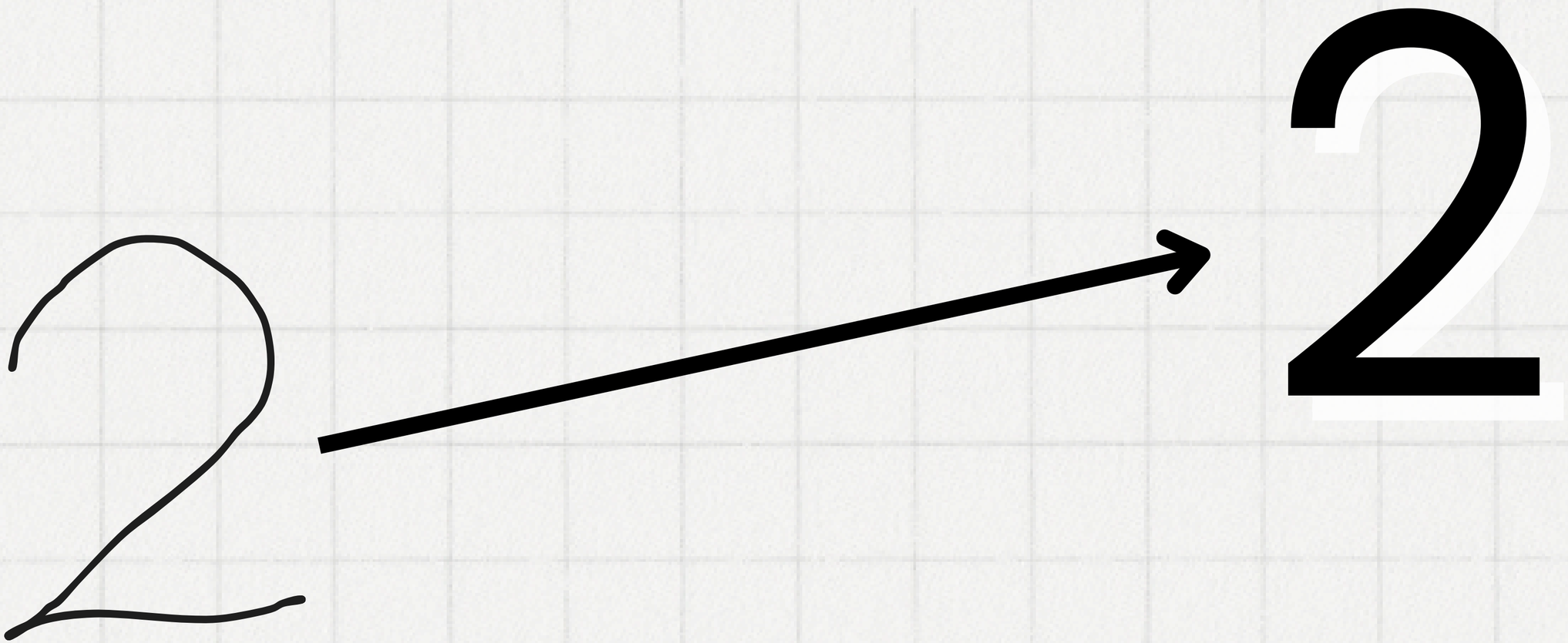
Handwritten Character Recognition

Made by Sescu Matei

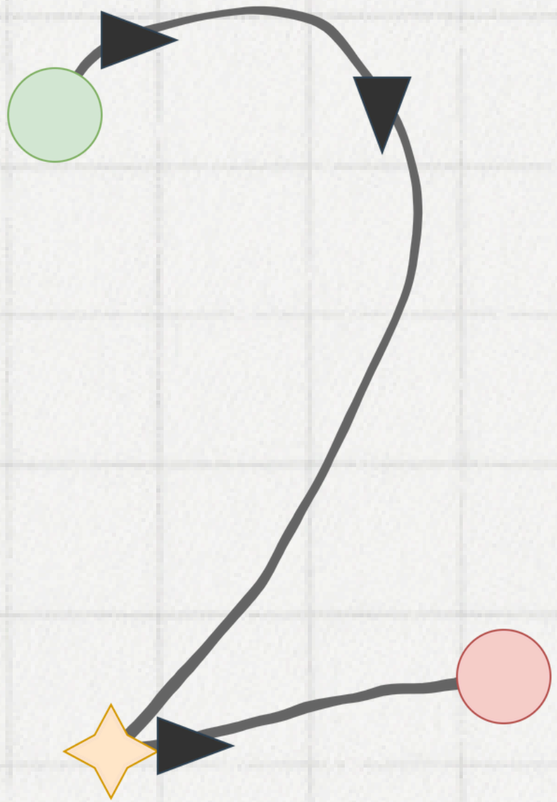
Winter 2024

What is handwriting recognition?

Why is my title handwritten character recognition?



The GRAIL method



Motivation for using GRAIL method

Easy to use for anyone who knows how to write

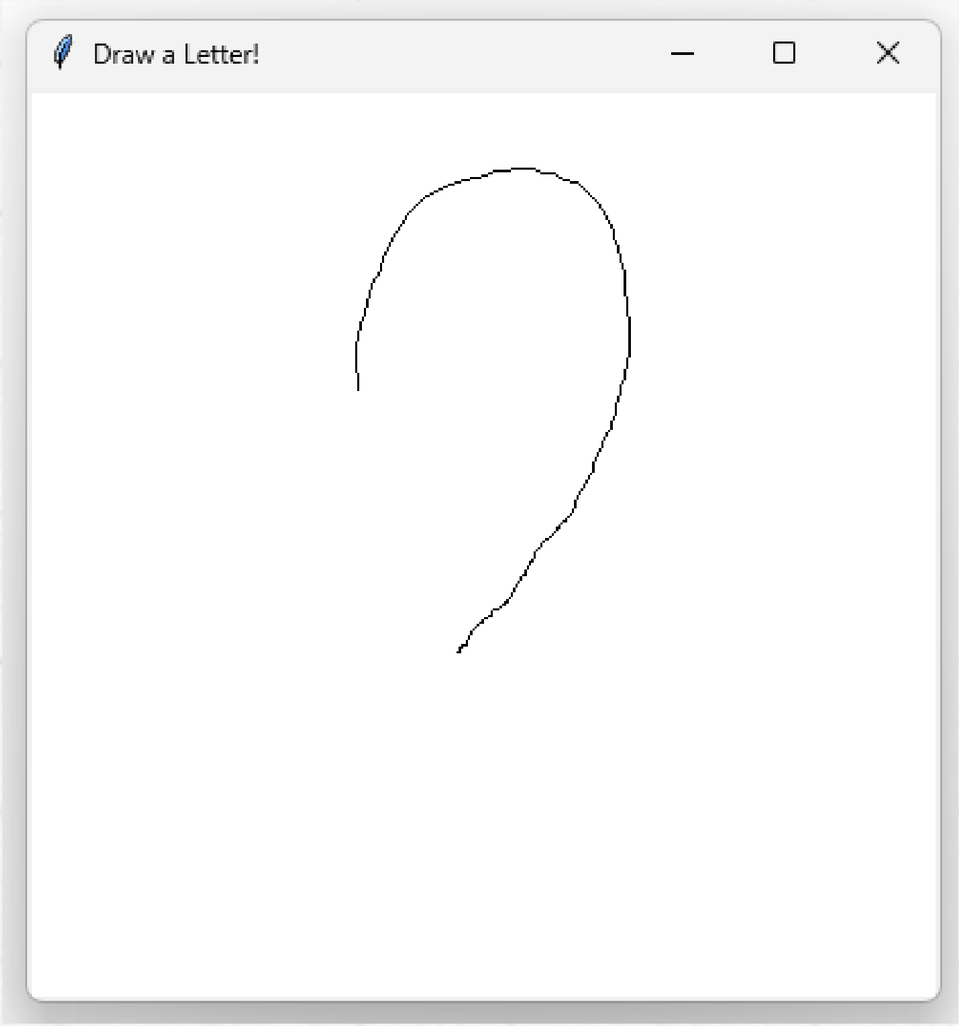
Lightweight and efficient

Solution without machine learning

Problem?

No drawing interface in SQL

Single stroke canvas using python

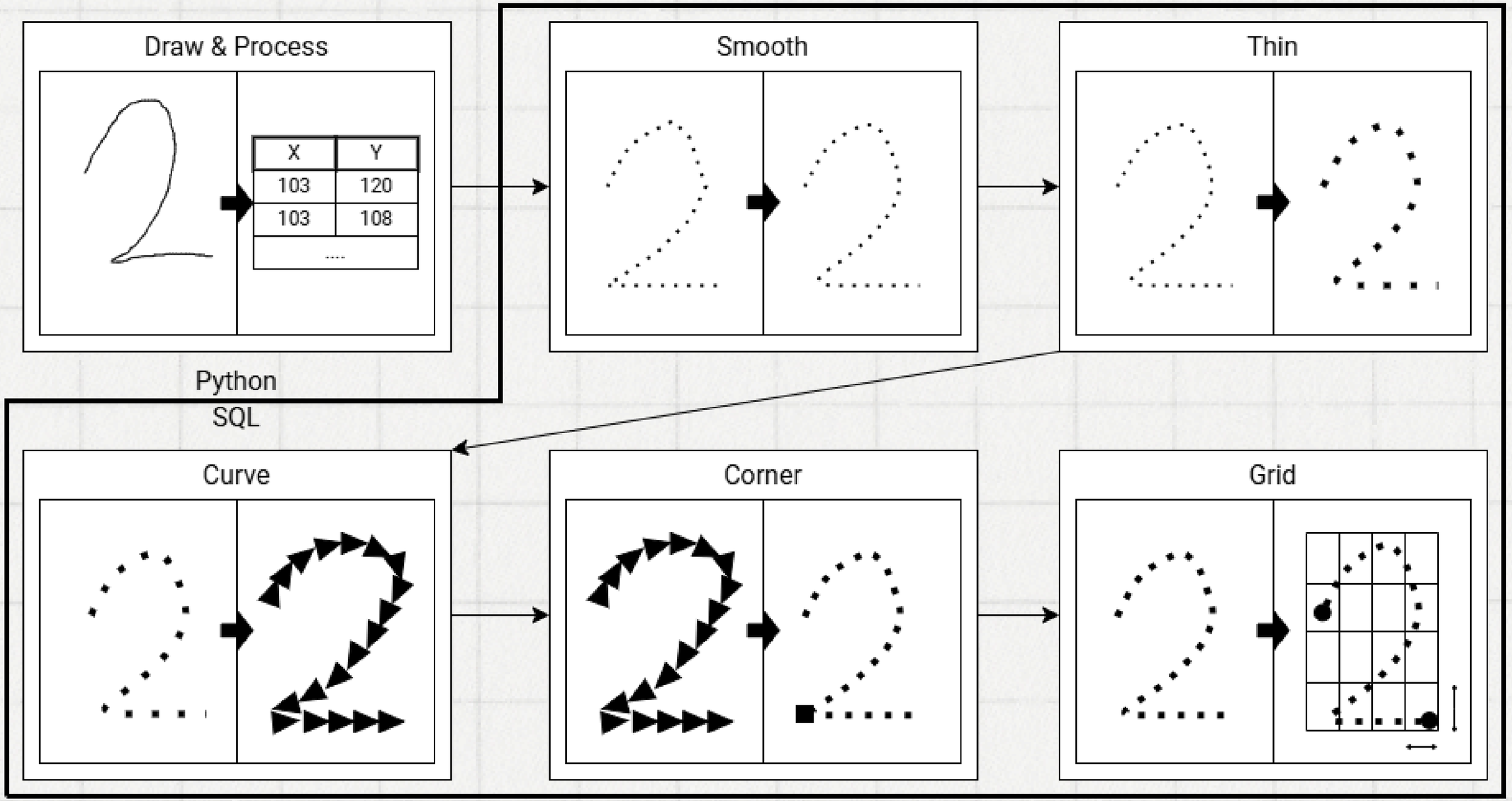


Why python?

