Assignment 5 : Volume Rendering Using Ray-casting

NAME: 叶者
STUDENT NUMBER: 2018533140
EMAIL: YEZHE@SHANGHAITECH.EDU.CN

1 INTRODUCTION

1.1 Environment

For this assignment, I use CMake 3.16.0-rc1 to generate the project and use Visual Studio 2019 to compile.

1.2 Overview

In this assignment, I have implemented the requirements, including:

· Volume Preprocessing
· Interpolator
· Classifier
· Compositor
· Renderer (including Sampler)
· Volumetric Shadow Implementation

2 IMPLEMENTATION DETAILS

2.1 Volume Preprocessing

During volume preprocessing, the gradient of the volume is calculated and stored. In this assignment, I use central difference to calculate the gradient.

First, if the target volume is at the boundary (whether x, y, z), I simply use its inner neighbor's gradient as its gradient.

Then for a volume data at index (x, y, z), marked as $v_{x,y,z}$, the gradient is calculated as:

$$
\begin{aligned}
grad(v_{x,y,z})_x &= \frac{den(v_{x+1,y,z}) - den(v_{x-1,y,z})}{||pos(v_{x+1,y,z}) - pos(v_{x-1,y,z})||} \\
grad(v_{x,y,z})_y &= \frac{den(v_{x,y+1,z}) - den(v_{x,y-1,z})}{||pos(v_{x,y+1,z}) - pos(v_{x,y-1,z})||} \\
grad(v_{x,y,z})_z &= \frac{den(v_{x,y,z+1}) - den(v_{x,y,z-1})}{||pos(v_{x,y,z+1}) - pos(v_{x,y,z-1})||}
\end{aligned}
$$

where $grad(\cdot)$ means gradient, $den(\cdot)$ means density, $pos(\cdot)$ means position.

2.2 Interpolator

I implemented 2 interpolators: Nearest Neighbor Interpolator and Trilinear Interpolator.

2.2.1 Nearest Neighbor Interpolator. The implementation is very simple. Just compare the position to the eight volume position in the voxel, and use the density and the gradient of the closet volume data.

2.2.2 Trilinear Interpolator. Let the interpolated position to be p, the 8 volume data in the voxel be c_{000} to c_{111}. Assume we interpolate value t.

Let

$$x_d = \frac{p.x - c_{000}.x}{c_{100}.x - c_{000}.x}$$
$$y_d = \frac{p.y - c_{000}.y}{c_{010}.y - c_{000}.y}$$
$$z_d = \frac{p.z - c_{000}.z}{c_{001}.z - c_{000}.z}$$

First we interpolate along x axis.

$$t_{00} = c_{000}.t(1 - x_d) + c_{100}x_d$$
$$t_{01} = c_{001}.t(1 - x_d) + c_{101}x_d$$
$$t_{10} = c_{010}.t(1 - x_d) + c_{110}x_d$$
$$t_{11} = c_{011}.t(1 - x_d) + c_{111}x_d$$

Then we interpolate along y axis.

$$t_0 = t_{00}(1 - y_d) + t_{10}y_d$$
$$t_1 = t_{01}(1 - y_d) + t_{11}y_d$$

Then we interpolate along z axis to get the final interpolated value t.

$$t = t_0(1 - z_d) + t_1$$

2.3 Classifier

First I use **tinycolormap** to map density value to color. Then I do local Phong shading.

The normal used in the Phong shading is the normalized gradient. The ambient is set to $(0.05, 0.05, 0.05)$.

The diffuse and specular calculation is very similar to the previous assignments.

For the transfer function, I use the gradient norm to enhance the structure of the smoke.

The output color is defined as:

$$color = G{\cdot}PhongResult{\cdot}dt{\cdot}density{\cdot}(1{+}\gamma{\cdot}\frac{||gradient||}{MaxGradientNorm})$$

Where G is to amplify the color and is set to 5, γ is the coefficient of gradient effect, and is set to 0.05. dt is the sampling step.

The output transparency value (the transparency vector has same components)is defined as:

$$tran = e^{-density \cdot dt} \cdot (1 + \gamma \cdot \frac{||gradient||}{MaxGradientNorm})$$

Where γ is the coefficient of gradient effect, and is set to 0.05. dt is the sampling step.

2.4 Compositor

I implement two scheme of compositing: forward scheme and backward scheme.

