Fast Direct Manipulation Programming with Patch-Reconciliation Correspondence

EPIC Data Lab Retreat • April 16, 2025 • Berkeley, CA

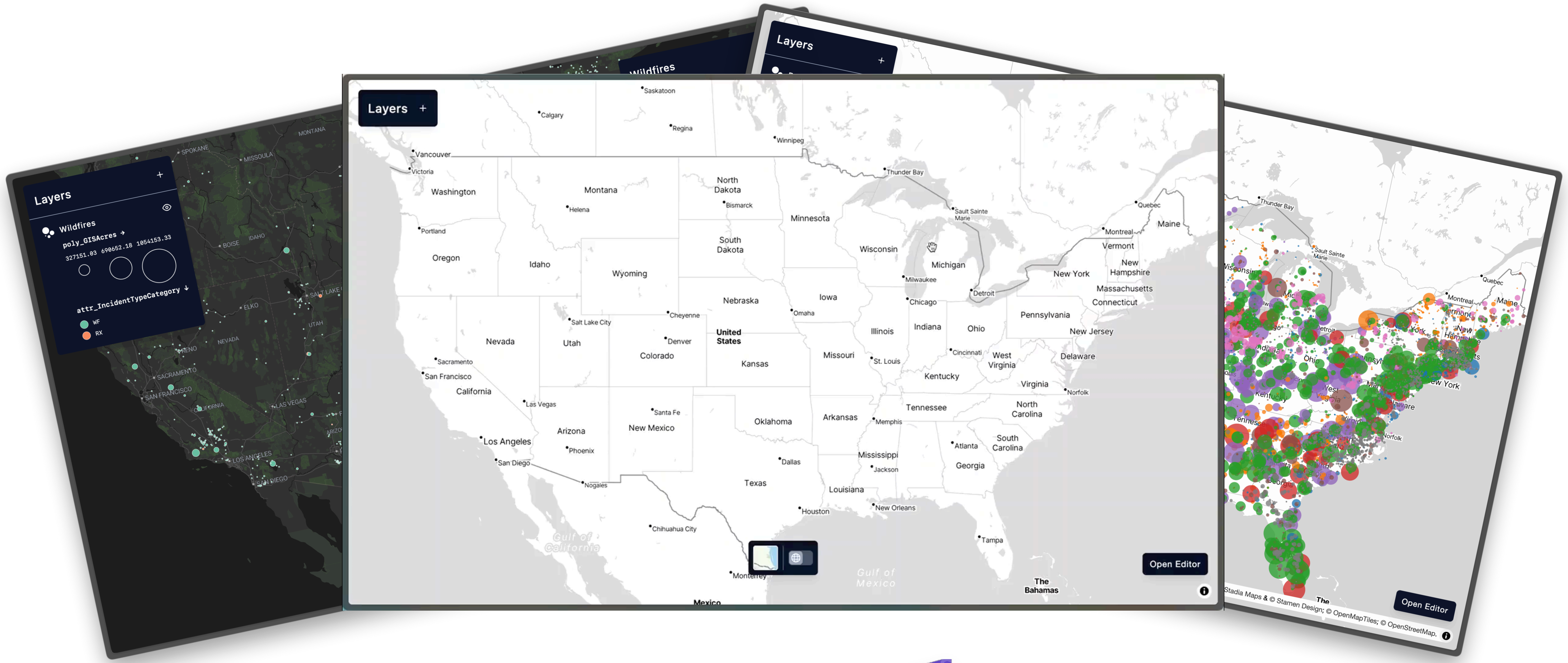
Parker Ziegler
peziegler@cs.berkeley.edu

Justin Lubin
justinlubin@berkeley.edu

Sarah E. Chasins
schasins@cs.berkeley.edu

cartokit

A Direct Manipulation Programming System for Interactive Maps



cartokit

A Direct Manipulation Programming System for Interactive Maps

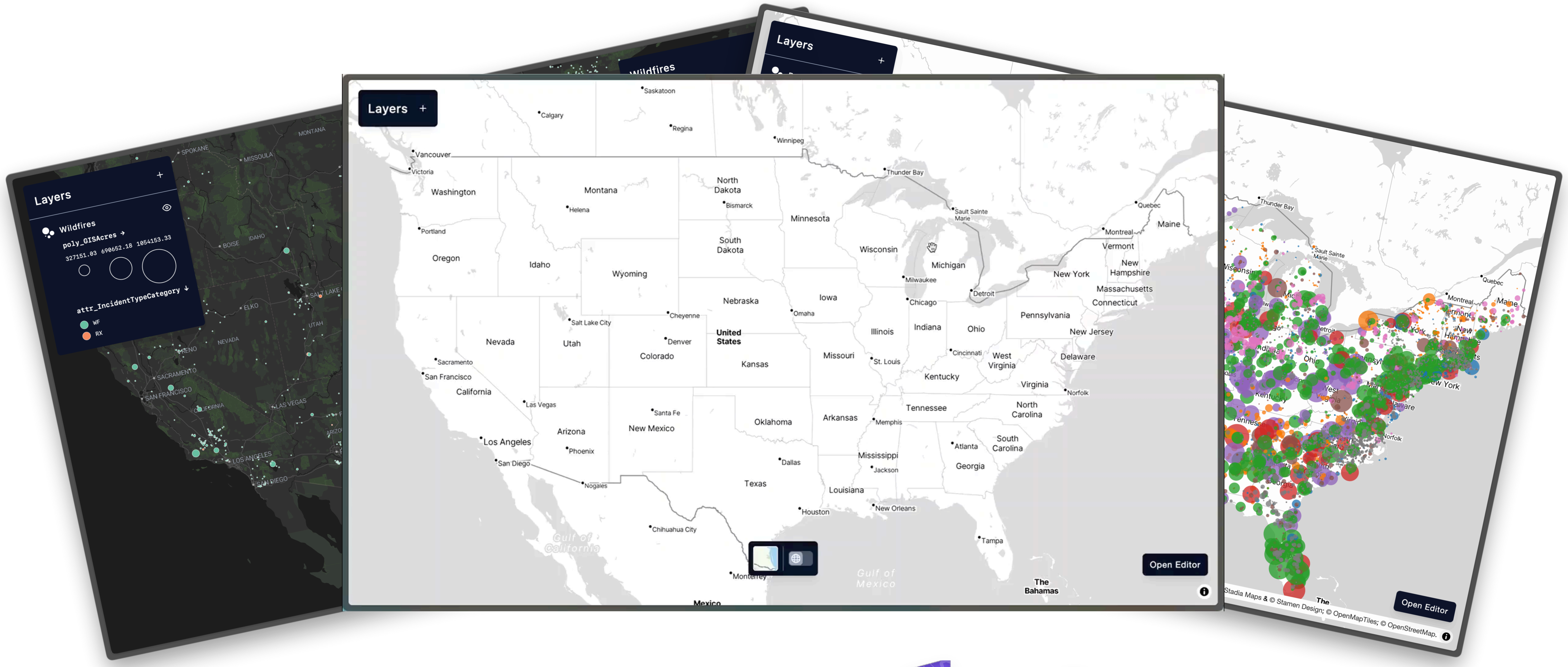
Used by journalists at

NICAR25 Minneapolis

FT
FINANCIAL TIMES

MUCKROCK

Grist



cartokit

A Direct Manipulation Programming System for Interactive Maps

1k+

Monthly Sessions

Users from

20

Countries

Used by journalists at

NICAR25 Minneapolis

T

wp

FT FINANCIAL TIMES

MUCKROCK

Grist

Direct Manipulation Interfaces

Direct Manipulation Interfaces

> *“act on **displayed objects of interest** using physical, incremental, and reversible actions whose effects are **immediately visible on the screen**”*

Direct Manipulation Interfaces

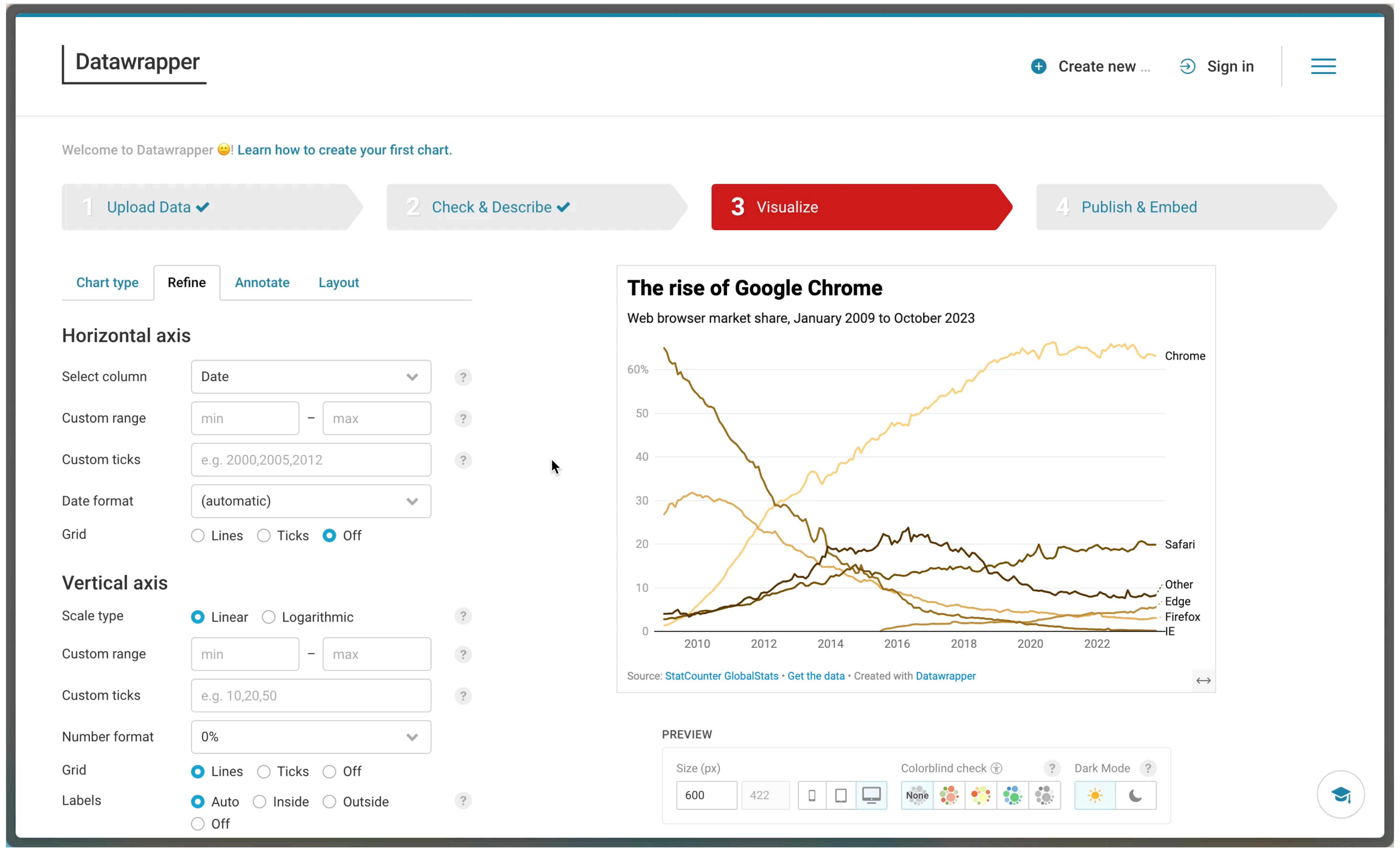
> *“act on **displayed objects of interest** using physical, incremental, and reversible actions whose effects are **immediately visible on the screen**”*

Data Visualization and Analysis

QGIS mapbox

tableau Datawrapper

Direct Manipulation Interfaces



Datawrapper

Welcome to Datawrapper 😊! Learn how to create your first chart.

1 Upload Data ✓ · 2 Check & Describe ✓ · 3 Visualize · 4 Publish & Embed

Chart type · Refine · Annotate · Layout

Horizontal axis

Select column: Date

Custom range: min – max

Custom ticks: e.g. 2000,2005,2012

Date format: (automatic)

Grid: ○ Lines ○ Ticks ● Off

Vertical axis

Scale type: ● Linear ○ Logarithmic

Custom range: min – max

Custom ticks: e.g. 10,20,50

Number format: 0%

Grid: ● Lines ○ Ticks ○ Off

Labels: ● Auto ○ Inside ○ Outside ○ Off

The rise of Google Chrome

Web browser market share, January 2009 to October 2023

Source: StatCounter GlobalStats • Get the data • Created with Datawrapper

PREVIEW

Size (px): 600 422

Colorblind check: None

Dark Mode

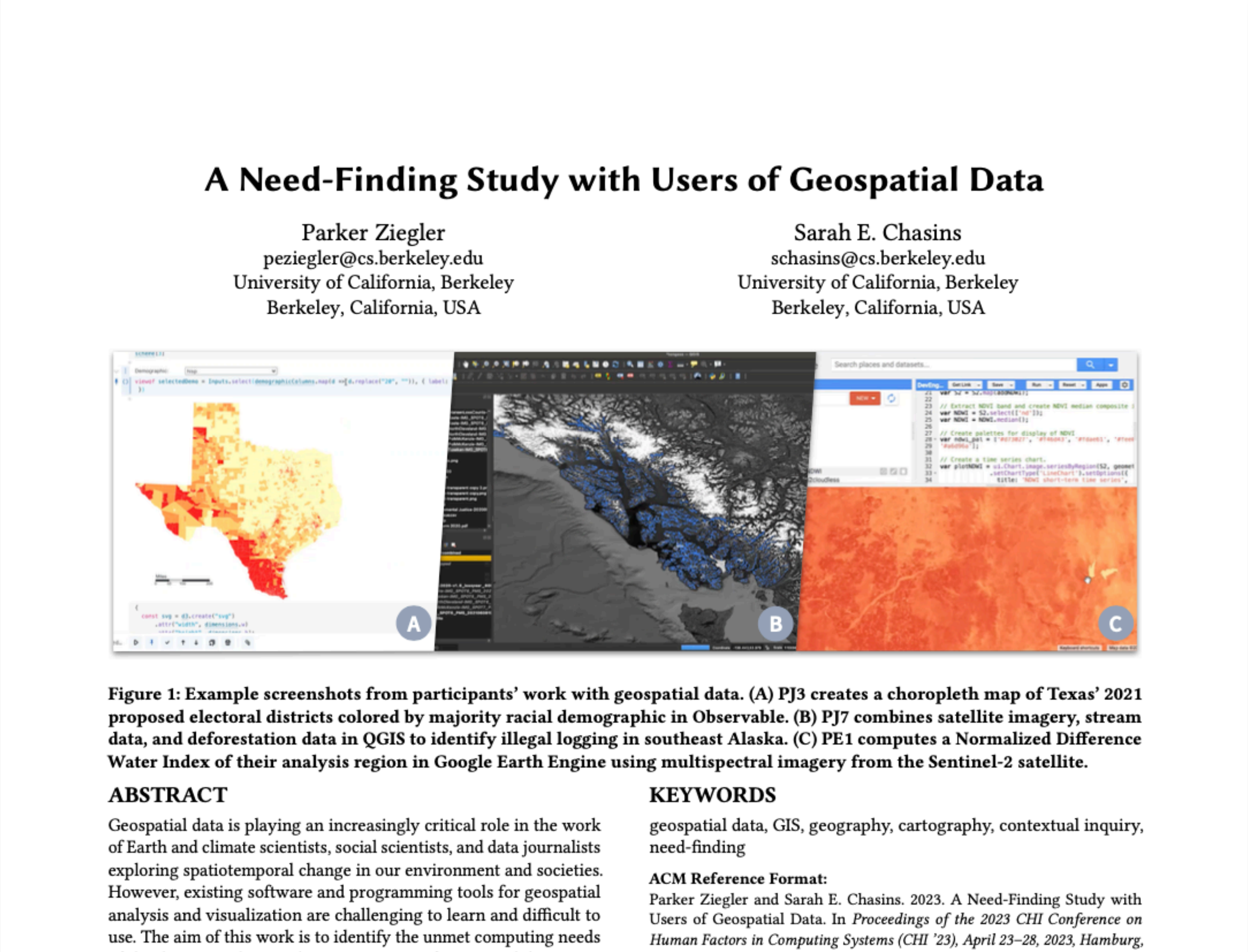
Datawrapper

Direct Manipulation Interfaces

These systems play an instrumental role in today's newsrooms.

Observational Research (CHI '23)

Direct Collaboration (Grist Data Desk)



A Need-Finding Study with Users of Geospatial Data

Parker Ziegler
peziegler@cs.berkeley.edu
University of California, Berkeley
Berkeley, California, USA

Sarah E. Chasins
schasins@cs.berkeley.edu
University of California, Berkeley
Berkeley, California, USA

Figure 1: Example screenshots from participants' work with geospatial data. (A) PJ3 creates a choropleth map of Texas' 2021 proposed electoral districts colored by majority racial demographic in Observable. (B) PJ7 combines satellite imagery, stream data, and deforestation data in QGIS to identify illegal logging in southeast Alaska. (C) PE1 computes a Normalized Difference Water Index of their analysis region in Google Earth Engine using multispectral imagery from the Sentinel-2 satellite.

