KTH Stockholm
CSC :: CST
Introduction to Visualization and Computer Graphics, Spring 2017, DH2320
Tino Weinkauf, Himangshu Saikia, Wiebke Köpp

Homework assignment No. 06

Due March 3, 2017

Introduction:
In this assignment you will be rendering a volume data set using the given framework. Unzip the archive and use cmake to obtain the solution files. Set `raycasting_task` as your start-up project.

For simplicity, you only need to do an orthographic projection of the slices of the volume in the XY plane. In other words, our viewing ray is parallel to the z-axis. It enters the volume at z_{min} and leaves it at z_{max}. Technically, to collect all sample points along a ray, you just need to query the volume for all vertex values for a given (x, y)-pair. The function `VolumeRaycaster::getAverage(int x, int y)` explains this.

We use a transfer function to obtain color and opacity for a given data value. The class `VolumeRaycaster` has already a transfer function as a member. To query it, use `tf.interpolateColor(datavalue)`. This function returns a `Vec4d`, which is a RGBA color tupel. Assuming a `Vec4d Color`, you get the opacity value using `Color.a()`.

The data set is a CT Scan of the human abdomen and pelvis. It contains also a stent in the abdominal aorta. The data is courtesy of Michael Meißner, Viatronix Inc., USA, and has been obtained from `http://www.volvis.org`.

This assignment works with two files:

- `raycasting_task/VolumeRaycaster.cpp`
  Here, you are going to implement different compositing schemes. Each scheme has its own function and you need to return the correct RGBA color.
- `raycasting_task/raycasting_task.cpp`
  Here, you can switch between the different tasks. Have a look at the `main()` function and the respective `Task()` functions.

Task 6.1: Simple Compositing Schemes **4+4 P**

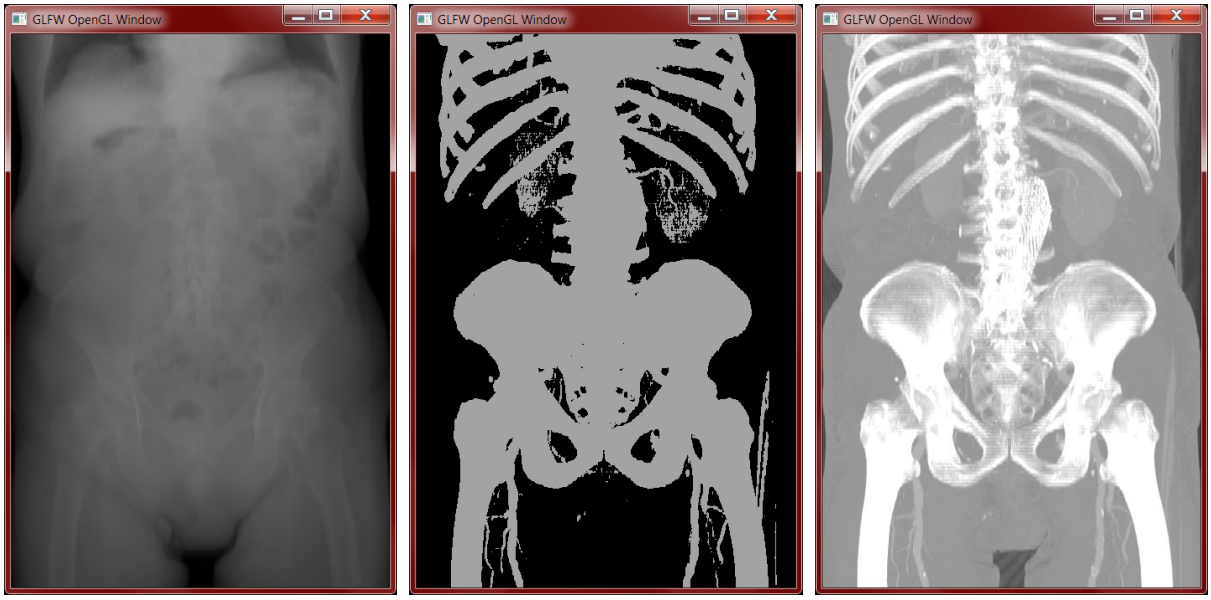
The *Average Intensity* strategy has already been implemented and serves as an example. See the function `VolumeRaycaster::getAverage(int x, int y)`.

Implement the following strategies:

(a) **First Hit.** Implement the first hit strategy in the function
`Vec4d VolumeRaycaster::getFirstHit(int x, int y, double isovalue)`.
We are interested in seeing the bones, which have data values around 0.65.

