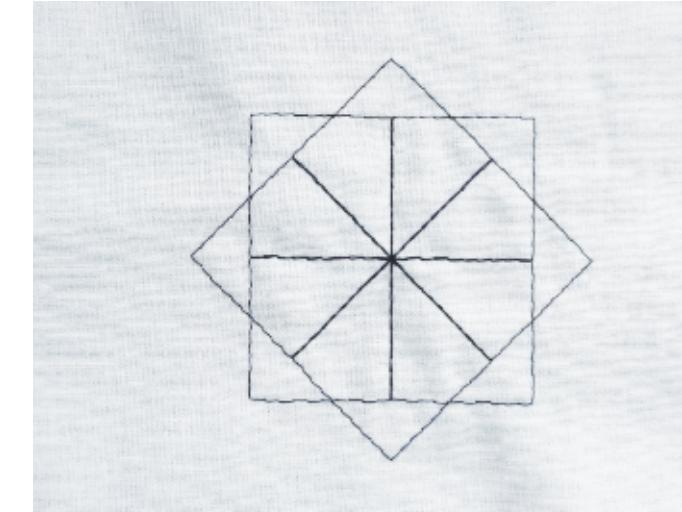
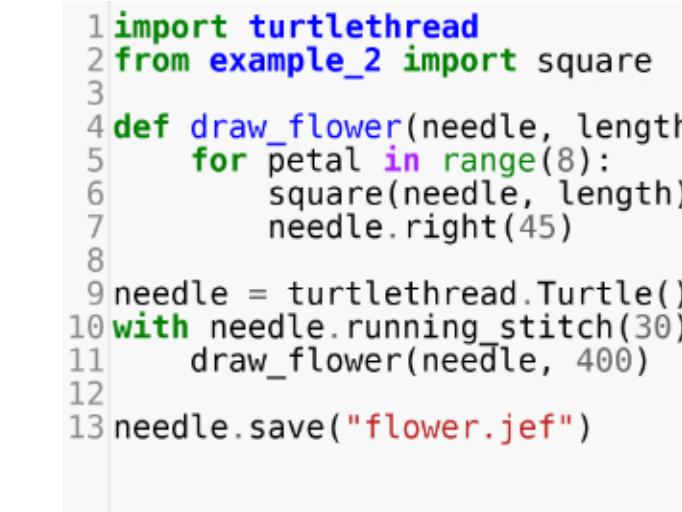
