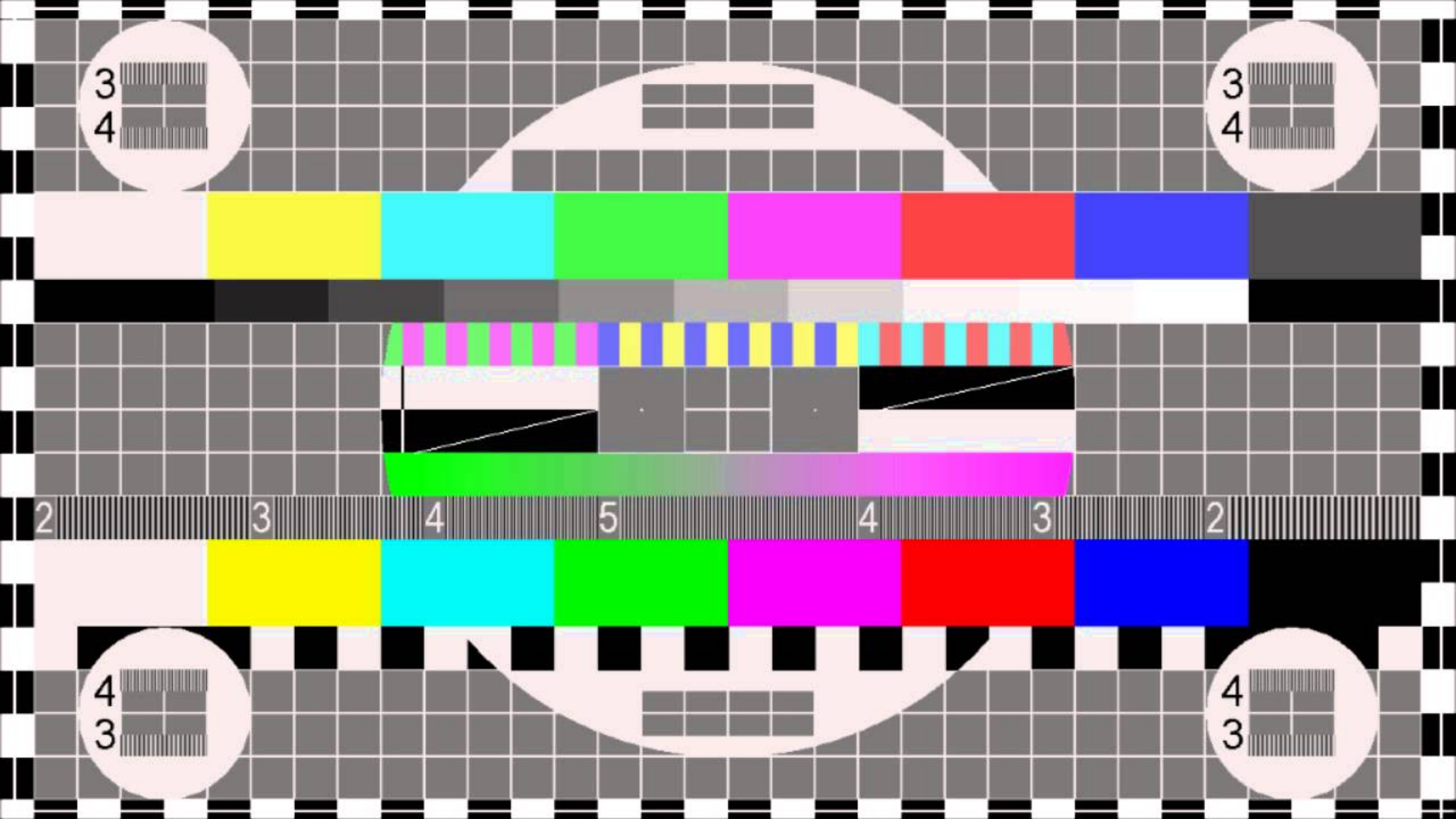
Картинки со звуком

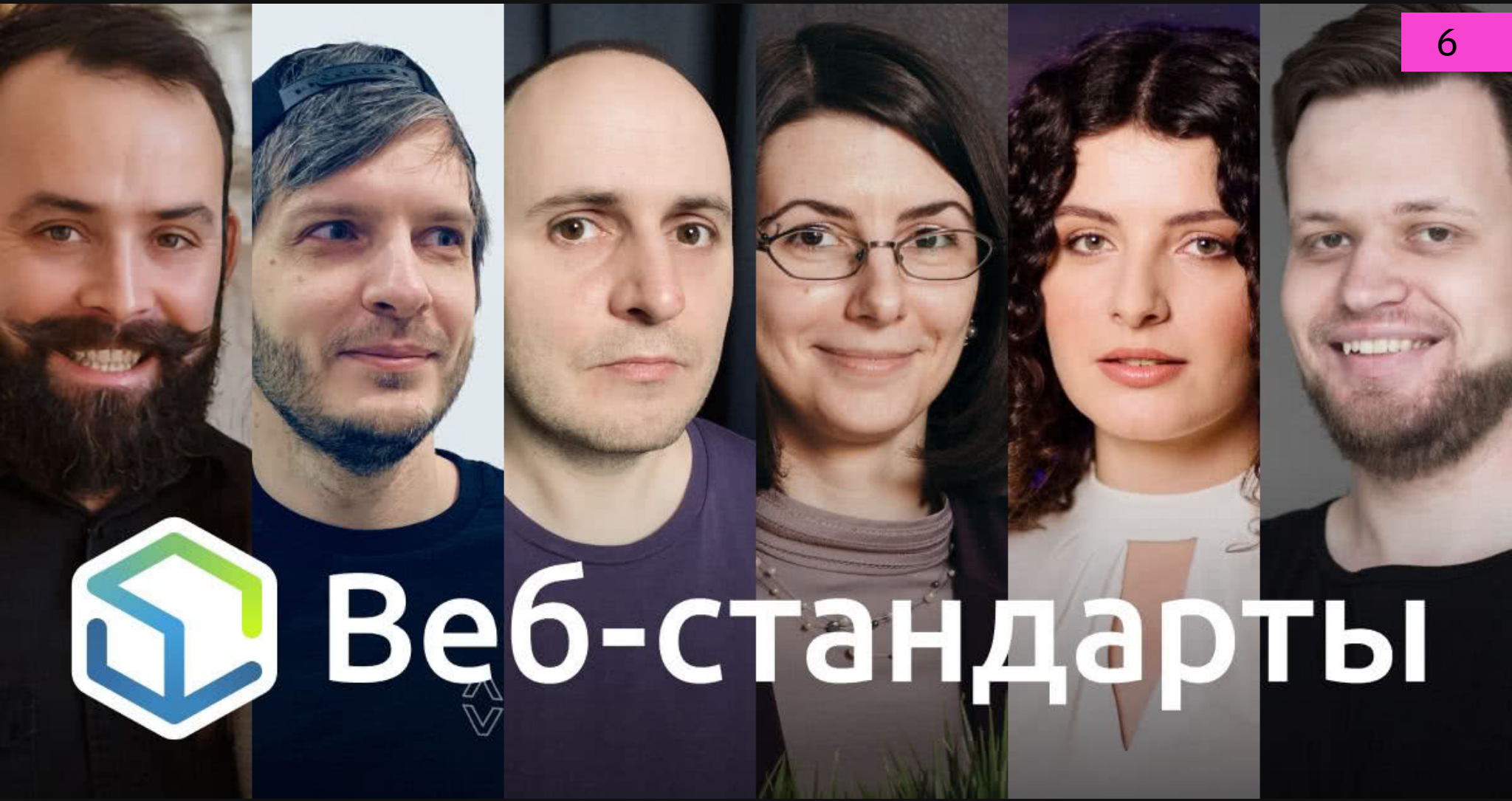
Никита Дубко, HR-Tech@Яндекс

Кто я?

- доброжелюбный бородач
- редактор новостей в Веб-стандартах
- пятый голос одноименного подкаста
- Google Developer Expert по вебу
- старший разработчик в HR-Tech Яндекса

