LAB 3: VOLUME RENDERING, INTERACTION & STEREO

ANDRÉS MARTÍNEZ MORA

CM2004: VISUALIZATION IN MEDICAL IMAGING

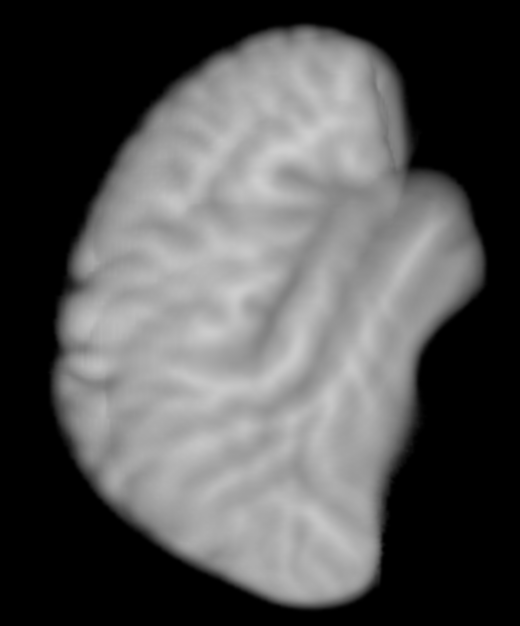
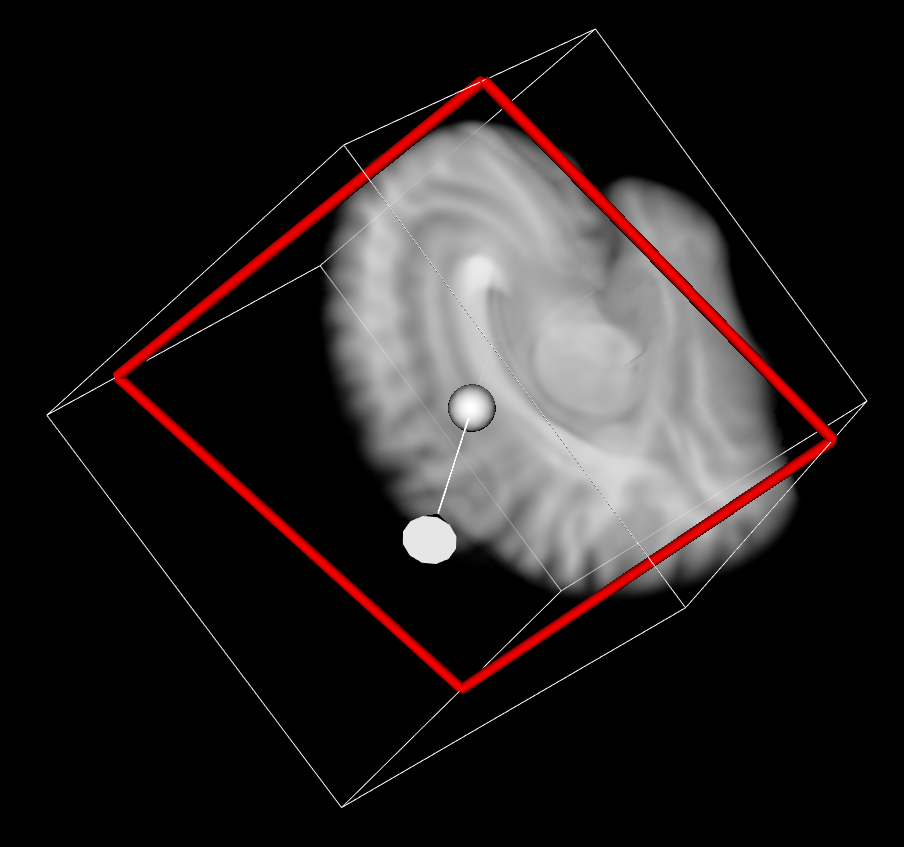
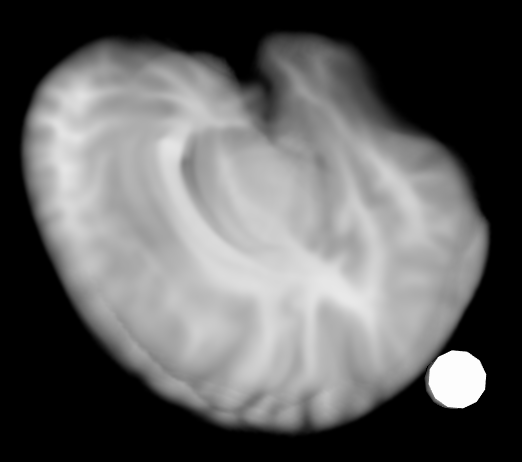
Exercise 1

