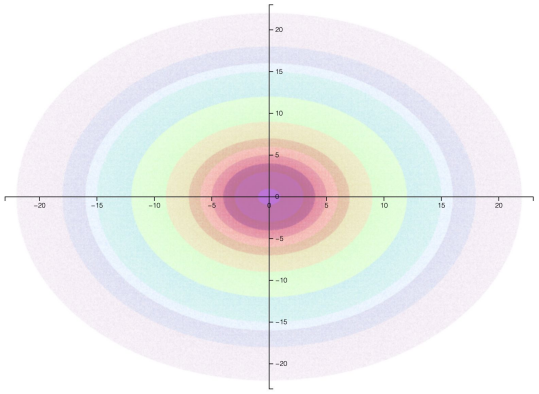
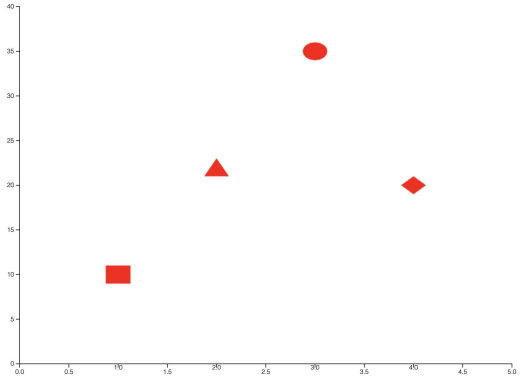
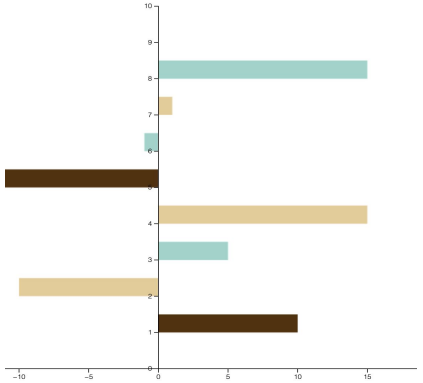
Epiviz.gl: A declarative approach to high-performance visualizations

Presentation by: Sam Rosen

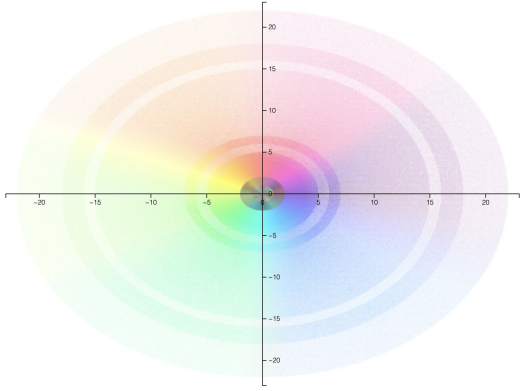


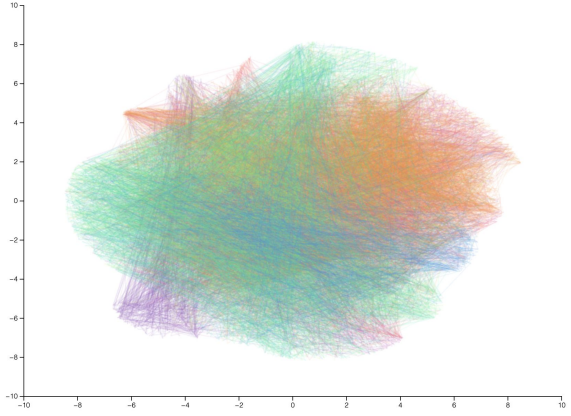


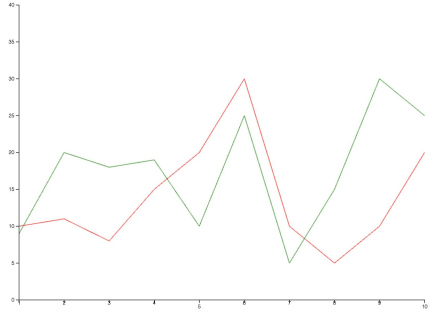


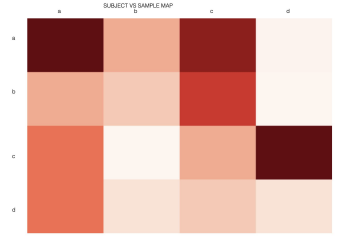
10 Million Points!
(~10 fps zoomed out, 60 fps zoomed in)

