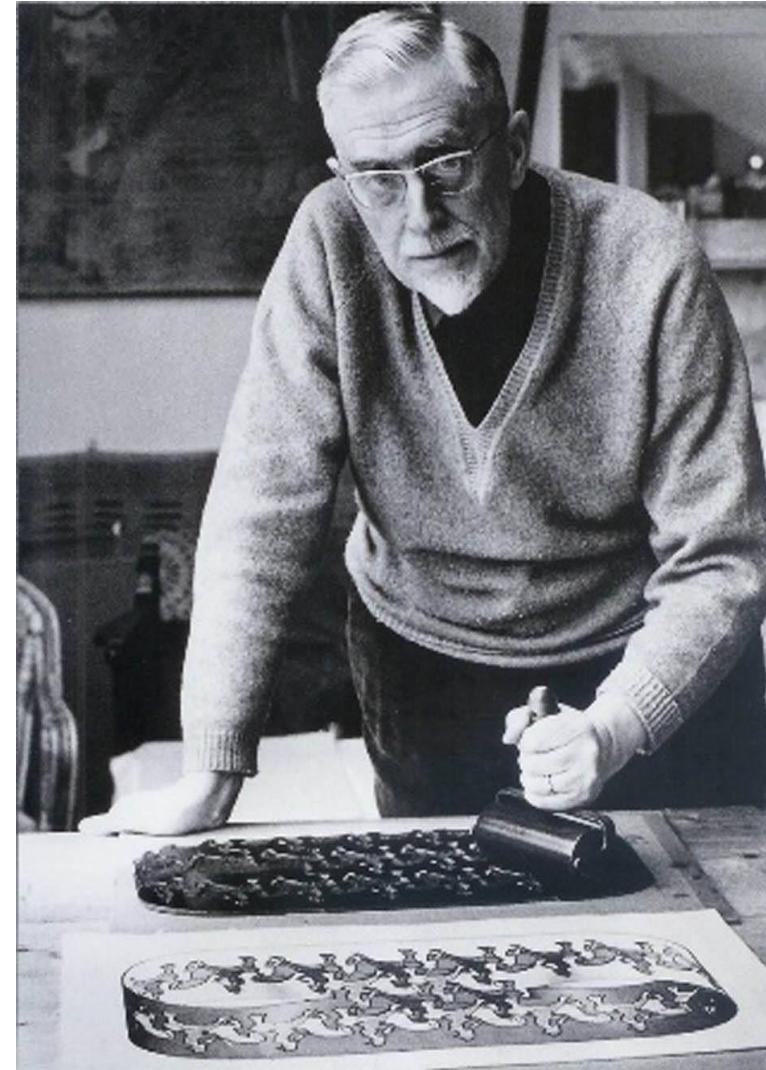


Maurits Cornelis Escher (1898-1972)

Leeuwarden, June 17



Escher

Graphical artist

Escher

Graphical artist

Optical illusions



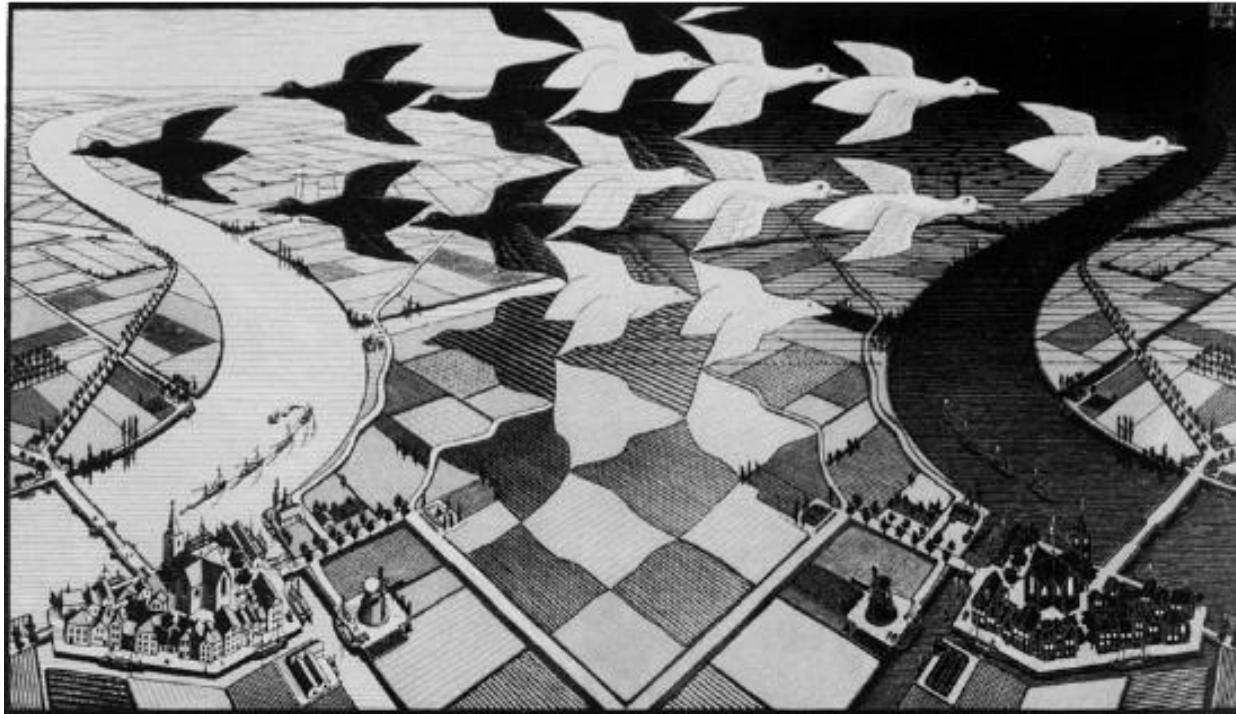
"Waterfall", 1961 Lithograph

Escher

Graphical artist

Optical illusions

Transformations



"Day and night", 1938 woodcut

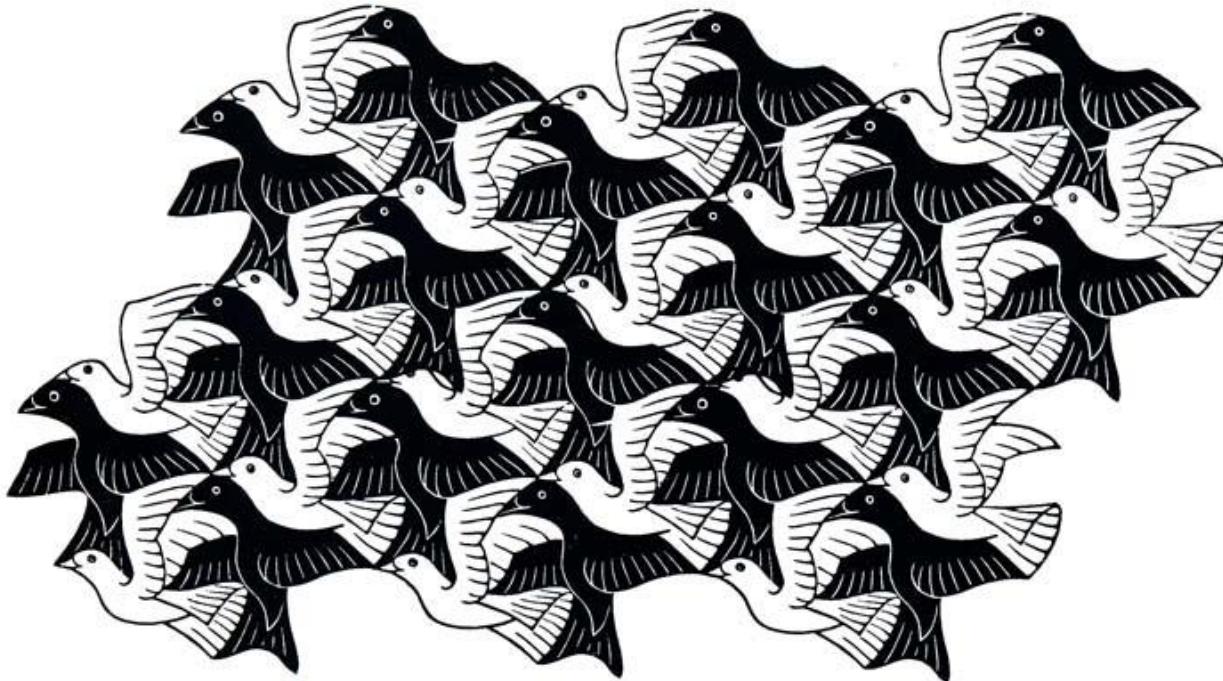
Escher

Graphical artist

Optical illusions

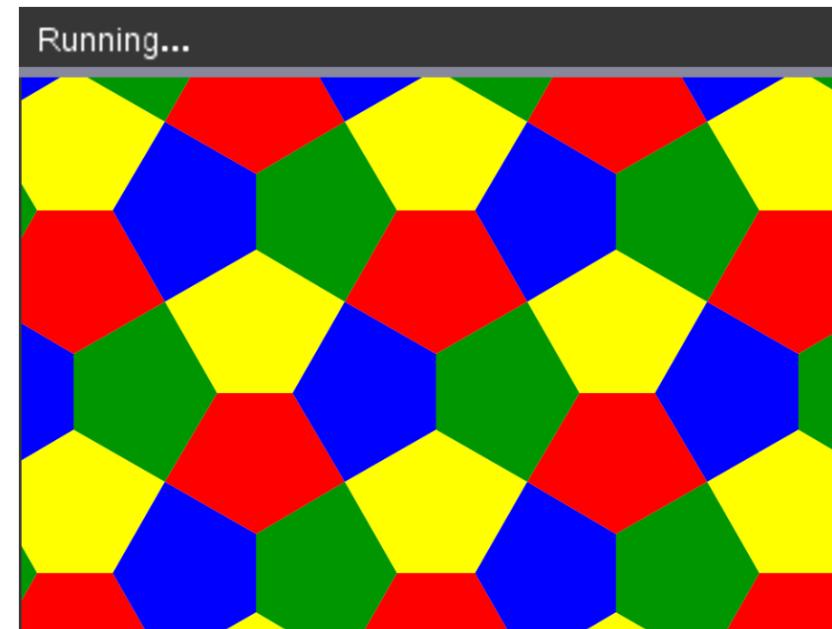
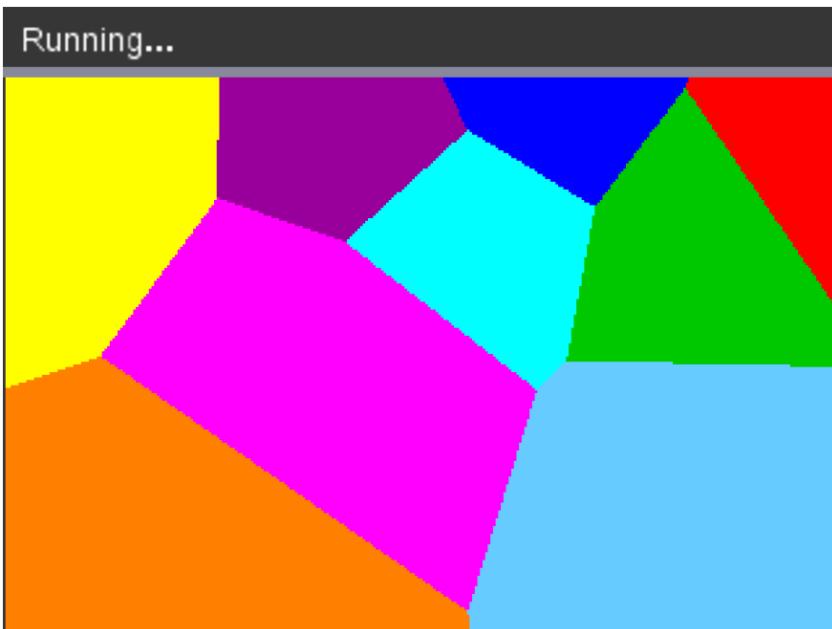
Transformations

Tessellations



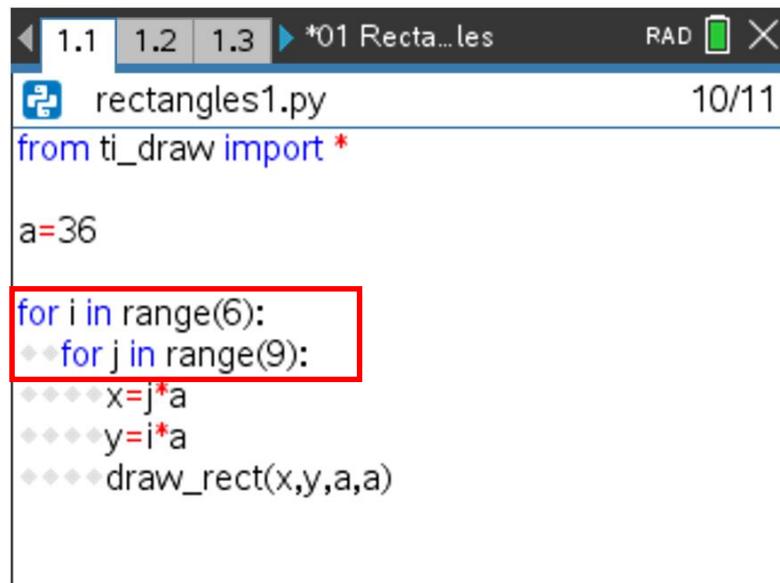
"Birds", 1949 woodcut

Tessellations



Regular tessellation

Rectangles



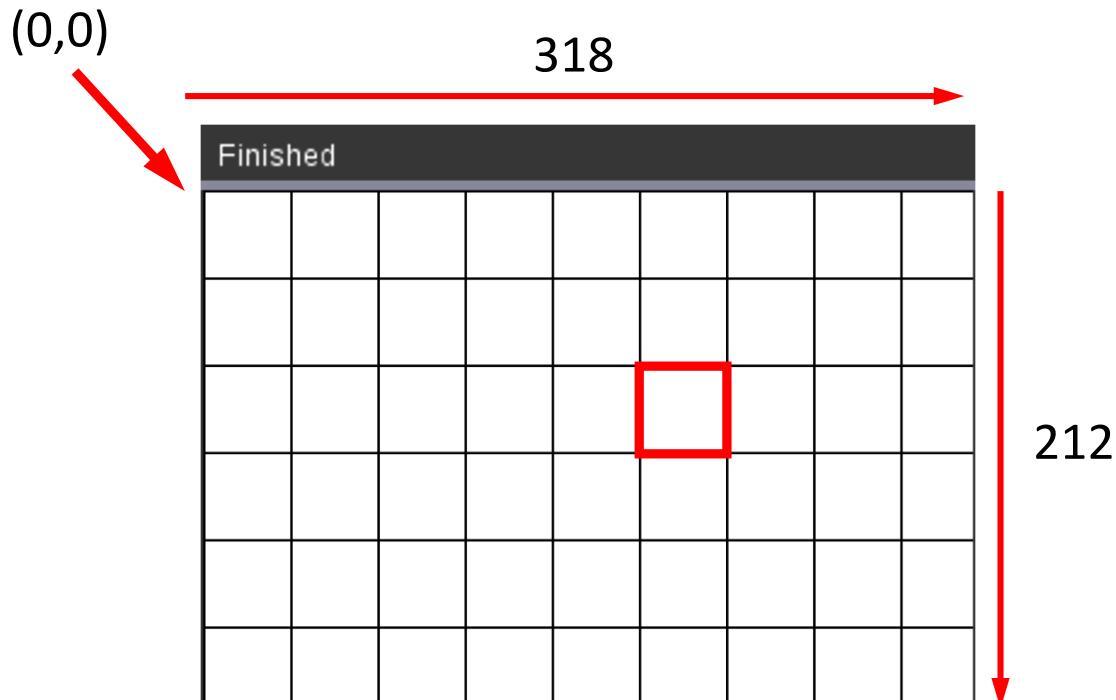
A screenshot of a Scratch workspace titled "01 Rectangles". The stage has a light blue background. The script area contains the following code:

```
from ti_draw import *
a=36
for i in range(6):
    for j in range(9):
        x=j*a
        y=i*a
        draw_rect(x,y,a,a)
```

The loop from `i` to 6 and `j` to 9 is highlighted with a red rectangle.

i : row number

j : column number



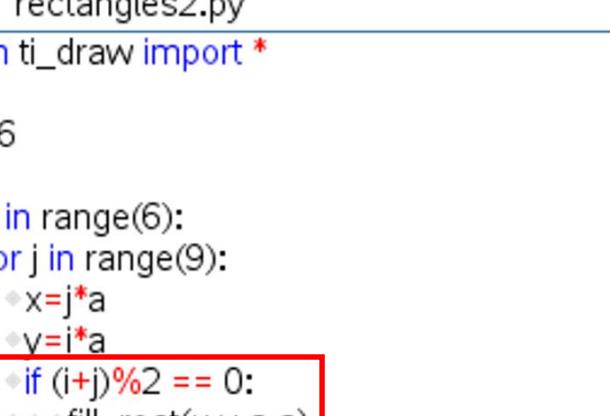
$$i = 2$$

$$j = 5$$

$$x = j \cdot 36 = 180$$

$$y = i \cdot 36 = 72$$

Rectangles

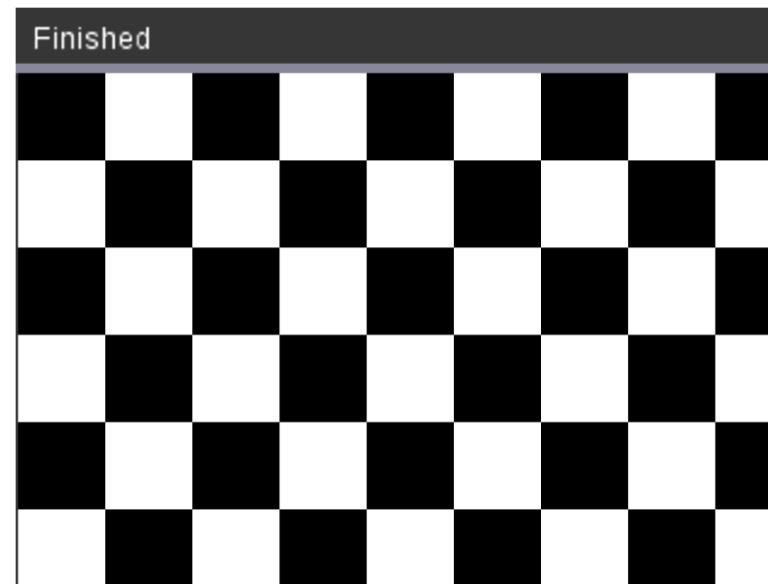


The image shows a TI-Nspire CX CAS calculator displaying a Python script for drawing rectangles. The script uses nested loops to iterate through a 6x9 grid. It calculates coordinates (x, y) based on indices i and j, where a is set to 36. It then checks if the sum of indices is even (using the condition $(i+j)\%2 == 0$). If true, it calls the `fill_rect` function to draw a rectangle at the calculated position with side length 'a'. The code is highlighted with a red box around the conditional statement.

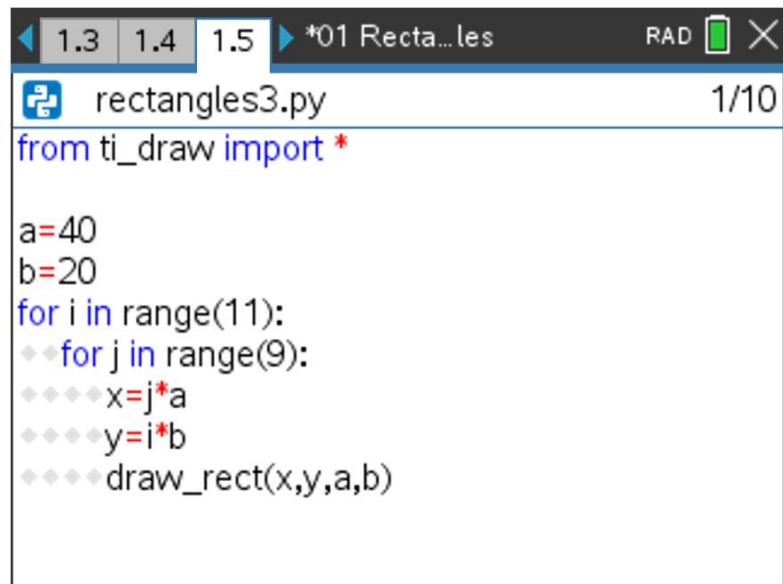
```
rectangles2.py
from ti_draw import *

a=36

for i in range(6):
    for j in range(9):
        x=j*a
        y=i*a
        if (i+j)%2 == 0:
            fill_rect(x,y,a,a)
```

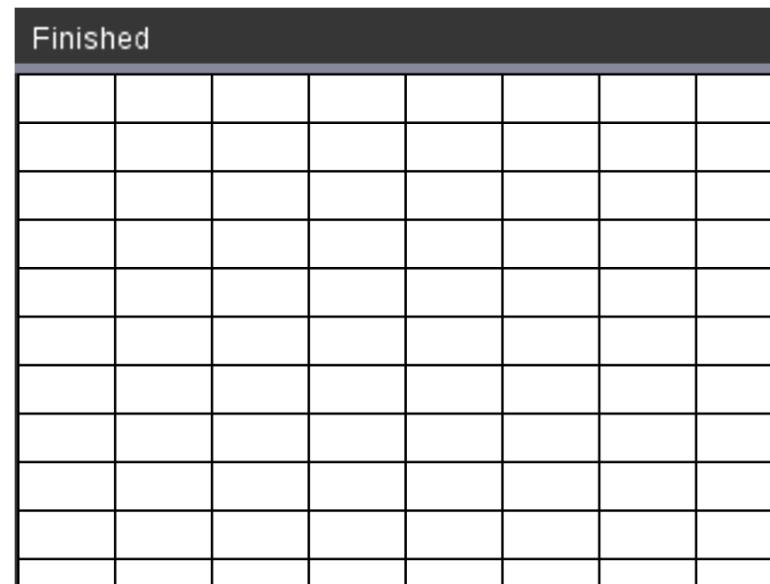


Rectangles



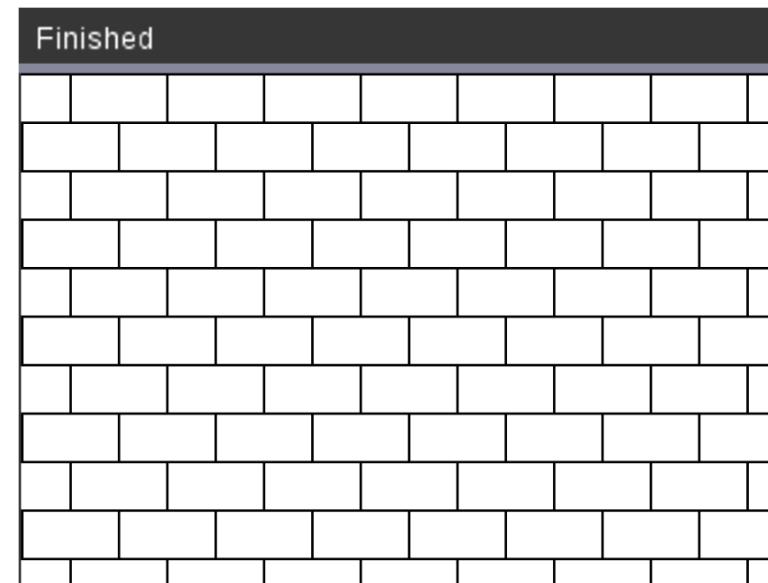
A screenshot of a Scratch workspace. At the top, there are navigation buttons (back, forward, search) and a title bar showing '1.3 | 1.4 | 1.5' and '*01 Rectangles'. Below the title bar, the script name 'rectangles3.py' is displayed next to a script icon. To the right, the score '1/10' is shown. The script itself contains the following code:

```
from ti_draw import *
a=40
b=20
for i in range(11):
    for j in range(9):
        x=j*a
        y=i*b
        draw_rect(x,y,a,b)
```



Rectangles

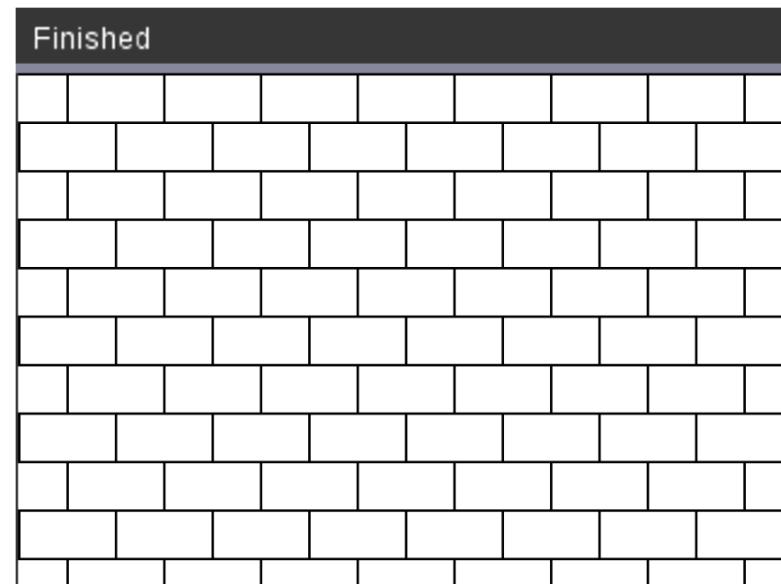
```
◀ 1.6 1.7 1.8 ▶ *01 Rectangles RAD ✚
rectangles4.py 2/12
from ti_draw import *
a=40
b=20
for i in range(11):
    for j in range(9):
        x=j*a-1
        if i%2 == 0:
            x = x-a/2
        y=i*b
        draw_rect(x,y,a,b)
```



p % q returns the remainder of p divided by q

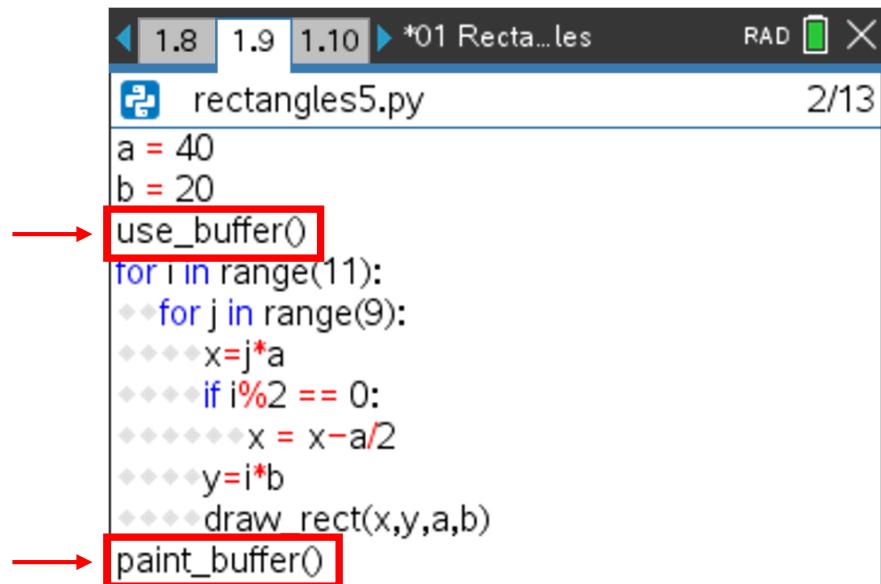
Rectangles

```
1.6 1.7 1.8 *01 Rectangles RAD X
rectangles4.py 2/12
from ti_draw import *
a=40
b=20
for i in range(11):
    for j in range(9):
        x=j*a-1
        if i%2 == 0:
            x = x-a/2
        y=i*b
        draw_rect(x,y,a,b)
```



even
odd
even
odd
even
odd
even
odd
even
odd
even
odd

Rectangles

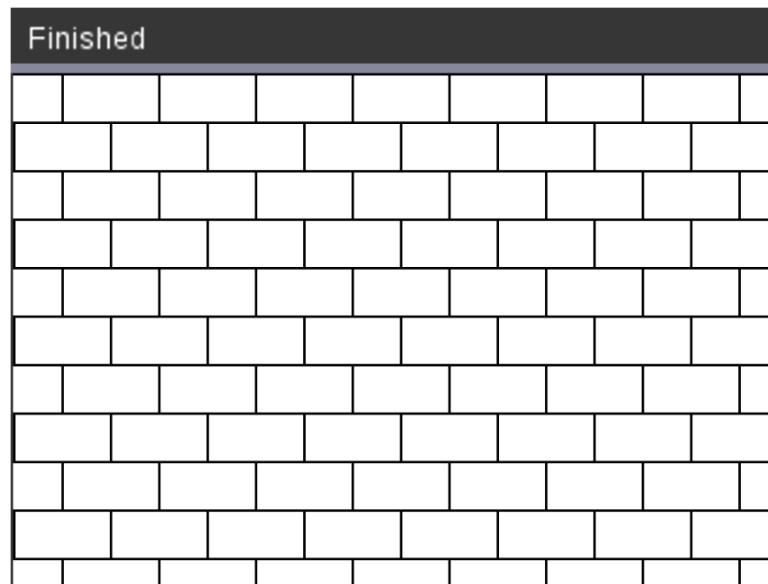


The screenshot shows a microcontroller development environment with the following details:

- Top bar: Version 1.8, 1.9, 1.10, *01 Rectangles, RAD, battery icon, close button.
- File list: rectangles5.py (selected), 2/13.
- Code area:

```
a = 40
b = 20
use_buffer()
for i in range(11):
    for j in range(9):
        x=j*a
        if i%2 == 0:
            x = x-a/2
        y=i*b
        draw_rect(x,y,a,b)
paint_buffer()
```

Two red arrows point to the function calls `use_buffer()` and `paint_buffer()`, which are highlighted with red boxes.