The forward scheme is implemented by:

$$C_{dst} = C_{dst} + (1 - \alpha_{dst})C_{src}$$
$$\alpha_{dst} = \alpha_{dst} + (1 - \alpha_{dst})\alpha_{src}$$

The backward scheme is implemented by:

$$C_{dst} = C_{src} + (1 - \alpha_{src})C_{dst}$$

2.5 Renderer (including Sampler)

The renderer is also simple. First generate rays from camera, and for each ray examine the intersection result. If the ray intersects with the volume, then it will have an entry point and an exit point, with ray parameters t_{start} and t_{end}.

Then we sample t from t_{start} to t_{end} by a fixed step dt, which is set to 0.005. For each sample, first use interpolator to interpolate the volume data, and then use classifier to generate optics data, then use compositor the blend the data.

2.6 Volumetric Shadow Implementation

The volumetric shadow is implemented in the renderer. It is implemented as follows: For each sampled position, shoot a ray to the light source, integral transmittance along the path, and then use the transmittance to effect the color.

The discrete integral is performed as follows:

$$transmittance *= e^{-\kappa \cdot density \cdot dt}$$

Where κ is the coefficient and is set to 5, *density* is the sampled density and dt is the **integral sampling** step length, which is set to 0.05 for performance.

3 RESULTS

3.1 Base result

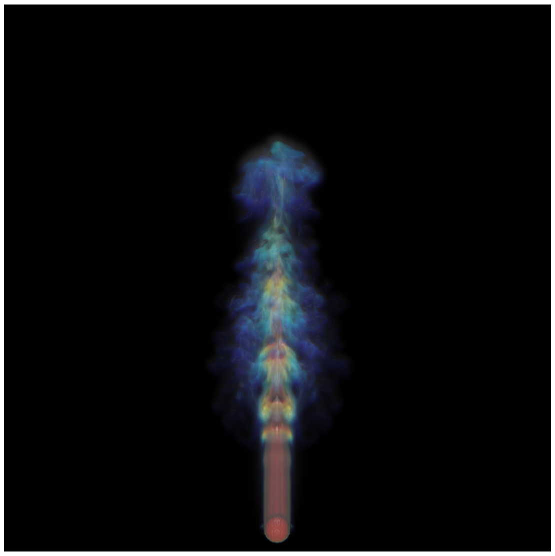

Fig. 1. Forward, Trilinear

3.2 Volumetric shadow comapre

To enhance the volumetric shadow effect, the light position is set to $(5, 0, 0)$.