All 60 fps





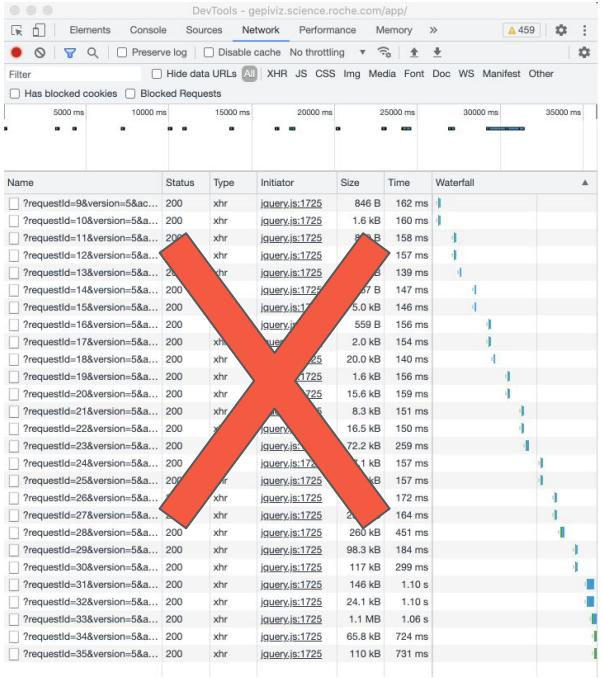




Motivation - Requirements

Simple Summary: Ease-of-use for visualizing (genomic) data

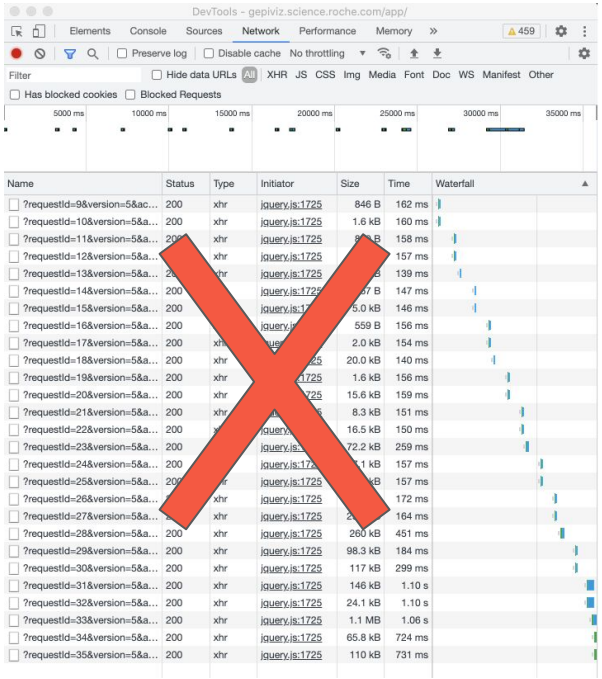
Avoid use of a backend



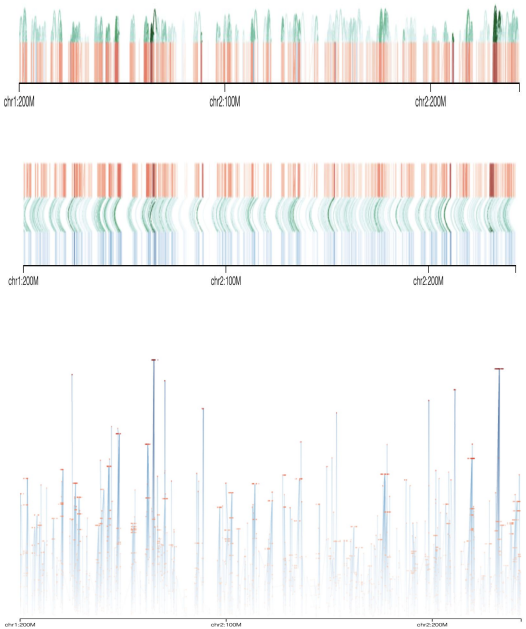
Motivation - Requirements

Simple Summary: Ease-of-use for visualizing (genomic) data

Avoid use of a backend



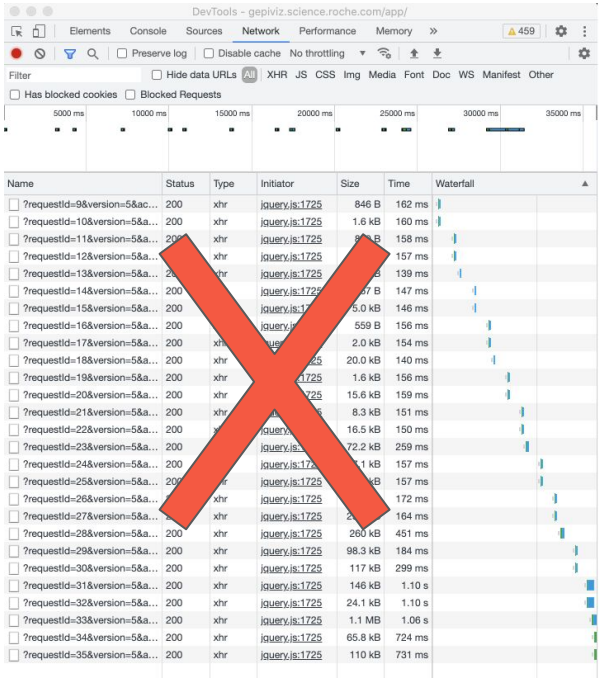
Declarative for flexibility and user choices