Polygons

`draw_rect (x, y, width, height)`

`fill_rect (x, y, width, height)`

`draw_poly (xlist, ylist)`

`fill_poly (xlist, ylist)`

Polygons

`draw_rect (x, y, width, height)`

`fill_rect (x, y, width, height)`

`draw_poly (xlist, ylist)`

`fill_poly (xlist, ylist)`

Polygons

Module tessellations

`multiply (polygon, factor)`

`rotate (polygon, angle)`

`mirror_hor (polygon)`

`mirror_vert (polygon)`

`draw (x, y, polygon)`

`fill (x, y, polygon)`

`x_correction (polygon)`

`y_correction (polygon)`

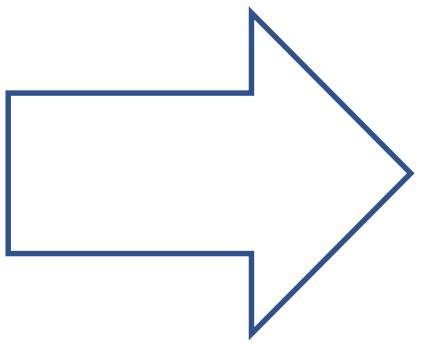
}

isometric grid

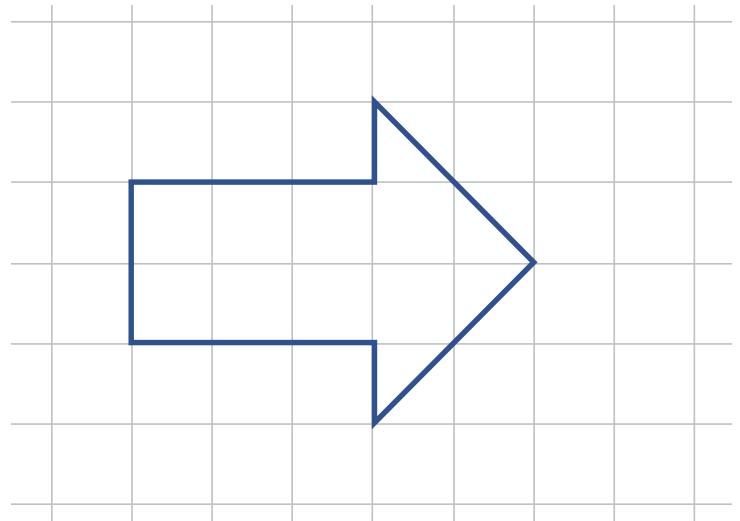
Polygon is a list with vertices

Window(-159,159,-106,106)

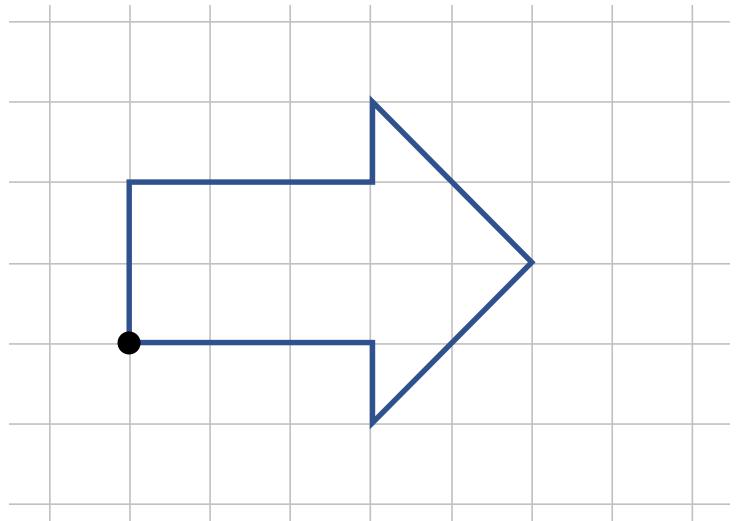
Polygons



Polygons

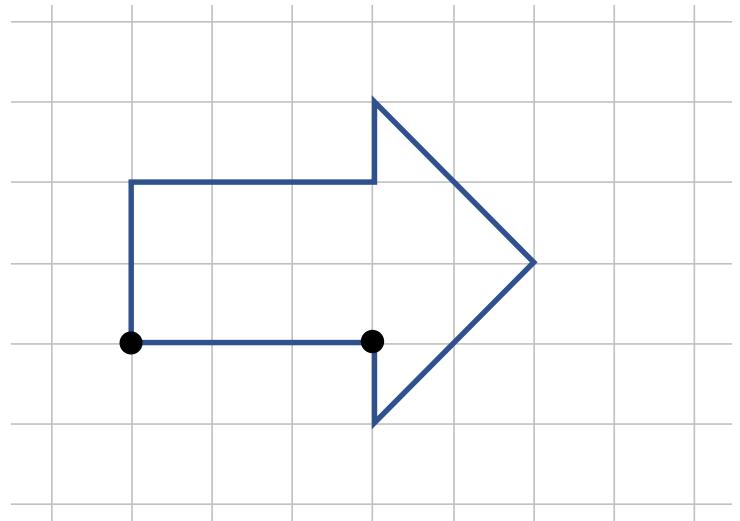


Polygons



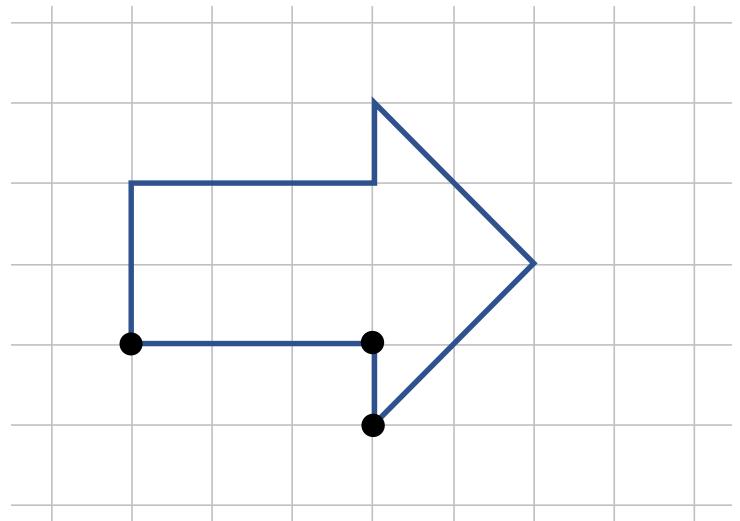
arrow = [(0,0)

Polygons



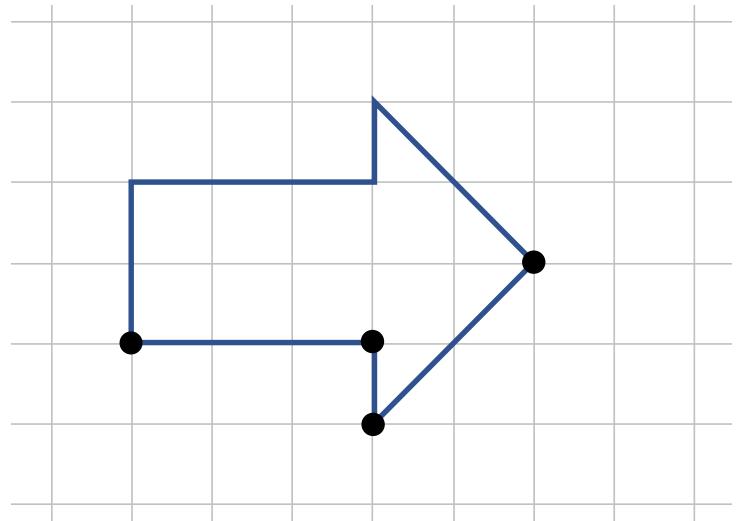
arrow = [(0,0), (3,0)]

Polygons



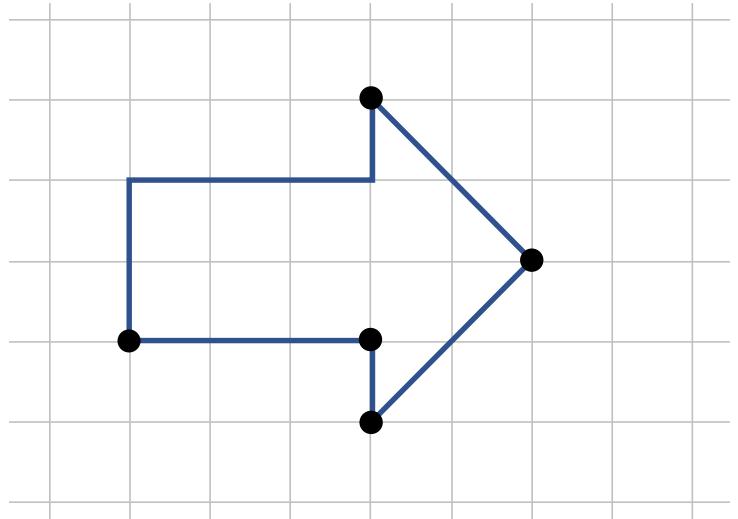
```
arrow = [ (0,0), (3,0), (3,-1)]
```

Polygons



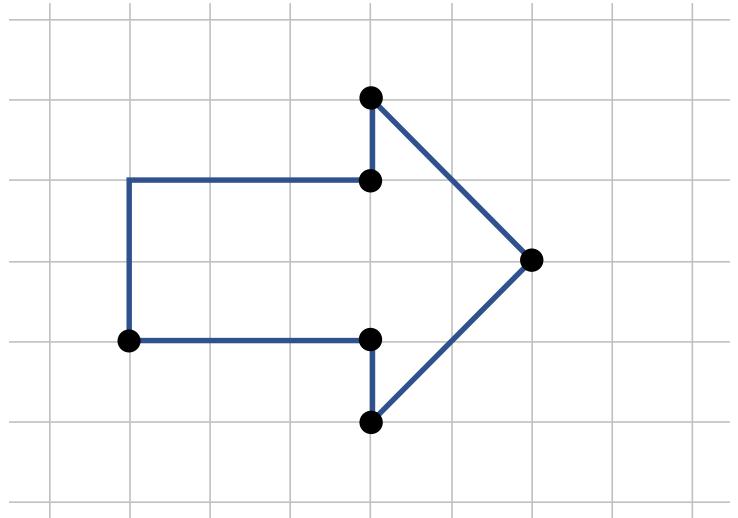
```
arrow = [ (0,0), (3,0), (3,-1), (5,1) ]
```

Polygons



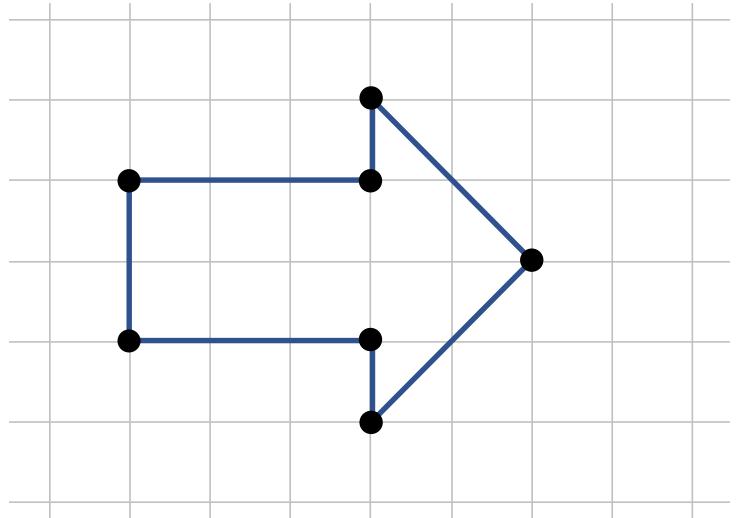
```
arrow = [ (0,0), (3,0), (3,-1), (5,1), (3,3) ]
```

Polygons



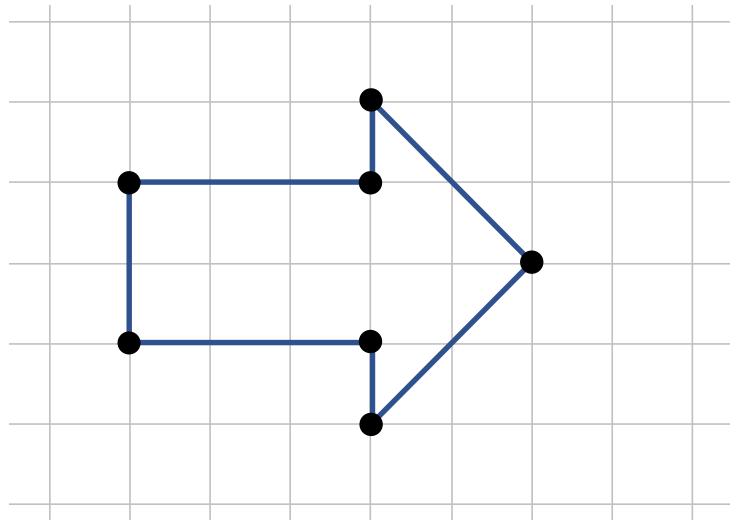
```
arrow = [ (0,0), (3,0), (3,-1), (5,1), (3,3), (3,2) ]
```

Polygons



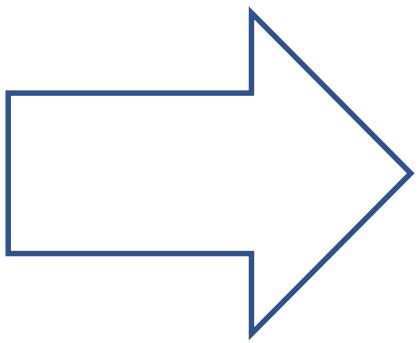
```
arrow = [ (0,0), (3,0), (3,-1), (5,1), (3,3), (3,2), (0,2) ]
```

Polygons



```
arrow = [ (0,0), (3,0), (3,-1), (5,1), (3,3), (3,2), (0,2), (0,0)]
```

Polygons



```
arrow = [ (0,0), (3,0), (3,-1), (5,1), (3,3), (3,2), (0,2), (0,0)]
```

Polygons

◀ 1.1 1.2 1.3 ▶ *02 Arrows RAD ☰ ×

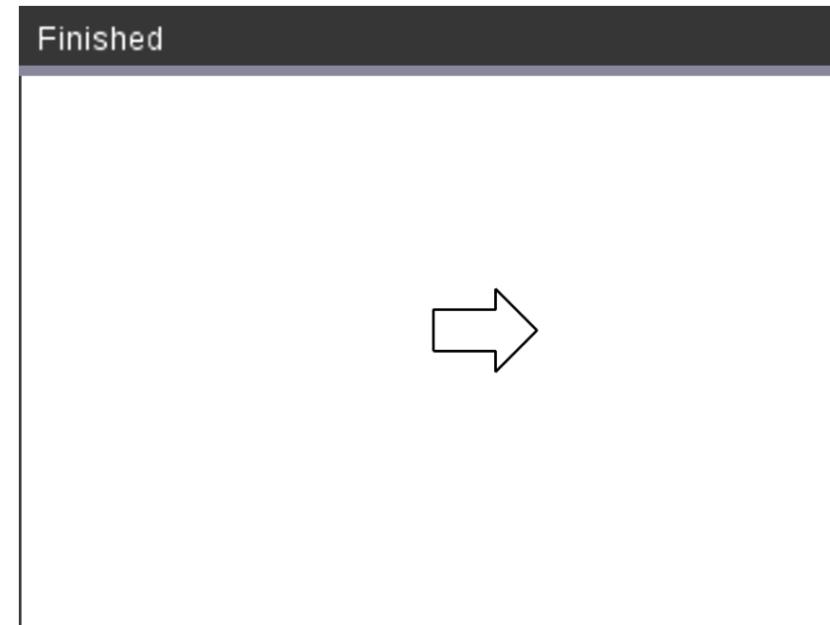
arrows1.py 8/10

```
from tessellation import *

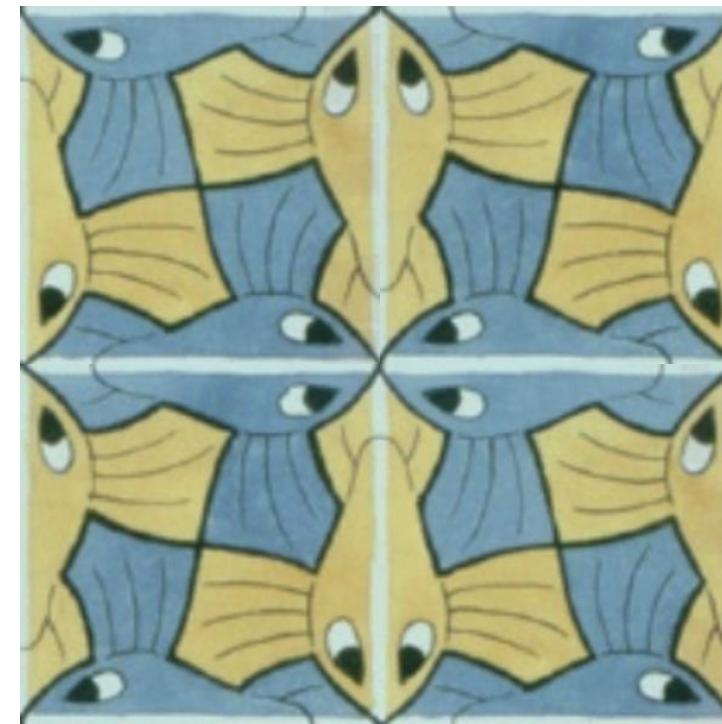
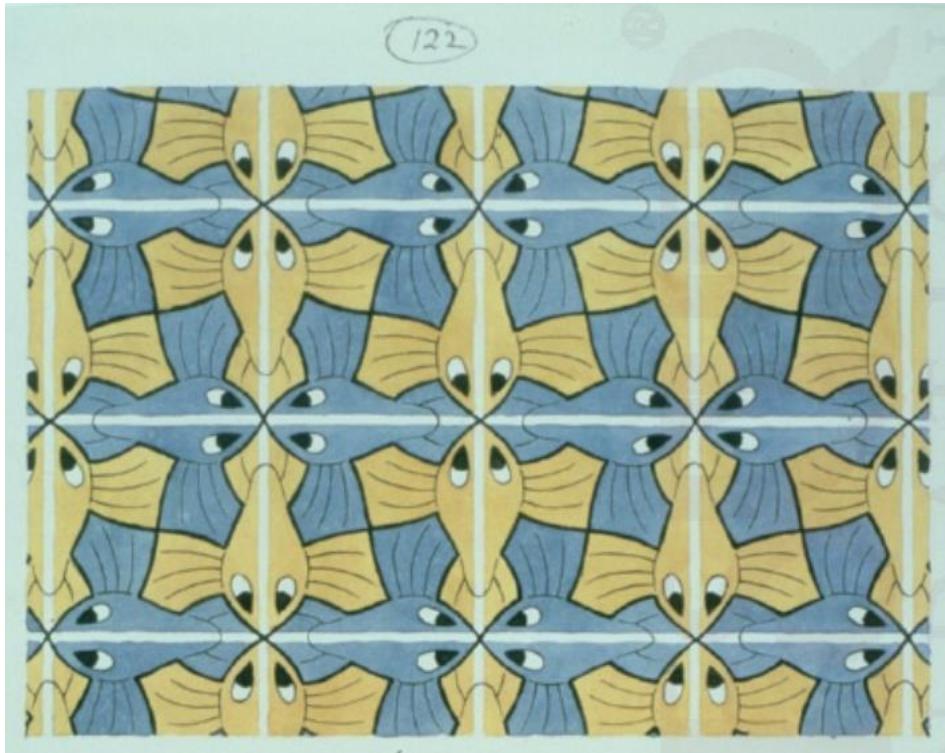
arrow = [
    (0,0), (3,0), (3,-1), (5,1),
    (3,3), (3,2), (0,2), (0,0)]

arrow = multiply(arrow, 8)

draw(0, 0, arrow)
```

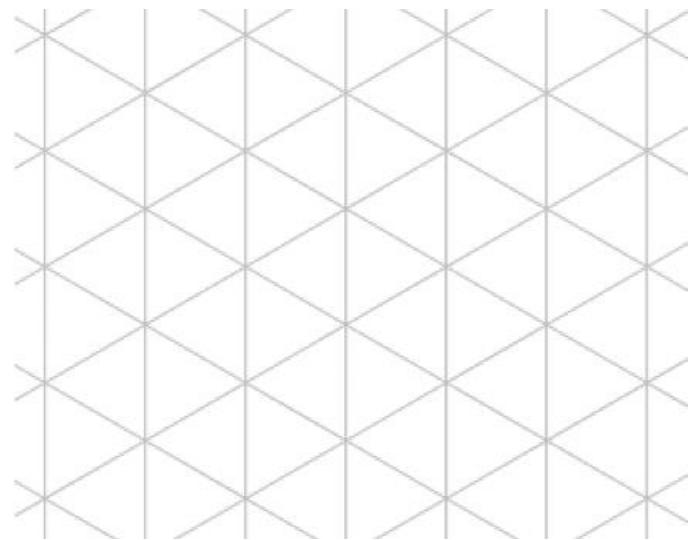


Polygons

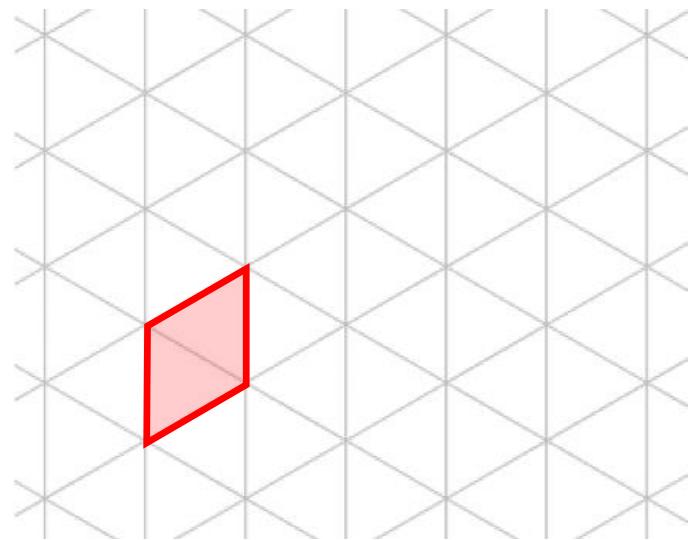


Example 1

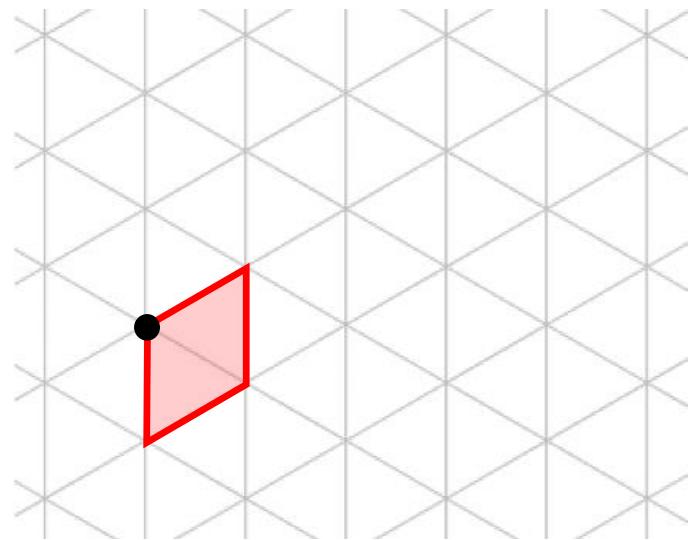
Isometric grid



Isometric grid

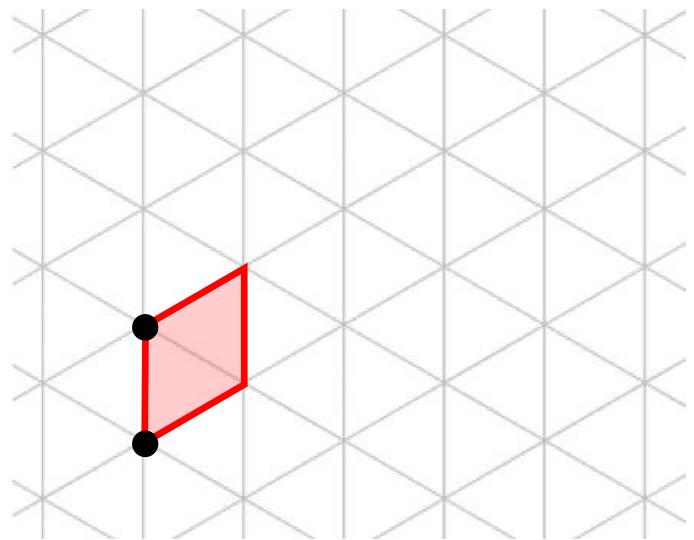


Isometric grid



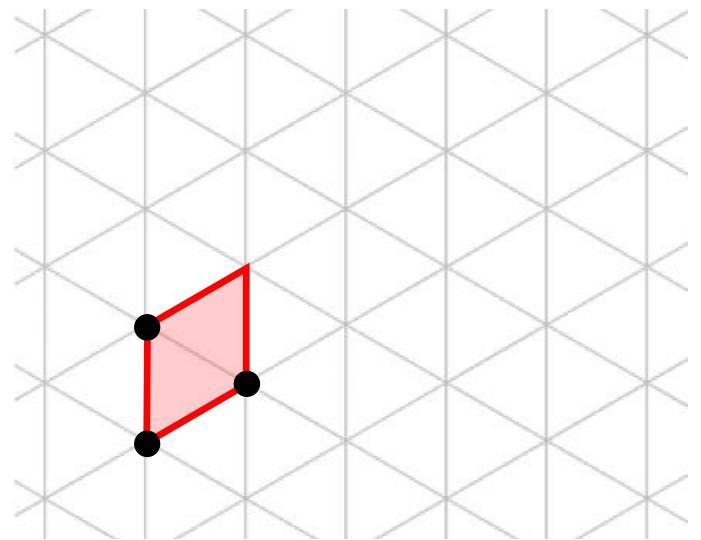
kite = [(0, 0)

Isometric grid



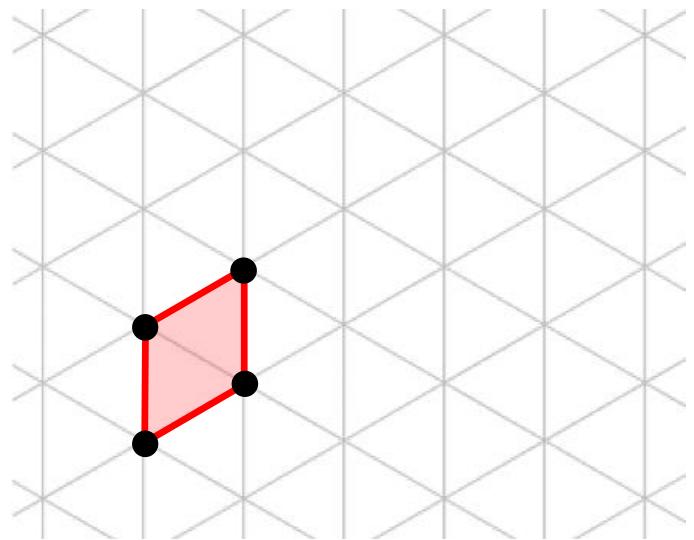
kite = [(0,0), (0,-1)

