Weird Web APIs I love to use when making art in the browser

By Nino Filiu

github.com/ninofiliu/talks

About me

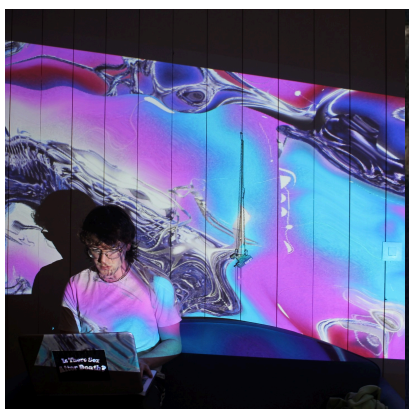
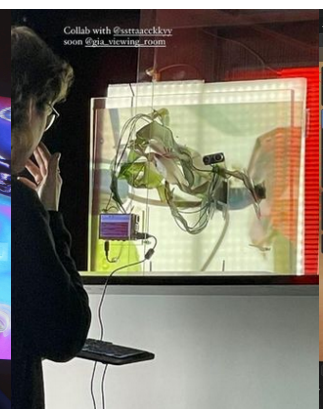
I make video games and interactive art with computers and electronics + many other things

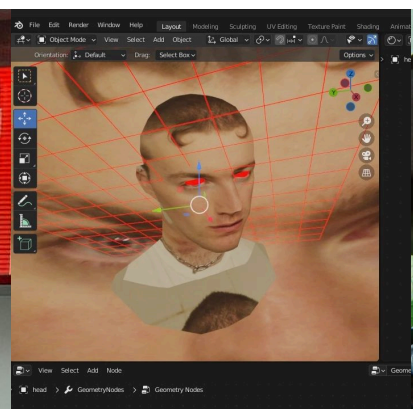
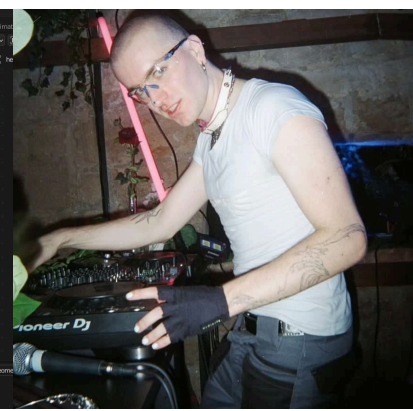
👱 26yo

🏳️‍🌈 he/they

🎓 Telecom Paris + Technische Universität Berlin

🔗 ninofiliu.com


Short company promo

We’re hiring!

