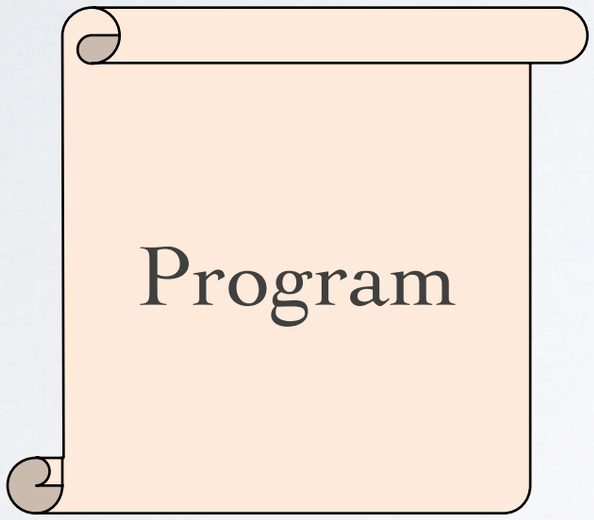
PROGRAMMATIC AND DIRECT MANIPULATION

TOGETHER AT LAST

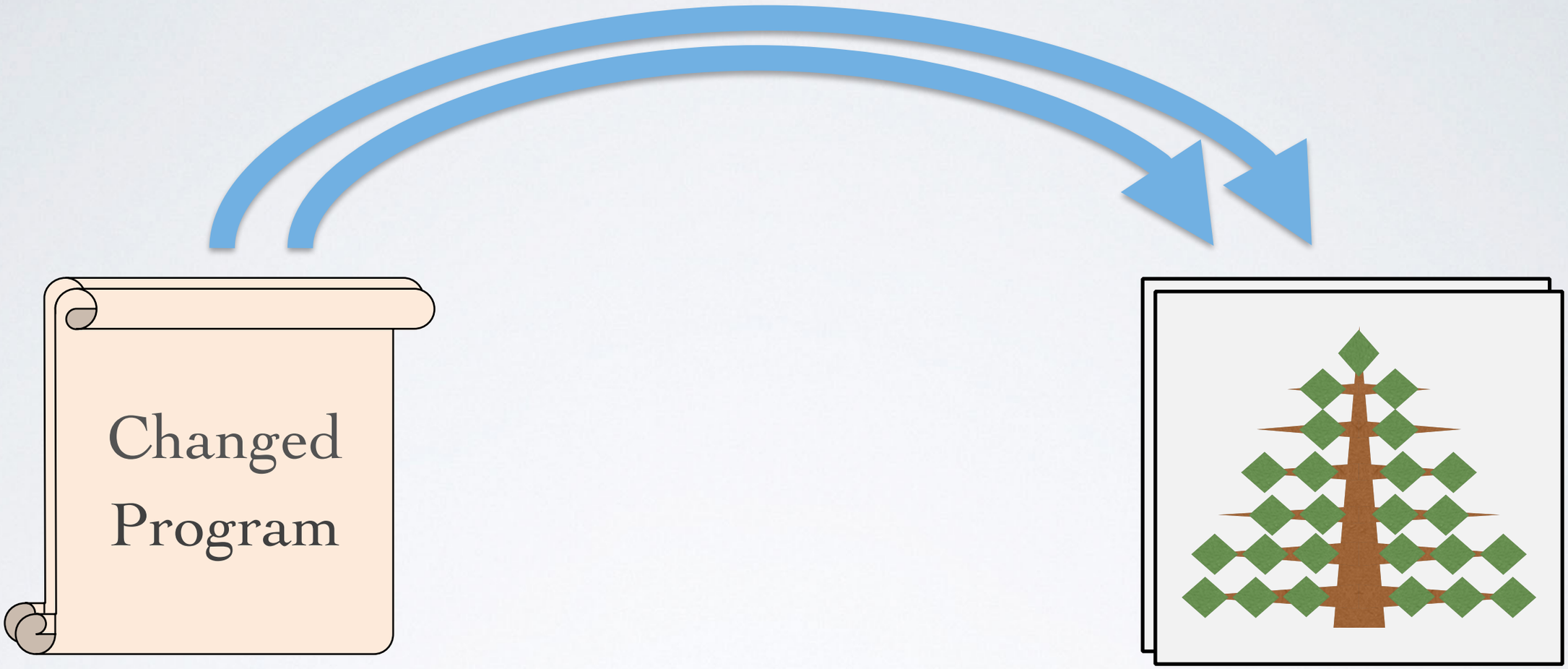
Ravi Chugh, Brian Hempel,
Mitchell Spradlin, Jacob Albers

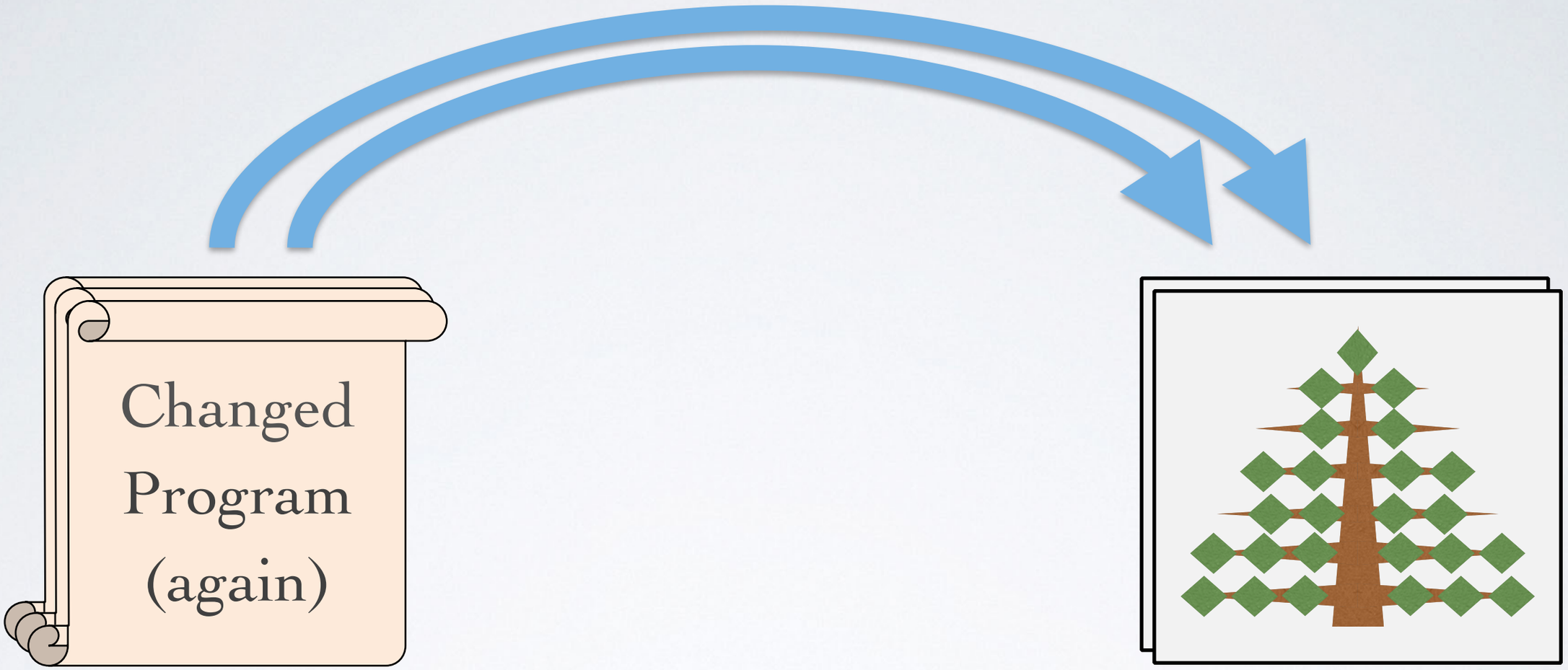
THE UNIVERSITY OF
CHICAGO


Program

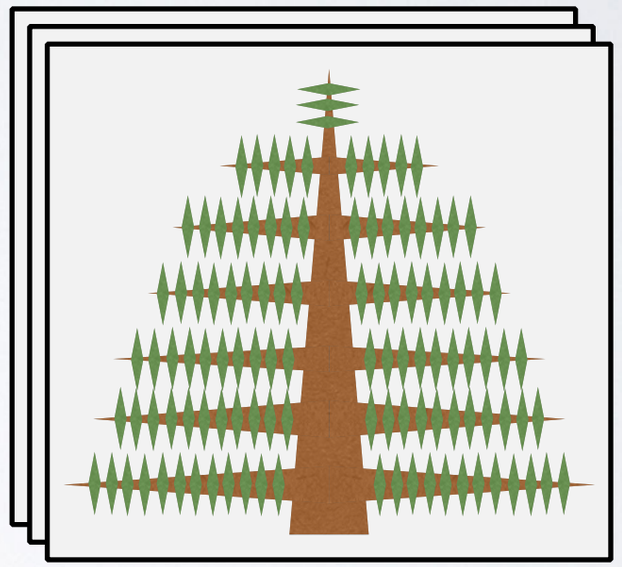

Program

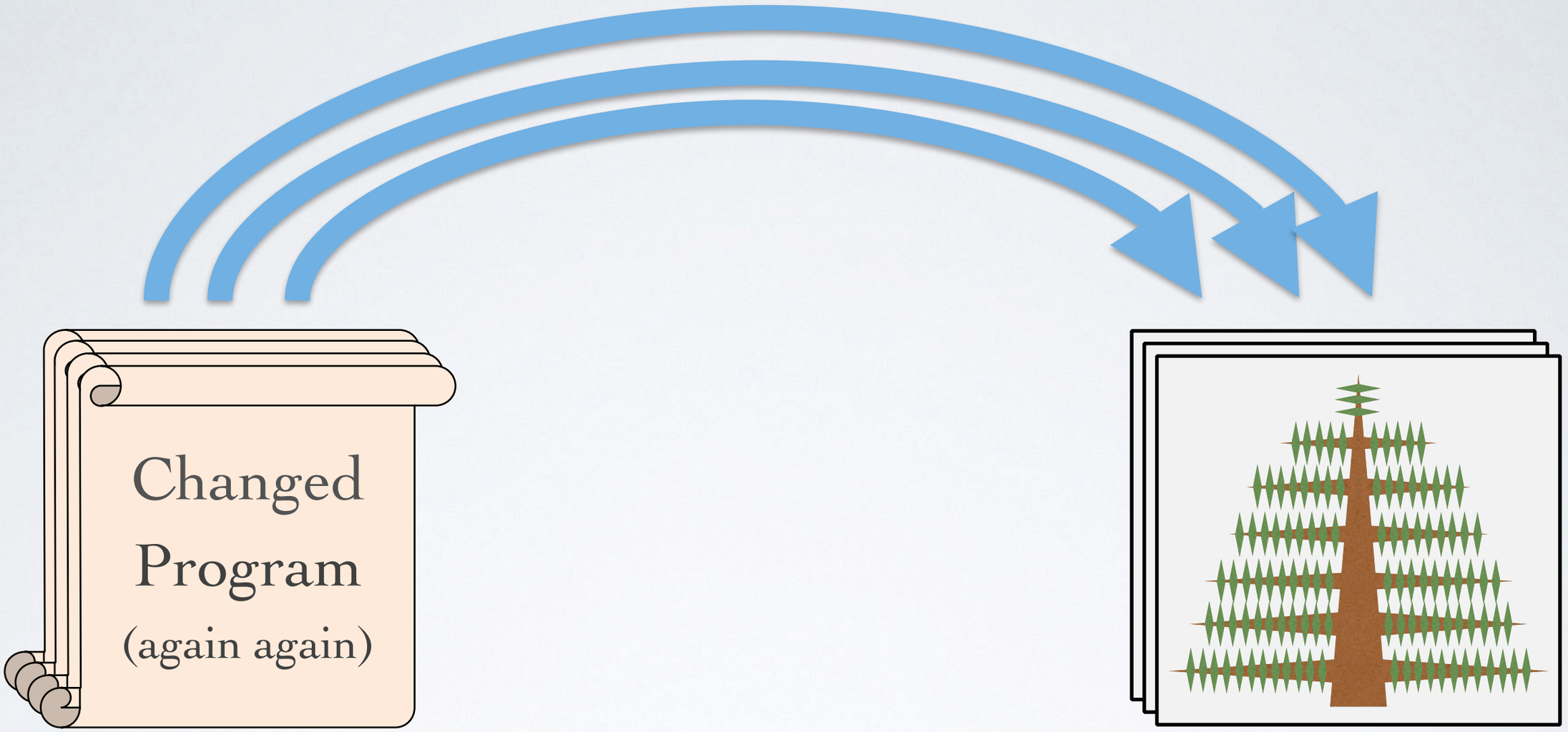

Changed
Program


Changed
Program

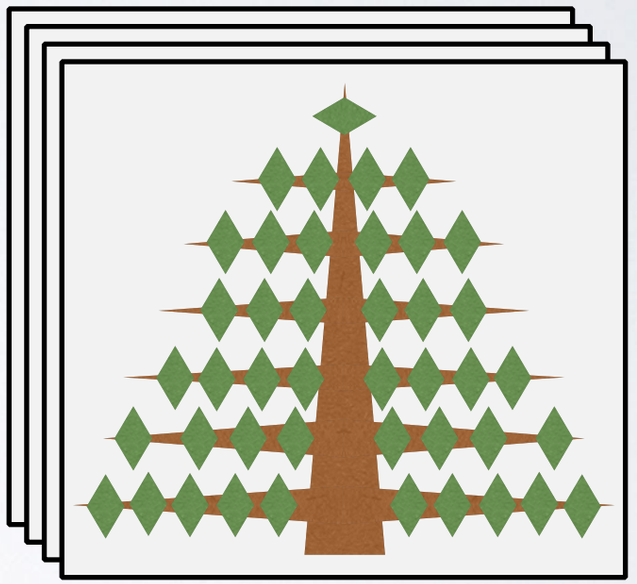

Changed
Program
(again)

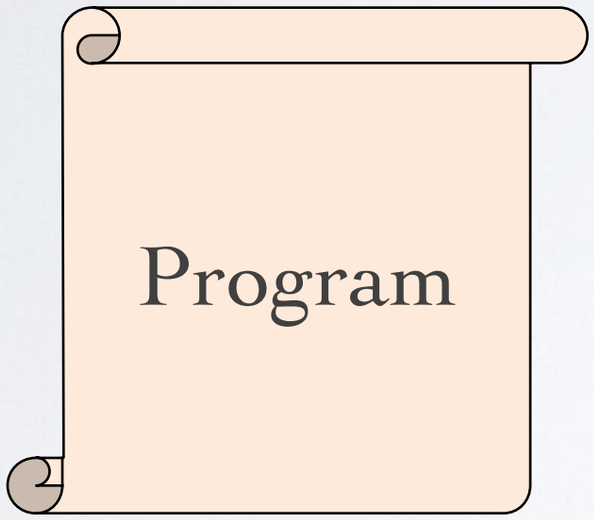
Changed Program (again)




Changed
Program
(again again)

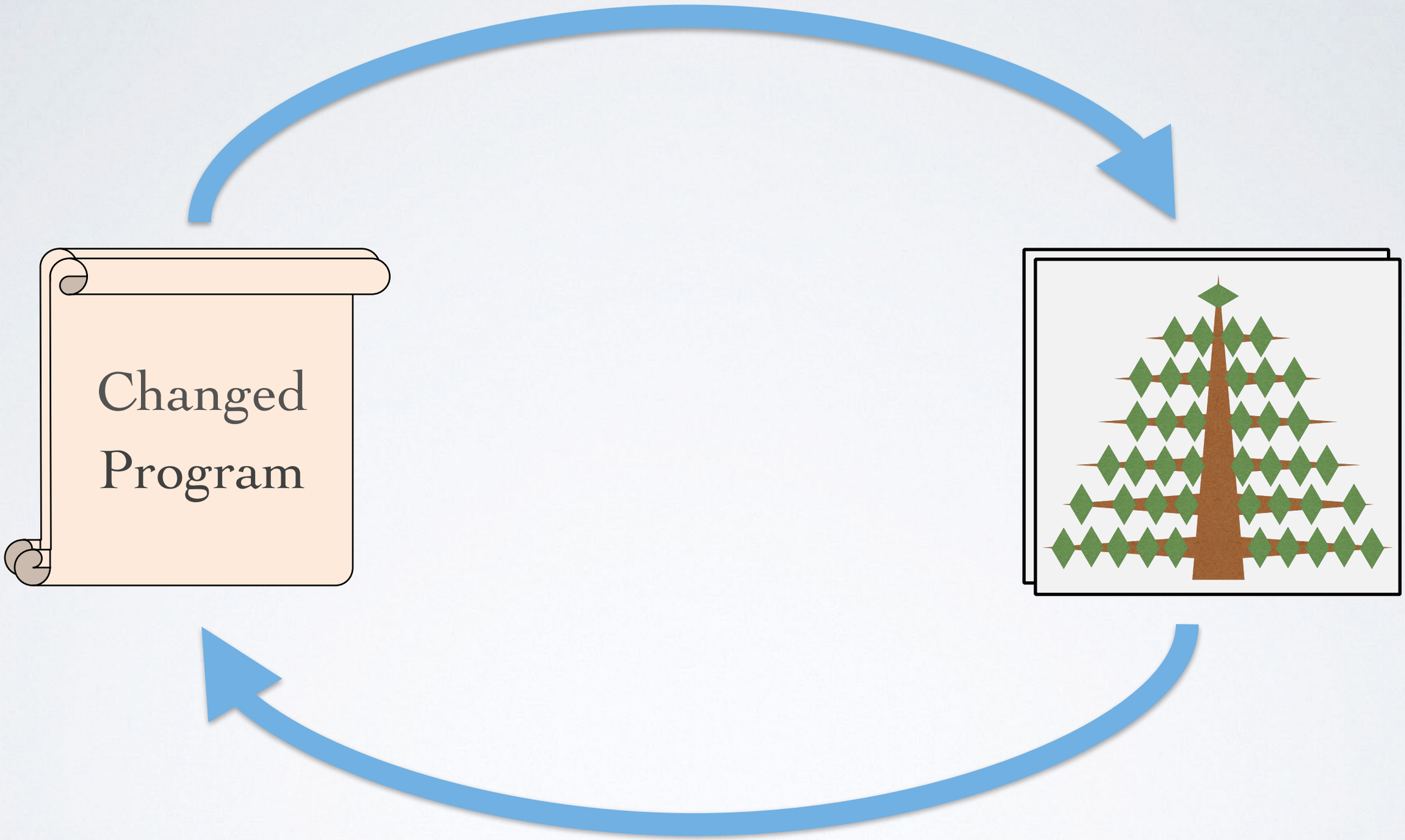
Changed
Program
(again again)




Program

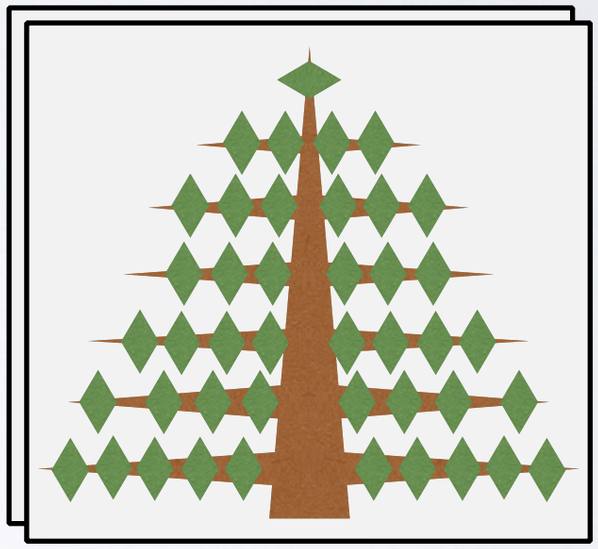

Program


Program


Changed
Program

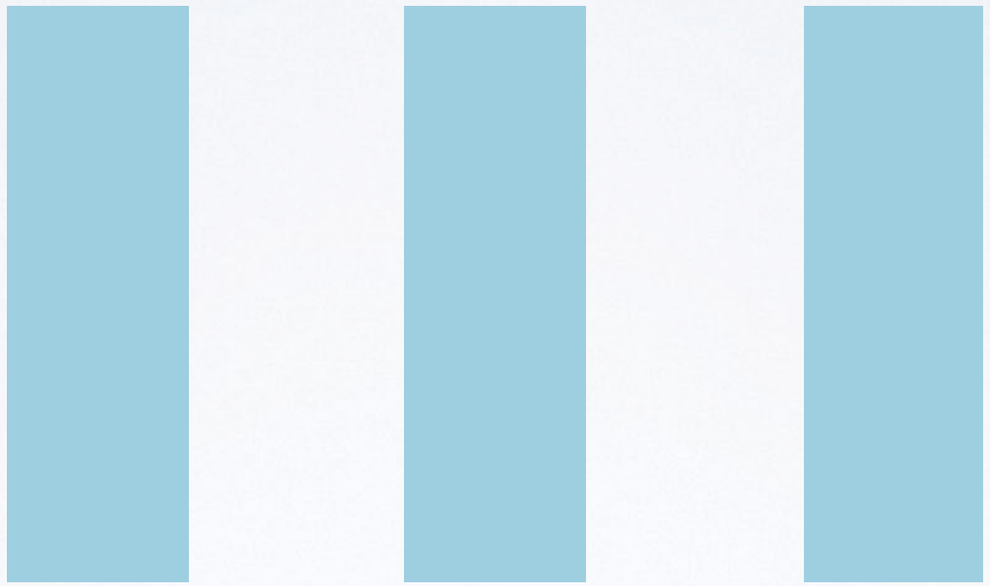
GOAL: Live Sync

Changed Program



Fast + Intuitive + Automatic

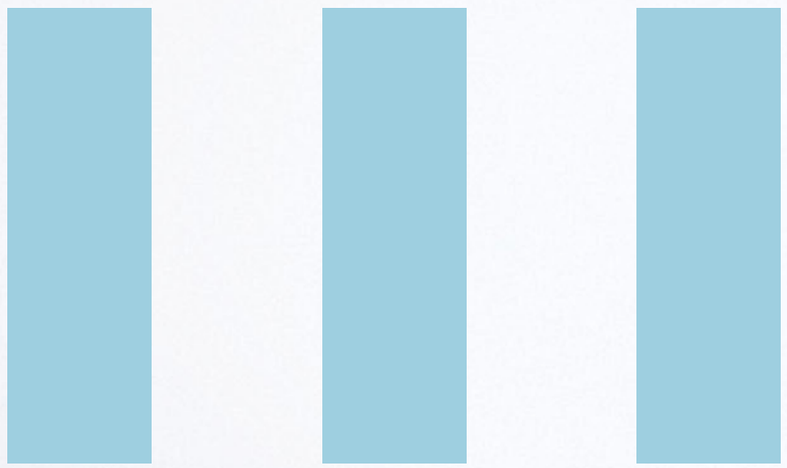
SIMPLE EXAMPLE



```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

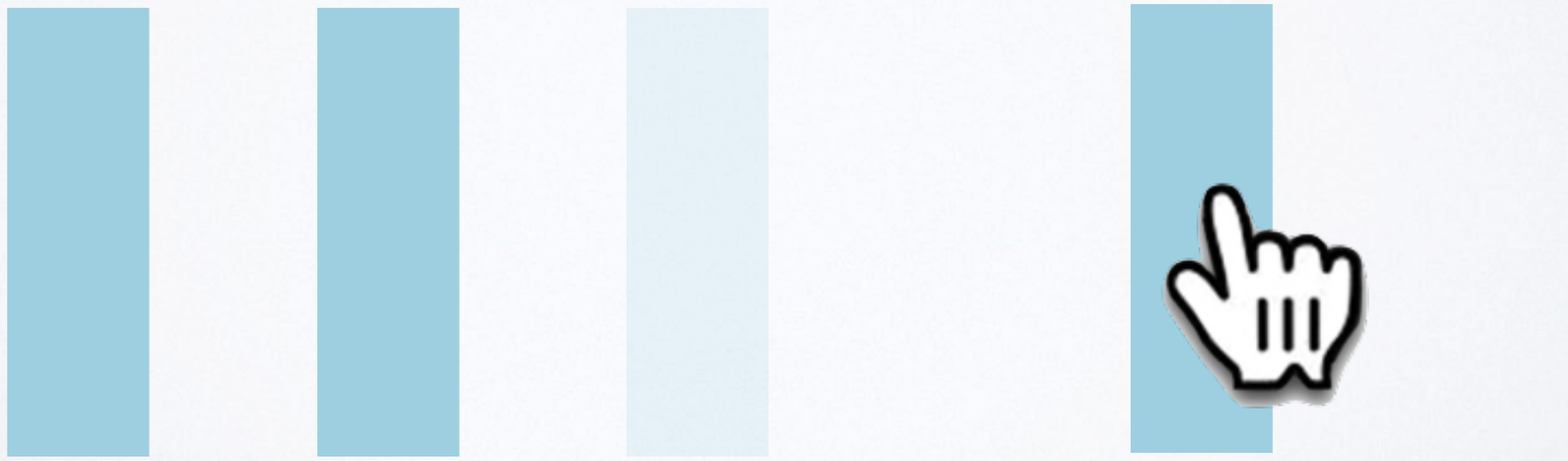


```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

x2 = 110 x2’ = 155

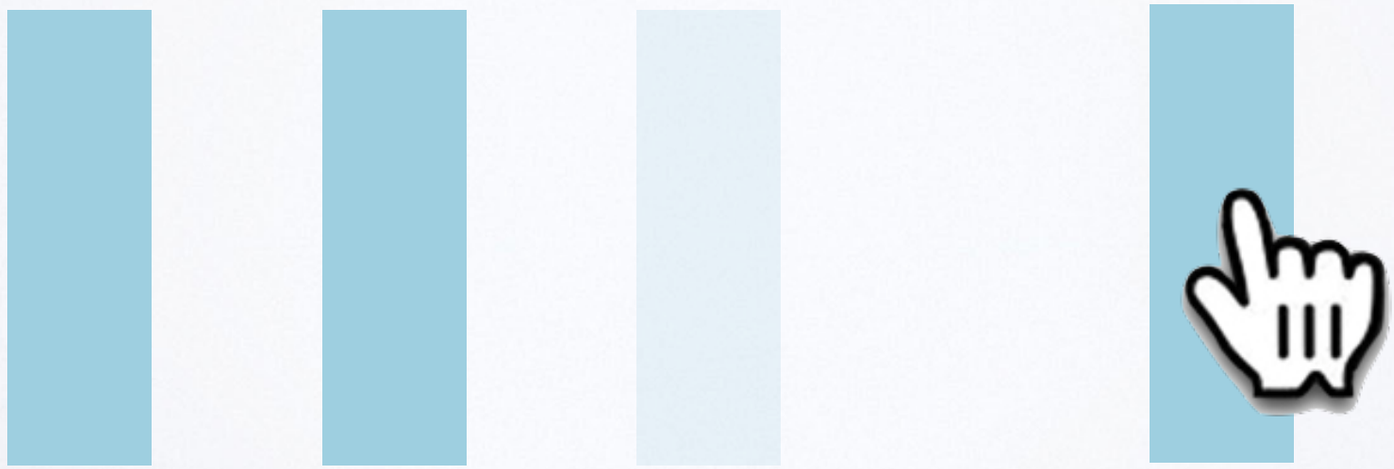


```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

GOAL

Change program so xi = 155 when i = 2

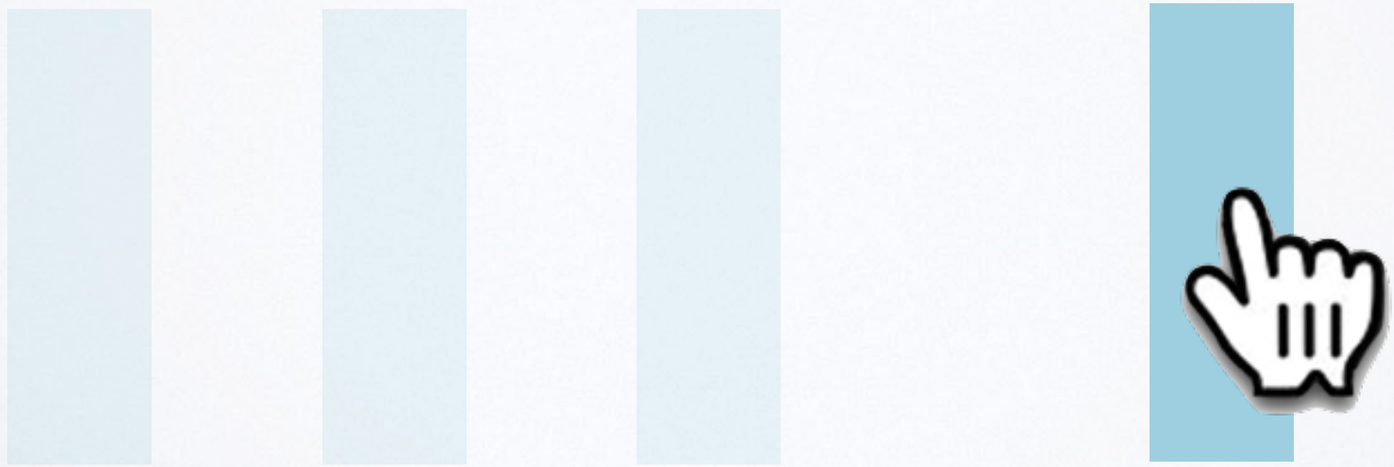
x2 = 110 x2’ = 155



```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

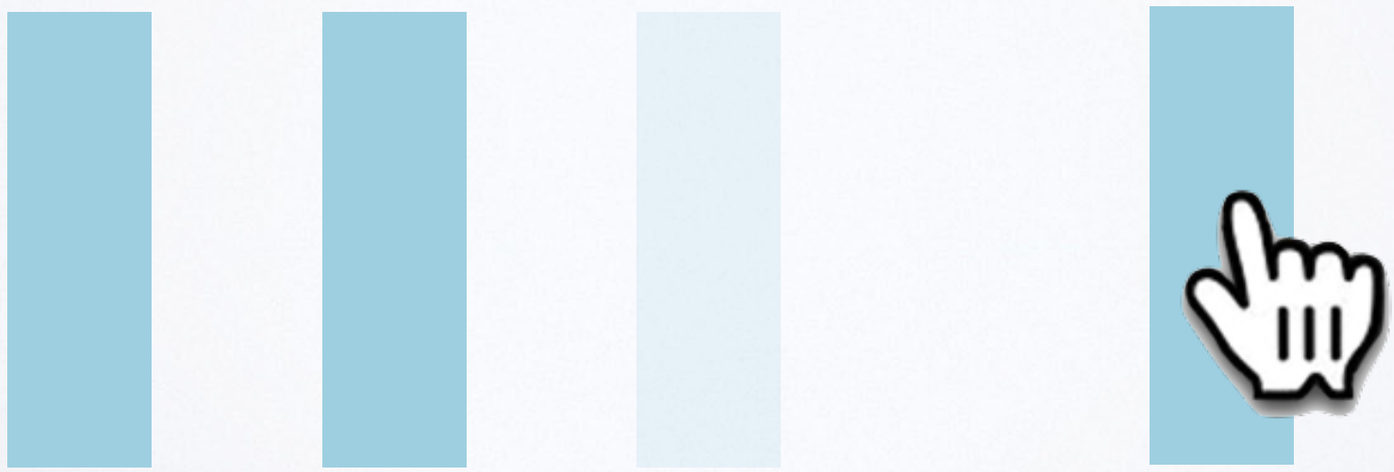
x2’ = 155



```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = if i=2 then 155 else x0+i*sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

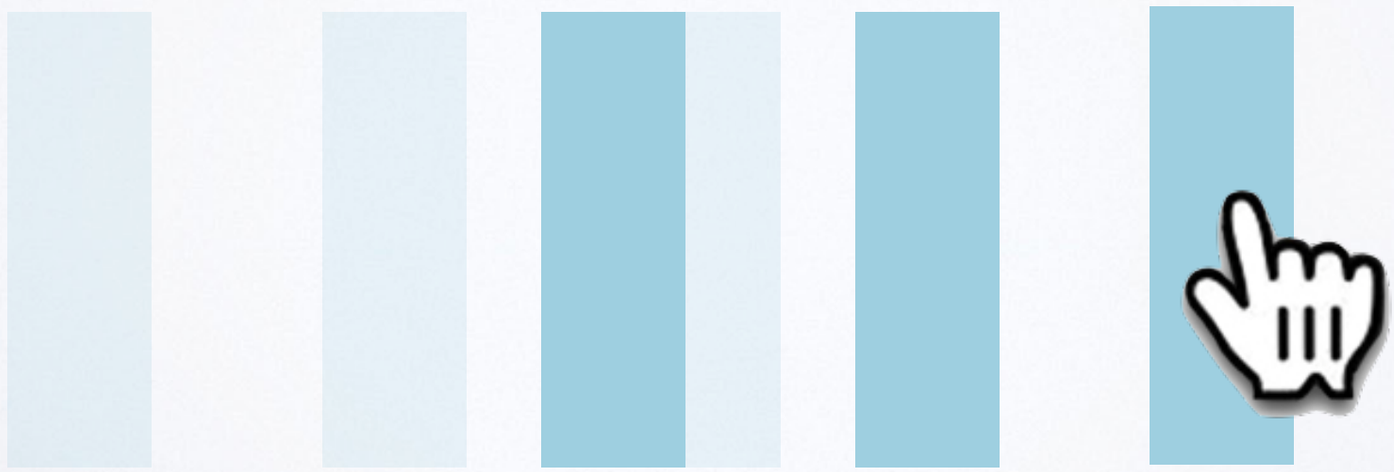
x2’ = 155



```
let (x0, y0,  w,  h,  sep, n) =
    (95, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

