UNIVERSITÄT STUTTGART
Institut für Visualisierung und Interaktive Systeme (VIS)

Thomas Ertl, Steffen Frey *Stuttgart, 29. 06. 2020*

Scientific Visualization (Assignment 9)

Exercise 9. 1 [10 Points] Direct and Indirect Volume Visualization

Prerequisites Obtain the shader skeleton `volume-renderingglsl.sec` from ILIAS and load it into *ShaderToy* (`https://www.shadertoy.com/new`; a brief description and introduction video is available in assignment 06). Note that ILIAS automatically renamed the file, feel free to restore the original filename `volume-rendering.glsl` after downloading. After the code has been compiled, a colored cube should be visible as shown in Figure 1(a).

The goal of this task is to visualize a synthetic 3D scalar-valued data set using various direct and indirect volume visualization techniques. For this, you have to complete different parts of the shader skeleton, which are marked with `TODO` comments. The `MODE` define at the beginning of the code is used to switch between the different techniques. Remember to set it accordingly after you have completed the respective part of the code. Implement the following parts of the shader skeleton:

1. **Ray Marching** (1 Point): Familiarize yourself with the structure of the program and the ray construction in `mainImage`. Complete the ray marching by calculating the current position on the ray using the ray origin and direction (`rayOrig`, `rayDir`) and the iteration variable `x`. Store the result in the variable `pos`.

2. **Early Ray Termination** (1 Point): Implement *Early Ray Termination* by completing the according function `earlyRayTermination`. Check if the accumulated color is nearly opaque and possibly terminate the integration by setting the variable `terminate` to `true`.

3. **Line Integration** (1 Point): Implement the transfer function `lineIntegration`. In this mode, the values along a ray shall be summed up as opacity of an arbitrary (bright) color. Note that it might be necessary to scale the values by a constant factor to obtain a visible X-ray like image. An exemplary rendering result is shown in Figure 1(b). Attach a screenshot of your solution.

4. **Maximum Intensity Projection** (1 Point): Implement the *Maximum Intensity Projection* (MIP) branch in `mainImage`. Store the highest value encountered along a ray and map it to color by calling the function `maximumIntensityProjection`. You might use the temporary variable `maxValue` for this. Store the result in `finalColor`. Note that you also have to complete the next task to obtain a visible result.

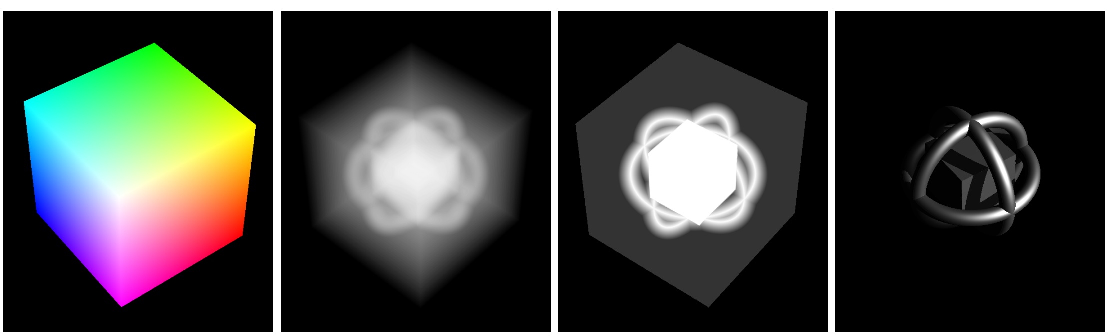
5. **MIP Transfer Function** (1 Point): Implement the transfer function `maximumIntensityProjection`. In this mode, the highest value encountered along a ray shall be mapped linearly to a fully opaque color. An exemplary rendering result is shown in Figure 1(c). Attach a screenshot of your solution.

6. **Isosurface Rendering** (1 Point): Implement isosurface rendering by providing a suitable condition for executing the according code block in `mainImage`. You might read the temporary variable `sampleValueLast`. The iso value is set in the define `ISO_VALUE`. An exemplary rendering result (including shadows) is shown in Figure 1(d). Attach a screenshot of your solution.

7. **Shadow Ray** (2 Points): Add shadows to the rendered isosurface by sampling the volume data set along a shadow ray and setting the variable `shadow` to `true`, if a value was encountered that is greater than the value defined in `SHADOW_THRESHOLD`. You might use the variable `sampleValueShadow` to store sampled values. Note that you also have to complete the next task to obtain a visible result.

8. **Shadow Color** (1 Point): Manipulate `finalColor` if the rendered fragment of the isosurface is shadowed. Darken `finalColor` by a fixed factor while retaining its alpha value. An exemplary rendering result is shown in Figure 1(d). Attach a screenshot of your solution.

9. **Classification** (1 Point): The synthetic volume data set contains a small string at the center. Make it visible by implementing the transfer function `classification`. The function should highlight the string while masking out the rest of the volume. Attach a screenshot of your solution and write down the found string in the `TODO` comment.



(a) (b) (c) (d)

Figure 1: Exemplary renderings created using the different program modes: (a) test image, (b) line integration, (c) maximum intensity projection and (d) isosurface rendering

Submit the complete, modified shader as code/text file. Use the *jpg* or *png* image format for your screenshots.

Submission Deadline: 2020-07-3, 23:55

please hand in your submission through the ILIAS system.

.