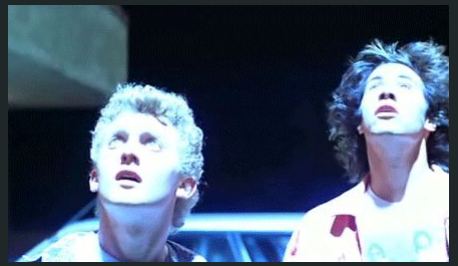
Express Yourself: Creative Coding and Generative Art

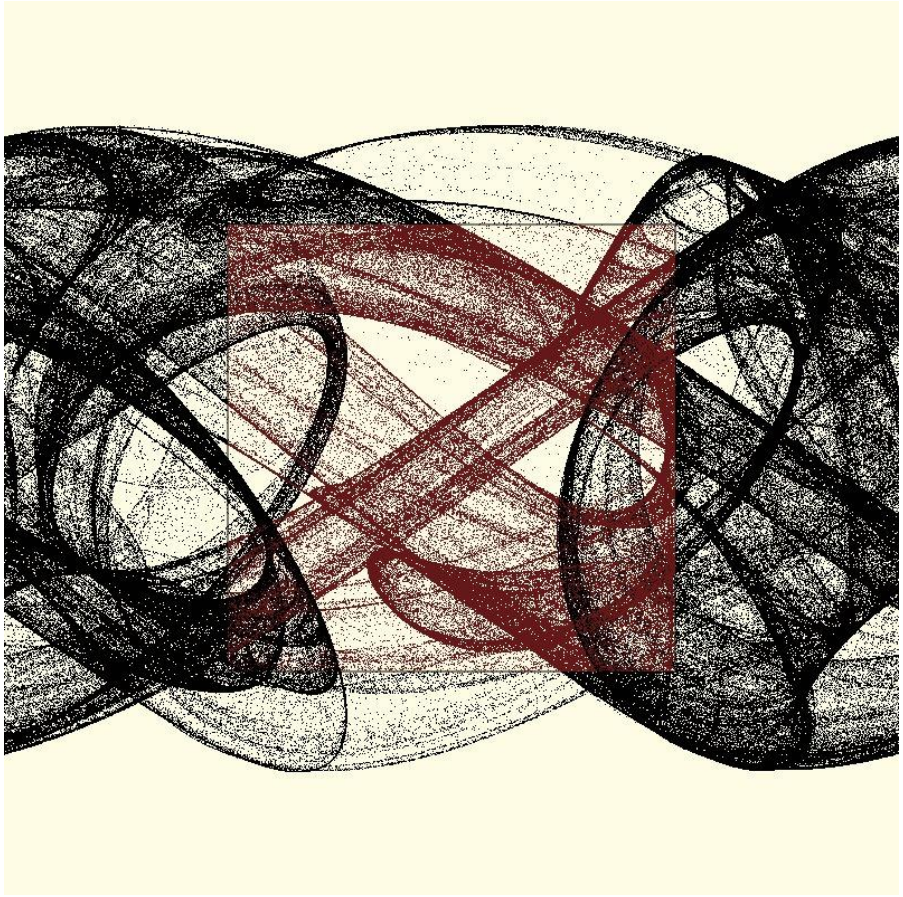
School of Computing // Seminar Series

Fredericks / 2024

http://efredericks.net

https://discord.gg/pNaTDKH





https://editor.p5js.org/frederer/sketches/2HV0XJKmJ

Strange attractor - colored based on location

Things to talk about

Creative coding
Generative art
Generative art research

FIRST

What comes to your mind when you think of **creative coding?**

How about **generative art?**

https://www.patrik-huebner.com/datadesigndictionary/creative-coding/

Creative coding

All about discovery and exploration!
Or ➜ writing algorithms and having an **immediate and visual output**

Often used to learn coding!

Characterized by abstractions to facilitate learning or **understanding**

Popularized by this guy ➜
Daniel Shiffman - TheCodingTrain

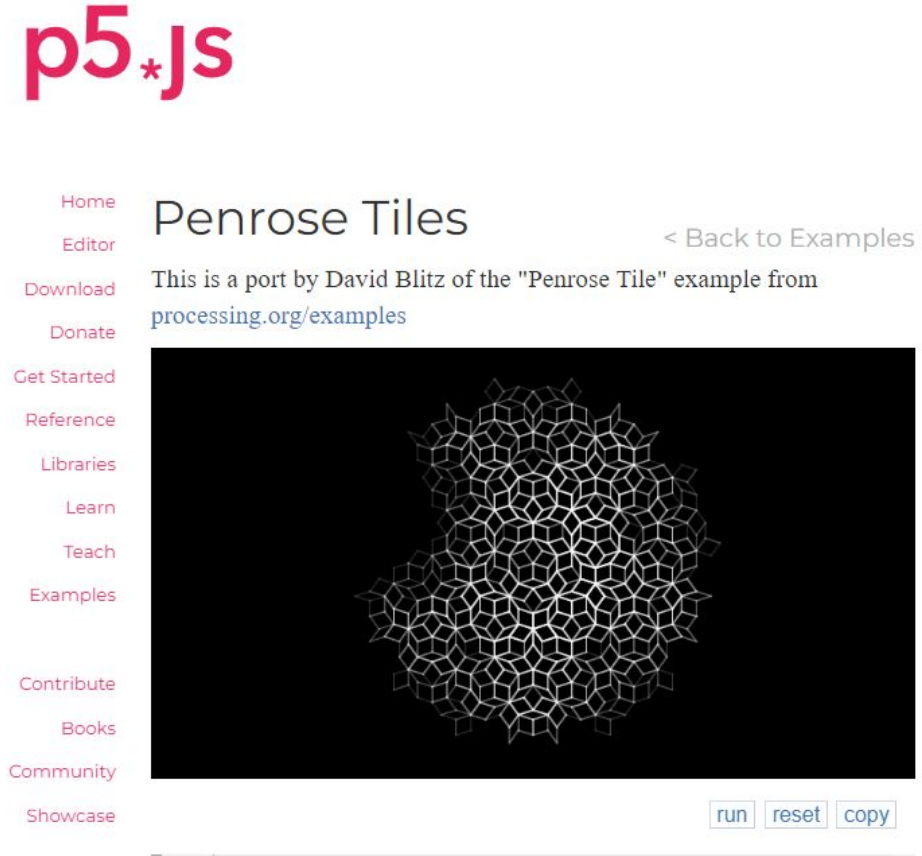
https://www.youtube.com/c/TheCodingTrain/

