ORC Layout: Adaptive GUI Layout with OR-Constraints

Yue Jiang University of Maryland, College Park

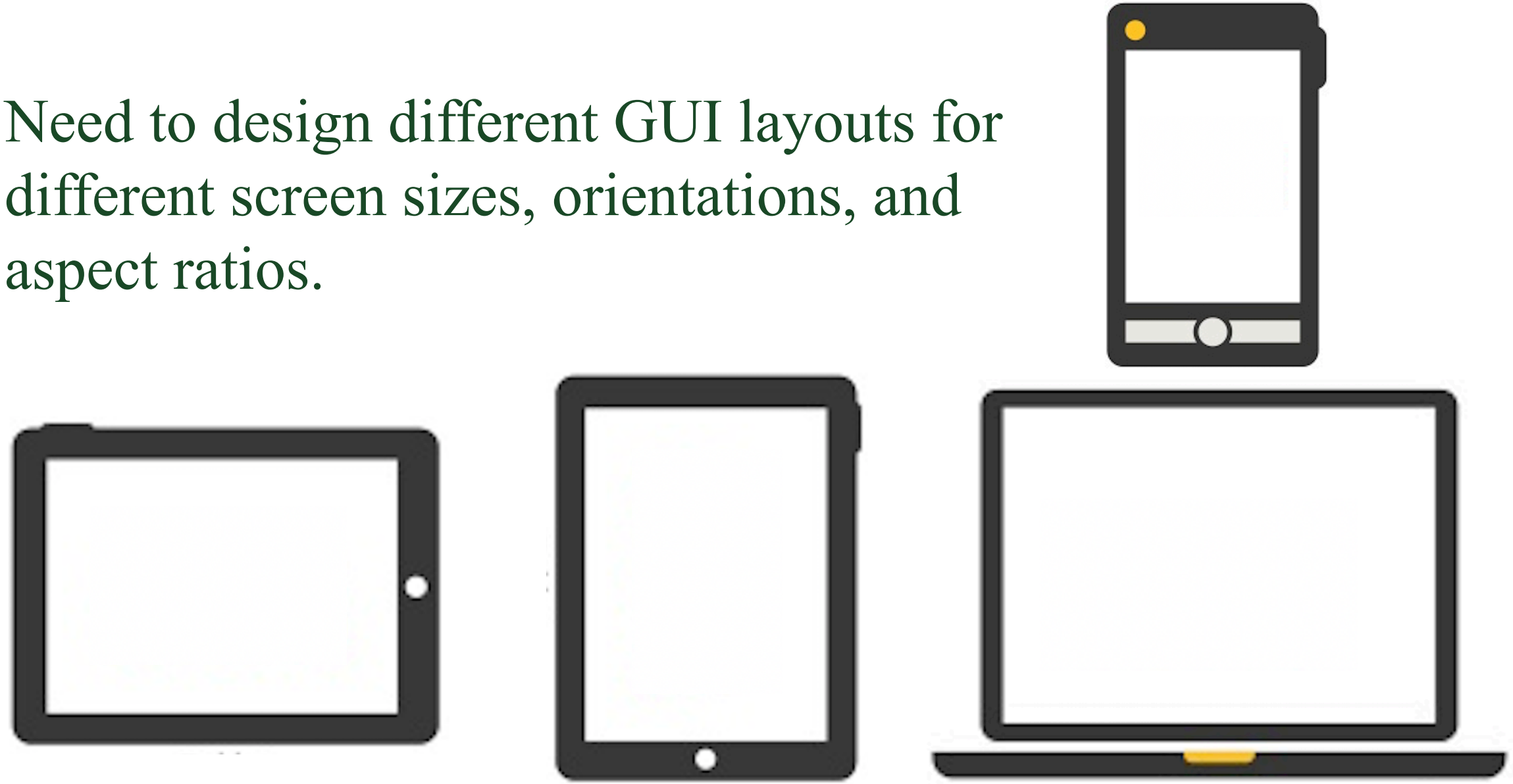
Ruofei Du Google, San Francisco

Christof Lutteroth University of Bath, Bath, United Kingdom

Wolfgang Stuerzlinger SIAT, Simon Fraser University, Vancouver, Canada

Motivation

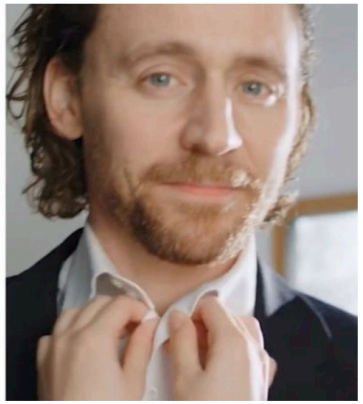
Need to design different GUI layouts for different screen sizes, orientations, and aspect ratios.



Flow Layout

BBC News

Must See



Why China loves Tom Hiddleston's 'creepy' ad

How to Watch

BBC World News TV
The latest global news, sport, weather and documentaries

Listen Live

BBC World Service Radio
Stories from around the world

'We've marched 270 miles for Brexit'

Rare tiger cubs make debut at Sydney Zoo

Australian lake an Insta hit after turning naturally pink

Prodigy fans 'raise roof' at Keith Flint funeral

Quiz of the Week: Who's the 'culturally significant' rapper?

