Modeling and Simualtion of Appearance Final Project: Clouds and other participating media

Pedro José Pérez García, 756642

1 Introduction and motivation

For this final project we have implemented a method to render participating media, both homogeneous and heterogeneous, bounded and unbounded, which is built on top of the path tracers we have developed during the subject. The final result allows the creation of unbounded homogeneous volumes (1 per scene), as well as both homogeneous and heterogeneous bounded volumes. In this report we showcase and detail the implemented features, along with some renders that illustrate the results.

Figure 1: Our final render, 4096 samples per pixel.

On the left we have our final render that shows the implemented features, which has been inspired on Figure 2, already shown for the Look and Dev session earlier in the course. This image features two clouds (bounded heterogeneous volumes loaded from `.vdb` files) and a global homogeneous volume, serving as a very thin atmospheric fog that makes everything look smoother the more distant it is with respect to the observer.

The diffuse elements were added in order to seek resemblance with the original image, and only serve aesthetic purposes (image composition, etc), so no focus was put into their materials and appearance.

2 Participating media: Quick overview

A volume is considered to be any kind of medium composed of particles that interact with light, changing the energy that a photon holds, or varying their trajectory. It is more an accumulation of these particles than an object by itself. The interaction between light and volumes produces several phenomena, like in-scattering, out-scattering, absorption and extinction. We can model them with the Radiative Transfer Equation (RTE, first mentioned by Kajiya [1]), as well as some Phase Function that models the direction the light will be scattered towards, depending on the medium, the position within it, and some other intrinsic characteristics (for examlpe, Henyey-Greenstein has a parameter that defines the isotropy of the function, g).

Since we cannot model each of this particles physically, we do it in a probabilistic way, defining coefficients that model the probability of one of the aforementioned phenomena happening, like μ_t, the probability of extinction per meter. Here's where the concept of transmittance between two points $Tr(x_o \longleftrightarrow x_z)$ appears. We can define transmittance as the amount of light that goes unaffected through a medium, and we can model it using the Beer-Lambert law, which states that $Tr(x_o \longleftrightarrow x_z) = e^{-\int_{x_o}^{x_z} \mu_t(x_s)\, dx_s}$. Since participating media can be homogeneous, meaning that the density (usually the same as extinction probability) of the medium is constant through it, but it's much more common in nature to find heterogeneous mediums, like clouds or smoke. In this case, density varies through the volume. So we have to estimate the integral for the transmittance since we no longer have a constant value that trivializes the calculation. Similarly, heterogeneous media also makes sampling distances more difficult. For this, in this project we use null-collision techniques (in use since 2008 [2]), like Woodcock tracking [3][4][5] (also known as delta tracking) for the distance estimation, and Ratio Tracking [6] for the transmittance estimation.

3 Volumetric Path Tracing

We tried to create our own iterative version of the algorithm but after encountering some small bugs, we also got inspiration from the one in the PBR Book [7] in some parts of the implementation.

First, it is important to comment one important design choice we made at the beginning of the implementation. We have decided that the best way to implement volumes is that, hierarchically speaking, they are children to a mesh, that will act as their bounding box. This is different from what we have seen in the PBR Book, which defines interfaces that go from one medium to another, but they are limited to 1:1 medium changes, while in ours, one volume can touch several others, the only limitation is the geometry of the scene. The only case where we don't require a volume to be parented by a mesh is in the global one, which will be homogeneous and infinite. There can only be one of this type per scene. If we create more than one, the `.xml` file parsing process will fail, or ignore the following global volumes. Always requiring a mesh to parent a volume comes very handy when performing null-collision techniques, so it felt like the right approach. We also implemented two phase functions for our volumes, Henyey-Greenstein and Rayleigh. Some illustrative renders about this topic can be found in Appendix B.

3.1 Through one volume: Woodcock and Ratio Tracking

For performing the mean free path sampling within a volume, we can do it directly in homogeneous mediums, but we require special techniques in heterogeneous mediums. One of them is the Woodcock tracking. This null-collision technique aims to simplify the fact that $p(t) \propto Tr(x_o \longleftrightarrow x_t)$ and we can't estimate the transmittance when we have heterogeneous values for $\mu_t(x_s)$ for every different x_s between our two points, x_o and x_t.

