Eurographics Conference on Visualization (EuroVis) (2014)
N. Elmqvist, M. Hlawitschka, and J. Kennedy (Editors)

Volumetric Ray Casting Implementation in VPT

Rok Cej

University in Ljubljana, Faculty of Computer and Information Science

Abstract

Volumetric ray casting is a method for visualizing 3D volumetric data, which is commonly used in medical imaging. It uses ray-volume intersection tests to render the data without relying on any geometric primitives, such as triangles. Transfer functions are used to assign color to the data based on its properties. This paper describes the implementation of volumetric ray casting in VPT framework, which uses JavaScript and WebGL 2.0. The program is platform-independent and runs in real-time.

Keywords: Ray casting - Volume rendering - WebGL.

1. Introduction

This project was created as an assignment for my Advanced Computer Graphics class. The goal of this project is to implement volumetric ray casting in VPT framework, which was developed in JavaScript and WebGL 2.0. The framework supports reading volumetric data by converting it into a 3D WebGL texture, which can be sampled through a shader.

2. Features

2.1. Ray casting

Ray casting algorithm is implemented on the GPU as a fragment shader, where each fragment (pixel) represents a ray leaving the camera. An intersection test is performed to see if the ray intersects the bounding volume of the object. If an entry- and an exit-point are determined, the volume is sampled on evenly spaced intervals between the two points.

Since volume data is stored as a texture, linear filtering can be used to achieve interpolation between neighboring voxels. The color and opacity of each voxel are determined using a transfer function that takes voxel data as input. Colors are accumulated in a front-to-back manner, taking opacities into account. Once the ray exits the volume or the accumulated samples become opaque, the final colour is obtained.

2.2. Random sampling

If all rays sample data at the same intervals, visual artifacts start appearing (unless the intervals are much smaller than voxels), since entire slices of volume parallel to the viewing plane are being skipped. The solution to this is to provide each ray with a random starting offset. One possible way of generating a random offset for each fragment is to generate noise based on a random seed, and then use the fragment coordinates to sample the noise. My solution uses a random unit vector as a seed for the noise.

2.3. Image convergence

Having a random offset for each ray gets rid of artifacts, but the result looks noisy. This can be improved by accumulating rendered frames and displaying their average. For that we need to have an additional accumulation buffer, as well as keep track of the number of rendered frames. The resulting image quickly converges and eliminates the noise.

2.4. Voxel gradients

By default, the volume data only has a density component. We can use that to pre-compute internal gradients, which can be used to improve visualization. They are calculated using the Sobel operator, which uses an edge detection technique. Greater gradients indicate greater changes in density around a given voxel.

Gradients have several useful features. We can compute their magnitude, and use it as an input to the transfer function, giving us more control over visualization. Furthermore, if we multiply voxel opacity with gradient magnitude, homogeneous regions will become less visible and high-variance

regions will stand out more. Lastly, we can use the gradient direction to compute a voxel normal, which allows us to implement shading.

2.5. Lights

All lights have a color and intensity parameters. Intensity specifies the amount of color emitted when sampling a light.

2.5.1. Light types

The following types of lights are implemented:

- **Uniform (ambient) light**: light that is uniformly present throughout the volume. It has no position or direction, therefore it ignores any reflection models.
- **Point light**: light with no surface area and a given position. It also has an attenuation parameter, which can simulate light fall-off that is inversely proportional to the squared distance.
- **Directional light**: light that is infinitely far away with a given direction.

2.5.2. Multiple lights

The program supports multiple simultaneous lights of any type, which can be added or removed through the UI. The maximum number of lights can be set arbitrarily.

2.6. Reflection models

The program supports multiple materials, which reflect light differently. Each has an ambient illumination parameter, along with other material-specific parameters. The following models are implemented:

- **Lambertian**: has a diffuse term that weights the reflected light with the cosine of the angle between the light ray and the normal.
- **Phong**: on top of a lambertian diffuse term, it also has a specular term, which weights the reflected light as the cosine of the angle between the view ray and the reflected light ray, to the power of shininess.
- **Blinn-Phong**: similar to the Phong model, the only difference is that the specular term uses the cosine between the normal and the halfway vector between the light ray any the view ray.

