

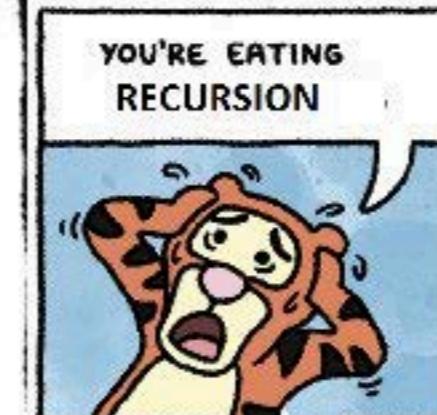
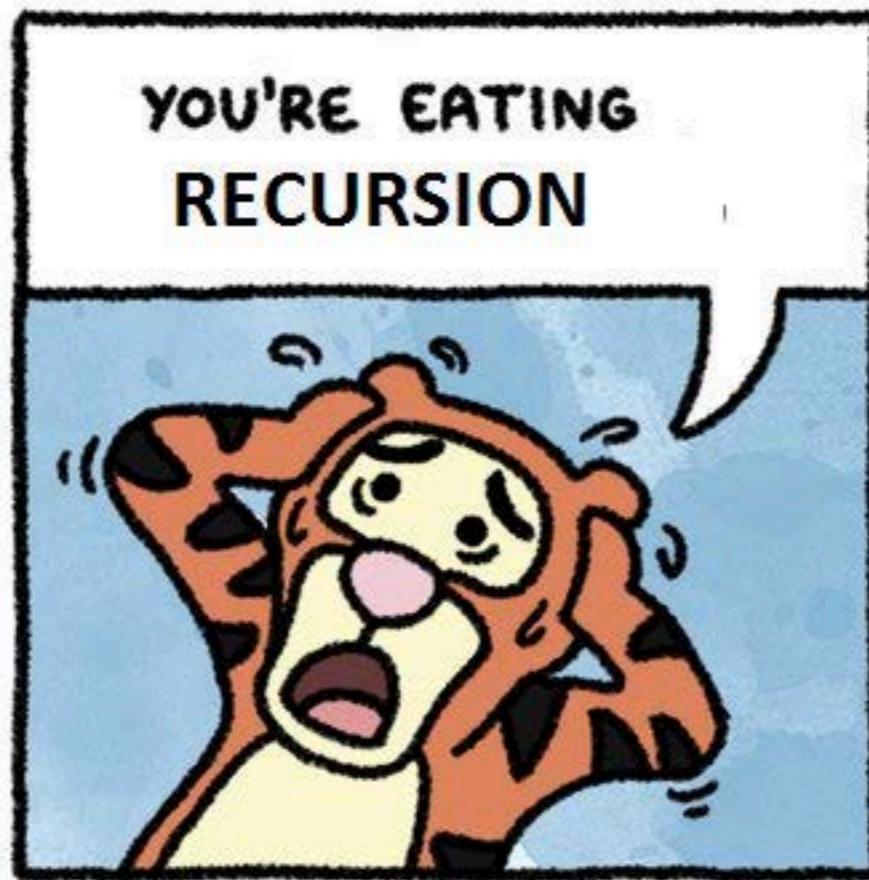
Module 07 RECURSION

Here we go again

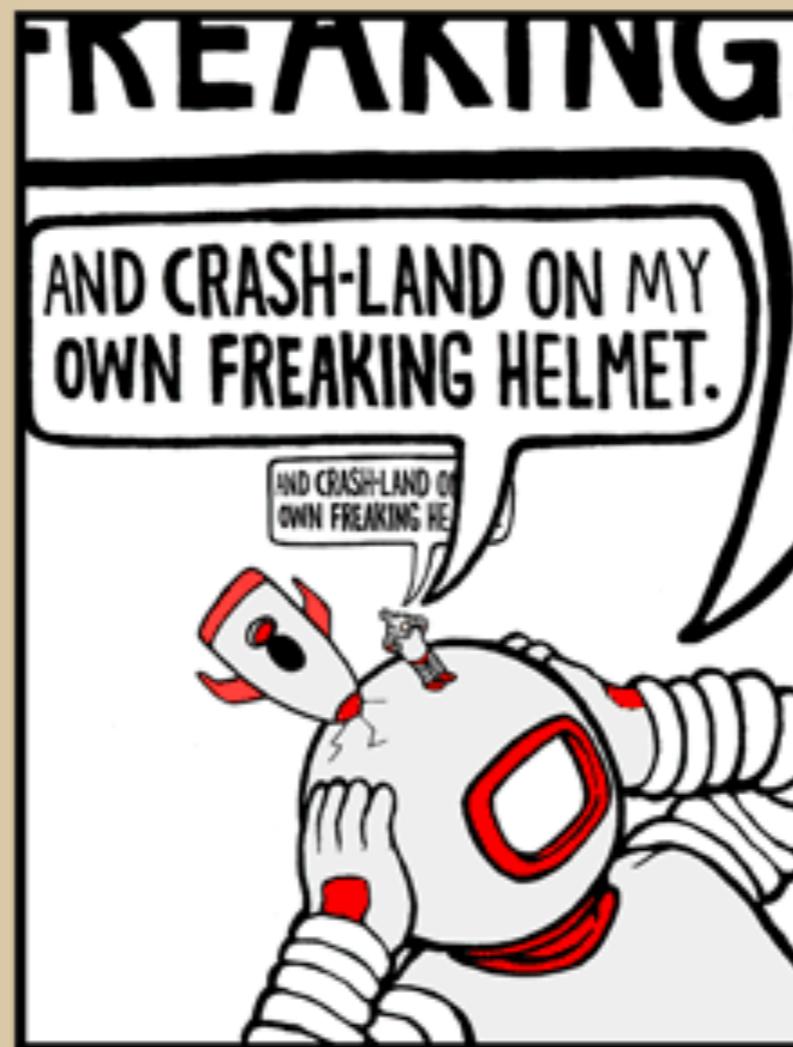


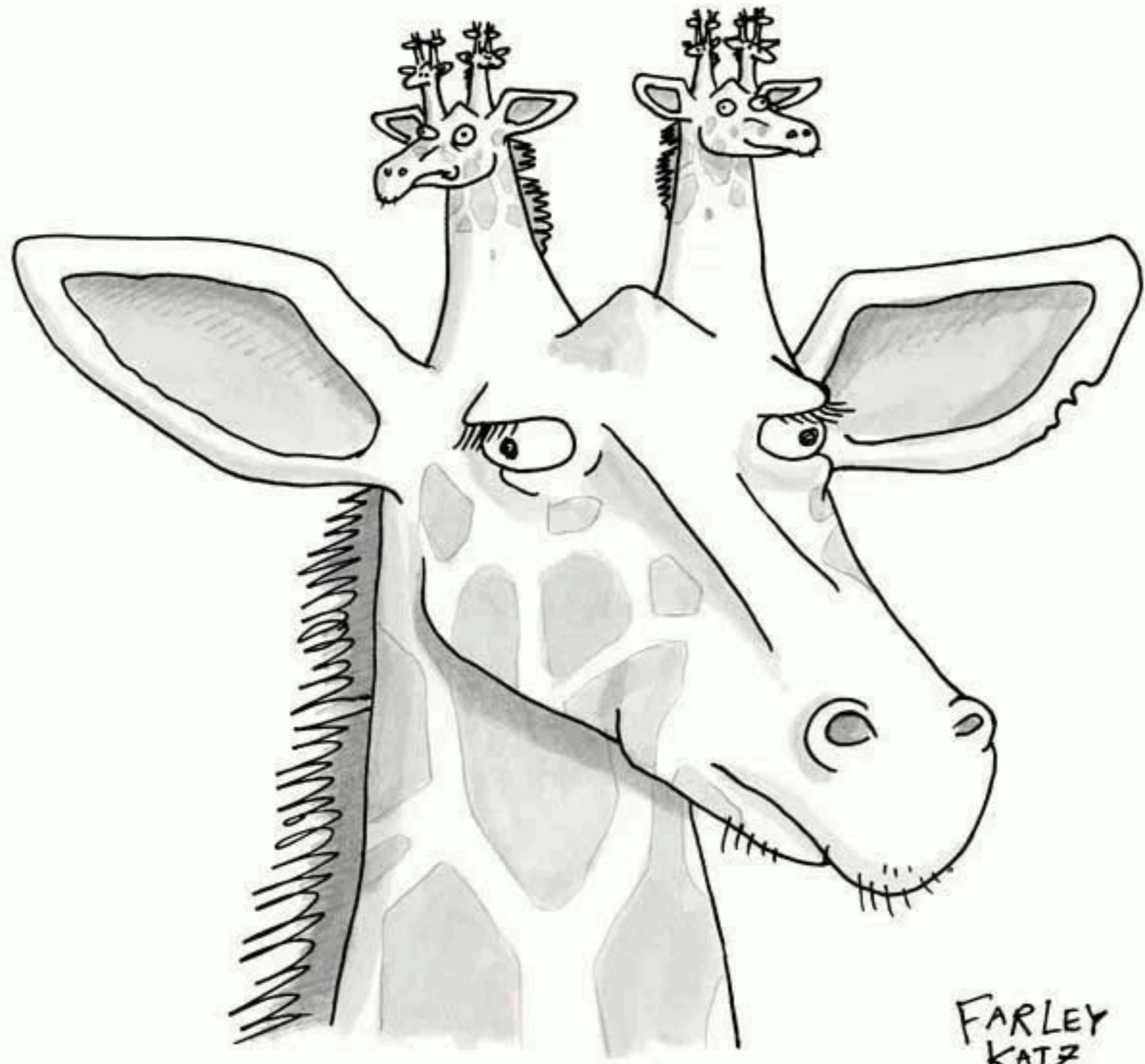
SAFELY ENDANGERED

SWEET JESUS, POOH!
THAT'S NOT HONEY



[@Zerglinator]?





FARLEY
KATZ



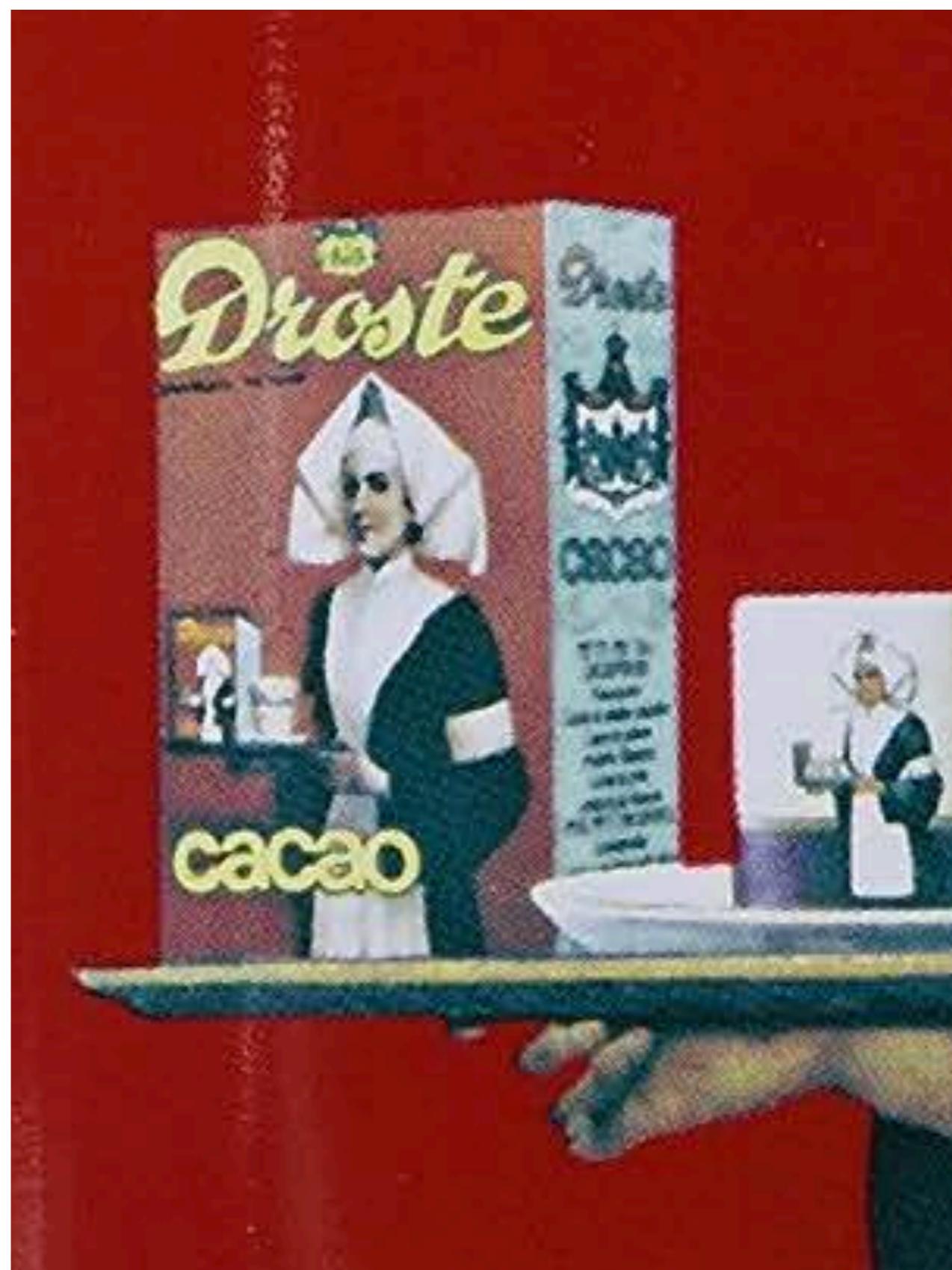
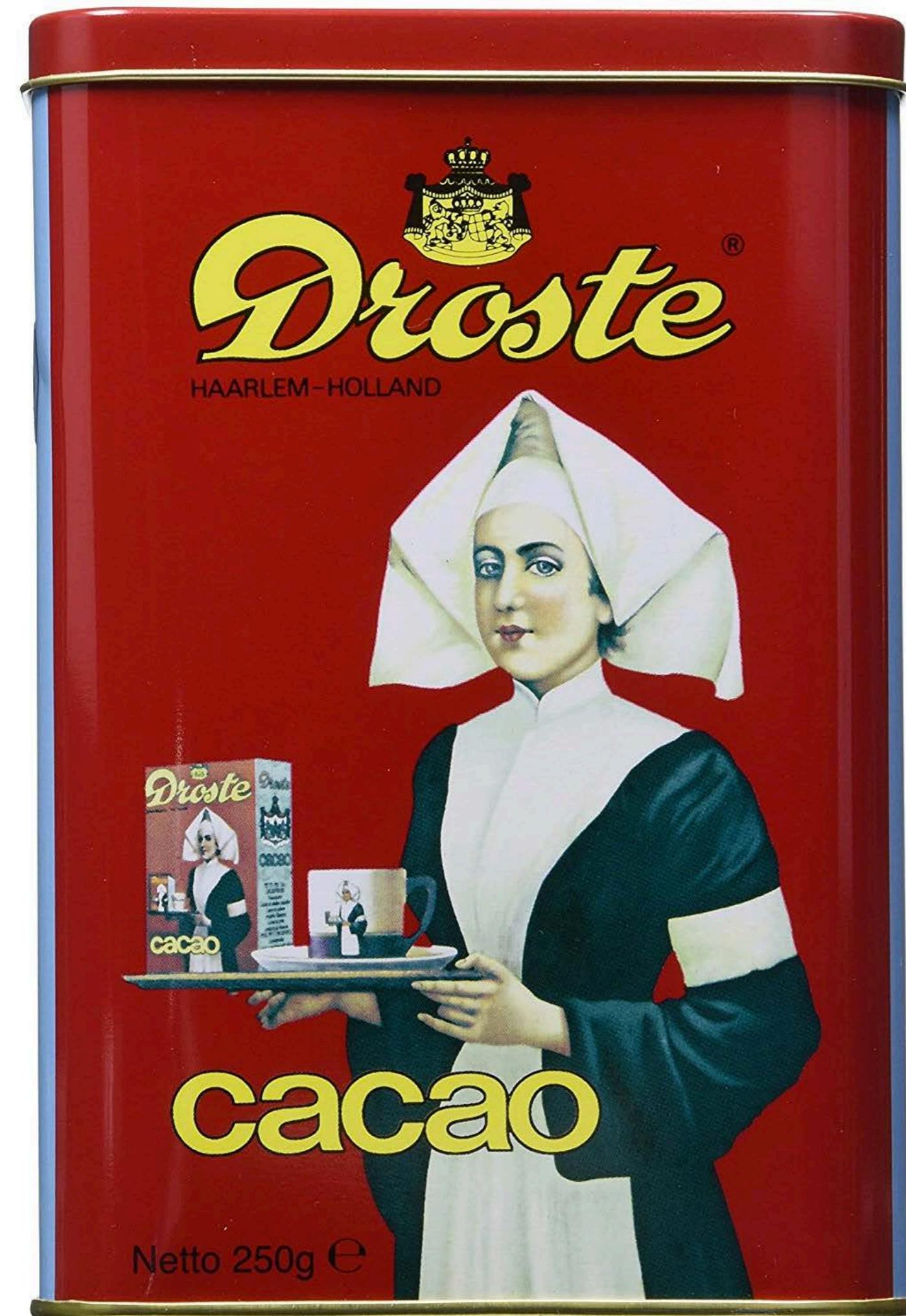
Droste®

HAARLEM - HOLLAND



cacao

Netto 250g e

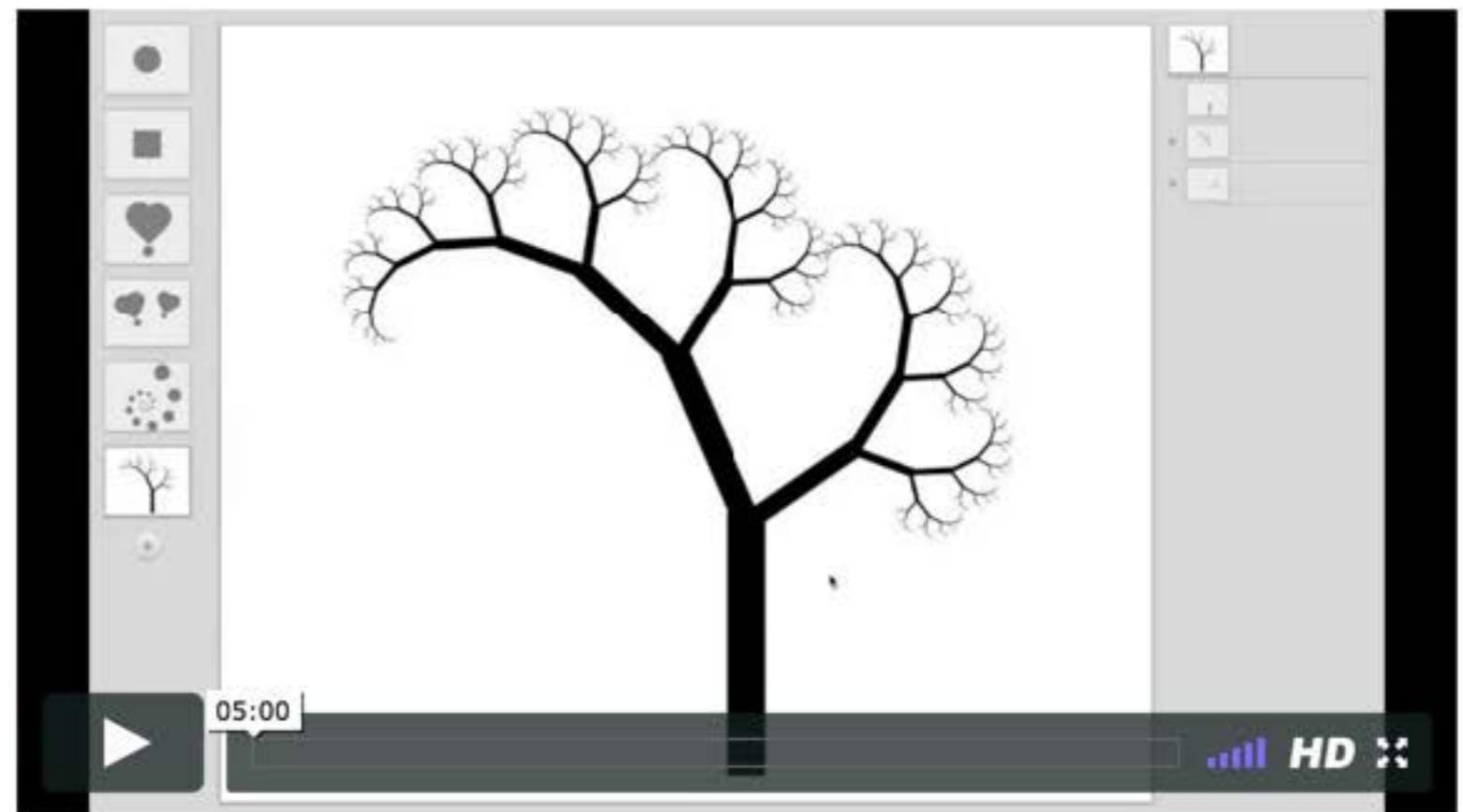


recursivedrawing.com

Recursive Drawing is an exploration of user interface ideas towards the development of a spatially-oriented programming environment.

