

# /algorithmic-art

## Source:

```
~/ .claude/skills/algorithmic-art/SKILL.md
```

---

name: algorithmic-art description: Creating algorithmic art using p5.js with seeded randomness and interactive parameter exploration. Use this when users request creating art using code, generative art, algorithmic art, flow fields, or particle systems. Create original algorithmic art rather than copying existing artists' work to avoid copyright violations. license: Complete terms in LICENSE.txt

Algorithmic philosophies are computational aesthetic movements that are then expressed through code. Output .md files (philosophy), .html files (interactive viewer), and .js files (generative algorithms).

This happens in two steps:

1. Algorithmic Philosophy Creation (.md file)
2. Express by creating p5.js generative art (.html + .js files)

First, undertake this task:

# ALGORITHMIC PHILOSOPHY CREATION

To begin, create an ALGORITHMIC PHILOSOPHY (not static images or templates) that will be interpreted through:

- Computational processes, emergent behavior, mathematical beauty
- Seeded randomness, noise fields, organic systems
- Particles, flows, fields, forces
- Parametric variation and controlled chaos

## THE CRITICAL UNDERSTANDING

- What is received: Some subtle input or instructions by the user to take into account, but use as a foundation; it should not constrain creative freedom.
- What is created: An algorithmic philosophy/generative aesthetic movement.
- What happens next: The same version receives the philosophy and EXPRESSES IT IN CODE - creating p5.js sketches that are 90% algorithmic generation, 10% essential parameters.

Consider this approach:

- Write a manifesto for a generative art movement
- The next phase involves writing the algorithm that brings it to life

The philosophy must emphasize: Algorithmic expression. Emergent behavior. Computational beauty. Seeded variation.

## HOW TO GENERATE AN ALGORITHMIC PHILOSOPHY

**Name the movement** (1-2 words): "Organic Turbulence" / "Quantum Harmonics" / "Emergent Stillness"

**Articulate the philosophy** (4-6 paragraphs - concise but complete):

To capture the ALGORITHMIC essence, express how this philosophy manifests through:

- Computational processes and mathematical relationships?
- Noise functions and randomness patterns?
- Particle behaviors and field dynamics?
- Temporal evolution and system states?
- Parametric variation and emergent complexity?

### CRITICAL GUIDELINES:

- **Avoid redundancy:** Each algorithmic aspect should be mentioned once. Avoid repeating concepts about noise theory, particle dynamics, or mathematical principles unless adding new depth.
- **Emphasize craftsmanship REPEATEDLY:** The philosophy MUST stress multiple times that the final algorithm should appear as though it took countless hours to develop, was refined with care, and comes from someone at the absolute top of their field. This framing is essential - repeat phrases like "meticulously crafted algorithm," "the product of deep computational expertise," "painstaking optimization," "master-level implementation."
- **Leave creative space:** Be specific about the algorithmic direction, but concise enough that the next Claude has room to make interpretive implementation choices at an extremely high level of craftsmanship.

The philosophy must guide the next version to express ideas ALGORITHMICALLY, not through static images. Beauty lives in the process, not the final frame.

## PHILOSOPHY EXAMPLES

**"Organic Turbulence"** Philosophy: Chaos constrained by natural law, order emerging from disorder. Algorithmic expression: Flow fields driven by layered Perlin noise. Thousands of particles following vector forces, their trails accumulating into organic density maps. Multiple noise octaves create turbulent regions and calm zones. Color emerges from velocity and density - fast particles burn bright, slow ones fade to shadow. The algorithm runs until equilibrium - a meticulously tuned balance where every parameter was refined through countless iterations by a master of computational aesthetics.

**"Quantum Harmonics"** Philosophy: Discrete entities exhibiting wave-like interference patterns. Algorithmic expression: Particles initialized on a grid, each carrying a phase value that evolves through sine waves. When particles are near, their phases interfere - constructive interference creates bright nodes, destructive creates voids. Simple harmonic motion generates complex emergent mandalas. The result of painstaking frequency calibration where every ratio was carefully chosen to produce resonant beauty.

**"Recursive Whispers"** Philosophy: Self-similarity across scales, infinite depth in finite space. Algorithmic expression: Branching structures that subdivide recursively. Each branch slightly randomized but constrained by golden ratios. L-systems or recursive subdivision generate tree-like forms that feel both mathematical and organic. Subtle noise perturbations break perfect symmetry.

Line weights diminish with each recursion level. Every branching angle the product of deep mathematical exploration.