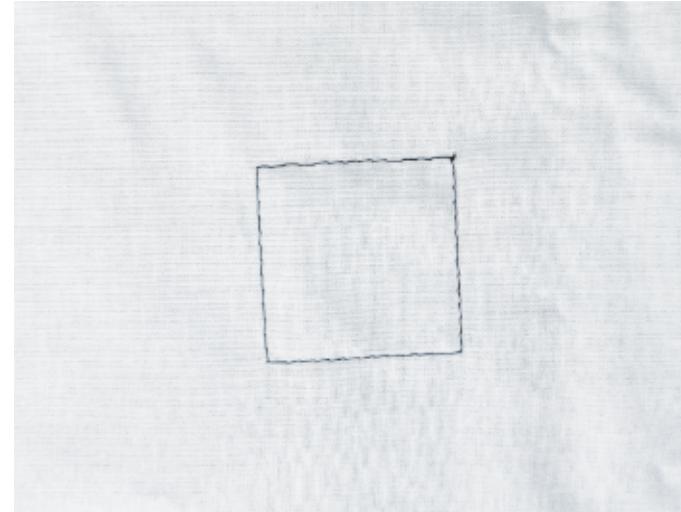
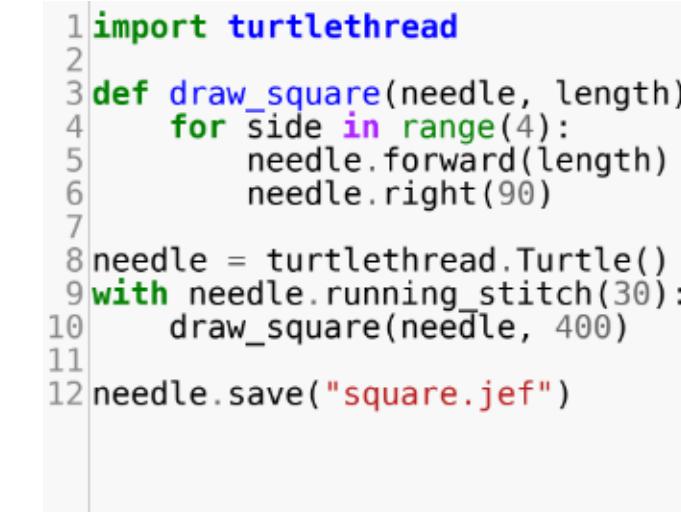
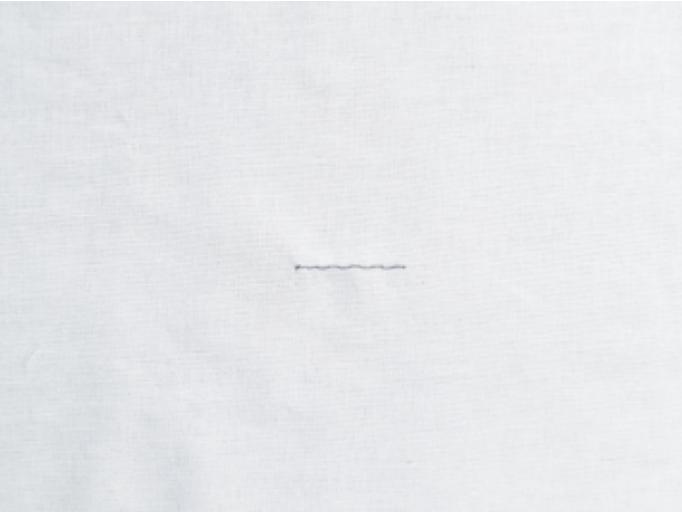