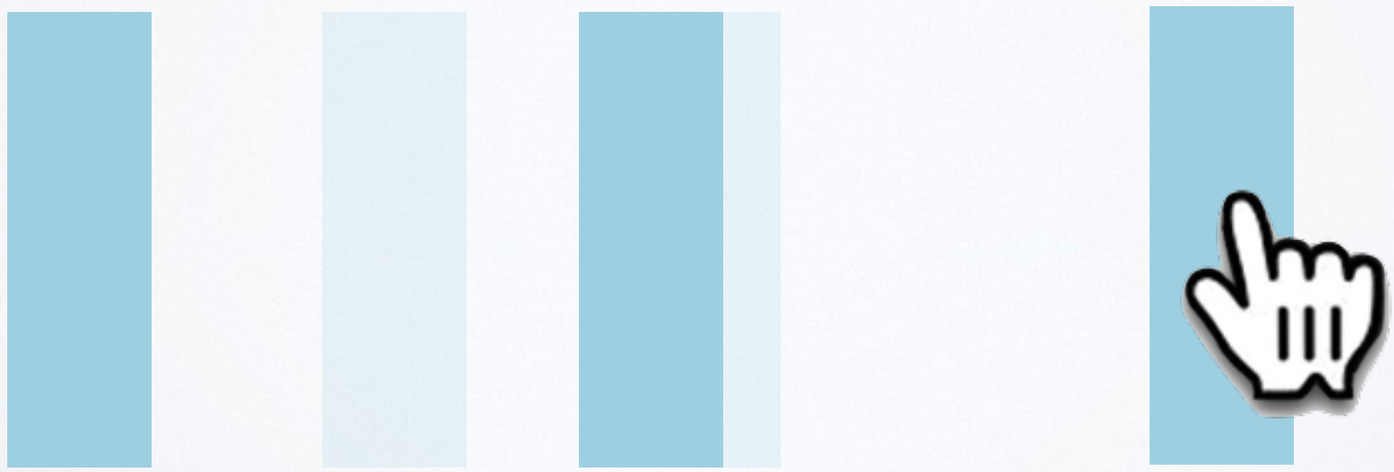
x2’ = 155



```
let (x0, y0,  w,  h,  sep,  n) =
    (95, 120, 20, 90, 52.5, 3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

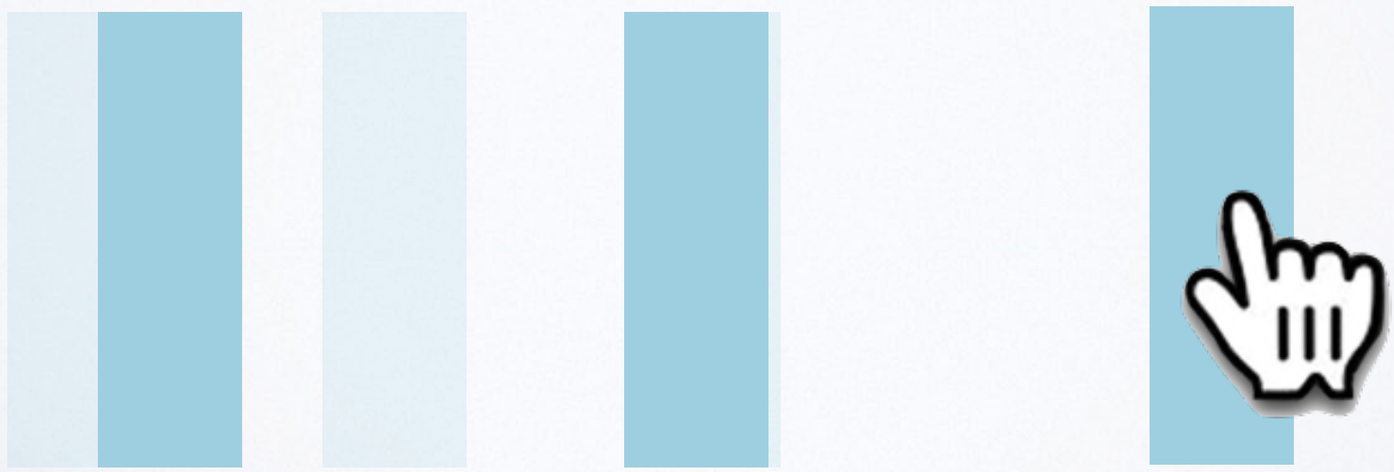
x2’ = 155



```
let (x0, y0,  w,  h,  sep, n) =
    (65, 120, 20, 90, 55,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

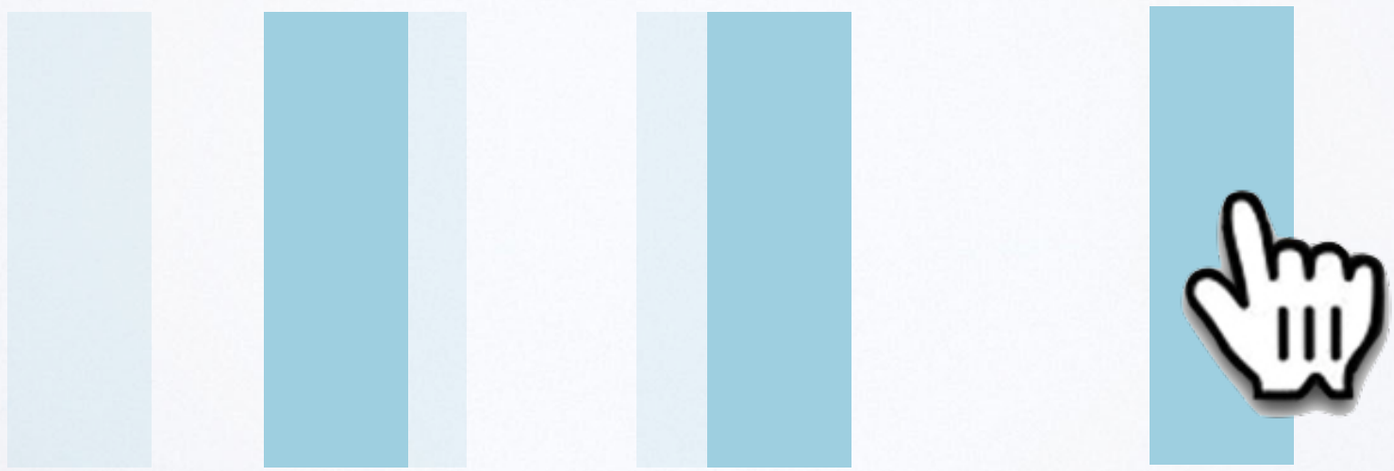
x2’ = 155



```
let (x0,  y0,   w,   h,   sep, n) =
    (75,  120,  20,  90,  40,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

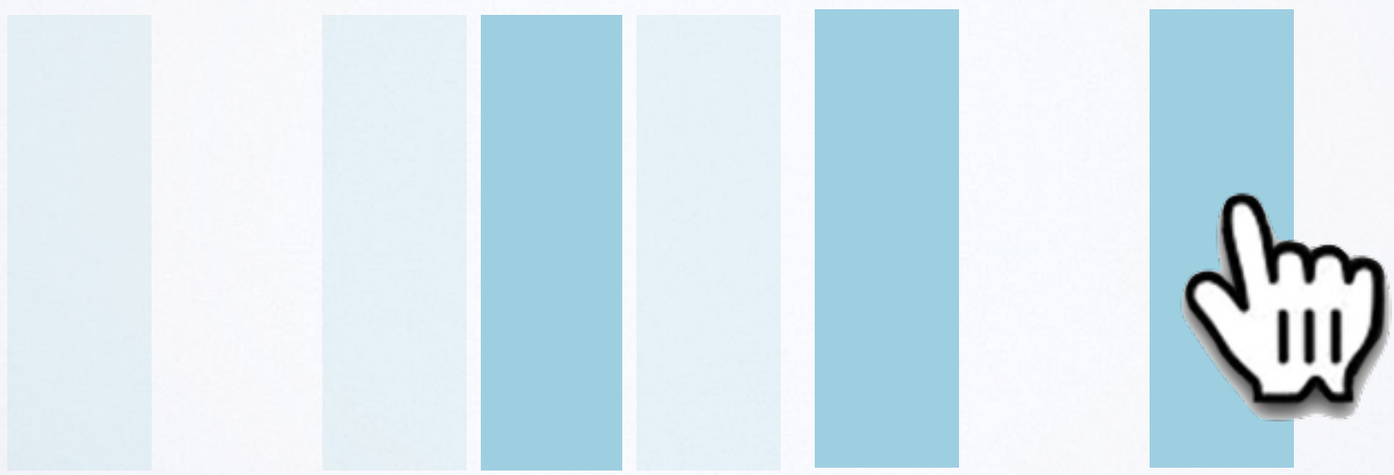
x2’ = 155



```
let (x0,  y0,  w,  h,  sep, n) =
    (95, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h
```

```
map boxi (zeroTo n)
```

x2’ = 155



155 = x0 + 2*sep

```
xi ↦ if i=2 then 155 else x0 + i * sep
```

```
x0 ↦ 75, sep ↦ 40
```

```
x0 ↦ 55, sep ↦ 50
```

```
x0 ↦ …, sep ↦ …
```

```
x0 ↦ 95
```

```
sep ↦ 52.5
```

x2’ = 155

155 = x0 + 2*sep

xi

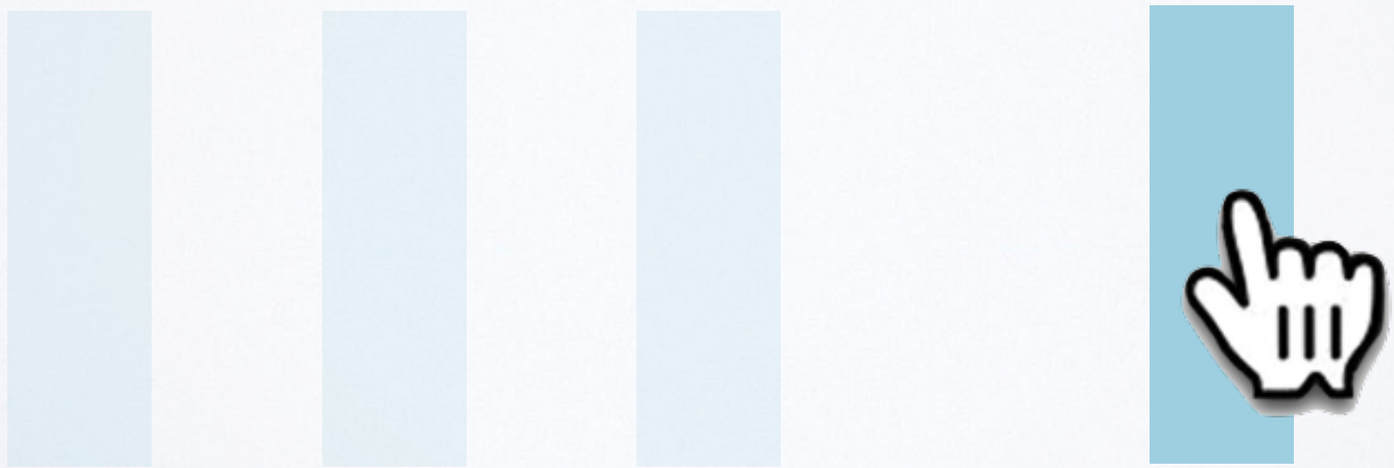
x0 ↦ 75, se

, sep ↦ …

GOAL

Intuitive Changes

x2’ = 155



155 = x0 + 2*sep

```
xi ↦ if i=2 then 155 else x0 + i * sep
```

```
x0 ↦ 75, sep ↦ 40
```

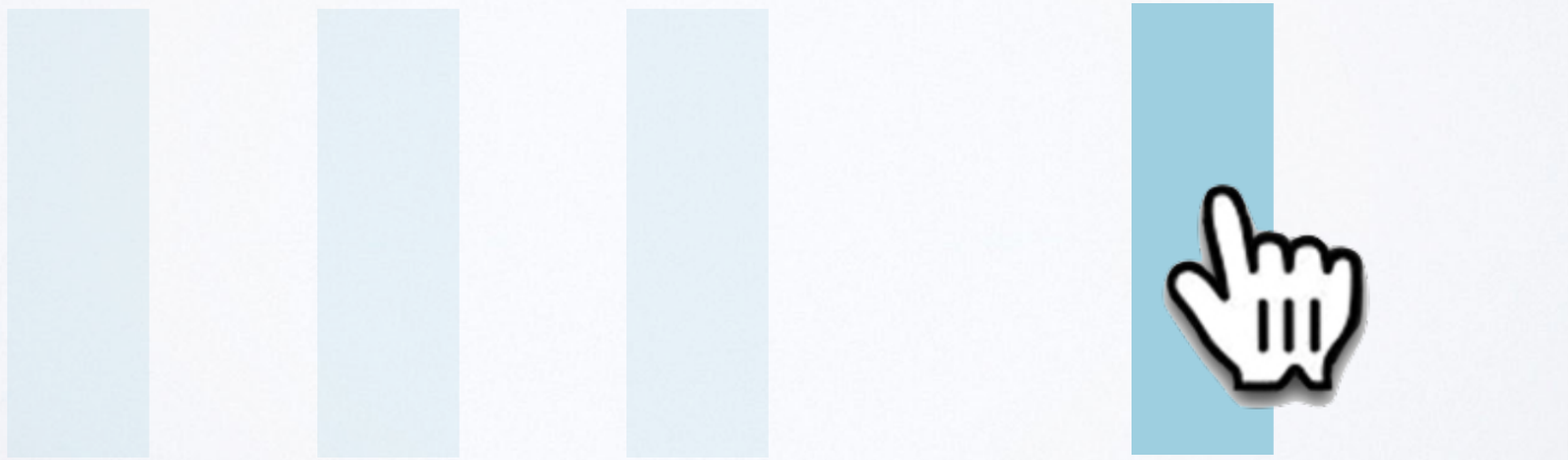
```
x0 ↦ 55, sep ↦ 50
```

```
x0 ↦ …, sep ↦ …
```

```
x0 ↦ 95
```

```
sep ↦ 52.5
```

x2’ = 155



155 = x0 + 2*sep

xi ↦ if i=2 then 155 else x0 + i * sep

x0 ↦ 75, sep ↦ 40

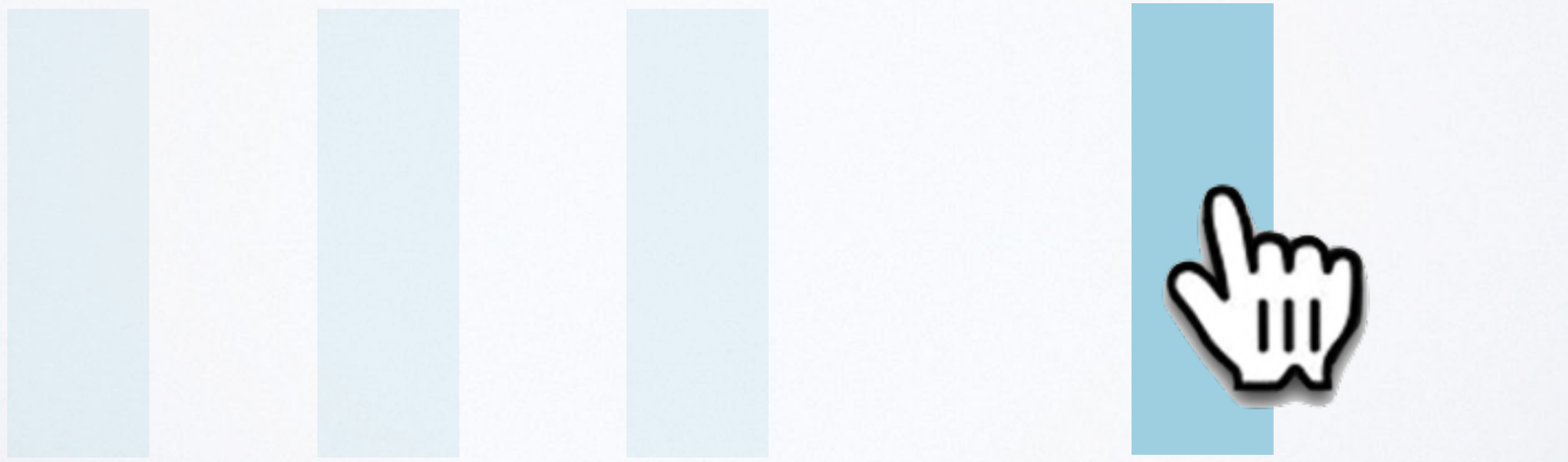
x0 ↦ 55, sep ↦ 50

x0 ↦ …, sep ↦ …

x0 ↦ 95

sep ↦ 52.5

x2’ = 155



$$155 = x0 + 2*sep$$

$$xi \mapsto \text{if } i=2 \text{ then } 155 \text{ else } x0 + i * sep$$

$$x0 \mapsto 75,\ sep \mapsto 40$$

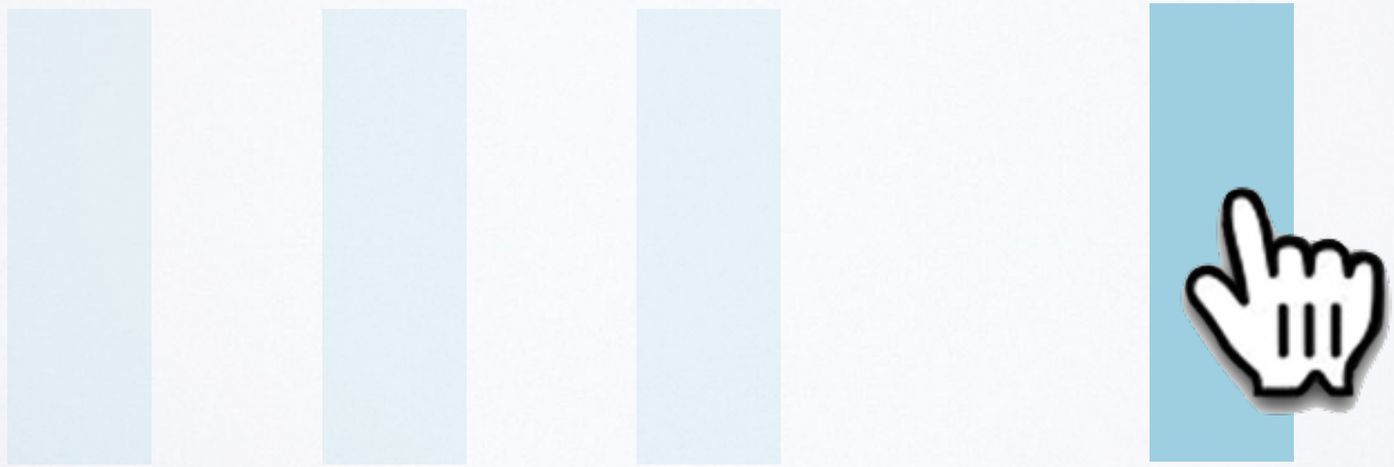
$$x0 \mapsto 55,\ sep \mapsto 50$$

$$x0 \mapsto \ldots,\ sep \mapsto \ldots$$

$$x0 \mapsto 95$$

$$sep \mapsto 52.5$$

$$x2' = 155$$



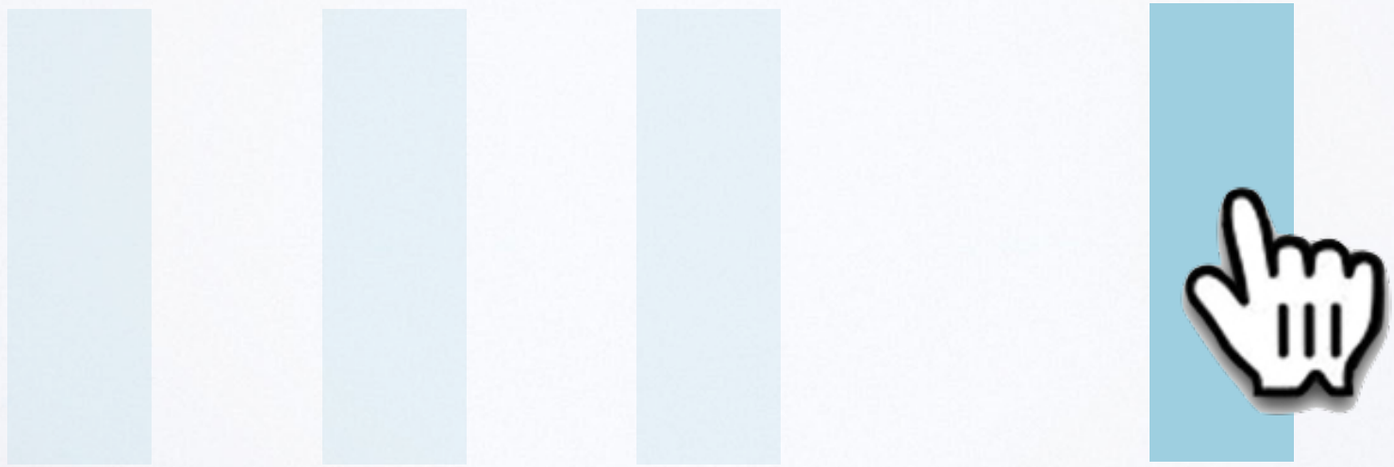
$$155 = x0 + 2*sep$$

“Small Updates”

$x0 \mapsto 95$

$sep \mapsto 52.5$

$x2' = 155$



x0 ↦ 95

?

sep ↦ 52.5

x2’ = 155

x0 H

52.5

GOAL

Automatic

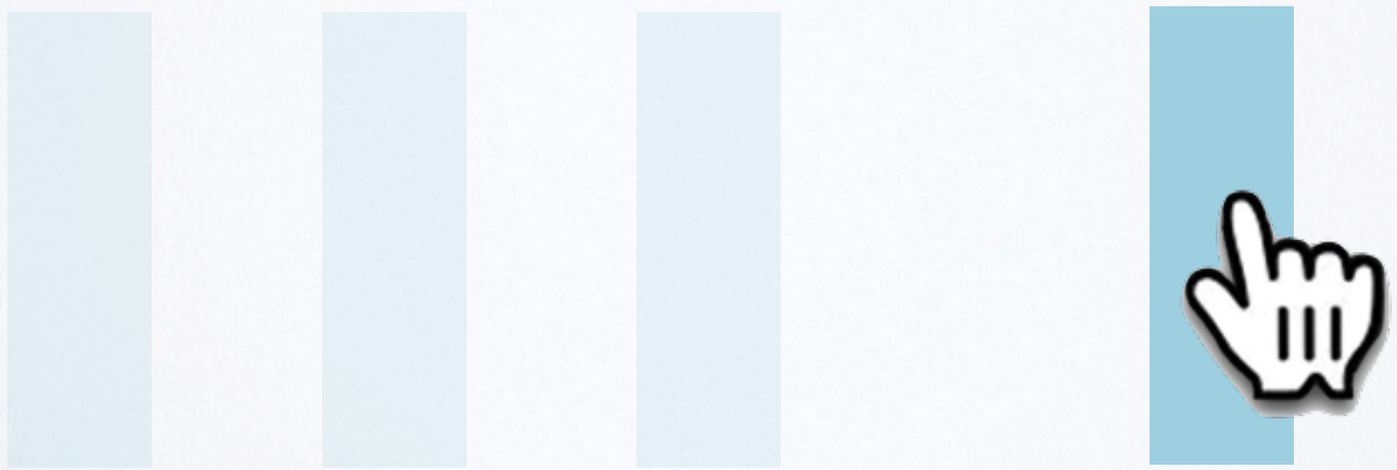
x2’ = 155

A HEURISTIC

`x0 ↦ 95`

`sep ↦ 52.5`

`x2’ = 155`

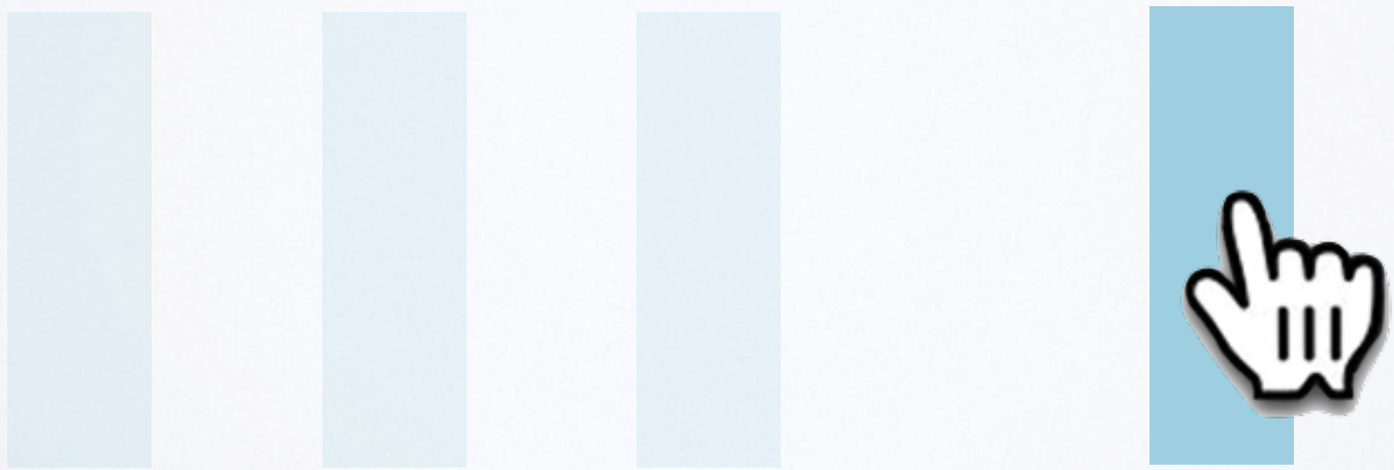


A HEURISTIC

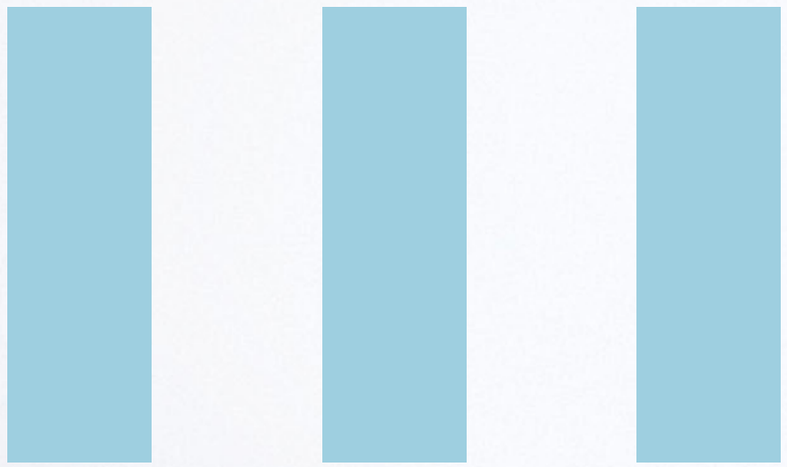
x0 ↦ 95

sep ↦ 52.5

x2’ = 155



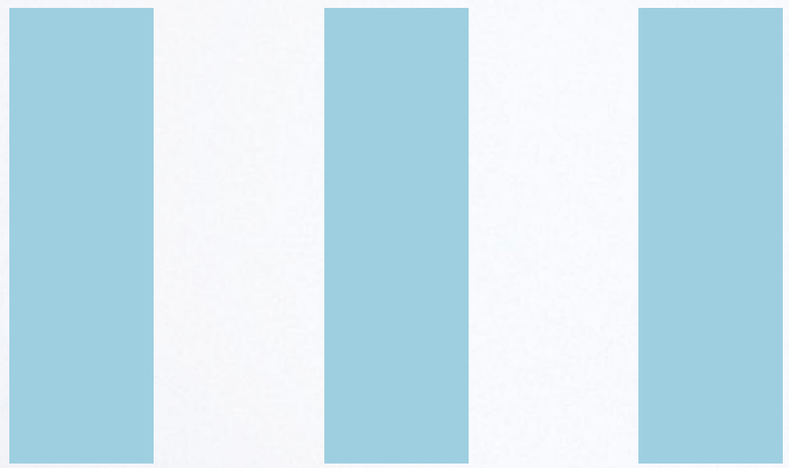
A HEURISTIC



A HEURISTIC

`x0 ↦ …`

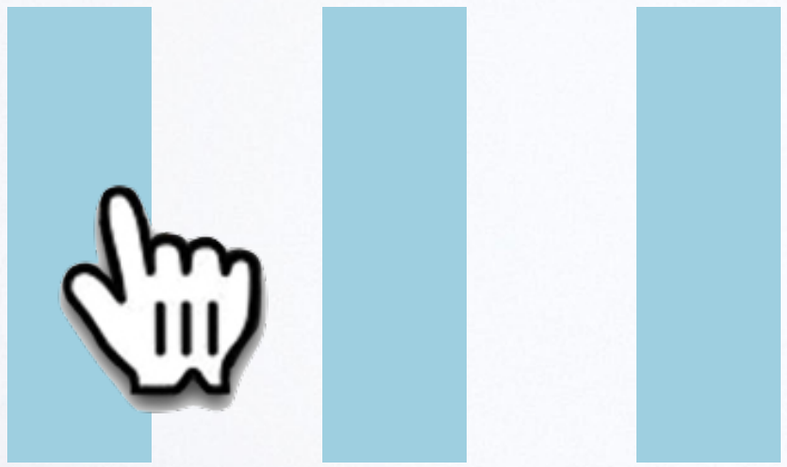
`sep ↦ …`



A HEURISTIC

```
x0 ↦ …
```

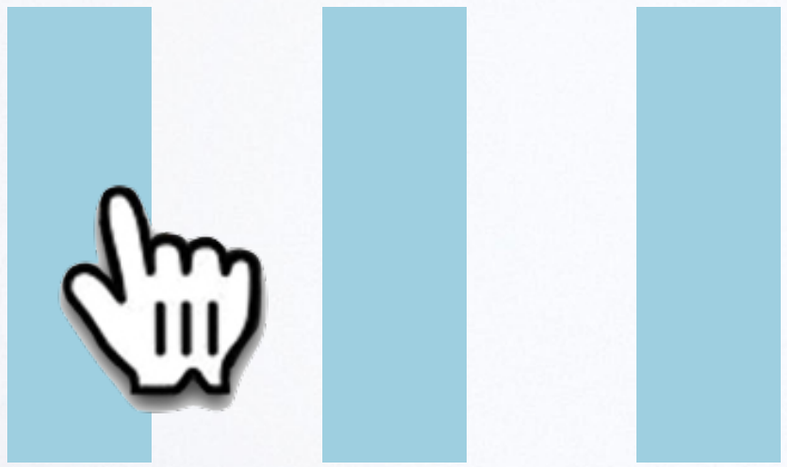
```
sep ↦ …
```



A HEURISTIC

x0 ↦ …

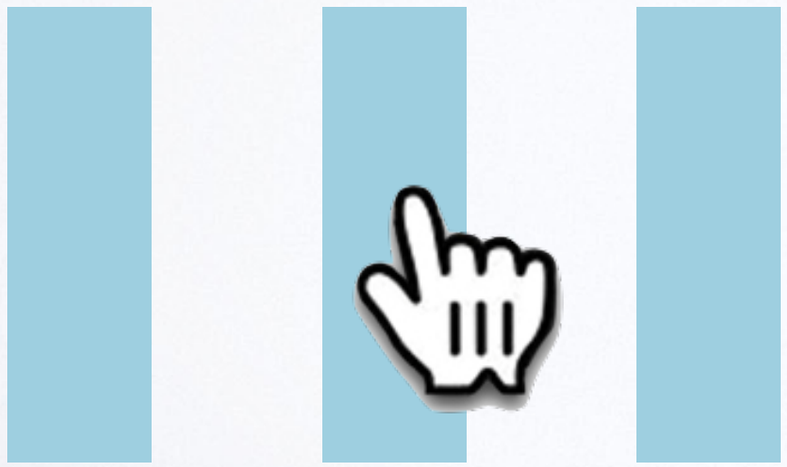
sep ↦ …



A HEURISTIC

x0 ↦ …

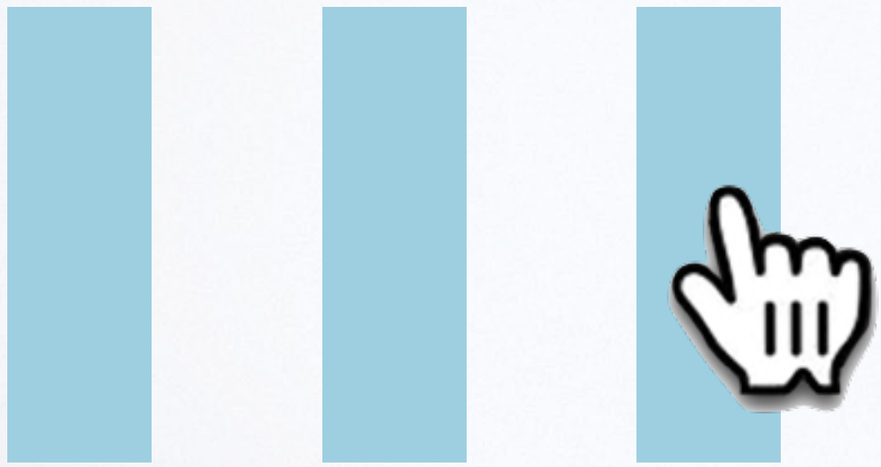
sep ↦ …



A HEURISTIC

x0 ↦ …

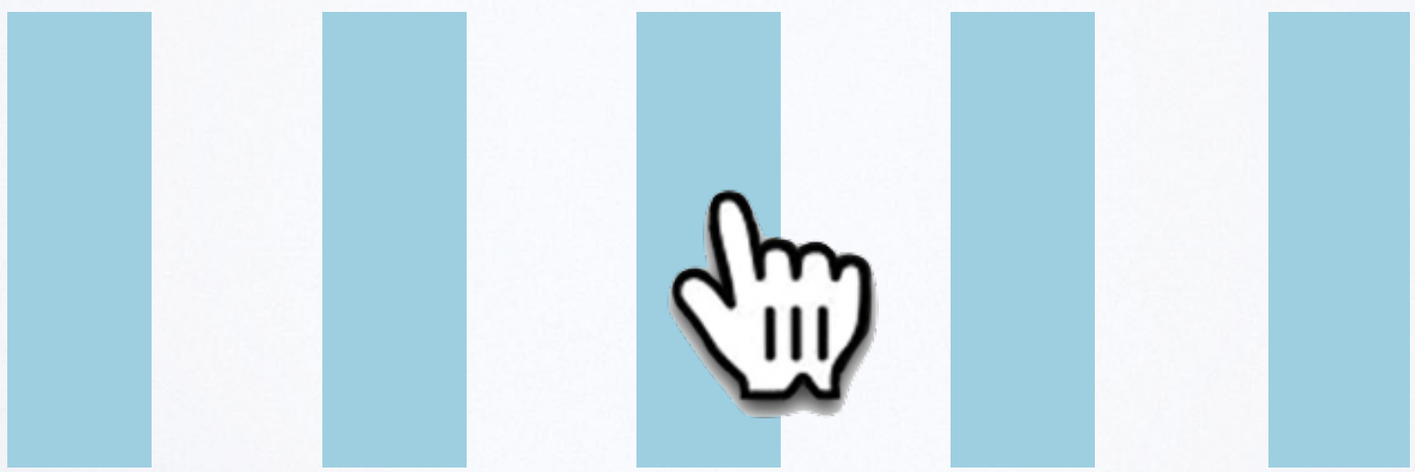
sep ↦ …



A HEURISTIC

x0 ↦ …

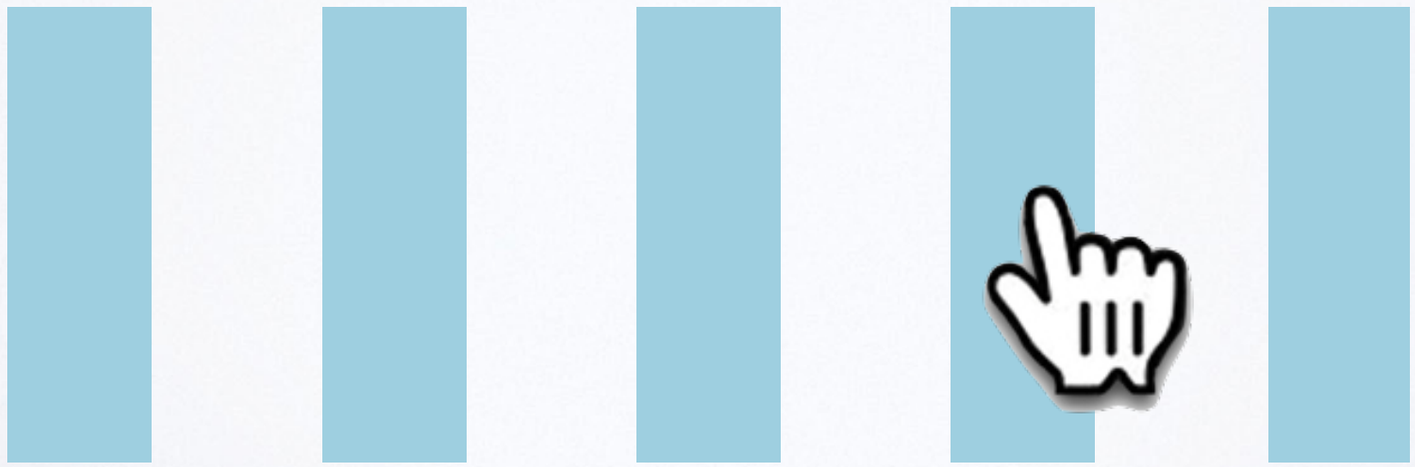
sep ↦ …



A HEURISTIC

x0 ↦ …

sep ↦ …

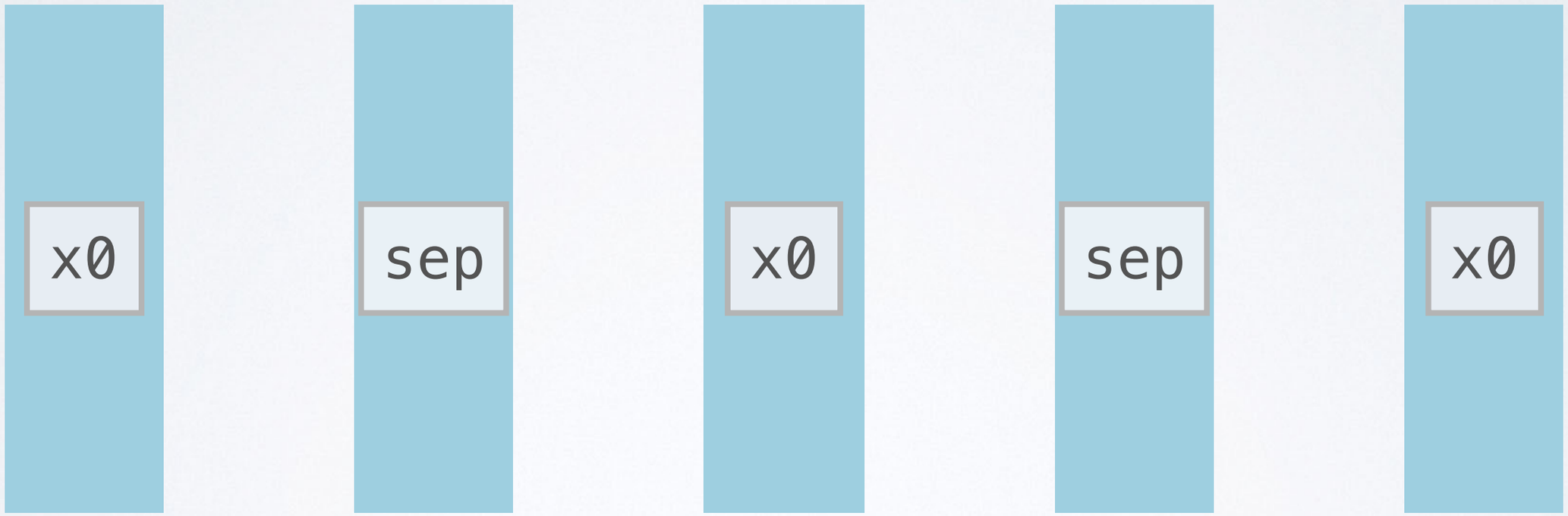


A HEURISTIC

x0 ↦ …

sep ↦ …

A HEURISTIC



Small Updates

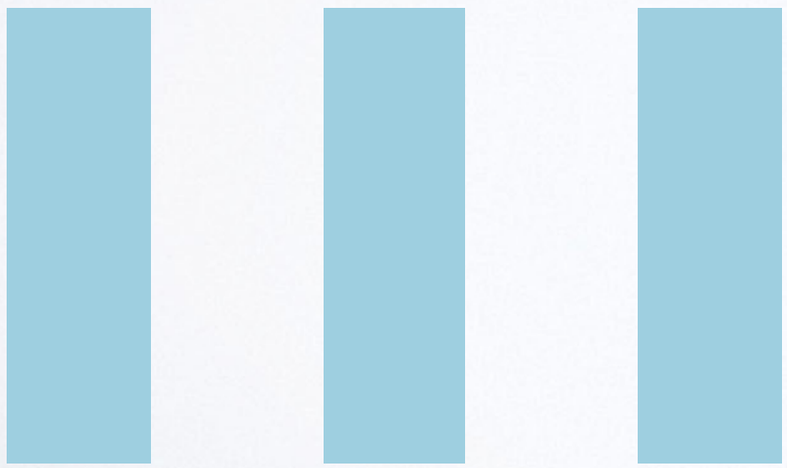
Heuristics

Traces

Simple Solver

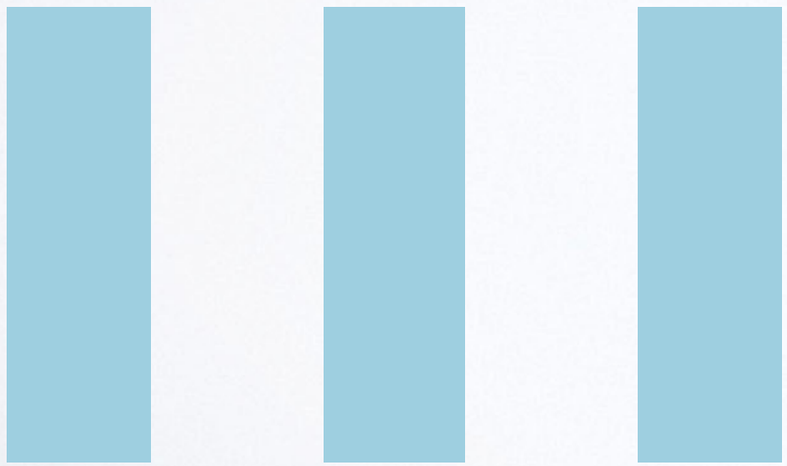
```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