In Exercise 1, a pipeline for plane clipping is prepared. The plane is initially stated as a global variable, being connected to the volume mapper. For every interaction event VTK requires a callback, which is initially defined in a separate function and afterwards in a separate class, so to execute the desired pipeline in every interaction event.

Plane clipping has to be defined through a widget that receives as input the image to render, has a conexion to the callback and is outputted into the Render Interaction Window.

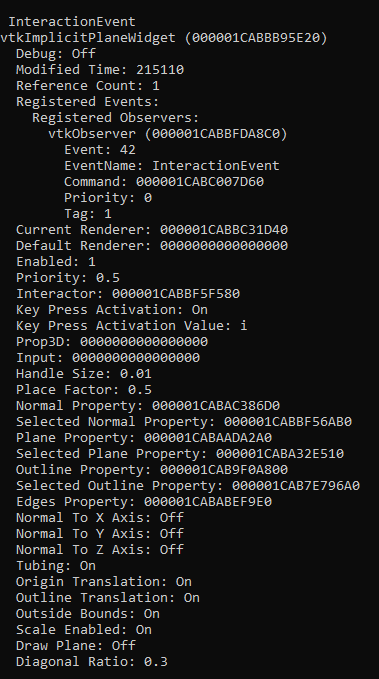
In order that interaction occurs, we have to press “i” after the OpenGL window is displayed. Then, a white dot appears in screen. By left or right clicking that dot, different clip planes and auxiliary measurement tools are shown together with the resulting visualization.

Some of the plane clipping results are the following:



The left image shows the visualization before connecting the plane clipping widget, the central image shows the visualization with the white dot for plane clipping interaction and the right image shows a visualization with an interaction variant.

During interaction, the object and event parameters were printed in the terminal, getting the following:

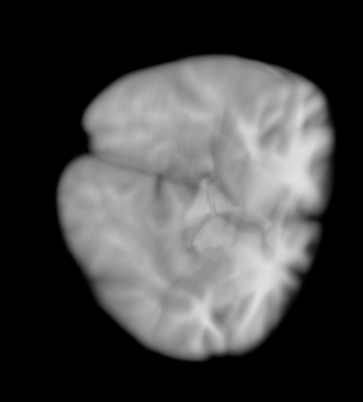
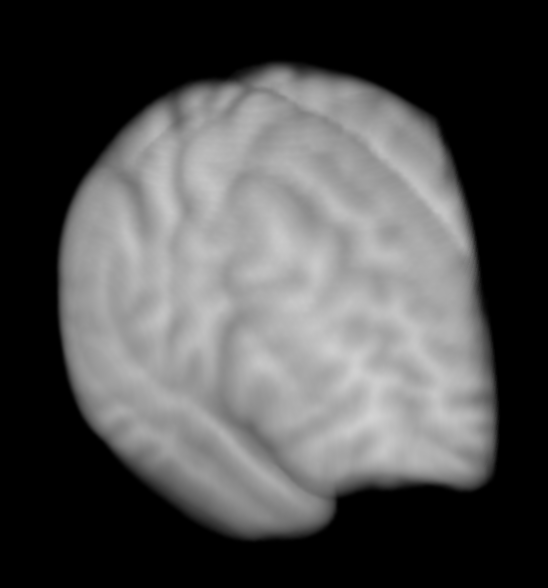