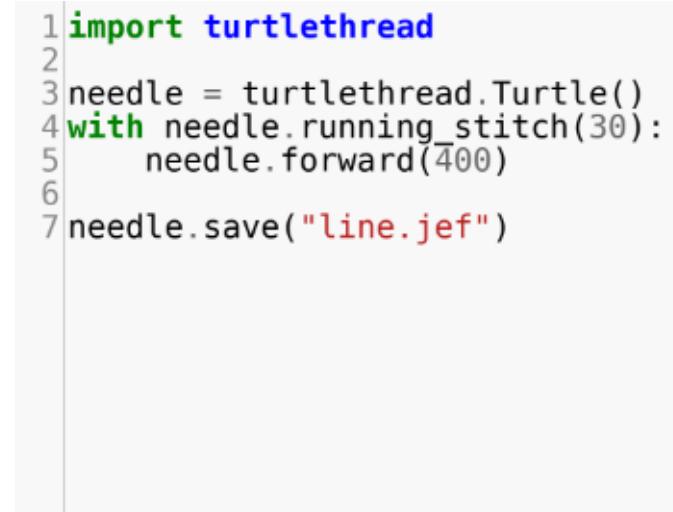
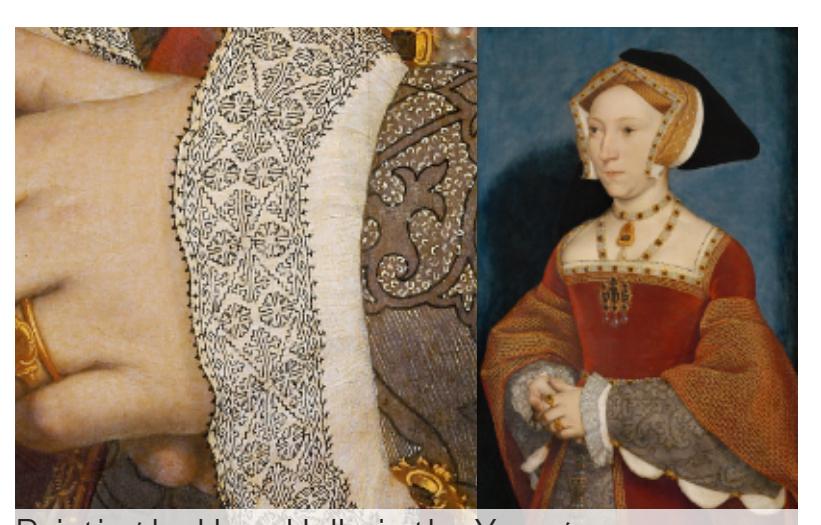
