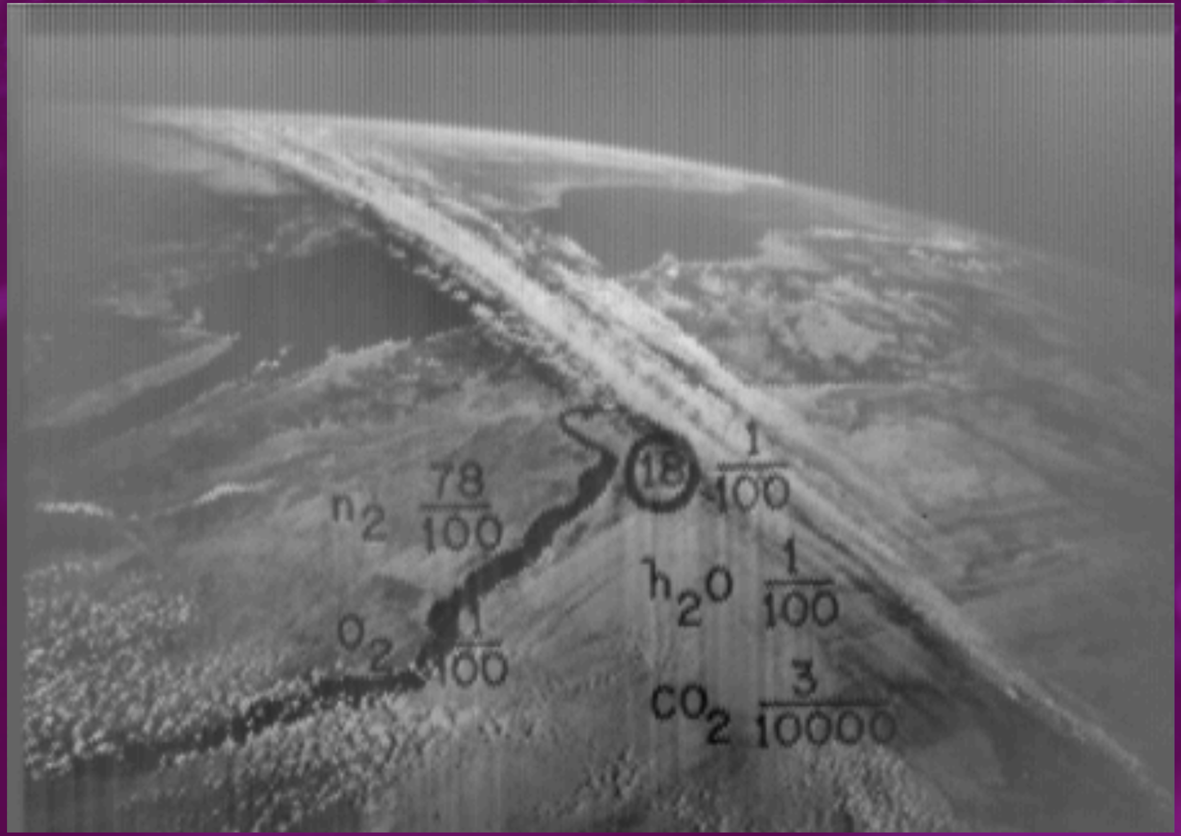
@damncabbage



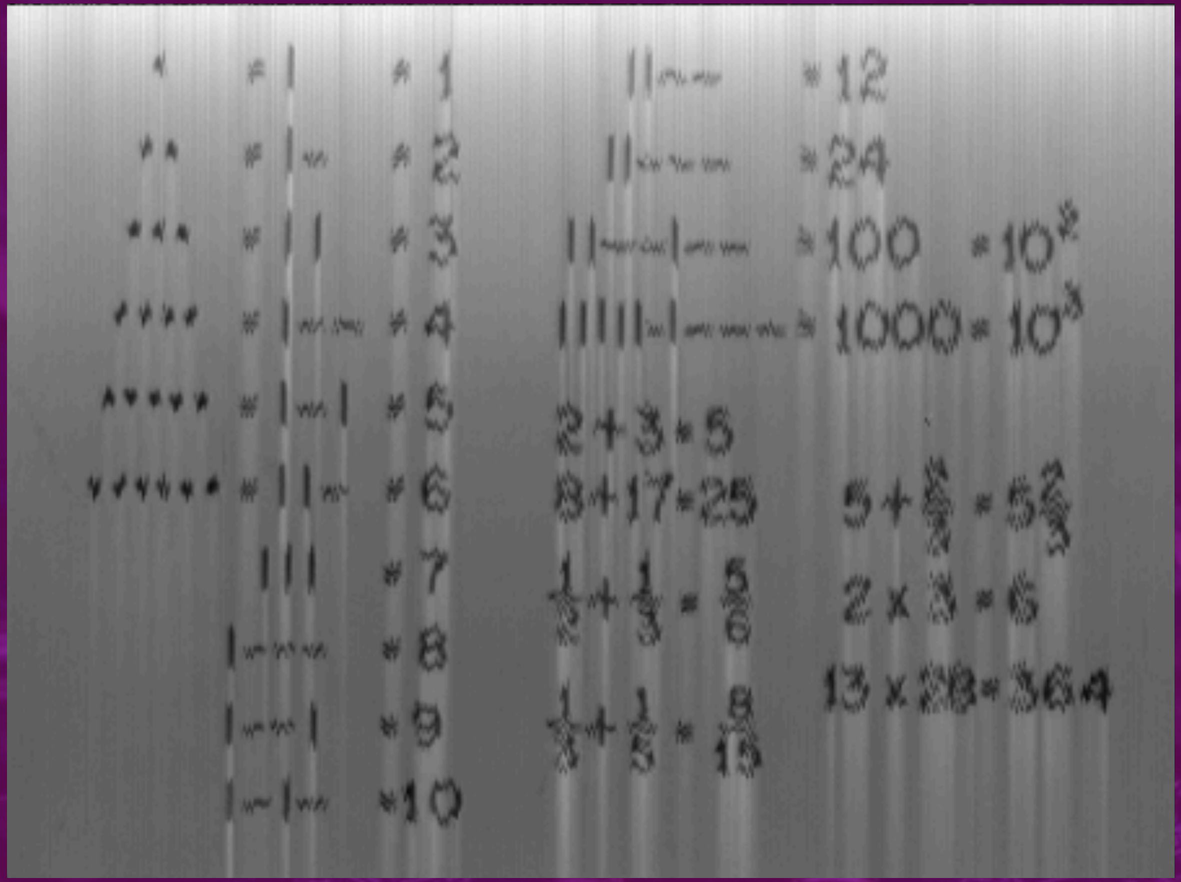


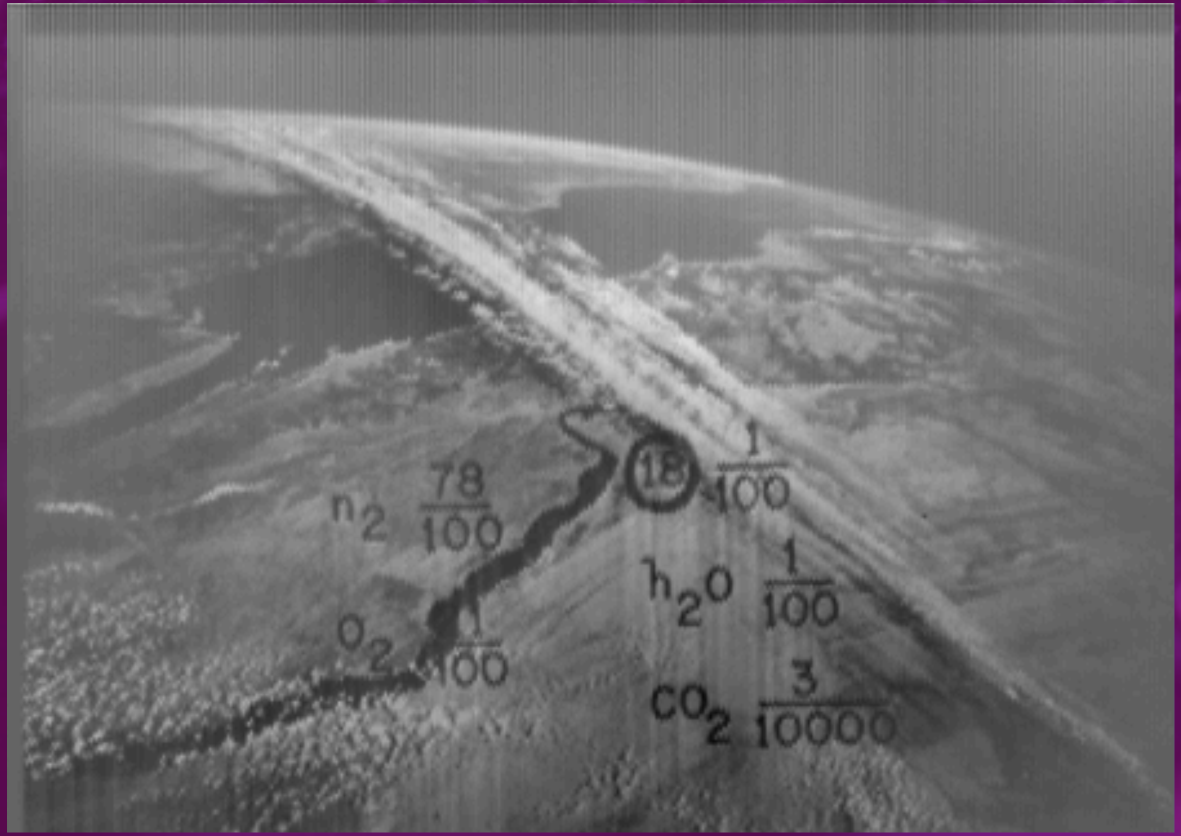
@damncabbage

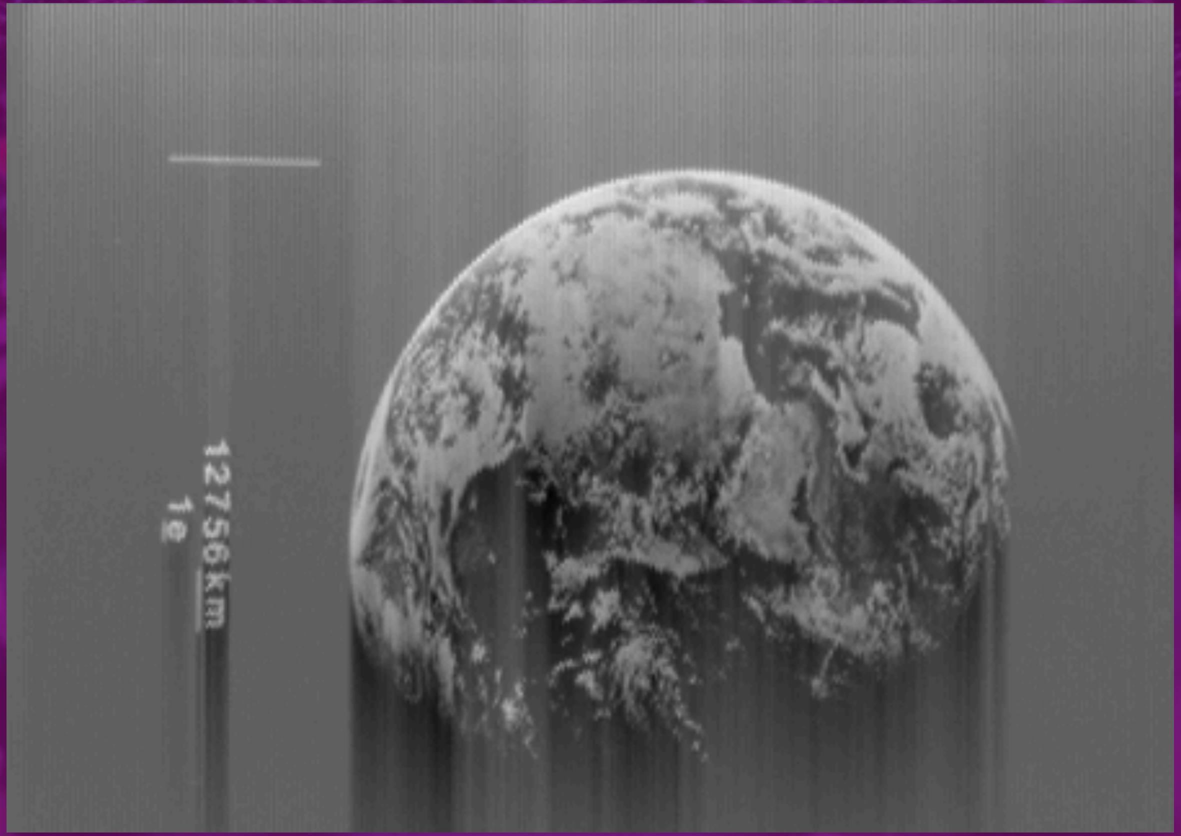