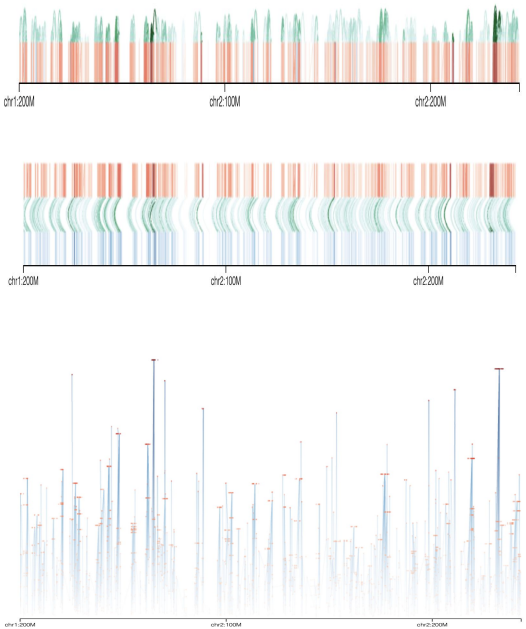
Motivation - Requirements

Simple Summary: Ease-of-use for visualizing (genomic) data

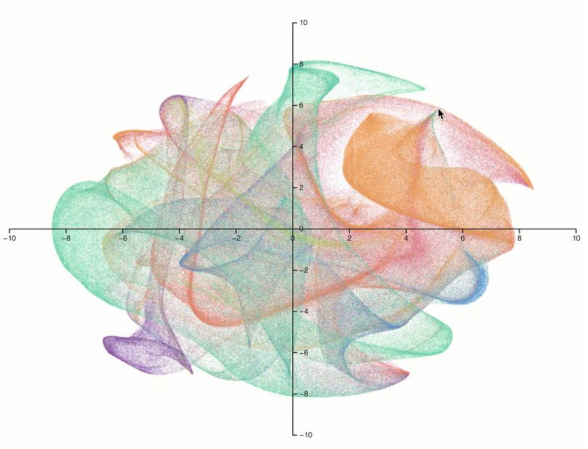
Avoid use of a backend



Declarative for flexibility and user choices



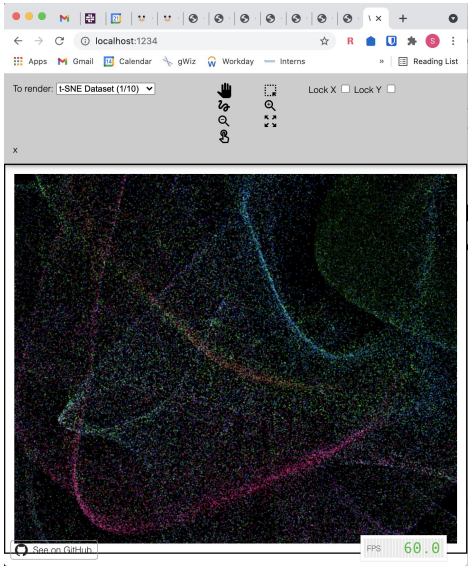
Enable visualization of large amounts of data with precision



Motivation - Technologies

Web Workers and OffScreenCanvas

- Rendering on own thread, avoids UI processing
- Keeps application responsive



Main Thread
Creates worker
Transfers canvas to worker

Web Worker
No access to DOM
Independent namespace
Runs on own thread

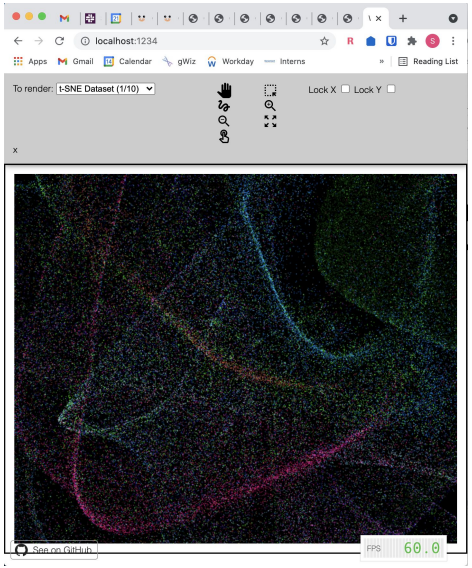
Main thread and Worker can post messages to each other with basic JavaScript objects copied

Initial demo of Web Workers: https://www.jkanche.com/uepiviz/

Motivation - Technologies

Web Workers and OffScreenCanvas

- Rendering on own thread, avoids UI processing
- Keeps application responsive



Main Thread
Creates worker
Transfers canvas to worker

Web Worker
No access to DOM
Independent namespace
Runs on own thread

Main thread and Worker can post messages to each other with basic JavaScript objects copied

Initial demo of Web Workers: https://www.jkanche.com/uepiviz/

WebAssembly (WASM)

- Allows native performance for compiled JavaScript
- Not yet used in epiviz.gl, but would make a noticeable difference

```
;; INFO asc module.ts --textFile module.wat --binaryFile module.wasm -O3 --runtime stub
(module
 (type $i32_=>_i32 (func (param i32) (result i32)))
 (memory $0 0)
 (export "fib" (func $module/fib))
 (export "memory" (memory $0))
 (func $module/fib (param $0 i32) (result i32)
  (local $1 i32)
  (local $2 i32)
  (local $3 i32)
  i32.const 1
  local.set $1
  local.get $0
  i32.const 0
  i32.gt_s
  if
   loop $while-continue|0
    local.get $0
    i32.const 1
    i32.sub
    local.tee $0
    if
     local.get $1
     local.get $2
     i32.add
     local.get $1
     local.set $2
     local.set $1
```

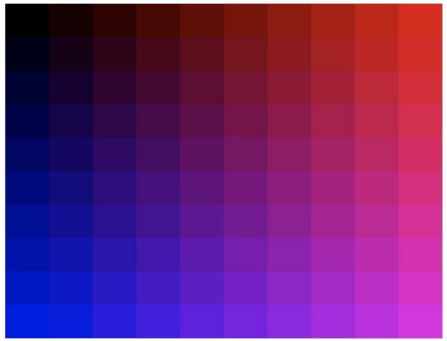
Credit: assemblyscript.org

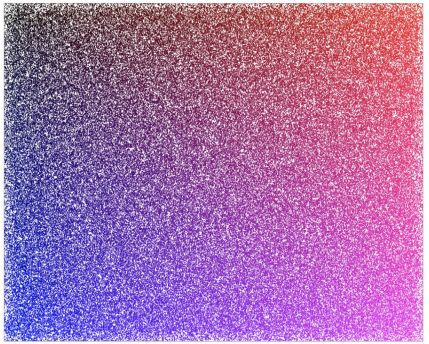
Initial Investigation

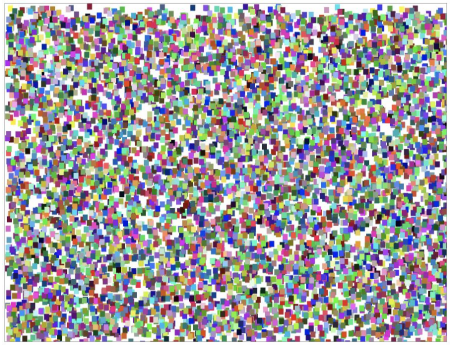
Live demo: https://samgrosen.github.io/offscreen-canvas-scrolling/

