on DRAWING

Browser as as an interactive canvas frame

03 19 2025

Clara + Ritu

Pixels

→ Most basic, the smallest unit of a digital image or display

Pixels

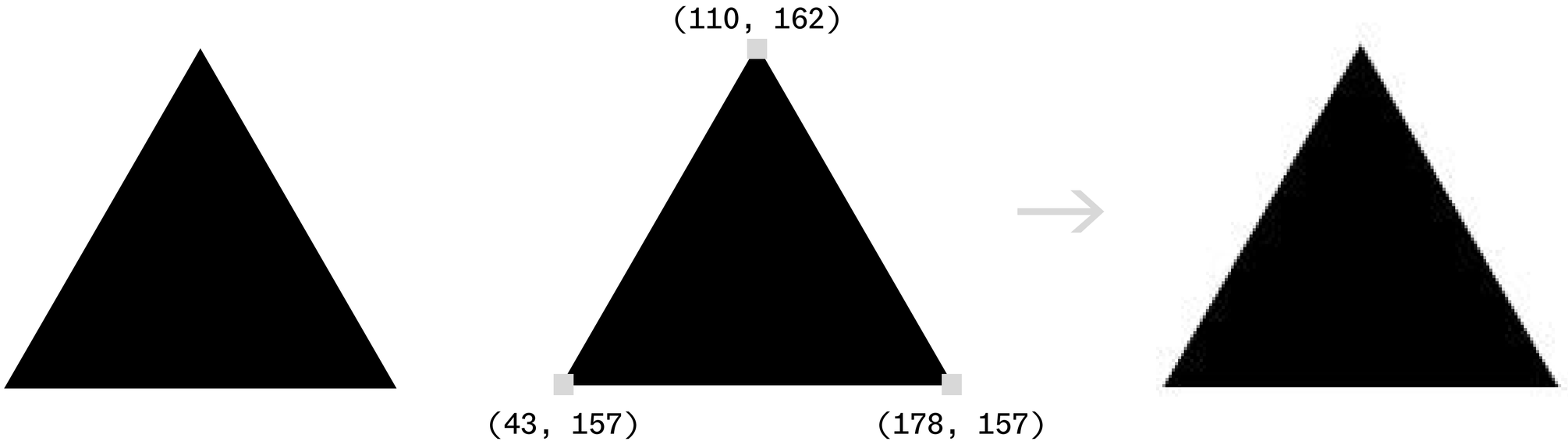
→ pixel as a single point of color in a grid of millions, much like a tiny tile in a grand mosaic

Rasterization

→ transforms geometric data into pixels that are displayed usually on a screen

Clara + Ritu

Rasterization



Clara + Ritu

Rasterization

→ vector files can be saved/exported as raster files, but raster files cannot be turned into vector files

Rasterization

Pro-tip: You should keep a vector copy at all times, even if you end up using the raster file for specific applications

Vectors

images created using mathematical formulas that define shapes, lines, and curves, allowing them to be scaled infinitely without losing quality, unlike raster images which are made of pixels.

