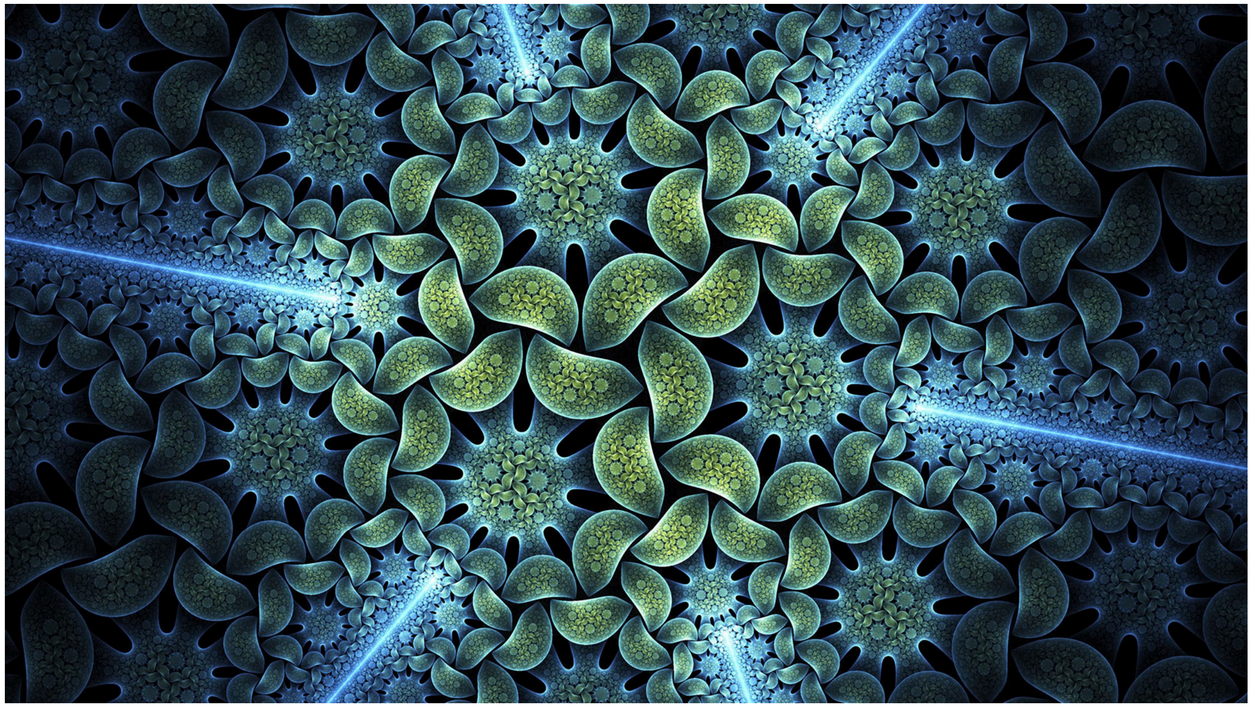
Thinking Recursively

CS 106B

Programming Abstractions
Fall 2016
Stanford University
Computer Science Department



Announcements

- Assignment 2 due Saturday
- Remember no LaIR on Friday night
- New texts for random writer

Today's Goal

1. More practice with recursion
2. Be able to write graphical recursive functions

One of the programs on assignment 3 will use graphical recursion

Recursion Definition

Recursion:

A problem solving technique in which problems are solved by reducing them into **smaller problems** ***of the same form***.

Recursion Template

The structure of recursive functions is typically like the following:

```
recursiveFunction:
   if (test for simple case) {
      Compute the solution without recursion
   } else {
      Break the problem into subproblems of the same form
      Call recursiveFunction on each subproblem
      Reassemble the results of the subproblems
   }
}
```

Three Musts of Recursion

1. Your code must have a case for all valid inputs.

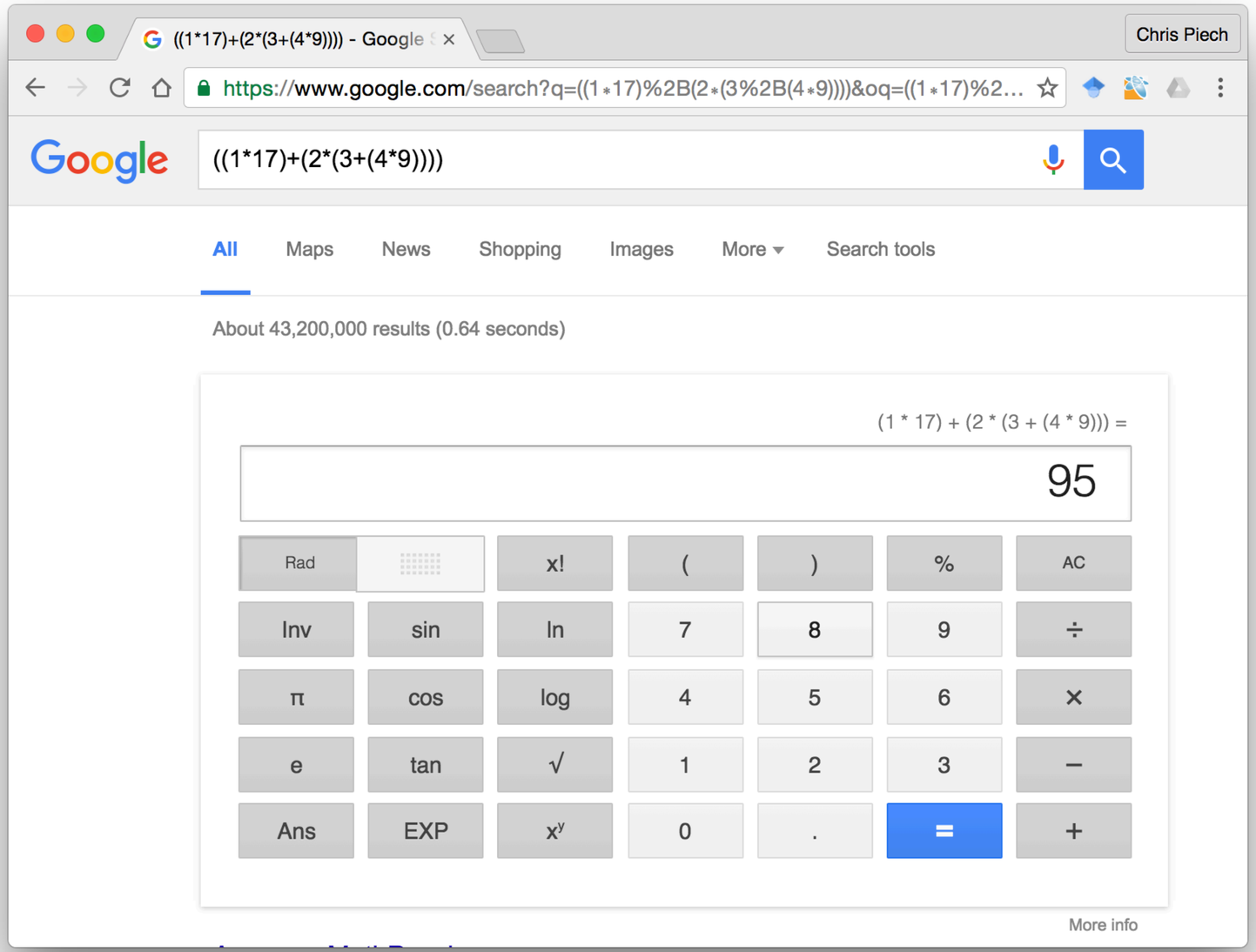
2. You must have a base case (makes no recursive calls).

