TNM067 — Scientific Visualization
2. Scalar Fields and Volume Rendering

September 6, 2022

1 Introduction

This laboratory exercise will introduce scalar field visualization. The first part will include generating a scalar field based on the wave function of a hydrogen atom. In the second part, you will explore volume rendering by using existing processors in inviwo.

Important: The labs might require more hours than available in the scheduled lab sessions. It is important that you work on the assignments outside of sessions in order to be able to finish the labs.

1.1 Change log

- 2017-09-01: Initial Version.
- 2017-09-04: Typo fixes.
- 2020-09-02: Update for 2020, Update for distance teaching.
- 2021-07-19: Typo fixes, clarifications for some tasks, changed variable name Voxel to DataPoint, new task for marching tetrahedra tests.
- 2022-07-08: Fix to match code.
- 2022-08-31: Splitting up lab2 into lab2 and lab3

2 Documentation

On the following links you can read more about Inviwo and get help with questions related to Inviwo.

- Inviwo: http://www.inviwo.org
- Inviwo API Documentation: https://inviwo.org/inviwo/doc/
- Inviwo Issue tracker: https://github.com/inviwo/inviwo/issues

3 Part 1 - Hydrogen Wave Function

In the first part of the lab, we will generate a scalar field from the wave function of a hydrogen atom. From the wave function it is possible to calculate the probability of finding an electron at each position in space. The generated probability density plot will be used as a scalar field throughout the lab, on which we will test multiple scalar field visualization techniques.

3.1 The Wave Function

The wave function consists of two parts: the radial function $R_{nl}(r)$ and the spherical harmonics $Y_{lm}(\theta, \phi)$ (providing the angular part). The subscripts n, l, m are called *the three principal quantum numbers* and in this case you will use the wave function for $n = 3$, $l = 1$ and $m = 0$. Using spherical coordinates:

$$R_{3,1}(r) = \frac{4\sqrt{2}}{3}\left(\frac{Z}{3a_0}\right)^{\frac{3}{2}}\left(\frac{Zr}{3a_0}\right)\left(1 - \frac{Zr}{6a_0}\right)e^{-Zr/3a_0} \tag{1}$$

$$Y_{1,0} = cos\theta \tag{2}$$

giving the complete wave function:

$$\Psi_{3,1,0}(r, \theta, \phi) = \frac{1}{81\sqrt{6\pi}}\left(\frac{Z}{a_0}\right)^{\frac{3}{2}}\frac{Z^2r^2}{a_0^2}e^{\frac{-Zr}{3a_0}}\left(3cos^2\theta - 1\right) \tag{3}$$

from which the electron probability density function is calculated as

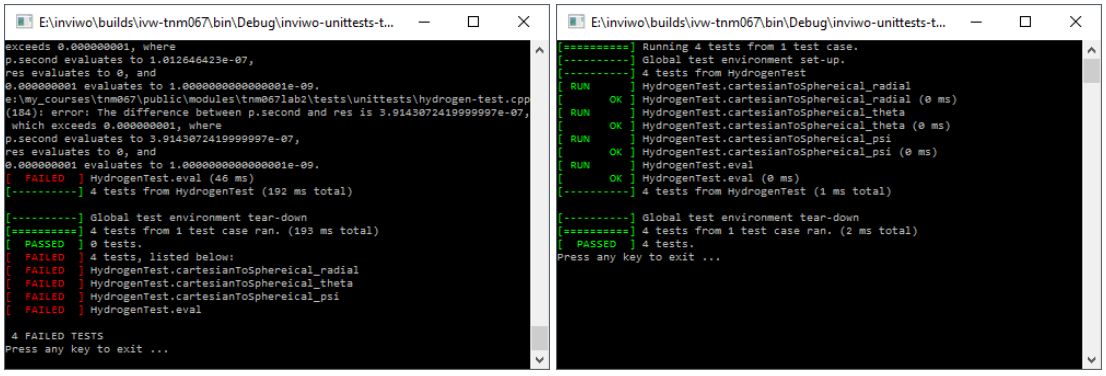
$$P_{3,1,0} = |\Psi_{3,1,0}(r, \theta, \phi)|^2 \tag{4}$$

In your code, do you have to calculate the absolute value of $\Psi_{3,1,0}(r, \theta, \phi)$? The constant a_0 is the *Bohr radius* (0.53Å) and Z is the charge of an electron. Because this case regards a single electron hydrogen, you should set:

$$Z = 1, \quad a_0 = 1 \tag{5}$$

If you feel the urge to read more about the wave function and the probability density function you can do so at http://academic.reed.edu/chemistry/roco/Density/cloud.html or read the book *Physics of Atoms and Molecules* by B.H. Brandsen and C.J. Joachain.