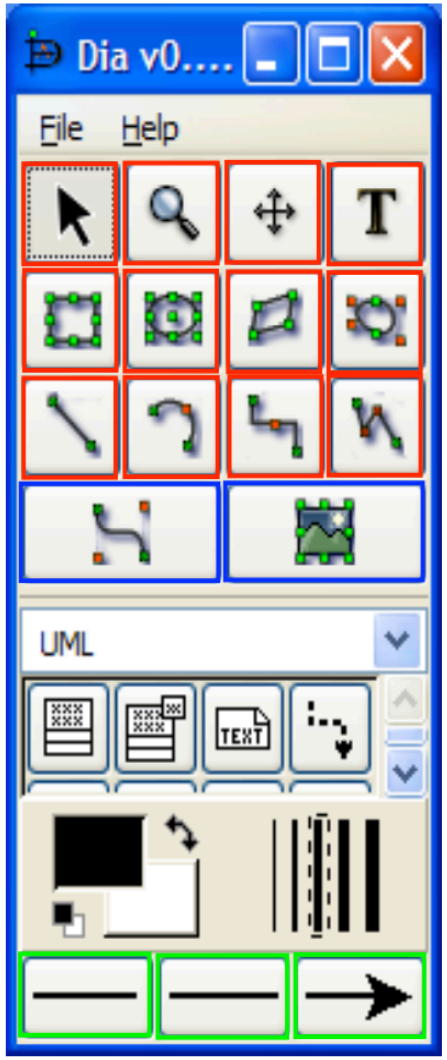
Most watched

1. ▶ Rare tiger cubs make debut at Sydney Zoo
2. ▶ First-time cover star at the age of 80
3. ▶ 'It's vital that children can see that people like me exist'
4. ▶ Why Trudeau's sorry for saying thank you
5. ▶ 'We've marched 270 miles for Brexit'

Full Story

Limitation: cannot restrict positions and relative sizes

Constraint-based Layout



Constraints:

Same size → Size(Red1) == Size(Red2) == …

Same height as above
Double width as above
→
Height(Blue) == Height(Red)
Width(Blue) == Width(Red) * 2

Limitations:

1. Widgets cannot move relative to other ones.
2. Device diversity a long-term challenge.

OR-constrained Layouts (ORC Layouts)

Goal: Unify constraint-based and flow layouts

Approach: OR-constraints

Input:

1. A set of constraints
2. Widget min/pref/max sizes
3. Window size

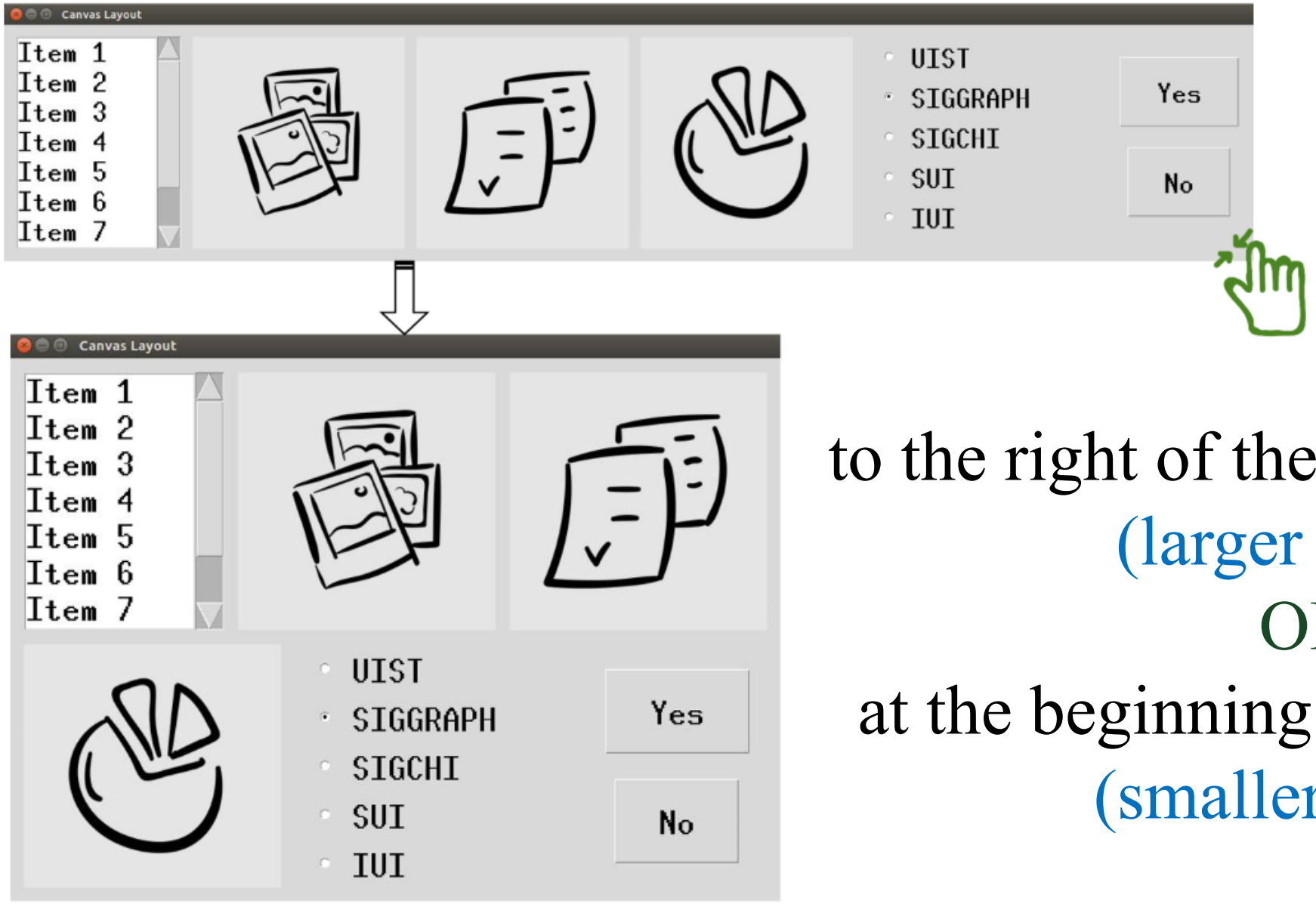
Output:

1. Widget sizes
2. Widget positions

OR-Constraints

- **Hard Constraints** must be satisfied.
- **Soft Constraints** are satisfied if possible.
  Their importance depends on weights.

