Pattern Creator

Simon Widmann

Problem

- ECP limited to simple shapes
- Small file sizes required

Idea

- Draw arbitrary geometrical shapes
- Shapes represented as parametrizations of closed curves
- Use low amount of rectangles to fill the shapes
- Implement as an easy to use Python package

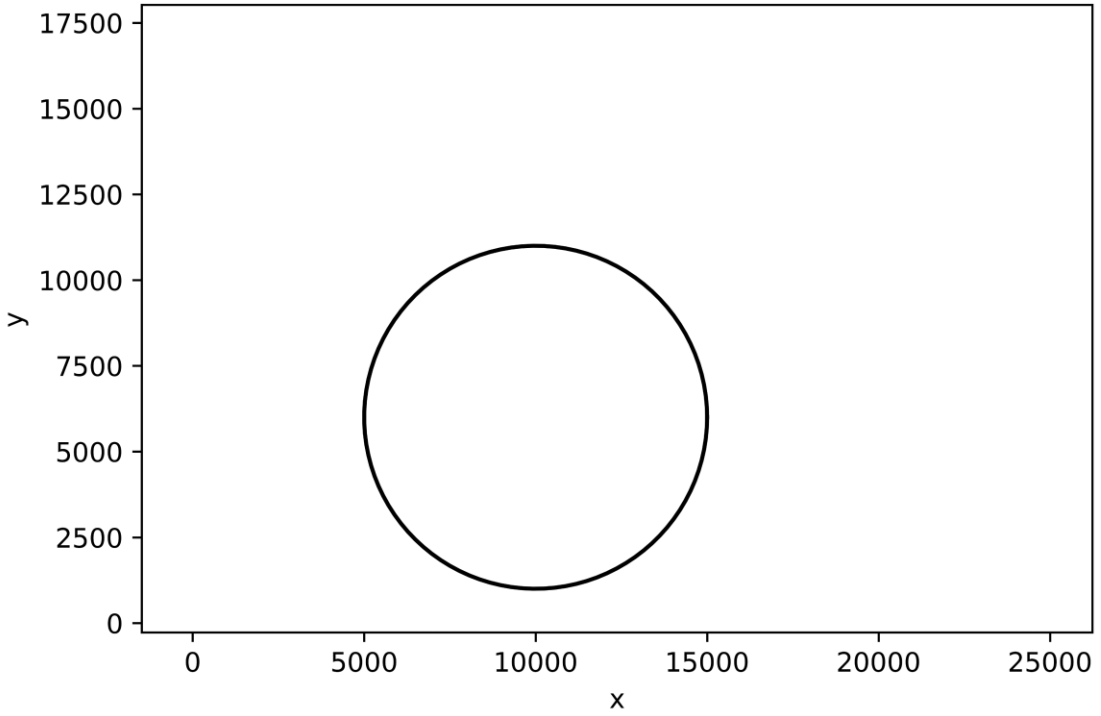
Definition of shapes

Parametrization of a **closed** curve $\vec{x}(t)$

Example:

$$\vec{x}_{\text{Circle}}(t) = \begin{pmatrix} x_0 + r \cdot \cos(2\pi t) \\ y_0 + r \cdot \sin(2\pi t) \end{pmatrix}$$

with $t \in [0, 1)$.



