CS171 Assignment 4: Rendering Isosurfaces by Volumetric Techniques

Introduction

In this assignment, you are required to implement the volume rendering algorithm for visualization of isosurfaces modeled by analytical implicit functions. You need to design your own transfer functions for the retrieval of optical properties.

In the following, we will give you the specifics about what you need to accomplish, as well as some related guidelines in order to assist your programming.

Programming Requirements

- [**must**] Implement gradient evaluation for the following implicit volume data: [30%]
  - Genus 2: $2y(y^2-3x^2)(1-z^2)+(x^2+y^2)^2-(9z^2-1)(1-z^2)=0$
  - Wineglass: $x^2+y^2-(\ln(z+3.2))^2-0.09=0$
  - Porous surface:

    $$\begin{aligned}&(2.92(x-1)x^2(x+1)+1.7y^2)^2\cdot(y^2-0.88)^2\\&+(2.92(y-1)y^2(y+1)+1.7z^2)^2\cdot(z^2-0.88)^2\\&+(2.92(z-1)z^2(z+1)+1.7x^2)^2\cdot(x^2-0.88)^2-0.02=0\end{aligned}$$
- [**must**] Design and implement an aliasing-free transfer function for visualizing isosurfaces with Phong lighting. [30%]
- [**must**] Implement front-to-back volume rendering with early ray termination. [40%]
- [**optional**] Simultaneously render multiple transparent isosurfaces with different iso-values of the same volume data. [15%]
- [**optional**] Use adaptive step size to accelerate the rendering process, i.e. use large step size for fully transparent area. [15%]

Notes

- Please be noticed that you are NOT allowed to use OpenGL in this assignment.
- As the computation in this assignment is relatively heavy, we encourage you to use OpenMP for acceleration.
- We will offer a code skeleton for you. Please understand the functionality of each declared class and method in the code skeleton before you start. Also please carefully read README.md in the project template to check the compilation and execution instructions.
- To verify your implemented algorithms, you can use various scene settings.

Submission

You are required to submit the following things through GitHub repository.

- Project code in the Coding folder.
- A PDF-formatted report which describes what you have done in the Report folder.

Submission deadline: 22:00, Dec 8, 2024

Grading Rules

- You can choose to do the [optional] item, and if you choose to do it, you will get additional scores based on the additional work you have done. But the maximum additional score will not exceed 15% of the entire score of this assignment.
- NO CHEATING! If found, your score for the entire assignment is zero. You are required to work INDEPENDENTLY. We fully understand that implementations could be similar somewhere, but they cannot be identical. To avoid being evaluated inappropriately, please show your understanding of code to TAs.
- Late submission of your assignment will subject to score deduction.
- After the deadline of this assignment, we will schedule an on-site demonstration. The specific date and venue will be posted on Piazza.

Skeleton Project/ Report Template

- The skeleton program and report template will be provided once you accept the assignment link of GitHub classroom. If you accept the assignment through the link properly, a repository which contains the skeleton project and report template will be created under your GitHub account.
- Please follow the template to prepare your report.

Implementation Guide

Here, we will instruct you to finish each part of this assignment. It is recommended to first overview the entire code skeleton and then fill each part by following your own understanding. This guide will not cover the optional parts, which mostly rely on your individual study.

Git Classroom

Accept the assignment in this link through Git classroom to start your assignment.

1. Implicit surfaces

In this assignment, the visualization targets are analytical implicit surfaces, which can be regarded as level sets of specific equations $\{(x, y, z)|F(x, y, z) = c\}$. For visualization purpose, you may also use normals on the surface, which can be calculated by $\mathbf{n}(x_0, y_0, z_0) = (F_x(x_0, y_0, z_0), F_y(x_0, y_0, z_0), F_z(x_0, y_0, z_0))$, i.e., the gradient ∇F of the impl $_0, y_0, z_0)$, which can be obtained as the close-form expression. For more details of implicit surfaces, please refer to Wikipedia.

You are required to complete the GenusTwoSurface::computeGradient, WineGlassSurface::computeGradient and PorousSurface::computeGradient methods, which are expected to return a gradient at the given point.