ABSTRACT

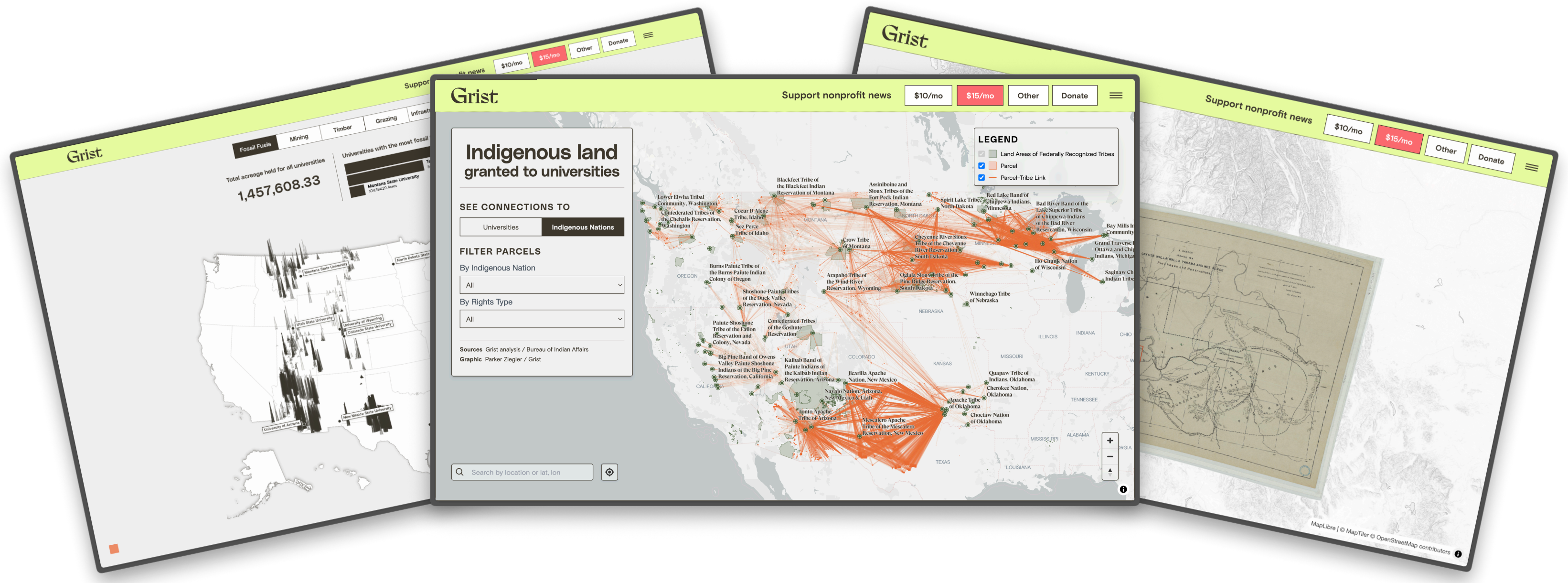
Geospatial data is playing an increasingly critical role in the work of Earth and climate scientists, social scientists, and data journalists exploring spatiotemporal change in our environment and societies. However, existing software and programming tools for geospatial analysis and visualization are challenging to learn and difficult to use. The aim of this work is to identify the unmet computing needs

KEYWORDS

geospatial data, GIS, geography, cartography, contextual inquiry, need-finding

ACM Reference Format:
Parker Ziegler and Sarah E. Chasins. 2023. A Need-Finding Study with Users of Geospatial Data. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg,*

Grist



Direct Manipulation Interfaces

These systems play an instrumental role in today’s newsrooms.

✅ Instantaneous feedback

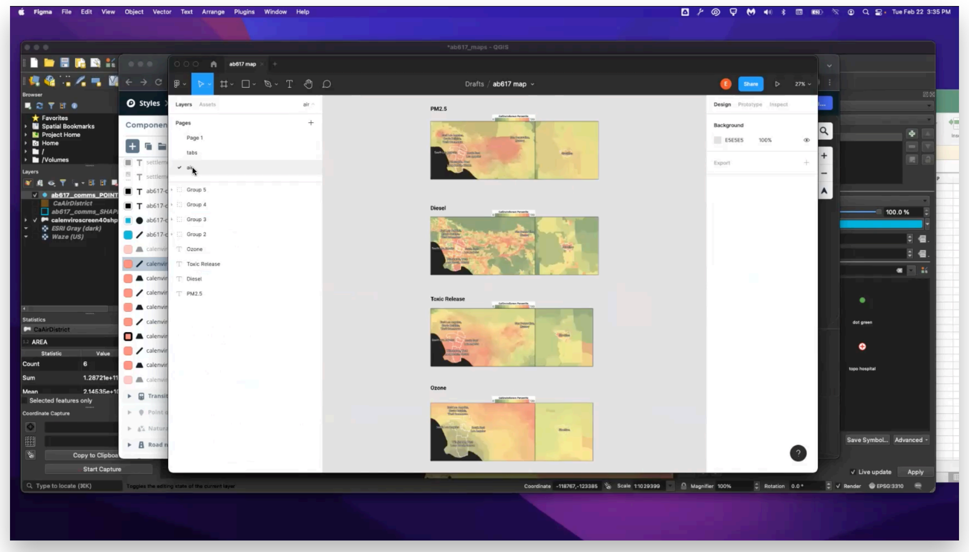
✅ Easy to explore variants

✅ Smart defaults

Direct Manipulation Interfaces

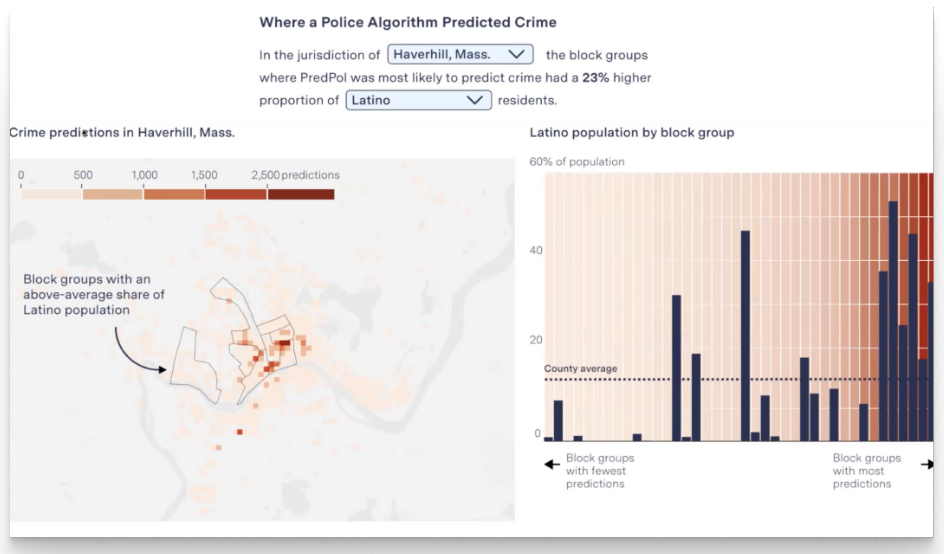
These systems play an instrumental role in today's newsrooms.

- ✅ Instantaneous feedback
- ✅ Easy to explore variants
- ✅ Smart defaults



useful for prototyping **"before I code anything"**

"I have several different versions...it's predominantly **thinking through what is the user experience** and what kind of information we want the reader to be focused on"



Where a Police Algorithm Predicted Crime

In the jurisdiction of Haverhill, Mass. the block groups where PredPol was most likely to predict crime had a **23%** higher proportion of Latino residents.

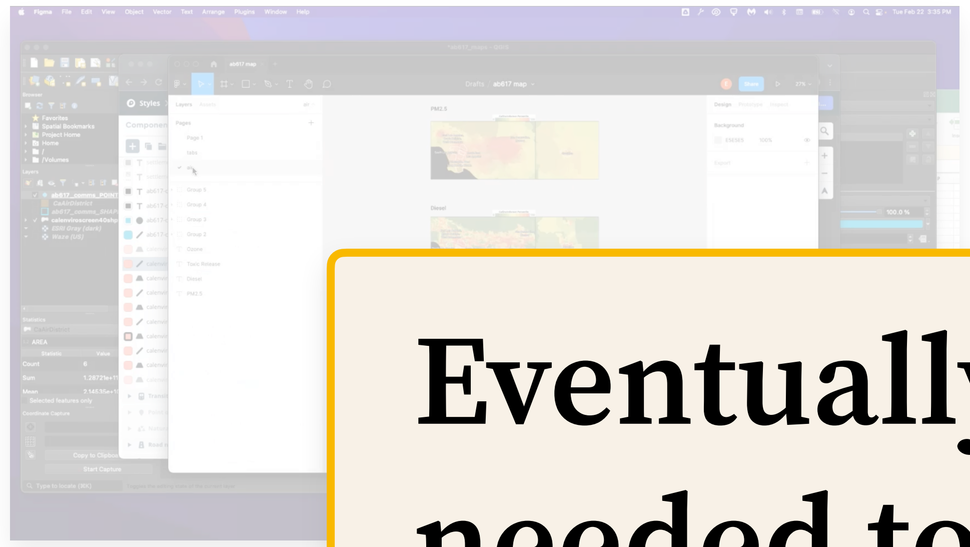
Direct Manipulation Interfaces

These systems play an instrumental role in today’s newsrooms.

✅ Instantaneous feedback

✅ Easy to explore variants

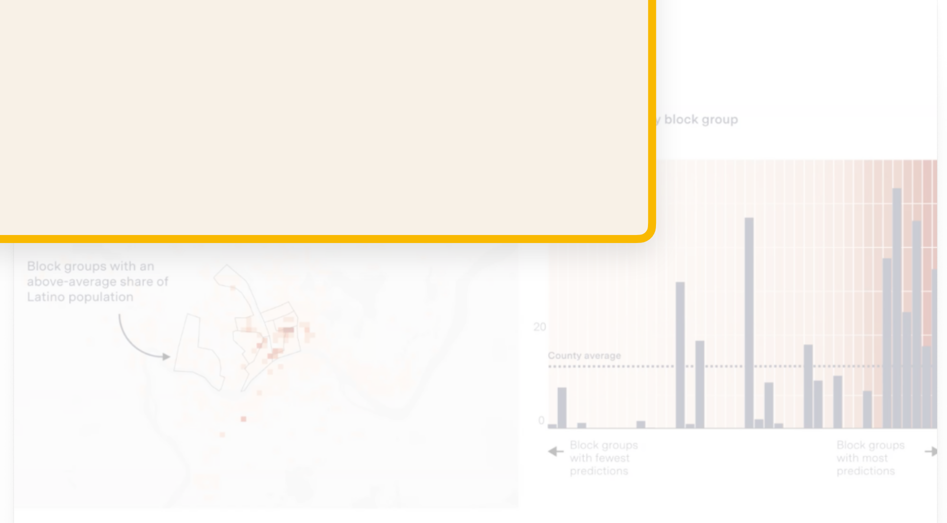
✅ Smart defaults



useful for prototyping
“before I code

Eventually, they needed to author programs from scratch.

“I have sev
versions…i
thinking th
user experience and what kind of information we want the reader to be focused on”



Direct Manipulation Interfaces

These systems play an instrumental role in today’s newsrooms.

- ✅ Instantaneous feedback
- ✅ Easy to explore variants
- ✅ Smart defaults

- ❌ Difficult to customize
- ❌ Difficult to abstract
- ❌ Difficult to escape

Direct Manipulation Interfaces

These systems play an instrumental role in today’s newsrooms.

- ✅ Instantaneous feedback
- ✅ Ea ts
- ✅ Smart defaults

This is what programming is made for!

- ❌ Difficult to customize
- ❌ Difficult to abstract
- ❌ Difficult to escape

Direct Manipulation Interfaces

These systems play an instrumental role in today’s newsrooms.

- ✅ Instantaneous feedback
- ✅ Ea ts
- ✅ Smart defaults

This is what programming is made for!

- ✅ Extremely expressive
- ❌ Difficult to abstract
- ❌ Difficult to escape

Direct Manipulation Interfaces

These systems play an instrumental role in today's newsrooms.

- ✅ Instantaneous feedback
- ✅ Ea ts
- ✅ Smart defaults

This is what programming is made for!

- ✅ Extremely expressive
- ✅ Built for abstraction
- ❌ Difficult to escape

Direct Manipulation Interfaces

These systems play an instrumental role in today’s newsrooms.

- ✅ Instantaneous feedback
- ✅ Ea ts
- ✅ Smart defaults

This is what programming is made for!

- ✅ Extremely expressive
- ✅ Built for abstraction
- ✅ Highly portable

Programming
Direct Manipulation ~~Interfaces~~

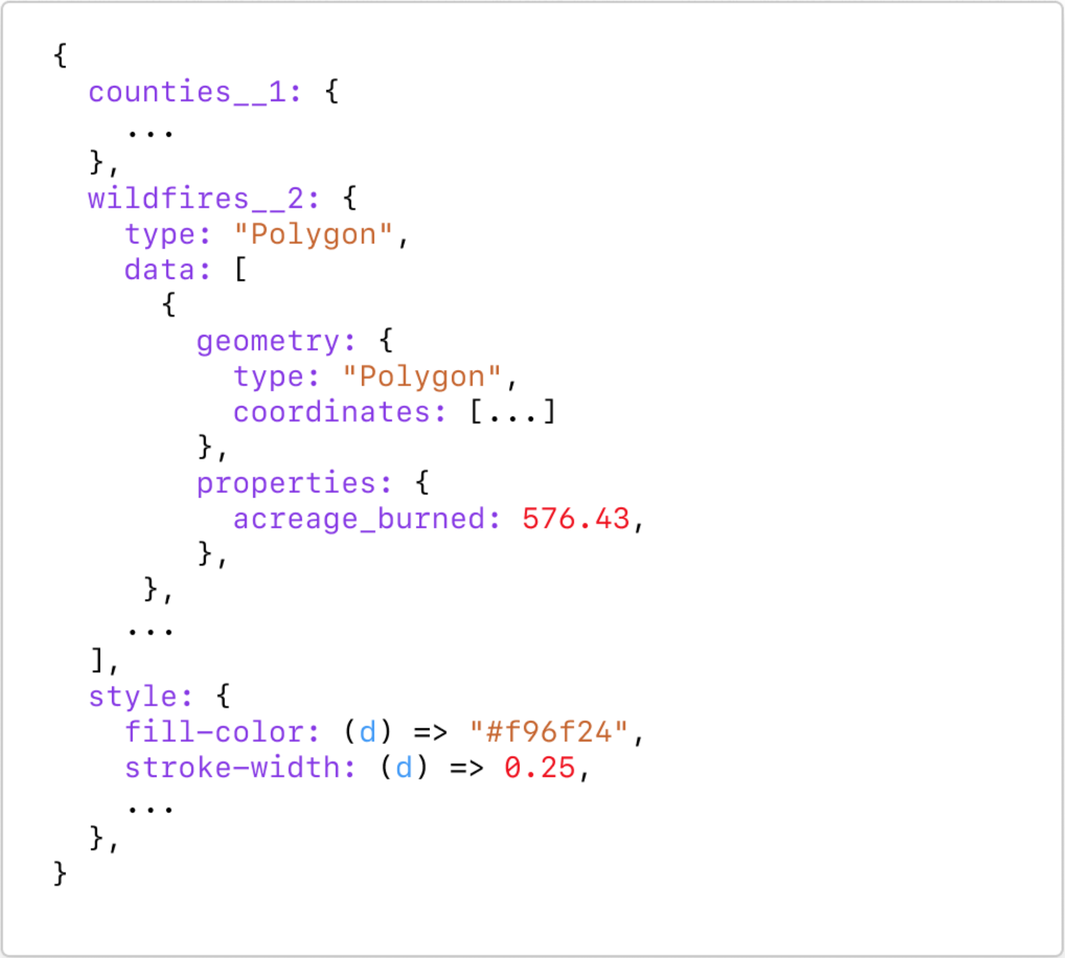
- ✅ Instantaneous feedback
- ✅ Easy to explore variants
- ✅ Smart defaults