Vectors

→ Looks perfect, though in reality it isn’t either

Rafaël Rozendaal

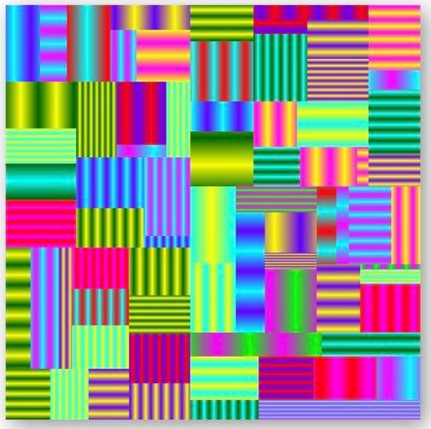
RR

Internet

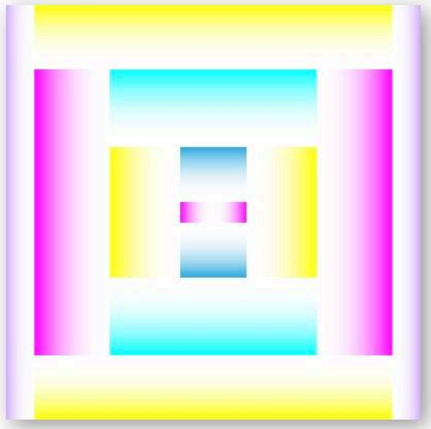
Exhibitions

Texts

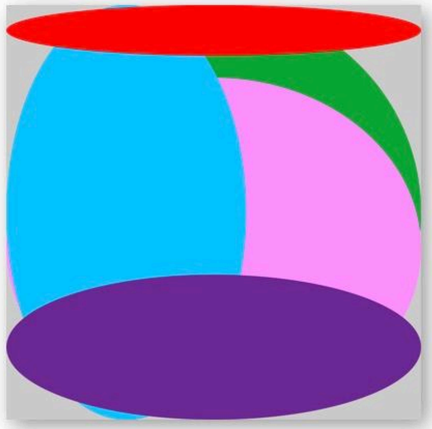
Info



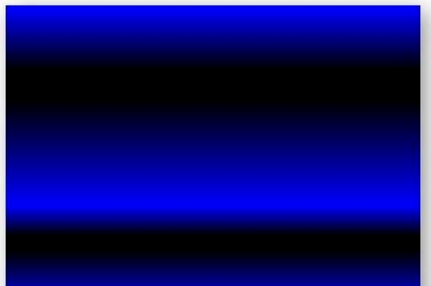
Endless Nameless

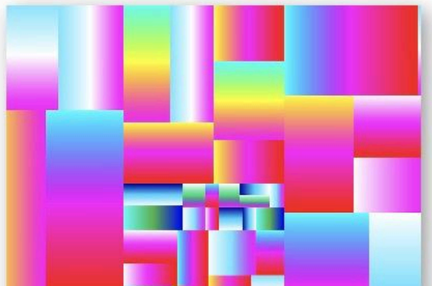


In Bloom



Open

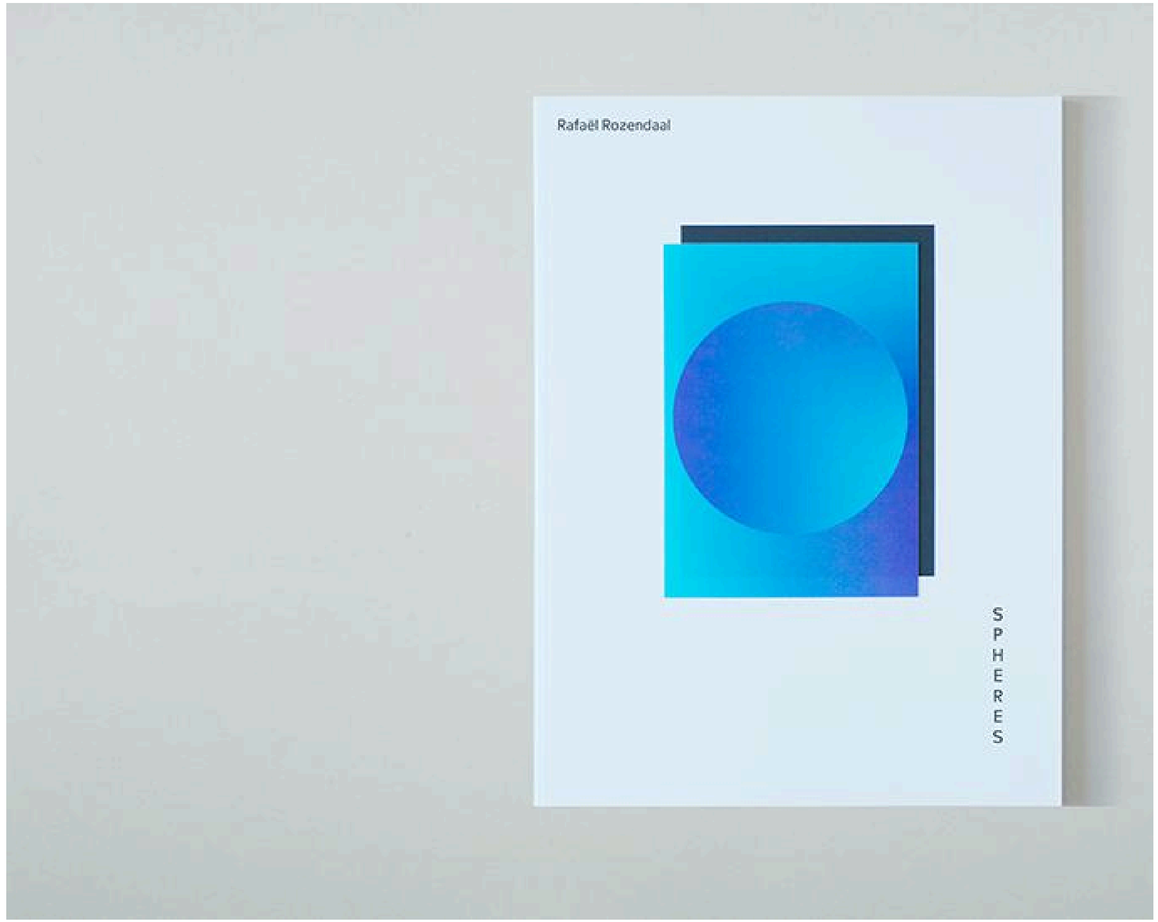




on DRAWING

Clara + Ritu

Vectors



‘Vectors are honest about the fact that they are computer imagery. It is clear that they are made on a computer, they’re not trying to be real’

Rafaël Rozendaal

Compression by Abstraction: A Conversation About Vectors (Reading)

on DRAWING

Clara + Ritu