Слышу глазами 

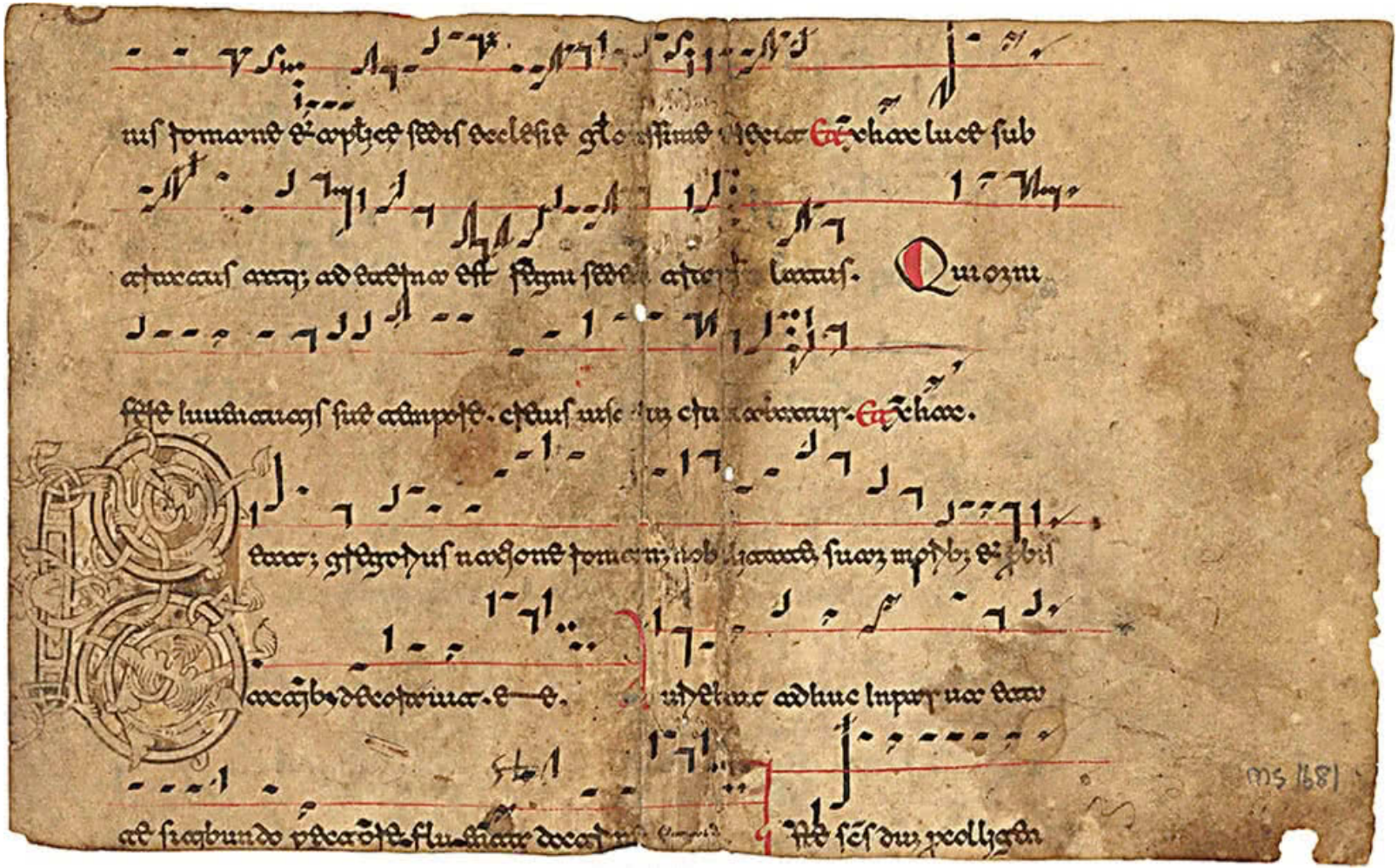

3
4
3
4
2
3
4
5
4
3
2
4
3
4
3


Веб-стандарты

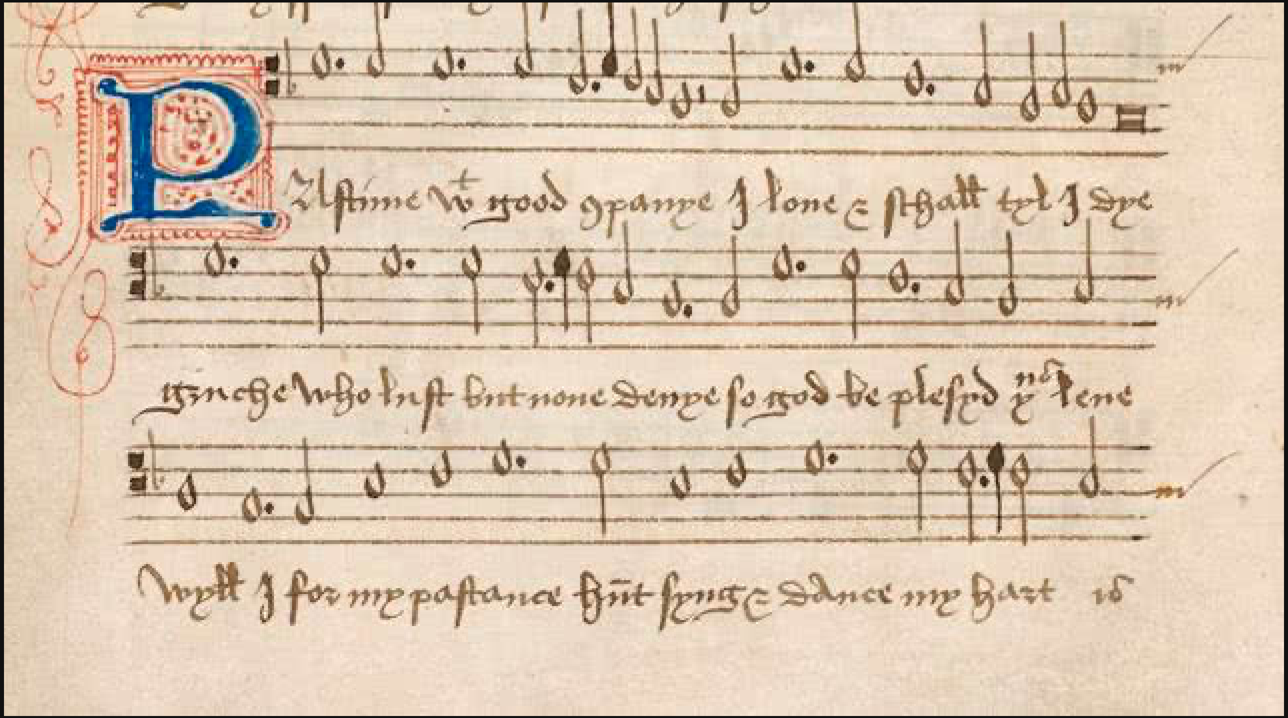
Я вижу, где смеётся Маша.

Вадим Макеев

Пятиминутка истории



Гвидо из Ареццо, XI век

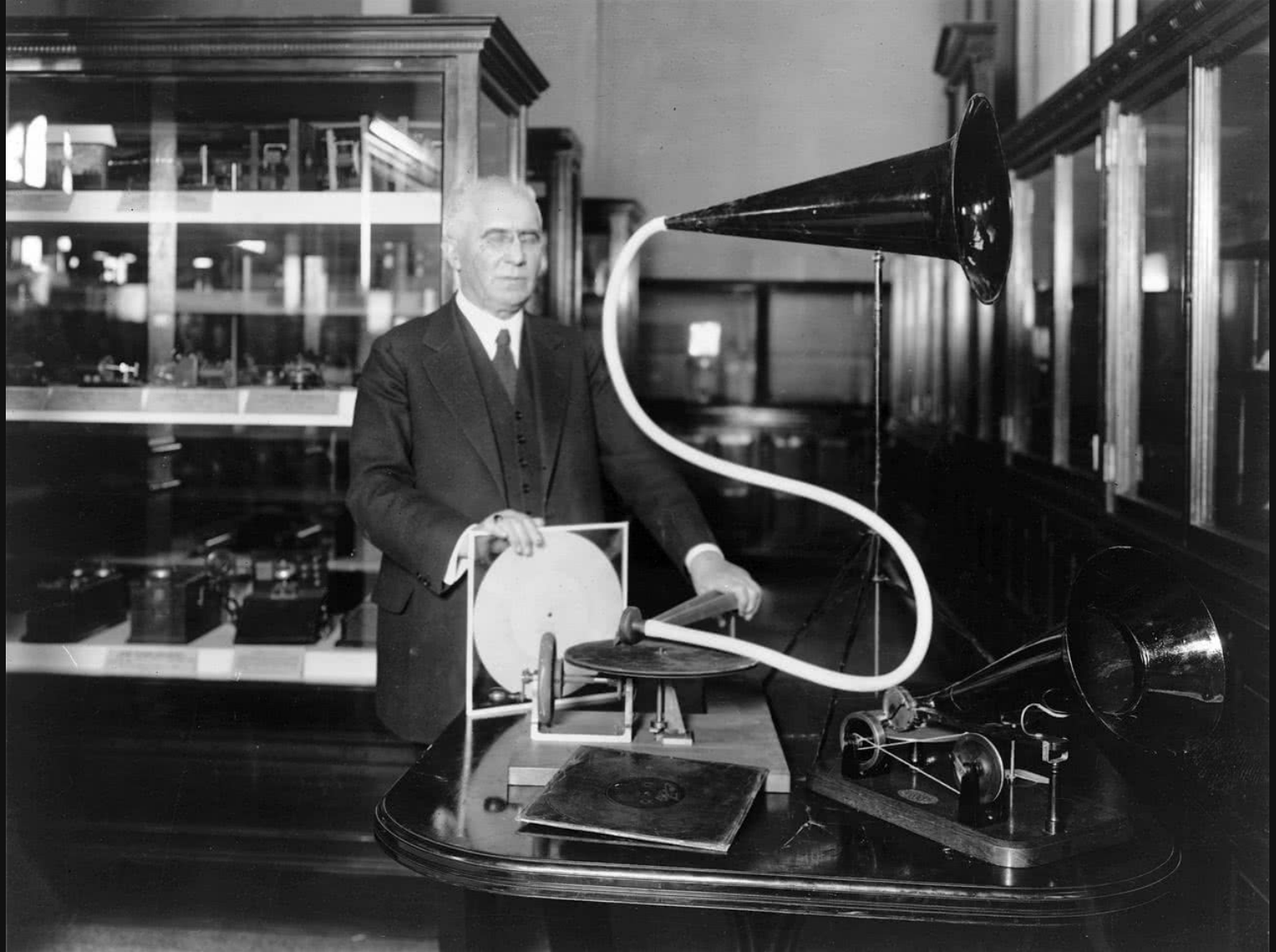

Pastime wt good companye I love & shall tyl I dye
grucche who lust but none denye so god be plesyd thus leve
wyll I for my pastance hunt syng & dance my hart is


Ideal Sublime Harmonic

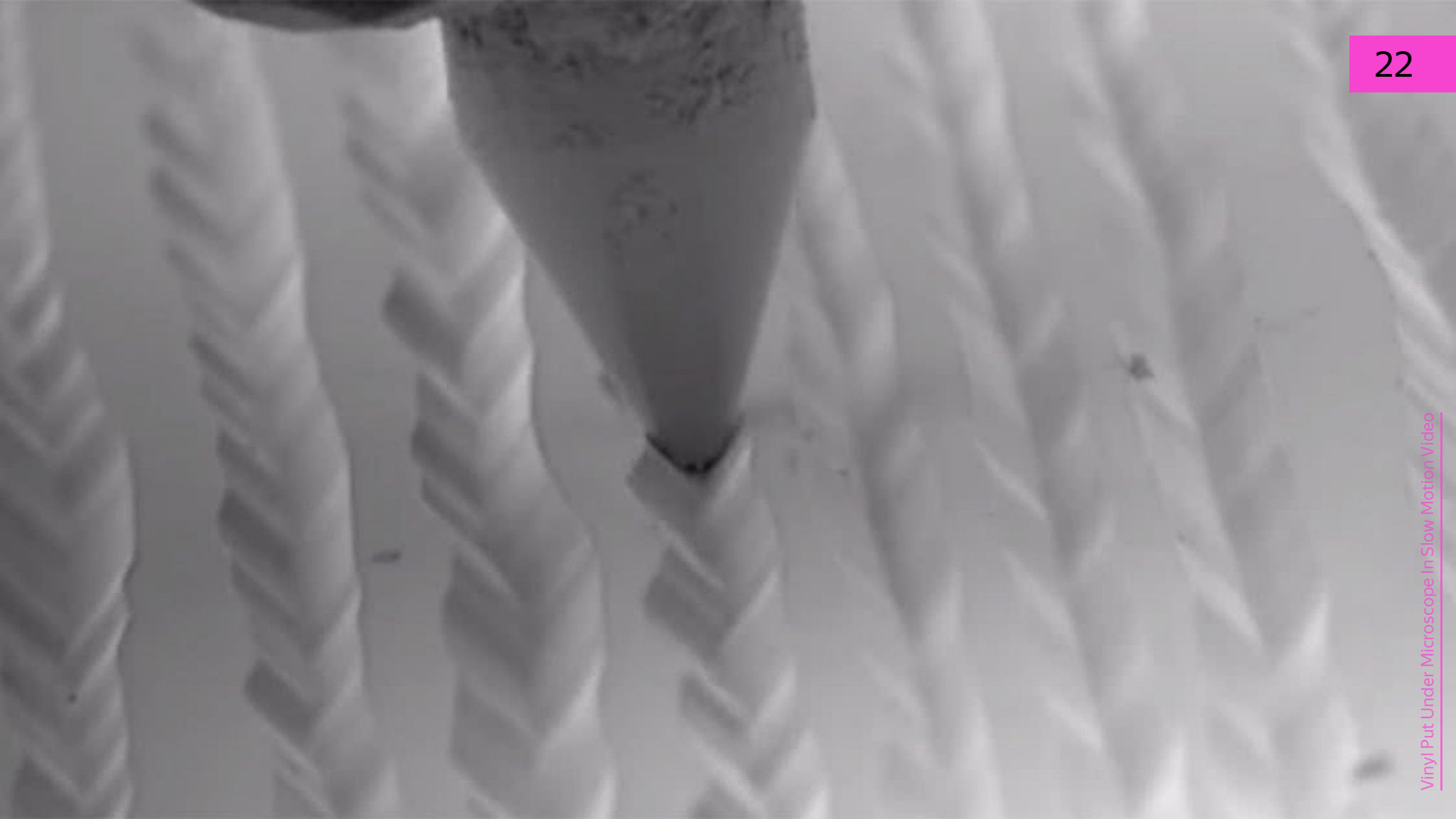
1877


1,00
0,20
FRANCE
CHARLES CROS
1842
1888
CHARLES CROS
3 DECEMBRE 1977
FABREZAN & PARIS
CENTENAIRE
DU
PHONOGRAPHE
1877–1977
Charles Cros