Examples of Creative Coding

https://p5js.org/examples/

- Game of life
- L-systems
- Cellular automata
- Particle systems

Possibilities are endless!

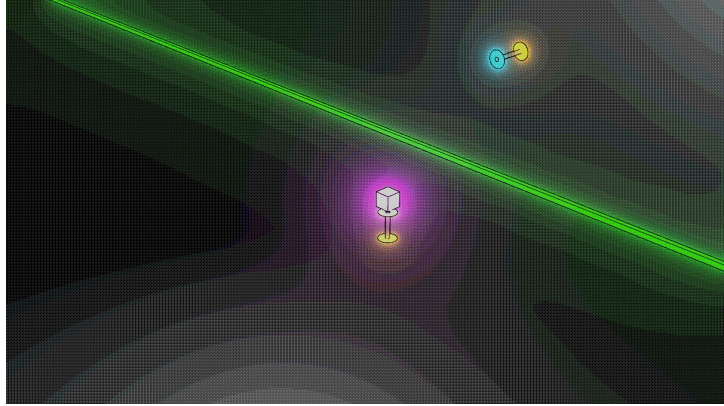


Creative coding outputs

Is a digital display our only form of output?

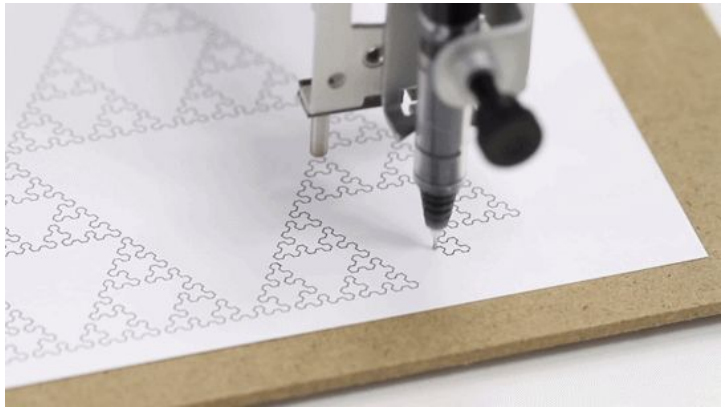
Absolutely not!

- Possible to take your creations and make them a reality!
  - Pen plotting, 3D printing, etc.



Blender - physics + post-processing

https://mary.codes/blog/art/3d_printing_generative_art_with_p5_and_blender/



https://prostheticknowledge.tumblr.com/post/153869265156/axidraw-v3-new-version-of-precise-plotter-drawing

How can we do this?

Really, any language that supports visuals can be used

Application-based:

- **Python**, C++, Java ➜ all support writing to windows

Web-based:

- JavaScript + HTML ➜ write to <canvas>
  - Shaders ➜ write to <canvas> *much faster*

- **p5js** ➜ abstraction over JavaScript (to write to <canvas>)
  - Horribly slow, but much quicker to prototype!
  - Also supports shaders
  - And *very* well-documented

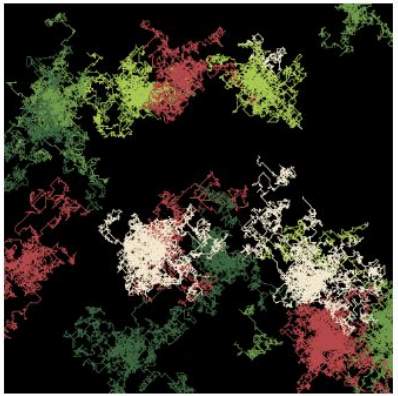
All you really need to do is:

Have an algorithm, output, or dataset you want to *visualize*

- What is this visualization going to show?
- To represent?

Have a method for representing it in a visual manner

- How are we going to draw this *thing* in such a way that it will have **meaning**?



(e) Drunkard's Walk

Generative art

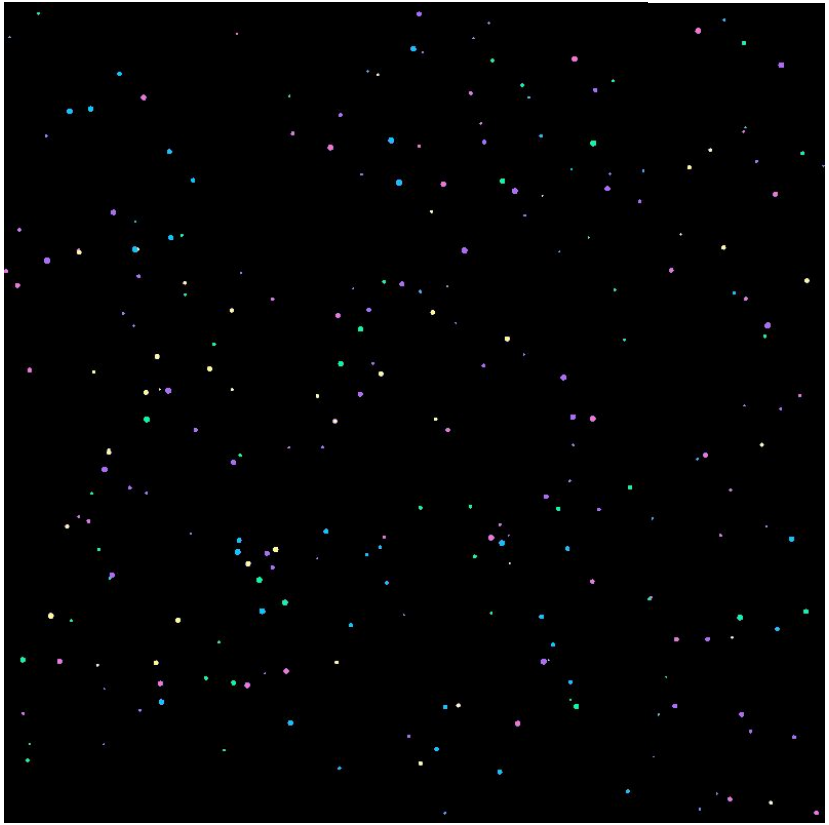
https://editor.p5js.org/frederer/sketches/DbWjEErKy