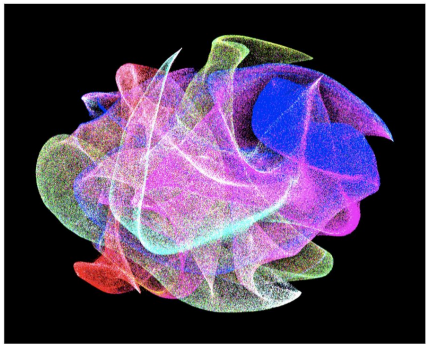
Framework	Interactive	Offscreen	Render all test sets
WebGL	Yes	Yes	Yes
Base Canvas	Yes	Yes	Somewhat
PIXI.js	Yes	With polyfill	No

- Also investigated **THREE.js** and **Babylon.js**, but both were deemed insufficient before final tests
- PIXI.js is used in Gosling.js, another WebGL accelerated declarative graphics framework









Test sets used: Square gradient, jittered gradient, random points, and 1.5M points

Why WebGL?

- Offset **most** calculations (rasterization, clipping, viewport) to GPU
- Use of WebGL only has less memory overhead than popular libraries
- twgl.js makes writing WebGL much less verbose
- Ultimately **very good** at rendering millions of **simple shapes** very fast

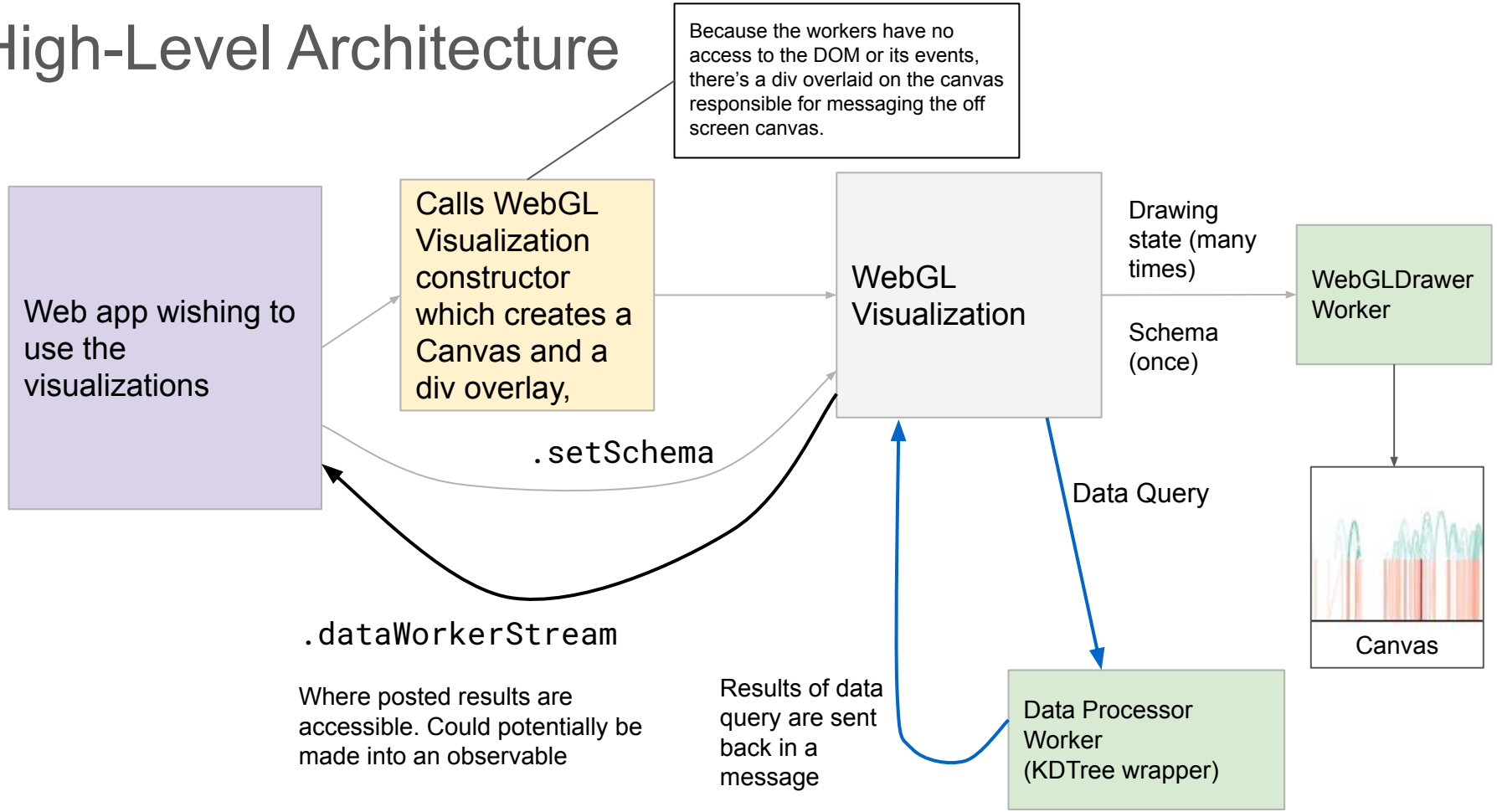
What's the catch?