x2 = 110



```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

x2 = 110



TRACES

```
let a = 3 in
let b = 5 in
a + b
```

TRACES

```
let a = 3 in
let b = 5 in
a + b
```

⇓

8

TRACES

```
let a = 3ᵃ in
let b = 5ᵇ in
```

a + b

⇓

8

TRACES

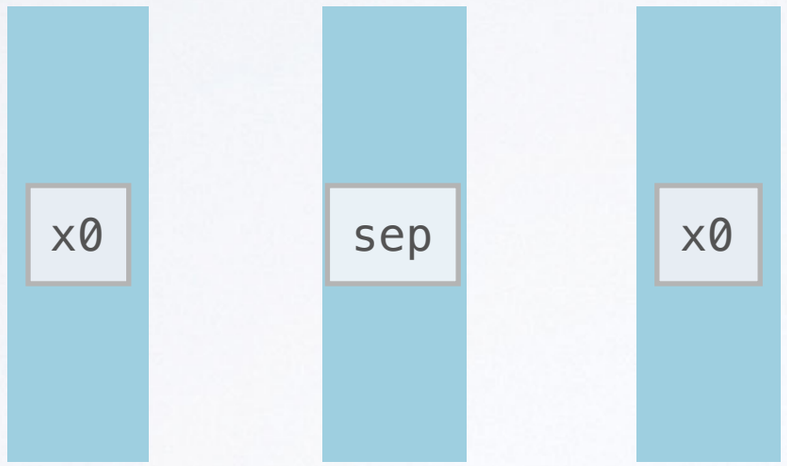
```
let a = 3ᵃ in
let b = 5ᵇ in
```

a + b

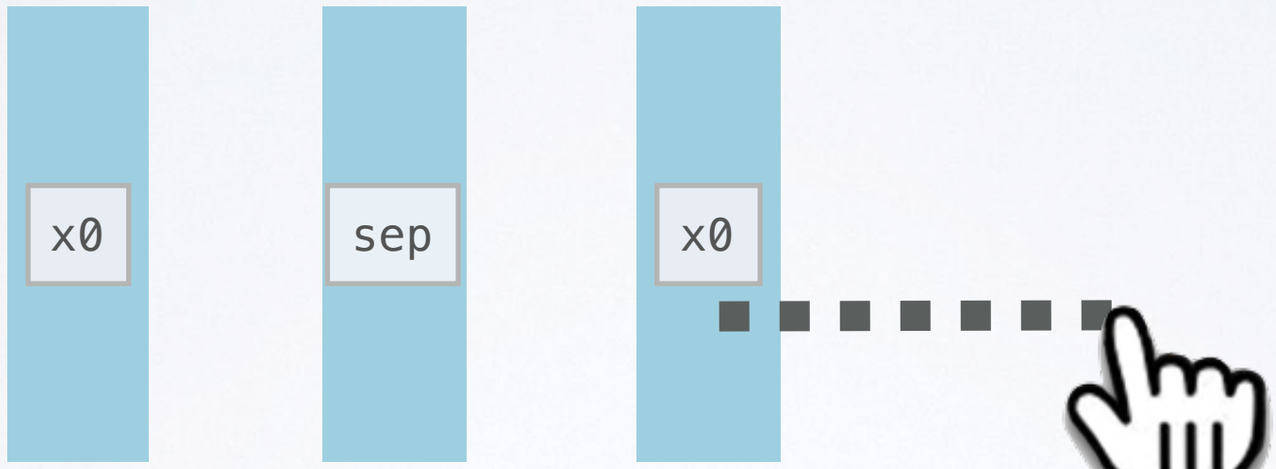
⇓

8^{a+b}

$$x2 = 110^{x0 + 2*sep}$$

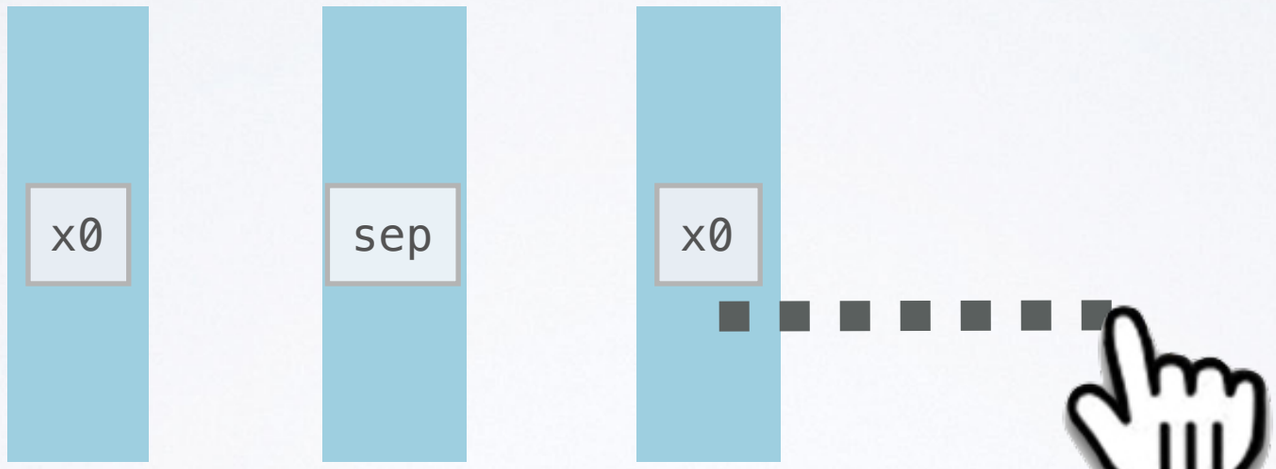


$$x2 = 110^{x0 + 2*sep}$$

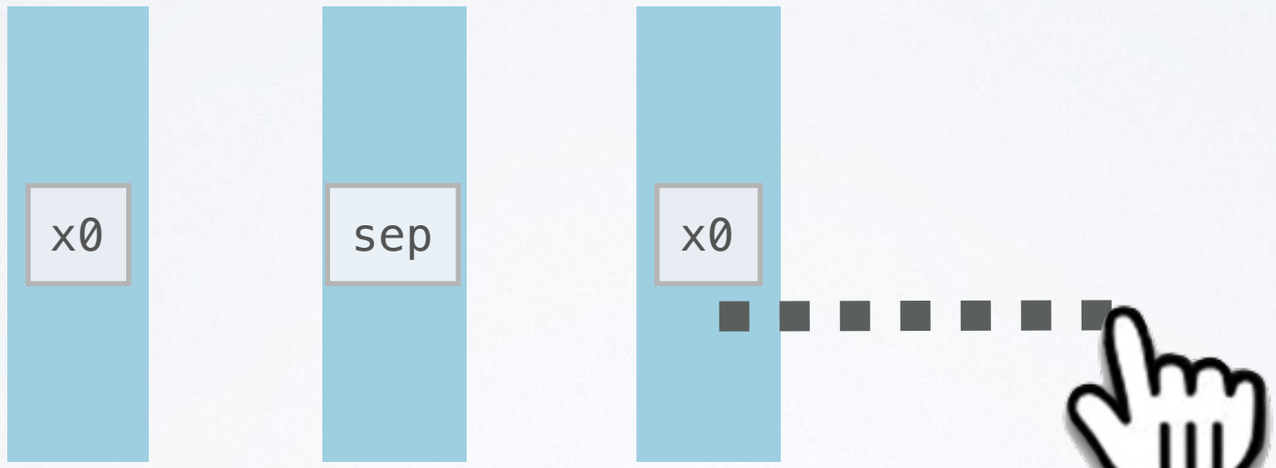


=

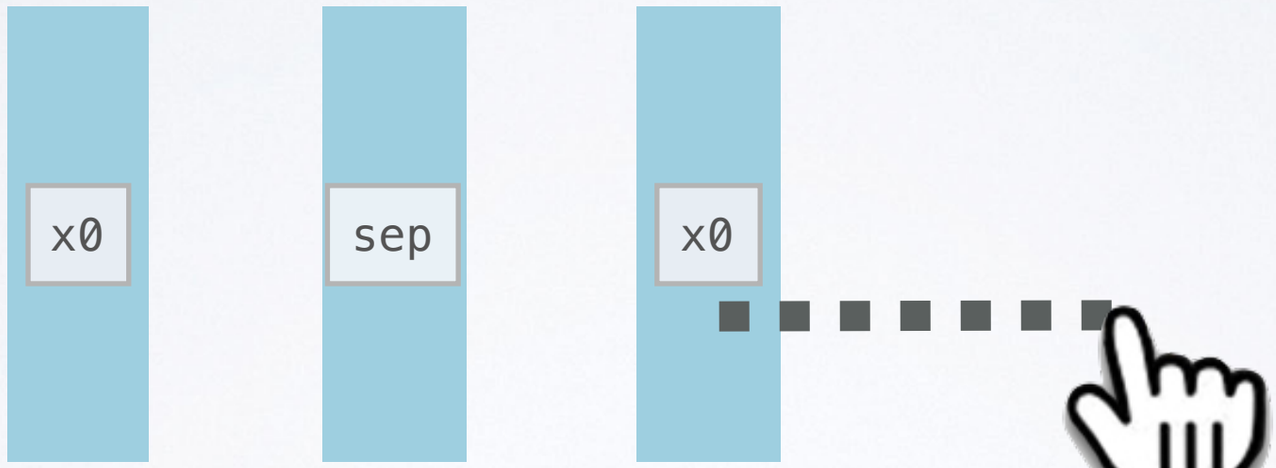
x2 = 110$^{x0 + 2*sep}$



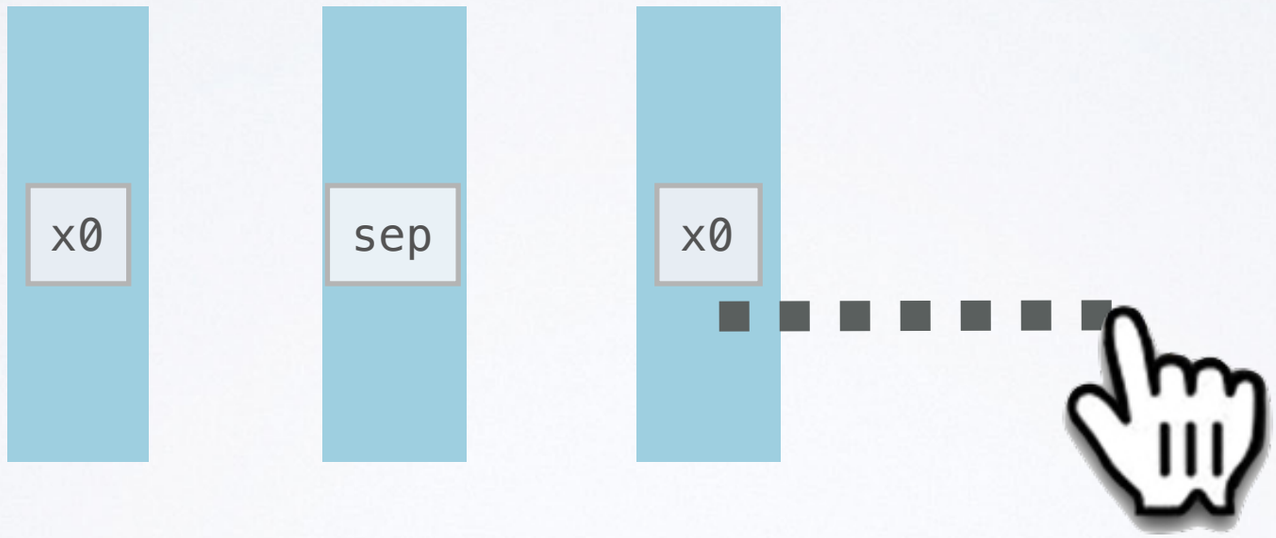
= x0 + 2*sep



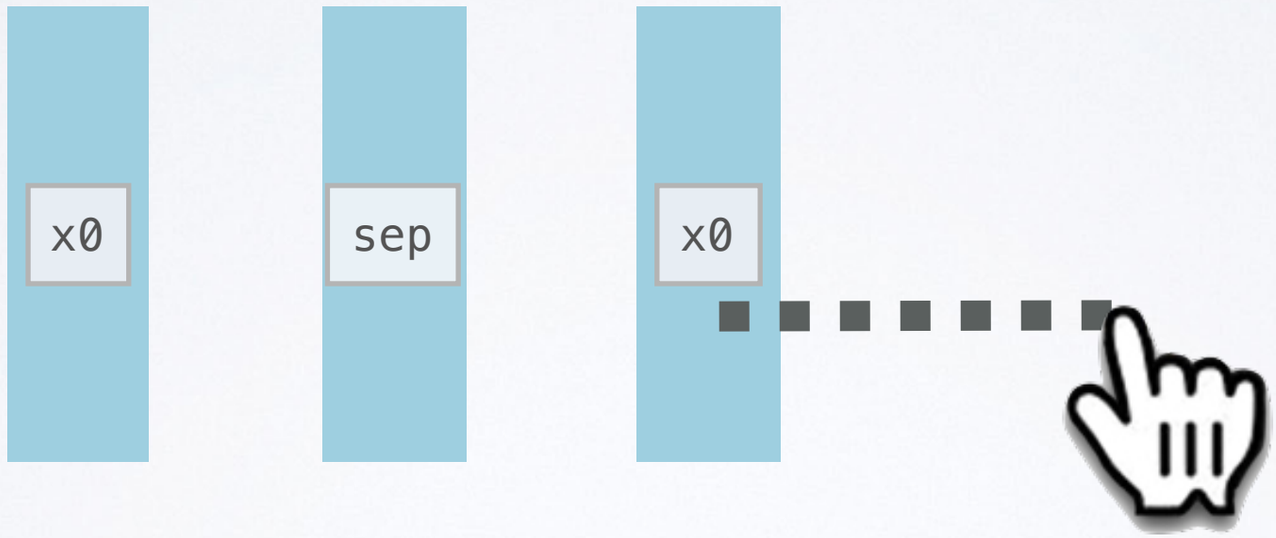
$$155 = x0 + 2*sep$$



$$155 = \square^{x0} + 2*sep$$

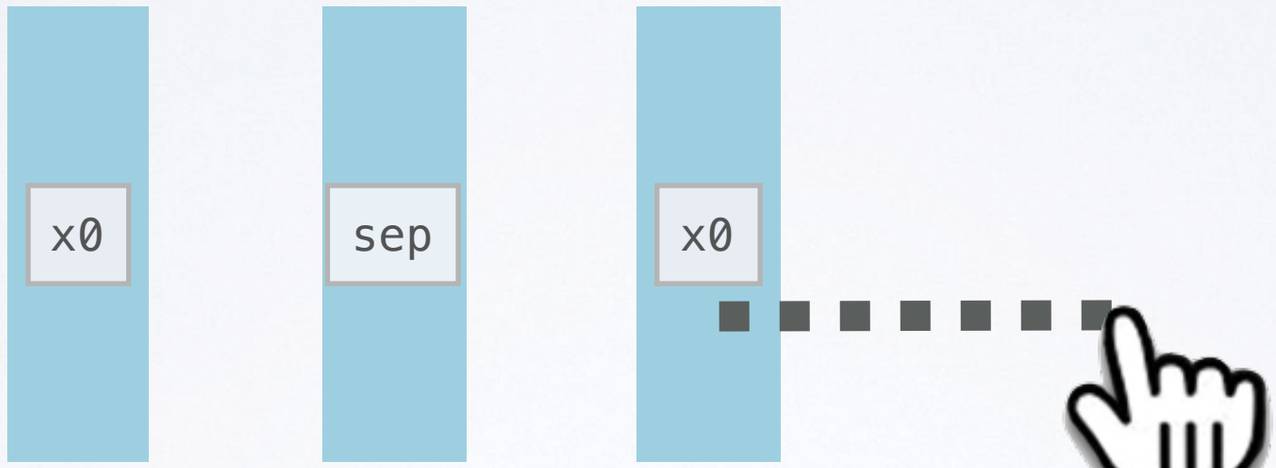


$$155 = \square^{x0} + 2*30$$

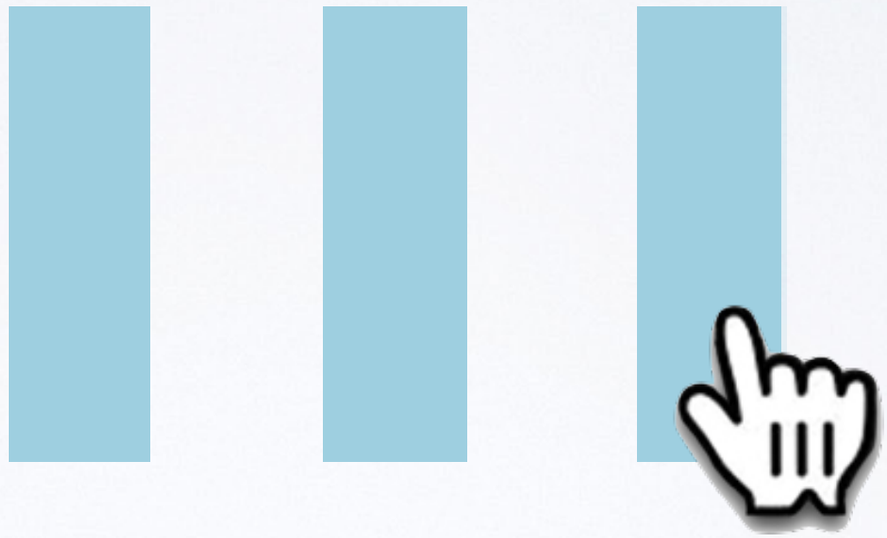


x0

155 = 95 + 2*30



$$155 = \overset{x0}{\boxed{95}} + 2*30$$



TRACE LANGUAGE

$$e \Downarrow n^t$$

TRACE LANGUAGE

$$e \Downarrow n^{t}$$

$$
\begin{aligned}
t \ ::= \ & x \\
\mid \ & t_1 + t_2 \\
\mid \ & t_1 * t_2 \\
\mid \ & \sin t \\
\mid \ & \mathrm{pow}\ t_1\ t_2 \\
& \ldots
\end{aligned}
$$

TRACE LANGUAGE

$$e \Downarrow n^{t}$$

$$
\begin{aligned}
t \ ::= \ & x \\
| \ & t_1 + t_2 \\
| \ & t_1 * t_2 \\
| \ & \sin t \\
| \ & \mathrm{pow}\ t_1\ t_2 \\
& \ldots
\end{aligned}
$$

$$n' = t$$

TRACES

```
let a = 3ᵃ in
let b = 5ᵇ in
```

```
a + b
```

TRACES

```
let a = 3[a] in
let b = (true ? 5[b1] : 6[b2]) in
```

a + b

TRACES

$$\texttt{let a} = 3^{a} \texttt{ in}$$

$$\texttt{let b} = (\texttt{true} \texttt{ ? } 5^{b1} \texttt{ : } 6^{b2}) \texttt{ in}$$

$$a + b$$

$$\Downarrow$$

$$8^{a+b1}$$

$$
\begin{aligned}
t ::= {} & x \\
\mid {} & t_1 + t_2 \\
\mid {} & t_1 * t_2 \\
\mid {} & \sin t \\
\mid {} & \mathrm{pow}\ t_1\ t_2 \\
& \ldots
\end{aligned}
$$

SOLVER

Can solve for **x** if **x** only occurs once.

`5*sin(x) = n` ✓

x$*(y + y^2) = n$ ✓

$\mathbf{x}^2 = n$ ✓

SOLVER

Can solve for **x** if **x** only occurs once.

`5*sin(x) = n` ✓

`x*(y + y²) = n` ✓

`x*(x + y²) = n` ✗

`x² = n` ✓

`x*x = n` ✗

SOLVER

Can solve for **x** if **x** only occurs once.

$5*\sin(x) = n$ ✓

$x*(y + y^2) = n$ ✓

$x*(x + y^2) = n$ ✗

$x^2 = n$ ✓

$x*x = n$ ✗

Good for 80% of equations.

Small Updates

Heuristics

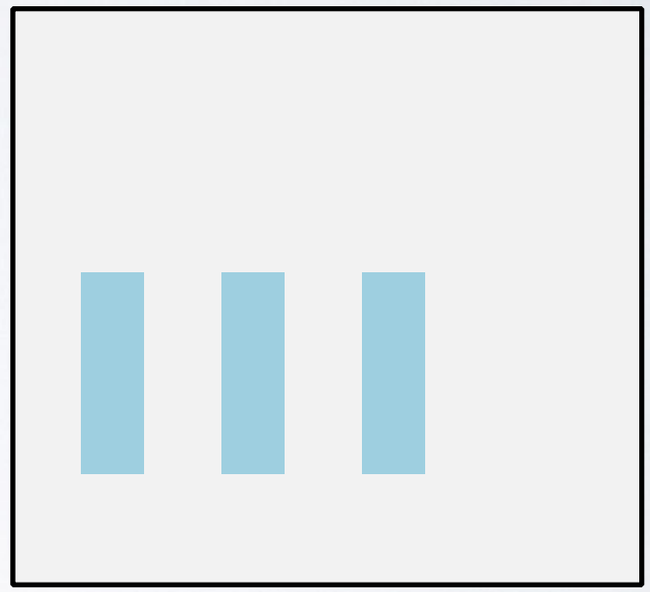
Traces

Simple Solver

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

1. Evaluate with **Traces**

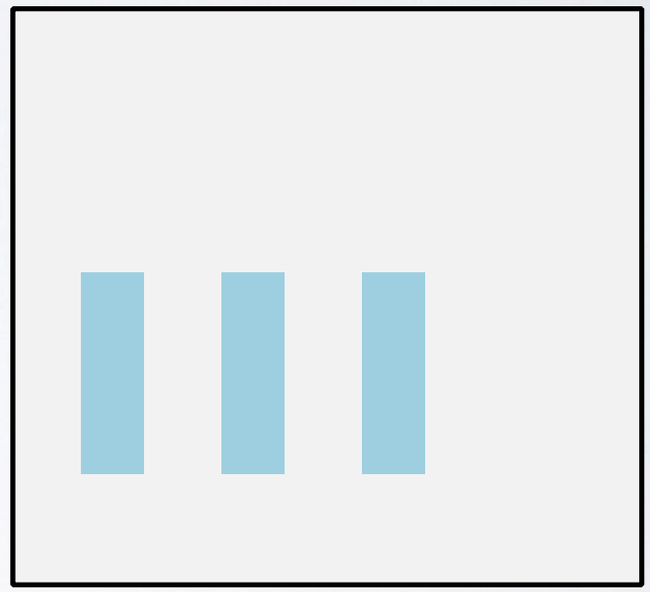
```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

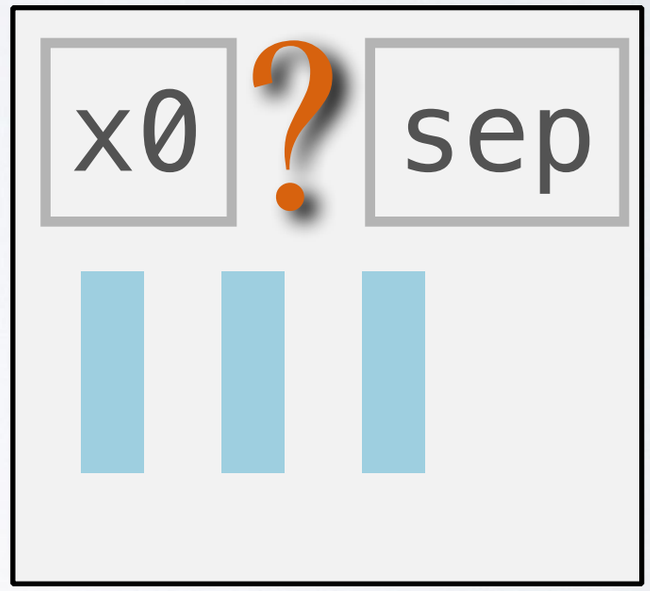
```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

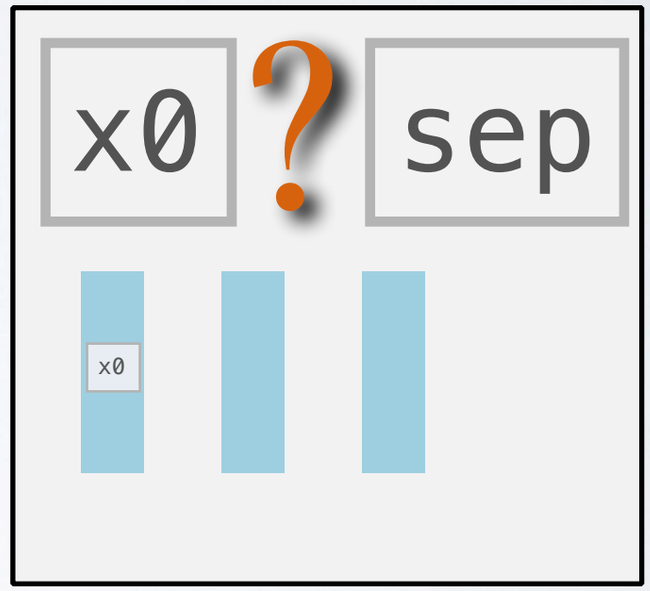
```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

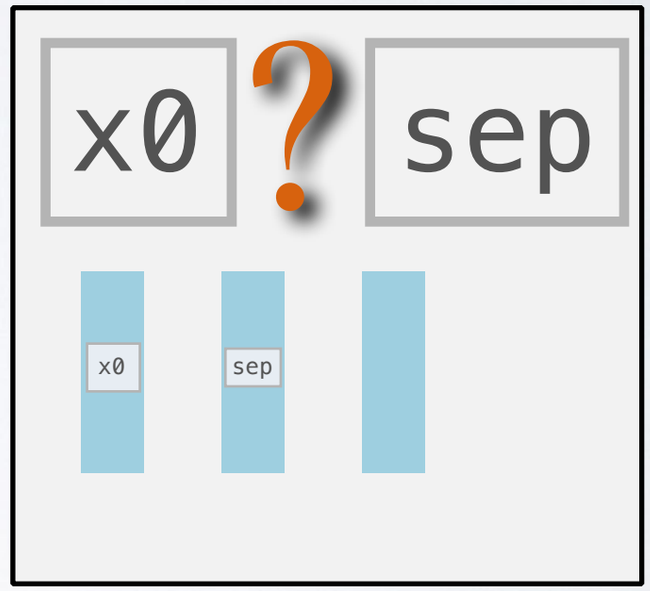
```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

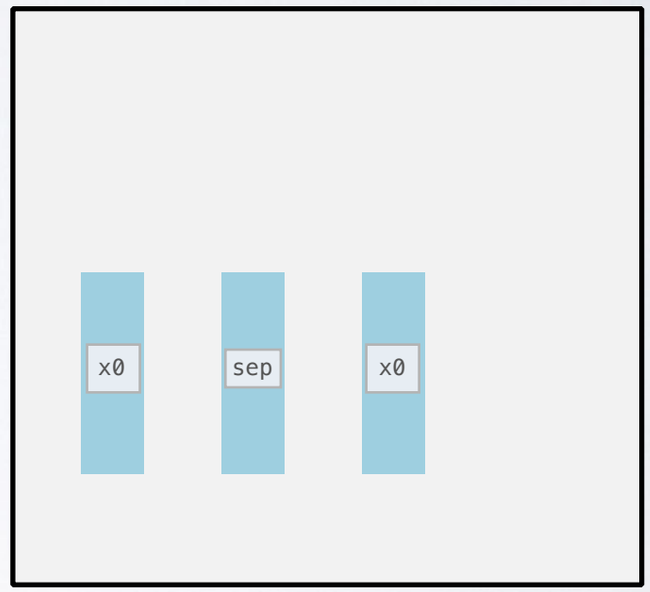
```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

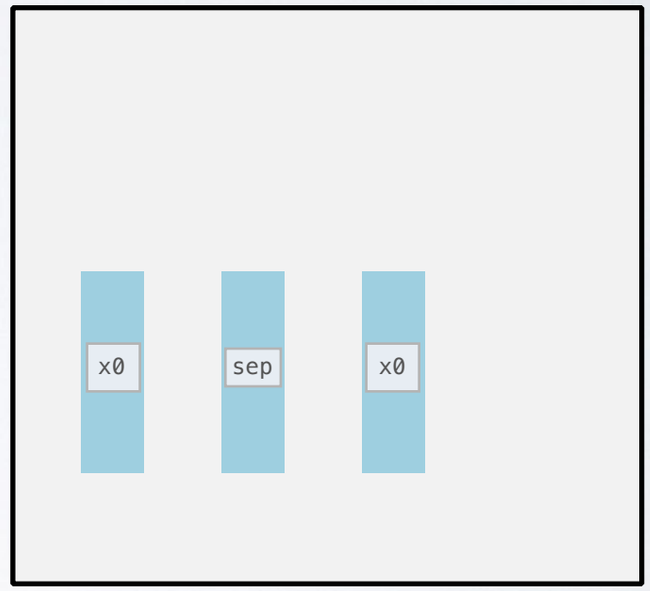


1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

3. Live Sync

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



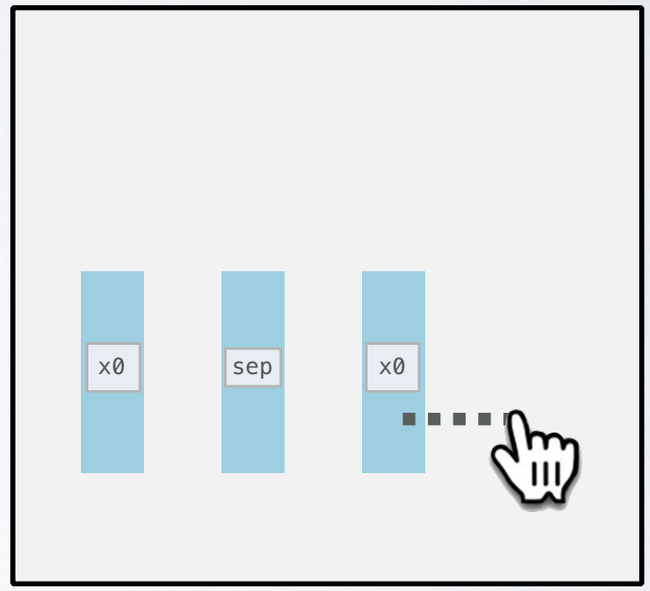
1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