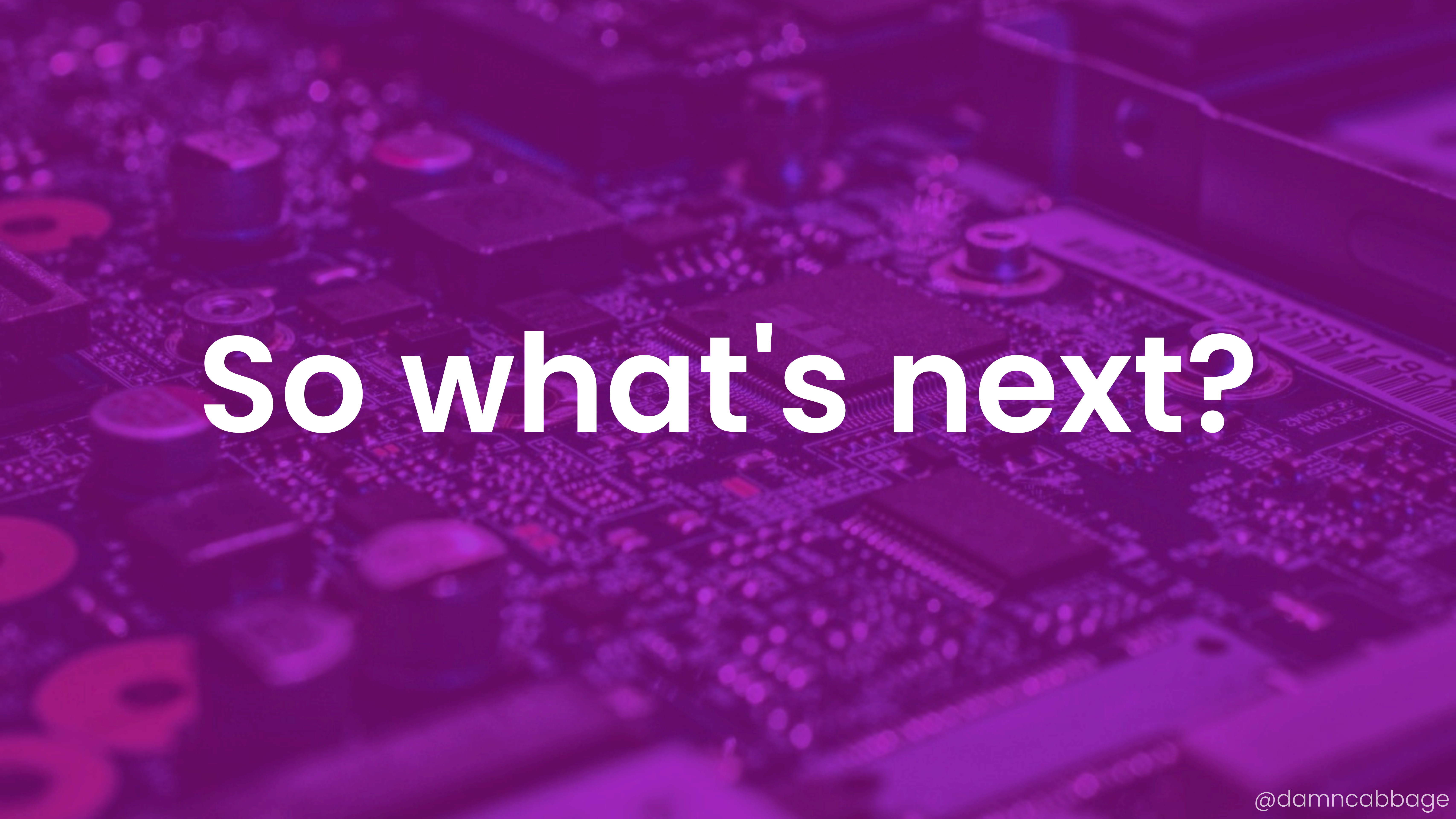


@damncabbage

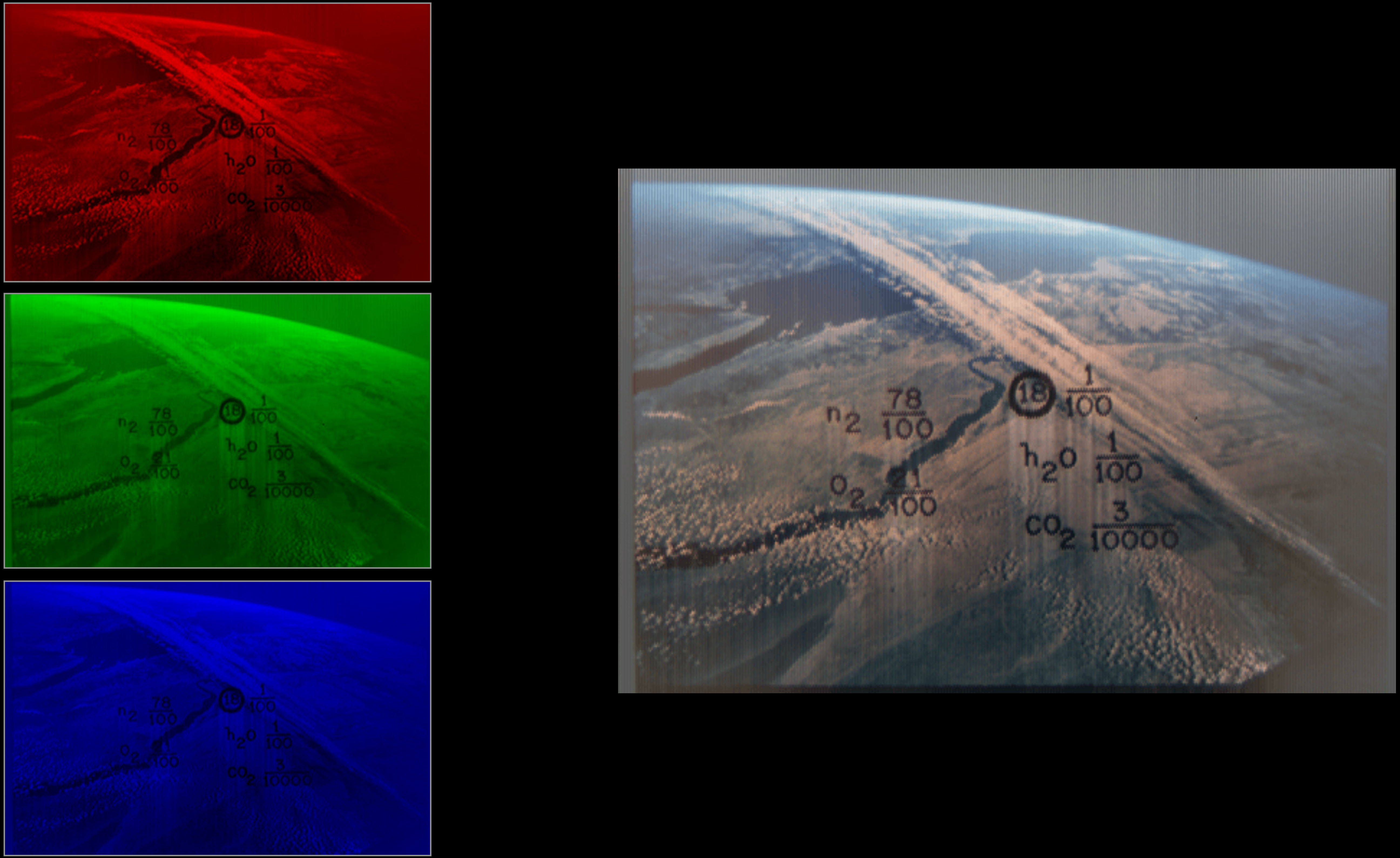

=1
=2
=3
=4
=5
=6
=7
=8
=9
=10
=12
=24
=100 =10²
=1000 =10³
2+3=5
8+17=25
1/2+1/3=5/6
1/3+1/5=8/15
2×3=6
13×28=364


