Volumetric Lighting

Ryan Heffernan and Brendan Rufo

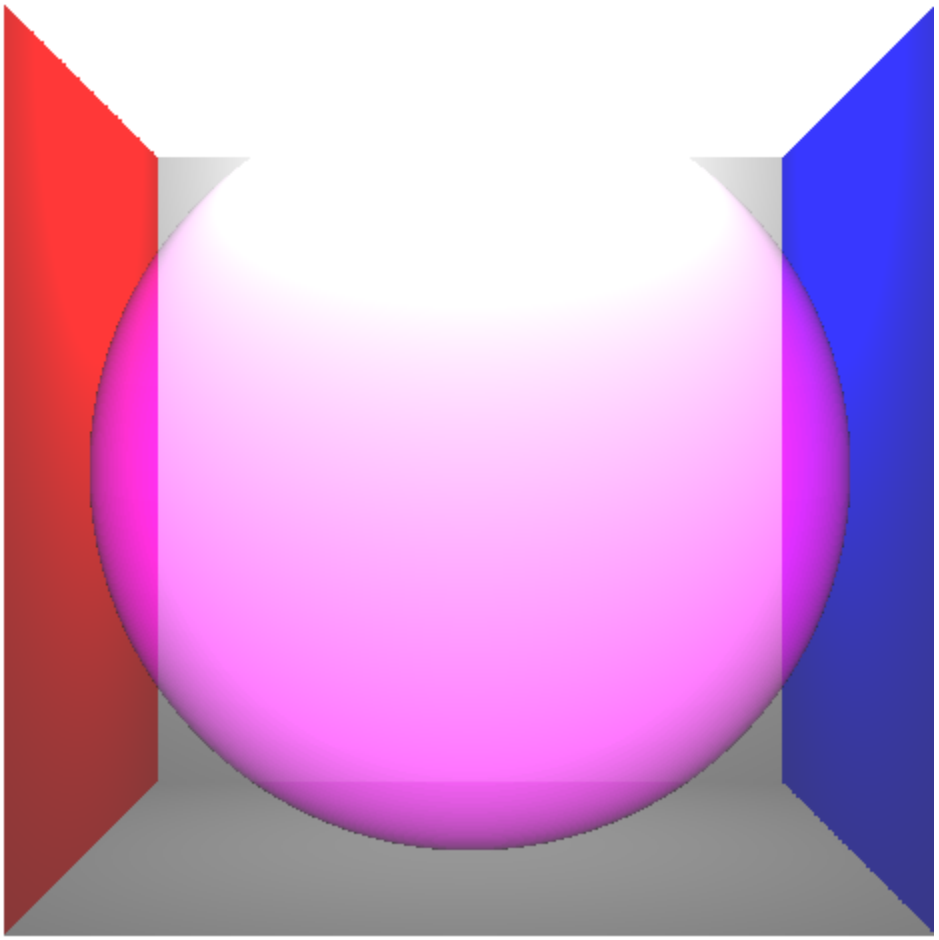


Figure 1: an example of a pink volume density placed inside the Cornell box.

Abstract

Described in this paper is our implementation of Volumetric Lighting. Our approach involved the representation of participating media within volume densities. The resulting implementation is an efficient rendering of volumetric lighting effects using a combination of ray tracing and ray marching techniques. It opts to sacrifice the precision of scattering in exchange for faster rendering and implementational ease to allow for rapid ideation of scenes requiring participating media.

1. Motivation

1.1 Decay of Artificial Light

It was first suggested to one of the authors that the focus of this project should be the Decay of Artificial Light. Ryan Heffernan's partner is an artist who explained the concept of lighting in drawings to get gradually darker the further away something is from light in the scene. While DAL was fascinating, it is technically simplistic. Because of this, there is little prior work on the topic and would not make for a good paper. The original vision for a final production of that project would be to render a dark scene illuminated only by a weak light source, like a single flash light illuminating a dark hallway. When you start to break down that scene, however, there is another lighting effect at play that gives you the eerie vibe; the beams of light from the flashlight being visible due to dust particles in the air. With the same final production in mind, we decided to make our project focus on Volumetric Lighting.