- ✅ Extremely expressive
- ✅ Built for abstraction
- ✅ Highly portable

Direct Manipulation Programming

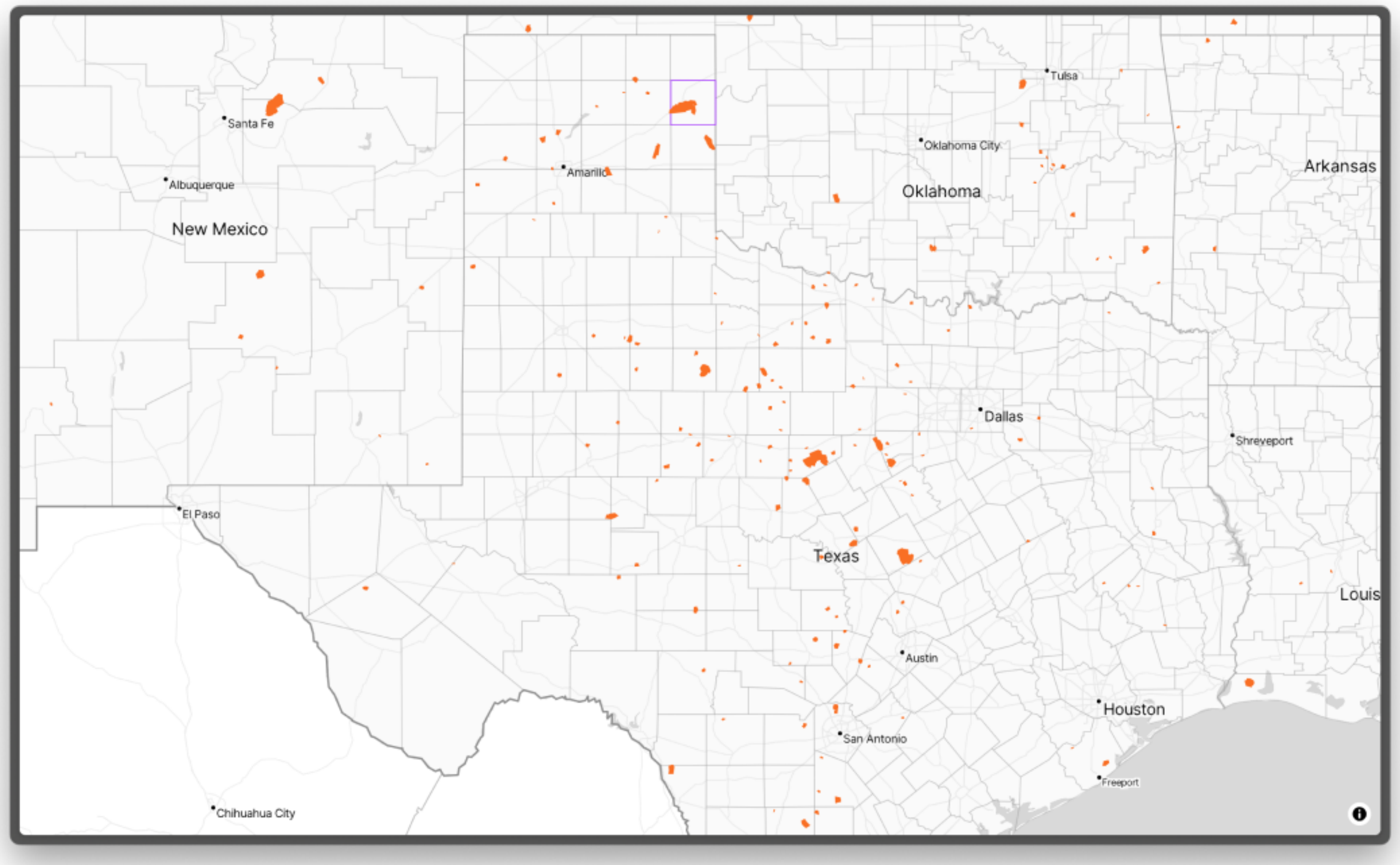
Direct Manipulation Programming

Program



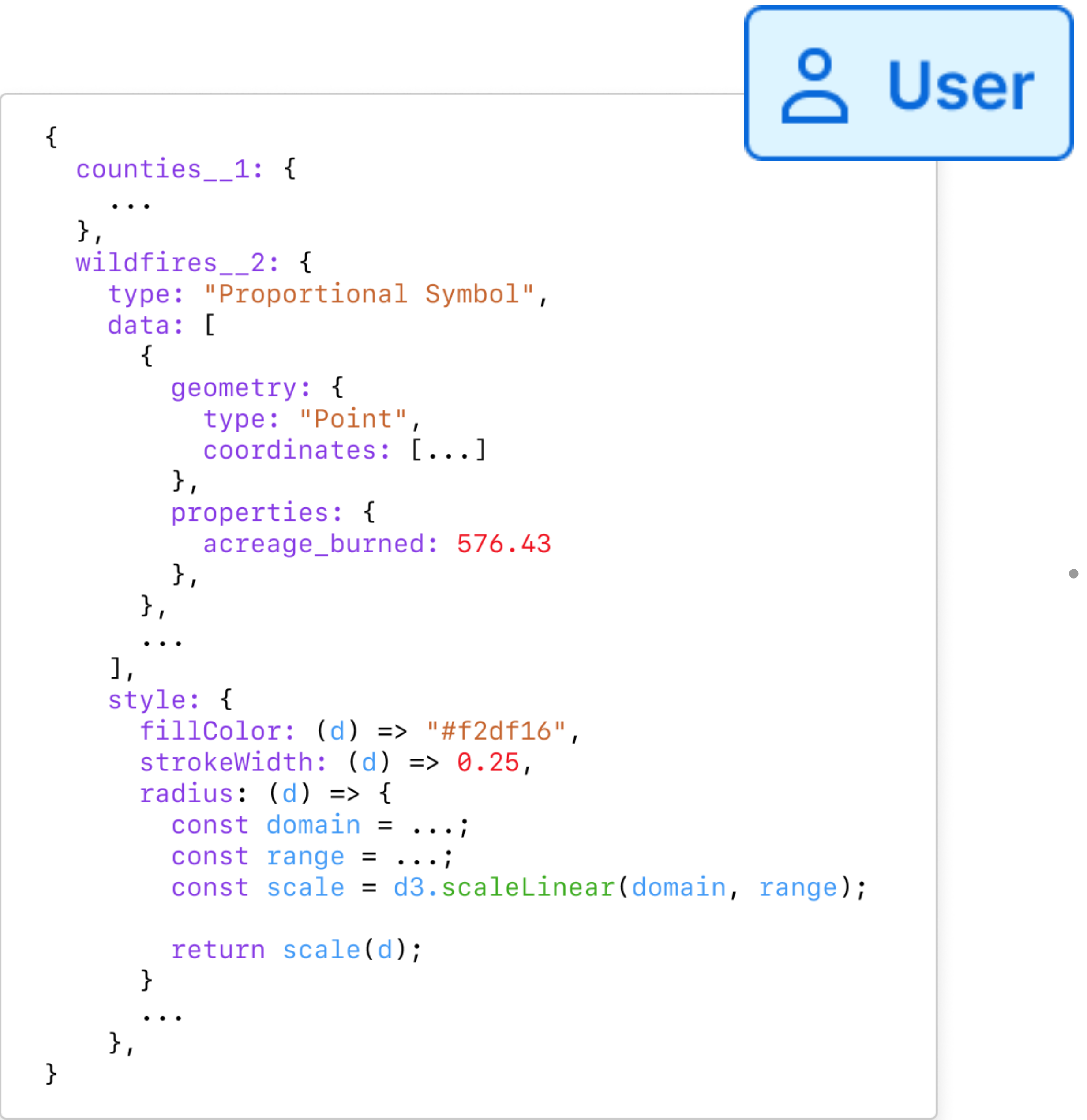
```
{
  counties__1: {
    ...
  },
  wildfires__2: {
    type: "Polygon",
    data: [
      {
        geometry: {
          type: "Polygon",
          coordinates: [...]
        },
        properties: {
          acreage_burned: 576.43,
        },
     },
    ...
  ],
  style: {
    fill-color: (d) => "#f96f24",
    stroke-width: (d) => 0.25,
    ...
  },
}
```

Output



Direct Manipulation Programming

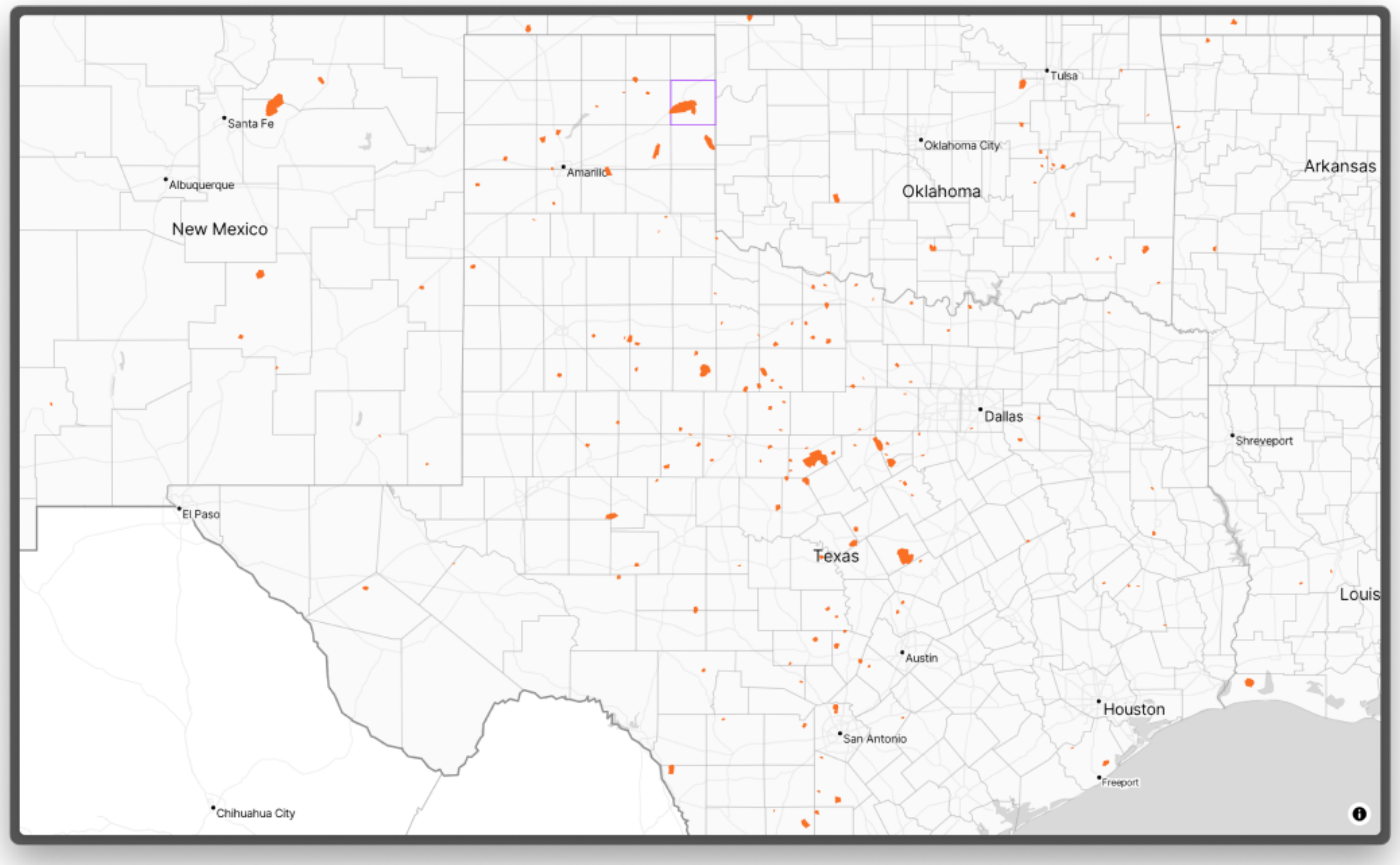
Program



```
{
  counties__1: {
    ...
  },
  wildfires__2: {
    type: "Proportional Symbol",
    data: [
      {
        geometry: {
          type: "Point",
          coordinates: [...]
        },
        properties: {
          acreage_burned: 576.43
        },
      },
      ...
    ],
    style: {
      fillColor: (d) => "#f2df16",
      strokeWidth: (d) => 0.25,
      radius: (d) => {
        const domain = ...;
        const range = ...;
        const scale = d3.scaleLinear(domain, range);

        return scale(d);
      }
      ...
    },
}
```

Sync

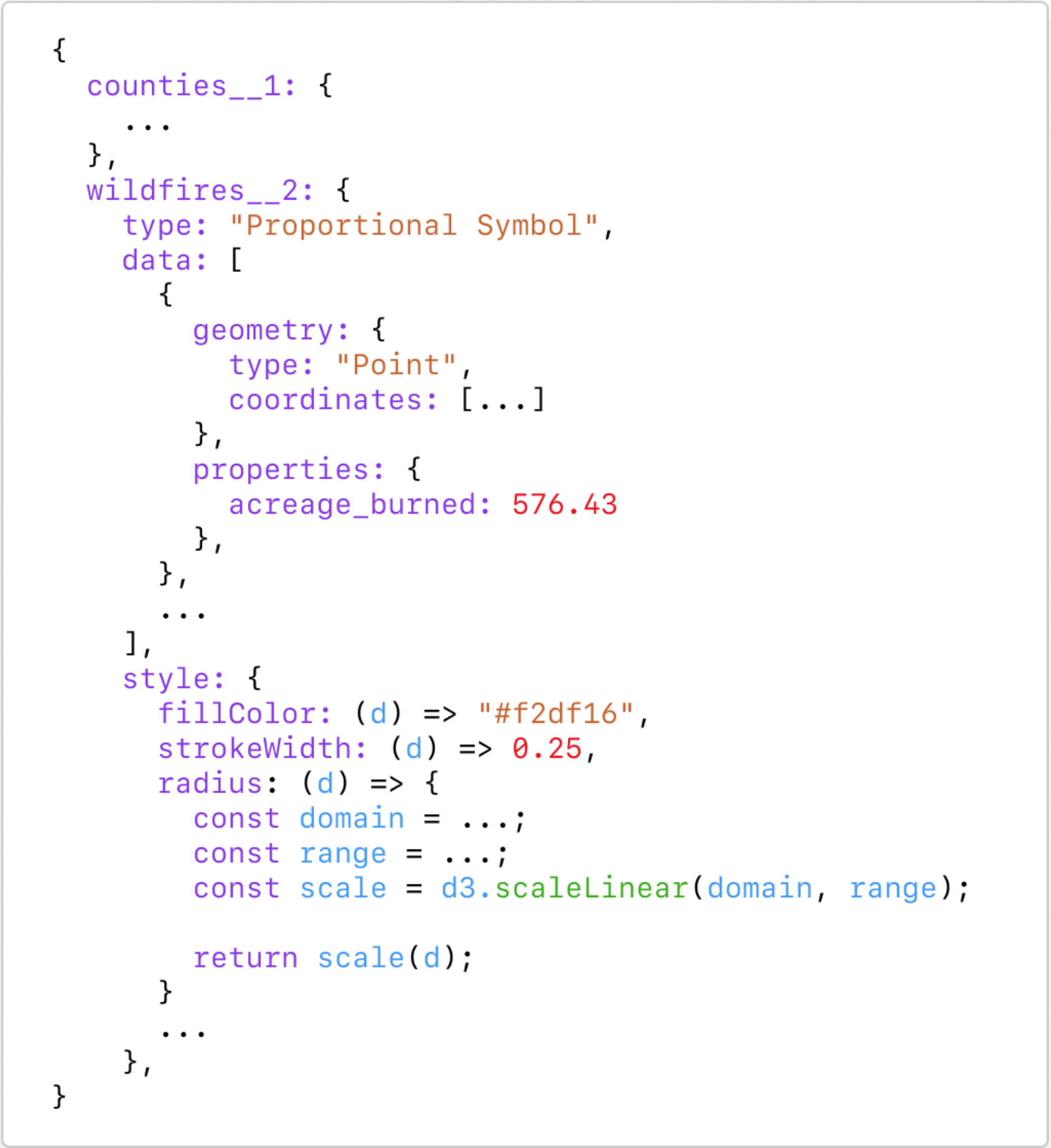
Output



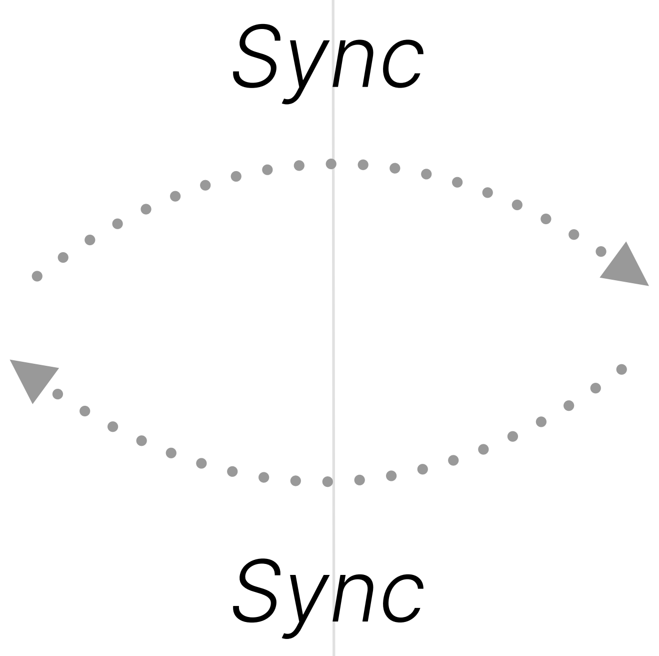
Direct Manipulation Programming

Program

Output

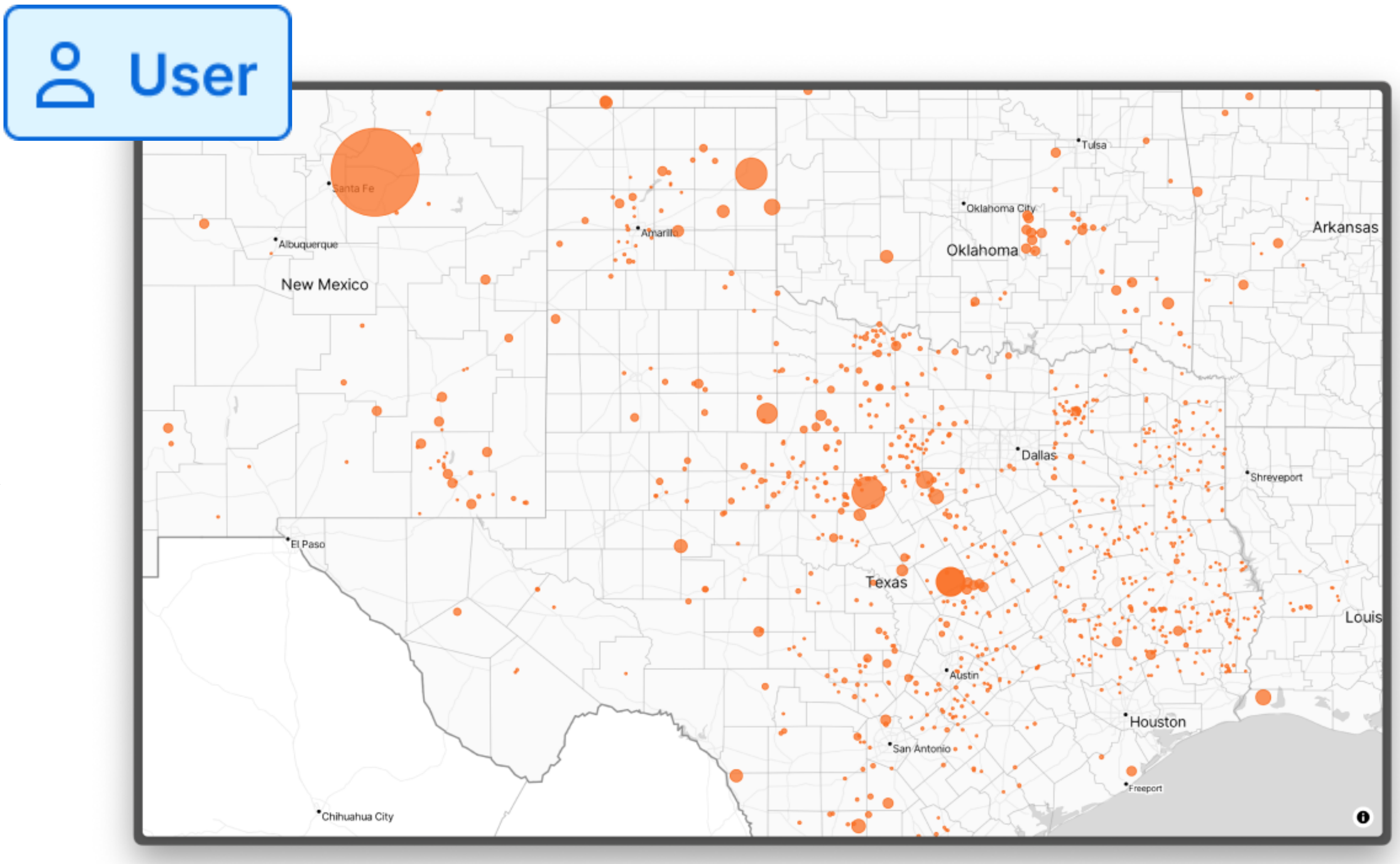


```
{
  counties__1: {
    ...
  },
  wildfires__2: {
    type: "Proportional Symbol",
    data: [
      {
        geometry: {
          type: "Point",
          coordinates: [...]
        },
        properties: {
          acreage_burned: 576.43
        },
      },
      ...
    ],
    style: {
      fillColor: (d) => "#f2df16",
      strokeWidth: (d) => 0.25,
      radius: (d) => {
        const domain = ...;
        const range = ...;
        const scale = d3.scaleLinear(domain, range);

        return scale(d);
      }
      ...
    },
}
```