ℹ️ Toucan Toco

🚀 Dataviz

👷‍♂️ Healthy and modern codebase

💚 Safe environment

💬 Welcome To The Jungle

Canvas API

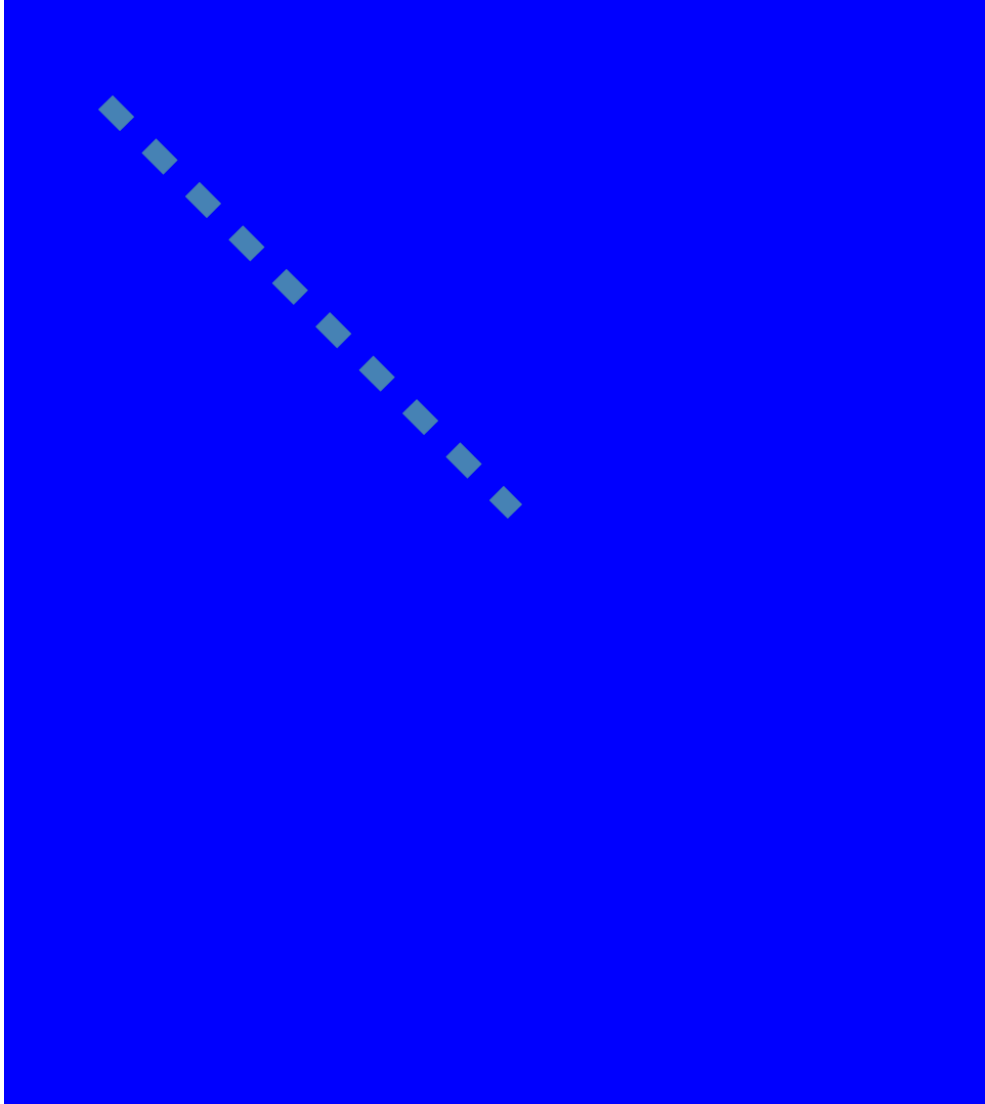
Drawing a dashed line on a blue background

```
<canvas></canvas>
```

```
// create a canvas and a 2D context
const canvas = document.querySelector("canvas")!;
canvas.width = 300;
canvas.height = 200;
const ctx = canvas.getContext("2d")!;

// fill the whole canvas blue
ctx.fillStyle = "blue";
ctx.fillRect(0, 0, 300, 200);

// draws a dashed line
ctx.strokeStyle = "steelblue";
ctx.lineWidth = 10;
ctx.setLineDash([20, 20]);
ctx.moveTo(50, 50);
ctx.lineTo(250, 150);
ctx.stroke();
```



Canvas API

More fun canvas capabilities

Supports text

```
ctx.font = "50px Roboto";
ctx.fillText("Hello World", 0, 0);
```

Painting images

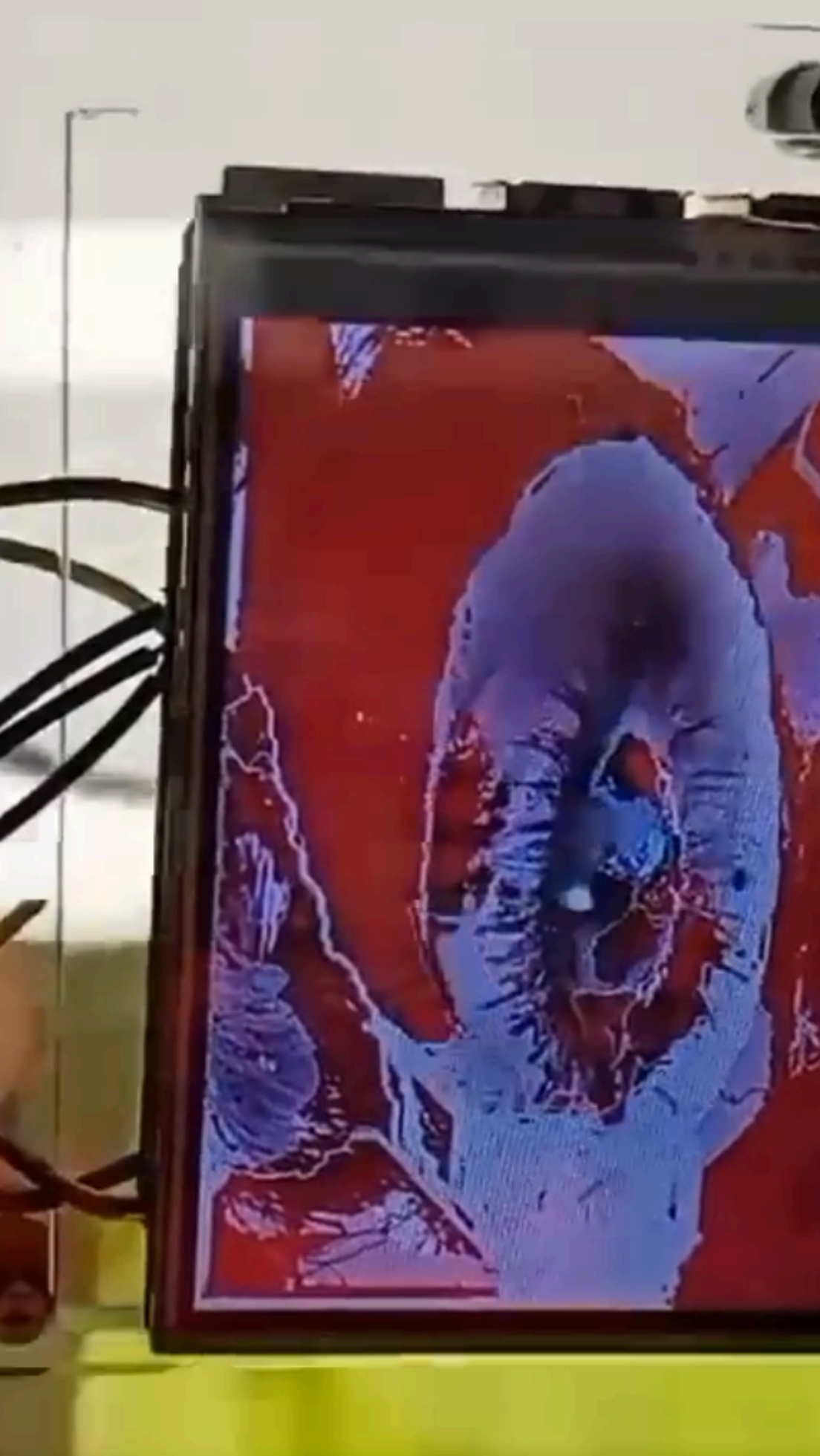
```
ctx.drawImage(document.querySelector("img"), 0, 0);
```