n2 78/100
18 1/100
o2 1/100
h2o 1/100
co2 3/10000


12756km
1e

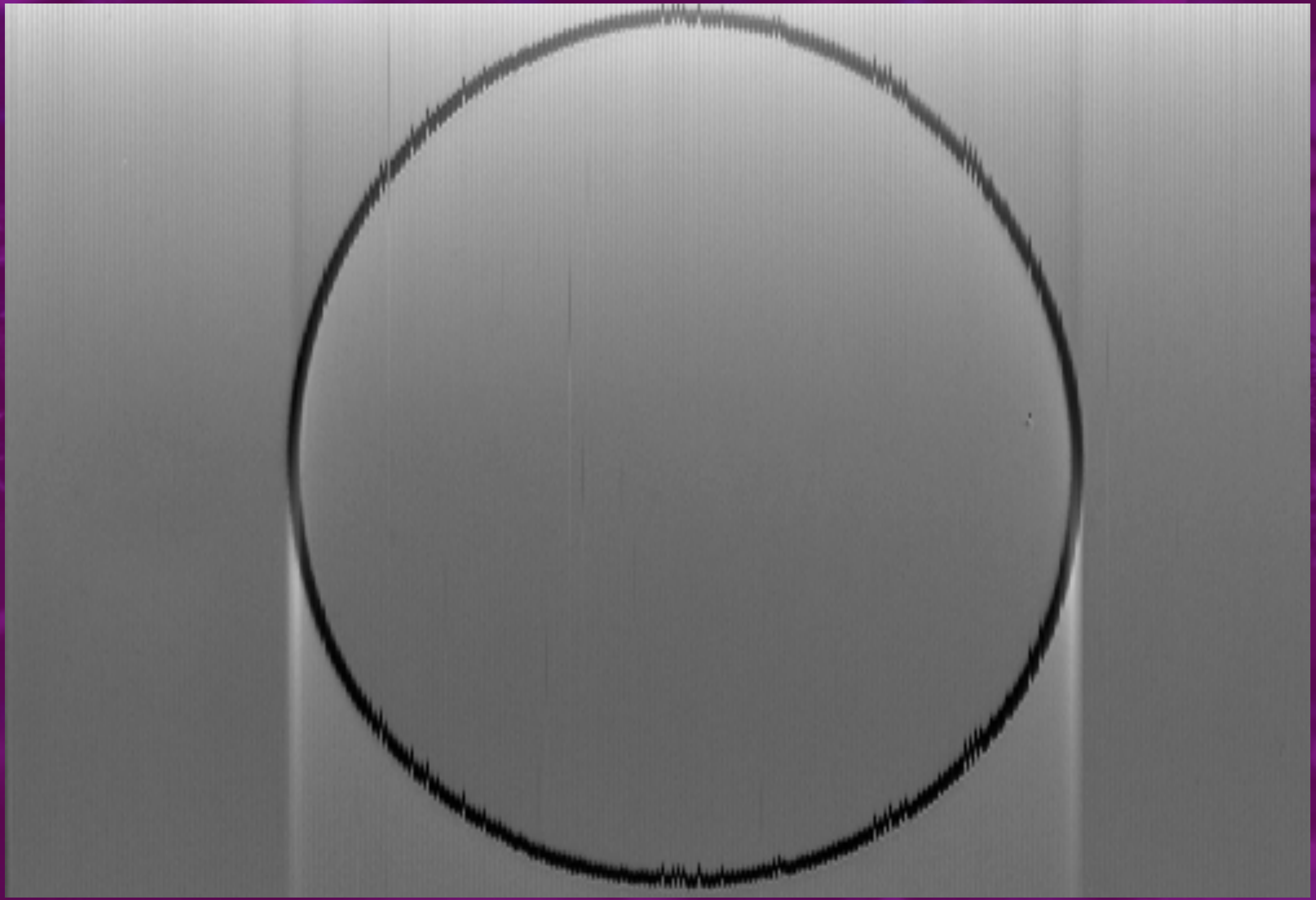


So what's next?



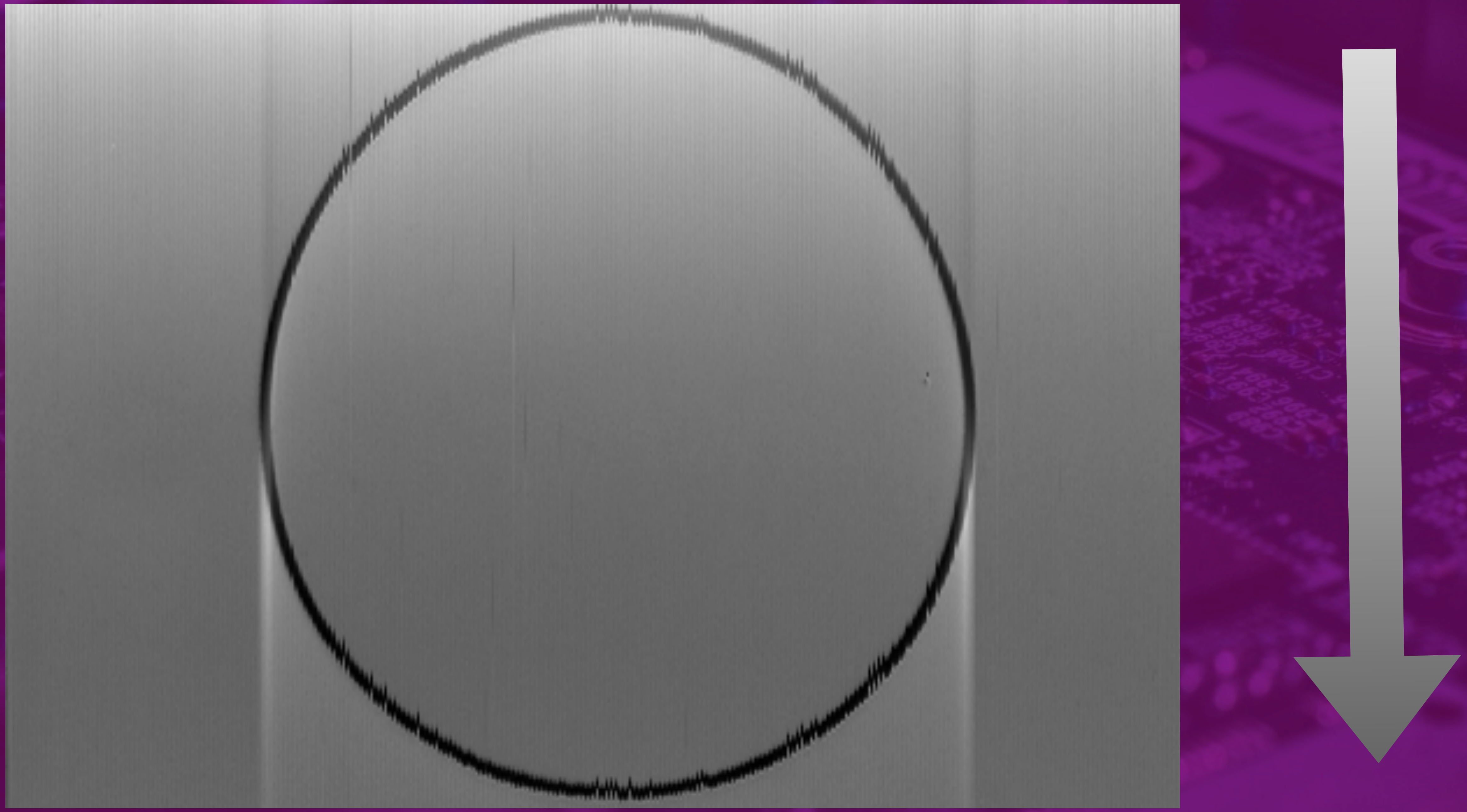
https://boingboing.net/2017/09/05/how-to-decode-the-images-on-th.html

Fix the Weird Gradient

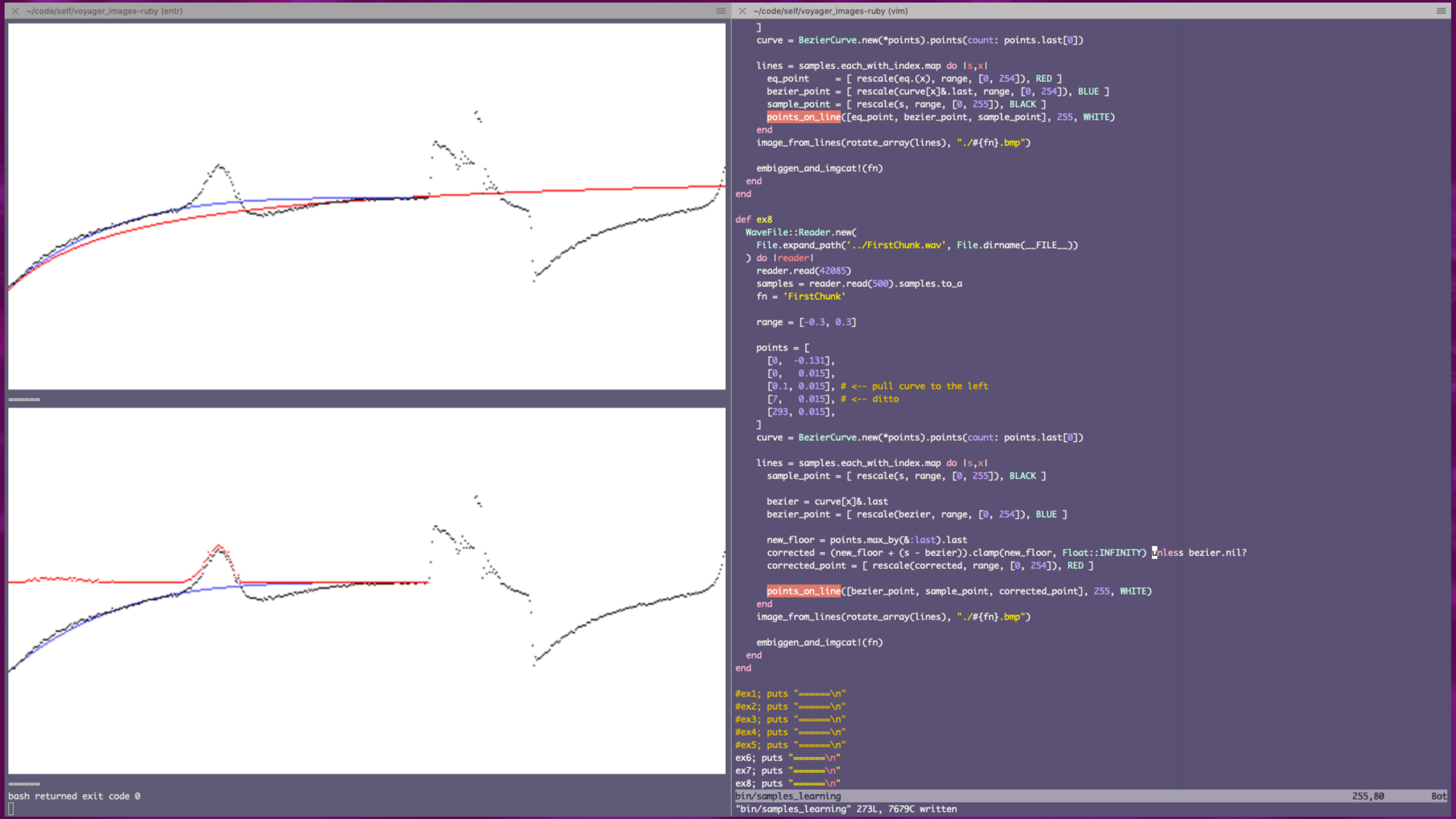


@damncabbage

Fix the Weird Gradient



Fix the Weird Gradient



~/code/self/voyager_images-ruby (entr)