3. When you make a recursive call it should be to a simpler instance (forward progress towards base case)

Start with Delicious Example

((1+3)*(2*(4+1)))

Google Search I'm Feeling Lucky


((1*17)+(2*(3+(4*9)))) - Google
Chris Piech
https://www.google.com/search?q=((1*17)%2B(2*(3%2B(4*9))))&oq=((1*17)%2...
Google
((1*17)+(2*(3+(4*9))))
All
Maps
News
Shopping
Images
More
Search tools
About 43,200,000 results (0.64 seconds)
(1 * 17) + (2 * (3 + (4 * 9))) =
95
Rad
x!
(
)
%
AC
Inv
sin
ln
7
8
9
÷
π
cos
log
4
5
6
×
e
tan
√
1
2
3
−
Ans
EXP
x^y
0
.
=
+
More info

“((1+3)*(2*(4+1)))”

95


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Challenge

Implement a function which evaluates an expression string:

“((1+3)*(2*(4+1)))”

Only * or +

“(7+6)”

“(((4*(1+2))+6)*7)”

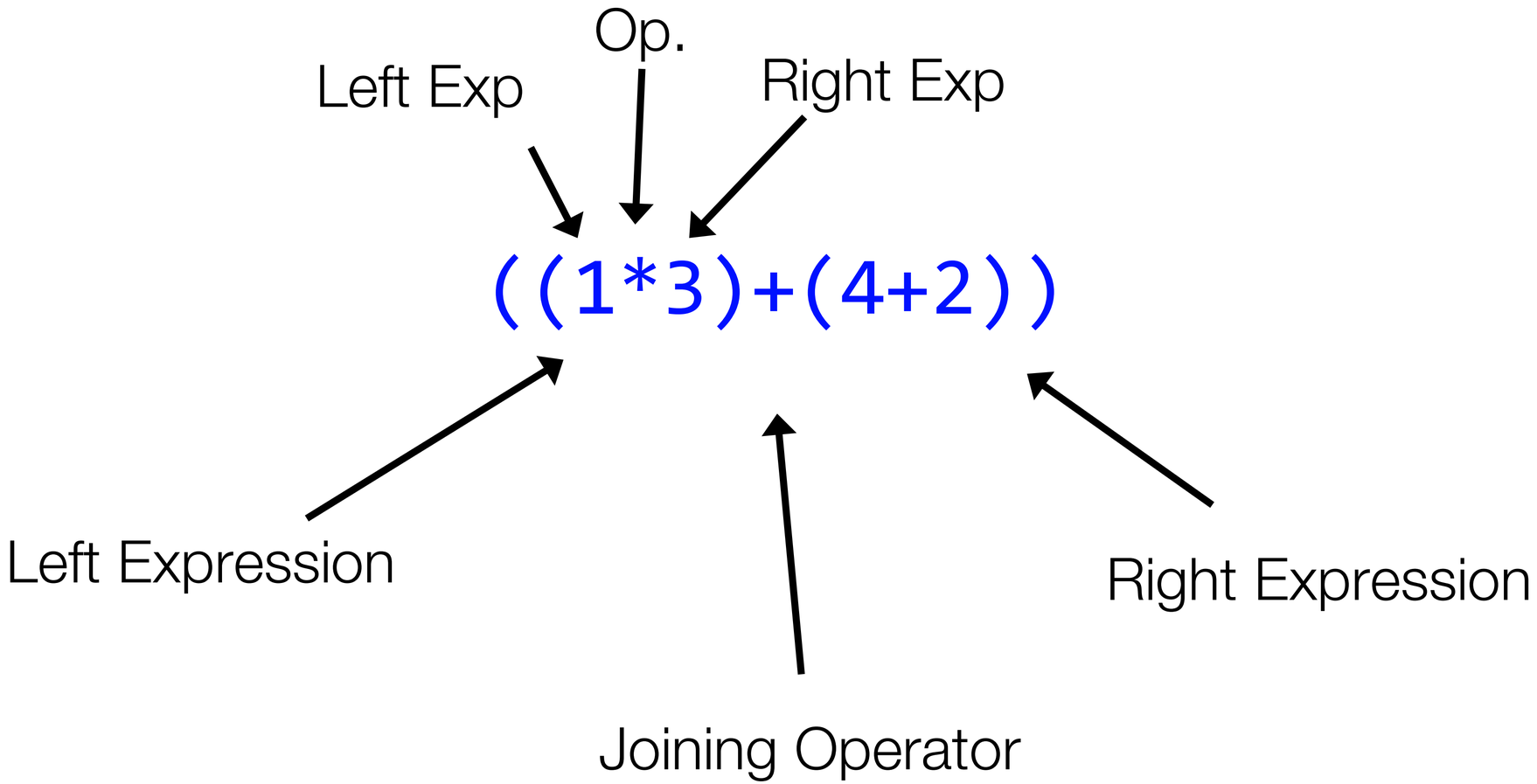
Anatomy of an Expression

An expression is always one of these three things

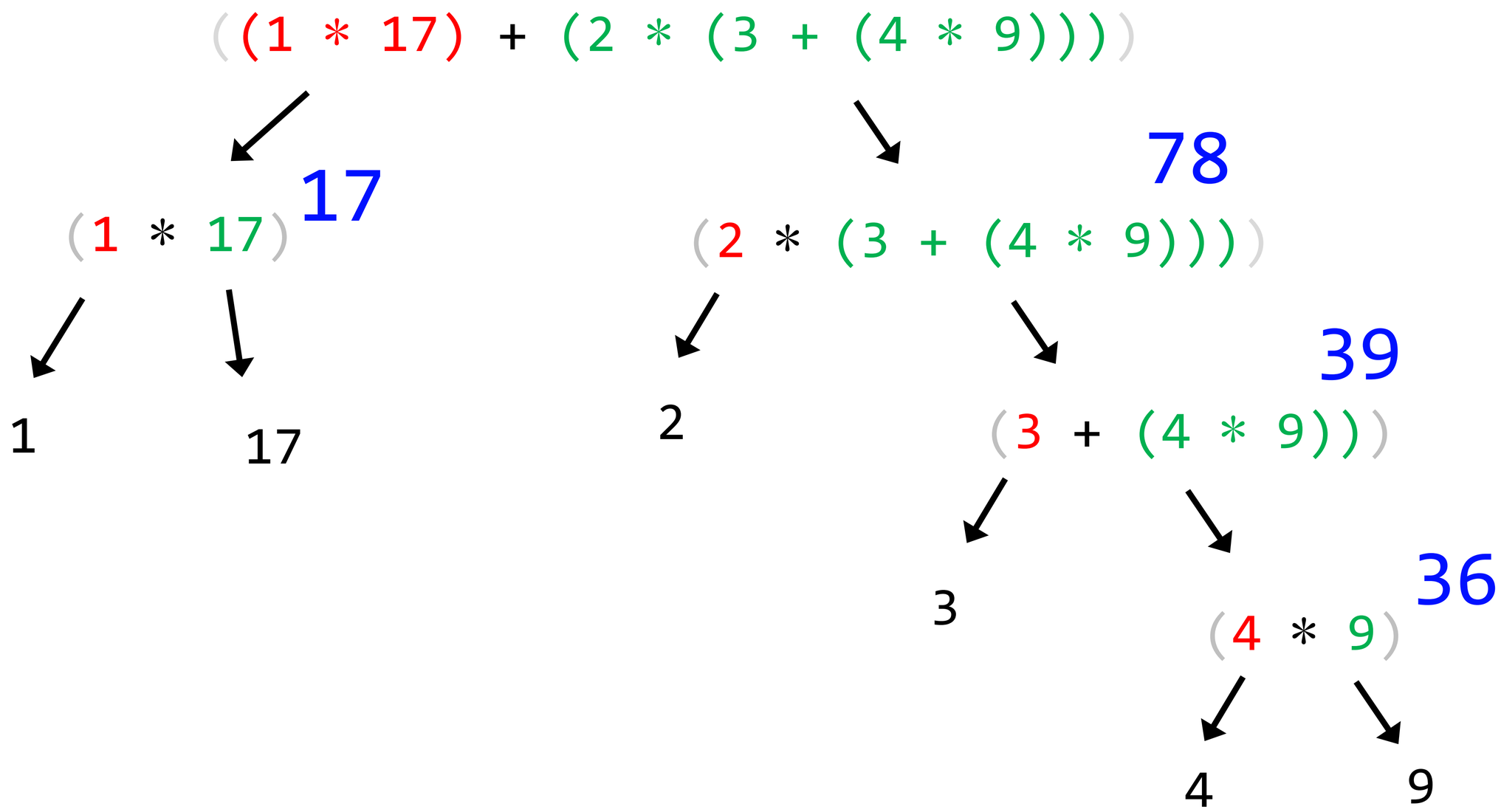
expression

- *number*
- *(expression + expression)*
- *(expression * expression)*

Anatomy of an Expression



How do we evaluate ((1*17)+(2*(3+(4*9))))? 95



It is Recursive?

```
((1*3)+(4+2))
```

The big instance of this problem is:

```
((1*3)+(4+2))
```

The smaller instances are:

```
(1*3)
```
and
```
(4+2)
```

Task

Write this function

```
int evaluate(string exp);
“((1*3)+(4+2))” //returns 11
```

Using these library functions

```
stringIsInteger(exp)
stringToInteger(exp)
```

And these exp helper functions

```
//returns ‘+’
char op = getOperator(exp);
//returns “(1*3)”
string left = getLeftExp(exp);
//returns “(4+2)”
string right = getRightExp(exp);
```