There is a set of equations which enables conversion between the Cartesian coordinate system and the spherical coordinate system. The angles ϕ and θ are easily mixed up because the notations are sometimes different in maths and physics, the notation here is that of physics. You can read more about spherical coordinates at http://mathworld.wolfram.com/SphericalCoordinates.html.



(a) (b)

Figure 1: Task 1 and 2 are using unit test to verify your solutions.

To go from the spherical coordinate system to the Cartesian coordinate system, use:

$$x = r \sin\theta \cos\phi \tag{6}$$
$$y = r \sin\theta \sin\phi \tag{7}$$
$$z = r \cos\theta \tag{8}$$

To go from the Cartesian coordinate system to the spherical coordinate system, use:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{9}$$
$$\theta = \cos^{-1}(z/r) \tag{10}$$
$$\phi = \tan^{-1}(y/x) \tag{11}$$

For task 1 and 2, set the startup project in Visual Studio to inviwo-unittests-tnm067lab2. This project contains unittests that will test code implemented in Task 1 and 2. At first, all test will fail (see Figure 1) . When you have implemented task 1, three out of four test will pass and after task 2, all test should pass. When all tests pass, change the startup project back to inviwo.

Task 1 — Cartesian to spherical conversion:
Implement the function *HydrogenGenerator::cartesianToSpherical* based on equations (9-11).

Task 2 — Implement the wave function:
Implement the function *HydrogenGenerator::eval* to evaluate equation (4). A common problem for some students with this task is to get lost in the amount of parentheses. Therefore, we strongly suggest to split the equation up on to smaller parts. Equation (12) shows an example way to divide the equation in to 5 parts whose product builds the final value.

$$\Psi_{3,1,0}(r,\theta,\phi) = \frac{1}{81\sqrt{6\pi}} \left(\frac{Z}{a_0}\right)^{\frac{3}{2}} \frac{Z^2 r^2}{a_0^2} e^{\frac{-Zr}{3a_0}} \left(3cos^2\theta - 1\right) \quad (12)$$

4 Part 2 - Volume Rendering

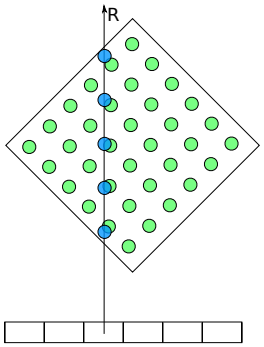


Figure 2: Casting a ray from the screen into the volume. Samples are taken along the ray which are composited to form the final pixel color.

4.1 Raycasting

Raycasting is an algorithm for performing direct volume rendering. The core of the algorithm is based on sending one ray, R, per pixel from the camera and then take samples along the ray inside a volume (see Figure 2). At each sample there is an amount of illumination, $I(x, y, z)$, from the light sources. Not all illumination is scattered in the direction to the camera, it depends on for instance on the local density, $D(x, y, z)$, at the point. It is not trivial to calculate the illumination, to do it correctly you must take the attenuation of the light source and possible shadows into account. A simplified approach is to calculate the gradient, i.e normal, at the sample point and then perform for instance Phong shading to obtain the intensity.

In OpenGL rays can be formed by rendering a proxy geometry around the volume. Keep in mind

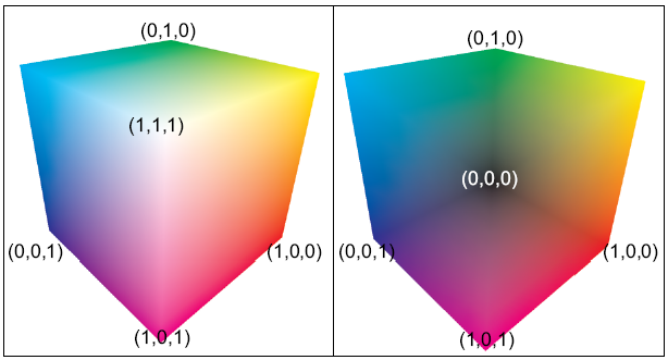


Figure 3: Texture coordinates of the front and back face rendered cubes.

