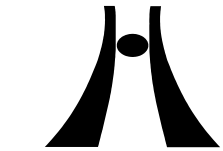
Code:_____



HÖGSKOLAN
I GÄVLE

Faculty of Engineering and Sustainable Development

A rendering method for simulated emission nebulae

Adam Carlson
June 2011

Bachelor Thesis, 15 credits, C
Computer Science

Computer Science program
Examiner: Stefan Seipel
Supervisor: Peter Jenke

A rendering method for simulated emission nebulae

by

Adam Carlson

Faculty of Engineering and Sustainable Development
University of Gävle

S-801 76 Gävle, Sweden

Email:
adam.carlson89@gmail.com
ndv08acn@student.hig.se

Abstract

Emission nebulae are some of the most beautiful stellar phenomena. The newly formed hot stars inside the nebulae ionize the surrounding gas making it glow in variety of colors. The focus of this work is to find a method for interactive rendering of simulated emission nebulae. A rendering program has been developed to render and generate nebulae. The emission light color is evaluated as a function of the accumulated density between the gas and the ionizing star. The rendering program can render a large variety of nebulae from any viewpoint with interactive performance on PC hardware. The method proposed in this work is visually accurate to real nebulae.

Contents

1 Introduction

1.1 Background

Nebulae are some of the most beautiful and impressive phenomena in the sky. Some of the brighter nebulae can be observed with a telescope but most are too faint to make out due to their low surface brightness and long distance from Earth.

Nebulae are composed of interstellar gas and dust that is distributed over huge areas. They become visible either by reflecting light (reflection nebulae) from nearby stars or by emitting light (emission nebulae) when the gas is ionized by the photons from nearby stars. Denser dust will also block light from passing, creating darker regions which can have dramatic visual effect. Nebulae are the birthplace of new stars and it is these hot young stars that illuminate a nebula. Eventually the stellar wind from these stars will disperse the surrounding gas and the nebula will disappear, leaving a cluster of newborn stars behind. An example of an emission nebula can be seen in Figure 1.

The color of light emitted by a nebula depends on the distribution of elements and the energy of the photons from the nearby stars [1]. The energy of photons emitted from the star decreases as they pass through more dense areas of gas. This makes the emitted color dependent on the distance from the star and the density of the gas in the path of the photons. As the cold dense gas in a nebula is pushed back by the stellar wind, a boundary forms between the colder gas and the hot gas surrounding the star. These boundaries can form many different shapes and emit light in many colors as the photon energy will rapidly decrease as the gas density increases.

Emission nebulae exist as two types, diffuse nebulae and planetary nebulae [1]. Diffuse nebulae are large regions of star forming gas while planetary nebulae are isolated and formed around stars when they eject most of their gas at the end of their lifecycle.

A problem with nebulae is however that they are very difficult to observe due to their large distance from Earth. Because of this large distance the only thing that can be observed from Earth are 2D projections of the original 3D nebulae [2].

By using an interactive 3D application, simulated nebulae can be explored from any viewpoint and without the need for a powerful telescope. This would make the wonderful visuals nebulae presents accessible to anyone. To fully capture the dramatic visuals and contrasts of nebulae a method for calculating the emitted colors of the gas and its internal boundaries needs to be devised.

Earlier work in this area has shown that this kind of volume rendering can be very slow [3] [4] and this makes it difficult to interact with the visualization.