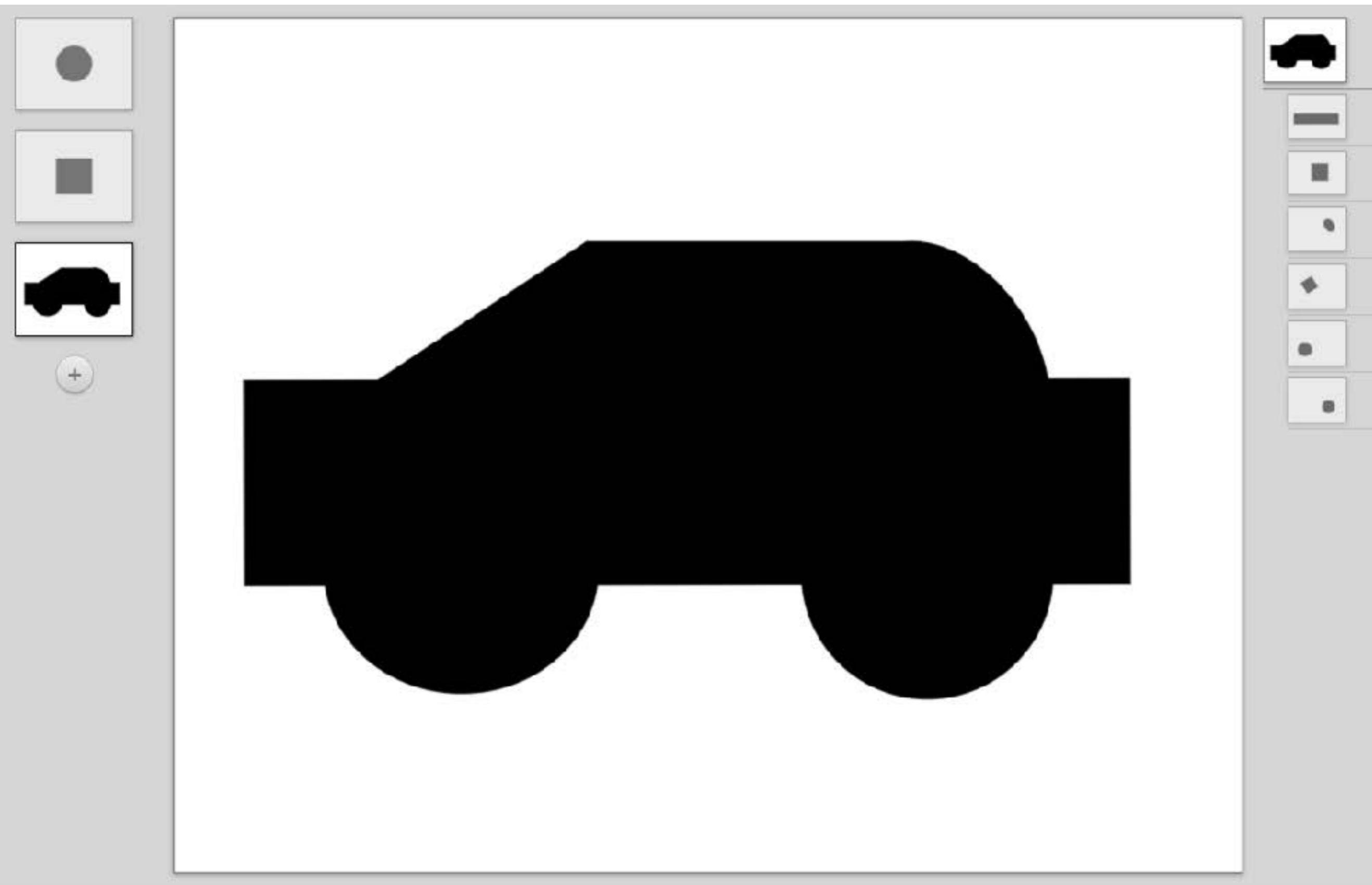
Start Drawing!

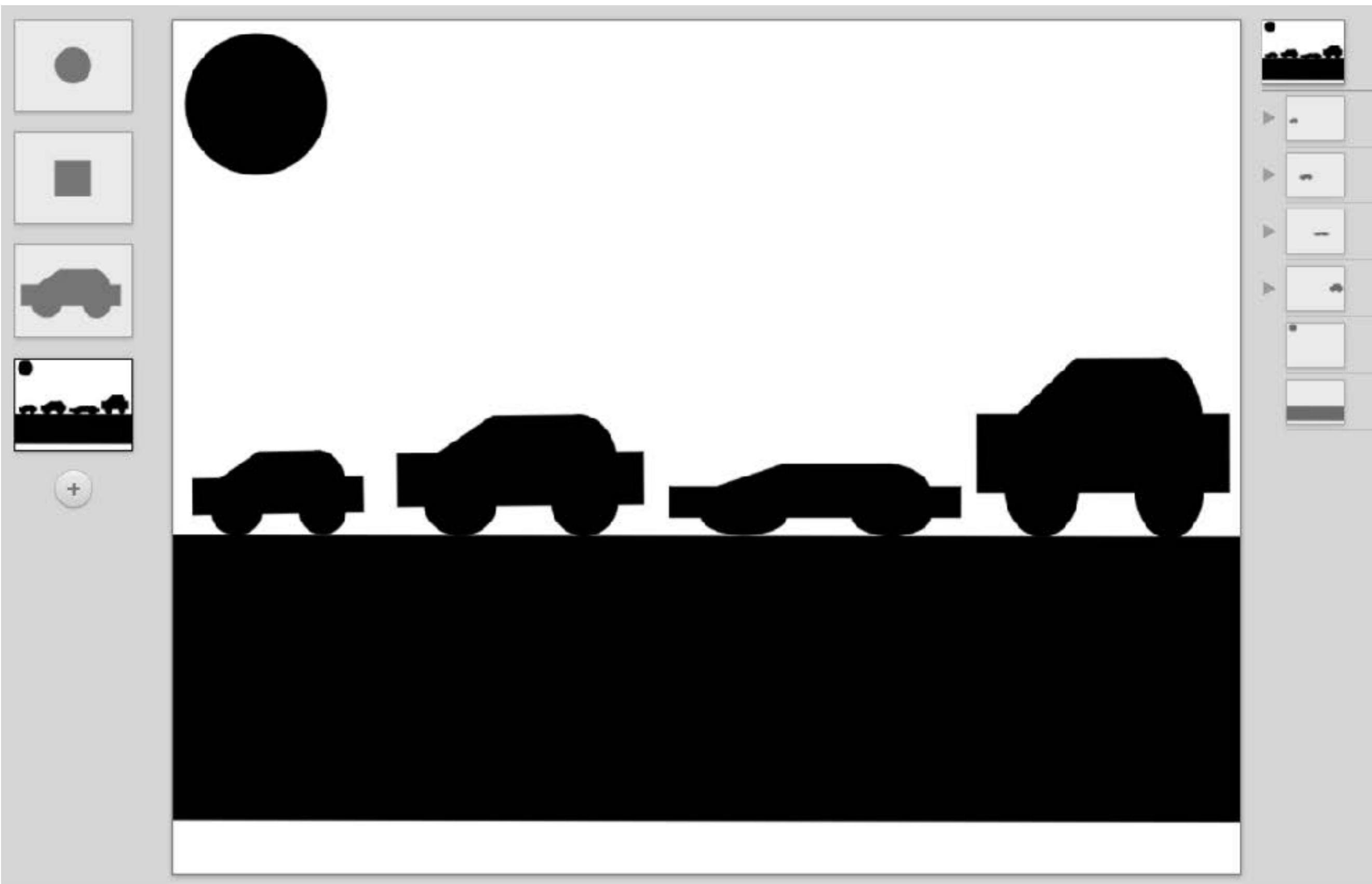
Complete list of instructions
(written up by Rory O'Kane)

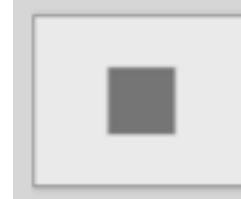
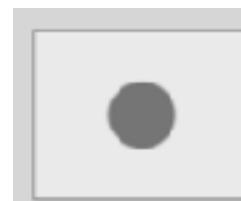


Recursive Drawing is by **Toby Schachman** for his **ITP** thesis project
Alternative Programming Interfaces for Alternative Programmers.

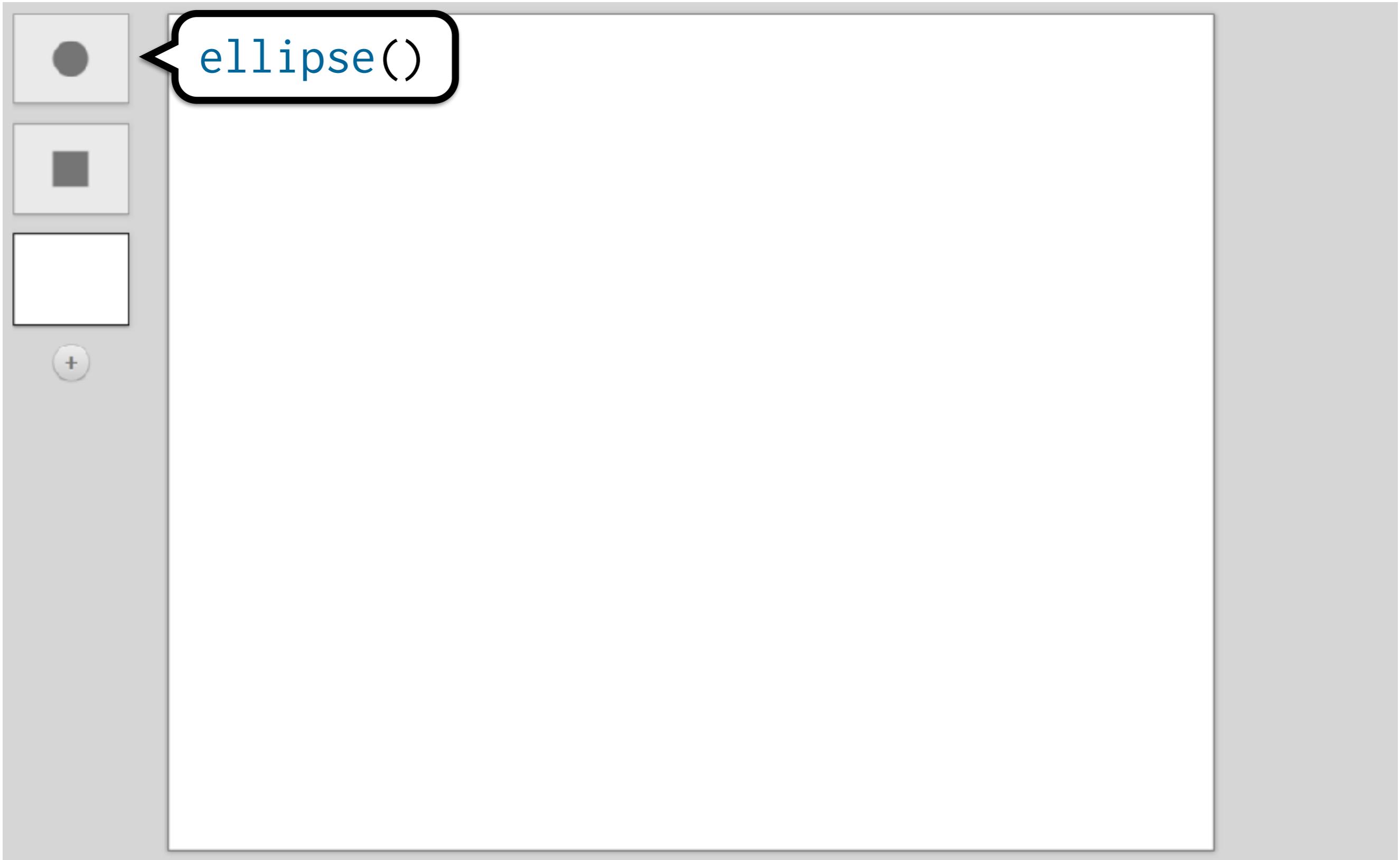
Fork me on GitHub







+



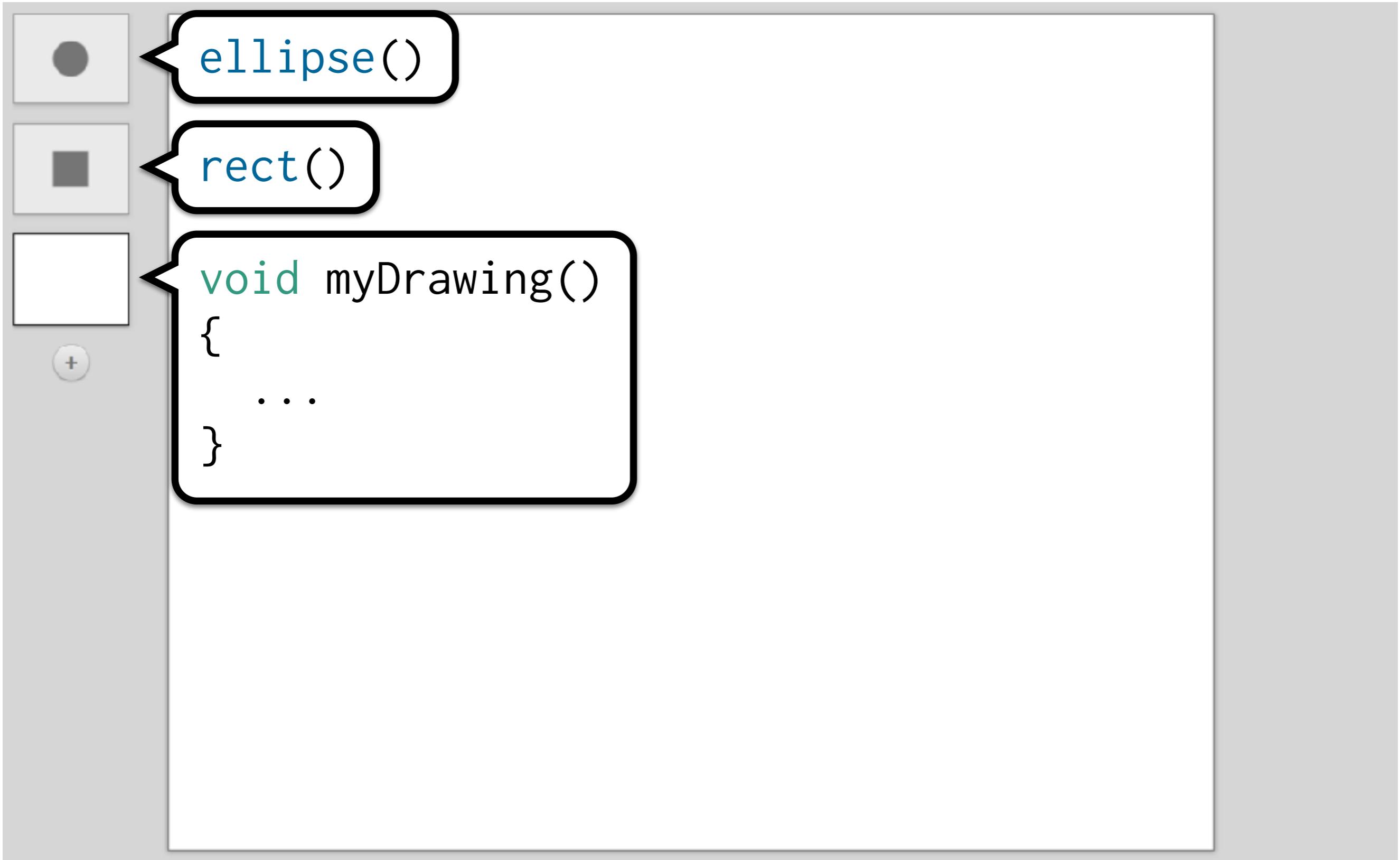
ellipse()



A Scratch workspace interface. On the left is a vertical toolbar with four icons: a grey circle, a grey square, a white rectangle, and a plus sign. To the right is a large, empty white canvas area. Two speech bubble-shaped callouts are overlaid on the screen. The top callout contains the text "ellipse()" in blue font, and the bottom callout contains the text "rect()" in blue font. Both callouts have black outlines and point towards the center of the workspace.

ellipse()

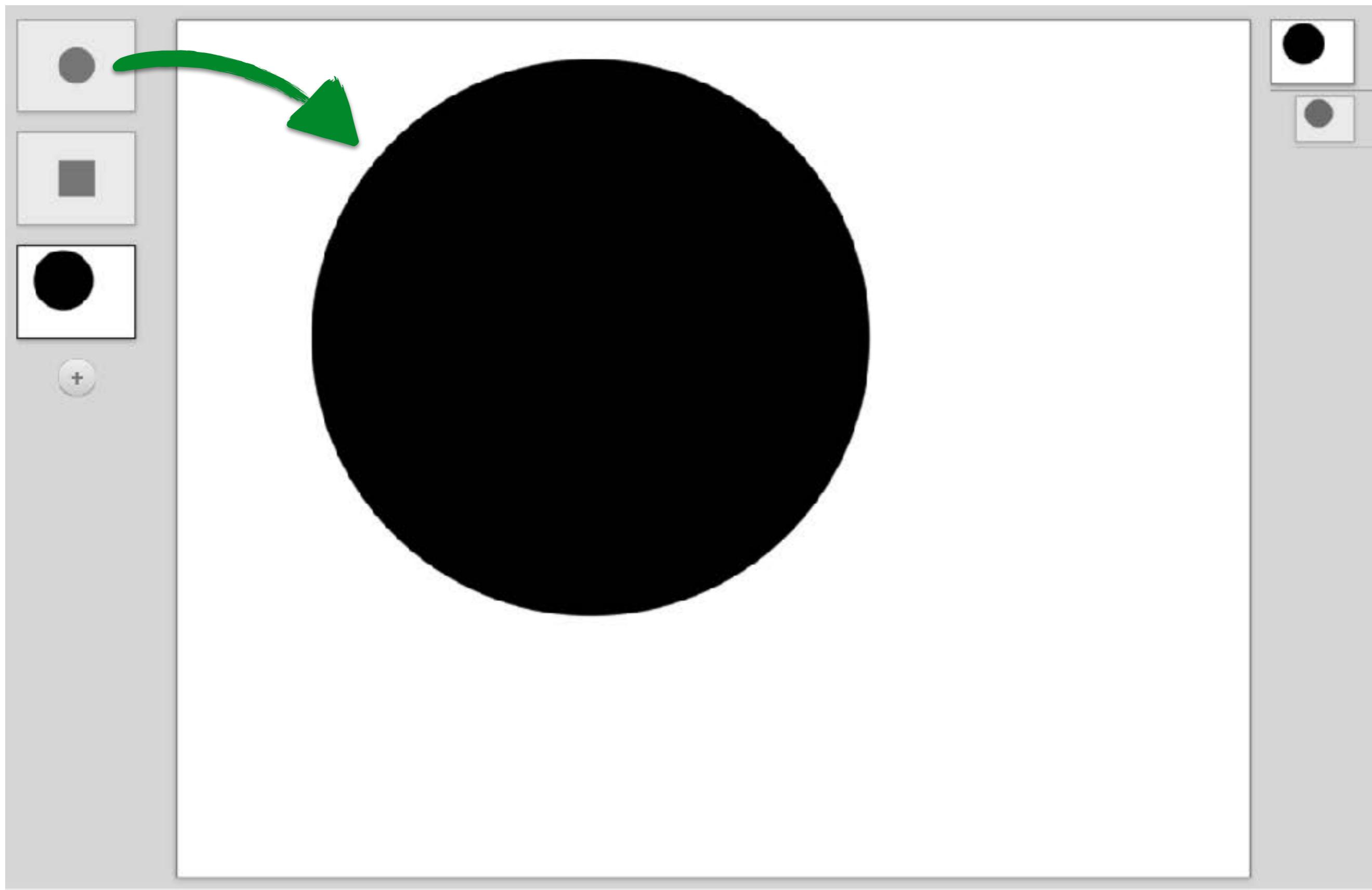
rect()

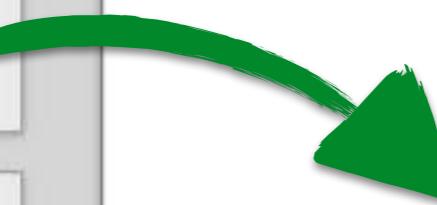


ellipse()

