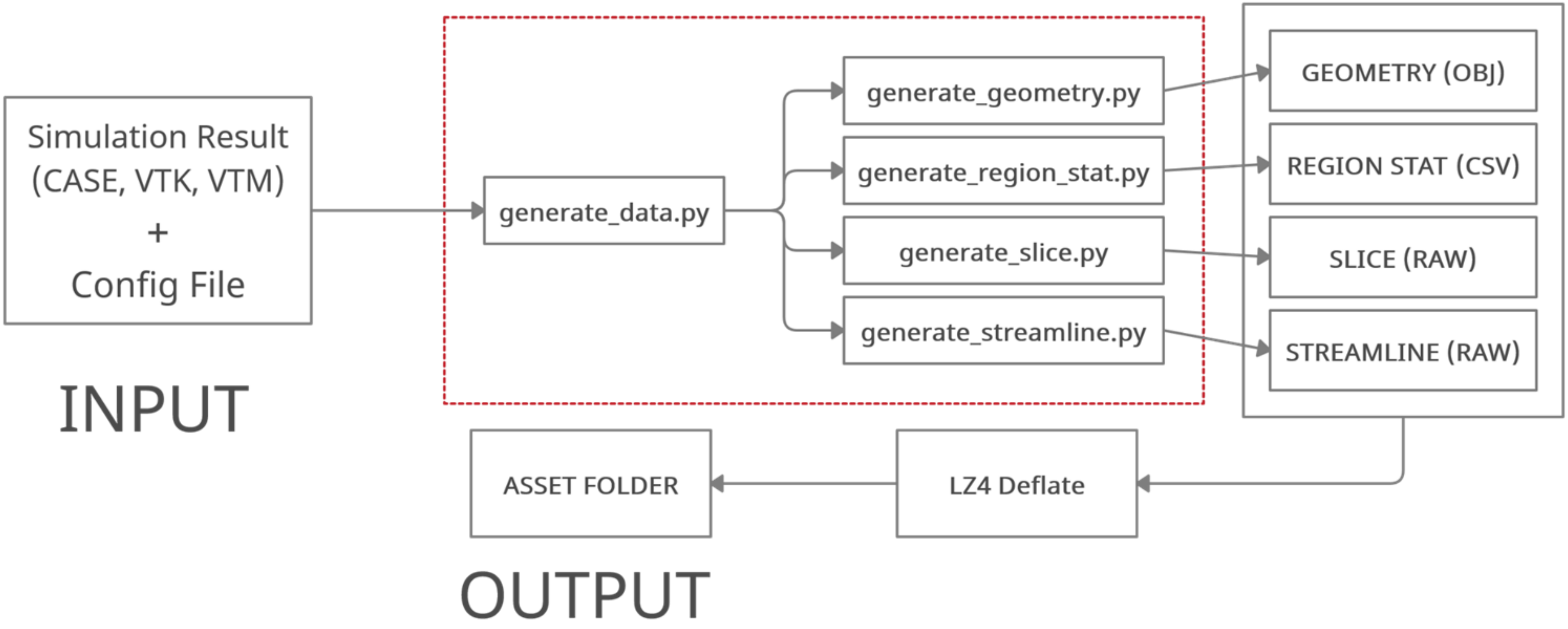
Simulation Visualization & Data Processing