3. Results

These are some of the images that can be generated using this algorithm. The model used is the head of a baby.

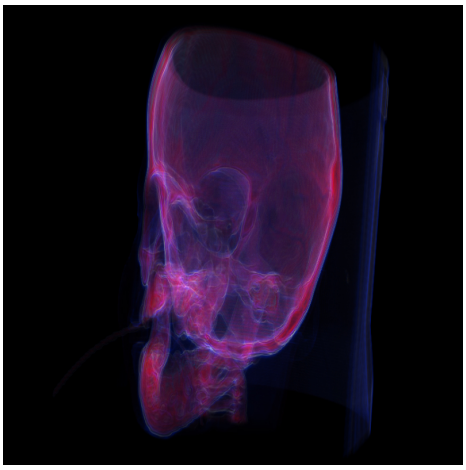

Figure 1: *Uniform lighting, custom transfer function.*

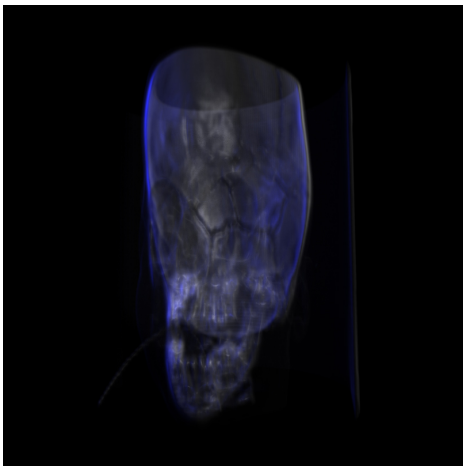

Figure 2: *Two point lights (white and blue), default transfer function, Phong material.*

4. References

- VPT, https://github.com/terier/vpt. Last accessed 15 May 2020.
- Real-Time Interactive Platform-Agnostic Volumetric PathTracing in WebGL 2.0, http://lgm.fri.uni-lj.si/wp-content/uploads/2018/07/1537821891.pdf. Last accessed 15 May 2020.
- Volume Visualization With Ray Casting, http://web.cs.wpi.edu/~matt/courses/cs563/talks/powwie/p1/ray-cast.htm. Last accessed 15 May 2020.
- Volume Ray Casting, https://en.wikipedia.org/wiki/Volume_ray_casting. Last accessed 15 May 2020.
- Efficient Ray Tracing of Volume Data, https://dl.acm.org/doi/pdf/10.1145/78964.78965. Last accessed 15 May 2020.
- Lambertian Reflectance, https://en.wikipedia.

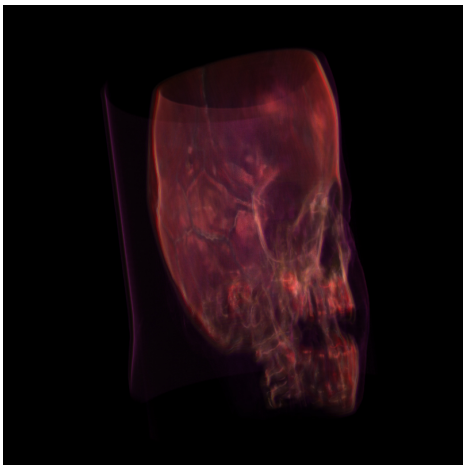

Figure 3: *Directional light, custom transfer function, Phong-Blinn material.*

org/wiki/Lambertian_reflectance. Last accessed 15 May 2020.

- Phong Reflection Model, https://en.wikipedia.org/wiki/Phong_reflection_model. Last accessed 15 May 2020.
- Blinn-Phong Reflection Model, https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_reflection_model. Last accessed 15 May 2020.