Isometric grid



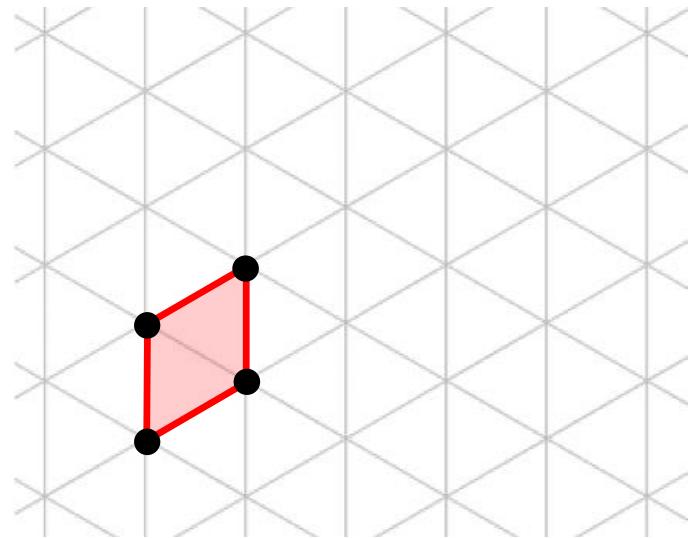
$$\text{kite} = [(0, 0), (0, -1), \left(\frac{1}{2}\sqrt{3}, -\frac{1}{2}\right)]$$

Isometric grid



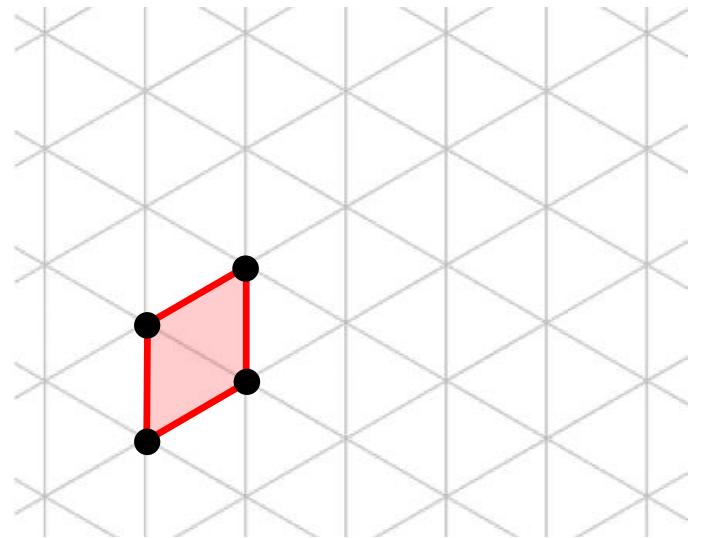
$$\text{kite} = [(0, 0), (0, -1), \left(\frac{1}{2}\sqrt{3}, -\frac{1}{2}\right), \left(\frac{1}{2}\sqrt{3}, \frac{1}{2}\right)]$$

Isometric grid



kite = [(0, 0), (0, -1), $(\frac{1}{2}\sqrt{3}, -\frac{1}{2})$, $(\frac{1}{2}\sqrt{3}, \frac{1}{2})$, (0, 0)]

Isometric grid

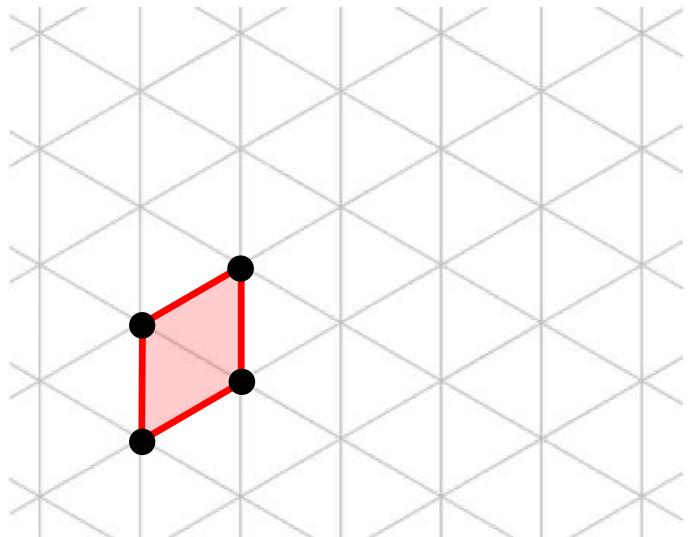


kite = [(0,0), (0,-1), $\left(\frac{1}{2}\sqrt{3}, -\frac{1}{2}\right)$, $\left(\frac{1}{2}\sqrt{3}, \frac{1}{2}\right)$, (0,0)]

kite = [(0,0), (0,-1), $(1, -\frac{1}{2})$, $(1, \frac{1}{2})$, (0,0)]

kite = x_correction(kite)

Isometric grid



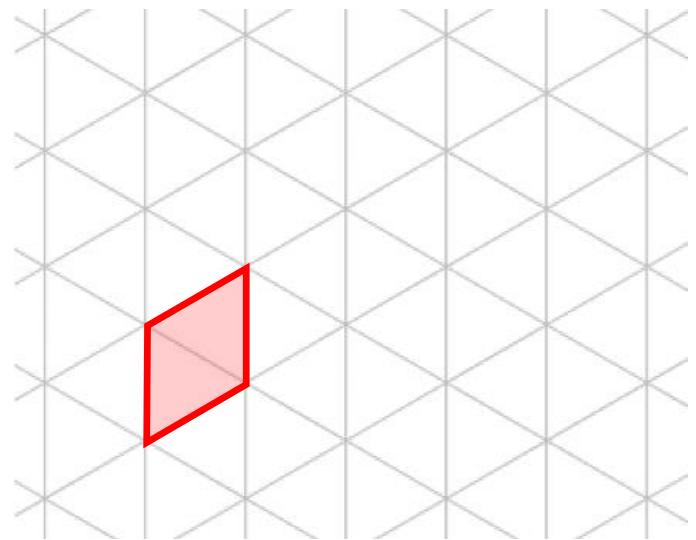
```
kite = [ (0,0), (0,-1), (1, -½), (1, ½), (0,0) ]
```



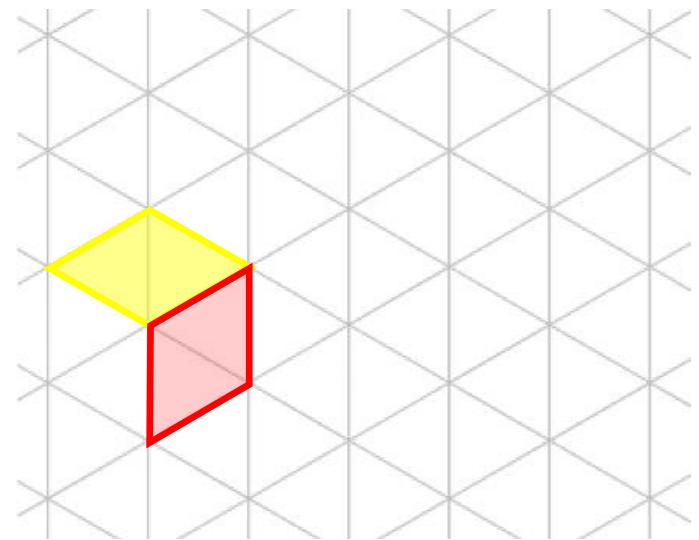
```
kite = x_correction( kite )
```

```
kite = [ (0,0), (0,-1), (½√3, -½), (½√3, ½), (0,0) ]
```

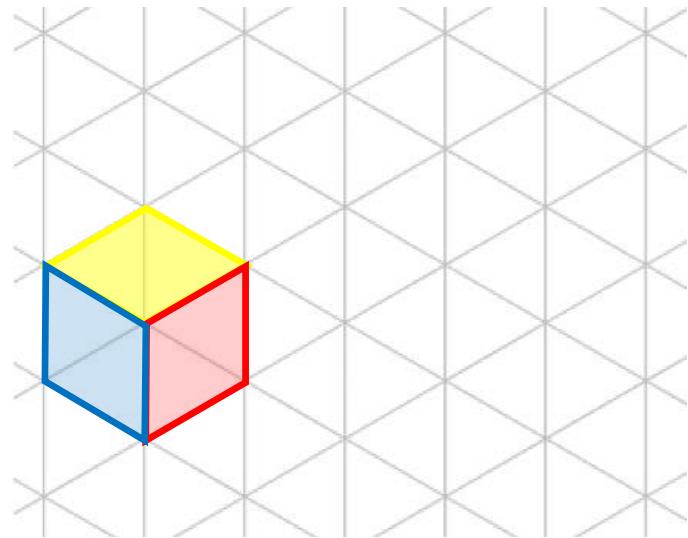
Isometric grid



Isometric grid



Isometric grid

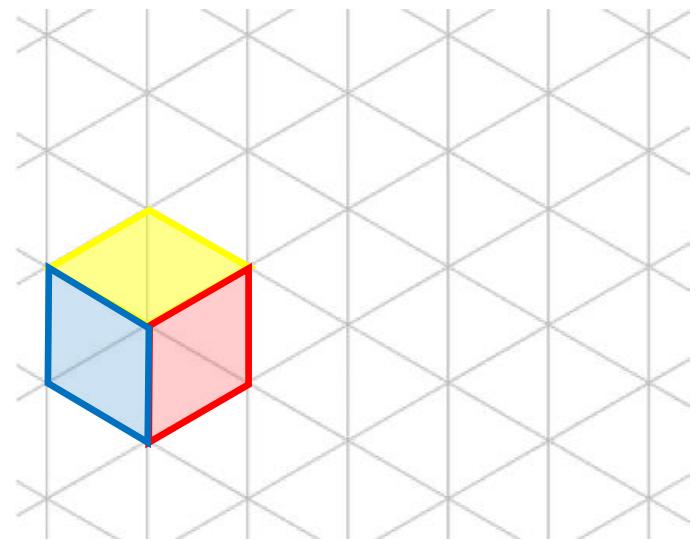


```
kite = [(0,0), (0,-1), (1,-0.5), (1,0.5), (0,0)]  
kite = x_correction(kite)
```

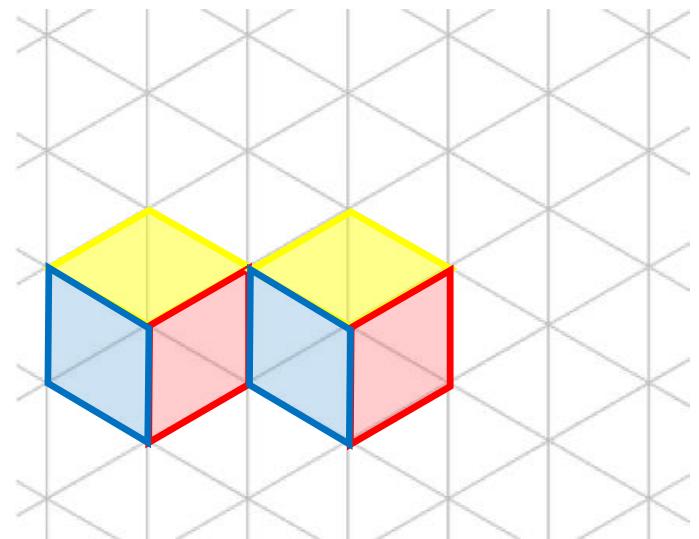
```
colors = [(255,0,0), (255,255,0), (0,0,255)]
```

```
def hexagon(x,y):  
    for i in range(3):  
        kyte = rotate(kite, 120*i)  
        set_color(colors[i])  
        fill(x,y,kyte)
```

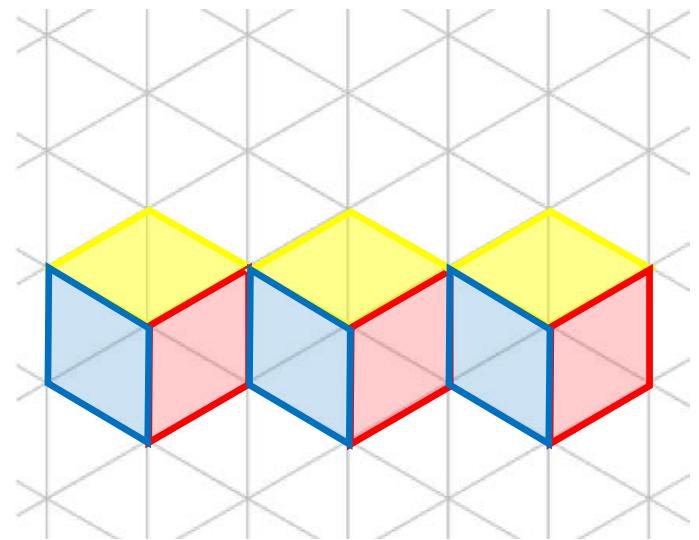
Isometric grid



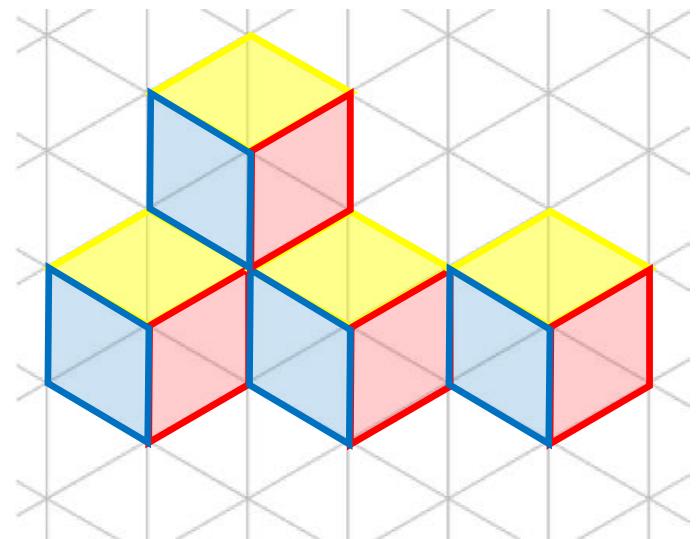
Isometric grid



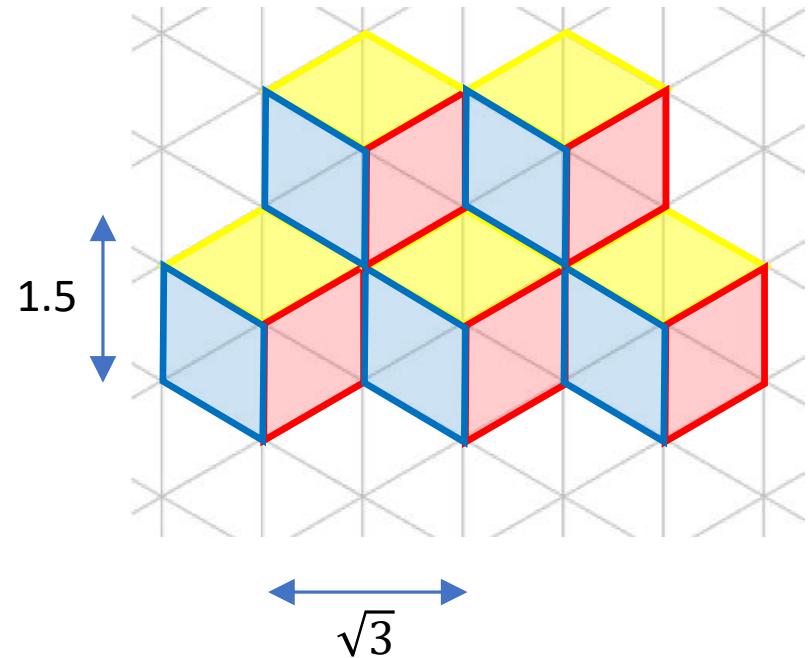
Isometric grid



Isometric grid



Isometric grid

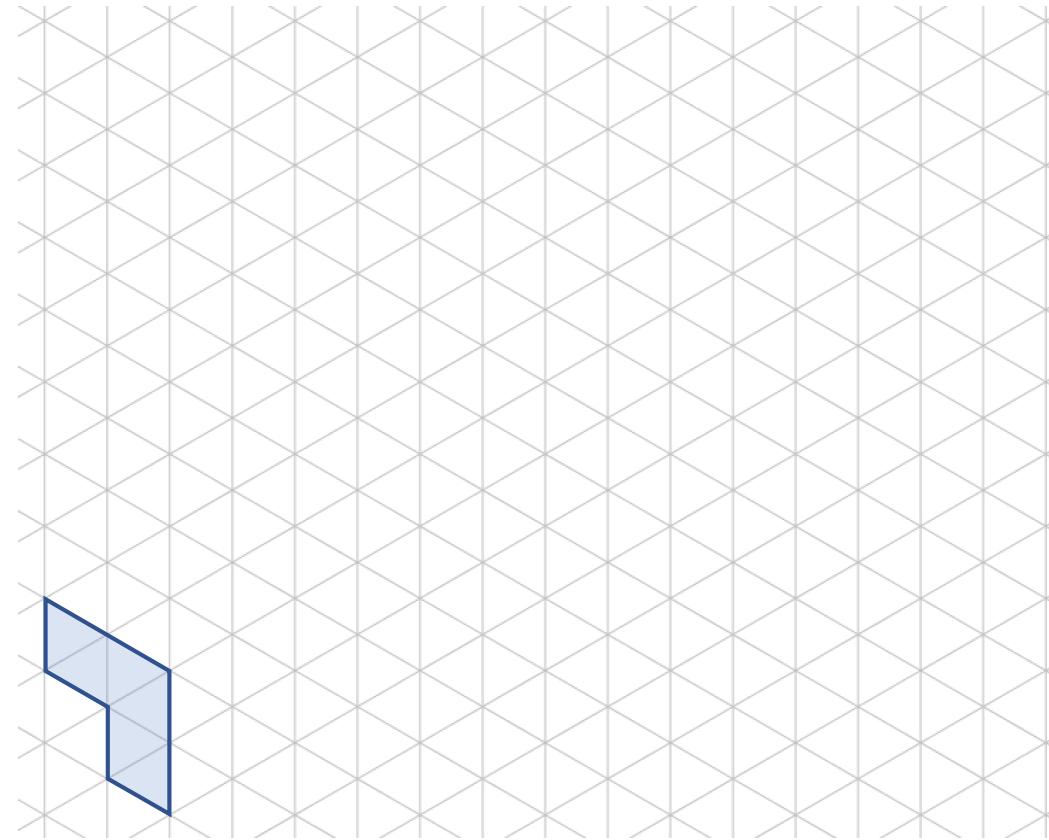


```
for i in range(-2,3):
    for j in range(-3,4):
        x = j*sqrt(3)*a
        if i%2 == 0:
            x = x+0.5*sqrt(3)*a
        y = i*1.5*a
        hexagon(x,y)
```