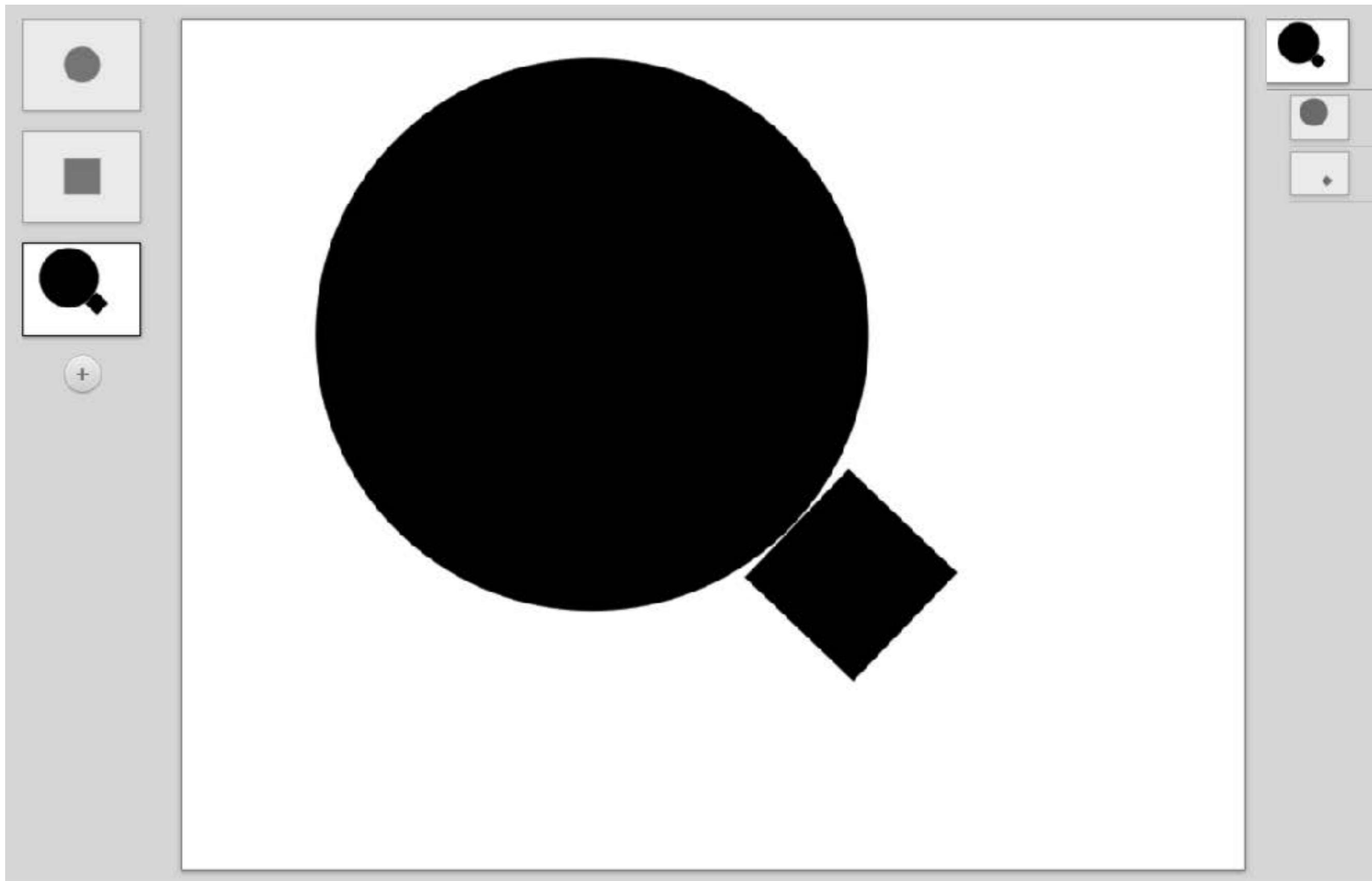
rect()

```
void myDrawing()
{
    ...
}
```



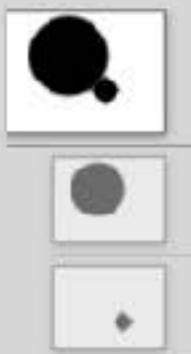
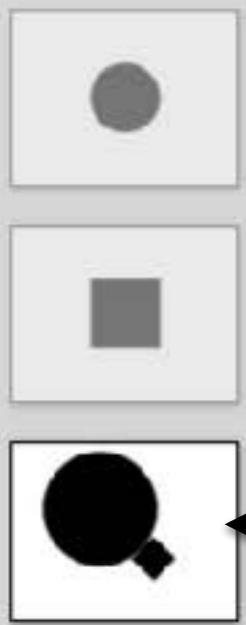
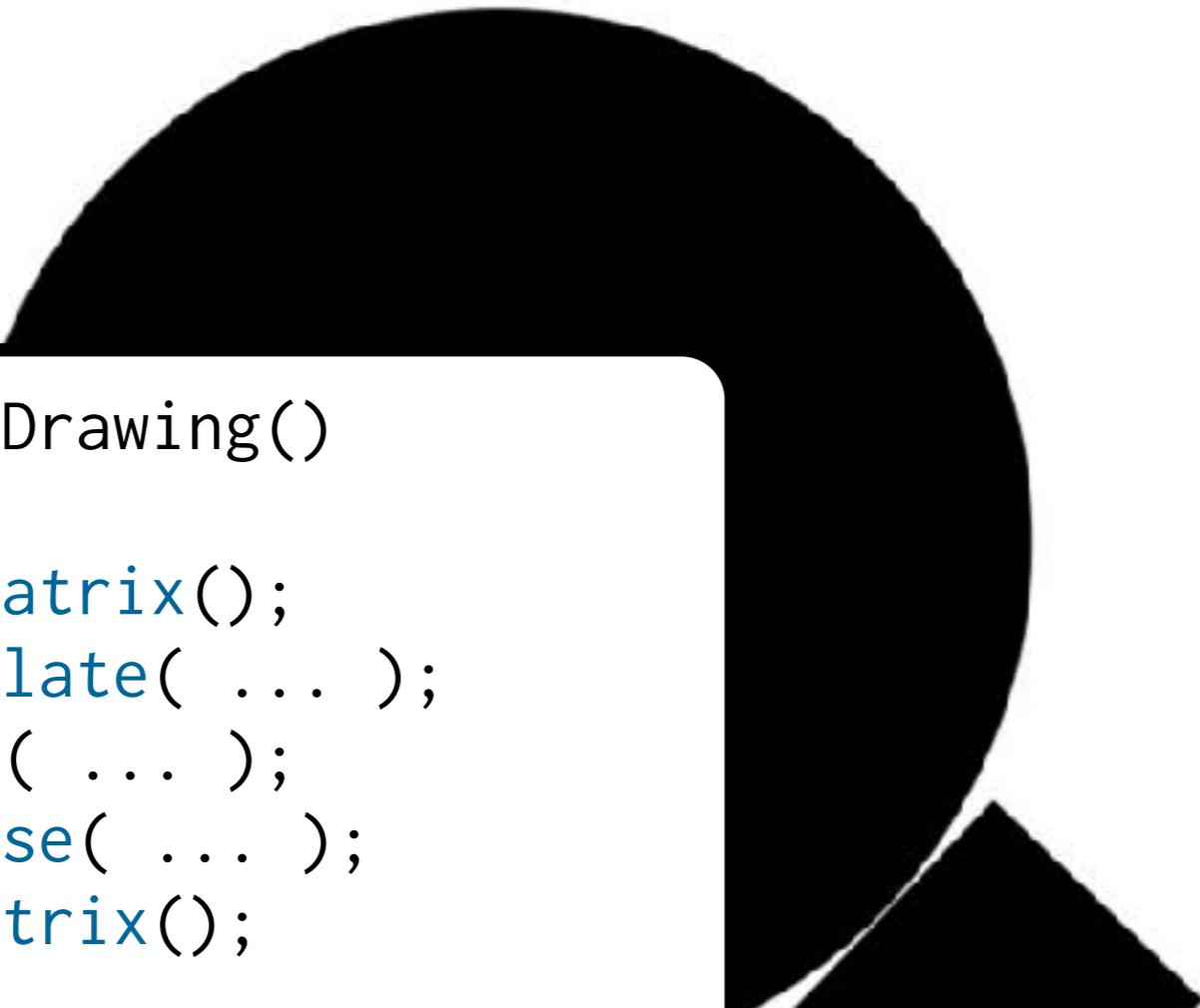


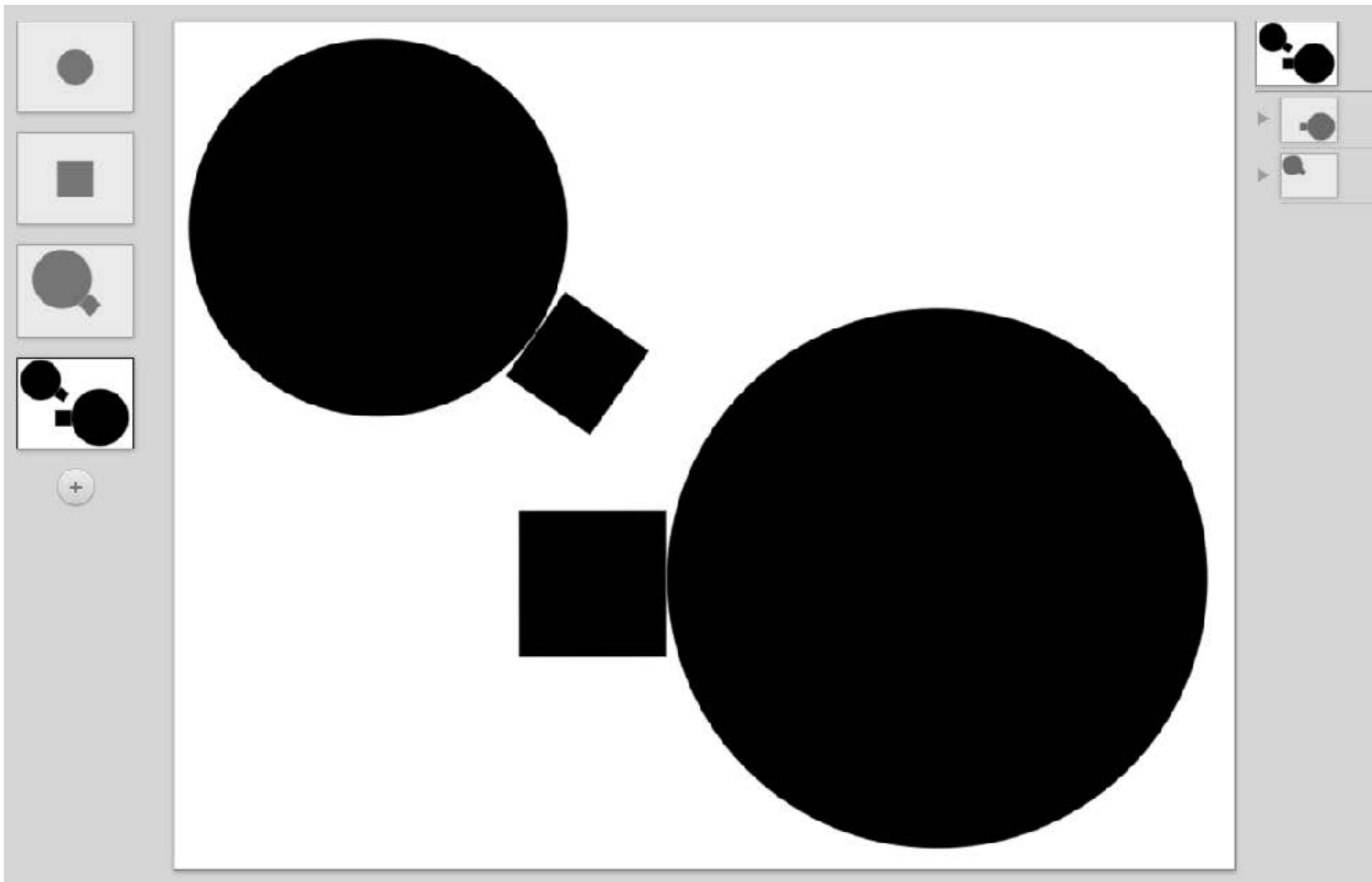
```
void myDrawing()
{
    pushMatrix();
    translate( ... );
    scale( ... );
    ellipse( ... );
    popMatrix();
}
```



```
void myDrawing()
{
    pushMatrix();
    translate( ... );
    scale( ... );
    ellipse( ... );
    popMatrix();

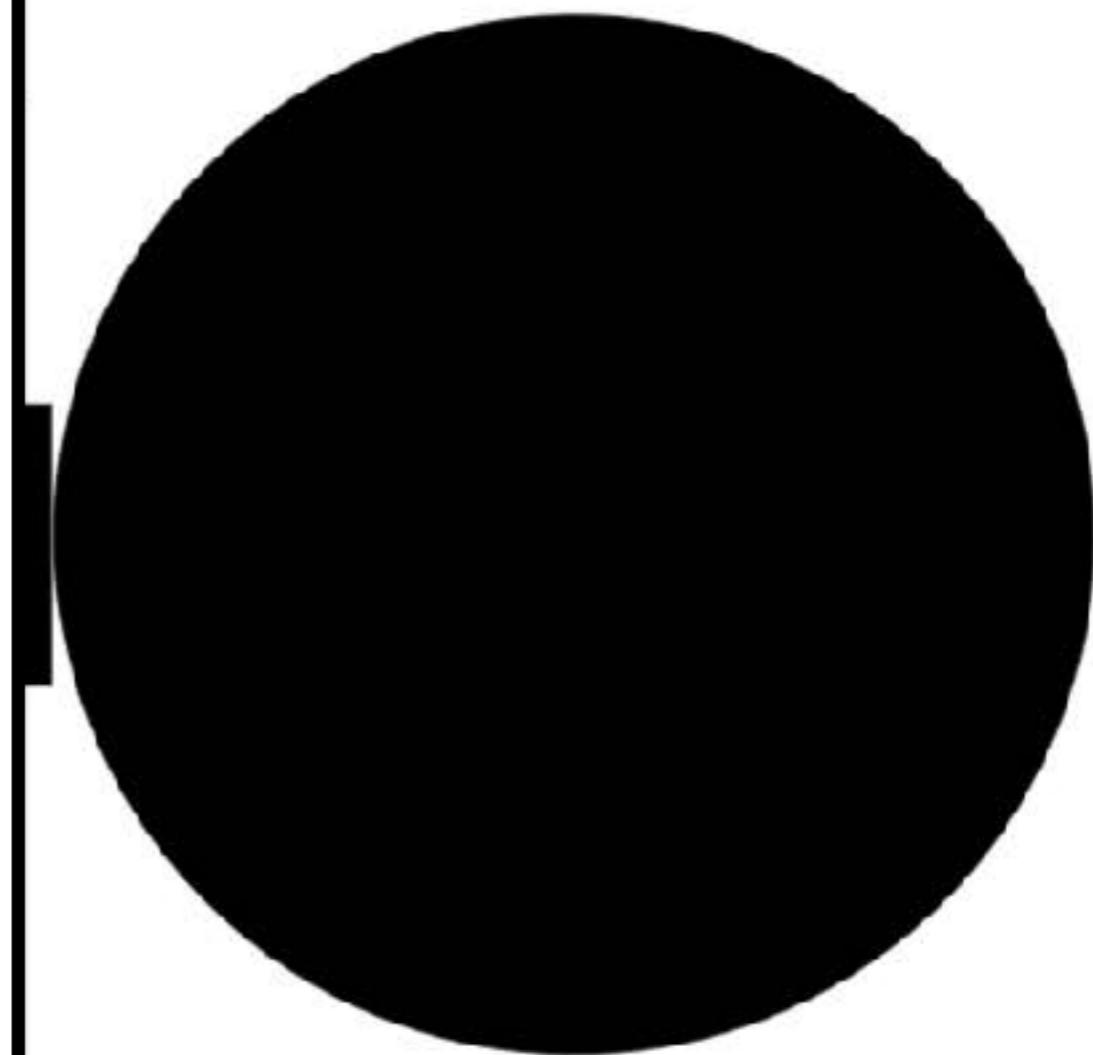
    pushMatrix();
    translate( ... );
    rotate( ... );
    scale( ... );
    rect( ... );
    popMatrix();
}
```