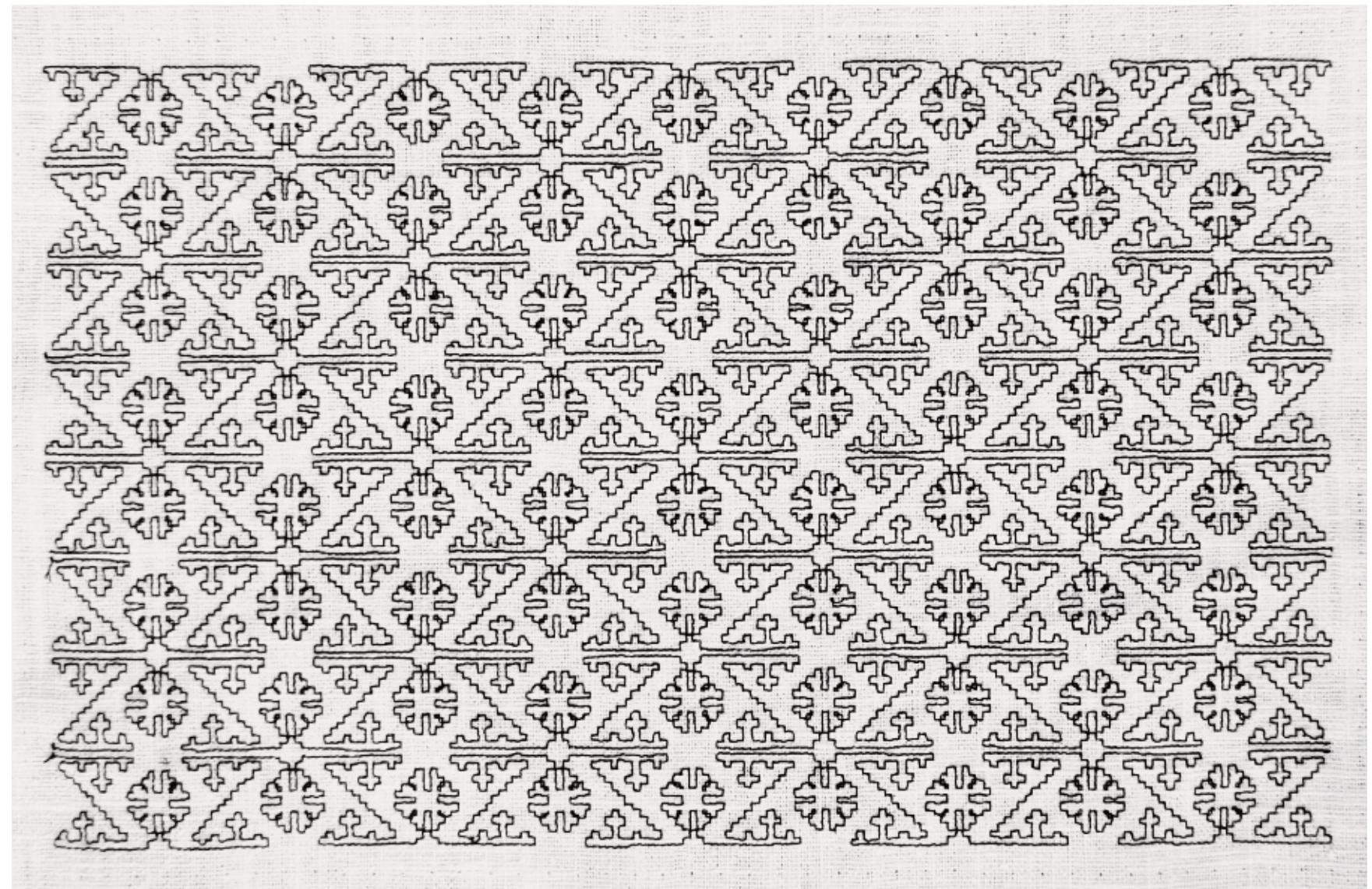
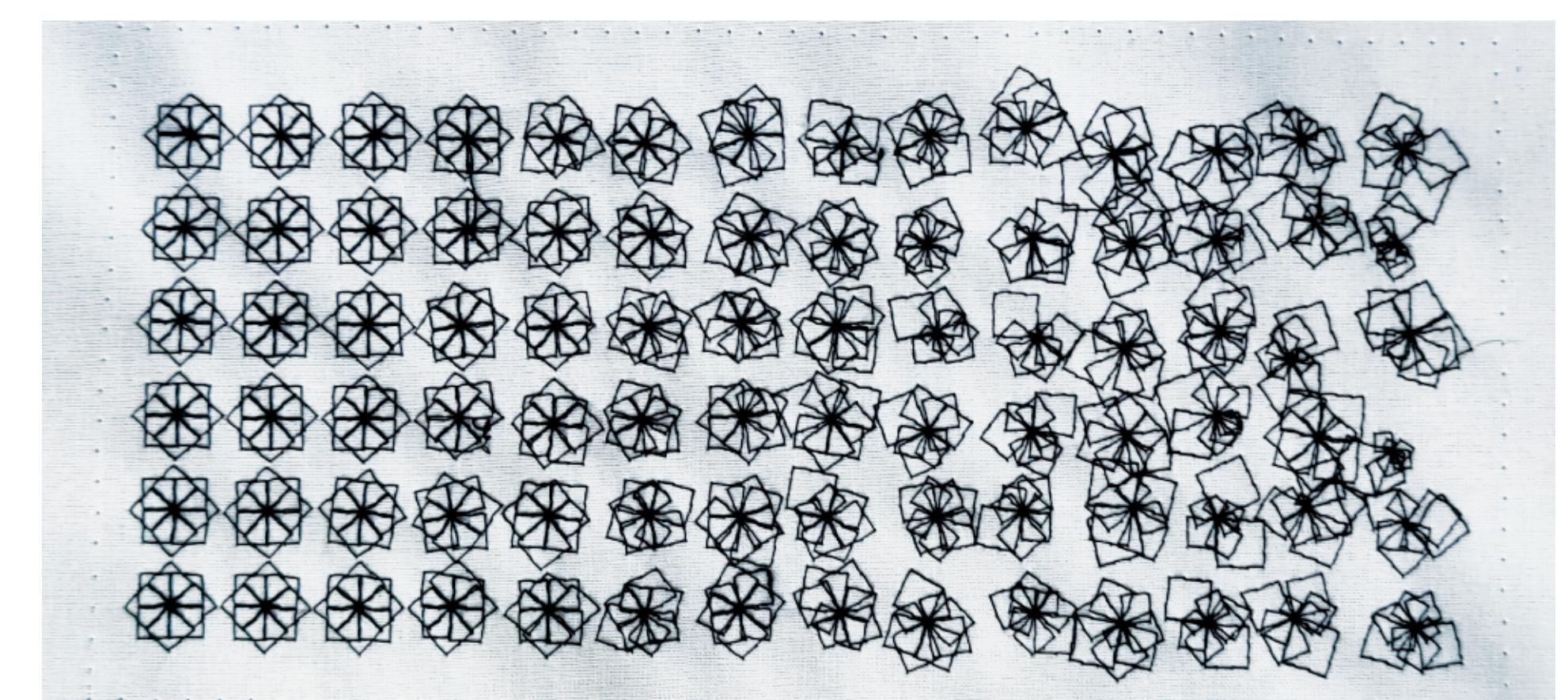