- Write shader code (hard)
- Limit transfer of data to GPU (slow)
- Special consideration for complex shapes (outside libraries)
- Build vertices/buffers in JavaScript (bottleneck)

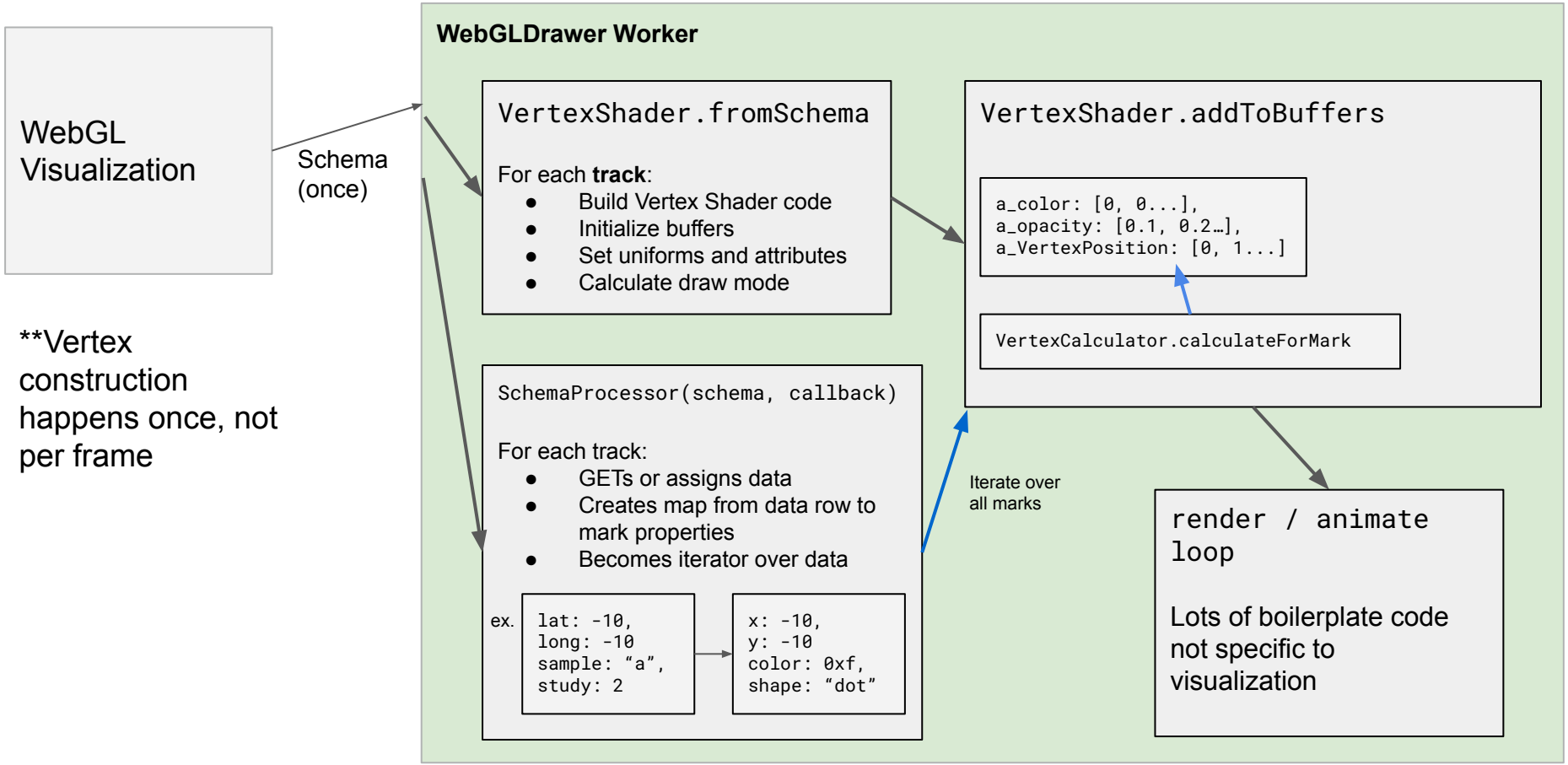
GPU

- Specialized computing
- Runs very few programs at time
- Limited API
- Built with graphics as priority