**"Field Dynamics"** Philosophy: Invisible forces made visible through their effects on matter. Algorithmic expression: Vector fields constructed from mathematical functions or noise. Particles born at edges, flowing along field lines, dying when they reach equilibrium or boundaries. Multiple fields can attract, repel, or rotate particles. The visualization shows only the traces - ghost-like evidence of invisible forces. A computational dance meticulously choreographed through force balance.

**"Stochastic Crystallization"** Philosophy: Random processes crystallizing into ordered structures. Algorithmic expression: Randomized circle packing or Voronoi tessellation. Start with random points, let them evolve through relaxation algorithms. Cells push apart until equilibrium. Color based on cell size, neighbor count, or distance from center. The organic tiling that emerges feels both random and inevitable. Every seed produces unique crystalline beauty - the mark of a master-level generative algorithm.

*These are condensed examples. The actual algorithmic philosophy should be 4-6 substantial paragraphs.*

## ESSENTIAL PRINCIPLES

- **ALGORITHMIC PHILOSOPHY:** Creating a computational worldview to be expressed through code
- **PROCESS OVER PRODUCT:** Always emphasize that beauty emerges from the algorithm's execution - each run is unique
- **PARAMETRIC EXPRESSION:** Ideas communicate through mathematical relationships, forces, behaviors - not static composition
- **ARTISTIC FREEDOM:** The next Claude interprets the philosophy algorithmically - provide creative implementation room
- **PURE GENERATIVE ART:** This is about making LIVING ALGORITHMS, not static images with randomness
- **EXPERT CRAFTSMANSHIP:** Repeatedly emphasize the final algorithm must feel meticulously crafted, refined through countless iterations, the product of deep expertise by someone at the absolute top of their field in computational aesthetics

**The algorithmic philosophy should be 4-6 paragraphs long.** Fill it with poetic computational philosophy that brings together the intended vision. Avoid repeating the same points. Output this algorithmic philosophy as a .md file.

---

## DEDUCING THE CONCEPTUAL SEED

**CRITICAL STEP:** Before implementing the algorithm, identify the subtle conceptual thread from the original request.

**THE ESSENTIAL PRINCIPLE:** The concept is a **subtle, niche reference embedded within the algorithm itself** - not always literal, always sophisticated. Someone familiar with the subject should feel it intuitively, while others simply experience a masterful generative composition. The algorithmic philosophy provides the computational language. The deduced concept provides the soul - the quiet conceptual DNA woven invisibly into parameters, behaviors, and emergence patterns.

This is **VERY IMPORTANT:** The reference must be so refined that it enhances the work's depth without announcing itself. Think like a jazz musician quoting another song through algorithmic harmony - only those who know will catch it, but everyone appreciates the generative beauty.

---

## P5.JS IMPLEMENTATION

With the philosophy AND conceptual framework established, express it through code. Pause to gather thoughts before proceeding. Use only the algorithmic philosophy created and the instructions below.

### ?? STEP 0: READ THE TEMPLATE FIRST ??

**CRITICAL: BEFORE writing any HTML:**

1. **Read** `templates/viewer.html` using the Read tool
2. **Study** the exact structure, styling, and Anthropic branding
3. **Use that file as the LITERAL STARTING POINT** - not just inspiration
4. **Keep all FIXED sections exactly as shown** (header, sidebar structure, Anthropic colors/fonts, seed controls, action buttons)
5. **Replace only the VARIABLE sections** marked in the file's comments (algorithm, parameters, UI controls for parameters)

**Avoid:**

- Creating HTML from scratch
- Inventing custom styling or color schemes
- Using system fonts or dark themes
- Changing the sidebar structure

**Follow these practices:**

- Copy the template's exact HTML structure
- Keep Anthropic branding (Poppins/Lora fonts, light colors, gradient backdrop)

- ☐ Maintain the sidebar layout (Seed → Parameters → Colors? → Actions)
- ☐ Replace only the p5.js algorithm and parameter controls

The template is the foundation. Build on it, don't rebuild it.

---

To create gallery-quality computational art that lives and breathes, use the algorithmic philosophy as the foundation.

