



# ORC Layout: Adaptive GUI Layout with OR-Constraints

**Yue Jiang**

University of Maryland, College Park

Ruofei Du

Google, San Francisco

Christof Lutteroth

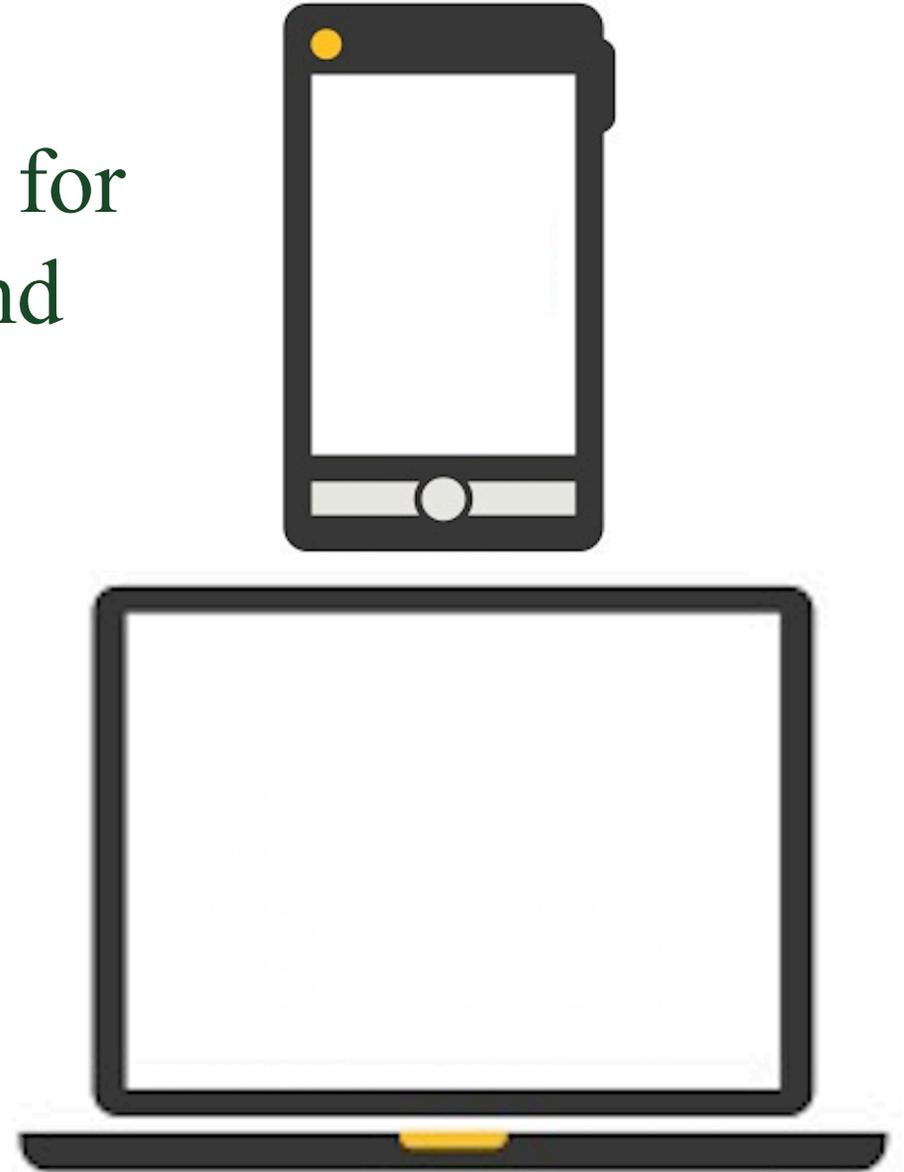
University of Bath, Bath, United Kingdom

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?

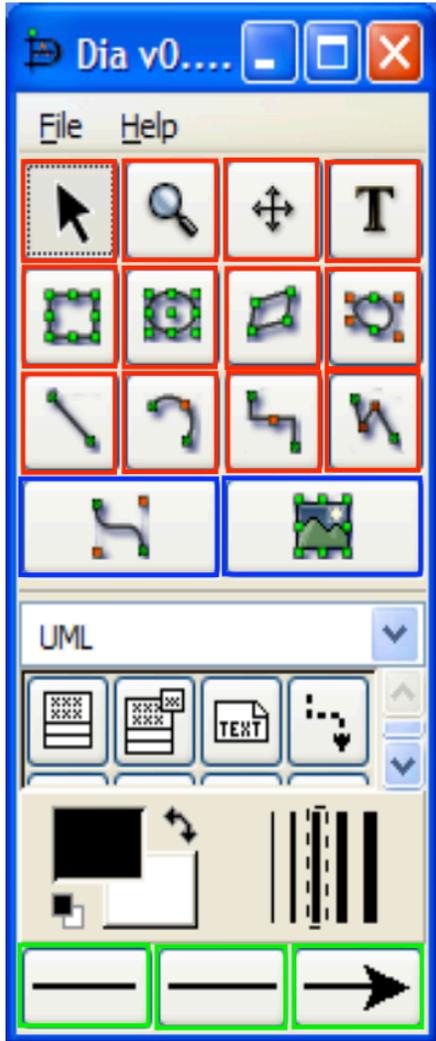
### Most watched

- 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

$$\longrightarrow \text{Size}(\text{Red1}) == \text{Size}(\text{Red2}) == \dots$$

