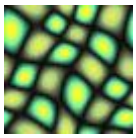
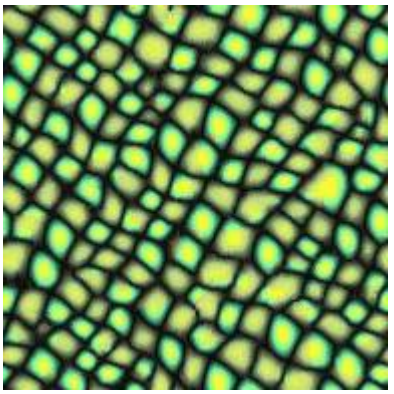
GSoC'22 Proposal

p5-texturesynth.js





Input Sample **Synthesised Result**

(src: http://graphics.stanford.edu/projects/texture/)

✱ Project Abstract

Texture Synthesis is the process of algorithmically constructing a large digital image from a small digital sample image by taking advantage of its structural content. This project would involve developing a texture synthesis library for artistic filling of the background in p5.js. The main focus is to introduce a simple, easy-to-use library to generate various textures. It would provide artists and game developers tools to fill textures in the objects in their sketches with ease.

✱ Personal Details

Name: Kartikeya Saxena
Github Handles: ktk53x, anexas5
Gitlab Handle: https://gitlab.com/anexa_s
Email ID: kartikeya.0005@gmail.com
Discourse handle: anexas
p5 editor sketches: https://editor.p5js.org/anexas/sketches

About Me

Overview

I am a Final Year undergraduate pursuing **B.Tech** in **Computer Science and Engineering** from the **Indian Institute of Technology, Guwahati (IITG), India**.
I am an experienced competitive coder and have solved a significant amount of problems on various platforms like **Leetcode**, **Codeforces**, **Atcoder** and **Codechef** with C++ as my primary language. Python and Javascript have been my secondary languages. I have good working experience with Git and GitHub.

Motivation

I was introduced to the world of creative coding by watching the Coding Train youtube channel by Daniel Shiffman. I was immediately hooked to his coding challenge video playlist and gained interest in generative art as a culmination of science, maths, coding and art. I enjoy creating p5.js sketches in my free time and have made a lot of sketches at https://editor.p5js.org/anexas/sketches. My views align with the organisation's mission to promote software literacy within the visual arts, and visual literacy within technology-related fields. This is my first time contributing to an open source organisation and I am really looking forward to contributing to the Processing Foundation's projects and learning more about them.

Projects

- **Data Mining [Incremental Clustering Algorithms]**
  I have worked with Prof. Amit Awekar as a part of a group project on developing incremental data mining algorithms. We have proposed a novel technique to the paper on SUBCLU by Karin Kailing, Hans-Peter Kriegel and Peer Kröger. We have proposed an algorithm that can take new data in and incrementally apply our novel technique combined with the static version and achieve a 20x speedup over the original static algorithm without much loss in accuracy compared to the static algorithm.
  [Work completed, might get published]
  Code, Datasets, Reports

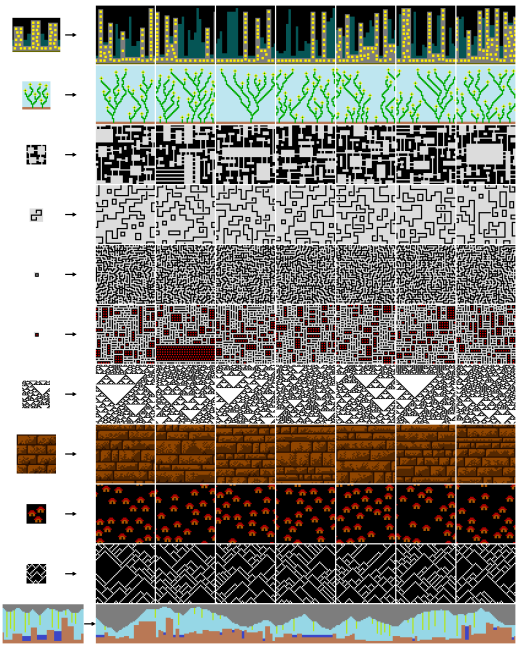
- **Routing in Lightning Network [Blockchain and Public Channel Network]**
  I am working with Prof. Kalpesh Kapoor as a part of my BTech thesis project on developing an efficient algorithm for routing in Lightning Networks. We have developed an algorithm which is an improved version of the earlier proposed Speedy Murmurs Algorithm by Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, Ian Goldberg and are currently simulating the performance of the algorithm compared to other state of the art algorithms.
  [Work under progress, might get published]