Creative coding - but aesthetically pleasing

Take those visualizations and make them works of art!

Tangentially - there is also generative AI

- Midjourney, DALL-E, etc.
- Not talking about that today



Flow field visualization

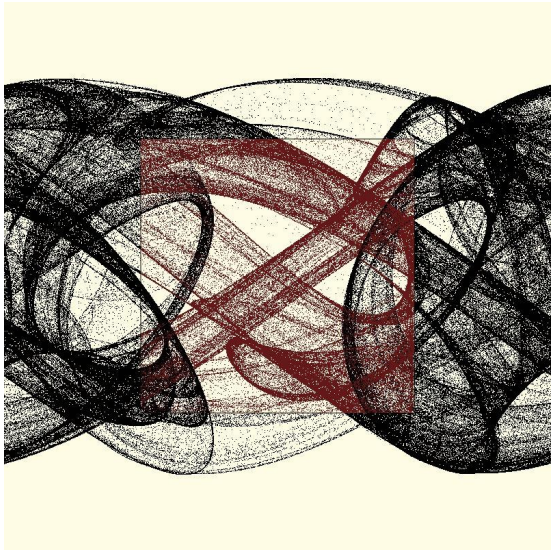
And then … random numbers

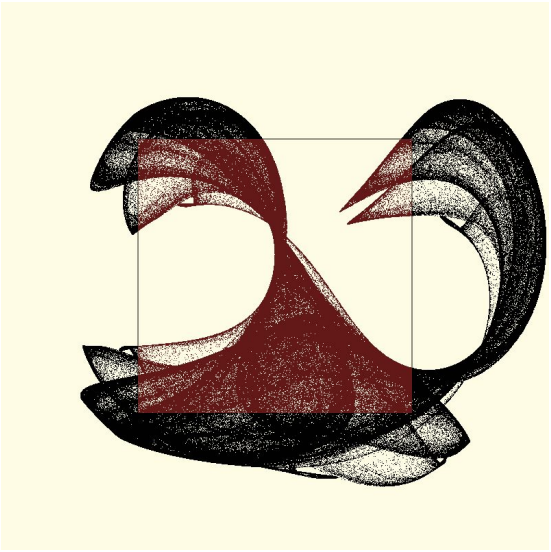
```
function clifford(x, y) {
  let _x = sin(a * y) + c * cos(a * x);
  let _y = sin(b * x) + d * cos(b * y);
  return { x: _x, y: _y };
}
```

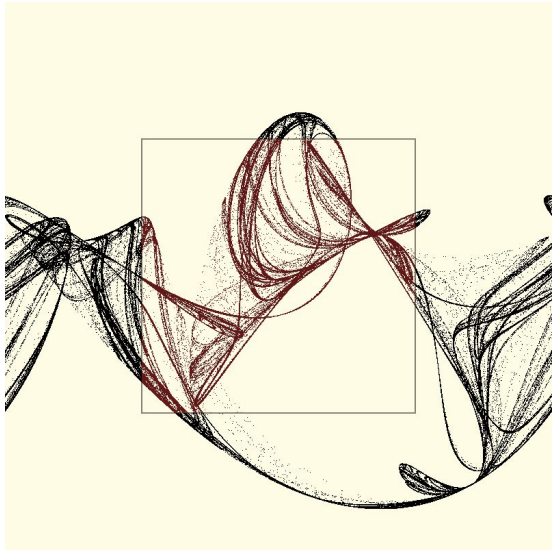
Same algorithm ➔ different outputs

How do we know which parameters to use?

- *More importantly, how do we revisit old outputs?*







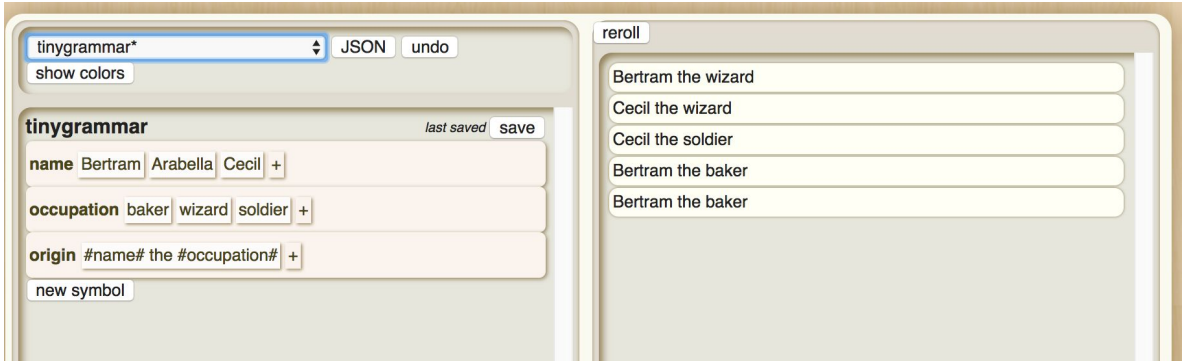
Generative <*X*>

We don't need to simply be making nifty looking things on a screen...