Understanding Rasterization in 2D and 3D Graphics

From “Chapter 12: From Geometry to Pixels. Interactive Computer Graphics. 8th edition” by Angel & Shreiner (2020, p.299)

Clara + Ritu

Frame Rate and Animation (Documentation)

Web as Medium

on DRAWING

Clara + Ritu

webGL

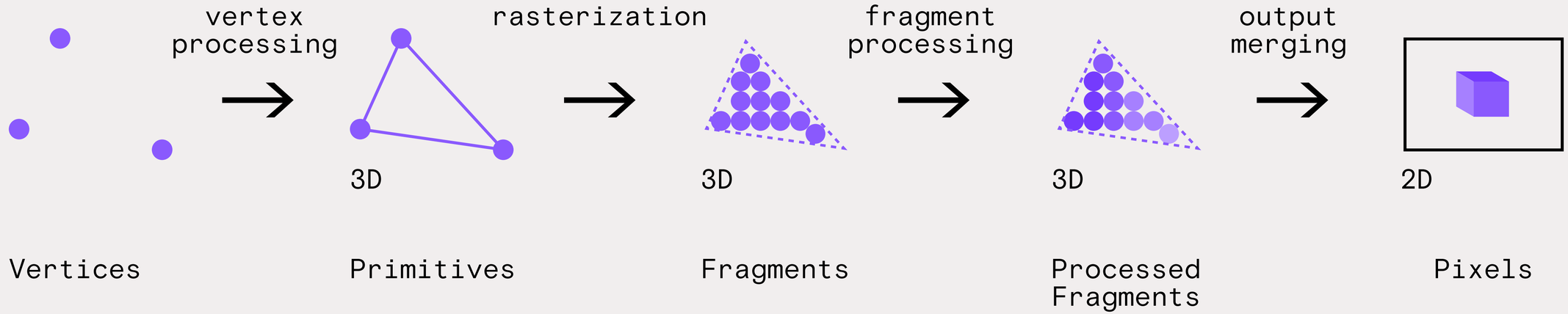
→ a javascript API for implementing interactive 2D and 3D vector graphics in the browser
→ runs graphics using GPU directly inside HTML canvas element, without the use of an external plug-in

Clara + Ritu

Shader Programs

→ Vertex Shader
graphics processing of mathematical operations on the objects' vertex data
→ Fragment Shader
stores the color values of every pixel in each fragment