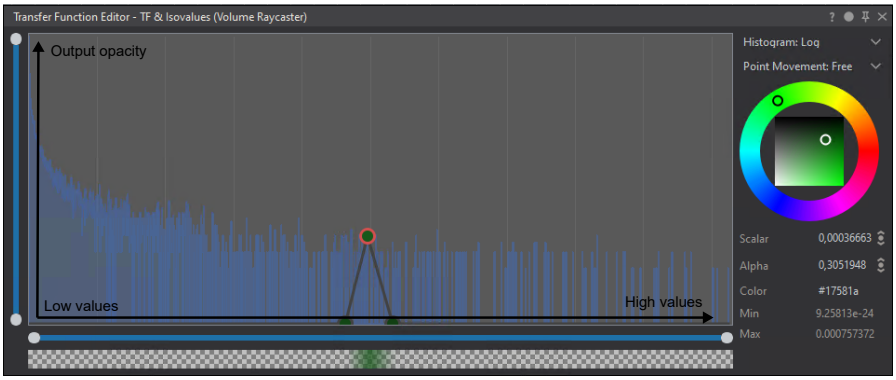


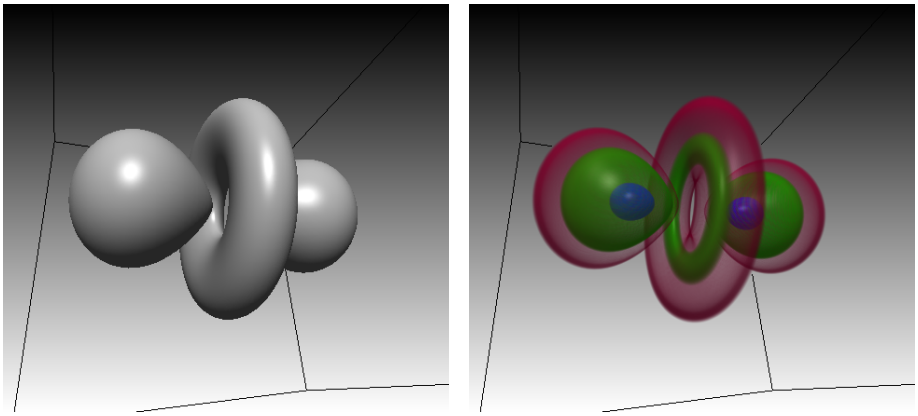
Figure 4: Inviwo's transfer function editor. Inviwo has a quite flexible transfer function editor. On the background you see blue bars representing the histogram of the voxel-intensities of the volume. You add and remove points by right clicking and move points by clicking and dragging. The location of the points on the x-axis represent what voxel-intensities it will be applied on and the height on the y-axis represent the output opacity of current sample. With the current setup all samples along the ray will have zero opacity, except those who samples whose intensity falls into the range around the green peak-point.

that a ray is an origin with a direction. Thus we must find out the origin and the direction for each ray. This can be done by rendering two cubes where the colors represent the coordinates. Each corner will have a coordinate and they will then be interpolated from the vertex shader to the fragment (pixel) shader, see figure 3. To get the origin a cube can be rendered with ordinary front face culling, this is where the ray will start/enter the volume. To get the direction of the ray the cube can be rendered using back face culling, the direction will then be the back face subtracted from the front face. Practically this can be done by storing the result of rendering the back face in a framebuffer and then use the front face to generate the fragments that will contain the origin of the rays.

For the following tasks in this part, you will use built in components in Inviwo to visualize the hydrogen, no programming will be needed for these tasks. You might want to save the workspace so you don't have to recreate it everytime you restart Inviwo. **If you are unsure how to connect components in Inviwo to render a volume, start by looking at the example workspace boron under File->Example Workspaces->Core->boron.inv**.

Task 3 — ISO Raycaster:
Inviwo has a processor called "ISO Raycaster" which uses a raycaster to visualize ISO surfaces of a volume. Extend your network by adding another canvas processor and the processors needed to visualize the hydrogen using the "ISO Raycaster". When you have your network you can see different ISO-surfaces by changing the ISO-value property on the "ISO Raycaster". See figure 5(a) for an example result.

Task 4 — Raycasting:
Similar to the "ISO Raycaster" Inviwo has a processor called "Volume Raycaster", this uses a transfer function, lighting and front-to-back compositing to let the user visualize multiple surfaces by using transparency. Open the transfer function editor by clicking the transfer function property in the "Volume Raycaster". Try to create a transfer function mimicking the example image in figure 5(b), which visualizes three different iso-values with different color and opacity. A good start for this is to try to make copy of the transfer function you can see in Figure 4.
Note: You can select and move several points at once by drawing a square around them.



(a) (b)

Figure 5: Possible results when visualizing the hydrogen using an ISO Raycaster and a Volume Raycaster.

Task 5 — Explore transfer functions:
Continue to explore the volume rendering by showing some different features in the dataset using **at least two new transferfunctions**. In the Volume Raycaster you can show ISO surfaces by changing the rendering settings (selection in the Rendering dropdown) and adding an isovalue in the transfer function editor.