List of courses I have done so far as part of my BTech Programme: **Course Material**

Project Description

- **Why p5-texturesynth.js?**
  An artist creating digital media for a virtual world or a video game or in graphic design, often requires a large number of textures for these kinds of designs like concrete walls, fabrics, leaves etc. But the problem occurs when they wish to fill, say an entire road with a concrete texture but they only have a small patch at their disposal. The easiest solution is to repeatedly copy paste the patch in a tile fashion but this results in unsatisfactory design which suffers from seams.
  p5-texturesynth.js will try to resolve these problems by procedural generation of the texture using the input sample image in a satisfactory and elegant manner.

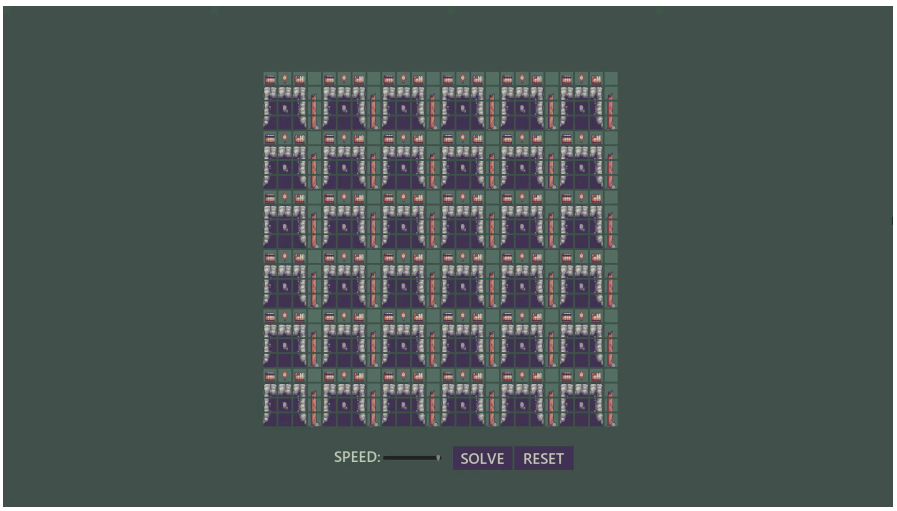
- **Algorithms**
  Texture synthesis is an active area of research and many algorithms have been developed for it. Following are some of the well performing practical algorithms in the literature.
  Wave Function Collapse:
  (by Maxim Gumin)