1.2 Volumetric Lighting

There are a few ways to approach rendering volumetric lighting in computer graphics. The approach that we decided to take was to model the participating media in the scene and render that. Participating media refers to particles in the air that can interact with light in the scene by absorbing, emitting, or scattering. For example, the participating media in the aforementioned dusty hallway scene would be the dust particles floating in the air. This particular example led us to model our volumetric lighting with discrete media volumes, as opposed to rendering the entire scene assuming that the media occupies all empty space.

2. Related Work

There was no shortage of scholarly works on Volumetric Lighting to look upon. There were two main sources of prior works we looked at while doing preliminary research. The less useful of the two types were physics based papers that focused on light scattering, and how photons interact with participating media. The useful types were other papers in the field of computer graphics that discuss how to model light scattering in a manner that was both understandable and useful to this project.

The paper by Kajiya and Von Herzon describes a method of rendering volume densities using light scattering equations with ray tracing. This provided great insight into the math required for achieving volumetric lighting. Novak et al is a great paper that describes achieving similar results using Monte Carlo ray tracing methods, but is more focused on modeling volumetric lighting using light transport

methods rather than participating media. Walczyk is less of a scholarly article and more of a “how-to” on ray marching, but was very useful in determining whether or not ray marching was the right method for us to use in this project.

A previous student project from 2017 by Gemmell and Maicus was very helpful in our understanding of volumetric lighting. From their project we were able to get an idea about where we should start with our implementation, and was a gentle guidance to our project.

3. Implementation

3.1 Representation of Participating Media

Our initial attempts at participating media included using a particle system to keep track of the various particles relevant to the participating media. While this would have allowed us to handle heterogeneous media with ease, we quickly ran into some computational concerns and shifted to using Volume Densities to model media. In order to create robust volume densities, it became necessary to implement a method for importing the location and properties of the various densities. As such, we implemented a plain text file reader that allowed us to read the center, radius, and color of the volume density, as well as the Absorbance, Outscattering, Emittance, and Inscattering of the volume. In addition to storing these values, our Volume_Density object can be used to check if it is intersecting with any rays being cast from the camera or if an arbitrary point is within the volume. It additionally computes the square of its radius the first time it is asked, which it then stores for the remainder of runtime to speed up intersection calculation. In exchange for exact representation, the volume density solution allows for much less computation and similar results.

3.2 Ray Marching

At first we sought to use Ray Marching to render our scene. Using the method described by Walczyk, our goal was to march through the scene using the distance of the closest object as our step distance each time we marched, also known as distance aided ray marching. The thing about distance aided ray marching, however, is that it is trivialized when you are able to efficiently find the closest intersection of your ray at every step. Because our implementation is an extension of the homework three framework, we have access to the CastRay function which allows us to find this point of closest intersection. Because of this, we made the decision to hybridize distance aided ray marching.

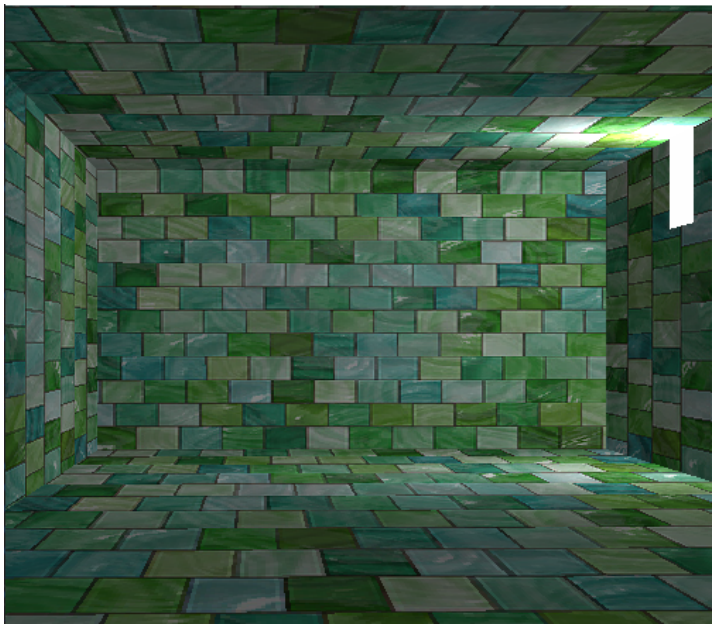


Figure 2: Our final hallway scene rendered with textured planes and without any volume densities

