# Demo: Drawing Algorithms As Modular Objects

## A Framework for Procedurally Generated Visual Arts and Music

Xingyu Dong
University of Pennsylvania

Daniel Průša
Czech Technical University

Michael Wehar
Haverford College

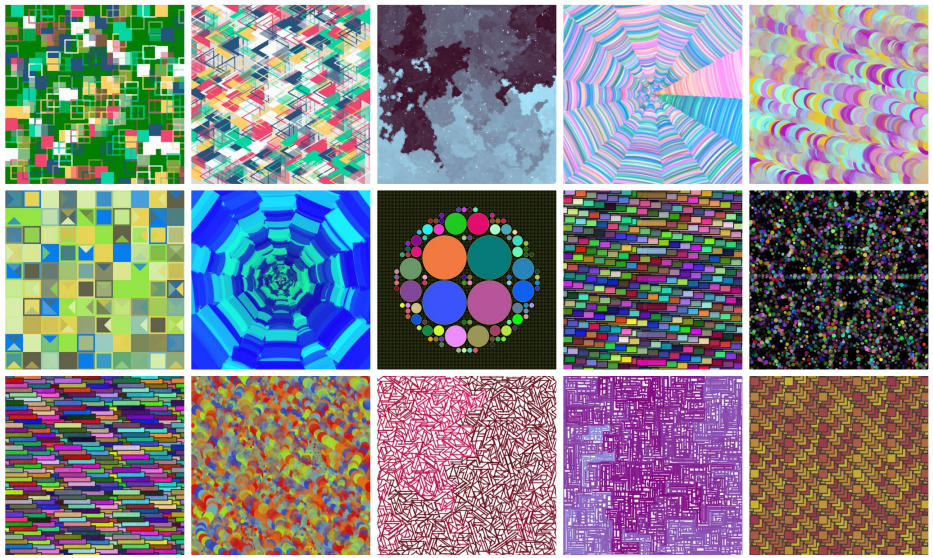
**Chen Xu**
University at Buffalo

Functional Art, Music, Modelling and Design (FARM 2026)

# Outline

# Algorithms are a medium for art and teaching

- Lower the barrier between **computing, art, and education**.
- Give learners immediate visual feedback as code executes.
- Let students exhibit the artifacts produced by their programs.

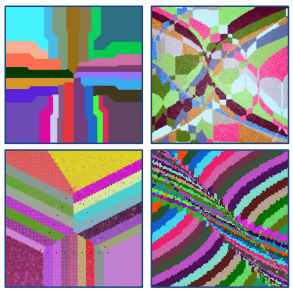


Earlier AlgoArt gallery examples

**Goal**

The image is both the output and a trace of the process.

# AlgoArt grew through student projects




Voronoi experiments

Algorithmically generated art on clothing

- Teaching modules and multi-generational capstone coding projects accumulated over time.
- Projects expanded to Voronoi Diagram art, augmented-reality process views, and wearable designs.

The next challenge was reuse: can every drawing algorithm exhibit similar structure and controls?

