Volumetric Renderer Improvements

Advanced Computer Graphics

Victor Ruiz Jiménez **NIA:174227**
Arnau Ruiz Fernández **NIA:192961**

I. Introduction

Our work is based in exploring different tools to improve our basic Ray Tracing Volumetric Renderer. We will explain in the following report the introduction of some techniques like Jittering, Phong illumination model, a transfer function, Isosurfaces and Gradients and Volume Clipping to achieve a better visualisation in our geometry and scene.

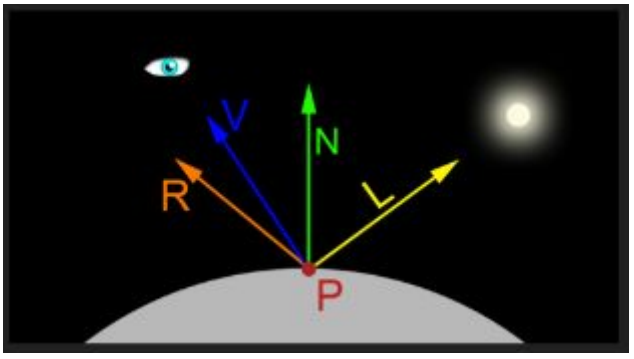
II. Implementation

Volume Clipping

This technique consists on cutting regions of the volume from the visualization. It can help to hide unimportant regions of volumes.
We implemented a separated 'if' condition to keep calculating the values of our volume (our ray is still tracing the volume) and we made a condition between two thresholds (maximum and minimum) for not printing any pixel out of this range. Then by modifying this threshold range we print only the pixels that we want. We implemented in the render screen some faders that controls the maximum and minimum coordinates we want to draw, in the three coordinates axis.

Local illumination model

We implemented Phong illumination model, to approximate the amount of light ('material') and object is receiving based on the orientation of the object, the position of the light and some material properties. These equations are computed per pixel, that gives a better result but takes more time to compute (more computation work that Goureud illumination model). We computed the 'P' point of the volume, its normal 'N', the position of the light 'L', the eye position 'V', and the reflection of the light 'R'.



Then we use these parameters to compute the sum of lights (ambient, diffuse and specular lights), the light equation.

$$Ip = KaIa + \sum(Kd(L \bullet N)Id + Ks(R \bullet V)^{\alpha}Is)$$

- Adaptations:

Isosurfaces and Gradients

Isosurfaces are surfaces in the volume with the same scalar value. The Gradients, are derivatives and represents the orientation of local surfaces within the volume. There are multiple ways to compute the gradient. Some are numerical, others are analytical. In 3D, we have to compute the partial derivatives of 'x', 'y' and 'z' coordinates of all the points, as the gradients in 3D are the normal vectors of isosurfaces.

The normal vector of the light 'N' will be our calculated gradient.

- Render screen:

The material parameters are 3 faders corresponding to the ambient constant light, the diffuse and the specular light. A fourth fader corresponds to the shininess factor.
With 3 faders more we can control the position of the light on an exact coordinate along the 3 axis.
We implemented a light color RGB button, to change the color of the light.

We are now able to appreciate illuminated, shadow and shiny areas.

Transfer function

These functions are mappings between volume properties and visual properties. We will play with color properties. We will make it fast and simple. We implemented a function capable of separating printed pixels depending of their voxel intensities. the transparency, with a simple 'if' command and some conditions. With this kind of functions we could see different parts of the volume, in our case, the bones, organs and flesh coloured differently.

Jittering

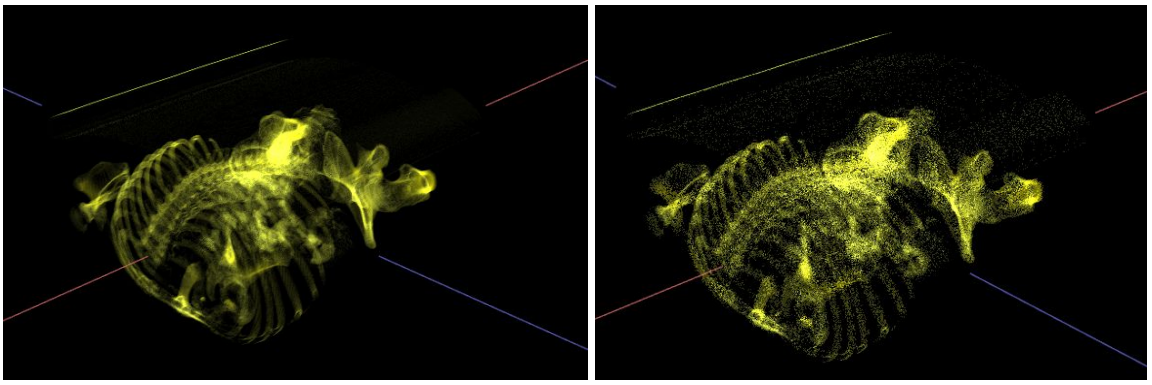
When we have a longer step_vector, some artifacts appear. Jittering is a technique that improves the framerate without raising the size, by offsetting the starting positions with random noise.