Solution

```
((1*3)+(4+2))
```

`int evaluate(expression):`

- If *expression* is a number, return *expression*
- Otherwise, break up *expression* by its *operator*:
  - *leftResult* = `evaluate(`*leftExpression*`)`
  - *rightResult* = `evaluate(`*rightExpression*`)`
  - `Return` *leftResult operator rightResult*

Solution

```
int evaluate(string exp) {
    if (stringIsInteger(expression)) {
        return stringToInteger(expression);
    } else {
        char op = getOperator(exp);
        string left = getLeftExp(exp);
        string right = getRightExp(exp);
        int leftResult = evaluate(left);
        int rightResult = evaluate(right);

        if (op == '+') {
            return leftResult + rightResult;
        } else if (op == '*') {
            return leftResult * rightResult;
        }
    }
}
```

Solution

```
int evaluate(string exp) {
    if (stringIsInteger(expression)) {
        return stringToInteger(expression);
    } else {
        char op = getOperator(exp);
        string left = getLeftExp(exp);
        string right = getRightExp(exp);
        int leftResult = evaluate(left);
        int rightResult = evaluate(right);

        if (op == '+') {
            return leftResult + rightResult;
        } else if (op == '*') {
            return leftResult * rightResult;
        }
    }
}
```

exp = “((1*3)+((4*5)+2))”

op = ‘+’

left = “(1*3)”

right = “((4*5)+2)”

leftResult = 3

rightResult = 22

Helper Methods

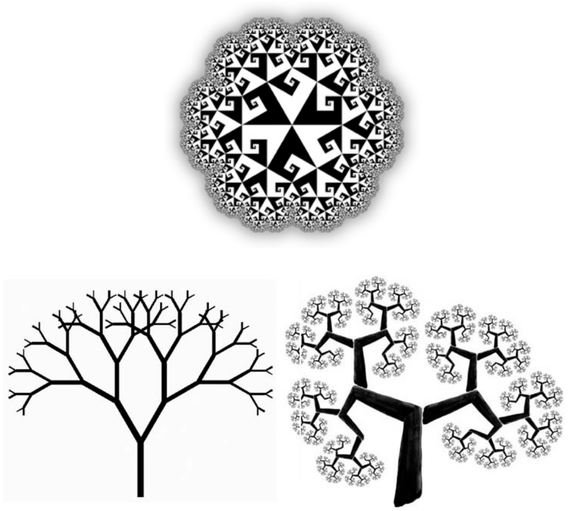
Here is the key function behind the helper methods:

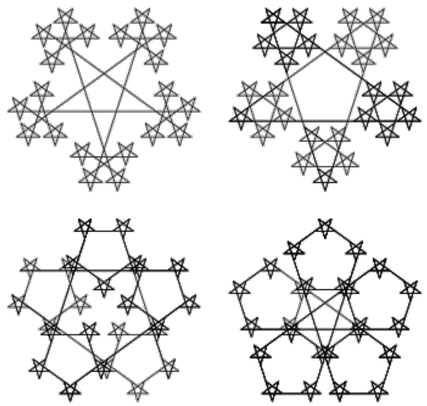
```
int getOppIndex(string exp){
    int parens = 0;
    for (int i = 0; i < exp.length(); i++) {
        char c = exp.charAt(i);
        if (c == '(') { parens++; }
        else if (c == ')') { parens--; }
        if (parens == 0 && (c == '+' || c == '*')) {
            return i;
        }
    }
}
```

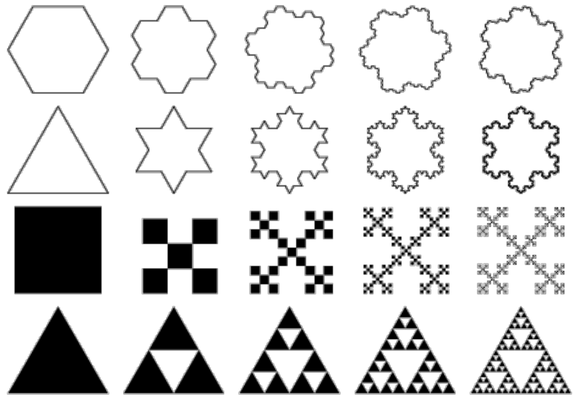
Recursion you can *see*

Fractal

fractal: A recurring graphical pattern. Smaller instances of the same shape or pattern occur within the pattern itself.







Fractals in Nature

Many natural phenomena generate fractal patterns:

1. earthquake fault lines
2. animal color patterns
3. clouds
4. mountain ranges
5. snowflakes
6. crystals
7. DNA
8. ...

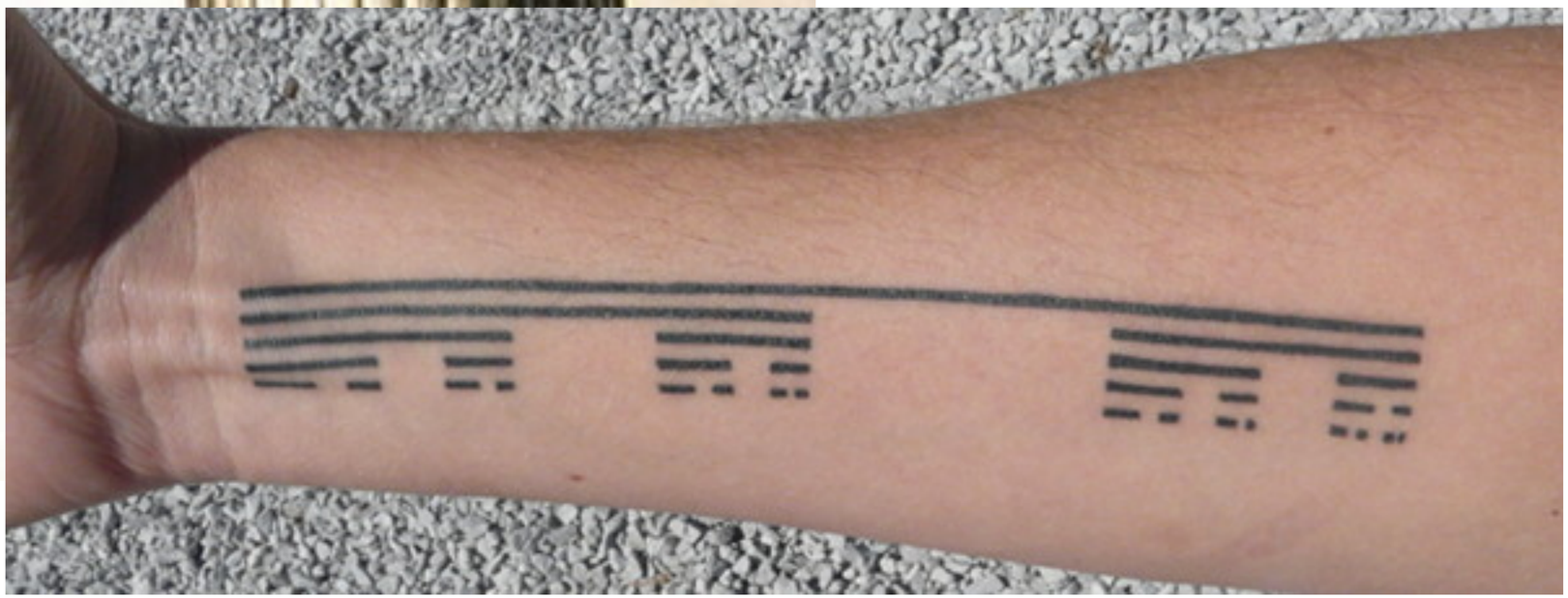
Ready?


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Let’s go!