Because it’s simple and straightforward

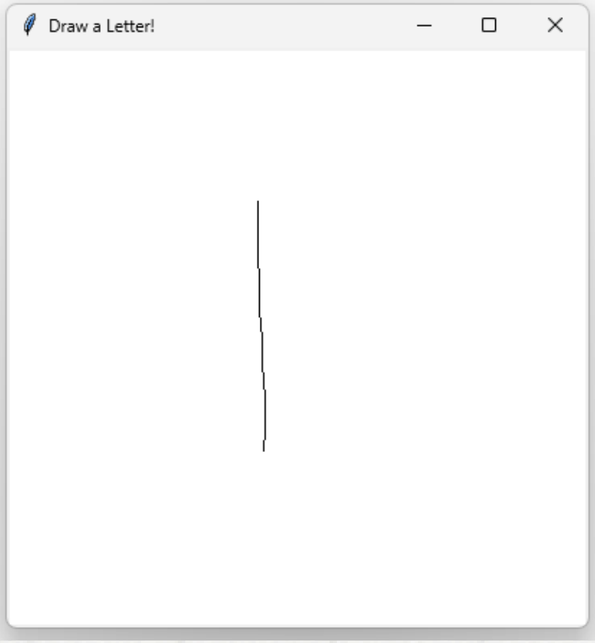
Because of my past experience with it

Steps

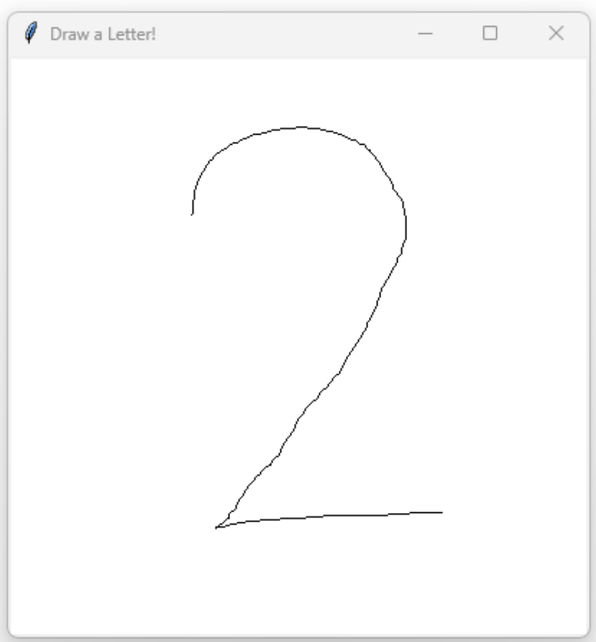

Draw & Process
X
Y
103
120
103
108
....
Smooth
Thin
Python
SQL
Curve
Corner
Grid

1. **Draw & Process**
2. **Smooth**
3. **Thin**
4. **Curve**
5. **Corner**
6. **Grid**

Draw & Process




```
(cnt, Xpoz, Ypoz)

(0, 174, 106)
(1, 174, 110)
(2, 174, 116)
 ... ... ...
(54, 178, 278)
(55, 178, 279)
(56, 178, 280)
```




```
(cnt, Xpoz, Ypoz)

(0, 127, 110)
(1, 128, 109)
(2, 128, 106)
 ... ... ...
(227, 300, 317)
(228, 301, 317)
(229, 302, 317)
```

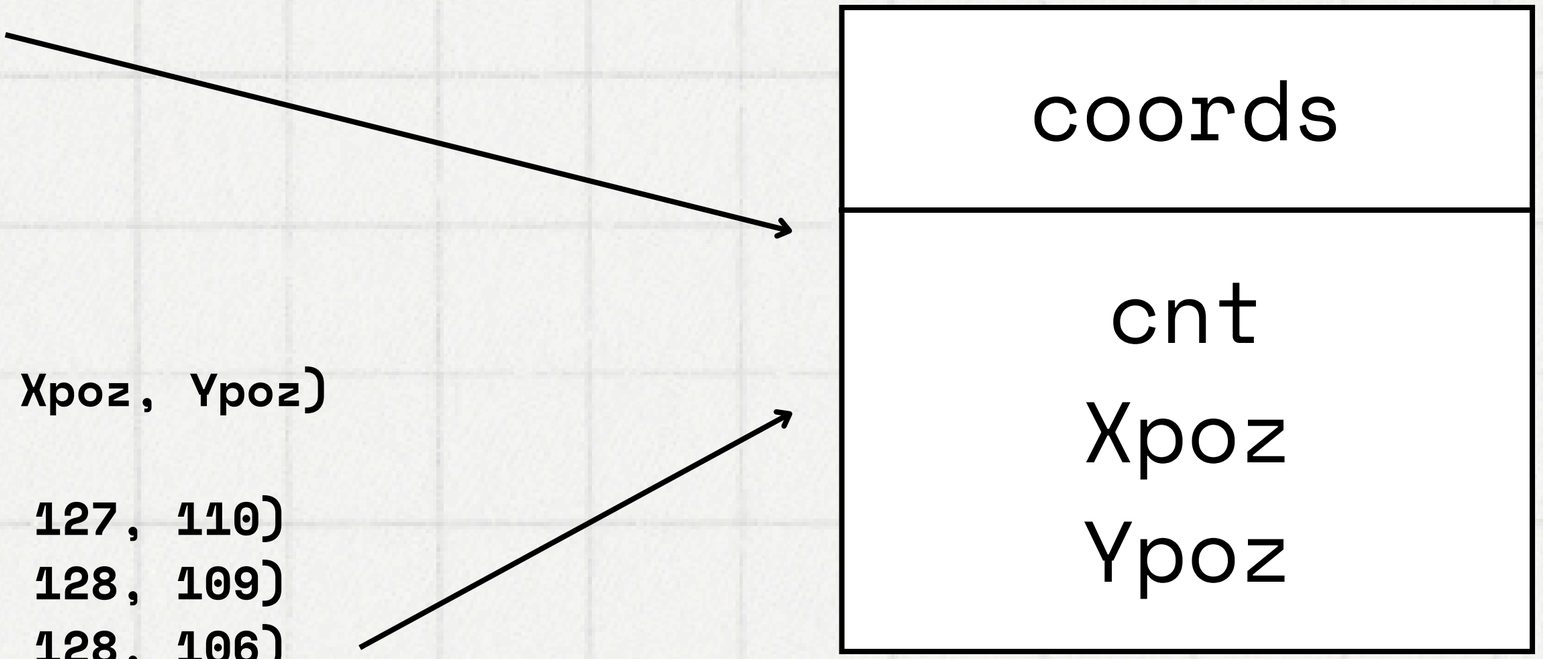
Draw & Process

(cnt, Xpoz, Ypoz)

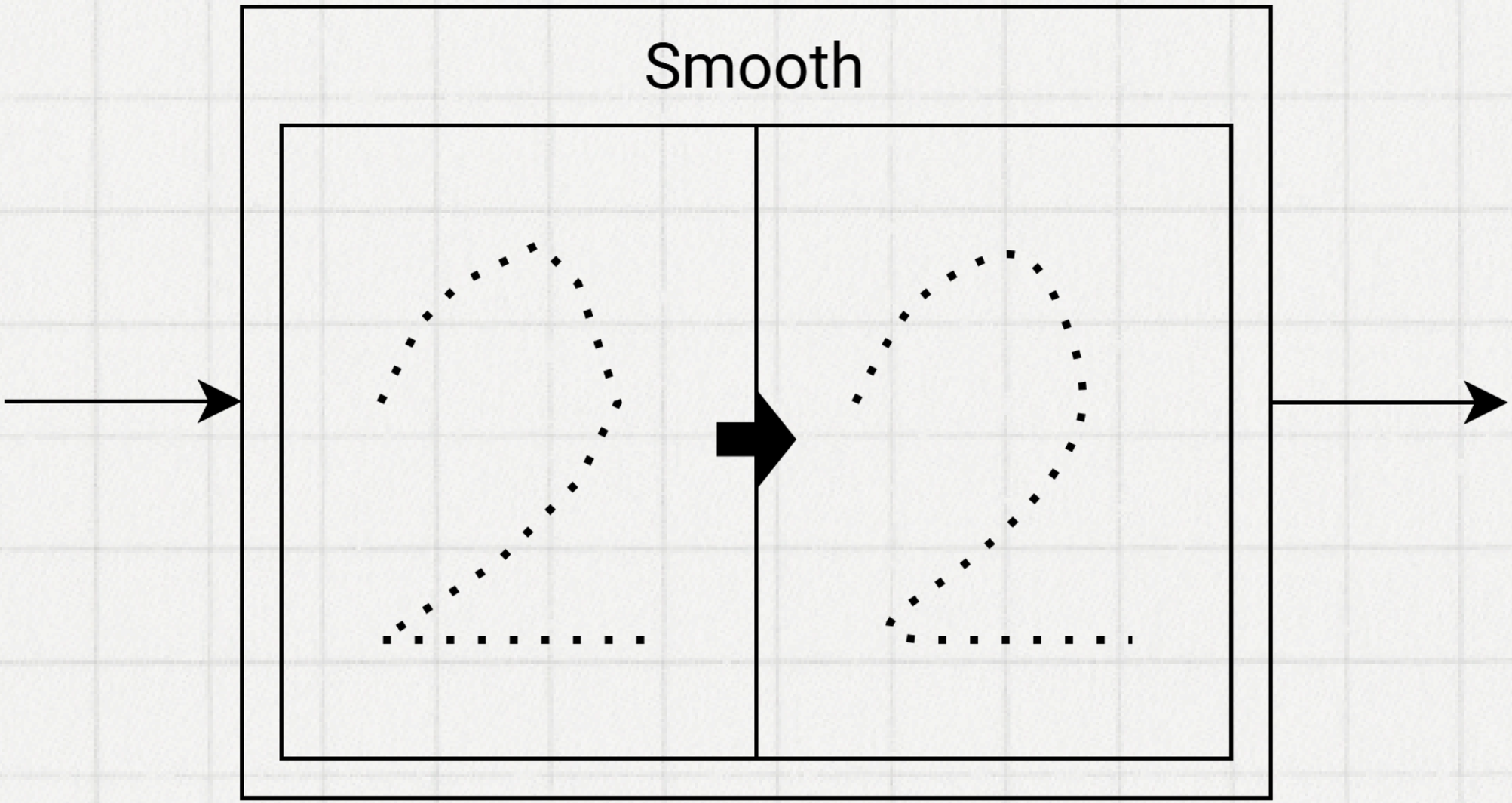
(0, 174, 106)
(1, 174, 110)
(2, 174, 116)
...
(54, 178, 278)
(55, 178, 279)
(56, 178, 280)

(cnt, Xpoz, Ypoz)

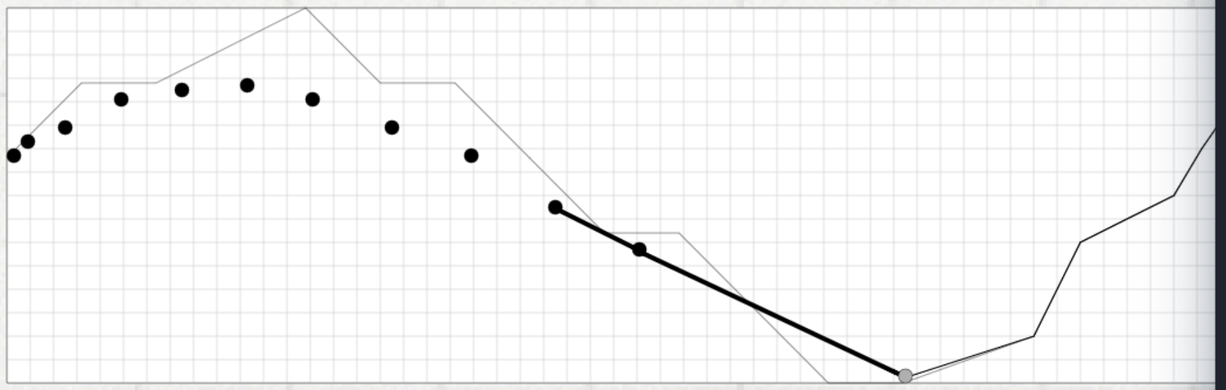
(0, 127, 110)
(1, 128, 109)
(2, 128, 106)
...
(227, 300, 317)
(228, 301, 317)
(229, 302, 317)