Matyas Constans SZE

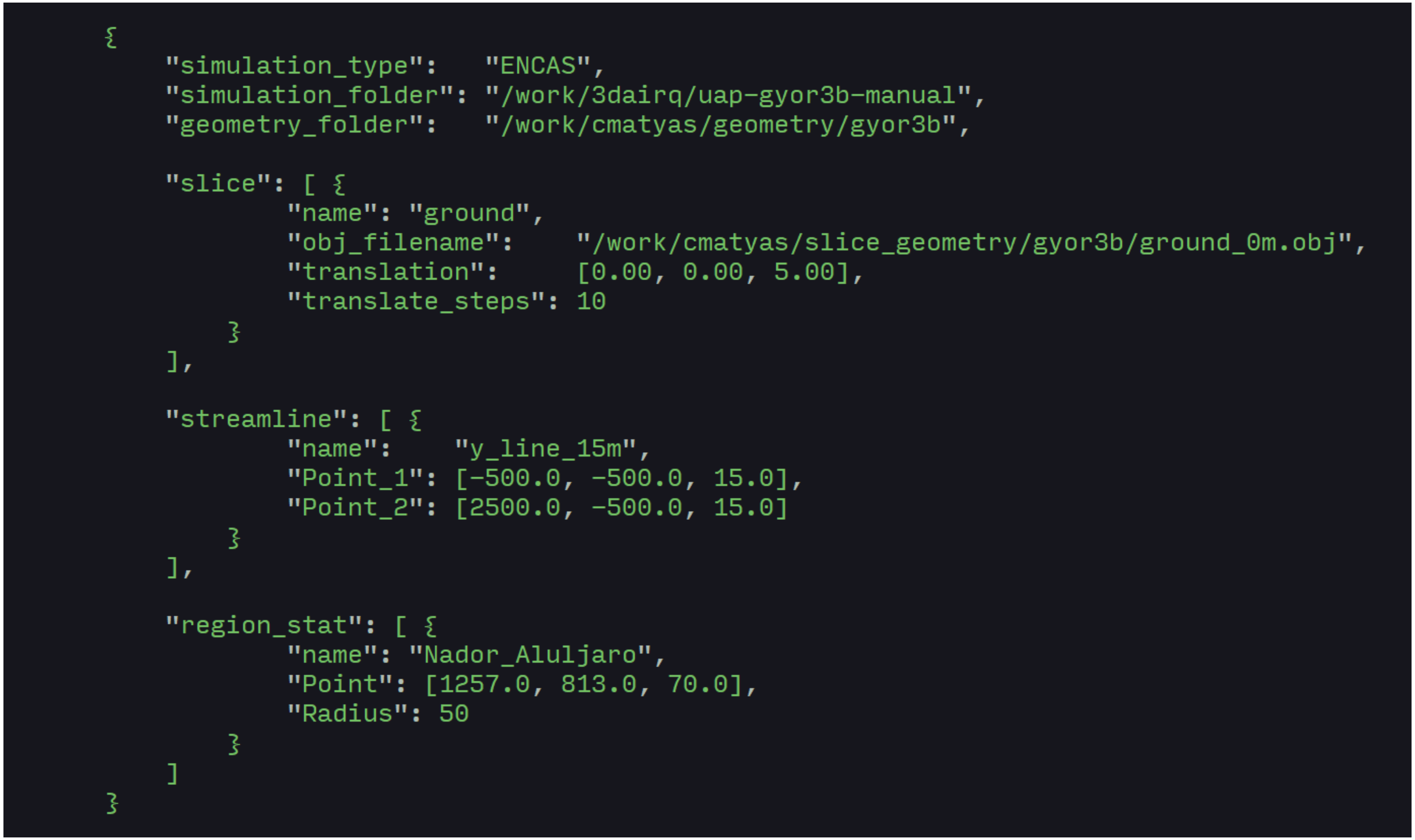
Files available at:

```
/work/demo-users/workshop-files/stat
```

