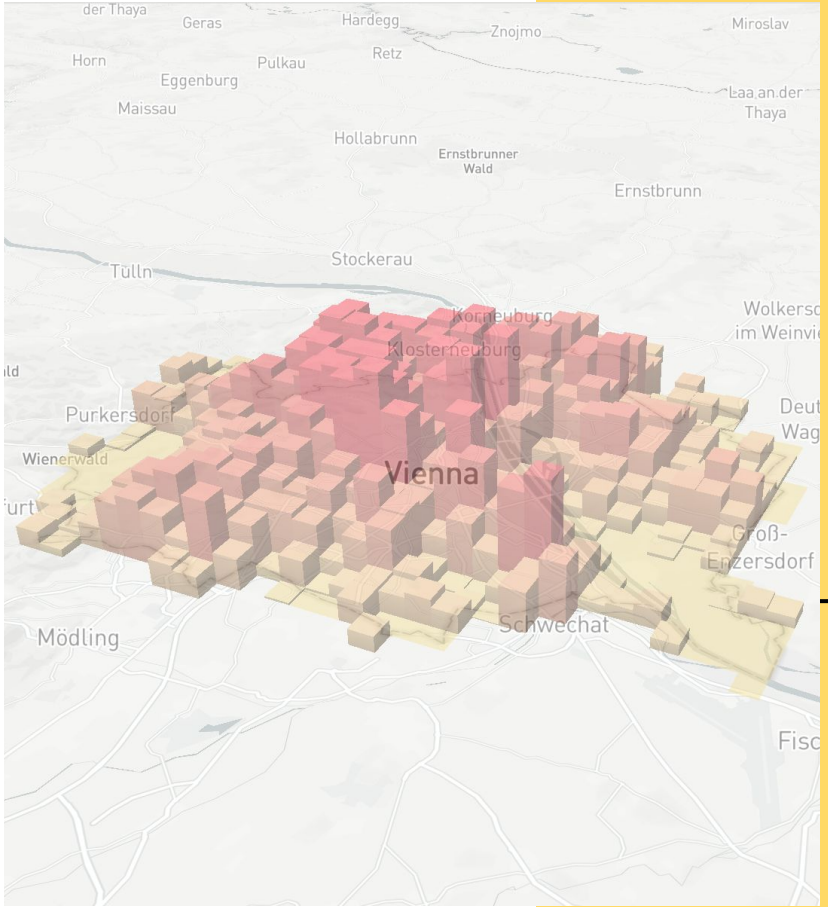


Developing a DSL for visualizing geospatial data

Christopher Stelzmüller
Sebastian Tanzer

Präsentations- und Arbeitstechnik, 343.306
Johannes Kepler Universität Linz

TABLE OF CONTENTS

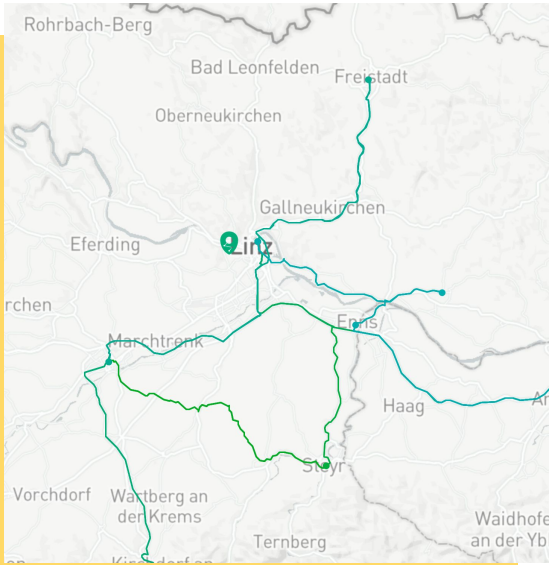
01 Motivation

02 Current Situation

03 Alternative? DSL!

04 Conclusion

MOTIVATION - why do we need Visualisations?