1. Draw & Process
2. **Smooth**
3. Thin
4. Curve
5. Corner
6. Grid


Smooth

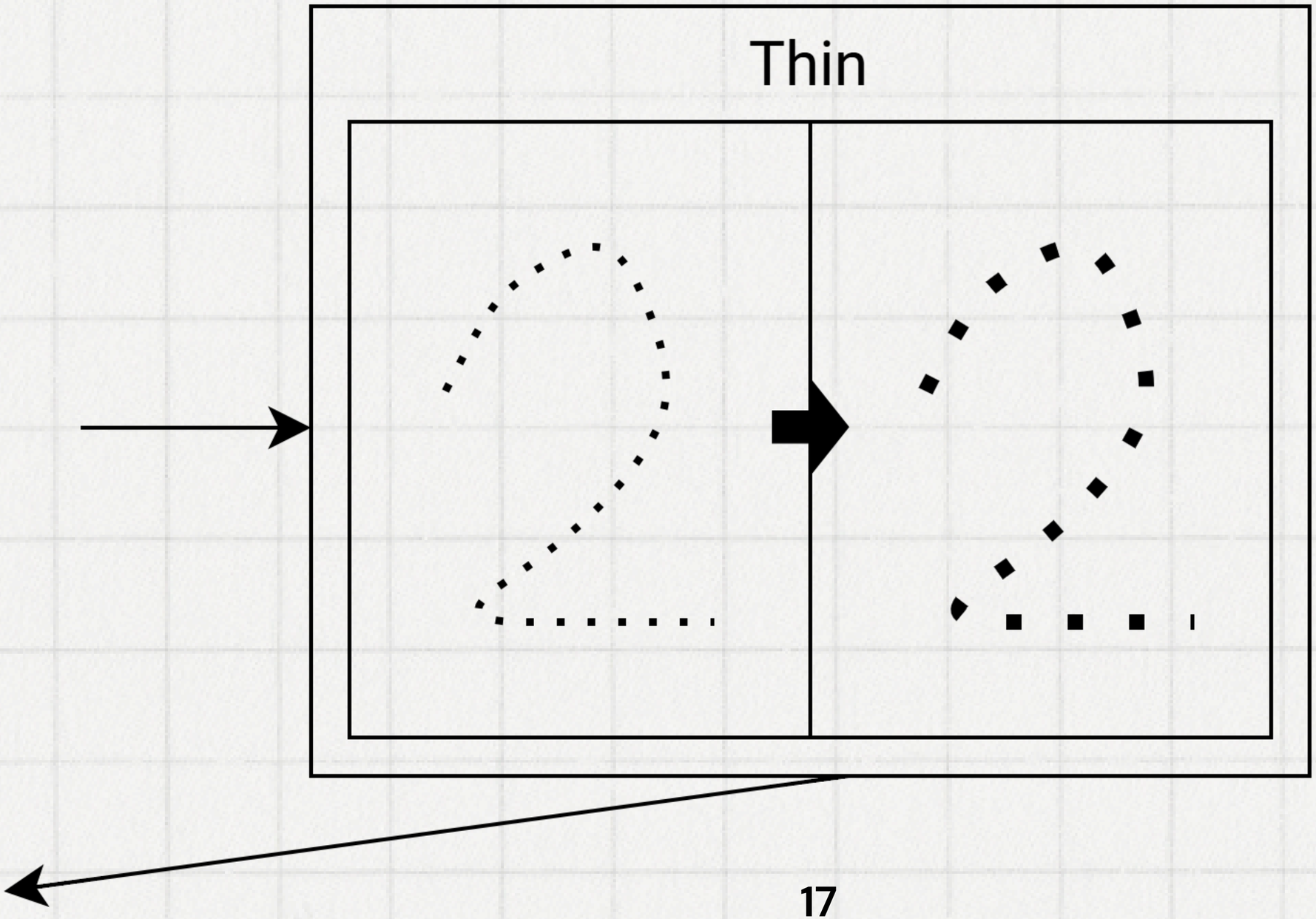
Smooth



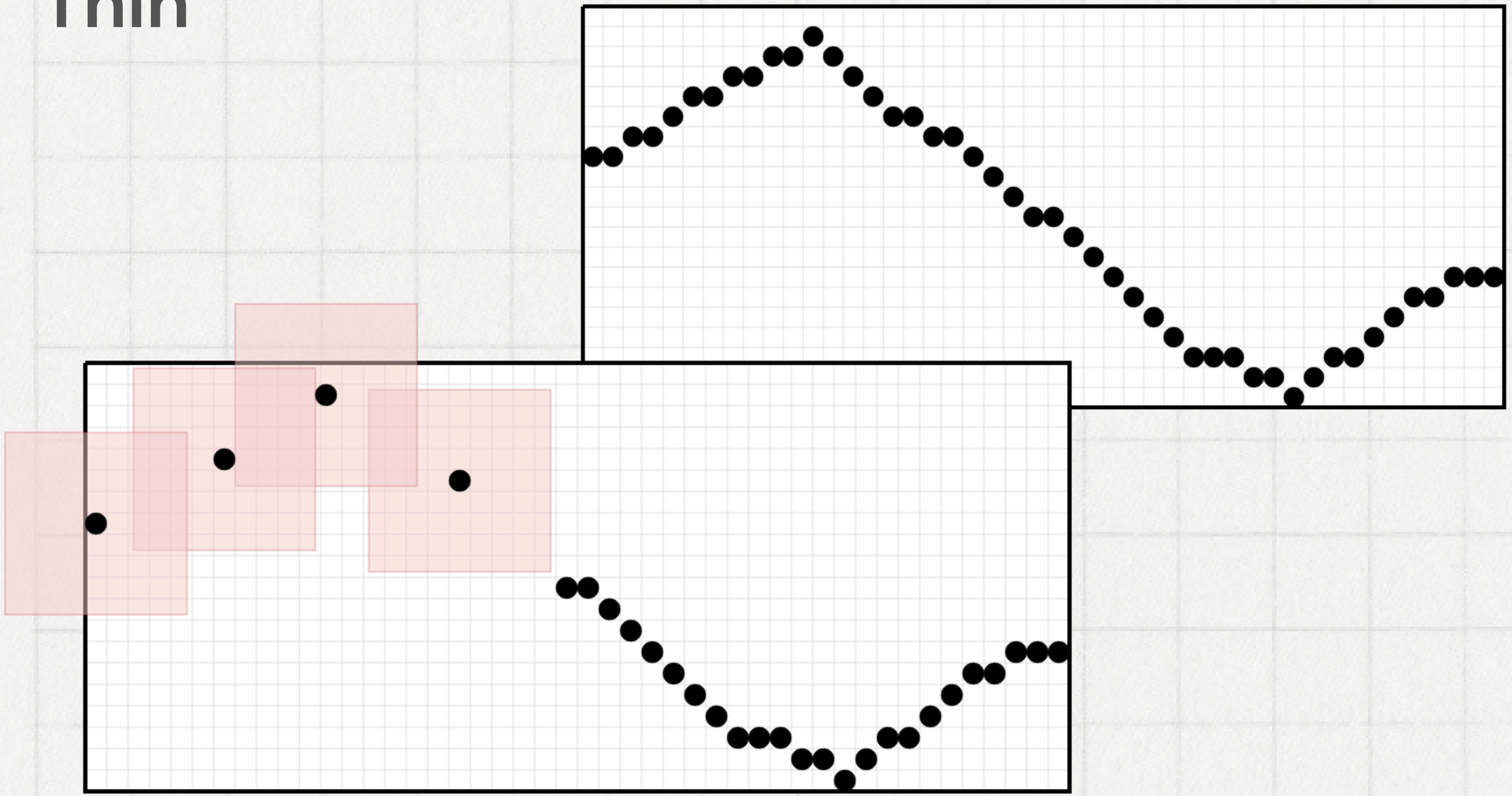
Smooth Step

```
smooth(cnt, x, y) AS (
  SELECT cnt, x::real, y::real
  FROM tablet
  WHERE cnt = 0

  UNION ALL

  SELECT t.cnt,
    (0.75 * s.x + 0.25 * t.x) :: real AS x,a
    (0.75 * s.y + 0.25 * t.y) :: real AS y
    -- 75% smoothing
  FROM smooth s
  JOIN tablet t ON t.cnt = s.cnt + 1
  WHERE t.cnt IS NOT NULL
  AND s.cnt ⩽ (SELECT MAX(cnt)
               FROM tablet)
)
```

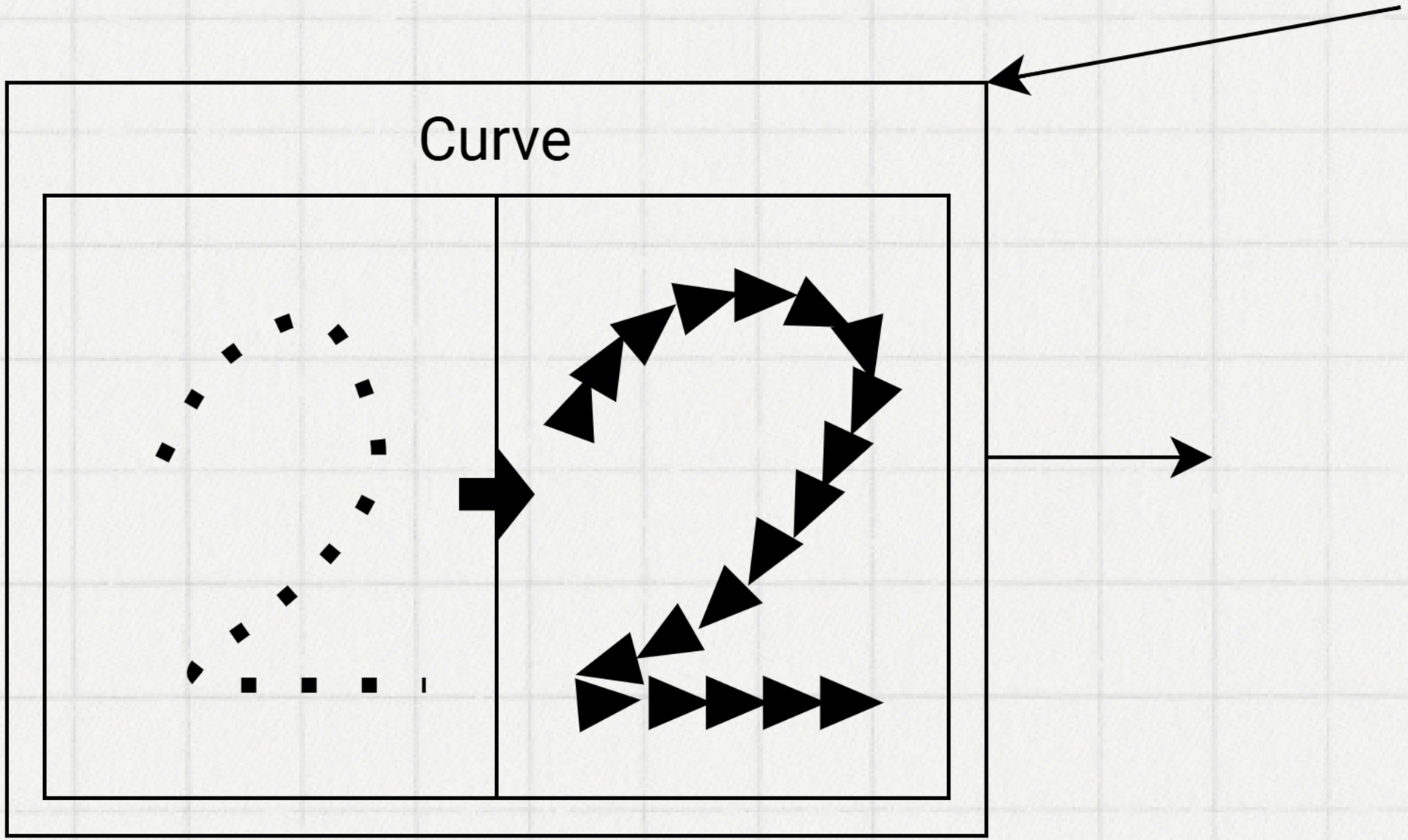
1. Draw & Process
2. Smooth
3. Thin
4. Curve
5. Corner
6. Grid