Could be generative music (or visualizers)

- https://junshern.github.io/algorithmic-music-tutorial/part1.html

Could be storytelling



Onto the research...

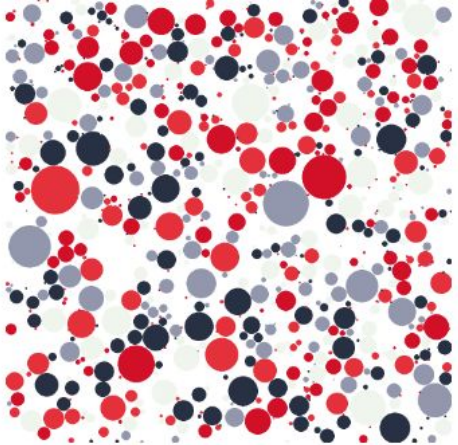
Art + genetic algorithms = research

Main concept:

- Take all of these disparate generative art techniques that we like to make
- Combine them with a search algorithm
- ???
- New art!

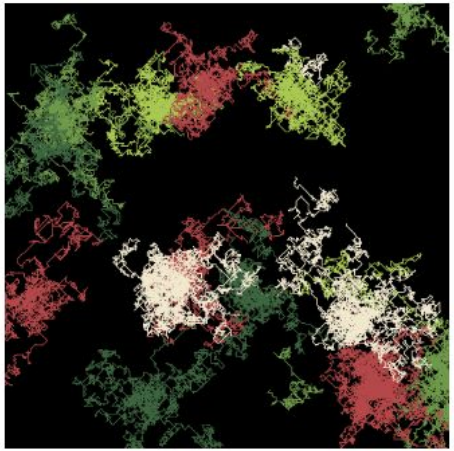
(a) Stippled

(b) Cellular Automata



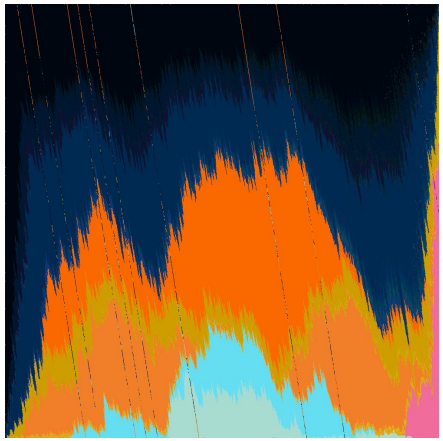
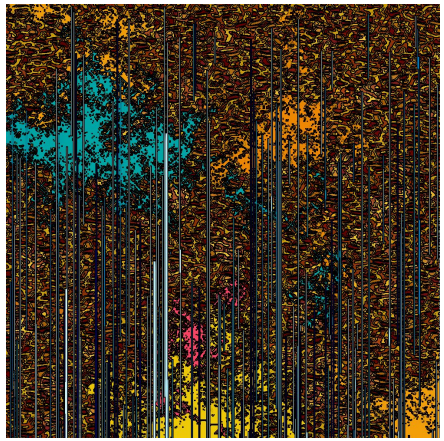
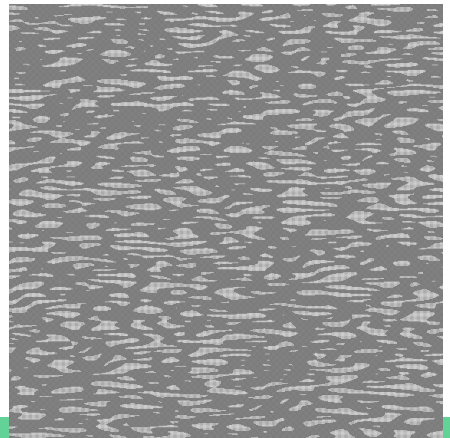
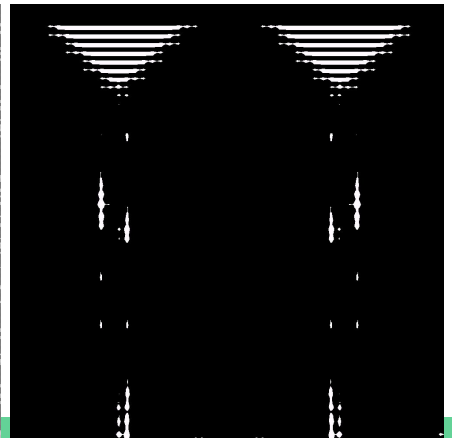
(c) Circle Packing

(d) Flow Field



(e) Drunkard's Walk

Generative art *research* - GenerativeGI

1 workshop paper (best research award)
1 invited journal extension (accepted for pub)

link to blog post ➜

GenerativeGI

Generative art + GI (genetic improvement)

What is GI then?

- Evolutionary computation applied to *source code* to improve program
  - EC: see last week's talk

- Genome is the source code itself

What is our source code?

- A "drawing program"

What is our fitness function?