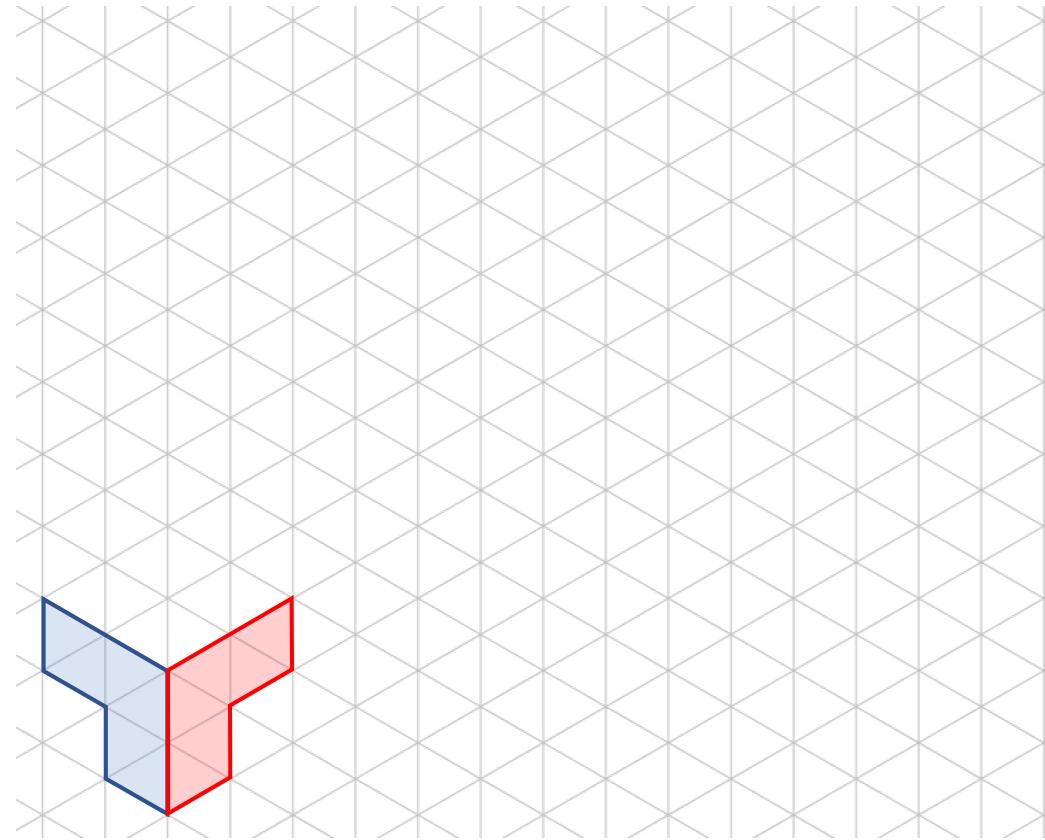
T-shapes



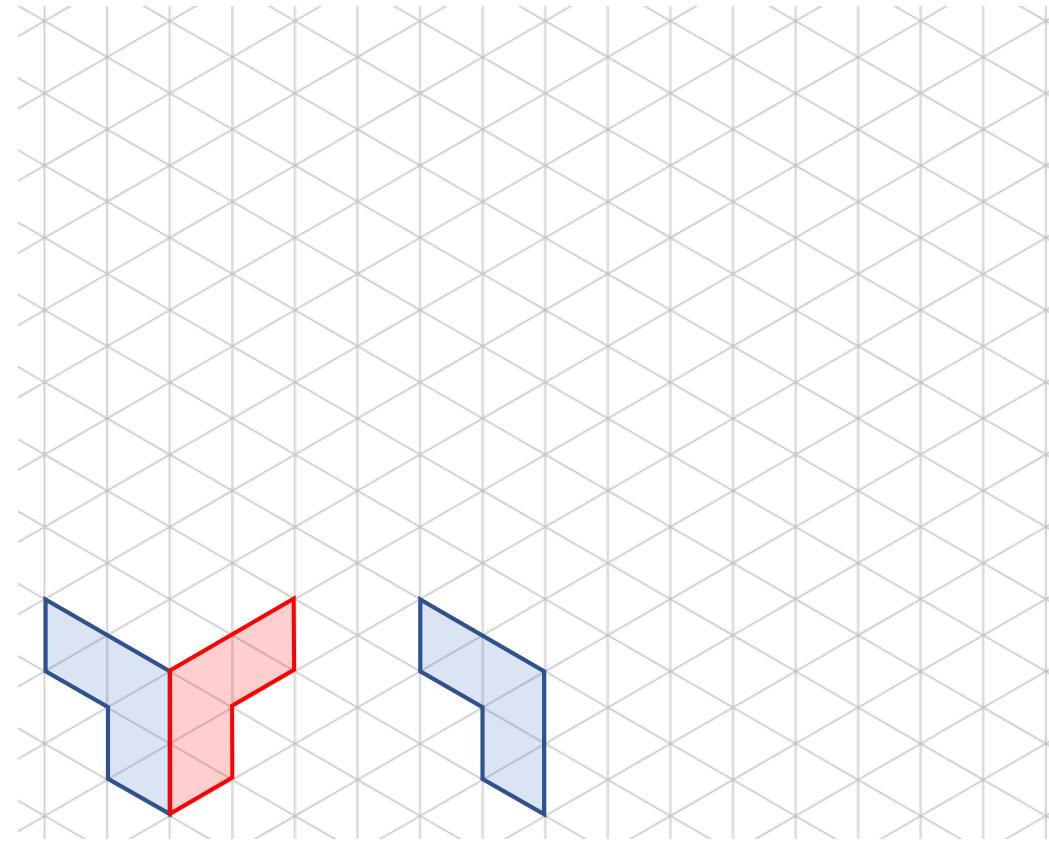
T-shapes



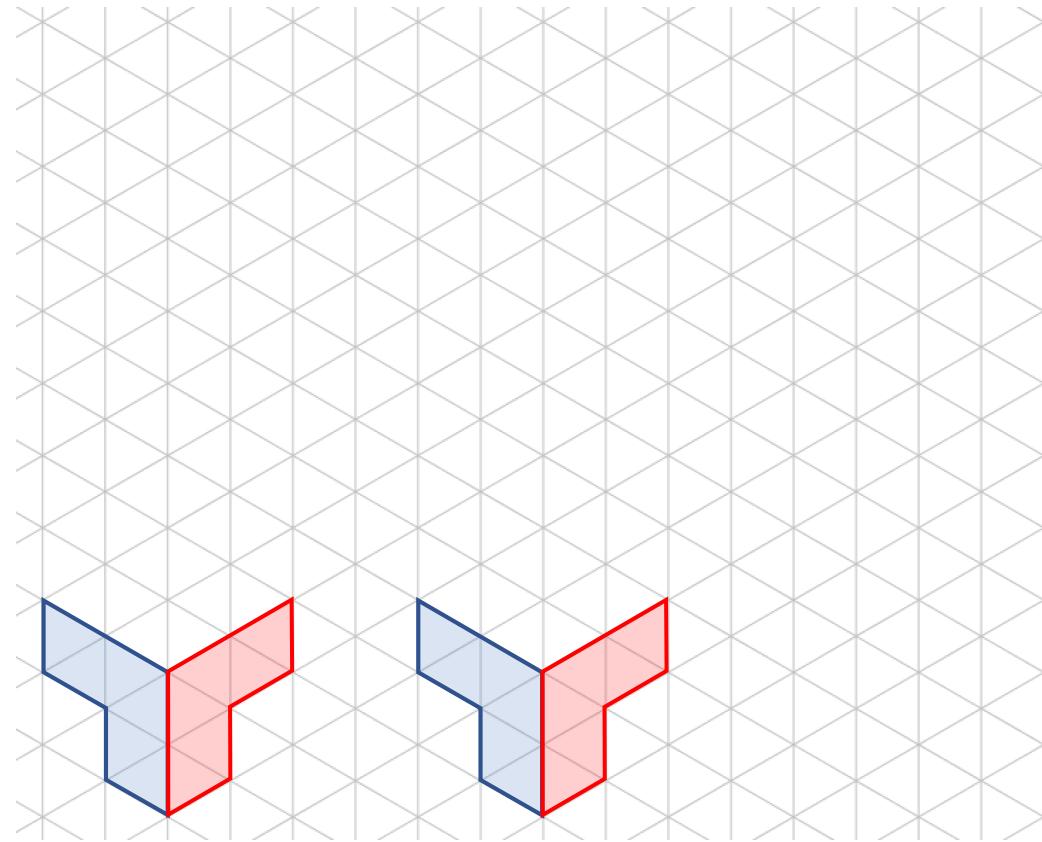
T-shapes



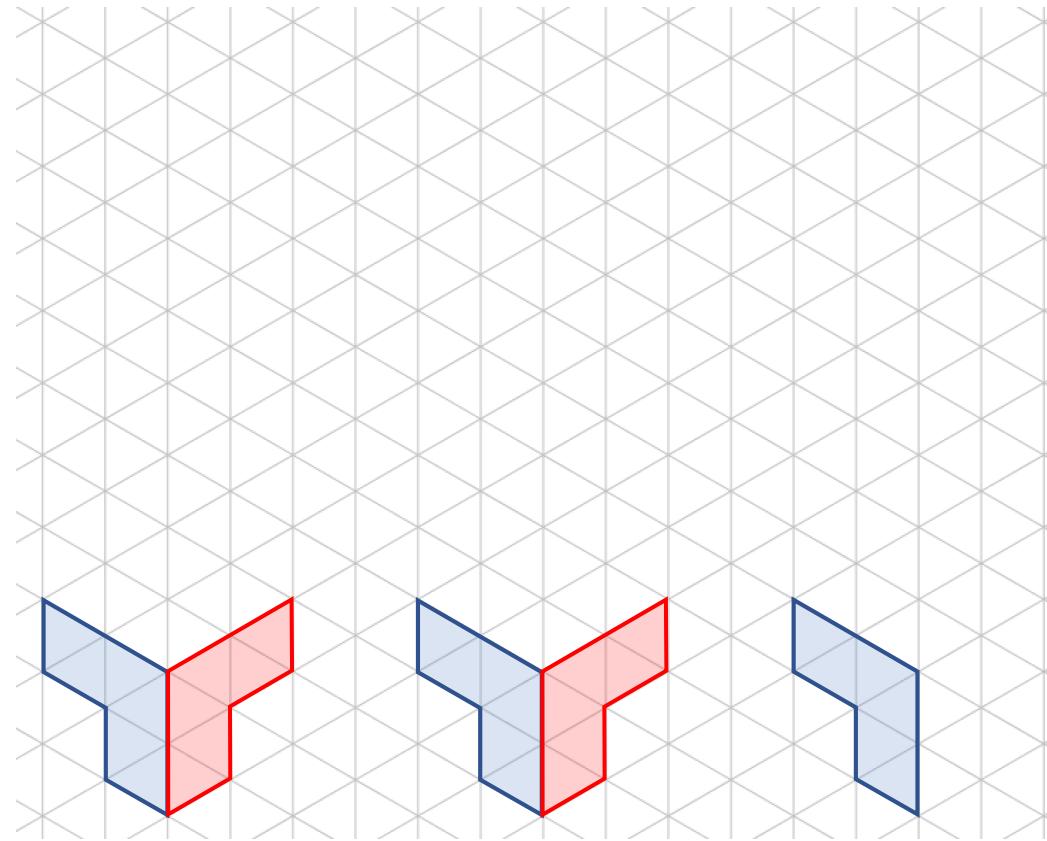
T-shapes



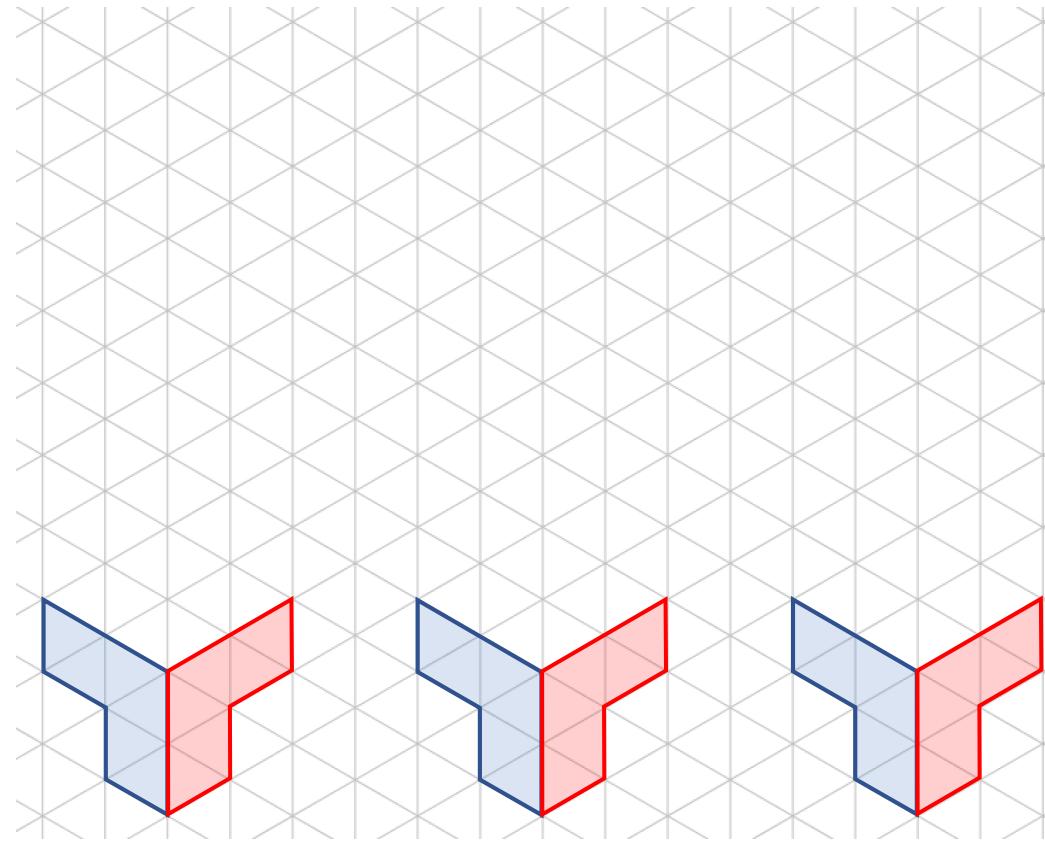
T-shapes



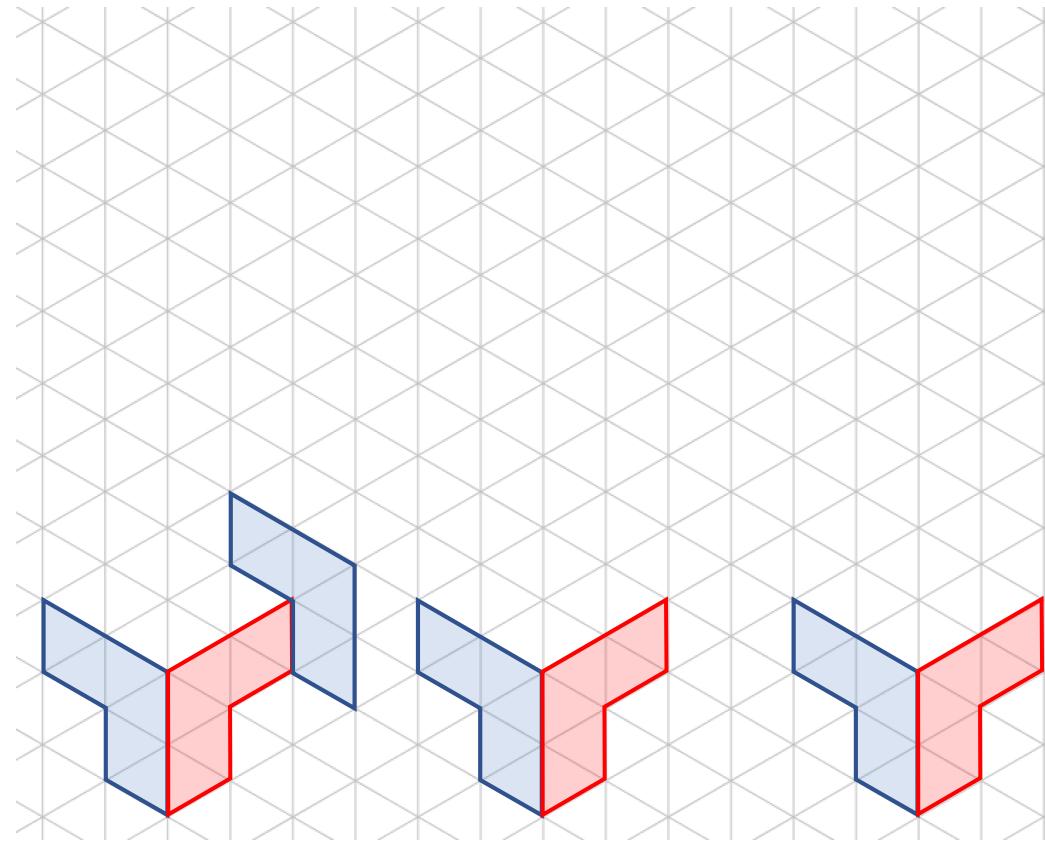
T-shapes



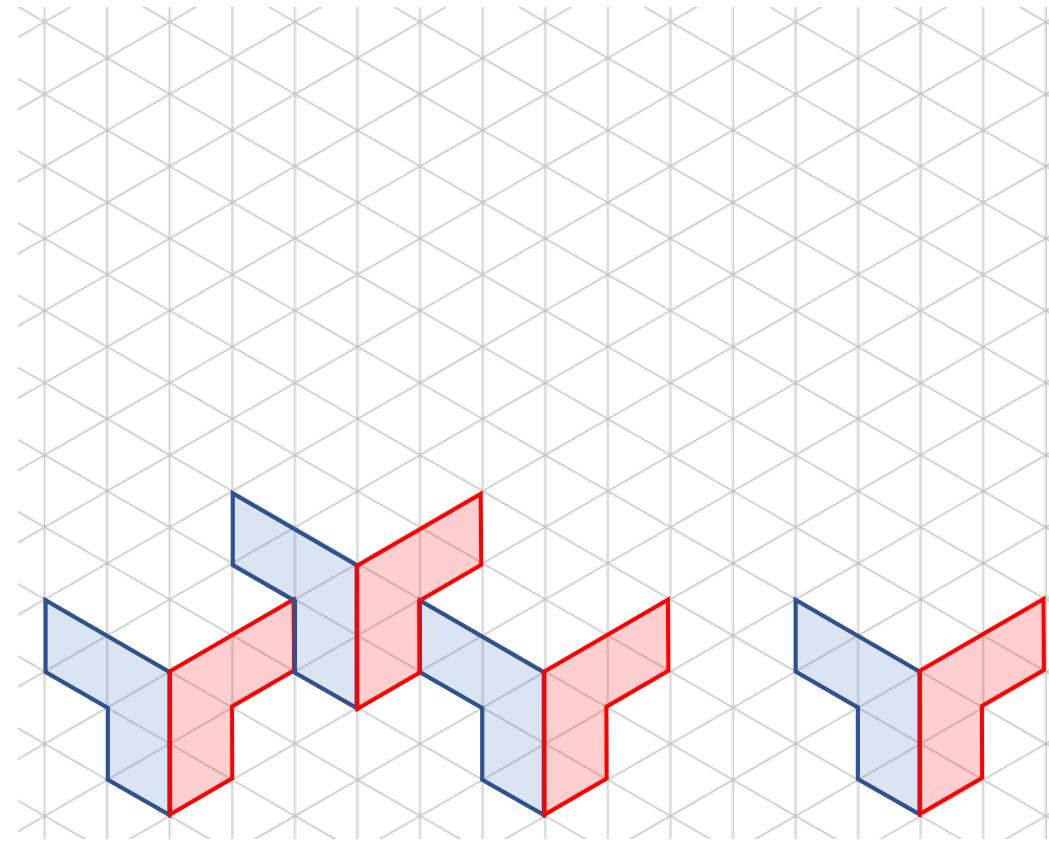
T-shapes



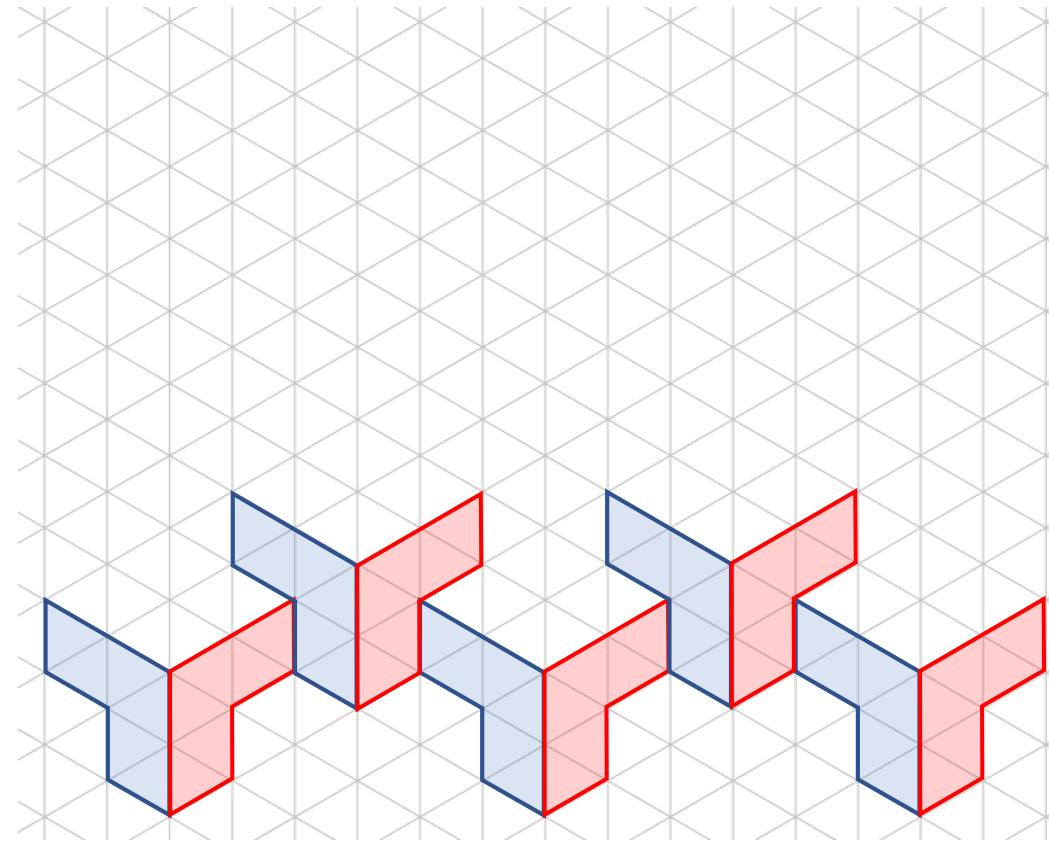
T-shapes



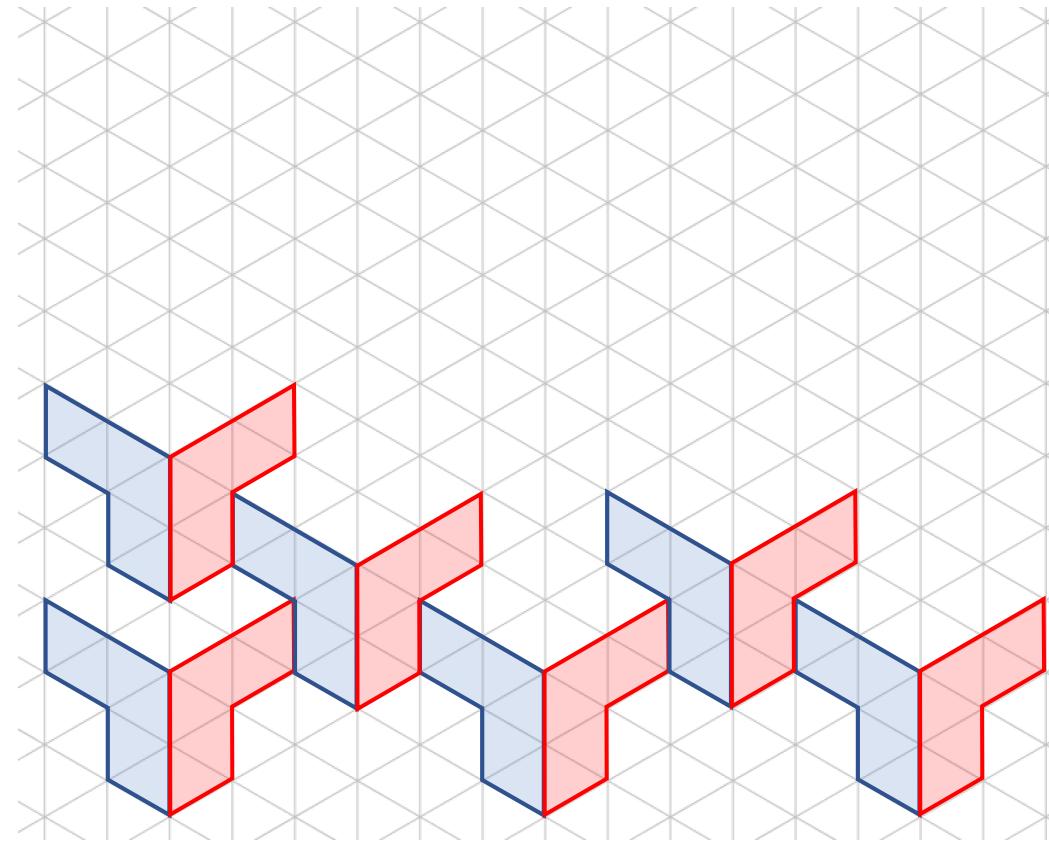
T-shapes



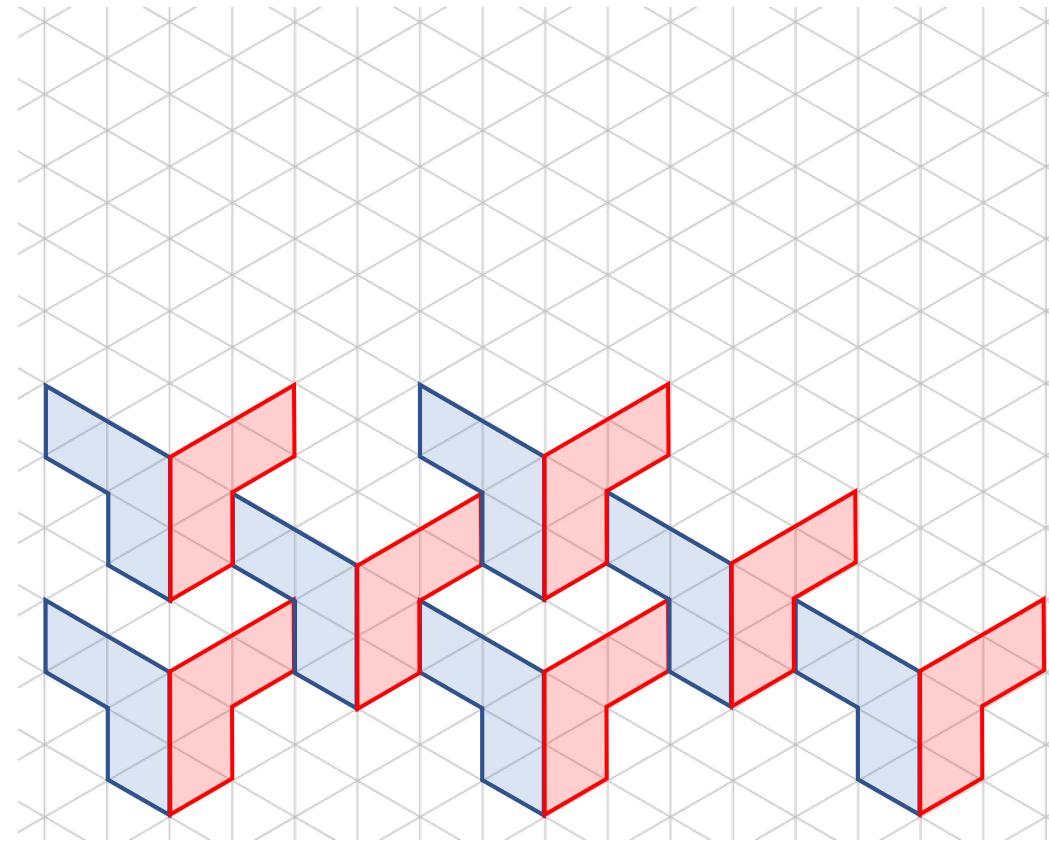
T-shapes



