Mediarts 206

DATA VIS +

P5JS +

WEATHER

WEATHER LINES

GRANT PEDERSEN & MASHA NIKISHINA

How do we visualize what is around us?

By fusing art with real-time data, data offers an ever-changing reflection of our natural environment. For example, how can we create a piece that fluidly adapts to wind speed and temperature variations, transforming the data constantly around us into an artwork that is constantly updating and reflecting the world.

This was the inspiration behind "Weather Lines." Our artwork attempted to incorporate OpenWeather's data API with noise particle functions to create a piece that mimics air itself.

Concept Statement

In the age of data visualization, art and science converge to create interactive experiences that engage, inform, and inspire. Weather Lines, set against the backdrop of environmental awareness, seeks to bridge meteorological data with visual aesthetics. By leveraging real-time weather data from any city, we have created a dynamic sketch that provides not just a quantitative understanding of the city's climate but also an intuitive, visual grasp of its ambiance. Using temperature and wind speed as primary inputs, the sketch translates these into a choreography of moving particles on a canvas, where each particle's motion and color embody the nuances of the weather.

We have two motivations behind this project. Firstly, in an era overloaded with information, there's a profound need for representations that simplify without undermining. Raw weather data, while crucial, can be mundane and inaccessible to many. Translating this data into a visual form can amplify its reach and impact. Secondly, the project underscores the potential of generative art in the realm of data representation. By automating artistic decisions based on data, we can craft visualizations that are both objective in their foundation and subjective in their interpretation.

Our Main Question

Can real-time weather data, when transformed into a visual medium using generative art principles, provide a more engaging and intuitive understanding of a city's climate, compared to traditional data presentation methods?

Generative Steps & Math

#	Generative Steps
1.	Particle Initialization
2.	Temperature Mapping
3.	Wind Speed Mapping
4.	Noise-Driven Movement
5.	Canvas Interaction

1. Particle Initialization: Begin with 3000 particles, each initialized at a random position on the canvas, with a random direction of motion.
2. Temperature Mapping: Map the real-time temperature of a city to a color gradient between blue (cold) and red (warm).
3. Wind Speed Mapping: Adjust the speed of the particles based on real-time wind speed data. Faster wind results in quicker particle movement.
4. Noise-Driven Movement: Utilize Perlin noise to dictate the direction of particle movement, ensuring fluidity and unpredictability.
5. Canvas Interaction: Should a particle drift beyond the canvas's boundaries, reinitialize its position to a random location within the canvas.

Math

These equations form the mathematical backbone of the project, translating raw weather data into a visually engaging experience. They link temperature to the vibrancy of particle movement, wind speed to particle speed, and numerical temperature values to an intuitive color spectrum. This mathematical transformation merges data with aesthetics, enhancing our understanding of climate through art and creating a dynamic, informative visualization.

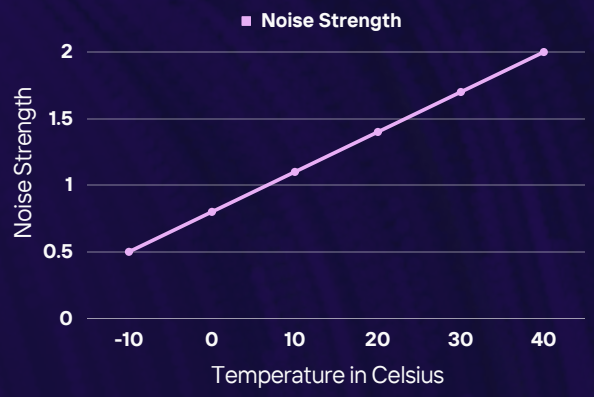
Math continues next page.

Math cont.

Temperature-based Noise Strength: The code maps the temperature (in Celsius) to a noiseStrength value between 0.5 and 2.
Equation:

noiseStrength = map(temp, -10, 40, 0.5, 2)

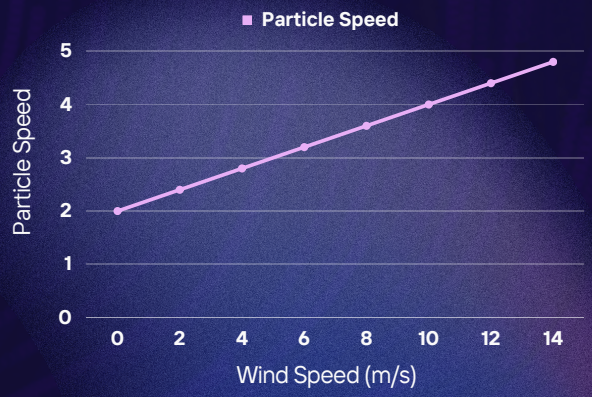
Graph:



Wind Speed-based Particle Speed: The code maps wind speed to the particle's speed value between 2 and 5.
Equation:

particleSpeed = map(windSpeed, 0, 15, 2, 5)