Phonograph



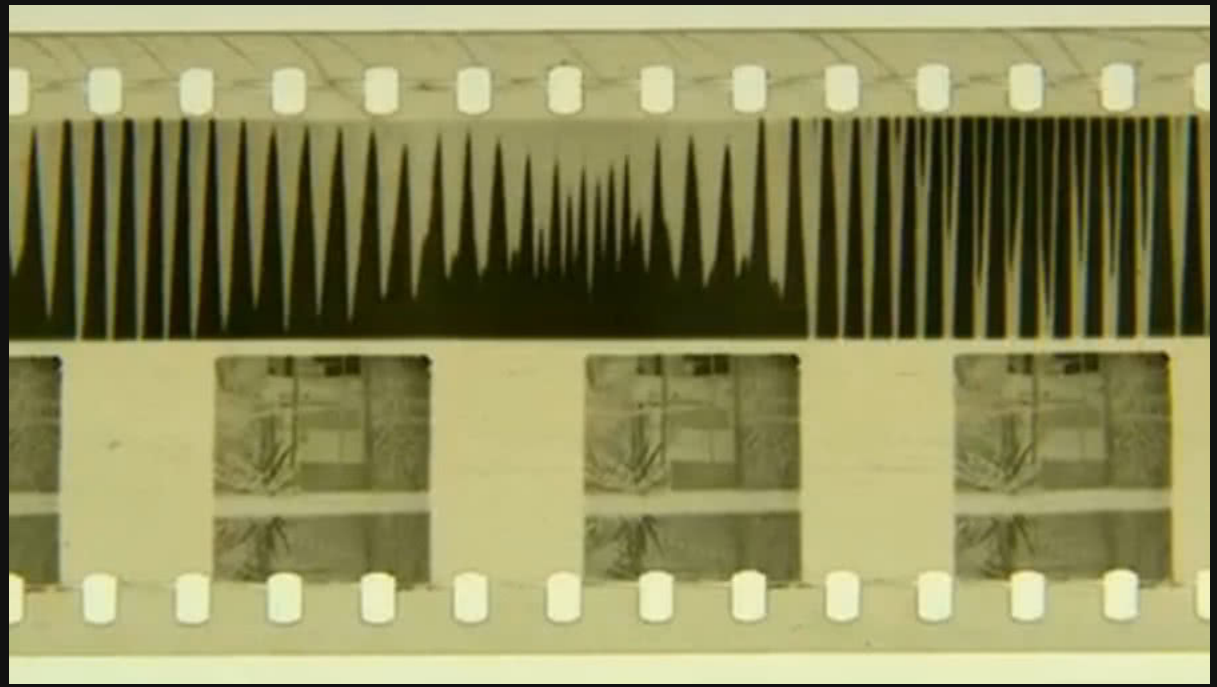
Emile Berliner

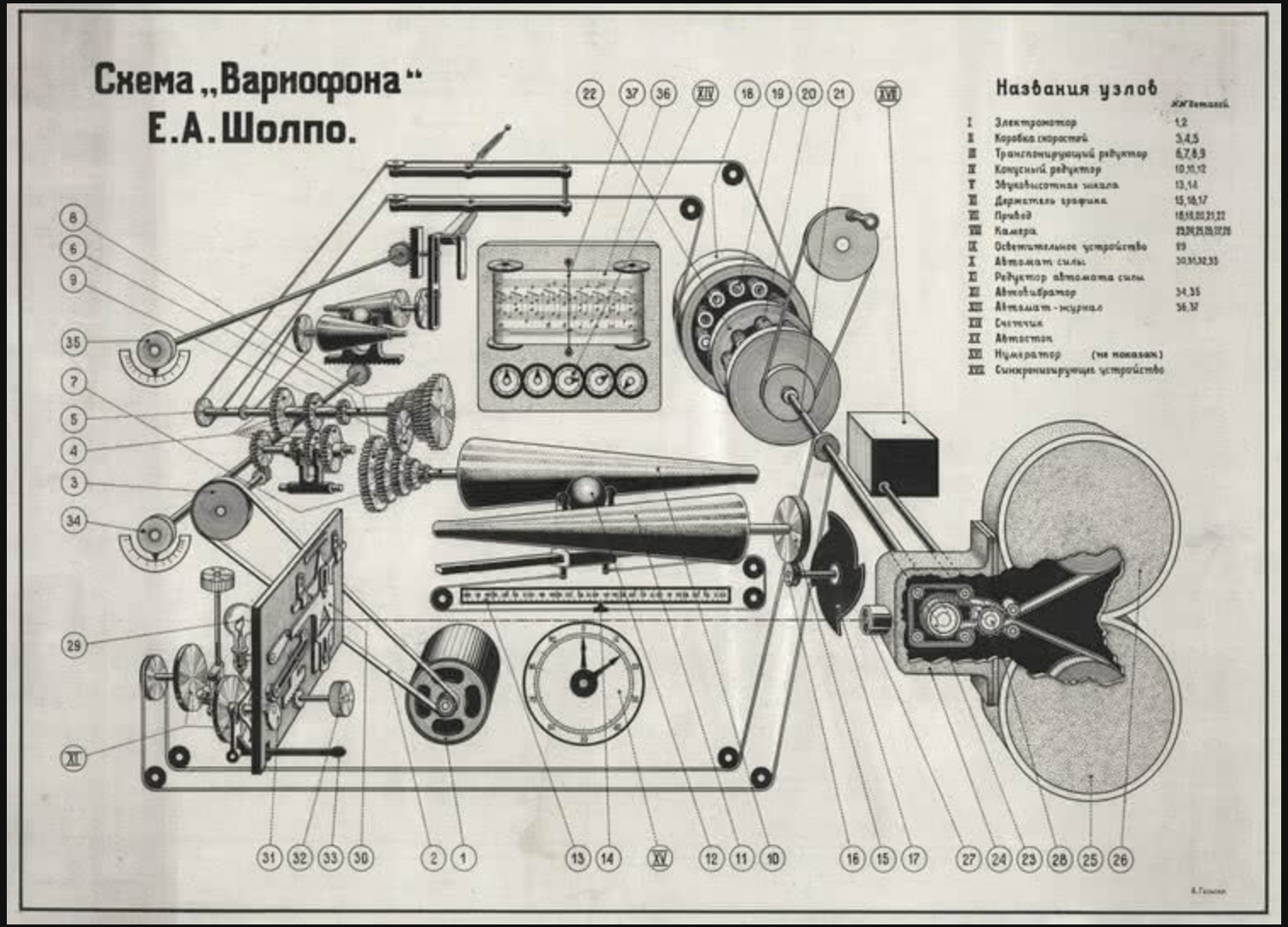


Vinyl Put Under Microscope In Slow Motion Video

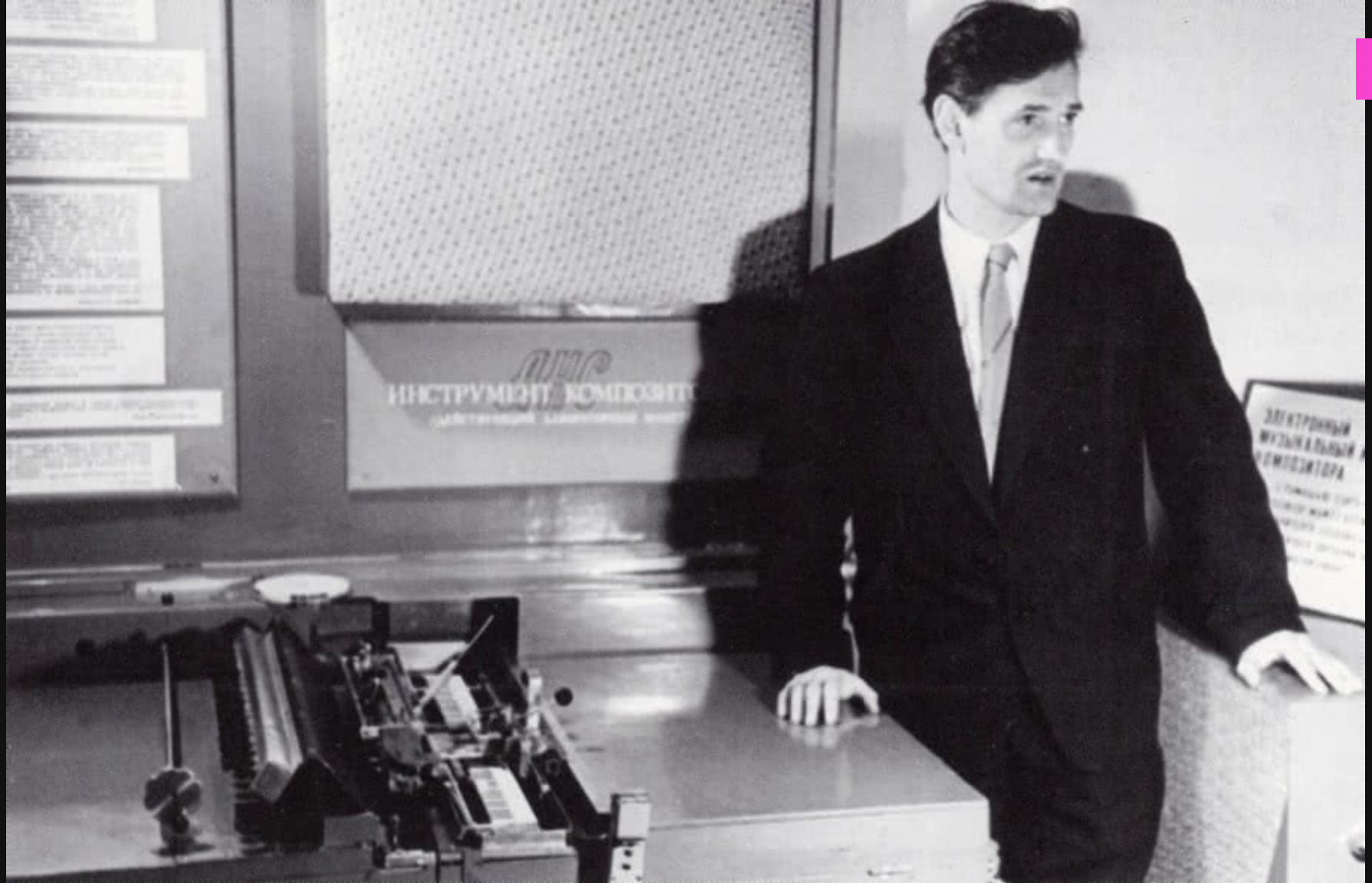
Пластинки изменили мировую экономику 

1904





Синтезированный «бумажный звук»



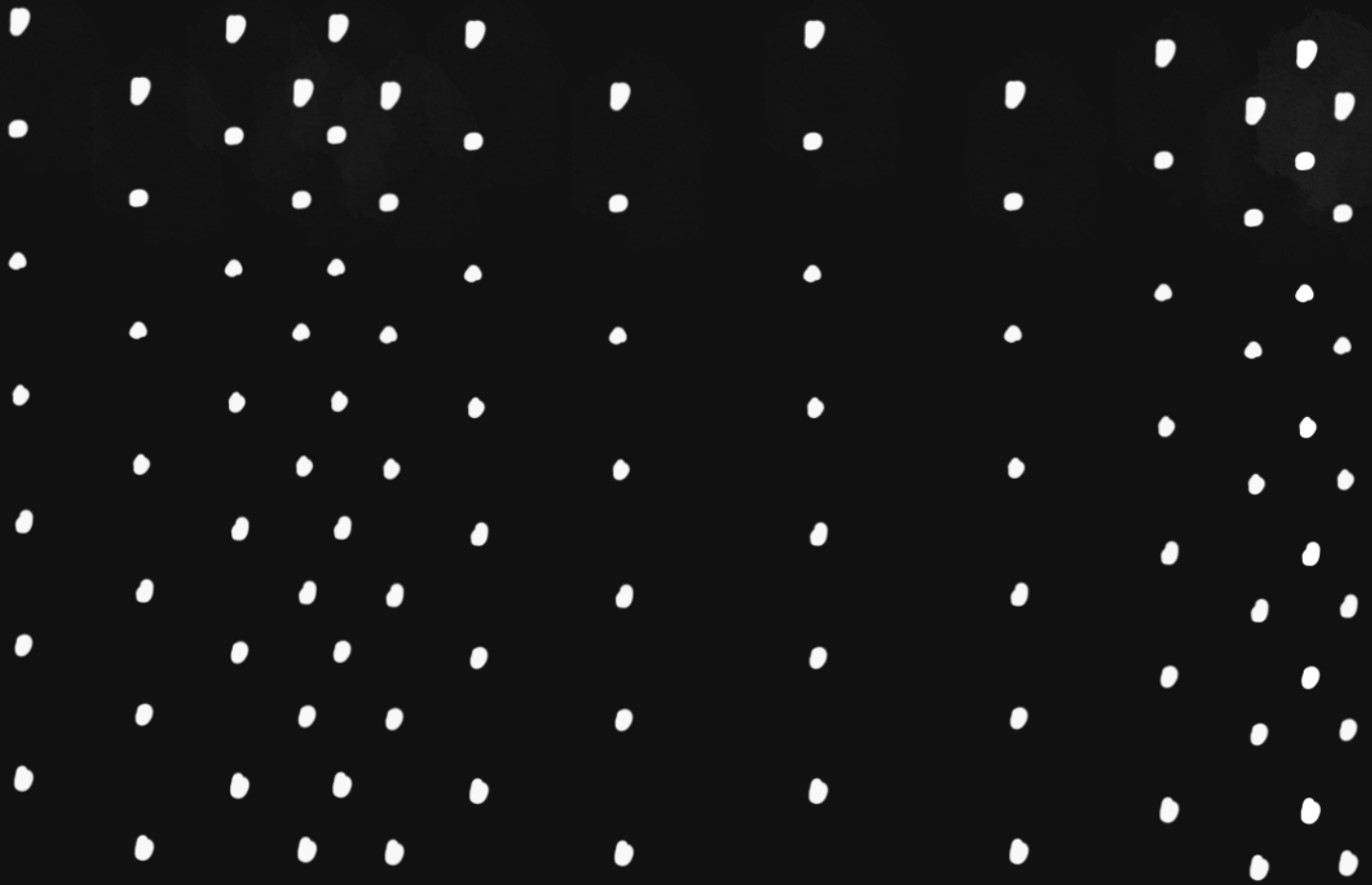
Евгений Мурзин показывает свое детище. ВДНХ. 1962