The creative art of algorithmic embroidery

Marie Roald & Yngve Mardal Moe



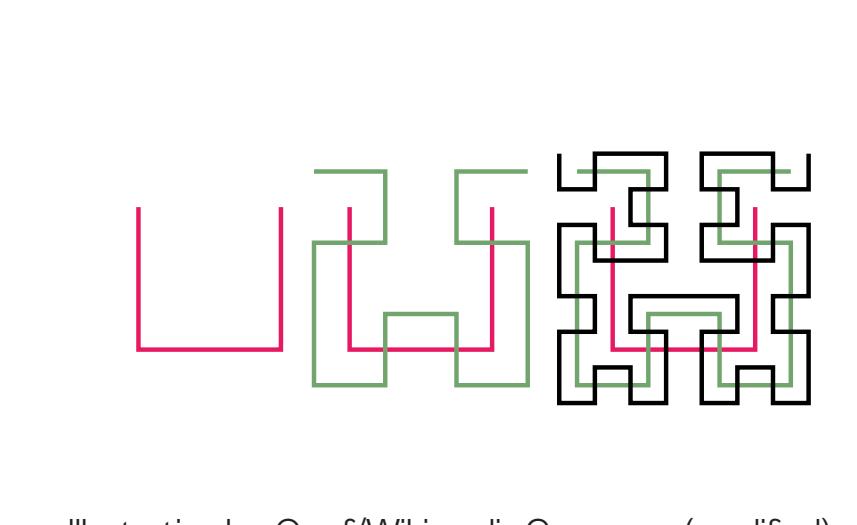
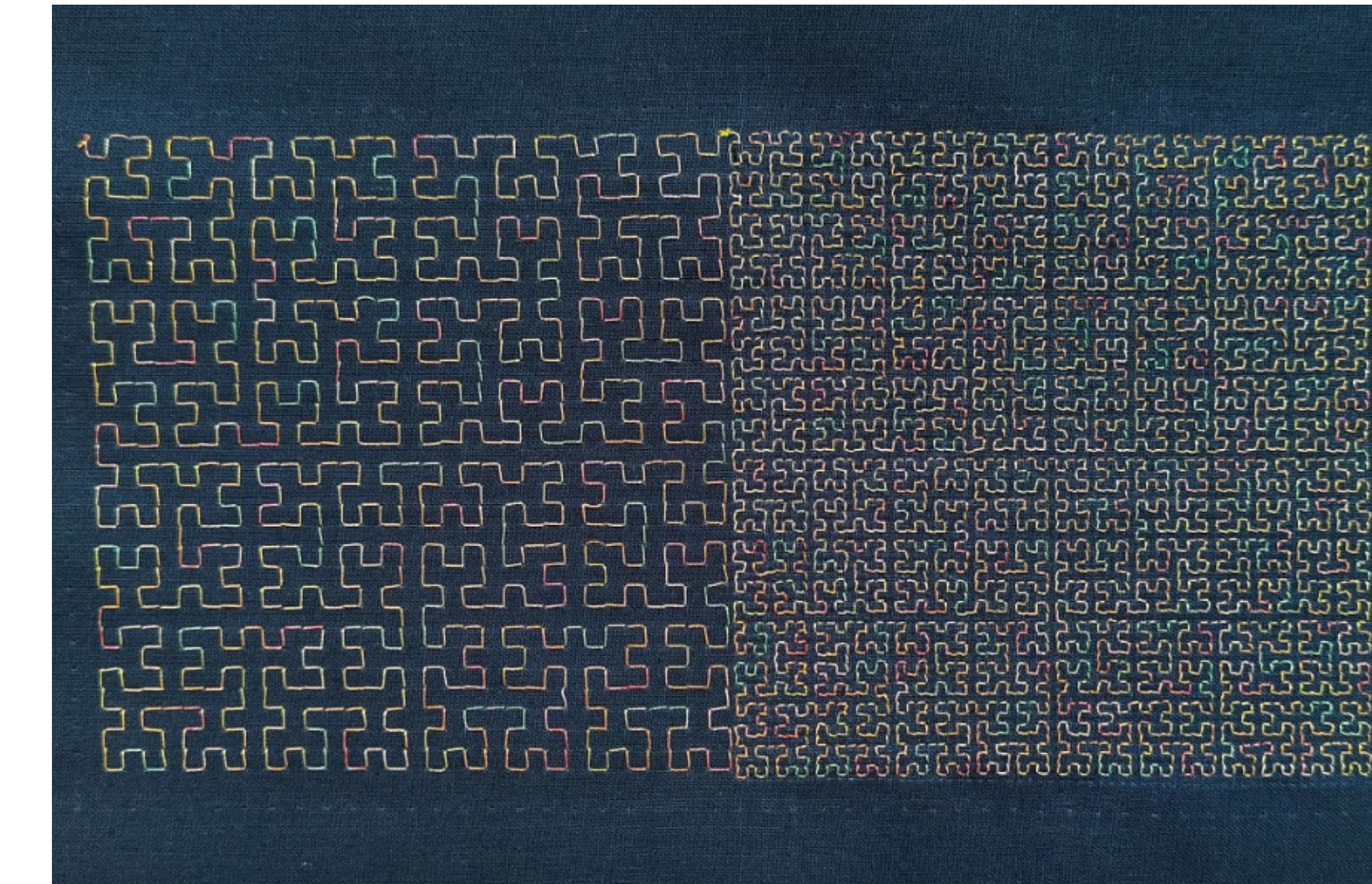
```
1 import turtlethread
2 from random import gauss
3 from example_2 import draw_square
4
5 def random_flower(needle, length, noise):
6     for petal in range(5):
7         # Choose random size and angle
8         l = gauss(length, noise)
9         a = gauss(0, 360)
10        needle.forward(l)
11        draw_square(needle, l)
12        needle.right(a)
13
14 needle = turtlethread.Turtle()
15 for x in range(0, 1951, 150):
16    n = 0.4 * (x / 1950)**1.1
17    for y in range(-150, 150):
18        for i in range(5):
19            d = gauss(0, 50*n)
20            da = gauss(0, 50*n)
21            needle.forward(d)
22            needle.left(da)
23            da = gauss(0, 90*n)
24            # Move to random offset
25            with needle.running_stitch(30):
26                needle.goto(x + dx, y + dy)
27                needle.setheading(da)
28
29    # Embroider random flower
30    with needle.running_stitch(25):
31        random_flower(needle, 50, n)
32
33 needle.save("field_schotter.jef")
```



