- Basic GPU-Based Ray Casting:

  This can be found in the shader folder. Where the vertex shader it works like the standard vertex shader plus calculate the direction vector.

  Next the fragment shader in the `main` is the core algorithm. Here we have to explain three things: `stepSize` is calculated from the longest diagonal in the cube divided by the number of textures (smallest dimension) and a magic number resolution to have more or less samplings; next `powSize` is just the longest diagonal; and finally the `insideCube` function it's a hard-coded function to detect if the ray is outside a cube in (0,0,0) of (1,1,1) size.

  The Ray casting algorithm works like any other where in each step from front to back adds the colors and exits if it's outside or has alpha equal to 1.

- Phong Shading:

  In the `raycasting.frag` it's inside a function `RCPhong` here it's adapted to the ray casting where gets the Normal from the neighbors and use the same color for all the color types (specular, matte, ..).

- Interactive transfer function editing:

  Using the old library `QWT` I've used the `QWT Plot` and `QWT Curve` to make the Look up table. If you have problems install it using `sudo apt install libqwt-qt5-dev libqwt-qt5-6`.

  Then I've done small changes in `glwidget.cc:193-201` to send the data to the shader and `main_window.cc:16-17` to initialize the classes `plotx.cpp`, the class to draw one plot, and `plotRGBA.cpp` the class manager for the LUT.