Оптическая реализация алгоритма преобразования Фурье 🤯

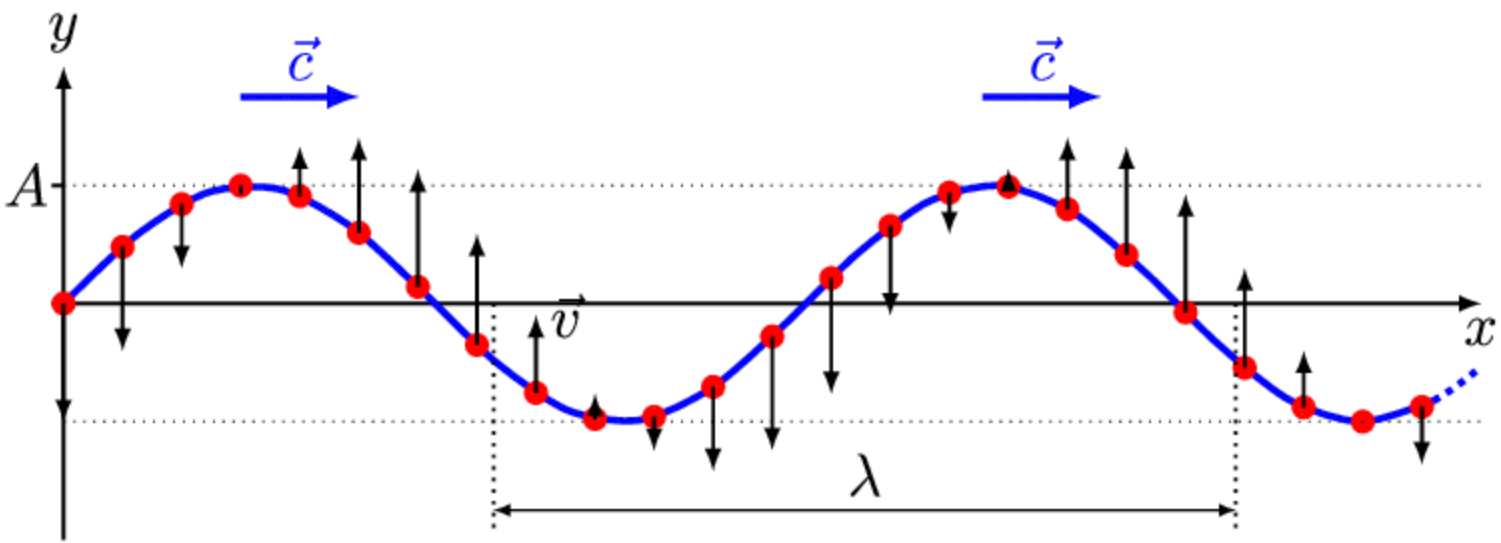
Что такое **звук**?

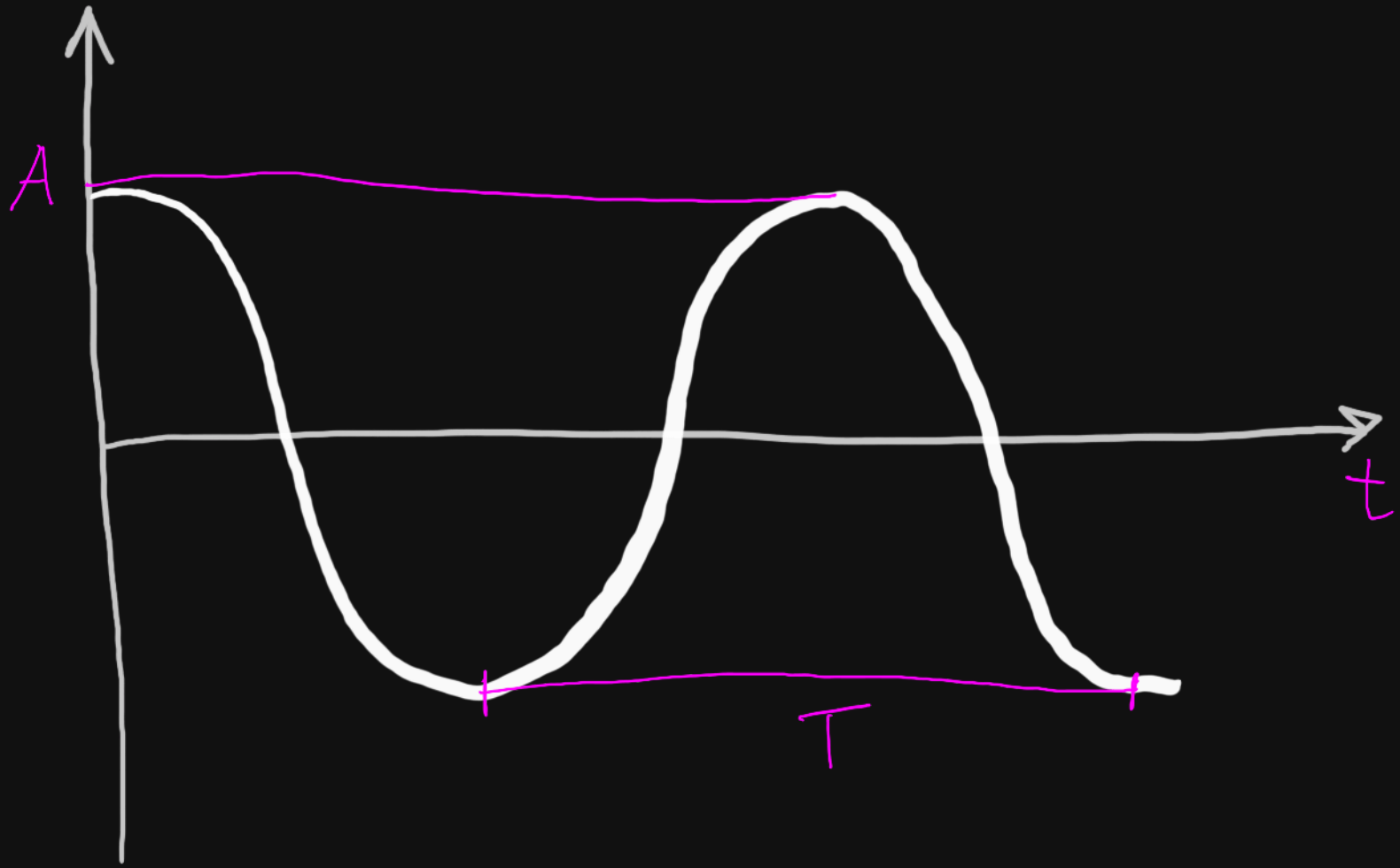
Дисклеймер

Автор не имеет докторской степени по физике звука. Дальнейшие объяснения основаны на собственном опыте и активном гуглении. Если вы считаете, что автор не прав, не стесняйтесь вступать в дискуссии после доклада. Если это кто-то читает, то вот вам интересный факт: шум моря, который мы слышим через морскую раковину, на самом всего лишь звук крови, протекающей по нашим сосудам.