OR-Constraints



to the right of the previous widget
(larger weight)
OR
at the beginning of the next row
(smaller weight)

Z3 Solver

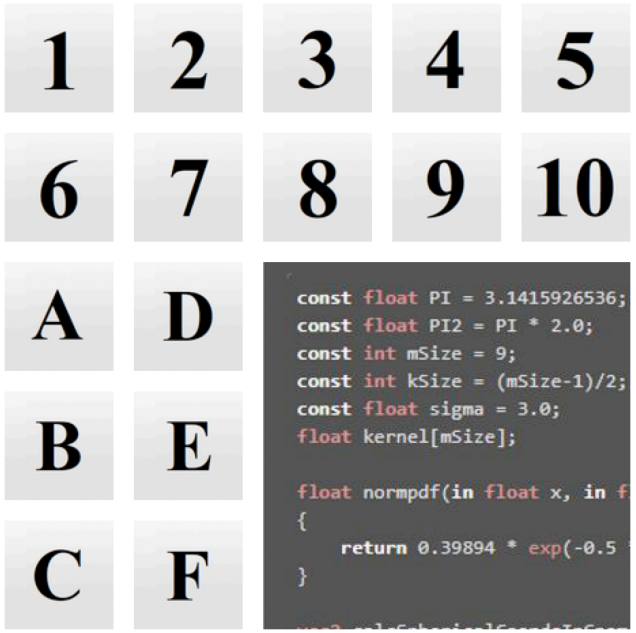
OR Constraints → more branches

Microsoft Z3 Solver:

- Can solve OR-constraints
- Support incremental solving
  (fast if #widget not too large)

ORC Patterns

Low-level constraints tedious and error prone



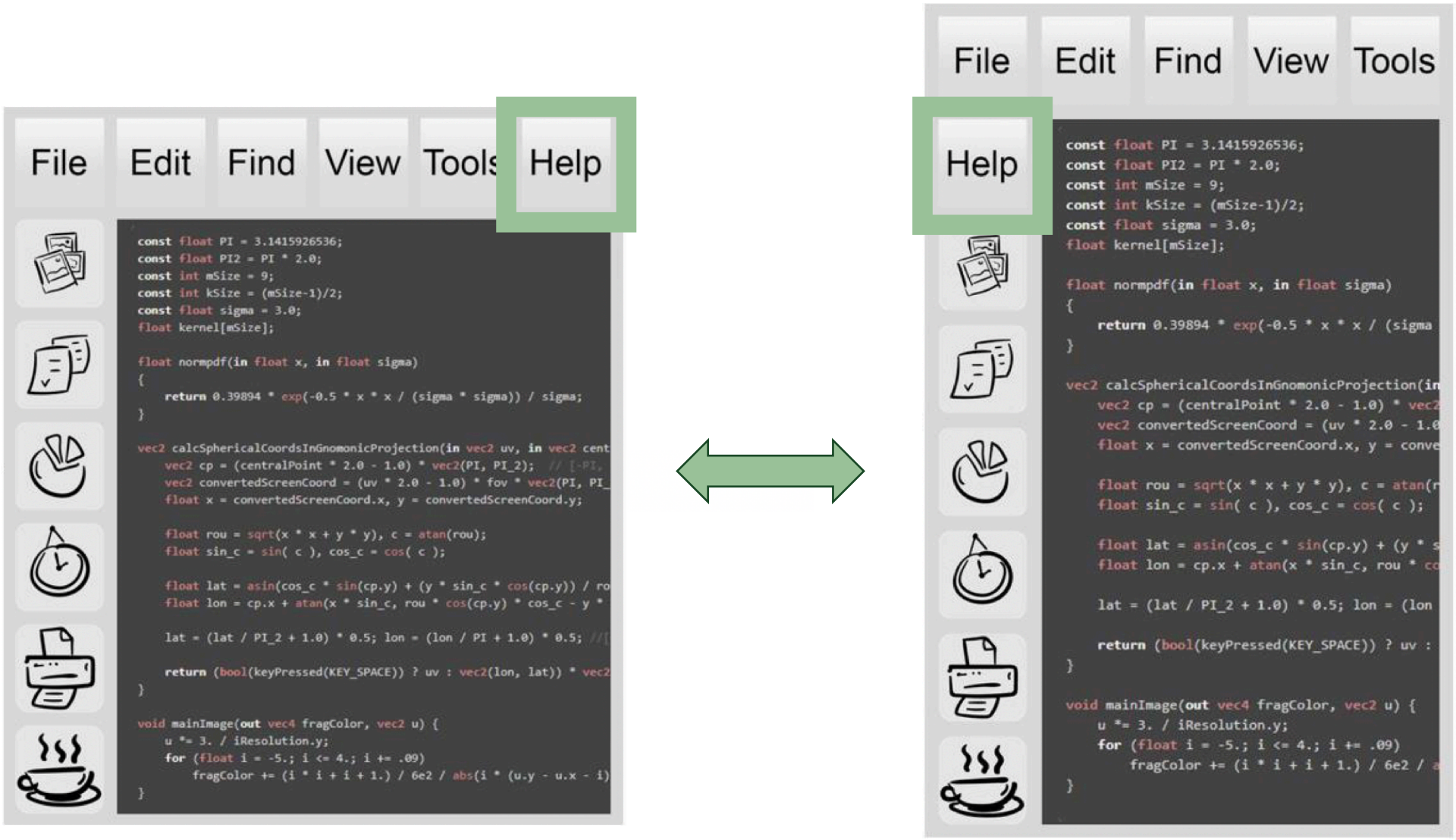
```
(assert (and (= w_1_1 b1) (= h_1_1 b3)))
(assert (and (<= w_1_2 b2) (<= h_1_2 b4)))
(assert (and (<= w_2_2 b2) (<= h_2_2 b4)))
(assert (and (<= w_3_2 b2) (<= h_3_2 b4)))
(assert (or (and (= w_2_1 (+ w_1_2 15)) (= h_2_1 h_1_1))
    (and (= w_2_1 15) (>= h_2_1 (+ h_1_2 15)) (= h_2_1 (+ h_1_2 15)))))
(assert (let ((a!1 (and (= w_3_1 15)
                (>= h_3_1 (+ h_1_2 15))
                (>= h_3_1 (+ h_2_2 15))
                (or (= h_3_1 (+ h_1_2 15)) (= h_3_1 (+ h_2_2 15))))))
  (or (and (= w_3_1 (+ w_2_2 15)) (= h_3_1 h_2_1)) a!1)))
(assert (and (= b1 15) (= b2 475) (= b3 15) (= b4 475)))
(assert (= (- w_1_2 w_1_1) (- b2 b1)))
(assert (let ((a!1 (and (= (- w_2_2 w_2_1) 80) (>= (- (- b4 h_1_2) 15) 555)))
      (a!2 (and (= (- w_2_2 w_2_1) 175) (< (- (- b4 h_1_2) 15) 555))))
  (or a!1 a!2)))
(assert (= (- w_3_2 w_3_1) (- (- b2 w_2_2) 15)))
(assert (= w_3_1 (+ w_2_2 15)))
(assert (or (and (= (- h_1_2 h_1_1) 80) (>= (- b2 b1) 935))
    (and (= (- h_1_2 h_1_1) 175) (< (- b2 b1) 935))))
(assert (= (- h_2_2 h_2_1) (- (- b4 h_1_2) 15)))
(assert (= (- h_3_2 h_3_1) (- (- b4 h_1_2) 15)))
(assert (= h_3_1 h_2_1))
(assert-soft (= w_2_1 (+ w_1_2 15)) :weight 14900)
(assert-soft (= h_2_2 h_1_2) :weight 1000)
(assert-soft (= w_3_1 (+ w_2_2 15)) :weight 14800)
(assert-soft (= h_3_2 h_2_2) :weight 1000)
```

