Aurora Borealis Simulation

Project Team CHS: Matthew Acevski, Dong Ha Lee, Vid Homšak, Andres Perez Fadon
Supervisor: Tsz Choi
Imperial College London
22 June 2020

Abstract

The goal of this project was to generate a realistic computer simulation of the aurora using basic physics and computational methods. The aurora is a geomagnetic phenomenon caused by high-energy electrons in the solar wind colliding with atoms in the atmosphere, exciting them to a higher energy level, which then de-excite to emit light. Using our understanding of the aurora, we simulated the phenomenon, starting from the motion of a single electron and building up to a realistic macroscopic model as a series of vertical sheets with random, natural folds.

I. Background

Initially, we intended to analyse magnetic field data of the Sun from satellite data. However, after discussions with our supervisor and amongst ourselves, we decided that we wanted to simulate an aurora as it is a geomagnetic phenomenon. Our early goals were to develop a model, at a molecular level first, of charged particles in a magnetic field then, form a basic simulation of the aurora at a macroscopic level.

Our next goals were to simulate it for a much greater number of particles (potentially using multiple processing) and have a function to produce lights of different intensities according to the different number densities of elements which produce the lights in the aurora. Also, we would try to make a function to yield the output intensities for each of the different wavelengths of light.

Our end goal was to produce a simulation of the aurora based on random collisions of electrons (i.e. there are random deflections in the path of the electrons) which takes atmospheric data to show correct intensities of RGB lights at every point on the aurora simulation as seen from the position of the observer. If this was successful, we could consider what we may observe if we change some variables, such as changing the composition of the atmosphere.

II. Description of Project Work

A. *Summary of the project work*

Weeks 1-2: The project started as a culmination of research into the Physics behind aurorae as well as Vid and Andres starting a basic particle simulation in Python of a charged particle in a uniform magnetic field which we believed to be a rudimentary visualisation of a particle interacting with Earths magnetic field in the upper atmosphere.

Week 3: Vid and Andres went on to improve their code by adding more particles to see their grouped motion in a magnetic field. We noticed that, as we added more and more particles, the grouped movements became far more chaotic and would take too long to be feasible in a macroscopic model; resulting in what we believed to be a poor representation of an aurora. The approach was changed to simulate it on a much larger scale by coding the motion ourselves, rather than relying on the particles to move, based on a magnetic field. This meant we could achieve a simulation which is more visually accurate. However, we were encountering problems such as implementing time-dependent motion with lots of particles while maintaining a reasonable frame-rate. Our supervisor, Kelvin, was asked to send an email requesting the use of the college's computers which had a lot more processing power than ours, which would reduce the processing time and lag.

Week 4: We all progressed into deeper research into the Physics, including searching for past papers written on the phenomena which gave us a good understanding of how to model the motion of Aurora using a combination of randomised sinusoidal motion. We found that a critical step in simulating the aurora is finding an initial coordinate P, from which we can simulate an electron scattering through the atmosphere. The method suggested by Baranoski et al. [1] was to sample a sine wave at pseudo-random intervals. Then, the sine wave is interrupted, attached to a Bézier curve, and continued with the same sine wave. This simulates folds in the aurora.

However, using the method suggested above leads to several problems:

- The intersection points between the Bézier curve and the sine waves will lead to unnatural transitions.
- The process is not automated.
- Bézier curves are difficult to implement computationally.

Andres found an alternative approach using 2D-Perlin noise (an algorithm which produces a correlated pseudo-random number). The advantage of Perlin noise is that the number produced will be related and close to numbers produced from similar parameters. Andres added a biased Perlin noise (as the average value of Perlin noise is 0) which produced smooth, automatic transitions [2], [3].

Upon achieving what we believed to be accurate motion of aurora, we went onto discuss how we could implement a typical aurora's colour scheme. Upon research done mostly by Dong, we saw that the base of the aurora is predominantly bright green in colour and the higher you go, it gets redder in colour; this is due to the different densities of various elements in our atmosphere as you increase in altitude.

Matthew searched for data that could be used to accurately model the density of Oxygen and Nitrogen in the air as a function of altitude so we could model the colour proportions correctly. He found a website endorsed by NASA which gave data readings for air and element density at equal intervals of altitude at any coordinate on Earth [4].

Dong began to look into the Beer-Lambert law to understand how the attenuation of light changes for different mediums and wavelengths. Taking the data reading found by Matthew, Dong attempted to create a function which returns the attenuation of light intensities as it passes through the air of different densities. He extrapolated from mass attenuation coefficients in the x-ray range to find the attenuation coefficients in the visible range. However, the function failed to yield sensible absorption values.

Week 5: Not much work was done on the project this week due to exams, we simply planned for the next couple weeks to start the report write up and finish implementing the colouring.