Same height as above

$$\longrightarrow \text{Height}(\text{Blue}) == \text{Height}(\text{Red})$$

Double width as above

$$\text{Width}(\text{Blue}) == \text{Width}(\text{Red}) * 2$$

## 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

Constraint1 OR Constraint2 OR Constraint3 ...

Soft

Soft

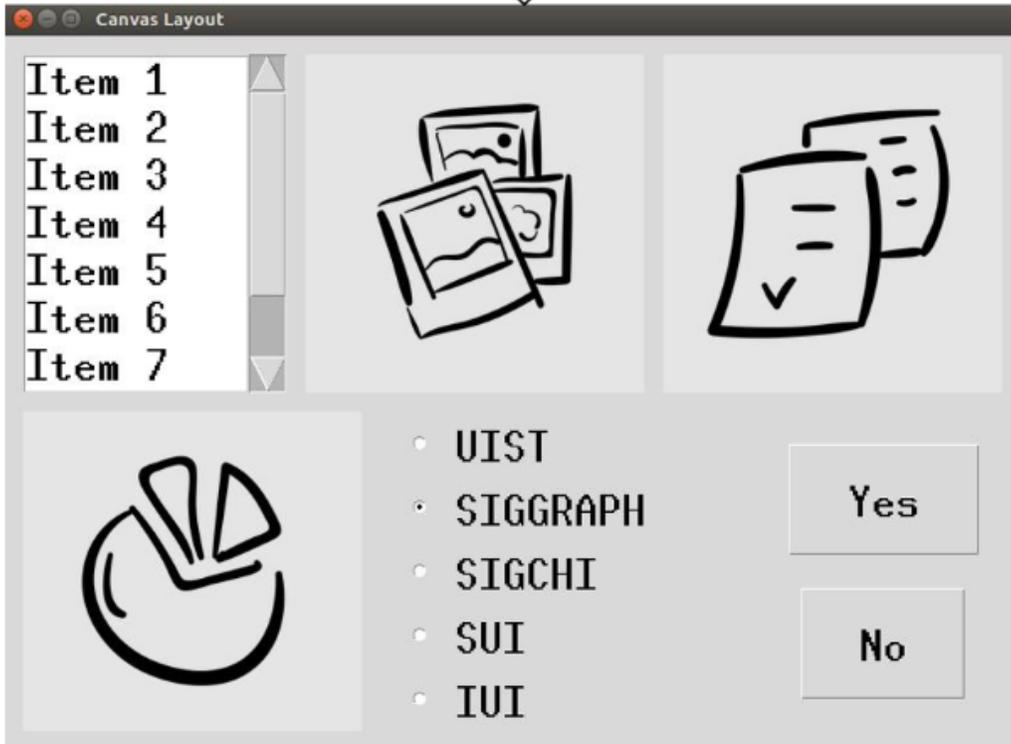
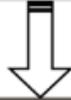
Soft

---

Hard

- **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  $\rightarrow$  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

1	2	3	4	5
6	7	8	9	10
A	D	<pre>const float PI = 3.1415926536; const float PI2 = PI * 2.0; const int mSize = 9; const int kSize = (mSize-1)/2; const float sigma = 3.0; float kernel[mSize];  float normpdf(in float x, in f {     return 0.39894 * exp(-0.5</pre>		
B	E			
C	F			