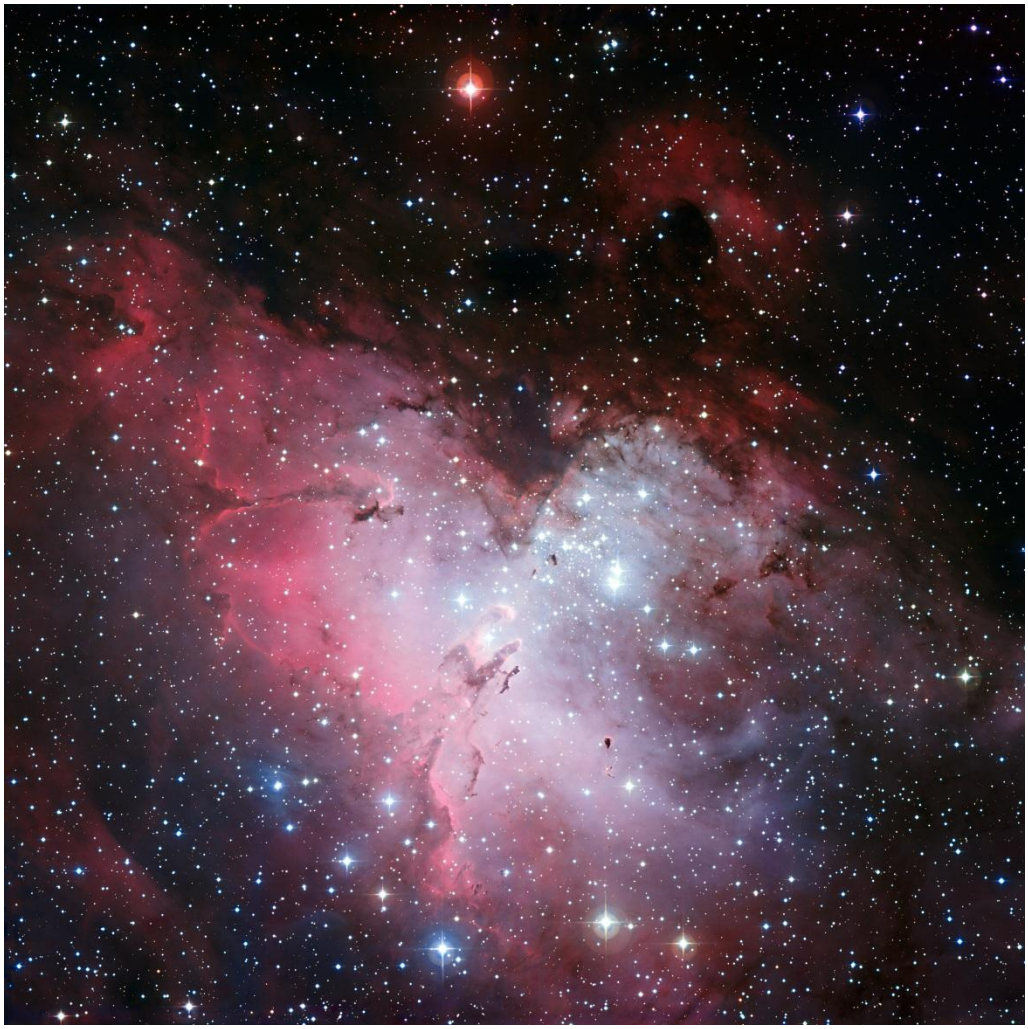


Figure 1: The Eagle nebula (by ESO [5]*)*

1.2 Aim

The aim of this research is to find a method to render diffuse emission nebulae in an efficient way while producing realistic visuals. The rendering method should allow real-time interaction allowing users to explore the nebula. This method will calculate the emission from gas based on the ionization from hot stars in the nebulae and the distribution of the emitted colors.

1.3 Related work

There has been some previous work in the area of visualizing nebulae. In 2001 Nadeau et al. [6] created a visualization technique for emission nebulae. Their technique uses a 3D mesh constructed from actual data collected from the Orion nebula. However, rendering a 2 minute flythrough of the nebula took 12 hours on a supercomputer with over 900 cores which means that this method is not appropriate for interactive visualization.

Later, Magnor et al. worked with reconstructing and rendering planetary nebula [2] [7]. Their work allows efficient rendering of emissive nebula on PC hardware [7]. This work uses a purely emissive model of the nebula and is limited to planetary nebulae. Their work does not calculate the emission color or the effect of gas density on which elements are ionized. Instead they calculate the emission color based on images taken of the nebulae they are basing their rendering on.