Post-Processing Pipeline Overview

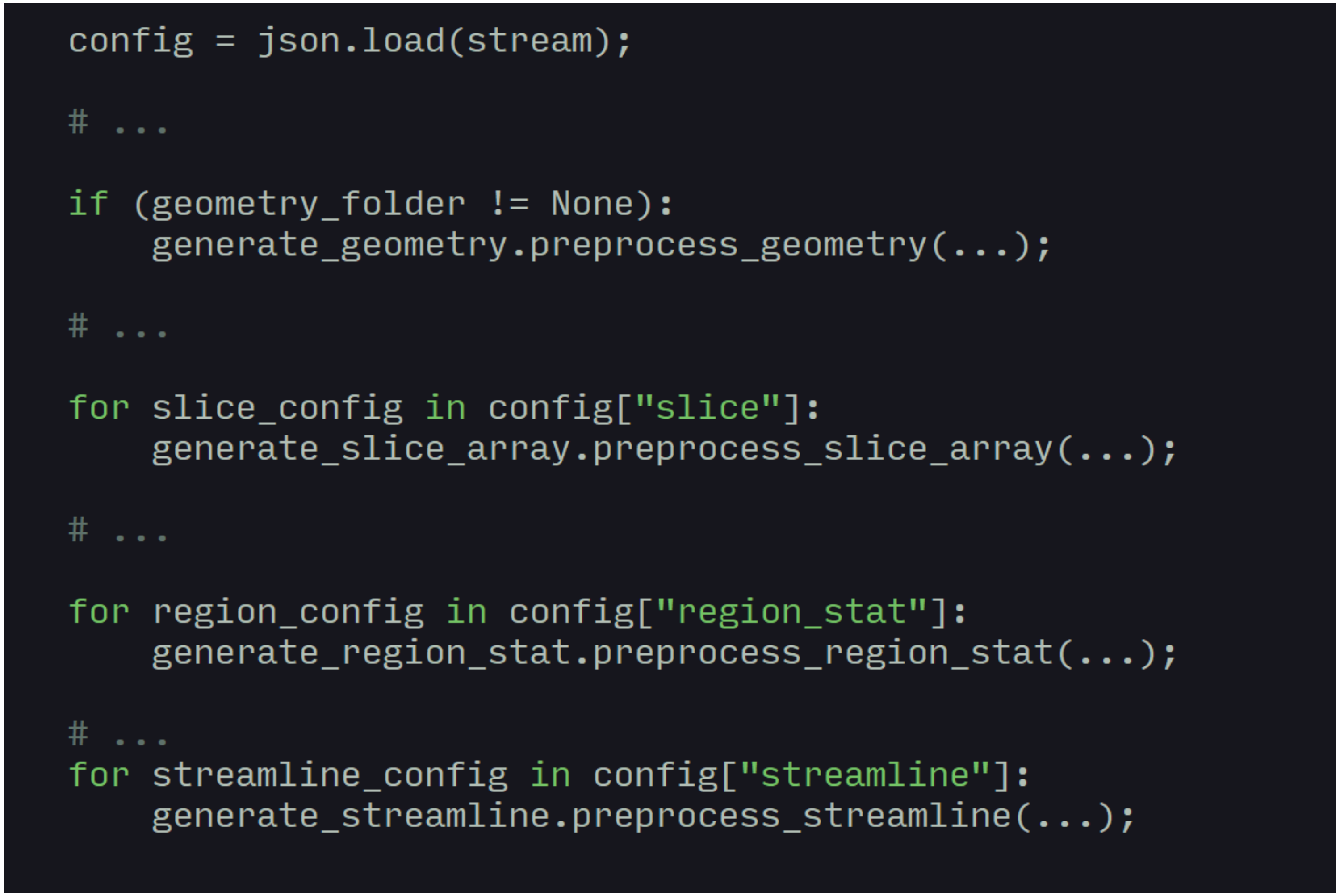


Config File Example



```
{
    "simulation_type":   "ENCAS",
    "simulation_folder": "/work/3dairq/uap-gyor3b-manual",
    "geometry_folder":   "/work/cmatyas/geometry/gyor3b",

    "slice": [ {
            "name": "ground",
            "obj_filename":    "/work/cmatyas/slice_geometry/gyor3b/ground_0m.obj",
            "translation":     [0.00, 0.00, 5.00],
            "translate_steps": 10
        }
    ],

    "streamline": [ {
            "name":    "y_line_15m",
            "Point_1": [-500.0, -500.0, 15.0],
            "Point_2": [2500.0, -500.0, 15.0]
        }
    ],

    "region_stat": [ {
            "name": "Nador_Aluljaro",
            "Point": [1257.0, 813.0, 70.0],
            "Radius": 50
        }
    ]
}
```