3. Live Sync

User **Changes** Output;



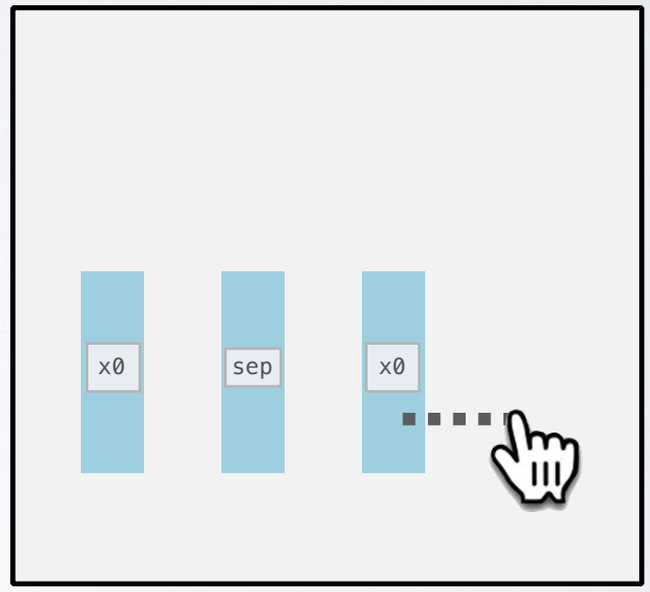
1. Evaluate with **Traces**

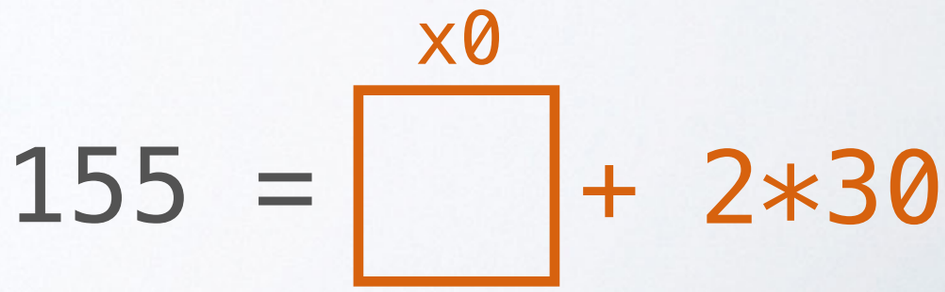
2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

3. Live Sync

User **Changes** Output;





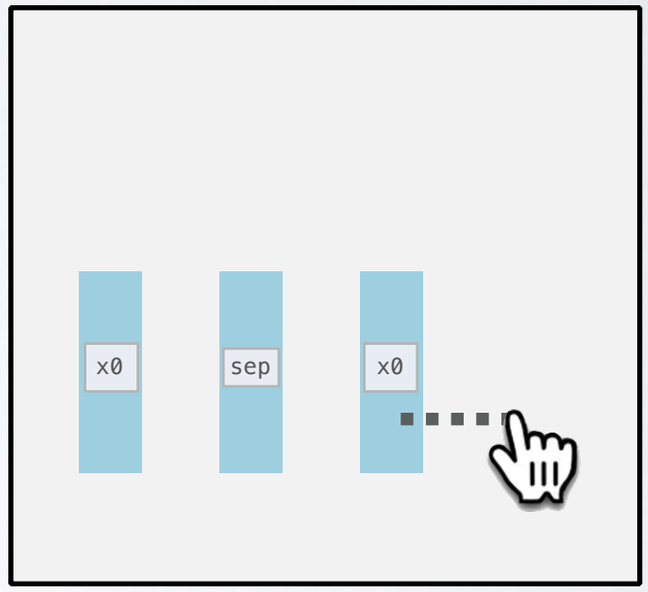
1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

3. Live Sync

User **Changes** Output;
Solve Trace Equation;



$$155 = \square + 2*30$$

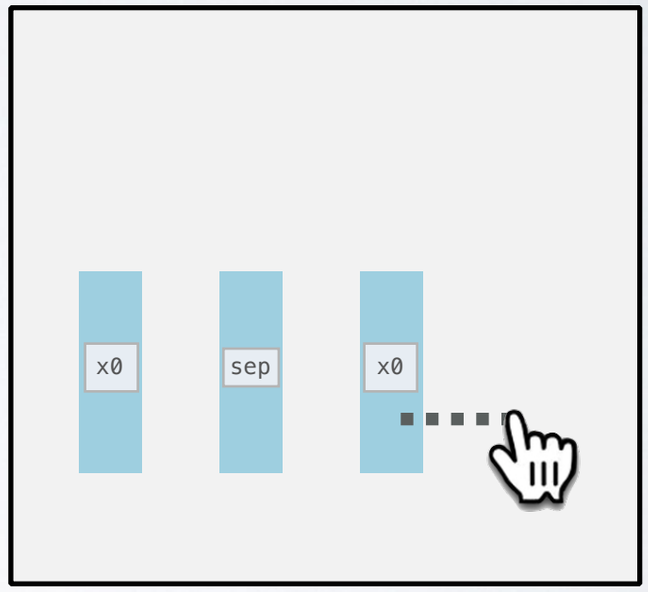
1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```

3. Live Sync

User **Changes** Output;
Solve Trace Equation;

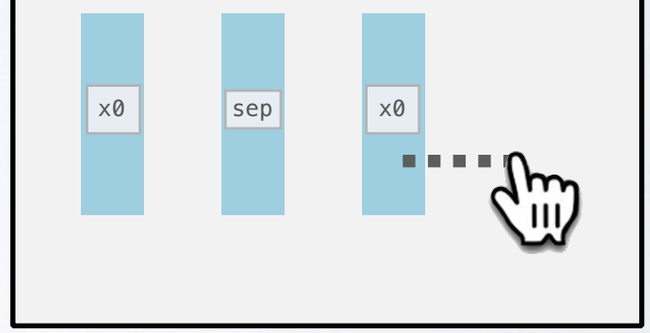


$$155 = \boxed{95}^{x0} + 2*30$$

1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (95, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



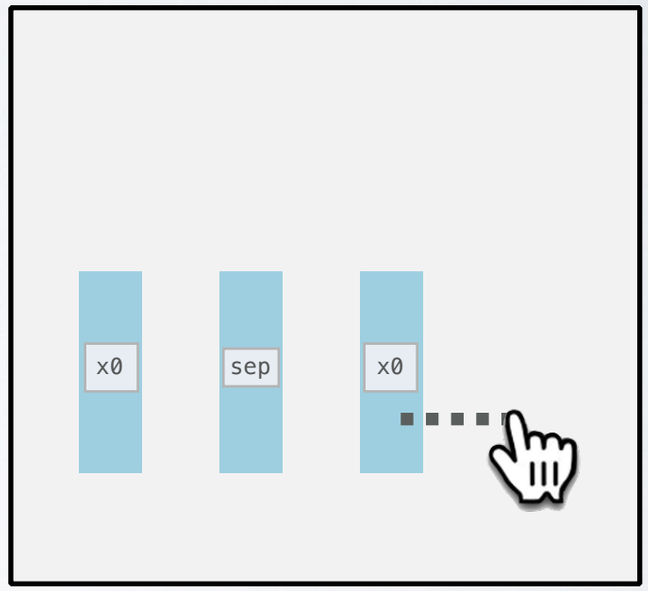
3. Live Sync

User **Changes** Output;
Solve Trace Equation;
Apply **Small Update**;

1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (95, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



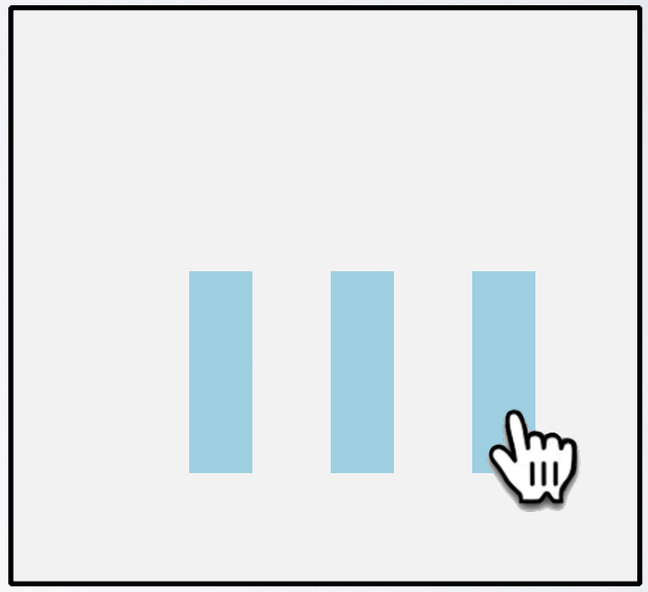
3. Live Sync

User **Changes** Output;
Solve Trace Equation;
Apply **Small Update**;
Re-run and Render

1. Evaluate with **Traces**

2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (95, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



3. Live Sync

User **Changes** Output;
Solve Trace Equation;
Apply **Small Update**;
Re-run and Render

Sketch-n-Sketch

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def [x0 y0  w  h  sep n]
     [50 120 20 90 30  3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

3 Boxes PLDI

Edit Code

Undo

Redo

[Zones] Hidden

[Widgets] Shown

[Out] Canvas

sketch-n-sketch v0.4.2

```
(def [x0 y0  w  h  sep n]
     [168 127 20 90 30  3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

3 Boxes PLDI

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown

[Heuristics] Fair

[Orientation] Vertical

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def [x0 y0  w  h  sep n]
     [69 128 20 90 76  3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

3 Boxes PLDI

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def [x0 y0  w  h  sep n]
     [210 126 20 90 34  3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

3 Boxes PLDI

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown

[Out] Canvas

sketch-n-sketch v0.4.2

```
(def [x0 y0  w  h  sep n]
     [53 126 20 90 34  3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

3 Boxes PLDI

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown

[Heuristics] Fair

[Orientation] Vertical

rect1 Interior (ACTIVE)
126{y0} 53{x0}

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def [x0 y0  w  h  sep n]
     [127 124 20 90 34  3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

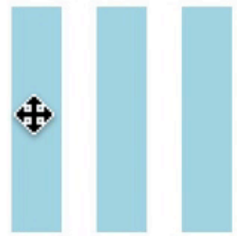
3 Boxes PLDI

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def [x0 y0  w  h  sep n]
     [76 115 20 90 45! 3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

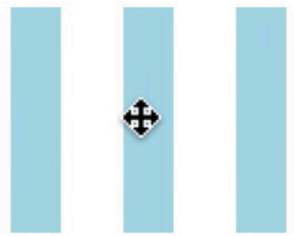
3 Boxes PLDI

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



rect2 Interior (ACTIVE)
115{y0} 76{x0}

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def [x0 y0  w  h  sep n]
     [42 118 17 131 45! 3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

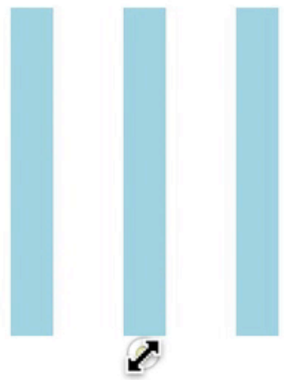
3 Boxes PLDI

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

```
(def [x0 y0  w  h  sep n]
     [32 118 27 92 45! 7{1-20}])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

3 Boxes PLDI

Edit Code

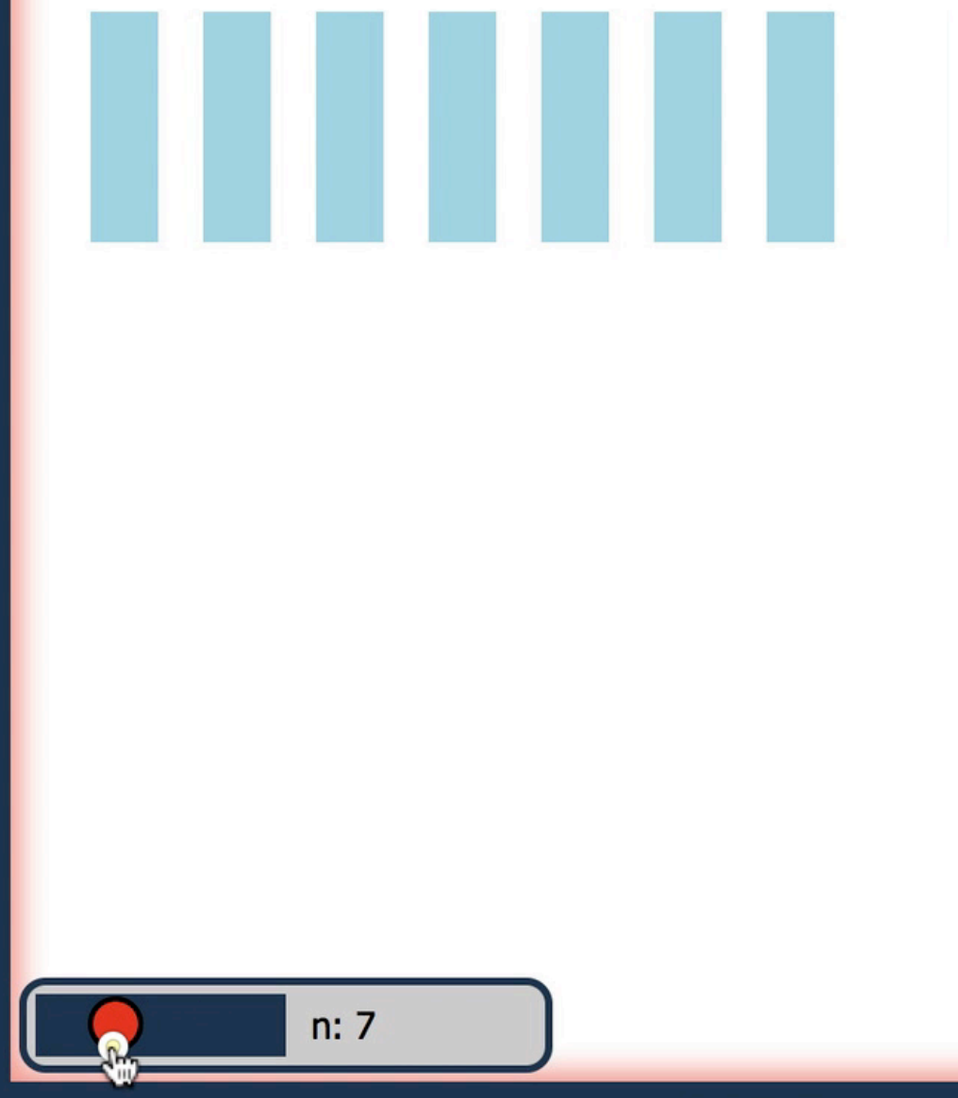
Undo Redo

[Zones] Hidden

[Widgets] Shown

[Heuristics] Fair

[Orientation] Vertical



[Out] Canvas

sketch-n-sketch v0.4.2

```
(def [x0 y0  w  h  sep amp n]
     [50 120 20 90 30  60  12!{1-30}])

(def twoPi (* 2! (pi)))

(def boxi (\i
  (let xi (+ x0 (* i sep))
  (let yi (- y0 (* amp (sin (* i (/ twoPi n)))))
  (rect 'lightblue' xi yi w h)))))

(svg (map boxi (zeroTo n)))
```

Wave Boxes F

Edit Code

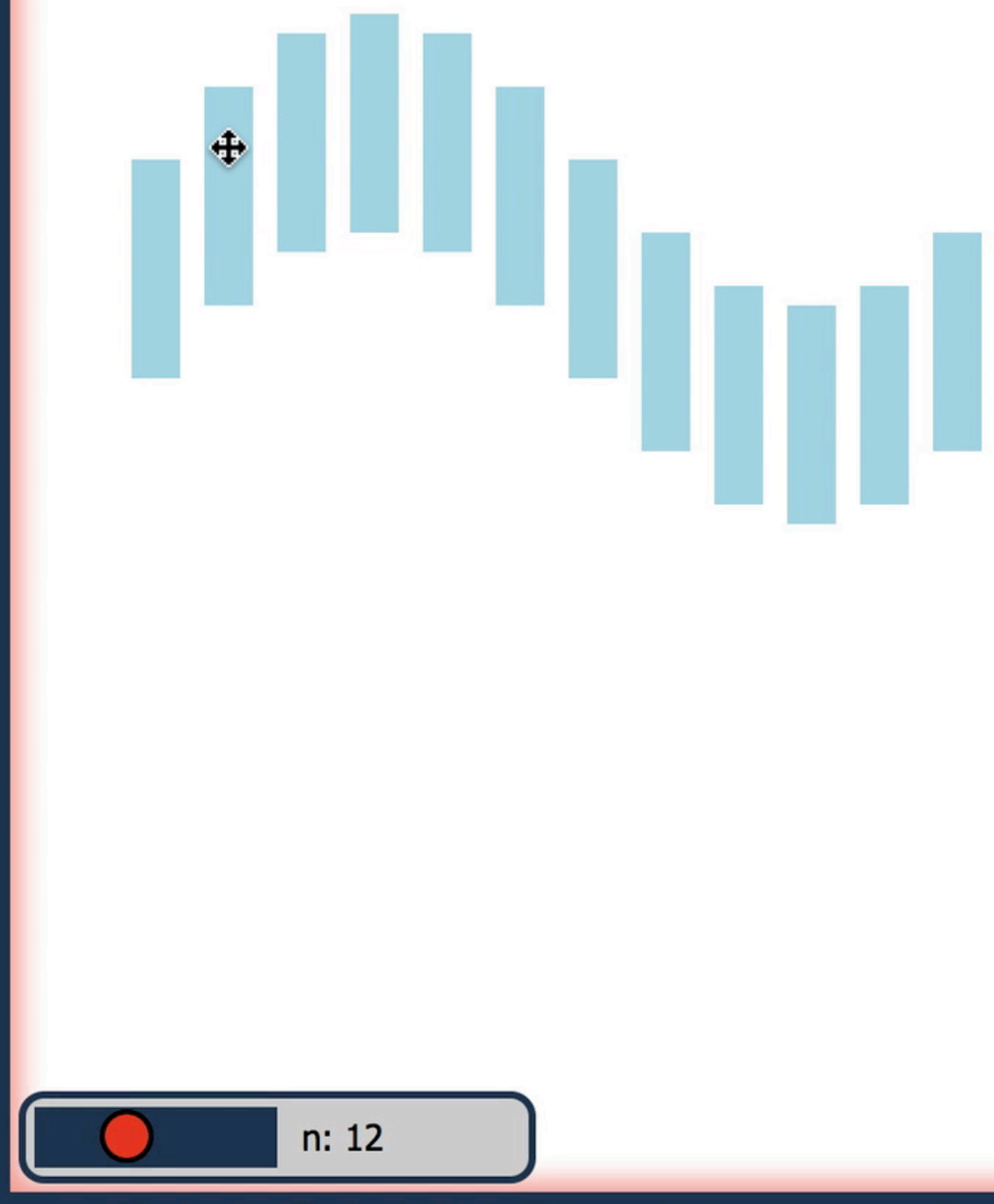
Undo Redo

[Zones] Hidden

[Widgets] Shown

[Heuristics] Fair

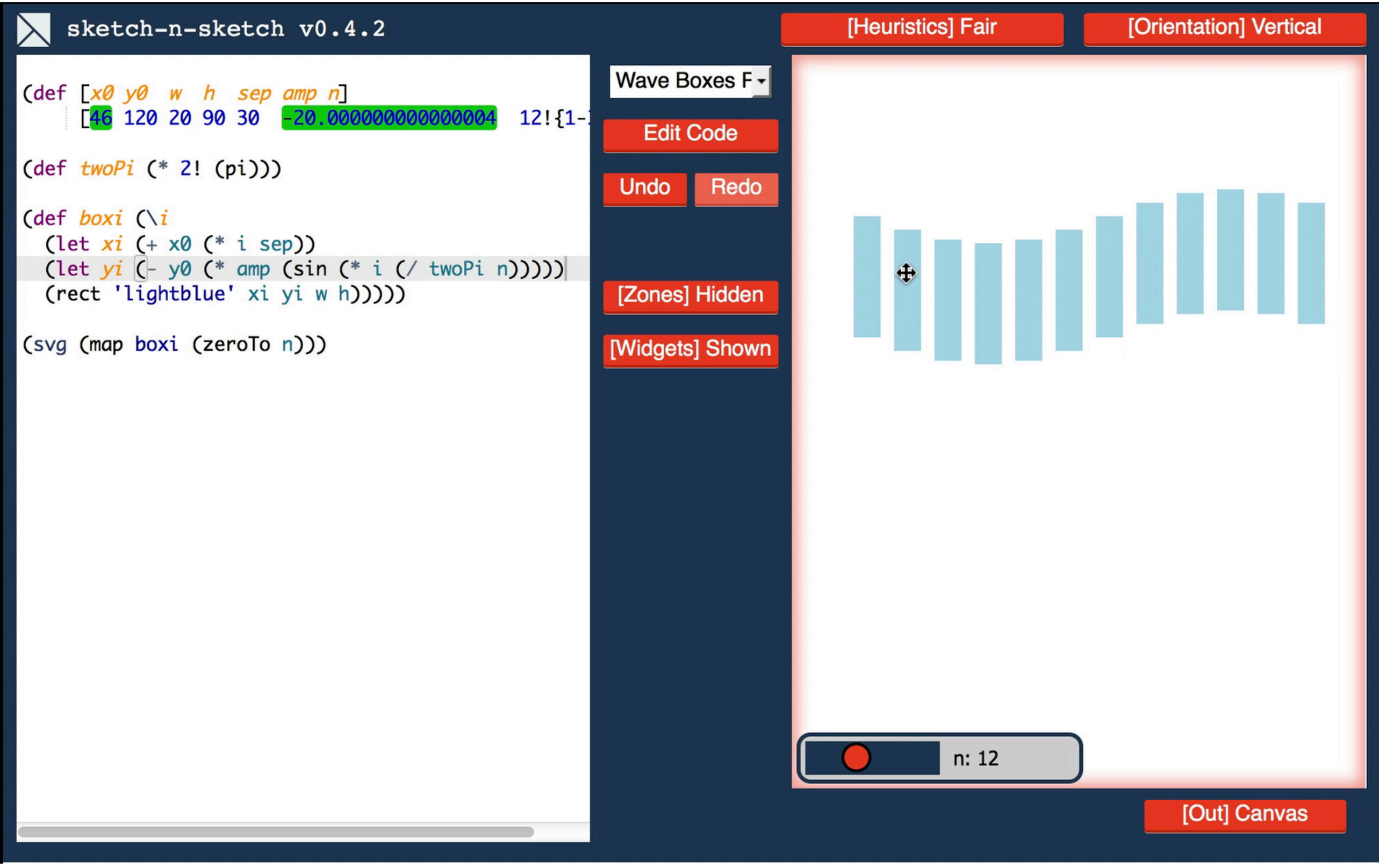
[Orientation] Vertical

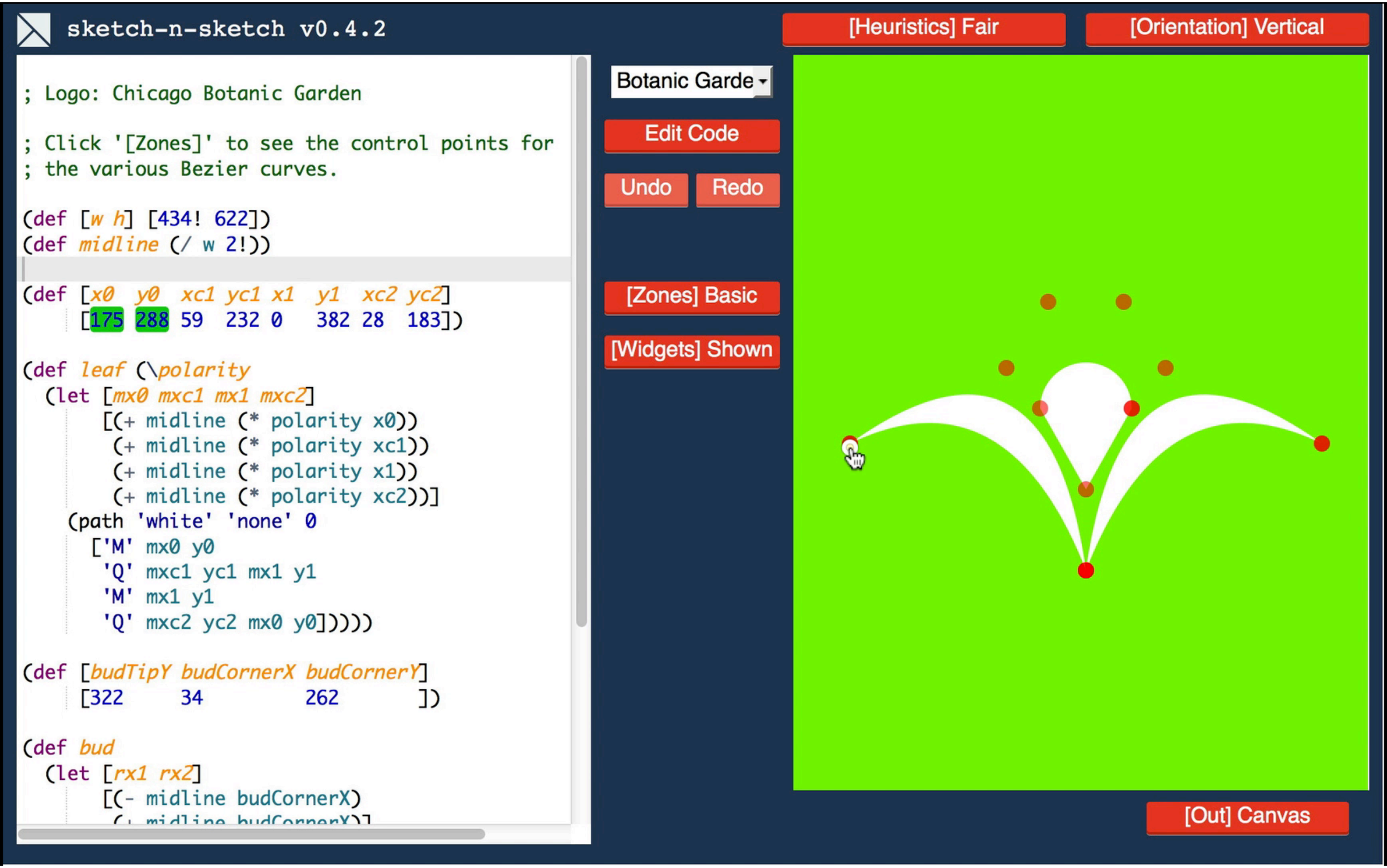


n: 12

rect2 Interior (ACTIVE)
60{amp} 50{x0}

[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
(def [x0 y0 w h sep amp n]
[46 120 20 90 30 -20.000000000000004 12!{1-
(def twoPi (* 2! (pi)))
(def boxi (\i
(let xi (+ x0 (* i sep))
(let yi (- y0 (* amp (sin (* i (/ twoPi n)))))
(rect 'lightblue' xi yi w h)))))
(svg (map boxi (zeroTo n)))
Wave Boxes F
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Shown
n: 12
[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; Logo: Chicago Botanic Garden
; Click '[Zones]' to see the control points for
; the various Bezier curves.
(def [w h] [434! 622])
(def midline (/ w 2!))
(def [x0 y0 xc1 yc1 x1 y1 xc2 yc2]
[175 288 59 232 0 382 28 183])
(def leaf (\polarity
(let [mx0 mxc1 mx1 mxc2]
[(+ midline (* polarity x0))
(+ midline (* polarity xc1))
(+ midline (* polarity x1))
(+ midline (* polarity xc2))]
(path 'white' 'none' 0
['M' mx0 y0
'Q' mxc1 yc1 mx1 y1
'M' mx1 y1
'Q' mxc2 yc2 mx0 y0]))))
(def [budTipY budCornerX budCornerY]
[322 34 262])
(def bud
(let [rx1 rx2]
[(- midline budCornerX)
Botanic Garde
Edit Code
Undo
Redo
[Zones] Basic
[Widgets] Shown
[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Logo: Chicago Botanic Garden

; Click '[Zones]' to see the control points for
; the various Bezier curves.

(def [w h] [434! 622])
(def midline (/ w 2!))

(def [x0  y0  xc1 yc1 x1  y1  xc2 yc2]
     [143 184 59  232 0   382 28  183])

(def leaf (\polarity
  (let [mx0 mxc1 mx1 mxc2]
       [(+ midline (* polarity x0))
        (+ midline (* polarity xc1))
        (+ midline (* polarity x1))
        (+ midline (* polarity xc2))]
    (path 'white' 'none' 0
      ['M' mx0 y0
       'Q' mxc1 yc1 mx1 y1
       'M' mx1 y1
       'Q' mxc2 yc2 mx0 y0]))))

(def [budTipY budCornerX budCornerY]
     [322     34         262       ])

(def bud
  (let [rx1 rx2]
       [(- midline budCornerX)
```

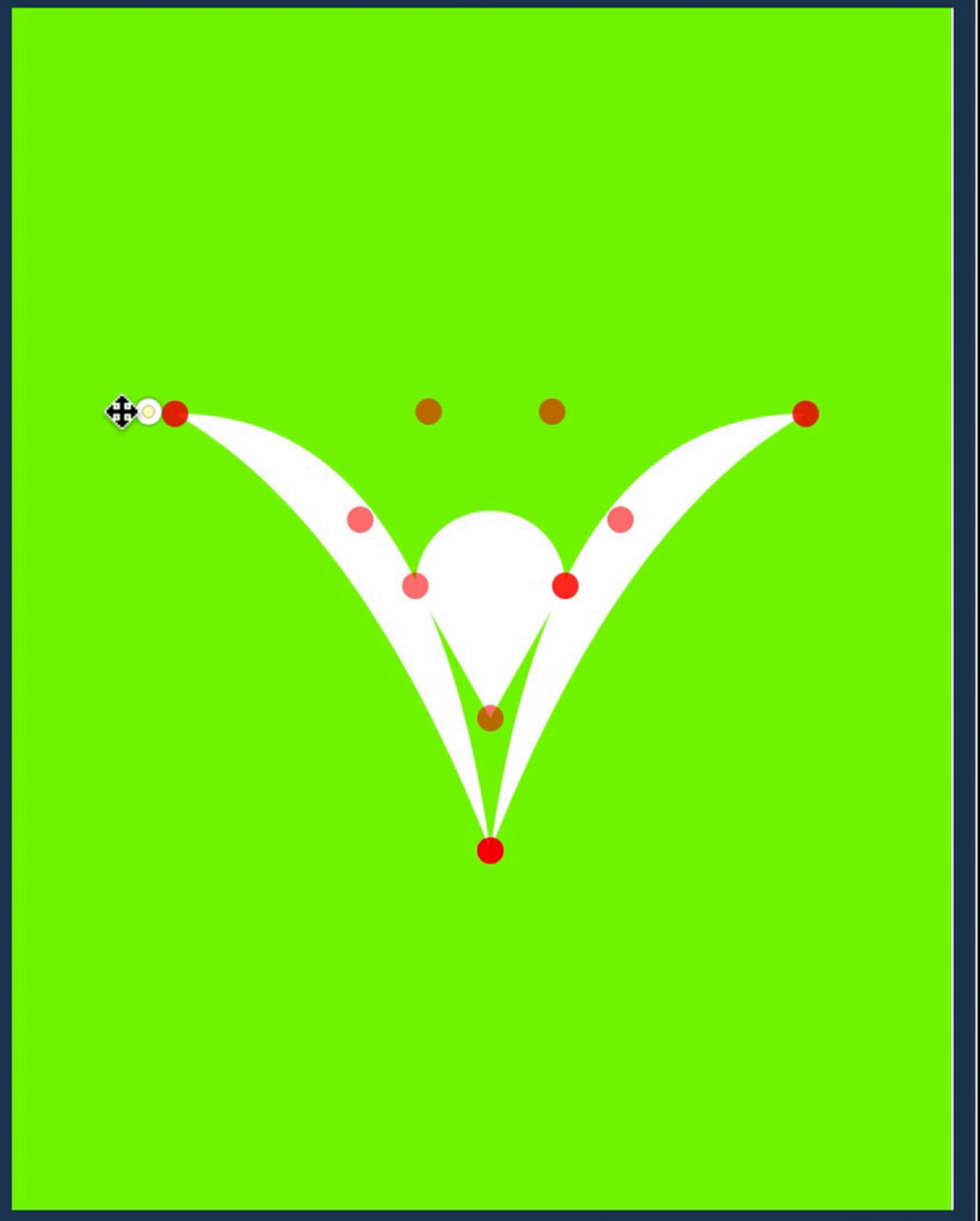
Botanic Garde

Edit Code

Undo Redo

[Zones] Basic

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; sketch-n-sketch logo
;
(let [x0 y0 w h delta] [50 50 200 200 10]
(let [xw yh w2 h2] [(+ x0 w) (+ y0 h) (div w 2) (d
(let poly (\pts (polygon 'black' 'none' 0 pts))
(svg [
  (rect 'white' x0 y0 w h)
  (poly
    [[(+ x0 delta) y0]
     [xw y0]
     [xw (- yh delta)]])
  (poly
    [[x0 (+ y0 delta)]
     [x0 (- yh delta)]
     [(- (+ x0 w2) delta) (+ y0 h2)]])
  (poly
    [[(+ x0 delta) yh]
     [(- xw delta) yh]
     [(+ x0 w2) (+ (+ y0 h2) delta)]])
])))))
```

Logo

Edit Code

Undo

Redo

[Zones] Hidden

[Widgets] Shown

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; sketch-n-sketch logo
;
(let [x0 y0 w h delta] [1 196 200 200 10]
(let [xw yh w2 h2] [(+ x0 w) (+ y0 h) (div w 2) (d
(let poly (\pts (polygon 'black' 'none' 0 pts))
(svg [
  (rect 'white' x0 y0 w h)
  (poly
    [[(+ x0 delta) y0]
     [xw y0]
     [xw (- yh delta)]])
  (poly
    [[x0 (+ y0 delta)]
     [x0 (- yh delta)]
     [(- (+ x0 w2) delta) (+ y0 h2)]])
  (poly
    [[(+ x0 delta) yh]
     [(- xw delta) yh]
     [(+ x0 w2) (+ (+ y0 h2) delta)]])
])))))
```