Graph:



Temperature-based Color Interpolation: The color of the particles changes between cold (blue) and warm (red) based on the current temperature.
Equation:

currentColor=lerpColor(coldColor,warmColor,map(temp,−10,40,0,1))

Representation:

-10 to 40 Celsius

Particle Movement using Noise: The particles move based on the Perlin noise function, which gives them a natural, fluid-like movement.
Equation:

$$\text{angle} = \text{noise}(x) * 2\pi * \text{noiseStrength}$$

Where $x =$

$$\left(\frac{\text{loc.x}}{\text{noiseScale}}, \frac{\text{loc.y}}{\text{noiseScale}}, \frac{\text{frameCount}}{\text{noiseScale}} \right)$$

What does the math show?

Illustrates how weather data, specifically temperature and wind speed, can be transformed into visual elements that convey meaningful information to viewers

Code Explained

Variables & Arrays

```
let weatherData;
let num = 3000;
var particles_a = [];
var particles_b = [];
var particles_c = [];
var fade = 100;
var radius = 3;

let noiseScale = 300;
let noiseStrength = 1.2;
```

- weatherData: A variable to store the fetched weather data.
- num: The number of particles (set to 3000).
- particles a, particles b, particles_c: Arrays to store instances of Particle objects.
- fade: The alpha value (transparency) of particles.
- radius: The size of particles.
- noiseScale and noiseStrength: Values to tweak the movement of particles.

Code Explained cont.

Setup()

```
function setup() {
 createCanvas(1200, 900);
 let city="Moscow"; //replace with other city name we used moscow to generate this
pages background :)
 let apiKey = [redacted]
 let url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}`;

 loadJSON(url, gotWeatherData,handleError);
 noStroke();
 for (let i=0; i<num; i++) {

 let loc_a = createVector(random(width*1.2), random(height), 2);
 let angle_a = random(TWO_PI);
 let dir_a = createVector(cos(angle_a), sin(angle_a));
 let loc_b = createVector(random(width*1.2), random(height), 2);
 let angle_b = random(TWO_PI);
 let dir_b = createVector(cos(angle_b), sin(angle_b));
 let loc_c = createVector(random(width*1.2), random(height), 2);
 let angle_c = random(TWO_PI);
 let dir_c = createVector(cos(angle_c), sin(angle_c));

 particles_a[i] = new Particle(loc_a, dir_a, 2);
 particles_b[i] = new Particle(loc_b, dir_b, 2);
 particles_c[i] = new Particle(loc_c, dir_c, 3);

 }
}
```

- The canvas of size 1200x900 pixels is created.
- A URL is formed to fetch weather data for a city using an API key.
- The loadJSON function fetches the weather data using the formed URL and passes it to gotWeatherData on success or handleError on error.
- A loop creates 3000 Particle objects for each particle array (particles_a, particles_b, particles_c). These particles have random positions and directions.

Code Explained cont.

gotWeatherData() & handleError()

```
function gotWeatherData(data) {
 weatherData = data;
}
function handleError(err) {
 console.error("There was an error
fetching the weather data:", err);
}
```

- gotWeatherData stores the fetched weather data in the weatherData variable.
- handleError logs any errors during the data fetch.

Code Explained cont.

draw()

```
function draw() {
 if (weatherData) {
 let temp = weatherData.main.temp - 273.15;
 noiseStrength = map(temp, -10, 40, 0.5, 2);
 let windSpeed = weatherData.wind.speed;
 for (let i = 0; i < num; i++) {
 particles_a[i].speed = map(windSpeed, 0, 15, 2, 5);
 }
 let coldColor = color(0, 0, 255);
 let warmColor = color(255, 0, 0);
 let currentColor = lerpColor(coldColor, warmColor, map(temp, -10, 40, 0, 1));
 fill(0, 5);
 noStroke();
 rect(0, 0, width, height);
 for (let i = 0; i < num; i++) {
 fill(currentColor.levels[0], currentColor.levels[1], currentColor.levels[2], fade);
 particles_a[i].move();
 particles_a[i].update(radius);
 particles_a[i].checkEdges();
 particles_b[i].move();
 particles_b[i].update(radius);
 particles_b[i].checkEdges();
 particles_c[i].move();
 particles_c[i].update(radius);
 particles_c[i].checkEdges();
 }
 }
}
```

- Continuously updates the canvas.
- If weatherData exists:
  - Temperature is converted from Kelvin to Celsius.
  - noiseStrength is adjusted based on temperature.
  - Particle speed (particles_a) is adjusted based on wind speed.
  - Color is determined by temperature: cold temperatures give a bluish color, and warm temperatures give a reddish color. Intermediate temperatures will result in a mix.
  - A black rectangle with slight opacity is drawn to create a fading effect.
  - All particles are moved, checked against canvas edges, and then drawn on the canvas.