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Cantor Fractal



Cantor Fractal

Parts of a Cantor set image…
are Cantor set images

Cantor Fractal

Start

End

Another cantor set

Also a cantor set

Levels of Cantor

6 Levels

Levels of Cantor

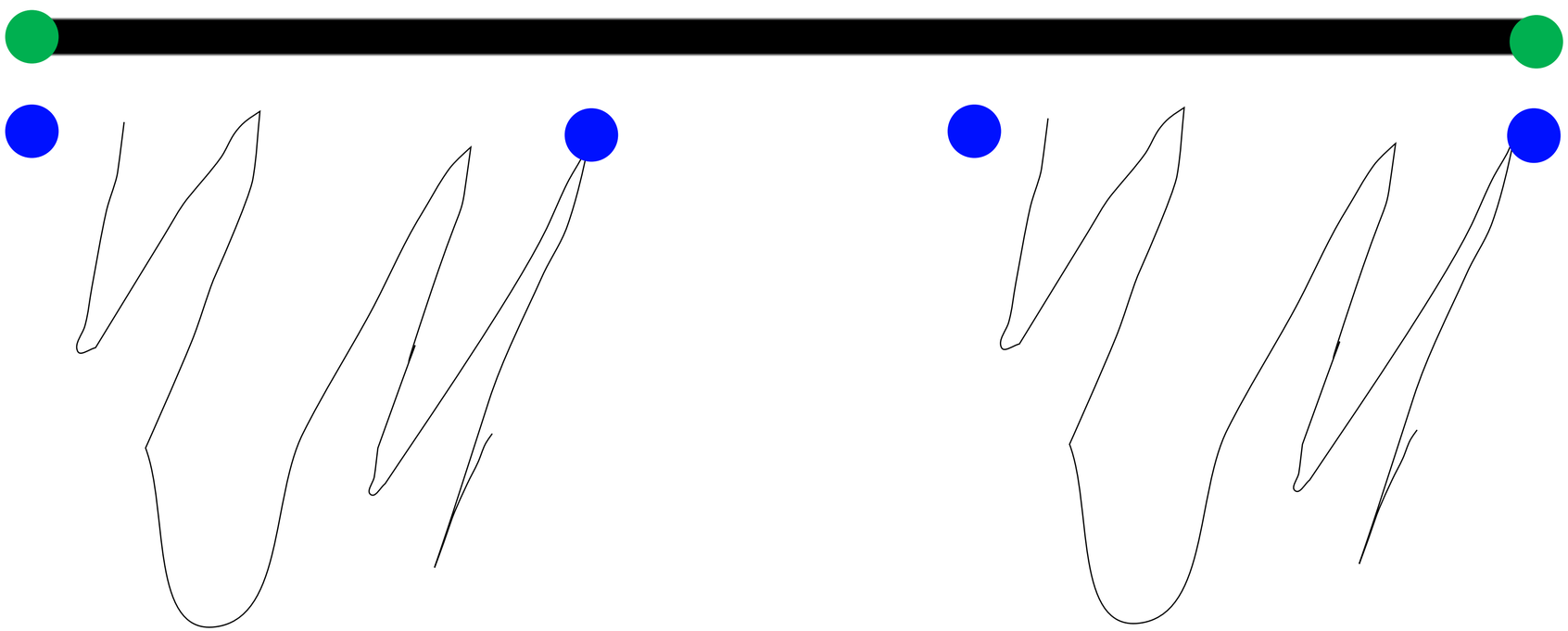
Levels of Cantor

1 Level

How to Draw a Level 1 Cantor?

How to Draw a Level n Cantor?