Sync

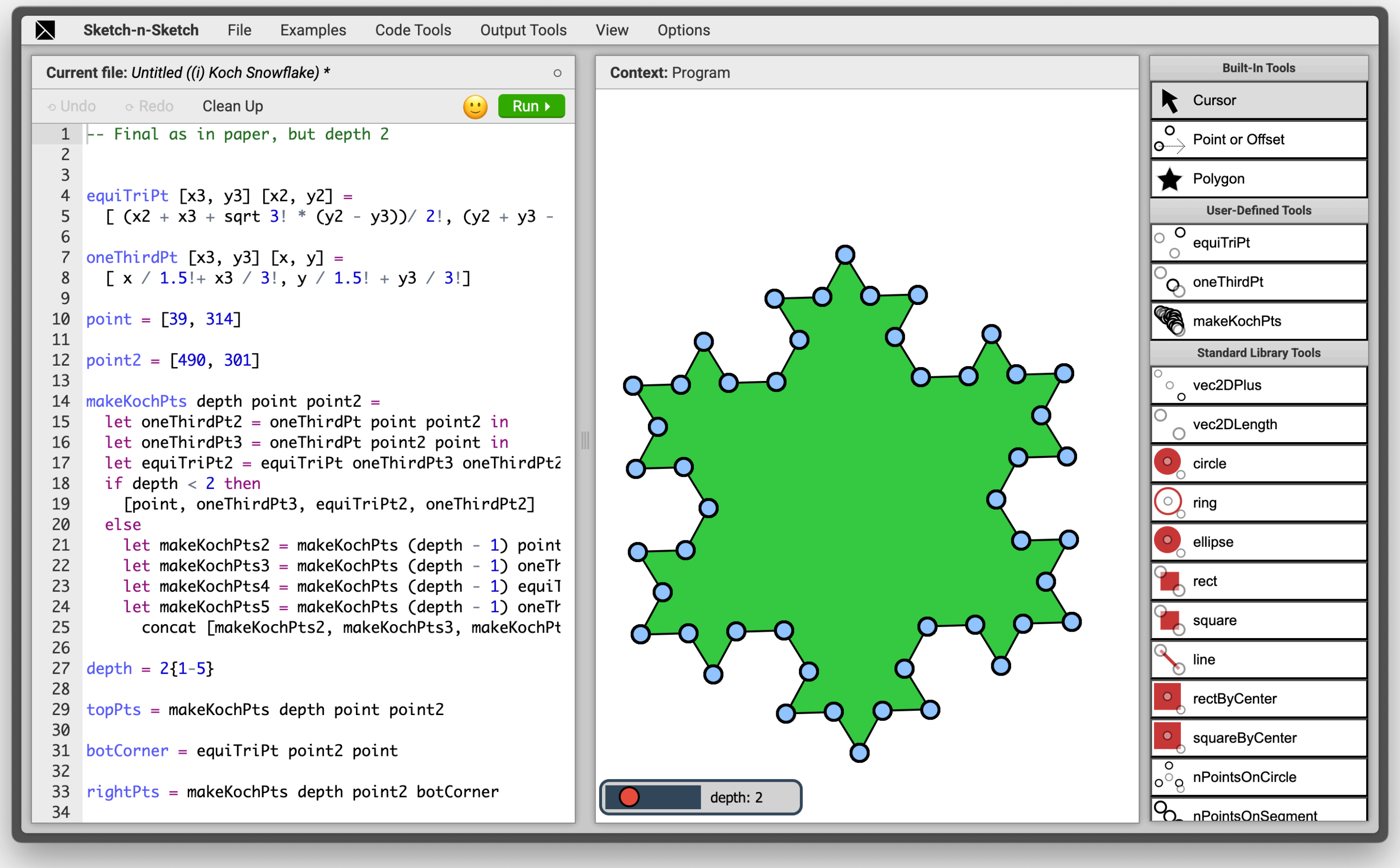
Sync



User

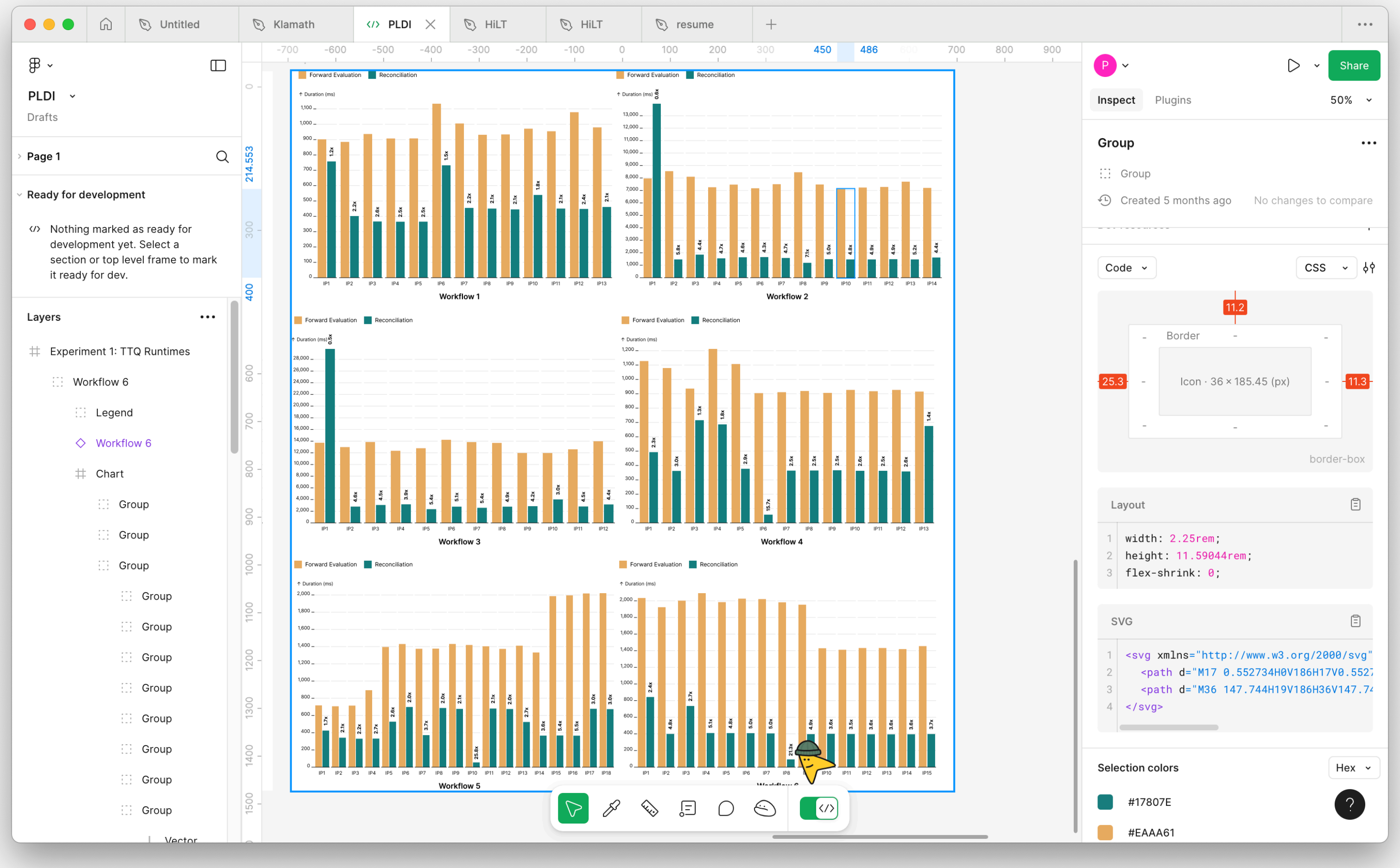
Direct Manipulation Programming

Sketch-n-Sketch (+livelits, BiOOP)



```
-- Final as in paper, but depth 2


equiTriPt [x3, y3] [x2, y2] =
  [ (x2 + x3 + sqrt 3! * (y2 - y3))/ 2!, (y2 + y3 -

oneThirdPt [x3, y3] [x, y] =
  [ x / 1.5!+ x3 / 3!, y / 1.5! + y3 / 3!]

point = [39, 314]

point2 = [490, 301]

makeKochPts depth point point2 =
  let oneThirdPt2 = oneThirdPt point point2 in
  let oneThirdPt3 = oneThirdPt point2 point in
  let equiTriPt2 = equiTriPt oneThirdPt3 oneThirdPt2
  if depth < 2 then
    [point, oneThirdPt3, equiTriPt2, oneThirdPt2]
  else
    let makeKochPts2 = makeKochPts (depth - 1) point
    let makeKochPts3 = makeKochPts (depth - 1) oneTh
    let makeKochPts4 = makeKochPts (depth - 1) equiT
    let makeKochPts5 = makeKochPts (depth - 1) oneTh
      concat [makeKochPts2, makeKochPts3, makeKochPt

depth = 2{1-5}

topPts = makeKochPts depth point point2

botCorner = equiTriPt point2 point

rightPts = makeKochPts depth point2 botCorner

```

Figma Dev Mode



Nothing marked as ready for development yet. Select a section or top level frame to mark it ready for dev.

```
width: 2.25rem;
height: 11.59044rem;
flex-shrink: 0;
```

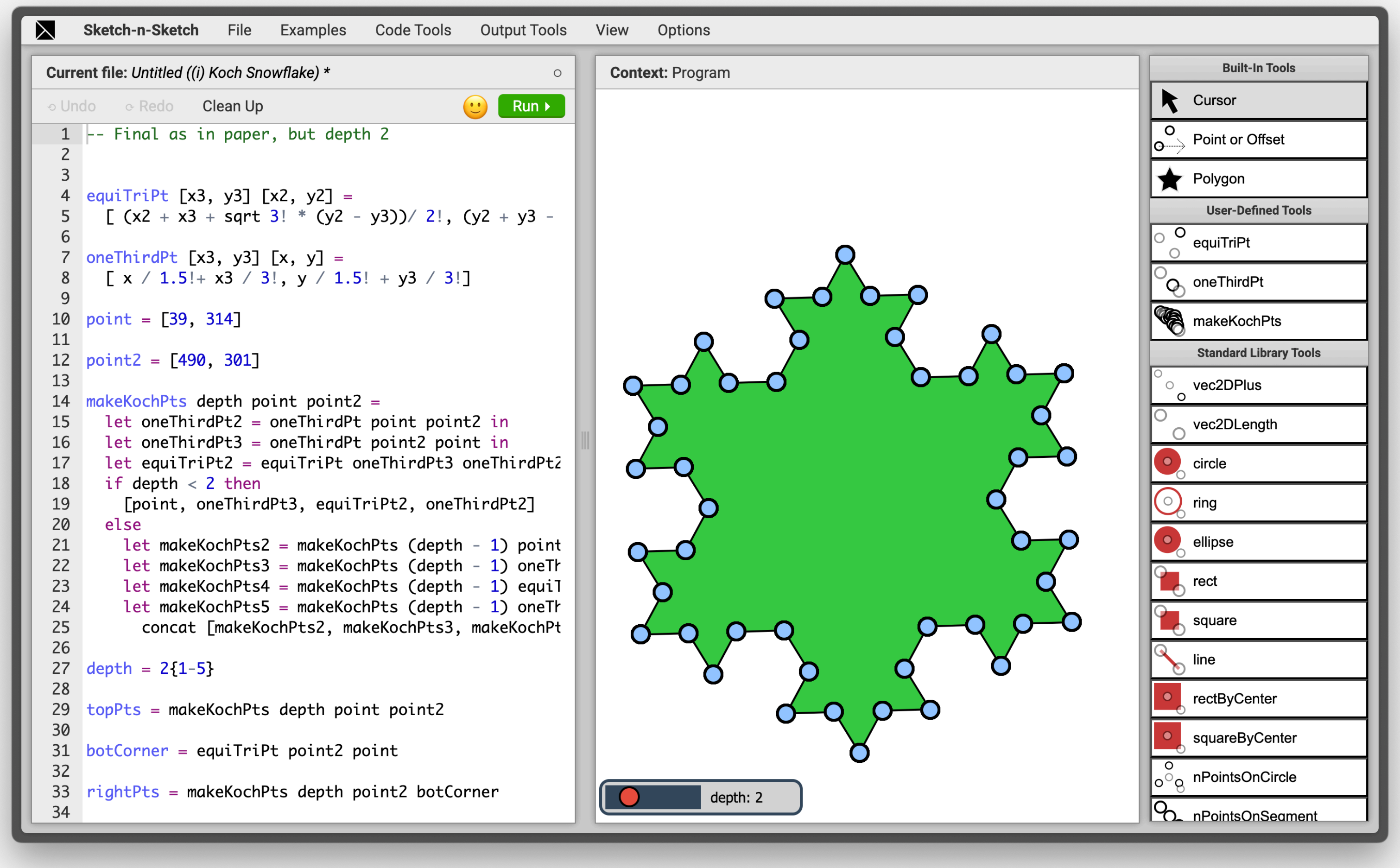
```
<svg xmlns="http://www.w3.org/2000/svg"
  <path d="M17 0.552734H0V186H17V0.5527
  <path d="M36 147.744H19V186H36V147.74
</svg>
```

🟩 #17807E

🟧 #EAAA61

Direct Manipulation Programming

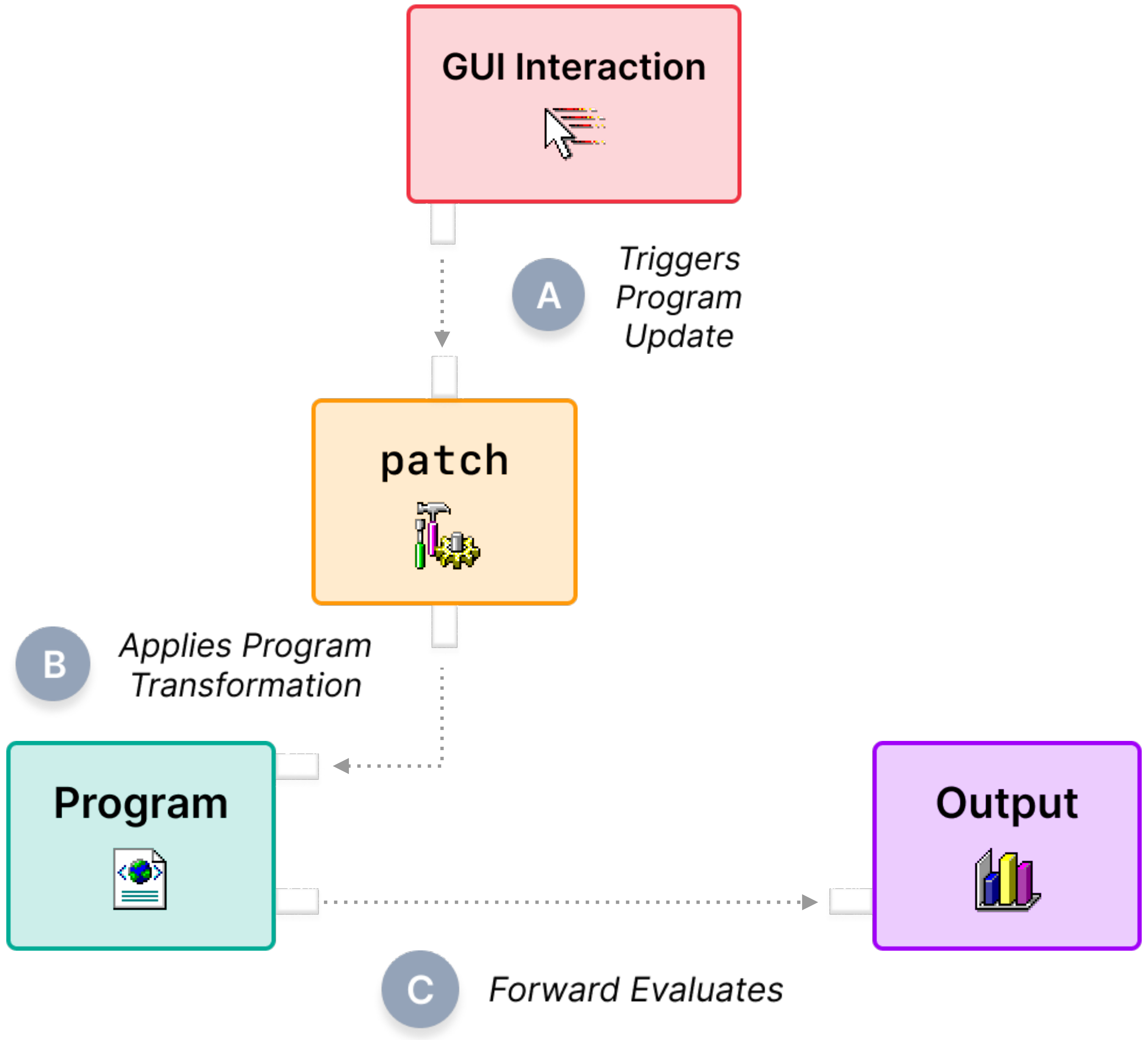
Sketch-n-Sketch (+livelits, BiOOP)



Current file: *Untitled ((i) Koch Snowflake)* *

```
-- Final as in paper, but depth 2


equiTriPt [x3, y3] [x2, y2] =
  [ (x2 + x3 + sqrt 3! * (y2 - y3))/ 2!, (y2 + y3 -

oneThirdPt [x3, y3] [x, y] =
  [ x / 1.5!+ x3 / 3!, y / 1.5! + y3 / 3!]

point = [39, 314]

point2 = [490, 301]

makeKochPts depth point point2 =
  let oneThirdPt2 = oneThirdPt point point2 in
  let oneThirdPt3 = oneThirdPt point2 point in
  let equiTriPt2 = equiTriPt oneThirdPt3 oneThirdPt2
  if depth < 2 then
    [point, oneThirdPt3, equiTriPt2, oneThirdPt2]
  else
    let makeKochPts2 = makeKochPts (depth - 1) point
    let makeKochPts3 = makeKochPts (depth - 1) oneTh
    let makeKochPts4 = makeKochPts (depth - 1) equiT
    let makeKochPts5 = makeKochPts (depth - 1) oneTh
      concat [makeKochPts2, makeKochPts3, makeKochPt

depth = 2{1-5}

topPts = makeKochPts depth point point2

botCorner = equiTriPt point2 point

rightPts = makeKochPts depth point2 botCorner