# Outline

# AlgoArt gives every algorithm the same controls

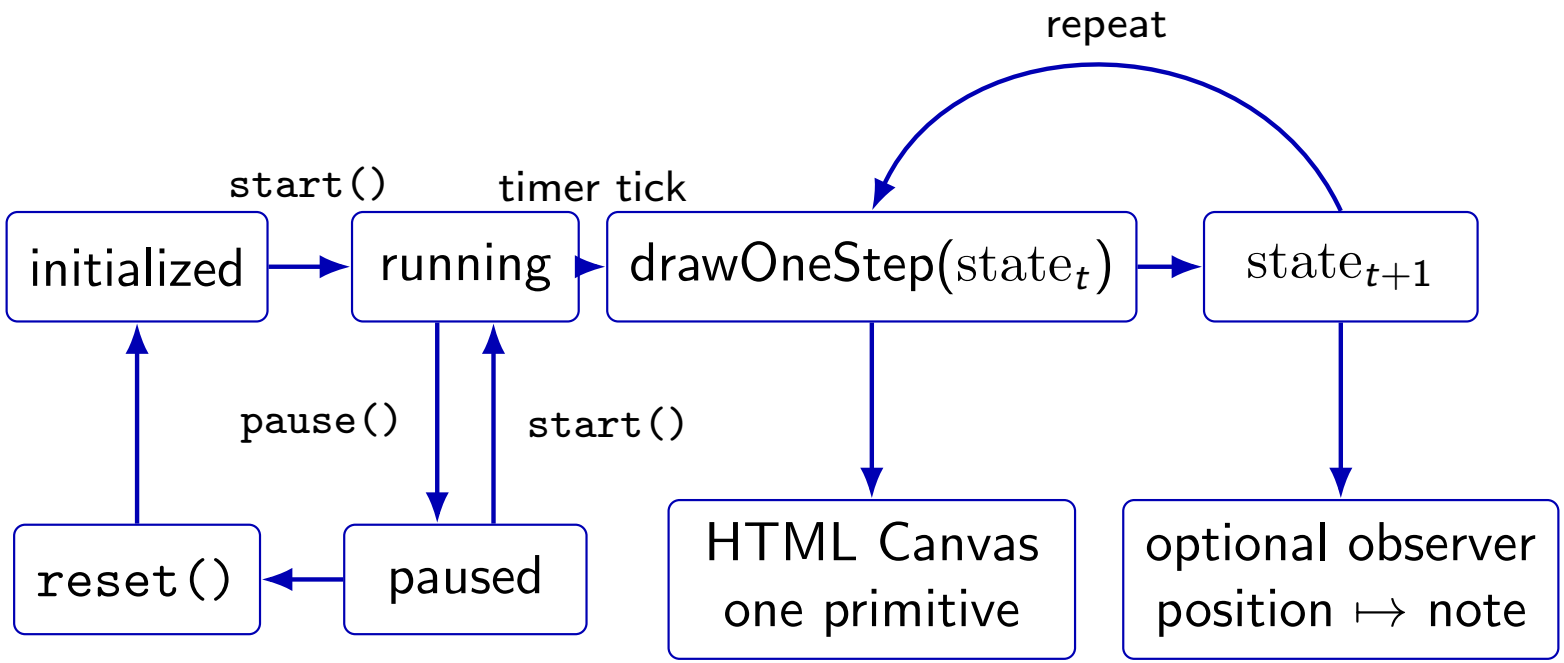
**Standard methods**

`initialize()`
`start()`
`pause()`
`reset()`
`drawOneStep()`

- A JavaScript object stores the algorithm's state; a separate file stores its parameters.
- Each `drawOneStep()` call draws to an external HTML Canvas.
- Other functions accept the object as input, connecting it to interfaces, analysis, or sound.

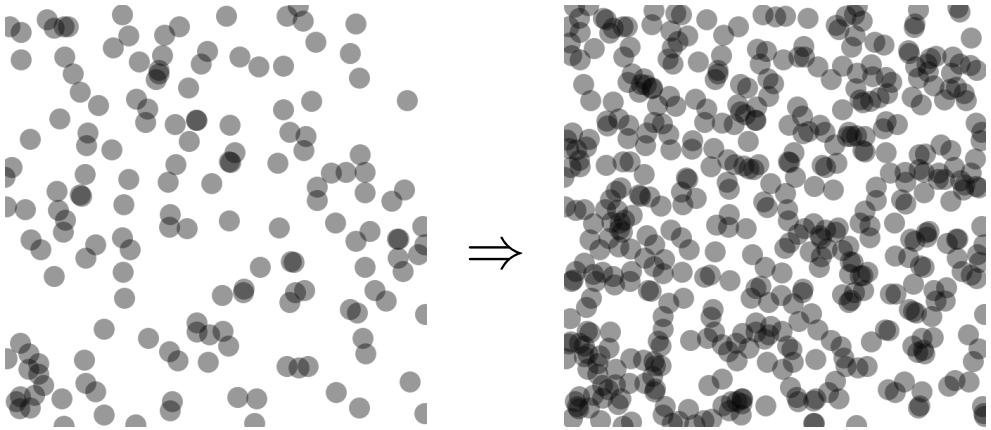
**Same structure & controller; different drawing behavior.**

# State makes drawing controllable




A drawing is a sequence of small, observable state transitions—not one opaque render call.