Magnor et al. has also worked on reflection nebulae [3]. This model visualizes dust that is directly lit by the nearby stars and is based on physical models from astrophysics research. Their method runs at 7.5 frames per second on a 2005 top of the line graphics card which is almost too low for user interaction. This method calculates the reflected light in the nebula and its scattering through the volume. They use a Monte-Carlo simulation to create a pre-computed table of light scattering probabilities. Unlike other work in the area, they do not use real world data in their model; instead they use Perlin noise to generate the volumetric data.

Perlin noise is a procedural algorithm that generates a pseudo-random volume of values [8] [9]. The Perlin noise function generates values based on a set of coordinates that is fed to its input. Perlin noise is coherent, that means that there are no discontinuities in the generated values and the result appears smooth. By multiplying the input coordinates, a higher frequency of noise can be obtained. By multiplying the resulting values the noise can be made smaller, changing its amplitude. By combining noise created with different frequencies and amplitudes it is possible to generate noise volumes with many different shapes and forms. By adding more high frequency noise the result will become more turbulent while a low frequency noise will create large shapes.

A model for rendering volumes with both emission and absorption has been devised as can be seen in the paper by Max [4]. This can also be extended to calculate both single scattering and multiple scattering. Scattering does not, however, contribute much to the visuals of emission nebulae as the light emitted is much weaker than that of the stars in reflection nebulae [1]. The rendering method proposed by Magnor et al. for light scattering is unfortunately very slow even though it has been greatly optimized with a pre-computed table. It appears that the existing methods for calculating light scattering all have high execution times [4], which is a weakness when using such complex calculations.

2 Theory

2.1 Emission Nebulae

Diffuse emission nebulae (or H II regions) are large regions of interstellar gas. These gas clouds contain many hot young stars that ionize the surrounding gas. The hot ionized gas expands into the dense neutral gas, hollowing the nebula from within.

The high energy ultraviolet radiation from the hot young stars in the nebula ionizes the atoms of the surrounding gas (an ion is an atom that has lost one or more electrons). The electrons knocked loose from the ions will recombine with other ions and are recaptured at a higher energy level. As the electrons decay to lower energy levels they emit new photons at specific wavelengths. The different wavelengths emitted by different elements lad to a variety of colors in the emitted light [1].

At larger distances from the star(s) more of the high energy photons will have been absorbed into the gas leaving only lower energy photons to ionize it. These lower energy photons will ionize different elements. Further out, beyond the central high energy zone the now lower energy photons will ionize oxygen making it emit light at a specific wavelength and color. Hydrogen will still be highly ionized in this region and its emission weak. When even more energy has been absorbed, hydrogen will be less ionized and the emission from hydrogen will increase while the emission from oxygen fades out. At the boundary between ionized and neutral gas nearly all the photons will have been absorbed and hydrogen will recombine to neutral atoms making the colors of hydrogen strong. At the neutral side of the boundary sulfur will emit strongly in its color. This boundary is called an ionization front. Only a nebula with sufficient density will contain an ionization front. The way the different elements emit light at different wavelengths makes the ionization fronts light up stronger due to the

combined color from oxygen, hydrogen and sulfur. The strong boundary in the front will also vary greatly in color, making it a distinctive feature in images of nebulae.

2.2 Volume rendering

The gas clouds of a nebula are inherently voluminous. Therefore it is appropriate to render nebulae as a volume instead of polygonal surfaces. An appropriate technique for volume rendering is ray casting [10]. Ray casting works by casting rays through the data set (volume) and sampling the volume values at regular intervals along the ray. One ray is cast for each pixel on the screen and the encountered values are evaluated to determine what color the screen pixel should have.

To perform ray casting it is necessary to know the entry and exit points of the rays in the volume. For each ray an intersection test is performed with a box representing the volume. If the ray misses the box no rendering needs to be performed but if the ray intersects it the entry and exit points are stored. These values are used to calculate a ray that can be used to step through the volume, sampling it along the way. A sampling rate needs to be set to determine the length of each step and how many steps are necessary to sample through the entire volume. A lower sampling rate will require fewer calculations but will also increase aliasing and risks missing small details [10].