Rendering Pipeline



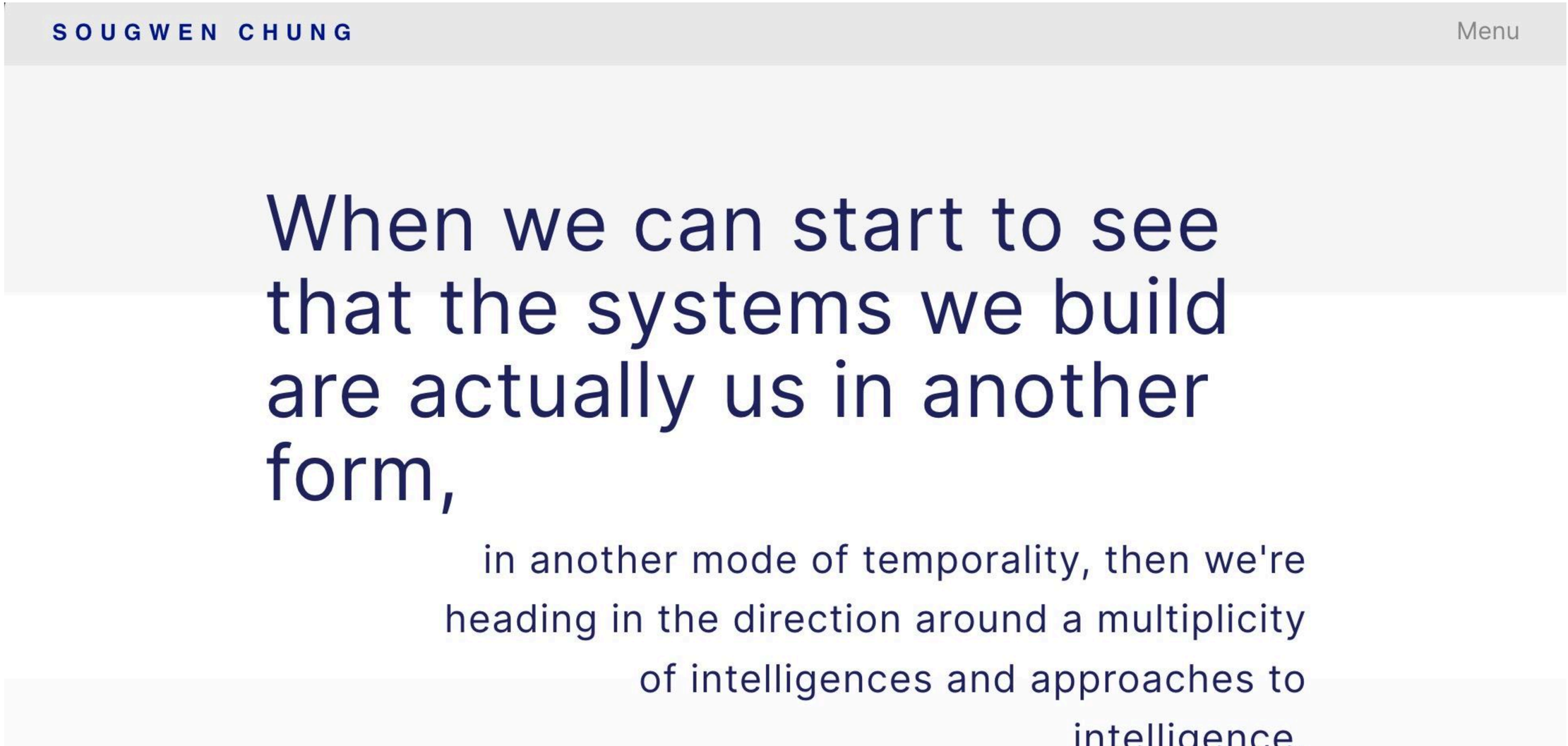
Zach Lieberman (Website)

‘As an artist I'm constantly thinking about new types of drawing tools, and what does drawing in the 21st century look like -- ink space is research in that realm.’

Ink Space App (Demo) (Github)

Clara + Ritu

Sougwen Chung (Website)



Sougwen Chung

GENESIS, 2024 (Work)

Web as Medium

on DRAWING

Clara + Ritu

Sougwen Chung

GENESIS, 2024 (Work)

Web as Medium

on DRAWING

Clara + Ritu

Casey Reas (website)

American artist of conceptual, procedural and minimal artworks, exploring ideas through the contemporary lens of software

Casey Reas (website)