〰 Упругие волны механических колебаний


y
$\vec{c}$
$\vec{c}$
A
$\vec{v}$
x
λ


A
t
T

$$y = y_0 \cdot \sin(\omega t + \varphi)$$

$$f = \frac{1}{T}$$

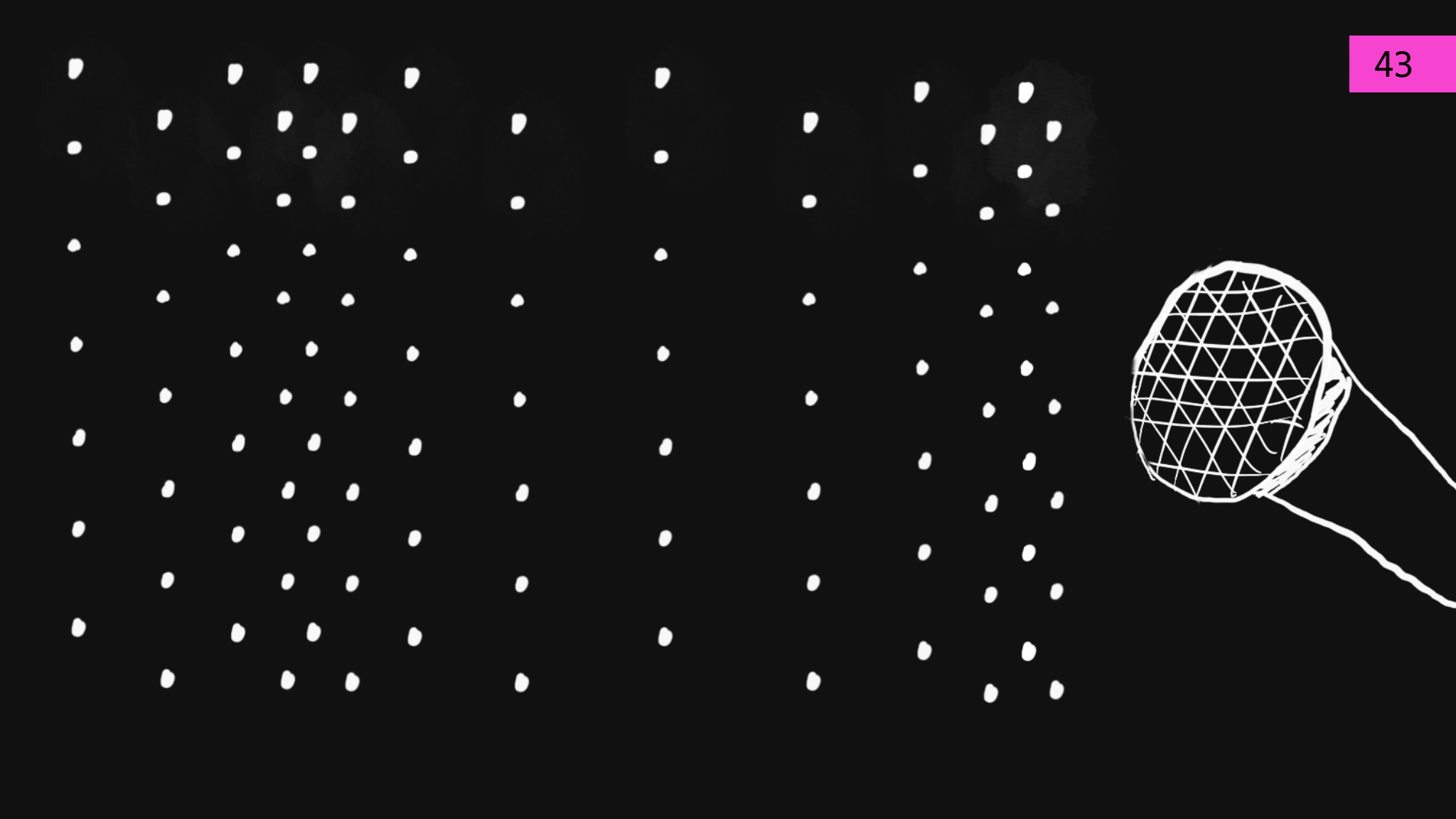
16–20 000 Гц

Инфразвук < 16 Гц

Ультразвук > 20 000 Гц

$$p = p_0 \cdot \sin(\omega t + \varphi)$$

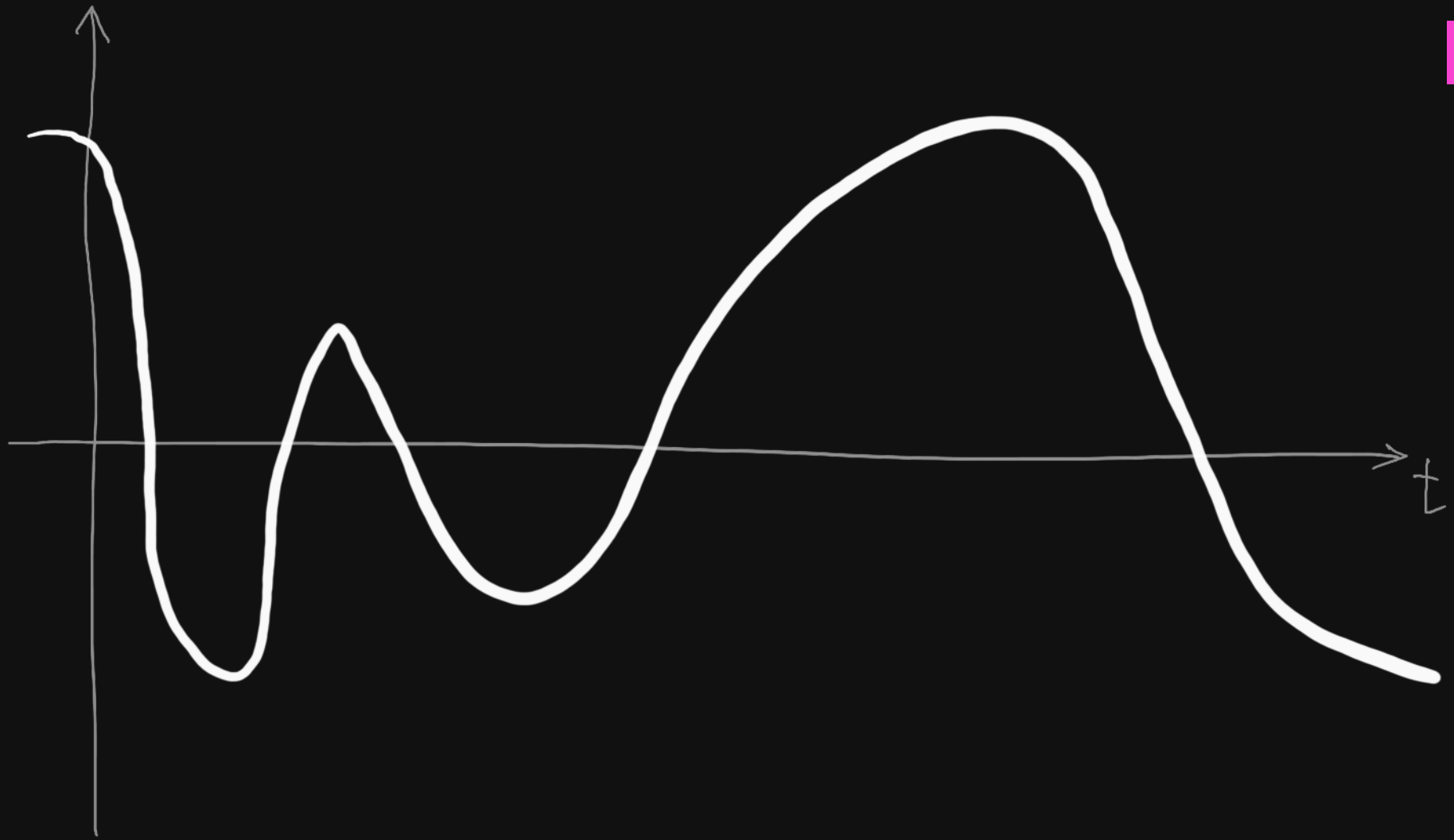
$$L_p = 20 \lg \frac{p}{20\ \mu\text{Па}}$$

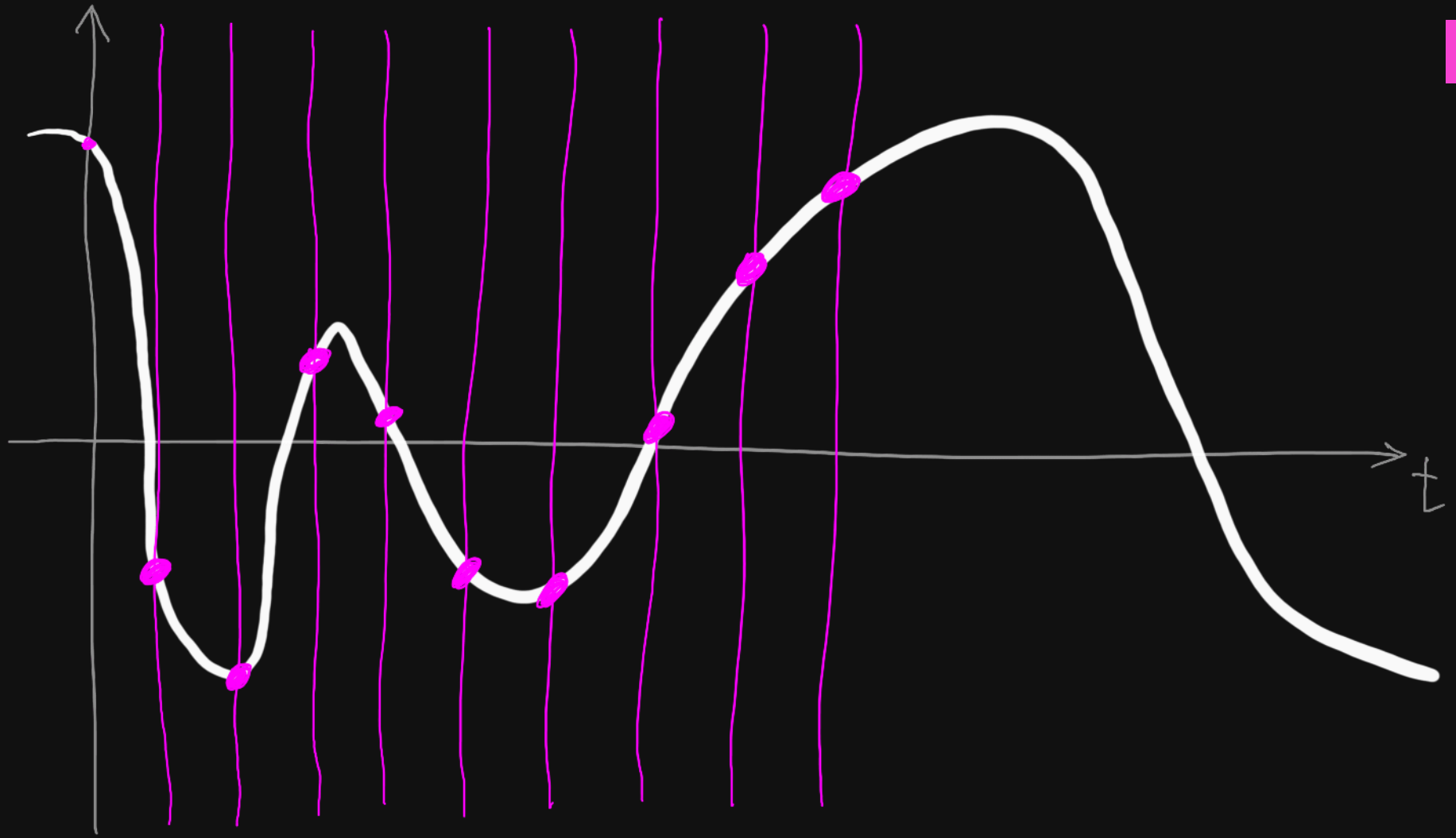


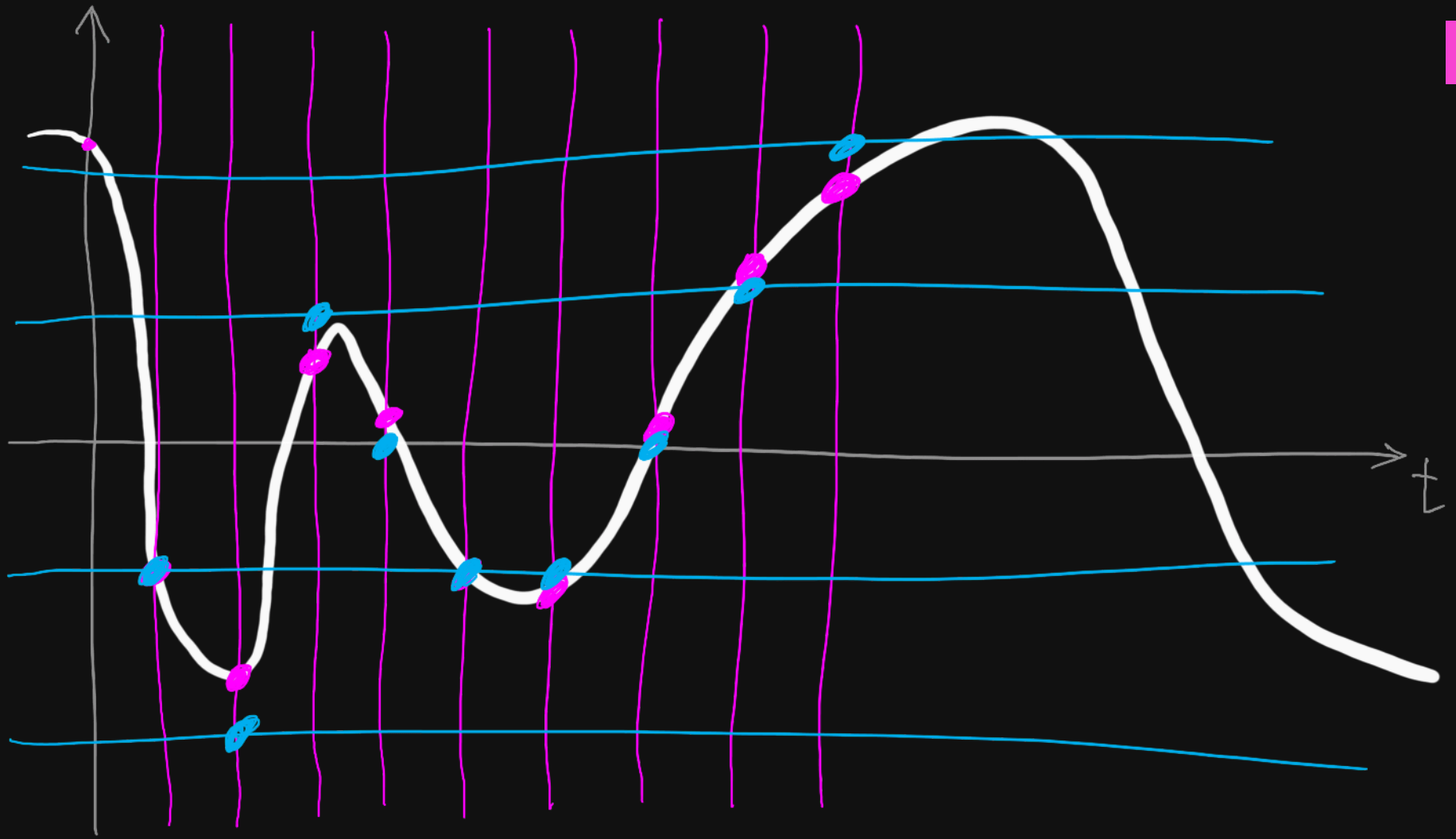
1933

📖 Теорема Котельникова (Найквиста-Шеннона)

> *Любую функцию F(t), состоящую из частот от 0 до f_1, можно непрерывно передавать с любой точностью при помощи чисел, следующих друг за другом через $1/(2f_1)$ секунд*