3 Method

To render the nebulae a volume renderer was implemented. Several generated nebulae were rendered to test the program. The renderer was implemented with OpenCL [11] and C++. The method used for rendering the nebulae is divided into three steps: nebula generation, emission color calculation and volume rendering.

3.1 Nebula generation

As real gas distribution in nebulae is hard to accurately determine and because it is interesting to render nebulae from other viewpoints than Earth a procedural approach was used to generate the gas distribution. For each of the three primary elements - hydrogen, oxygen and sulfur - a noise function is used to generate the distribution of the gas.

The Perlin noise function [8] [9] is evaluated multiple times, each one with a higher frequency and lower amplitude until the next step will be too small to see (the noise takes up less than two pixels). The noise is normalized to the range 0 to +1 to give a uniform distribution of gas. For the neutral gas the noise is generated in the same way except the negative values are clipped, giving a gas distribution with holes and cavities. This simulates the effect of the ionized gas expanding into the surrounding neutral gas.

To allow the generation of many different nebulae a seed value is provided to the program. This is an integer that is added to the input coordinate of the Perlin noise function. The seed is bit shifted to the left one step further for each of the elements and the neutral gas, giving different noises for all of them. The result of the four noise functions is stored respectively in the four channels of a 3D texture. To avoid the nebula appearing cubical all values are diminished with distance from the center and fade out in a spherical fashion so the corners of the cube are all dark.

3.2 Emission

To improve rendering performance the calculation of emission color is performed separately before a nebula is rendered. The emission color and intensity is primarily

dependent on the optical depth (accumulated gas density) between the ionizing star(s) and the ions in the gas.

The gas density has previously been calculated and was stored in a 3D texture. This texture is now available for use to calculate the emission from the gas. By using OpenCL's built-in 3D textures it is possible to automatically interpolate the volume when it is sampled. To allow rendering of the nebula from any direction it is necessary to store the emission colors in a volume that can be rendered later. It is therefore easy to simply evaluate each voxel in the gas density volume and store the calculated color at the same coordinate in a volume of the same dimensions as the density volume.

To calculate the color of each voxel the accumulated depth between each voxel and the ionizing star(s) is needed. To simplify the implementation only one star or a star cluster is considered in this work. The position of the star is specified in relation to the center of the nebula. For each voxel a ray is cast toward the stars position and the gas density of all four texture channels is sampled and summed along the ray. A step size of 1/400 of length of the volume sides is used. In this program the volume is considered as a box with corners in (-1 -1 -1) and (1 1 1). As the accumulated density values become very large the samples from three elements are divided by 1000. The neutral gas samples are divided by 10 in order to make the ionization fronts appear at the density increase at the boundary of the neutral and ionized gas. The accumulated density then needs to be evaluated to see how much emission each element will contribute.