Web as Medium

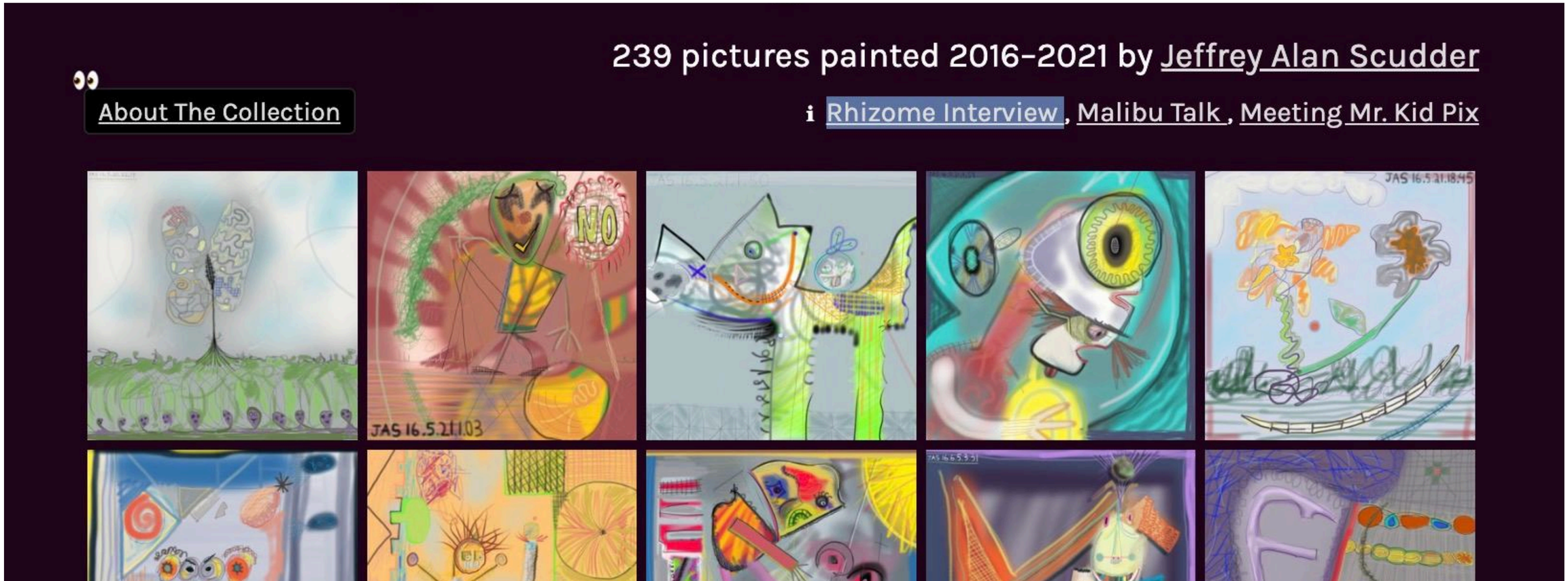
on DRAWING

Clara + Ritu

Jeffrey Scudder (website) (Rhizome)

‘In my digital paintings, I like to define some spatial and durational limits before drafting. A digital painting is a record of interactions, stored in a visual frame.’