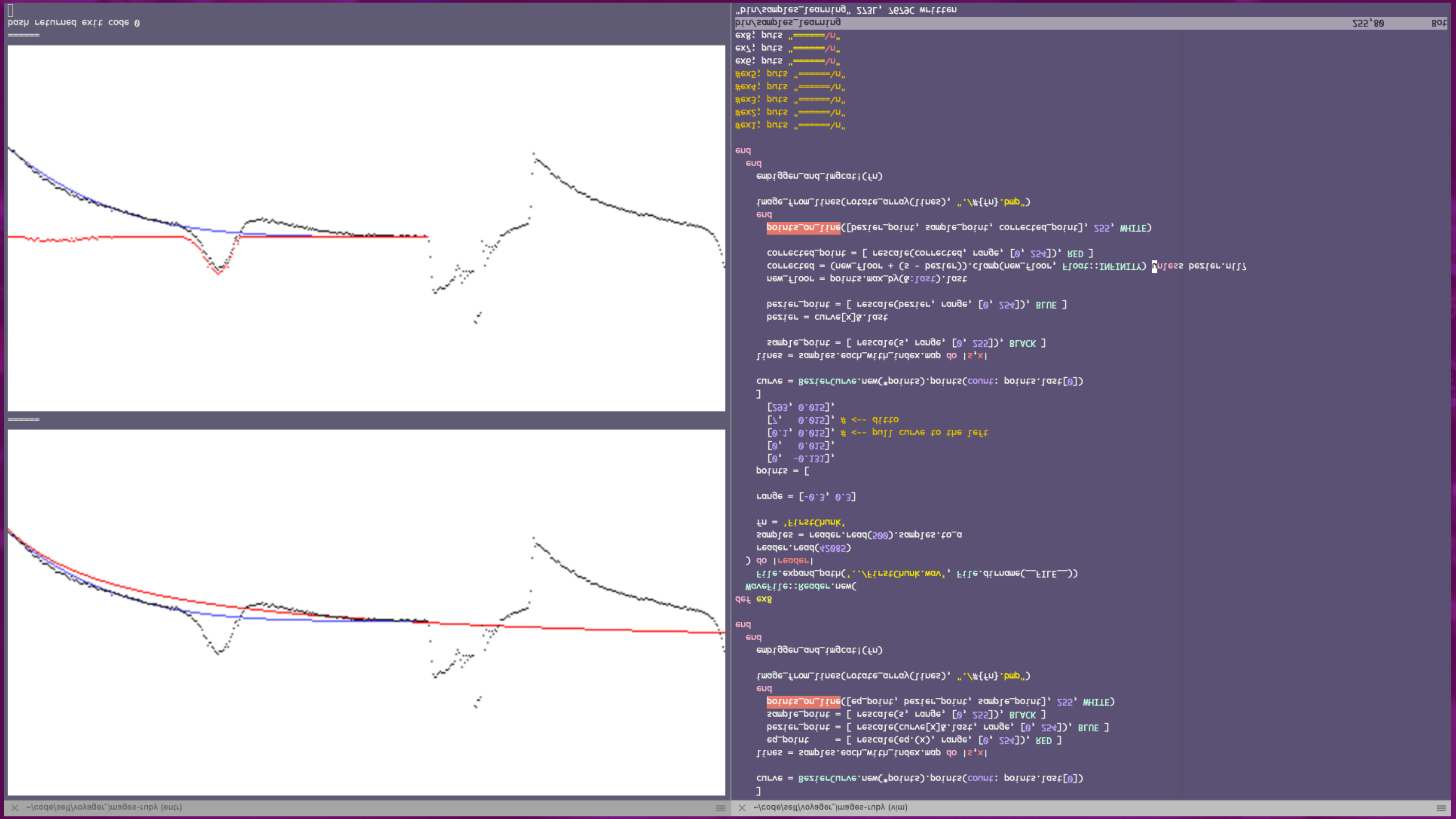
```
======
======
bash returned exit code 0
```

~/code/self/voyager_images-ruby (vim)

```ruby
    ]
    curve = BezierCurve.new(*points).points(count: points.last[0])

    lines = samples.each_with_index.map do |s,x|
      eq_point     = [ rescale(eq.(x), range, [0, 254]), RED ]
      bezier_point = [ rescale(curve[x]&.last, range, [0, 254]), BLUE ]
      sample_point = [ rescale(s, range, [0, 255]), BLACK ]
      points_on_line([eq_point, bezier_point, sample_point], 255, WHITE)
    end
    image_from_lines(rotate_array(lines), "./#{fn}.bmp")

    embiggen_and_imgcat!(fn)
  end
end

def ex8
  WaveFile::Reader.new(
    File.expand_path('../FirstChunk.wav', File.dirname(__FILE__))
  ) do |reader|
    reader.read(42085)
    samples = reader.read(500).samples.to_a
    fn = 'FirstChunk'

    range = [-0.3, 0.3]

    points = [
      [0,  -0.131],
      [0,   0.015],
      [0.1, 0.015], # <-- pull curve to the left
      [7,   0.015], # <-- ditto
      [293, 0.015],
    ]
    curve = BezierCurve.new(*points).points(count: points.last[0])

    lines = samples.each_with_index.map do |s,x|
      sample_point = [ rescale(s, range, [0, 255]), BLACK ]

      bezier = curve[x]&.last
      bezier_point = [ rescale(bezier, range, [0, 254]), BLUE ]

      new_floor = points.max_by(&:last).last
      corrected = (new_floor + (s - bezier)).clamp(new_floor, Float::INFINITY) unless bezier.nil?
      corrected_point = [ rescale(corrected, range, [0, 254]), RED ]

      points_on_line([bezier_point, sample_point, corrected_point], 255, WHITE)
    end
    image_from_lines(rotate_array(lines), "./#{fn}.bmp")

    embiggen_and_imgcat!(fn)
  end
end

#ex1; puts "======\n"
#ex2; puts "======\n"
#ex3; puts "======\n"
#ex4; puts "======\n"
#ex5; puts "======\n"
ex6; puts "======\n"
ex7; puts "======\n"
ex8; puts "======\n"
bin/samples_learning                                                    255,80     Bot
"bin/samples_learning" 273L, 7679C written
```

@damncabbage

Fix the weird gradient



```
======
======
bash returned exit code 0
```

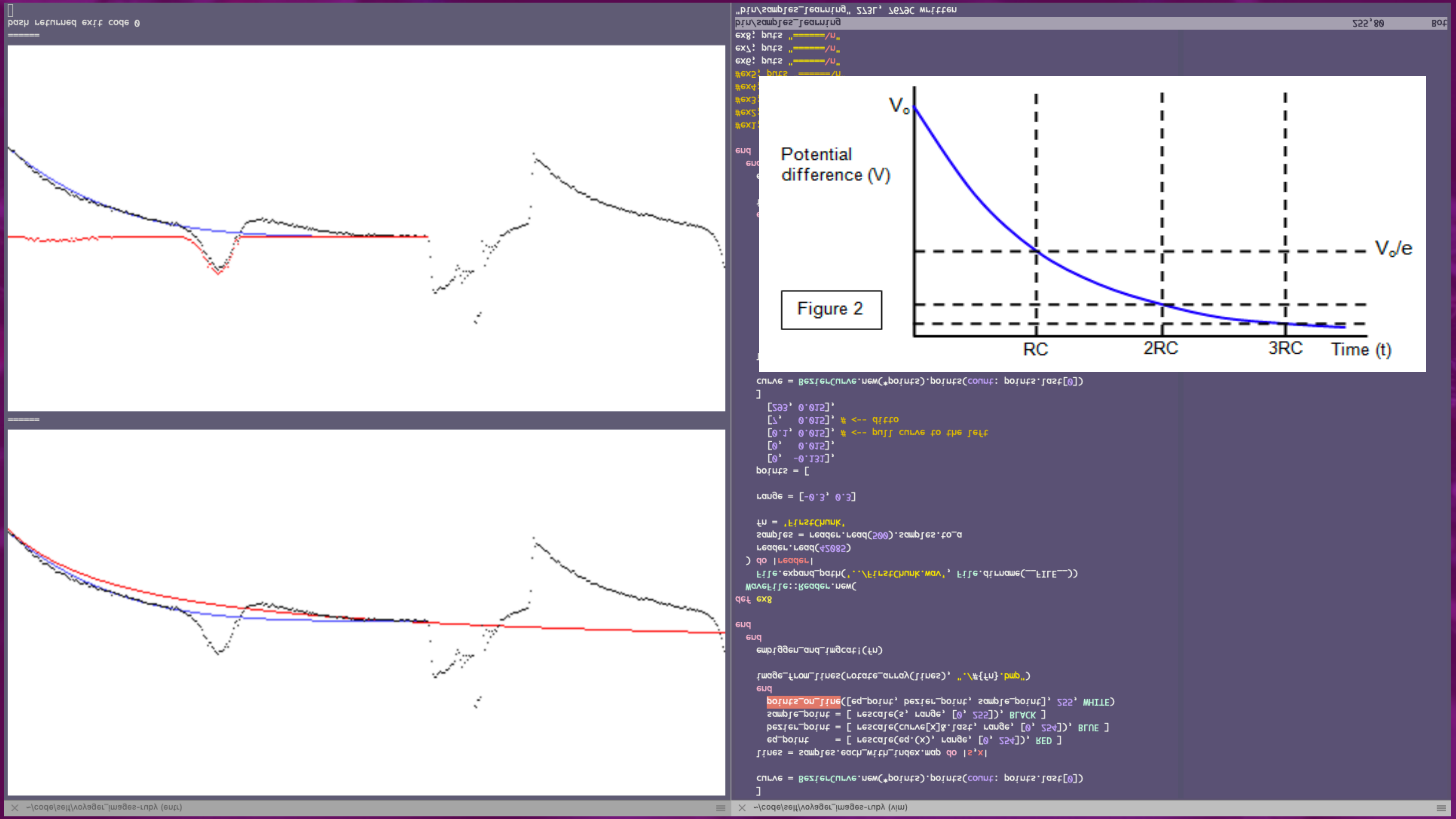
~\code\self\voyager_images-ruby (entr)

```ruby
  ]
  curve = BezierCurve.new(*points).points(count: points.last[0])

  lines = samples.each_with_index.map do |s,x|
    ed_point     = [ rescale(ed.(x), range, [0, 254]), RED ]
    bezier_point = [ rescale(curve[x]&.last, range, [0, 254]), BLUE ]
    sample_point = [ rescale(s, range, [0, 255]), BLACK ]
    points_on_line([ed_point, bezier_point, sample_point], 255, WHITE)
  end
  image_from_lines(rotate_array(lines), "./#{fn}.bmp")

  embiggen_and_imgcat!(fn)
  end
end

def ex8
  WaveFile::Reader.new(
    File.expand_path('../FirstChunk.wav', File.dirname(__FILE__))
  ) do |reader|
    reader.read(42082)
    samples = reader.read(200).samples.to_a
    fn = 'FirstChunk'

    range = [-0.3, 0.3]

    points = [
      [0,  -0.131],
      [0,   0.015],
      [0.1, 0.015], # <-- pull curve to the left
      [7,   0.015], # <-- ditto
      [263, 0.015],
    ]
    curve = BezierCurve.new(*points).points(count: points.last[0])

    lines = samples.each_with_index.map do |s,x|
      sample_point = [ rescale(s, range, [0, 255]), BLACK ]

      bezier = curve[x]&.last
      bezier_point = [ rescale(bezier, range, [0, 254]), BLUE ]

      new_floor = points.max_by(&:last).last
      corrected = (new_floor + (s - bezier)).clamp(new_floor, Float::INFINITY) unless bezier.nil?
      corrected_point = [ rescale(corrected, range, [0, 254]), RED ]

      points_on_line([bezier_point, sample_point, corrected_point], 255, WHITE)
    end
    image_from_lines(rotate_array(lines), "./#{fn}.bmp")

    embiggen_and_imgcat!(fn)
  end
end

#ex1; puts "======\n"
#ex2; puts "======\n"
#ex3; puts "======\n"
#ex4; puts "======\n"
#ex5; puts "======\n"
ex6; puts "======\n"
ex7; puts "======\n"
ex8; puts "======\n"
bin/samples_learning                                                255,80         Bot
"bin/samples_learning" 273L, 7676C written
```