Better approach:

Designers → choose a template & modify parameters

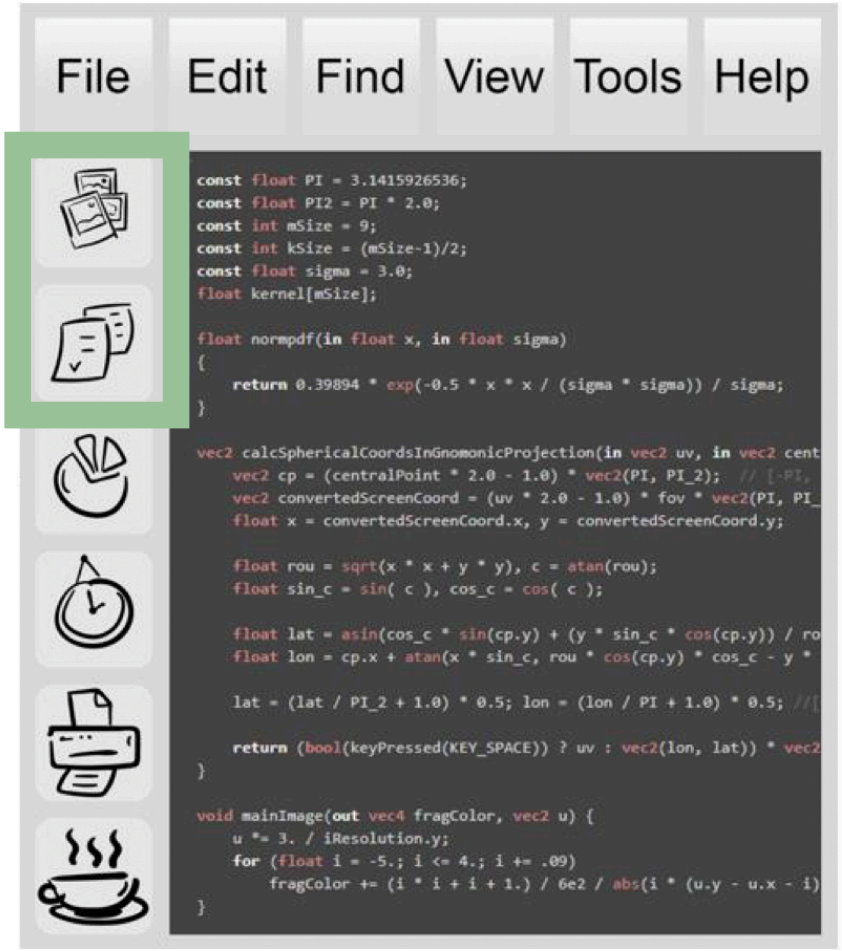
System → automatically maintain low-level constraints

