Interactive Visual Data Analysis

Assignment 6 – *Vector Fields and Flow Metrics*

In this assignment we will begin working on flow visualization. For a start, we will employ our existing volume rendering capabilities to visualize simple scalar flow metrics like velocity or vorticity magnitude.

6.1. Loading Vector Fields

Extend your data loader to handle 3- and 4-component flow fields (data type half or float).

- Make sure to respect the `SliceThickness` and `Timestep` fields.
- Create only 4-component textures on the GPU, i.e. `DXGI_FORMAT_R16G16B16A16_FLOAT` / `DXGI_FORMAT_R32G32B32A32_FLOAT`. Pad the data with zeros if only 3 components are stored in the .raw file.
- Just like for volume data: support single vector fields and sequences of vector fields. When sampling the vector data, use tri-linear interpolation (→*SamplerState*) and an additional linear interpolation between timesteps.
- If there is a `MeshFileName` entry, load and display the corresponding `.ply` file. The meshes are given in data world space `[0, Resolution*SliceThickness]`.

Note: All the interesting vector sequences are located in the additional repository folder `pra1314/bigdata` due to their size.

6.2. Metric Computation

Write a compute shader that computes a metric scalar field of equal resolution from the current vector field by evaluating a user-selectable flow metric.

- When metric rendering is enabled, the field should be recomputed every frame. As usual, interpolate between time-steps of the vector field in case of sequences.
- Implement velocity magnitude, divergence and vorticity magnitude. Additionally, implement at least one vortex core criterion, e.g. Q or λ_2 (see slides).
- Use central differences to approximate all first-order derivatives. At the boundary, use forward/backward differences (for some metrics this might not work very well and you should prefer simply setting the metric values at the boundary to zero).
- In general, metrics do not necessarily evaluate to values in the range $[0; 1]$. Provide some GUI elements to let the user choose which range of values he's able to see. Also, provide the possibility to flip the sign of the computed metric (=reverse range of values).

6.3. Metric Volume Rendering

Render the metric scalar field in addition to all existing, opaque objects (as of now, this includes bounding boxes and meshes, but soon we will have more “stuff flying around” like e.g. particles). You should support DVR and iso-surfaces.

6.4. Visualize!

The same procedure as every week! Commit some nice screenshots in PNG format to a folder called `screenshots/assignment6/` outside of your solution directory.

The working solution must be committed until **November 27, 09:00am**. If anything is not working as described here or if you want a specific SVN-Revision to be rated, explain yourself in the `readme.txt` file within your `solution` directory.