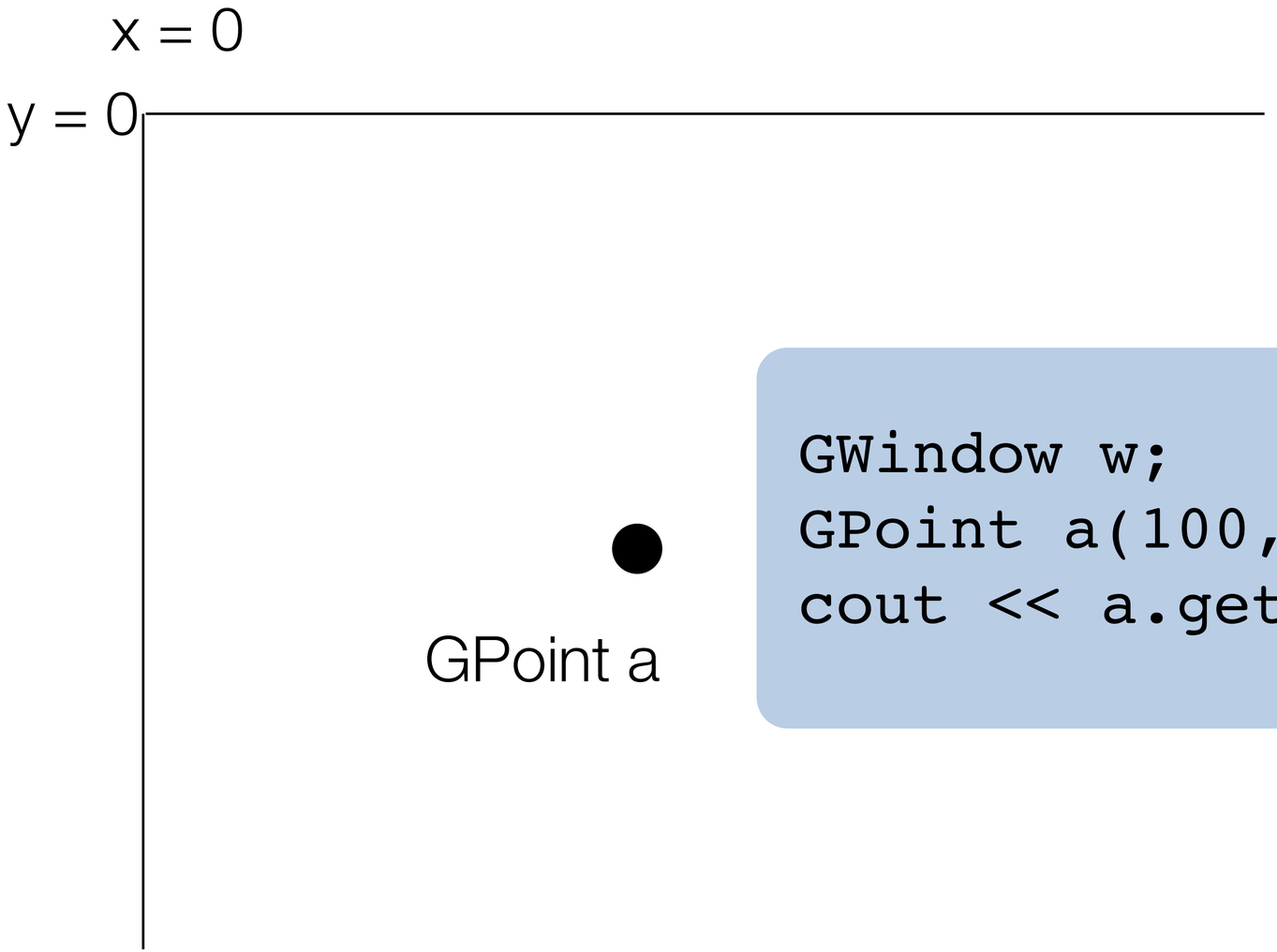
1 Draw a line from start to finish



2 Draw a Cantor of size n-1

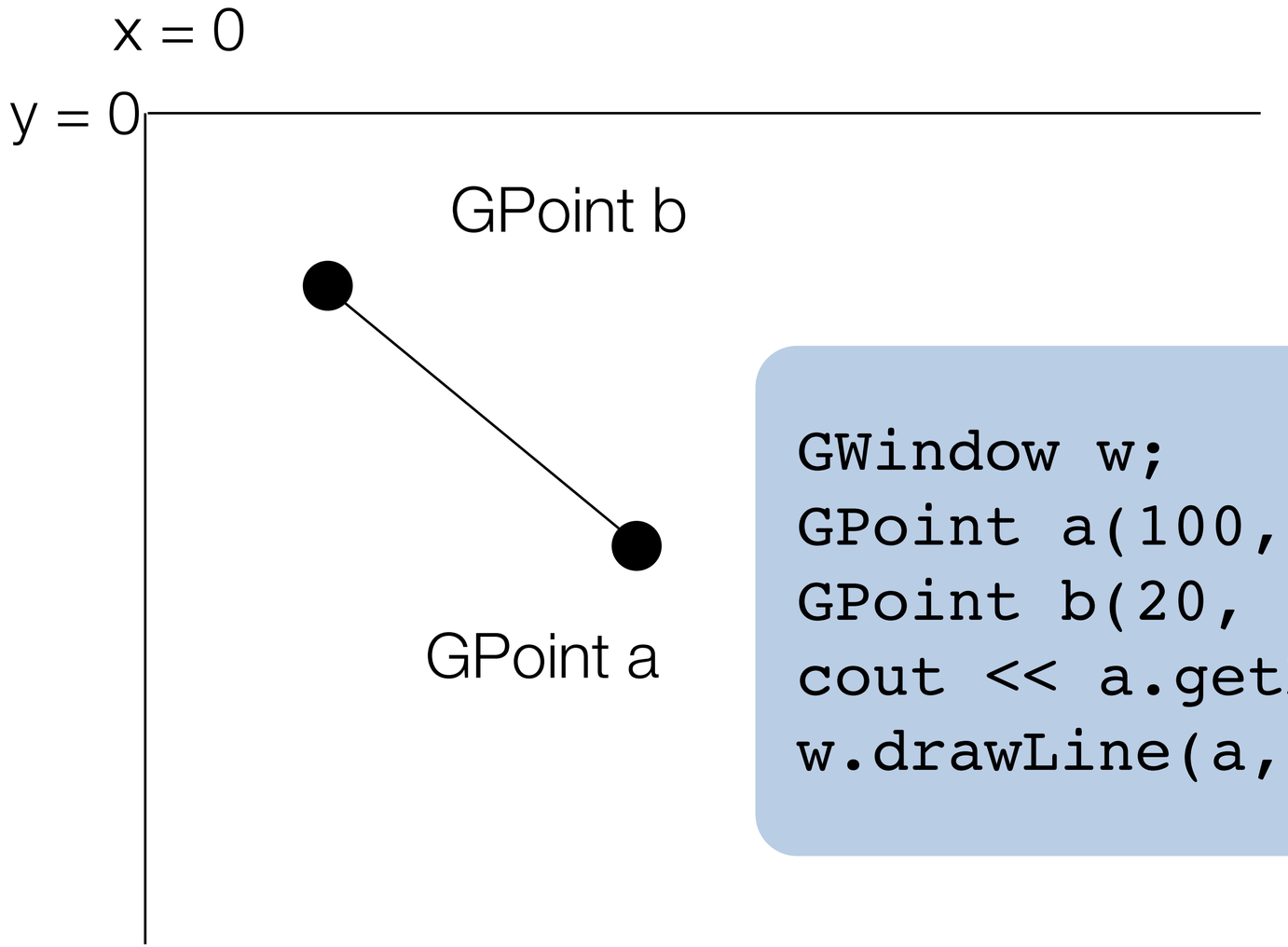
3 Draw a Cantor of size n-1

GPoint



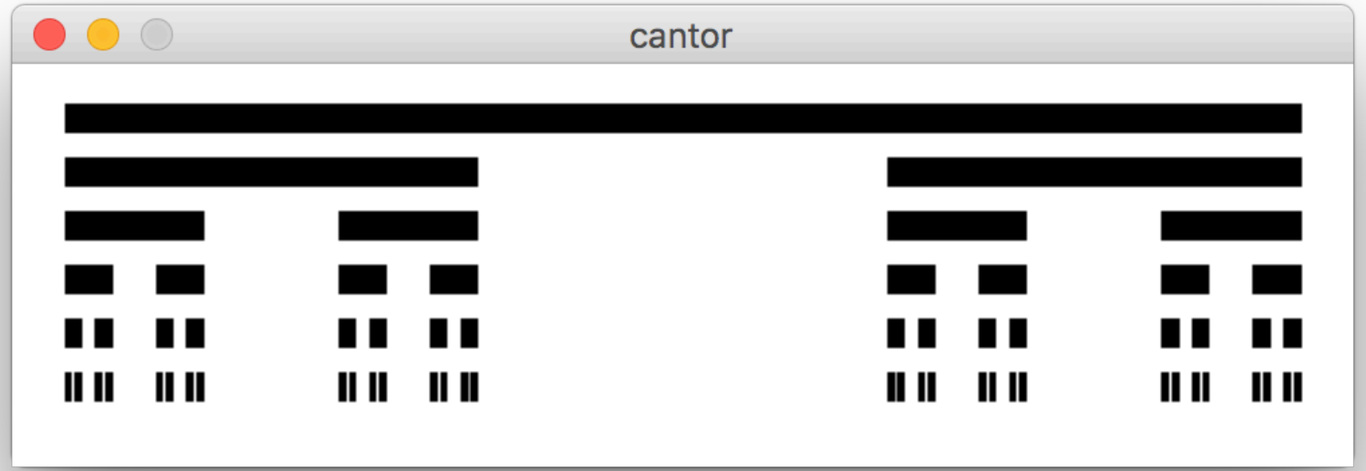

```
GWindow w;
GPoint a(100, 100);
cout << a.getX() << endl;
```

GPoint

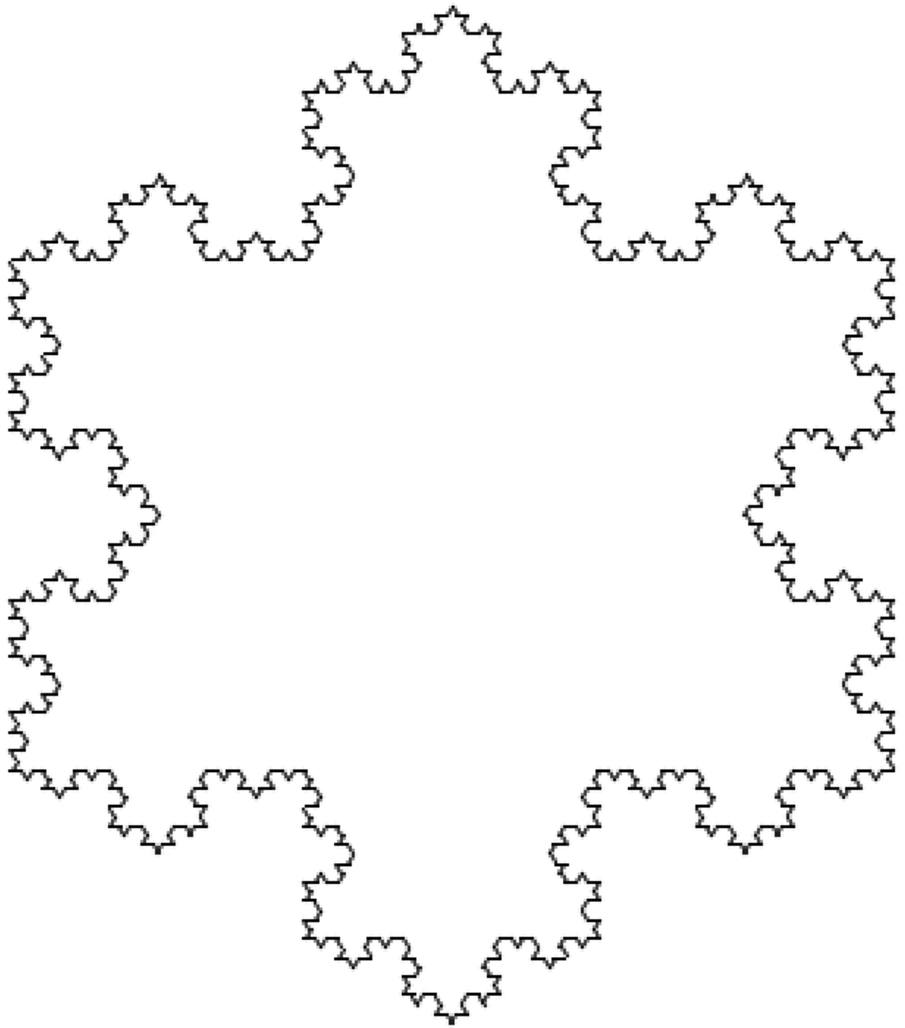


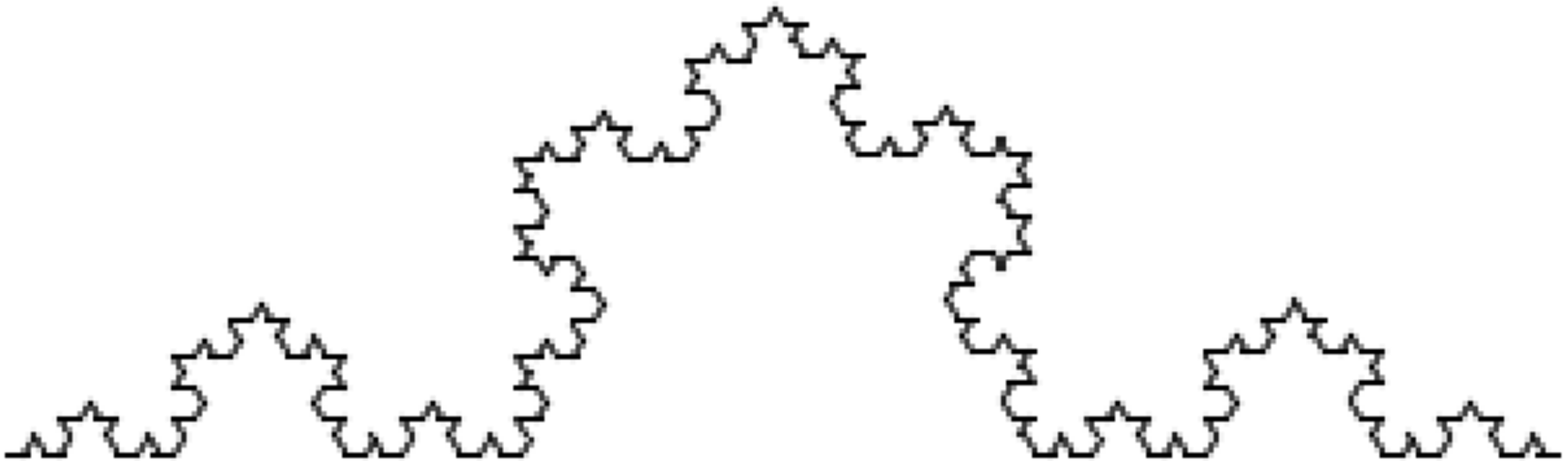

```
GWindow w;
GPoint a(100, 100);
GPoint b(20, 20);
cout << a.getX() << endl;
w.drawLine(a, b);
```