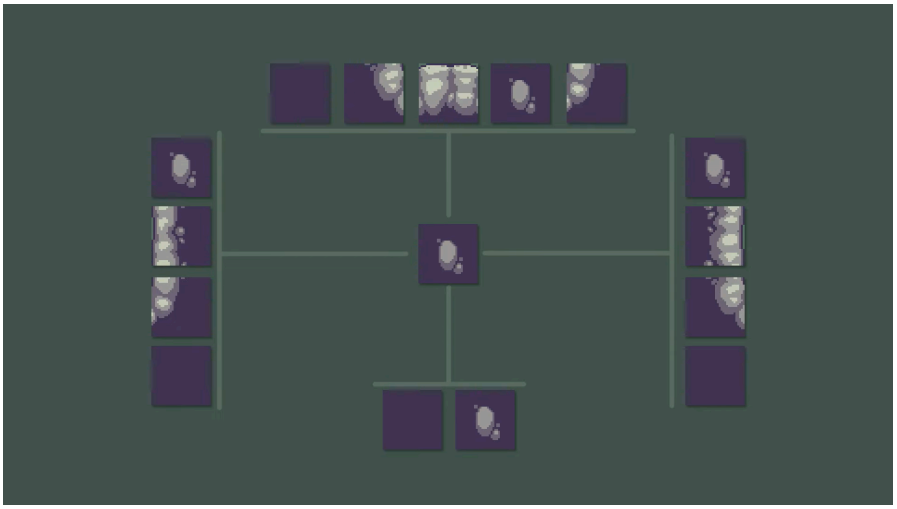
Sample output

(src: https://github.com/mxgmn/WaveFunctionCollapse)

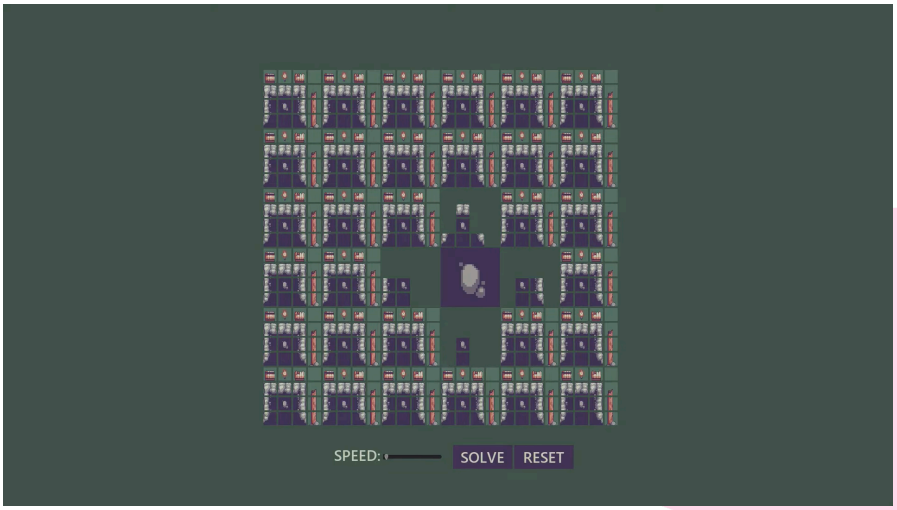
This texture synthesis algorithm is inspired from the concepts of quantum mechanics. In quantum mechanics, wave function collapse occurs when a wave function—initially in a superposition of several eigenstates—reduces to a single eigenstate due to interaction with the external world. This interaction is called an "observation".
In the context of textures, the algorithm starts with a grid of cells each occupying the superposition of all the possible states of that cell.



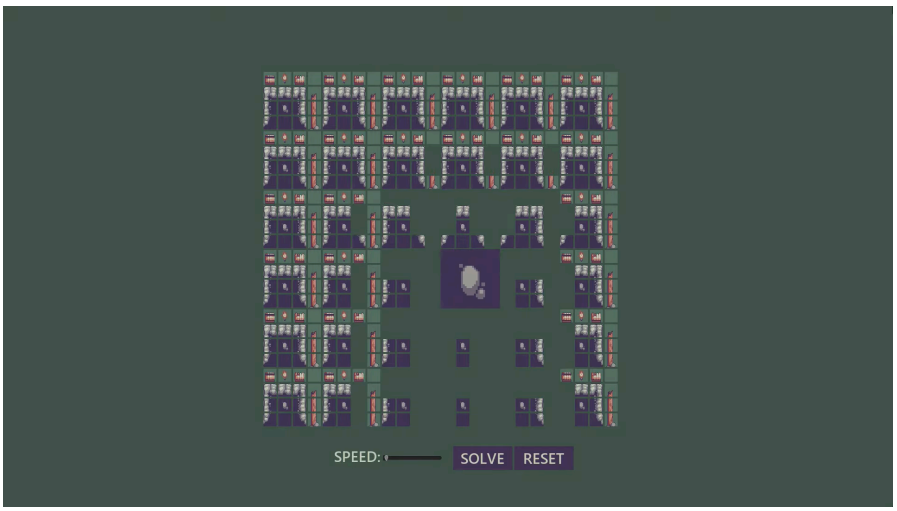
Each cell comes with a set of adjacency rules which only allow certain cells to be its neighbours.