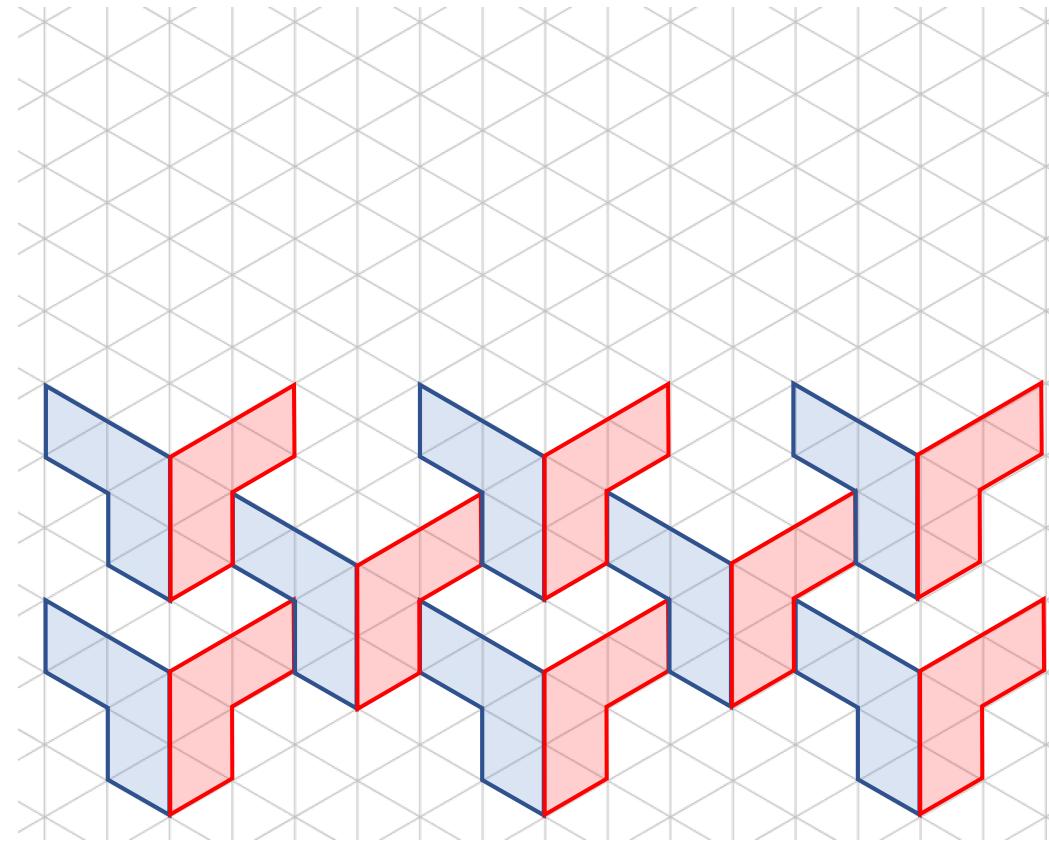
T-shapes



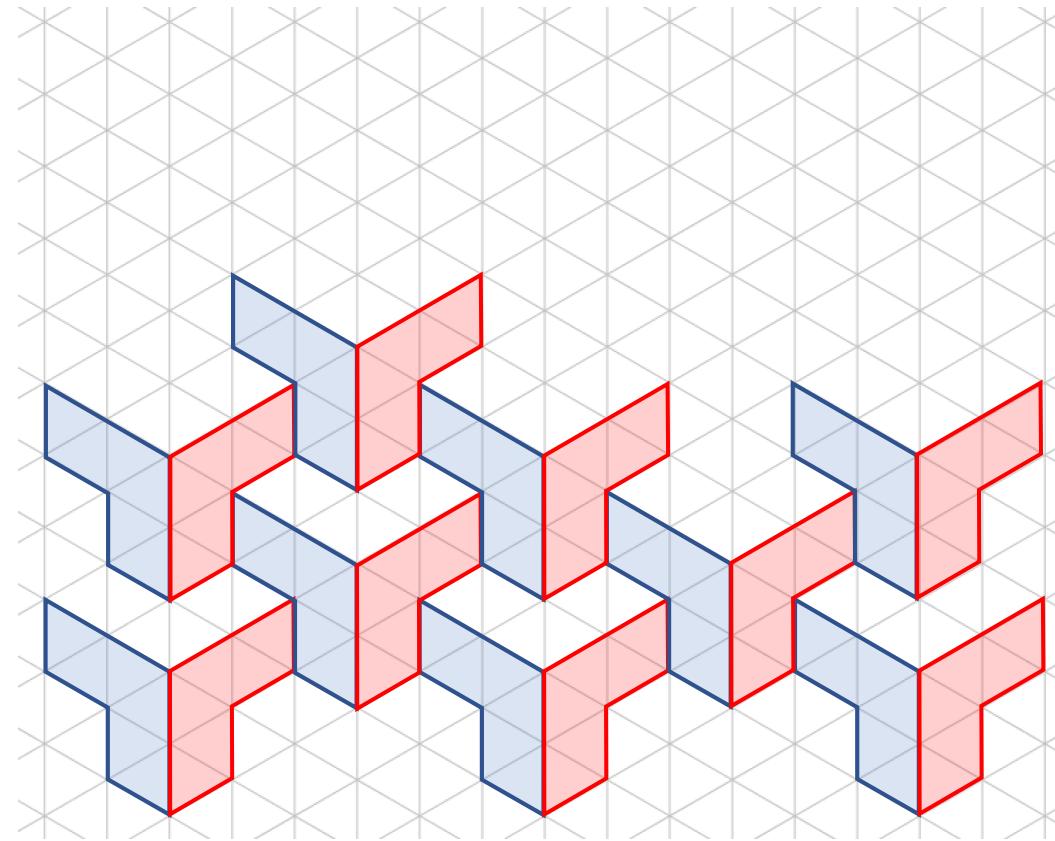
T-shapes



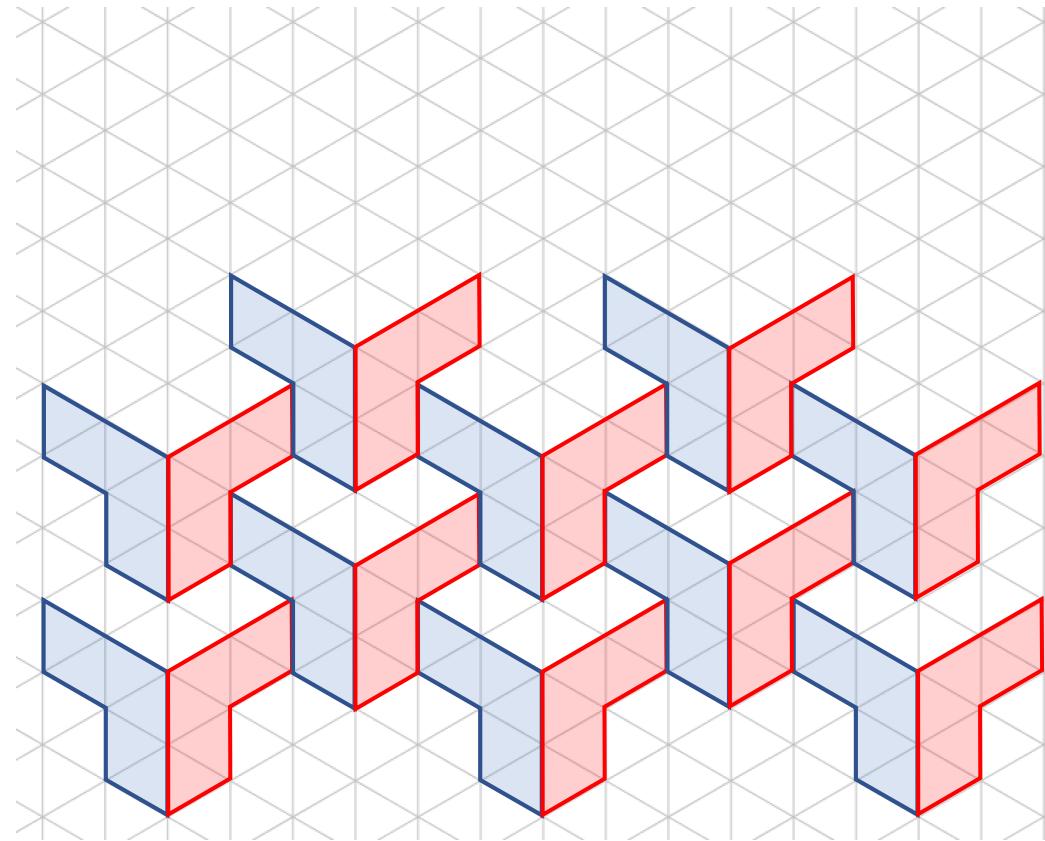
T-shapes



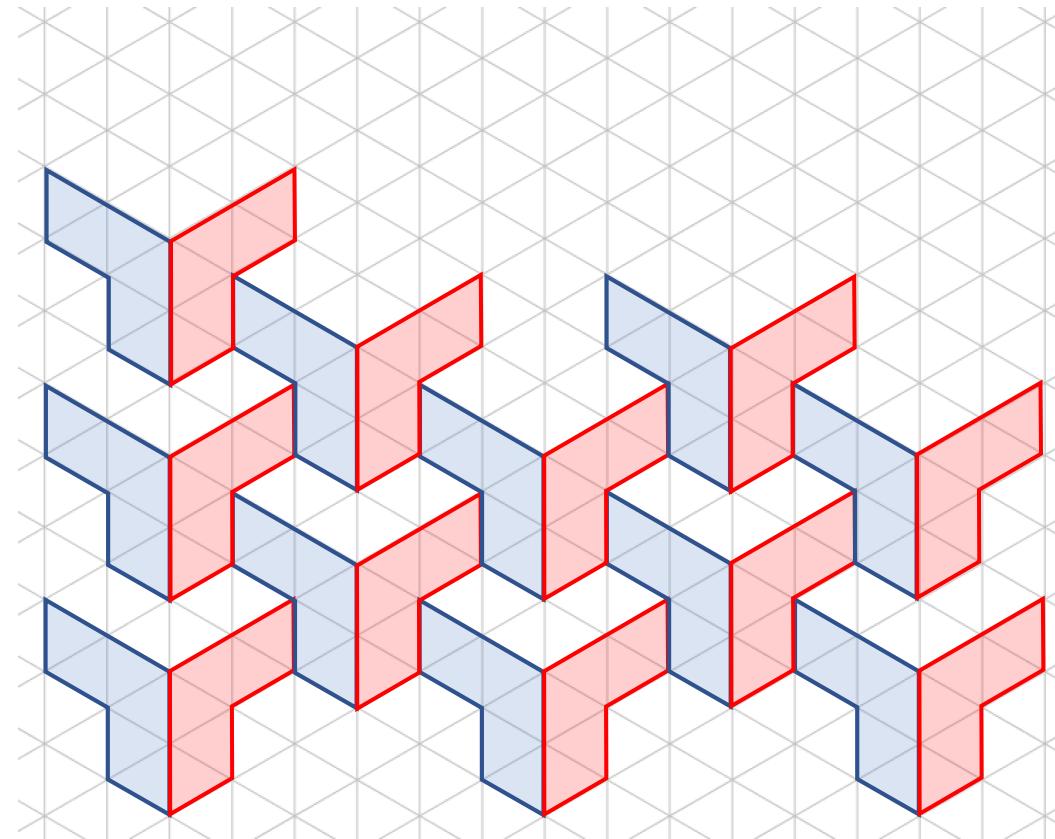
T-shapes



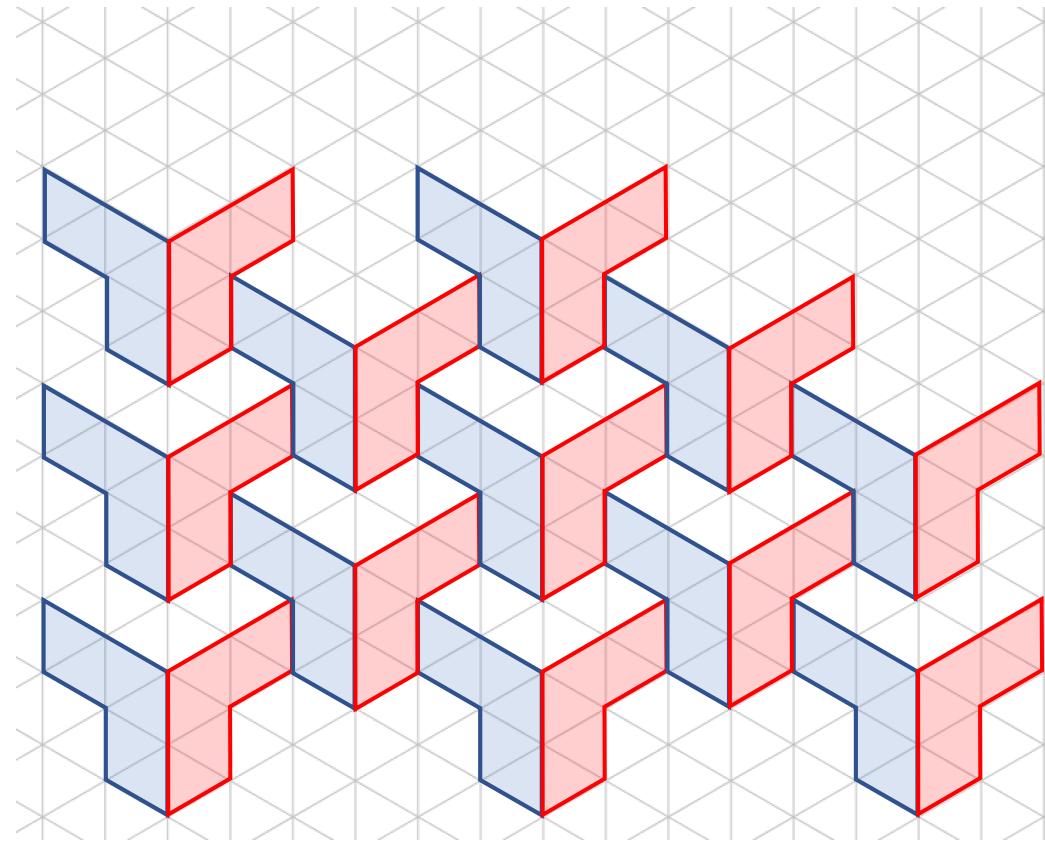
T-shapes



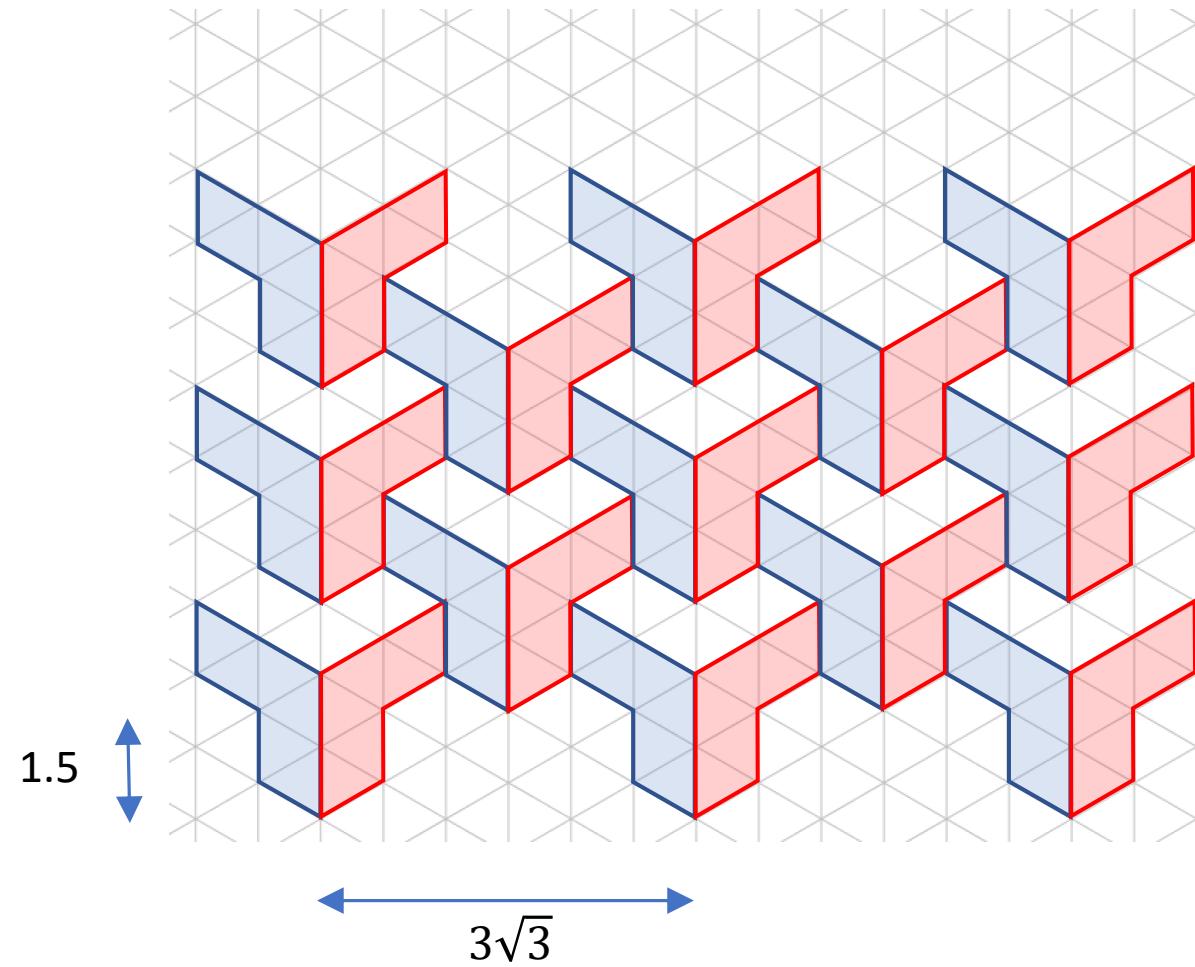
T-shapes



T-shapes



T-shapes



Tessellations

Tessellations

Cairo

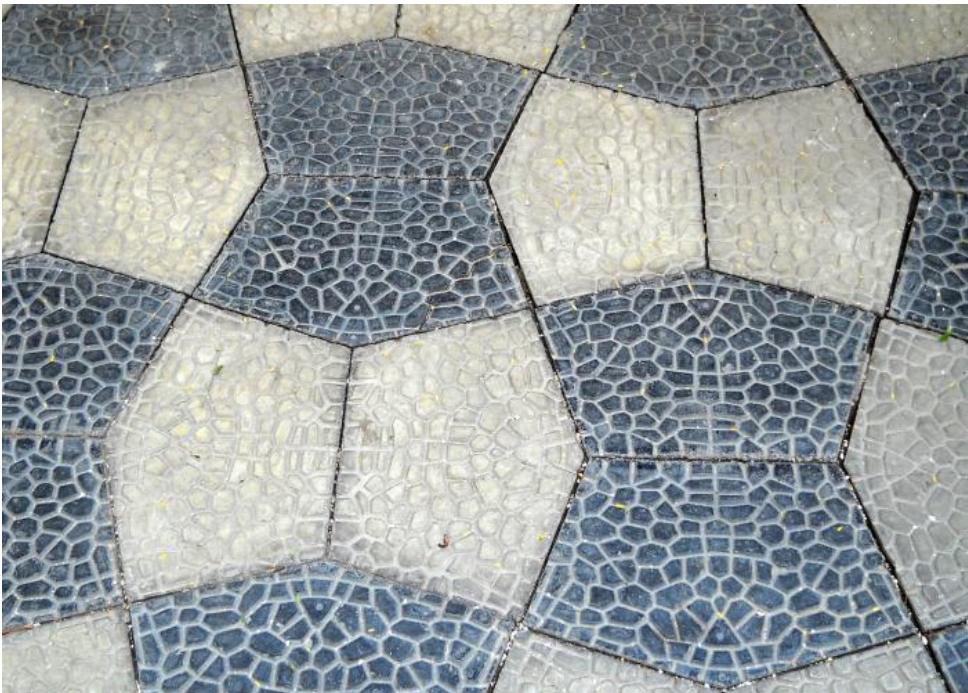
Fish

Circular Fish

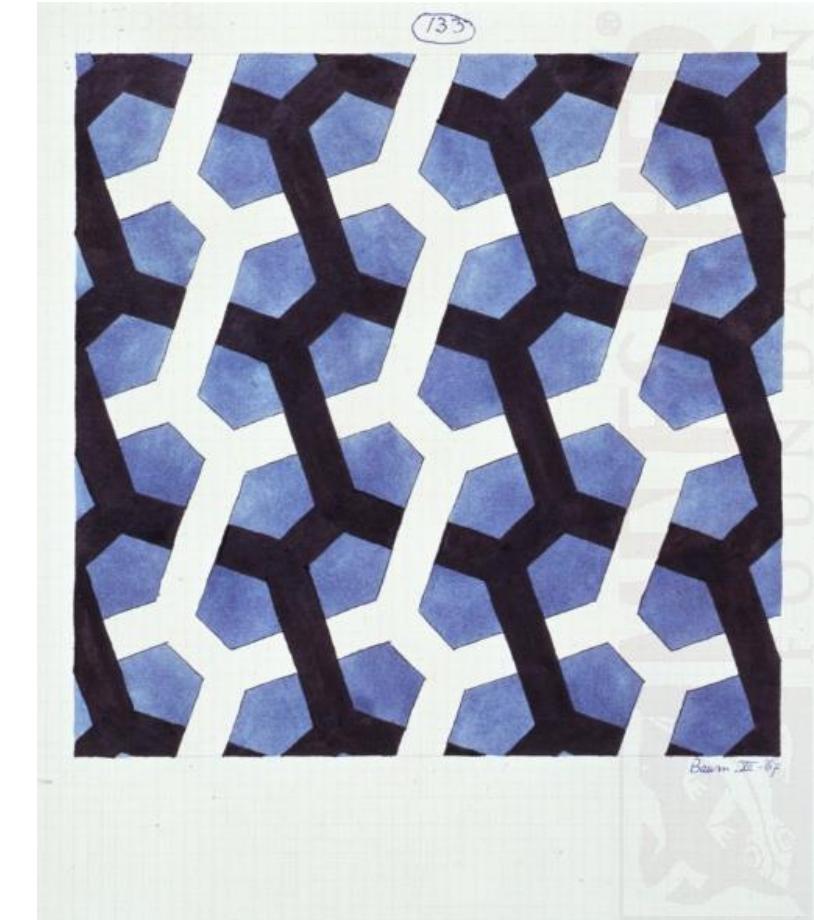
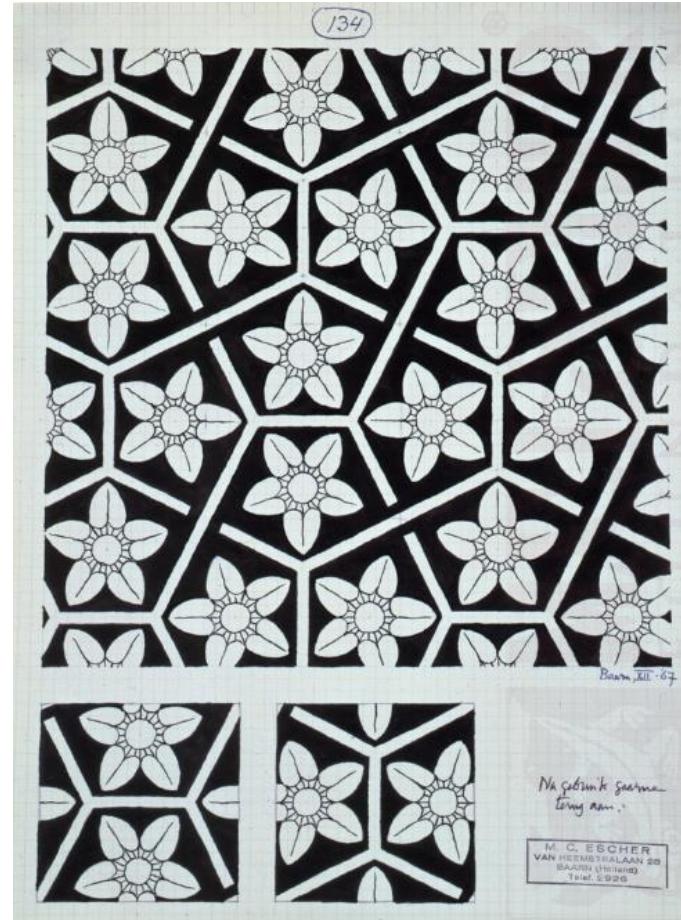
Penrose