Thin

Thin



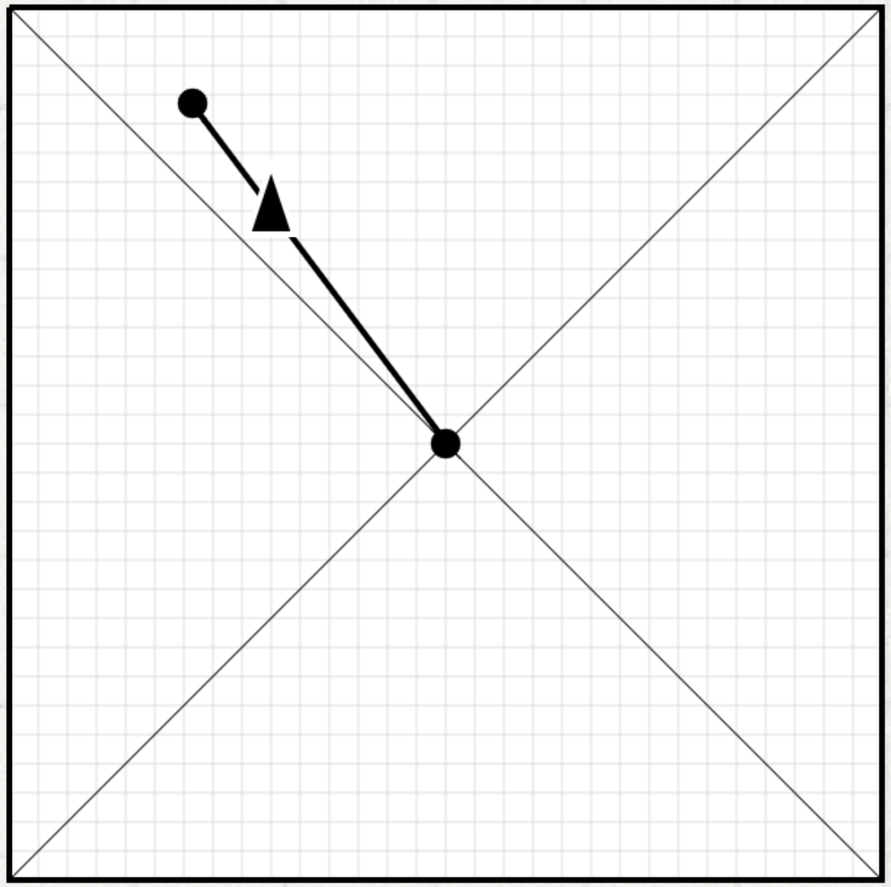
1. Draw & Process
2. Smooth
3. Thin
4. **Curve**
5. Corner
6. Grid


Curve

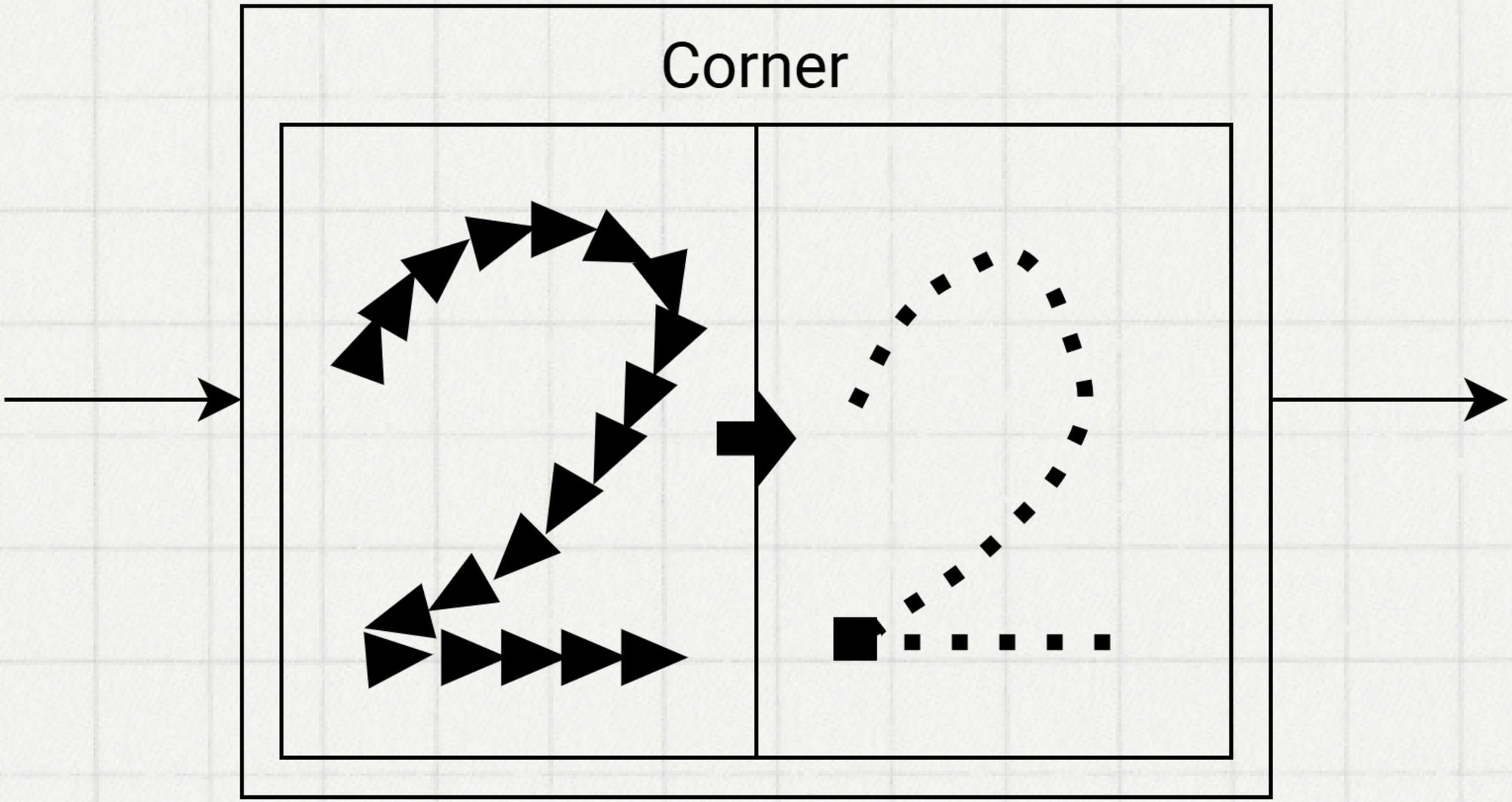
Curve Step

```
curve(cnt, x, y, direction) AS (
  SELECT cnt, x, y, direction
  FROM (
    SELECT cnt, x, y,
           CASE
         WHEN lag(x) OVER (ORDER BY cnt) IS NULL THEN
              (enum_range(NULL::directions))[
                ((degrees(atan2(lead(y) OVER
           (ORDER BY cnt) - y, lead(x) OVER
           (ORDER BY cnt) - x)) + 360) / 90)::int % 4 + 1
              ]::directions
         ELSE
              (enum_range(NULL::directions))[
                ((degrees(atan2(-y + lag(y) OVER
           (ORDER BY cnt), x - lag(x) OVER
           (ORDER BY cnt))) + 360) / 90)::int % 4 + 1
              ]::directions
           END AS direction
    FROM thin
    GROUP BY cnt, x, y
    )
  WHERE cnt =/= 0
)
```

```
CREATE TYPE directions
AS ENUM('R', 'U', 'L', 'D');
```



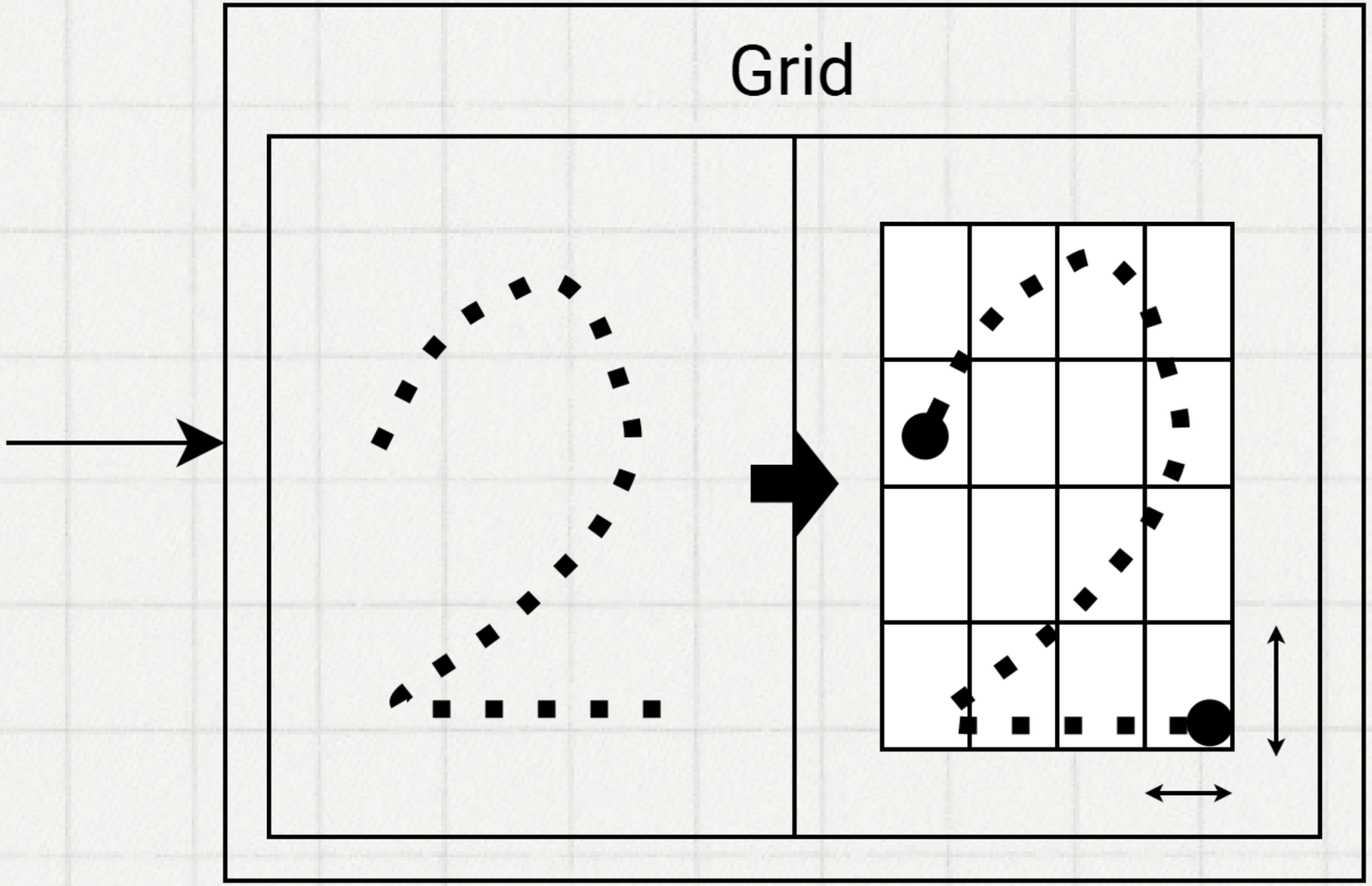
1. Draw & Process
2. Smooth
3. Thin
4. Curve
5. **Corner**
6. Grid