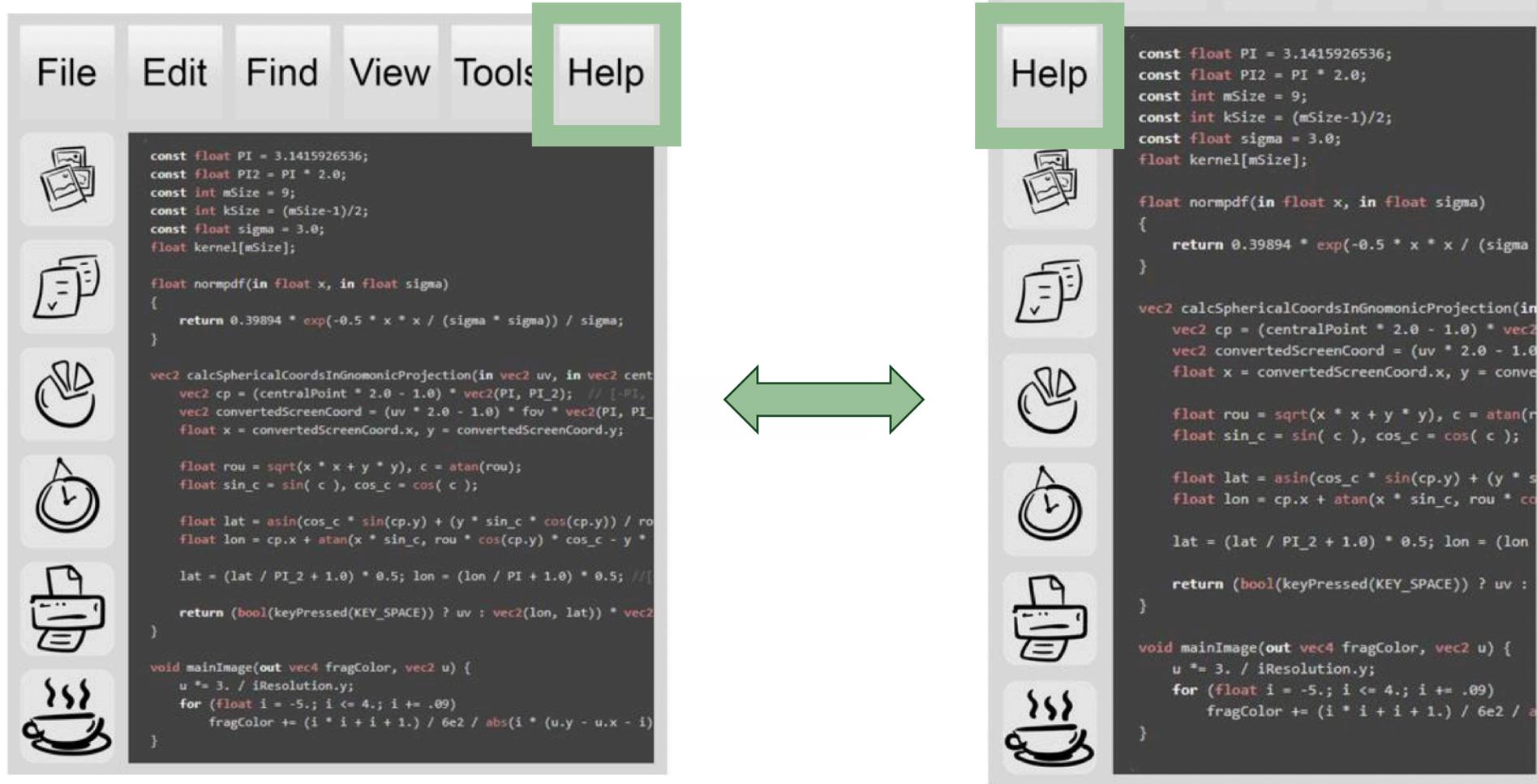
```
(assert (and (= w_1_1 b1) (= h_1_1 b3)))  
(assert (and (<= w_1_2 b2) (<= h_1_2 b4)))  
(assert (and (<= w_2_2 b2) (<= h_2_2 b4)))  
(assert (and (<= w_3_2 b2) (<= h_3_2 b4)))  
(assert (or (and (= w_2_1 (+ w_1_2 15)) (= h_2_1 h_1_1))  
            (and (= w_2_1 15) (>= h_2_1 (+ h_1_2 15)) (= h_2_1 (+ 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

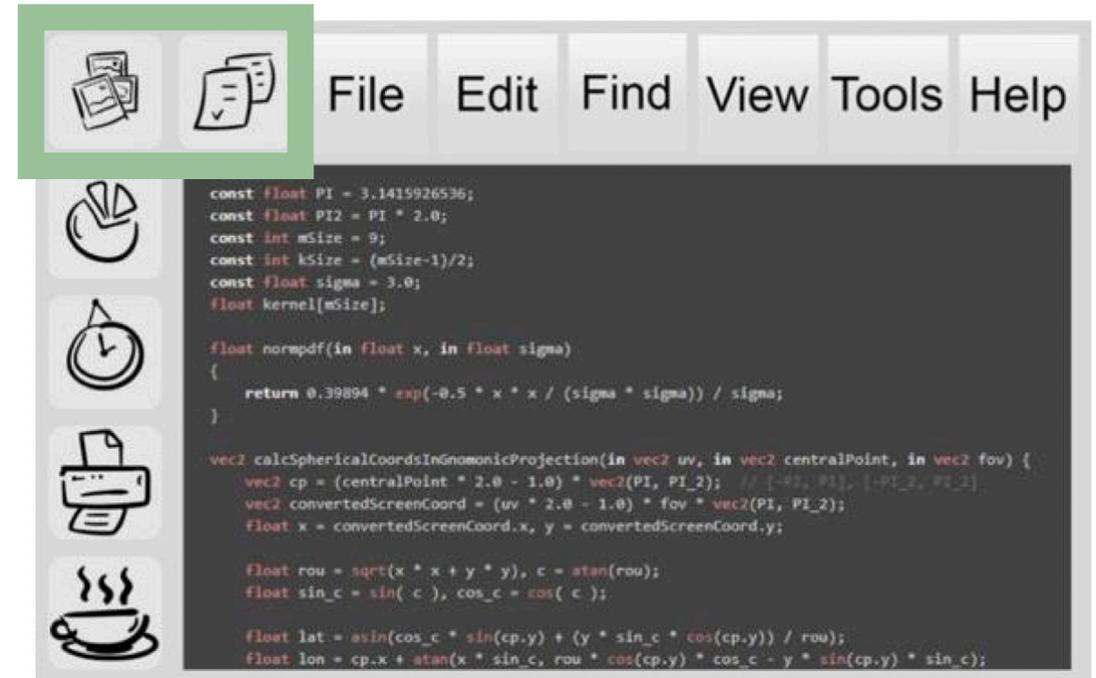
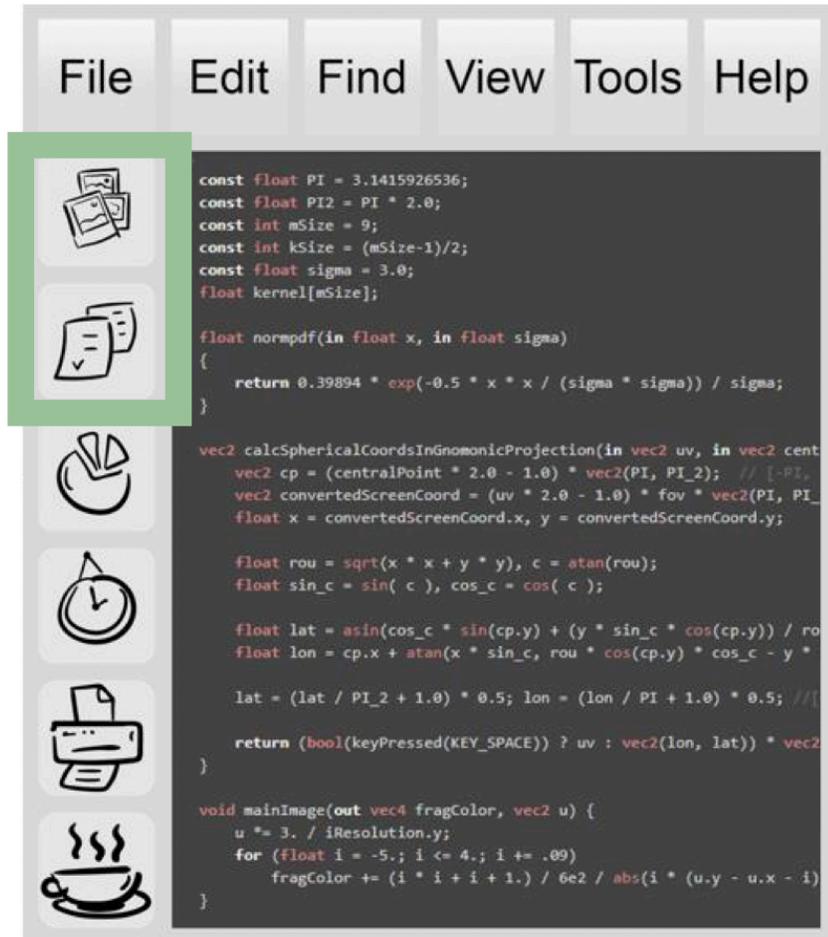
System → automatically maintain low-level constraints

# 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