For that, we create a majorant, $\hat{\mu}$, which will be constant through all the volume. In our case we have chosen the highest value of any of the channels for μ_t. Then, we sample small steps using the majorant, in the same way we would do it with a homogeneous medium. Lastly, we have to decide if the steps are real or null. For this we can directly use the coefficient between μ_t and the majorant $\hat{\mu}$ and a random number, ξ. If a step isn't real, we "walk" in that direction and sample another step (equivalent to saying that we have a delta phase function in the null parts of the volume), until we get out of the volume's bounds (then the bound is the final sampled point) or until we get a real collision, determined when $\xi < \frac{\mu_t(x)}{\hat{\mu}}$. This method is performed in `samplePathStep()` in `volume_vdb.cpp` for homogeneous media, and calls the homonimous function in `volumedatabase.cpp`.

We can also estimate the transmittance between two points x_o and x_z by iteratively running samples in the same way starting in x_o, but this time we also accumulate a product of ratios, $\prod \frac{\mu_n(x)}{\hat{\mu}}$, until we surpass x_z. The resulting throughput of ratios is the estimation of the transmittance. The corresponding code is located in the `ratioTracking()` function, in the `volume_vdb.cpp` file. This function is needed mostly by the `transmittance()` function in the same file, and is mostly called when tracing shadow rays that traverse several volumes (and then those individual transmittances get multiplied in `volumetric_transmittance()`, in the integrator file, `path_mis_participating.cpp`).

3.2 Through several volumes: In which volume are we now?

Once we have the ability to render heterogeneous volumes, we need to bound them. This process has two steps: First, we have to create the bounding boxes for the volumes, the meshes we mentioned earlier. We discuss this in the next section. The other part of the process involves tracking which volume is our path currently inside. We achieve this by using two pointers that tell us which is the current volume, and another one that points to the volume that we will be inside for the next iteration.

This imposes one restriction, we cannot have intersecting volumes. Although we tried to, keeping track of that was too complex for the time available and we had to discard it. In case of implementing it, we probably could have done it by restricting the volumes to be axis-aligned and running something like the Kay and Kajiya Algorithm [8], and when in an intersection, sample one of those volumes uniformly or weighted, based on their spatial densities.

There is one special case, though. When casting shadow rays towards the light sources we have to note the different volumes that ray goes through, and for every segment, approximate the transmittance. Finally, we compute the transmittance of the entire ray by multiplying all of the k transmittances for each segment: $Tr(x_0 \longleftrightarrow x_k) = \prod_i^k Tr(x_i \longleftrightarrow x_{i+1})$

4 Measured materials: Loading volumes from files

The final step to achieve the desired scene wasn't rendering related, but required some implementation work. We wanted to load real volumetric files, contained in `.vdb` files.

For this purpose, we tried to make use of the OpenVDB [9], but unfortunately we couldn't compile it with Nori. Instead of giving up, we discovered a repository originally thought for Mitsuba [10], that converts those files into `.vol` ones, a Mitsuba-specific format [11], but easy to understand and to load. So we actually load those files after being converted with the code from that repository.

Finally, we use the data inside the `.vol` files to get the min and max points of the bounding box for the volumes, and we coded a simple Blender add-on [12] that creates a cuboid that acts as bounding box, and only has 8 vertices. It is also necessary to slightly edit the `.xml` scene file to account for the volume transform, which is outputted in the terminal by the add-on to make things easier.

References

[1] J. Kajiya and B. von herzen, "Ray tracing volume densities," *ACM SIGGRAPH Computer Graphics*, vol. 18, pp. 165–174, 07 1984.

[2] M. Raab, D. Seibert, and A. Keller, "Unbiased global illumination with participating media," in *Monte Carlo and Quasi-Monte Carlo Methods 2006*, A. Keller, S. Heinrich, and H. Niederreiter, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 591–605.

[3] E. Woodcock, T. Murphy, P. Hemmings, and S. Longworth, "Techniques used in the gem code for monte carlo neutronics calculations in reactors and other systems of complex geometry," in *Proc. Conf. Applications of Computing Methods to Reactor Problems*, vol. 557, no. 2. Argonne National Laboratory, 1965.