Corner

Corner Step

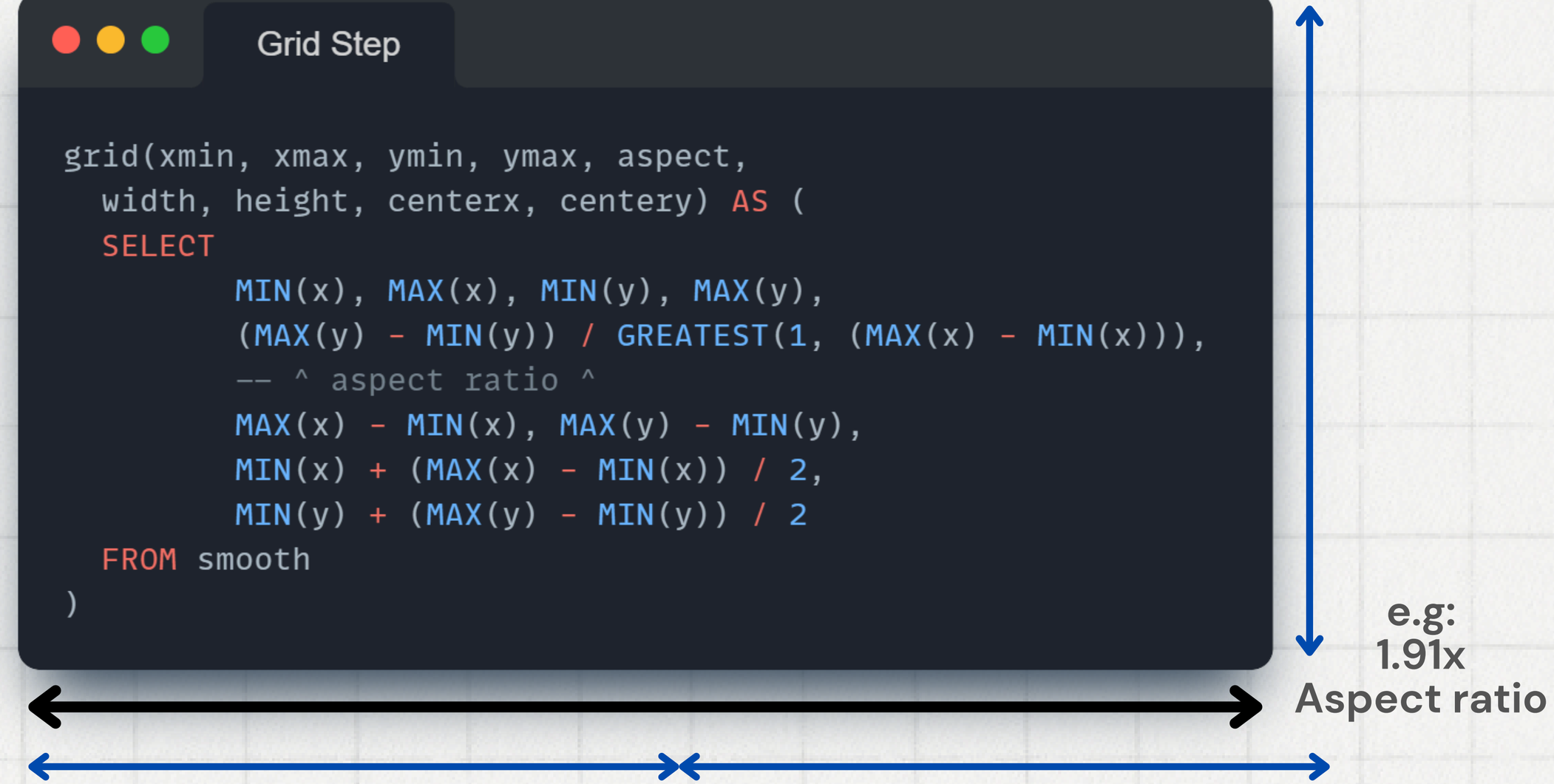
```
corner(cnt, x, y) AS (
  SELECT cnt, x, y, angle_change
  FROM (
    SELECT cnt, x, y,
           degrees(abs(atan2(y - lag(y) OVER
      (ORDER BY cnt), x - lag(x) OVER (ORDER BY cnt))
           - atan2(lead(y) OVER
      (ORDER BY cnt) - y, lead(x) OVER (ORDER BY cnt) - x)))
           AS angle_change
    FROM curve
  ) AS _
  WHERE angle_change > 60
  -- Angle threshold for detecting a corner (in degrees)
)
```

1. Draw & Process
2. Smooth
3. Thin
4. Curve
5. Corner
6. **Grid**


Grid

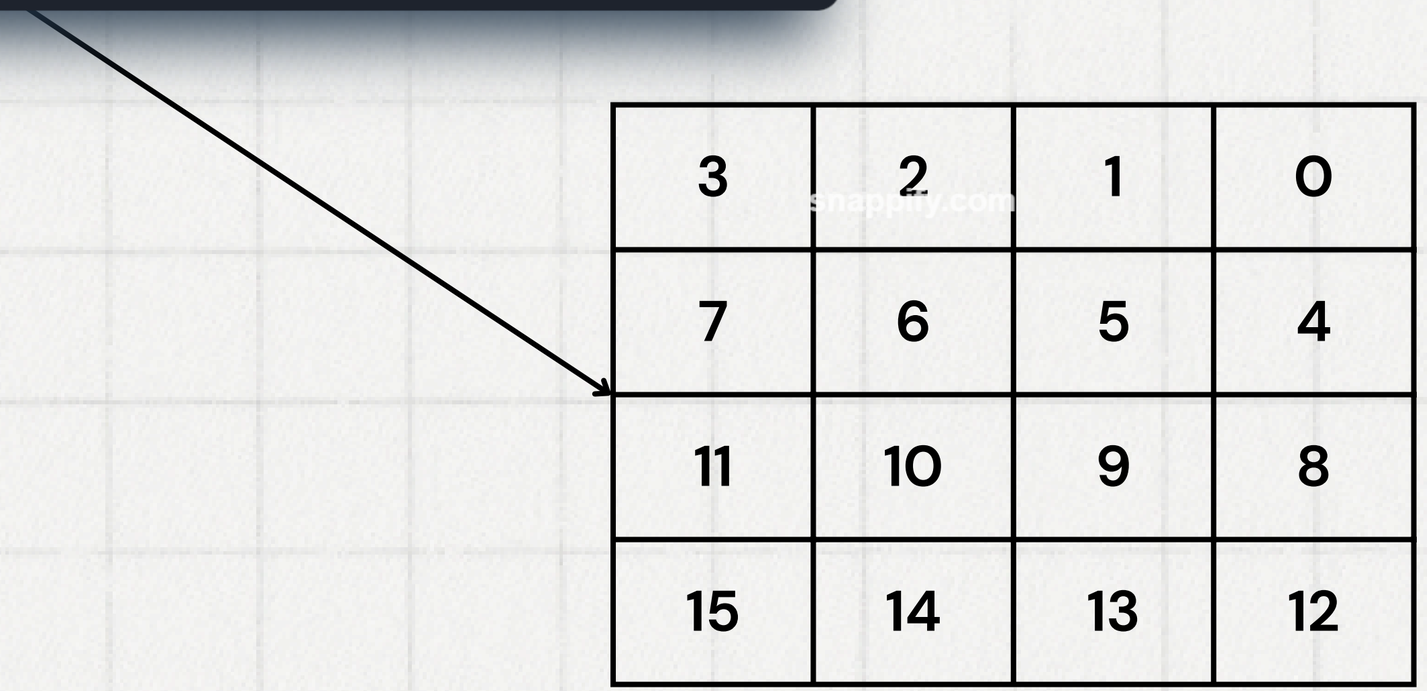
Grid Step

```
grid(xmin, xmax, ymin, ymax, aspect,
  width, height, centerx, centery) AS (
  SELECT
         MIN(x), MAX(x), MIN(y), MAX(y),
         (MAX(y) - MIN(y)) / GREATEST(1, (MAX(x) - MIN(x))),
         -- ^ aspect ratio ^
         MAX(x) - MIN(x), MAX(y) - MIN(y),
         MIN(x) + (MAX(x) - MIN(x)) / 2,
         MIN(y) + (MAX(y) - MIN(y)) / 2
  FROM smooth
)
```



Grid Position Function

```sql
CREATE OR REPLACE FUNCTION grid_position
  (width REAL, height REAL, xmin REAL, ymin REAL, x REAL, y REAL)
RETURNS INT AS $$
    SELECT GREATEST(0,
           (3 - (FLOOR(4 * (x - xmin) / (width + 1)) :: INT)) -- Columns
           + 4 * (FLOOR(4 * (y - ymin) / (height + 1)) :: INT) -- Rows
    );
$$ LANGUAGE SQL IMMUTABLE;
```

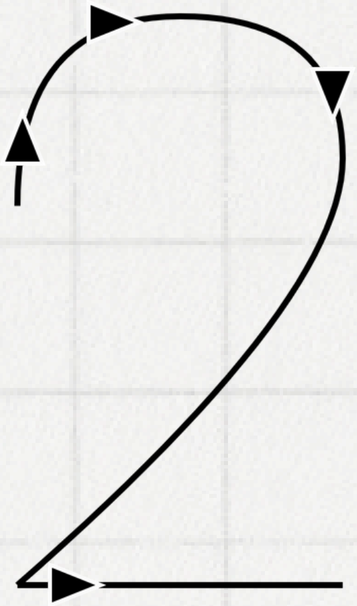


Tables

Tables

possible_characters	example for 2
first_four_directions candidate_characters	{"U","R","D","R"} {"2","3","8","B","D","P","R"}

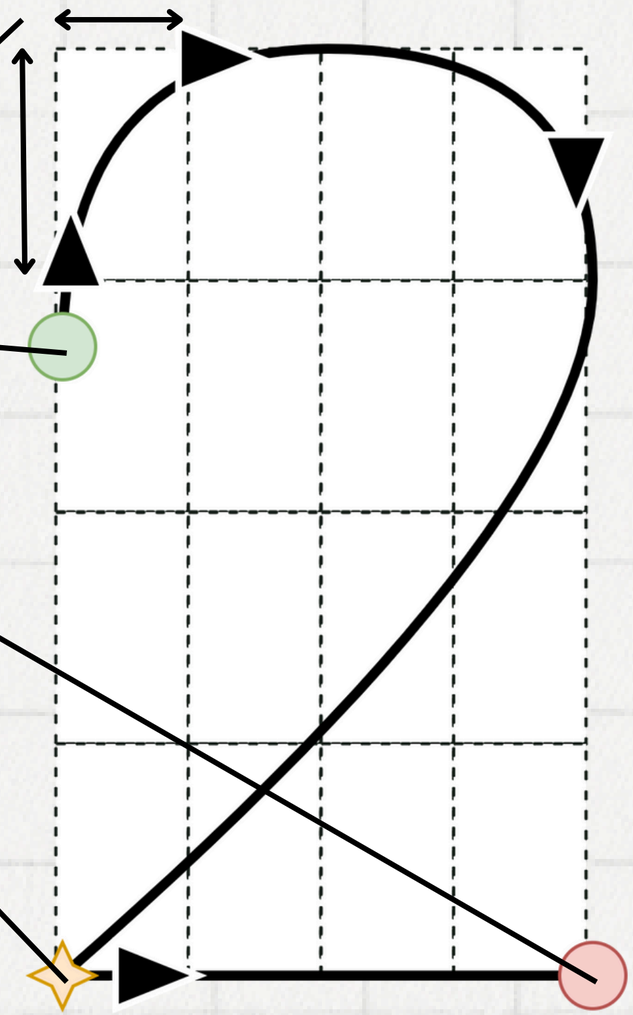
U	R	D	L
⬆	➡	⬇	⬅



Tables

criteria	example for 2
candidate_characters	{"2","3","8","B","D","P","R"}
character	2
start	7
finish	12
corners	{15}
aspect_range	1.517