Note: you can utilize software packages, like Wolfram Mathematica, to solve function derivatives if you think manual derivation is tedius.

Related files: implicit_geom.hpp, implicit_geom.cpp

2. Transfer Function

Transfer functions are designed for retrieving the optical properties, like (colored) emission and transparency, according to input values. In this assignment, the input values are implicit function values, i.e., for instance, if we want to render the surface, we can design a transfer function that returns transparency of 0 for those regions that satisfy the iso-value, and return transparency of 1 for the points of other regions.

Theoretically, transfer functions for handling isosurfaces are supposed to be like a shifted Dirac delta function, which only yields non-zero responses when inputs are exactly equal to the specified isovalue. However, this function may cause serious aliasing artifacts. Thus, we often compromise to approximate it by using a Gaussian function.

Once you find the responses of the input values, you can map them to optical properties, i.e., (colored) emission and transparency. You are required to perform Phong lighting on the isosurface to compute its colored radiance. The base color of the surface can be chose arbitrarily, but we encourage you to use color mapping to visualize various properties of the surface, including normals, curvatures, or any other properties you are interested in.

For this part, you need to fill the IsosurfaceClassifier::transfer function to find the responses of input values and transfer the response values to optical properties.

Note: Colormaps would be very useful for visualization. For your convenience, we have included a color mapping library.

Related files: classifier.hpp, classifier.cpp

3. Volume Renderer

With transfer functions, we can implement a volume renderer for visualizing implicit surfaces. This paper introduces basic formulation of volume rendering. The concepts and formulation of volume rendering are in our lecture slides, where the composition schemes are given on p. 60. Since not all the contents are covered in lectures, you can review the tutorials on week 11 & 12 which describes the details of volume rendering. Besides, the slides from SJTU elaborates most of the elements in volume rendering as well, which may help you to understand and finish this assignment.

Also note, in order to make your rendering procedure efficient, your front-to-back volume rendering are obligated to perform early ray termination.

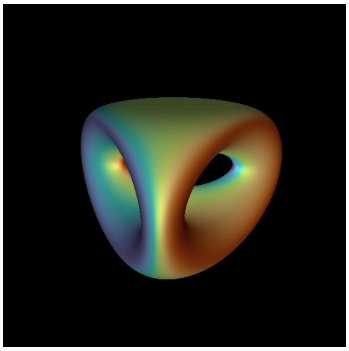
The code you need to complete is located in the VolumeRenderer::renderFrontToBack method.

Note: the ImplicitGeometry::bboxRayIntersection method can detect and compute entry/exit points of intersection between the given ray and the bounding box of an implicit geometry.

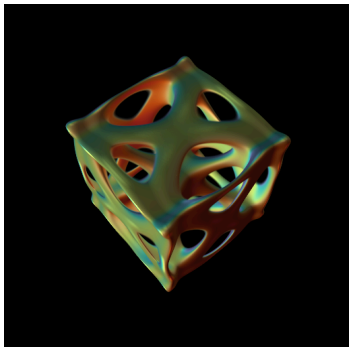
Related files: volume_renderer.hpp, volume_renderer.cpp

Expected Results

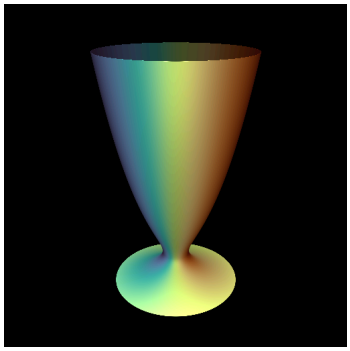
The first three figures are visualization of normals' x-components on surfaces (with Phong shading), the last one is Phong shading of a surface with pure color:



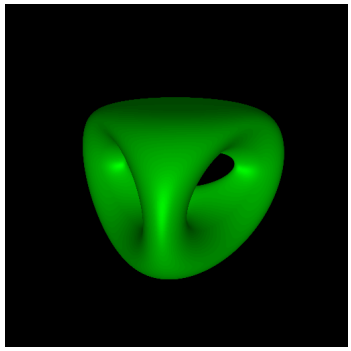
Genus 2



Porous Surface



Wineglass



Genus 2 (pure color)
