

evaluation with
provenance tracing

semantic analysis

well-typed, functional
programming language

widgets, draw tools,
and refactoring menus

algebraic simplification
via the REDUCE solver

heuristics for human-like
code generation

```
1 equiTriPt [x3, y3] [x2, y2] =  
2 [ (x2 + x3 + sqrt 3! * (y2 - y3))/ 2!, (y2 + y3 - sqrt 3! * (x2 - x3)) / 2! ]  
3  
4 oneThirdPt [x3, y3] [x, y] =  
5 [ x / 1.5!+ x3 / 3!, y / 1.5! + y3 / 3! ]  
6  
7 point = [39, 314]  
8  
9 point2 = [498, 385]  
10  
11 makeKochPts depth point point2 =  
12 let oneThirdPt2 = oneThirdPt point point2 in  
13 let oneThirdPt3 = oneThirdPt point2 point in  
14 let equiTriPt2 = equiTriPt oneThirdPt2 oneThirdPt3 in  
15 if depth = 2 then  
16 [point, oneThirdPt3, equiTriPt2, oneThirdPt2]  
17  
18 else  
19 let makeKochPts2 = makeKochPts (depth - 1) point oneThirdPt3 in  
20 let makeKochPts3 = makeKochPts (depth - 1) oneThirdPt3 equiTriPt2 in  
21 let makeKochPts4 = makeKochPts (depth - 1) equiTriPt2 oneThirdPt2 in  
22 let makeKochPts5 = makeKochPts (depth - 1) oneThirdPt2 point2 in  
23 concat [makeKochPts2, makeKochPts3, makeKochPts4, makeKochPts5]  
24  
25 depth = 3 [1-5]  
26  
27 topPts = makeKochPts depth point point2  
28  
29 botCorner = equiTriPt point2 point  
30  
31 rightPts = makeKochPts depth point2  
32  
33 leftPts = makeKochPts depth botCorner  
34  
35 snowflakePts = concat [topPts, rightPts, leftPts]  
36  
37 polygon =  
38 let pts = snowflakePts in  
39 let [color, strokeColor, strokeWidth] = [24, 360, 2] in  
40 polygon color strokeColor strokeWidth pts  
41  
42 svg (concat [  
43 [polygon]  
44 ])  
45
```

$\Rightarrow \dots \Rightarrow v$

Program
Transformations