01

Intuitivity

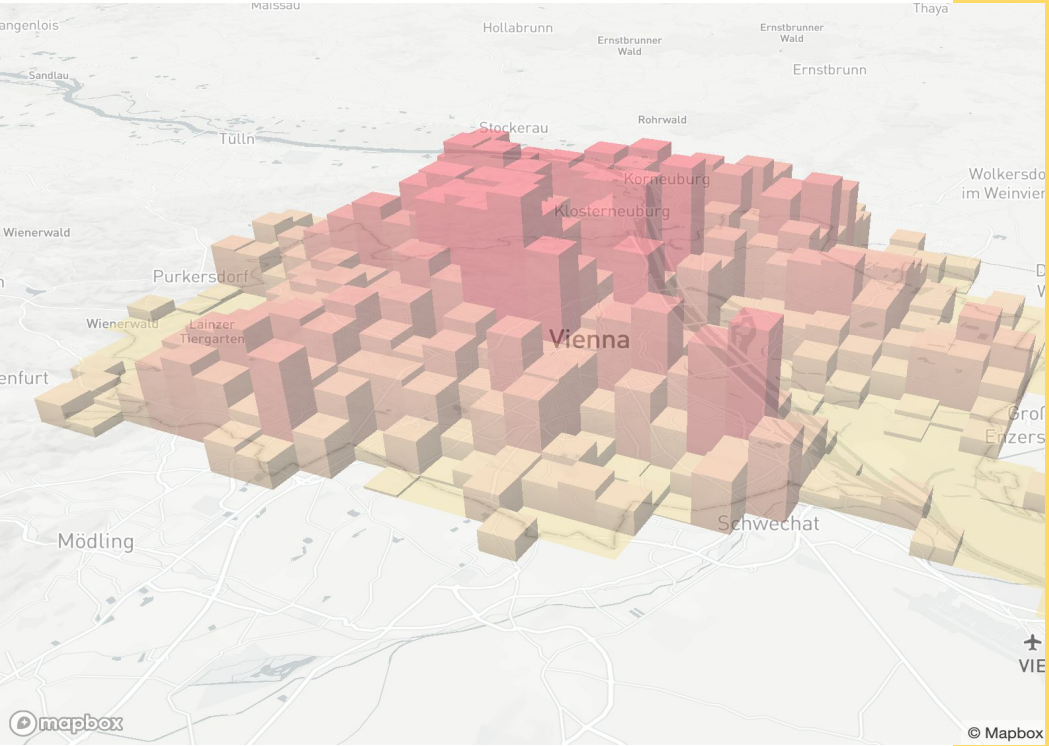
Visualizing geospatial data can make complex topics and data easier to understand

02

Problems with existing solution

Creating visualizations is often repetitive and error-prone

AN EXAMPLE



DSL for visualizing geospatial data

Visualizing Vienna's Population distribution

CURRENT SITUATION


```
{
  "id": "population",
  "type": "fill-extrusion",
  "source": {
    "type": "geojson",
    "data": "./data/vienna_population.geojson"
  },
  "paint": {
    "fill-extrusion-color": [
      "interpolate",
      ["linear"],
      ["*", 40, ["sqrt", ["get", "tot_p"]]],
      0,
      "#ecda9a",
      5600,
      "#ee4d5a"
    ],
    "fill-extrusion-height": [
      "*",
      40,
      ["sqrt", ["get", "tot_p"]]
    ],
    "fill-extrusion-opacity": 0.5
  }
}
```