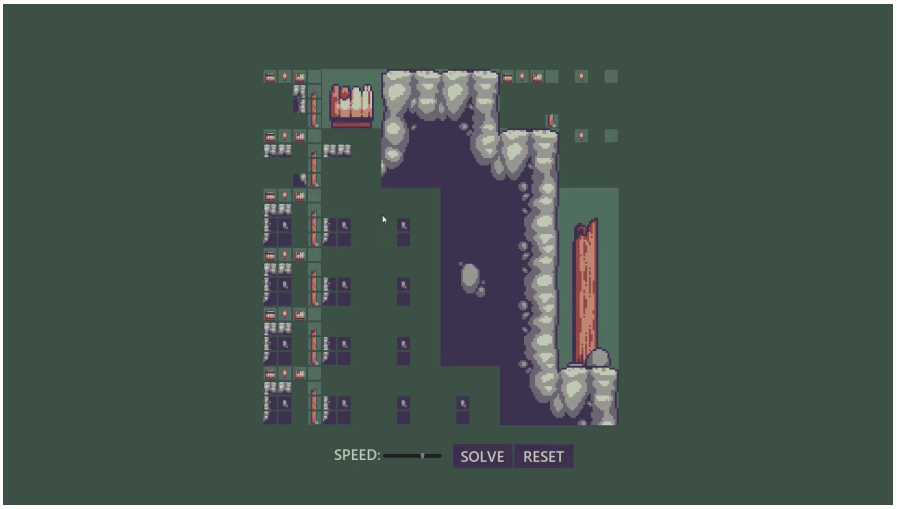
The wave function collapse algorithm randomly picks a cell with lowest entropy (lowest number of possible states) and collapses it into a single tile propagating the consequences of the adjacency rule to its neighbours.

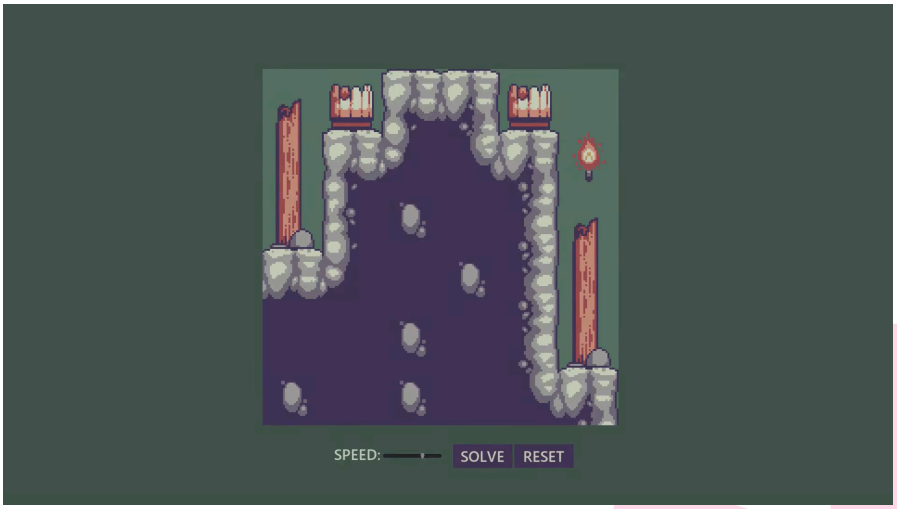


These affected neighbouring cells further propagate the consequences to their neighbours in a ripple like fashion.



This composes a single iteration of the algorithm and the algorithm continues until all cells contain a single tile generating a final texture.