Blackwork is a counted thread embroidery where a black thread is used on white linen fabric. It was prevalent among British royals in the late 15th-16th century.

Early blackwork is very geometric, and this code example forms a pattern inspired by that on Jane Seymour's cuff. Three motifs are repeated and mirrored to create the design: a staircase and a cross and a flower.

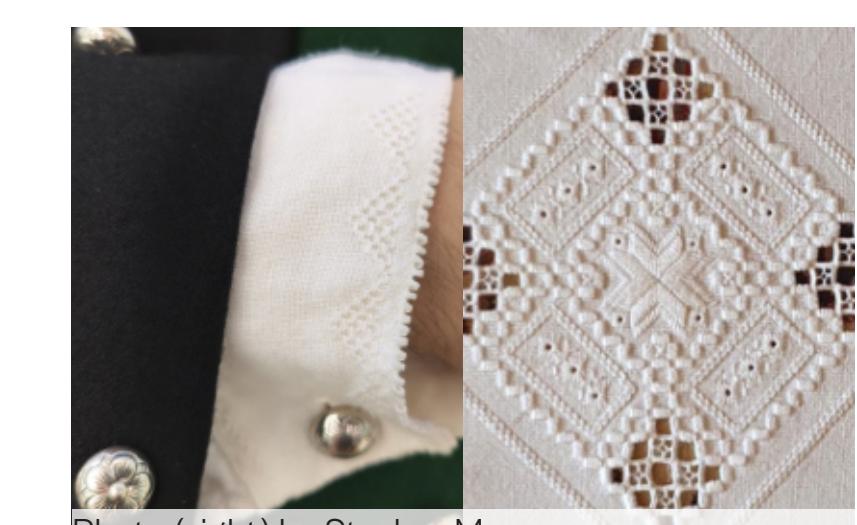
```
1 import turtlethread
2 # The following imports are available on GitHub
3 from blackwork_utils import (
4     draw_staircase,
5     draw_staircase,
6     draw_flower,
7     )
8
9
10 def draw_column(pen, unit, d, length):
11     for n in range(1, length):
12         if n % 2 == 0:
13             draw_staircase(pen, unit)
14         else:
15             draw_staircase(pen, unit)
16         pen.forward(unit**2)
17
18     # Mirror symmetry
19     for n in range(1, d):
20         direction = d**m
21
22         # Straight line
23         pen.forward(1**2*unit)
24         pen.left(90)
25         pen.forward(1**2*unit)
26         pen.forward(1**2*unit)
27         pen.left(90*direction)
28
29     # Staircase and crosses
30     draw_staircase(pen, unit, direction)
31     draw_staircase(pen, unit, direction)
32     draw_staircase(pen, unit, direction)
33     draw_staircase(pen, unit, direction)
34
35     # Flower separator
36     if m == 1:
37         pen.forward(unit**2)
38         draw_flower(pen, unit, -direction)
39         pen.forward(unit**2)
40
41     # Flower
42     if m == 2:
43         pen.forward(unit**2)
44         draw_flower(pen, unit, -direction)
45         pen.forward(unit**2)
46
47     # Add space
48     if m == 3:
49         pen.forward(unit**2)
50         draw_flower(pen, unit, -direction)
51         pen.forward(unit**2)
52
53     unit = 12
54
55 pen = turtlethread.Turtle()
56 pen.jump(0)
57 for offset in range(4):
58     if offset == 0:
59         with pen.jump_stitch():
60             pen.goto(0 + 2 * offset * 14 * unit, 0)
61     # Draw left column
62     with pen.running_stitch(1**2):
63         draw_column(pen, unit, 1, 5)
64
65     # Move to top
66     with pen.jump_stitch():
67         pen.goto(0 + 2 * offset * 14 * unit, 0)
68
69     # Draw right column
70     with pen.jump_stitch(1**2):
71         draw_column(pen, unit, -1, 5)
72
73 pen.save("blackwork.jef")
```



The **Hilbert curve** is a recursive curve that fills a square without crossing itself. It has several interesting properties and is used in various applications in computer science.