Script Overview

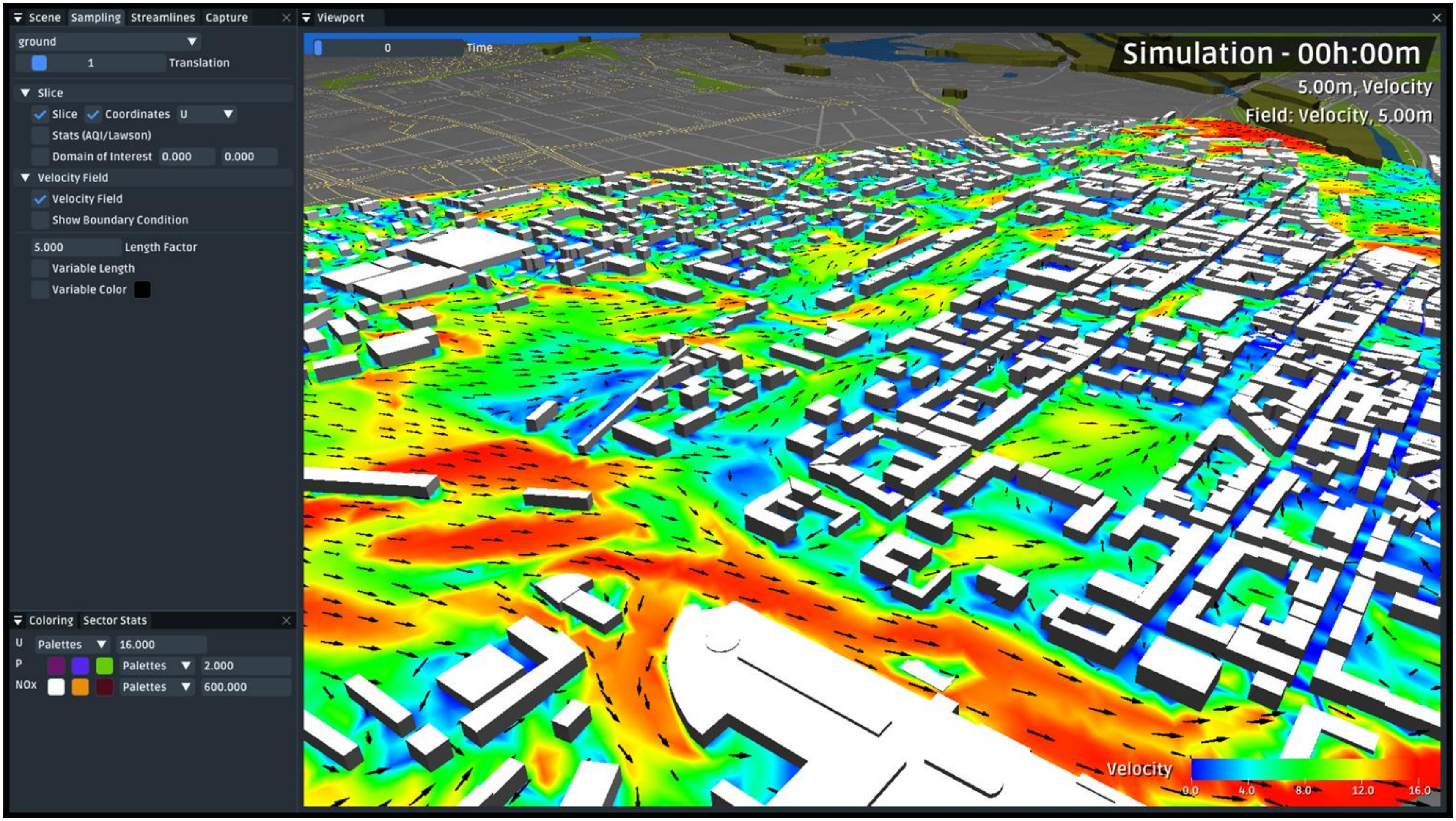


```
config = json.load(stream);

# ...

if (geometry_folder != None):
    generate_geometry.preprocess_geometry(...);

# ...

for slice_config in config["slice"]:
    generate_slice_array.preprocess_slice_array(...);

# ...

for region_config in config["region_stat"]:
    generate_region_stat.preprocess_region_stat(...);

# ...
for streamline_config in config["streamline"]:
    generate_streamline.preprocess_streamline(...);
```

Web Renderer Overview

- Written in idiomatic C99, compiled as C++
- Compiled with Emscripten
- Rendered with SDL2 / OpenGL ES 2.0
- Only SDL2 and IMGUI as external libraries
- Same look/feel as a desktop app, although not as cluttered

Web Renderer Overview



ground

1 Translation

▼ Slice

☑ Slice ☑ Coordinates U

□ Stats (AQI/Lawson)