~\code\self\voyager_images-ruby (vim)

@damncabbage

Fix the weird gradient

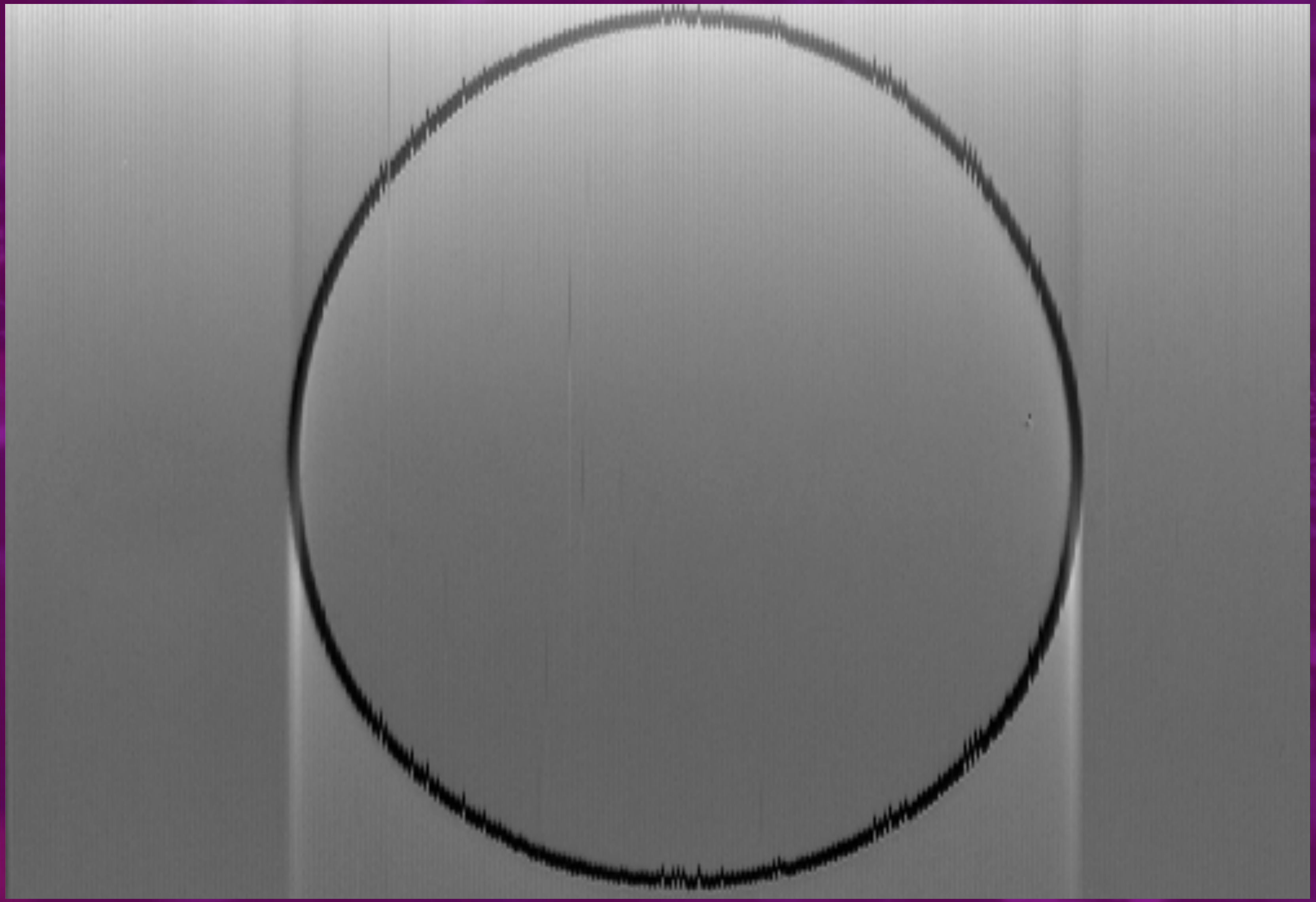


```
~/code/self/voyager_images-ruby (entr)
======
======
bash returned exit code 0
```

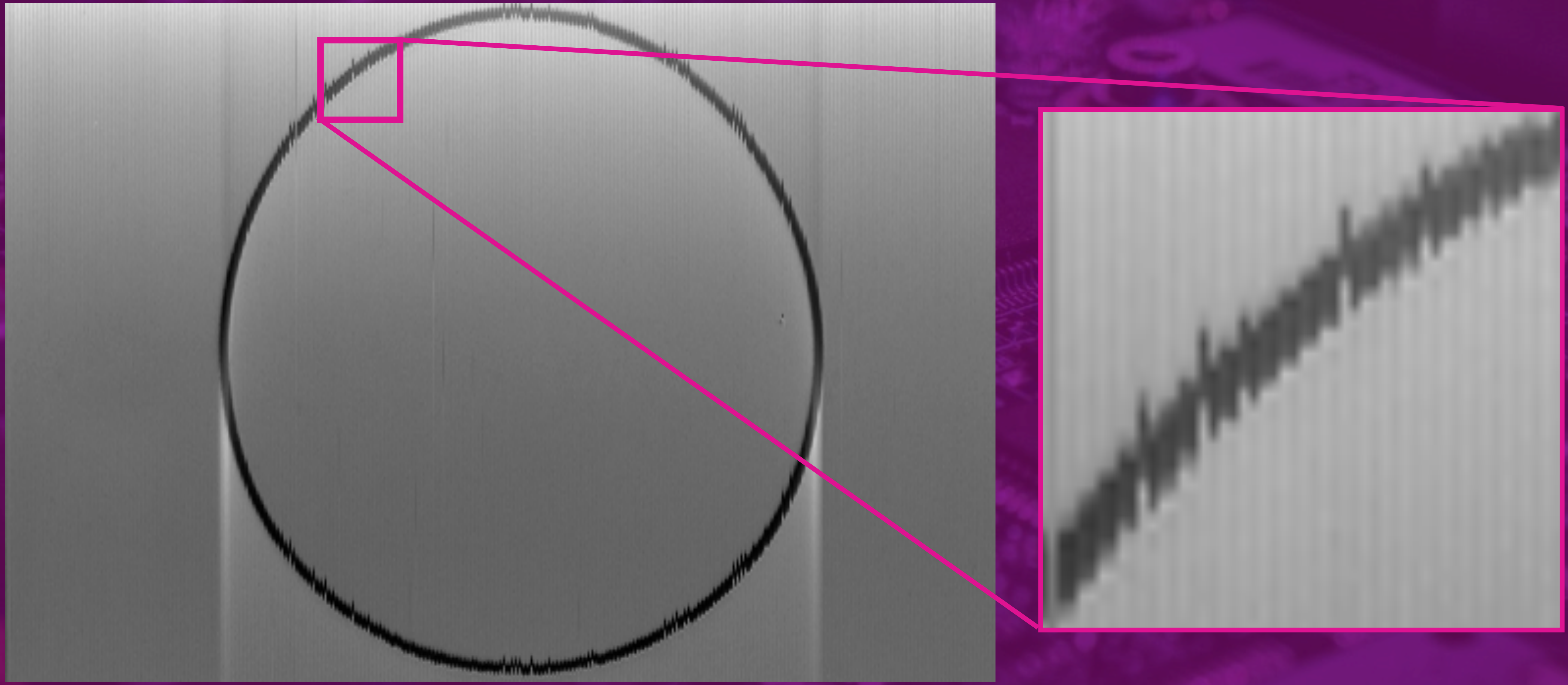
```
~/code/self/voyager_images-ruby (vim)
]
  curve = BezierCurve.new(*points).points(count: points.last[0])

  lines = samples.each_with_index.map do |s,x|
    ed_point     = [ rescale(ed.(x), range, [0, 254]), RED ]
    bezier_point = [ rescale(curve[x]&.last, range, [0, 254]), BLUE ]
    sample_point = [ rescale(s, range, [0, 255]), BLACK ]
    points_on_line([ed_point, bezier_point, sample_point], 255, WHITE)
  end

  image_from_lines(rotate_array(lines), "./#{fn}.bmp")

  embiggen_and_imgcat!(fn)
 end
end

def ex8
 WaveFile::Reader.new(
  File.expand_path('../FirstChunk.wav', File.dirname(__FILE__))
 ) do |reader|
  reader.read(42082)
  samples = reader.read(200).samples.to_a

  fn = 'FirstChunk'

  range = [-0.3, 0.3]

  points = [
   [0,  -0.131],
   [0,   0.015],
   [0.1, 0.015], # <-- pull curve to the left
   [7,   0.015], # <-- ditto
   [293, 0.015],
  ]
  curve = BezierCurve.new(*points).points(count: points.last[0])

 end
end

#ex1
#ex2
#ex3
#ex4
#ex5: puts "======\n"
ex6: puts "======\n"
ex7: puts "======\n"
ex8: puts "======\n"
bin/samples_learning                                              255,80         Bot
"bin/samples_learning" 223L, 7079C written
```