Logo

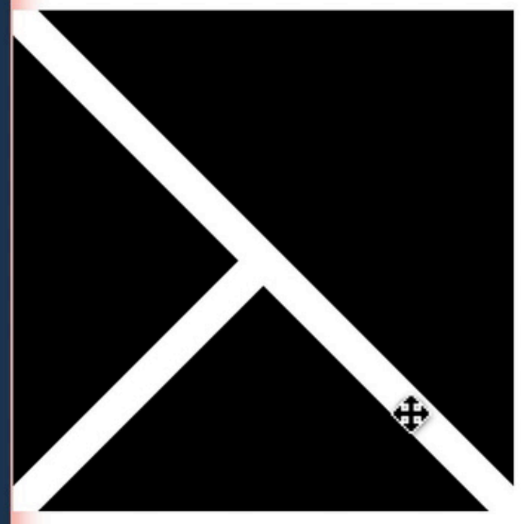
Edit Code

Undo

Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; sketch-n-sketch logo
;
(let [x0 y0 w h delta] [35 39 265 299 10]
(let [xw yh w2 h2] [(+ x0 w) (+ y0 h) (div w 2) (d
(let poly (\pts (polygon 'black' 'none' 0 pts))
(svg [
  (rect 'white' x0 y0 w h)
  (poly
    [[(+ x0 delta) y0]
     [xw y0]
     [xw (- yh delta)]])
  (poly
    [[x0 (+ y0 delta)]
     [x0 (- yh delta)]
     [(- (+ x0 w2) delta) (+ y0 h2)]])
  (poly
    [[(+ x0 delta) yh]
     [(- xw delta) yh]
     [(+ x0 w2) (+ (+ y0 h2) delta)]])
])))
```

Logo

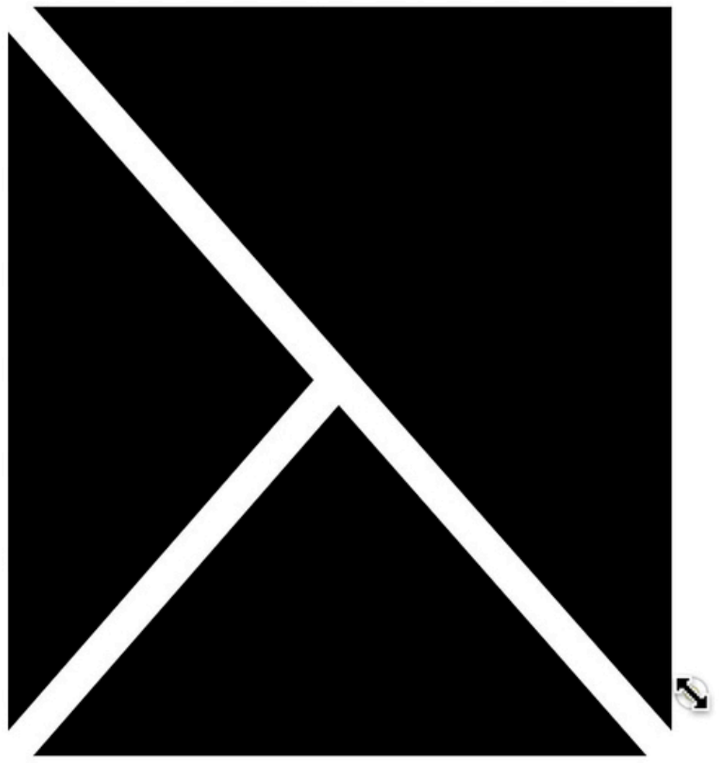
Edit Code

Undo

Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; sketch-n-sketch logo
;
(let [x0 y0 w h delta] [35 42 205 221 37]
(let [xw yh w2 h2] [(+ x0 w) (+ y0 h) (div w 2) (d
(let poly (\pts (polygon 'black' 'none' 0 pts))
(svg [
  (rect 'white' x0 y0 w h)
  (poly
    [[(+ x0 delta) y0]
     [xw y0]
     [xw (- yh delta)]])
  (poly
    [[x0 (+ y0 delta)]
     [x0 (- yh delta)]
     [(- (+ x0 w2) delta) (+ y0 h2)]])
  (poly
    [[(+ x0 delta) yh]
     [(- xw delta) yh]
     [(+ x0 w2) (+ (+ y0 h2) delta)]])
])))
```

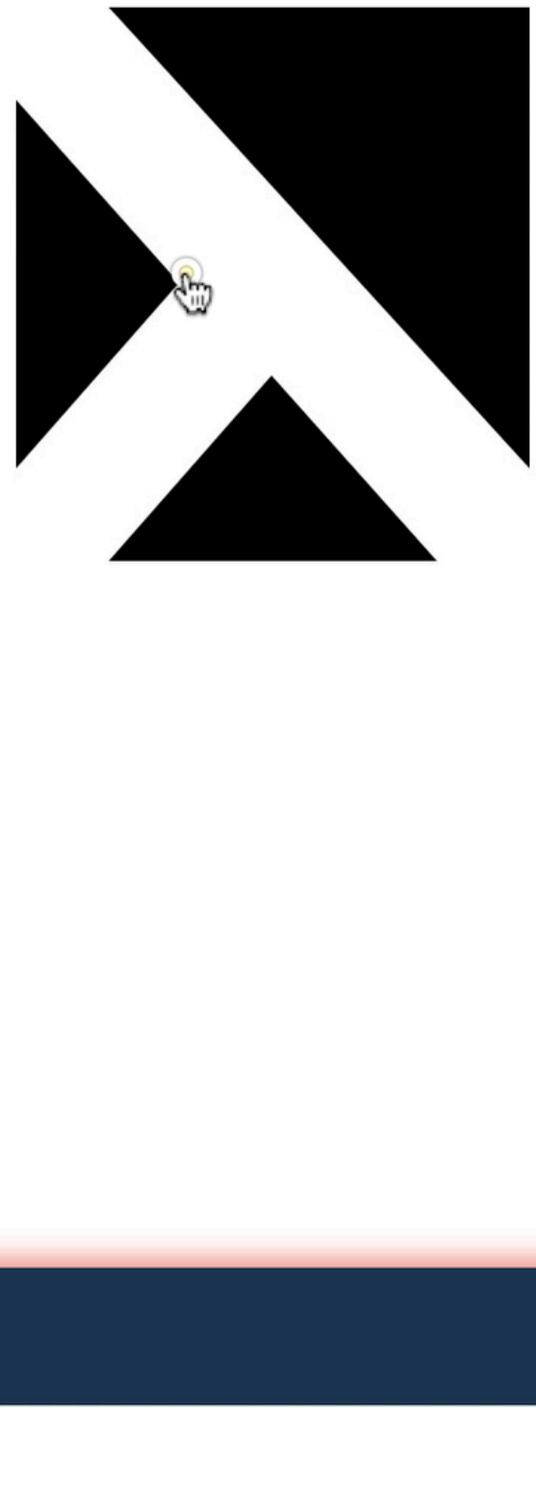
Logo

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Take this ferris wheel for a spin!
; Try:
;  - Stretching the passenger cars
;  - Stretching the central hub
;  - Dragging the central hub
;  - Setting showSliders to false

(def wheel (\(cx cy rCenter wCar rCap numSpokes sp
  (let rim      [(ring 'darkgray' 6 cx cy spokeLen]
  (let center   [(circle 'black' cx cy rCenter)]
  (let frame    [(nStar 'goldenrod' 'darkgray' 3 n
  (let spokePts (nPointsOnCircle numSpokes rotAngl
  (let cars     (mapi (\[i [x y]] (squareCenter (i
  (let hubcaps  (map (\[x y] (circle 'black' x y r
    (concat [rim cars center frame hubcaps])
)))))))

(def [cx cy spokeLen rCenter wCar rCap] [194 266 1
(def [numSpokes rotAngle] [5!{3-15} 0!{-3.14-3.14}]

(svg (wheel cx cy rCenter wCar rCap numSpokes spok
```

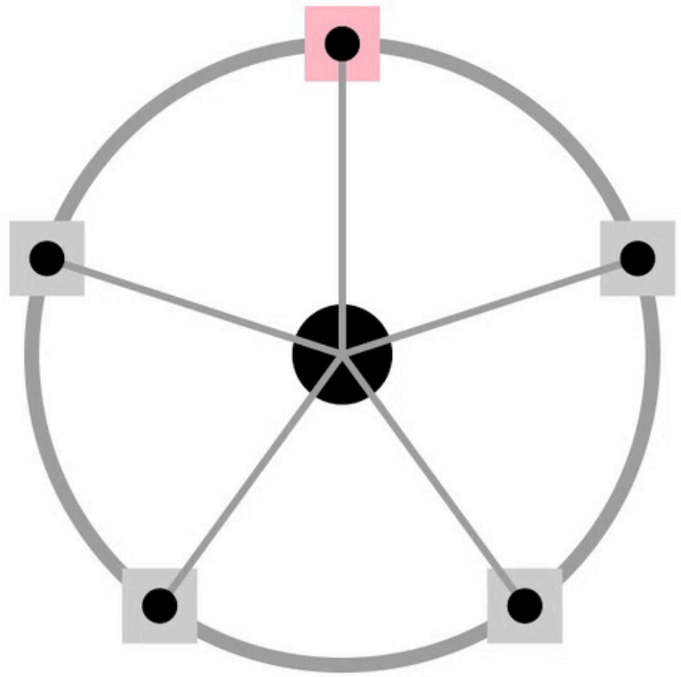
Ferris Wheel

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



rotAngle: 0

numSpokes: 5

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Take this ferris wheel for a spin!
; Try:
;  - Stretching the passenger cars
;  - Stretching the central hub
;  - Dragging the central hub
;  - Setting showSliders to false

(def wheel (\(cx cy rCenter wCar rCap numSpokes sp
  (let rim      [(ring 'darkgray' 6 cx cy spokeLen
  (let center   [(circle 'black' cx cy rCenter)]
  (let frame    [(nStar 'goldenrod' 'darkgray' 3 n
  (let spokePts (nPointsOnCircle numSpokes rotAngl
  (let cars     (mapi (\[i [x y]] (squareCenter (i
  (let hubcaps  (map (\[x y] (circle 'black' x y r
    (concat [rim cars center frame hubcaps])
)))))))

(def [cx cy spokeLen rCenter wCar rCap] [194 266 1
(def [numSpokes rotAngle] [8!{3-15} 0!{-3.14-3.14}]

(svg (wheel cx cy rCenter wCar rCap numSpokes spok
```

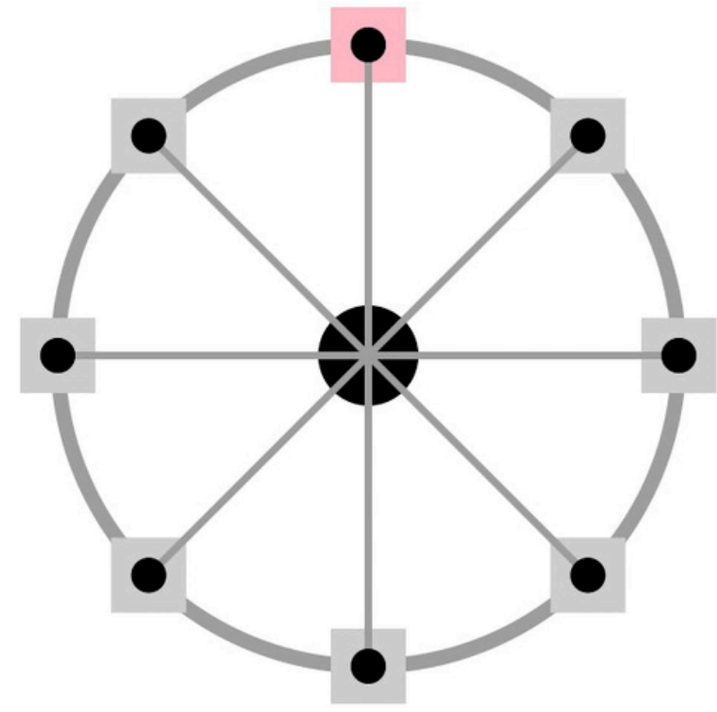
Ferris Wheel

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



rotAngle: 0

numSpokes: 8

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Take this ferris wheel for a spin!
; Try:
;  - Stretching the passenger cars
;  - Stretching the central hub
;  - Dragging the central hub
;  - Setting showSliders to false

(def wheel (\(cx cy rCenter wCar rCap numSpokes sp
  (let rim      [(ring 'darkgray' 6 cx cy spokeLen]
  (let center   [(circle 'black' cx cy rCenter)]
  (let frame    [(nStar 'goldenrod' 'darkgray' 3 n
  (let spokePts (nPointsOnCircle numSpokes rotAngl
  (let cars     (mapi (\[i [x y]] (squareCenter (i
  (let hubcaps  (map (\[x y] (circle 'black' x y r
    (concat [rim cars center frame hubcaps])
)))))))

(def [cx cy spokeLen rCenter wCar rCap] [194 266 1
(def [numSpokes rotAngle] [14!{3-15} 0!{-3.14-3.14]

(svg (wheel cx cy rCenter wCar rCap numSpokes spok
```

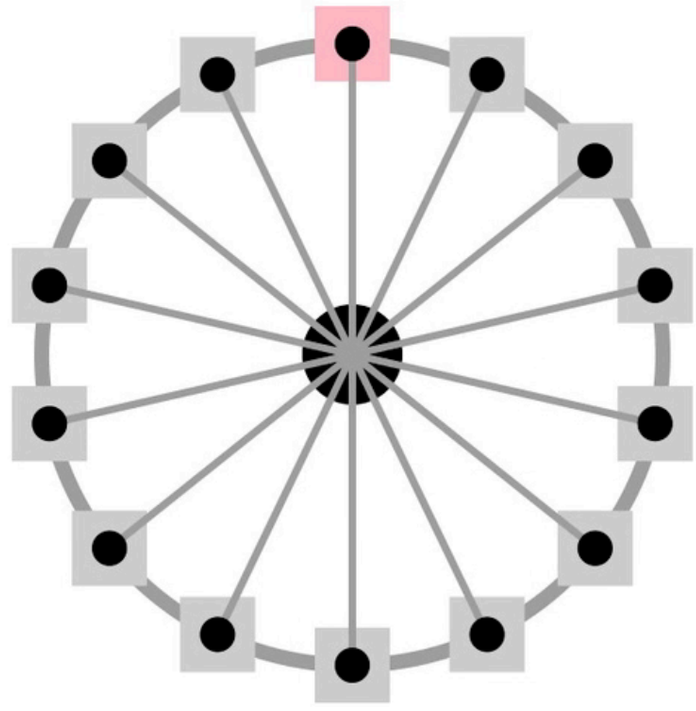
Ferris Wheel

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



rotAngle: 0

numSpokes: 14

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Take this ferris wheel for a spin!
; Try:
;  - Stretching the passenger cars
;  - Stretching the central hub
;  - Dragging the central hub
;  - Setting showSliders to false

(def wheel (\(cx cy rCenter wCar rCap numSpokes sp
  (let rim      [(ring 'darkgray' 6 cx cy spokeLen]
  (let center   [(circle 'black' cx cy rCenter)]
  (let frame    [(nStar 'goldenrod' 'darkgray' 3 n
  (let spokePts (nPointsOnCircle numSpokes rotAngl
  (let cars     (mapi (\[i [x y]] (squareCenter (i
  (let hubcaps  (map (\[x y] (circle 'black' x y r
    (concat [rim cars center frame hubcaps])
)))))))

(def [cx cy spokeLen rCenter wCar rCap] [194 266 1
(def [numSpokes rotAngle] [9!{3-15} 0!{-3.14-3.14}]

(svg (wheel cx cy rCenter wCar rCap numSpokes spok
```

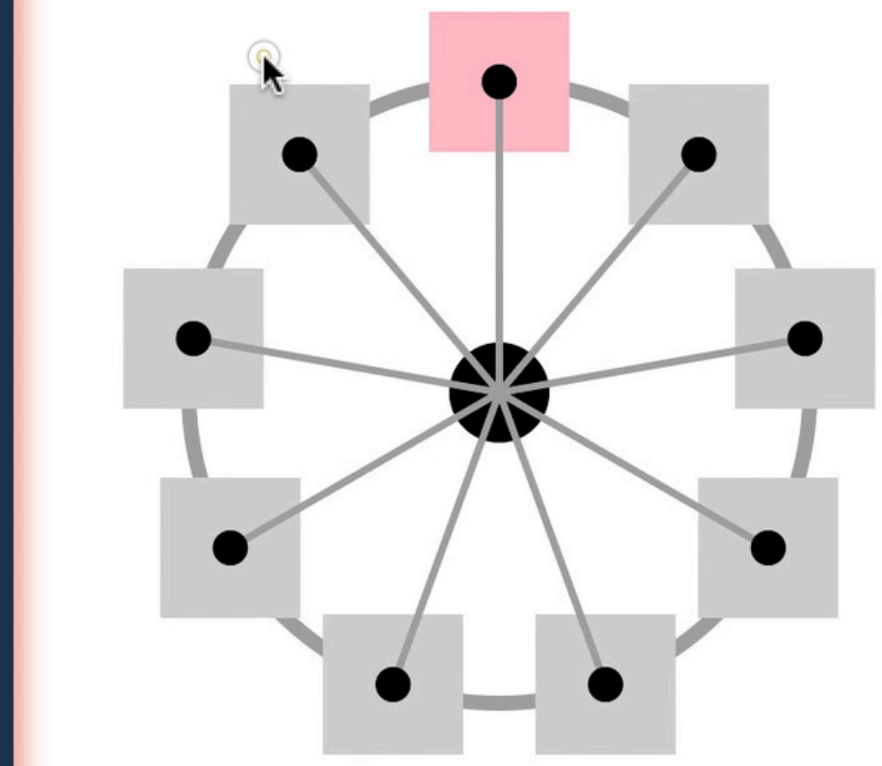
Ferris Wheel

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



rotAngle: 0

numSpokes: 9

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Take this ferris wheel for a spin!
; Try:
;  - Stretching the passenger cars
;  - Stretching the central hub
;  - Dragging the central hub
;  - Setting showSliders to false

(def wheel (\(cx cy rCenter wCar rCap numSpokes sp
  (let rim      [(ring 'darkgray' 6 cx cy spokeLen]
  (let center   [(circle 'black' cx cy rCenter)]
  (let frame    [(nStar 'goldenrod' 'darkgray' 3 n
  (let spokePts (nPointsOnCircle numSpokes rotAngl
  (let cars     (mapi (\[i [x y]] (squareCenter (i
  (let hubcaps  (map (\[x y] (circle 'black' x y r
    (concat [rim cars center frame hubcaps])
)))))))

(def [cx cy spokeLen rCenter wCar rCap] [194 266 1
(def [numSpokes rotAngle] [9!{3-15} 1.1932!{-3.14-

(svg (wheel cx cy rCenter wCar rCap numSpokes spok
```

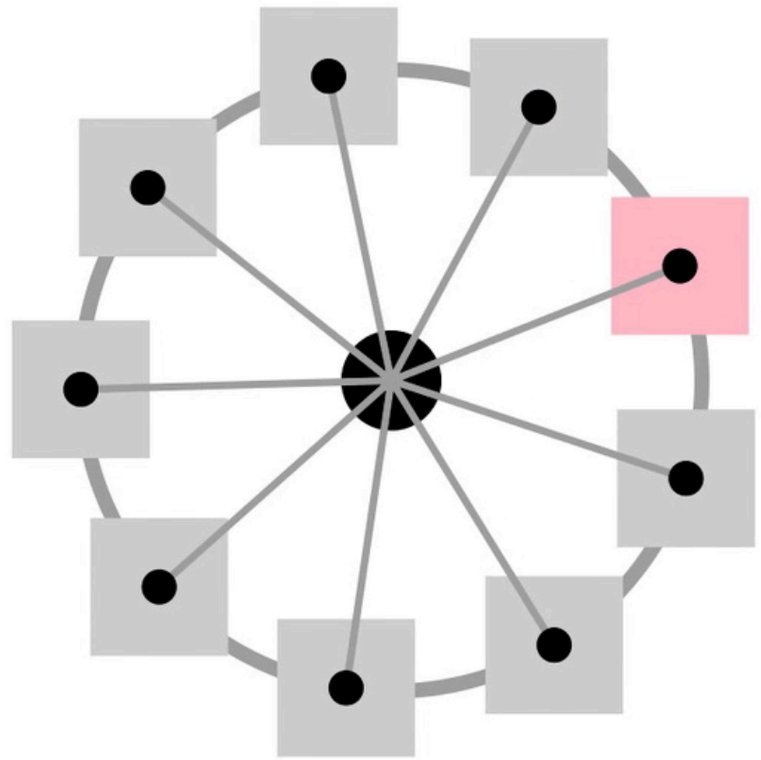
Ferris Wheel

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



rotAngle: 1.193..

numSpokes: 9

[Out] Canvas

sketch-n-sketch v0.4.2

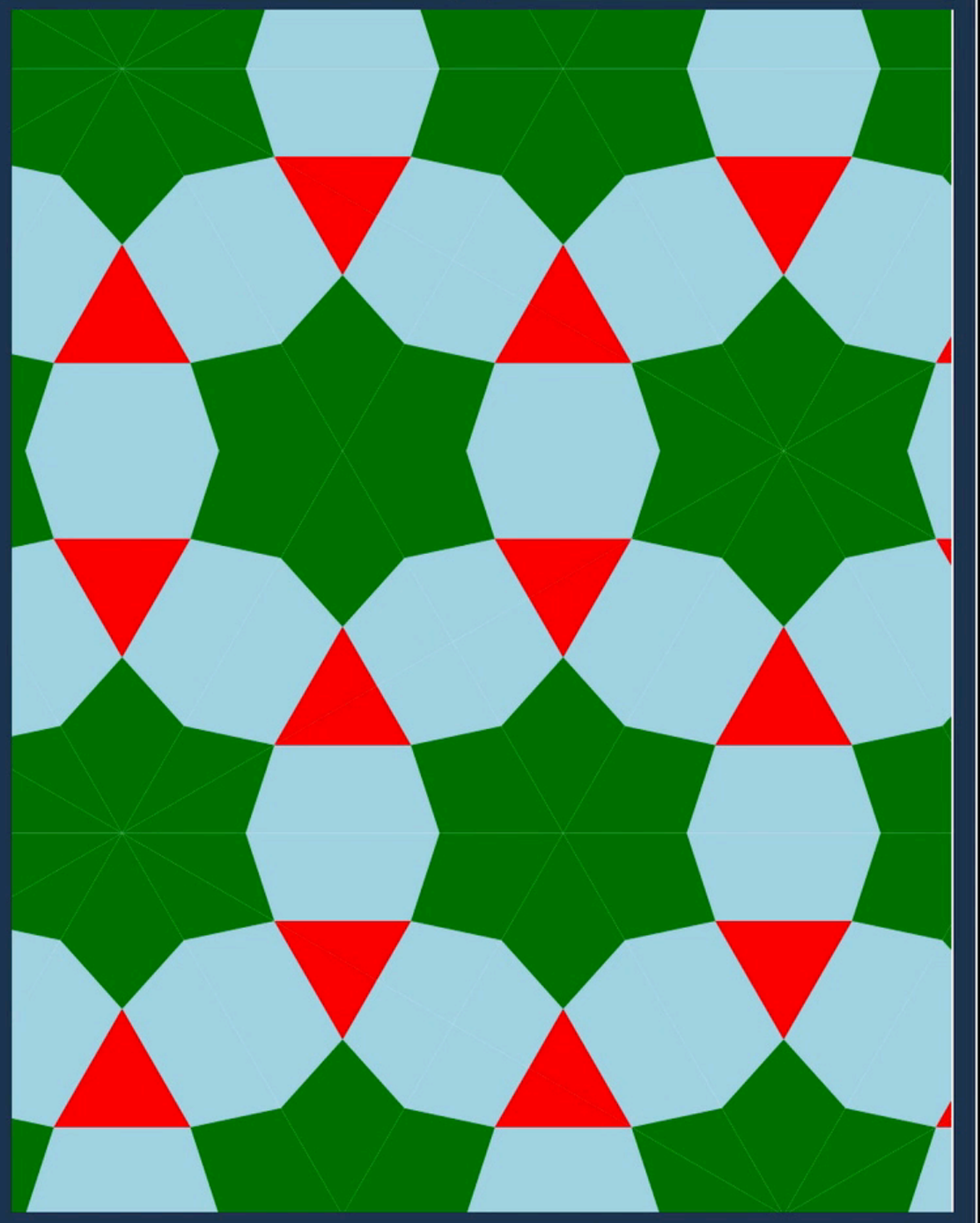
```
(def primitiveHexagon [
  (transformGroup x y false 0 0 0 primitive)
  (transformGroup x y true  0 0 0 primitive)
  (transformGroup x y false 60 0 0 primitive)
  (transformGroup x y true  60 0 0 primitive)
  (transformGroup x y false 120 0 0 primitive)
  (transformGroup x y true  120 0 0 primitive)
  (transformGroup x y false 180 0 0 primitive)
  (transformGroup x y true  180 0 0 primitive)
  (transformGroup x y false 240 0 0 primitive)
  (transformGroup x y true  240 0 0 primitive)
  (transformGroup x y false 300 0 0 primitive)
  (transformGroup x y true  300 0 0 primitive)
])

(def primitiveHexagonColumn [
  (transformGroup x y false 0 0 (* -2 innerRadius
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 0 (* 2 innerRadius)
  (transformGroup x y false 0 0 (* 4 innerRadius)
])

(def primitiveHexagonColumns [
  (transformGroup x y false 0 (* -1 (* radius (+
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 (* 1 (* radius (+ 1
  (transformGroup x y false 0 (* 3 radius) 0 prim
])

(svg primitiveHexagonColumns)
```

Tessellation Ta

Run Code

[Heuristics] Fair

[Orientation] Vertical



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def primitiveHexagon [
  (transformGroup x y false 0 0 0 primitive)
  (transformGroup x y true  0 0 0 primitive)
  (transformGroup x y false 60 0 0 primitive)
  (transformGroup x y true  60 0 0 primitive)
  (transformGroup x y false 120 0 0 primitive)
  (transformGroup x y true  120 0 0 primitive)
  (transformGroup x y false 180 0 0 primitive)
  (transformGroup x y true  180 0 0 primitive)
  (transformGroup x y false 240 0 0 primitive)
  (transformGroup x y true  240 0 0 primitive)
  (transformGroup x y false 300 0 0 primitive)
  (transformGroup x y true  300 0 0 primitive)
])

(def primitiveHexagonColumn [
  (transformGroup x y false 0 0 (* -2 innerRadius
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 0 (* 2 innerRadius)
  (transformGroup x y false 0 0 (* 4 innerRadius)
])

(def primitiveHexagonColumns [
  (transformGroup x y false 0 (* -1 (* radius (+
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 (* 1 (* radius (+ 1
  (transformGroup x y false 0 (* 3 radius) 0 prim
])

(svg primitive)
```

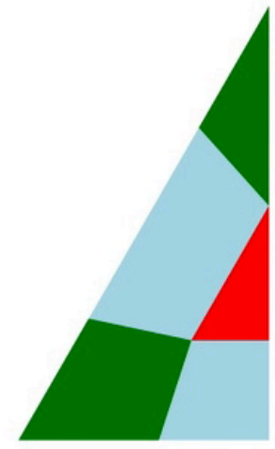
Tessellation Ta

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

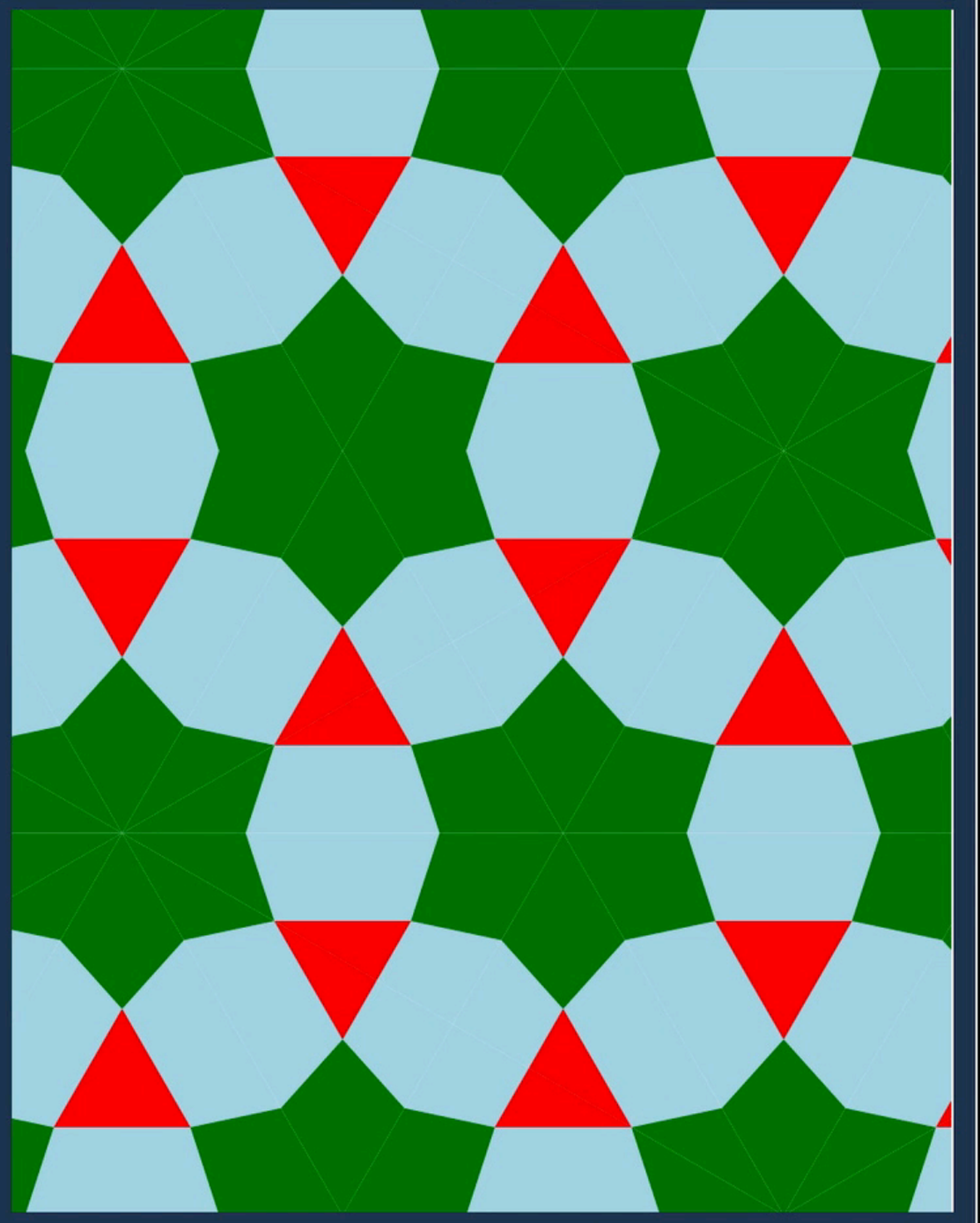
```
(def primitiveHexagon [
  (transformGroup x y false 0 0 0 primitive)
  (transformGroup x y true  0 0 0 primitive)
  (transformGroup x y false 60 0 0 primitive)
  (transformGroup x y true  60 0 0 primitive)
  (transformGroup x y false 120 0 0 primitive)
  (transformGroup x y true  120 0 0 primitive)
  (transformGroup x y false 180 0 0 primitive)
  (transformGroup x y true  180 0 0 primitive)
  (transformGroup x y false 240 0 0 primitive)
  (transformGroup x y true  240 0 0 primitive)
  (transformGroup x y false 300 0 0 primitive)
  (transformGroup x y true  300 0 0 primitive)
])

(def primitiveHexagonColumn [
  (transformGroup x y false 0 0 (* -2 innerRadius
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 0 (* 2 innerRadius)
  (transformGroup x y false 0 0 (* 4 innerRadius)
])

(def primitiveHexagonColumns [
  (transformGroup x y false 0 (* -1 (* radius (+
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 (* 1 (* radius (+ 1
  (transformGroup x y false 0 (* 3 radius) 0 prim
])

(svg primitiveHexagonColumns)
```

Tessellation Ta

Run Code

[Heuristics] Fair

[Orientation] Vertical



[Out] Canvas

sketch-n-sketch v0.4.2

```
(def primitiveHexagon [
  (transformGroup x y false 0 0 0 primitive)
  (transformGroup x y true  0 0 0 primitive)
  (transformGroup x y false 60 0 0 primitive)
  (transformGroup x y true  60 0 0 primitive)
  (transformGroup x y false 120 0 0 primitive)
  (transformGroup x y true  120 0 0 primitive)
  (transformGroup x y false 180 0 0 primitive)
  (transformGroup x y true  180 0 0 primitive)
  (transformGroup x y false 240 0 0 primitive)
  (transformGroup x y true  240 0 0 primitive)
  (transformGroup x y false 300 0 0 primitive)
  (transformGroup x y true  300 0 0 primitive)
])

(def primitiveHexagonColumn [
  (transformGroup x y false 0 0 (* -2 innerRadius
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 0 (* 2 innerRadius)
  (transformGroup x y false 0 0 (* 4 innerRadius)
])

(def primitiveHexagonColumns [
  (transformGroup x y false 0 (* -1 (* radius (+
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 (* 1 (* radius (+ 1
  (transformGroup x y false 0 (* 3 radius) 0 prim
])

(svg primitiveHexagonColumns)
```

Tessellation Ta

Edit Code

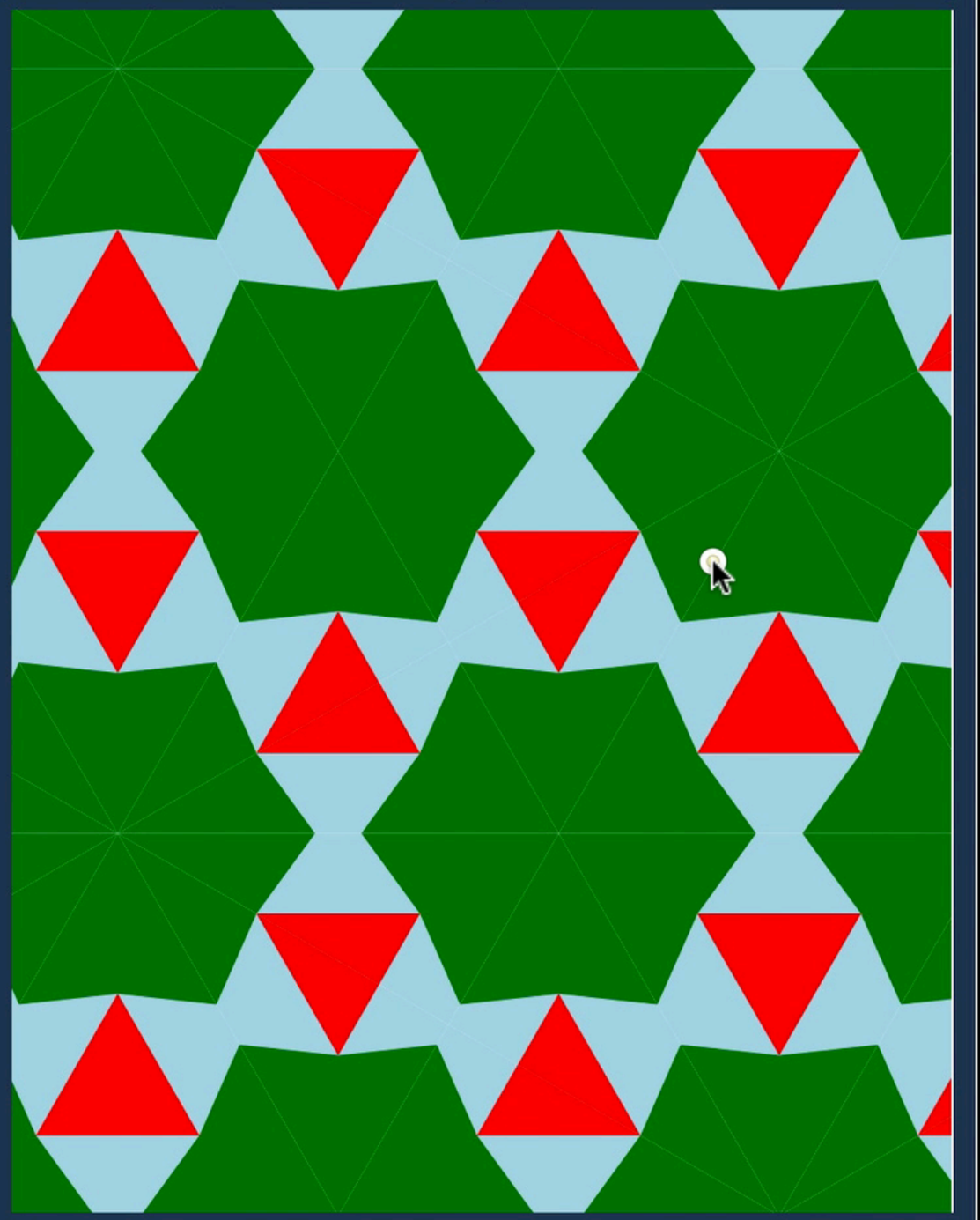
Undo Redo

[Zones] Hidden

[Widgets] Shown

[Heuristics] Fair

[Orientation] Vertical



[Out] Canvas

sketch-n-sketch v0.4.2

```
(def primitiveHexagon [
  (transformGroup x y false 0 0 0 primitive)
  (transformGroup x y true  0 0 0 primitive)
  (transformGroup x y false 60 0 0 primitive)
  (transformGroup x y true  60 0 0 primitive)
  (transformGroup x y false 120 0 0 primitive)
  (transformGroup x y true  120 0 0 primitive)
  (transformGroup x y false 180 0 0 primitive)
  (transformGroup x y true  180 0 0 primitive)
  (transformGroup x y false 240 0 0 primitive)
  (transformGroup x y true  240 0 0 primitive)
  (transformGroup x y false 300 0 0 primitive)
  (transformGroup x y true  300 0 0 primitive)
])

(def primitiveHexagonColumn [
  (transformGroup x y false 0 0 (* -2 innerRadius
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 0 (* 2 innerRadius)
  (transformGroup x y false 0 0 (* 4 innerRadius)
])

(def primitiveHexagonColumns [
  (transformGroup x y false 0 (* -1 (* radius (+
  (transformGroup x y false 0 0 0 primitiveHexago
  (transformGroup x y false 0 (* 1 (* radius (+ 1
  (transformGroup x y false 0 (* 3 radius) 0 prim
])

(svg primitiveHexagonColumns)
```