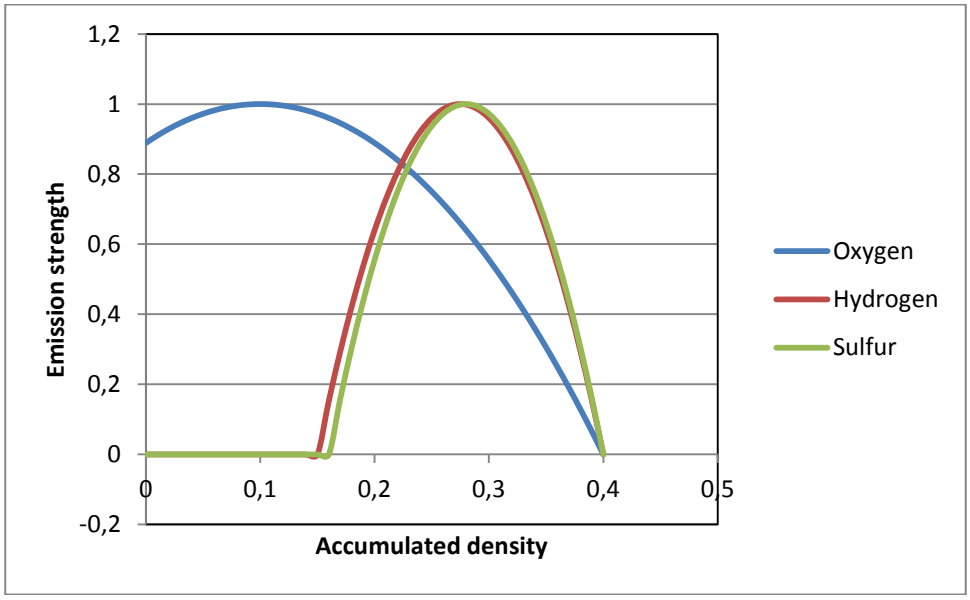


Figure 2: Emission strength for each element as a function of the accumulated density.

$$f(x) = \begin{cases} 1 - \left(\left(x - \left(a + \frac{b-a}{2} \right) \right) * \frac{2}{b-a} \right)^2, & a > x > b \\ 0, & a \leq x \leq b \end{cases}$$

Figure 3: The function for determining emission strength. This is a curved ramp between the constants 'a' and 'b'.

To make the rendering easier a simple formula has been devised for determining emission strength. This formula is based on the fact that the emission strength of each element is primarily dependent on the optical depth to the star. The order of the elements as they appear with increasing distance from the star is accurate but the actual values used to determine the onset of a new element were found using a primarily trial and error approach based on images of real nebulae. In the function *x*

represents the optical depth, or the accumulated density in a point of the gas cloud. The values *a* and *b* represent the start and end point of the curved ramp.

Oxygen is the first element to contribute visual emission around the star. Further out, at the potential ionization front, hydrogen and sulfur start to emit light. The following *a* and *b* values were chosen to represent the three elements: oxygen: -0.2, 0.4; hydrogen: 0.15, 0.4 and sulfur: 0.16, 0.4. The emission strength is used to determine the amount of each color in each point of the nebula volume. The emission strength of each element is then multiplied with the stored density of that element, allowing for each element to vary in density through the nebula.

The color associated with each element can either be the actual emitted color as seen by a telescope or a set of 'fake' colors, often used in nebula images. While testing the program two sets of colors were used: blue, green and red for oxygen, hydrogen and sulfur respectively, or just red as the natural color of hydrogen.

Finally, the density of the neutral gas is copied from the sampled texture into the new emission volume.

3.3 Volume rendering

The volume containing the emission colors will need to be rendered to the screen. The emission color volume is loaded as an OpenCL 3D texture. Each pixel on the screen has a ray cast from the viewpoint in the current view direction. An intersection test is performed on the rays to see if they pass through the volume. For all the rays that pass the volume the entry and exit points are used to calculate the ray path through the volume. Each ray starts at its entry point in the volume and is stepped towards the exit point. For each step through the volume a color is sampled and blended with the previous color. The value of the neutral gas is used as an alpha value in the blending, making dense regions of neutral gas turn opaque, blocking the light emitted behind it.

The step size used when stepping through the volume has an impact on runtime performance and also on rendering quality. A good step size balancing performance and quality is 1/200 of the volumes side. A larger step size will lead to aliasing and a smaller will not visually improve the rendering.

Due to the many samples taken the accumulated color values become very large. To avoid saturating the colors each color channel is multiplied by 0.016 when sampled and the alpha value is multiplied by 0.02. A larger alpha value will make the neutral gas more opaque, adding dark clouds to the rendering.

4 Result

The nebula rendering program that was implemented can be used to render a variety of simulated nebulae from any viewpoint. The rendering algorithm calculates ionization of the gas from a star cluster inside a nebula. The spectral color of the emission depends on the depth between each voxel and the illuminating stars (with depth meaning the accumulated density of gas and dust) and the density of each of the elements: hydrogen, oxygen and sulfur.