End

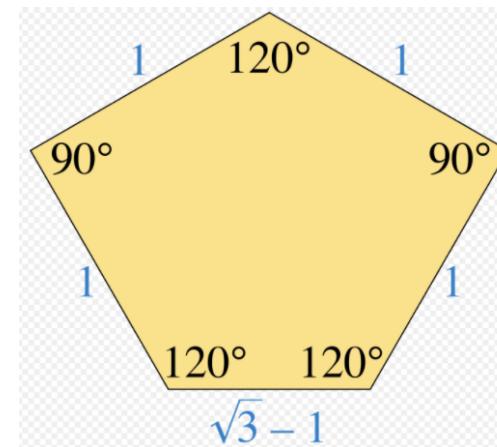
Cairo



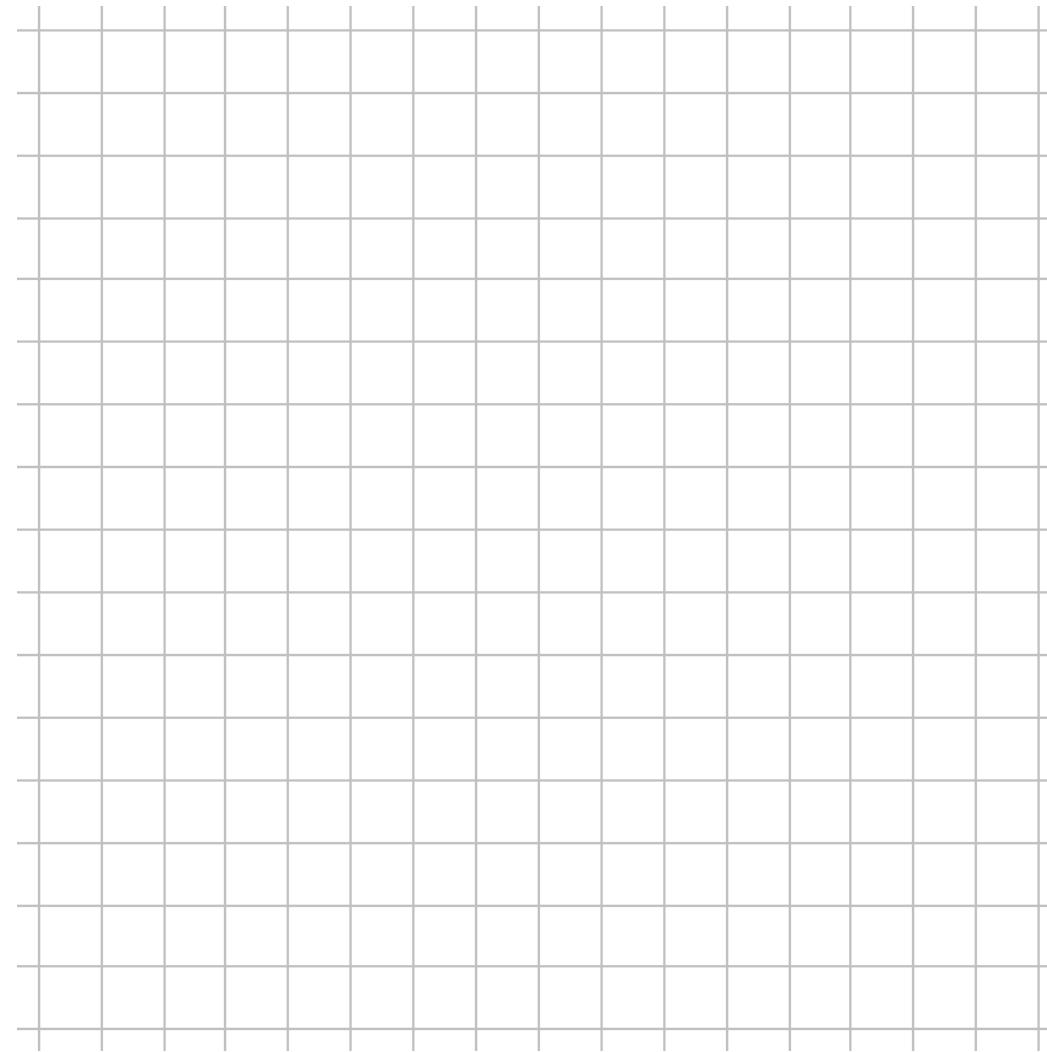
Cairo



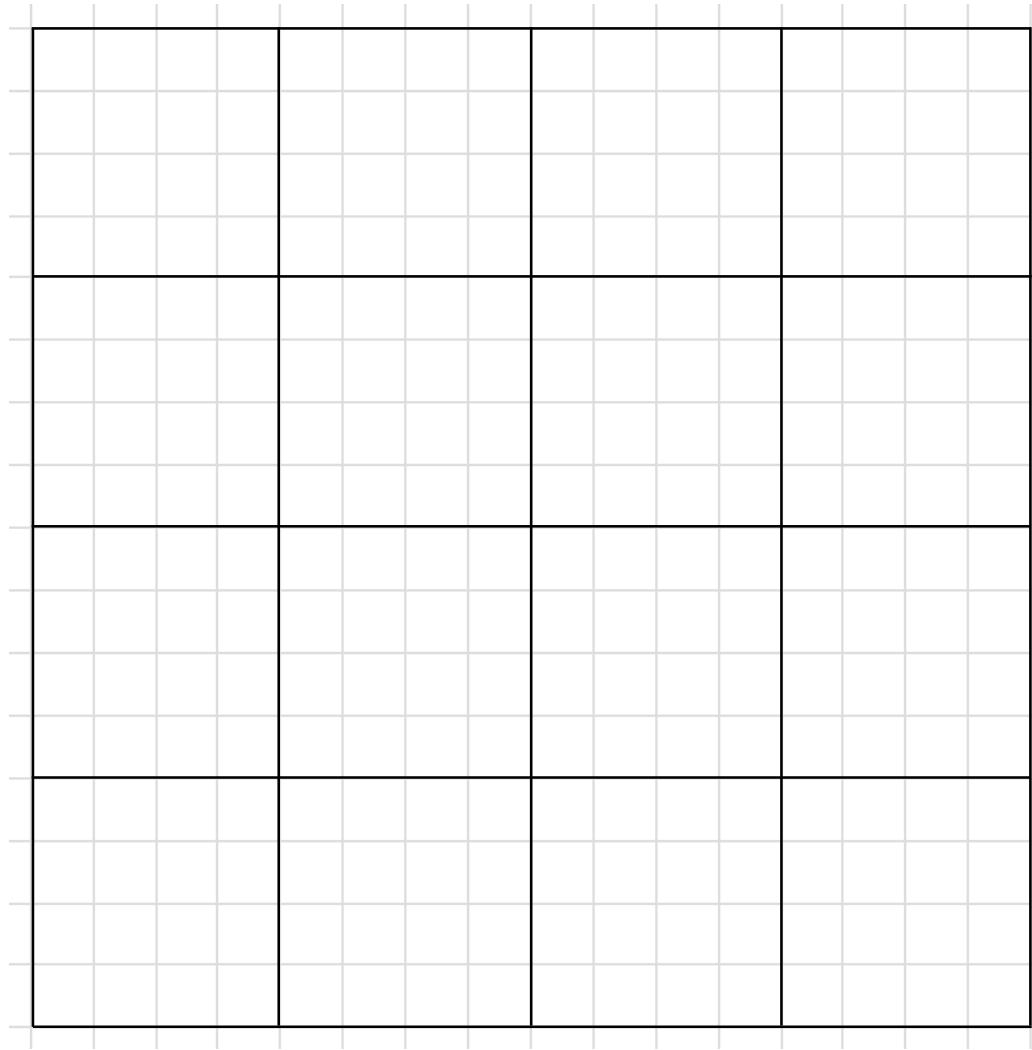
Cairo



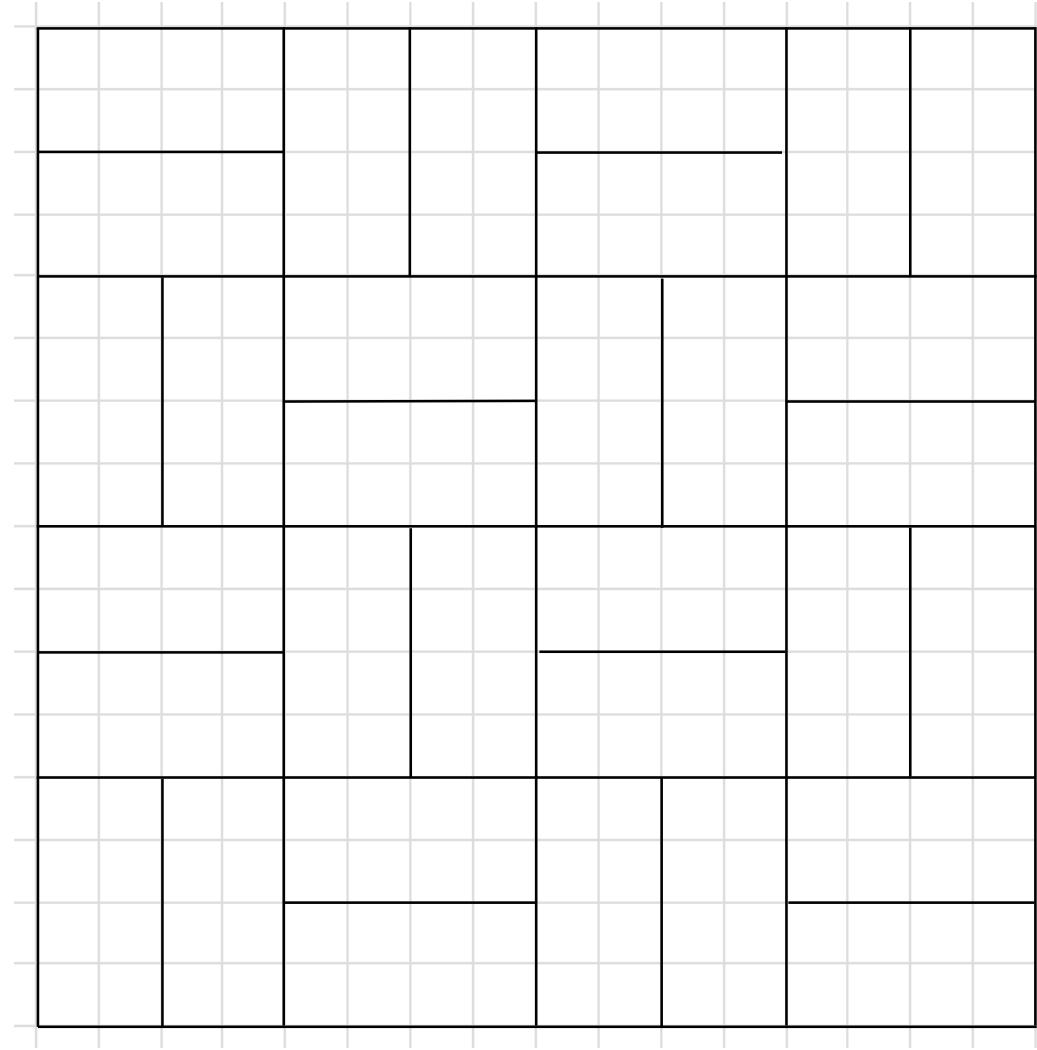
Cairo



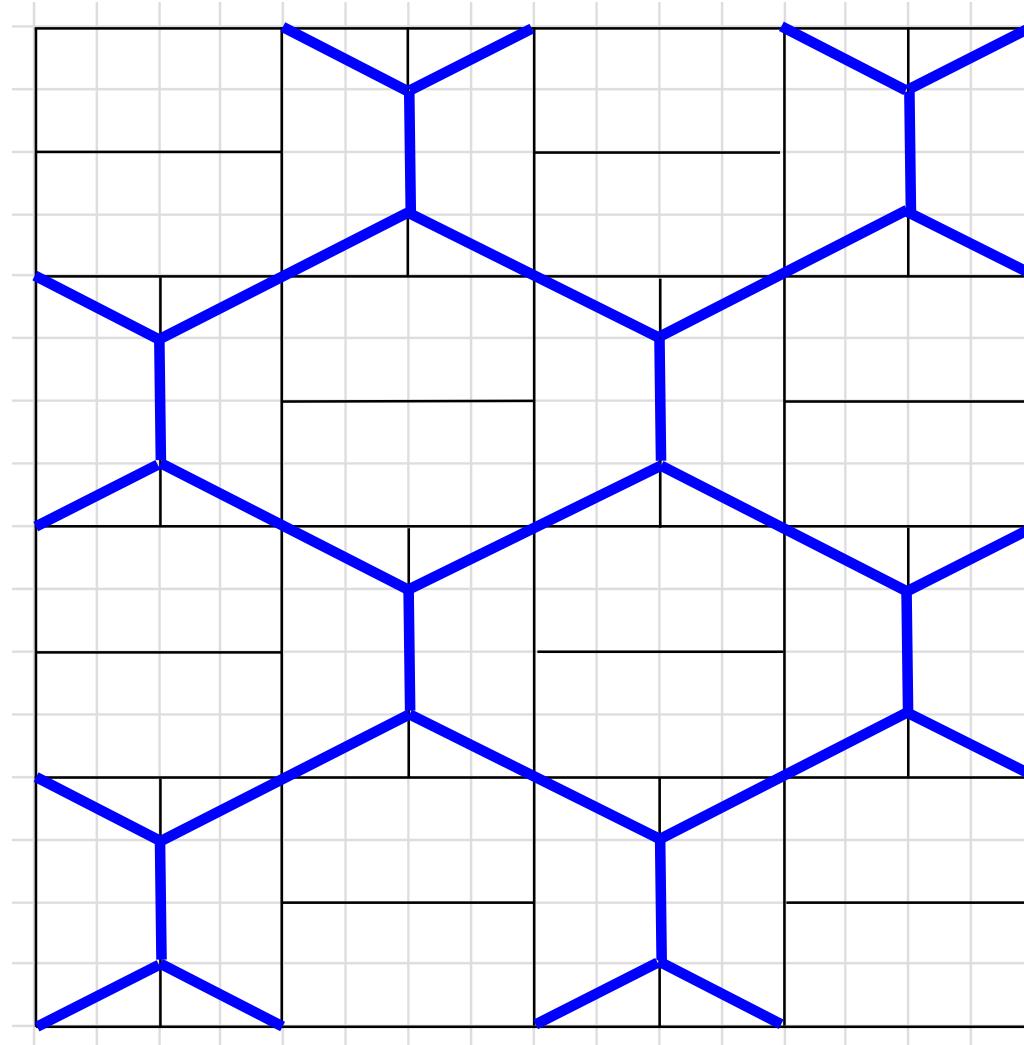
Cairo



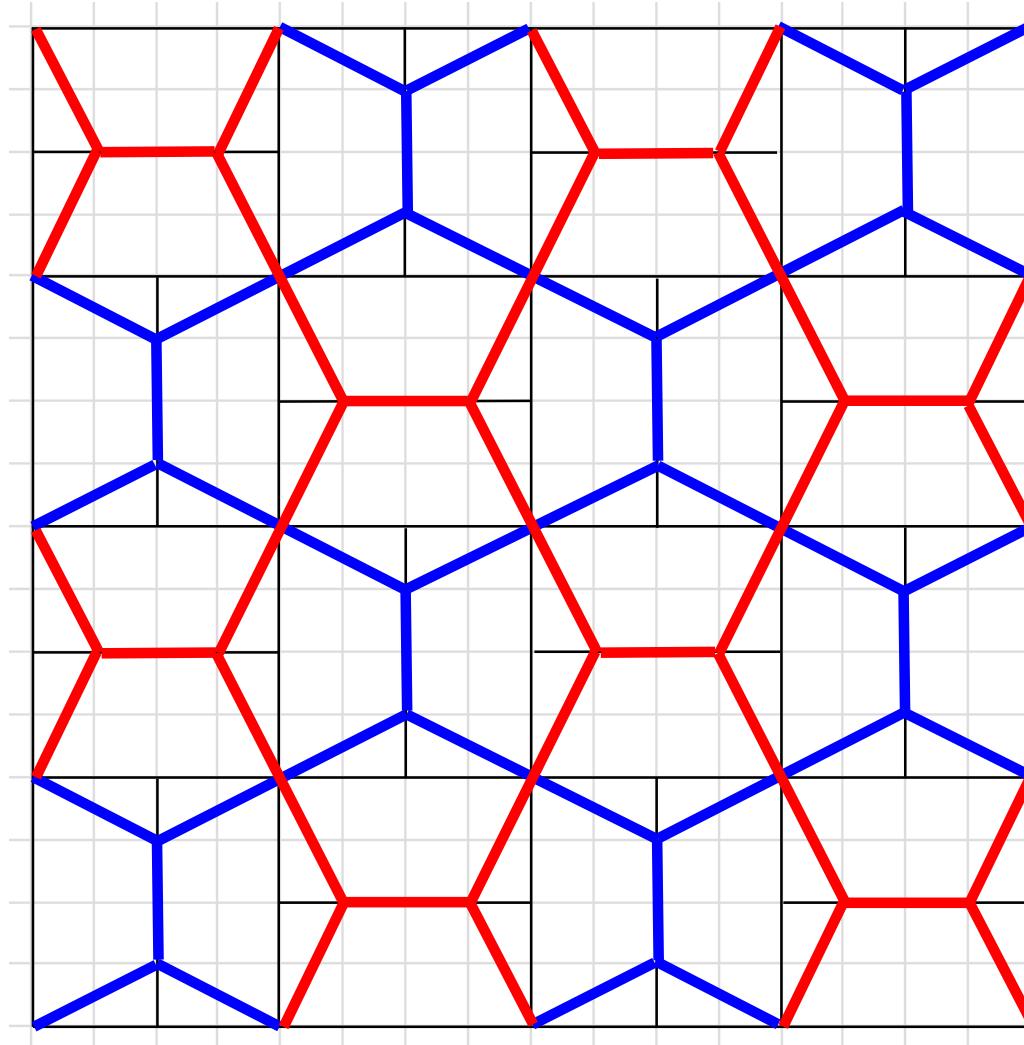
Cairo



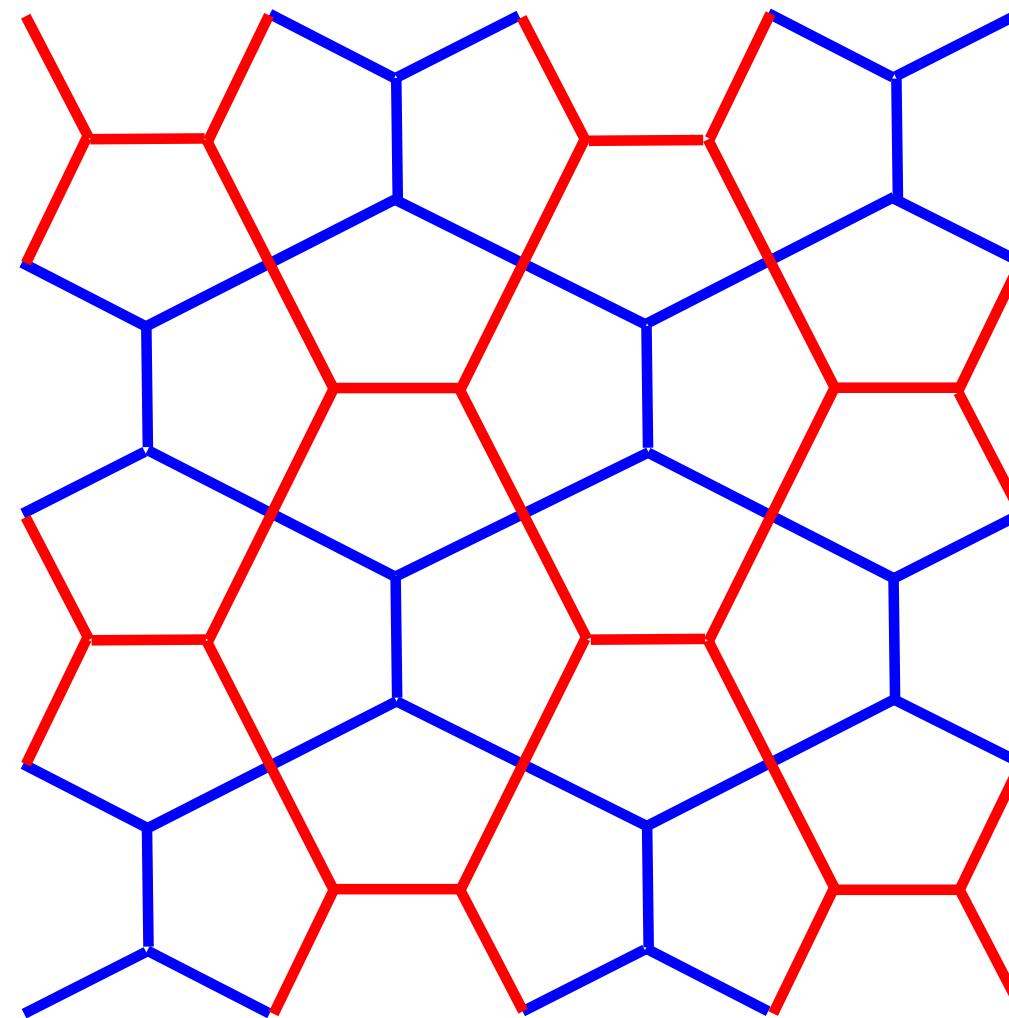
Cairo



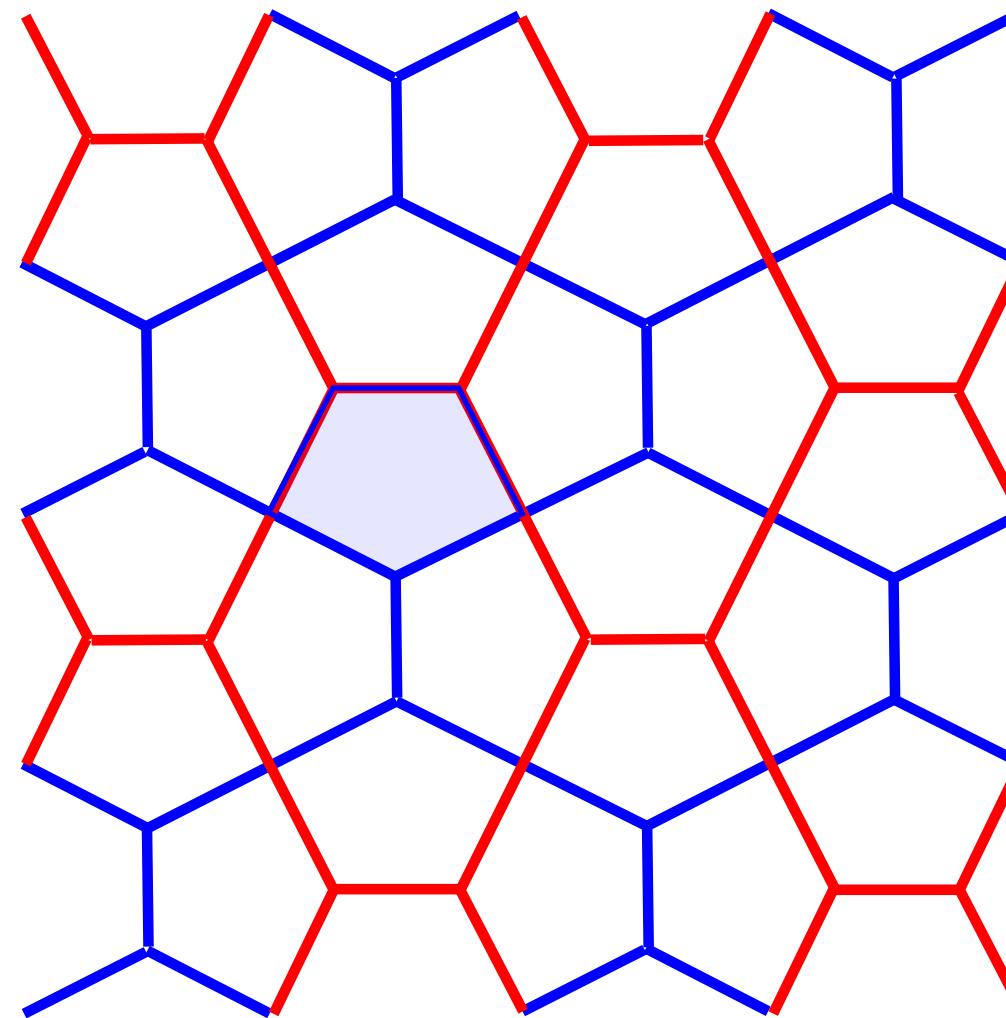
Cairo



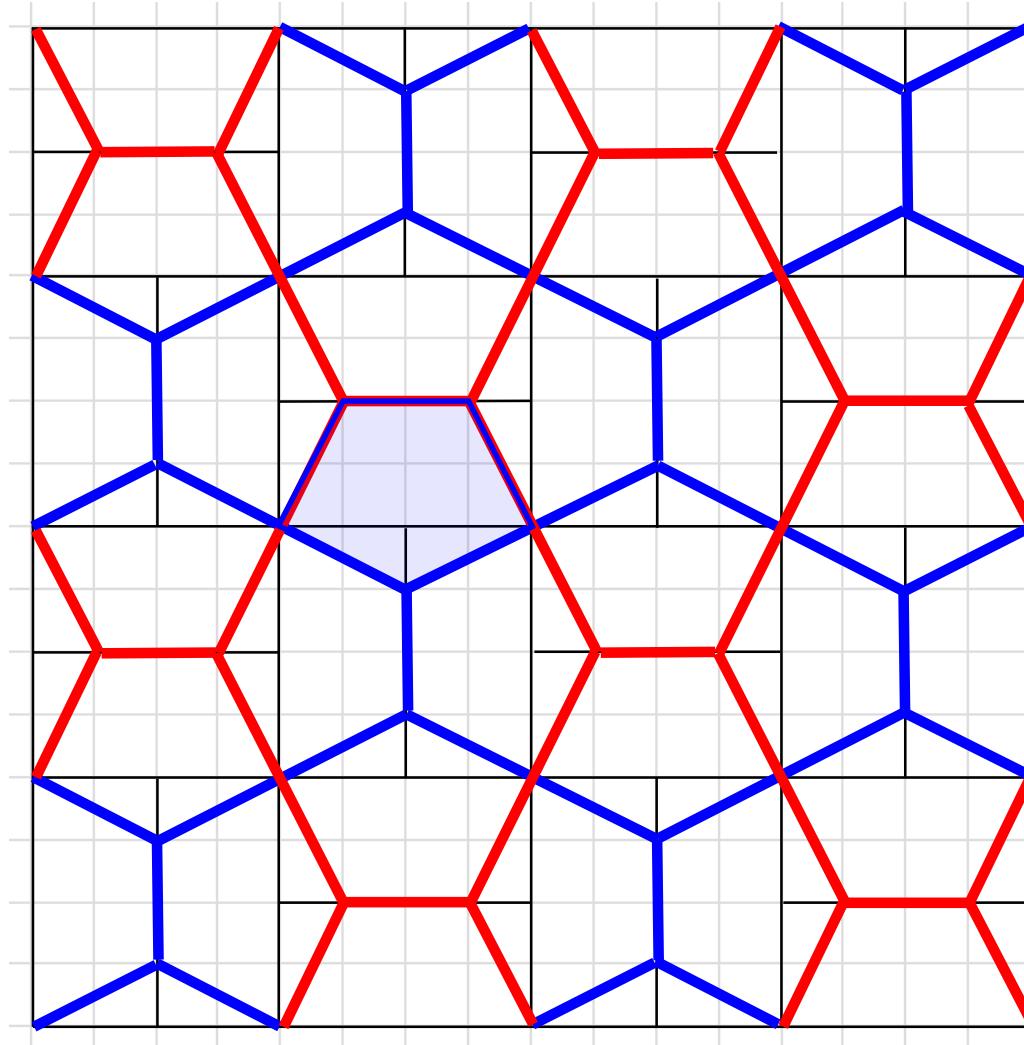
Cairo



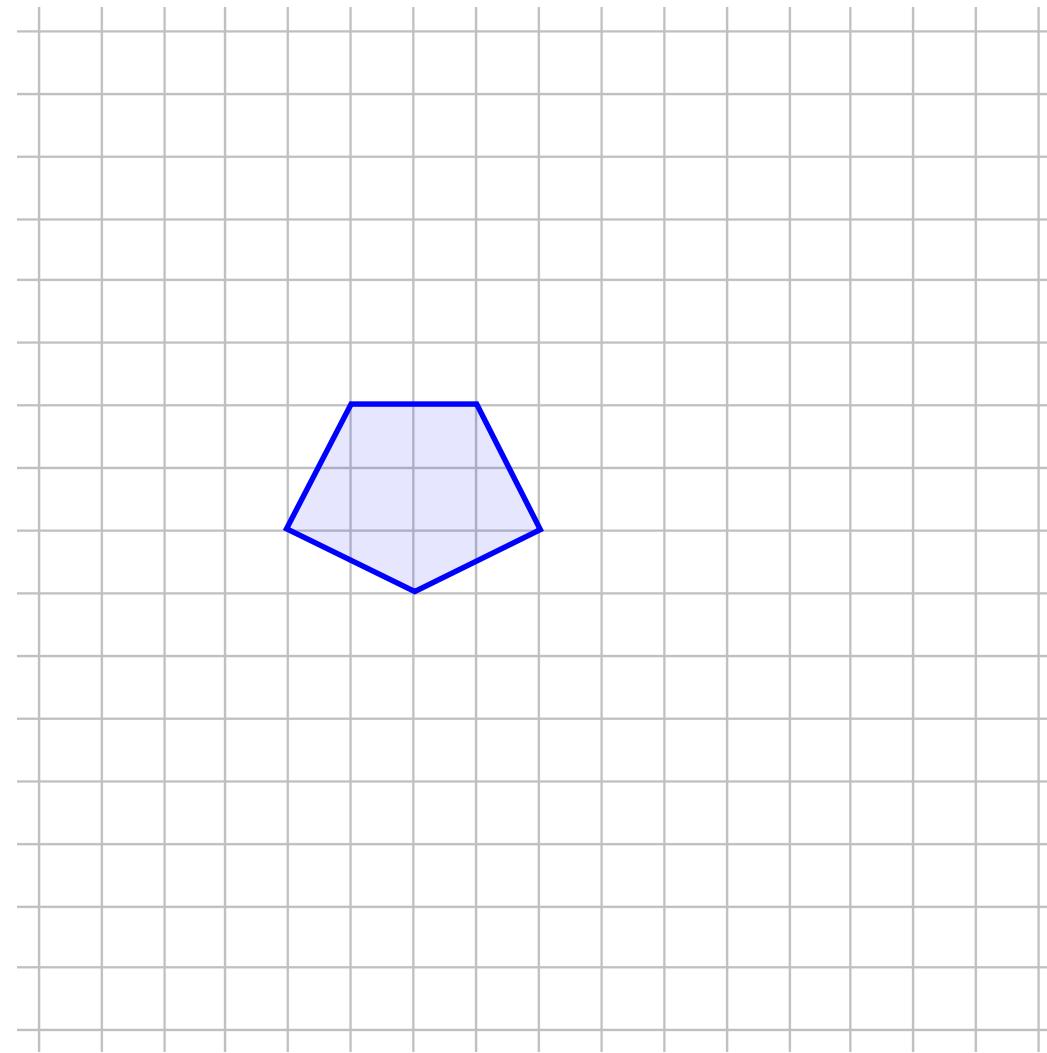
Cairo



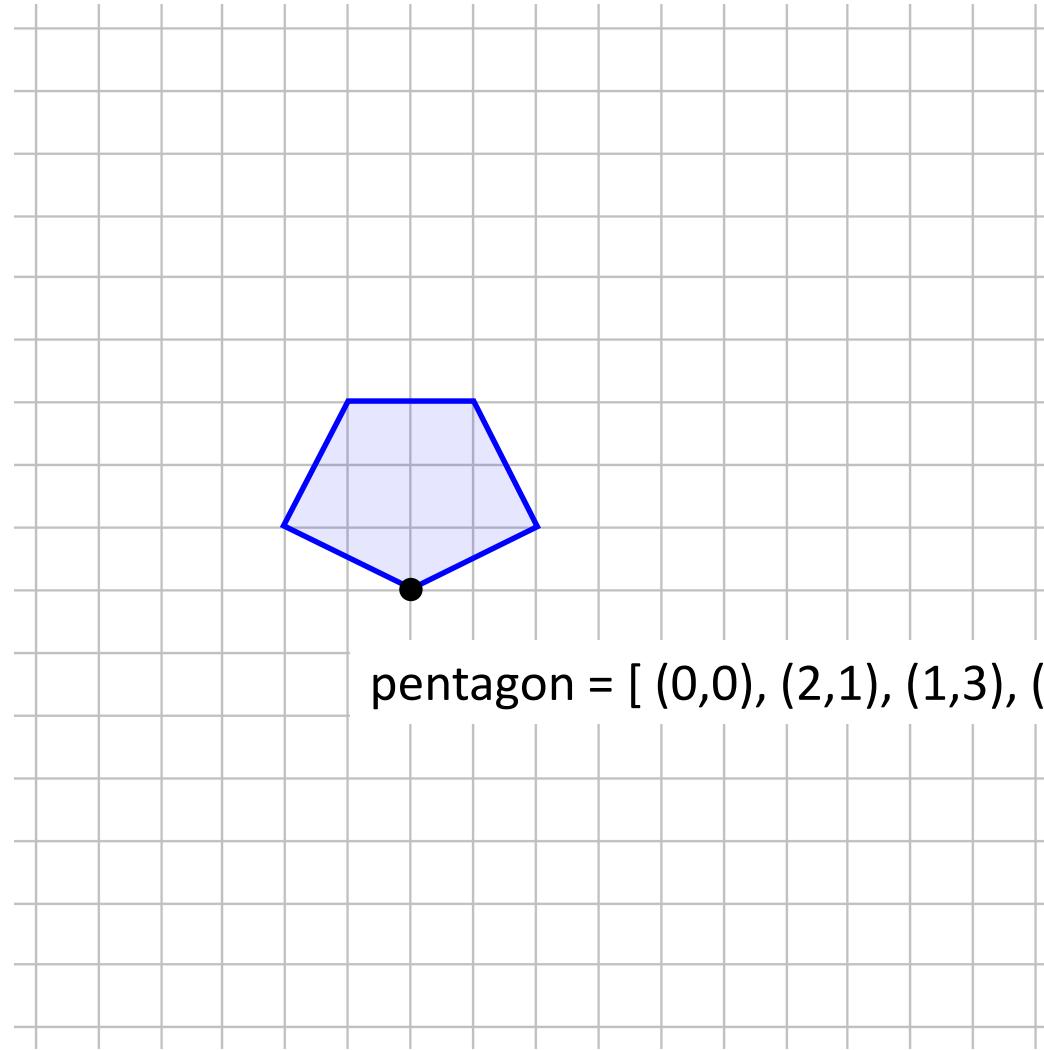
Cairo



Cairo

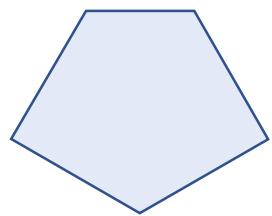


Cairo

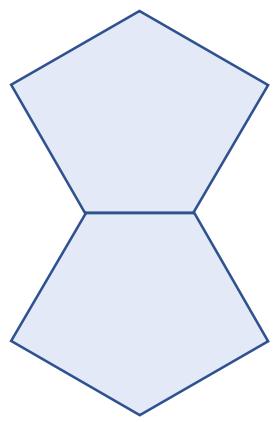


pentagon = [(0,0), (2,1), (1,3), (-1,3), (-2,1), (0,0)]