High-Level Architecture



Declaration to Visualization Pipeline

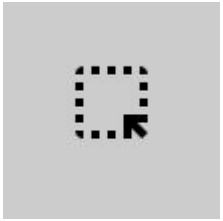


**Vertex construction happens once, not per frame

Current Tools

`MouseReader`

Panning



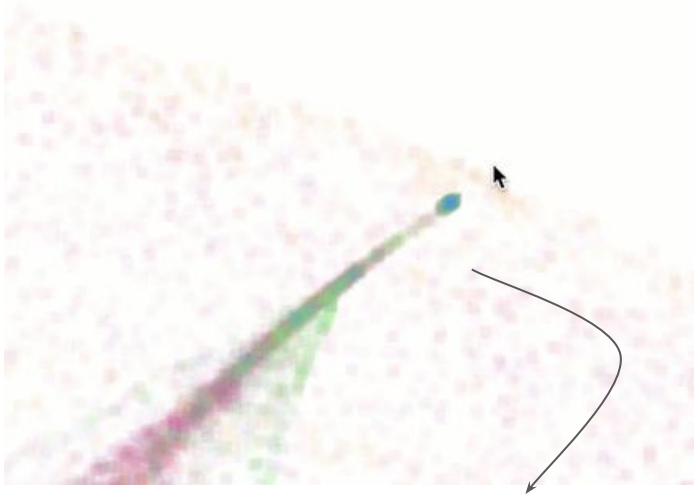
`DataProcessor`
`Web Worker`

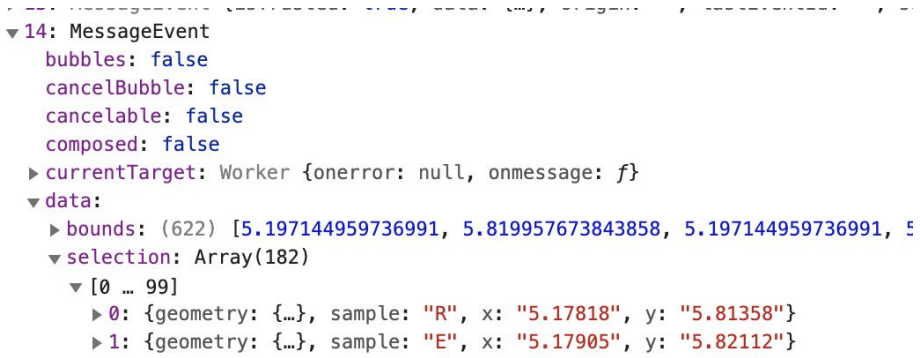
Box select

- Supercluster

Lasso select

- Supercluster
- simplify.js
- turf.js

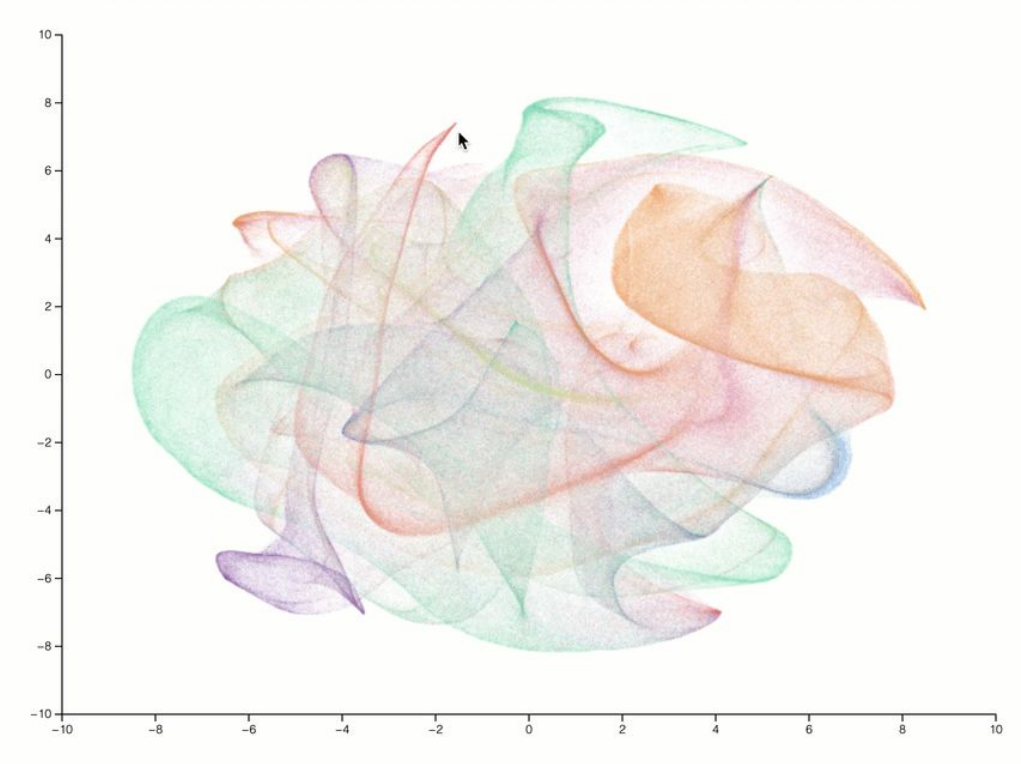




```
{
  "xAxis": "center",
  "yAxis": "center",
  "defaultData":
"http://localhost:1234/tsne.a6dfdcb6.csv"
,
  "tracks": [
    {
      "mark": "point",
      "x": {
        "attribute": "x",
        "type": "quantitative",
        "domain": [-10,10],
      },
      "y": {
        "attribute": "y",
        "type": "quantitative",
        "domain": [-10,10],
      },
      "color": {
        "attribute": "sample",
        "type": "categorical",
        "cardinality": 32,
        "colorScheme":
"interpolateRainbow"
      },
      "opacity": {
        "value": 0.05
      }
    }
  ]
}
```