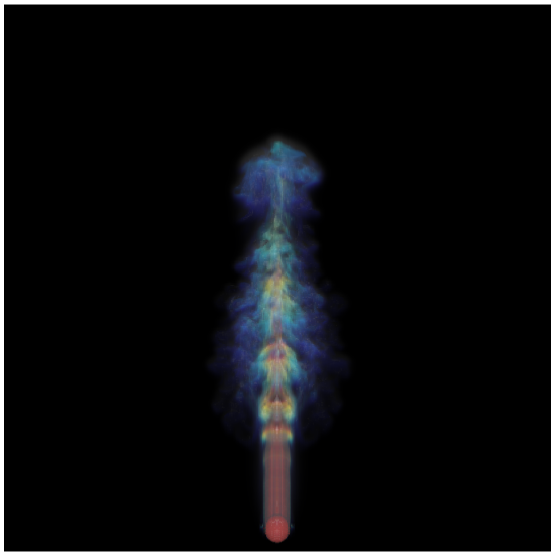

Fig. 2. Forward, Nearest Neighbor

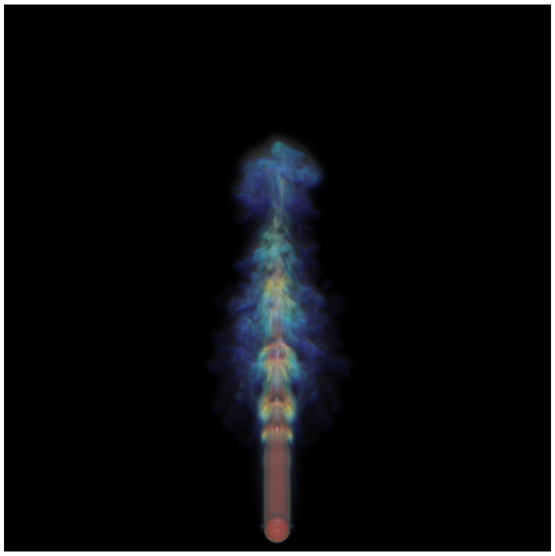

Fig. 3. Backward, Trilinear

student number: 2018533140
email: yezhe@shanghaitech.edu.cn

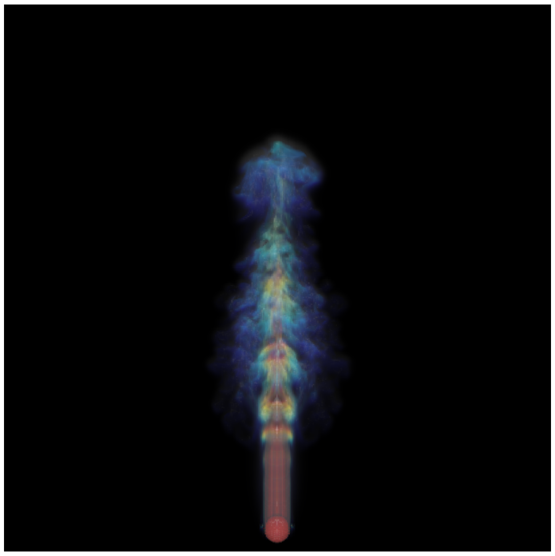
Fig. 4. Backward, Nearest Neighbor

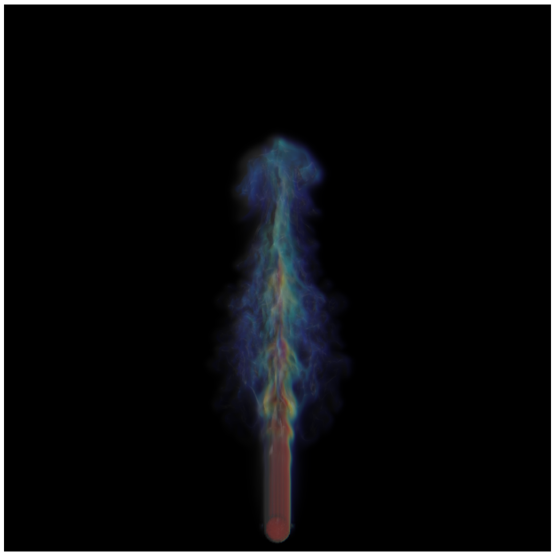
Fig. 6. With volumetric shadow

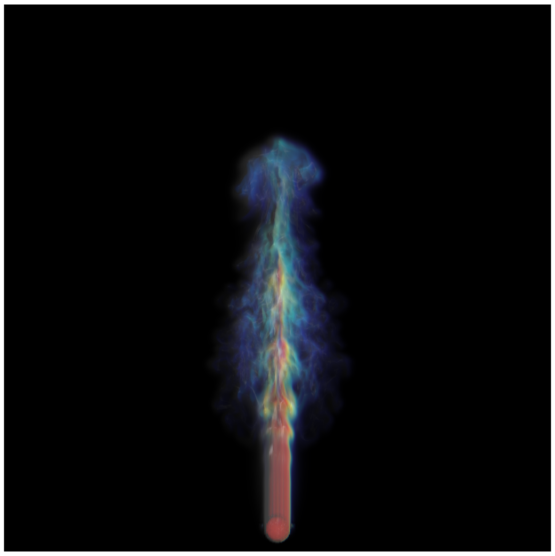
Fig. 5. Without volumetric shadow

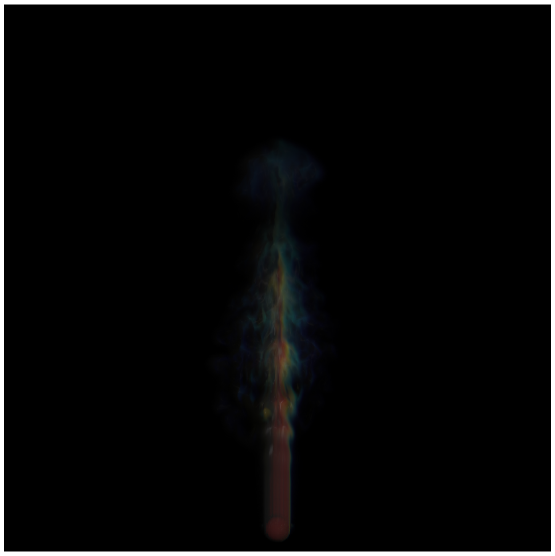
Fig. 7. Difference between above two image