In simplified terms, we raytrace until we intersect with a participating media. From there, we shoot a ray through the media in the direction towards the target pixel, and we shoot a ray towards each light source. These auxiliary rays we use ray marching on, which combined will give us our final answer for what color that pixel should be as well as allowing us to get the volumetric lighting effect.

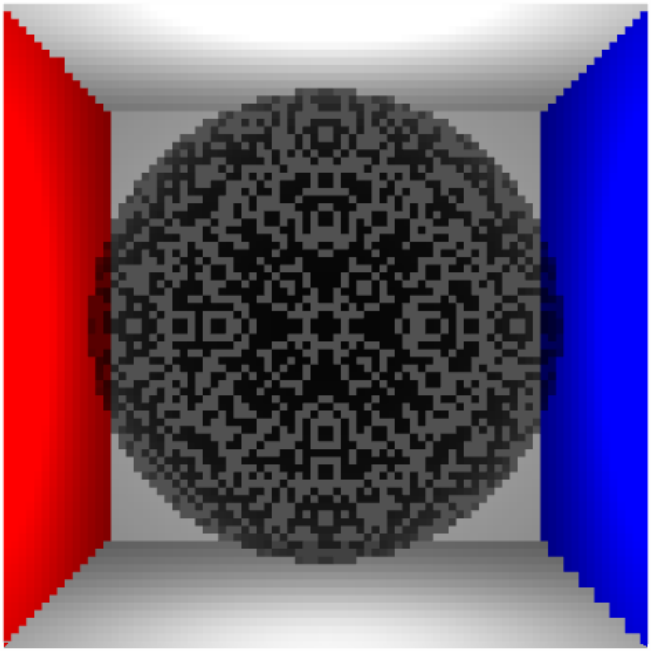


Figure 3: Erroneous media representation rendered with ray marching in Cornell Box

A simple bug which produced an accidentally very interesting pattern in the participating media occurred once we had detected intersection with participating media and moved to ray marching, epsilon error led to incorrectly calculating as no longer being inside the media, resulting in some portions darkening based on the depth of volume, while others immediately exited the ray marching portion.

3.3 Lighting

Once basic Ray Marching and Volume Densities were implemented, the next step was to actually allow for the lights present in the scene to interact with the traits of the participating media during the ray tracing of the scene. This was accomplished using the two ray marches that occurred after intersecting with the participating media, one deeper into the volume along the original ray, and one toward the light source. The marching along the original ray's trajectory is used to determine how much of the participating media's color contributes to the color of the pixel, which considers the stored Absorbance value, and the marching towards the light source is used to determine how much brightness the pixel should receive from the light source, which considers the stored Inscattering value. In Figures 3 and, for example, the bottom right density is notably different colors in the two figures. In Figure 3, the volume is immediately under the top right density, and the rays march through both it and the top right density which greatly reduces the brightness of the resulting pixel. In Figure 4, the top right density is offset to specifically avoid the darkening of the lower density, allowing the red and purple to be revealed.