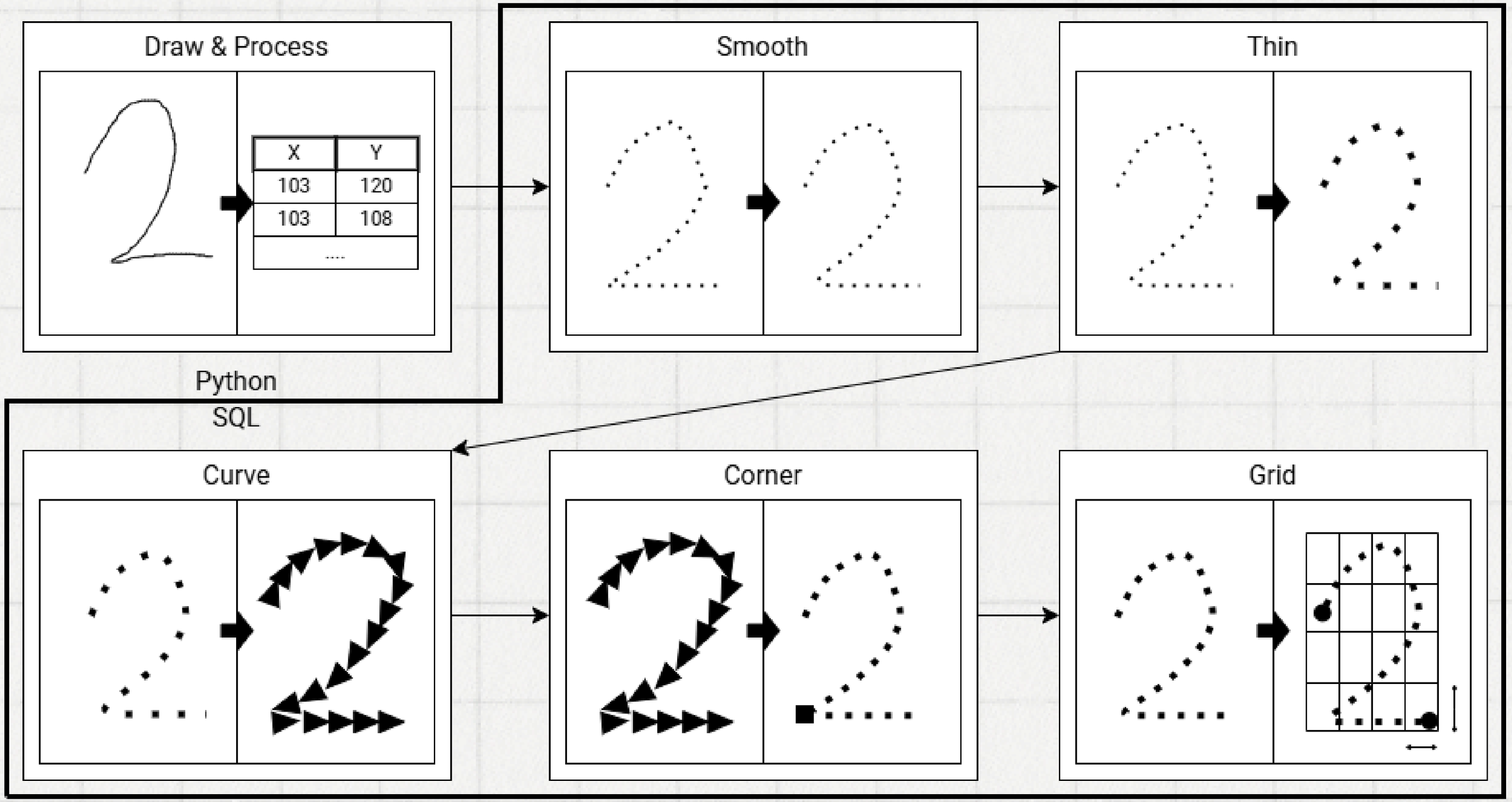
3	2	1	0
7	6	5	4
11	10	9	8
15	14	13	12



Result

character	directions	start	finish	corners	aspect	center
2	{U,R,D,L,R}	7	12	{15}	1.5388119	-201,213