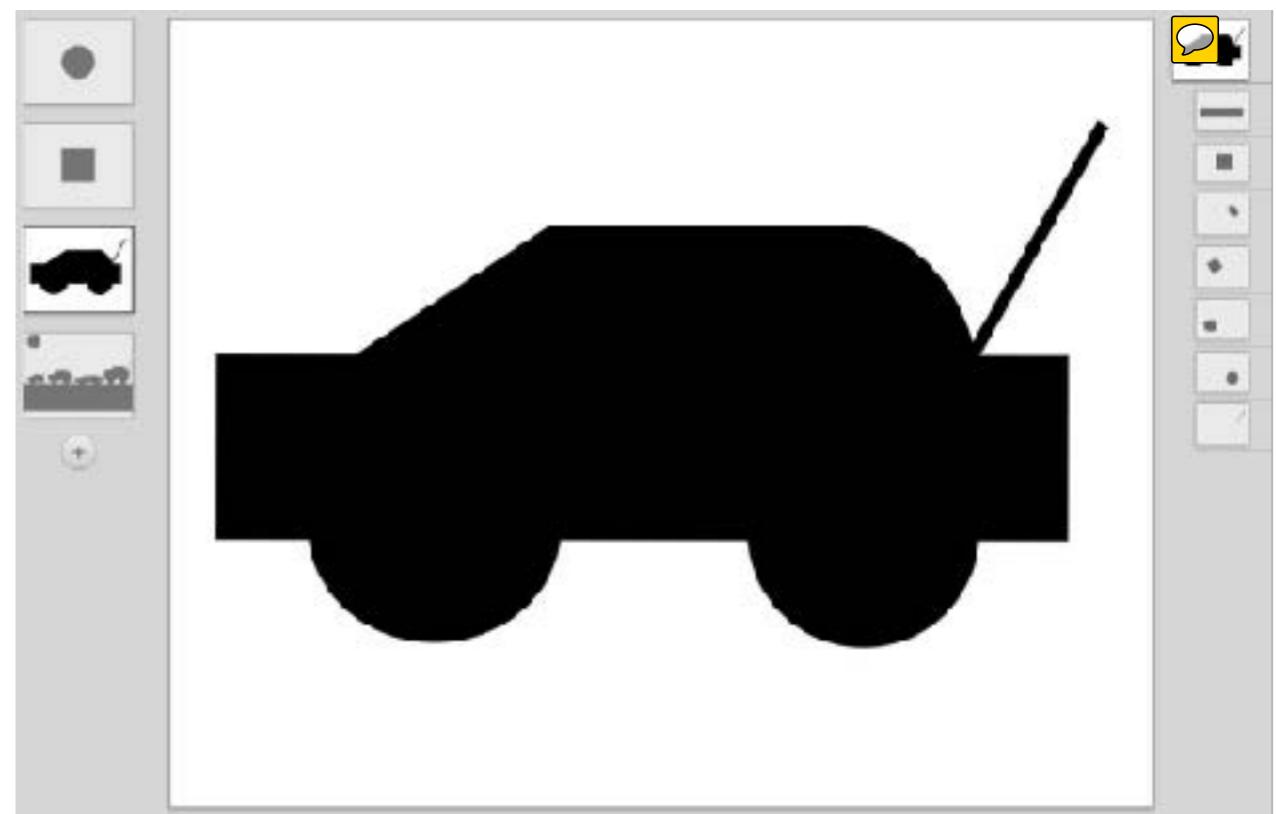


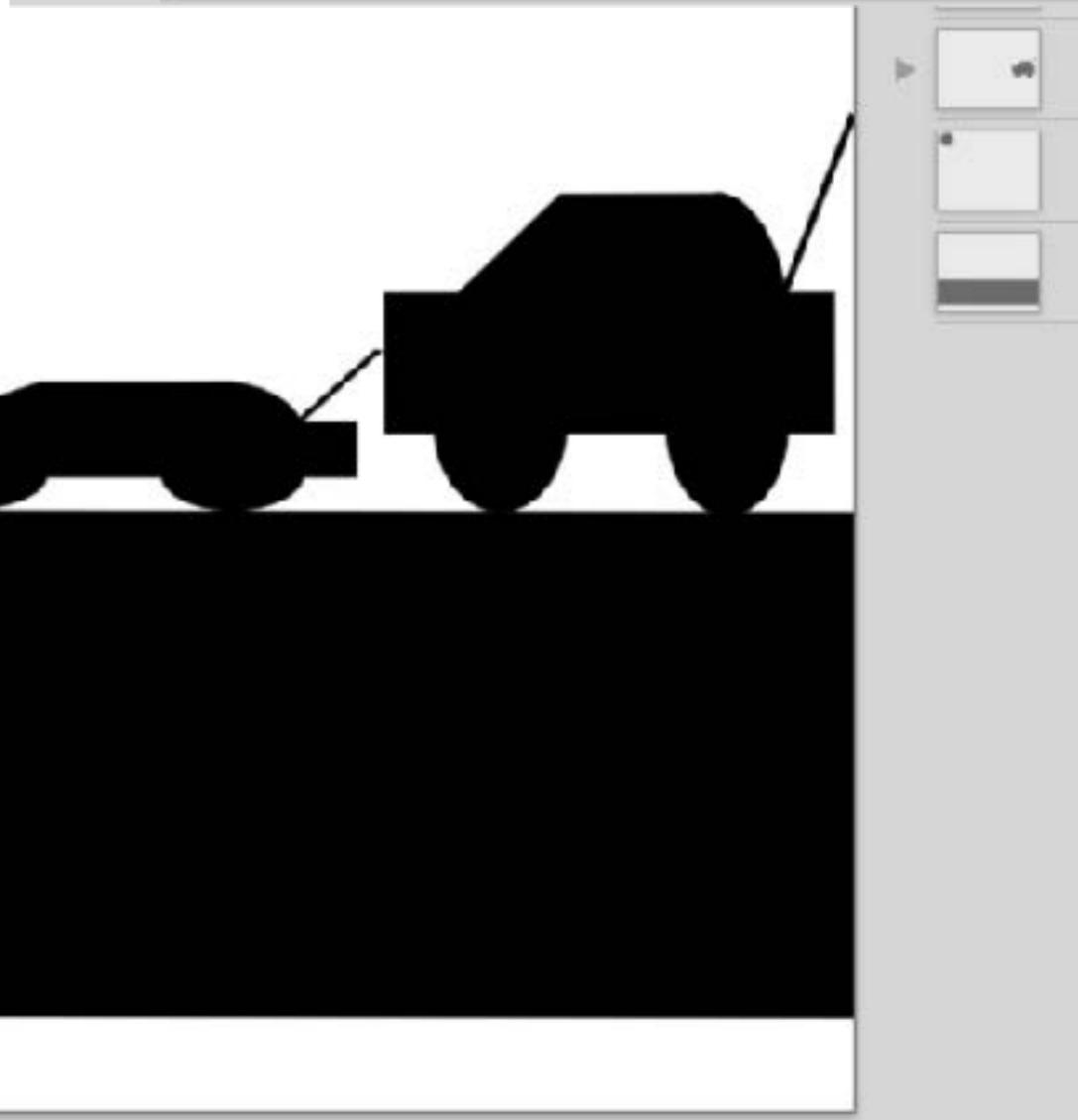
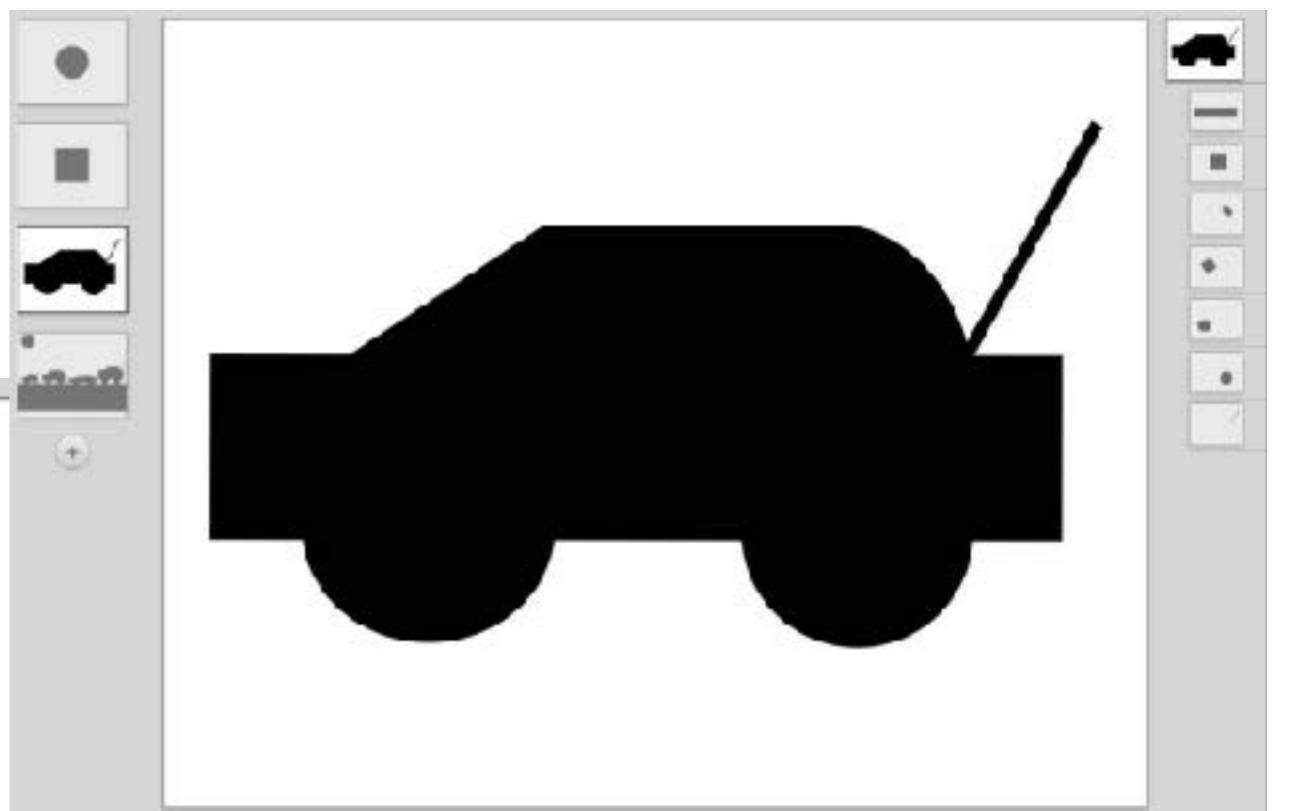
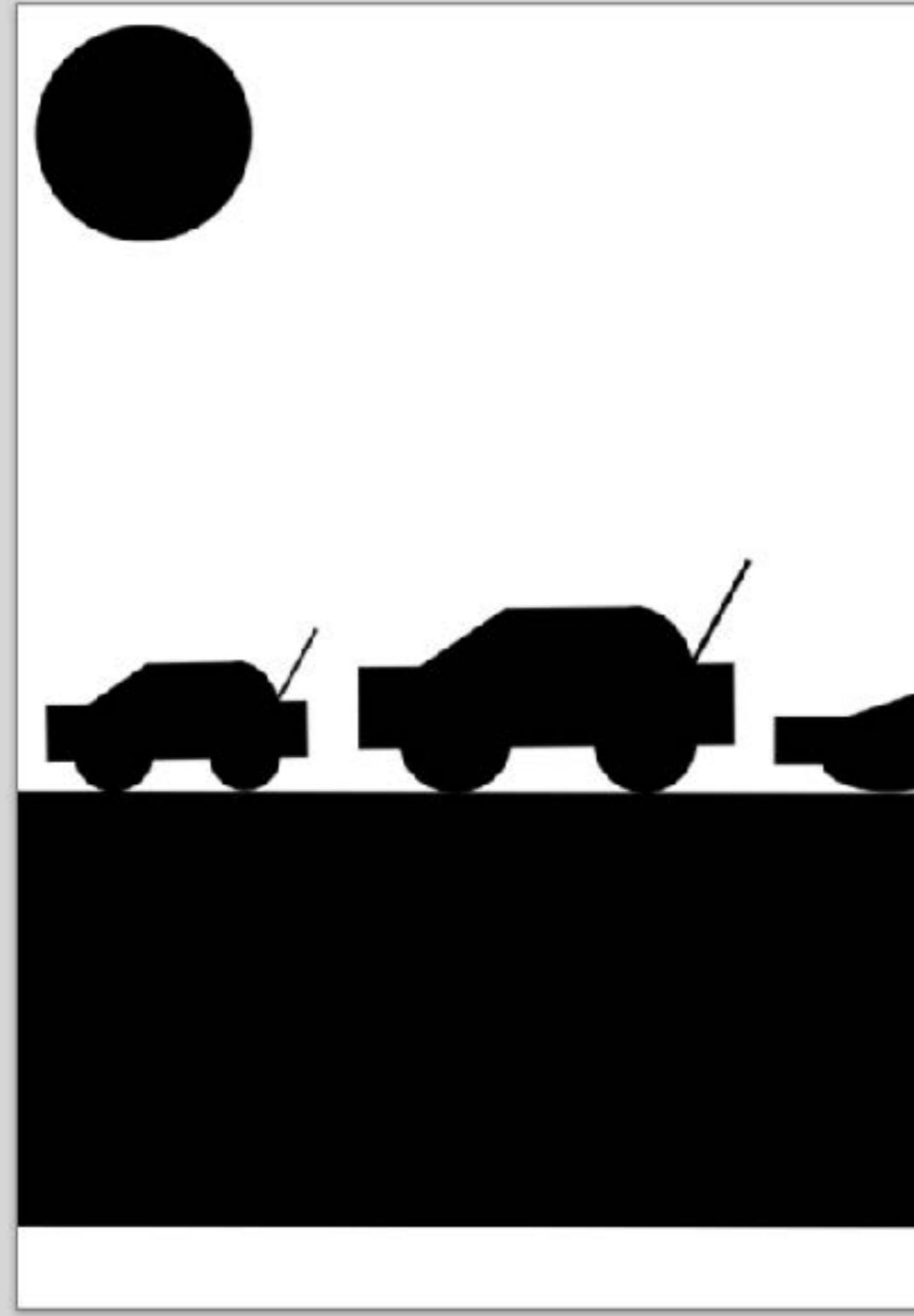


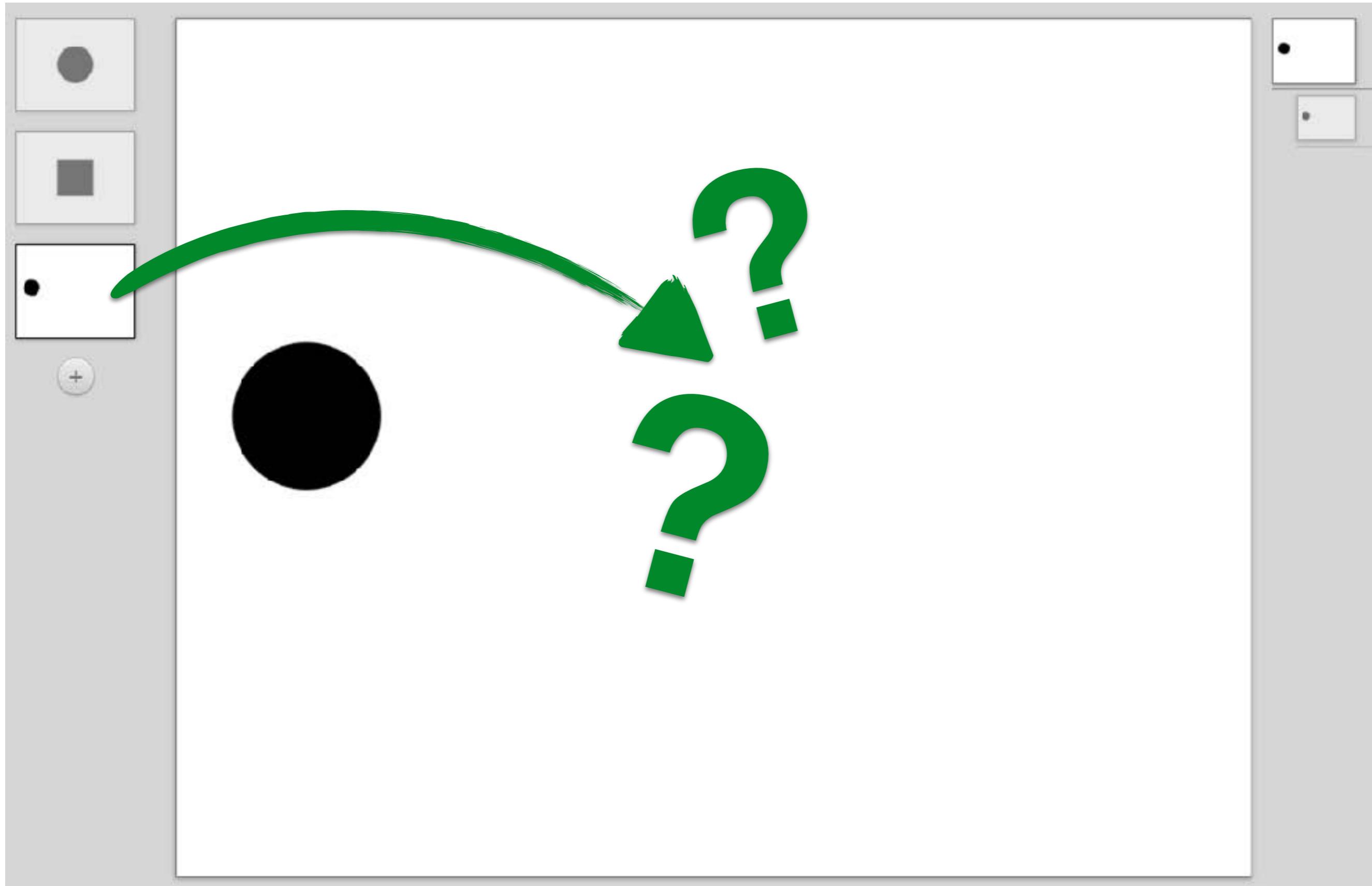
```
void myOtherDrawing()
{
    pushMatrix();
    translate( ... );
    rotate( ... );
    scale( ... );
    myDrawing();
    popMatrix();

    pushMatrix();
    translate( ... );
    rotate( ... );
    scale( ... );
    myDrawing();
    popMatrix();
}
```









```
void myDrawing()
{
    pushMatrix();
    translate( ... );
    scale( ... );
    ellipse( ... );
    popMatrix();

    pushMatrix();
    translate( ... );
    rotate( ... );
    scale( ... );
    myDrawing();
    popMatrix();
}
```



```
void myDrawing()
{
    pushMatrix();
    translate( ... );
    scale( ... );
    ellipse( ... );
    popMatrix();

    pushMatrix();
    translate( ... );
    rotate( ... );
    scale( ... );
    myDrawing();
    popMatrix();
}
```



**This doesn't actually work,
but it's on the right track.**

How to draw myDrawing:

- 1. Draw a circle**
- 2. Take one step east**
- 3. Draw myDrawing**

How to draw myDrawing:

- 1. Draw a circle**
- 2. Take one step east**
 - 1. Draw a circle**
 - 2. Take one step east**
 - 3. Draw myDrawing**

How to draw myDrawing:

- 1. Draw a circle**
- 2. Take one step east**
 - 1. Draw a circle**
 - 2. Take one step east**
 - 1. Draw a circle**
 - 2. Take one step east**
 - 3. Draw myDrawing**

How to draw myDrawing:

1. Draw a circle
2. Take one step east
 1. Draw a circle
 2. Take one step east
 1. Draw a circle
 2. Take one step east
 1. Draw a circle
 2. Take one step east
 1. Draw a circle
 2. Take one step east

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    myDrawing();
    popMatrix();
}
```

```
void setup()
{
    size( 600, 100 );
    background( 100 );
    myDrawing();
}
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    myDrawing();
    popMatrix();
}

void setup()
{
    size( 600, 100 );
    background( 100 );
    myDrawing();
}
```

**What's wrong with
this code?**

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );

    myDrawing();

    popMatrix();
}
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );

    popMatrix();
}
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );

    popMatrix();
    popMatrix();
}
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );

    popMatrix();

    popMatrix();
    popMatrix();
}
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    popMatrix();
    popMatrix();

    popMatrix();
    popMatrix();
}
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    ellipse( 50, 50, 100, 100 );
    popMatrix();
    translate( 150, 0 );

    popMatrix();
    popMatrix();
}

popMatrix();
popMatrix();
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    ellipse( 50, 50, 100, 100 );

    popMatrix();
    popMatrix();

    popMatrix();
    popMatrix();
}
```

```
void myDrawing()
{
    ellipse( 50, 50, 100, 100 );

    pushMatrix();
    translate( 150, 0 );
    myDrawing();
    popMatrix();
}
```

```
void setup()
{
    size( 600, 100 );
    background( 100 );
    myDrawing();
}
```

```
void myDrawing( int levels )
{
    ellipse( 50, 50, 100, 100 );

    if ( levels > 0 ) {
        pushMatrix();
        translate( 150, 0 );
        myDrawing( levels - 1 );
        popMatrix();
    }
}

void setup()
{
    size( 600, 100 );
    background( 100 );
    myDrawing();
}
```

```
void myDrawing( int levels ) {
```

```
    ellipse( 50, 50, 100, 100 );
```

```
    if ( levels > 0 ) {
```

```
        pushMatrix();
```

```
        translate( 150, 0 );
```

```
        myDrawing( levels - 1 );
```

```
        popMatrix();
```

```
}
```

```
}
```

```
void setup()
```

```
{
```

```
    size( 600, 100 );
```

```
    background( 100 );
```

```
    myDrawing();
```

```
}
```

How many more
levels of recursion?

```
void myDrawing( int levels ) {  
    ellipse( 50, 50, 100, 100 );  
  
    if ( levels > 0 ) {  
        pushMatrix();  
        translate( 150, 0 );  
        myDrawing( levels - 1 );  
        popMatrix();  
    }  
}  
  
void setup()  
{  
    size( 600, 100 );  
    background( 100 );  
    myDrawing();  
}
```

How many more levels of recursion?

Are we done yet?

```
void myDrawing( int levels )
```

```
{
```

```
    ellipse( 50, 50, 100, 100 );
```

```
    if ( levels > 0 ) {
```

```
        pushMatrix();
```

```
        translate( 150, 0 );
```

```
        myDrawing( levels - 1 );
```

```
        popMatrix();
```

```
}
```

```
}
```

```
void setup()
```

```
{
```

```
    size( 600, 100 );
```

```
    background( 100 );
```

```
    myDrawing();
```

```
}
```

How many more
levels of recursion?

Are we done yet?

Make progress
towards the end.

```
void myDrawing( int levels )
{
    ellipse( 50, 50, 100, 100 );

    if ( levels > 0 ) {
        pushMatrix();
        translate( 150, 0 );
        myDrawing( levels - 1 );
        popMatrix();
    }
}

void setup()
{
    size( 600, 100 );
    background( 100 );
    myDrawing( 6 );
}
```

```
void myDrawing( int levels )
{
    ellipse( 50, 50, 100, 100 );

    if ( levels > 0 ) {
        pushMatrix();
        translate( 150, 0 );
        myDrawing( levels - 1 );
        popMatrix();
    }
}

void setup()
{
    size( 600, 100 );
    background( 100 );
    myDrawing( 6 ); Choose a number of
} levels at the outset.
```

Anatomy of recursion

Every recursive function has three features:

Anatomy of recursion

Every recursive function has three features:

One or more *recursive cases*: places where the function calls itself to solve a sub-problem.

Anatomy of recursion

Every recursive function has three features:

One or more *recursive cases*: places where the function calls itself to solve a sub-problem.

One or more *base cases*: places where the function does something without calling itself.

Anatomy of recursion

Every recursive function has three features:

One or more *recursive cases*: places where the function calls itself to solve a sub-problem.

One or more *base cases*: places where the function does something without calling itself.

Every recursive call *makes progress towards a base case*.

Factorial

$$1! = 1$$

$$2! = 1 \times 2 = 2$$

$$3! = 1 \times 2 \times 3 = 6$$

$$4! = 1 \times 2 \times 3 \times 4 = 24$$

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

...

$$n! = 1 \times 2 \times 3 \times 4 \times \dots \times n$$

Factorial

$$1! = 1$$

$$2! = 1 \times 2 = 2$$

$$3! = 1 \times 2 \times 3 = 6$$

$$4! = 1 \times 2 \times 3 \times 4 = 24$$

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

...

$$n! = n \times (n-1)!$$

Factorial

$$1! = 1$$

$$2! = 1 \times 2 = 2$$

$$3! = 1 \times 2 \times 3 = 6$$

$$4! = 1 \times 2 \times 3 \times 4 = 24$$

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

...