# Dots separates behavior from parameters



partway through



after drawing finishes

**One call adds one dot.**

```
drawOneStep()
```

```
p = randomPoint()
drawCircle(p, radius,
   color, alpha)

steps += 1
```

**Parameters:** speed, maximum steps, radius, color, opacity

Change the parameters; keep the shared contract.

# Outline

# Coding challenges reward efficient answers

**Input**

An $m \times n$ binary matrix.

$\Rightarrow$

**Task**

Enumerate every eligible rectangle.

$\Rightarrow$

**Measure**

Time and space.

**Enumerating Rectangles With Empty Interior**

The paper adapts a previously developed solver into a drawing algorithm.

# Which rectangles are eligible?

**Matrix $M$**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

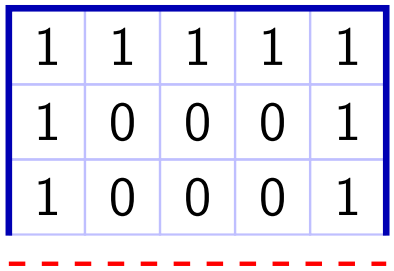
Highlighted block: rows 1–3, columns 2–5

**Eligible block**

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

- every border entry is 1;
- every interior entry is 0.

# A staple is a partial rectangle




bottom edge is not known yet

- The top border contains only 1s.
- The left and right sides contain only 1s.
- Below the top, entries between the sides are 0.
- A later row may close the open bottom.

The row sweep carries these candidates forward.

# Each row updates the active staples

**Close**

1 0 0 0 1

1 1 1 1 1

A row of 1s supplies the bottom border.

**Continue**

1 0 0 0 1

1 0 0 0 1

Matching side entries extend the staple.

**Start**

0 0 0 0 0

1 1 1 1 1

A new top border creates new candidates.

## Prior result

$O(mn)$ time and $O(n)$ space.

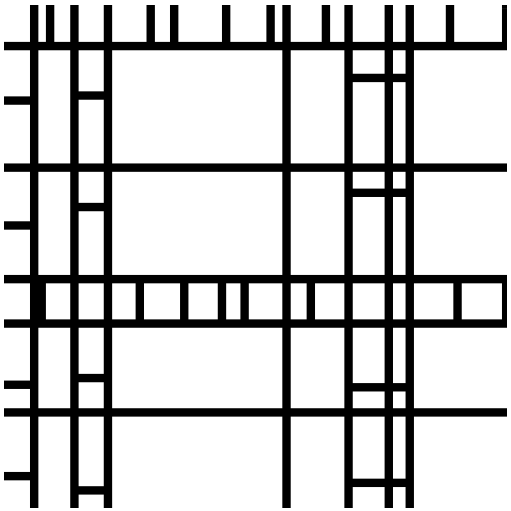
The scan visits the matrix row by row and stores only linear workspace.

**Algorithm and Complexity**

Finding Maximum and Minimum Size Matrices. Abdelmonsef et al. (FUN 2026)