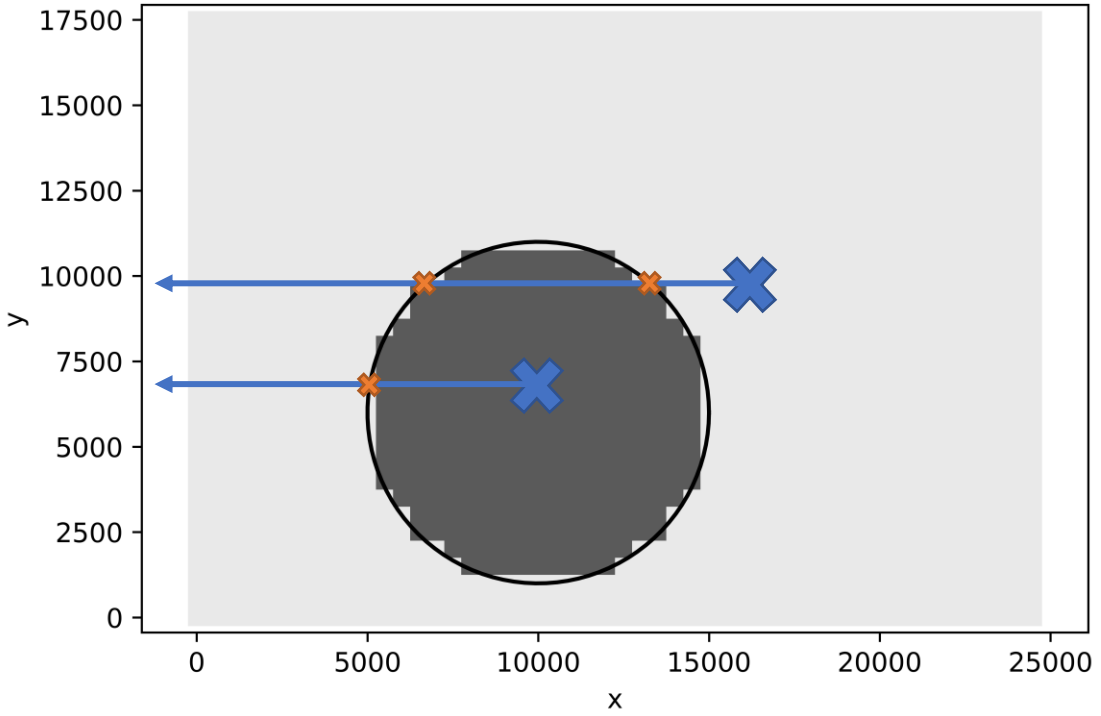
Translation of the shape into pixels

$$\vec{x}_{\text{Circle}}(t) = \begin{pmatrix} x_0 + r \cdot \cos(2\pi t) \\ y_0 + r \cdot \sin(2\pi t) \end{pmatrix}$$

- Black / White image
- Pixel as the smallest possible rectangle
- $\mathbf{1} \rightarrow$ pixel **is** part of a shape, $0 \rightarrow$ pixel **is not** part of a shape