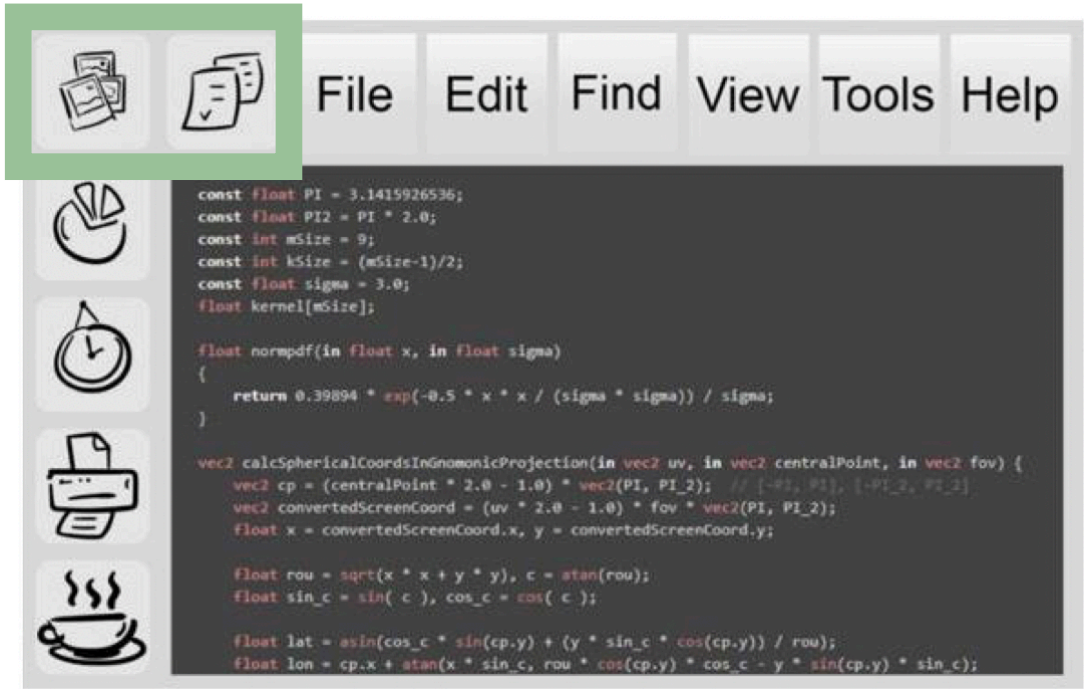
Pattern #1: Connected Layout Pattern



Top toolbar widgets → left toolbar

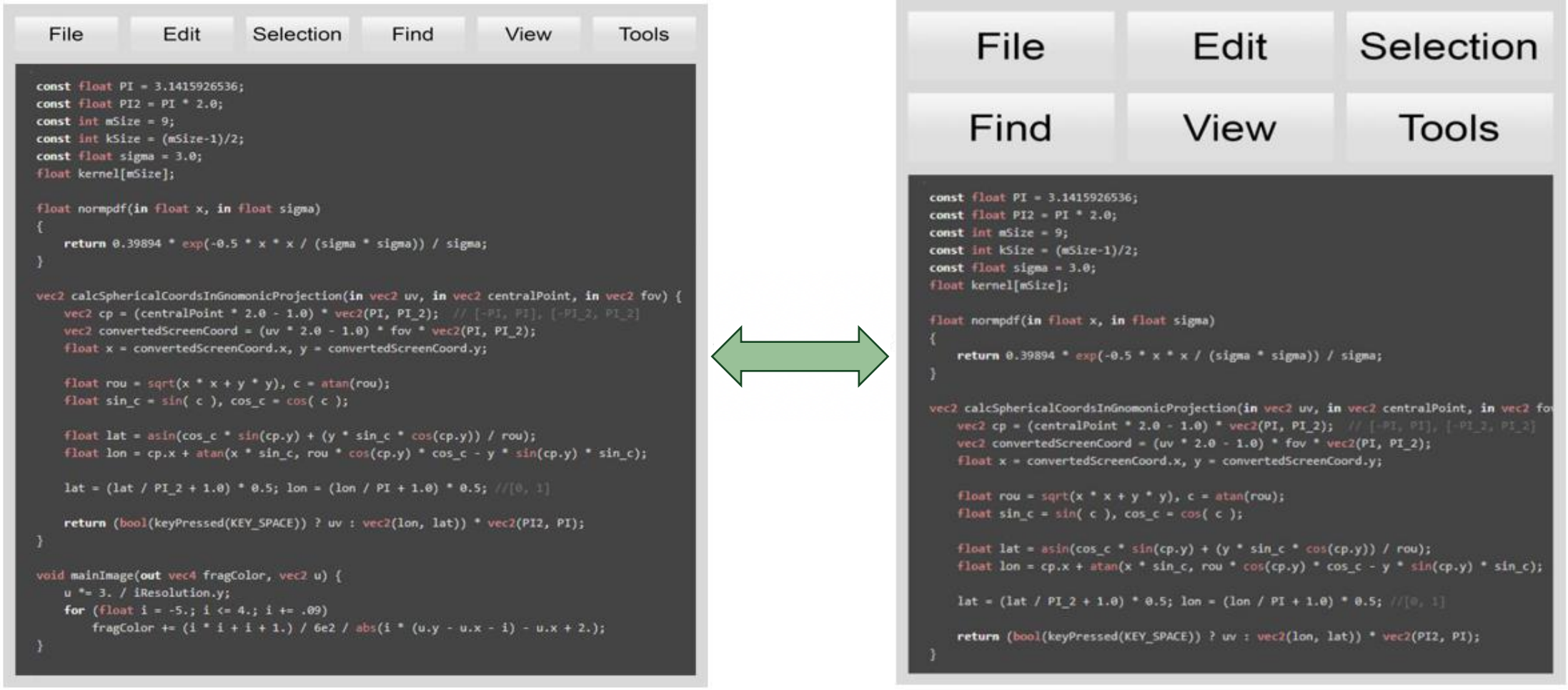
Pattern #1: Connected Layout Pattern