Read pixel data

```
const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
imageData.data[2] = 255;
ctx.putImageData(imageData, 0, 0);
```

Encode image

```
canvas.toDataURL("image/jpg", 0.9); //  ...
```



WebUSB

Allows navigators to communicate with USB devices

- poor browsing support
- OS-dependent permissions
- "raw" USB protocol

Groundbreaking in some industries, but it was such a hassle to use it IMO.

If you can use another API, do it

Gamepad API

Allows navigators to communicate with gamepads

WebMIDI

Allows navigators to communicate with MIDI devices

MIDI: Musical Instrument Digital Interface

Simple protocol:

- input and output channels
- messages have three bytes
- specs defines their meaning: `[144, 60, 127]` = note on, middle C, full velocity

WebMIDI debugger

From personal XP, it's usually easier to play around with the debugger and understand the patterns rather than reading the docs

Only one Chrome for now

WebMIDI

Log all messages on all inputs

```
const access = await navigator.requestMIDIAccess();
for (const input of access.inputs) {
  input.addEventListener("midimessage", (evt) => {
    const [a, b, c] = evt.data;
  });
}
```

WebMIDI

Play a note on all outputs

```
const access = await navigator.requestMIDIAccess();
for (const output of access.outputs) {
  output.send([144, 60, 127]);
}
```

WebMIDI

Sacrifice Demo

WebGL

What are they?

GPUs: more cores than CPUs, but can only run a few programs over all of them

Useful for graphics programming (same programs for many pixels)

OpenGL: driver for GPU

GLSL: OpenGL shading language

WebGL: run OpenGL in a `<canvas>`

WebGPU: successor to WebGL, can ask the GPU to execute arbitrary tasks, not just graphics - also machine learning and fluid simulations

WebGL

Draw one triangle: part 1

```
const vertexShaderSource = `#version 300 es
in vec4 a_position;
void main() {
  gl_Position = a_position;
}
`;

const fragmentShaderSource = `#version 300 es
precision highp float;
out vec4 outColor;
void main() {
  outColor = vec4(1, 0, 0.5, 1);
}
`;
```

WebGL

Draw one triangle: part 2

```
const x = <T>(x: T | null | undefined): T => {
  if (x == null) throw new Error("should not be nullish");
  return x;
};

const createShader = (
  gl: WebGL2RenderingContext,
  type: number,
  source: string
) => {
  const shader = x(gl.createShader(type));
  gl.shaderSource(shader, source);
  gl.compileShader(shader);
  if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS))
    throw new Error(`${gl.getShaderInfoLog(shader)}`);
  return shader;
};
```