```

patch-eval Architecture



GUI Interaction

A *Triggers Program Update*

patch

B *Applies Program Transformation*

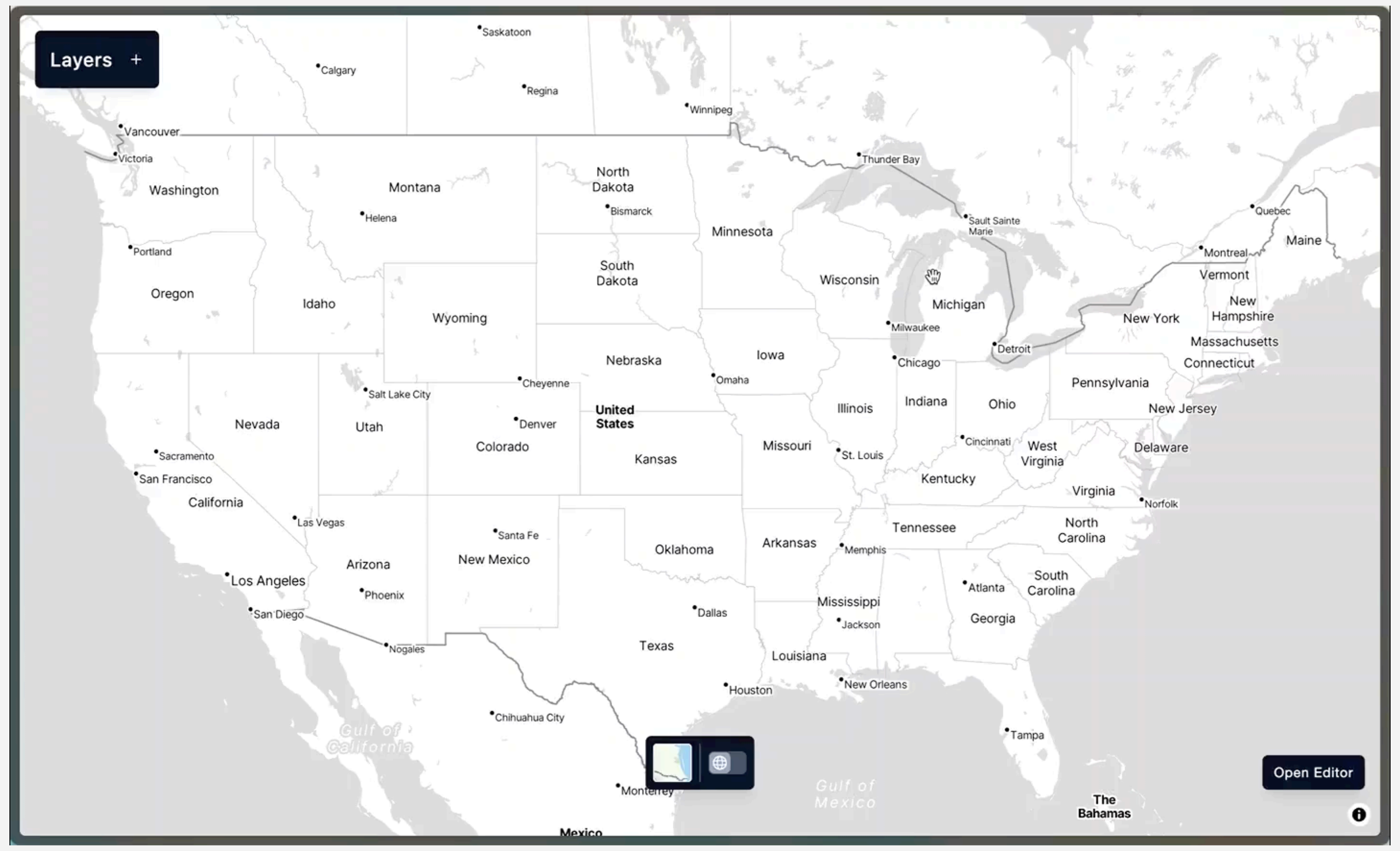
Program

Output

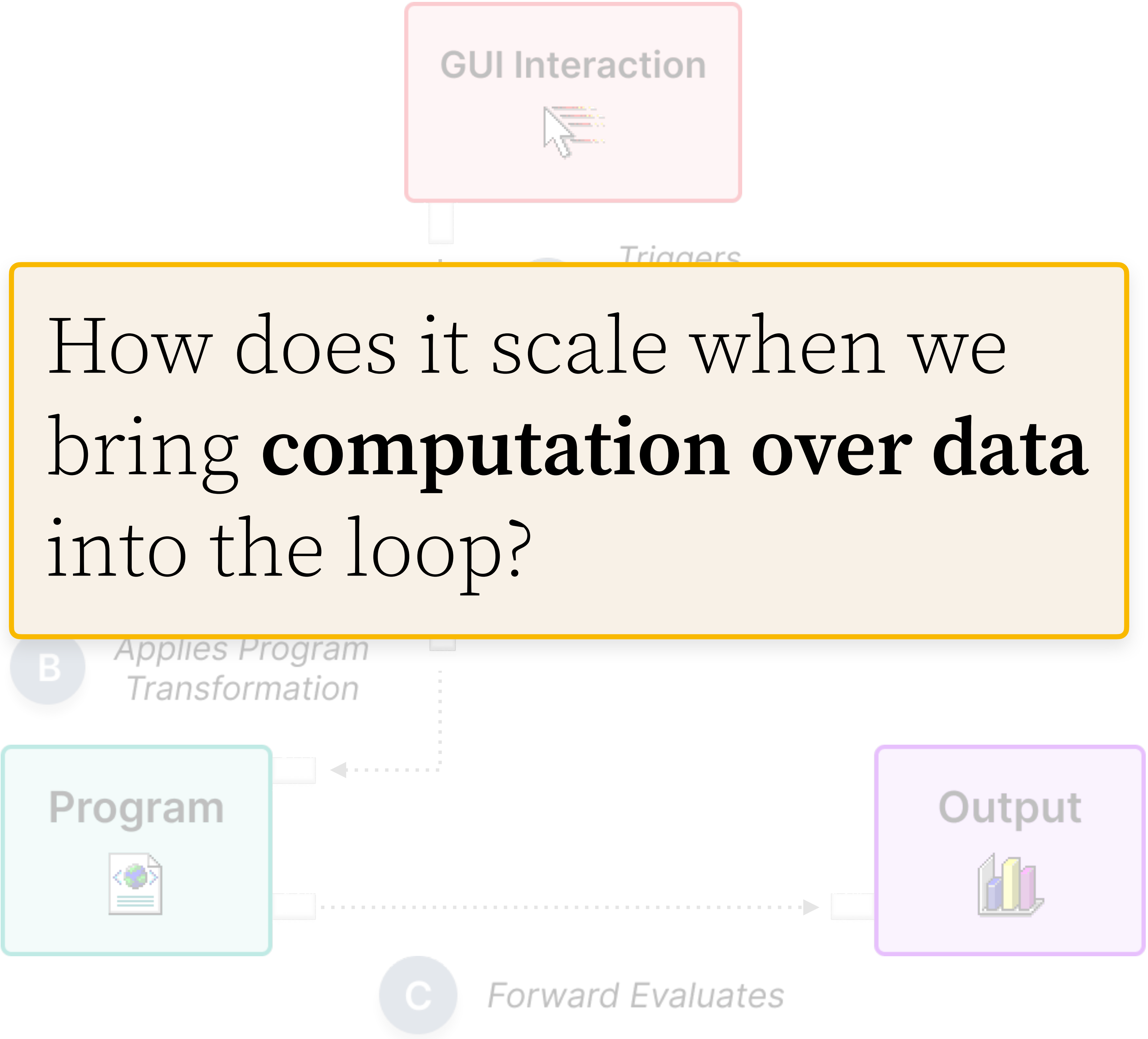
C *Forward Evaluates*

Direct Manipulation Programming

cartokit



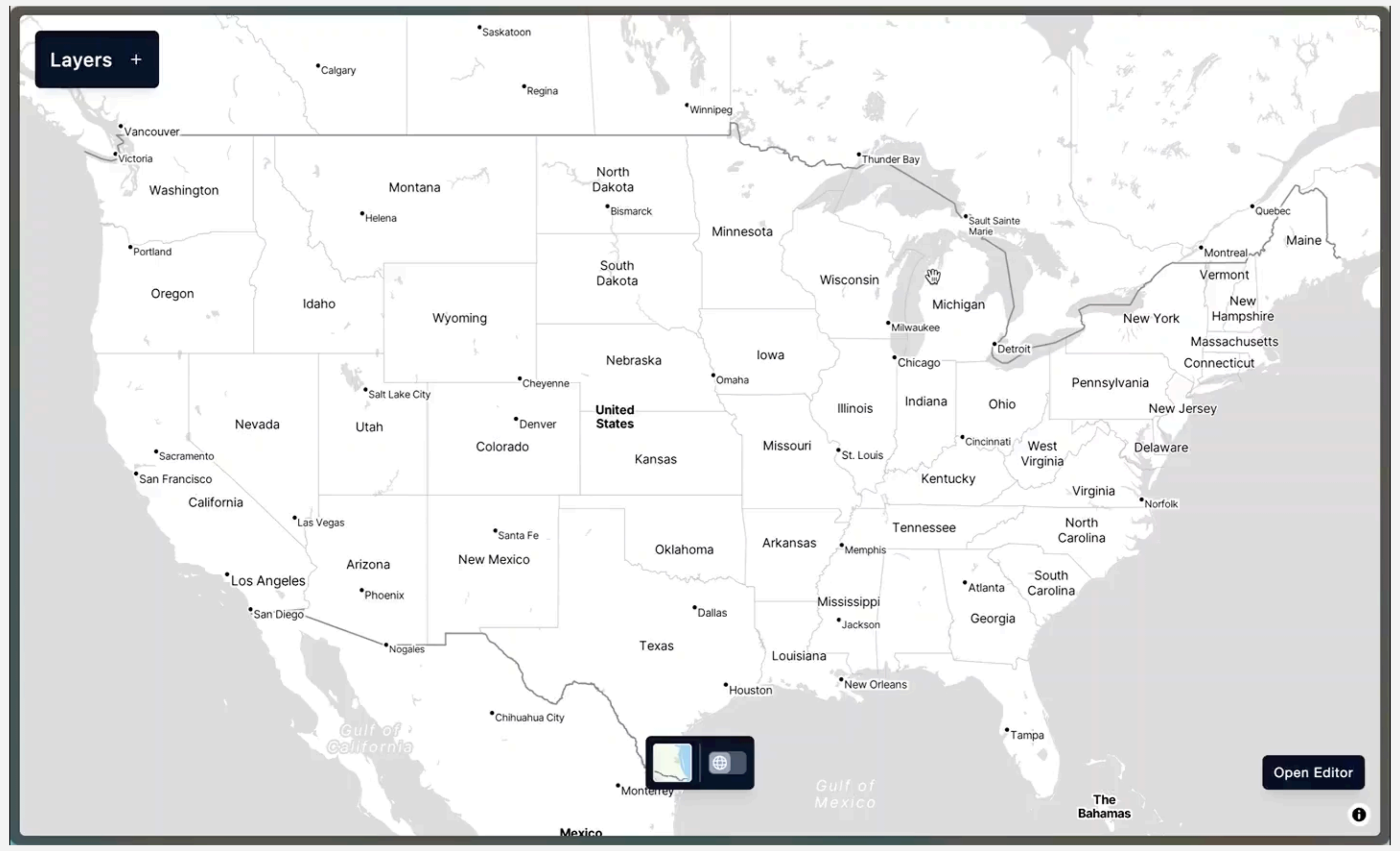
patch–eval Architecture



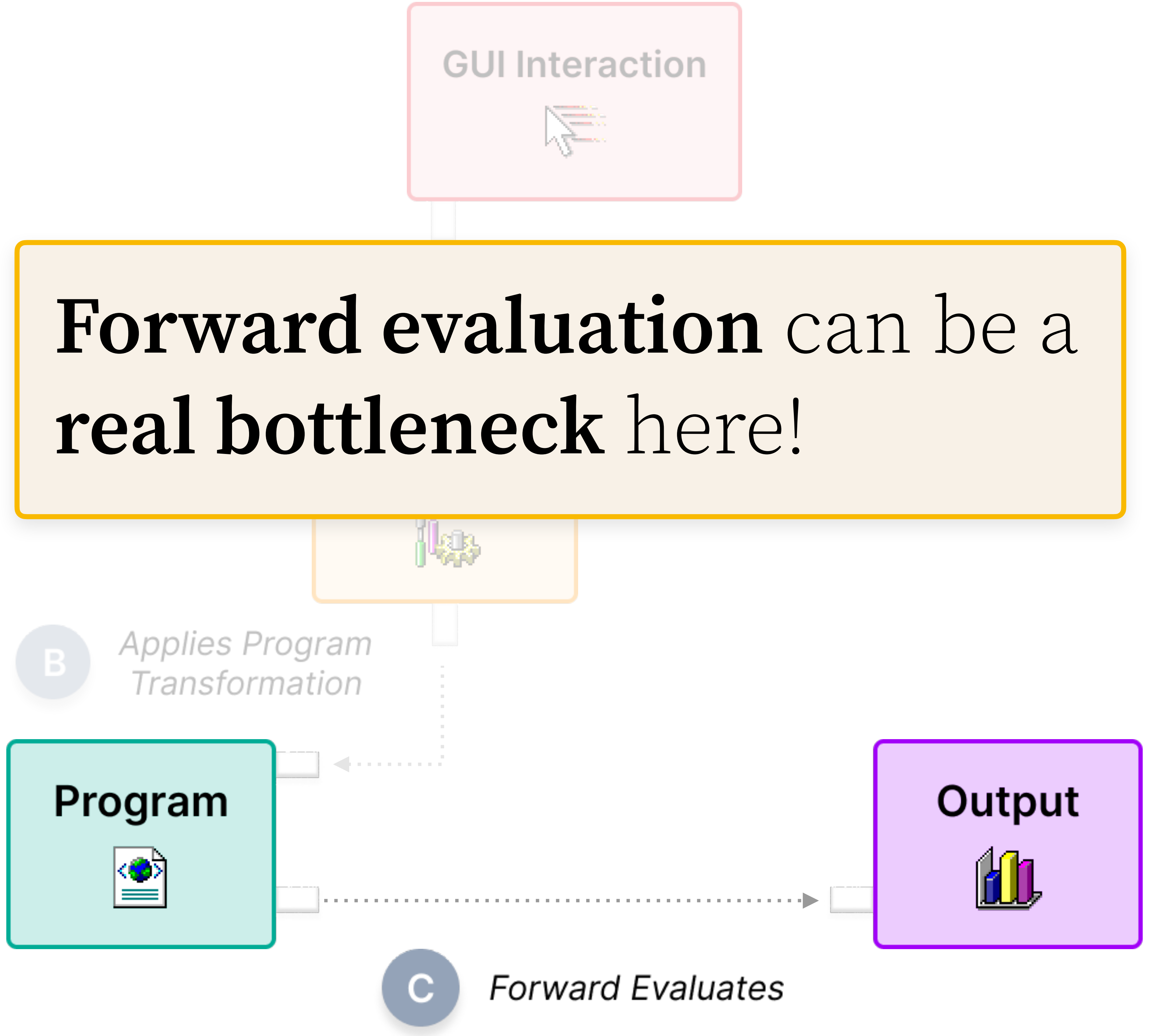
How does it scale when we bring **computation over data** into the loop?