(b) **Maximum Intensity.** Implement the maximum intensity strategy in the function
`Vec4d VolumeRaycaster::getMaximum(int x, int y)`.

If you implemented everything correctly, you will see results similar to Figure 1.



(a) Task 1: Average (b) Task 1: First Hit (c) Task 1: Maximum Intensity

Figure 1: Results of Task 1 with a simple *Gray Ramp* transfer function.

Task 6.2: Accumulation Compositing 6+6 P

Hint: *Switch to* `Task2()` *in* `raycasting_task/raycasting_task.cpp`.

Implement the accumulation compositing scheme using

(a) the *front-to-back* strategy in `Vec4d VolumeRaycaster::accummulateFrontToBack(int x, int y)`,

(b) the *back-to-front* strategy in `Vec4d VolumeRaycaster::accummulateBackToFront(int x, int y)`.

If you implemented everything correctly, you will see results similar to Figure 2a.

You can test your implementation by switching to `Task2_Test()` in `raycasting_task/raycasting_task.cpp`. It renders the volume with both strategies and computes the difference between the images. The result needs to be pitch black.

Task 6.3: Transfer Function Design 5 P

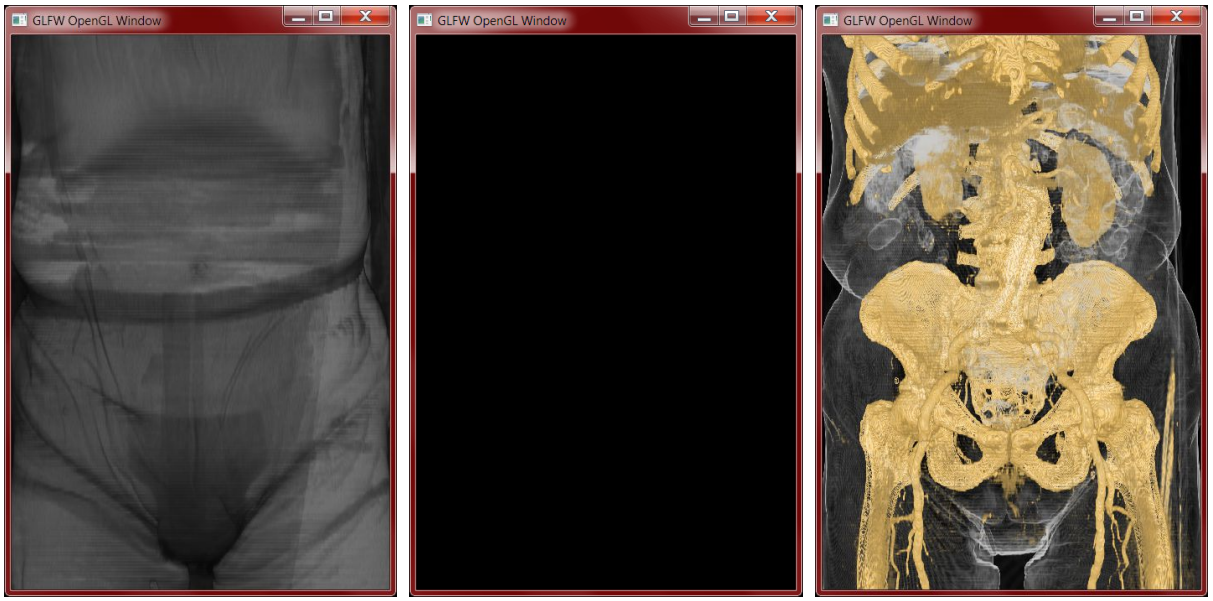
Hint: *Switch to* `Task3()` *in* `raycasting_task/raycasting_task.cpp`.

Consider the function `Task3()` in `raycasting_task/raycasting_task.cpp`. It renders the volume using *back-to-front* accumulation.

Your task is to define a transfer function such that the following features become visible:

- outer skin and clothes, which have data values around 0.4.
- bones and vessels, which have data values around 0.65.

Use different colors and opacities to create an insightful visualization! See Figure 2c for an example.



(a) Task 2: Accumulation with *front-to-back* or *back-to-front* strategy using a simple *Gray Ramp* transfer function.

(b) Task 2 Test: Difference image between *front-to-back* and *back-to-front* strategy. Both strategies shall create the exact same image. Hence the difference image needs to be black.

(c) Task 3: Accumulation with *front-to-back* or *back-to-front* strategy using a well-designed transfer function.

Figure 2: Results of Task 2 and 3. Note how an appropriate transfer function reveals the interesting structures of the data.