Scatter Plot

~30 fps zoomed out, 60 fps otherwise

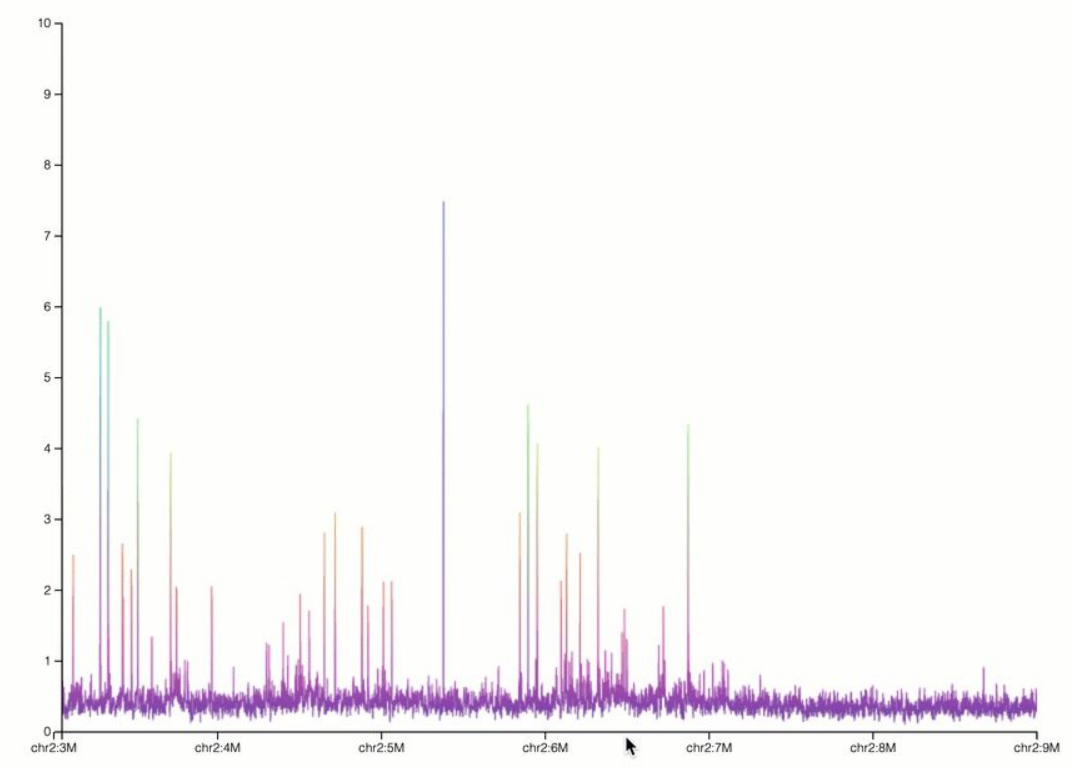


~1.5 million points

Line Track

60 fps

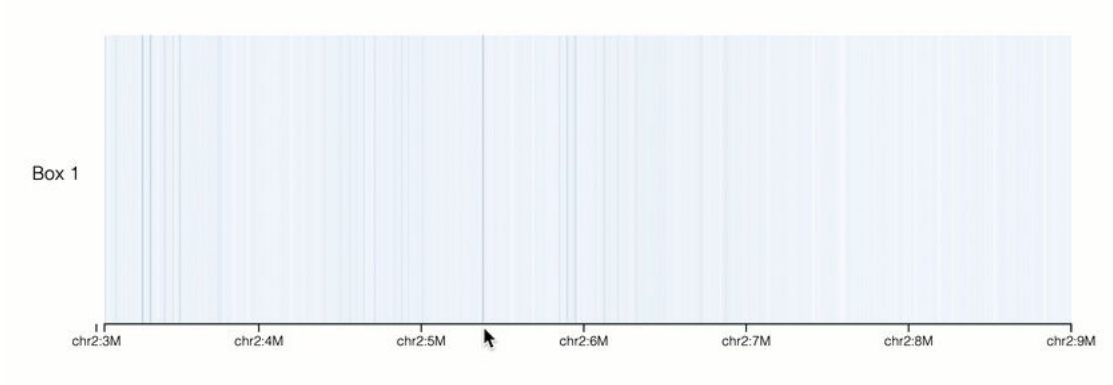
```
{
  "defaultData":
"http://localhost:1234/box-track.4acbc46c.csv"
,
  "tracks": [
    {
      "mark": "line",
      "x": {
        "type": "genomic",
        "chrAttribute": "chr",
        "geneAttribute": "start",
        "domain": [
          "chr2:3049800",
          "chr2:9001000"
        ],
        "genome": "hg38"
      },
      "y": {
        "type": "quantitative",
        "attribute": "score",
        "domain": [0, 10]
      },
      "color": {
        "type": "quantitative",
        "attribute": "score",
        "domain": [0, 8],
        "colorScheme": "interpolateBlues"
      }
    }
  ]
}
```



Box Track

60 fps

```
{
  "margins": {"left": "10%"},
  "labels": [
    {
      "y": 0.05,
      "x": -1.1,
      "text": "Box 1",
      "fixedX": true
    }
  ],
  "xAxis": "zero",
  "yAxis": "none",
  "defaultData":
"http://localhost:1234/box-track.4acbc46c.csv",
  "tracks": [
    {
      "mark": "rect",
      "x": {
        "type": "genomicRange",
        "chrAttribute": "chr",
        "startAttribute": "start",
        "endAttribute": "end",
        "domain": [
          "chr2:3049800",
          "chr2:9001000"
        ],
        "genome": "hg38"
      },
      "y": {
        "value": 0
      },
      "height": {
        "value": 10
      },
      "color": {
        "type": "quantitative",
        "attribute": "score",
        "domain": [0, 8],
        "colorScheme": "interpolateBlues"
      }
    }
  ]
}
```

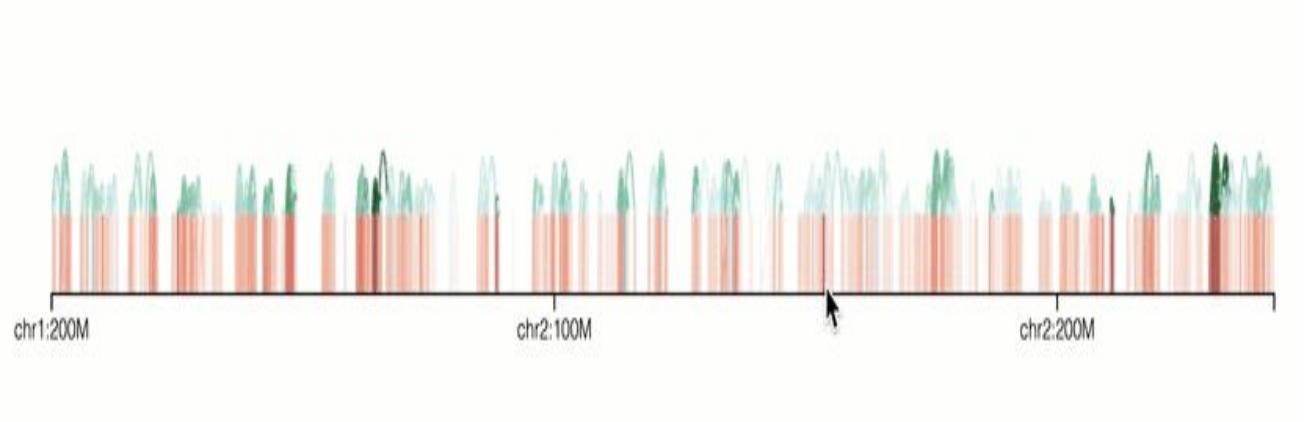


Arc Track

```
{
  "mark": "arc",
  "x": {
    "type": "genomicRange",
    "chrAttribute": "region1Chrom",
    "startAttribute": "region1Start",
    "endAttribute": "regionEnd",
    "domain": [
      "chr2:38000",
      "chr2:243149000"
    ],
    "genome": "hg19"
  },
  "width": {
    "type": "genomicRange",
    "chrAttribute": "region2Chrom",
    "startAttribute": "region2Start",
    "endAttribute": "region2End",
    "domain": [
      "chr2:38000",
      "chr2:243149000"
    ],
    "genome": "hg19"
  },
  "y": {
    "value": 0.1
  },
  "height": {
    "value": 0
  },
  "color": {
    "type": "quantitative",
    "attribute": "value",
    "domain": [0, 60],
    "colorScheme": "interpolateBuGn"
  }
}
```