□ Domain of Interest 0.000 0.000

▼ Velocity Field

☑ Velocity Field

□ Show Boundary Condition

5.000 Length Factor

□ Variable Length

□ Variable Color ■

0 Time

Coloring Sector Stats

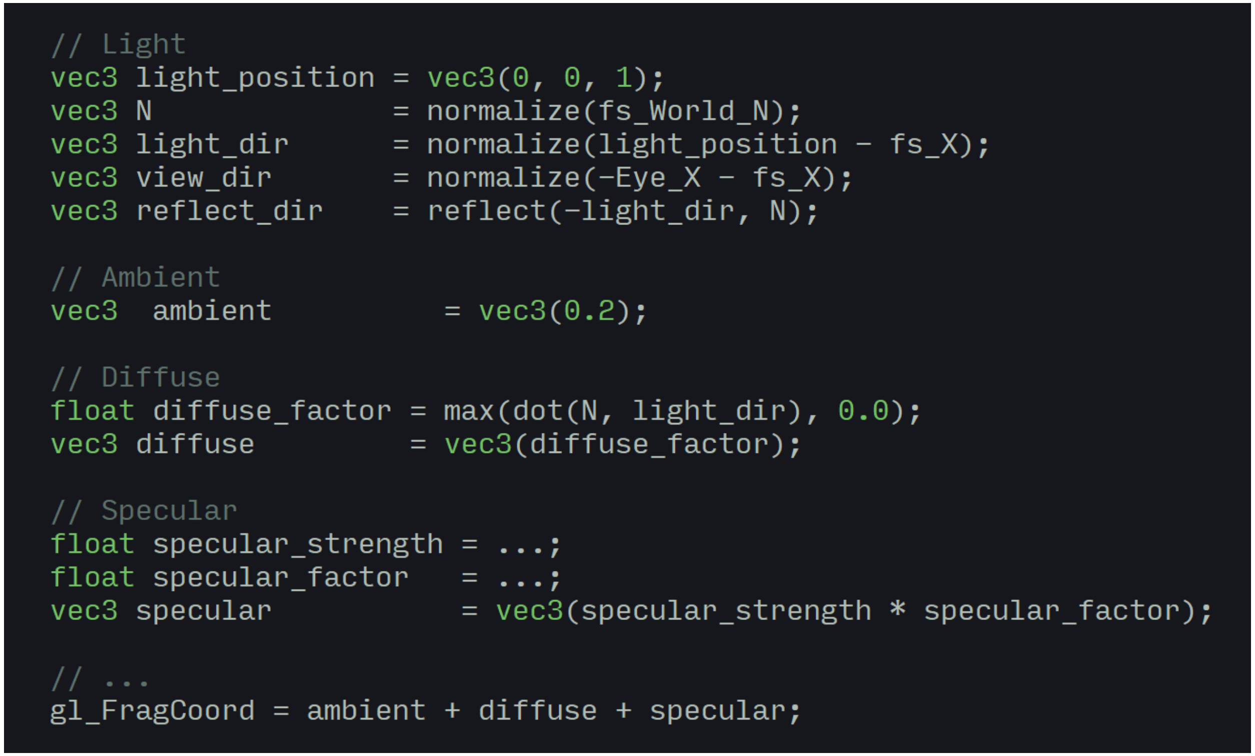
U Palettes 16.000

P 🟪 🟪 🟩 Palettes 2.000

NOx ⬜ 🟧 🟥 Palettes 600.000

Rendering

- Rendering is fairly basic (since ES 2.0 is pretty far behind).
- PHONG model (Diffuse + Specular + Ambient).