Tessellation Ta

Edit Code

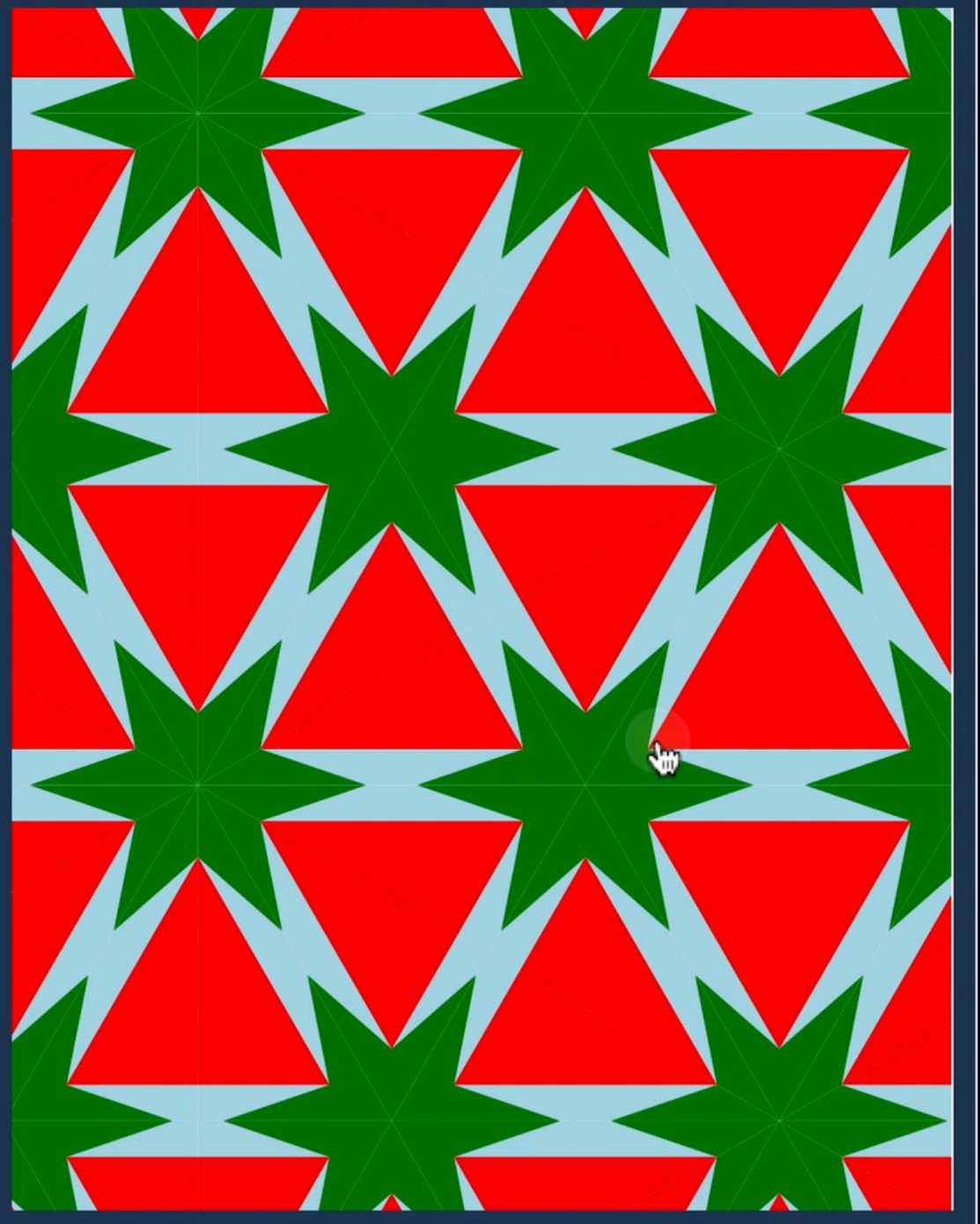
Undo Redo

[Zones] Hidden

[Widgets] Shown

[Heuristics] Fair

[Orientation] Vertical



[Out] Canvas

Custom User Interface Widgets

sketch-n-sketch v0.4.2

```
;
; The ni constants get adjusted by the sliders,
; and then clamped to fit within the [min, max] r
; Also try changing the min and max constants bel
;
(let [min max] [0! 10!]
(let [n1 n2 n3 n4] [7.1 5 5 5]
(let [m1 m2 m3 m4] (map (clamp min max) [n1 n2 n3
;
; Both the horizontal and vertical slider abstrac
; below take a dropBall parameter:
;  - if true, the ball can slide off the rail;
;  - if false, the ball disappears when off the r
;
(let horizSlider (\(dropBall xStart xEnd y minVa
  (let [rPoint wLine rBall] [4! 3! 10!]
  (let [xDiff valDiff] [(- xEnd xStart) (- maxVal
  (let xBall (+ xStart (* xDiff (/ (- curVal minV
  (let xBall_ (clamp xStart xEnd xBall)
  (let rBall_ (if dropBall (if (= xBall_ xBall) r
    [ (circle 'black' xStart y rPoint)
      (circle 'black' xEnd y rPoint)
      (line 'black' wLine xStart y xEnd y)
      (circle 'black' xBall y rBall_)
    ])))))
;
(let vertSlider (\(dropBall yStart yEnd x minVal
  (let [rPoint wLine rBall] [4! 3! 10!]
  (let [yDiff valDiff] [(- yEnd yStart) (- maxVal
  (let yBall (+ yStart (* yDiff (/ (- curVal minV
```

Sliders

Edit Code

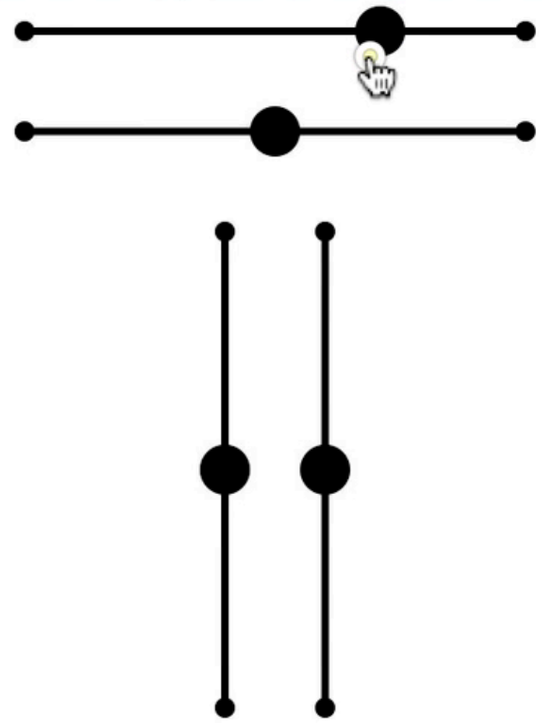
Undo Redo

[Zones] Hidden

[Widgets] Shown

[Heuristics] Fair

[Orientation] Vertical



m1 = 7.1
m2 = 5
m3 = 5
m4 = 5

[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def roundedRect (\(fill x y w h rxSeed rySeed)
  (let off 20!
  (let [wHalf hHalf] [(/ w 2!) (/ h 2!)]
  (let [rx rxSlider] (hSlider true x (+ x wHalf) (
  (let [ry rySlider] (vSlider true y (+ y hHalf) (
  (concat [
    [ (consAttr (consAttr (rect fill x y w h) ['rx
    rxSlider
    rySlider
  ])))))))

(svg (concat [
  [(rect 'black' 0 0 0 0)] ; TODO remove dummy
  (roundedRect 150 101 100 80 200 30 46)
]))
```

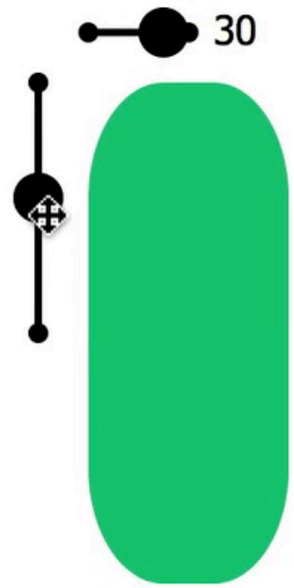
Rounded Rect

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def roundedRect (\(fill x y w h rxSeed rySeed)
  (let off 20!
  (let [wHalf hHalf] [(/ w 2!) (/ h 2!)]
  (let [rx rxSlider] (hSlider true x (+ x wHalf) (
  (let [ry rySlider] (vSlider true y (+ y hHalf) (
  (concat [
    [ (consAttr (consAttr (rect fill x y w h) ['rx
    rxSlider
    rySlider
  ])))))))

(svg (concat [
  [(rect 'black' 0 0 0 0)] ; TODO remove dummy
  (roundedRect 150 101 100 227 112 30 46)
]))
```

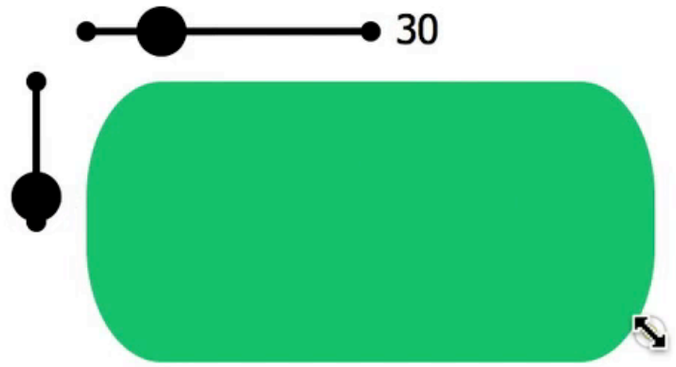
Rounded Rect

Edit Code

Undo Redo

[Zones] Hidden

[Widgets] Shown



[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
(def roundedRect (\(fill x y w h rxSeed rySeed)
  (let off 20!
  (let [wHalf hHalf] [(/ w 2!) (/ h 2!)]
  (let [rx rxSlider] (hSlider true x (+ x wHalf) (
  (let [ry rySlider] (vSlider true y (+ y hHalf) (
  (concat [
    [ (consAttr (consAttr (rect fill x y w h) ['rx
    rxSlider
    rySlider
  ])))))))

(svg (concat [
  [(rect 'black' 0 0 0 0)] ; TODO remove dummy
  (roundedRect 150 101 100 132 173 30 46)
]))
```

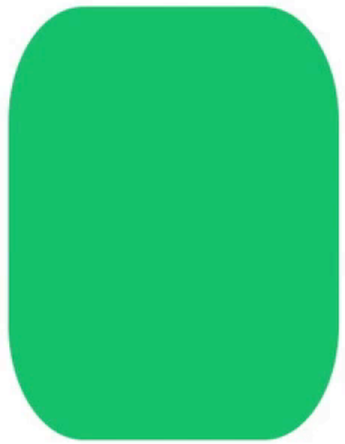
Rounded Rect

Edit Code

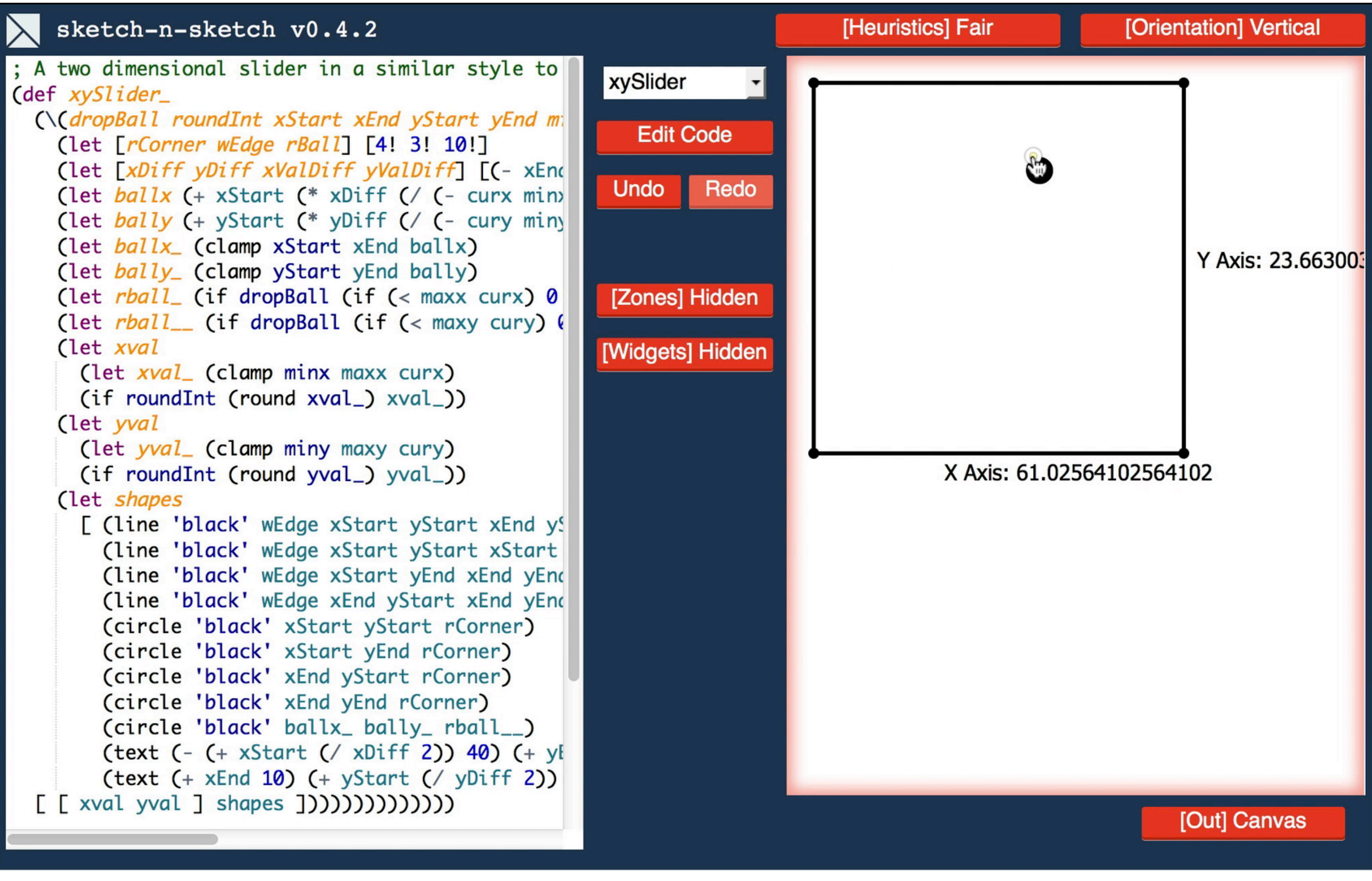
Undo Redo

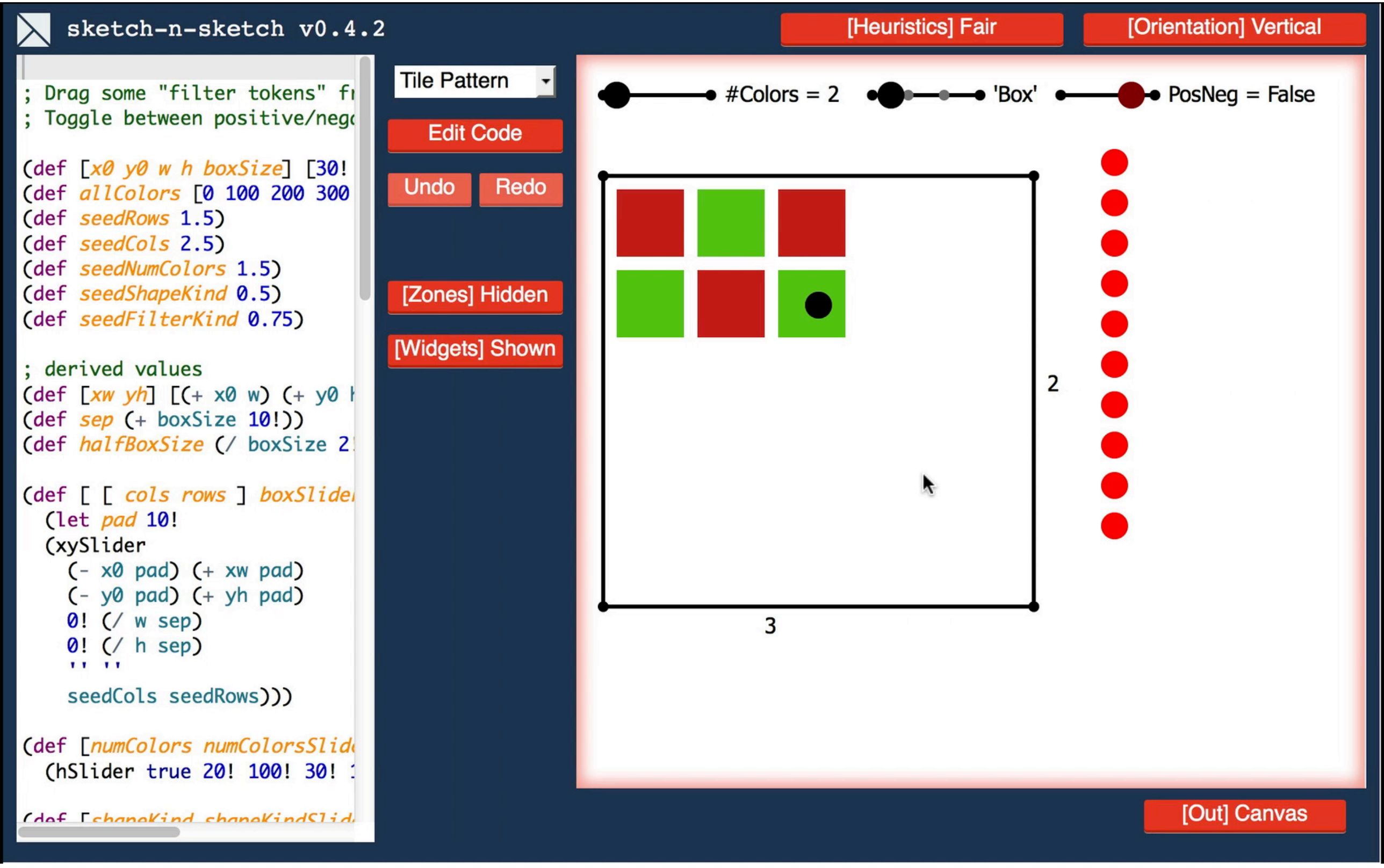
[Zones] Hidden

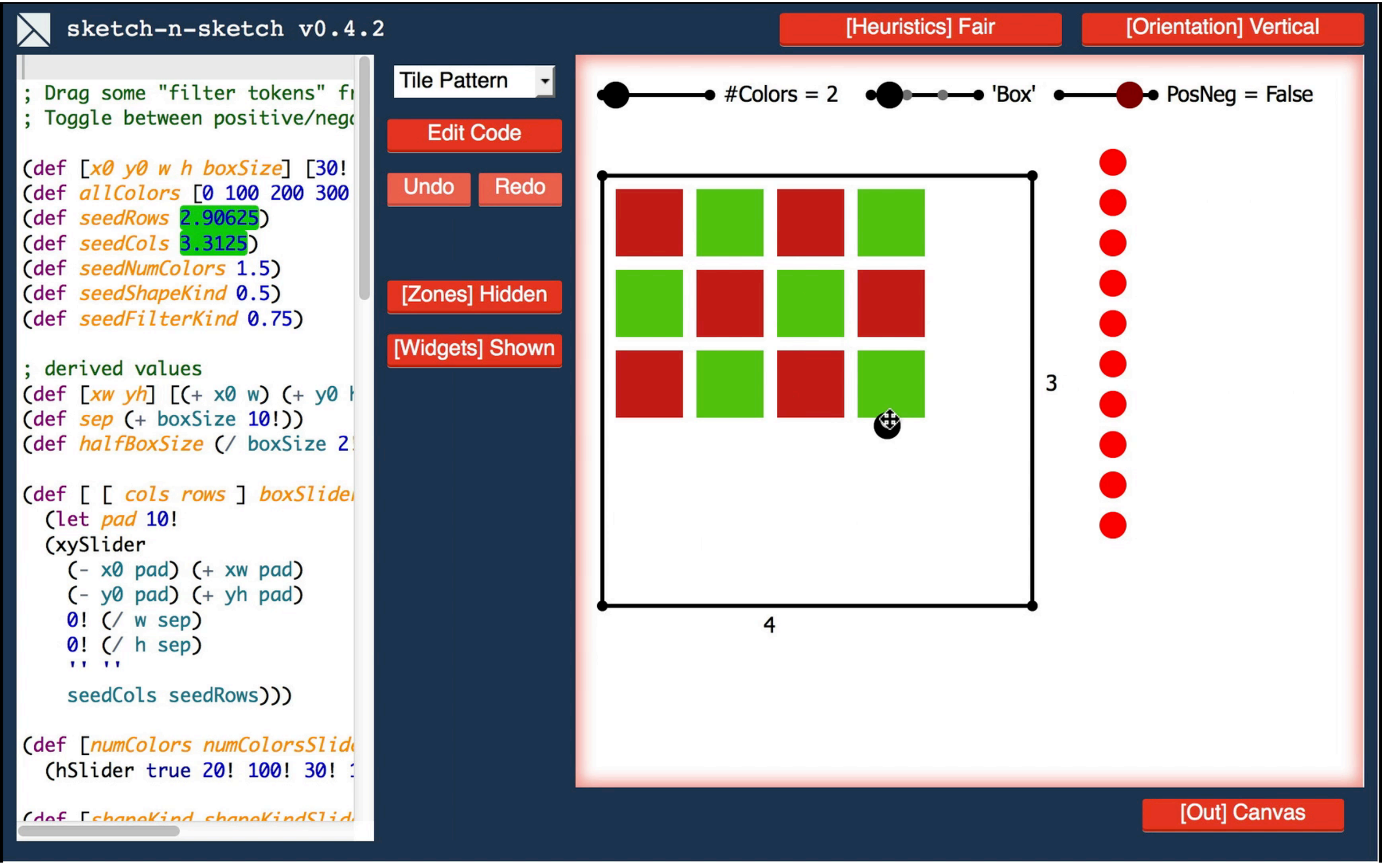
[Widgets] Hidden



[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; A two dimensional slider in a similar style to
(def xySlider_
(\(dropBall roundInt xStart xEnd yStart yEnd m
(let [rCorner wEdge rBall] [4! 3! 10!]
(let [xDiff yDiff xValDiff yValDiff] [(- xEnd
(let ballx (+ xStart (* xDiff (/ (- curx minx
(let bally (+ yStart (* yDiff (/ (- cury miny
(let ballx_ (clamp xStart xEnd ballx)
(let bally_ (clamp yStart yEnd bally)
(let rball_ (if dropBall (if (< maxx curx) 0
(let rball__ (if dropBall (if (< maxy cury) 0
(let xval
(let xval_ (clamp minx maxx curx)
(if roundInt (round xval_) xval_))
(let yval
(let yval_ (clamp miny maxy cury)
(if roundInt (round yval_) yval_))
(let shapes
[(line 'black' wEdge xStart yStart xEnd yS
(line 'black' wEdge xStart yStart xStart
(line 'black' wEdge xStart yEnd xEnd yEnd
(line 'black' wEdge xEnd yStart xEnd yEnd
(circle 'black' xStart yStart rCorner)
(circle 'black' xStart yEnd rCorner)
(circle 'black' xEnd yStart rCorner)
(circle 'black' xEnd yEnd rCorner)
(circle 'black' ballx_ bally_ rball__)
(text (- (+ xStart (/ xDiff 2)) 40) (+ yE
(text (+ xEnd 10) (+ yStart (/ yDiff 2))
[[xval yval] shapes]))))))))))))
xySlider
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Hidden
Y Axis: 23.663003
X Axis: 61.02564102564102
[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; Drag some "filter tokens" fr
; Toggle between positive/nega
(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 1.5)
(def seedCols 2.5)
(def seedNumColors 1.5)
(def seedShapeKind 0.5)
(def seedFilterKind 0.75)
; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2!
(def [[cols rows] boxSlider
(let pad 10!
(xySlider
(- x0 pad) (+ xw pad)
(- y0 pad) (+ yh pad)
0! (/ w sep)
0! (/ h sep)
'' ''
seedCols seedRows)))
(def [numColors numColorsSlid
(hSlider true 20! 100! 30! 1
Tile Pattern
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Shown
#Colors = 2
'Box'
PosNeg = False
2
3
[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; Drag some "filter tokens" fr
; Toggle between positive/nega
(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 2.90625)
(def seedCols 3.3125)
(def seedNumColors 1.5)
(def seedShapeKind 0.5)
(def seedFilterKind 0.75)
; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2!
(def [[cols rows] boxSlider
(let pad 10!
(xySlider
(- x0 pad) (+ xw pad)
(- y0 pad) (+ yh pad)
0! (/ w sep)
0! (/ h sep)
'' ''
seedCols seedRows)))
(def [numColors numColorsSlide
(hSlider true 20! 100! 30! 1
Tile Pattern
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Shown
#Colors = 2
'Box'
PosNeg = False
3
4
[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Drag some "filter tokens" fr
; Toggle between positive/nega

(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 1.5)
(def seedShapeKind 0.5)
(def seedFilterKind 0.75)

; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2!

(def [ [ cols rows ] boxSlider
  (let pad 10!
  (xySlider
    (- x0 pad) (+ xw pad)
    (- y0 pad) (+ yh pad)
    0! (/ w sep)
    0! (/ h sep)
    '' ''
    seedCols seedRows)))

(def [numColors numColorsSlide
  (hSlider true 20! 100! 30! 1

(def [shapeKind shapeKindSlid
```

Tile Pattern

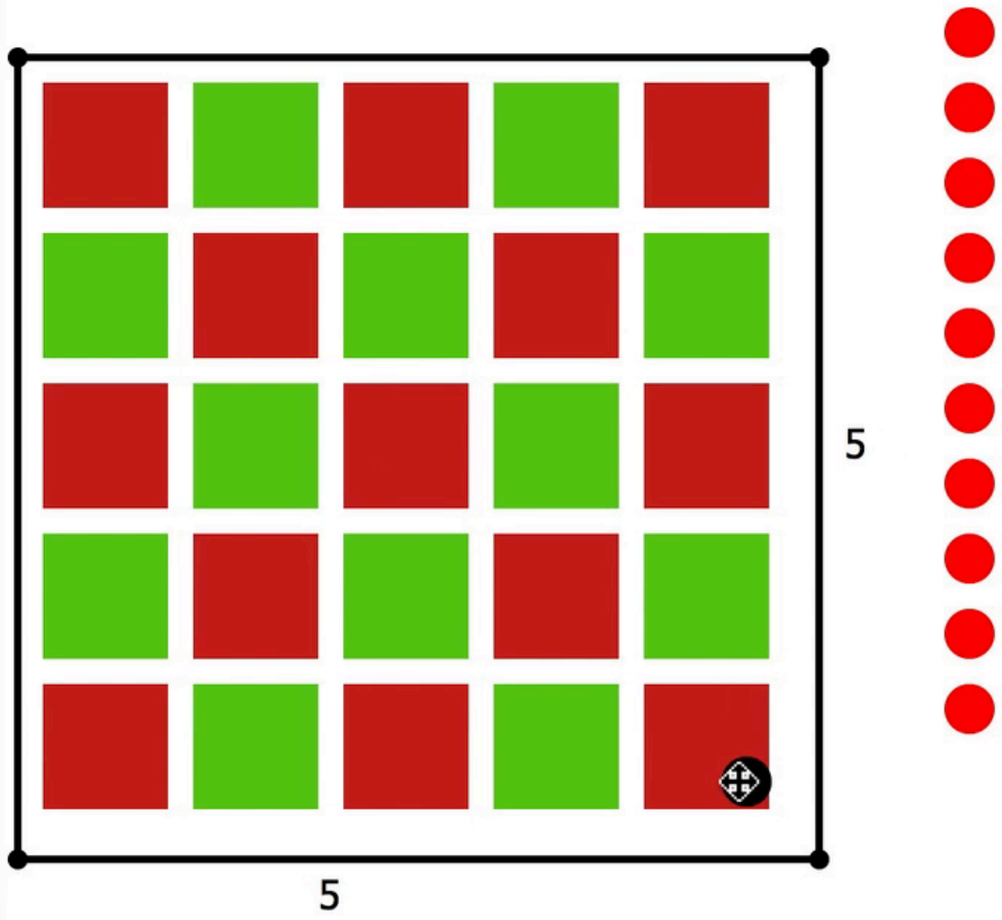
Edit Code

Undo Redo

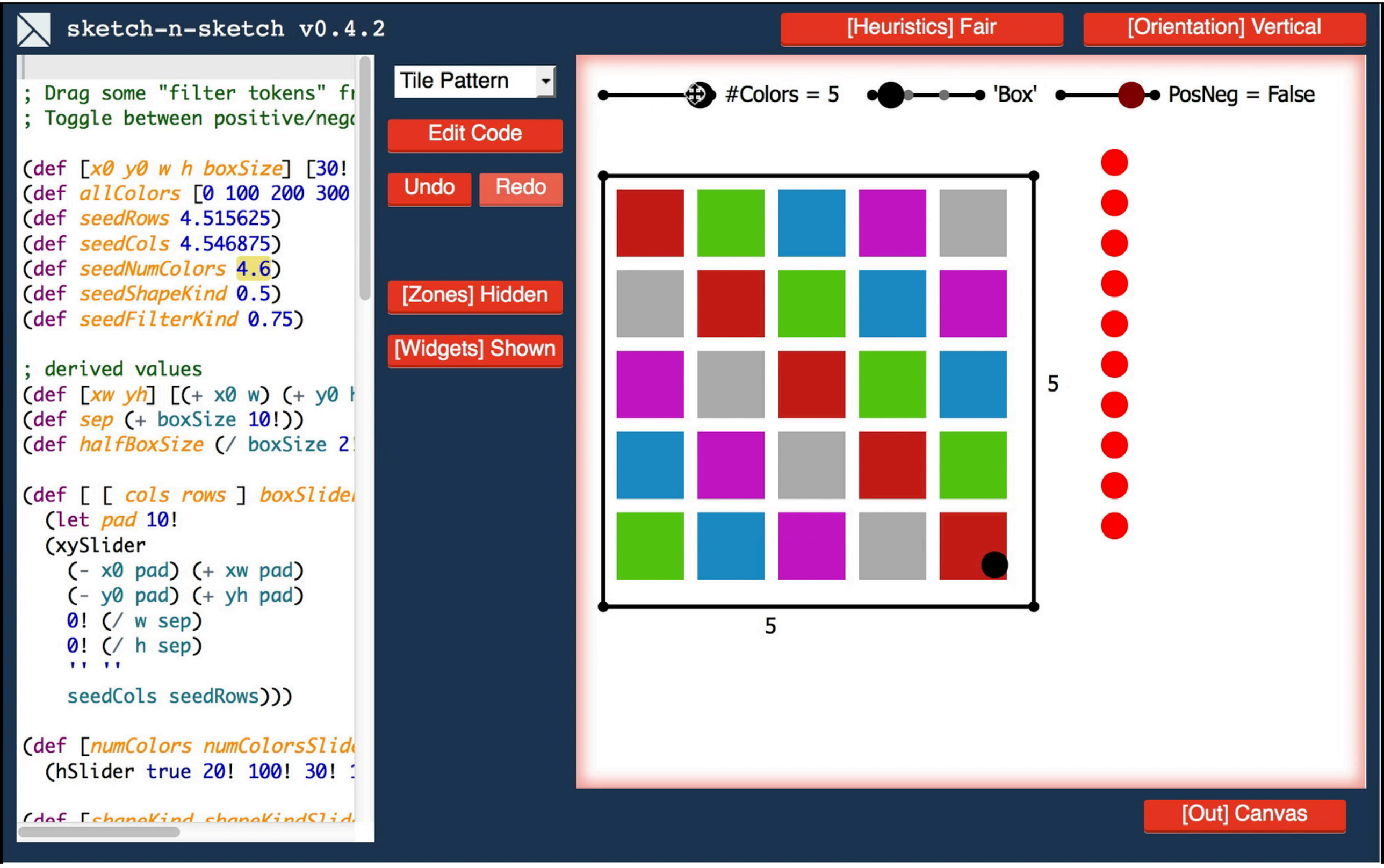
[Zones] Hidden

[Widgets] Shown

#Colors = 2 'Box' PosNeg = False



[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; Drag some "filter tokens" fr
; Toggle between positive/nega
(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 4.6)
(def seedShapeKind 0.5)
(def seedFilterKind 0.75)
; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2
(def [[cols rows] boxSlider
(let pad 10!
(xySlider
(- x0 pad) (+ xw pad)
(- y0 pad) (+ yh pad)
0! (/ w sep)
0! (/ h sep)
'' ''
seedCols seedRows)))
(def [numColors numColorsSlid
(hSlider true 20! 100! 30! 1
Tile Pattern
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Shown
#Colors = 5
'Box'
PosNeg = False
5
5
[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Drag some "filter tokens" fr
; Toggle between positive/nega

(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 1.15)
(def seedShapeKind 0.5)
(def seedFilterKind 0.75)

; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2!