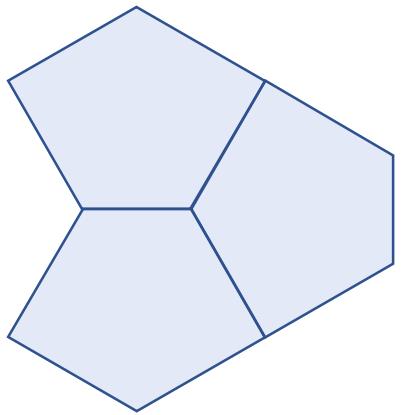
Cairo



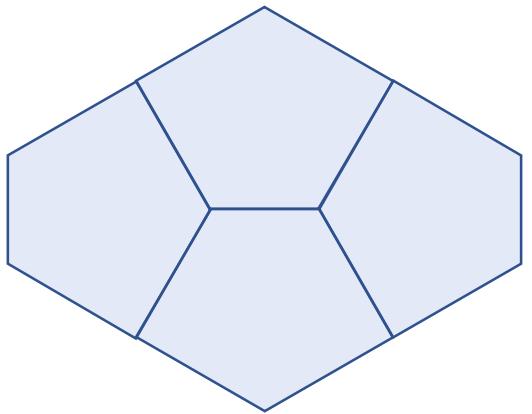
Cairo



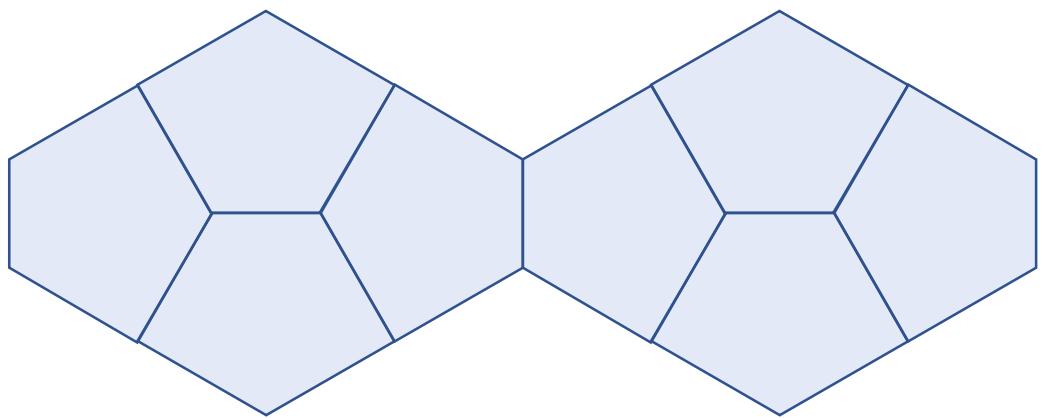
Cairo



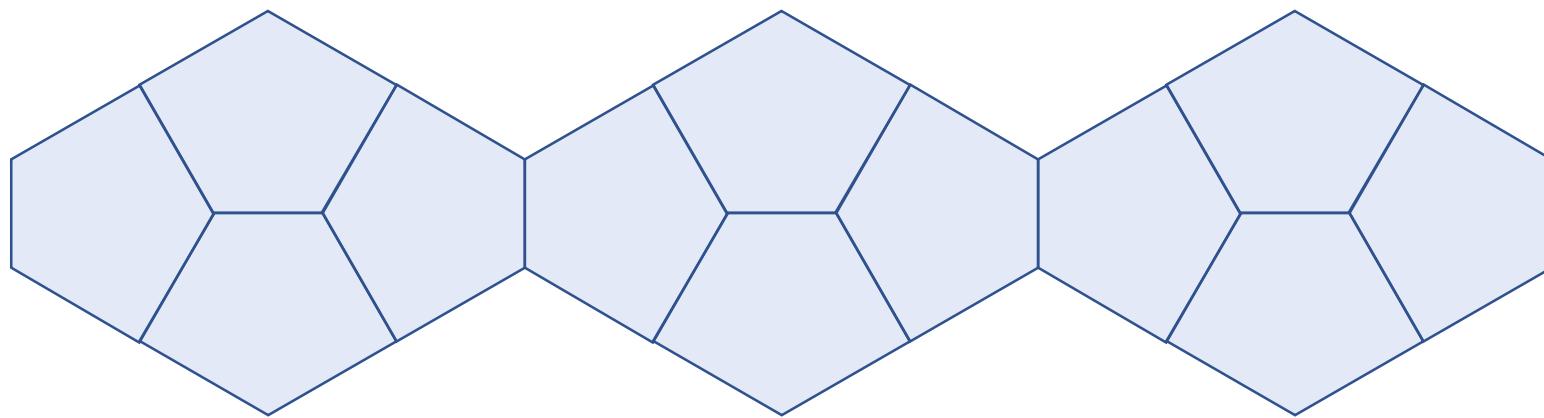
Cairo



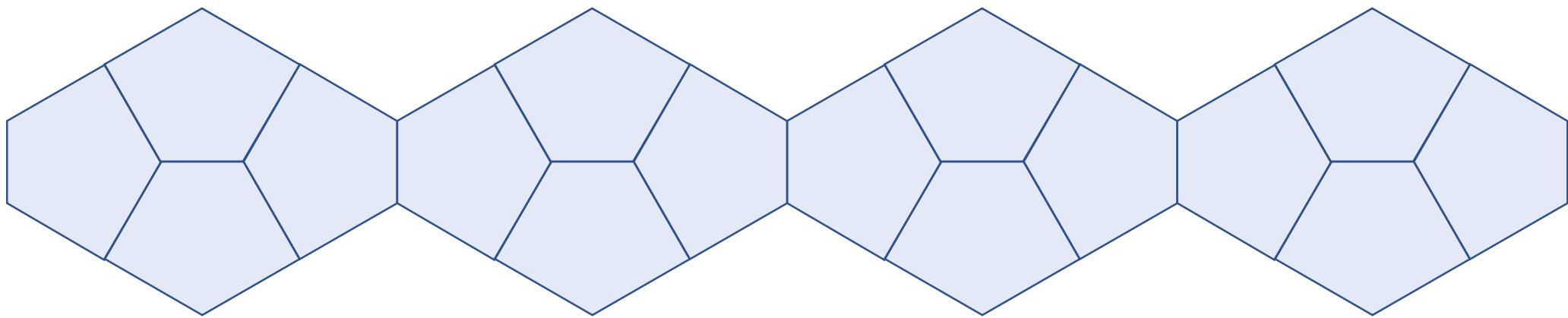
Cairo



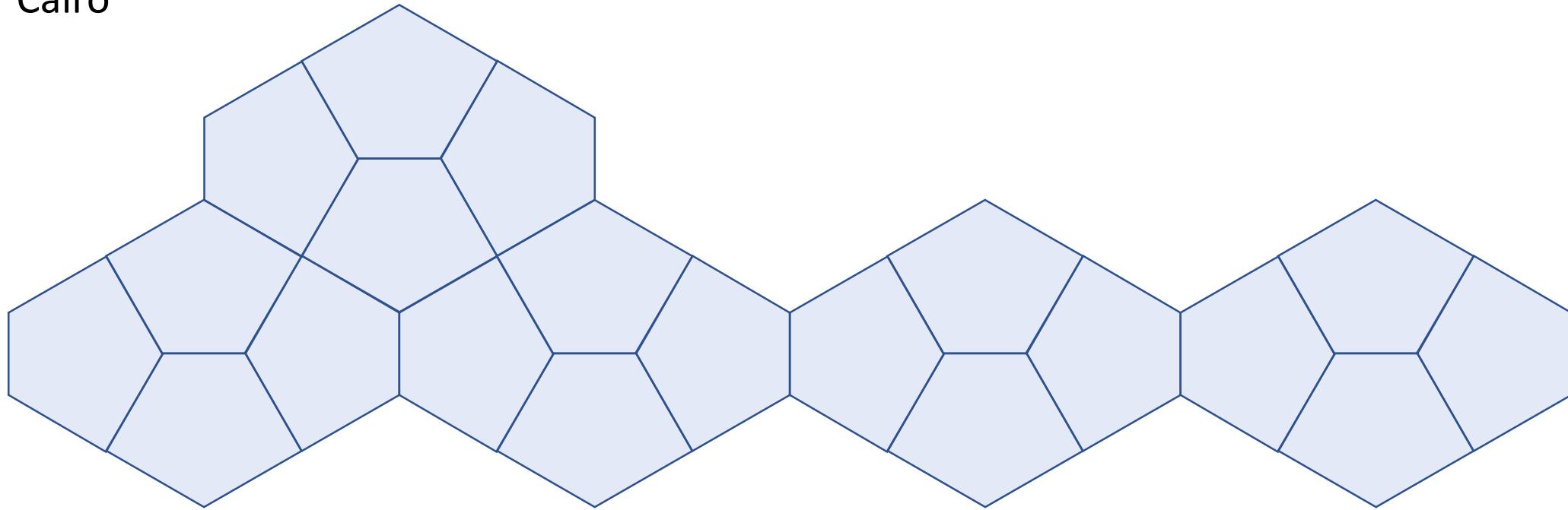
Cairo



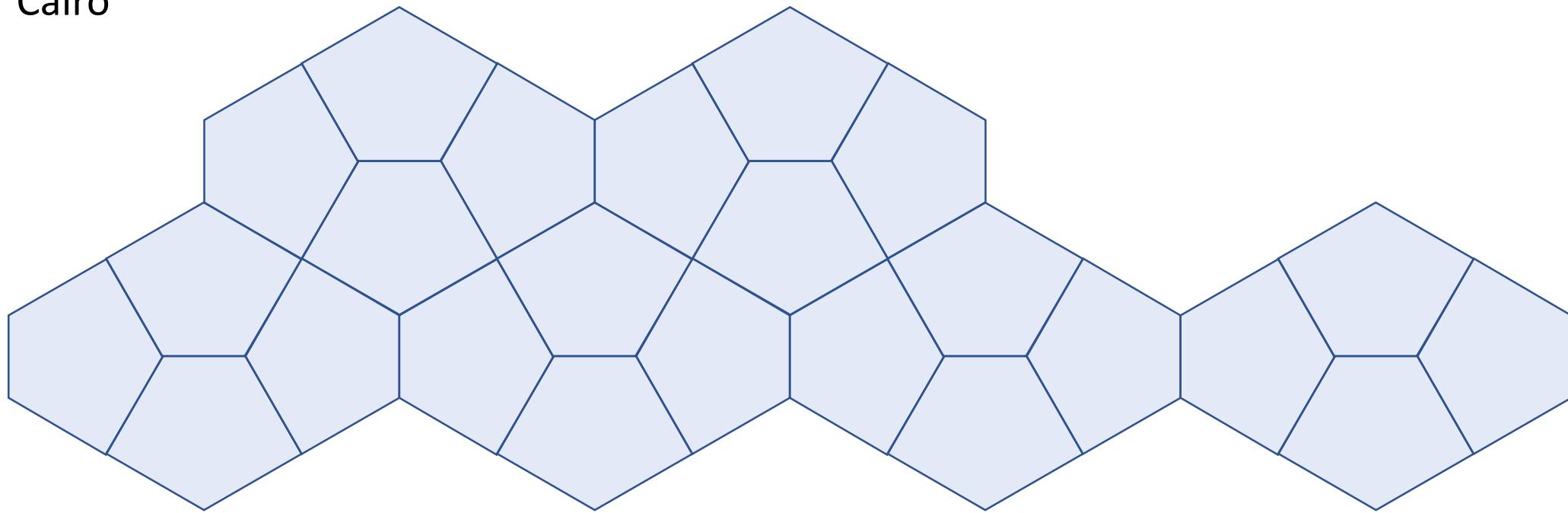
Cairo



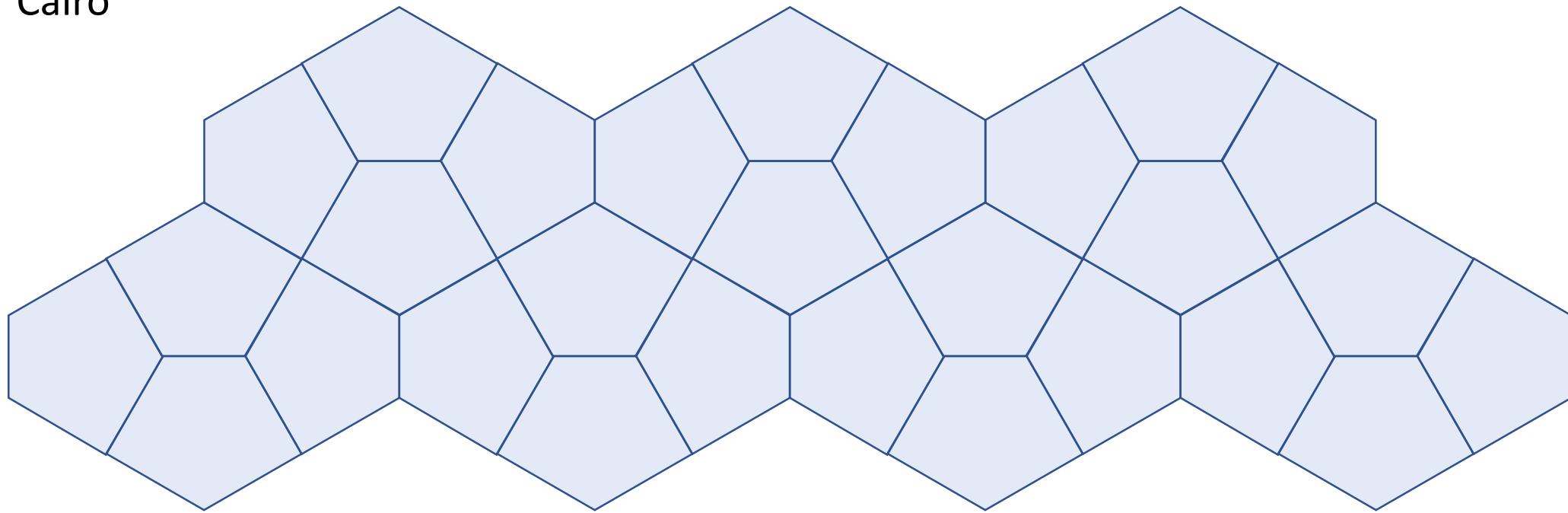
Cairo



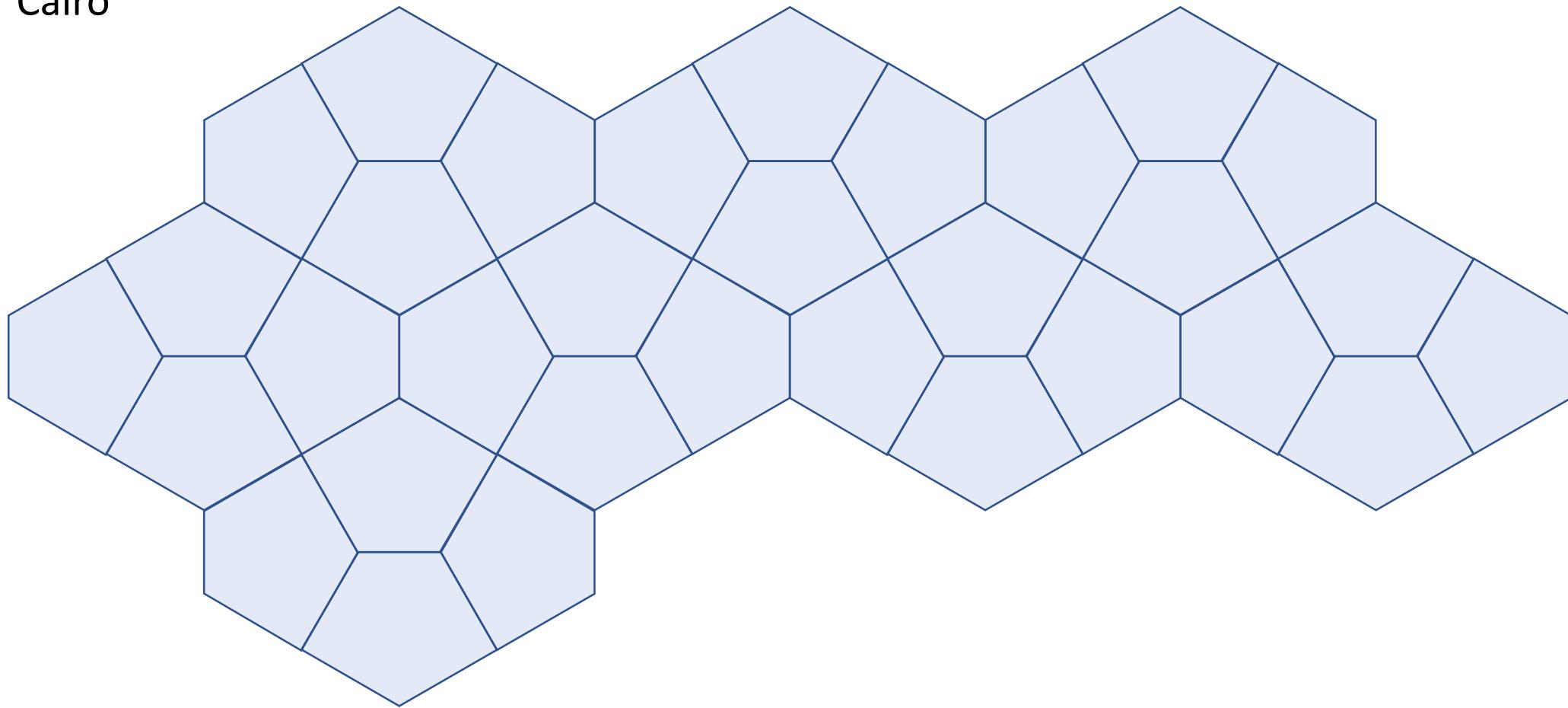
Cairo



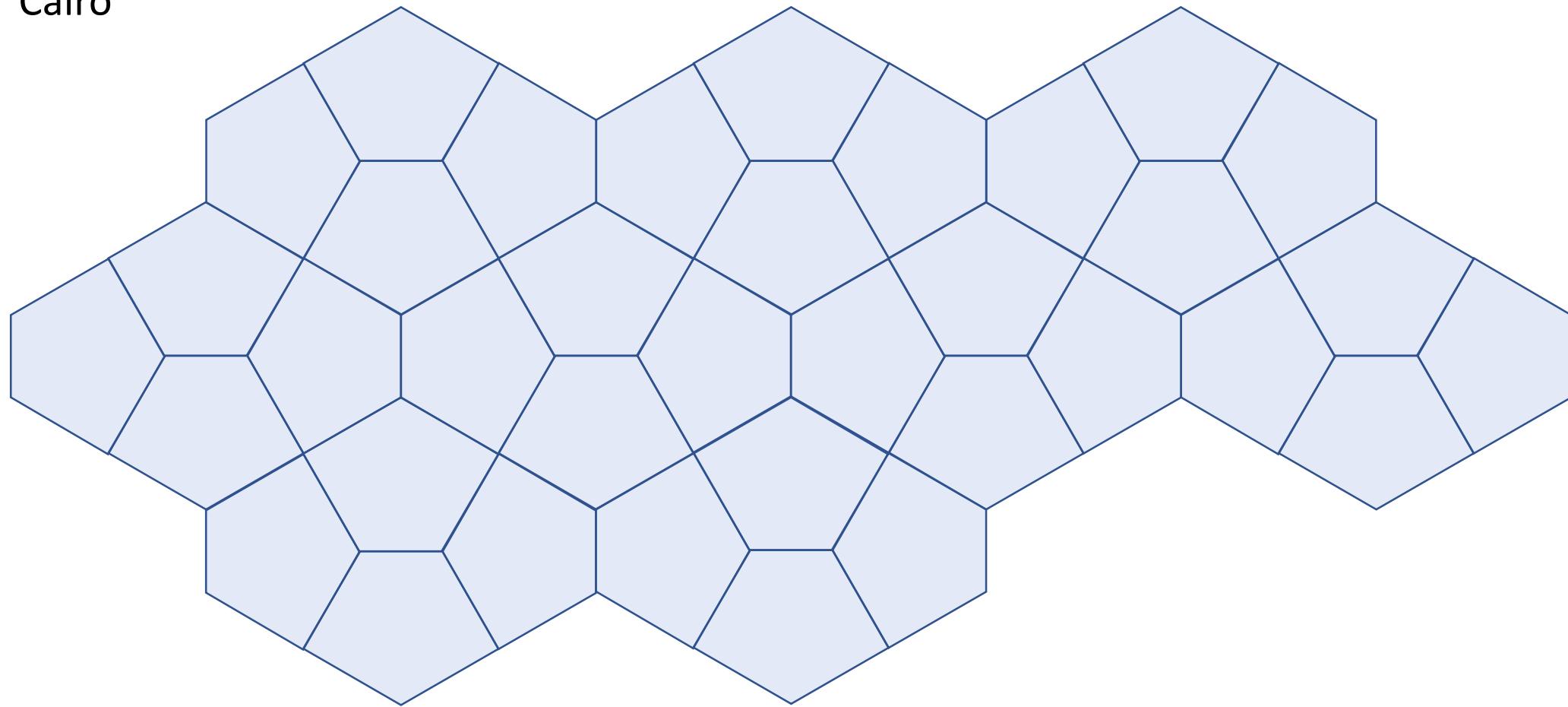
Cairo



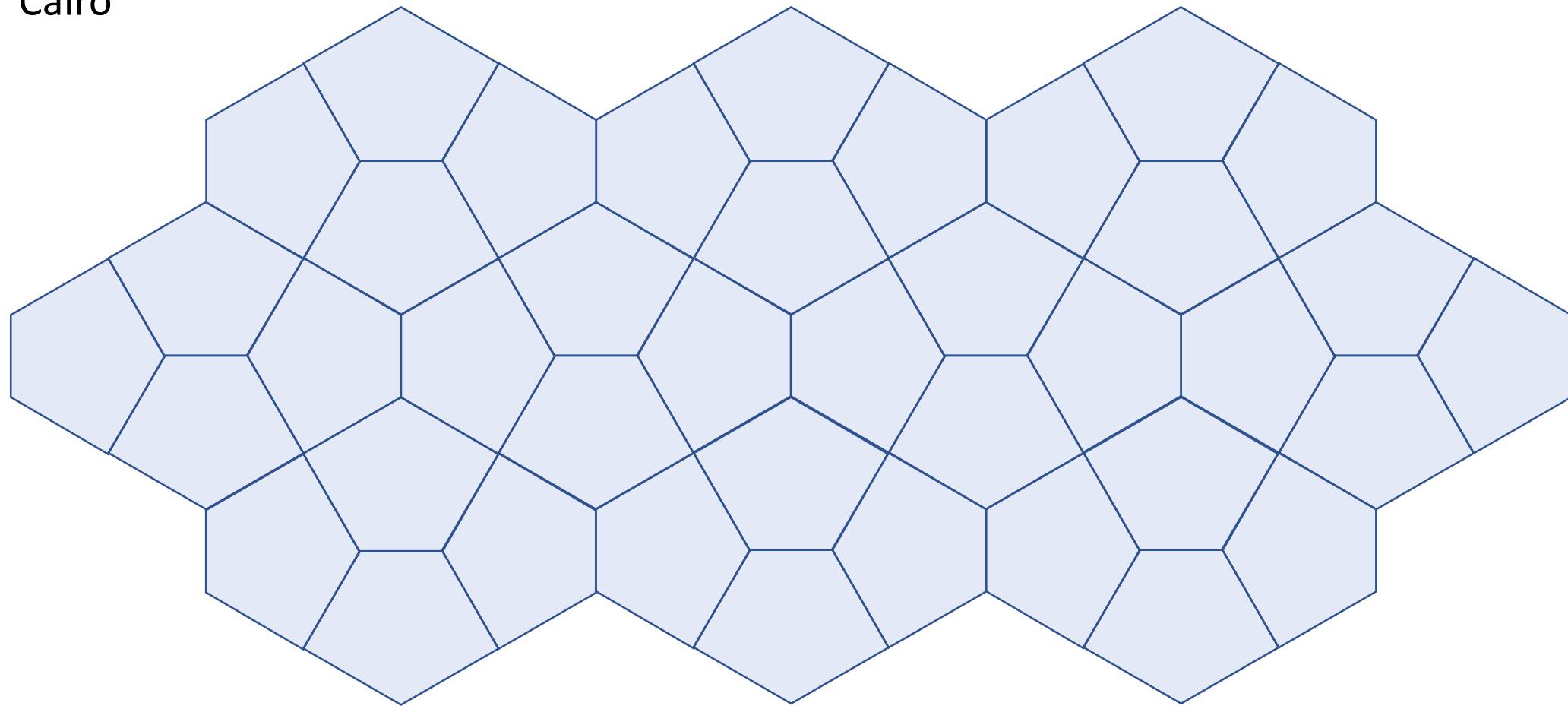
Cairo



Cairo



Cairo



Cairo

Back

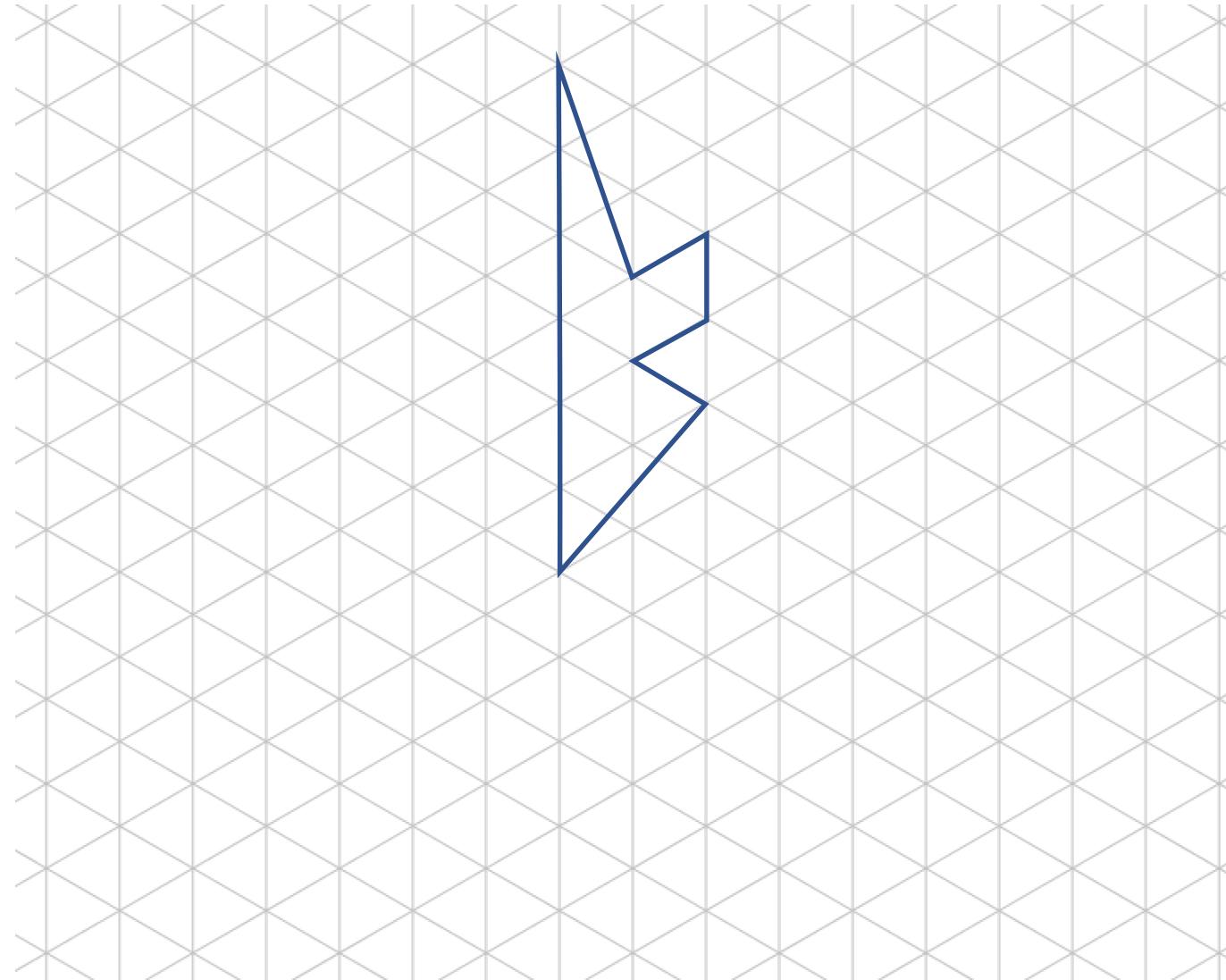
Fish



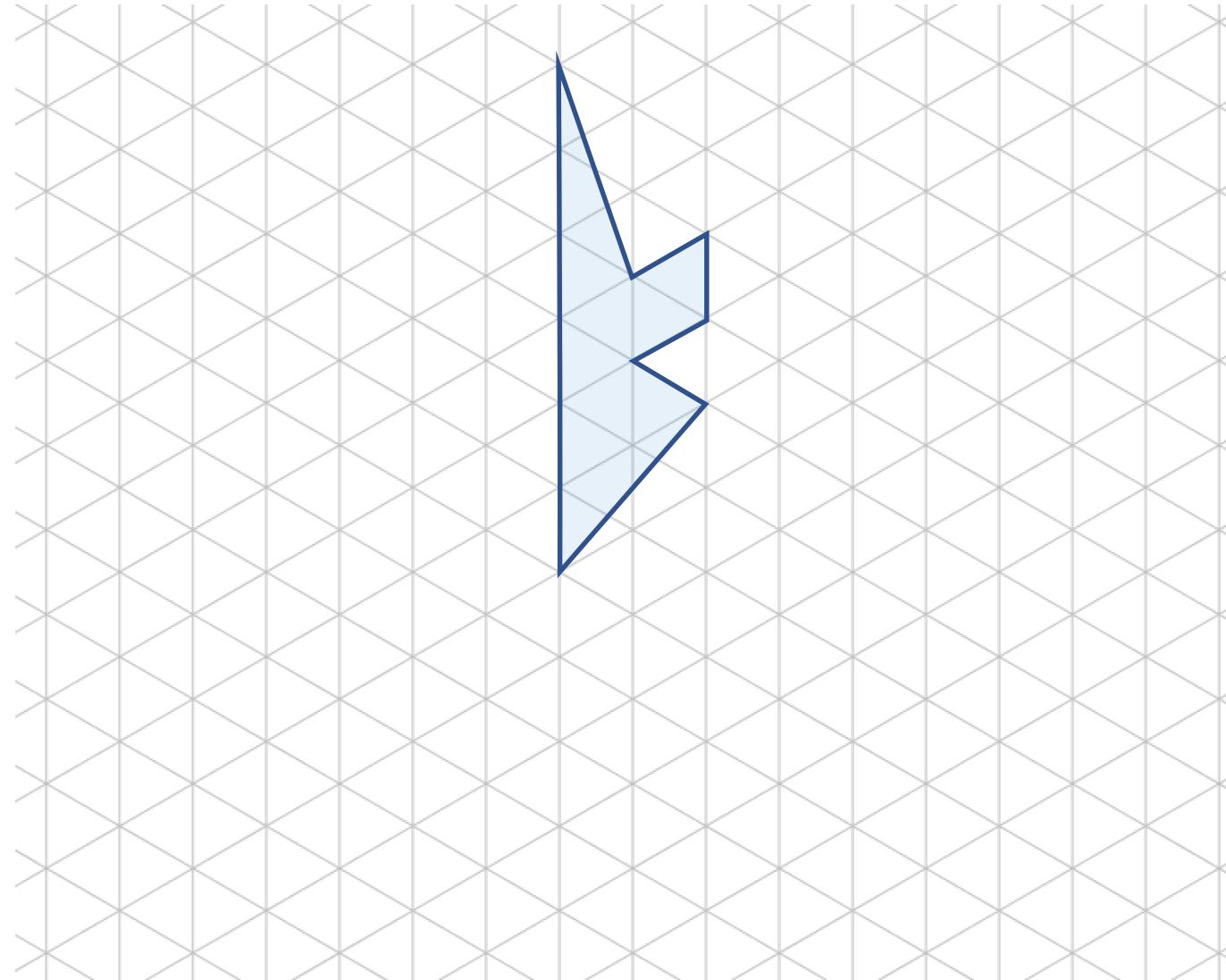
Fish



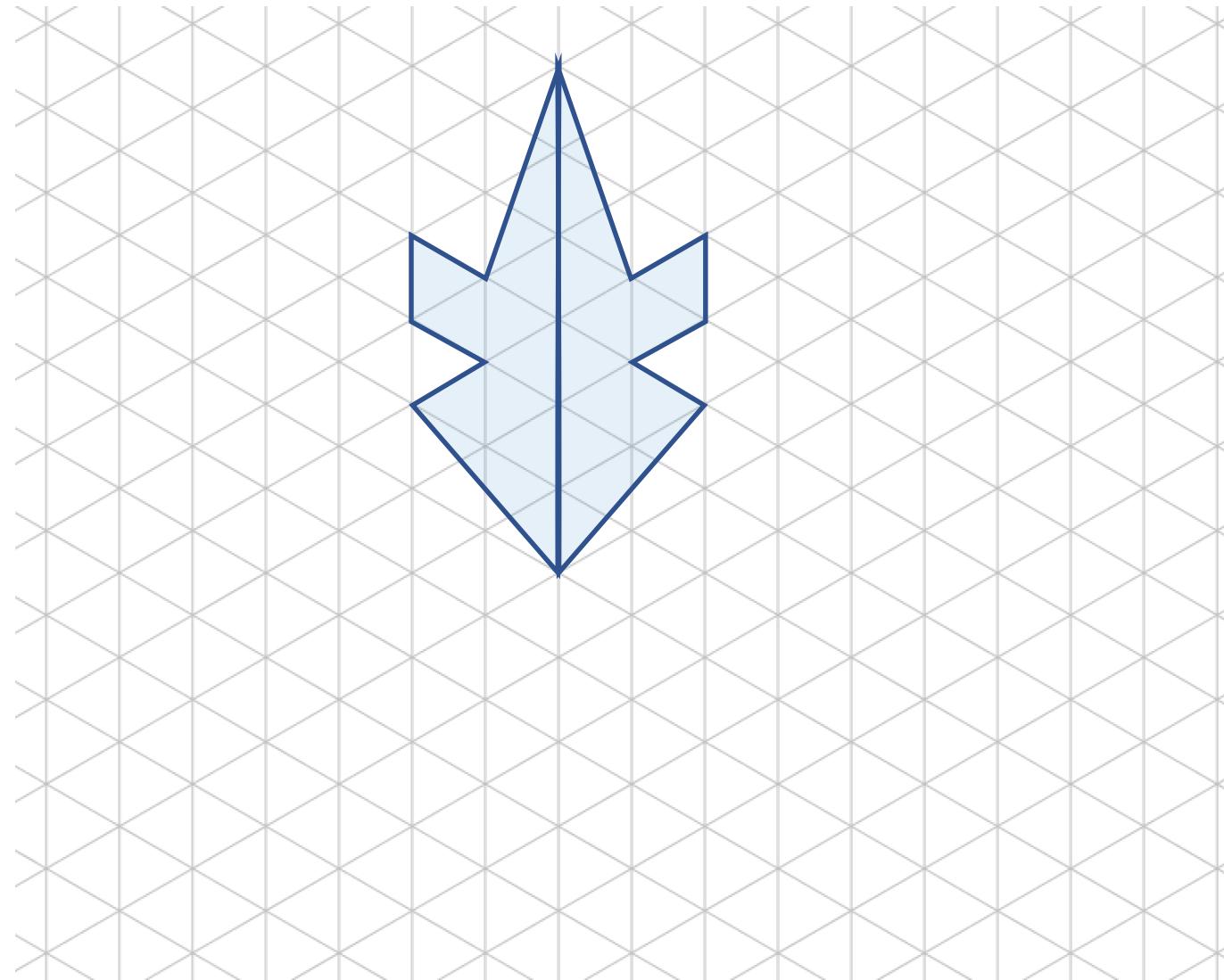
Fish



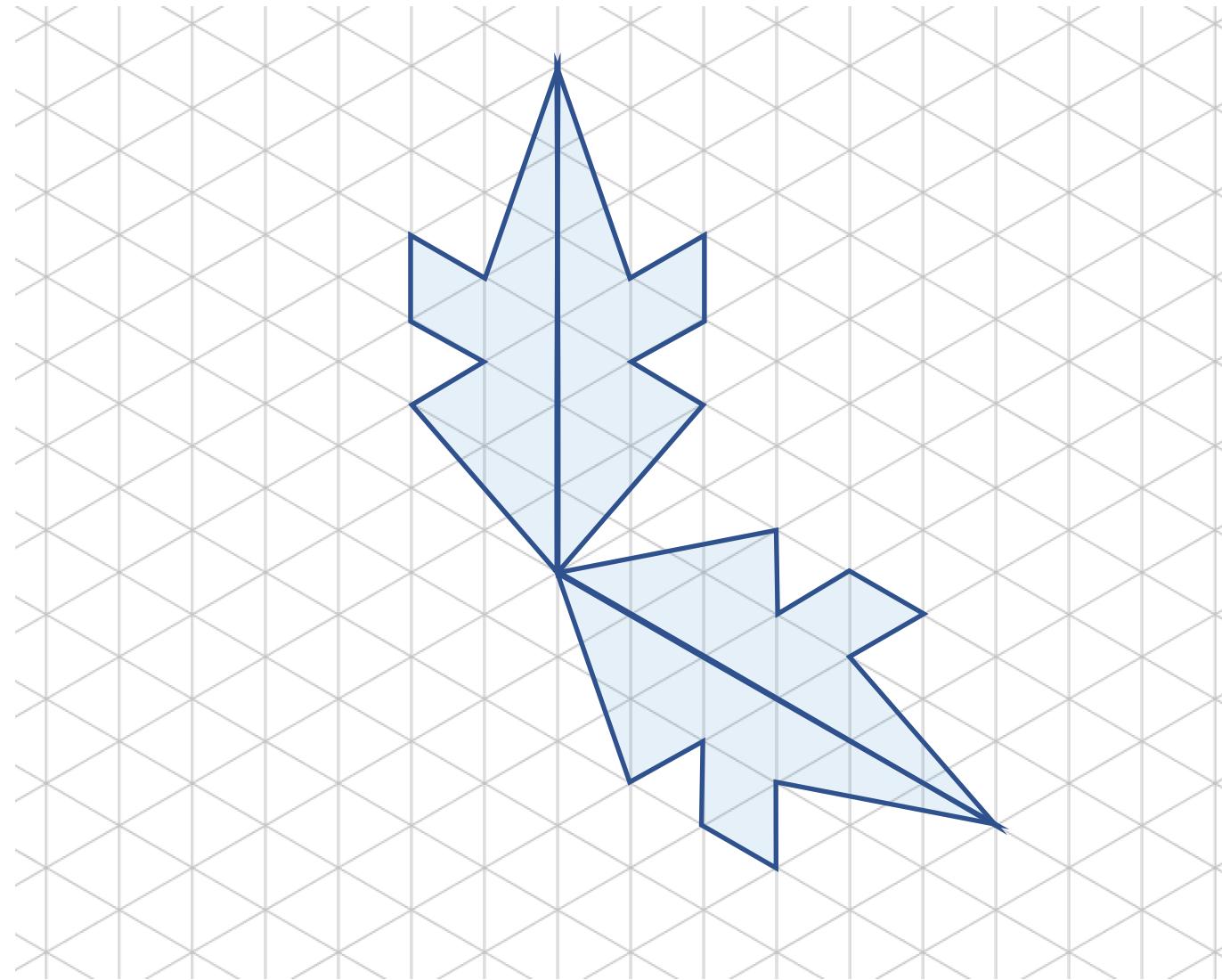
Fish



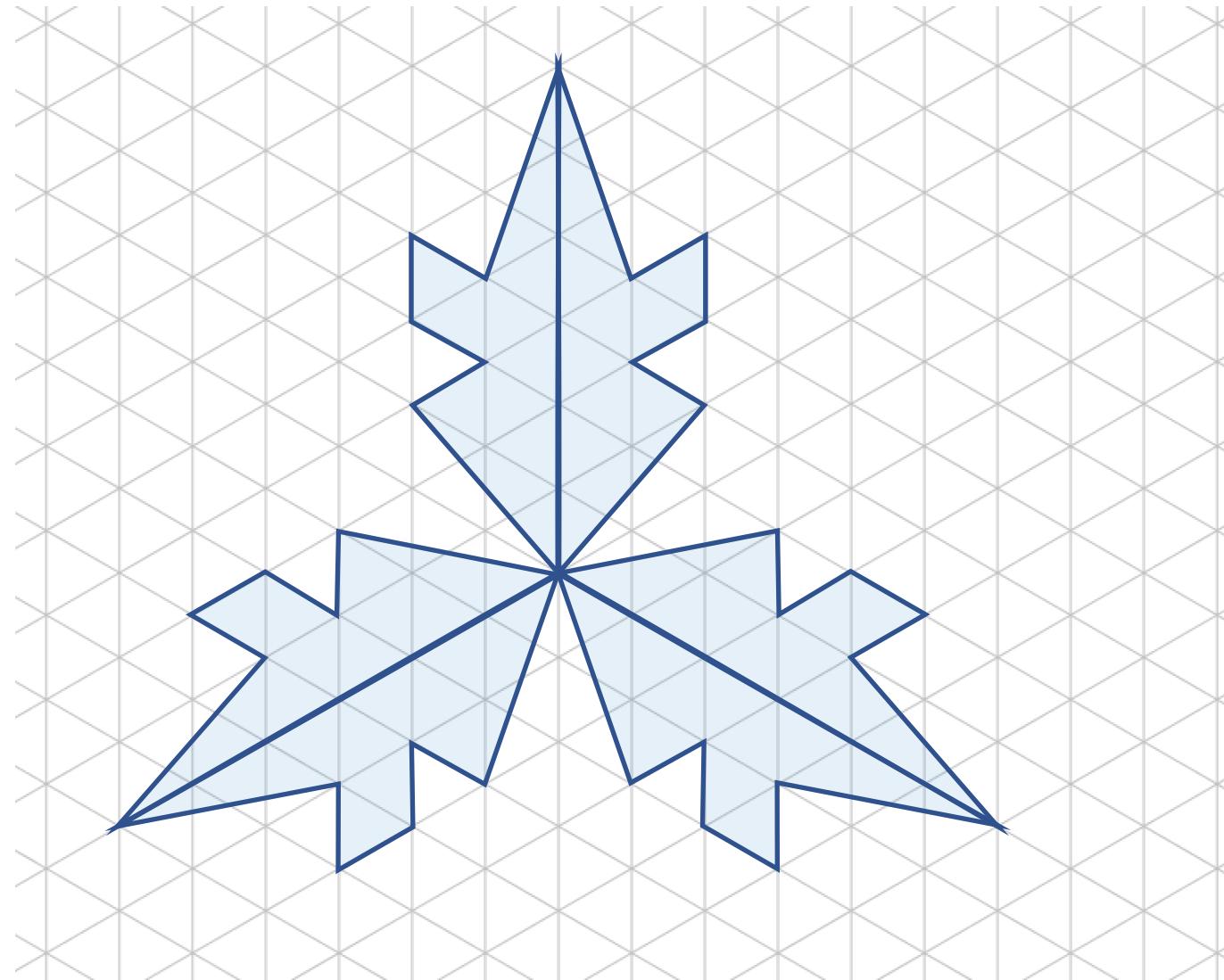
Fish