To construct the Hilbert curve, the code uses a recursive function that replaces each line with a new Hilbert curve. The embroidery pattern consists of recursion levels 5 (left) and 6 (right).

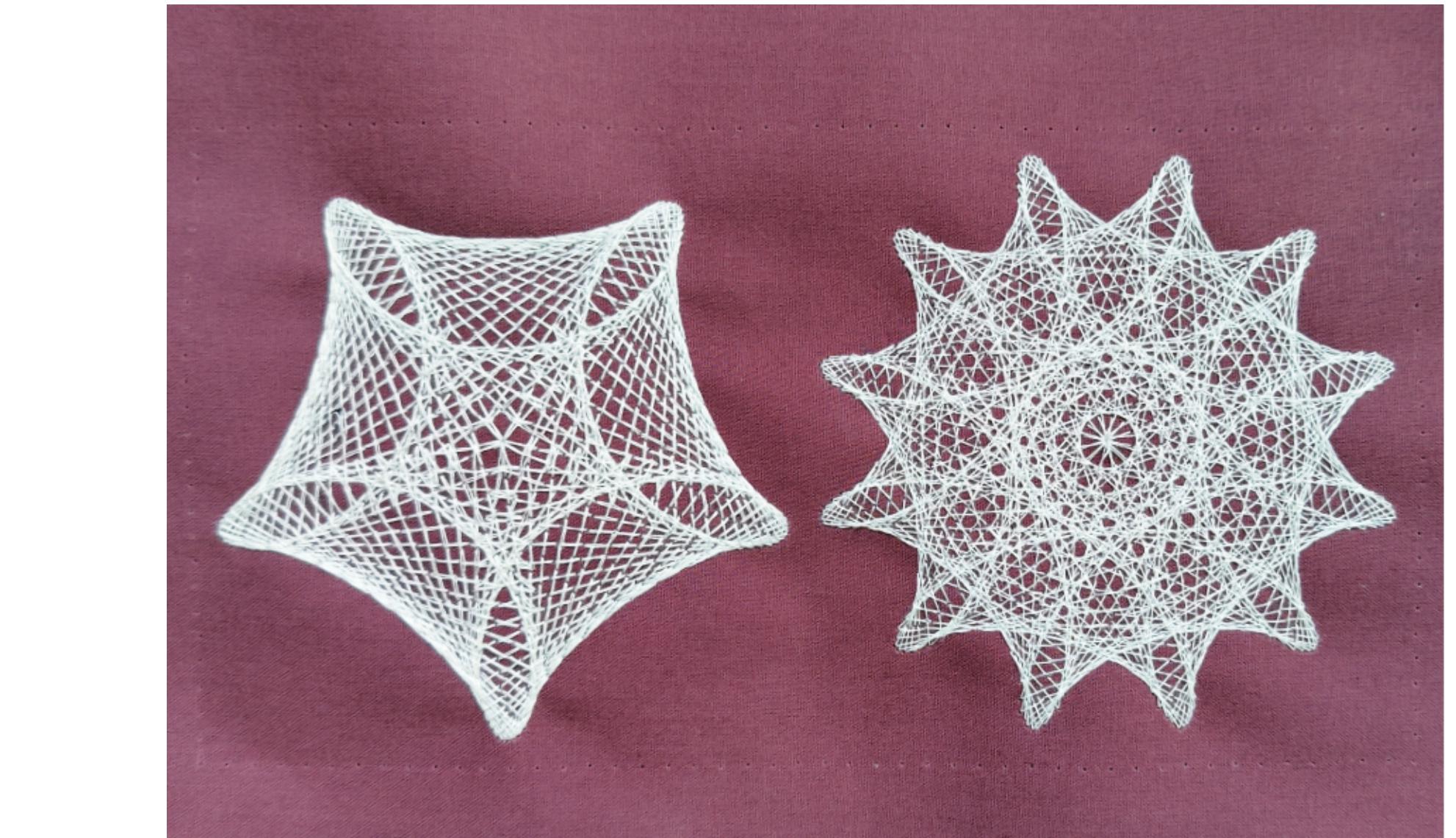
```
1 import turtlethread
2 # The following import is available on GitHub
3 from whitework_utils import draw_parallelogram
4
5 def hilbert(pen, length, n, angle=90):
6     if n == 0:
7         pen.forward(length)
8         pen.right(angle)
9         pen.forward(length)
10    hilbert(pen, length, n-1, -angle)
11    pen.left(angle)
12    hilbert(pen, length, n-1, angle)
13    pen.forward(length)
14    hilbert(pen, length, n-1, angle)
15    pen.left(angle)
16    pen.forward(length)
17    hilbert(pen, length, n-1, -angle)
18    pen.right(angle)
19
20    # Add space
21    pen.forward(18)
22
23 pen = turtlethread.Turtle()
24
25 # Draw first, low resolution, curve
26 with pen.running_stitch(1**2):
27     hilbert(pen, 2**6, 5)
28
29    # Flower separator
30    if m == 1:
31        pen.forward(18)
32
33    # Draw second, high resolution, curve
34    with pen.running_stitch(18):
35        hilbert(pen, 18, 5)
36
37    pen.save("hilbert.jef")
```



