Interactive Visual Data Analysis

Assignment 10 – *Streak Lines and Tube Rendering*

In this assignment we will add streak lines to our tool and improve the visual appearance of our stream/streak lines by rendering them as three-dimensional tubes.

10.1. Streak Lines

Implement streak lines. Like for stream lines, the starting positions should be distributed randomly within a probe. The user should be able to turn streak lines on/off for each probe and to select the number and length of the streak lines, i.e. two parameters out of the following three: lifetime T, seeding interval τ and vertex count m (see slides).

- Allocate a buffer to hold all the line vertices and write a compute shader that updates this buffer **every frame** by advecting and seeding particles (one thread per particle).
- Your seeding strategy has to ensure that streak line particles are seeded in fixed time intervals τ based on the *elapsedTime* parameter such that the visualization (the resulting shape of the streak lines) is independent of the frame rate and sequence play speed. Each line always has to start at the seeding point, whereas the end of the line is allowed to oscillate as a result of particle removal.
- Modify/extend the proposed seeding strategy from the slides as needed. **Caution:** You cannot avoid moving the *separator* by moving the particles through the buffer instead–this will lead to race conditions between different compute shader threads accessing the buffer.
- Ensure that τ/m can be chosen large/small enough such that we can see that your seeding strategy works. Another good idea would be a "Reseed Streak Lines" button.
- Parts of streak lines that leave the domain through the boundary should not be visible.

Add a render mode that uses simple line primitives to visualize your streak lines!

10.2. Tube Rendering

Add another render mode for both stream and streak lines that renders them as tubes of a fixed (user-defined) radius. You can either build the tubes from analytic spheres and cylinders, or you can use the geometry shader to extrude each line segment to a tube (with a fixed number of segments, e.g. 8). If you use the geometry shader approach, you have to make sure that there are no gaps at the points where two line segments meet, and that the tubes are not rotated (which would make them thinner in the middle, see slides). To achieve that, you'll need to track a per-vertex normal vector which rotates as little as possible along the line. For stream lines, it's possible to do this in the main advection pass, but for streak lines you'll need a separate pass!

Optional: Adapt the tube radius locally at each vertex, so that the volume of each tube segment stays approximately constant. If you used the analytic approach above, take note that the cylinders now become cones! However, for the purpose of calculating the volume, you can still assume that they're cylinders.

The working solution must be committed until **January 15, 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.