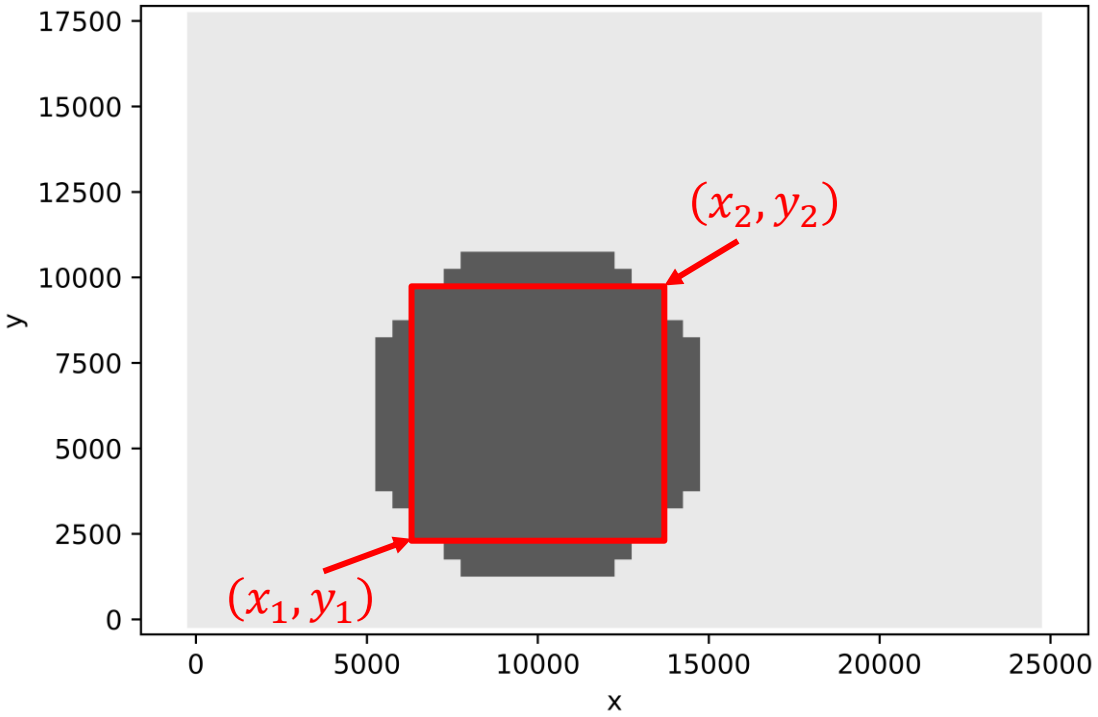
Raycasting method

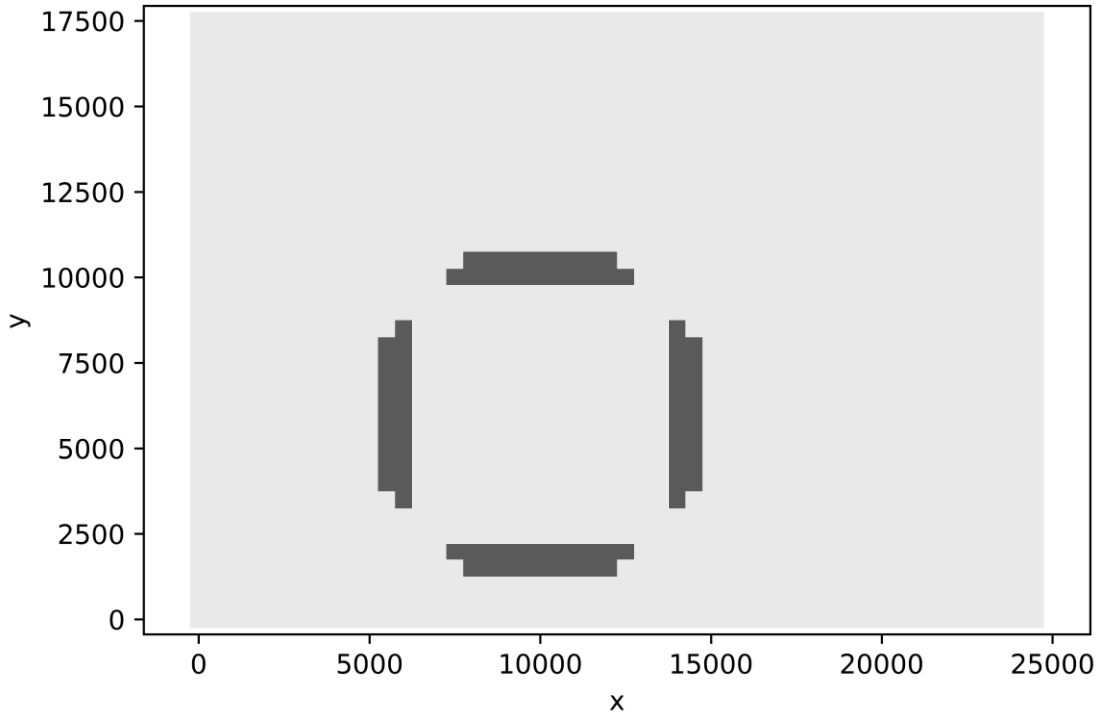
- Check for each pixel $\vec{p} = (x,\ y)$, if a ray going from $(x,\ y)$ to $(-\infty, y)$ intersects $\vec{x}(t)$ and odd (even) amount of times
- Odd (even) → Pixel is (not) inside of the shape