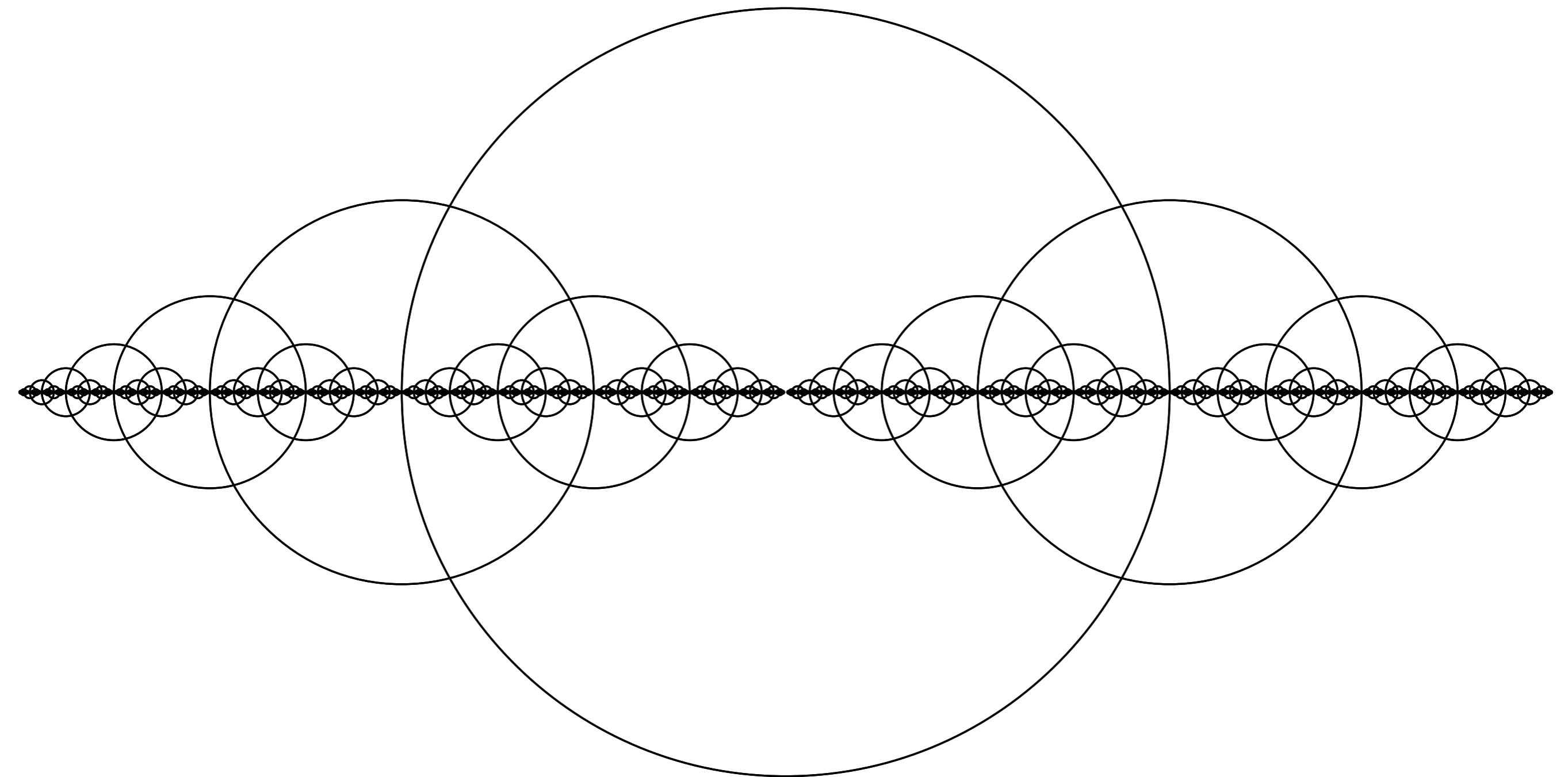
$$n! = n \times (n-1)!$$

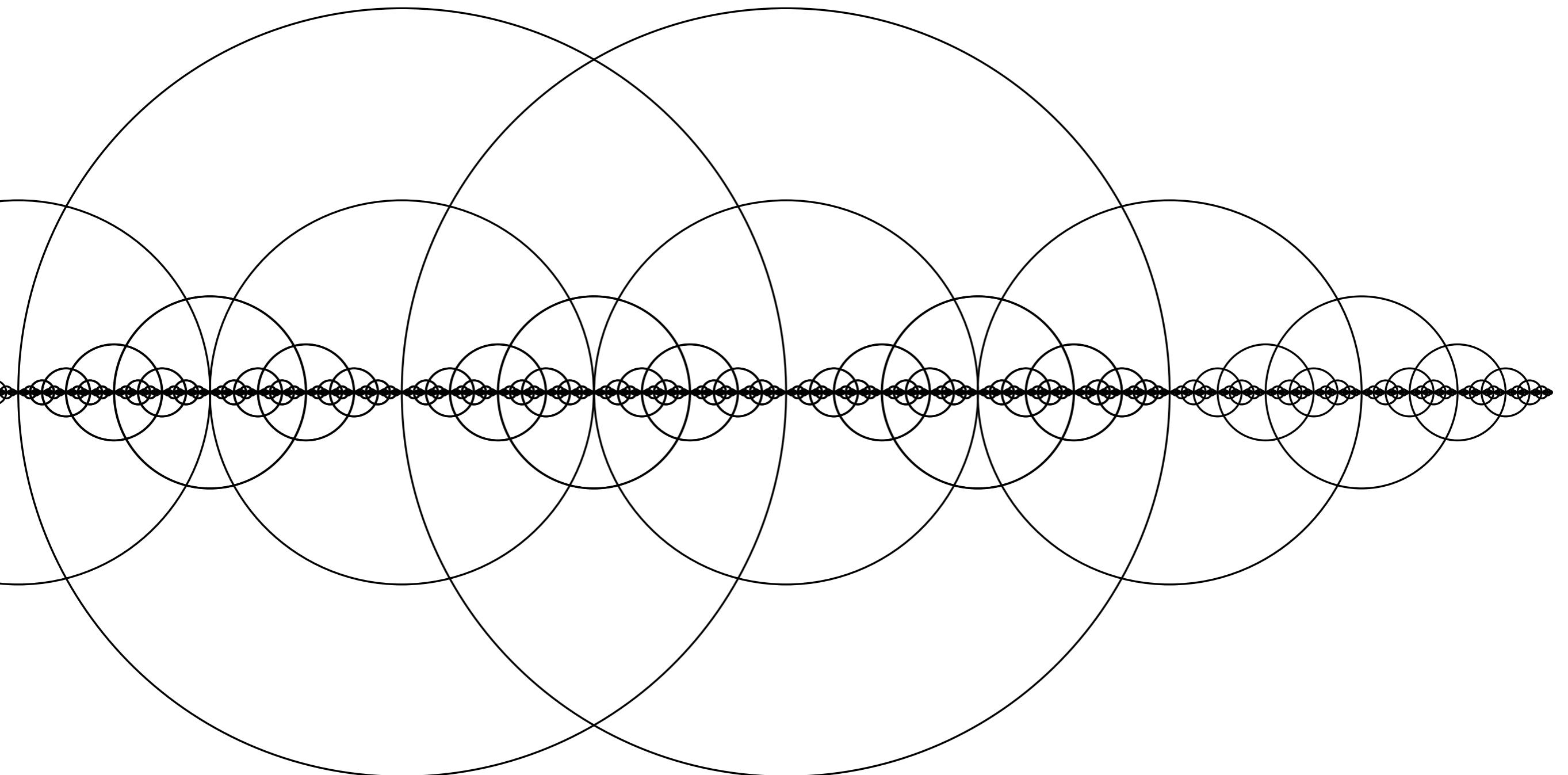
```
int fact( int n )
{
    return n * fact( n - 1 );
}
```

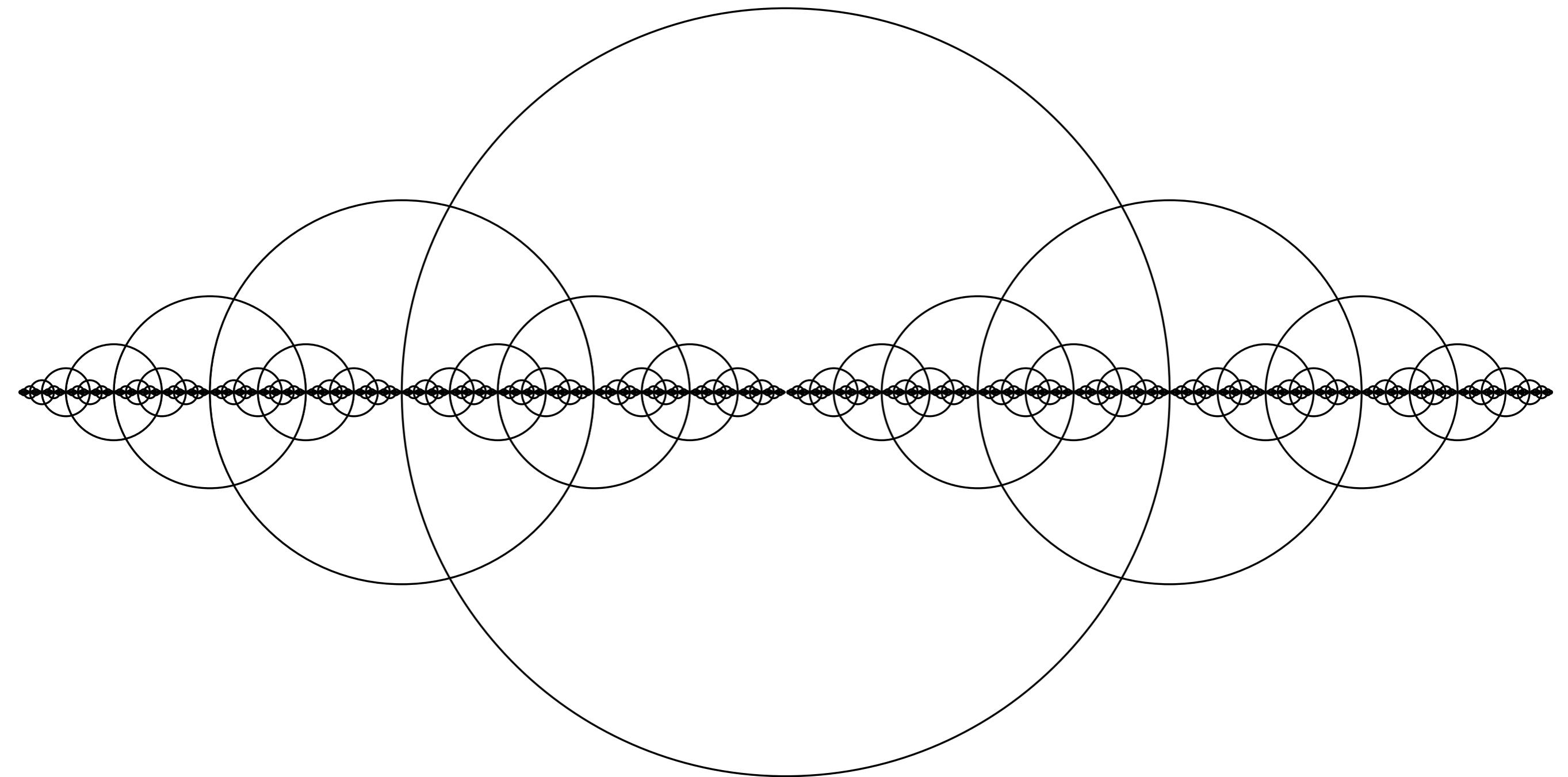
Factorial

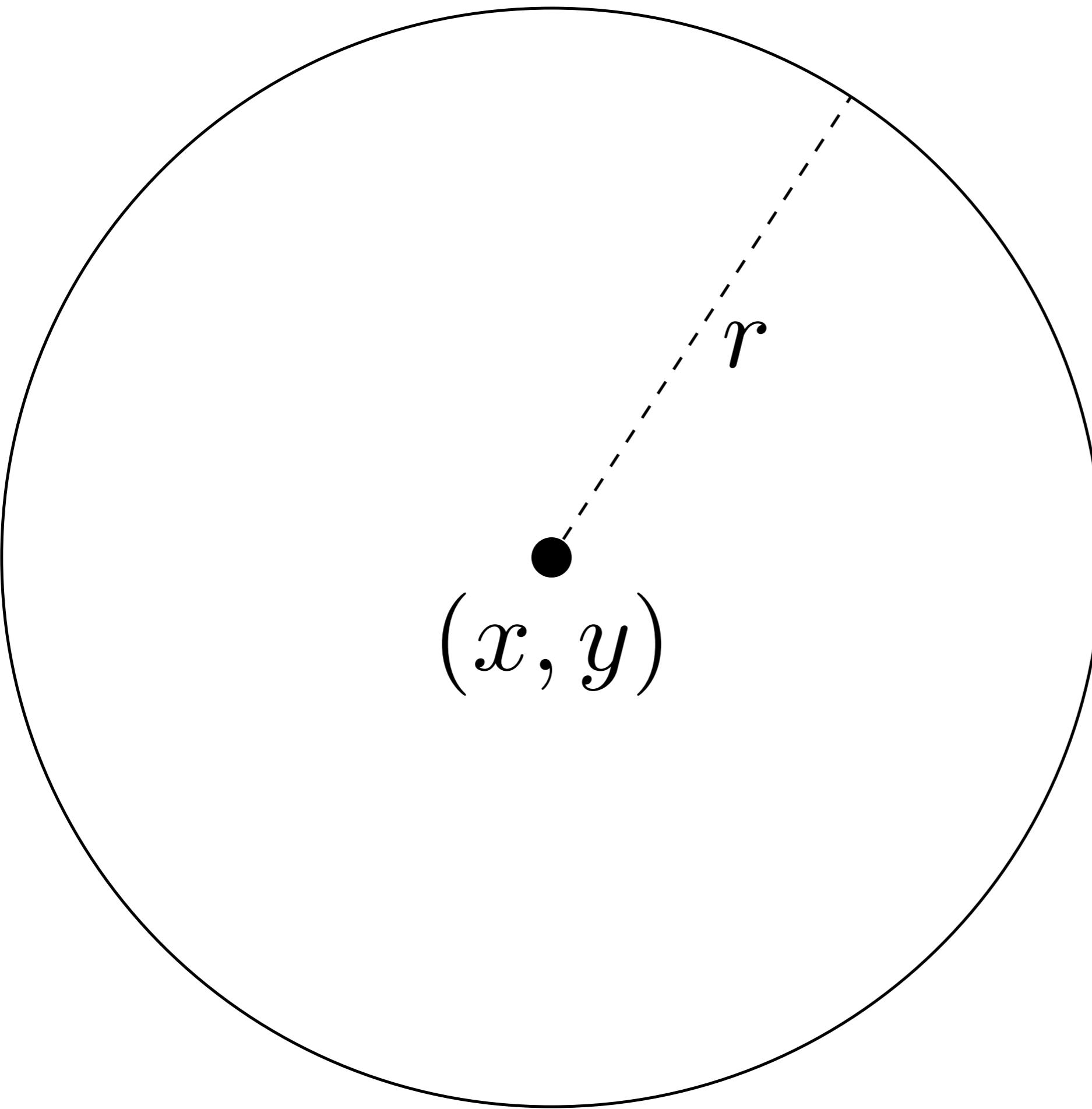
$$n! = n \times (n-1)!$$

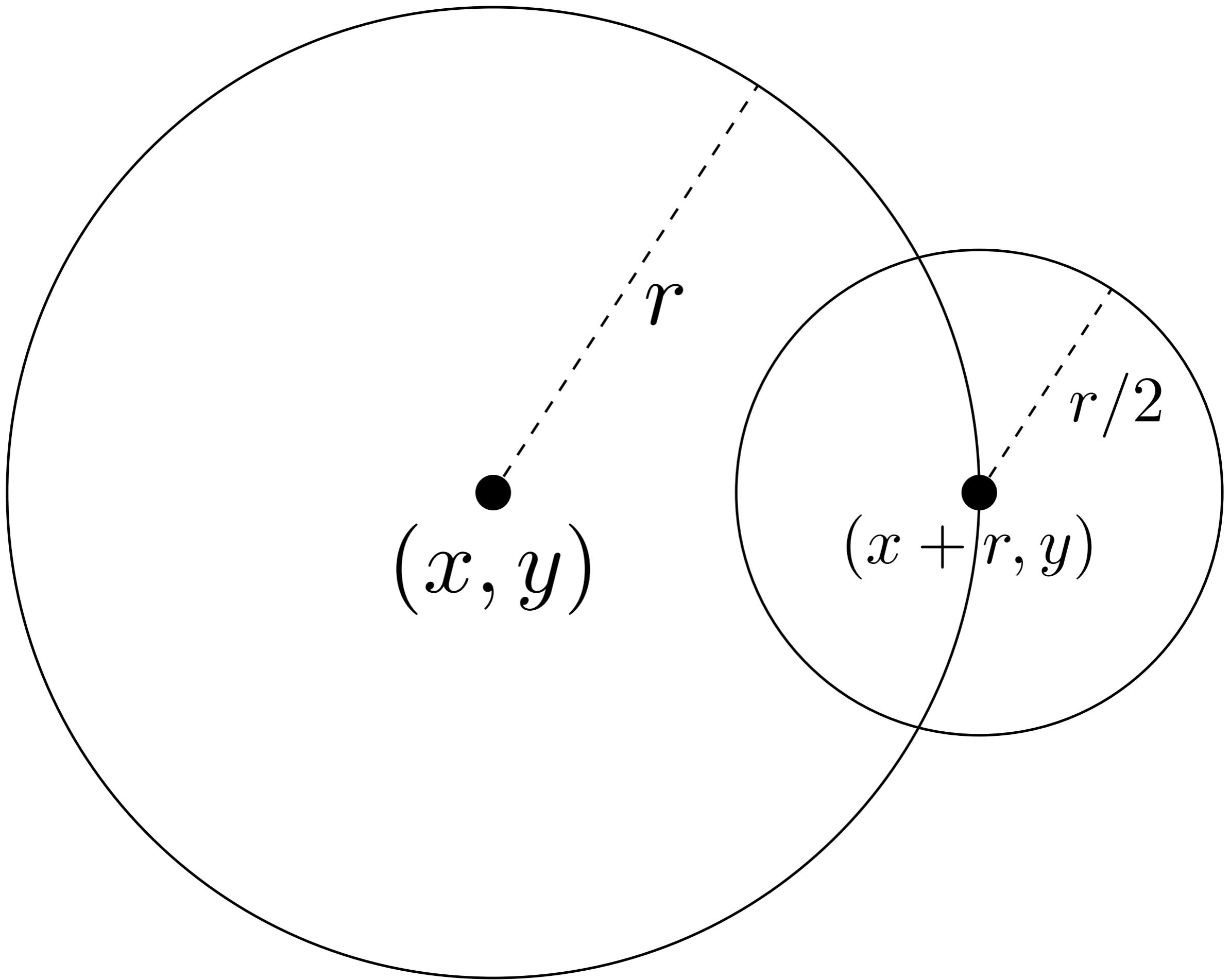
```
int fact( int n )
{
    if ( n == 1 ) {
        return 1;
    } else {
        return n * fact( n - 1 );
    }
}
```