t


t


t

частота дискретизации

44,1 кГц

разрядность квантования

16 бит

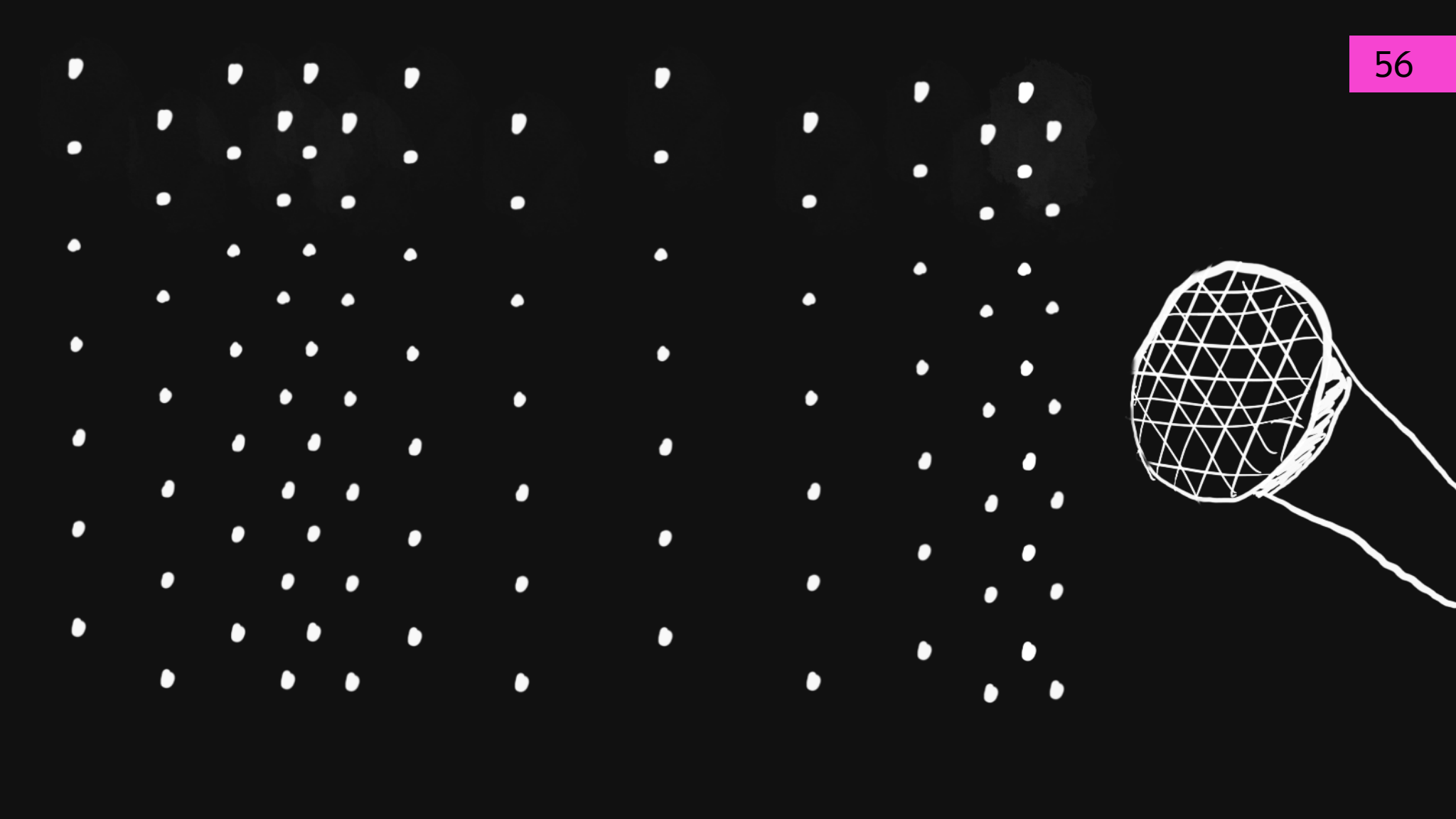
16 бит

65 536

PCM
Pulse Code Modulation

ИКМ
Импульсно-Кодовая Модуляция

АЦП
Аналогово-Цифровой Преобразователь



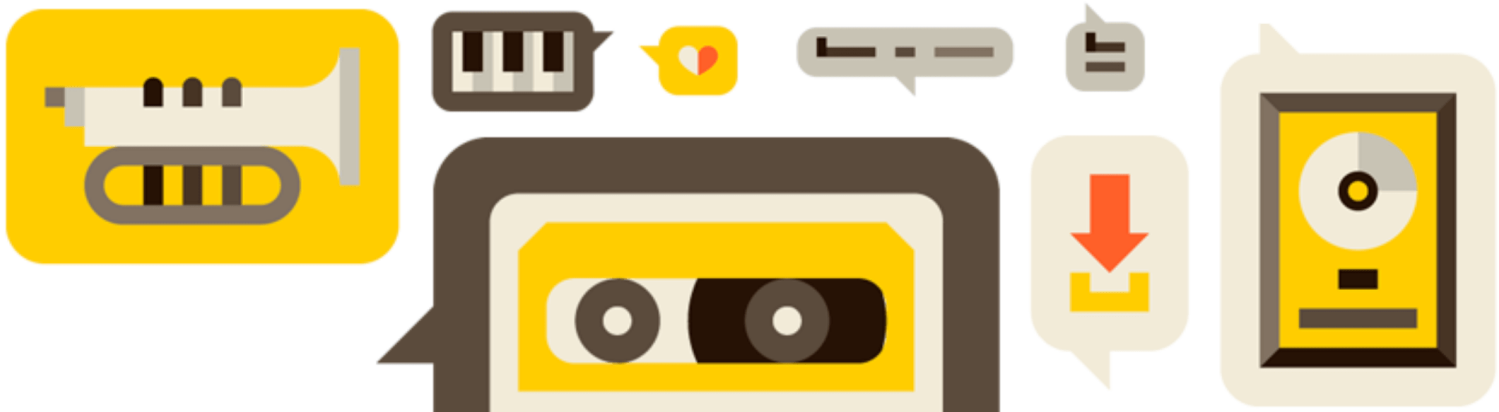
gheljenor 12 ноября 2015 в 19:17

Теория звука. Что нужно знать о звуке, чтобы с ним работать. Опыт Яндекс.Музыки

Блог компании Яндекс , Алгоритмы *, Математика *

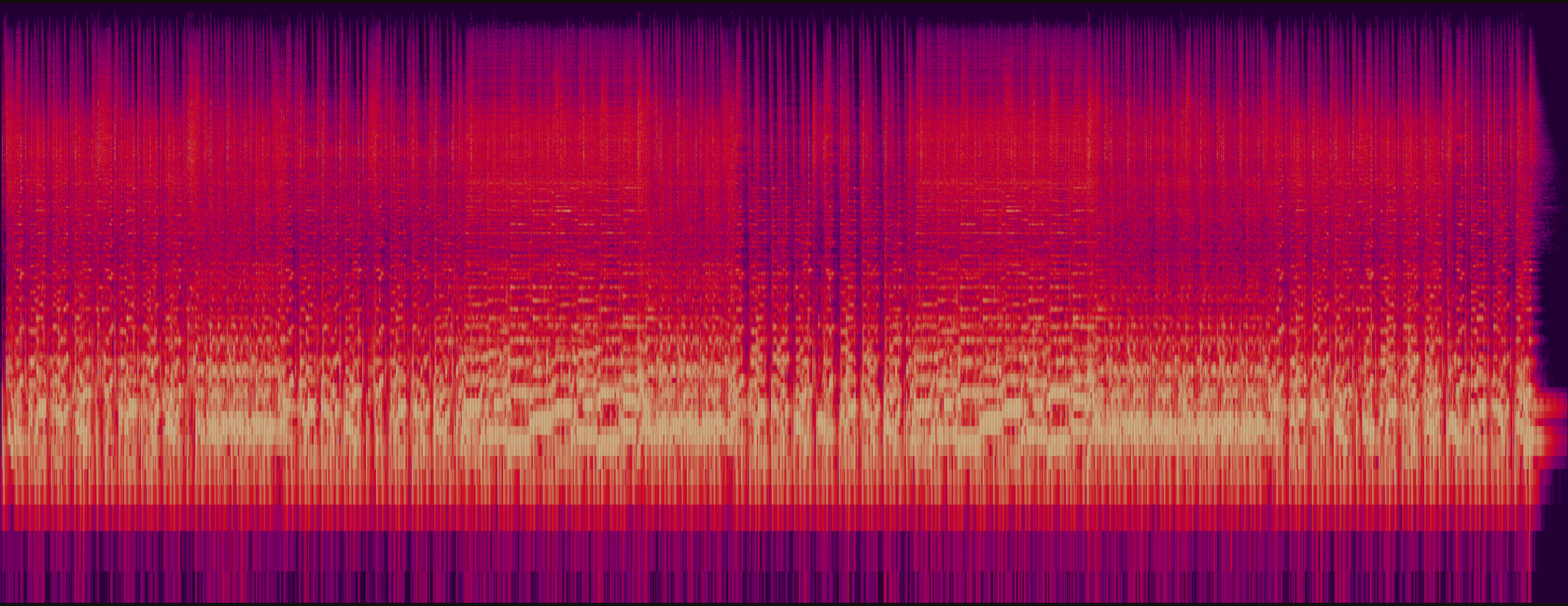
Звук, как и цвет, люди воспринимают по-разному. Например, то, что кажется слишком громким или некачественным одним, может быть нормальным для других.

Для работы над Яндекс.Музыкой нам всегда важно помнить о разных тонкостях, которые таит в себе звук. Что такое громкость, как она меняется и от чего зависит? Как работают звуковые фильтры? Какие бывают шумы? Как меняется звук? Как люди его воспринимают.

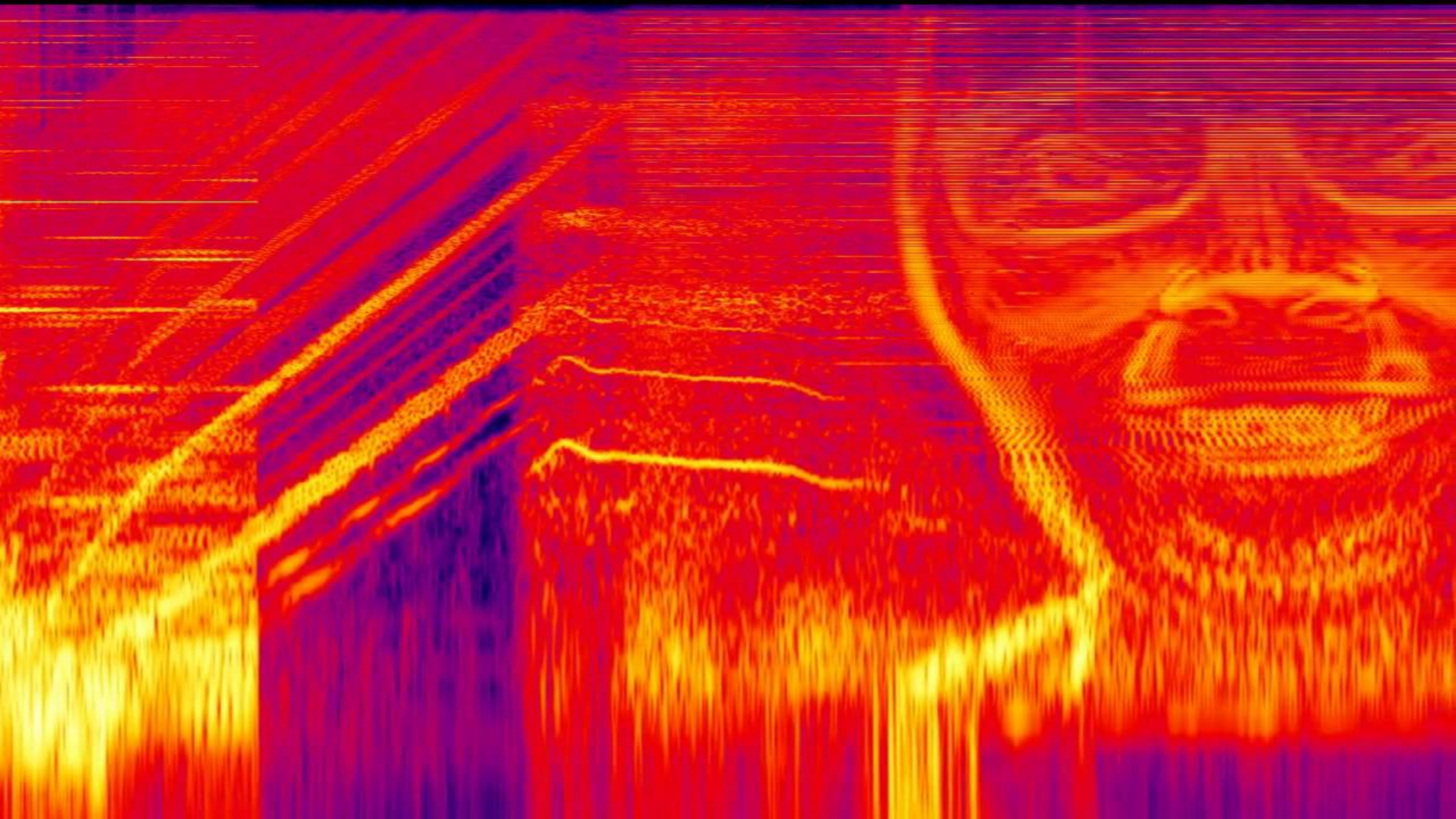


Время, частота, амплитуда

Время X,
частота Y,
амплитуда Z



Спектрограмма 🎨



Web Audio API


Advanced Sound for Games and Interactive Apps
Web Audio API
O'REILLY®
Boris Smus

W3C Editor's Draft

TABLE OF CONTENTS

Web Audio API

Editor's Draft, 28 September 2021

This version:
https://webaudio.github.io/web-audio-api/

Latest published version:
https://www.w3.org/TR/webaudio/

Previous Versions:
https://www.w3.org/TR/2021/CR-webaudio-20210114/
https://www.w3.org/TR/2020/CR-webaudio-20200611/
https://www.w3.org/TR/2018/CR-webaudio-20180918/
https://www.w3.org/TR/2018/WD-webaudio-20180619/
https://www.w3.org/TR/2015/WD-webaudio-20151208/
https://www.w3.org/TR/2013/WD-webaudio-20131010/
https://www.w3.org/TR/2012/WD-webaudio-20121213/
https://www.w3.org/TR/2012/WD-webaudio-20120802/
https://www.w3.org/TR/2012/WD-webaudio-20120315/
https://www.w3.org/TR/2011/WD-webaudio-20111215/

Feedback:
public-audio@w3.org with subject line "`[webaudio] … message topic …`" (archives)

Implementation Report:
implementation-report.html

Test Suite:
https://github.com/web-platform-tests/wpt/tree/master/webaudio

Issue Tracking:
GitHub

Editors:
Paul Adenot (Mozilla (https://www.mozilla.org/))
Hongchan Choi (Google (https://www.google.com/))

Former Editors:

yandex / audio-js Public

Watch 13 Star 156 Fo

Code Issues 5 Pull requests Actions Projects Wiki Security Insights

master audio-js / tutorial / web-audio-api.md Go to file ...

gheljenor Убраны ссылки на gitbug-pages Latest commit f4ccfae on 25 May 2016 History

1 contributor

397 lines (246 sloc) 73.4 KB Raw Blame

Web Audio API

Web Audio API - это технология, позволяющая существенно расширить возможности воспроизведения звука в браузере. К сожалению, на данный момент эта технология очень молодая и ее поддержка есть только в свежих версиях популярных десктопных браузеров и практически отсутствует в мобильных браузерах.