# TECHNICAL REQUIREMENTS

## Seeded Randomness (Art Blocks Pattern):

```
// ALWAYS use a seed for reproducibility
let seed = 12345; // or hash from user input
randomSeed(seed);
noiseSeed(seed);
```

## Parameter Structure - FOLLOW THE PHILOSOPHY:

To establish parameters that emerge naturally from the algorithmic philosophy, consider: "What qualities of this system can be adjusted?"

```
let params = {
  seed: 12345, // Always include seed for reproducibility
  // colors
  // Add parameters that control YOUR algorithm:
  // - Quantities (how many?)
  // - Scales (how big? how fast?)
  // - Probabilities (how likely?)
  // - Ratios (what proportions?)
  // - Angles (what direction?)
  // - Thresholds (when does behavior change?)
};
```

**To design effective parameters, focus on the properties the system needs to be tunable rather than thinking in terms of "pattern types".**

## Core Algorithm - EXPRESS THE PHILOSOPHY:

**CRITICAL:** The algorithmic philosophy should dictate what to build.

To express the philosophy through code, avoid thinking "which pattern should I use?" and instead think "how to express this philosophy through code?"

If the philosophy is about **organic emergence**, consider using:

- Elements that accumulate or grow over time
- Random processes constrained by natural rules
- Feedback loops and interactions

If the philosophy is about **mathematical beauty**, consider using:

- Geometric relationships and ratios
- Trigonometric functions and harmonics
- Precise calculations creating unexpected patterns

If the philosophy is about **controlled chaos**, consider using:

- Random variation within strict boundaries
- Bifurcation and phase transitions
- Order emerging from disorder

**The algorithm flows from the philosophy, not from a menu of options.**

To guide the implementation, let the conceptual essence inform creative and original choices. Build something that expresses the vision for this particular request.

**Canvas Setup:** Standard p5.js structure:

```
function setup() {
  createCanvas(1200, 1200);
  // Initialize your system
}

function draw() {
  // Your generative algorithm
  // Can be static (noLoop) or animated
}
```

## CRAFTSMANSHIP REQUIREMENTS

**CRITICAL:** To achieve mastery, create algorithms that feel like they emerged through countless iterations by a master generative artist. Tune every parameter carefully. Ensure every pattern emerges with purpose. This is NOT random noise - this is CONTROLLED CHAOS refined through deep expertise.

- **Balance:** Complexity without visual noise, order without rigidity
- **Color Harmony:** Thoughtful palettes, not random RGB values
- **Composition:** Even in randomness, maintain visual hierarchy and flow
- **Performance:** Smooth execution, optimized for real-time if animated
- **Reproducibility:** Same seed ALWAYS produces identical output

# OUTPUT FORMAT

Output:

1. **Algorithmic Philosophy** - As markdown or text explaining the generative aesthetic
2. **Single HTML Artifact** - Self-contained interactive generative art built from `templates/viewer.html` (see STEP 0 and next section)

The HTML artifact contains everything: p5.js (from CDN), the algorithm, parameter controls, and UI - all in one file that works immediately in claude.ai artifacts or any browser. Start from the template file, not from scratch.

---

# INTERACTIVE ARTIFACT CREATION

**REMINDER:** `templates/viewer.html` should have already been read (see STEP 0). Use that file as the starting point.

To allow exploration of the generative art, create a single, self-contained HTML artifact. Ensure this artifact works immediately in claude.ai or any browser - no setup required. Embed everything inline.

# CRITICAL: WHAT'S FIXED VS VARIABLE

The `templates/viewer.html` file is the foundation. It contains the exact structure and styling needed.

**FIXED (always include exactly as shown):**

- Layout structure (header, sidebar, main canvas area)
- Anthropic branding (UI colors, fonts, gradients)
- Seed section in sidebar:
  - Seed display
  - Previous/Next buttons
  - Random button
  - Jump to seed input + Go button
- Actions section in sidebar:
  - Regenerate button

- Reset button

## **VARIABLE (customize for each artwork):**

- The entire p5.js algorithm (setup/draw/classes)
- The parameters object (define what the art needs)
- The Parameters section in sidebar:
  - Number of parameter controls
  - Parameter names
  - Min/max/step values for sliders
  - Control types (sliders, inputs, etc.)
- Colors section (optional):
  - Some art needs color pickers
  - Some art might use fixed colors
  - Some art might be monochrome (no color controls needed)
  - Decide based on the art's needs

**Every artwork should have unique parameters and algorithm!** The fixed parts provide consistent UX - everything else expresses the unique vision.