Grouping pixels into larger rectangles

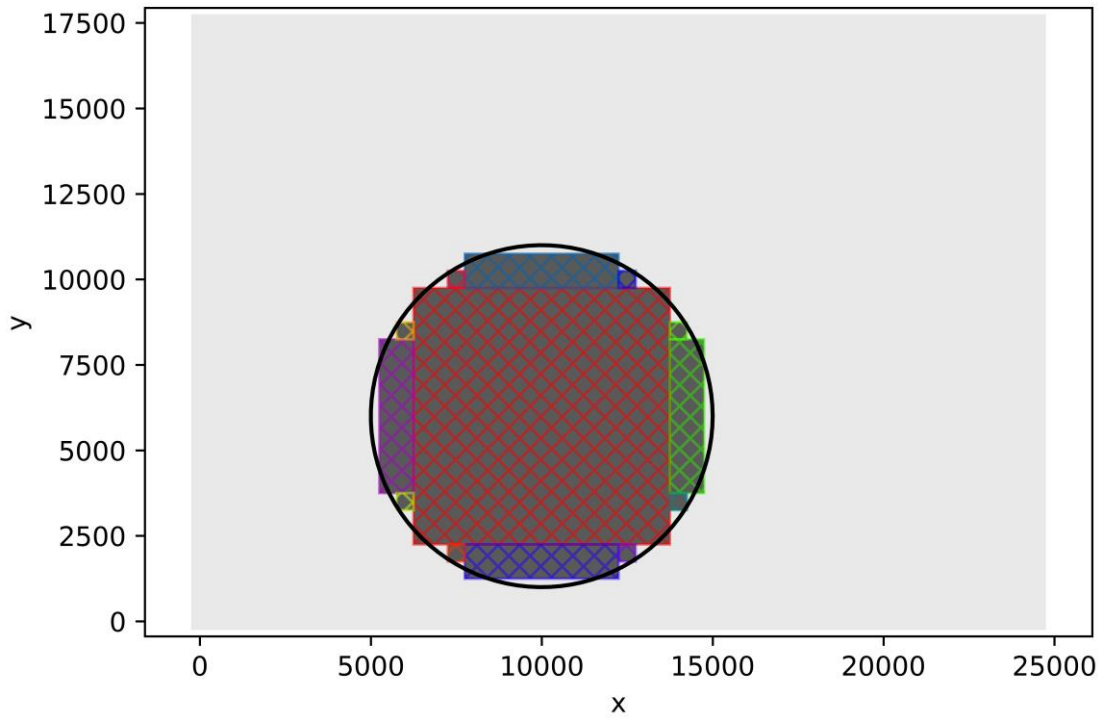
- Find the largest rectangle that fits into the shape in the image
- Save coordinates of two opposite corners $(x_1,\ y_1,\ x_2,\ y_2)$
- Remove the found rectangle from the image
- Repeat





Grouping pixels into larger rectangles

- Repeat until no pixel is left
- Convert rectangle corner coordinates into ECP code