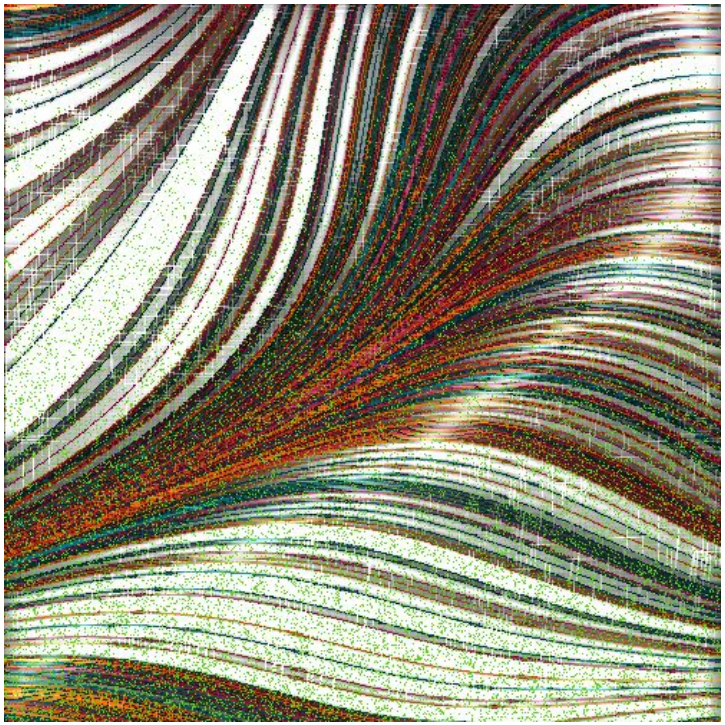
- **The hard part**

```
rules = {
  'ordered_pattern': ['#techniques#'],
  'techniques': ['#technique#', '#techniques#,#technique'],
  'technique': [
        'stippled:', 'wolfram-ca:#palette#',
        'flow-field:#flow-field-type#:#flow-field-zoom#',
        '#flow-field-type#:#palette#:#flow-field-zoom#',
        'flow-field-type': ['edgy', 'curves'],
        'flow-field-zoom': [str(x) for x in np.arange(0.001, 0.5,
                                  0.001)],
        'palette': [['#ff00ff','#ff0000','#00ff00'], ...],
    ...
}
```

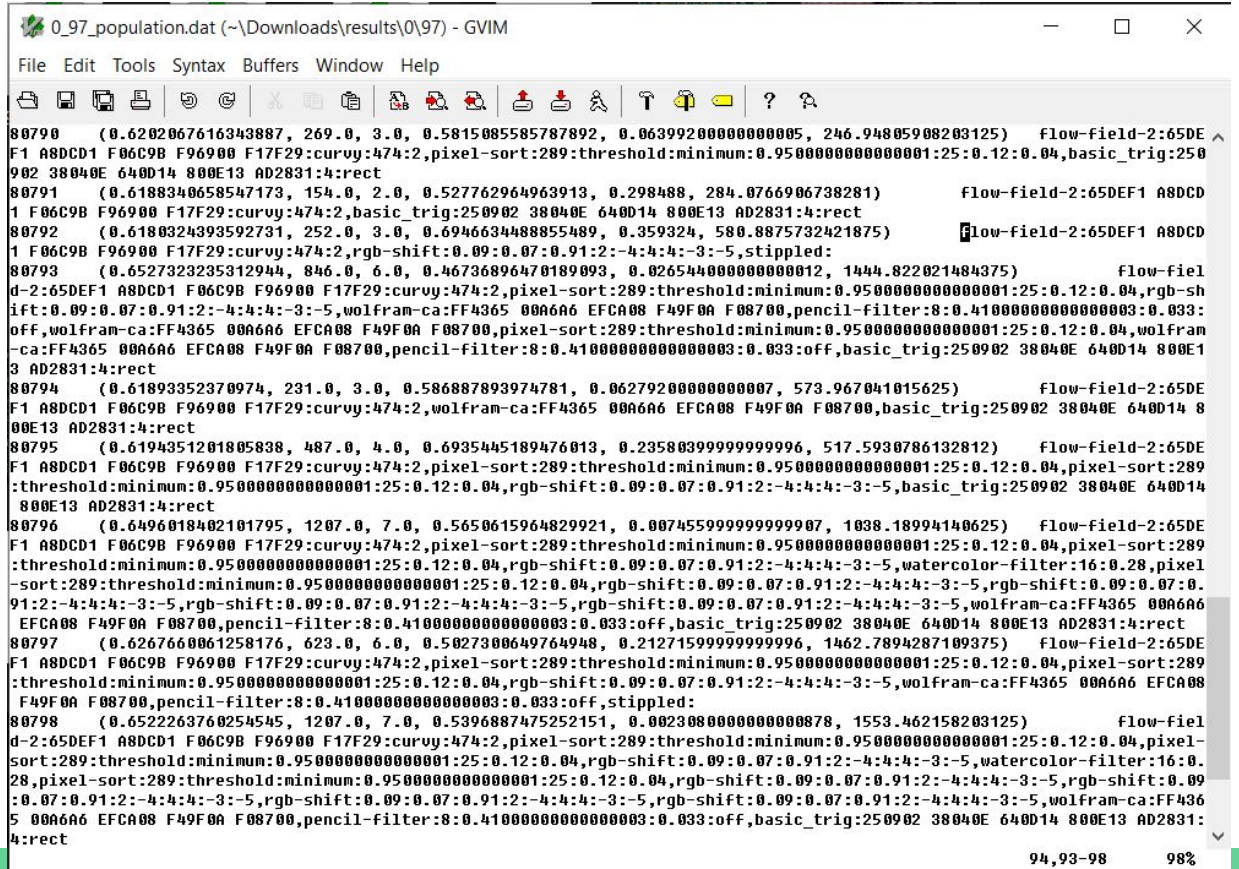
```
flatten('techniques') = flow-field:edgy:#ff00ff #ff0000
                        #00ff00:0.025
```

Resulting in something like this:

```
flow-field-2:65DEF1 A8DCD1 F06C9B
F96900 F17F29:curvy:474:2,

pixel-sort:289:threshold:minimum:0.9500
000000000001:25:0.12:0.04,

pencil-filter:8:0.41000000000000003:0.0
33:off,

stippled:
```



(Looks like this behind the scenes)



The **real** problem - fitness

How do we turn human preference into something measurable?