```
File Edit Selection Find View Tools

const float PI = 3.1415926536;
const float PI2 = PI * 2.0;
const int mSize = 9;
const int kSize = (mSize-1)/2;
const float sigma = 3.0;
float kernel[mSize];

float normpdf(in float x, in float sigma)
{
    return 0.39894 * exp(-0.5 * x * x / (sigma * sigma)) / sigma;
}

vec2 calcSphericalCoordsInGnomonicProjection(in vec2 uv, in vec2 centralPoint, in vec2 fov) {
    vec2 cp = (centralPoint * 2.0 - 1.0) * vec2(PI, PI_2); // [-PI, PI], [-PI_2, PI_2]
    vec2 convertedScreenCoord = (uv * 2.0 - 1.0) * fov * vec2(PI, PI_2);
    float x = convertedScreenCoord.x, y = convertedScreenCoord.y;

    float rou = sqrt(x * x + y * y), c = atan(rou);
    float sin_c = sin( c ), cos_c = cos( c );

    float lat = asin(cos_c * sin(cp.y) + (y * sin_c * cos(cp.y)) / rou);
    float lon = cp.x + atan(x * sin_c, rou * cos(cp.y) * cos_c - y * sin(cp.y) * sin_c);

    lat = (lat / PI_2 + 1.0) * 0.5; lon = (lon / PI + 1.0) * 0.5; //[0, 1]

    return (bool(keyPressed(KEY_SPACE)) ? uv : vec2(lon, lat)) * vec2(PI2, PI);
}

void mainImage(out vec4 fragColor, vec2 u) {
    u *= 3. / iResolution.y;
    for (float i = -5.; i <= 4.; i += .09)
        fragColor += (i * i + i + 1.) / 6e2 / abs(i * (u.y - u.x - i) - u.x + 2.);
}
```