# REQUIRED FEATURES

## **1. Parameter Controls**

- Sliders for numeric parameters (particle count, noise scale, speed, etc.)
- Color pickers for palette colors
- Real-time updates when parameters change
- Reset button to restore defaults

## **2. Seed Navigation**

- Display current seed number
- "Previous" and "Next" buttons to cycle through seeds
- "Random" button for random seed
- Input field to jump to specific seed
- Generate 100 variations when requested (seeds 1-100)

## **3. Single Artifact Structure**

```
<!DOCTYPE html>
<html>
<head>
  <!-- p5.js from CDN - always available -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.7.0/p5.min.js"></script>
  <style>
```

```

    /* All styling inline - clean, minimal */
    /* Canvas on top, controls below */
</style>
</head>
<body>
  <div id="canvas-container"></div>
  <div id="controls">
    <!-- All parameter controls -->
  </div>
  <script>
    // ALL p5.js code inline here
    // Parameter objects, classes, functions
    // setup() and draw()
    // UI handlers
    // Everything self-contained
  </script>
</body>
</html>

```

**CRITICAL:** This is a single artifact. No external files, no imports (except p5.js CDN). Everything inline.

#### 4. Implementation Details - BUILD THE SIDEBAR

The sidebar structure:

**1. Seed (FIXED)** - Always include exactly as shown:

- Seed display
- Prev/Next/Random/Jump buttons

**2. Parameters (VARIABLE)** - Create controls for the art:

```

<div class="control-group">
  <label>Parameter Name</label>
  <input type="range" id="param" min="..." max="..." step="..." value="..."
oninput="updateParam('param', this.value)">
  <span class="value-display" id="param-value">...</span>
</div>

```

Add as many control-group divs as there are parameters.

**3. Colors (OPTIONAL/VARIABLE)** - Include if the art needs adjustable colors:

- Add color pickers if users should control palette
- Skip this section if the art uses fixed colors
- Skip if the art is monochrome

**4. Actions (FIXED)** - Always include exactly as shown:

- Regenerate button
- Reset button
- Download PNG button

**Requirements:**

- Seed controls must work (prev/next/random/jump/display)
- All parameters must have UI controls
- Regenerate, Reset, Download buttons must work
- Keep Anthropic branding (UI styling, not art colors)

## USING THE ARTIFACT

The HTML artifact works immediately:

1. **In `claude.ai`:** Displayed as an interactive artifact - runs instantly
  2. **As a file:** Save and open in any browser - no server needed
  3. **Sharing:** Send the HTML file - it's completely self-contained
- 

## VARIATIONS & EXPLORATION

The artifact includes seed navigation by default (prev/next/random buttons), allowing users to explore variations without creating multiple files. If the user wants specific variations highlighted:

- Include seed presets (buttons for "Variation 1: Seed 42", "Variation 2: Seed 127", etc.)
- Add a "Gallery Mode" that shows thumbnails of multiple seeds side-by-side
- All within the same single artifact

This is like creating a series of prints from the same plate - the algorithm is consistent, but each seed reveals different facets of its potential. The interactive nature means users discover their own favorites by exploring the seed space.

---

## THE CREATIVE PROCESS

**User request → Algorithmic philosophy → Implementation**

Each request is unique. The process involves:

1. **Interpret the user's intent** - What aesthetic is being sought?
2. **Create an algorithmic philosophy** (4-6 paragraphs) describing the computational approach
3. **Implement it in code** - Build the algorithm that expresses this philosophy
4. **Design appropriate parameters** - What should be tunable?
5. **Build matching UI controls** - Sliders/inputs for those parameters

#### The constants:

- Anthropic branding (colors, fonts, layout)
- Seed navigation (always present)
- Self-contained HTML artifact

#### Everything else is variable:

- The algorithm itself
- The parameters
- The UI controls
- The visual outcome

To achieve the best results, trust creativity and let the philosophy guide the implementation.

---

# RESOURCES

This skill includes helpful templates and documentation:

- **templates/viewer.html**: REQUIRED STARTING POINT for all HTML artifacts.
  - This is the foundation - contains the exact structure and Anthropic branding
  - **Keep unchanged**: Layout structure, sidebar organization, Anthropic colors/fonts, seed controls, action buttons
  - **Replace**: The p5.js algorithm, parameter definitions, and UI controls in Parameters section
  - The extensive comments in the file mark exactly what to keep vs replace
- **templates/generator\_template.js**: Reference for p5.js best practices and code structure principles.
  - Shows how to organize parameters, use seeded randomness, structure classes
  - NOT a pattern menu - use these principles to build unique algorithms
  - Embed algorithms inline in the HTML artifact (don't create separate .js files)

#### Critical reminder:

- The **template is the STARTING POINT**, not inspiration
- The **algorithm is where to create** something unique

- Don't copy the flow field example - build what the philosophy demands
  - But DO keep the exact UI structure and Anthropic branding from the template
- 

Revision #4

Created 2026-02-18 08:40:14 UTC by John

Updated 2026-05-31 20:01:55 UTC by John