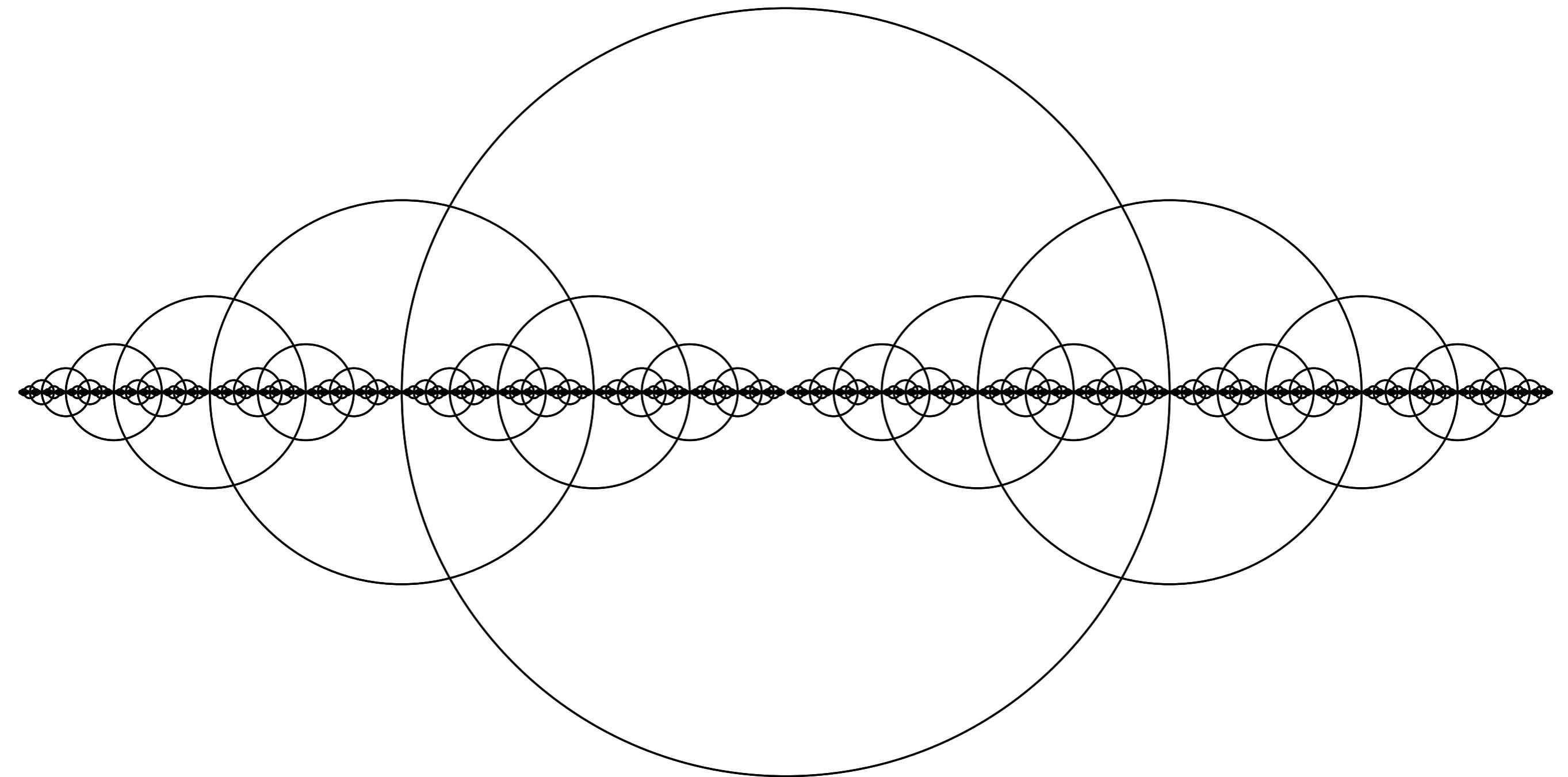


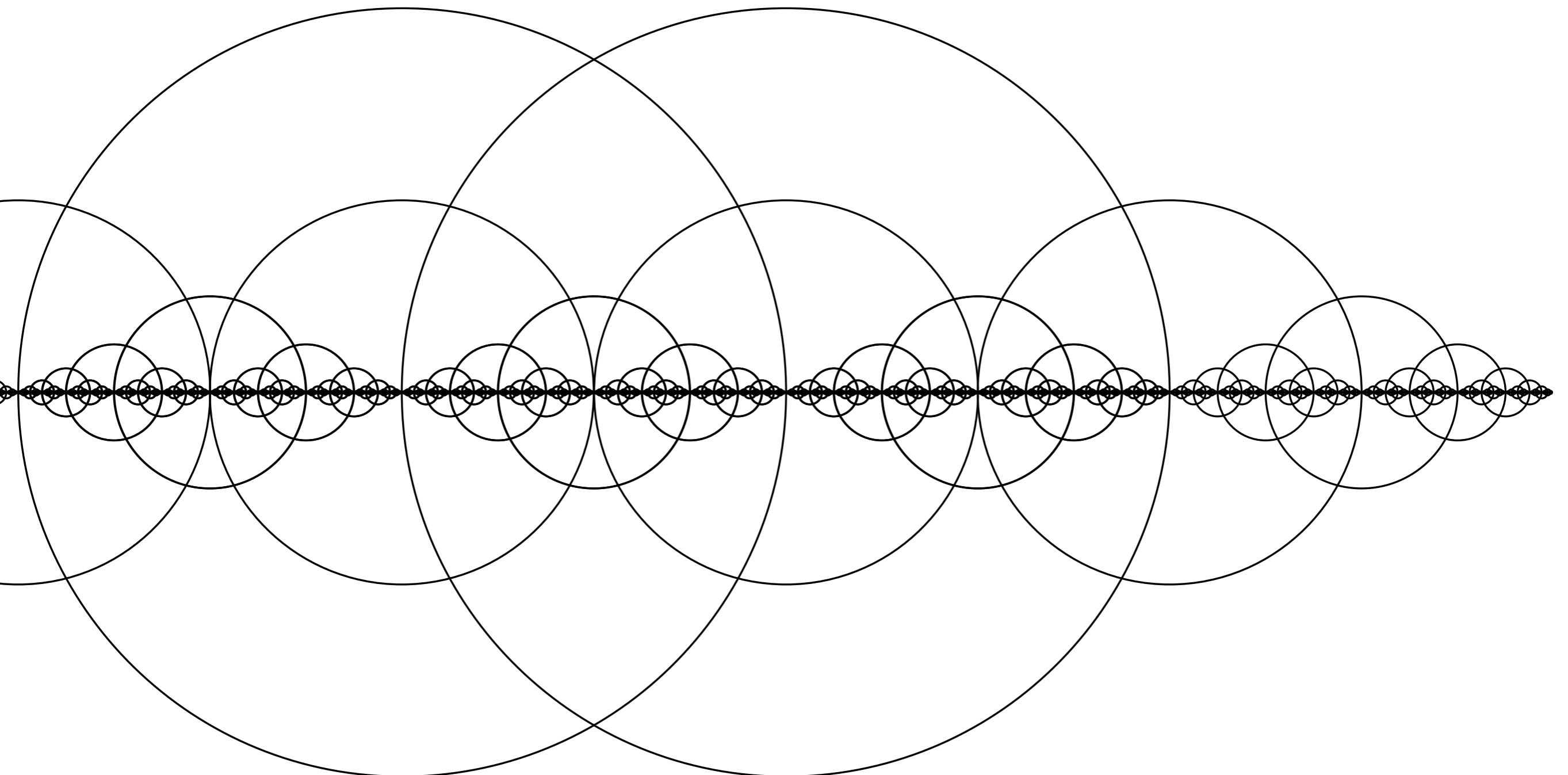


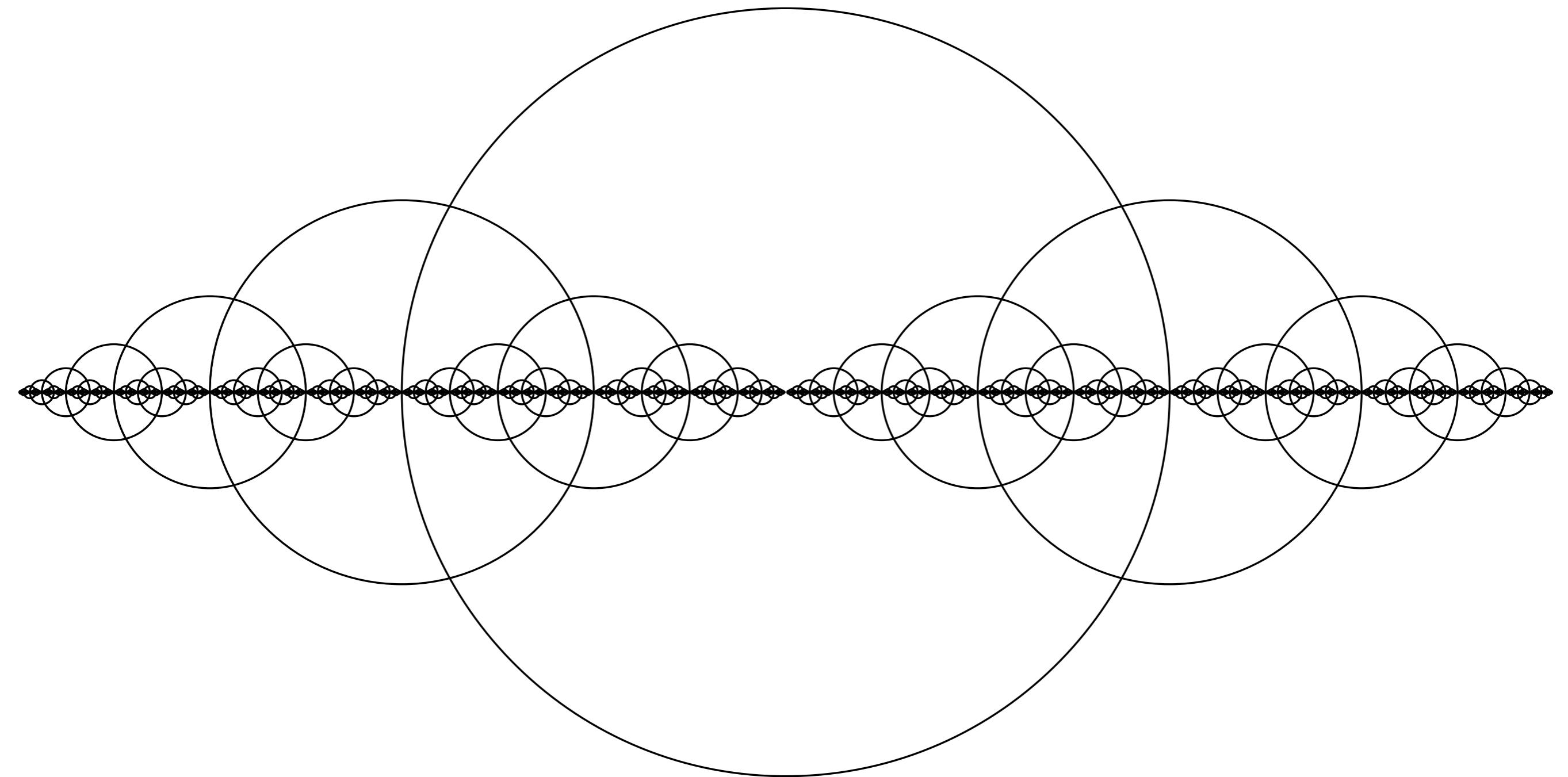


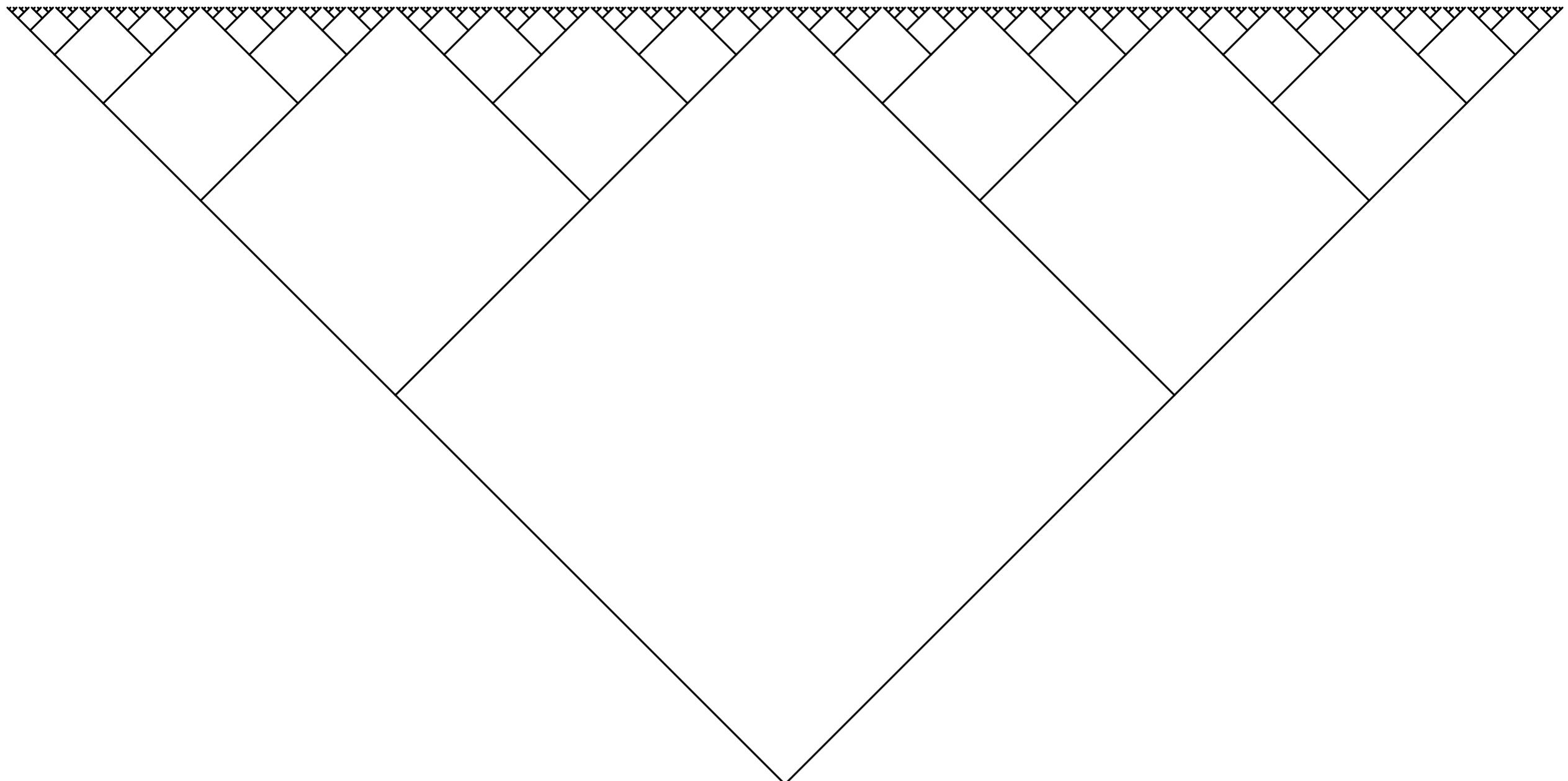




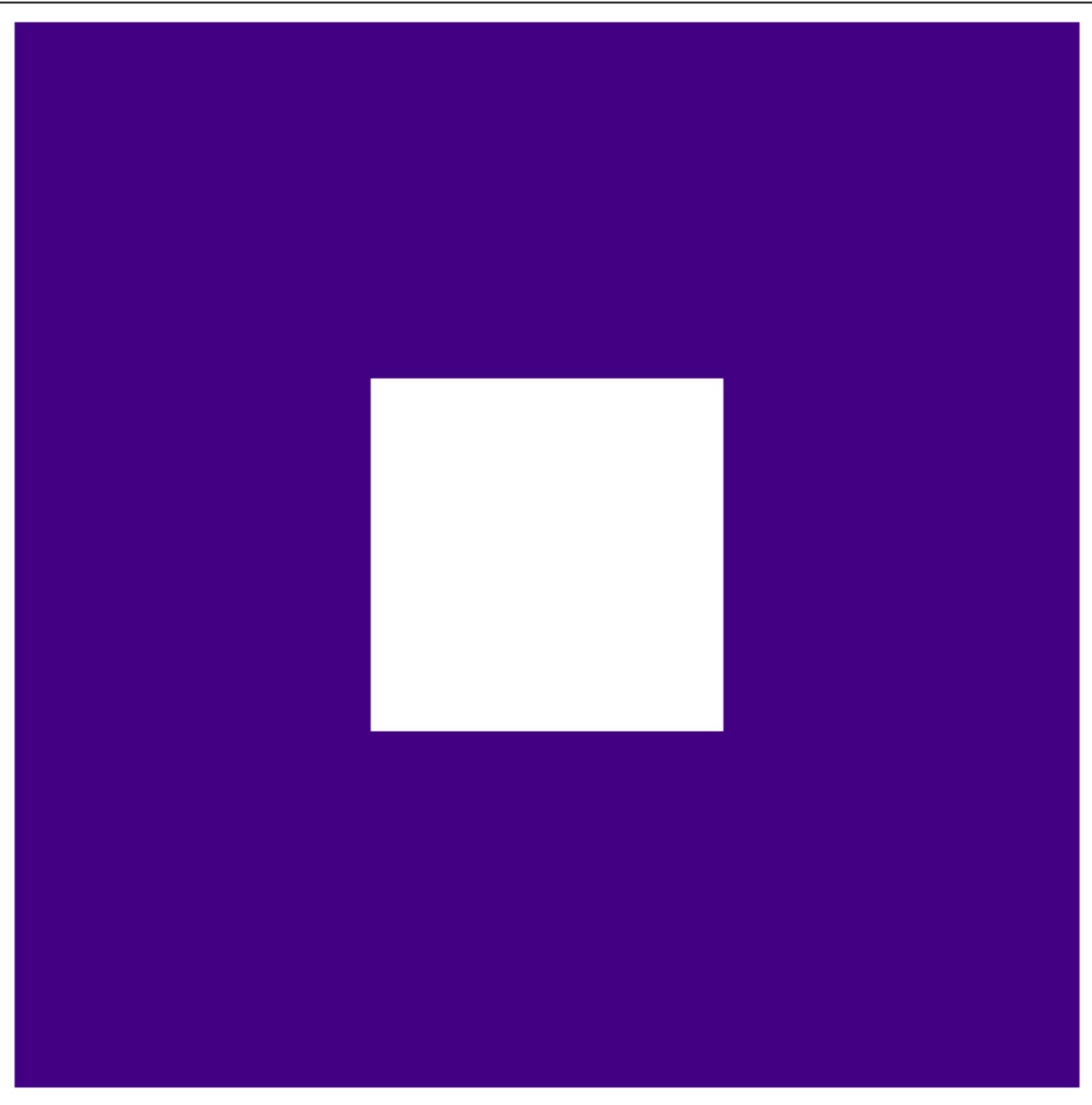




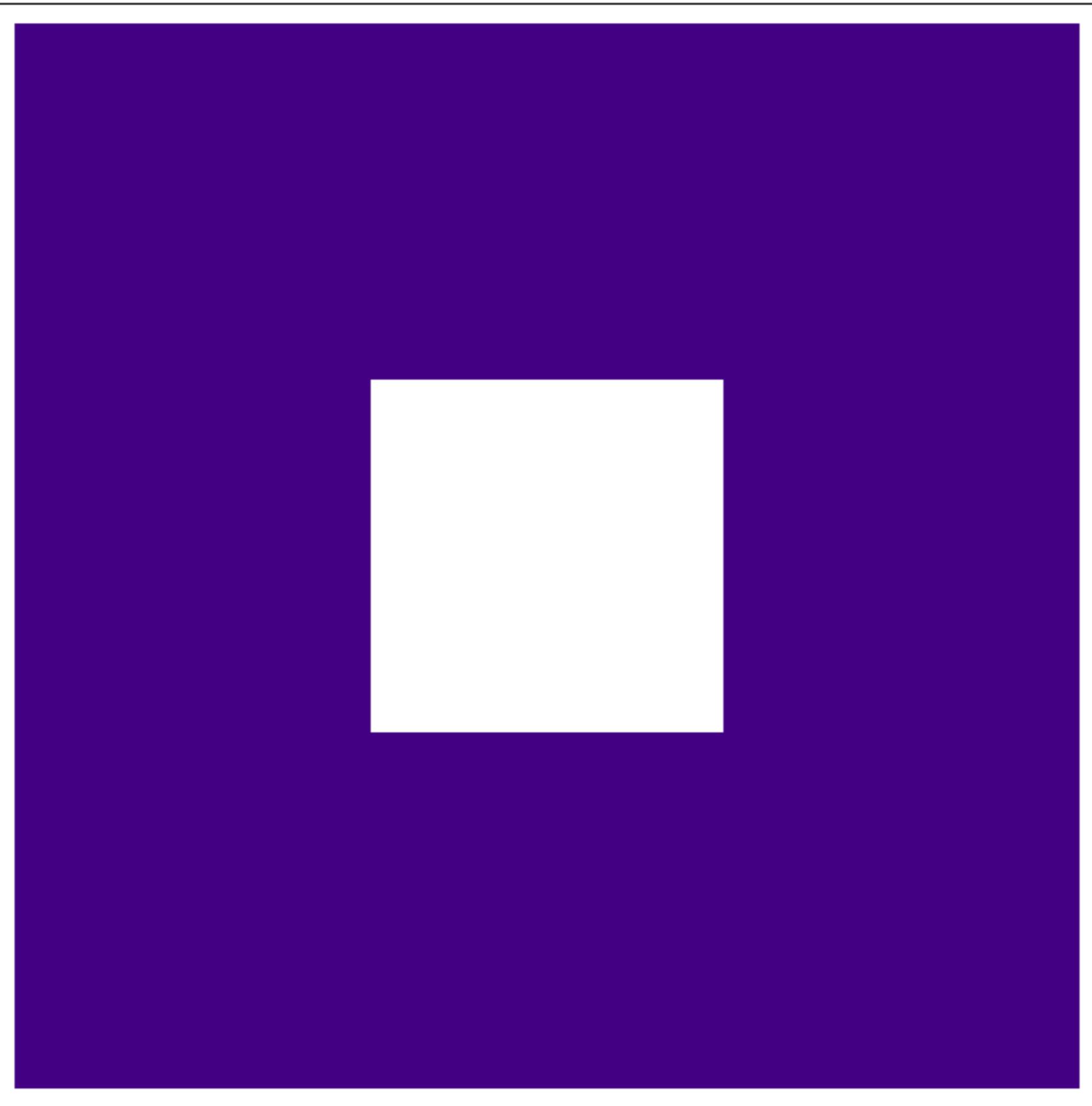




Sierpinski Carpet



Sierpinski Carpet



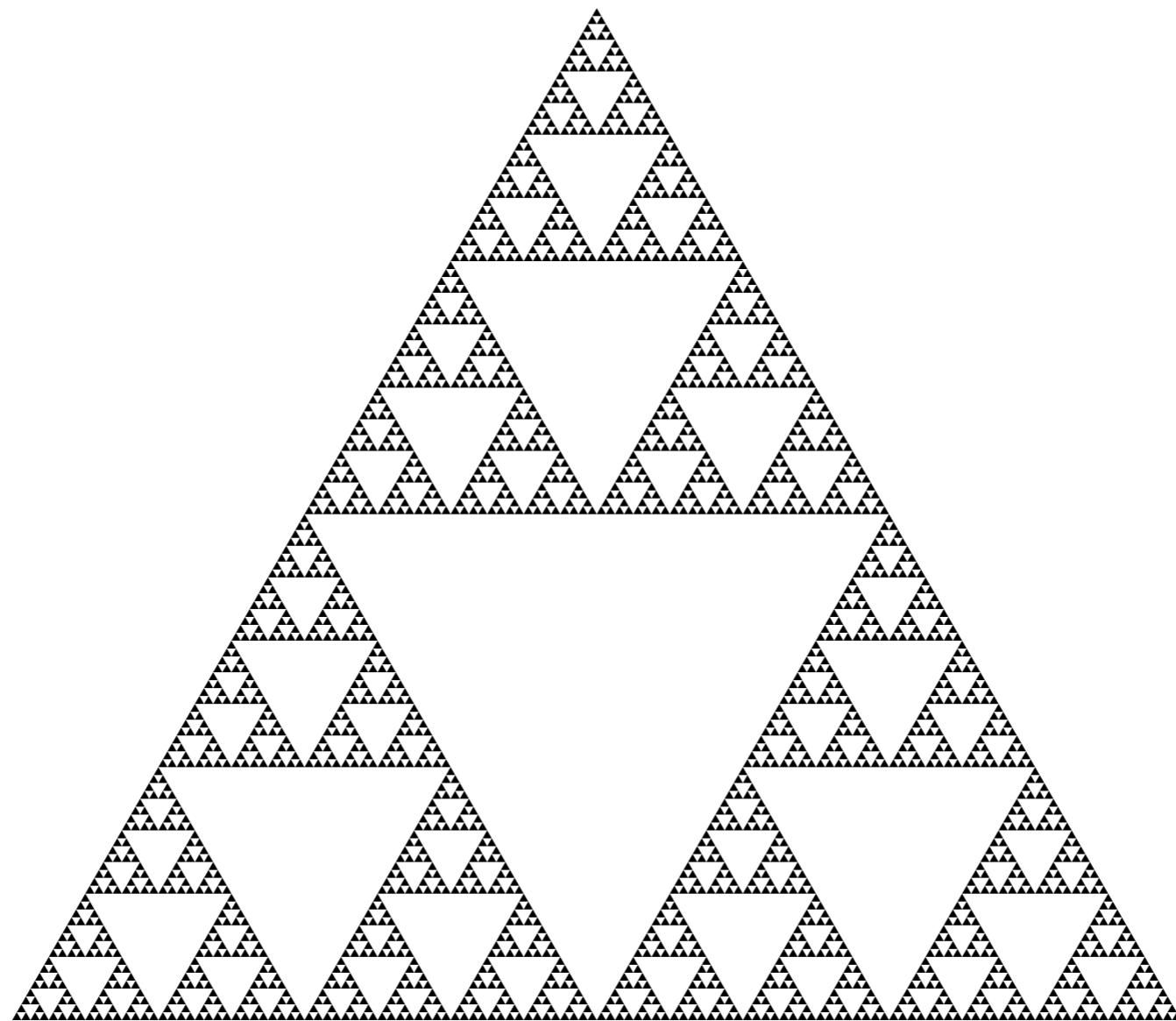
How to draw the carpet:

1. Divide a square into a 3×3 grid.
2. Draw the carpet in each cell of the grid, except for the central one.

How to draw the carpet at level n :

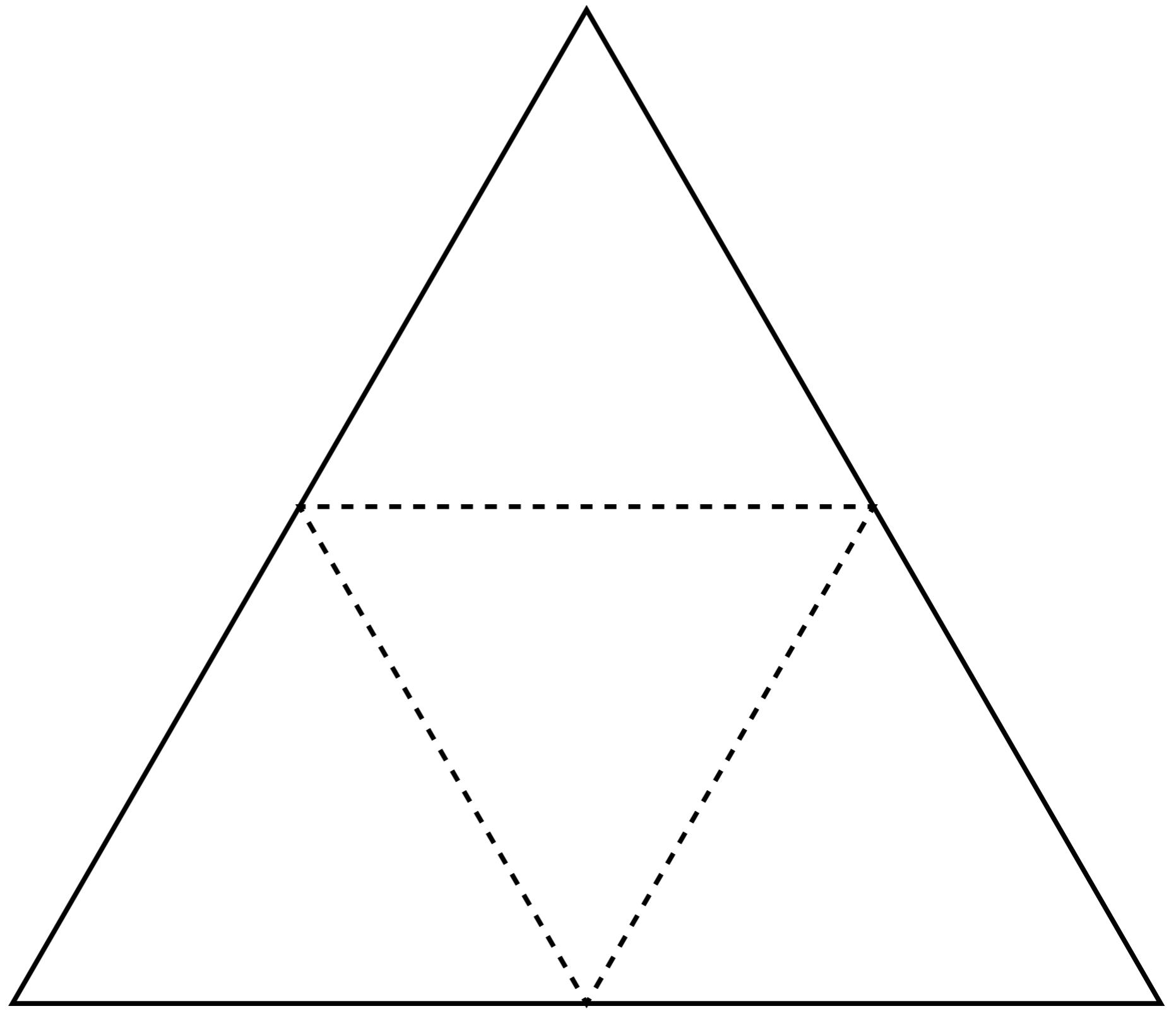