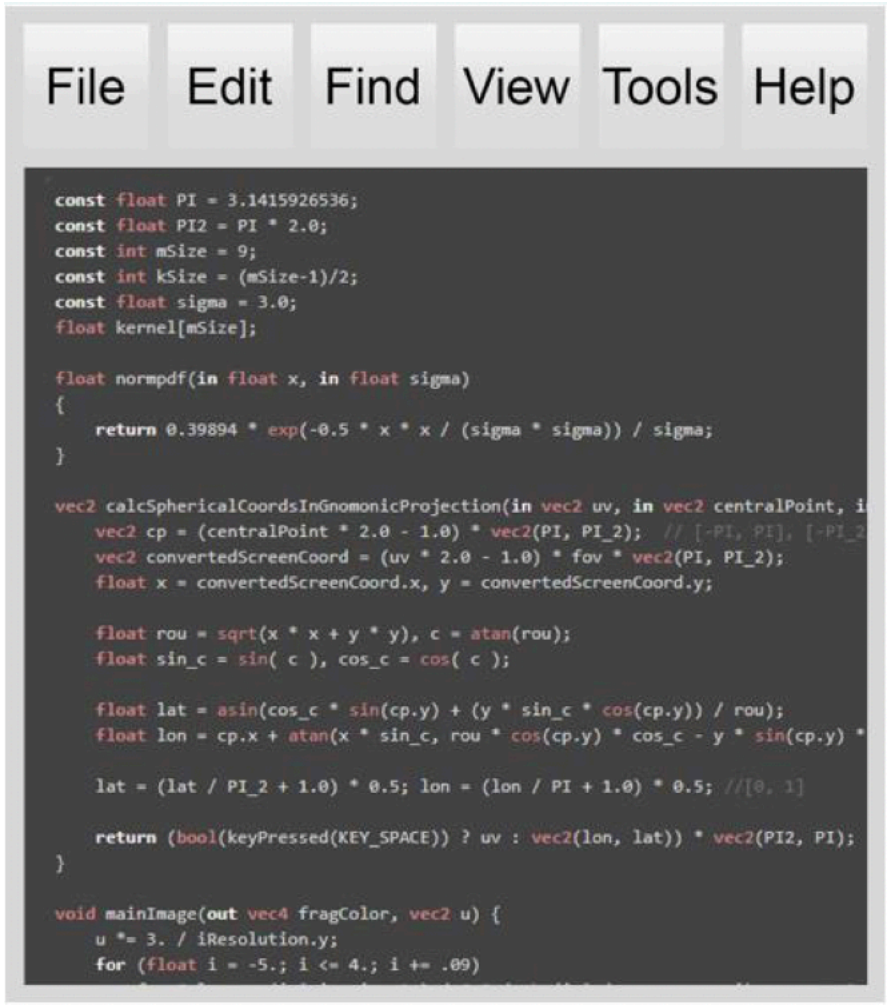
Left toolbar widgets → top toolbar

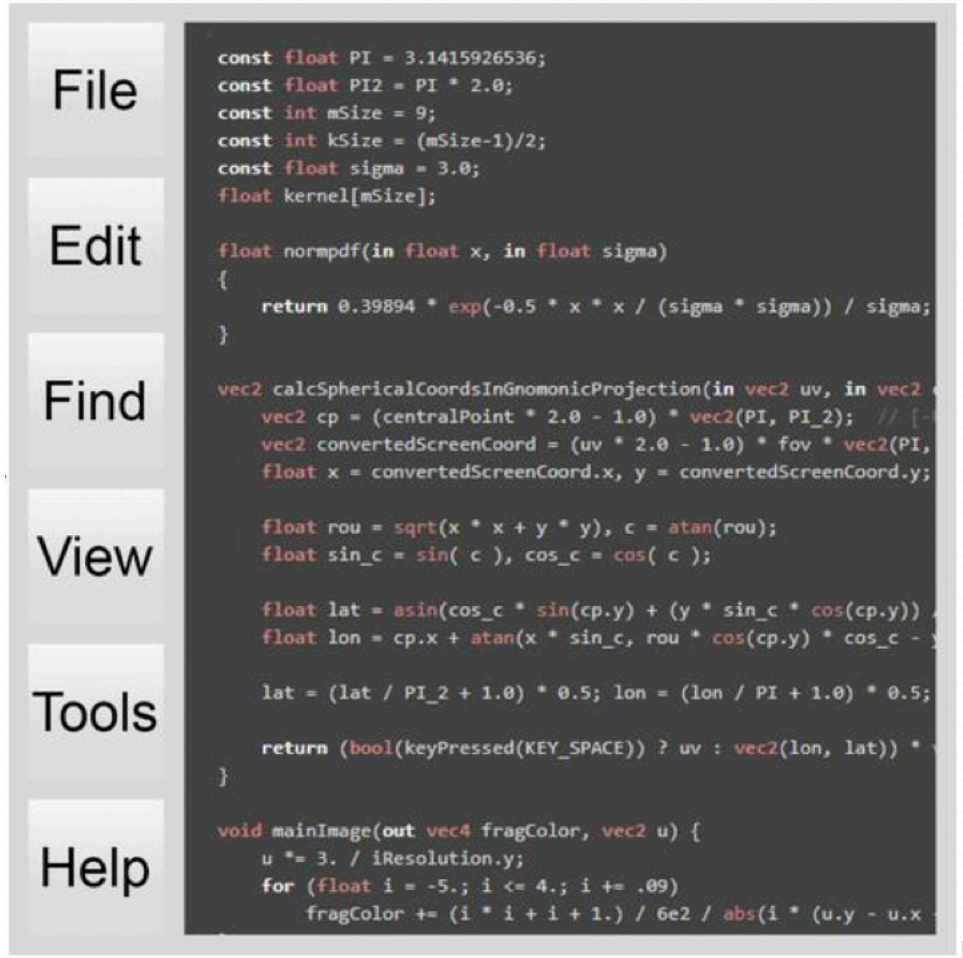
Pattern #2: Balanced Flow Layout Pattern



