Interactive Visual Data Analysis

Assignment 11 – *Smoke Surfaces*

In this final assignment you have to read and implement a flow visualization paper, which can be found here (`http://www.mpi-inf.mpg.de/~weinkauf/publications/absvonfunck08a.html`). In order to visualize streak surfaces interactively, the paper introduces a smoke metaphor to avoid the need for adapting the surface mesh during advection. We leave the theory to the paper, and in the following will only discuss implementation details.

Note: You do not need to implement the "Enhancements and Modifications" mentioned in Section 4 of the paper.

11.1. Seeding and Advection

Generate a streak surface consisting of $n \cdot m$ vertices: Seed n streak lines of length m from equidistant points on a straight line. To define the seeding line you can simply use your box probe if you want. E.g., you can seed from the longest central axis of the box (the restriction to axis-aligned seeding lines is ok).

Optional: Implement a line probe, i.e. a line whose start and end point can be moved separately by the user through mouse input.

Optional: Implement smoke time surfaces (section 4.2 of the paper). You can seed them e.g. from the faces of your box probe or a custom plane probe.

11.2. Rendering

Compute alpha values for each streak surface vertex in a separate compute shader which is called every time the streak surface has changed. Implement the following criteria: $\alpha_{density}$, α_{shape} and α_{fade} as described in the paper. Make the individual criteria toggle-able in your UI.

Optional: Implement $\alpha_{curvature}$ (requires the calculation of vertex normals in a separate compute shader).

Hint: On the surface vertices you will need to accumulate alpha values calculated on a per-triangle basis (find minimum value). Use one CS thread per vertex and for every vertex loop over all adjacent triangles and evaluate the alpha for each of those triangles.
Note: In total this process will evaluate the alpha values thrice for every triangle (each triangle is adjacent to exactly three vertices). The alternative is to use one CS thread per triangle and atomic operations (`InterlockedXYZ()` functions in HLSL) to access adjacent vertices. This may be less efficient—you should avoid atomic operations where possible.

To avoid the need for depth correct blending, use a constant smoke color and only vary the opacity of the fragments corresponding to the smoke. Render all opaque objects of the scene first, then deactivate depth write (but keep depth test enabled) and render the smoke surfaces using back-to-front alpha blending. You can extrude the surface triangles in a geometry shader, invoking either one GS for every triangle or one GS for every quad of the surface (two neighboring triangles).

It should be possible to render smoke surfaces and other opaque objects like particles or stream lines at the same time. You do **not** need to support simultaneous rendering of smoke surfaces and other transparent stuff (e.g. metric DVR).

Spoiler: We'll tackle varying smoke colors next week (order independent transparency).

11.3. Visualize

Like every week, take a few screenshots of some pretty renderings generated by your tool and commit them to a folder called `screenshots/assignment11/` outside of your solution directory.

The working solution must be committed until **January 22, 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.