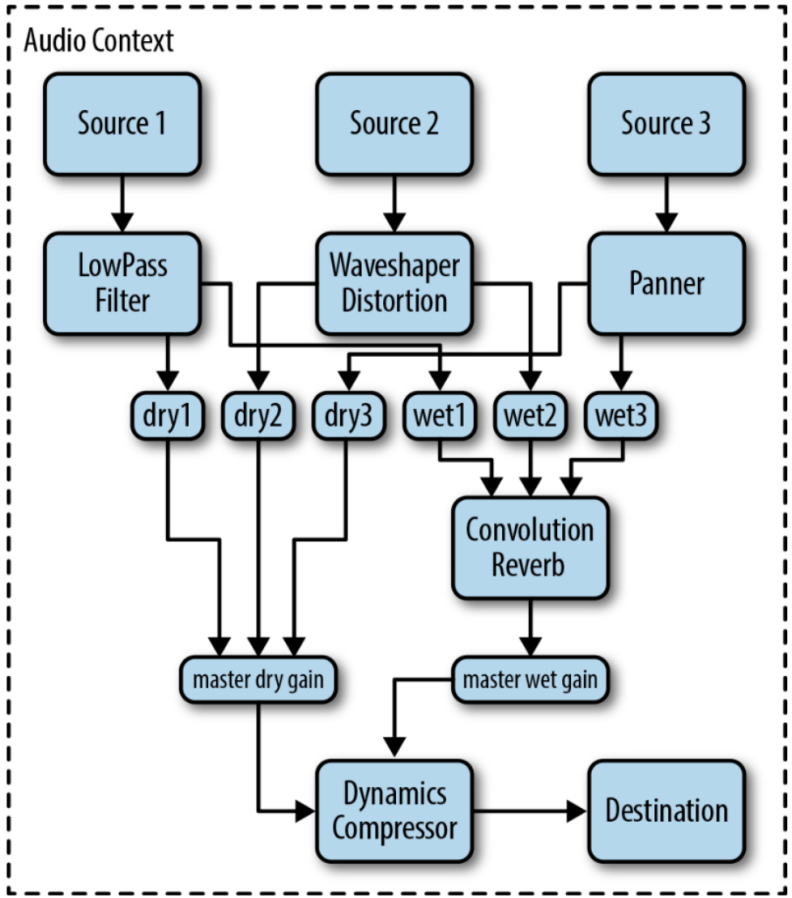
У Web Audio API стоит выделить 3 основных аспекта, которые особо не пересекаются и их можно рассматривать отдельно:

- возможности работы с различными источниками сигнала
- возможности по анализу сигнала
- возможности по цифровой обработке сигнала

В этой статье я постараюсь объяснить назначение каждого элемента. Она получается и так достаточно большой, так что большую часть информации об интерфейсах я опустил (она доступна по ссылкам на документацию), а основное внимание сосредоточил на вещах, которые не очевидны после прочтения документации.

Не про файлы □

Про сигналы 〰


Audio Context
Source 1
Source 2
Source 3
LowPass Filter
Waveshaper Distortion
Panner
dry1
dry2
dry3
wet1
wet2
wet3
Convolution Reverb
master dry gain
master wet gain
Dynamics Compressor
Destination

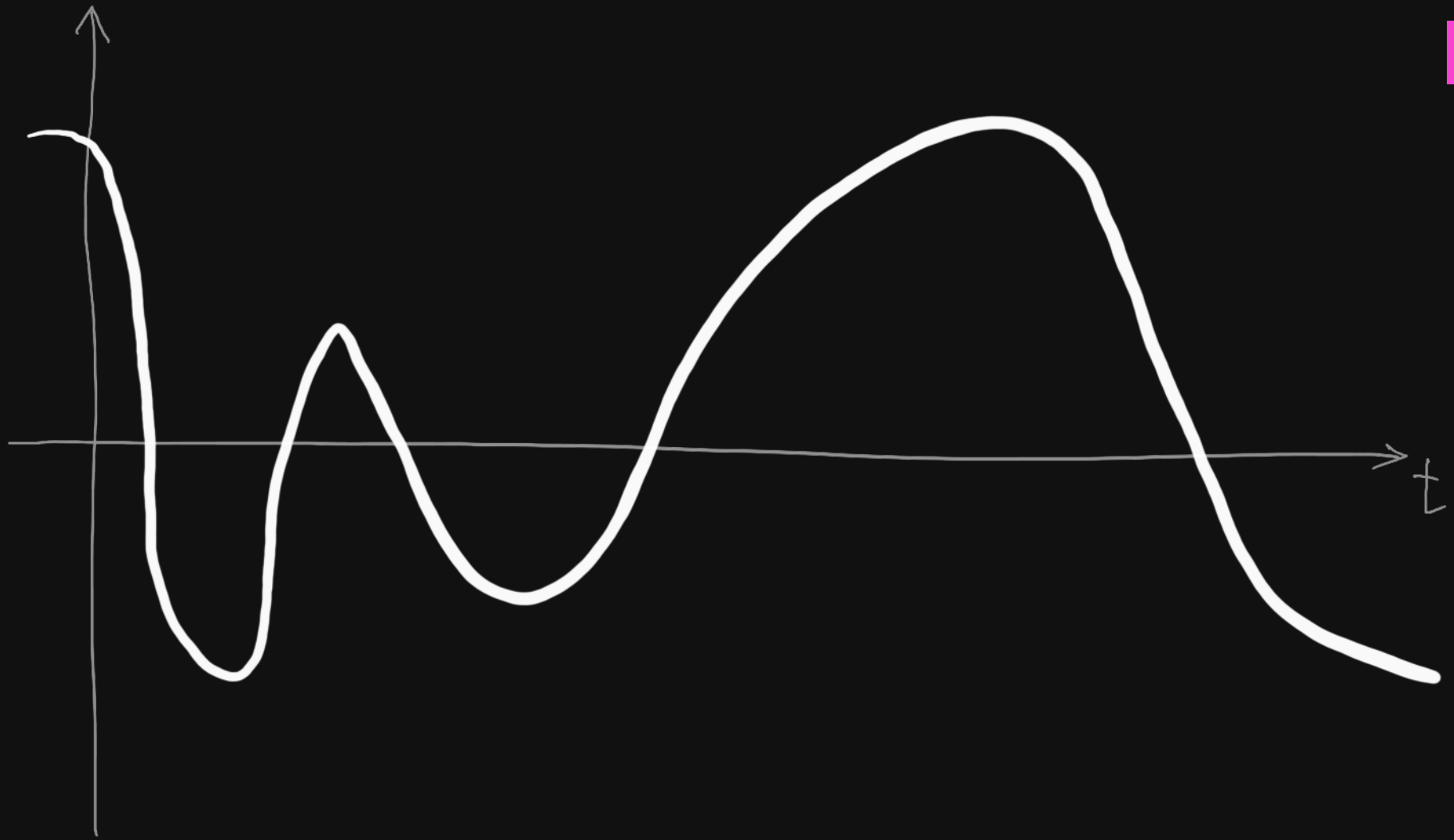
Доклад не про это 🤷

Демо 1.
Осцилограмма

```
const audioCtx = new (AudioContext || webkitAudioContext)();

const buffer = await audioCtx.decodeAudioData(file);

const source = audioCtx.createBufferSource();
source.buffer = buffer;

source.connect(audioCtx.destination);

source.start(0);
```

```
buffer: AudioBuffer
    duration: 91.53199546485261
    length: 4036561
    numberOfChannels: 2
```

```
buffer.getChannelData(0);

Float32Array(6506070) [0.0014954069629311562, 0.0019531846046447754,
0.0016785180196166039, 0.0018616290763020515, 0.0017700735479593277,
0.0017700735479593277, 0.0018005920574069023, 0.0017700735479593277,
0.0017700735479593277, 0.001831110566854477, 0.001831110566854477,
0.0019226660951972008, 0.0020142216235399246, 0.0020447401329874992,
0.0021057771518826485, 0.002227851189672947, 0.0023194067180156708,
0.0024719992652535544, 0.002533036284148693, 0.002624591812491417,
0.0027161473408341408, 0.0027161473408341408, 0.00277718435972929,
0.00277718435972929, 0.0028382213786244392, 0.00277718435972929,
0.0028077028691768646, 0.0028382213786244392, 0.0028382213786244392,
0.0028077028691768646, 0.00277718435972929, 0.0028382213786244392,
0.0028077028691768646, 0.002868739888072014, 0.0028077028691768646,
0.0028077028691768646, 0.0028077028691768646, 0.0028382213786244392, ...
```


t

```
const dotsCount = buffer.length;
const dotsInPixel = Math.floor(dotsCount / canvas.width);
const values = [];

for (let i = 0; i < canvas.width; i++) {
    let sum = 0;
    for (let j = i * dotsInPixel; j < (i + 1) * dotsInPixel; j++) {
        sum += Math.pow(buffer[j], 2);
    }
    values.push(Math.sqrt(sum));
}
const maxValue = Math.ceil(Math.max(...values));
```

```
const yCenter = Math.floor(h / 2);
const h = canvas.height;

for (let i = 0; i < canvas.width; i++) {
    const currentValue = Math.round(
        (values[i] / maxValue) * h * 0.8
    );
    canvas.fillRect(
        i,
        y + yCenter - currentValue / 2,
        1,
        currentValue
    );
}
```

Demo

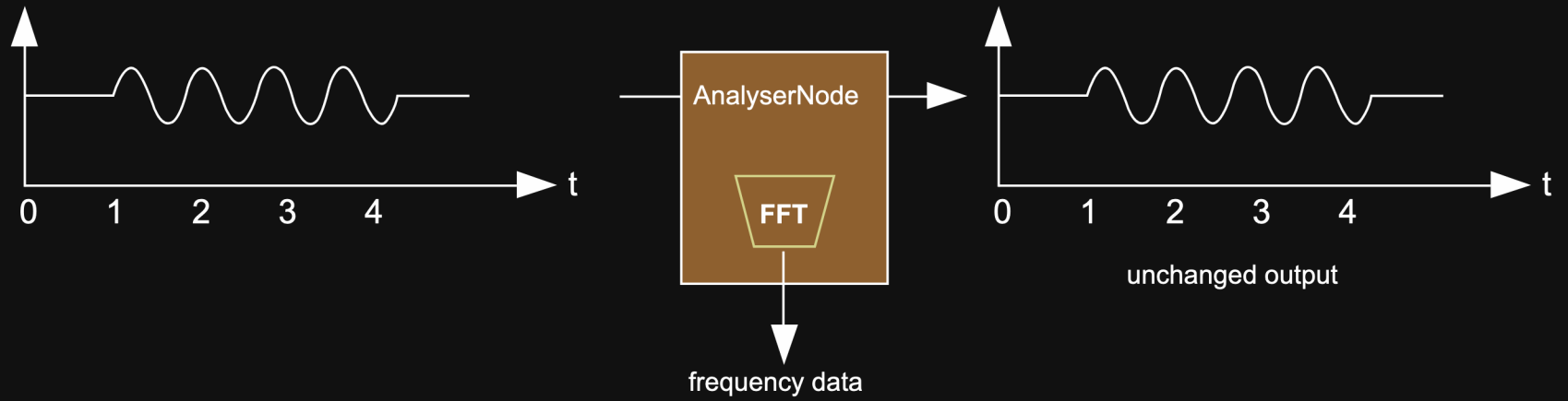
Демо 2.
Спектрограмма

```
const offlineCtx = new OfflineAudioContext(
    audioBuffer.numberOfChannels,
    audioBuffer.length,
    audioBuffer.sampleRate
);

const offlineSource = offlineCtx.createBufferSource();
offlineSource.buffer = audioBuffer;
offlineSource.channelCount = audioBuffer.numberOfChannels;
```

Не про файлы □

OfflineAudioContext 💡

```
const config = {
    fftResolution: 4096,
    smoothingTimeConstant: 0.02,
    processorBufferSize: 2048,
};

const analyzer = offlineCtx.createAnalyser();
analyzer.fftSize = config.fftResolution;
analyzer.smoothingTimeConstant = config.smoothingTimeConstant;
```


t
0
1
2
3
4
AnalyserNode
FFT
frequency data
t
0
1
2
3
4
unchanged output

FFT?