(def [ [ cols rows ] boxSlider
  (let pad 10!
  (xySlider
    (- x0 pad) (+ xw pad)
    (- y0 pad) (+ yh pad)
    0! (/ w sep)
    0! (/ h sep)
    '' ''
    seedCols seedRows)))

(def [numColors numColorsSlide
  (hSlider true 20! 100! 30! 1
```

Tile Pattern

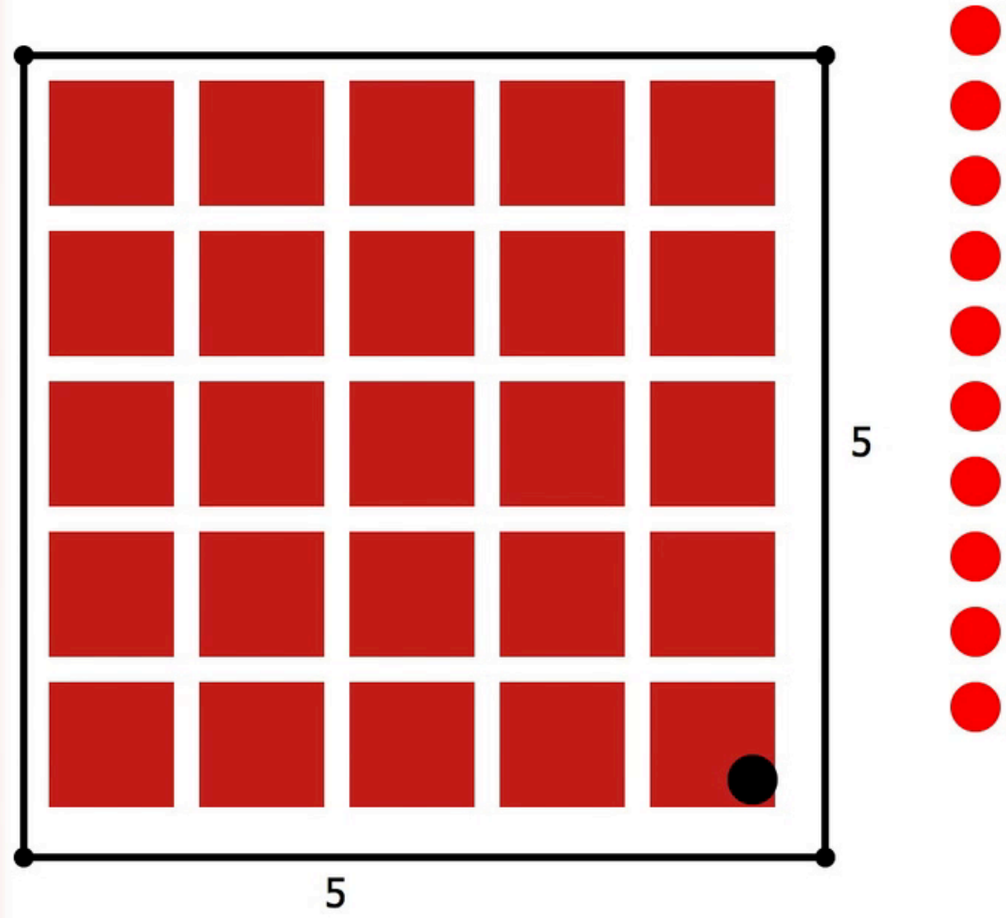
Edit Code

Undo Redo

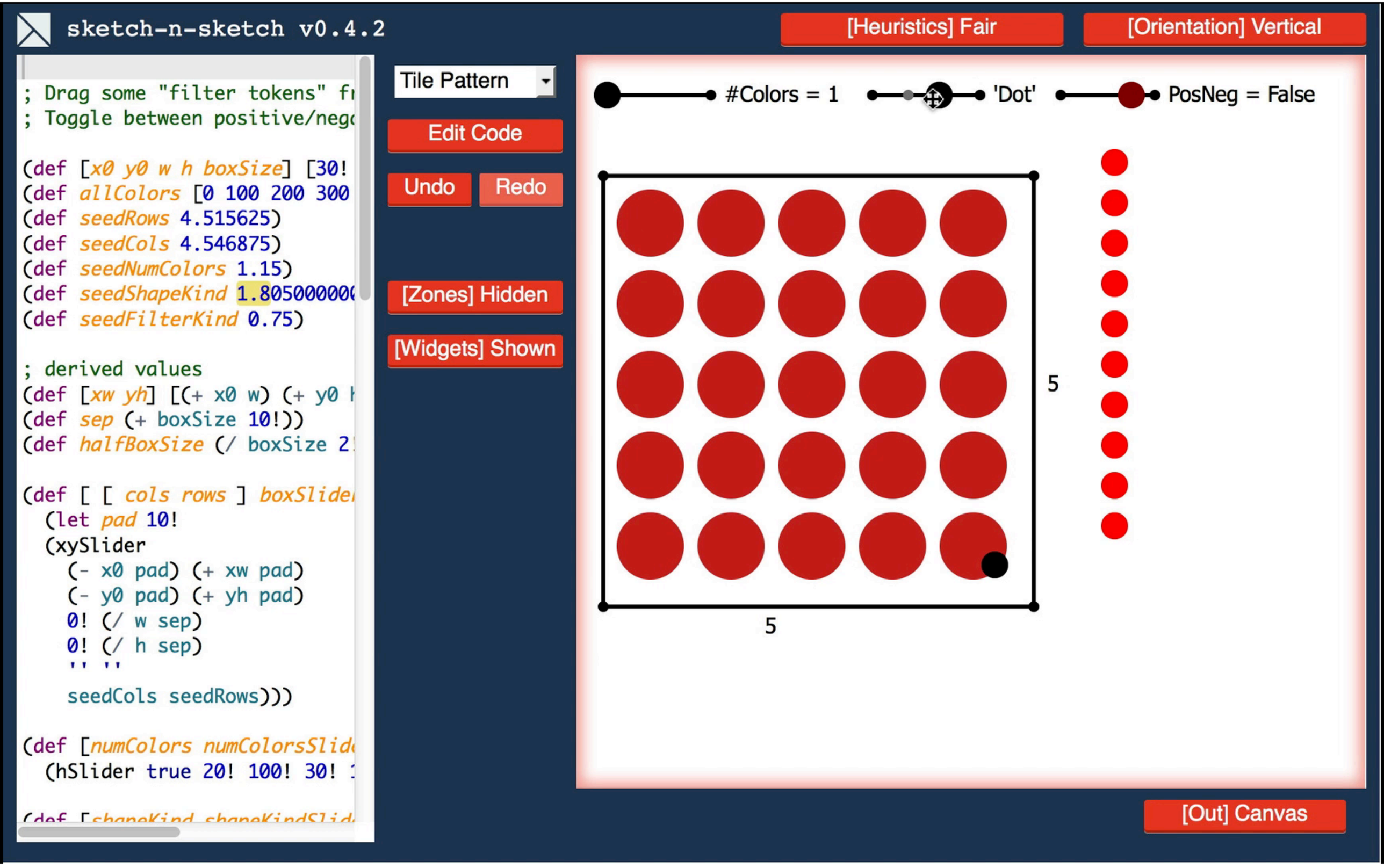
[Zones] Hidden

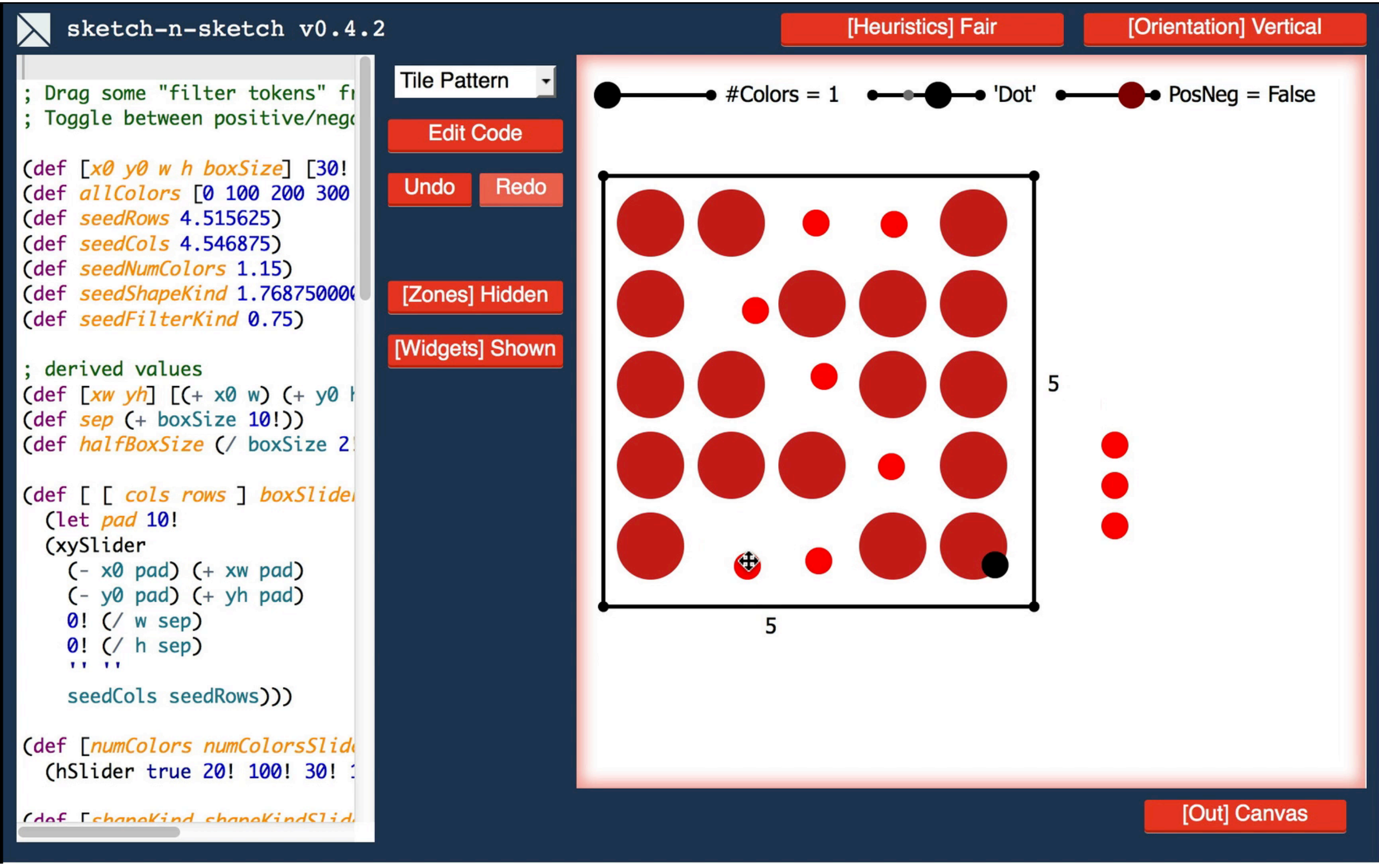
[Widgets] Shown

#Colors = 1 'Box' PosNeg = False



[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; Drag some "filter tokens" fr
; Toggle between positive/nega
(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 1.15)
(def seedShapeKind 1.805000000
(def seedFilterKind 0.75)
; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2!
(def [[cols rows] boxSlider
(let pad 10!
(xySlider
(- x0 pad) (+ xw pad)
(- y0 pad) (+ yh pad)
0! (/ w sep)
0! (/ h sep)
'' ''
seedCols seedRows)))
(def [numColors numColorsSlide
(hSlider true 20! 100! 30! 1
Tile Pattern
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Shown
#Colors = 1
'Dot'
PosNeg = False
5
5
[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; Drag some "filter tokens" fr
; Toggle between positive/nega
(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 1.15)
(def seedShapeKind 1.768750000
(def seedFilterKind 0.75)
; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2
(def [[cols rows] boxSlider
(let pad 10!
(xySlider
(- x0 pad) (+ xw pad)
(- y0 pad) (+ yh pad)
0! (/ w sep)
0! (/ h sep)
'' ''
seedCols seedRows)))
(def [numColors numColorsSlid
(hSlider true 20! 100! 30! 1
Tile Pattern
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Shown
#Colors = 1
'Dot'
PosNeg = False
5
5
[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Drag some "filter tokens" fr
; Toggle between positive/nega

(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 1.15)
(def seedShapeKind 1.768750000
(def seedFilterKind 0.3928571

; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2!

(def [ [ cols rows ] boxSlider
  (let pad 10!
  (xySlider
    (- x0 pad) (+ xw pad)
    (- y0 pad) (+ yh pad)
    0! (/ w sep)
    0! (/ h sep)
    '' ''
    seedCols seedRows)))

(def [numColors numColorsSlide
  (hSlider true 20! 100! 30! 1

(def [shapeKind shapeKindSlid
```

Tile Pattern

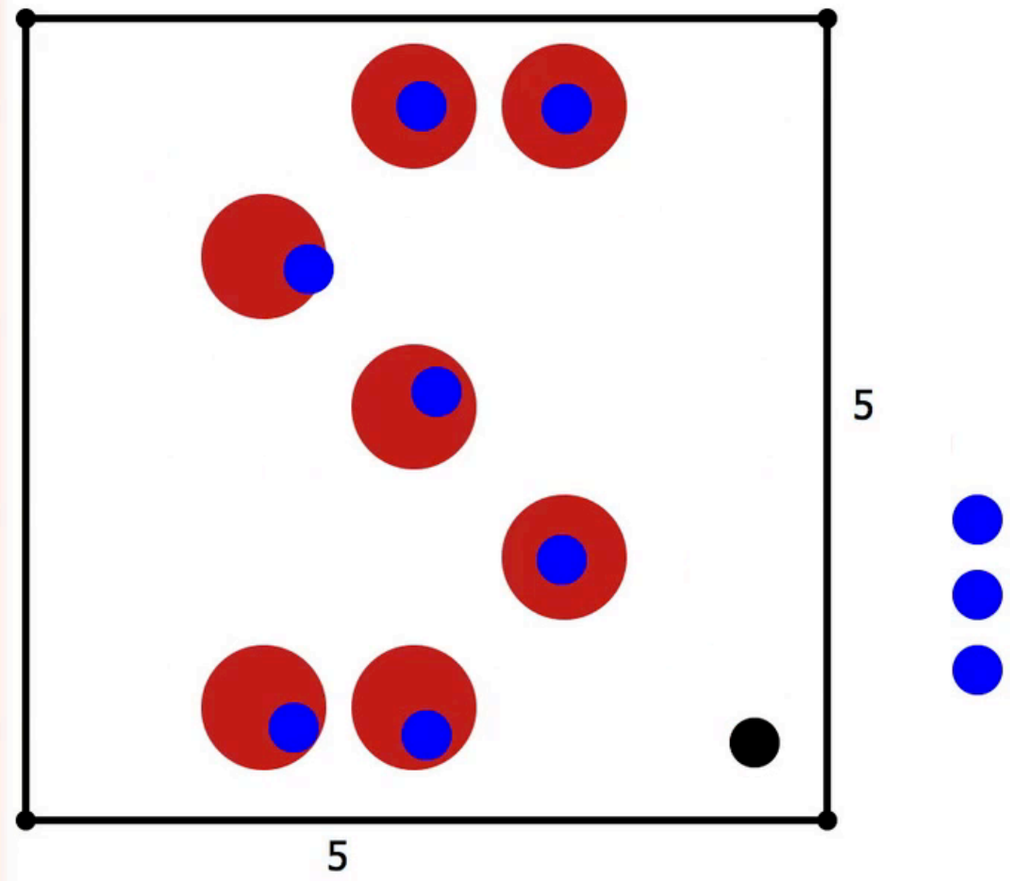
Edit Code

Undo Redo

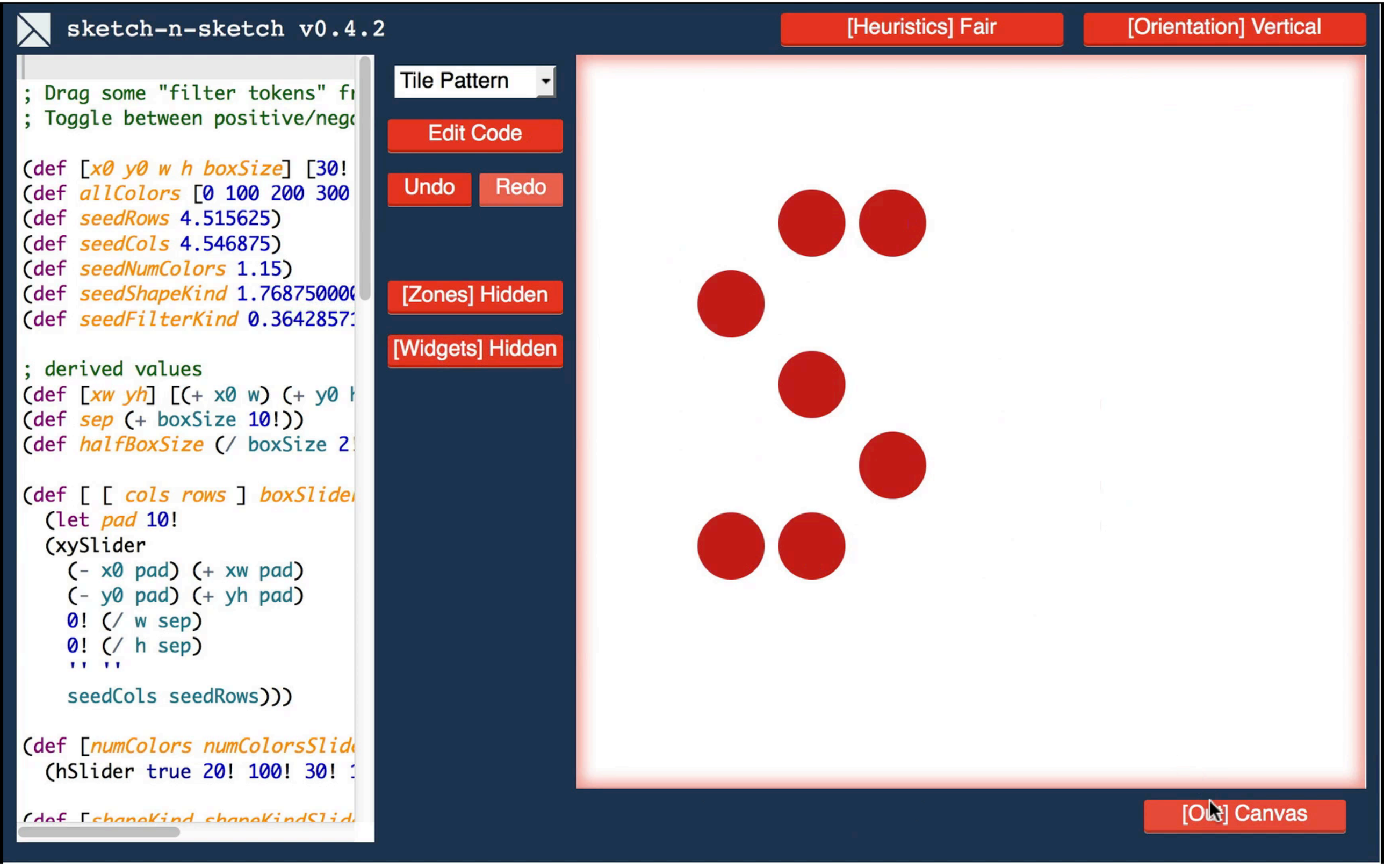
[Zones] Hidden

[Widgets] Shown

#Colors = 1 'Dot' PosNeg = True



[Out] Canvas


sketch-n-sketch v0.4.2
[Heuristics] Fair
[Orientation] Vertical
; Drag some "filter tokens" fr
; Toggle between positive/nega
(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 1.15)
(def seedShapeKind 1.768750000
(def seedFilterKind 0.3642857
; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2
(def [[cols rows] boxSlider
(let pad 10!
(xySlider
(- x0 pad) (+ xw pad)
(- y0 pad) (+ yh pad)
0! (/ w sep)
0! (/ h sep)
'' ''
seedCols seedRows)))
(def [numColors numColorsSlide
(hSlider true 20! 100! 30! 1
Tile Pattern
Edit Code
Undo
Redo
[Zones] Hidden
[Widgets] Hidden
[Out] Canvas

sketch-n-sketch v0.4.2

[Heuristics] Fair

[Orientation] Vertical

```
; Drag some "filter tokens" fr
; Toggle between positive/nega

(def [x0 y0 w h boxSize] [30!
(def allColors [0 100 200 300
(def seedRows 4.515625)
(def seedCols 4.546875)
(def seedNumColors 1.15)
(def seedShapeKind 1.768750000
(def seedFilterKind 0.3642857

; derived values
(def [xw yh] [(+ x0 w) (+ y0 h
(def sep (+ boxSize 10!))
(def halfBoxSize (/ boxSize 2

(def [ [ cols rows ] boxSlider
  (let pad 10!
  (xySlider
    (- x0 pad) (+ xw pad)
    (- y0 pad) (+ yh pad)
    0! (/ w sep)
    0! (/ h sep)
    '' ''
    seedCols seedRows)))

(def [numColors numColorsSlide
  (hSlider true 20! 100! 30! 1

(def [shapeKind shapeKindSlid
```

Tile Pattern

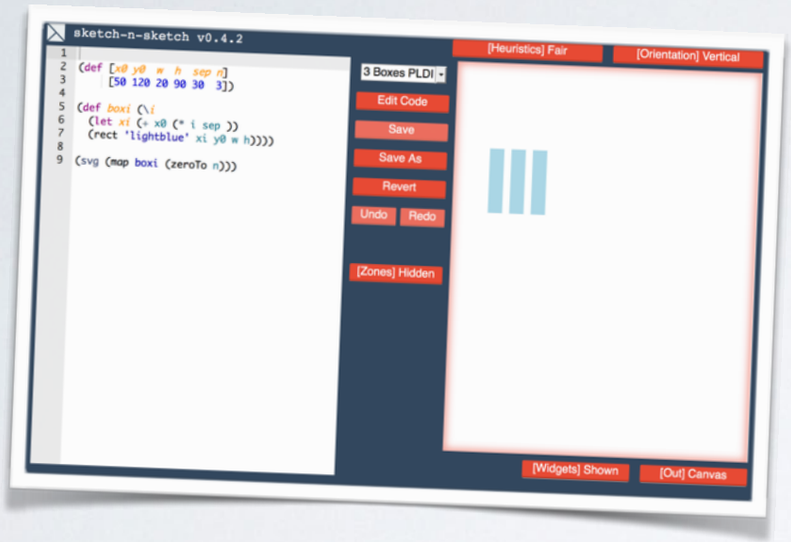
Edit Code

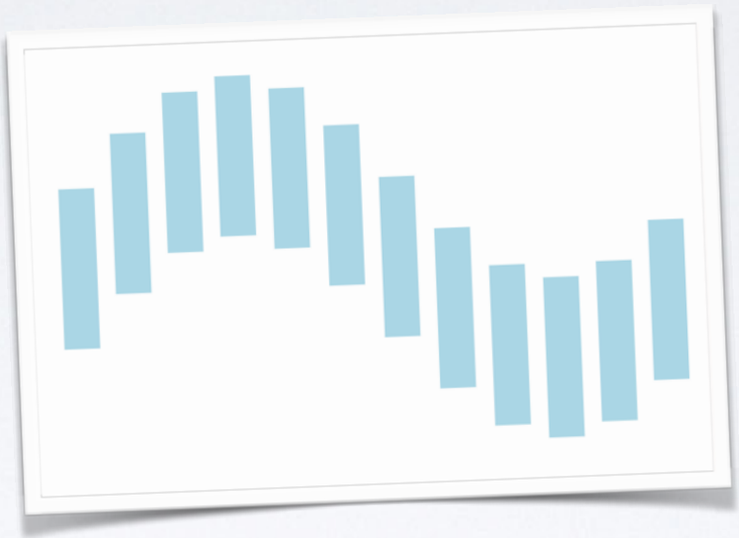
Undo Redo

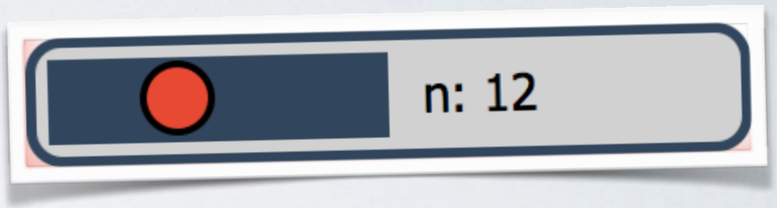
```
<svg xmlns='http://www.w3.org/2000/svg'>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='115' cy='185' r='25' fill='rgba(200, 55, 55, 1
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='115' cy='365' r='25' fill='rgba(200, 55, 55, 1
  <circle cx='175' cy='125' r='25' fill='rgba(200, 55, 55, 1
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='175' cy='245' r='25' fill='rgba(200, 55, 55, 1
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='175' cy='365' r='25' fill='rgba(200, 55, 55, 1
  <circle cx='235' cy='125' r='25' fill='rgba(200, 55, 55, 1
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='235' cy='305' r='25' fill='rgba(200, 55, 55, 1
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
  <circle cx='0' cy='0' r='0' fill='none'></circle>
</svg>
```

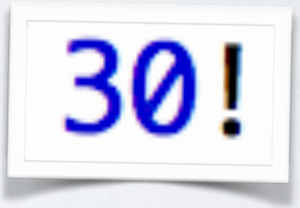
[Out] SVG

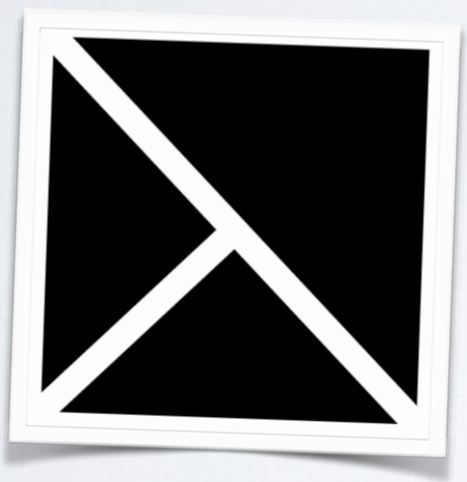
DEMOS

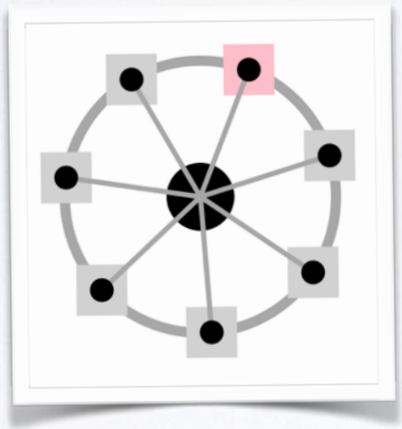

sketch-n-sketch v0.4.2
(def [x0 y0 w h sep n]
[50 120 20 90 30 3])
(def boxi (\i
(let xi (+ x0 (* i sep))
(rect 'lightblue' xi y0 w h))))
(svg (map boxi (zeroTo n)))
3 Boxes PLDI
Edit Code
Save
Save As
Revert
Undo
Redo
[Zones] Hidden
[Heuristics] Fair
[Orientation] Vertical
[Widgets] Shown
[Out] Canvas

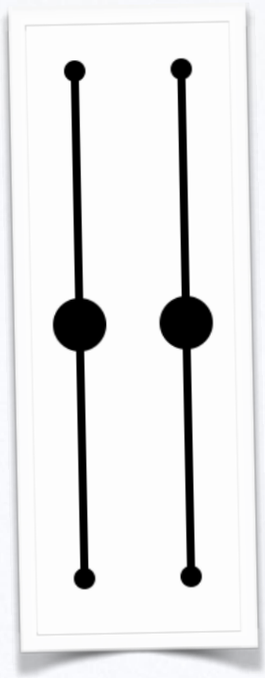


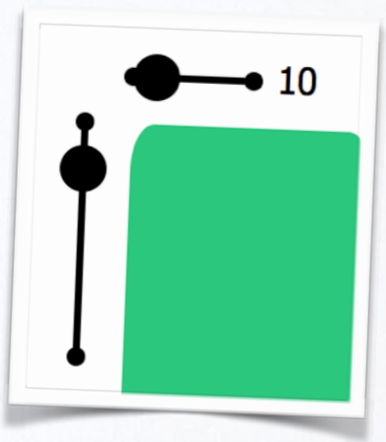

n: 12

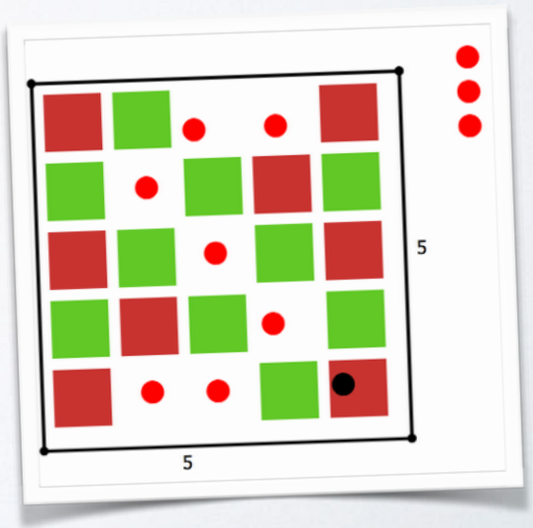

30!

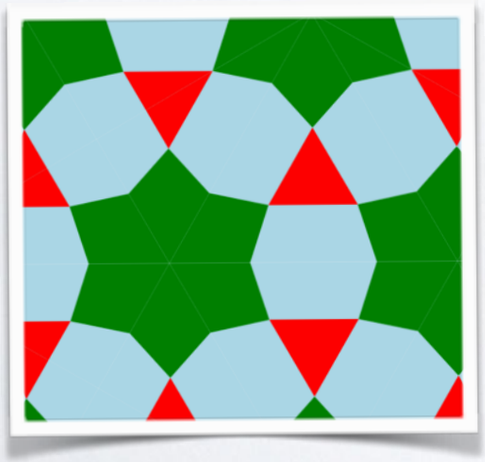


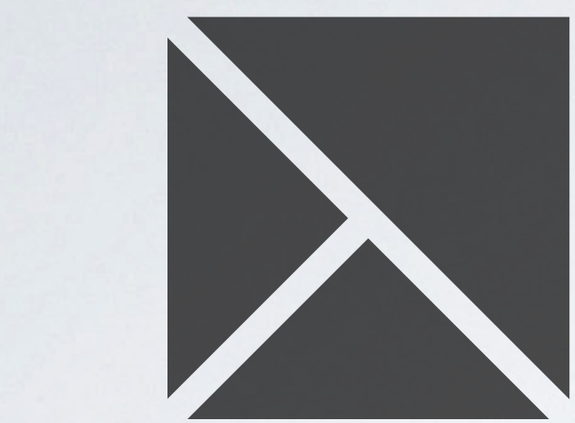





10

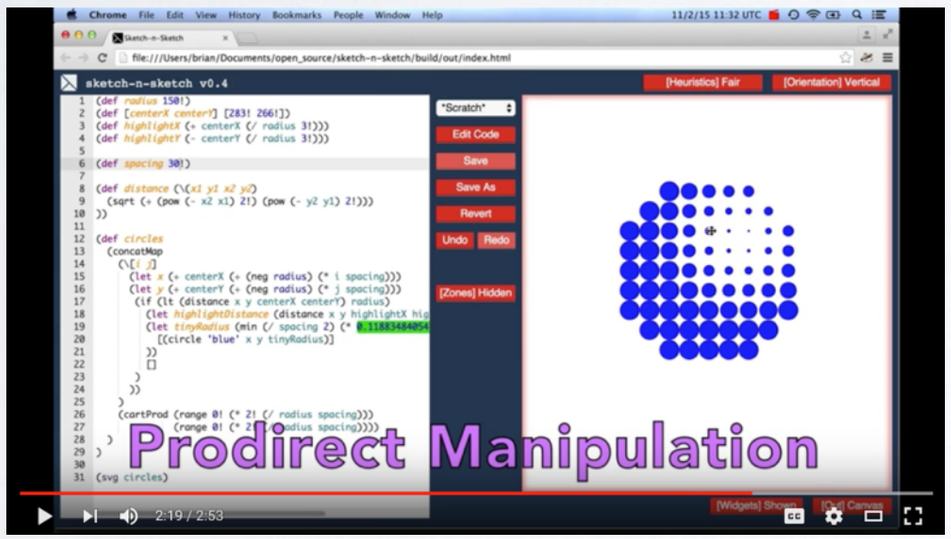

5
5





Sketch-n-Sketch

https://ravichugh.github.io/sketch-n-sketch



Demos on YouTube

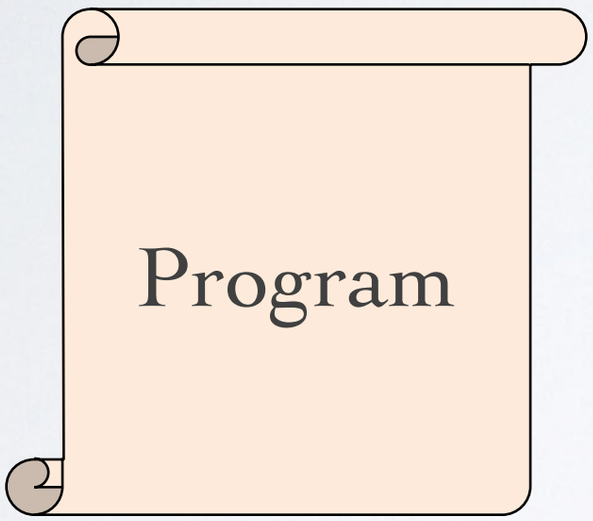
Just google “sketch n sketch”