We provide a function 'random(vec2)' that for two input coordinates 'x' and 'y', returns a random number (not exactly, is pre defined). The first thing the function does is to go from 2D to 1D. The numbers are chosen so they do not repeat typically. Also we have a

floating point addition there, so it does a mix. Lastly it amplifies the error in the 'sin()' implementation by multiplying and taking the fraction. Then this random float is saved in a variable and multiplied by the step vector to implement an offset of the starting positions.

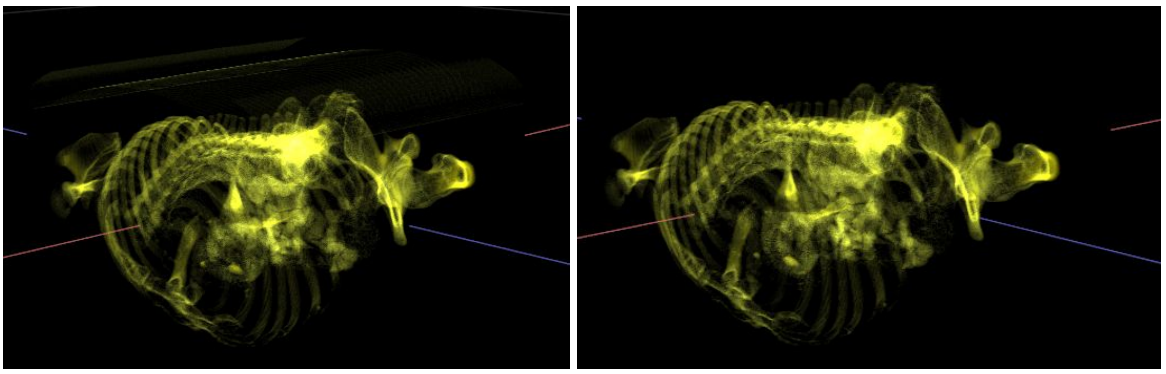
III. Improvements and results

After the Jittering implementation, we saw that now when we are increasing the step size of our volume renderer (step size fader in our render screen) this variable is been modified by the noise (noise artifacts appear when we have a bigger step size). Aliasing artifacts or wood pattern artifacts are not visible now, so we achieved a better visualisation with a bit of noise but more smooth.



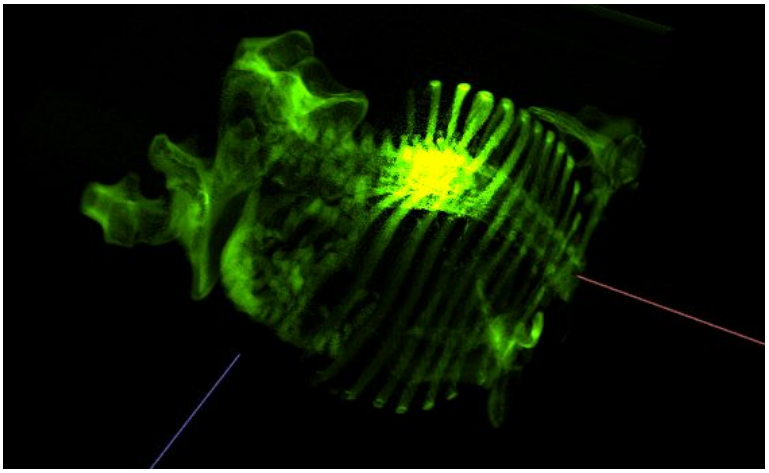
Step size = 0.01 Step size = 0.07

With the Volume Clipping technique we hide unimportant regions we do not want to visualize. In the first following image we observe the metal plate where the body is, while the second image remains hidden.



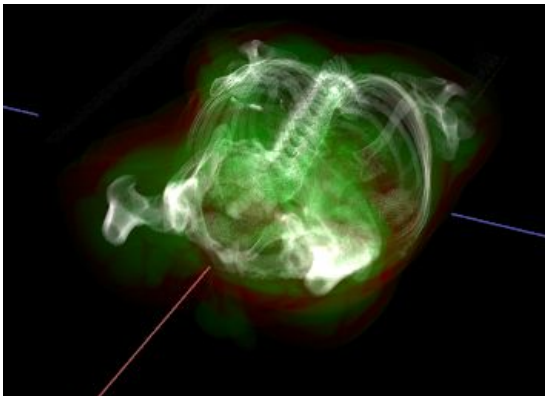
Images without vs with Volume Clipping implementation

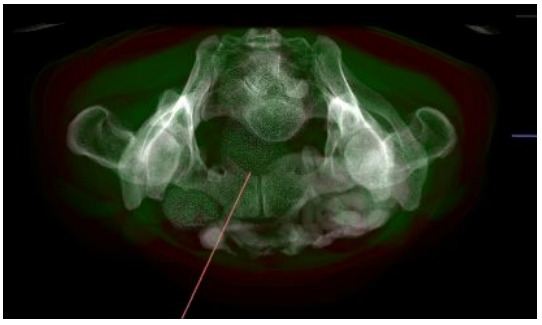
Implementation of Phong model illumination changing some fader parameters of our render:



Decreasing ambient and diffuse light (first and second fader of 'Material parameters'), with a high specular light, we could observe the light bulb affecting the volume.

After the implementation of the Transfer function (pressing the 'Switch preset button') we can easily see the flesh, bones and muscles.





With the volume segmented