Direct Manipulation Programming

cartokit



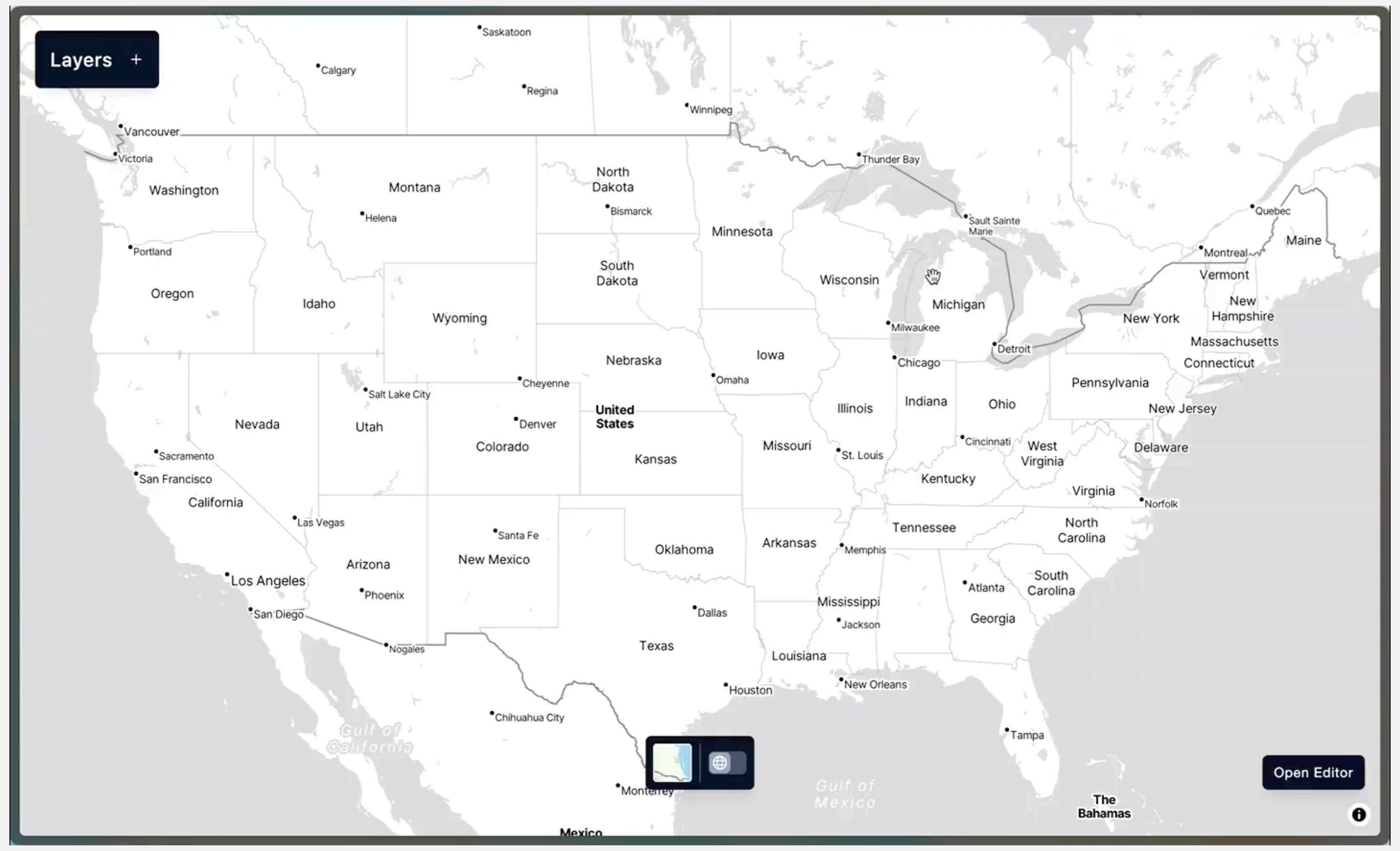
patch–eval Architecture



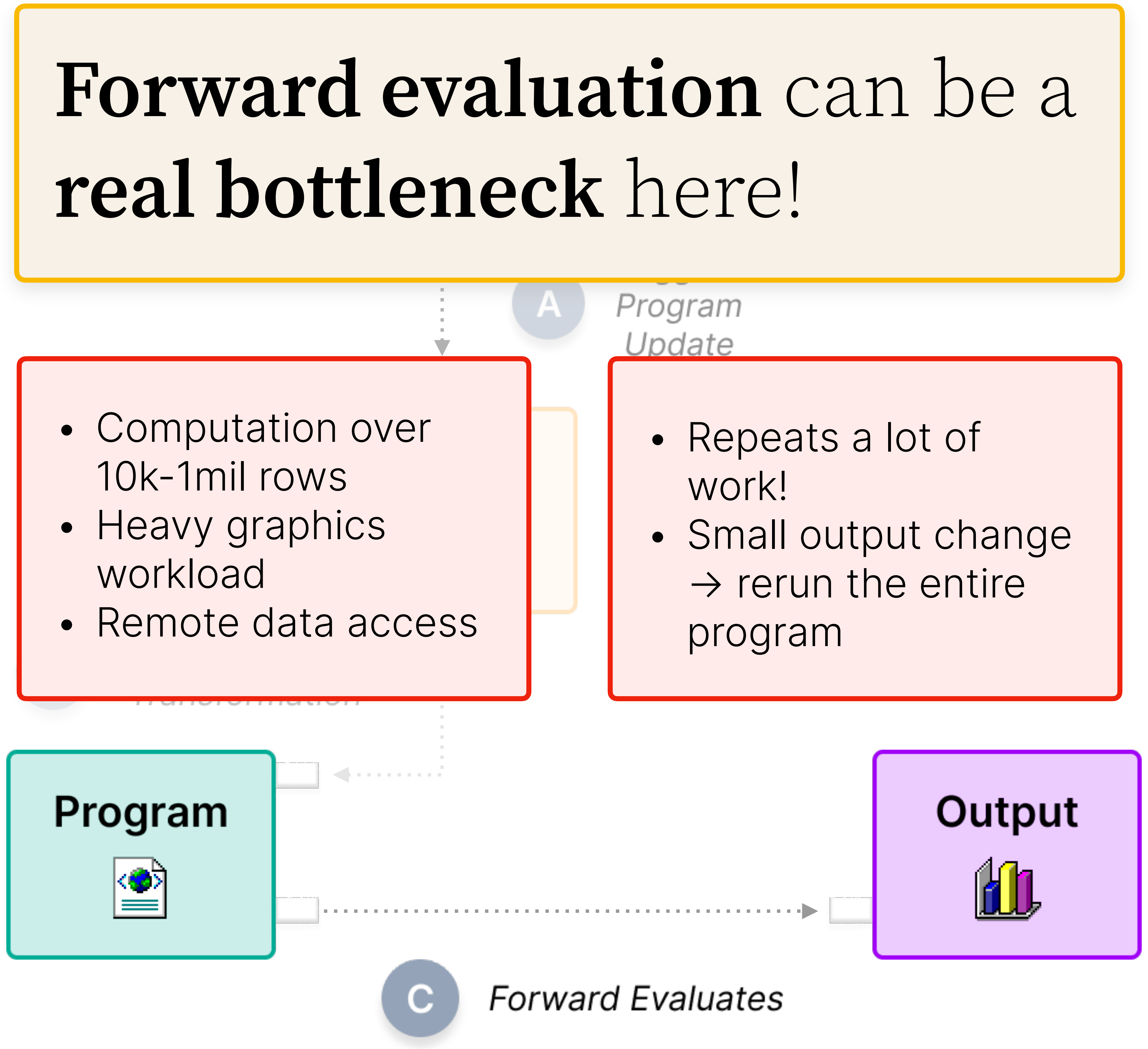
Forward evaluation can be a **real bottleneck** here!

Direct Manipulation Programming

cartokit



patch-eval Architecture



Forward evaluation can be a **real bottleneck** here!

- Computation over 10k-1mil rows
- Heavy graphics workload
- Remote data access

- Repeats a lot of work!
- Small output change → rerun the entire program

Fast Direct Manipulation Programming

with Patch-Reconciliation Correspondence

Fast Direct Manipulation Programming with Patch-Reconciliation Correspondence

Key Insight

Rather than re-evaluating the full program on every GUI interaction, **update the output directly**—ideally, in the smallest way possible.

Fast Direct Manipulation Programming
with Patch-Reconciliation Correspondence

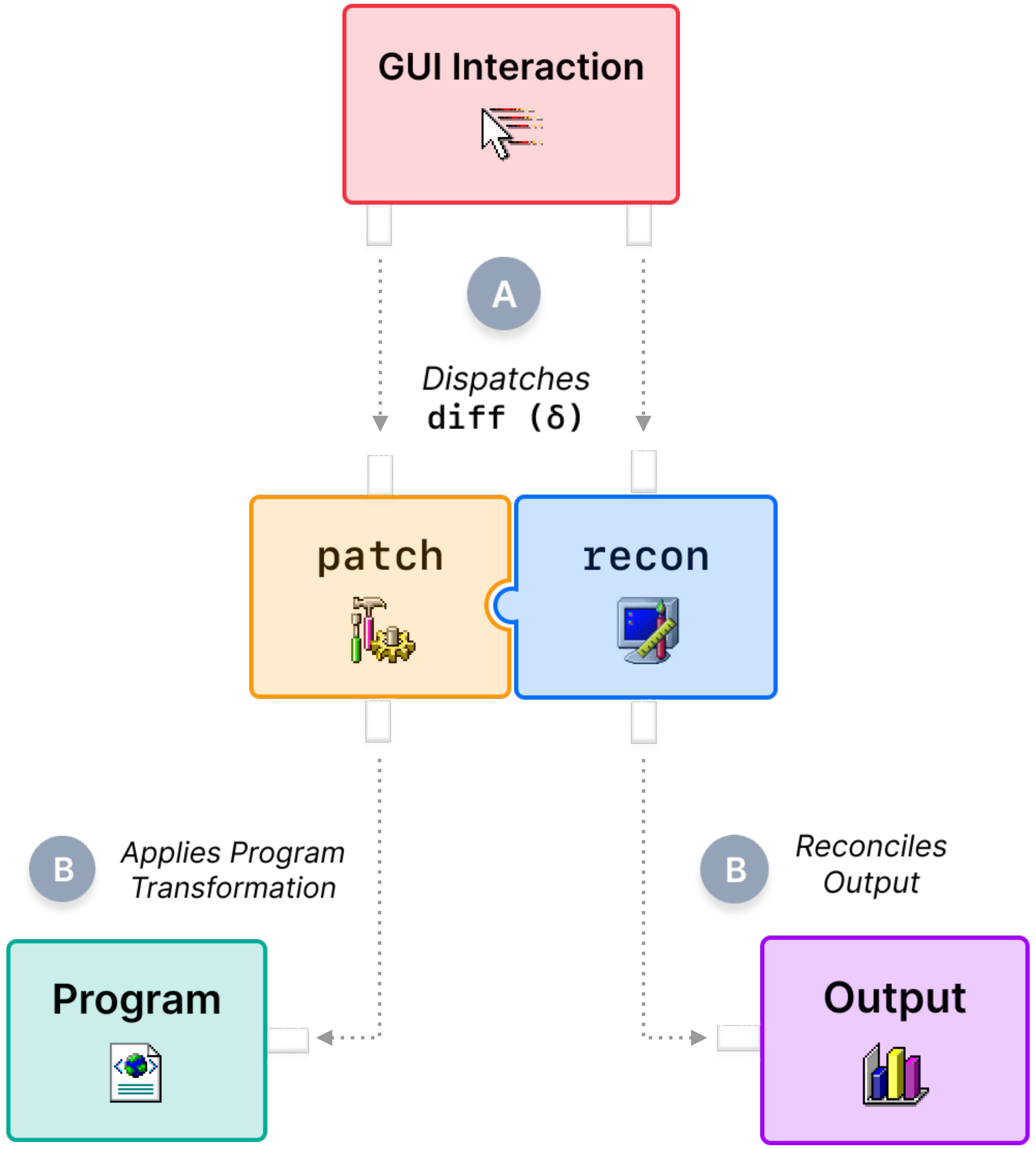
Key Insight

Rather than re-evaluating the full program on every GUI interaction, **update the output directly**—ideally, in the smallest way possible.

Key Implication

We can eliminate the need for program evaluation—the most expensive operation—**altogether!**

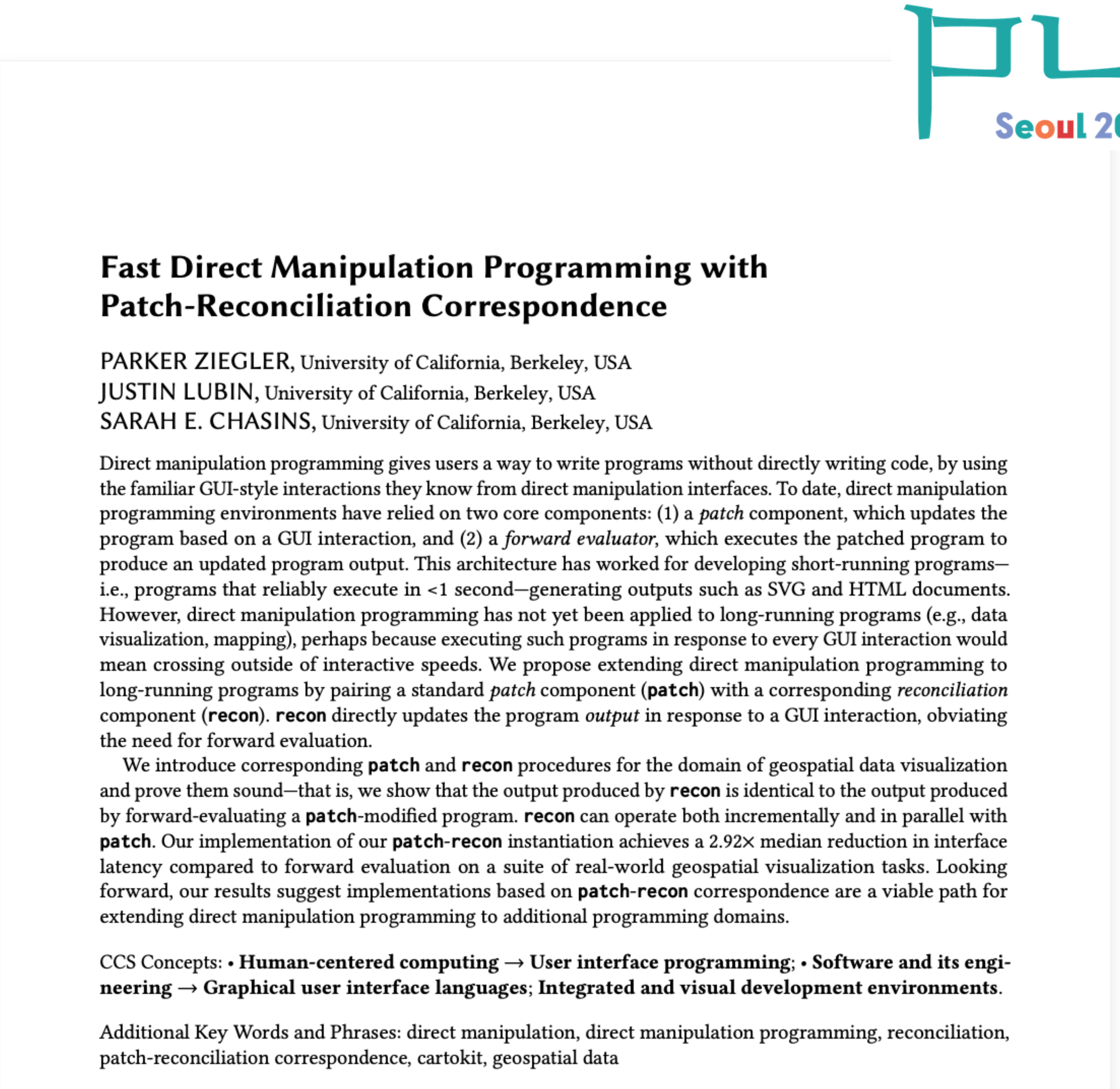
Patch-Reconciliation Correspondence



Patch-Reconciliation Correspondence

For full details, see the proofs in our upcoming PLDI paper!

PLDI
Seoul 2025



Fast Direct Manipulation Programming with Patch-Reconciliation Correspondence

PARKER ZIEGLER, University of California, Berkeley, USA
JUSTIN LUBIN, University of California, Berkeley, USA
SARAH E. CHASINS, University of California, Berkeley, USA

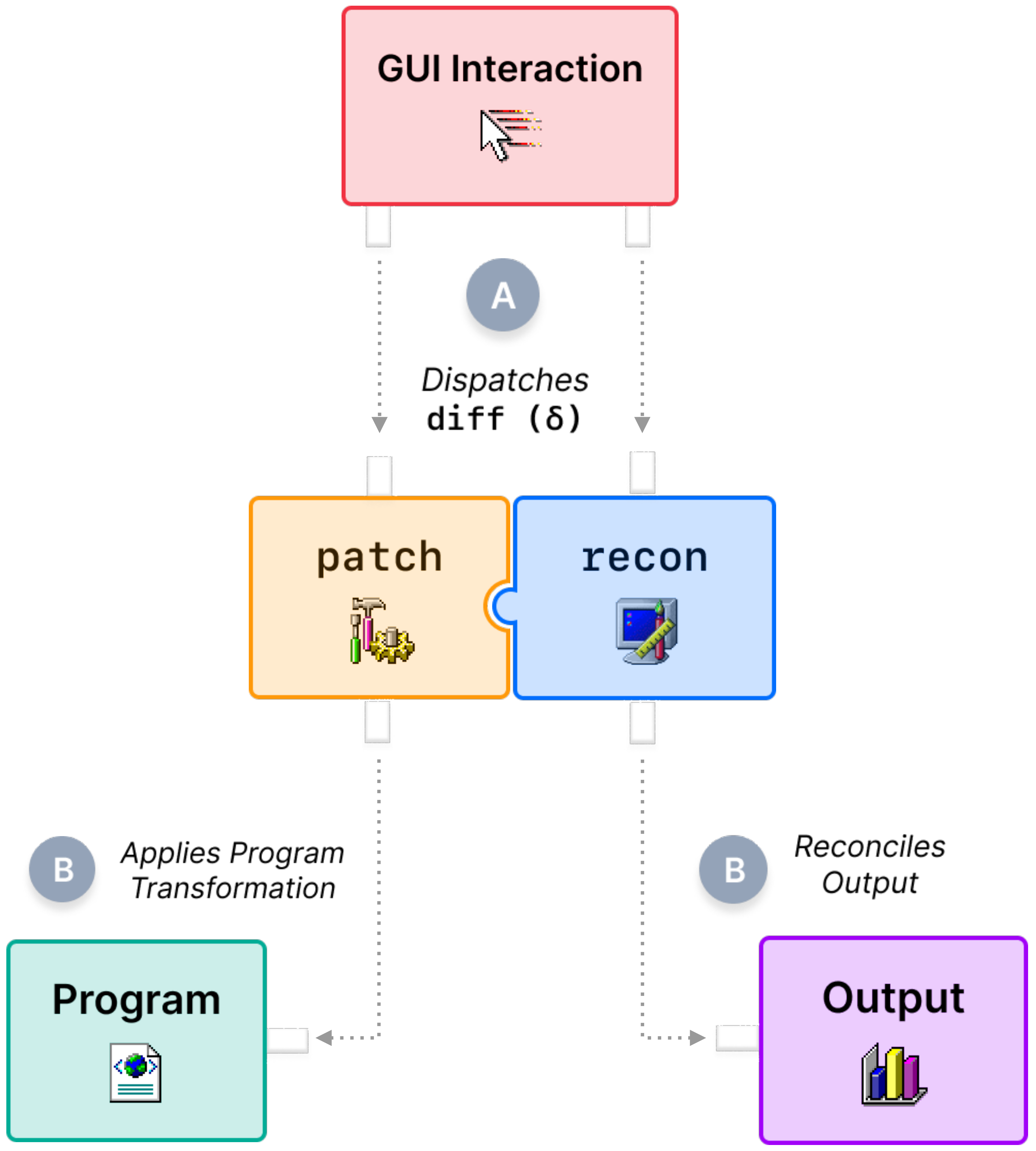
Direct manipulation programming gives users a way to write programs without directly writing code, by using the familiar GUI-style interactions they know from direct manipulation interfaces. To date, direct manipulation programming environments have relied on two core components: (1) a *patch* component, which updates the program based on a GUI interaction, and (2) a *forward evaluator*, which executes the patched program to produce an updated program output. This architecture has worked for developing short-running programs—i.e., programs that reliably execute in <1 second—generating outputs such as SVG and HTML documents. However, direct manipulation programming has not yet been applied to long-running programs (e.g., data visualization, mapping), perhaps because executing such programs in response to every GUI interaction would mean crossing outside of interactive speeds. We propose extending direct manipulation programming to long-running programs by pairing a standard *patch* component (**`patch`**) with a corresponding *reconciliation* component (**`recon`**). **`recon`** directly updates the program *output* in response to a GUI interaction, obviating the need for forward evaluation.