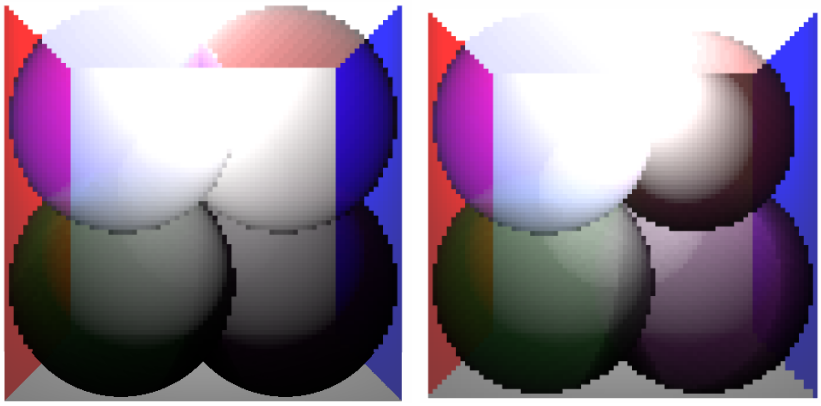


Figure 4, 5: Four volume densities of the colors blue, red, green, purple (left to right, top to bottom) in Cornell Box

We tested ray marching by using the Cornell box scene, and some modified variations of it, and the textured plane scene from the homework 3 ray tracing assignment, and seeing if the result was comparable when a volume density was present and when one was not. The modified variations of the

Cornell Box had their light source moved to the ceiling of the box. Some versions also included a darker back face to better highlight the appearance of white participating media, such as Figure 8.

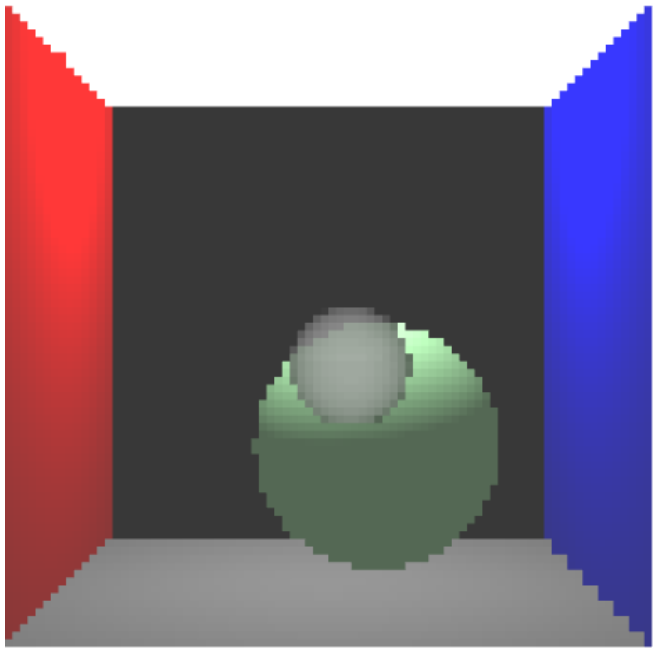


Figure 6: Participating Media incorrectly rendering in front of green diffuse sphere in modified Cornell Box

One long standing bug allowed for participating media to be rendered in scenes even when no portion of them should be visible from the scene, such as figure 5. The volume density is located at (0.25,-.5,-10), but is being rendered inside of the unit cube modified Cornell Box . This was due to not checking if the ray collided with a face prior to intersecting with the volume density. It was uncovered while trying to produce Figure 6.

4. Results

Our final product is a functional rendering of participating media to create volumetric lighting effects. We are able to render our scenes efficiently compared to other methods that exist. There are limitations to our implementation that will be discussed, but ultimately we are happy with the images we are able to produce. The implementation suffers from no measurable cost in the average case when compared to the rendering of the exact same scene ray traced without any participating media present.

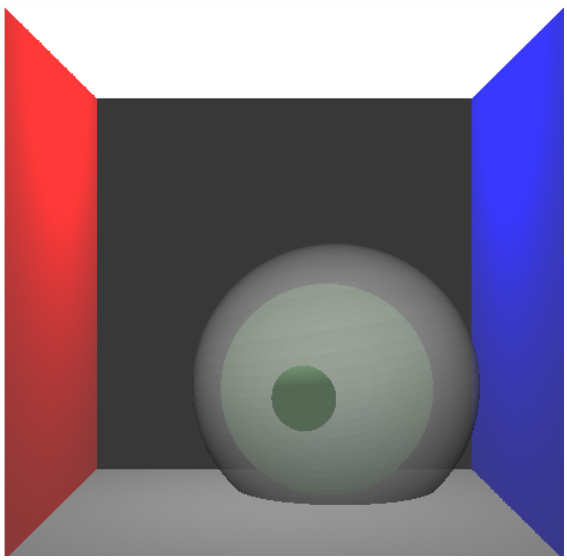


Figure 7: The green diffuse sphere in the modified Cornell Box partially peeks out of Participating Media