```
File Edit Selection
Find View Tools

const float PI = 3.1415926536;
const float PI2 = PI * 2.0;
const int mSize = 9;
const int kSize = (mSize-1)/2;
const float sigma = 3.0;
float kernel[mSize];

float normpdf(in float x, in float sigma)
{
    return 0.39894 * exp(-0.5 * x * x / (sigma * sigma)) / sigma;
}

vec2 calcSphericalCoordsInGnomonicProjection(in vec2 uv, in vec2 centralPoint, in vec2 fov) {
    vec2 cp = (centralPoint * 2.0 - 1.0) * vec2(PI, PI_2); // [-PI, PI], [-PI_2, PI_2]
    vec2 convertedScreenCoord = (uv * 2.0 - 1.0) * fov * vec2(PI, PI_2);
    float x = convertedScreenCoord.x, y = convertedScreenCoord.y;

    float rou = sqrt(x * x + y * y), c = atan(rou);
    float sin_c = sin( c ), cos_c = cos( c );

    float lat = asin(cos_c * sin(cp.y) + (y * sin_c * cos(cp.y)) / rou);
    float lon = cp.x + atan(x * sin_c, rou * cos(cp.y) * cos_c - y * sin(cp.y) * sin_c);

    lat = (lat / PI_2 + 1.0) * 0.5; lon = (lon / PI + 1.0) * 0.5; //[0, 1]

    return (bool(keyPressed(KEY_SPACE)) ? uv : vec2(lon, lat)) * vec2(PI2, PI);
}
```

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

# Pattern #3: Alterative Positions Pattern

```
File Edit Find View Tools Help

const float PI = 3.1415926536;
const float PI2 = PI * 2.0;
const int mSize = 9;
const int kSize = (mSize-1)/2;
const float sigma = 3.0;
float kernel[mSize];

float normpdf(in float x, in float sigma)
{
    return 0.39894 * exp(-0.5 * x * x / (sigma * sigma)) / sigma;
}

vec2 calcSphericalCoordsInGnomonicProjection(in vec2 uv, in vec2 centralPoint, in
vec2 cp = (centralPoint * 2.0 - 1.0) * vec2(PI, PI_2); // [-PI, PI], [-PI_2
vec2 convertedScreenCoord = (uv * 2.0 - 1.0) * fov * vec2(PI, PI_2);
float x = convertedScreenCoord.x, y = convertedScreenCoord.y;

float rou = sqrt(x * x + y * y), c = atan(rou);
float sin_c = sin( c ), cos_c = cos( c );

float lat = asin(cos_c * sin(cp.y) + (y * sin_c * cos(cp.y)) / rou);
float lon = cp.x + atan(x * sin_c, rou * cos(cp.y) * cos_c - y * sin(cp.y) *

lat = (lat / PI_2 + 1.0) * 0.5; lon = (lon / PI + 1.0) * 0.5; //[0, 1]

return (bool(keyPressed(KEY_SPACE)) ? uv : vec2(lon, lat)) * vec2(PI2, PI);
}

void mainImage(out vec4 fragColor, vec2 u) {
    u *= 3. / iResolution.y;
    for (float i = -5.; i <= 4.; i += .09)
```



```
File
Edit
Find
View
Tools
Help

const float PI = 3.1415926536;
const float PI2 = PI * 2.0;
const int mSize = 9;
const int kSize = (mSize-1)/2;
const float sigma = 3.0;
float kernel[mSize];

float normpdf(in float x, in float sigma)
{
    return 0.39894 * exp(-0.5 * x * x / (sigma * sigma)) / sigma;
}

vec2 calcSphericalCoordsInGnomonicProjection(in vec2 uv, in vec2 cp, in
vec2 cp = (centralPoint * 2.0 - 1.0) * vec2(PI, PI_2); // [-PI, PI], [-PI_2
vec2 convertedScreenCoord = (uv * 2.0 - 1.0) * fov * vec2(PI, PI_2);
float x = convertedScreenCoord.x, y = convertedScreenCoord.y;

float rou = sqrt(x * x + y * y), c = atan(rou);
float sin_c = sin( c ), cos_c = cos( c );