Cantor Fractal



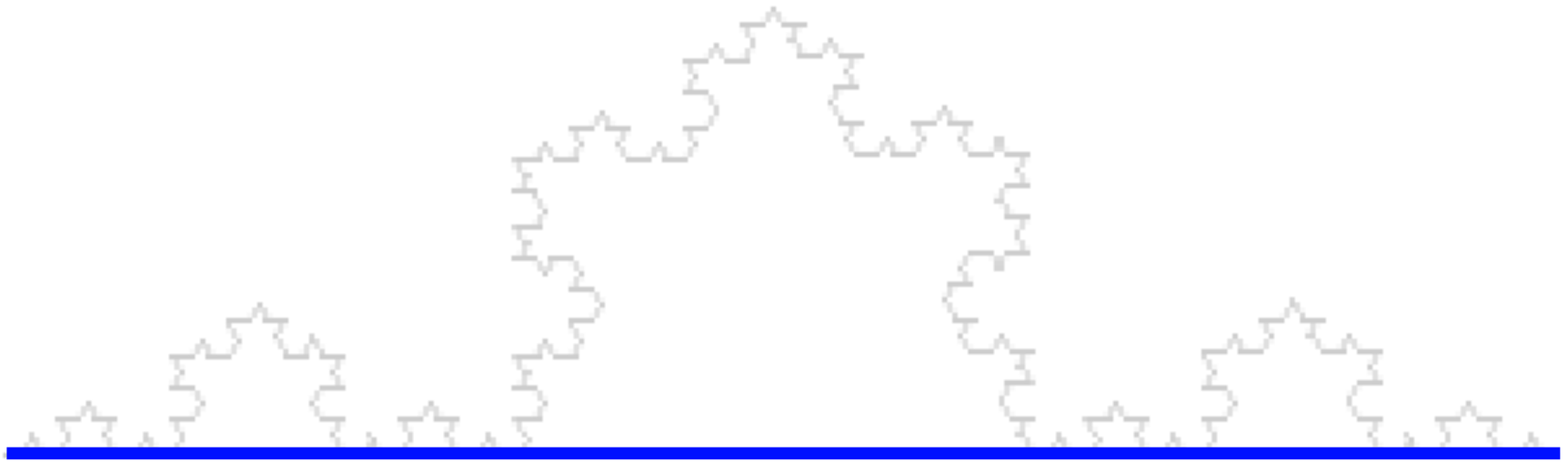
Snowflake Fractal



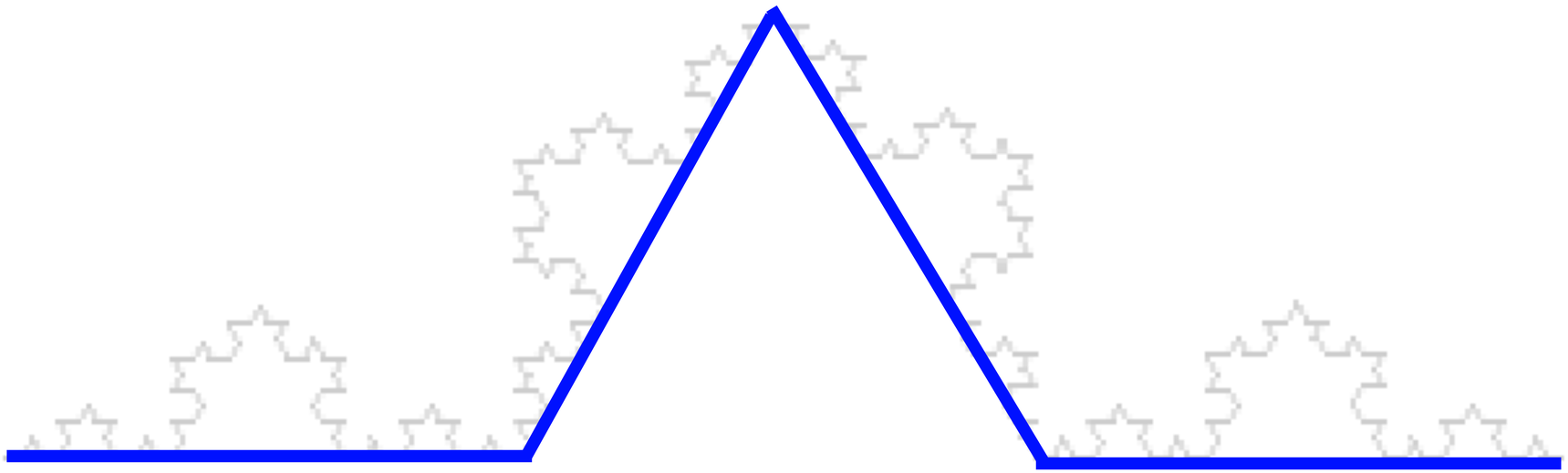



LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

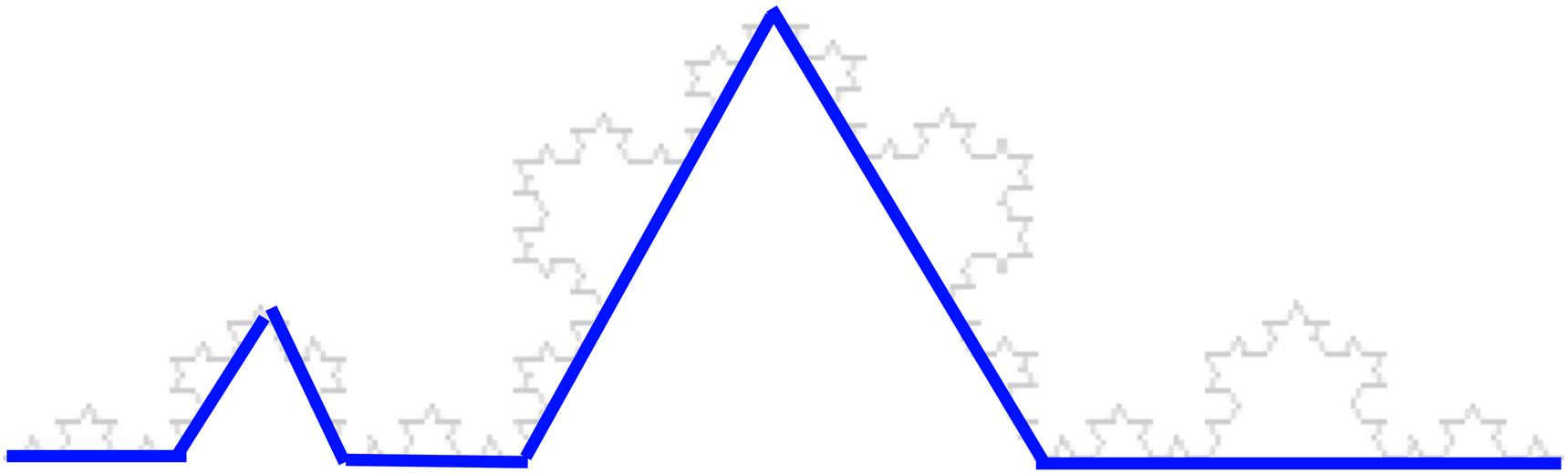
Depth 1 Snowflake Line



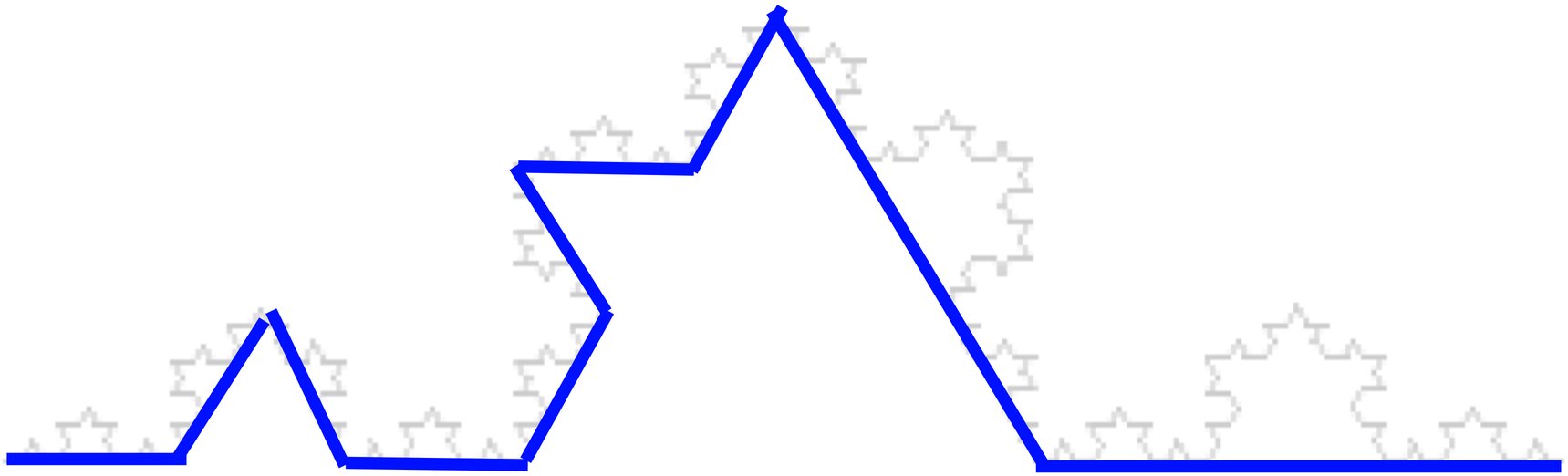