[4] M. Galtier, S. Blanco, J. Dauchet, M. El Hafi, V. Eymet, R. Fournier, M. Roger, C. Spiesser, and G. Terrée, "Radiative transfer and spectroscopic databases: A line-sampling monte carlo approach," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 172, pp. 83–97, 2016, eurotherm Conference No. 105: Computational Thermal Radiation in Participating Media V. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022407315003192

[5] W. Jarosz. (2018) Monte carlo methods for physically based volume rendering. [Online]. Available: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm

[6] J. Novák, A. Selle, and W. Jarosz, "Residual ratio tracking for estimating attenuation in participating media," *ACM Trans. Graph.*, vol. 33, no. 6, nov 2014. [Online]. Available: https://doi.org/10.1145/2661229.2661292

[7] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation (3rd ed.)*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Oct. 2016.

[8] T. L. Kay and J. T. Kajiya, "Ray tracing complex scenes," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, p. 269–278, aug 1986. [Online]. Available: https://doi.org/10.1145/15886.15916

[9] A. S. Foundation. (2023) Openvdb open-source c++ library. [Online]. Available: https://www.openvdb.org/

[10] s. Delio Vicini. (2022) mitsuba2-vdb-converter, a small utility to convert openvdb files to the mitsuba 2 volume format resources. [Online]. Available: https://github.com/mitsuba-renderer/mitsuba2-vdb-converter

[11] W. Jarosz, S. Speierer, N. Roussel, M. Nimier-David, D. Vicini, T. Zeltner, B. Nicolet, M. Crespo, V. Leroy, and Z. Zhang. (2022) Mitsuba 3.1.0 documentation. [Online]. Available: https://mitsuba.readthedocs.io/en/stable/

[12] B. team. (2023) Blender 3.6 python api documentation. [Online]. Available: https://docs.blender.org/api/current/index.html

A Reference image for the final render

Figure 2: Reference image for our final render. It depicts a sunset in Gold Coast, Australia. The sunset is directly illuminating with orange-like colors the upper layers of a progressive cumulonimbus with a progressive anvil. There is also a crane from the harbor in that city and some birds that appear in front of the cloud.

B Additional renders and examples

Figure 3: $g = -0.65$

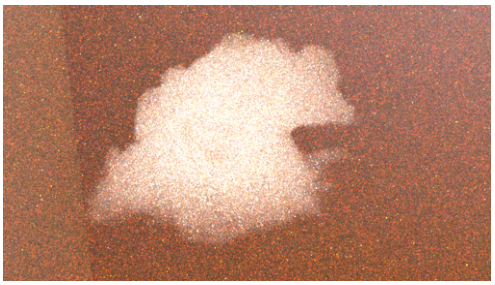

Figure 4: $g = 0.0$

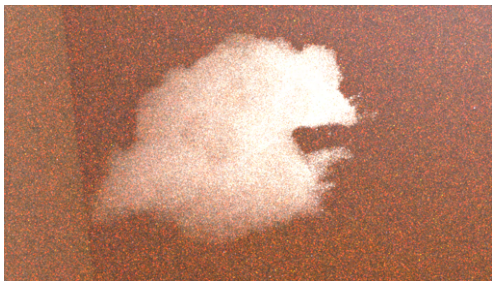

Figure 5: $g = 0.65$

Figure 6: Comparison between several values for the Henyey-Greenstein phase function in a heterogeneous volume. Light incides from the left. When g is negative, we get backward scattering, if it is 0, light gets scattered more evenly through the cloud, and if it's positive, it is scattered forward, to the right end of the cloud. The global volume is using the Rayleigh phase function, with altered coefficients (particle density, etc) to get a sunset effect with orange colors.