How do you objectively measure an artistic output?

- Can we turn to machine learning?
  - Bias in datasets?
- Can we turn to LLMs?
  - Copyright issues?
- Can we just ask a bunch of humans?
  - Different preferences?

Fitness functions (or, how we currently check "goodness")

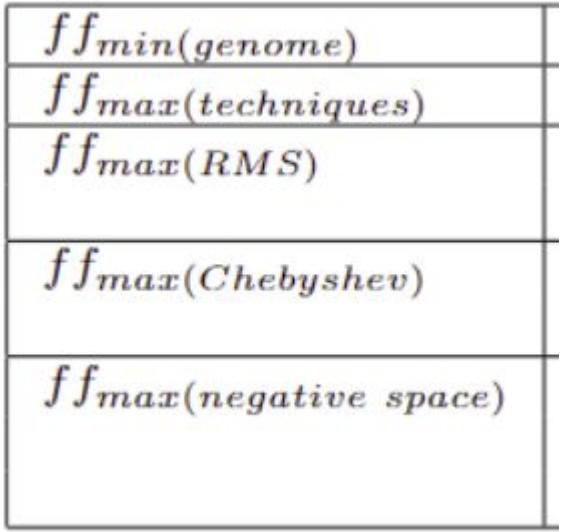
$ff_{min(genome)}$	Minimize the number of duplicate genes across individuals.
$ff_{max(techniques)}$	Maximize the diversity of included techniques in a genome.
$ff_{max(RMS)}$	Maximize the pixel difference between individual image objects using an RMS difference metric.
$ff_{max(Chebyshev)}$	Maximize the pixel difference between individual image objects using a Chebyshev difference metric.
$ff_{max(negative\ space)}$	Minimize the difference between a target "negative space" and the actual "negative space" within an image (70% was the target percentage used during evaluation).

Table 1: *GenerativeGI* fitness functions.

Now the question that is just **burning** through all your collective minds...

What happens if these fitness functions fight with each other?

- Traditionally these would be **weighted**
- Or, a multi-objective evolutionary algorithm used...
  - But that would take even longer to run



Lexicase selection

(or, what Prof. Moore has been researching a lot of)

Standard genetic algorithms use one or few fitness functions (and weight them)

- Difficult to focus on multiple fitnesses!

Lexicase - **many-objective** search

- Randomly shuffle fitness functions and evaluate all
- If one individual is significantly better than another individual then it is **selected** for procreation

Our goal: incorporate as many fitness functions as necessary to get to "real art"

What are we "looking for?"

For this, a glitch art aesthetic

`0 ticks`

`72000 ticks`

`120000 ticks`

https://happycoding.io/examples/p5js/images/pixel-sorter

Outcomes?

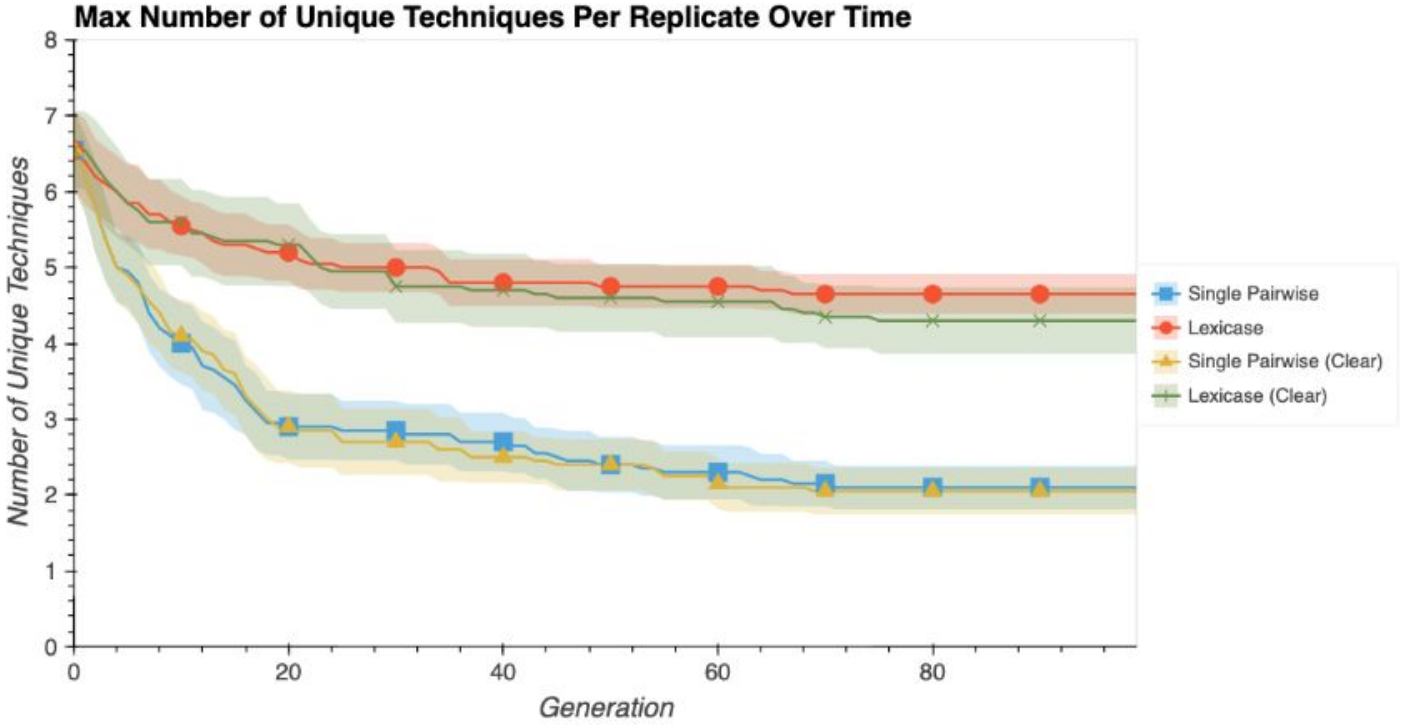


Fig. 16: Population maximum of the number of unique techniques in an individual's genome over time across replicates. Shaded areas represent 95% confidence intervals.