- Lack of static validation
- No IDE-support
- Repetitive
- Error-prone

DSL for visualizing geospatial data

ALTERNATIVE: A DSL

Improved Productivity

Readability & Communication

Editor Support & Validation

Reusability

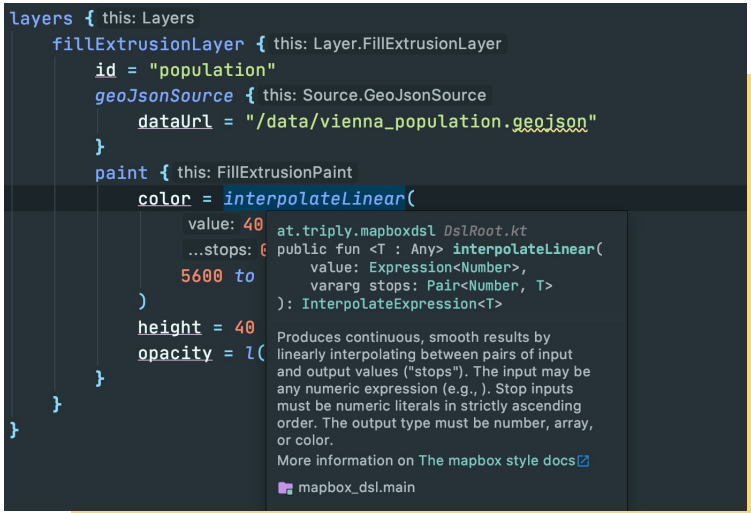
Our DSL Implementation

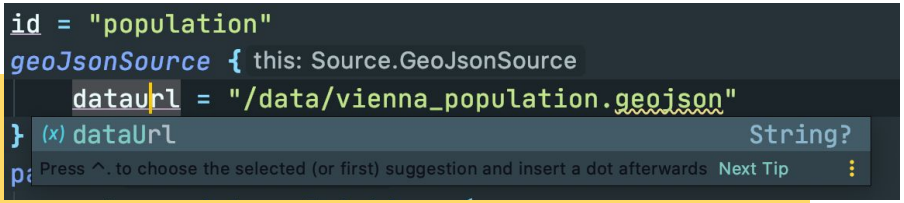

```
fillExtrusionLayer {
    id = "population"
    geoJsonSource {
        dataUrl = "/data/vienna_routes.geojson"
    }
    paint {
        color = interpolateLinear(
            40 * sqrt(get("tot_p")),
            0 to "#ecda9a",
            5600 to "#ee4d5a"
        )
        height = 40 * sqrt(get("tot_p"))
        opacity = l(.5)
    }
}
```

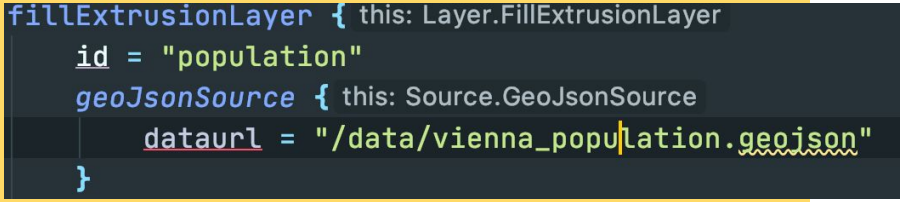
- Embedded DSL
- Written in Kotlin
- Similar Syntax to Java & C#
- Can be used in JVM-backend & Web frontend applications

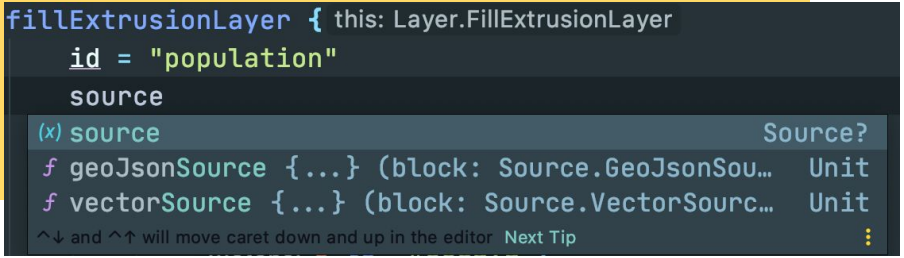
DSL for visualizing geospatial data

Advantages of the DSL









DSL for visualizing geospatial data

COMPARISON - before & after


```
{
  "id": "population",
  "type": "fill-extrusion",
  "source": {
    "type": "geojson",
    "data": "./data/vienna_population.geojson"
  },
  "paint": {
    "fill-extrusion-color": [
      "interpolate",
      ["linear"],
      ["*", 40, ["sqrt", ["get", "tot_p"]]],
      0,
      "#ecda9a",
      5600,
      "#ee4d5a"
    ],
    "fill-extrusion-height": [
      "*",
      40,
      ["sqrt", ["get", "tot_p"]]
    ],
    "fill-extrusion-opacity": 0.5
  }
}
```

```
fillExtrusionLayer {
    id = "population"
    geoJsonSource {
        dataUrl = "/data/vienna_routes.geojson"
    }
    paint {
        color = interpolateLinear(
            40 * sqrt(get("tot_p")),
            0 to "#ecda9a",
            5600 to "#ee4d5a"
        )
        height = 40 * sqrt(get("tot_p"))
        opacity = l(.5)
    }
}
```

DSL for visualizing geospatial data

CONCLUSION

- First draft of DSL implemented
- Usable for creating web-visualizations
- Future work:
  - Implement all functionality in mapbox style language
  - Enable usage in Android application

THANKS!

Do you have any questions?

github.com/tuesd4y/mapbox-dsl