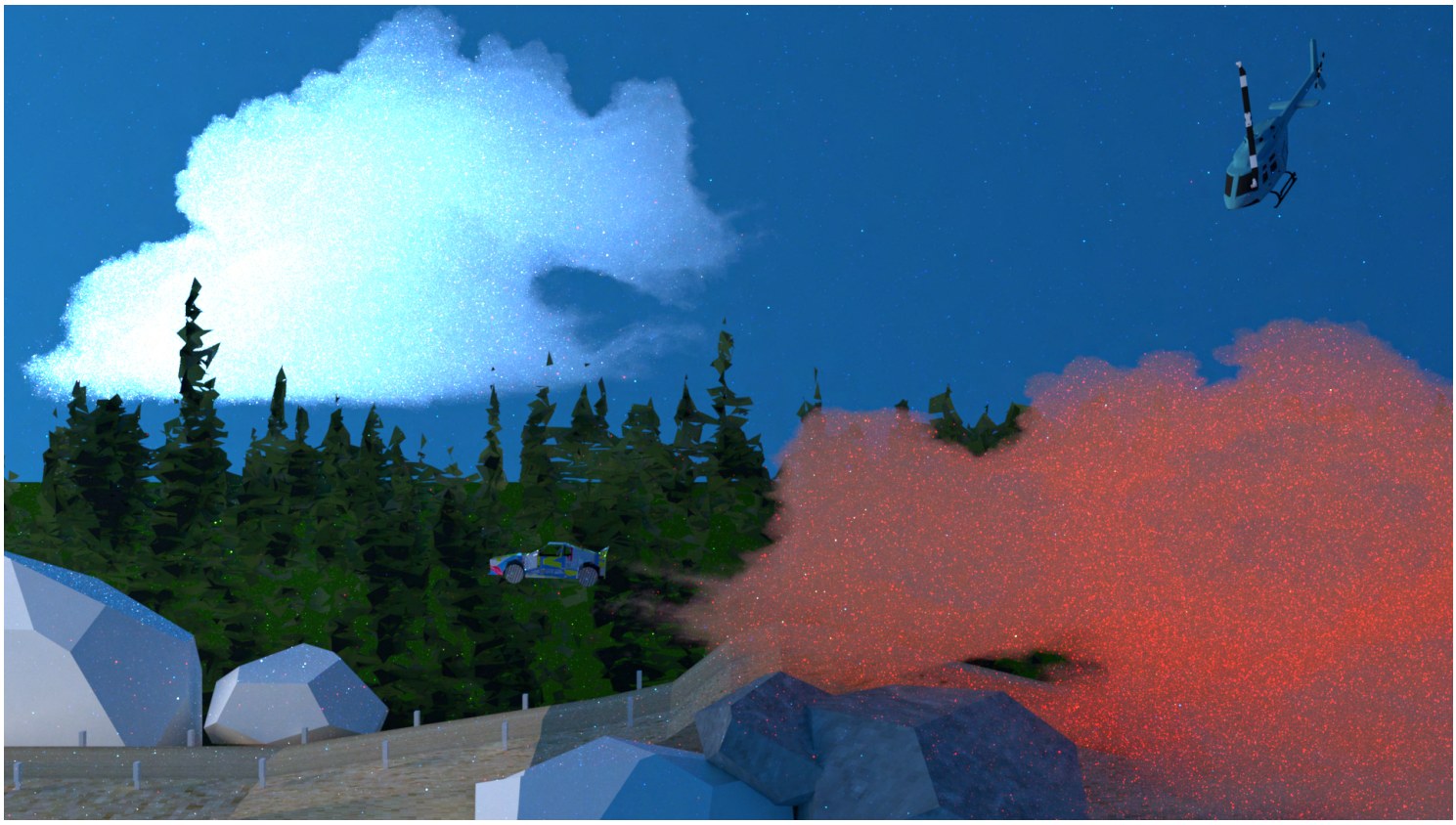

Figure 7: Another render featuring 2 bounded volumes in the scene, in this case a cloud and dust. The dust is in fact the same volume as the cloud, but scaled to look different. The scene also has global constant fog which gives the image its slightly dull appearance.