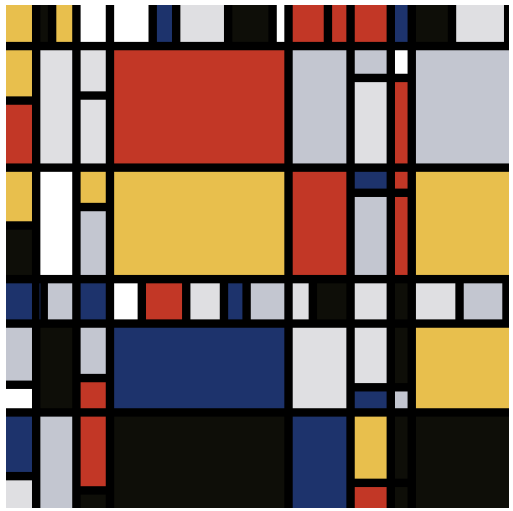
# Outline

# The same solver can color an image



input image

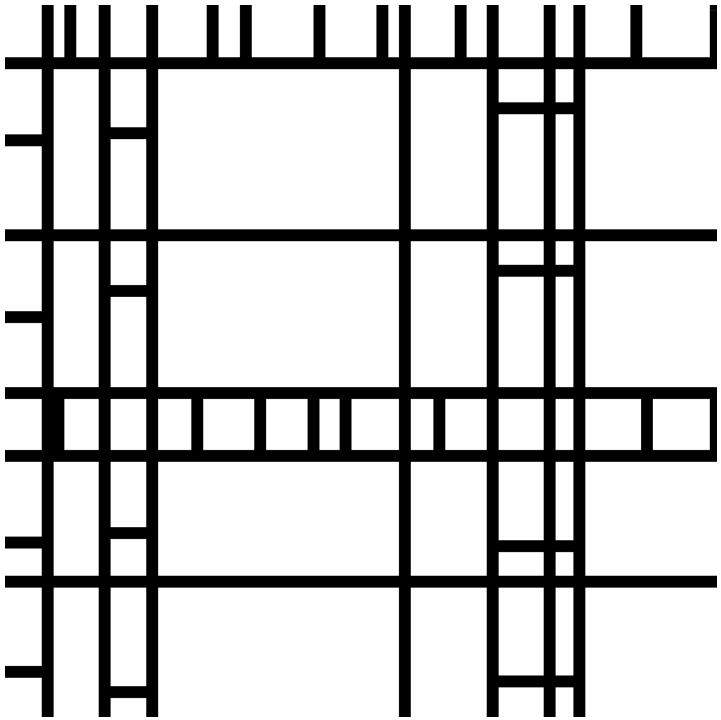
1. white or transparent $\mapsto 0$;
2. every other pixel $\mapsto 1$;
3. enumerate eligible rectangles;
4. fill each reported region.



colored output

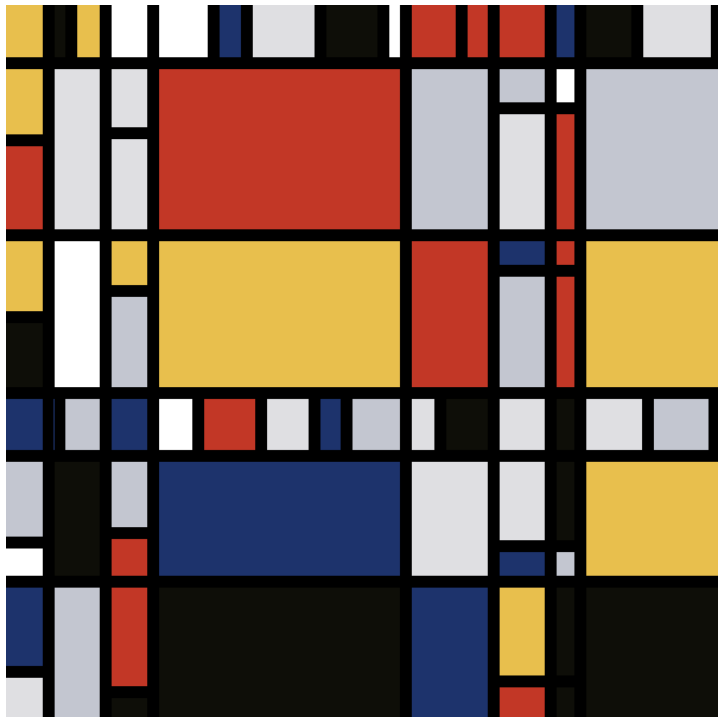
The detector is unchanged; only its output acquires a visual action.