1. If n is 0 then just draw a square.
2. Otherwise:
 1. Divide a square into a 3×3 grid.
 2. Draw the carpet at level $(n - 1)$ in each cell of the grid, except for the central one.

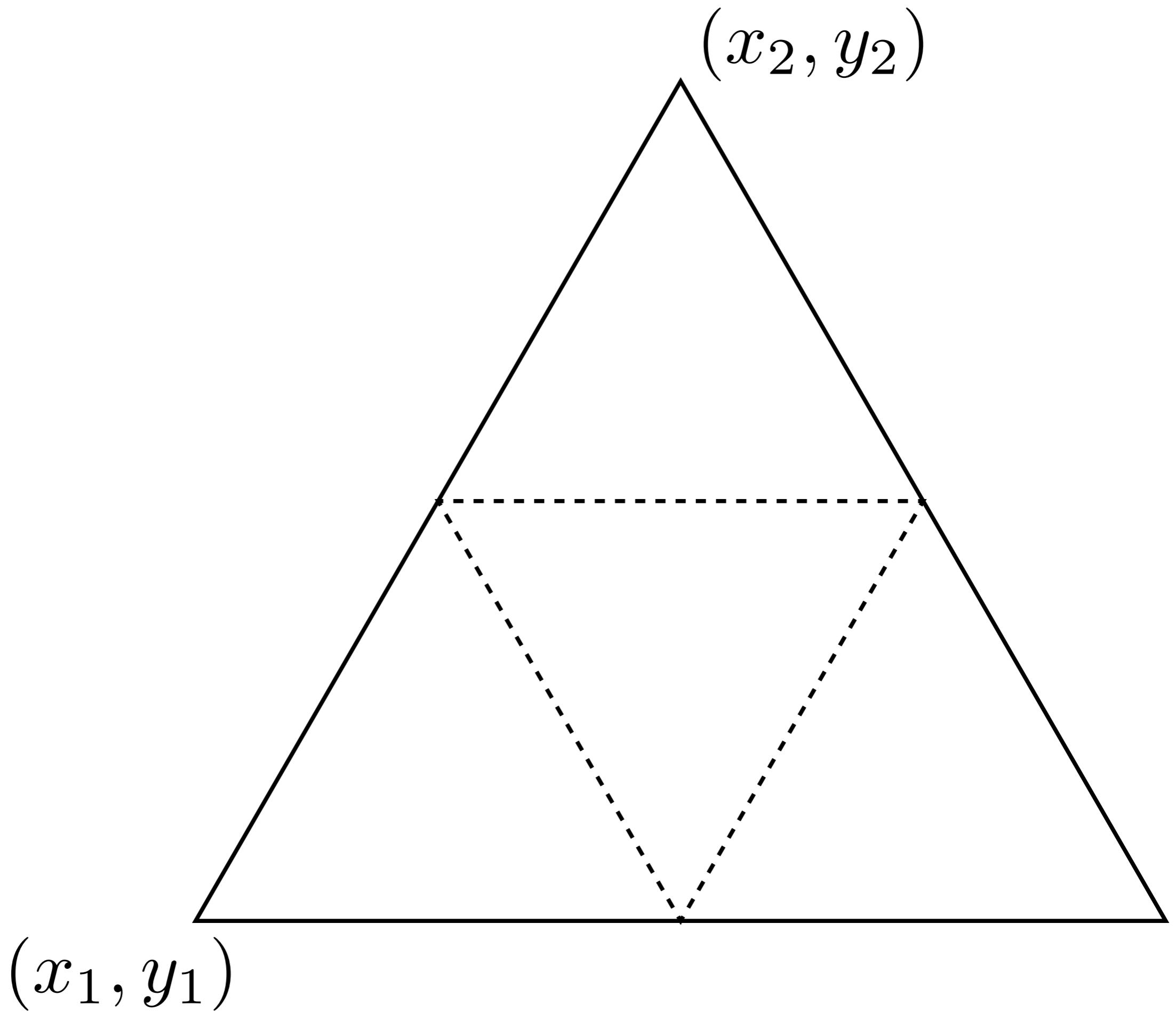
Sierpinski Gasket

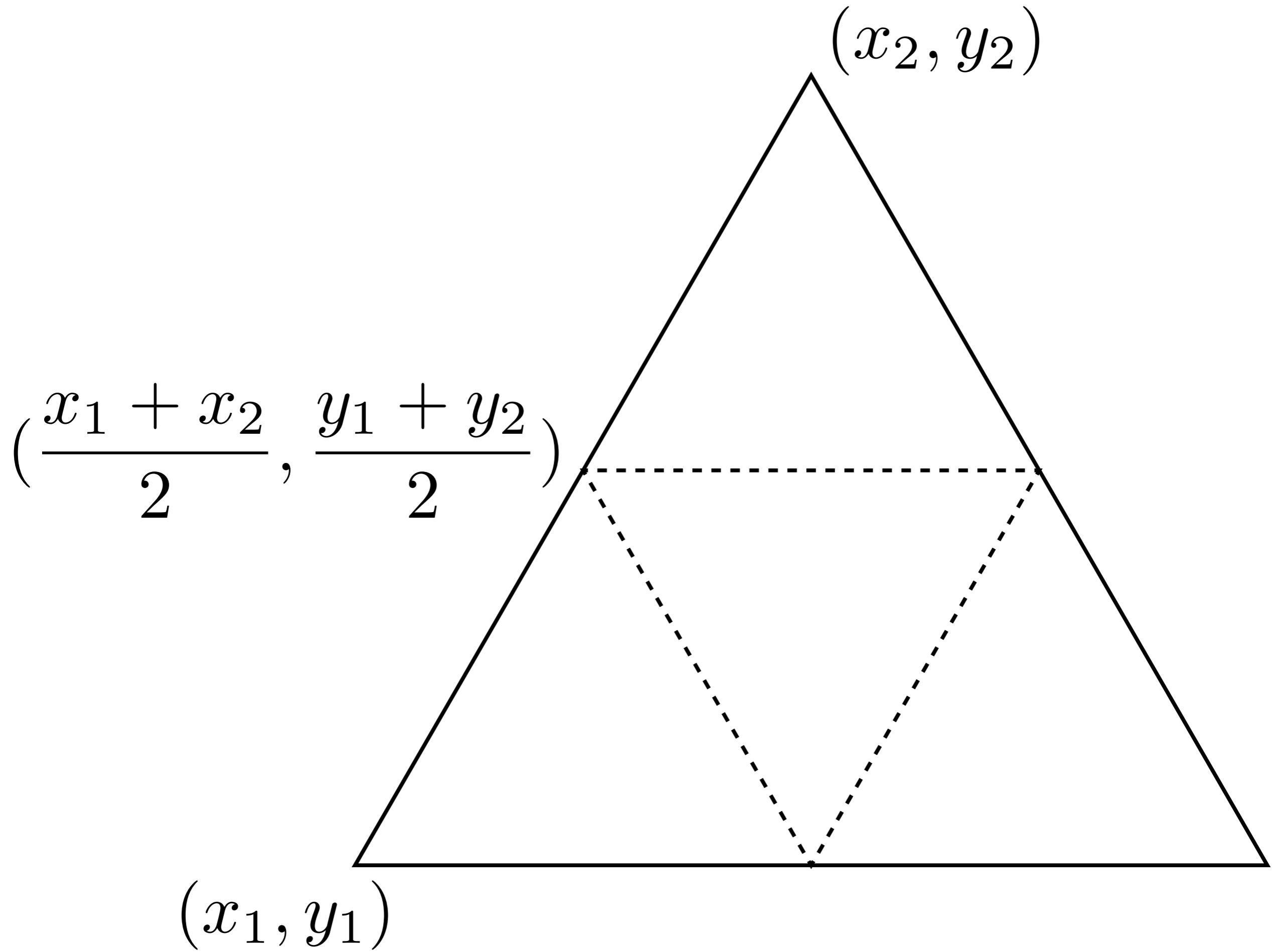


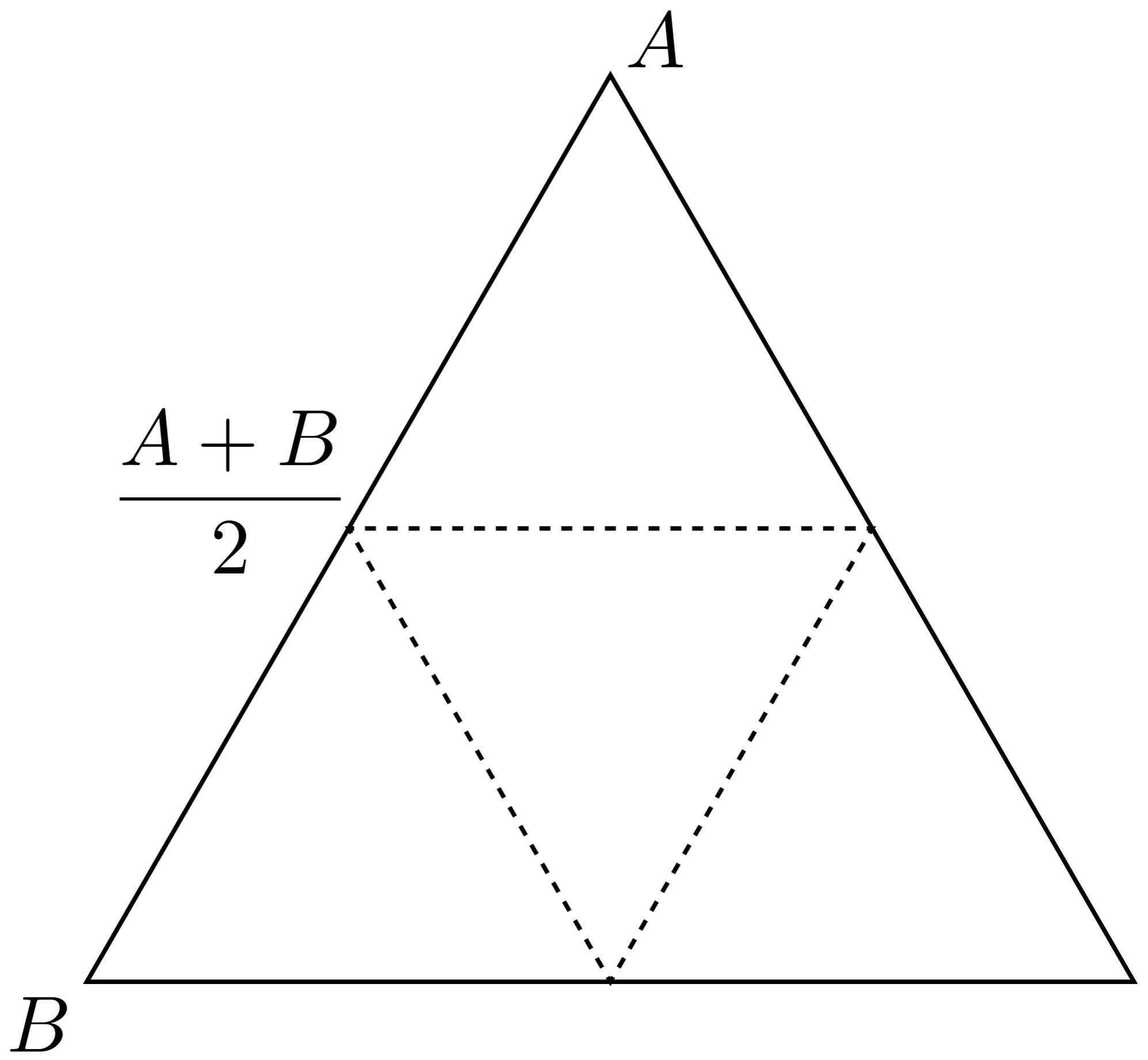
How to draw the gasket at level n :

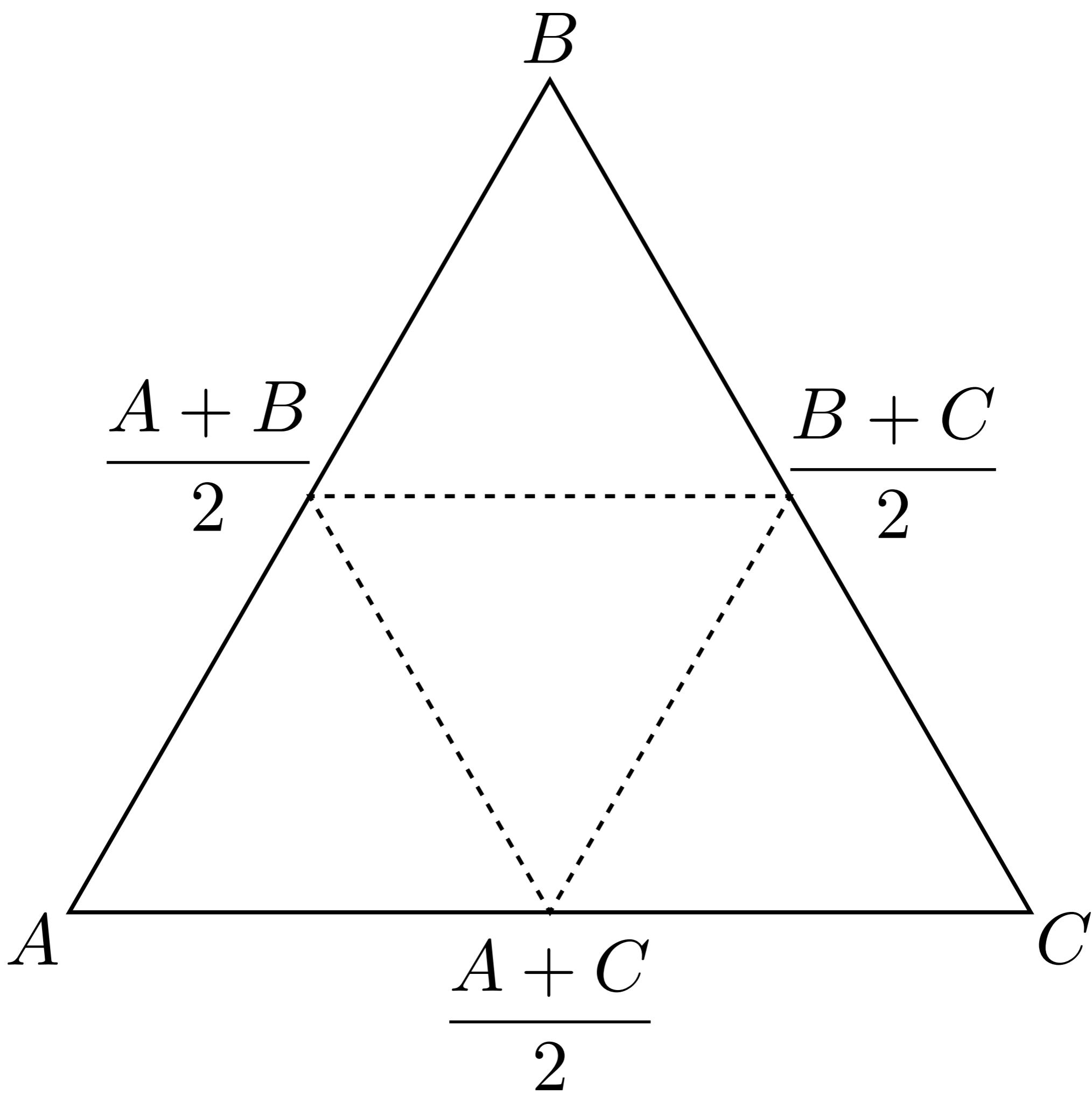
1. If n is 0 then just draw a triangle.
2. Otherwise:
 1. Divide a triangle into four.
 2. Draw the gasket at level $(n - 1)$ in each sub-triangle, except for the central one.