It can be seen that the color in the renderings varies in a way very similar to real nebula with a large volume of emissive oxygen near the center shifting into hydrogen as the distance to the stars increase, finally shifting again into sulfur at the largest distances from the stars before fading away as the radiation from the stars has been absorbed or weakened by distance. Ionization fronts are visible in the shift between hydrogen and sulfur. These sharp structures are formed when the density of gas rapidly increases in the boundary between ionized and neutral gas.

Each of the three primary elements found in nebulae (hydrogen, oxygen and sulfur or nitrogen [1]) can be mapped to different colors, allowing for either true

visible color or a different, ‘fake’ color scheme designed to make a nebula easier to observe and study.

Instead of using the physical equations for ionization and recombination of the elements in the nebulae a simpler method is used. This method simplifies the emission from each element to a function based on accumulated depth. This gives a “good-enough” approximation of the color variation through a nebula.

By varying the viewpoint around and inside the simulated nebula it can be studied both from an earth-like position and from the cloud structures of the expansion boundary and ionization front. Figure 6 shows the view from close to the ionization front in one of the simulated nebulae, showing some of the detail of the cloud like structures.

Figure 4 illustrates a nebula seen from a short distance with several ionization fronts visible and some obscuring neutral gas in the foreground. Due to the way the nebula volumes are generated it is possible to create a wide variety of nebulae with this program. By changing the seed value used to generate the volume many different dust distributions can be simulated. Not all seed values are however equally good. Depending on the chosen position of the ionizing star the gas may be most dense right on the star, making it only ionize a small dot before the density is too great and the emission fades out.

As the program precalculates the emission color the renderer only needs to perform a basic volume rendering each frame. This means that the program can work under the same performance conditions as a normal ray caster. The images in Figures 4 through 7 were taken at a screen resolution of 1024x768 and the rendering program ran at approximately 55 frames per second on an Nvidia GeForce 460 series graphics card.

5 Discussion

The nebula rendering program generates very beautiful pictures that match the features of real nebulae. When compared to real nebulae (Figure 1) it can be seen that this method rendering do not capture all the small details present in real nebulae. In order to render all the details a larger volume is required for the gas densities. As this program is implemented in OpenCL and designed to work on the graphics card a larger volume than 256^3 is not practical due to the memory size of a normal graphics card. There is room for significant improvement of the rendered images if a larger volume is used but this is not a shortcoming of the volume itself.

If more processing power were to be available it would be possible to eschew both the precalculation of emission color and the storage of gas densities by computing these values each frame. Currently this would however lead to the program becoming unresponsive to user input and the aim of this work was to find a rendering method that allows interaction.

The simplified rendering method that is proposed here looks good and appears to match actual nebulae quite closely. It is however not empirically tested to match real phenomena in nebulae and is not designed for physical correctness. An improved method could be devised to use physical equations to calculate the emission strength of the gas but this would require a much more complex calculations to make it work with all the details and features that make nebulae special. Even though this is a simplified model it is able to show a lot of detail such as the fragmented surface of ionization fronts.