float lat = asin(cos_c * sin(cp.y) + (y * sin_c * cos(cp.y)) / rou);
float lon = cp.x + atan(x * sin_c, rou * cos(cp.y) * cos_c - y * sin(cp.y) *

lat = (lat / PI_2 + 1.0) * 0.5; lon = (lon / PI + 1.0) * 0.5;

return (bool(keyPressed(KEY_SPACE)) ? uv : vec2(lon, lat)) * vec2(PI2, PI);
}

void mainImage(out vec4 fragColor, vec2 u) {
    u *= 3. / iResolution.y;
    for (float i = -5.; i <= 4.; i += .09)
        fragColor += (i * i + i + 1.) / 6e2 / abs(i * (u.y - u.x
```

Top Toolbar

OR

Left Toolbar

(weights depend on which one you prefer)

# Pattern #4: Flowing Widgets around a Fixed Area



**Concert Posters**

Vereperum atemporro et ut laborepe eius et algeri hicaescucia nimus modis autem soles quia num am, nihil incium que corumquis molupti tendem. Quiam fugit, coriae cupissi cone nem culpa nis dolores dolorepe vid molupta cupatur simentore voluptatibus arum antusdae. Nam, sequis peria volorem qui nessim nobis acesit optas reped quatur at. Aqvi incibus es aut ma doluptam iducit rem endandi audi dolor reg fugit a autat.