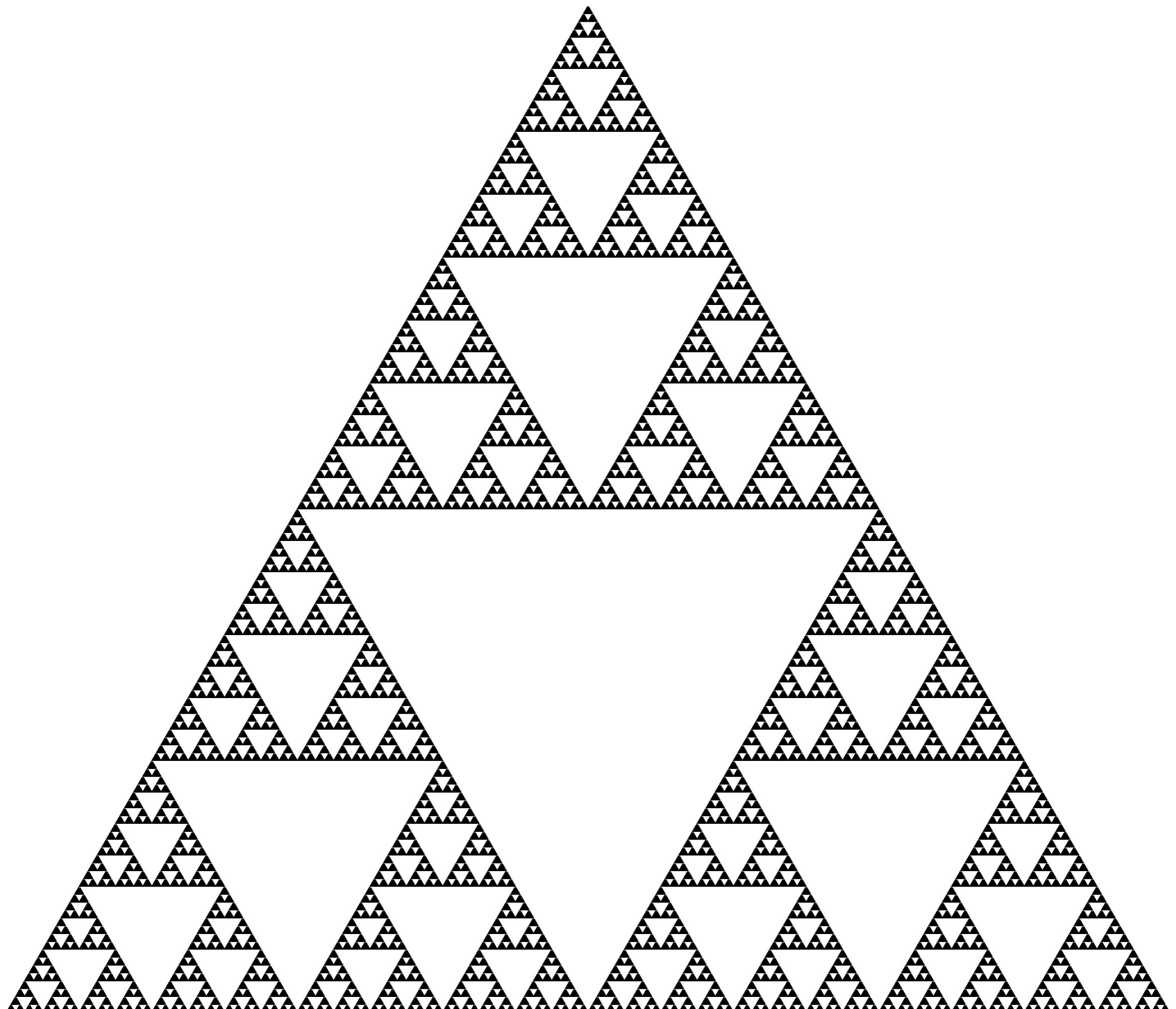


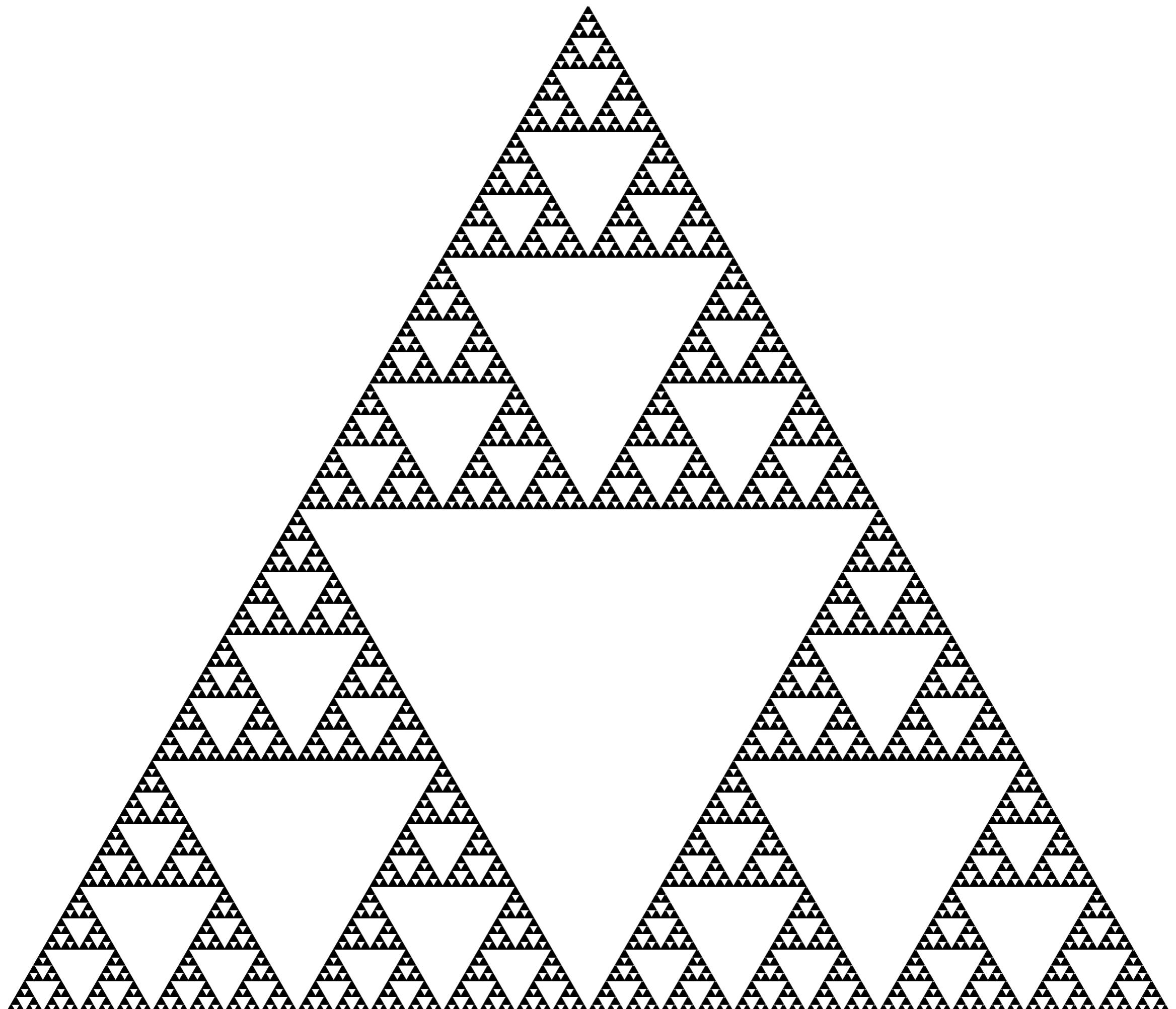


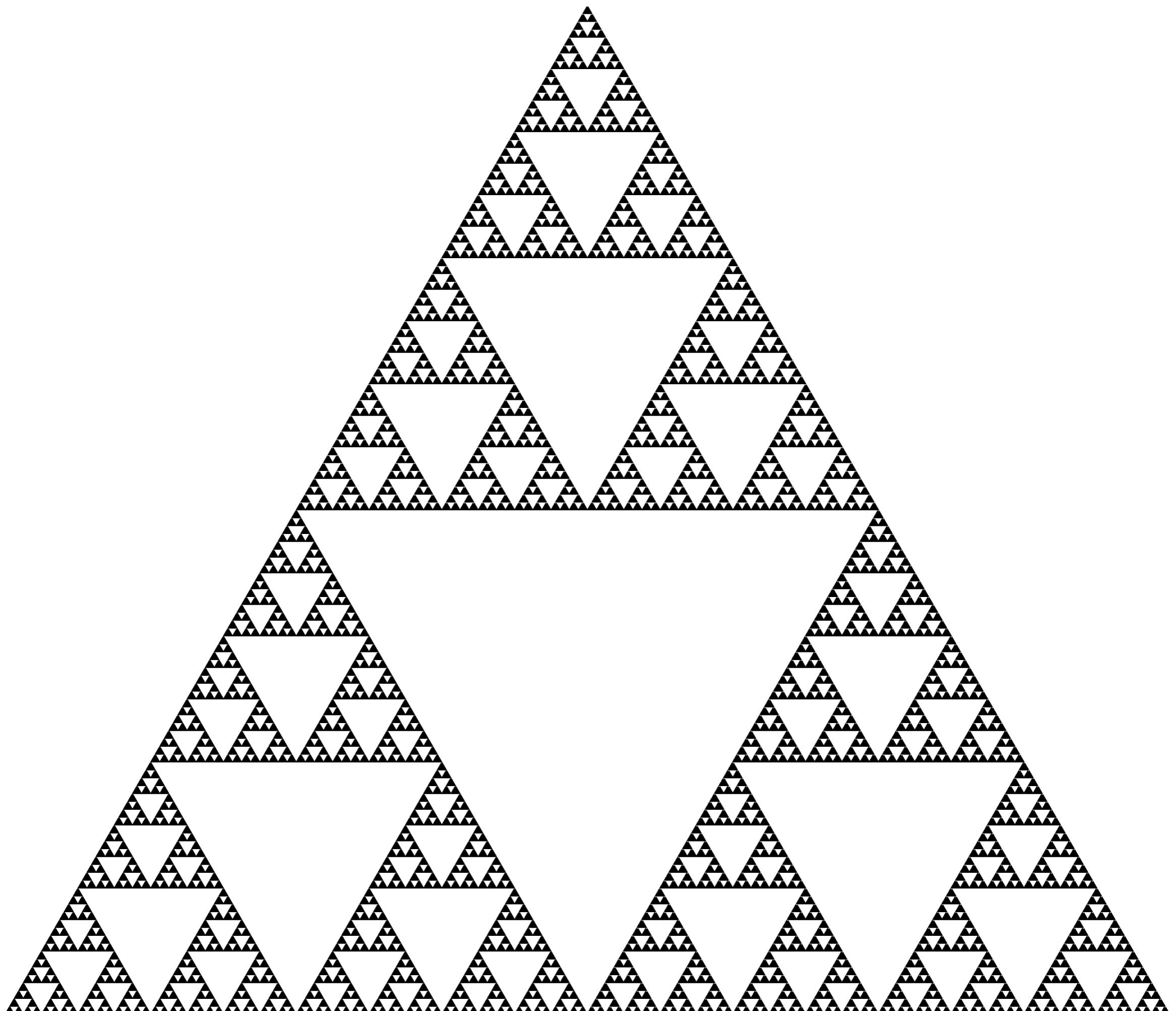


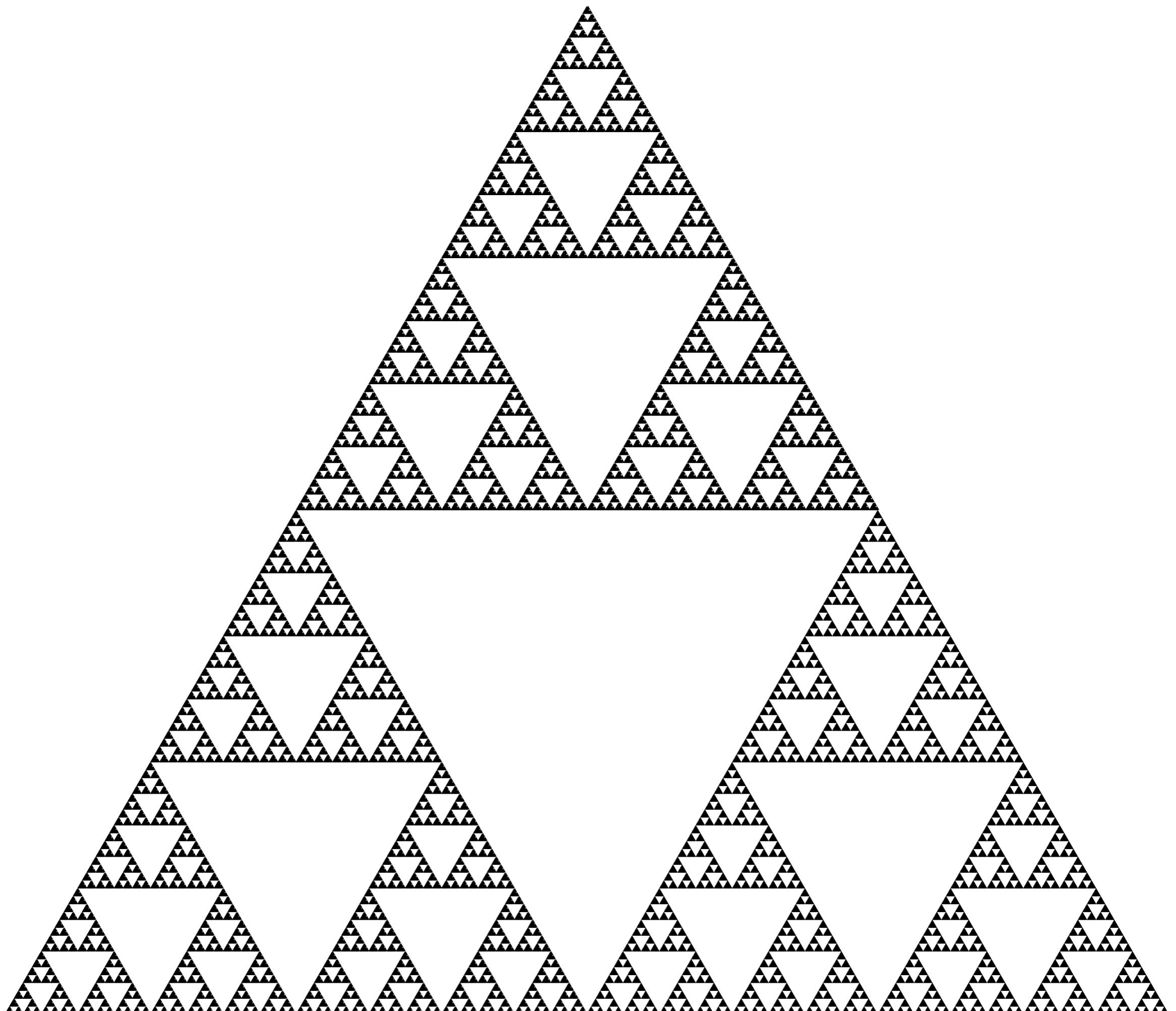












Sierpinskitaschen



Debs Gardner [seattlelocalfood.com]

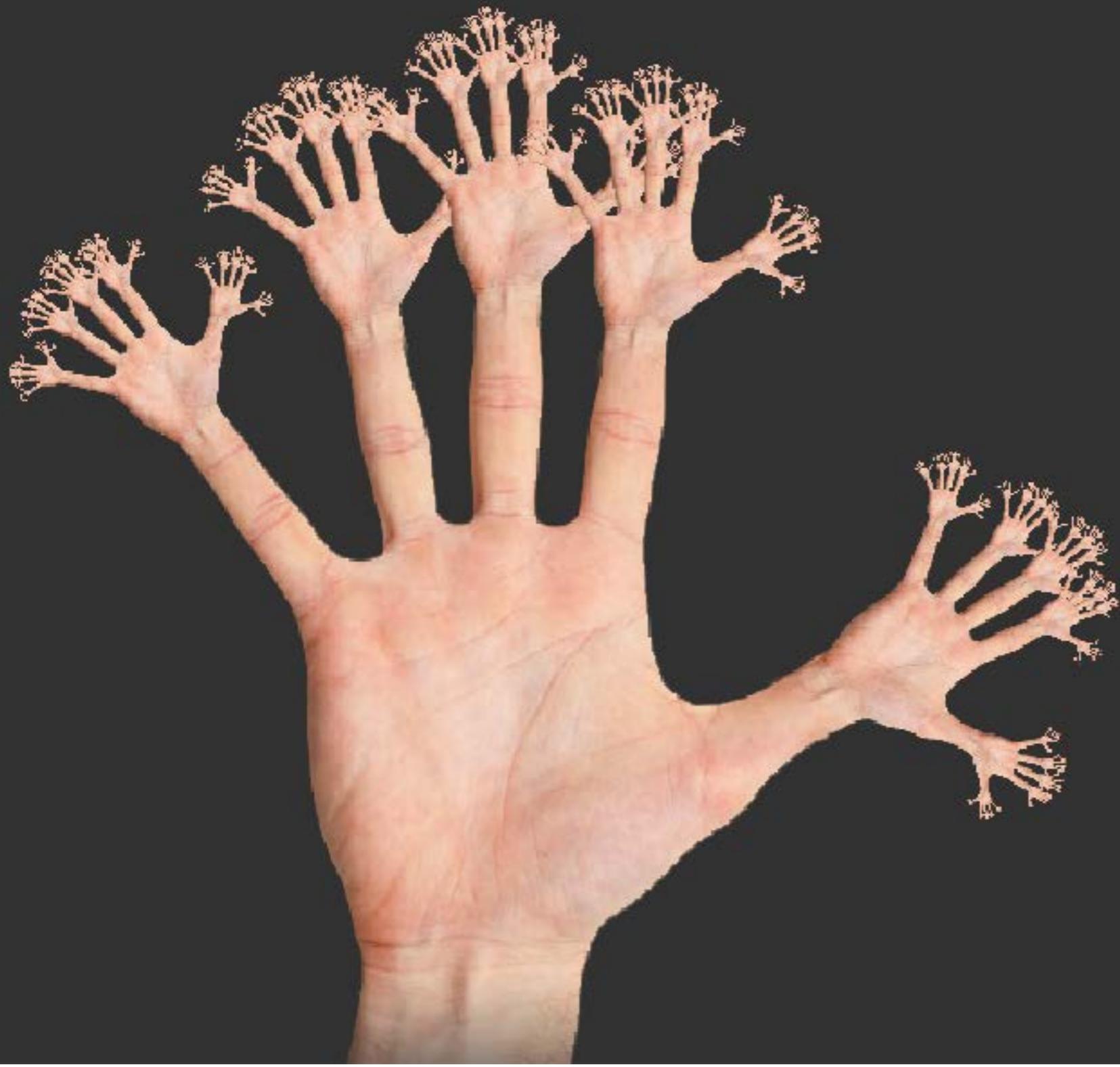


NEW COPY

SHOW OUTLINES

4

LEVELS





Museum of Mathematics, New York City