Interactive Visual Data Analysis

Assignment 5 – *Time-dependent Data*

In this assignment, we will render time-dependent volume data sets, and finish up the volume rendering part of this course.

5.1. Time-dependent data

Time-dependent volume data sets are given as sequences of time steps. Each time step is similar to a volume file that you handled so far. A sequence is defined by a single `.dat` file, and one `.raw` file per time step. In the `.dat` file, the `ObjectFileName` contains a `printf`-style tag (e.g. `%03i` for a 3-digit integer with leading zeros). [Hint: Use `sprintf_s`.] In addition, there is an additional `ObjectIndices: <min> <max> <step>` entry, which specifies the valid time step numbers (e.g. `ObjectIndices: 0 12 4` would specify the numbers 0, 4, 8 and 12). The `<step>` entry can also be omitted, in which case the step is 1. There may also be a `Timestep: <float>` entry, which specifies the distance between time steps in seconds.

- Adjust your `.dat` parser to handle the extended format.
- When loading a volume data set, load all specified time steps. You may assume that all data will fit into CPU memory.

For rendering, we will need two time steps in GPU memory for interpolation – the one immediately before and the one immediately after the current time.

- Create two 3D textures instead of one.
- Add GUI elements to control the time (e.g. play/stop, speed, current time). While "play" is enabled, update the current time in each frame.
- Before rendering, update the textures as required using `UpdateSubresource`. This is fairly expensive, so update only what is actually necessary – e.g. during playback, you will usually need to update at most one of the textures in each frame.
- In your shaders, sample from both textures and interpolate between the two values based on the current time.

5.2. Cleanup

This is the last assignment on (scalar) volume rendering – next week, we will start with flow visualization. So now is a good time to clean up your volume rendering tool, fix or add missing features, build nicer GUI controls etc. In particular, you should implement the following things if you haven't done so already:

- Make sure your tool also works correctly with non-cubic volumes, i.e. where the data size is not the same in all dimensions. For example, this is the case in the "VisHuman Foot" data set.
- Adapt your ray casting code to handle depth correctly. For iso-surfaces, it is sufficient to write out the z-value of the intersection into `SV_Depth`. However, for DVR, you have to implement a "manual z-test": In the pixel shader, read the depth of the current pixel. (You might have to

create a copy of the z buffer for this!) Stop the ray marching when you hit this depth. Since we are now treating depth manually, disable the hardware depth test **and depth writes** for the DVR shader. Hint: For testing this feature, use the `C60Large_zTest.dat` "data set".

- Optional: Make it possible to save/load configuration files which contain all relevant options, such as view point, transfer function, light position, etc.

5.3. Optional: ClearView

Implement a ClearView rendering mode in your visualization tool, as described in the paper available here:

http://wwwcg.in.tum.de/research/research/publications/2006/clearview-an-interactive-context-preserving-hotspot-visualization-technique.html

Warning: This is major task – don't start with this until everything else is done, and expect to spend quite some time here.

5.4. Visualize!

Once again, time to create some nice visualizations! Like last week, take a few screenshots (PNG format) of some pretty renderings generated by your tool and commit them to a folder called `screenshots/assignment5/` outside of your solution directory. If your screenshots need further explanation, write a small readme and/or simply choose meaningful filenames.

The working solution must be committed till **November 20, 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.