WebGL

Draw one triangle: part 3

```
const createProgram = (
  gl: WebGL2RenderingContext,
  vertexShader: WebGLShader,
  fragmentShader: WebGLShader
) => {
  const program = x(gl.createProgram());
  gl.attachShader(program, vertexShader);
  gl.attachShader(program, fragmentShader);
  gl.linkProgram(program);
  if (!gl.getProgramParameter(program, gl.LINK_STATUS))
    throw new Error(`${gl.getProgramInfoLog(program)}`);
  return program;
};
const canvas = document.createElement("canvas");
const gl = x(canvas.getContext("webgl2"));
const vertexShader = createShader(gl, gl.VERTEX_SHADER, vertexShaderSource);
const fragmentShader = createShader(
  gl,
  gl.FRAGMENT_SHADER,
  fragmentShaderSource
);
const program = createProgram(gl, vertexShader, fragmentShader);
```

WebGL

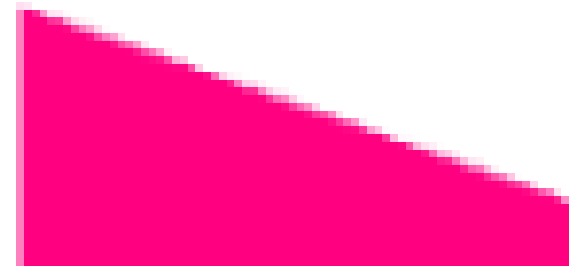
Draw one triangle: part 4

```
const locations = {
  a_position:,
};
gl.bindBuffer(gl.ARRAY_BUFFER, gl.createBuffer());
gl.bufferData(
  gl.ARRAY_BUFFER,
  new Float32Array([0, 0, 0, 0.5, 0.7, 0]),
  gl.STATIC_DRAW
);
const vao = gl.createVertexArray();
gl.bindVertexArray(vao);
gl.enableVertexAttribArray(locations.a_position);
gl.vertexAttribPointer(locations.a_position, 2, gl.FLOAT, false, 0, 0);
gl.viewport(0, 0, gl.canvas.width, gl.canvas.height);
gl.clearColor(0, 0, 0, 0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.useProgram(program);
gl.bindVertexArray(vao);

gl.drawArrays(gl.TRIANGLES, 0, 3);
```

WebGL

Output



WebGL

Made with WebGL

- ThreeJS: de-facto standard lib for 3D stuff on the web
- React Three Fiber: write ThreeJS stuff in JSX
- Spline: 3D software that runs inside the browser
- Womp: same, but focused on smooth shapes
- Cannon: 3D physics library
- Shadertoy: shader social network
- TensorflowJS: machine learning in the browser, including live pose detection & co

WebGL + TensorflowJS

Detect faces in webcam stream

```
import "@mediapipe/face_detection";
import "@tensorflow/tfjs-core";
import "@tensorflow/tfjs-backend-webgl";
import * as faceDetection from "@tensorflow-models/face-detection";

const stream = await navigator.mediaDevices.getUserMedia({ video: true });
const video = document.createElemnt("video");
video.srcObject = stream;
await video.play();
const faceDetector = await faceDetection.createDetector(
  faceDetection.SupportedModels.MediaPipeFaceDetector,
  {
    runtime: "mediapipe",
    solutionPath: "/pkgs/@mediapipe/face_detection",
  }
);

const loop = () => {
  const faces = await faceDetector.estimateFaces(video);
  requestAnimationFrame(loop);
};
loop();
```

WebGL + TensorflowJS

Used "in production"

First Contact

Web Audio

Fine-grained sound manipulation in the browser if you need more that `<audio>` 's `.play()` and `.pause()`

- samples (eg: `song.mp3` , `kick.wav`)
- audio generation (sine wave, triangle wav, custom...)
- amplitude modulation
- frequency modulation
- audio effects (filters, reverb, compression...)
- analysis node (Fourier transform)
- node graph of all these

Example: Simple microphone spectrogram

Web Audio

meh/10

Promising at first, but personally had more fun with

- TouchDesigner
- Ableton
- Vital

Web Codecs

Gives low-level access to individual frames of a video

I really like this API, here’s why

Web Codecs

How does .mp4 lossy-compresses stuff?

Encoding all frames as images would take too much space on disk, so we do this instead:

1. Exceptionally encode first frame as image
2. Encode "block movement" since last frame
3. Repeat as many times as possible

Movements are spatially uniform so output look ok

Only one movement per block is encoded, instead of RGBA for each pixel of the block, so we save space

Web Codecs

Datamosh

Playing these frames in an order they’re not supposed to be played at yields visually interesting results

But the usual workflow is pretty hardcore technically

That’s why I released Supermosh

Thanks!

:)

ninofiliu.com

github.com/ninofiliu/talks