Week 6: To finalise the simulation, Vid implemented a Ray Tracing package into Python called POV-ray [5] which we used to record the simulation as a ray-tracing file so we could compile a short video of a sped-up simulation of the aurora over time. We also did further research into modelling the colours of the aurora. We found a graph predicting the rate of photon emission at varying heights for different wavelengths of light [6] but there were no data or functions available for these graphs, so Andres copied the graph into the background of a graphing calculator and approximated a Poisson function over it and adjusted the parameters to fit the graph. Also, Andres incorporated the mean free path of an electron going through a medium, such that the mean length between collisions is inversely proportional to the density of the air at a given height.

Dong completed the attenuation function by assuming that the attenuation is dominated by Rayleigh scattering in the visible spectrum. By finding the refractive index of the air as a function of the wavelength of light and the density of air, the Rayleigh cross-section could be determined [7]. However, it was deemed that the attenuation factor was so minimal that it was not included in the main body of code. Although, if Mie scattering and atmospheric absorption are included, the attenuation factor may be more significant.

B. Python packages used in simulation

- Vpython – Used for 3D rendering

- Random – To calculate random variables for the motion of the aurora
- PIL – Used to capture each frame of circling electrons
- Noise – to implement the Perlin noise for curve generation
- Time – To calculate time between frames
- PyAutoGUI – For finding corners of the simulation and cropping the screenshots to the correct size
- POV-Ray – Used for frame recording as well as implementing Ray tracing into every file

III. Summary of Results

The initial aim of this project was to successfully simulate the physics behind the aurora borealis, and thus create a visual model for this phenomenon. Even though this was a highly ambitious aim, as the programming taught in the first year is limited to data analysis, we not only achieved our goals but also exceeded our expectations. Starting from first principles and carefully simulating the motion and scattering of charged particles interacting with the Earth's magnetic field, we obtained a high-resolution image, which highly resembles the aurora borealis. Furthermore, we ambitiously extended our aims to produce a video of the aurora changing over time. This was a challenging task, as it is hard to model a smooth transition between frames given that most of the parameters in each frame are random. Moreover, as we had to simulate many particles scattering many times per frame, and for many frames, computing power was an issue. Using a correlated noise function known as Perlin noise [2], [3], and high-end parallel computing service from Imperial College London, we calculated enough frames to produce a realistic 10-second video of an aurora evolving over time. This last task greatly exceeded what we initially thought was possible to do with so little knowledge and so little time.

IV. Conclusion

To conclude, the aims of the project were met as we successfully produced pictures and a video of the aurora borealis evolving over time. Yet, due to time constraints and technical issues with the rendering software POV-Ray with larger files, a limited number of particles were able to be simulated, which unfortunately causes the simulated aurora to look less realistic.

Had we not spent that much time initially on simulating the electromagnetic interactions of electrons in the geomagnetic field, we could have reached more extended goals, since we later figured that the molecular approach is unusable for simulating aurora on macroscopic scale. Also, using PIL to take snapshots of the screen turned out to be a very inefficient method. Hence the process of implementing this method of recording was unnecessary since we later used POV-Ray. Thus, due to time constraints, we were not able to simulate the aurora on other planets, add aerosol scattering to the attenuation factor or explored how it behaves if the real geomagnetic field was utilised, which were some of the extensions we had envisioned.

However, we have produced a time-evolving simulation of the aurora, which we were initially unsure if it was possible, due to our limited computing skills. Moreover, we improved the methodology used by Baranoski et al. by utilising the 2D-Perlin noise, which solves the issue of unnatural transitions in the waves by inserting Bézier curves and automates the process of simulating folds in the simulated aurora.

On the other hand, our simulation lacks a realistic feel firstly due to, only simulating a small number of particles and secondly and most importantly, due to a lack of a physical model justifying the Gaussian blur, which must be implemented since photons are scattered when travelling from aurora to an observer. We could not implement it as our method utilised VPython and POV-Ray that made it impossible to include it in our code. However, blurring was added in after effects in Premiere Pro, to improve image realism, yet it lacks any physical justification. In future, we should consider changing wavelengths and width of the aurora with time.

Furthermore, our simulation was limited in computational power because we used Python to write our code in. It has a global interpreter lock, which without utilising any special multiprocessing modules forbids running computations in parallel on multiple processors. We tried to implement multiprocessing into our code, but due to the lack of our knowledge, we failed. Thus, for speed, we should have aimed for C or any other faster language, which would allow running the code on multiple nodes using Imperial's high-end parallel computing service. This would enable simulations with numbers of particles orders of magnitude greater than what we have achieved. Moreover, a better alternative would be writing the code in a dedicated ray-tracing program/language, which would also allow us to implement the Gaussian blur the way it was done in the paper by Baranoski et. al.

V. Acknowledgements

We would like to recognise the contributions of our supervisor, Tsz Kelvin Choi, for his guidance throughout the project. Furthermore, he provided important resources for us, including access to the Imperial College London high-end parallel computing service, and helped us understand and improve the simulation by thinking more carefully about each stage of the algorithm. We would also like to acknowledge Bruce Sharewood, one of the developers of the python module VPython (visual python), which we used to render the data into a 3D canvas and then into a PNG file. He kindly answered our questions in a forum to stop some unnecessary processes which improved the program speed. This includes fixing a 5 year old issue with the module.