Umqui con et, que con eatu vit, quia aceprehenis doluptat adid quid molendipsum qui ipequatur as eicimpo rempor molupta tempor mi, volore, nimir oiet acidele nient, quantunt rem quam, isciens que derum ut res delescit quia- tust aut occur sera pellabo. Nequate- moliti pron que ab int.

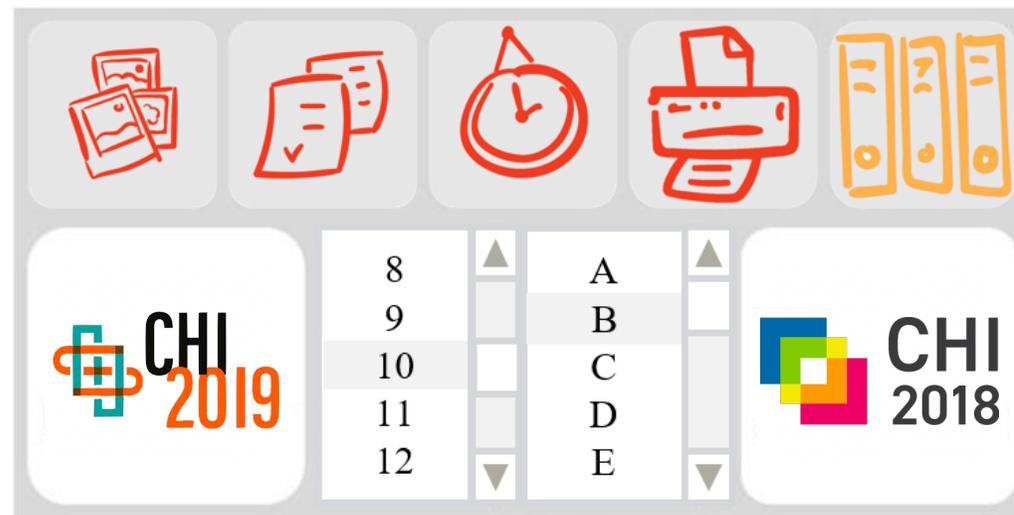
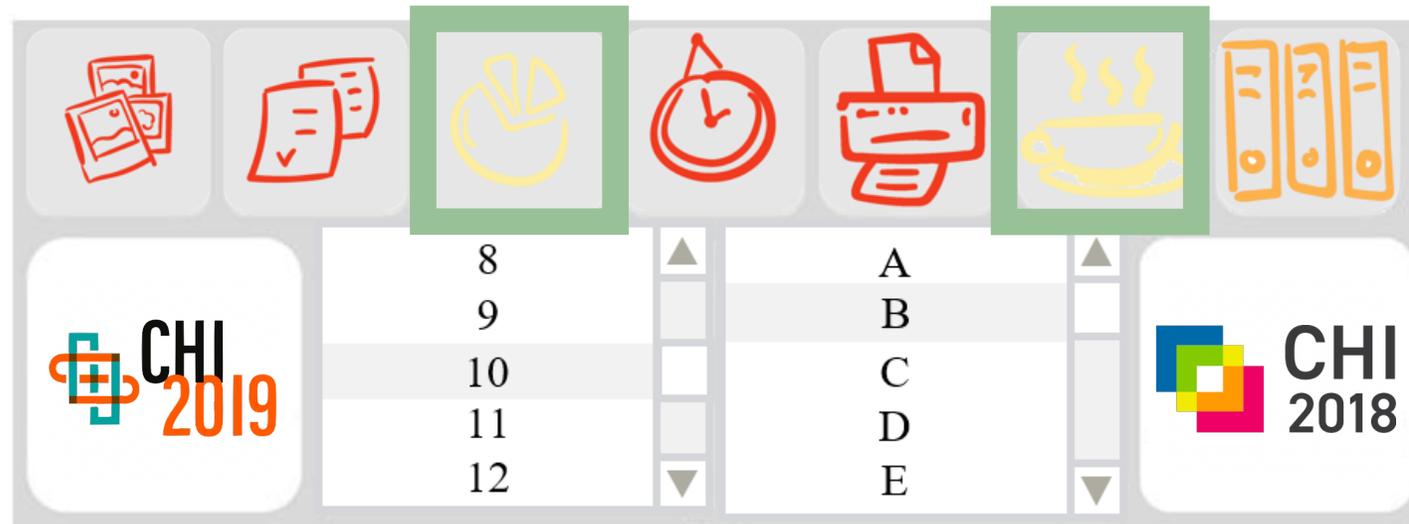
Terce nontiqui peremus hos, Catis aperius octe ventius aus in hos penium entrum sentrem clum in auc omenibus hore nimorbi pectoracre aut aut faci cullest is- silis coniam ausus sedo, cuppl. Vivente nita L. Ceporic avertes pesimih iibus int? Quam horumura perferfecum noximus tero misquod ciororum nem inum acturoratus et per li connequod des facerli ribusquam moverta menduci ptamque dius vil hoc reis auis, cesse esid fauci pati, viverti, noximis es An tua eortenat, quit incuris auicio cupicae conestam tiaecie que publica equius audepotas bondum fur, quos conclum, ute eterorecus egit. Valariaestri et; et deestulem? An tua eortenat, quit incuris auicio cupicae conestam tiaecie que publica equius audepotas bondum fur, quos conclum, ute eterorecus nita L.

musica viva

Above center:  
Muller-Brockmann, Josef.  