Exercise 2

This exercise aims to work with animation + plane clipping interaction events. To do so, the plane origin is set at the volume center, so that the rotation is done at the volume center. A timer callback class is designed so to define how the volume will be at each time for the interaction. In here, a time-dependent angle for the normal vector of the plane is computed. This angle is used to specify the clipping plane at each step of interaction time, inside the interaction window, requiring also an animation speed.

Changing the angles changes the plane for rotation, while changing the parameters for the timer callback class modifies the way the volume rotates.

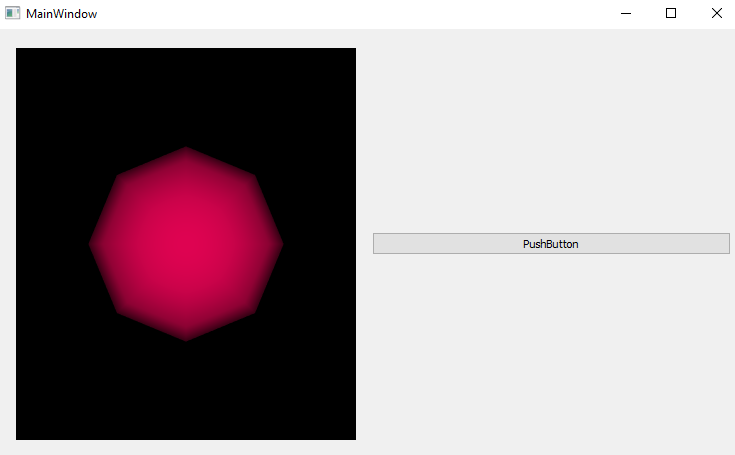
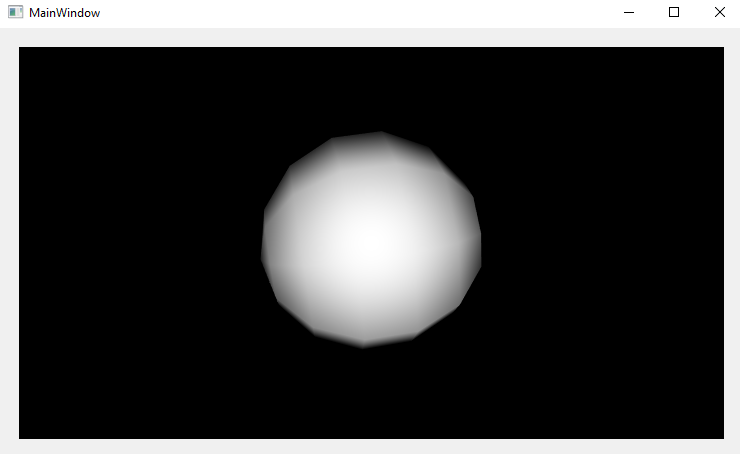
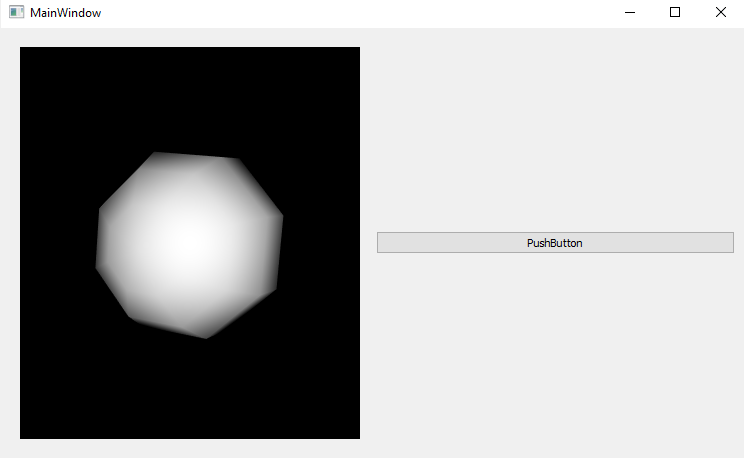
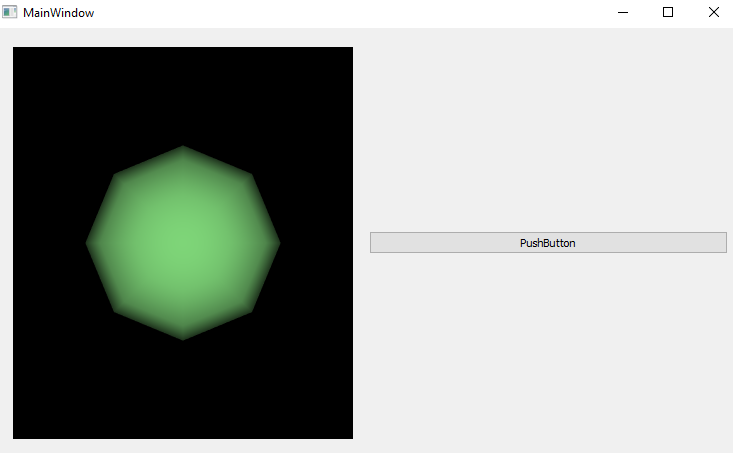
The timer is programmed so that it repeats the animation every 100 events. The results are the following. They show a rotating clip plane on the Z axis, clockwise along time.