We introduce corresponding **`patch`** and **`recon`** procedures for the domain of geospatial data visualization and prove them sound—that is, we show that the output produced by **`recon`** is identical to the output produced by forward-evaluating a **`patch`**-modified program. **`recon`** can operate both incrementally and in parallel with **`patch`**. Our implementation of our **`patch-recon`** instantiation achieves a 2.92× median reduction in interface latency compared to forward evaluation on a suite of real-world geospatial visualization tasks. Looking forward, our results suggest implementations based on **`patch-recon`** correspondence are a viable path for extending direct manipulation programming to additional programming domains.

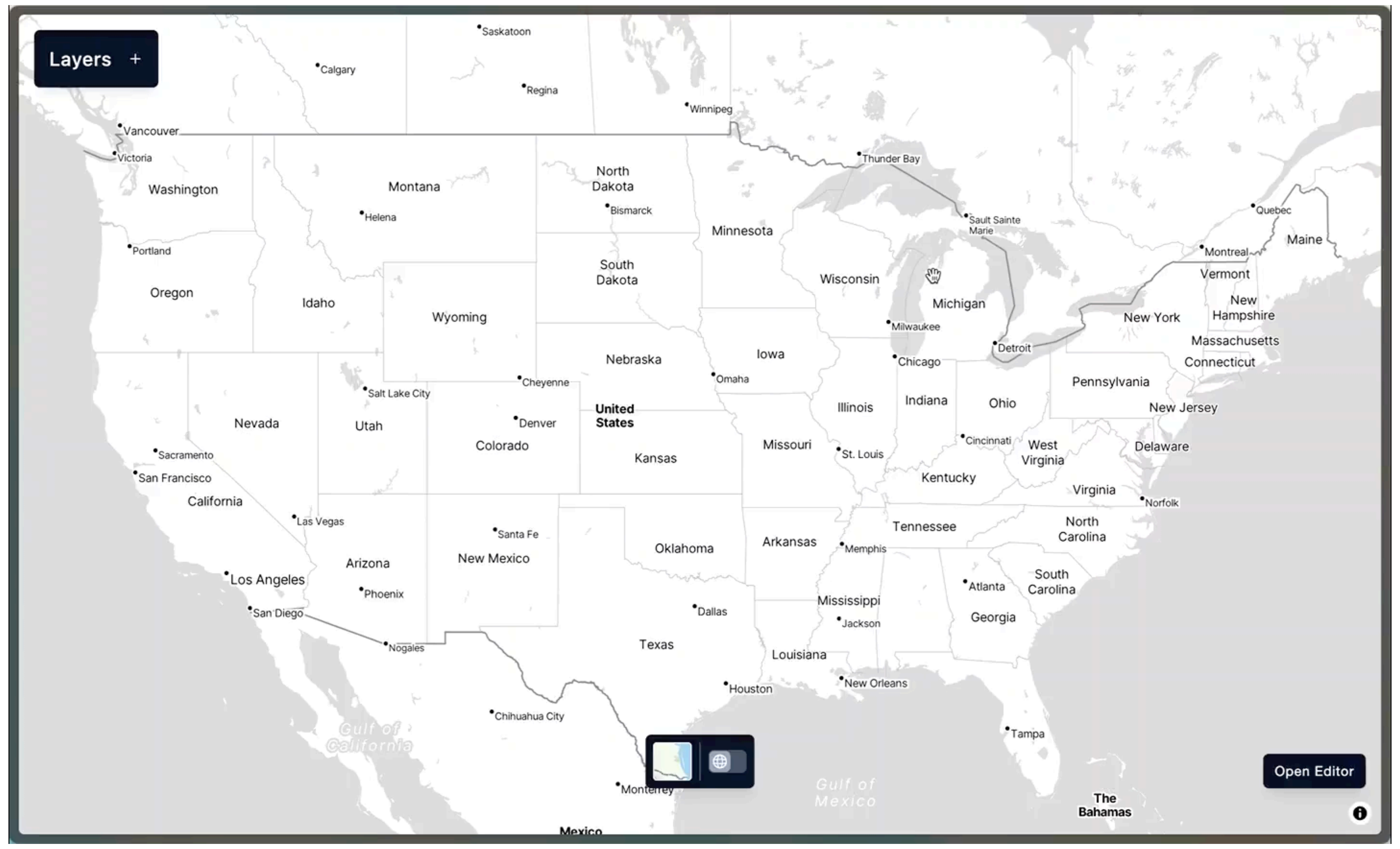
CCS Concepts: • **Human-centered computing → User interface programming**; • **Software and its engineering → Graphical user interface languages**; **Integrated and visual development environments**.

Additional Key Words and Phrases: direct manipulation, direct manipulation programming, reconciliation, patch-reconciliation correspondence, cartokit, geospatial data

patch–recon Architecture

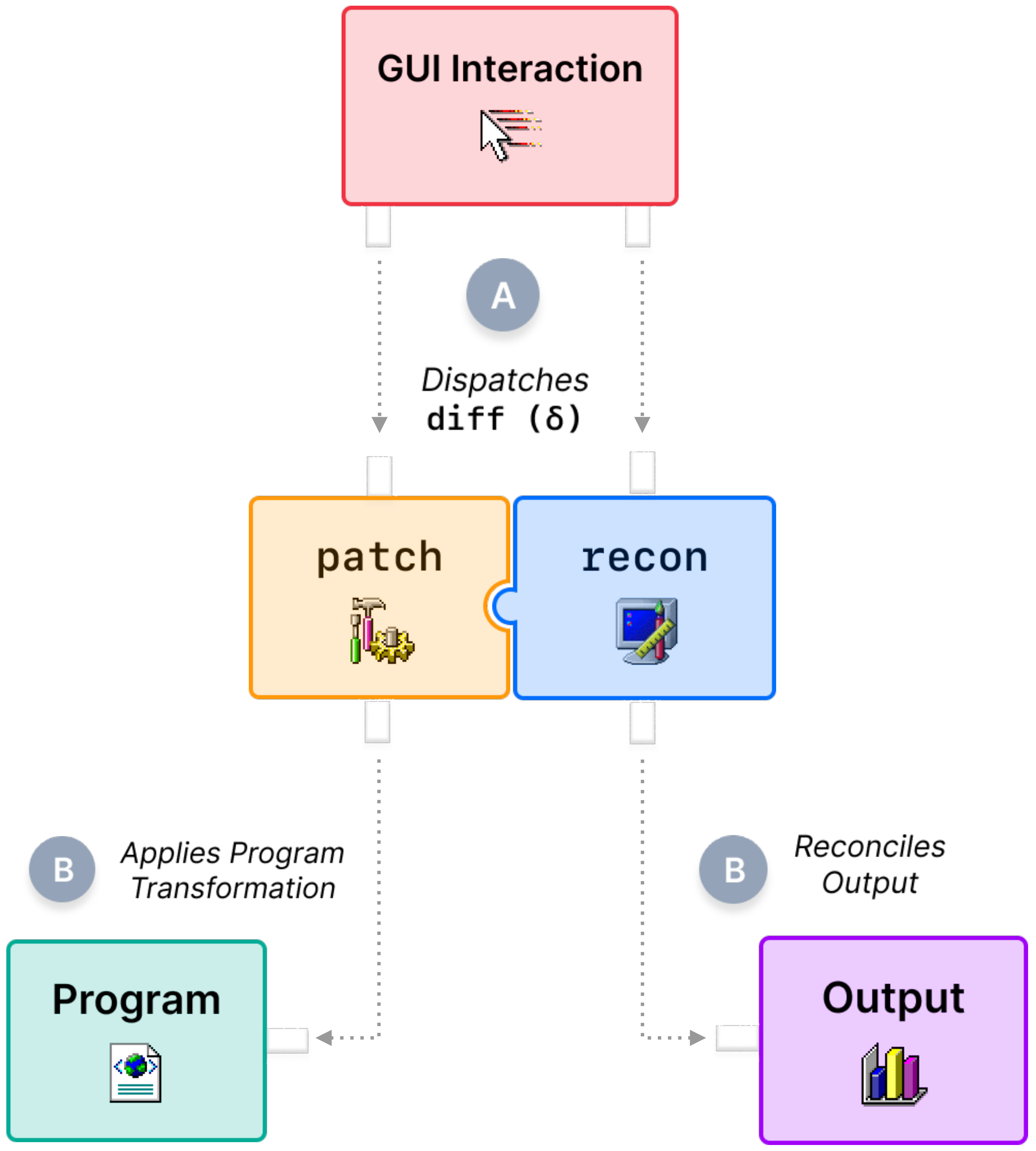


Patch-Reconciliation Correspondence



cartokit

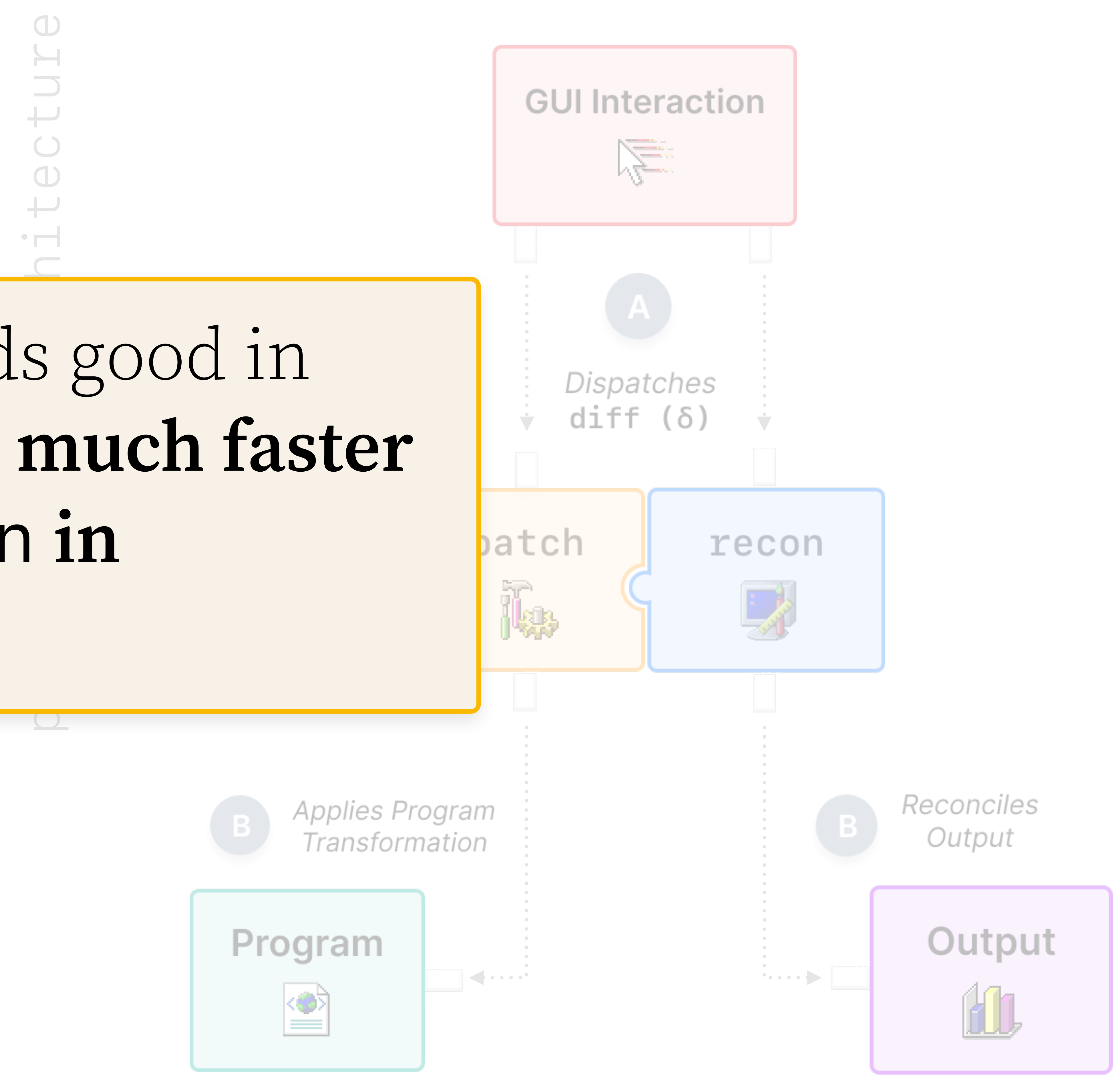
patch-recon Architecture



Patch-Reconciliation Correspondence

So this all sounds good in theory, but **how much faster is `patch–recon` in practice?**

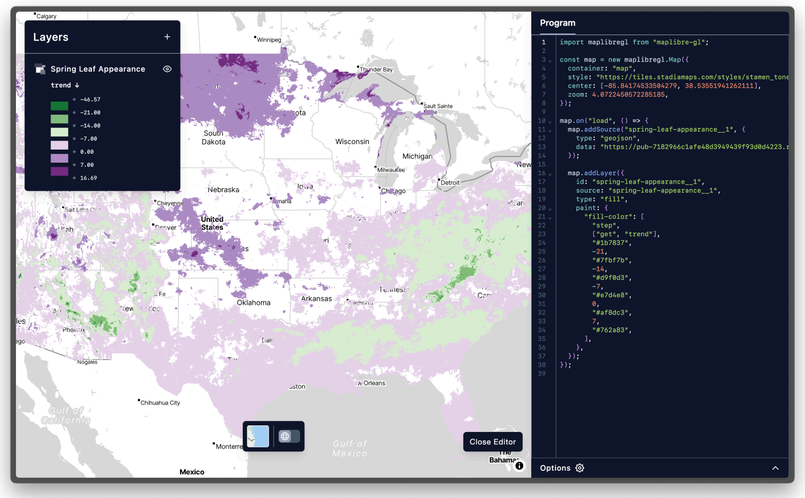
cartokit



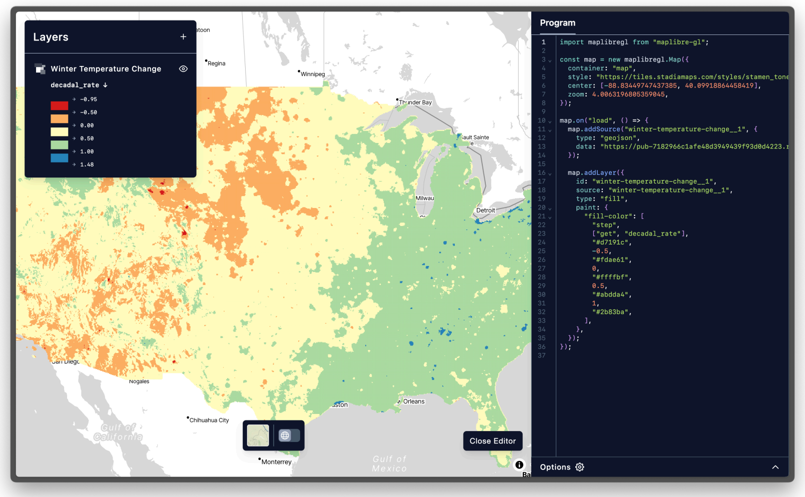
Patch-Reconciliation Correspondence

Benchmark Maps

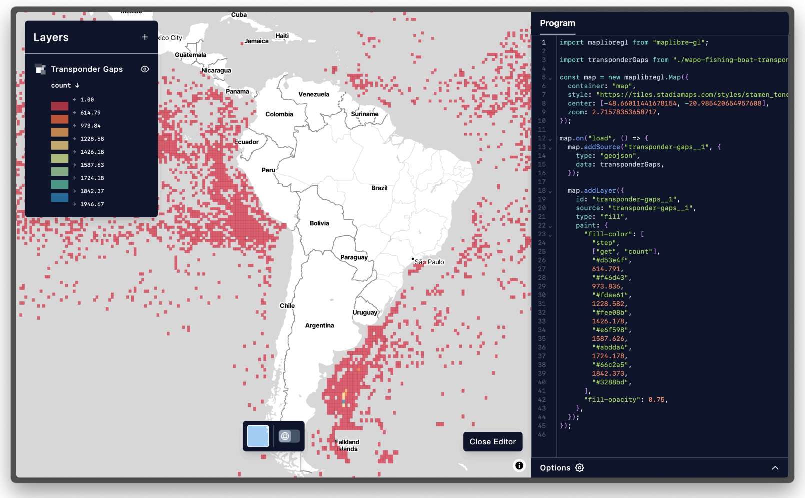
Maps of the April 2024 Total Solar Eclipse



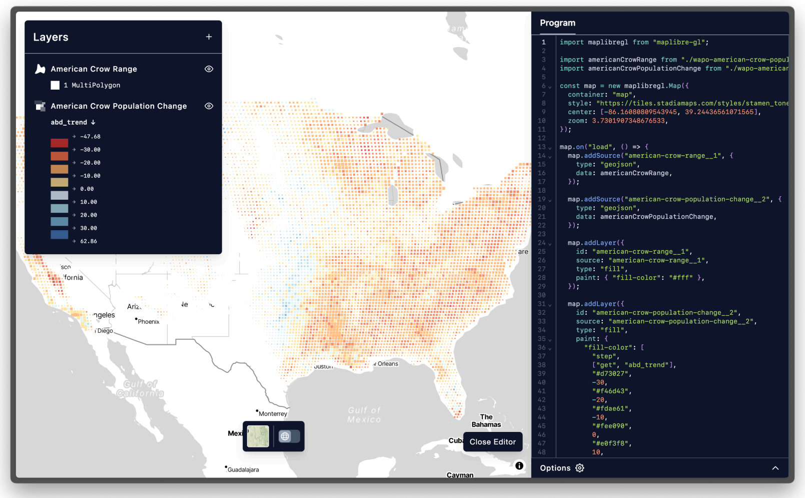
You're not crazy. Spring is getting earlier. Find out how it's changed in your town.