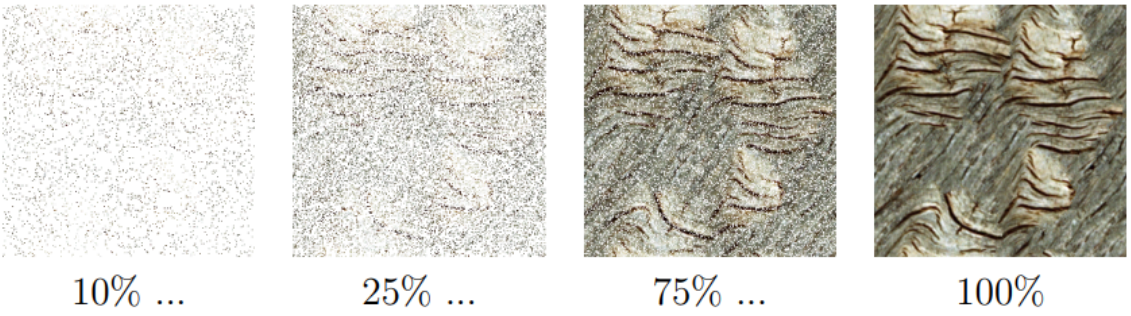


(src: https://www.youtube.com/watch?v=2SuvO4Gi7uY)

P.F. Harrison Algorithm

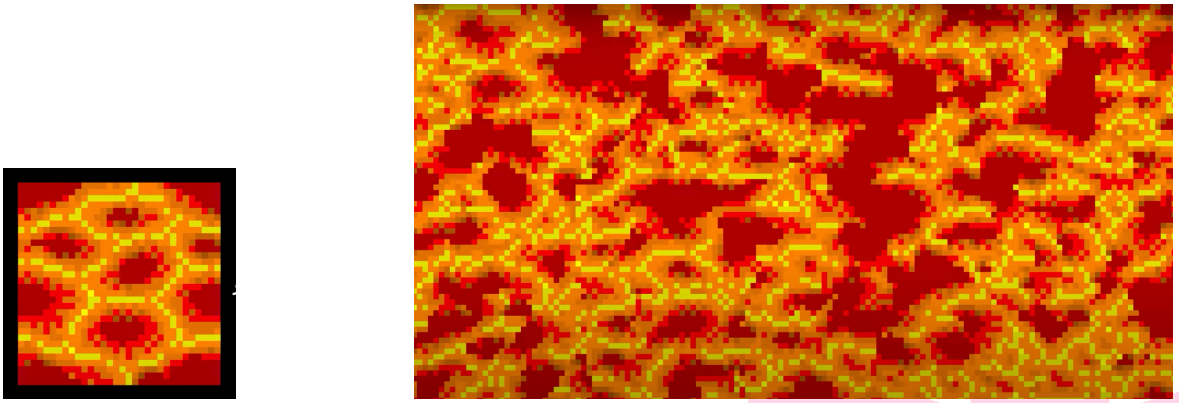
It is a practical simple and fast algorithm and is in current use in the form of an open-source GIMP plug-in.
The algorithm assumes that the input sample is taken from a Markov Random Field. If the image is a sample from a Markov Random Field, then the likelihood of the unknown pixel having a certain value may be determined solely from the values of a fixed neighbourhood of surrounding pixels. Knowledge of values beyond this neighbourhood provides no further information.

In this algorithm, pixel values are chosen one at a time in random order and their n nearest pixels that already have values are located. The input image is then searched for a good match to the pattern these pixels form and appropriate pixel values are copied from the match. Some early chosen pixel values are re-chosen and recomputed.



P.F. Harrison advancement
(src: https://logarithmic.net/pfh-files/thesis/dissertation.pdf)

When choosing each pixel value, only a small number of locations in the input texture are examined, yielding a considerable speed-up as compared to a search of the whole image. https://www.youtube.com/watch?v=8sUMBMpZNzk. This link consists of a nice visualisation of the way the algorithm works.