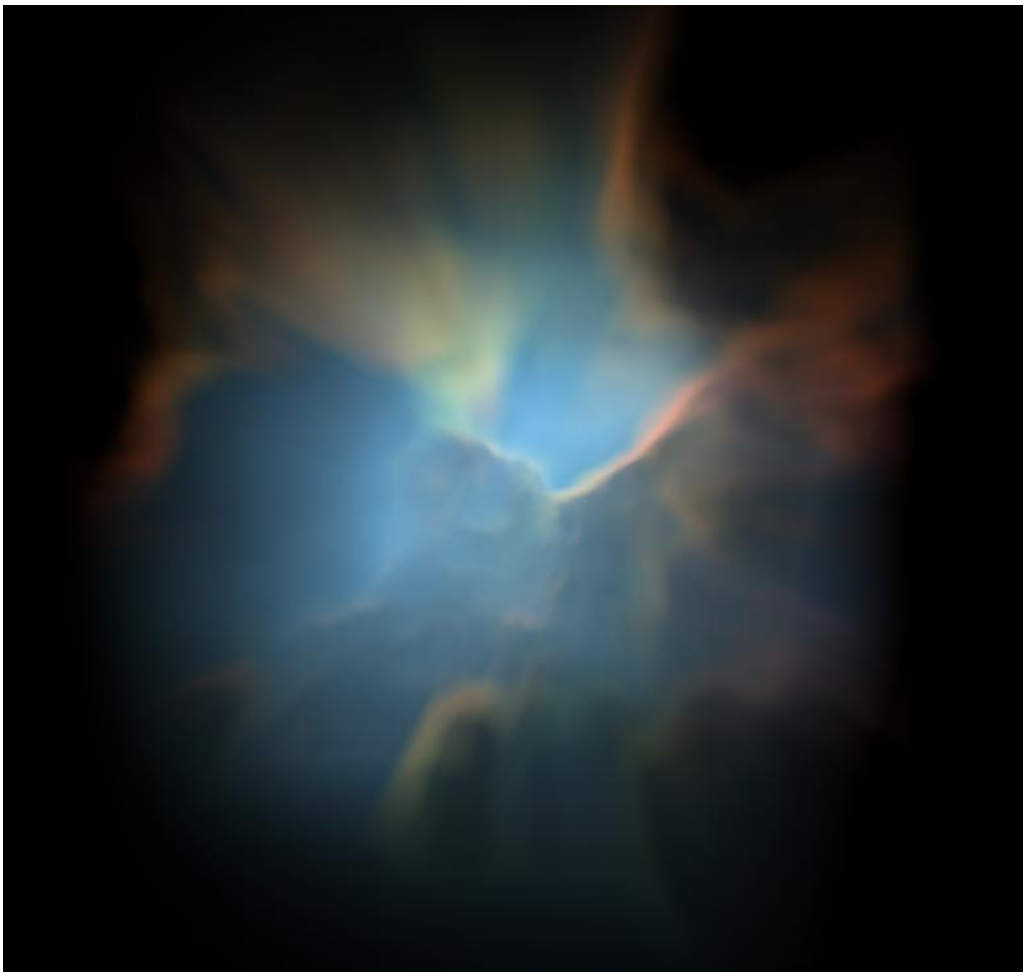


Figure 4: A rendered nebula with several ionization fronts visible through the blue oxygen.

Figure 5: High density gas makes this nebula glow mostly in the color of red (sulfur) and green (hydrogen).

Figure 6: Close-up of the interior of a nebula showing the ionization front behind a haze of oxygen.

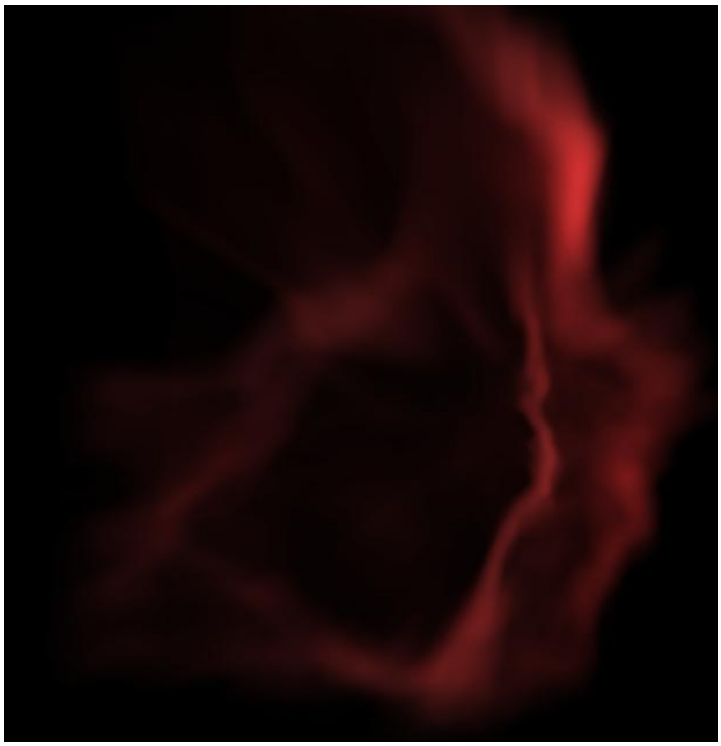


Figure 7: A hydrogen nebula in its natural dark red color.