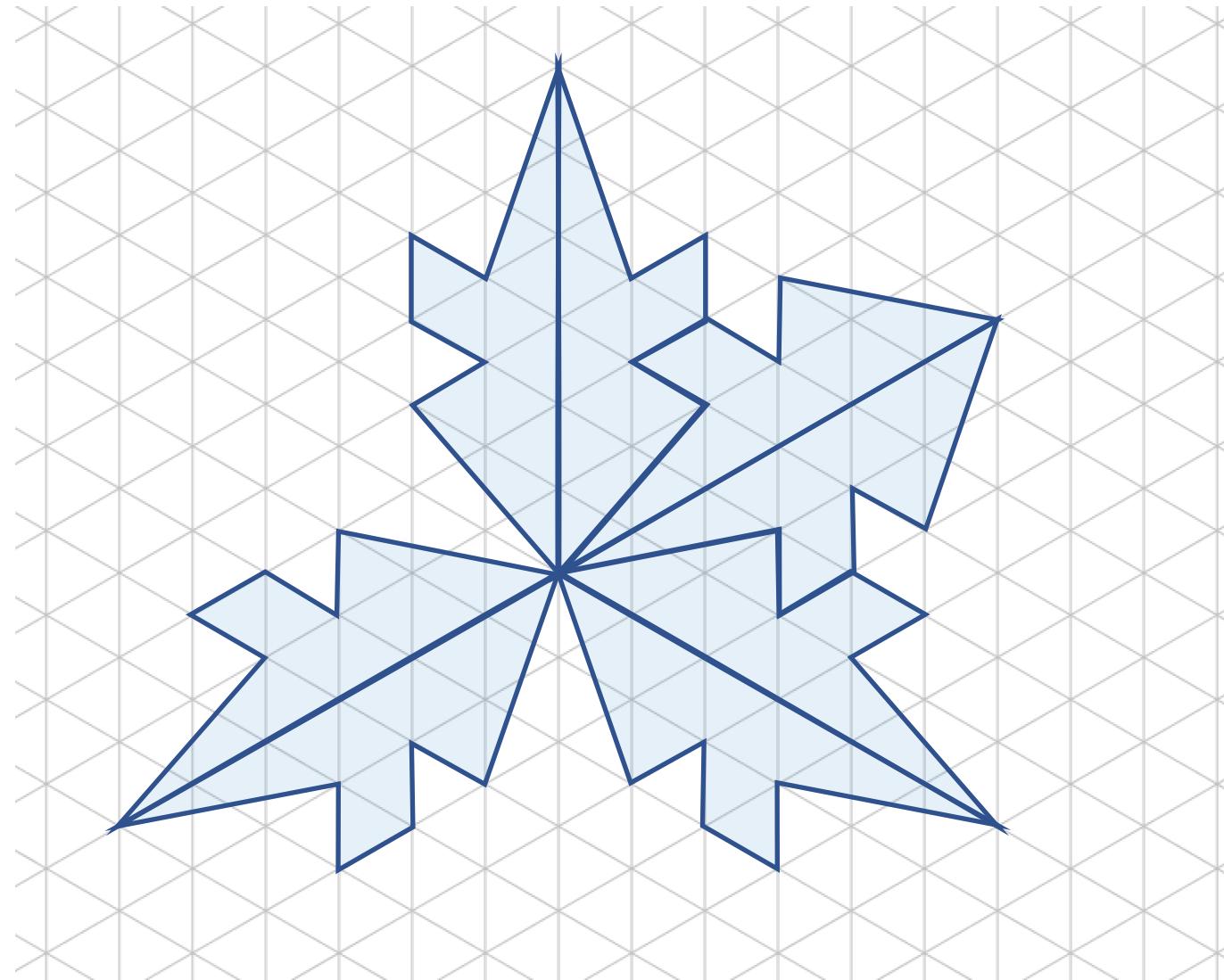
Fish



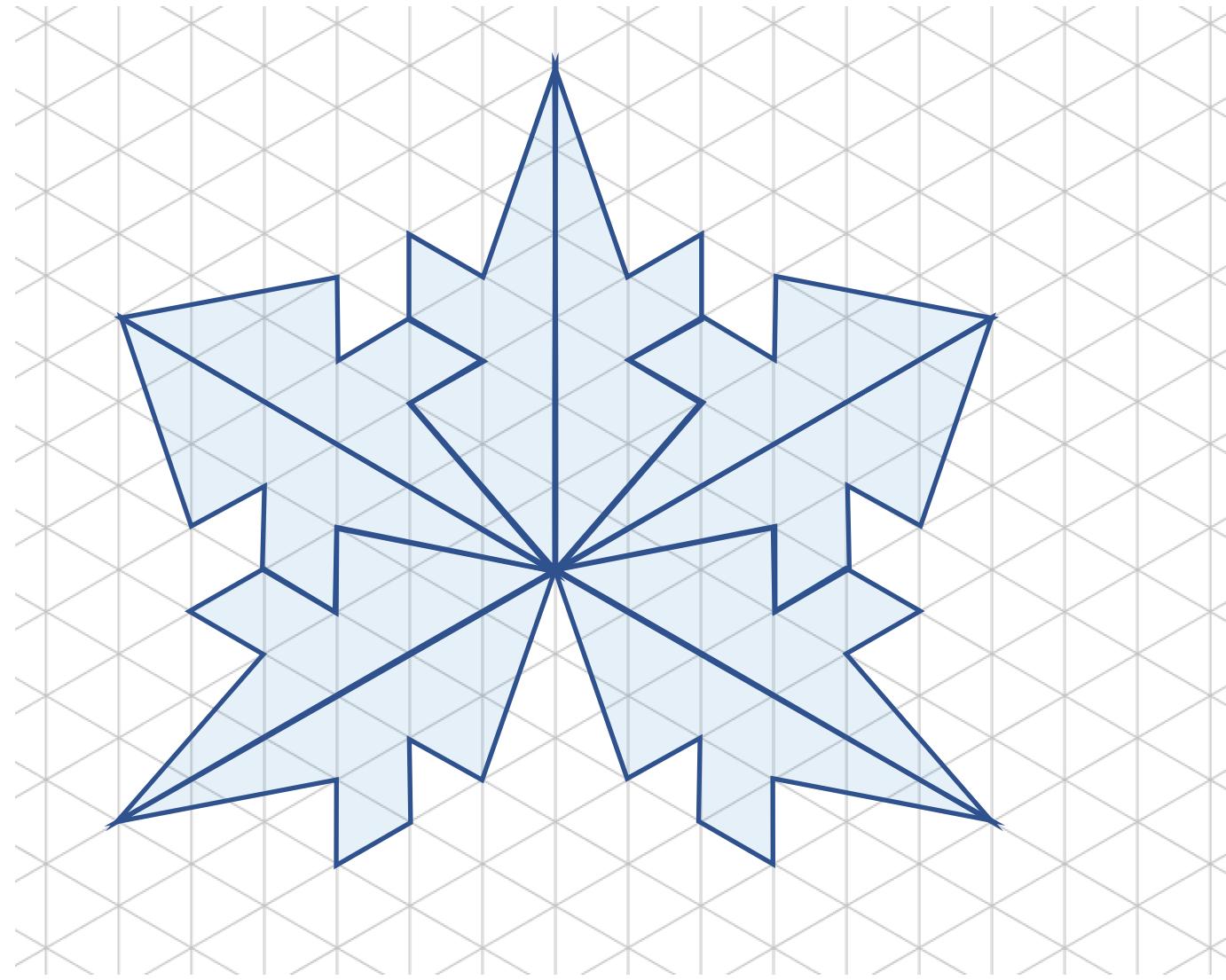
Fish



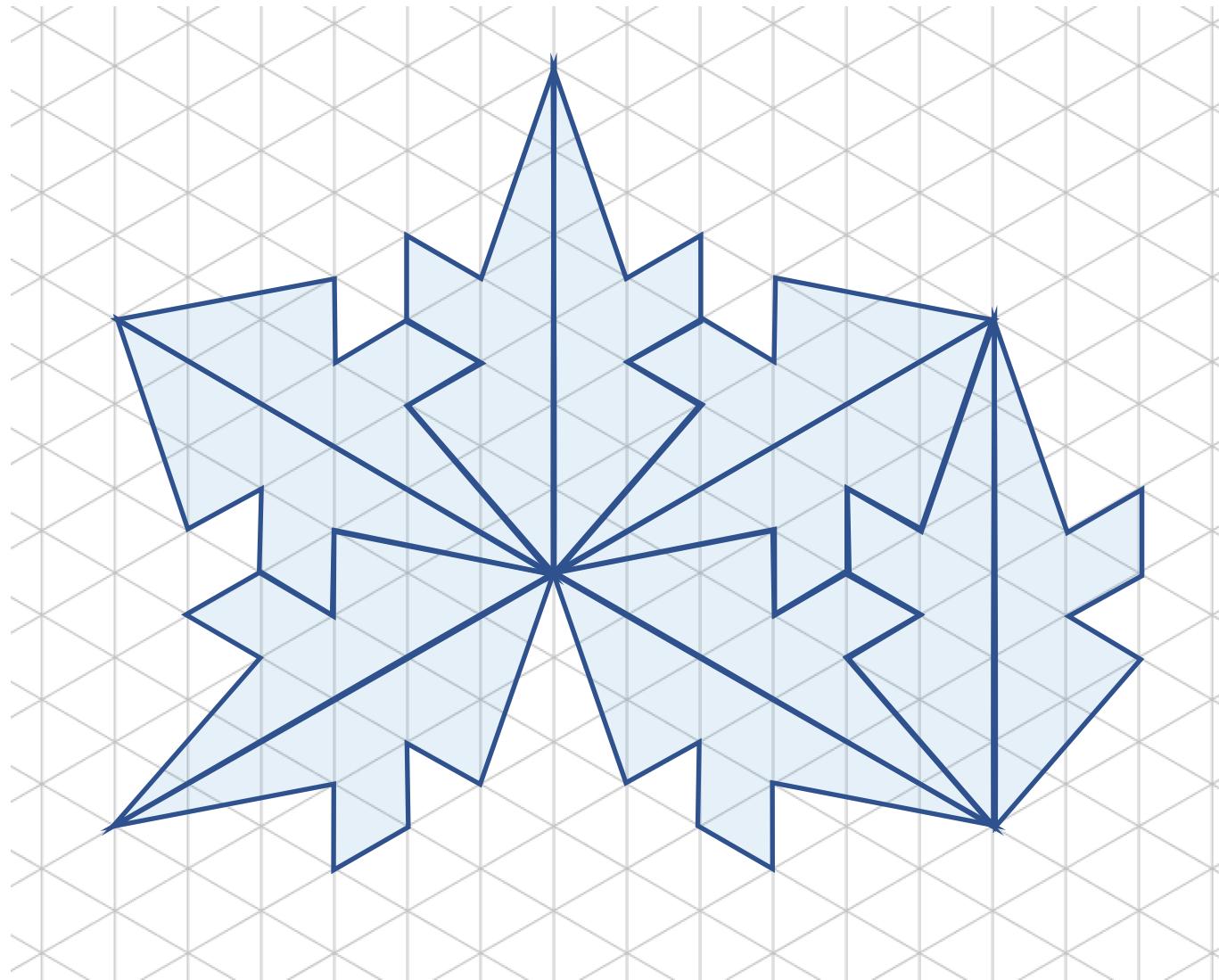
Fish



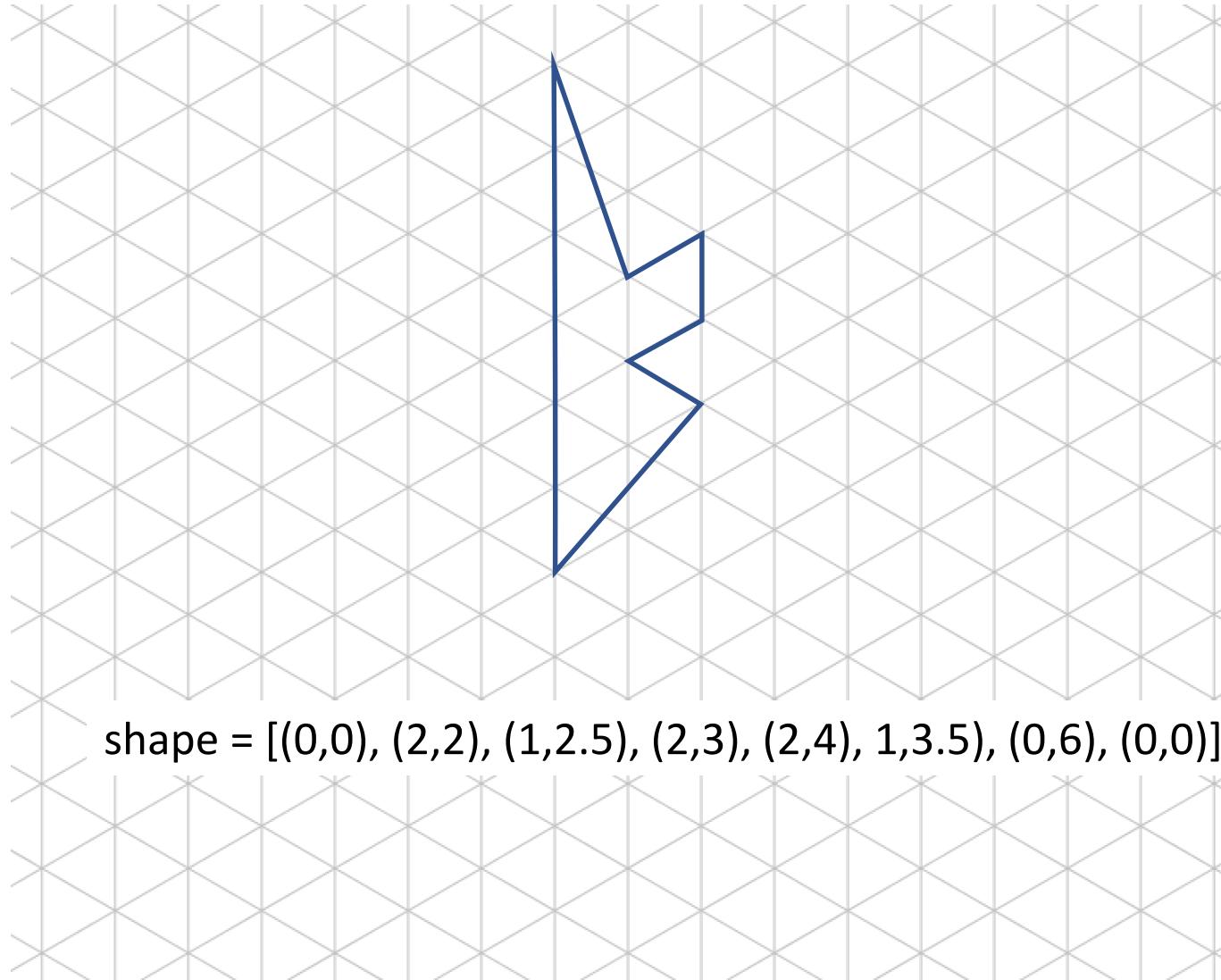
Fish



Fish



Fish



Fish

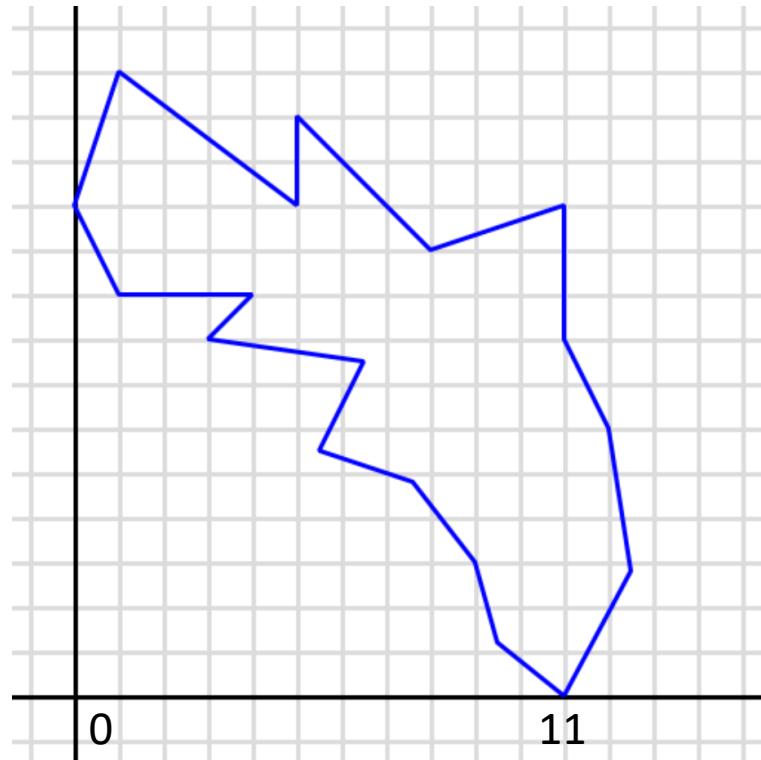
Back

Circular Fish



"Circular Fish", 1956 woodcut

Circular Fish

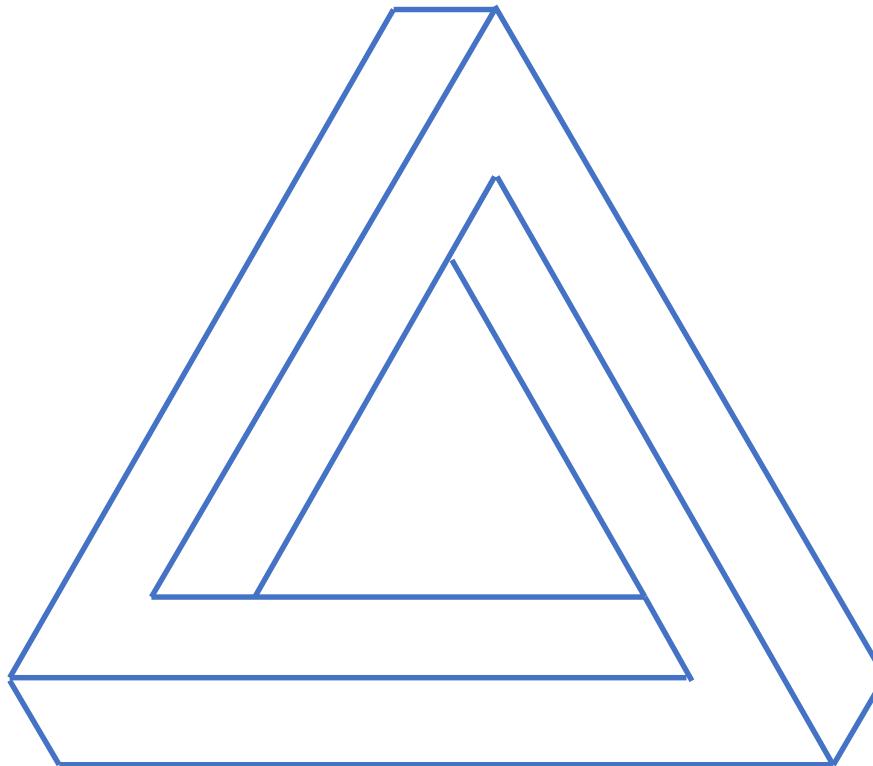


```
fish = [ (11, 0), (12.5, 2.8), (12, 6), (11, 8), (11, 11), (8, 10), (5, 13), (5, 11), (1, 14), (0, 11), (1, 9), (4, 9),  
       (3, 8), (6.5, 7.5), (5.5, 5.5), (7.55, 4.8), (9, 3), (9.5, 1.2), (11, 0)]
```

Circular Fish

Back

Penrose



Penrose



Penrose



Penrose



Penrose



Penrose



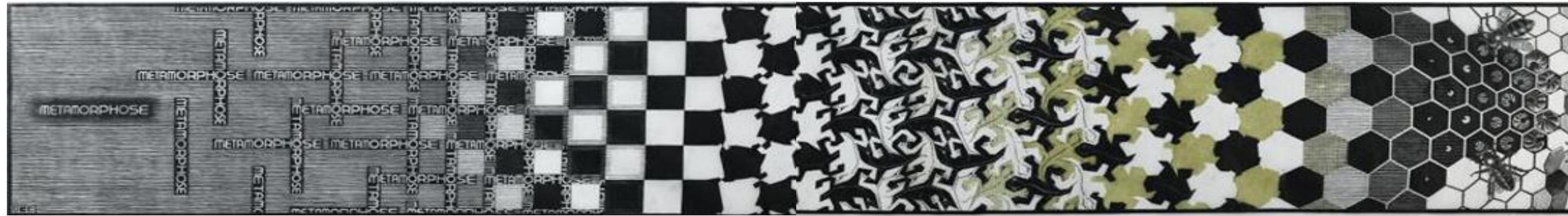
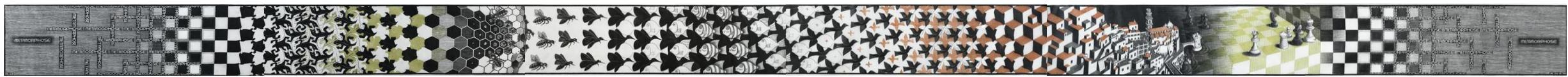
Penrose

Back

Examples

Triangles

Birds



"Metamorphose II", 1940 woodcut