Input Sample **Synthesised Result**
(src: https://www.youtube.com/watch?v=8sUMBMpZNzk)

API

OutputImageTexture = texture(inputImageTexture, inputWidth, inputHeight, outputWidth, outputHeight, algorithm, [options])

Example Sketch

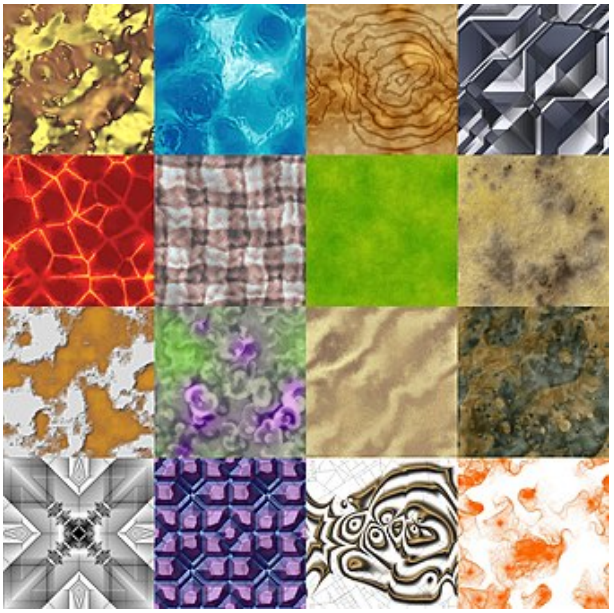
```
let inputImg;
let outputImg;
let options;

function preload() {
      inputImg = loadImage('input.png');

}

function setup() {
     createCanvas(400, 400);
      options = {
            algorithm: "wfc",
            tileSize: 3,
            periodic: false,
            neighbours: {
                  0: [-1, 0],
                  1: [1, 0],
                  2: [0, 1],
                  3: [0, -1]
            }
      };
      outputImg = texture(inputImg, inputImg.width, inputImg.height,
width, height, "wfc", options);
      image(outputImg, 0, 0);
}
```

Work Timeline

Phase/Week	Dates	Work Description
Community Bonding		
Community Bonding Period	May 20 - May 27	Understand the existing codebase of p5.js, figure out the parts of the code base necessary for building the library and being active in the community.
	May 28 - June 4	Reviewing existing algorithms (like Fast texture synthesis using Tree-structured Vector Quantization, Chaos Mosaic, Graphcut textures, P.Harrison, Wave function collapse, Multiresolution Stochastic Texture Synthesis) and its implementations in literature and finalising the algorithms and features for the library.
	June 5 - June 12	Discuss with the mentor the best way to go about the implementation.
Phase 1		
Week 1	June 12 - June 19	Breaking the goals of my project to several small trackable issues for better analysis of progress and milestones.
Week 2 & 3	June 20 - July 4	These two weeks would involve implementation of the wave function collapse and P.F. Harrisons algorithm along with automated test cases.
Week 4	July 5 - July 12	Updating documentation for the algorithms implemented and API functions.
Week 5	July 13 - July 24	Reviewing existing algorithms for additional features and its implementations in literature.
Phase 1 Evaluations July 12-July 16		
Phase 2		
Week 6	July 25 - Aug 1	Starting with the implementation of the additional features finalised.
Week 7	Aug 2 - Aug 9	Continue working on additional features and algorithms discussed along with automated test cases.
Week 8	Aug 10 - Aug 17	Updating documentation for the algorithms implemented and API functions.
Week 9	Aug 17 - Aug 24	Tutorial building and example creation of the algorithms implemented in the library.
Week 10	Aug 24 - Sep 4	Final touchup on the library. Making the final GSOC project Report.
Final Evaluations Aug 23 -Aug 30		