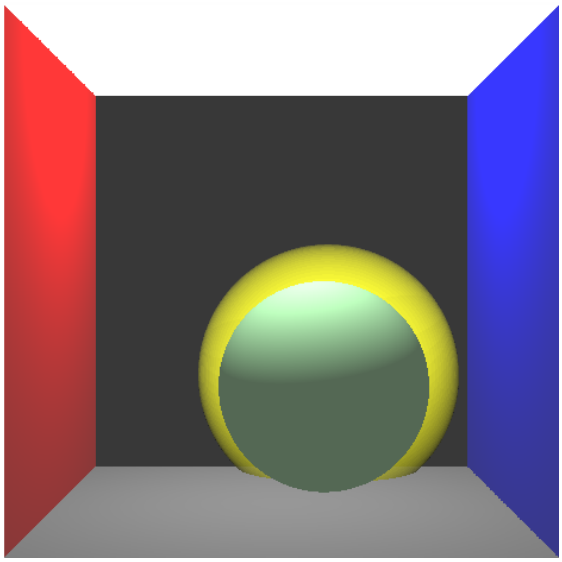


Figure 8: Green diffuse sphere in modified Cornell Box rendered with a yellow volume density behind it to give the image of a halo.

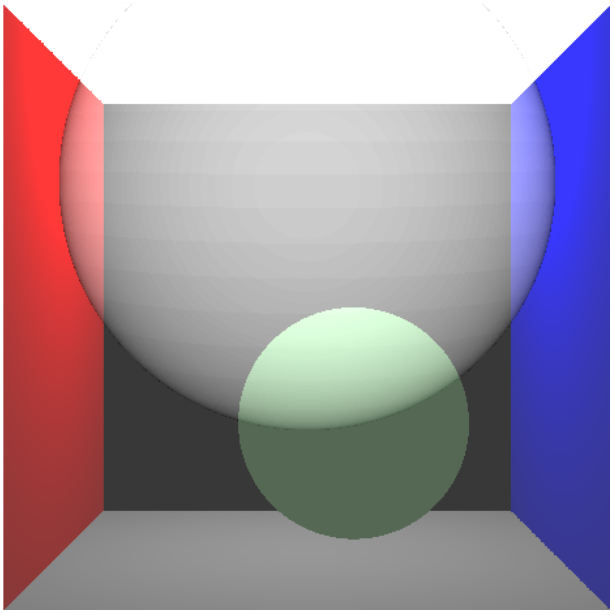


Figure 9: Green diffuse sphere partially behind participating media

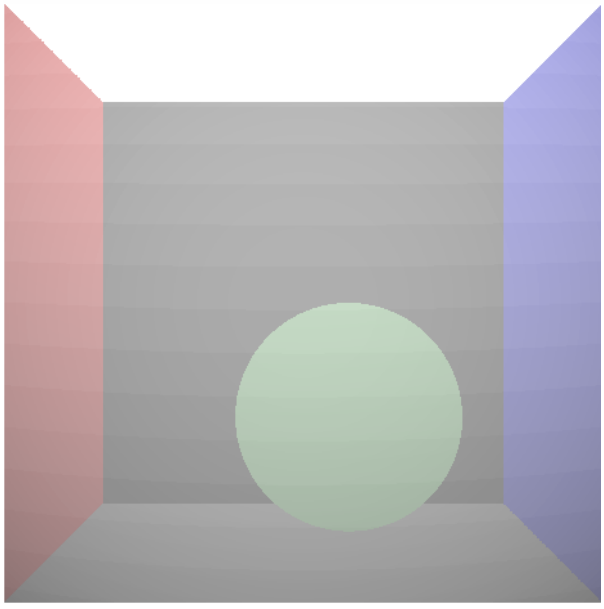


Figure 10: Green diffuse sphere entirely inside participating media

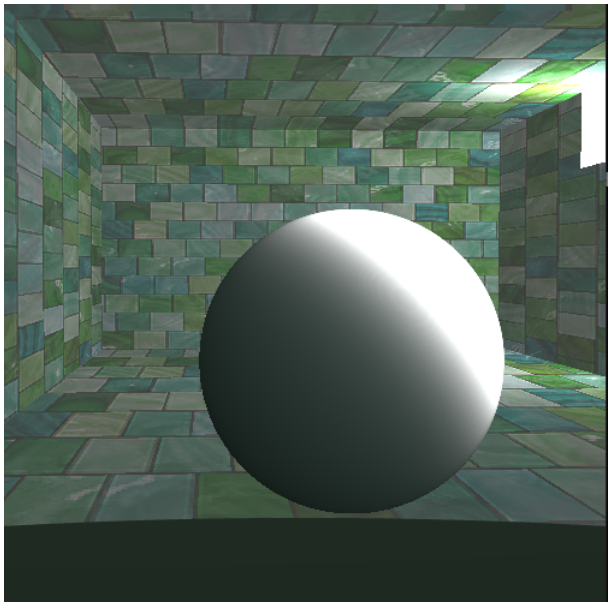
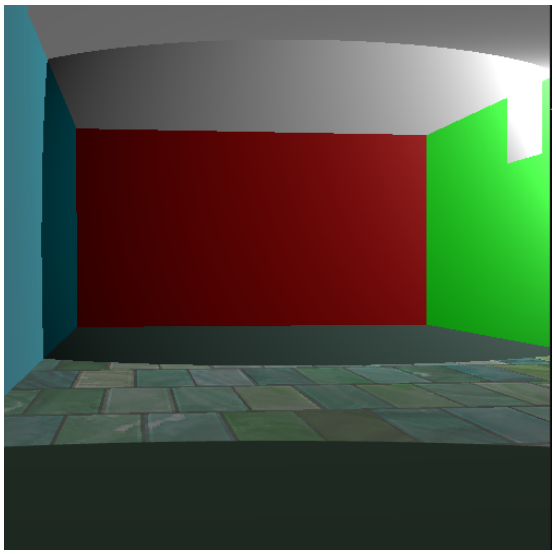


Figure 11: our final hallway scene rendered with a singular volume density. We discovered a compatibility issue with our volume densities and using textures, making the density appear extremely opaque.



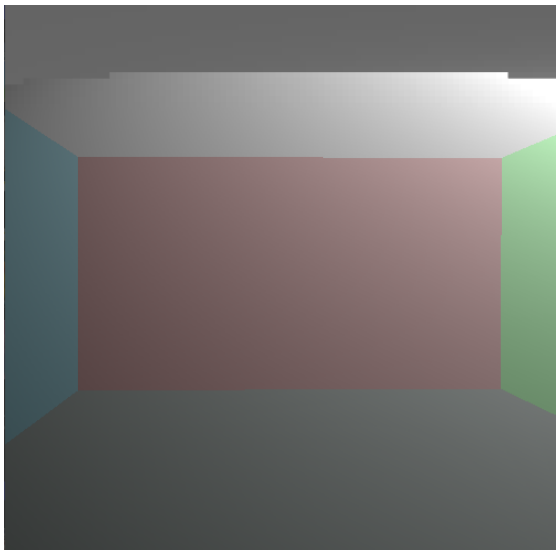


Figure 12, 13: the hallway scene re-rendered with a larger volume density and no textures on the walls.

5. Future Work

Because of the timeframe, there are some limitations of our project that we would like to improve on in order to create more realistic and aesthetically pleasing renderings.

One such limitation is the shape and attributes of our volume density. Currently, we only render using perfectly spherical volumes of participating media, though this does not accurately reflect how media occupies space in real life. Furthermore, we assume a perfectly uniform media, one that is uniformly dense and of a single particle type. Ideally we could render non-spherical densities of varying particle size, which would change how the light scatters as it passes through it. It is possible to work around some of these limitations by overlaying multiple volume densities with the desired traits. Figure 10 depicts one such example.