Jeffrey Scudder (website)
Radical Digital Painting



p5.js

a short demo on using p5.js to create a drawing tool

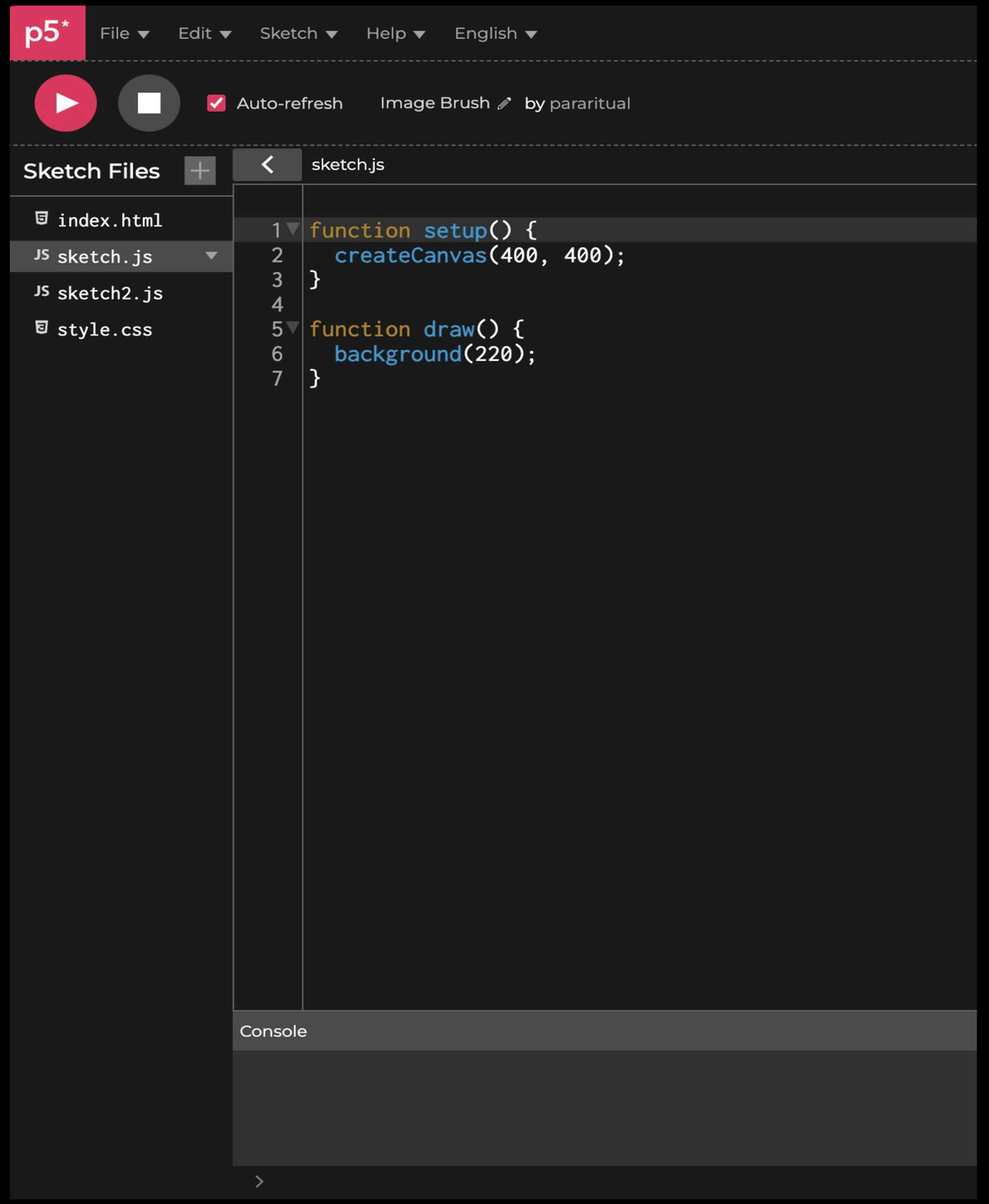
Web as Medium

on DRAWING

Clara + Ritu

→ open the p5.js editor

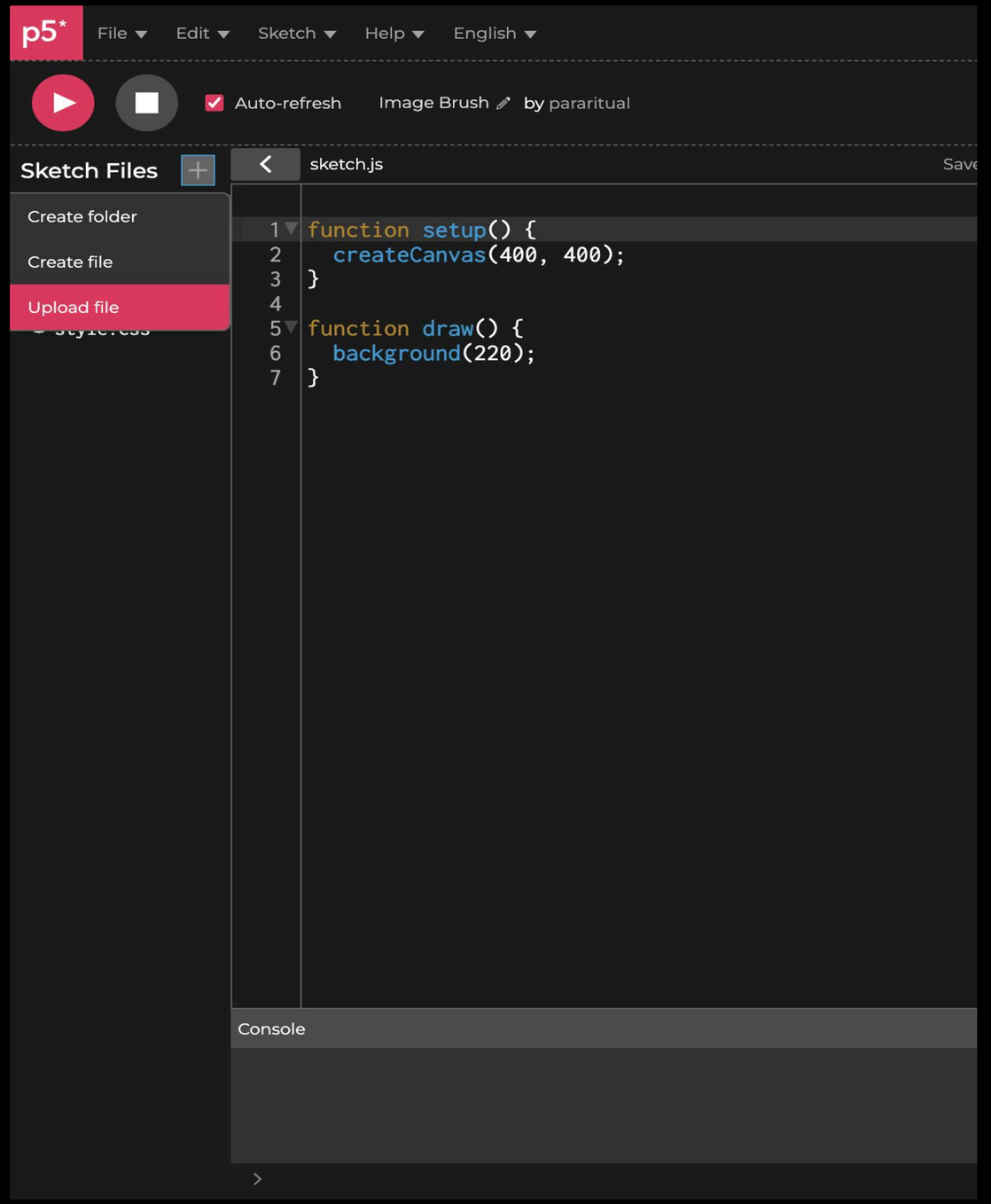
https://editor.p5js.org/



Clara + Ritu

→ upload minimum 3 images

Preferably same sizes or you can download example images here



Web as Medium

on DRAWING

Clara + Ritu

→ now let’s set the parameters

```
let imgA, imgB, imgC;
let images = [];
let k = 0;
let brushSizeSlider;
let clearButton;


function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```


Clara + Ritu

→ let’s preload our images

```
let imgA, imgB, imgC;
let images = [];
let k = 0;
let brushSizeSlider;
let clearButton;


function preload() {
  imgA = loadImage('a.png');
  imgB = loadImage('b.png');
  imgC = loadImage('c.png');
}


function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```


Clara + Ritu

→ setup your tool

add background, store images in an array, create a slider, create a button

```
function setup() {
  createCanvas(400, 400);
  background(255);
  images = [imgA, imgB, imgC];

  brushSizeSlider = createSlider(10, 100, 30);

  clearButton = createButton('Clear Canvas');

clearButton.mousePressed(clearCanvas);
}

function draw() {
  background(220);
}
```

Web as Medium

on DRAWING

Clara + Ritu

→ write draw functions

```
function draw() {
  background(220)
  if (mouseIsPressed) {
    let size =
brushSizeSlider.value();
    let nextImage = images[k];

    image(nextImage, mouseX, mouseY,
size, size);
  }
}
```


Clara + Ritu

→ create a mouseClicked function

```
function draw() {
  if (mouseIsPressed) {
    let size =
brushSizeSlider.value();
    let nextImage = images[k];

    image(nextImage, mouseX, mouseY,
size, size);
  }
}

function mouseClicked() {
  k = (k + 1) % images.length;
}
```


Clara + Ritu

→ define clearCanvas Function

```
function draw() {
  if (mouseIsPressed) {
    let size =
brushSizeSlider.value();
    let nextImage = images[k];

    image(nextImage, mouseX, mouseY,
size, size);
  }
}

function mouseClicked() {
  k = (k + 1) % images.length;
}

function clearCanvas() {
  background(255);
}
```


Clara + Ritu

→ now we have our own image brush!

Link for the demo code