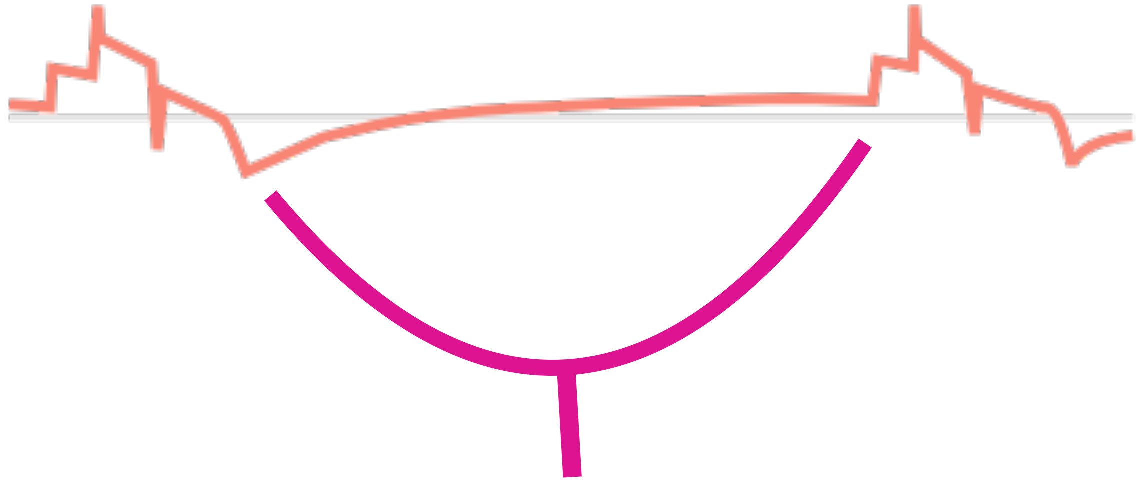
@damncabbage

Fix the 'Jitter'

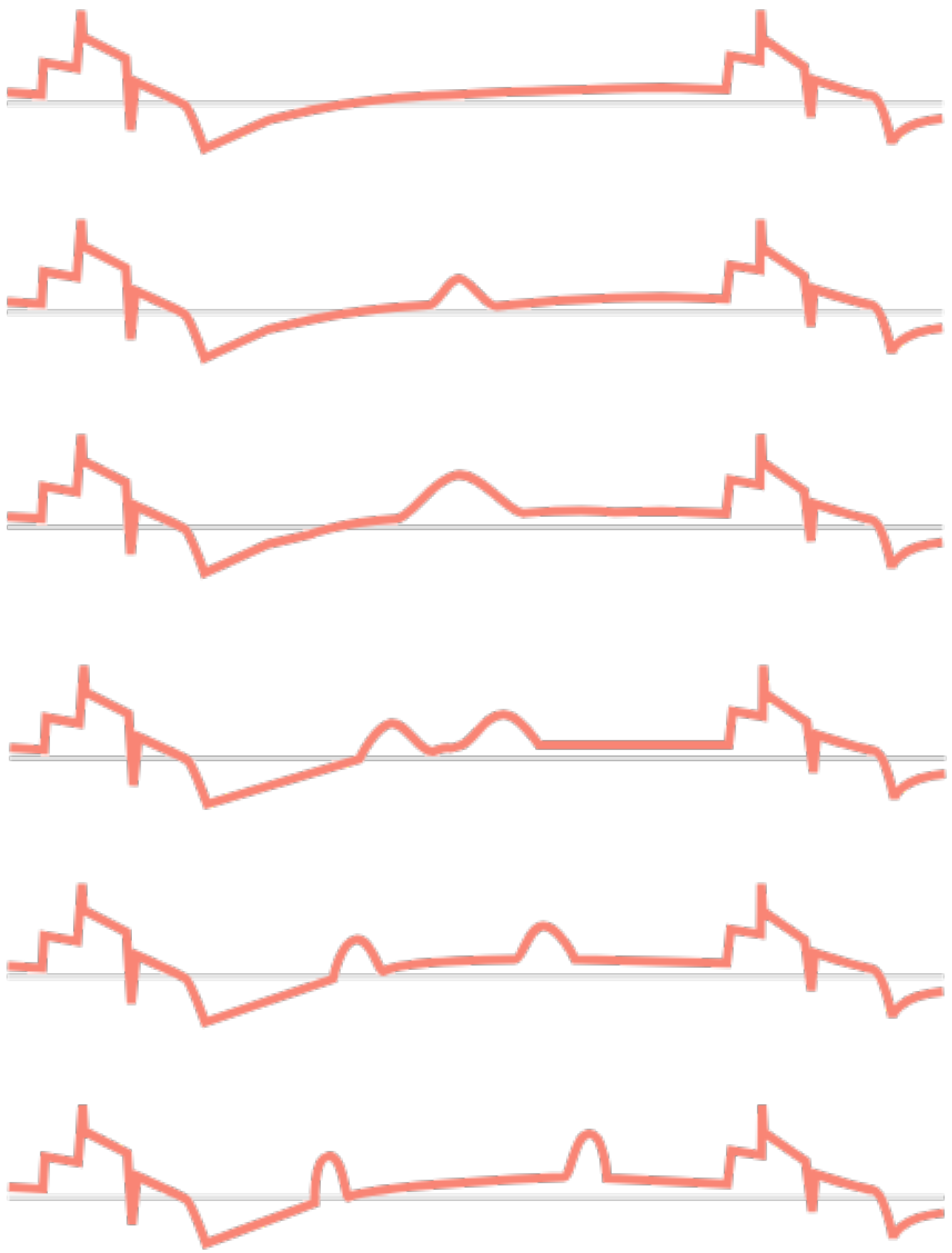


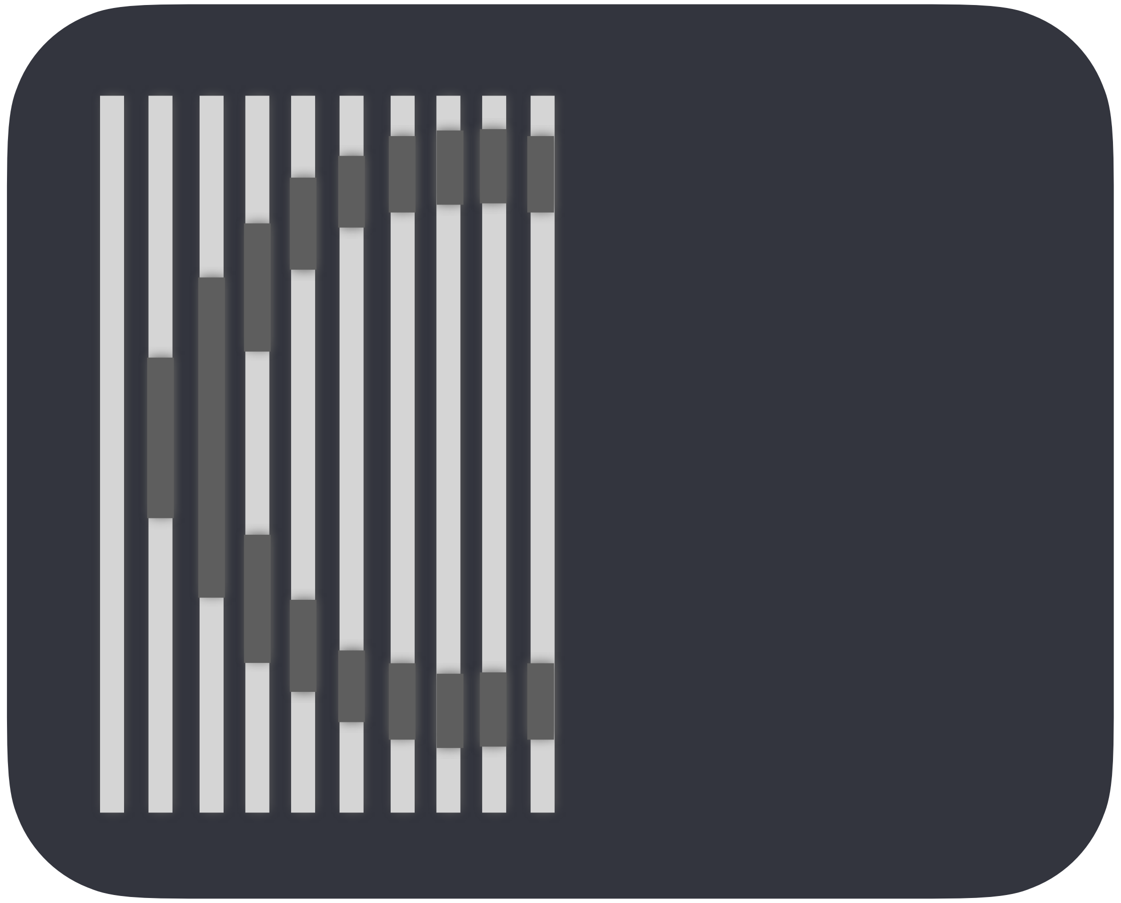
Fix the 'Jitter'