References

[1] Baranoski, Gladimir & Rokne, Jon & Shirley, Peter & Trondsen, Trond & Bastos, Rui. (2000). Simulating the aurora borealis. 2-432. 10.1109/PCCGA.2000.883852. [Accessed: 22/05/2020]
This is the paper we took the most inspiration from for the project as a whole. It helped us understand the intricacies of the mathematics behind simulating the movement of Aurora as well as giving us the idea to improve upon it using the concept of Perlin noise.

[2] Perlin noise (article) |Noise| Khan Academy [Internet]. Khan Academy. 2020. Available from: https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise. [Accessed: 03/06/2020].
We used this website and its videos to understand what Perlin noise is and how we could use it to create a good approximation of the time-dependent movement of aurora.

[3] noise [Internet]. PyPI. 2020. Available from: https://pypi.org/project/noise/. [Accessed: 05/06/2020]
This was used to help implement Perlin noise into our simulation.

[4] MSIS-E-90 Atmosphere Model [Internet]. Ccmc.gsfc.nasa.gov. 2020. Available from: https://ccmc.gsfc.nasa.gov/modelweb/models/msis_vitmo.php. [Accessed: 05/06/2020]
This website provided us with data for the densities of Oxygen and Nitrogen in our atmosphere as a function of height which helped us implement the varying colours in our simulation.

[5] FreeCAD \pov ray tutorial - FreeCAD Forum [Internet]. Forum.freecadweb.org. 2020. Available from: https://forum.freecadweb.org/viewtopic.php?t=32745. [Accessed: 30/05/2020]
This website was used for general help with POV-Ray while implementing ray tracing into our code.

[6] Aryal, S., Finn, S. C., Hewawasam, K., Maguire, R., Geddes, G., Cook, T., et al. (2018). Derivation of the energy and flux morphology in an aurora observed at midlatitude using multispectral imaging. Journal of Geophysical Research: Space Physics, 123, 4257–4271. Available from: https://doi.org/10.1029/2018JA025229 [Accessed: 17/06/2020]
This paper was key to the last part of the project, where we implemented the colouring of the Aurora. We approximated the graphs they predicted for Rate of Photon emission as a function of height.

[7] Thalman R, Zarzana KJ, Tolbert MA, Volkamer R. Rayleigh scattering cross-section measurements of nitrogen, argon, oxygen and air. [Online] Journal of Quantitative Spectroscopy and Radiative Transfer. Journal of Quantitative Spectroscopy and Radiative Transfer; 2014. p. 171–177. Available from: doi:10.1016/j.jqsrt.2014.05.030. [Accessed: 15/06/2020]
We used this paper for the Rayleigh cross-section and refractive indices of various components making up the atmosphere.

Bibliography

Bodhaine, B. A., Wood, N. B., Dutton, E. G., & Slusser, J. R. (1999). On Rayleigh Optical Depth Calculations. Journal of Atmospheric and Oceanic Technology, 16(11), 1854-1861. doi:10.1175/1520-0426(1999)0162.0.co;2. [Accessed: 15/06/2020]

We used this article to find appropriate approximations for the refractive index of dry air in standard conditions, dependent on wavelength.

Edlén B. The Refractive Index of Air. [Online] Metrologia. Metrologia; 1966. p. 71–80. Available from: doi:10.1088/0026-1394/2/2/002. [Accessed: 16/06/2020]

We used this article to approximate the refractive index of air, dependant on air pressure.

Ishikawa T, Yue Y, Iwasaki K, Dobashi Y, Nishita T. Modeling of aurora borealis using the observed data. 2013.. Available from: doi:10.1145/2461217.2461220. [Accessed: 22/05/2020]

This paper helped us gain a better understanding of the Physics behind how Aurora works.

Ityaksov D, Linnartz H, Ubachs W. Deep-UV absorption and Rayleigh scattering of carbon dioxide. [Online] Chemical Physics Letters. Chemical Physics Letters; 2008. p. 31–34. Available from: doi:10.1016
/j.cplett.2008.07.049. [Accessed: 15/06/2020]

This paper aided us in our understanding of the Rayleigh cross-section and Beer-Lambert Law.

Peele AG, Chantler CT, Paterson D, Mcmahon PJ, Irving THK, Lin JJA, et al.. Measurement of mass attenuation coefficients in air by application of detector linearity tests. [Online] Physical Review A. Physical Review A; 2002.. Available from: doi:10.1103/physreva.66.042702. [Accessed: 29/05/2020]

This paper was used to find the mass-attenuation coefficient of different elements found in the air at varying altitudes (could not obtain reasonable extrapolation to visible region).

Documentation for Classic VPython 6 [Internet]. Vpython.org. 2020. Available from: https://vpython.org
/contents/doc.html. [Accessed: 01/06/2020]

We used this website for general help with the VPython package, which we used to implement the ray-tracing element of our simulation.

[Internet]. YouTube. 2020. Available from: https://www.youtube.com/watch?v=5wZSt_LNq3U
feature=youtube. [Accessed: 16/05/2020]

We used this video as a starting point in understanding what the aurora is.