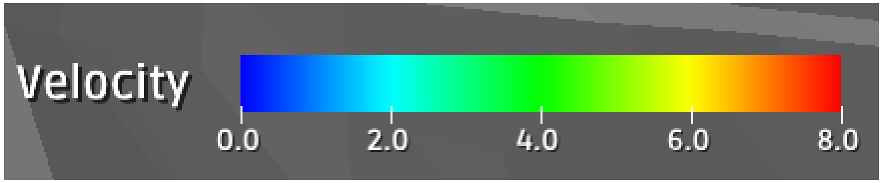


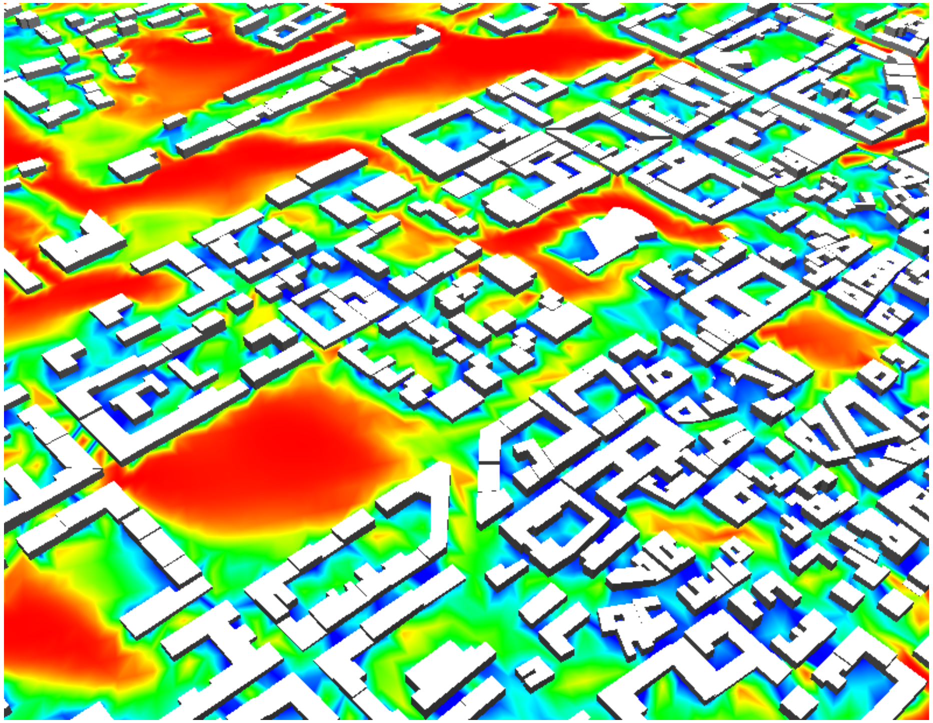
```
// Light
vec3 light_position = vec3(0, 0, 1);
vec3 N              = normalize(fs_World_N);
vec3 light_dir      = normalize(light_position - fs_X);
vec3 view_dir       = normalize(-Eye_X - fs_X);
vec3 reflect_dir    = reflect(-light_dir, N);

// Ambient
vec3  ambient          = vec3(0.2);

// Diffuse
float diffuse_factor = max(dot(N, light_dir), 0.0);
vec3 diffuse         = vec3(diffuse_factor);

// Specular
float specular_strength = ...;
float specular_factor   = ...;
vec3 specular           = vec3(specular_strength * specular_factor);

// ...
gl_FragCoord = ambient + diffuse + specular;
```

Rendering - Rendering Color Ranges

- When Rendering Color ranges (HSV Rainbow / Color Ranges, with interpolation between 3-4-5 color values), we use a custom shader.

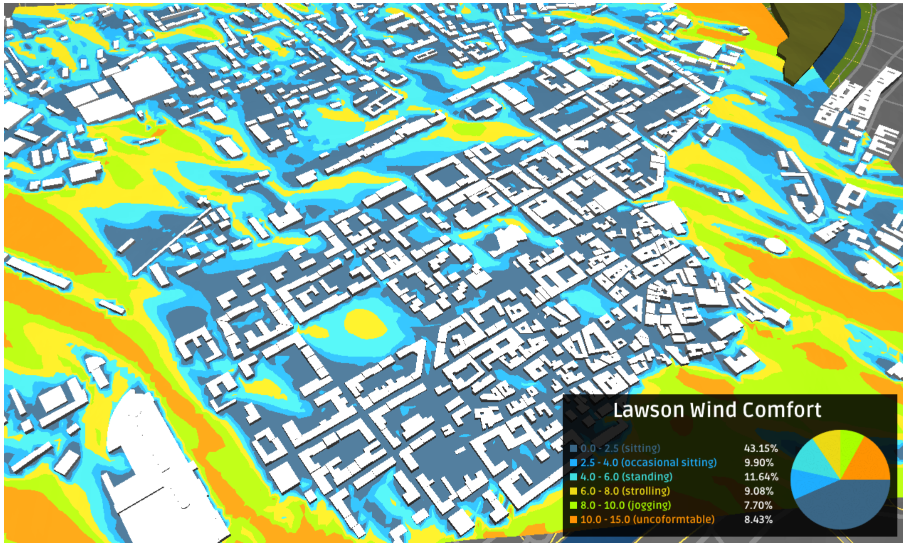
- We just pass that info down the GL pipeline, as an attribute buffer.