# Coloring makes coverage visible



input grid

$\Rightarrow$

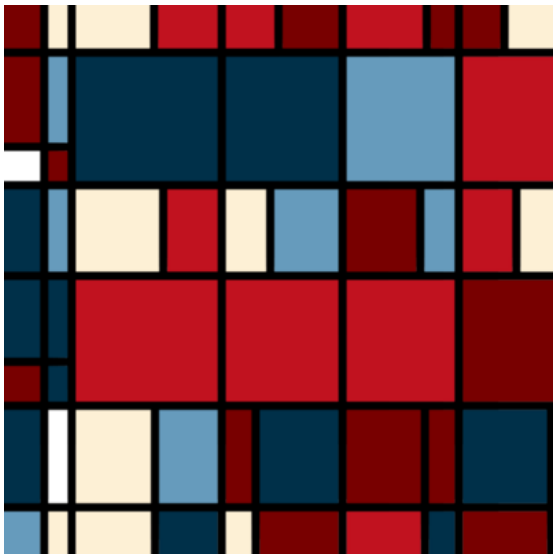
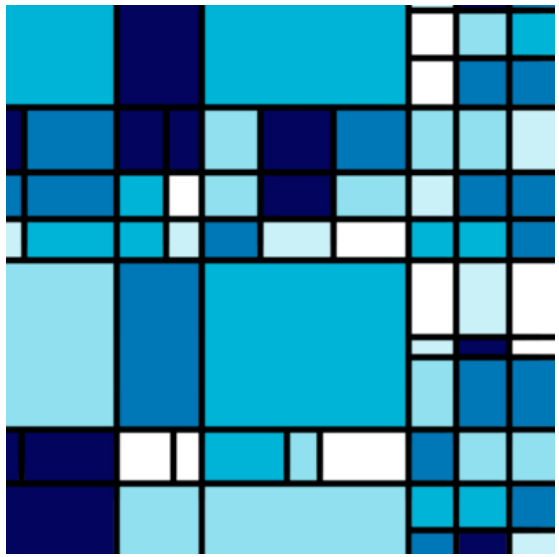
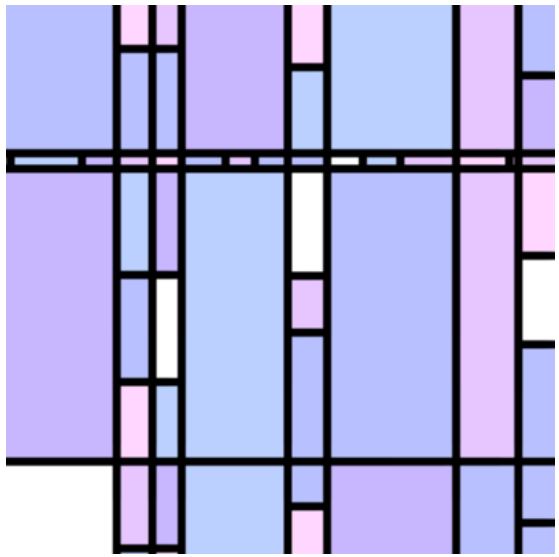


detected rectangles filled

- An eligible region left white points to a possible miss.
- An incorrect fill points to a possible false positive.

Visual inspection supports debugging; it does not replace proof or testing.

# Palettes turn rectangles into *Composition*

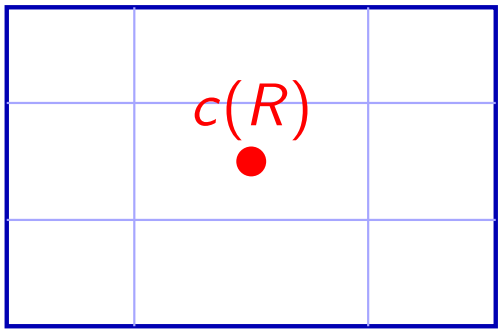




- Drawing algorithm generates a grid, then invokes the rectangle solver.
- A selected palette supplies fill colors for the reported regions.

**Swap the coloring policy without changing the detector or control interface.**

Visual style inspired by Piet Mondrian's "Composition" works.