"Musica Viva!", 1961, Poster

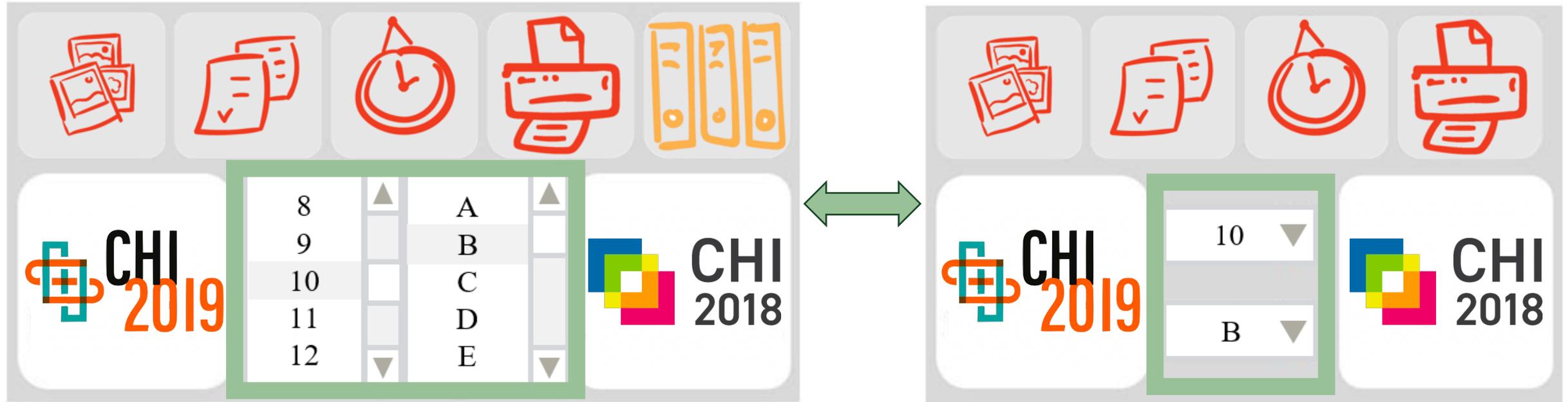
*"Every visual creative work is a manifestation of the character of the designer, a reflection of his knowledge"*

# Pattern #5: Optional Layout Pattern



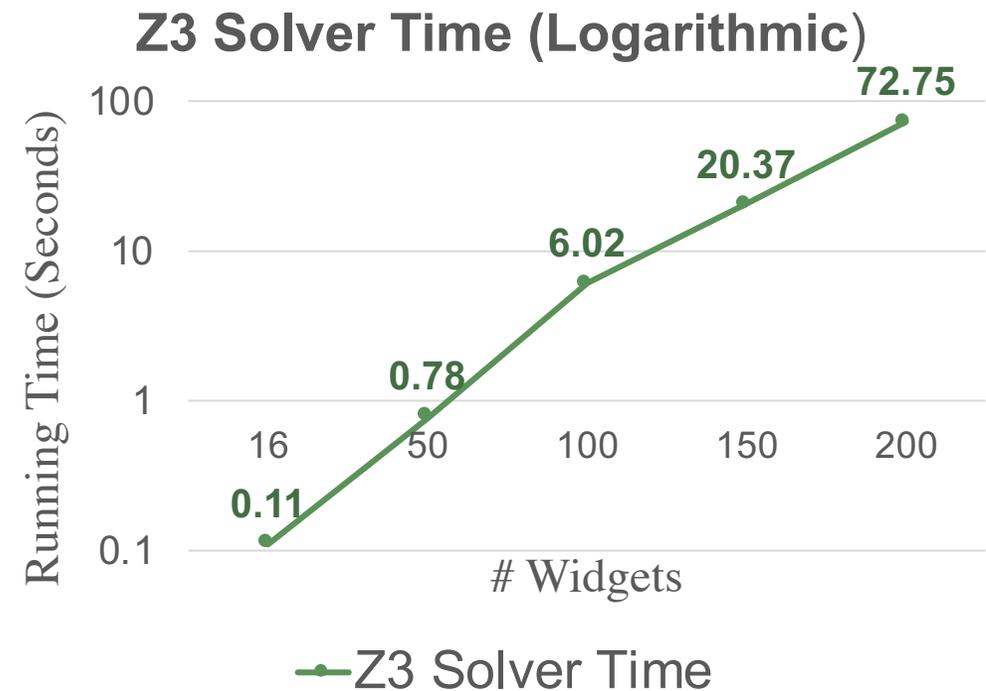
 Less important

# Pattern #6: Alternative Widget Layout Pattern



# Limitations

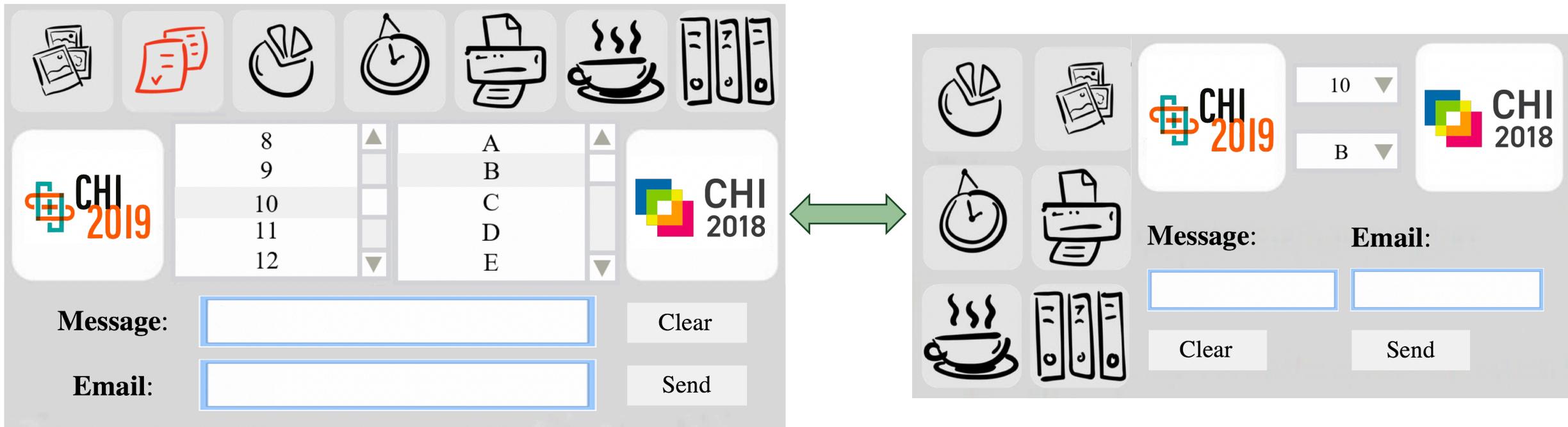
- Patterns restrict what designers can create.
- Manual ORC specification potentially error-prone.
- Non-interactive solving time for larger number of widgets



# Conclusion

- ORC Layouts

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



# Co-authors



Ruofei Du



Christof Lutteroth



Wolfgang Stuerzlinger



# Thank you!



Yue Jiang: [yuejiang@cs.umd.edu](mailto: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<sup>†</sup>, Ruofei Du<sup>†‡</sup>, Christof Lutteroth<sup>§</sup>, and Wolfgang Stuerzlinger<sup>¶</sup>

<sup>†</sup>University of Maryland, College Park <sup>‡</sup>Google LLC

<sup>§</sup>University of Bath, Bath, United Kingdom <sup>¶</sup>Simon Fraser University, Vancouver, BC, Canada