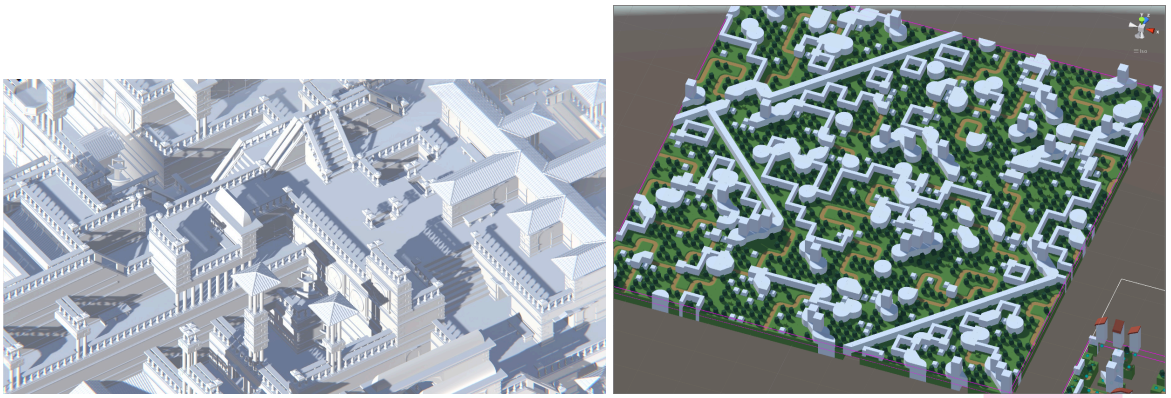
★ Additional Features (to be discussed)

1. This library can be improved further by exploring more texture synthesis algorithms and could also be extended to perform procedural texture generation without any input digital image. There are many physical and biological phenomena that can be simulated to create an entire texture without any initial input patch.



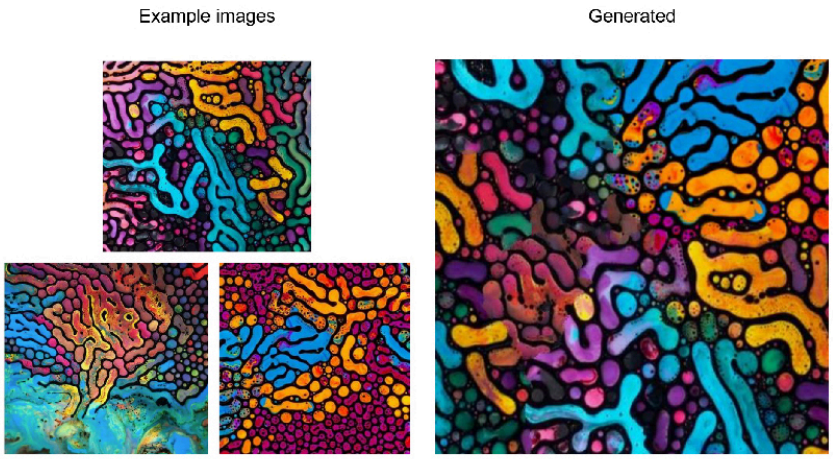
Procedurally generated tiling textures
(src: https://en.wikipedia.org/wiki/Procedural_texture)

2. Parallelisation of the discussed algorithms to improve performance.
3. The algorithms discussed above can be extended to create 3D texture using a sample 3D texture sample.



3D texture synthesis
(src: Marian42, Selfsame)

4. Multi example synthesis involving multiple image patches remixed to generate a single large texture.



Multi source texture synthesis
(src: Anastasia Opara)

★ Reference

- Texture Synthesis Wiki - https://en.wikipedia.org/wiki/Texture_synthesis
- Procedural Texture Wiki - https://en.wikipedia.org/wiki/Procedural_texture
- Wave Function Collapse - https://www.youtube.com/watch?v=2SuvO4Gi7uY
- Resynthesis algorithm of P.F. Harrisons algorithm - https://logarithmic.net/pfh-files/thesis/dissertation.pdf

Looking forward to working with the **Processing Foundation** on this!!

Literature Survey:

Parametric:
https://www.cns.nyu.edu/heegerlab/content/publications/Heeger-siggraph95.pdf
https://www.cns.nyu.edu/pub/lcv/portilla99-reprint.pdf

Non Parametric:
https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/papers/efros-iccv99.pdf
https://people.eecs.berkeley.edu/~efros/research/quilting/quilting.pdf