Handwritten Character Recognition



Code demonstration

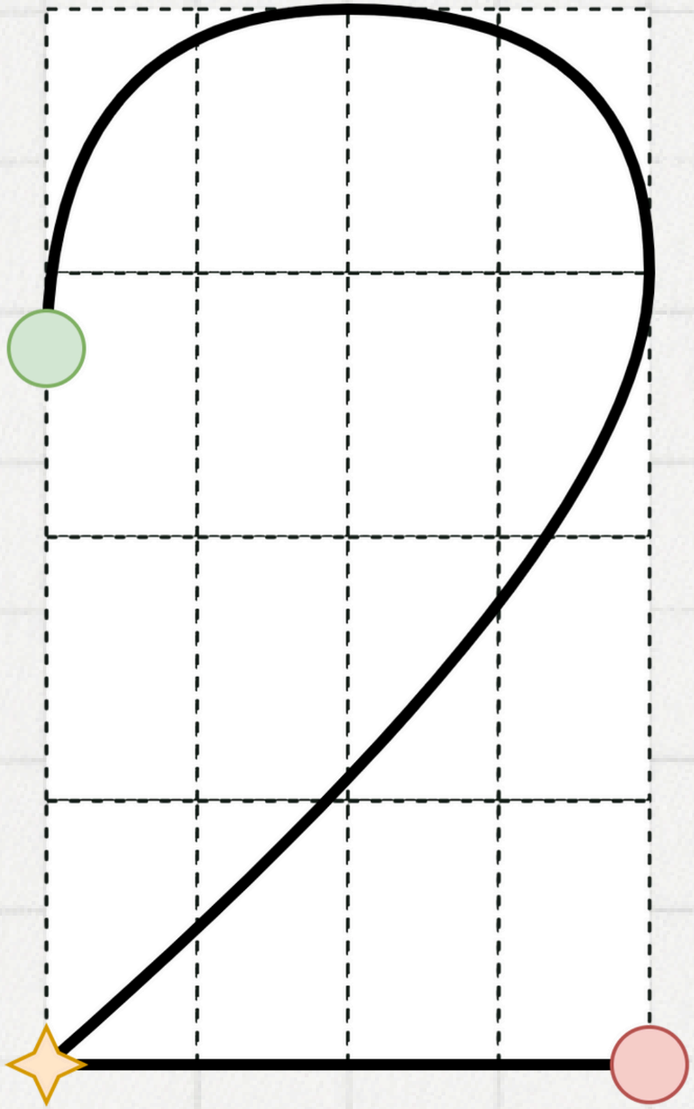
01
Python Canvas

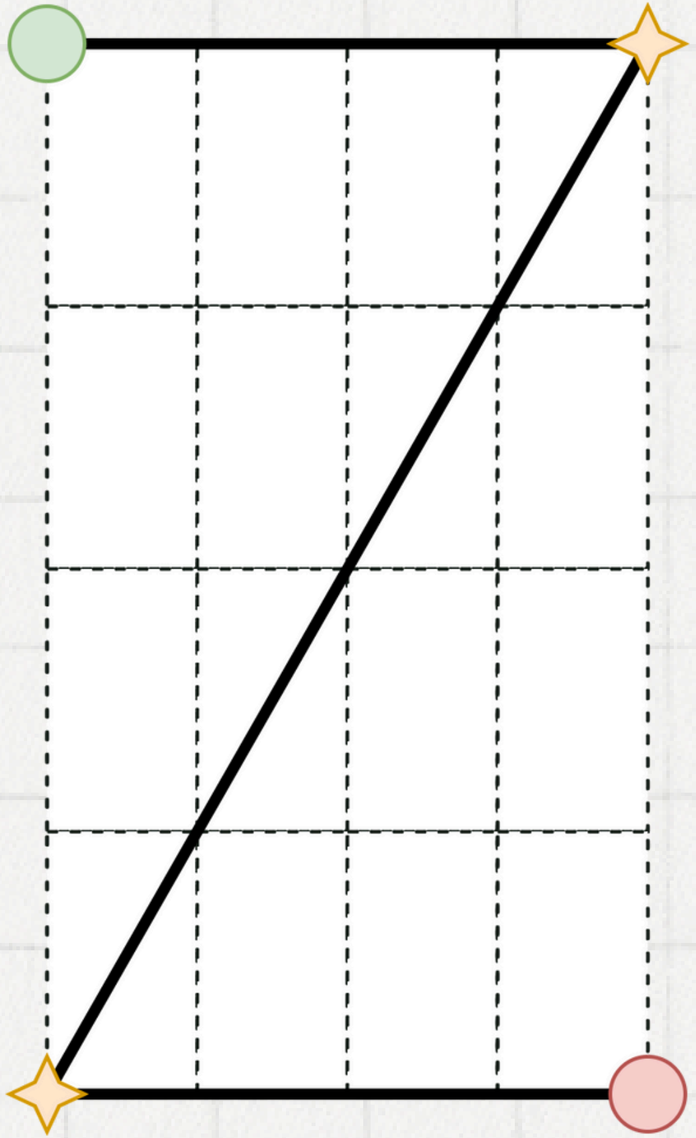
02
Table creation

03
Data processing

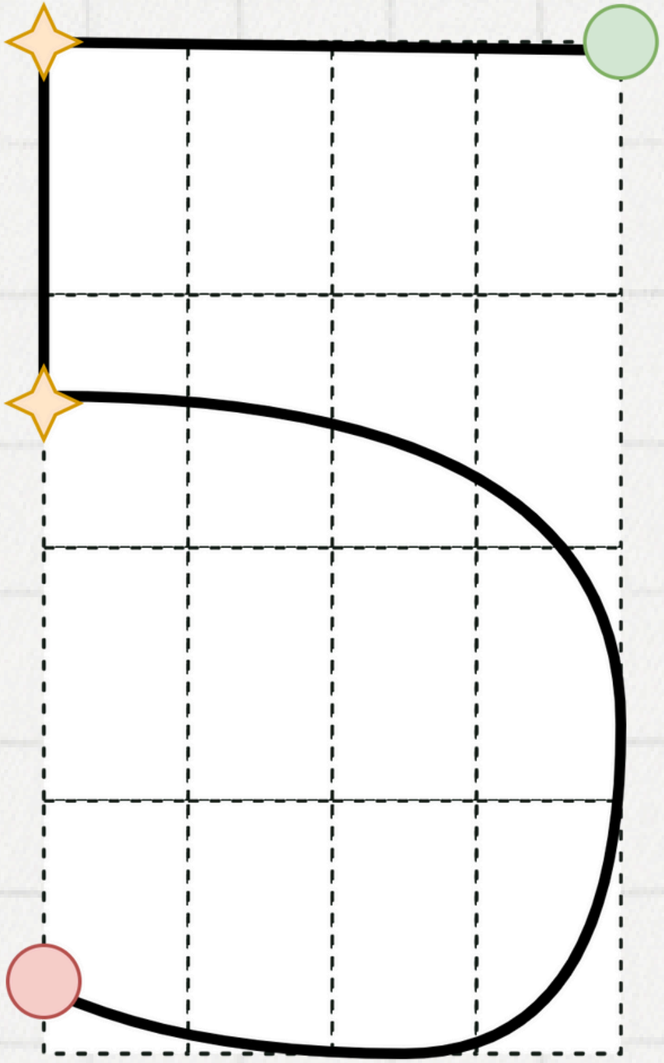
04
Best candidate

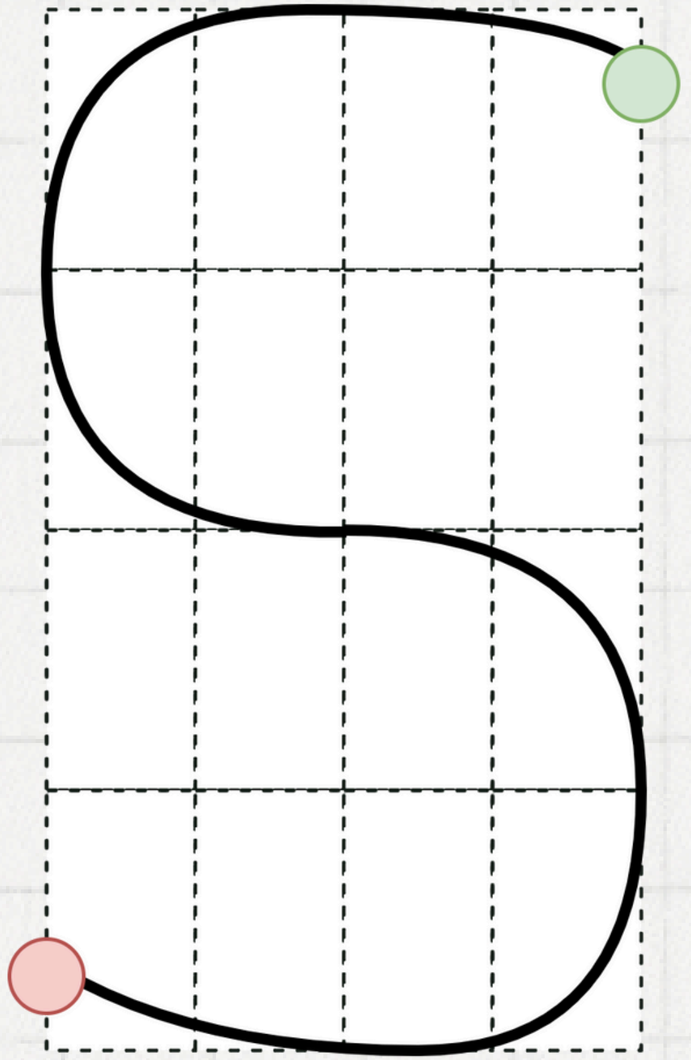
Case comparisons



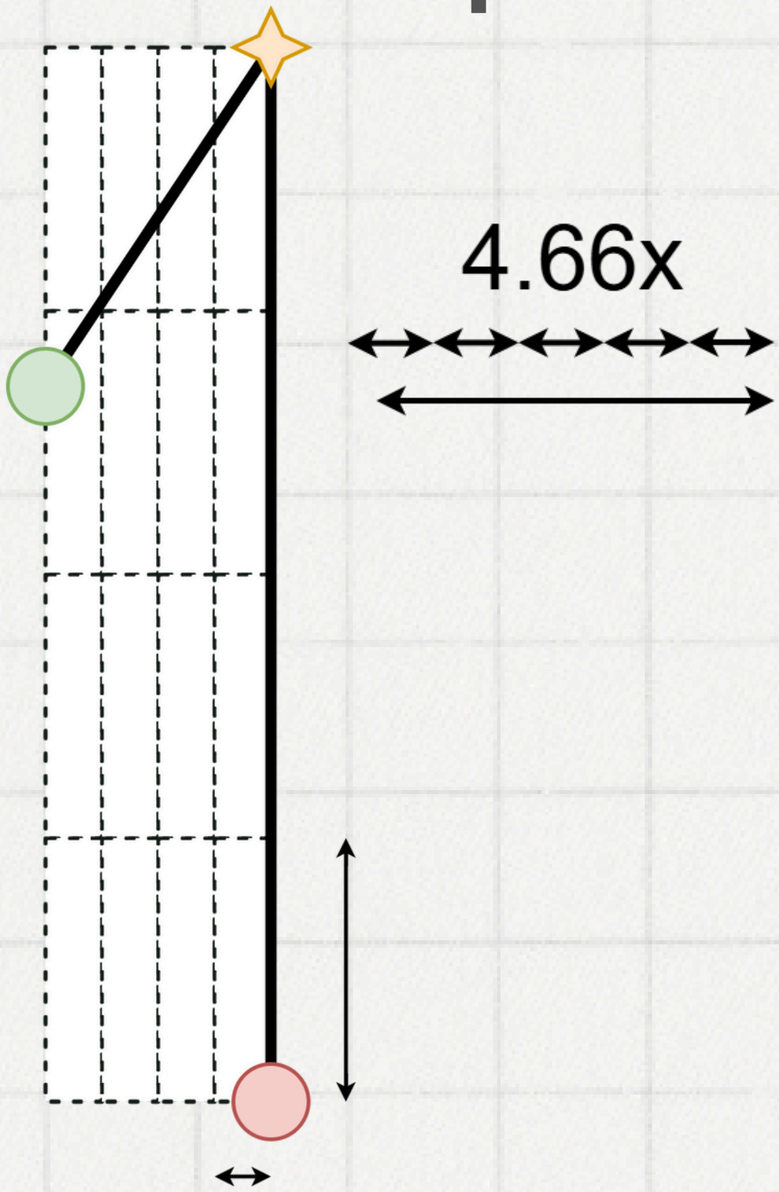


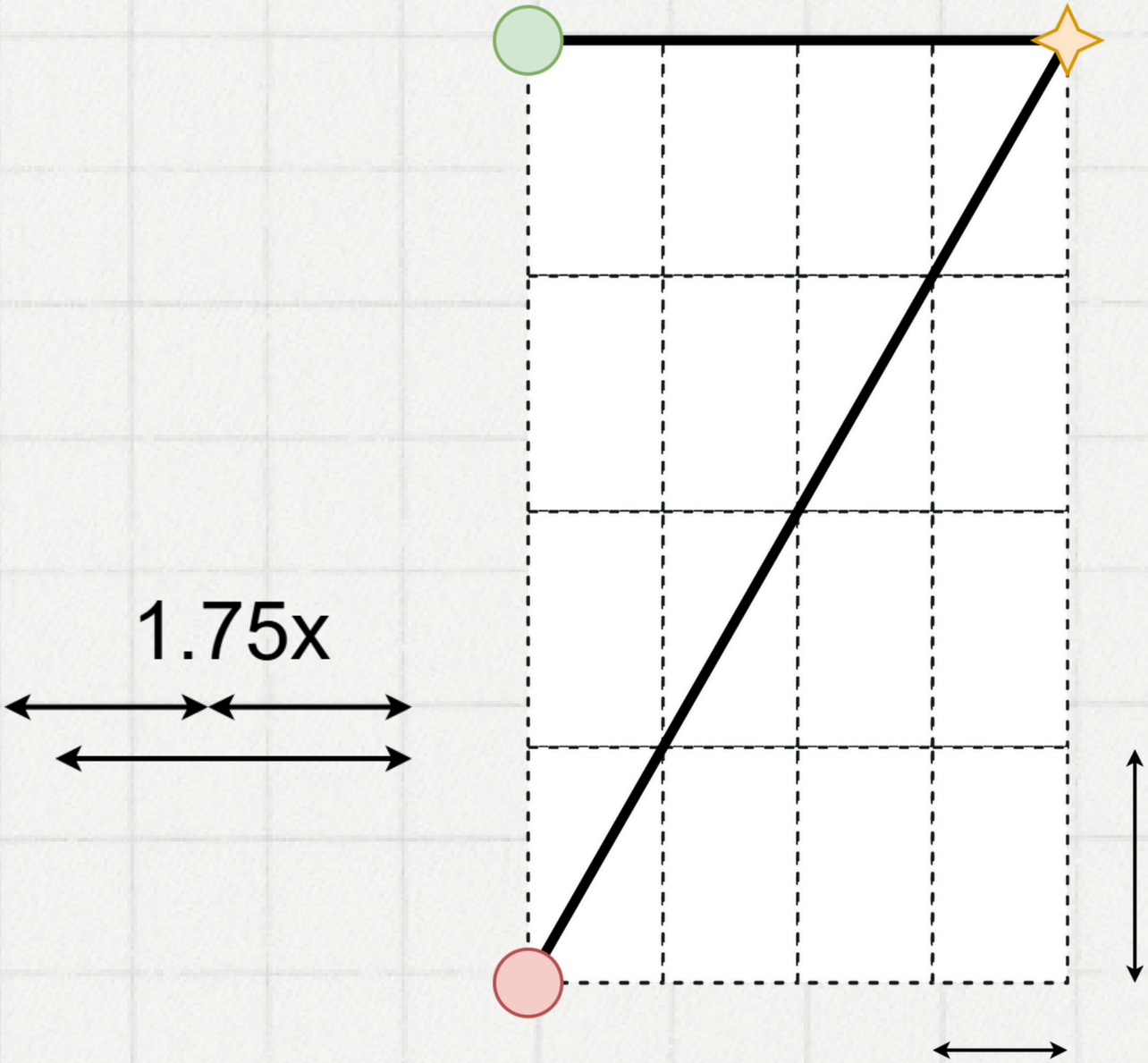
Case comparisons



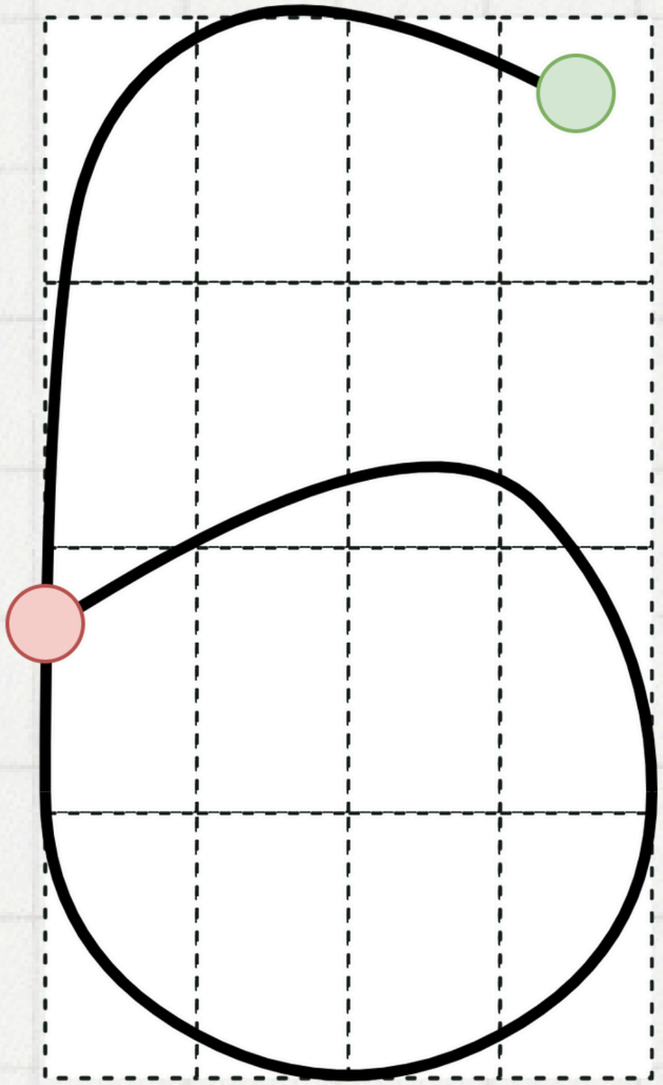


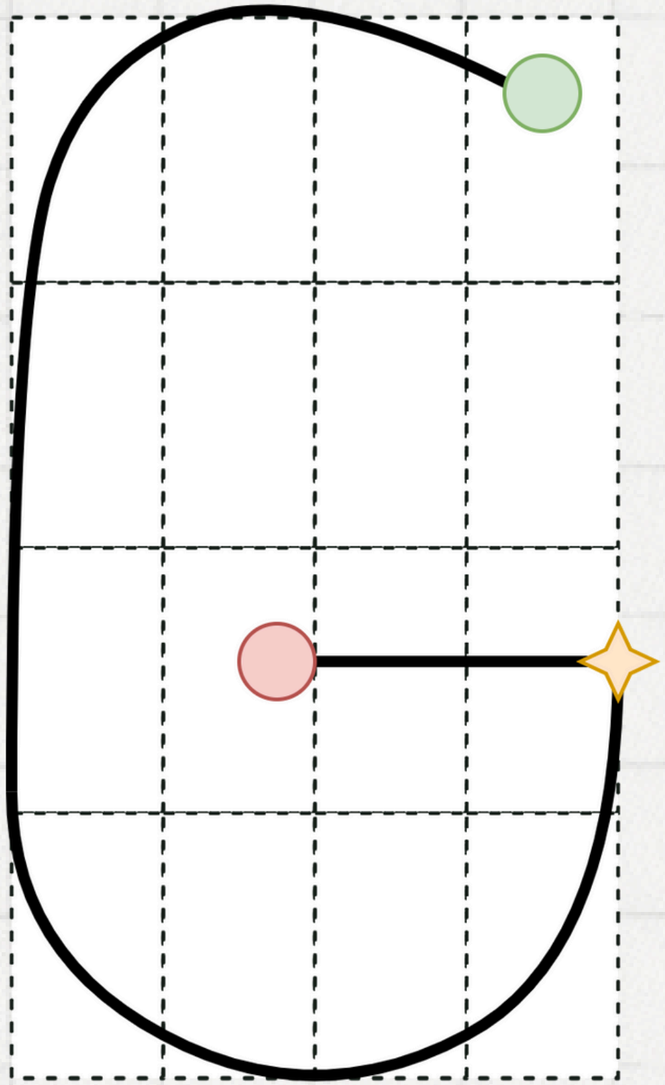
Case comparisons



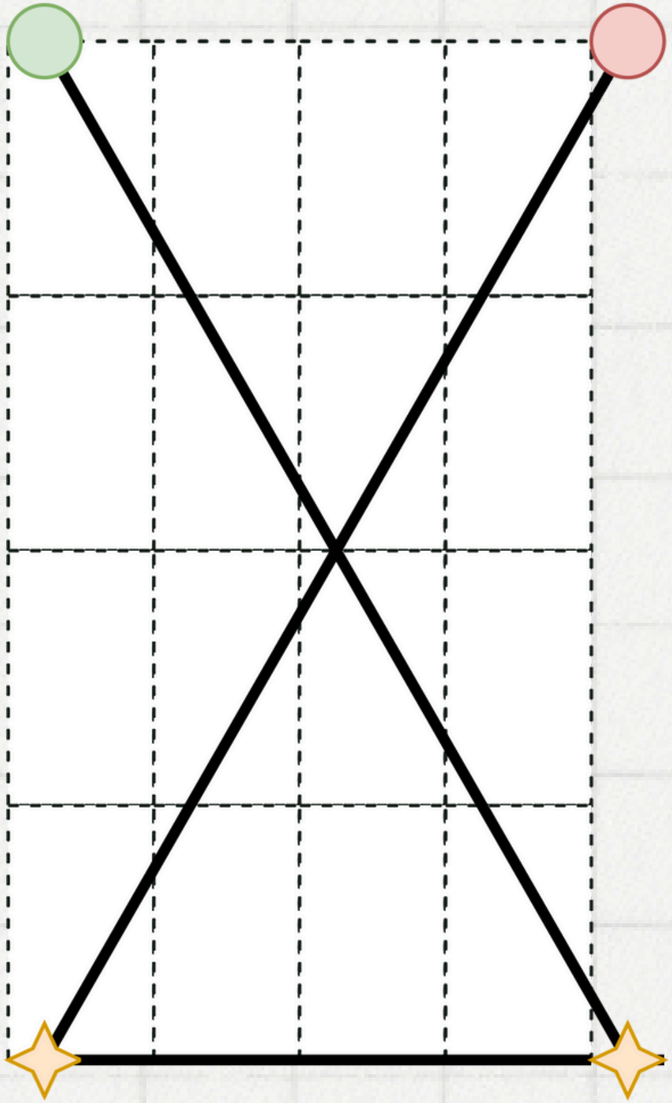