FUTURE WORK

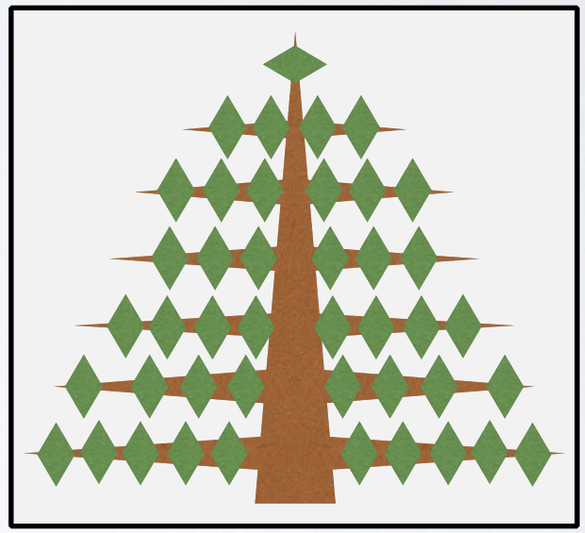
Traces

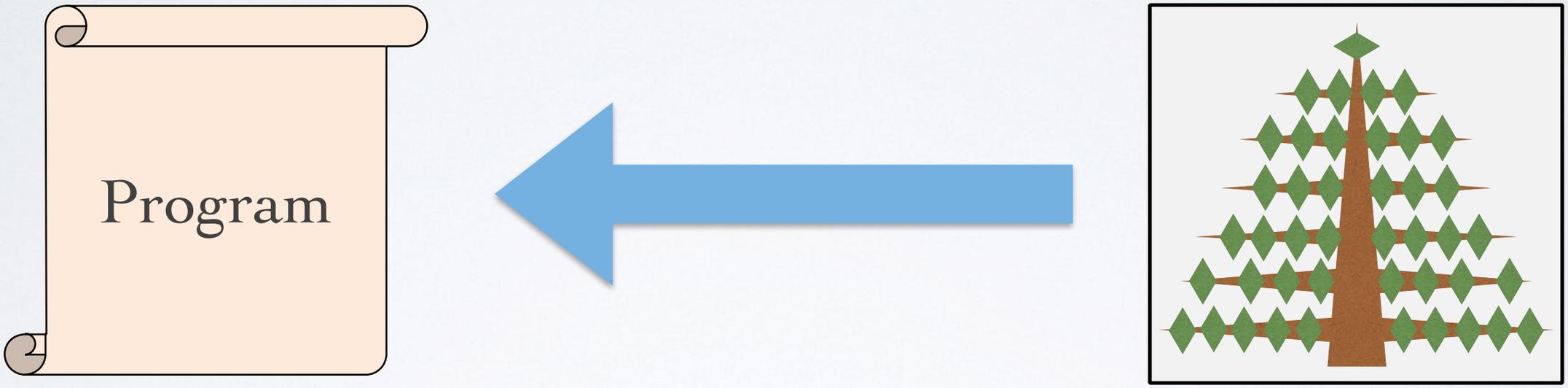
Heuristics

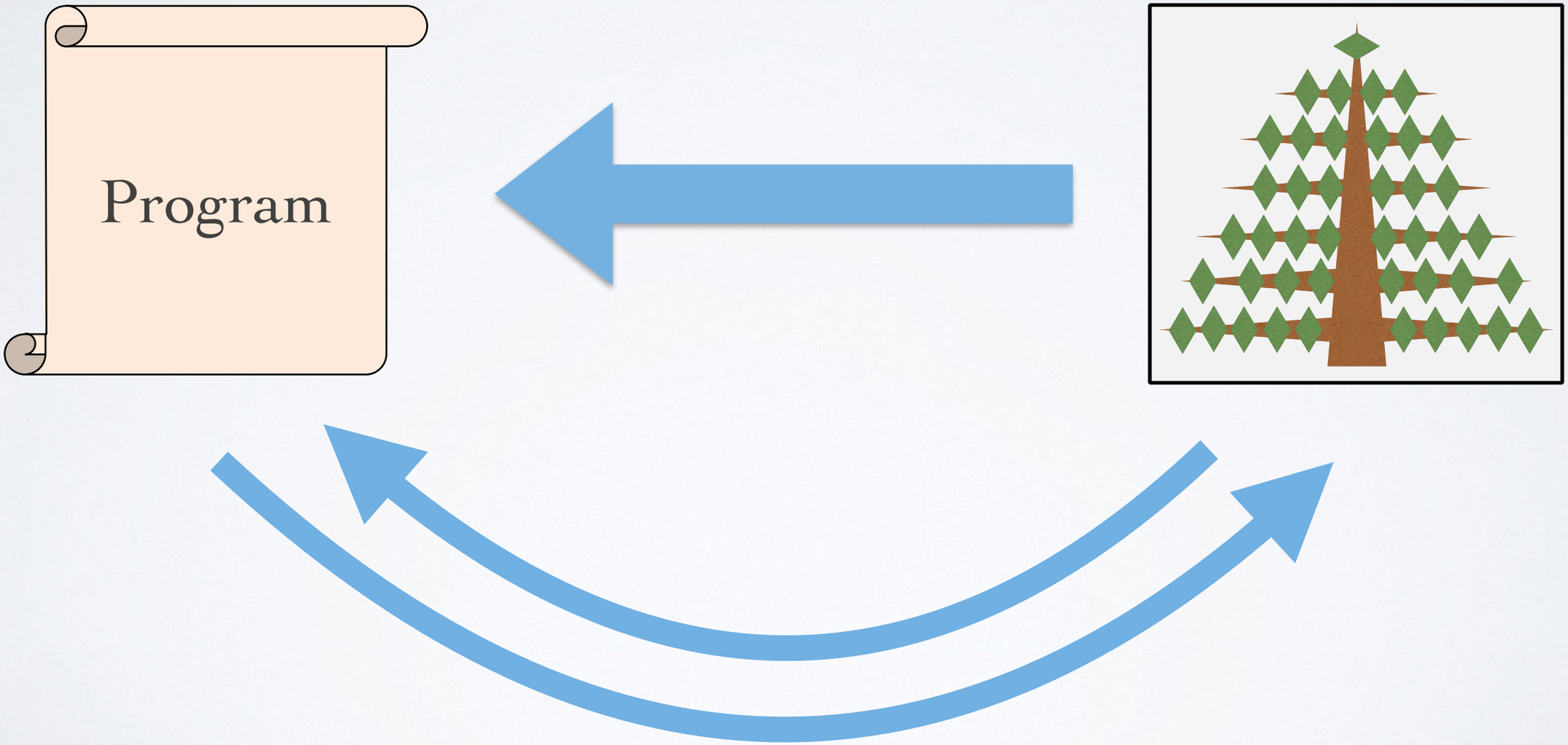
Solver

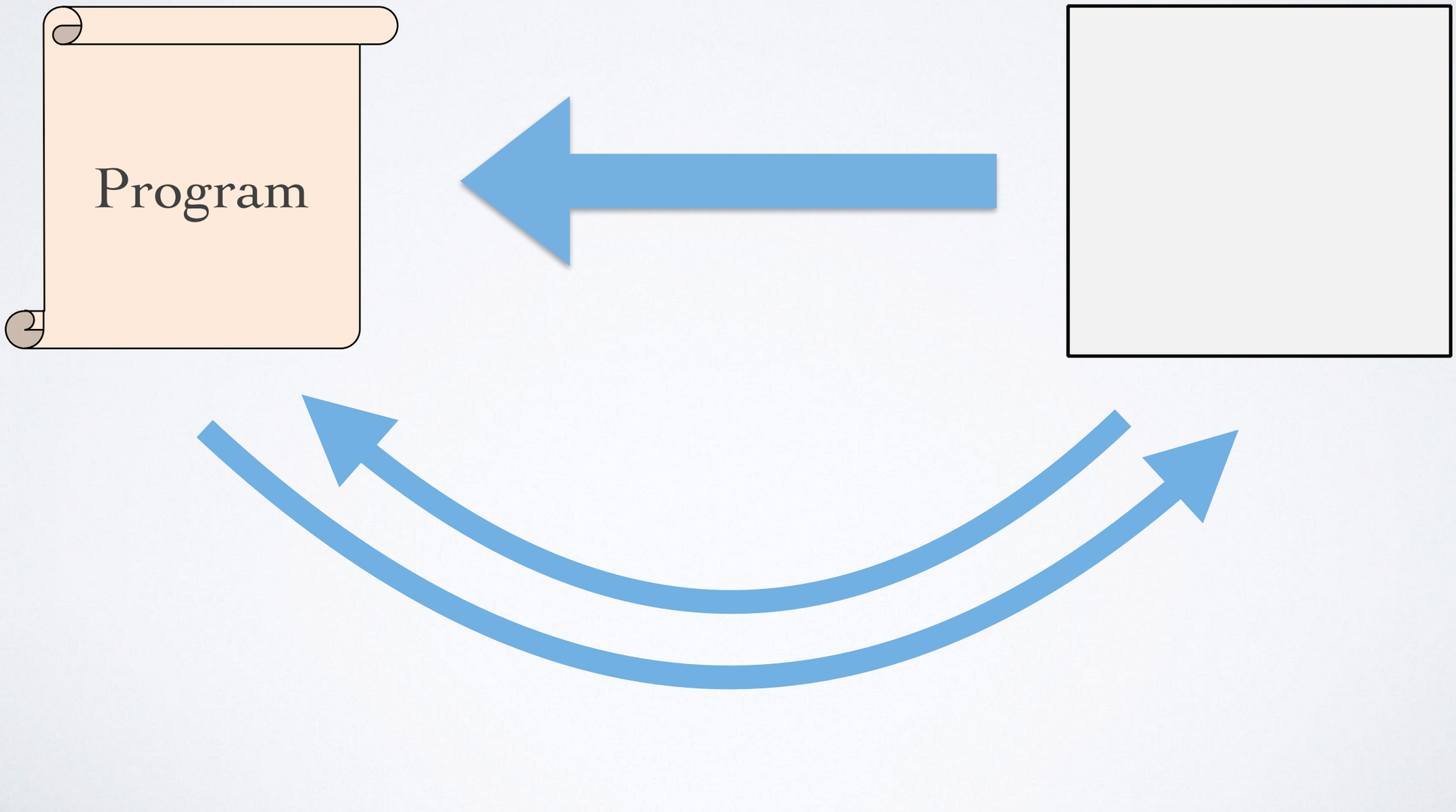

Program


Program



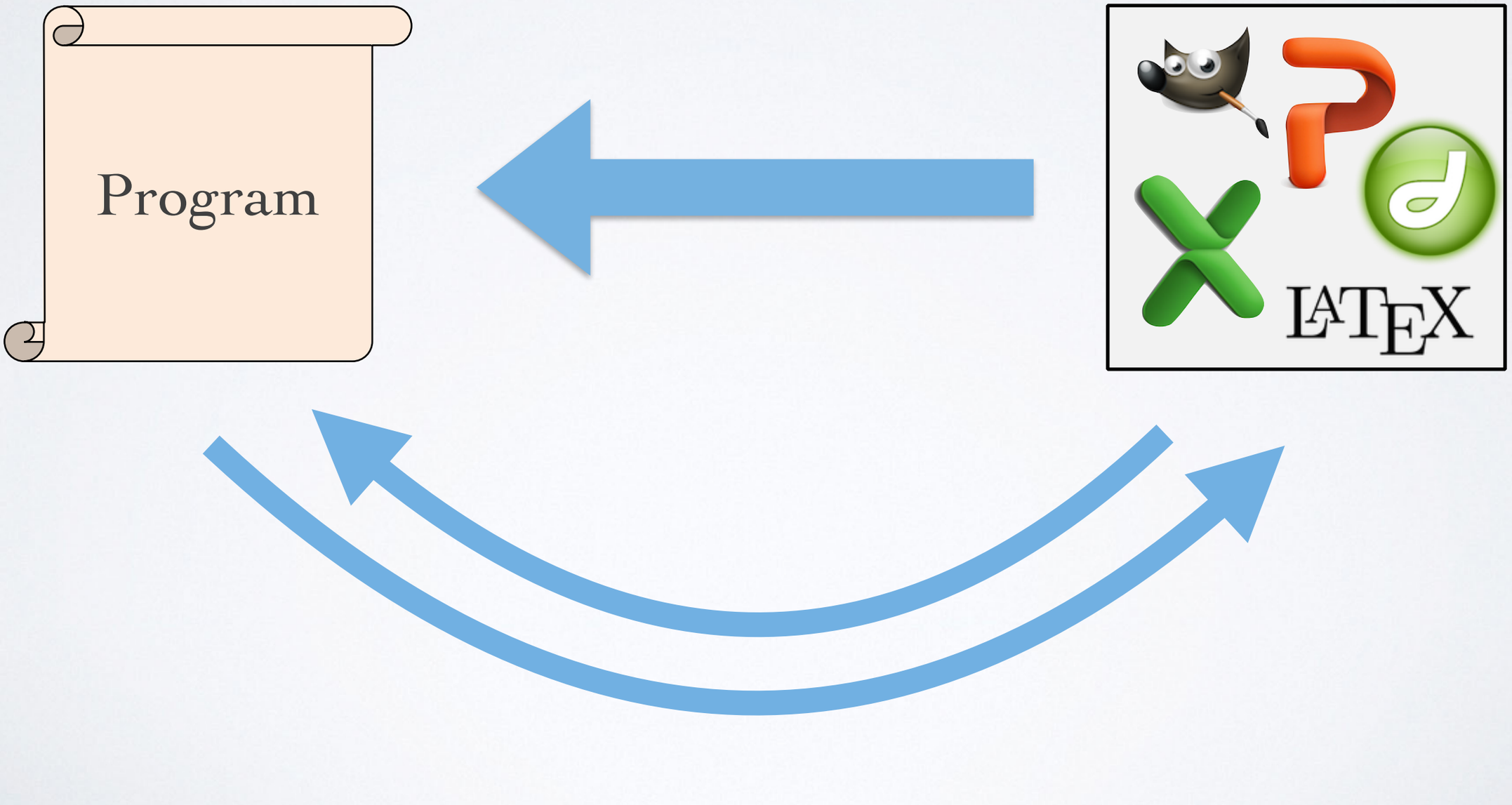

Program


Program


Program

PROGRAMMATIC AND DIRECT MANIPULATION

IN OTHER DOMAINS



RELATED WORK

PROGRAM SYNTHESIS

Old Program: "Sketch"

Constants: "Holes"

User Action: "Example"

PROGRAM ← OUTPUT

String Origin Analysis for PHP by Wang et al.

String Literals

LIVE SYNC

Numbers
+ Heuristics

CONSTRAINTS

Sketchpad, ThingLab, Babelsberg, …

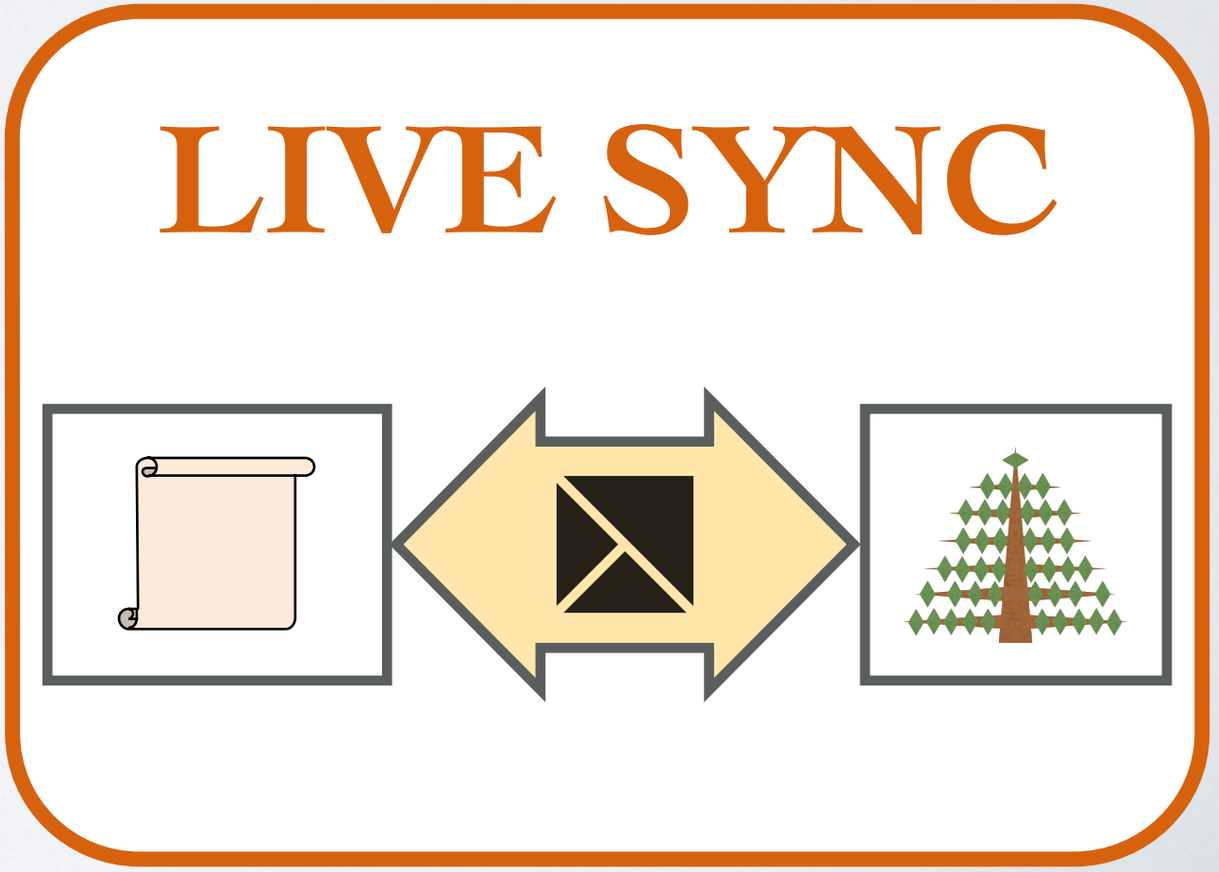
constraints in language

LIVE SYNC

constraints in synthesis only

BX PROGRAMMING

Boomerang, biXid, Beanbag…



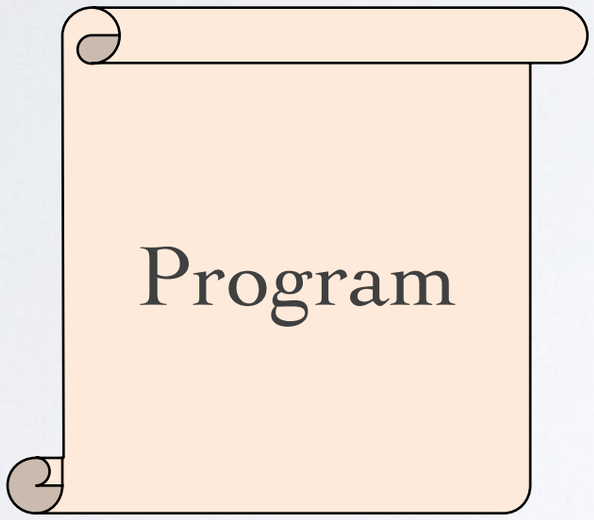
PROG. BY MANIPULATION

Victor, Apparatus, PBM for Layout, …

DM + DSL

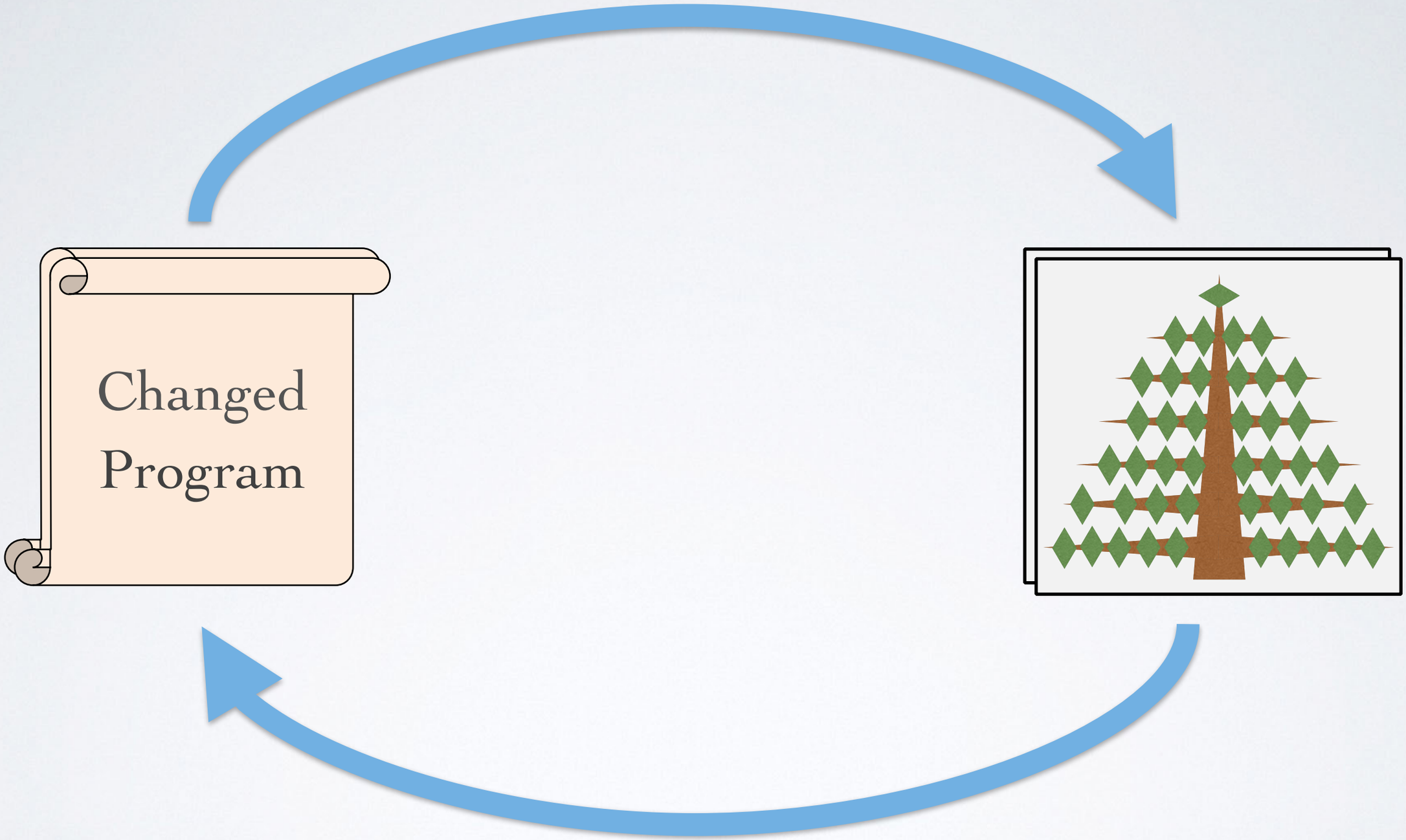
LIVE SYNC

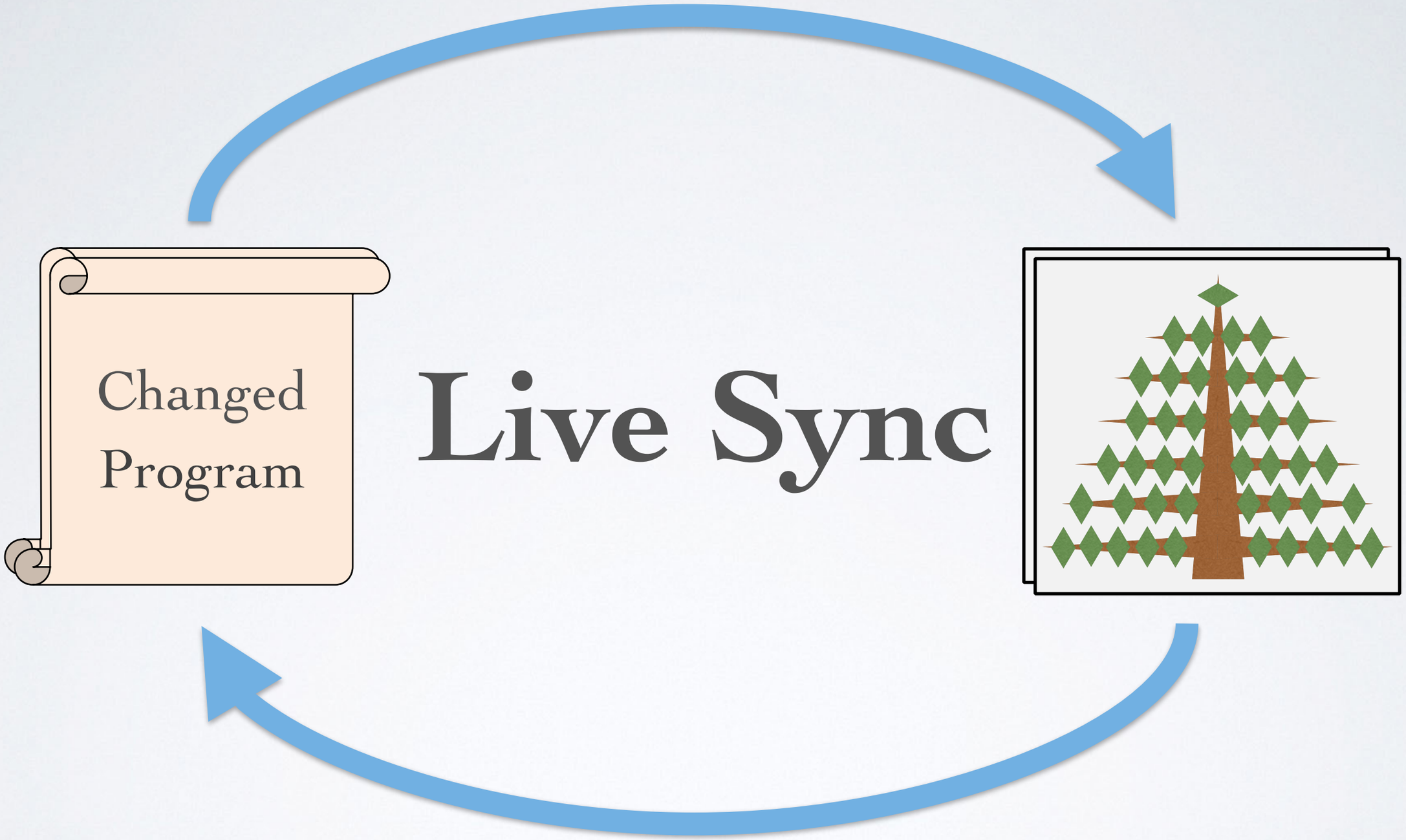
λ-calc + DM


Program


Program


Program

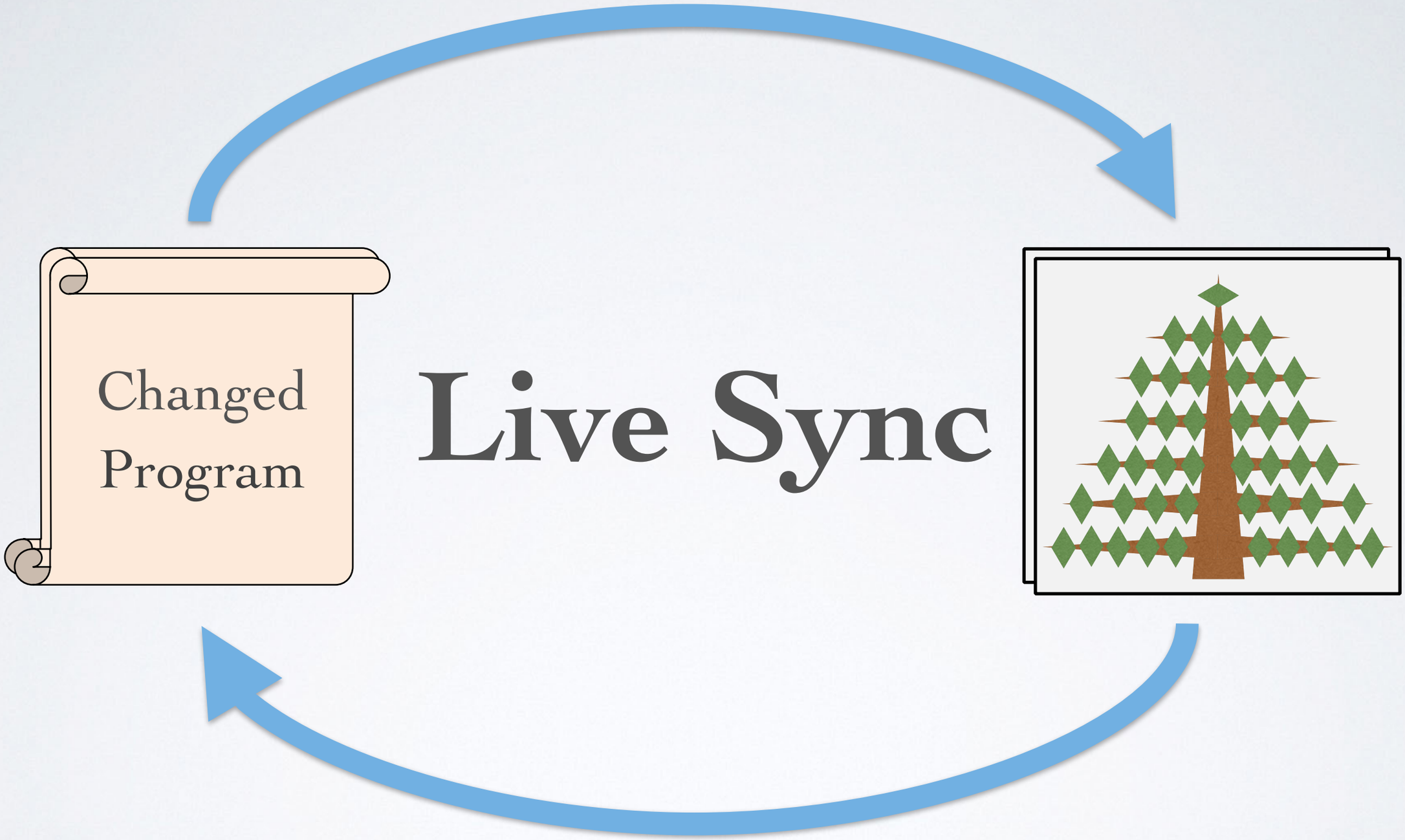

Changed
Program


Changed
Program
Live Sync

Live Sync

Fast + Intuitive + Automatic

Thank You!

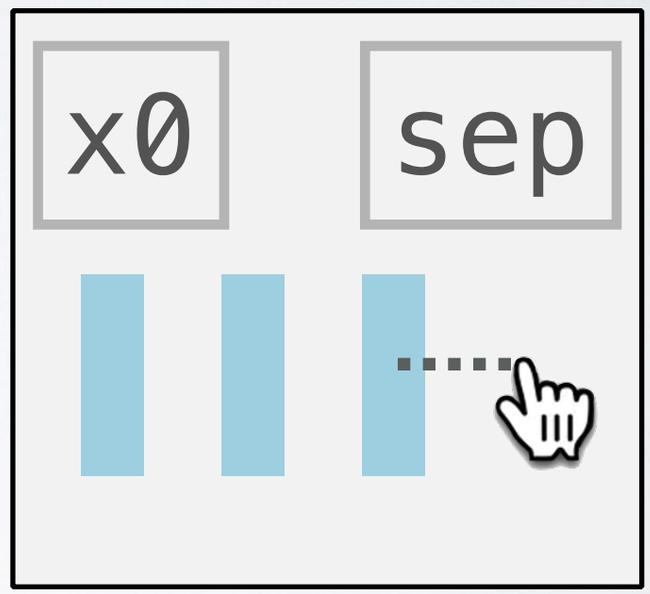


Fast + Intuitive + Automatic

1. Evaluate with **Traces**

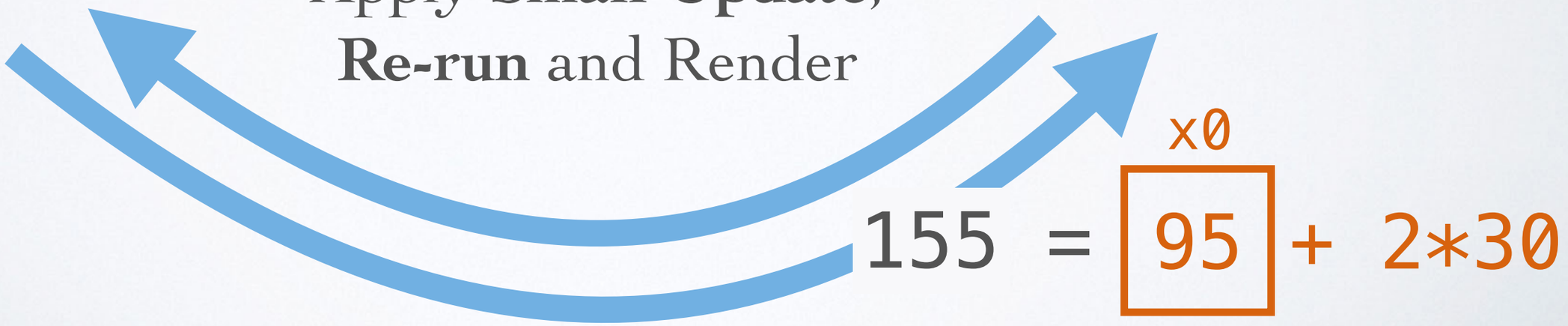
2. Prepare for User Actions with **Heuristics**

```
let (x0, y0,  w,  h,  sep, n) =
    (50, 120, 20, 90, 30,  3)

let boxi i =
  let xi = x0 + i * sep in
  rect 'lightblue' xi y0 w h

map boxi (zeroTo n)
```



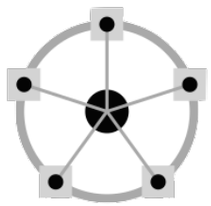
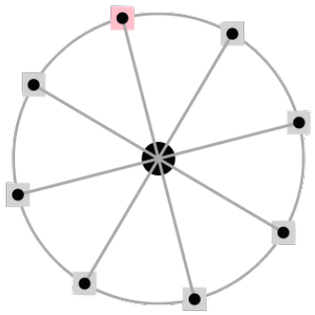
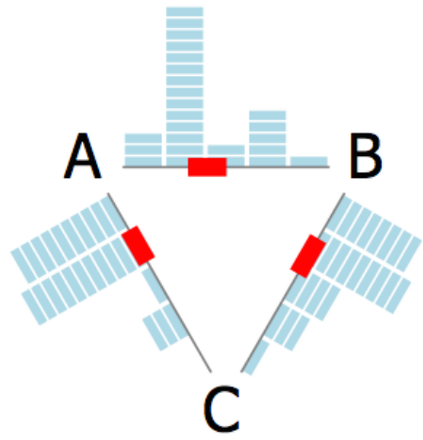
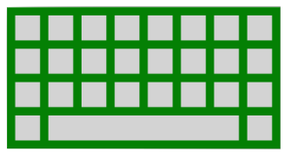
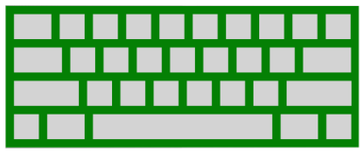
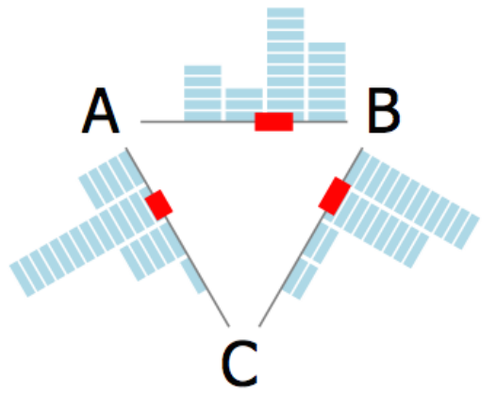
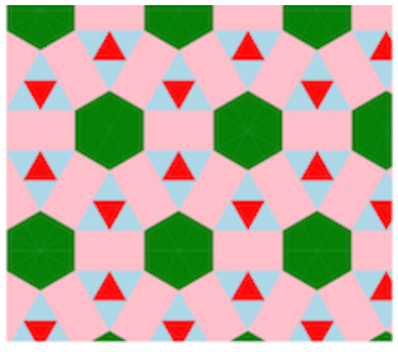
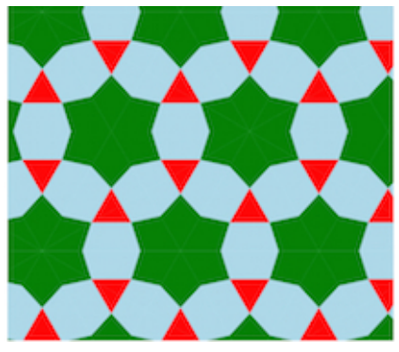
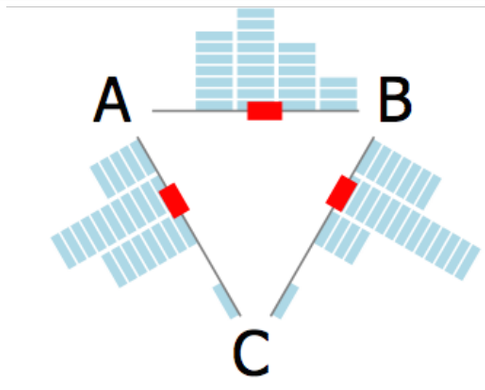
3. Live Sync

User **Changes** Output;
Solve Trace Equation;
Apply **Small Update**;
Re-run and Render



Extra Slides

USER STUDY

Before	After	Histograms
		
		
		

TIMINGS

Operation	Min	Med	Avg	Max
Parse	9 ms	53 ms	77 ms	520 ms
Eval	<1 ms	5 ms	12 ms	165 ms
Prepare	1 ms	13 ms	200 ms	6,789 ms
Solve	<1 ms	<1 ms	<1 ms	14 ms

SOLVABILITY

Unique Pre-Equations	**4,574**	
Outside Fragment	919	20%
Inside Fragment	3,655	
No Solution for $d = 1$	194	4%
Solution for $d = 1$	3,461	
No Solution for $d = 100$	438	10%
Solution for $d = 100$	3,023	66%

AMBIGUITY

Zones	**14,106**	
Inactive	991	7%
Active	13,115	
Unambiguous	4,856	34%
Ambiguous	8,259	59%