Code Explained cont.

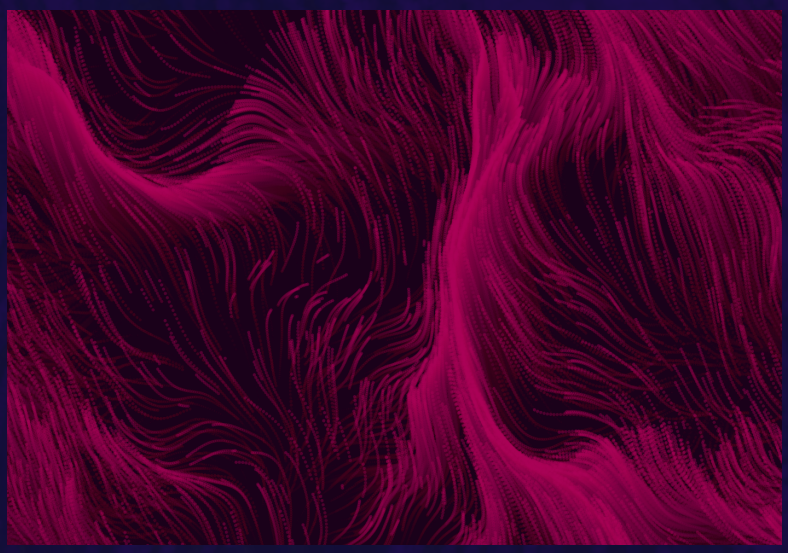
draw()

```
let Particle = function(loc_, dir_, speed_) {
this.loc = loc_;
this.dir = dir_;
this.speed = speed_;
this.d = 1;
};
Particle.prototype.run = function() {
this.move();
this.checkEdges();
this.update();
};
Particle.prototype.move = function(){
this.angle=noise(this.loc.x/noiseScale, this.loc.y/noiseScale,
frameCount/noiseScale)*TWO_PI*noiseStrength;
this.dir.x = cos(this.angle)+sin(this.angle)-sin(this.angle);
this.dir.y = sin(this.angle)-cos(this.angle)*sin(this.angle);
this.vel = this.dir.copy();
this.vel.mult(this.speed*this.d);
this.loc.add(this.vel);
};
Particle.prototype.checkEdges = function(){
if (this.loc.x < 0 || this.loc.x > width || this.loc.y < 0 || this.loc.y > height) {
this.loc.x = random(width*1.2);
this.loc.y = random(height);
}
};
Particle.prototype.update = function(r){
ellipse(this.loc.x, this.loc.y, r);
};
```

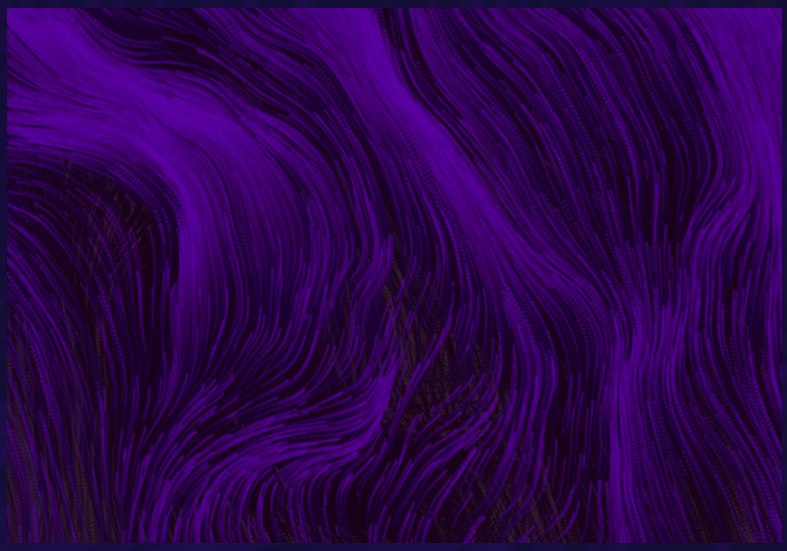
- Defined using a function constructor and prototypes.
- Represents a moving particle with properties such as location, direction, and speed.
  - run(): Calls move(), checkEdges(), and update().
  - move(): Adjusts the particle's direction based on noise and updates its position.
  - checkEdges(): Resets the particle's position if it goes beyond the canvas edges.
  - update(): Draws the particle as an ellipse on the canvas.

Results & Conclusion

Combining data and art provides a fresh way to understand information. In this project, we visualize city's changing weather with moving particles, colored by temperature and driven by wind. This approach simplifies data and shows the natural beauty in weather patterns. As we focus more on climate change and the environment, projects like this one help people connect with data and see it as a reflection of our world.



Atlanta, USA



Moscow, Russia