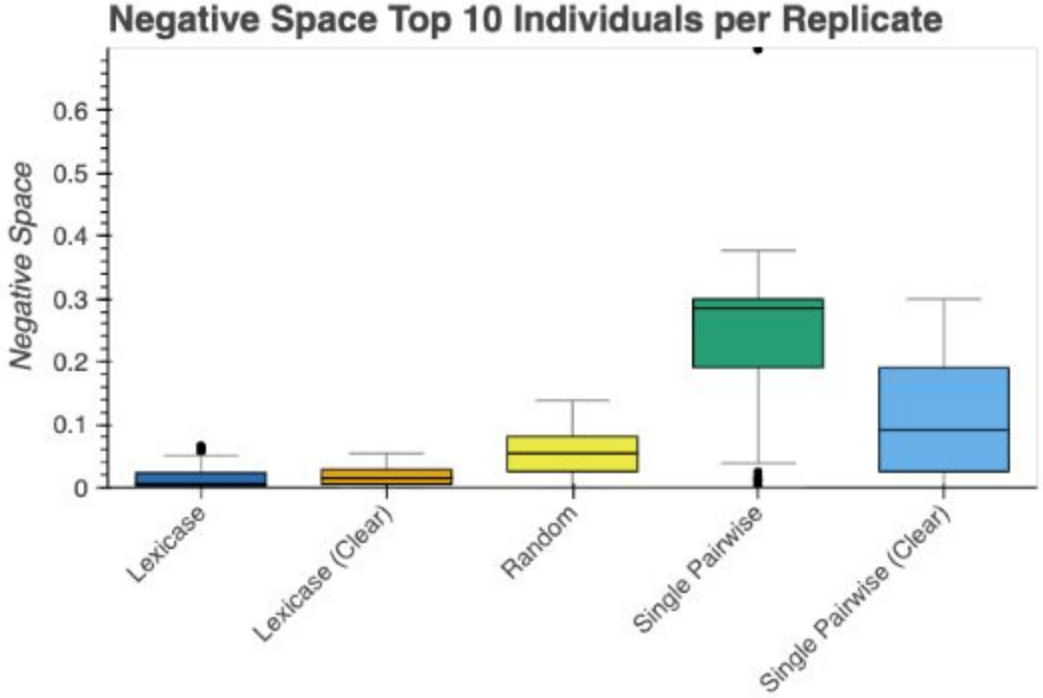
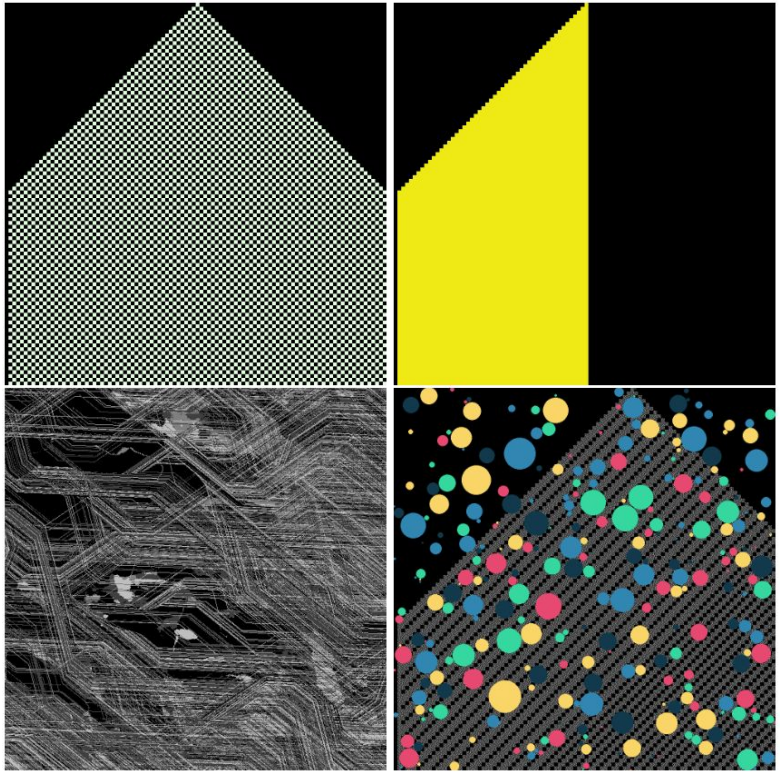
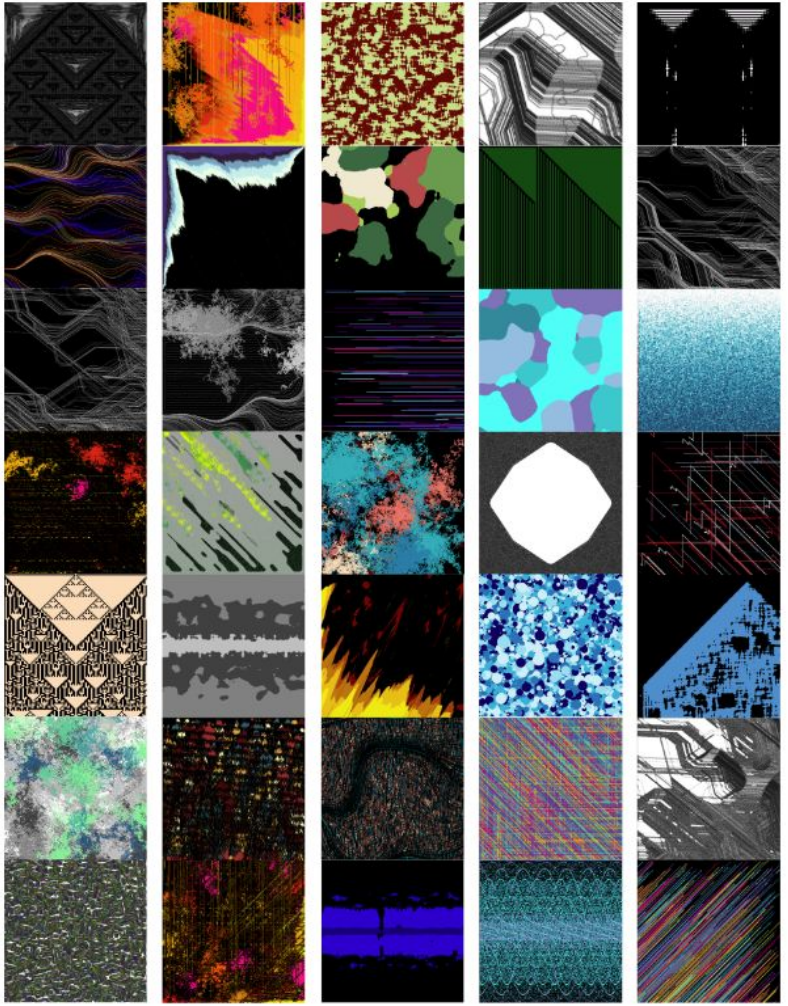