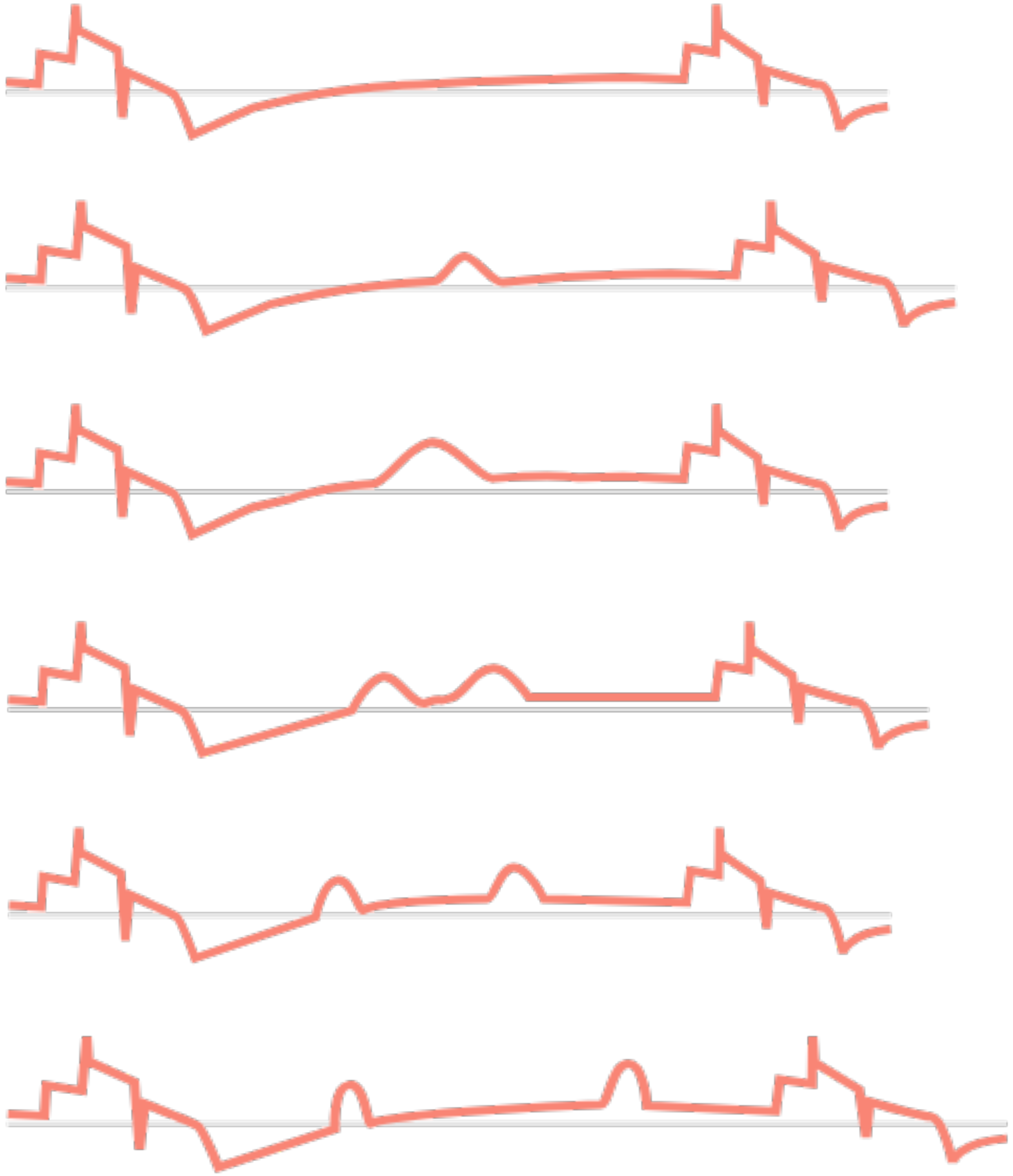


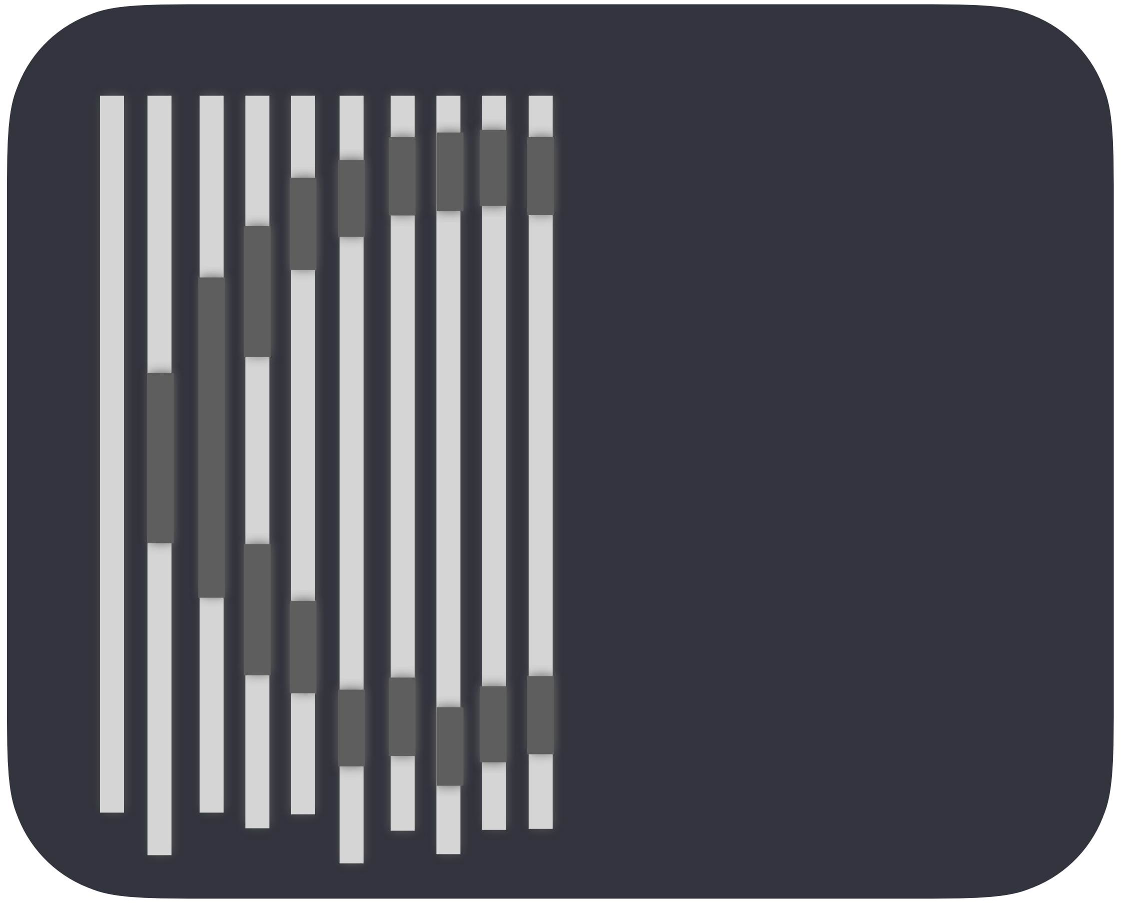


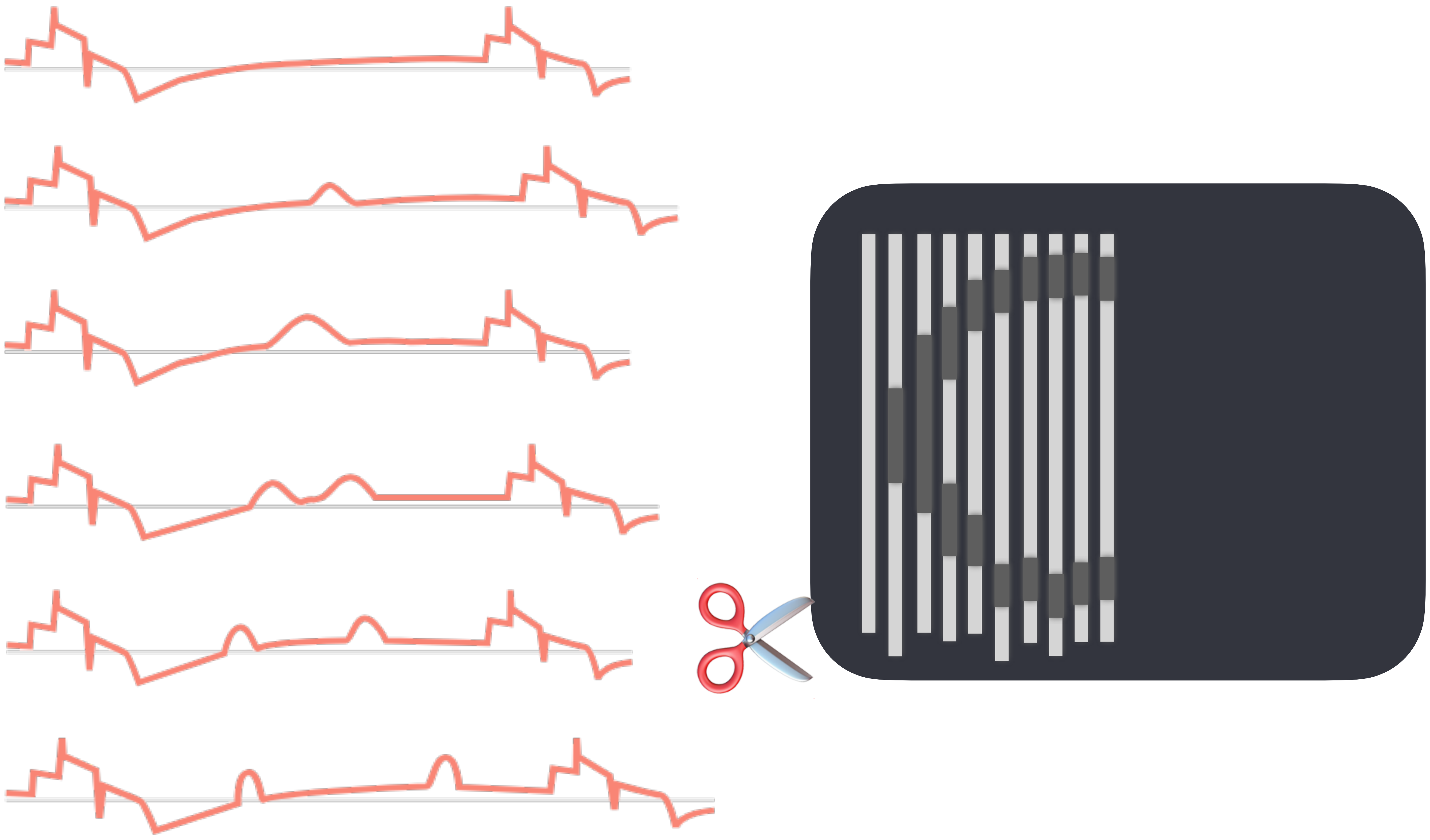
Useful bit

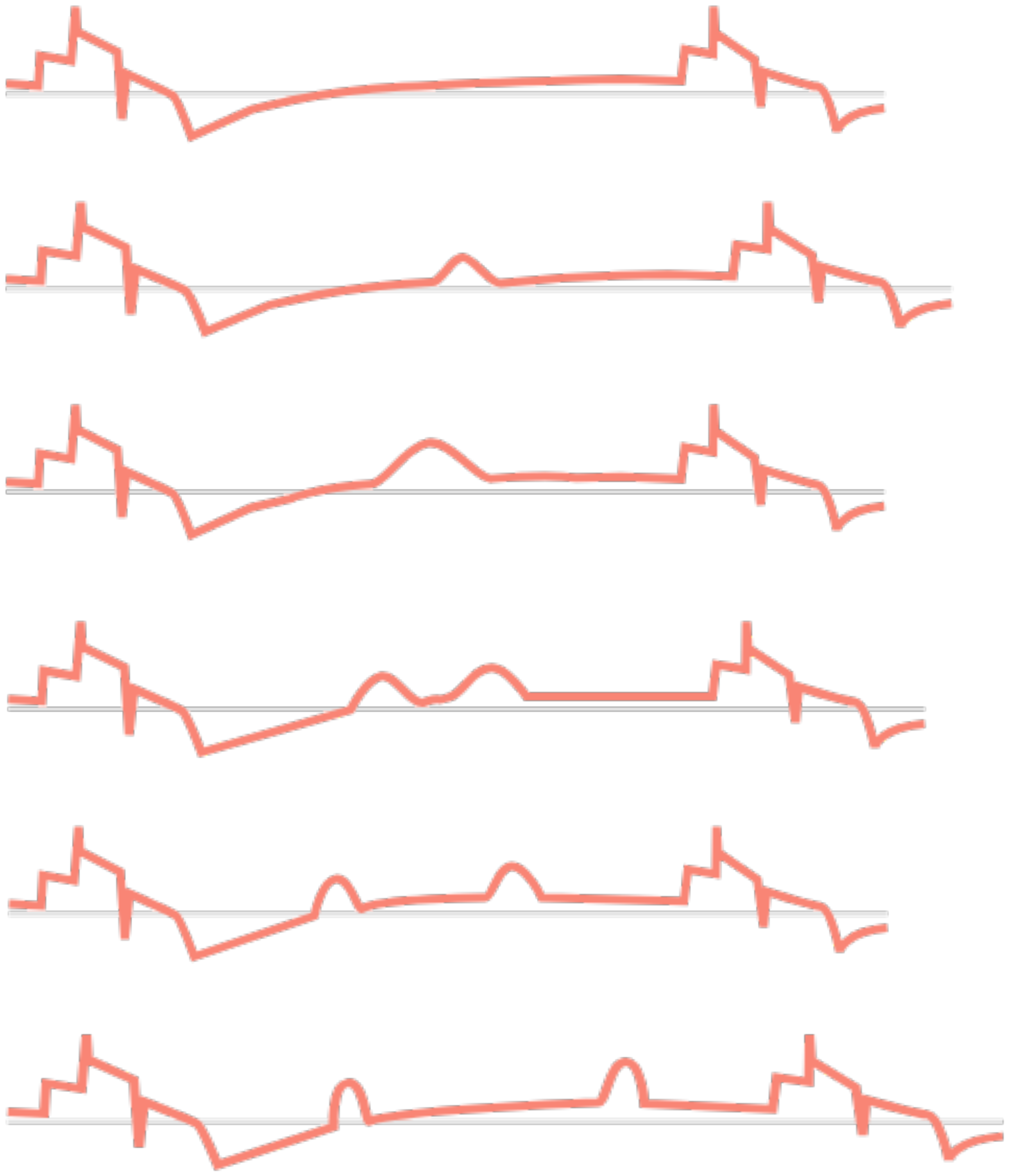


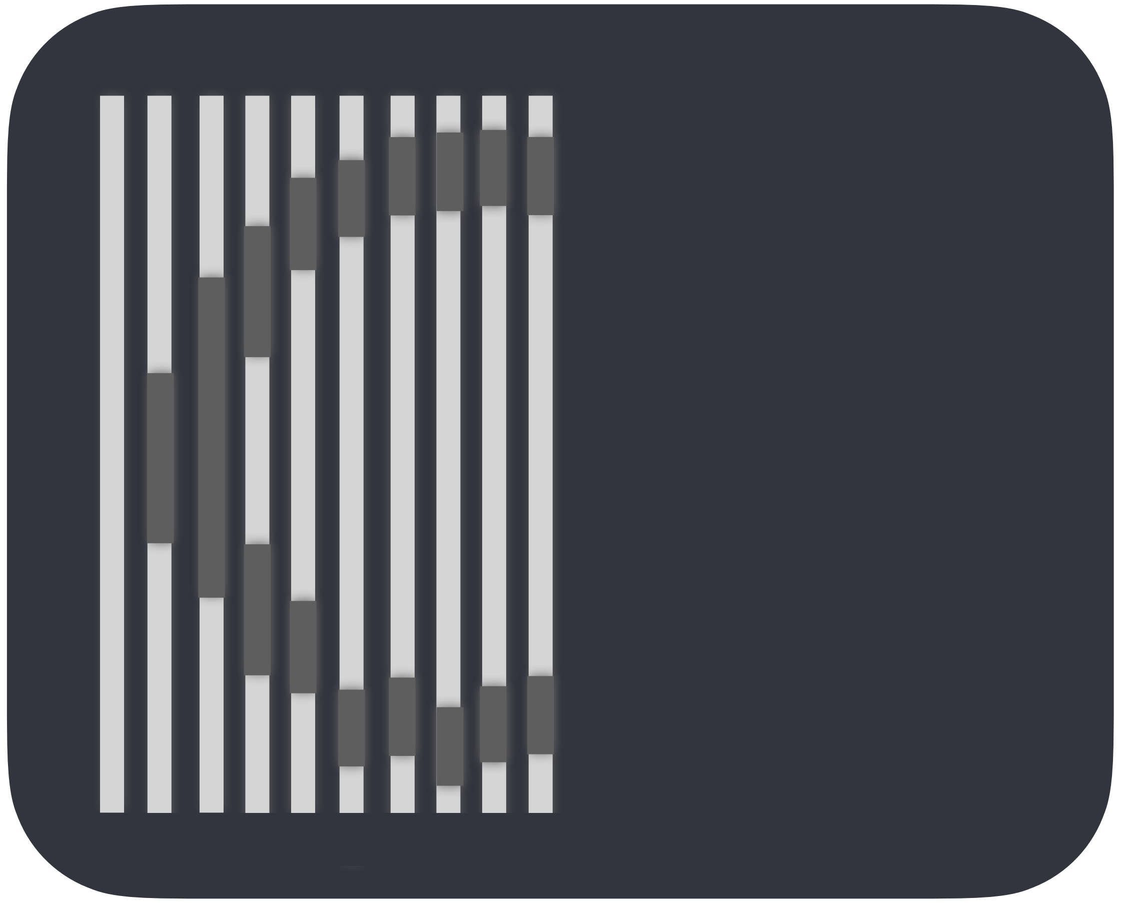










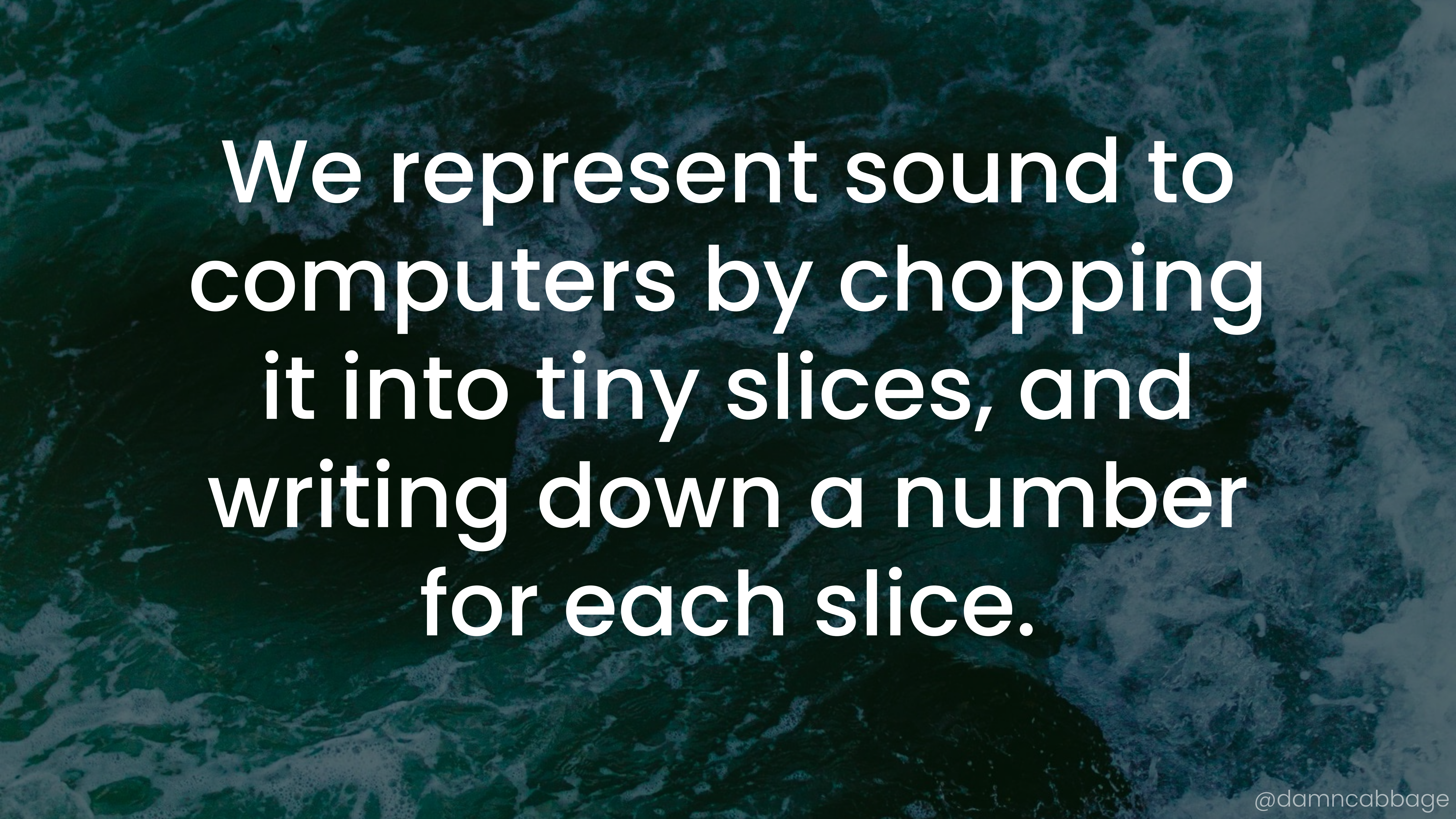


@damncabbage

Wrapping Up

Sound is a bunch of pressure waves.

How close they are together, and how powerful they are, changes how it sounds.



We represent sound to computers by chopping it into tiny slices, and writing down a number for each slice.

Pictures are made on old TVs by sweeping an electron beam around really quickly in a grill-like pattern.

We can represent images to computers to making a big grid of numbers.

Each number is a 'brightness' value. Groups of numbers instead can mean colours and transparency.

In 1977, humans sent a message to outer space.

An optimistic message, showing all the brightness and colour of the world.