White shapes embroidered on white cloth characterise **Norwegian whitework**, which is found, among other places, on decorative cloth and the shirts of the Norwegian national dress,

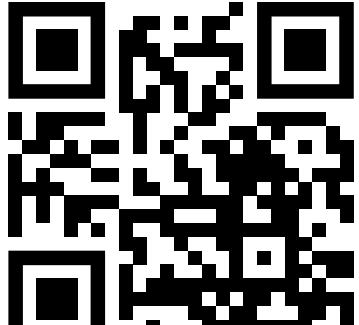
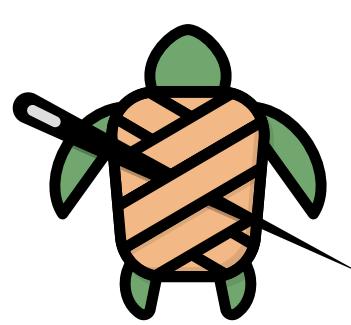
This example is inspired by a shirt cuff and a motif popular in Norway: eight parallelograms flipped and rotated to form a flower. The code repeats square corners to make the border and repeats a tight zigzag pattern to fill the parallelograms.

```
1 import turtlethread
2 # The following import is available on GitHub
3 from math import pi, sin, cos
4
5 def draw_rose(needle, n, d, r0):
6     theta = 0
7     for k in range(0, 360, 1):
8         r = r0 * sin(k * pi / 180)
9         x = r * cos(k * pi / 180)
10        y = r * sin(k * pi / 180)
11        needle.goto(x, y)
12
13 needle = turtlethread.Turtle()
14
15 # Draw first rose
16 with needle.running_stitch(50):
17     draw_rose(needle, 5, 57, 570)
18
19 # Move a bit more than two radii
20 with needle.running_stitch(150):
21     needle.forward(150)
22
23 # Draw second rose
24 with needle.running_stitch(50):
25     draw_rose(needle, 6, 71, 570)
26
27 needle.save("mauer_rose.jef")
```



The **Maurer rose**, named after Peter M. Maurer, who first introduced it, is a polar graph made up of lines connecting points on a rose curve, forming a rose-like pattern. By adjusting the rotation angle and scaling, we can get a variety of expressions demonstrating how simple mathematical operations can lead to beautiful and intricate designs.

The embroidery above shows roses with $n=5$, $d=97$ (left) and $n=6$, $d=71$ (right).



Find more examples and the documentation at:

turtlethread.com

All code examples from this poster are available at
github.com/TurtleThread/PyConUS24_poster

TurtleThread is a Python package inspired by *Turtlestitch* and powered by *pyembroidery* that lets you create embroidery patterns with turtle commands and context managers. This poster shows some examples that use loops, variables and recursion to create patterns inspired by traditional embroidery and mathematics

REFERENCES

Batsford Encyclopaedia of Embroidery Techniques. Swift, Gay (1984)

Blackwork Embroidery. Geddes, Elisabeth & McNeill, Moyra (1976)

The Art of Blackwork Embroidery. Drysdale, Rosemary (1975)

A Rose is a Rose... Maurer, Peter M (1987). DOI: 10.2307/2322215

Space-Filling Curves. Bader, Michael (2013). DOI: 10.1007/978-3-642-31046-1