Interactive Visual Data Analysis

Assignment 4 – *Direct Volume Rendering*

In this assignment we will implement *Direct Volume Rendering (DVR)*. Besides *Iso-surfaces*, it is the second of the two basic visualization techniques for scalar volumes.

4.1. Open File Dialog

Add an "open file" button to your GUI that lets the user select a .dat file for opening. Enable the appropriate visualization mode(s) of your application depending on the data type (*mesh* and/or *scalar volume* and soon also *vector volume*).

- Test the available data sets, and make sure that your scaling (of the bounding box, scalar field etc.) works correctly when the data set is not cubic.
- Repeatedly loading/closing different data sets in one program run should cause no crashes and produce no memory leaks.

Hint: You can use `GetOpenFileName()` from the Windows API.

4.2. DVR Pixel Shader

Add a *Direct Volume Rendering* mode to your tool. You can copy or re-use everything from your *Iso-surface* mode up to the point in the pixel shader where you have computed entry and exit point of the ray casting box.

In the remainder of the (DVR-) pixel shader: Step iteratively through the volume with a fixed step size h, starting at the entry point, and in every step do the following:

- Sample the volume → scalar value s.
- Apply the transfer function to get an RGBA value. For a start, apply a simple hardcoded transfer function like $s \rightarrow (1,1,1,s)$.
- Adjust the α-value to the step size h by multiplying it with h (and an additional user-defined scaling value). This makes the overall opacity in the final image (roughly) independent of h.
- Update the current accumulated RGBA value using the rule for front-to-back alpha blending.
- Terminate if you reach the exit point or if the accumulated α gets close to 1 (= early ray termination: everything behind won't have an effect on the final color), and return the accumulated value.

4.3. Alpha Blending

Since the DVR pixel shader returns transparent fragments, we need to do alpha blending in the output merger. Therefore you have to configure the blend state of the output merger for (back-to-front) alpha blending (see `D3D11_BLEND_DESC`).

- Be careful with pre-multiplied alpha: does your pixel shader return (C, α) or $(C\alpha, \alpha)$ as a result of the front-to-back alpha blending in **4.2**? Adjust the blend state accordingly.
- Adjust the rendering order of your scene: Draw all opaque objects first and then draw all transparent objects in back-to-front order. (In addition to the volume itself, you may have e.g. a transparent bounding box. You do not need to support multiple volumes at the same time.)

4.4. Transfer Functions

So far, so boring – we need more color and user interaction! Integrate the simple transfer function editor provided in `external/TransferFunctionEditor` into your tool. Feel free to copy the sources to your repository and modify them if the editor doesn't fit your needs.

The editor provides the transfer function in the form of a 1D RGBA texture.

- Bind this texture to your effect and replace the hardcoded transfer function from **4.2** by a 1D texture lookup: Use the scalar volume value $s \in [0;1]$ as a 1D texture coordinate to sample the transfer function texture (Hint: effect variable of type `Texture1D<float4>`).

4.5. Visualize!

Time to actually create some nice visualizations! Take a few screenshots (PNG format) of DVR renderings and commit them to a new, separate folder called `screenshots/assignment4/` outside of your solution directory. If your screenshots need further explanation, either write a small readme and/or simply choose meaningful filenames. The final folder structure should look like

```
.../yourlocaldir/external/
.../yourlocaldir/GROUP/solution/
.../yourlocaldir/GROUP/screenshots/assignment4/
```

Hint: The template project already provides a screenshot hotkey (F8).

The working solution must be committed till **November 13, 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.