Быстрое преобразование Фурье

$$\mathrm{Re}\,X[k] = \sum_{i=0}^{N-1} x[i] \cdot \cos(2\pi k i / N)$$

$$\mathrm{Im}\,X[k] = -\sum_{i=0}^{N-1} x[i] \cdot \sin(2\pi k i / N)$$

$$A[X_k] = \sqrt{\mathrm{Re}X_k^2 + \mathrm{Im}X_k^2}$$

$$\varphi[X_k] = \tan^{-1}\frac{\mathrm{Im}\,X_k}{\mathrm{Re}X_k}$$

```
offlineSource.connect(analyzer);

offlineCtx.createScriptProcessor =
    offlineCtx.createScriptProcessor || offlineCtx.createJavaScriptNode;

const processor = offlineCtx.createScriptProcessor(
    config.processorBufferSize,
    1,
    1
);

const channelFFtDataBuffer = new Uint8Array(
    (audioBuffer.length / config.processorBufferSize) *
        (config.fftResolution / 2)
);
```

```
let offset = 0;
processor.onaudioprocess = (e) => {
    const freqData = new Uint8Array(
        channelFFtDataBuffer.buffer,
        offset,
        analyzer.frequencyBinCount
    );
    analyzer.getByteFrequencyData(freqData);
    offset += analyzer.frequencyBinCount;
};

offlineSource.connect(processor);
processor.connect(offlineCtx.destination);
offlineSource.start(0);
```

getByteFrequencyData

```
const stride = generalAnalyzer.frequencyBinCount;

for (let i = 0; i < w; i++) {
    const startIndex = i * ticksPerLine * stride;

    for (let j = 0; j < stride; j++) {
        const index = startIndex + j;
        const db = data.channel[index] / 255;
        const hColor = Math.round((db * 120 + 280) % 360);
        const lColor = 10 + 70 * db + '%';
        spectreCanvasCtx.fillStyle =
            `hsl(${hColor}, 100%, ${lColor})`;
        spectreCanvasCtx.fillRect(
            i * cellWidth, h - cellHeight * j,
            cellWidth, cellHeight
        );
    }
}
```

Demo

Демо 3.
Spectrodrawing

Спектрограмма

↓

Звук

‹ New in Audio Processing

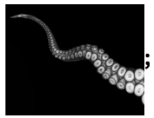
All Latest Features

Generate Sound from Image Using Inverse Spectrogram

Construct an audio signal from an image, assuming the image to be the power spectrogram of the original signal.

In[1]:=

```
image =
```



```
;
```

Use InverseSpectrogram to calculate the approximate inversion of the spectrogram operation. This assumes that the input image is the magnitude spectrogram and iteratively tries to reconstruct the phase.

In[2]:=

```
Audio[InverseSpectrogram[image], SampleRate → 8000]
```

Out[2]=

Потери ☹️

Ну и ладно 😄

```
const imgData = ctx.getImageData(0, 0, w, h).data;
const durationSeconds = 10;
const sampleRate = 44100;
const channelsCount = 1;
const samplesCount = Math.round(sampleRate * durationSeconds);
const maxPossibleFrequency = 20000; // Hz
const coeff = maxPossibleFrequency / canvas.height;
const samplesPerLineX = Math.floor(samplesCount / canvas.width);
const samples = [];

let maxFreq = -Infinity;
let yFactor = 2;
```

$$X[i] = \sum_{k=0}^{N/2} \frac{\mathrm{Re}\,X[k]}{N/2} \cos(2\pi k i / N) +$$

$$+ \sum_{k=0}^{N/2} - \frac{\mathrm{Im}\,X[k]}{N/2} \sin(2\pi k i / N)$$

```
for (let i = 0; i < samplesCount; i++) {
    const x = Math.floor(i / samplesPerLineX);
    let reX = 0;

    for (let y = 0; y < h; y += yFactor) {
        const j = (y * w + x) * 4;
        const sum = imgData[j] + imgData[j + 1] + imgData[j + 2];
        const volume = Math.pow((sum / (255 * 3)) * 100, 2);
        const freq = Math.round(coeff * (h - y + 1));
        reX += Math.floor(
            volume * Math.cos((freq * 2 * Math.PI * i) / sampleRate)
        );
    }

    samples.push(reX);
}
```

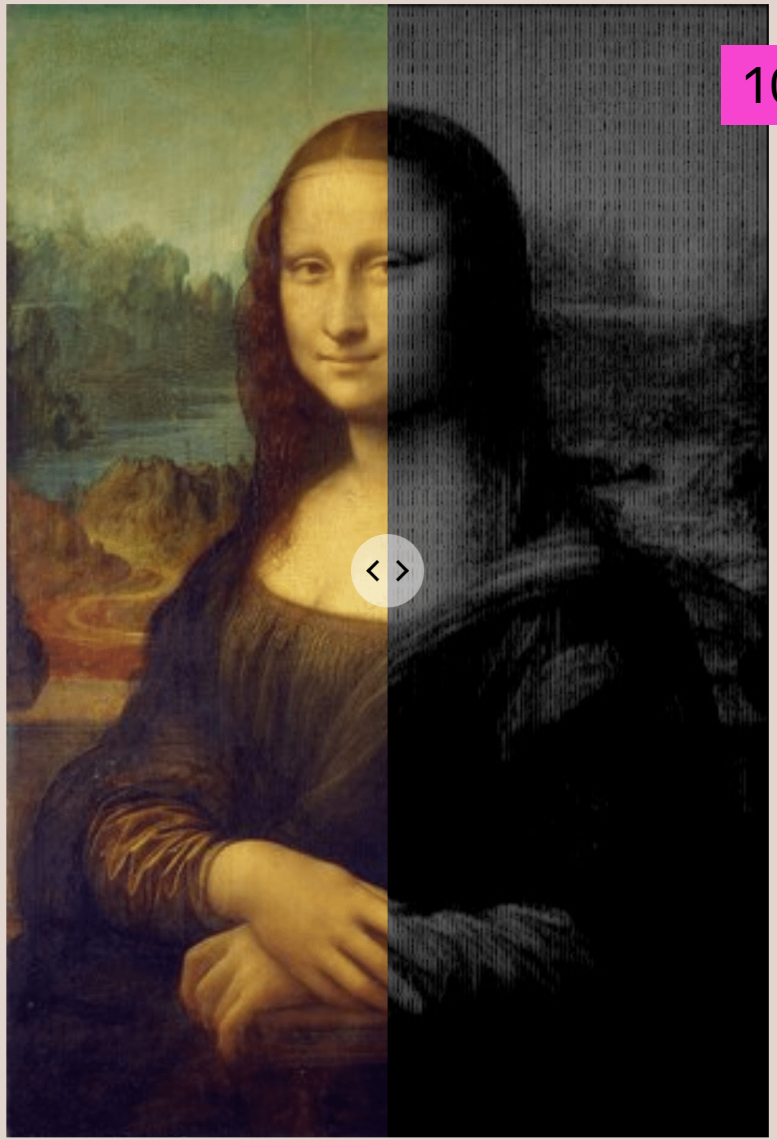
```
volume * Math.cos((freq * 2 * Math.PI * i) / sampleRate)
```

4 часа дебага спустя...

Demo

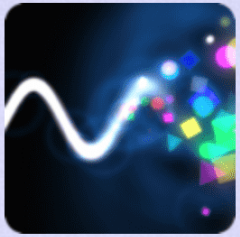
img-encode

Convert an image to sound spectrum



Virtual ANS

Последняя версия - 3.0.2c (12 апреля 2020)



Скачать:
Virtual ANS для Windows, Linux и macOS
Virtual ANS для iOS
Virtual ANS для Android

Список изменений

Вы можете сделать пожертвование, если хотите поддержать дальнейшее развитие Virtual ANS. Спасибо!

Donate

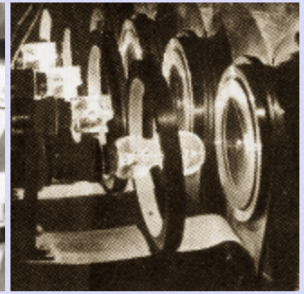
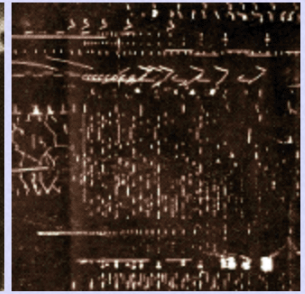
(если PayPal не подходит, попробуйте Яндекс.Деньги)

Описание
Инструкция
Скриншоты | **Видео**
Форум | **VK** | **FB**
Старые версии

Что такое Virtual ANS

Virtual ANS - симулятор легендарного фотоэлектронного синтезатора АНС, который впервые позволил рисовать музыку в виде спектрограммы, без участия живых инструментов и исполнителей. АНС был создан советским изобретателем Евгением Мурзиным в период с 1938 по 1958г.
Вы можете услышать звуки синтезатора в некоторых композициях Альфреда Шнитке, Станислава Крейчи, Эдуарда Артемьева, в фильмах Андрея Тарковского "Солярис", "Зеркало", "Сталкер", или, например, в сцене ночного кошмара из комедии Леонида Гайдая "Бриллиантовая рука".



Зачем? 😶

Визуальный поиск паттернов 👀

Распознавание речи по картинке

Стеганограмма 

Подпись файлов ✍️

Демо 4.
Музыка в картинку

амплитуда

число

картинка

пиксели

пиксель

R G B

R G B

числа

числа

числа


КАРТИНКИ — ЧИСЛА
ЗВУКИ — ЧИСЛА
ТЫ ЖЕ НЕ СТАНЕШЬ
СОХРАНЯТЬ ПЕСНИ В PNG?
imgflip.com

```
const buffers = [];
const buffersCount = source.buffer.numberOfChannels;

for (let i = 0; i < buffersCount; i++) {
    buffers.push(source.buffer.getChannelData(i));
}

const size = Math.ceil(Math.sqrt(buffers[0].length / 3));
const cellSize = 1;
const h = (canvas.height = size * cellSize * buffersCount);
const w = (canvas.width = size * cellSize);
```

```
for (let y = 0; y < size; y++) {
    for (let x = 0; x < size; x++) {
        const index = (y * size + x) * 3;
        const r = Math.floor((255 * (buffer[index] + 1)) / 2);
        const g = Math.floor((255 * (buffer[index + 1] + 1)) / 2);
        const b = Math.floor((255 * (buffer[index + 2] + 1)) / 2);
        const fillStyle = `rgb(${r}, ${g}, ${b})`;
        canvasCtx.fillStyle = fillStyle;
        canvasCtx.fillRect(
            x * cellSize,
            y * cellSize + i * w,
            cellSize,
            cellSize
        );
    }
}
```

Demo

Демо 5.
Картинка в музыку

```
const data = ctx.getImageData(0, 0, w, h).data;

const buffersCount = Math.round(h / w);
const bufferLength = (data.length * 3) / 4 / buffersCount;
const audioBuffer = new AudioBuffer({
    length: bufferLength,
    sampleRate: 44100,
    numberOfChannels: buffersCount,
});
```

```
for (let j = 0; j < buffersCount; j++) {
    const pixels = [];
    const dotsCount = data.length / 4 / buffersCount;
    for (let i = 0; i < dotsCount; i++) {
        const index = j * dotsCount * 4 + i * 4;
        const r = (data[index] * 2) / 255 - 1;
        const g = (data[index + 1] * 2) / 255 - 1;
        const b = (data[index + 2] * 2) / 255 - 1;
        pixels.push(r);
        pixels.push(g);
        pixels.push(b);
    }
    const buffer = Float32Array.from(pixels);
    audioBuffer.copyToChannel(buffer, j, 0);
}
```

```
const objectUrl = URL.createObjectURL(
    bufferToWave(abuffer, total_samples)
);

audio.src = objectUrl;
audio.title = filename + '.wav';

function bufferToWave(abuffer, len) {
    // ...
}
```

Demo

Squoosh 😈

Физика —
это интересно 🙃

Have Fun!

mefody.dev/talks/img-with-sound/

@dark_mefody

n.a.dubko@gmail.com