6 widgets → Each row has 1 OR 2 OR 3 OR 6 widgets in the toolbar

Pattern #3: Alterative Positions Pattern



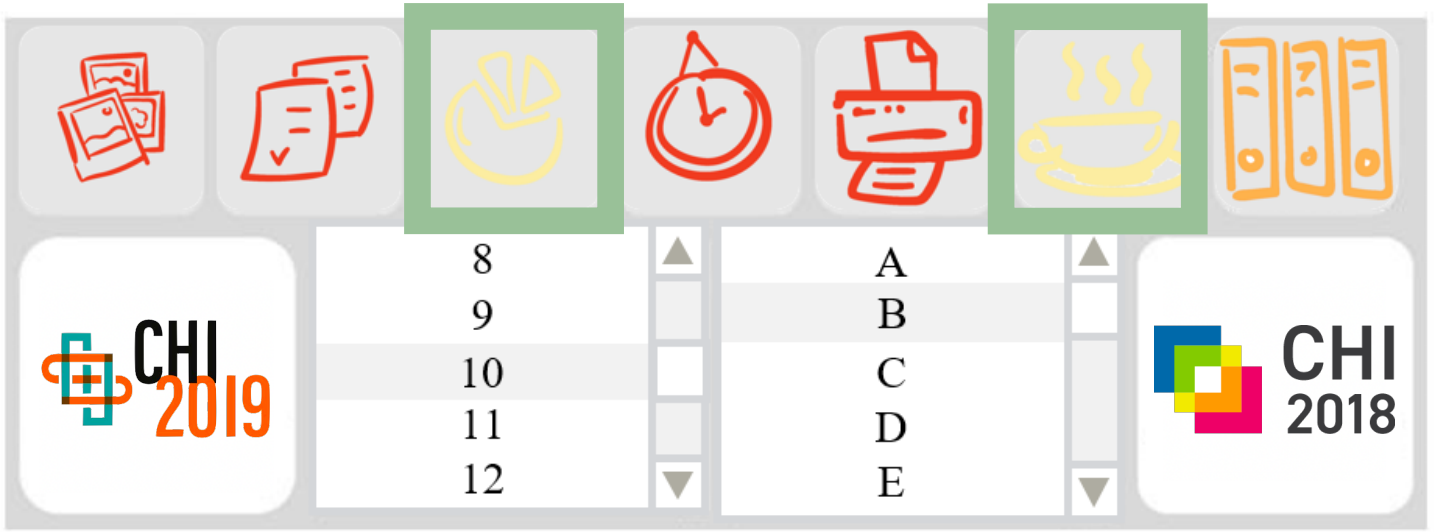


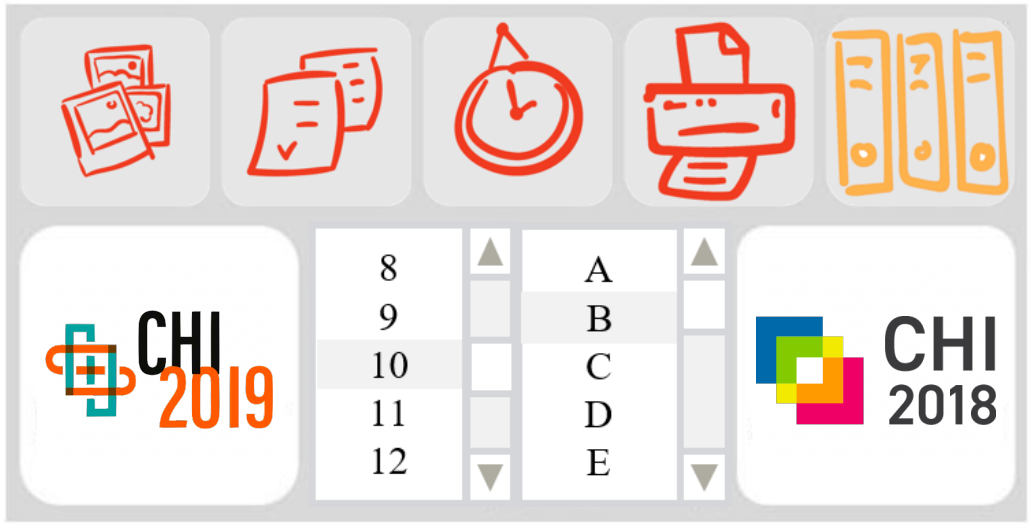
Top Toolbar OR Left Toolbar

(weights depend on which one you prefer)