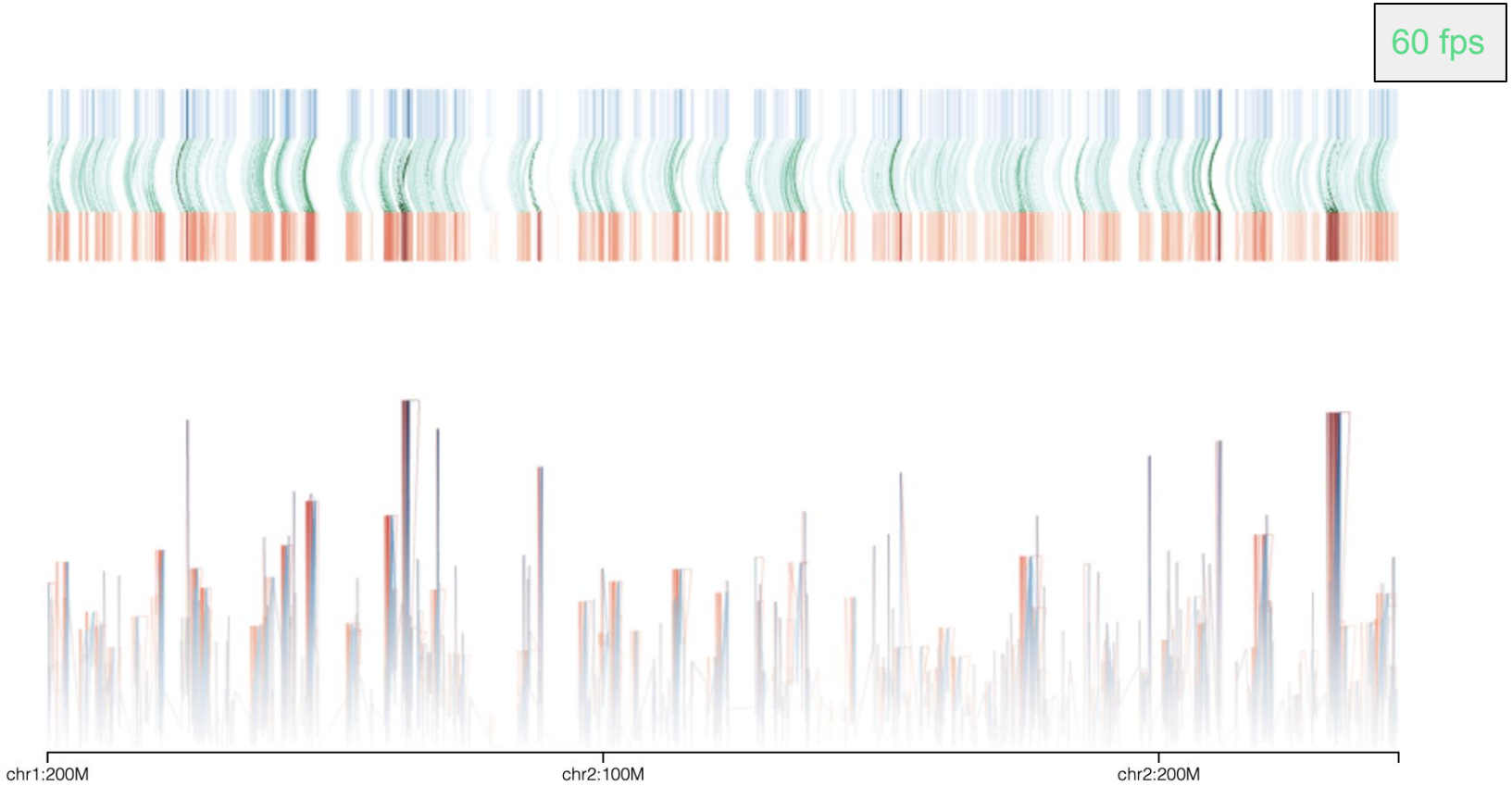
60 fps

~242 million base pairs in the 2nd chromosome

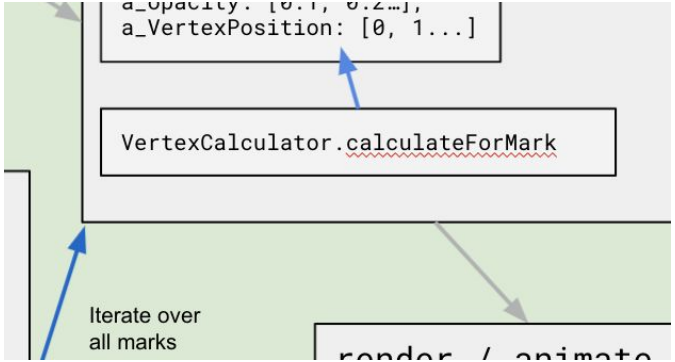


~400 base pairs in this region

Put them all together!



Future Features



WASM to speedup vertex generation

```
"x": {
  "attribute" : "x",
  "type": "quantitative"
  "domain": [0, 100000]
  "map": "log(x) ** 2",
  "filters": ["x < 10"]
}
```

Inline declaration functions

- Tooltips
- Legends
- Deployment as component
- Semantic Zooming
- Increasing chrome memory?
- Whatever your heart desires! (almost)
- Make a GitHub issue!