Depth 2 Snowflake Line



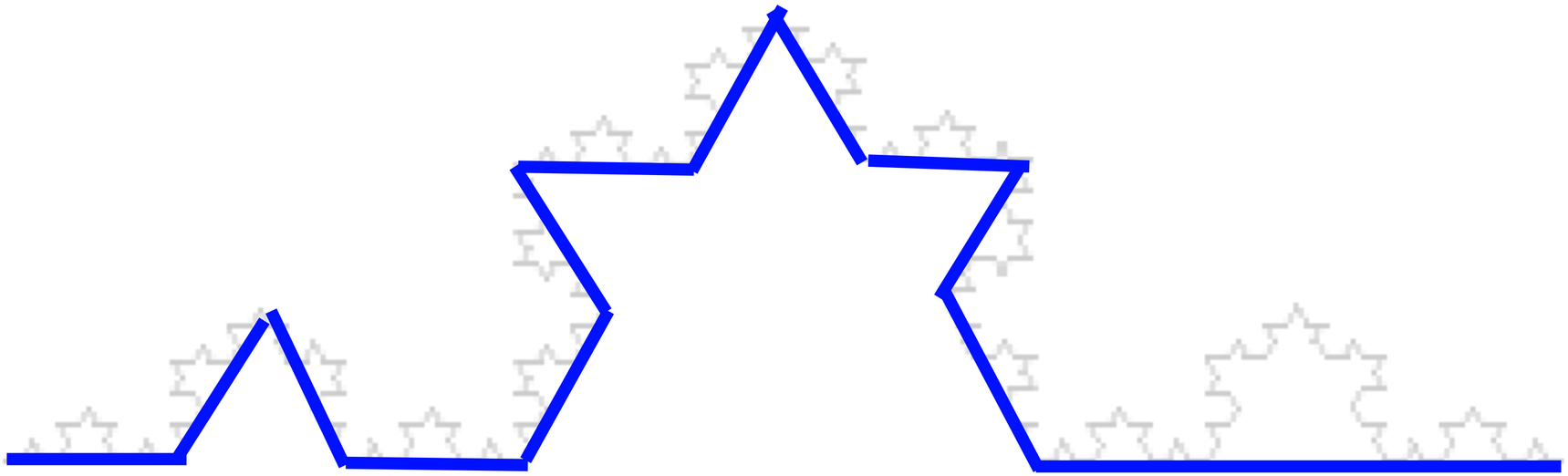
Depth 3 Snowflake Line (in progress)



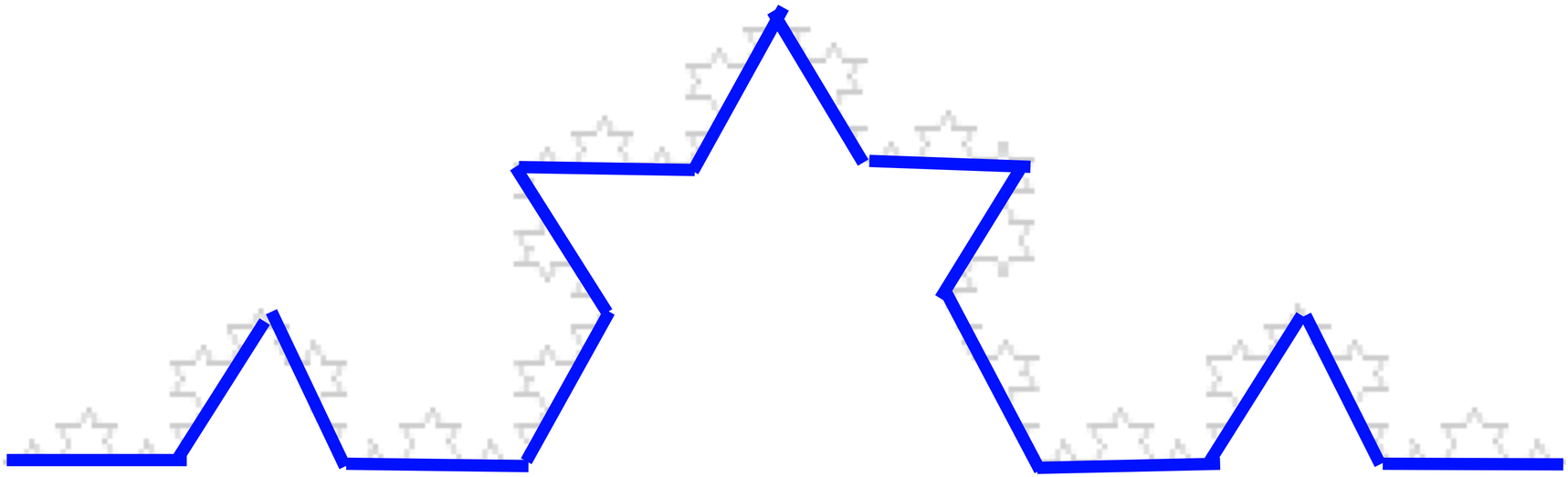
Depth 3 Snowflake Line (in progress)

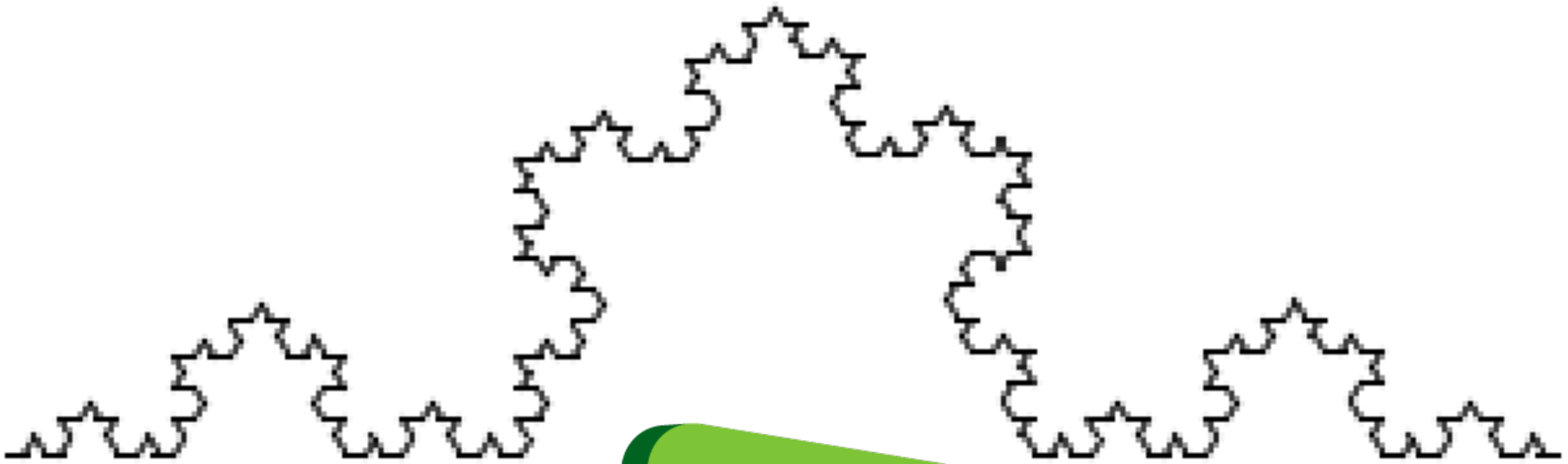


Depth 3 Snowflake Line (in progress)