Pattern #4: Flowing Widgets around a Fixed Area

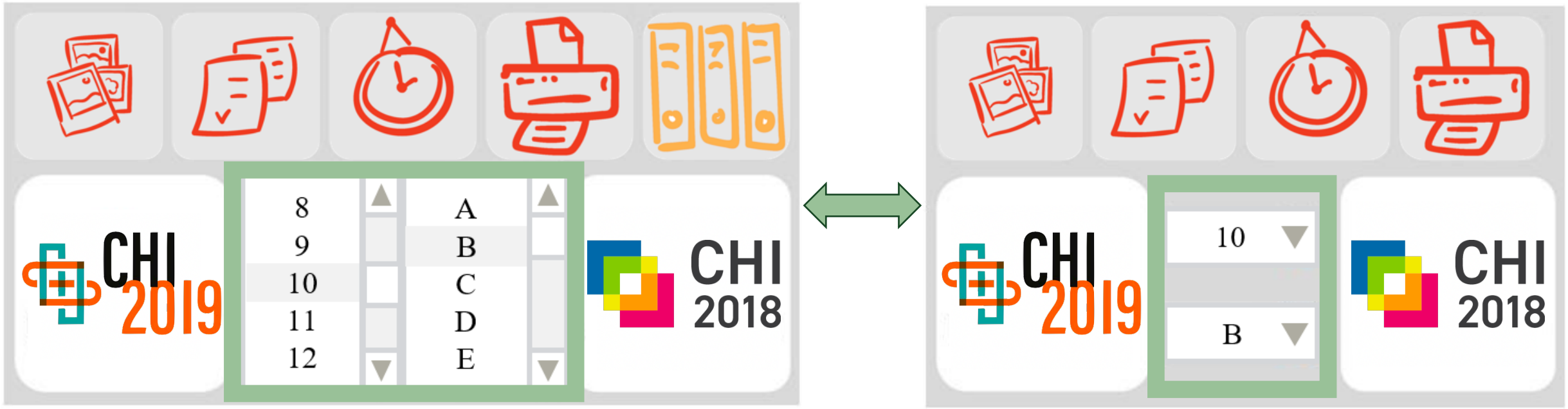
Pattern #5: Optional Layout Pattern





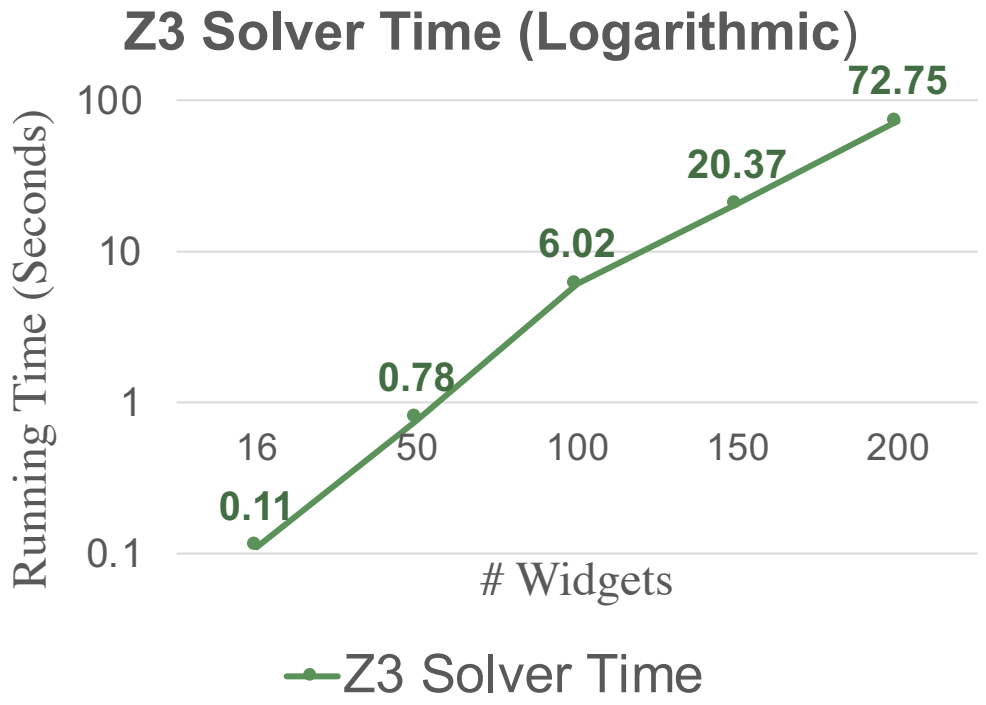
Less important

Pattern #6: Alternative Widget Layout Pattern



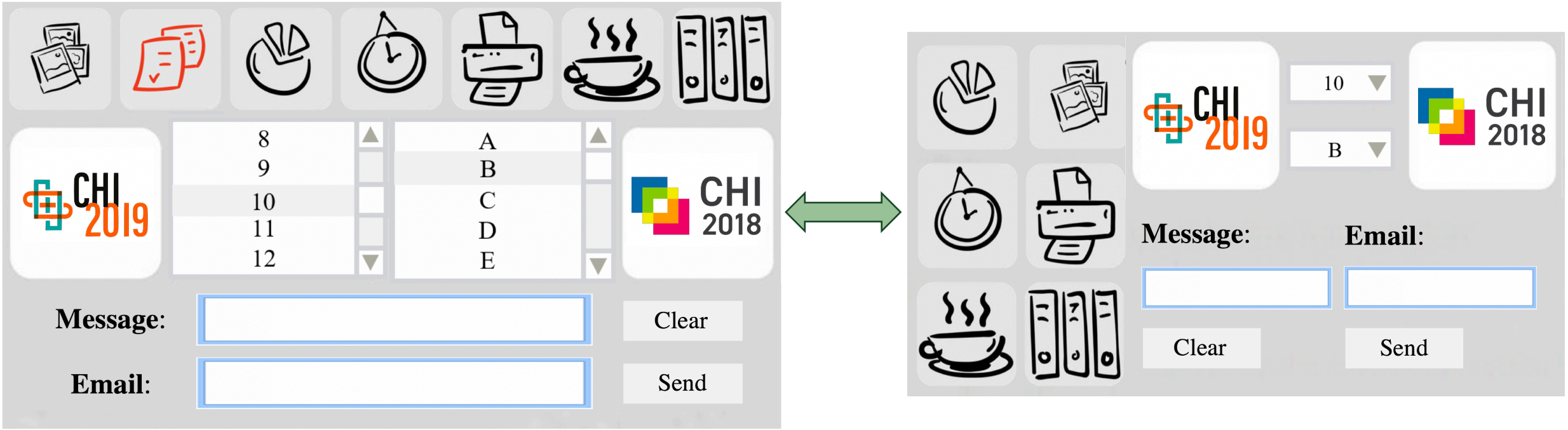
Limitations

- Patterns restrict what designers can create.
- Manual ORC specification potentially error-prone.
- Non-interactive solving time for larger number of widgets



Conclusion

- ORC Layouts
  - Introduce OR-constraints
  - Unify flow & constraint-based layouts
  - Enrich GUI layout design space



Co-authors

Ruofei Du

Google

Christof Lutteroth

UNIVERSITY OF BATH

Wolfgang Stuerzlinger

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

SFU SIMON FRASER UNIVERSITY

Thank you!

Yue Jiang: yuejiang@cs.umd.edu
https://cs.umd.edu/~yuejiang

Contributions:

- Add OR-constraints to standard hard/soft constraint systems.
- Adapt layouts to screens with different screen sizes, orientations, and aspect ratios with only a single specification.
- Unify flow & constraint-based layouts.

Yue Jiang[†], Ruofei Du[†‡], Christof Lutteroth[§], and Wolfgang Stuerzlinger[¶]
[†]University of Maryland, College Park [‡]Google LLC
[§]University of Bath, Bath, United Kingdom [¶]Simon Fraser University, Vancouver, BC, Canada