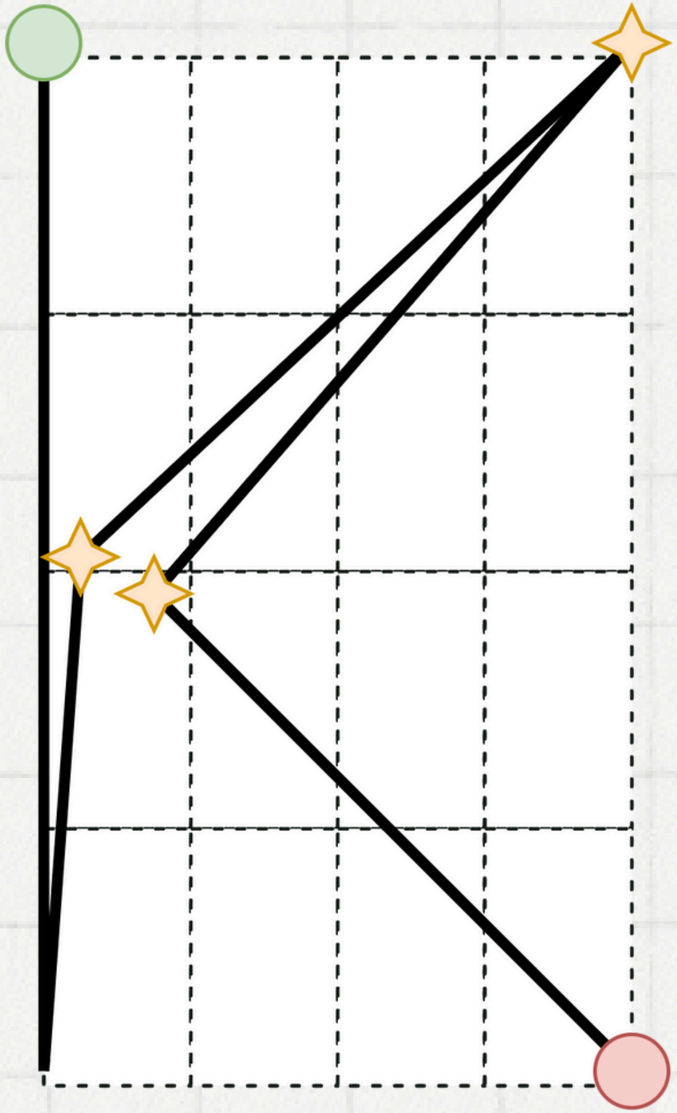
Case comparisons





Single stroke problems





Possible improvements

01. Better canvas to SQL integration

02. Refine edge case detection

03. Use the output to show the criteria visually

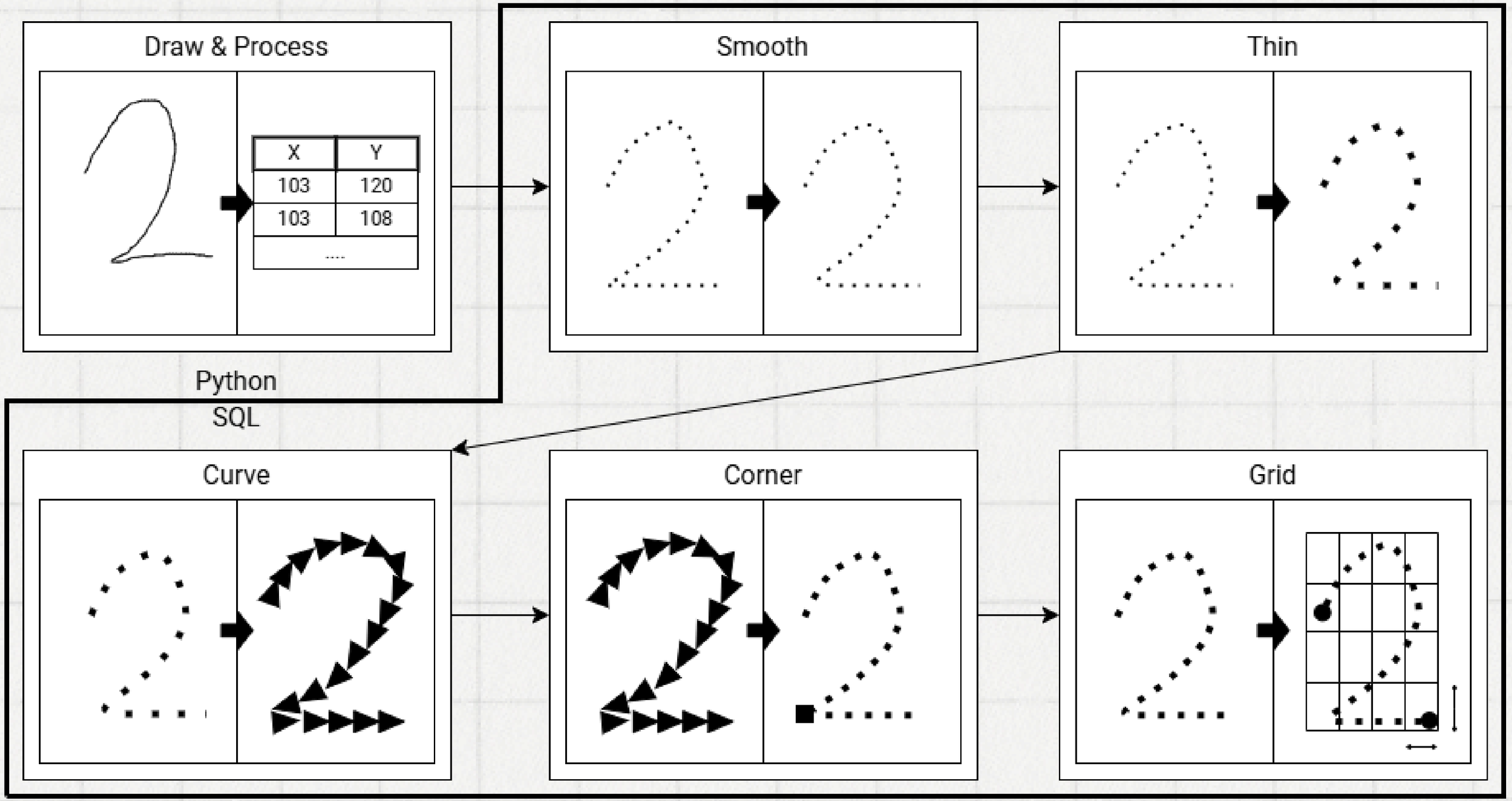
Conclusion

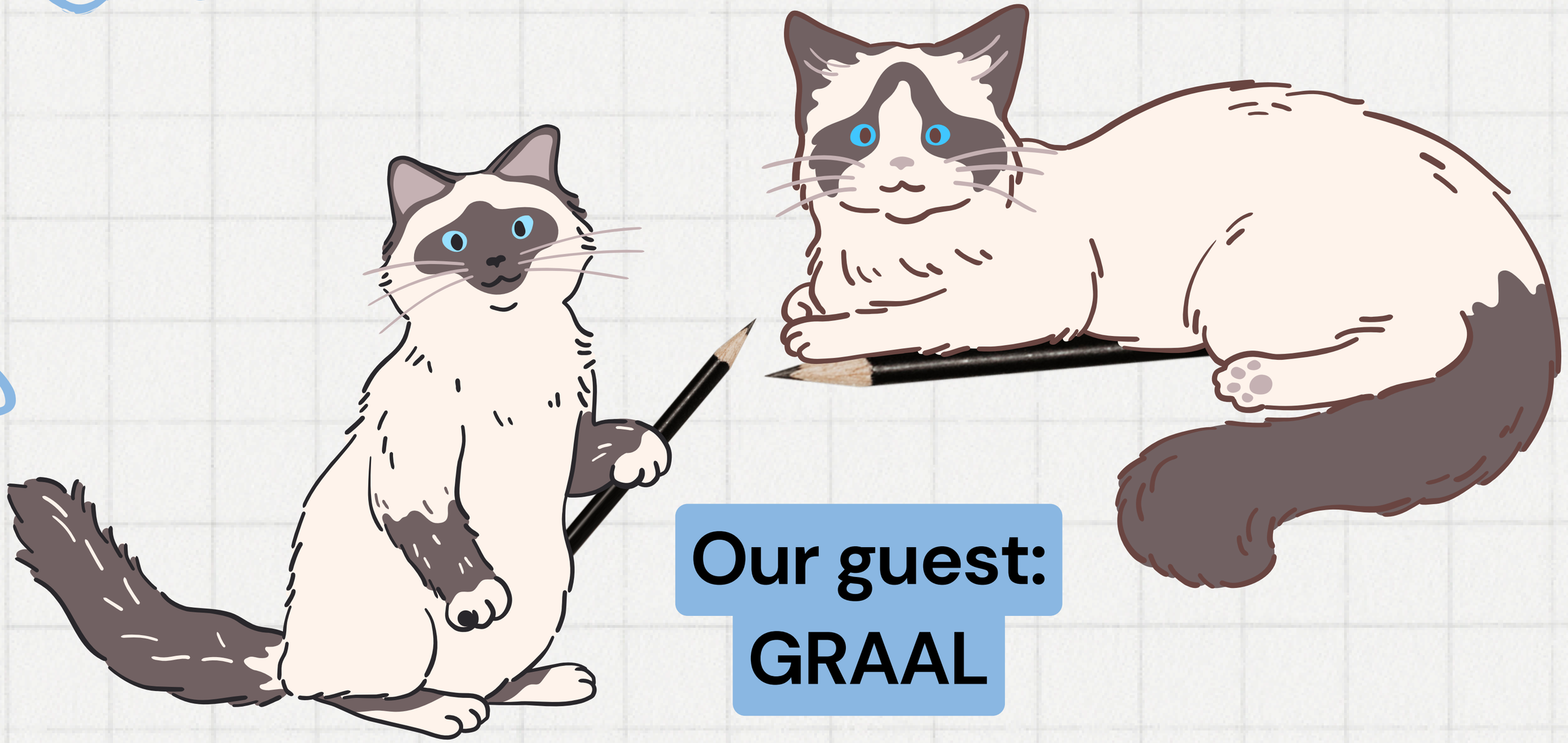
Lightweight and efficient
No need for machine learning
Interesting thought process

Interactive site and the blogpost that inspired me to use the algorithm

The site is not made by me

Handwritten Character Recognition





Our guest: GRAAL