Another limitation of our product is the lack of indication of exactly 'where' in the scene a volume density is centered. Figures 3 and 4 highlight this confusion, as it would be impossible to understand the reason behind their differences without the provided explanation.

Most significantly, attempting to use some other aspects of raytracing, such as support for textures, is not continued in our implementation. Basic ray traced shadows are implemented, however, currently volume densities do not leave shadows.

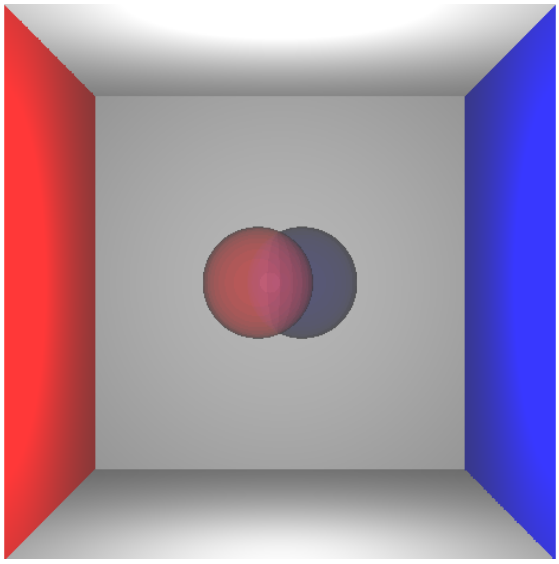


Figure 14: A Red and Blue volume density overlaid in a modified Cornell Box, showing a purple region in their overlap

Though not a limitation of our implementation, we would also like to implement the decay of artificial light that got us to this topic in the first place. This would probably have us consider irradiance caching, so that we might consider how light decays during interference from a participating media.

Adjacent to the technical implementation of this project was the creation of a final scene to render our participating media in. The hallway scene we used was created by a beginner 3D modeler and in the future we would like to spend much more time creating a scene that best shows off how our implementation works.

6. Team Workload

Both group members of this project worked on each portion, debugging or writing the bulk of the code.

Ryan Heffernan focused primarily on the ray marching portions of the project. He extended our starter code to allow for ray marching in the case of participating media, and was the one who came up with the implementation idea for the hybridized ray marching instead of using the distance aided method. In addition, Ryan was responsible for the creation of the hallway scene using blender.

Brendan Rufo implemented the representation of volume densities and was responsible for calculating what happens to the rays as they march through the media. He also extended our starter code to allow for the additional input of text files to represent the volume density, or densities, represented in the scene.

7. Conclusion

Ultimately we would call our project a success. Revisiting our core tasks, those being to implement ray marching in the case of participating media and to represent volume densities, we conclude that we have sufficiently met our expectations. There are definitely things we would like to improve on given we had more time, but we did complete what we set out to do, even if the methods we used weren't exactly what we imagined they would be. Spread mostly evenly over 4 weeks we spent approximately 30 hours working on this project.

8. References

Cerezo, Eva, et al. "A Survey on Participating Media Rendering Techniques." The Visual Computer, vol. 21, no. 5, 2005, pp. 303–328., https://doi.org/10.1007/s00371-005-0287-1.

Gemmell, Alec, and Evan Maicus. Volumetric Rendering of Participating Media - Computer Science. 2017, https://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S17/final_projects/alec_evan.pdf.

Kajiya, James T., and Brian P Von Herzen. “Ray Tracing Volume Densities.” ACM SIGGRAPH Computer Graphics, vol. 18, no. 3, 1984, pp. 165–174., https://doi.org/10.1145/964965.808594.

Novák, Jan, et al. “Monte Carlo Methods for Volumetric Light Transport Simulation.” Computer Graphics Forum, vol. 37, no. 2, 2018, pp. 551–576., https://doi.org/10.1111/cgf.13383.

Walczyk, Michael. “Ray Marching.” Michael Walczyk, https://michaelwalczyk.com/blog-ray-marching.html.