# Reference images create non-square pixels



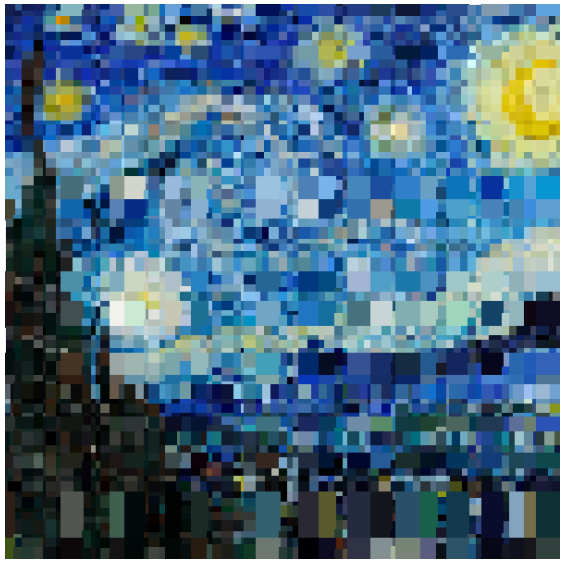

one detected rectangle $R$

**For each rectangle $R$:**

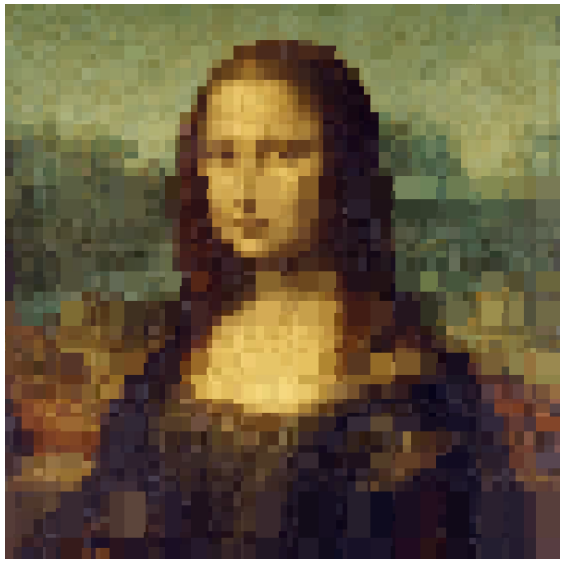
1. locate its center in a reference image $I$;
2. sample that single pixel;
3. fill the whole rectangle with the sampled color.

$$\text{color}(R) = I\big(c(R)\big)$$

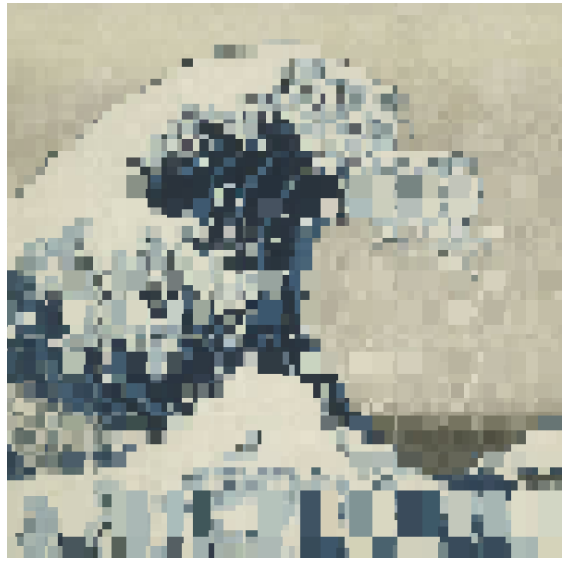
# The same geometry reinterprets art



*Starry Night* by Vincent van Gogh



*Mona Lisa* by Leonardo da Vinci



*The Great Wave Off Kanagawa* by Hokusai

Square pixels become rectangles of many aspect ratios; the detector stays the same.

Public-domain reference works; visual approach inspired by Adam Lister's works.

# Outline

# What this work contributes

1. **Framework:** stateful JavaScript drawing objects share five standard controls.
2. **Connection:** an algorithmic solution becomes a step-by-step drawing algorithm.
3. **Demonstration:** a prior $O(mn)$-time, $O(n)$-space rectangle scan supports art, education, and inspectable execution.

## Takeaway

**One algorithm can solve, draw, certify, and explain.**

# Next: richer regions and richer traces

**Extend the geometry**

- enumerate empty non-rectangular regions;
- visualize them with flood fill;
- explore tilings and segmentation.

**Extend the experience**

- find algorithms with meaningful visual traces;
- combine visual inspection with testing;
- reuse one engine across teaching, galleries, and music.

algoart.org · Creator Studio · rectangle solver · paper

# Thank You!

Questions or Comments?

AlgoArt · Composition gallery · FARM 2026 paper

Thanks to Lorraine (Khanh) Nguyen, the AlgoArt collaborators, the Swarthmore Algorithms Research Group participants, the Composition 2026 community, and the FARM reviewers.