Fig. 14: Negative Space score for the top 10 individuals based on their negative space score per replicate across treatments. Lower scores indicate more successful individuals.




Lexicase
Lexicase (Clear)
Random
Single Pairwise
Single Pairwise (Clear)

And, if you wish to try it out

https://github.com/GI2023-GenerativeGI/GI2023/tree/ASE-GI-Extension

Next steps (or, work in progress)

Two active projects

1) Incorporate a classifier to determine if our outputs are close to "real art"

2) Live display where users can interact with the evolutionary process


AND NOW FOR
SOMETHING
COMPLETELY
DIFFERENT

https://efredericks.github.io/gvsu-cis367/demos/voronoi-sin.html

Computing Seminar on March 14th: **Voronoi Diagrams**

Dividing Space Closest to Specific Points

Speakers: L Dettling and Dan Dietsche

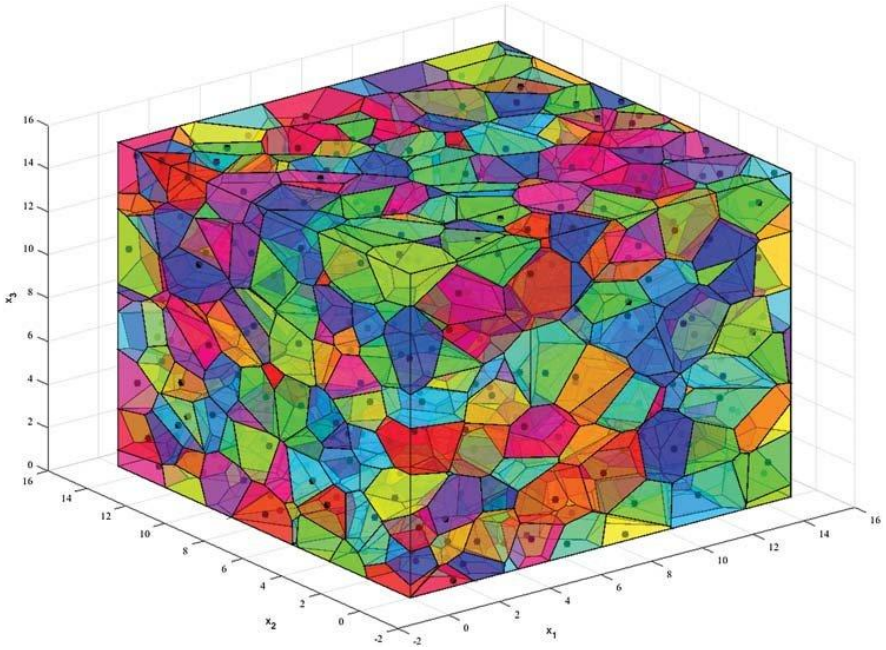


Fig 1. 3-D Voronoi Diagram
https://www.researchgate.net/figure/Initial-3D-Voronoi-diagram_fig2_351028108

Given a space and a set of ***n*** points, one divides the space such that all the space is grouped by which point is closest.

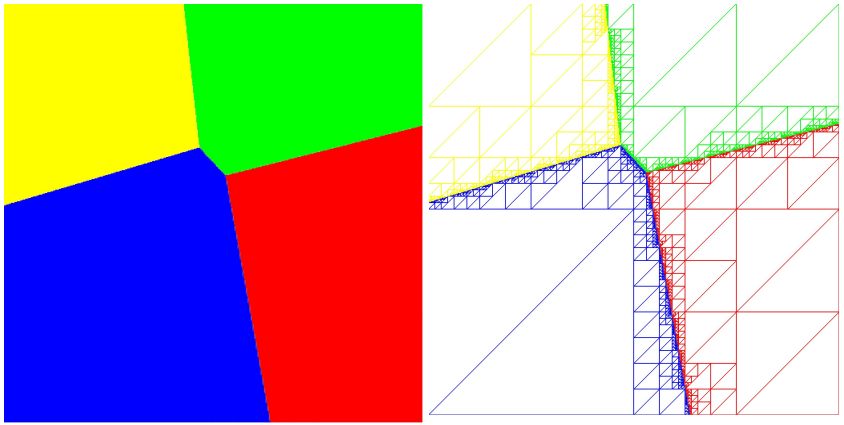


Fig 2. 2-D Voronoi Diagram with 4 seeds.