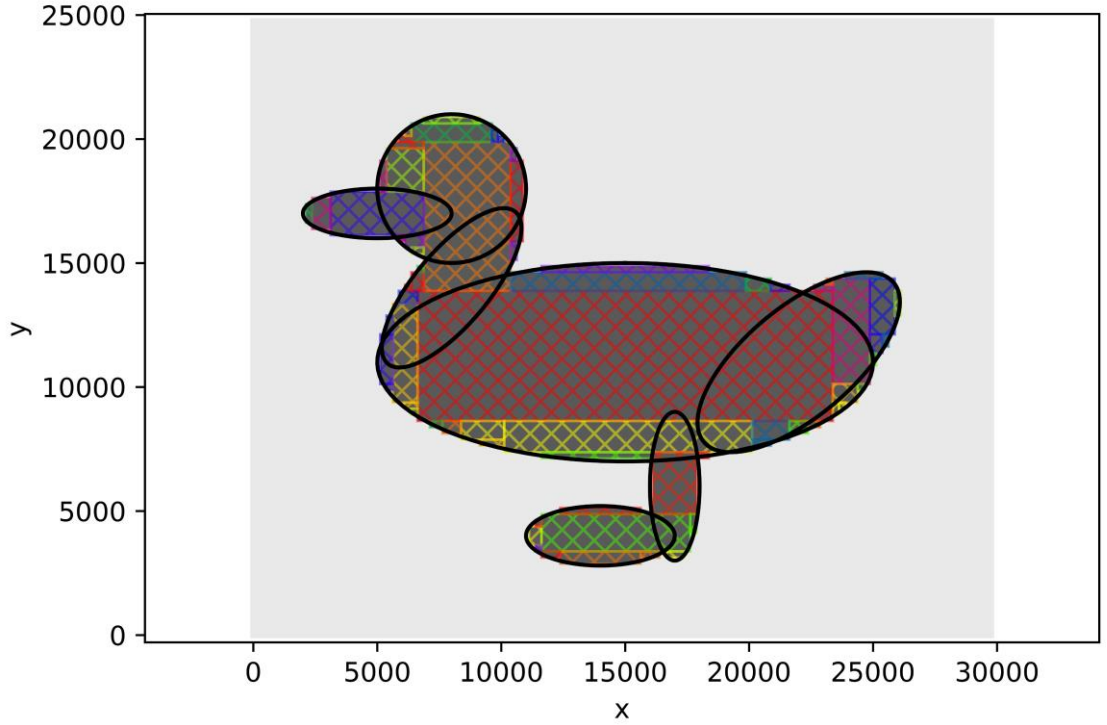


ECP .pat code

```
RECT 6250, 2250, 13750, 9750
RECT 5250, 3750, 6250, 8250
RECT 7750, 1250, 12250, 2250
RECT 7750, 9750, 12250, 10750
RECT 13750, 3750, 14750, 8250
RECT 5750, 3250, 6250, 3750
RECT 5750, 8250, 6250, 8750
RECT 7250, 1750, 7750, 2250
RECT 7250, 9750, 7750, 10250
RECT 12250, 1750, 12750, 2250
RECT 12250, 9750, 12750, 10250
RECT 13750, 3250, 14250, 3750
RECT 13750, 8250, 14250, 8750
```

More complicated shapes

- Combination of multiple curves
- Overlap is automatically supported
- Higher resolution with fast computation time



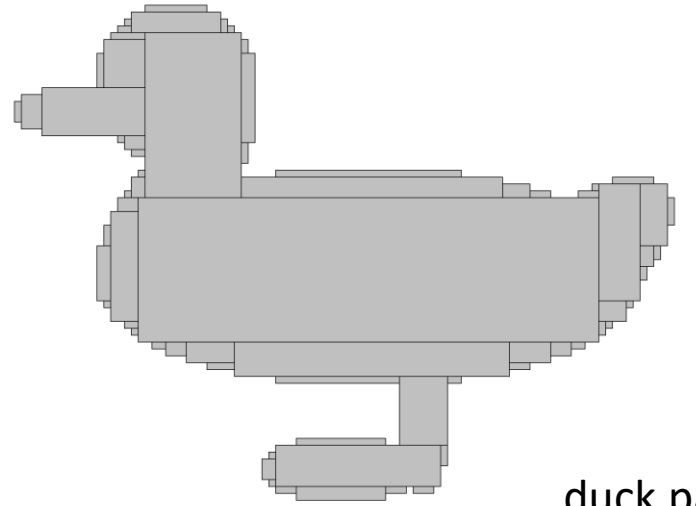
- Short code:

```python
from generate_pattern import Pattern, circle, ellipse
from math import pi


pattern = Pattern(3e4, 2.5e4, 250)
pattern.add_parametrized_shape(circle, 8e3, 18e3, 3e3)
pattern.add_parametrized_shape(ellipse, 1.5e4, 1.1e4, 0.4e4, 1e4, 90 / 180 * pi)
pattern.add_parametrized_shape(ellipse, 8e3, 14e3, 0.15e4, 0.4e4, -40 / 180 * pi)
pattern.add_parametrized_shape(ellipse, 5e3, 17e3, 0.1e4, 0.3e4, 90 / 180 * pi)
pattern.add_parametrized_shape(ellipse, 17e3, 6e3, 1e3, 3e3, 0 / 180 * pi)
pattern.add_parametrized_shape(ellipse, 14e3, 4e3, 1.2e3, 3e3, 90 / 180 * pi)
pattern.add_parametrized_shape(ellipse, 22e3, 11e3, 2.2e3, 5e3, -50 / 180 * pi)
pattern.visualize()
print(pattern.export_pattern())


```

ECP:



duck.pat