Depth 3 Snowflake Line (in progress)



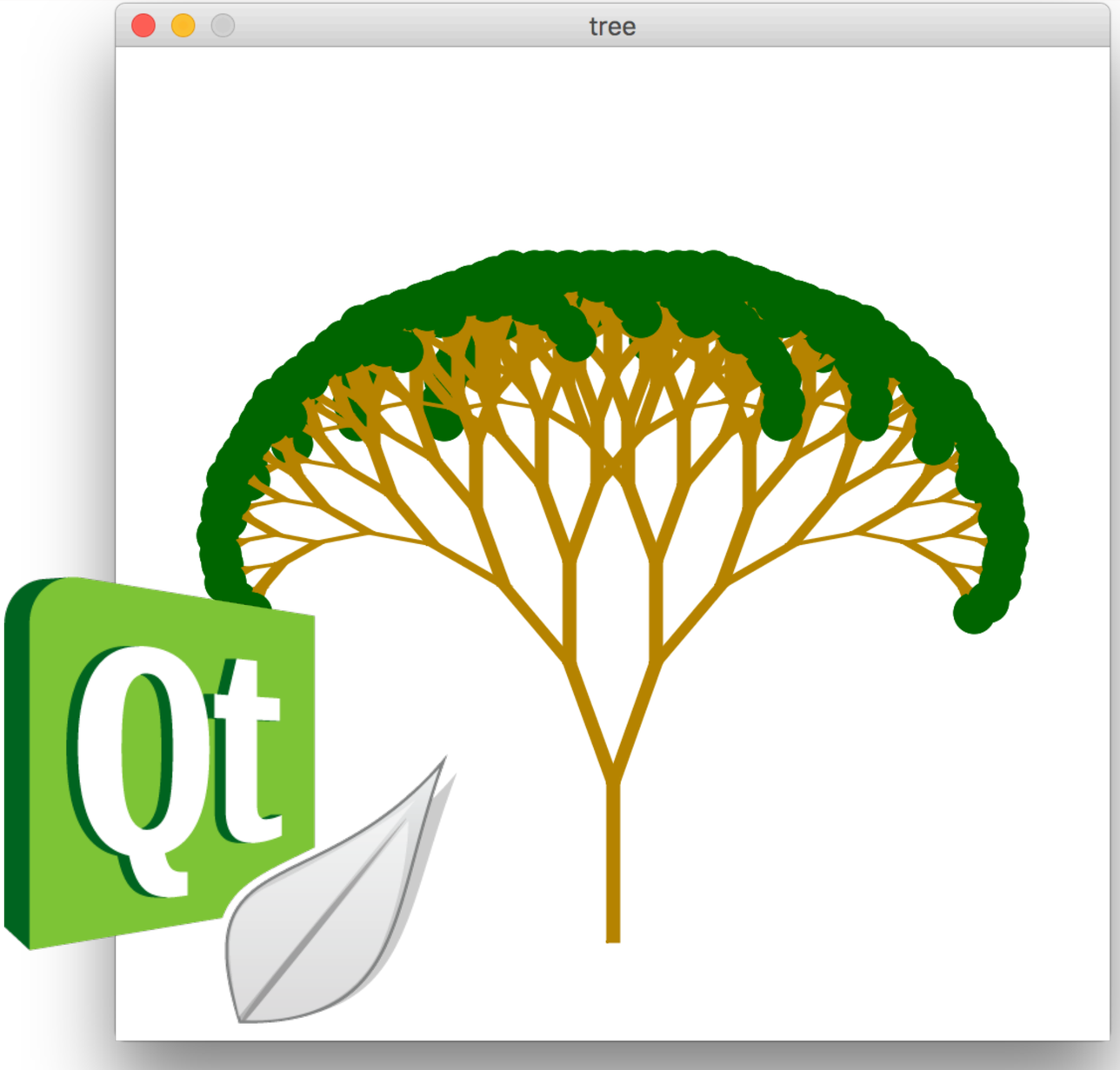



Qt

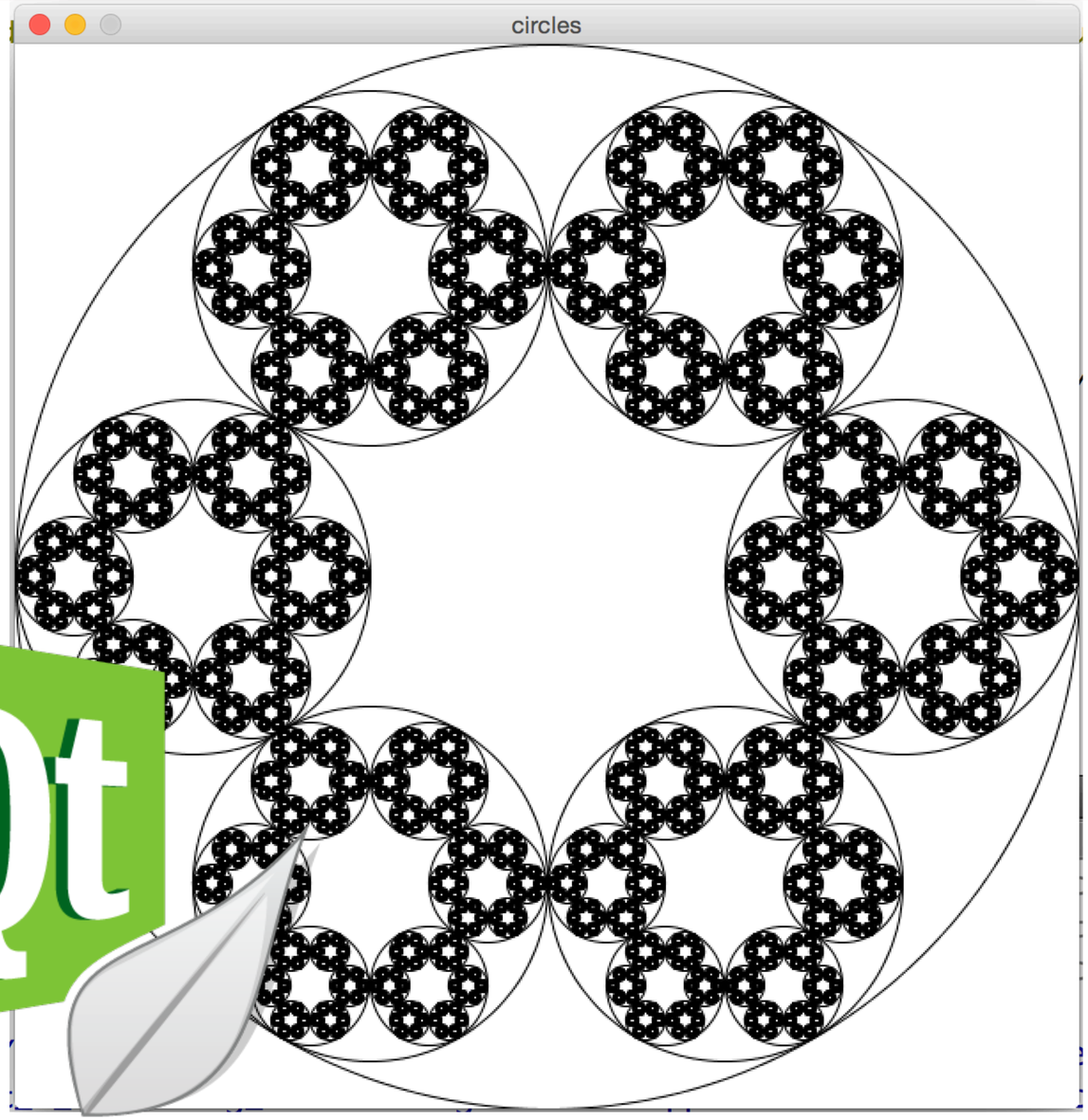

LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Extra Problems Online

Fractal Tree



Fractal Circles




Recursive
Ray Tracing


ydot

The End


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891