A disadvantage of the simplified emission calculation is the long streaks or rays that are visible in the upper region of Figure 4 where the ionizing star shines along the edge of the ionization front. This along with the blurriness of the close-up in Figure 6 is primarily dependent on the resolution of the volume but also on the way the emission function has been simplified. A potential improvement to remove some of the rays would be to jitter the sample positions in the emission step slightly, introducing some randomness.

The runtime performance of the program is more than sufficient for user interaction. Because of the precalculation step the performance is only dependent on the volume renderer. As the volume renderer does not differ from other volume renderers, performance depends primarily on screen resolution and sampling rate. The time for the precalculation is insignificant on the computer used for testing and restarting the program with a new seed value is not a problem.

A potential improvement of the program would be to implement a tone mapper or allowing the user to adjust image brightness and contrast depending on what part of a nebula is rendered. Some images can be oversaturated which makes it difficult to make out detail. In other cases it might be interesting for the user to change the brightness level in order to see features deeper in the nebula, behind obscuring gas. An example of this can be seen in Figure 6 where it could be interesting to see more detail in the distant ionization front.

6 Conclusion

The aim of this work was to find a method for rendering diffuse emission nebulae in an efficient manner. The method was to allow real-time interaction and enable users to explore nebulae.

This work has proposed a method for rendering emission nebulae. This method is visually accurate but is not physically correct. The method fully captures the important nebulae features such as ionization fronts and the distribution of emitted light colors. The rendering method is based on the ionization of the elements in a nebula and uses this as a base to determine the strength of the emission from the elements in the gas.

The rendering is efficient on modern PC hardware and works fast enough to allow full interaction. The rendering program created allows a user to fully explore a nebula from any viewpoint, both from outside and from inside the gas clouds.

An improved version of the renderer with support for higher resolution volumes could be used to generate images and videos of nebulae usable as realistic illustrations or just as beautiful images.

References

[1] D E Osterbrock and J G Ferland, *Astrophysics of Gaseous Nebulae and Active Galactic Nuclei*, 2nd ed. Sausalito CA, USA: University Science Books, 2006.

[2] M Magnor, G Kindlmann, C Hansen, and N Duric, "Reconstruction and visualization of planetary nebulae," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 5, pp. 485-496, September-October 2005.

[3] M A Magnor, K Hildebrand, A Lintu, and A J Hanson, "Reflection nebula visualization," in *IEEE Visualization 2005, Proceedings*, Minneapolis, October 2005, pp. 255-262.

[4] N Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99-108, June 1995.

[5] (2011) ESO - eso926a - The Eagle Nebula. [Online]. http://www.eso.org/public/images/eso0926a/

[6] D R Nadeu et al., "Visualizing Stars and Emission Nebulas," *Computer Graphics Forum*, vol. 20, no. 1, pp. 27-33, March 2001.

[7] M Magnor, G Kindlmann, N Duric, and C Hansen, "Constrained inverse volume rendering for planetary nebulae," in *IEEE Visualization 2004, Proceedings*, Austin, 2004, pp. 83-90.

[8] K Perlin, "An image synthesizer," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques (SIGGRAPH '85)*, San Francisco, 1985, pp. 287 - 296.

[9] K Perlin, "Improving noise," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*, San Antonio, 2002, pp. 681 - 682.

[10] W Schroeder, K Martin, and B Lorensen, *The Visualization Toolkit, An object oriented approach to 3D graphics*, 4th ed. Clifton Park NY, USA: Kitware, 2006.

[11] (2011) OpenCL. [Online]. http://www.khronos.org/opencl/