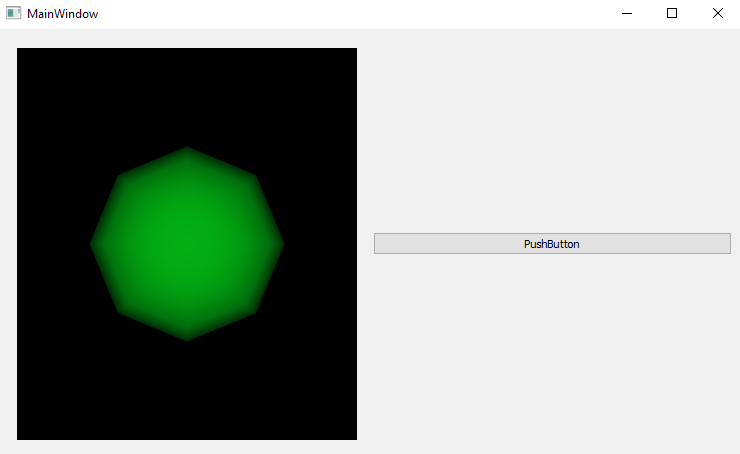


Exercise 3

VTK applications can also be controlled with buttons that control events, using the package “PyQt”. In here, the class “setup\_vtk\_renderer” provides the rendering window, the class “setup\_ui” establishes how the user interface will be designed, the class “on\_button\_clicked” tells the necessary actions to complete when the specified button in the interface is clicked and the “main” section executes everything.

In this exercise, we develop a user interface with PyQt for VTK rendering of a sphere source with a random color, so that the sphere randomly changes color every time a button in the User Interface is clicked. Results are displayed below:





The first image just shows the sphere, before the design of the user interface is applied and connected to the main pipeline. The second image shows the sphere when the user interface is created, but before the button is actually connected to the main pipeline. The rest of the images show the sphere in different random colors, just after the button as well as the user interface is connected into the main pipeline.

Exercise 4

The last exercise is a volume splatting application, where stereo based on anaglyphs (color multiplexing) is also applied in the pipeline, requiring to be observed with special glasses and to be displayed with a special screen. The results before and after stereo (in a normal screen) are depicted below. One can see in the stereo image red and blue-green traces, corresponding to the features of each image used in the stereo pipeline.