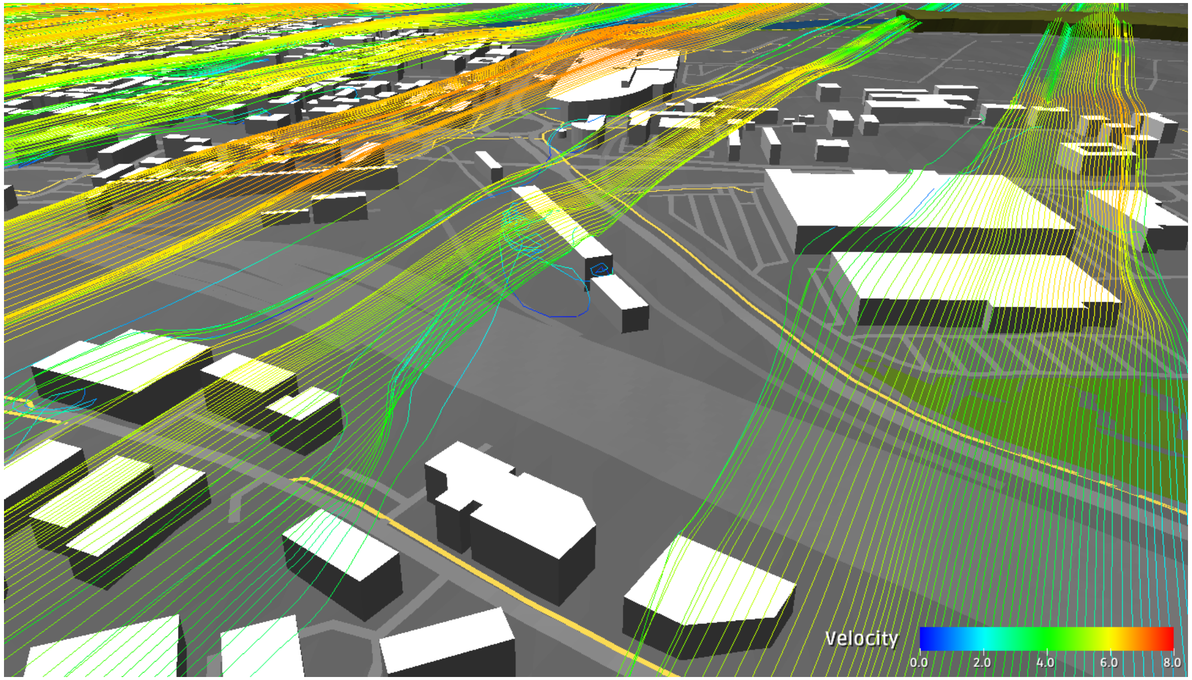
Computing Statistics on-the-fly

- Stats are computed on-the-fly (AQI/LAWSON)
- Take the triangulated mesh, and subdivide until we’re happy with the refinement level. We then discard based on the vertex values, and sum-up the areas on multiple threads. (single-threaded is enough most of the time)


Lawson Wind Comfort
0.0 - 2.5 (sitting) 43.15%
2.5 - 4.0 (occasional sitting) 9.90%
4.0 - 6.0 (standing) 11.64%
6.0 - 8.0 (strolling) 9.08%
8.0 - 10.0 (jogging) 7.70%
10.0 - 15.0 (uncoformtable) 8.43%

Streamline Construction / Rendering

- Streamlines are reconstructed, from point cloud, based on distance and inner product magnitude.
- Dynamic Window Flow, is rendered as a particle system. Simply with GL_POINT.



Hands-on session

1. **Generate Dataset, with simulation (FOAM)**

1. **Understanding pvpython and automated stat gen.**

```
command: ./region_stat_extractor.py
```

1. **Analyzing the .CSV files.**

1. **Visualization as probes (web_renderer)**