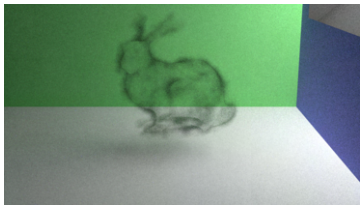

Figure 8: $\mu_t = 4, \mu_s = 1$

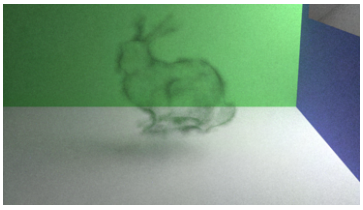

Figure 9: $\mu_t = 4, \mu_s = 2$

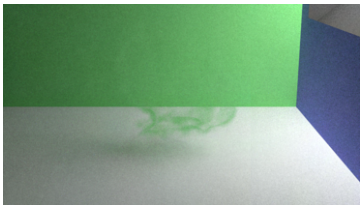

Figure 10: $\mu_t = 4, \mu_s = 4$

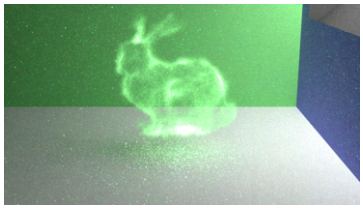

Figure 11: $\mu_t = 4, \mu_s = 8$

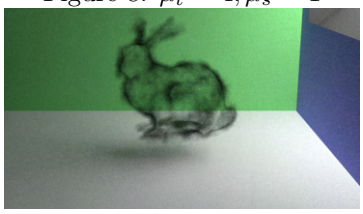

Figure 12: $\mu_t = 8, \mu_s = 1$

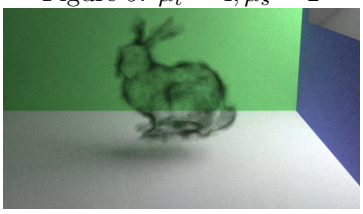

Figure 13: $\mu_t = 8, \mu_s = 2$

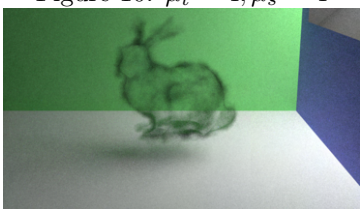

Figure 14: $\mu_t = 8, \mu_s = 4$

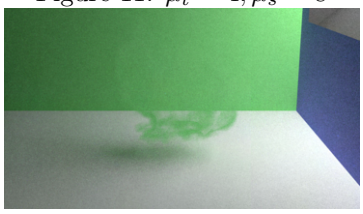

Figure 15: $\mu_t = 8, \mu_s = 8$

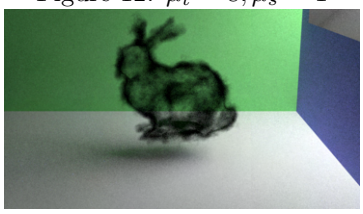

Figure 16: $\mu_t = 16, \mu_s = 1$

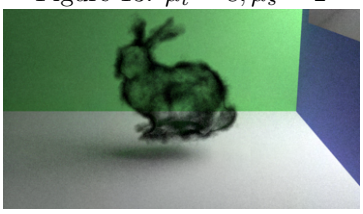

Figure 17: $\mu_t = 16, \mu_s = 2$

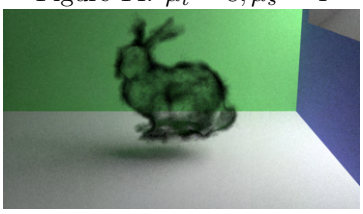

Figure 18: $\mu_t = 16, \mu_s = 4$

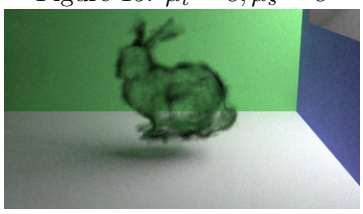

Figure 19: $\mu_t = 16, \mu_s = 8$

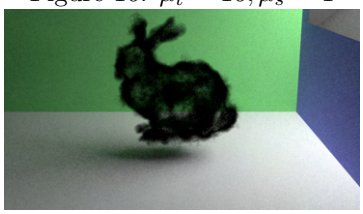

Figure 20: $\mu_t = 32, \mu_s = 1$

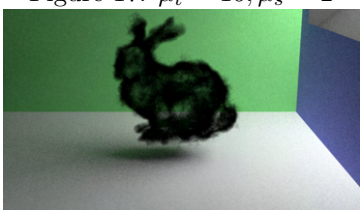

Figure 21: $\mu_t = 32, \mu_s = 2$

Figure 22: $\mu_t = 32, \mu_s = 4$

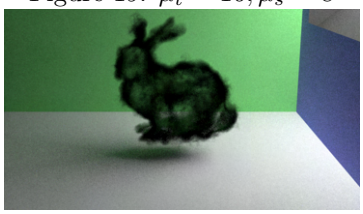

Figure 23: $\mu_t = 32, \mu_s = 8$

Figure 24: Comparison for several values for the scattering and extinction coefficient over the Stanford bunny volume. The scattering albedo is indirectly determined by $\frac{\mu_s}{\mu_t}$.