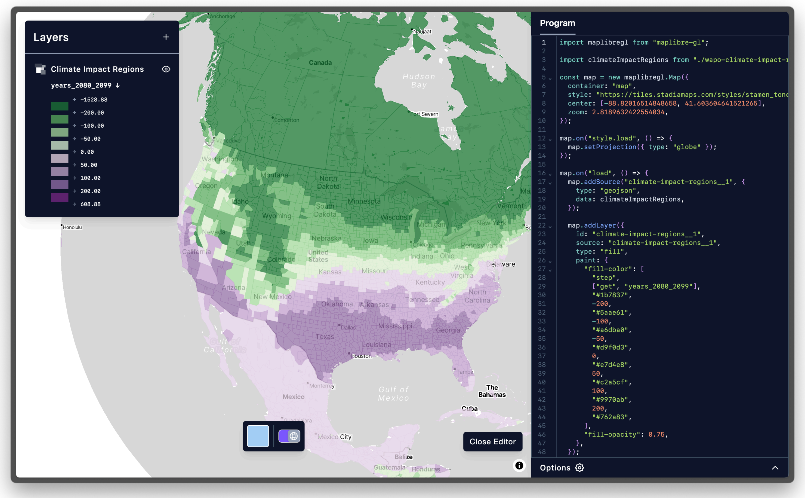
Winter is warming almost everywhere. See how it's changed in your town.



A boat went dark. Finding it could save the world's fish.



Bird populations are declining. Some are in your neighborhood.



Will global warming make temperature less deadly?

Results

2.92x

median reduction in UI latency

Patch-Reconciliation Correspondence

Results

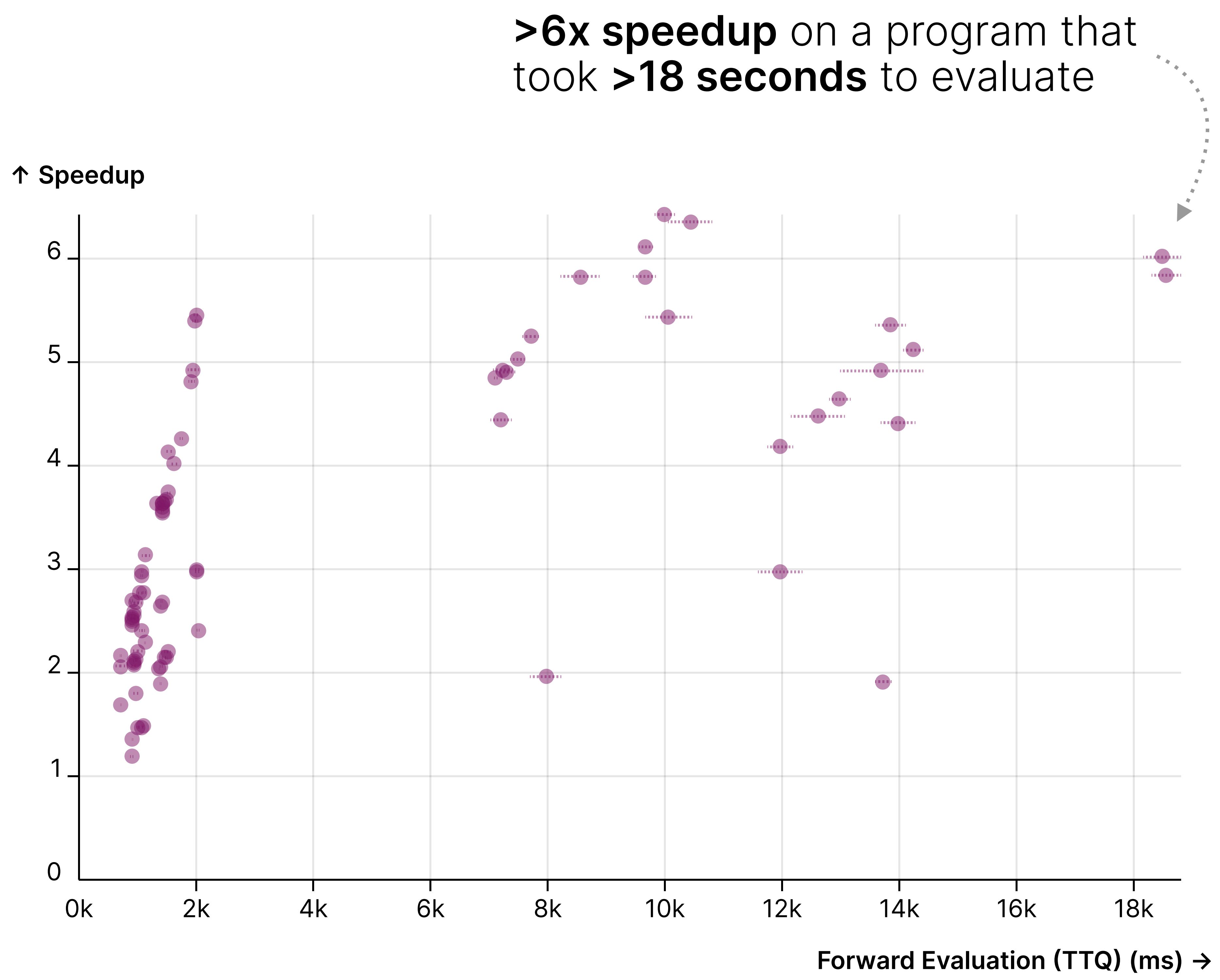


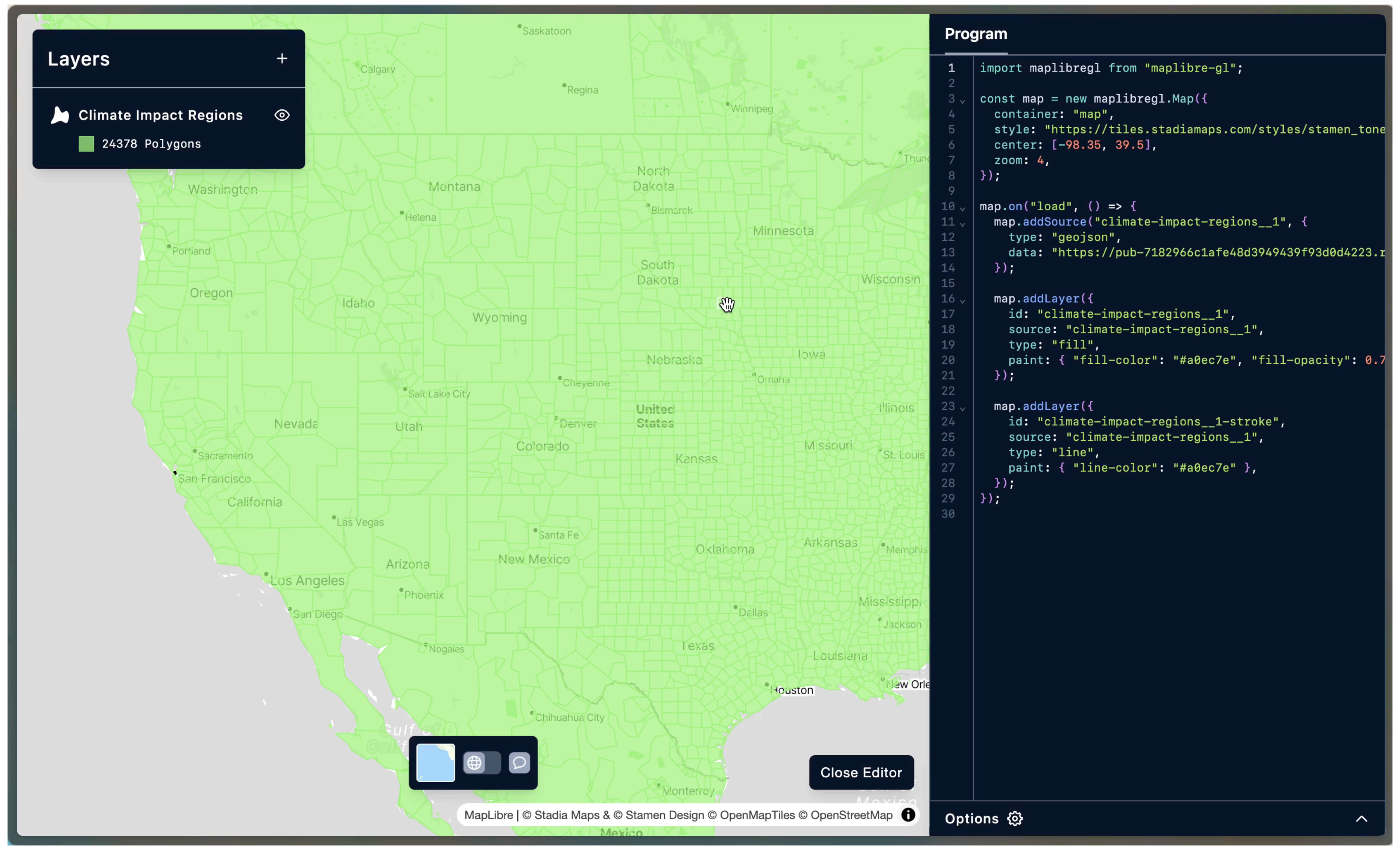
Fig 9. Comparing forward evaluation TTQ against speedup from reconciliation.

Longer-running programs see larger speedups!

0.732

correlation between eval time and speedup

Patch-Reconciliation Correspondence+LLMs

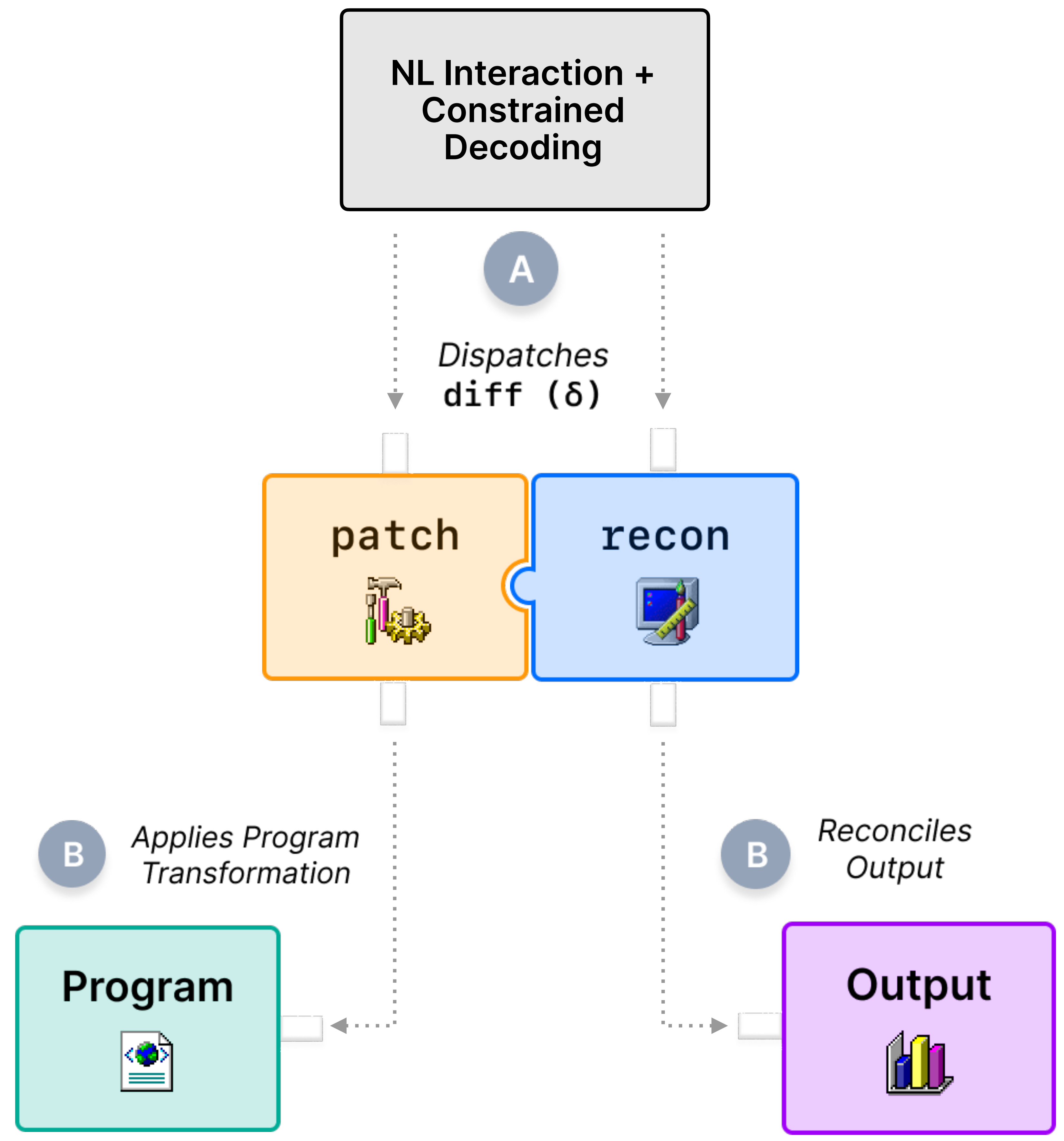

Layers
+
Climate Impact Regions
24378 Polygons
Close Editor
Options
MapLibre | © Stadia Maps & © Stamen Design © OpenMapTiles © OpenStreetMap

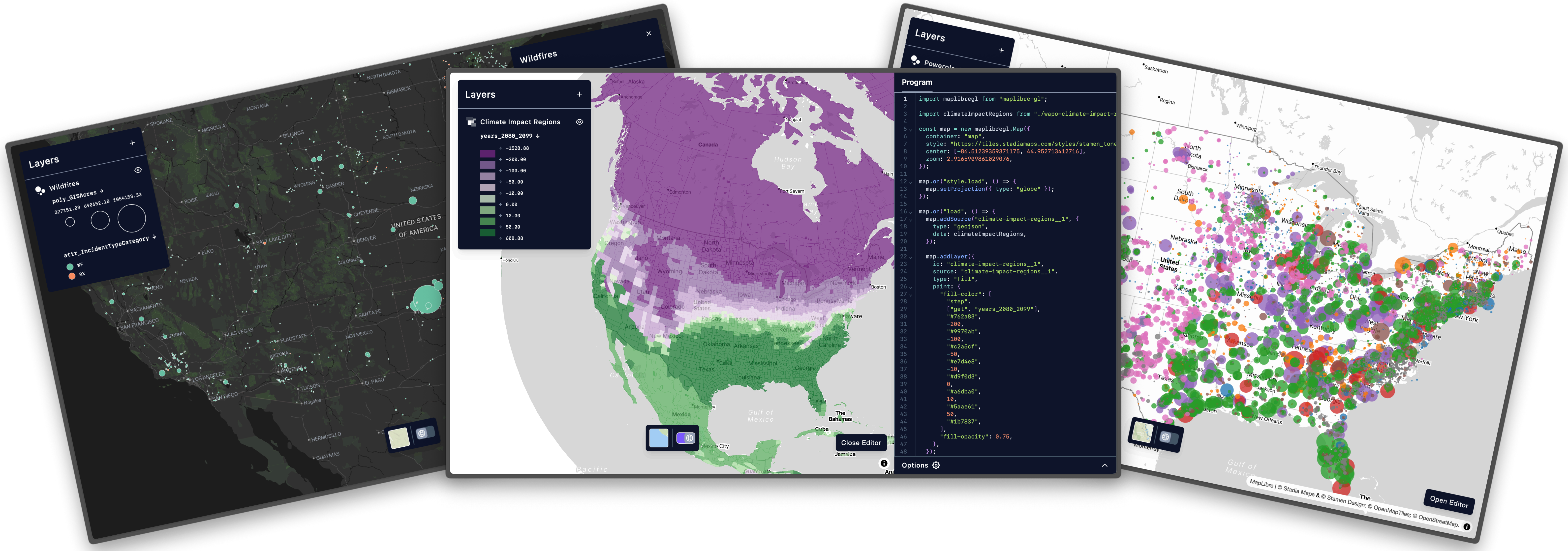
Program

```
import maplibregl from "maplibre-gl";

const map = new maplibregl.Map({
  container: "map",
  style: "https://tiles.stadiamaps.com/styles/stamen_tone
  center: [-98.35, 39.5],
  zoom: 4,
});

map.on("load", () => {
  map.addSource("climate-impact-regions__1", {
    type: "geojson",
    data: "https://pub-7182966c1afe48d3949439f93d0d4223.r
  });

  map.addLayer({
    id: "climate-impact-regions__1",
    source: "climate-impact-regions__1",
    type: "fill",
    paint: { "fill-color": "#a0ec7e", "fill-opacity": 0.7
  });

  map.addLayer({
    id: "climate-impact-regions__1-stroke",
    source: "climate-impact-regions__1",
    type: "line",
    paint: { "line-color": "#a0ec7e" },
  });
});
```

cartokit

patch-recon Architecture





```
import maplibregl from "maplibre-gl";

import climateImpactRegions from "./wapo-climate-impact-r

const map = new maplibregl.Map({
  container: "map",
  style: "https://tiles.stadiamaps.com/styles/stamen_tone
  center: [-86.51239359371175, 44.952713412716],
  zoom: 2.9165909861029076,
});

map.on("style.load", () => {
  map.setProjection({ type: "globe" });
});

map.on("load", () => {
  map.addSource("climate-impact-regions__1", {
    type: "geojson",
    data: climateImpactRegions,
  });

  map.addLayer({
    id: "climate-impact-regions__1",
    source: "climate-impact-regions__1",
    type: "fill",
    paint: {
      "fill-color": [
        "step",
        ["get", "years_2080_2099"],
        "#762a83",
        -200,
        "#9970ab",
        -100,
        "#c2a5cf",
        -50,
        "#e7d4e8",
        -10,
        "#d9f0d3",
        0,
        "#a6dba0",
        10,
        "#5aae61",
        50,
        "#1b7837",
      ],
      "fill-opacity": 0.75,
    },
  });
```

Fast Direct Manipulation Programming

with Patch-Reconciliation Correspondence

Get in touch!

peziegler@cs.berkeley.edu

parkie-doo.sh

Try it out!

CK alpha.cartokit.dev

github.com/parkerziegler/cartokit