Render Aurora with Photon Mapping and Volume Rendering

Shizong Li, Li-Che Chien, Songnan Wu, Wei-Yuan Hsing
shizongl@usc.edu, lichechi@usc.edu, songnanw@usc.edu, whsing@usc.edu

1 Introduction

Aurora is a splendid natural phenomenon of great visual beauty. Auroras are the result of disturbances in the magnetosphere caused by solar wind. These disturbances are sometimes strong enough to alter the trajectories of charged particles in both solar wind and magnetospheric plasma. These particles, mainly electrons and protons, precipitate into the upper atmosphere and collide with atoms causing light emission. Due to its unpredictable and changeable complexity, rendering a physically plausible aurora is always challenging. In our final project, we present an algorithm to simulate the aurora borealis by providing a complete pipeline to render aurora based on physical simulation and volumetric rendering. The aurora shape is generated by simulating a 2D footprint, then a volume grid is built based on the density of the footprint. We generate photons inside the volume for volumetric photon mapping. By simulating these light emissions along with the spatial and temporal distribution of the entering electrons, we are able to render the major visual aspects of the auroral displays. The rendering result is also enhanced by applying multiple kinds of noises and customized colormap. This approach also allows the representation of time-dependent features that characterize the dynamic nature of the aurora.

Our work is based on pbrt-v2 framework, which provides a simple way to read and render customized 3D models so we can focus on rendering aurora. We implemented all the functions mentioned in this article in C++ to do the rendering, including the generation of photons with beams, the volumetric photon mapping, noise and casted light.

Figure 1: Aurora Image from Patrick Endres

2 Algorithms and Techniques

2.1 Creating Aurora Curtains - Footprints

The charged particle fluxes, which aurora curtain are formed with, are driven by magnetohydrodynamics. Because the detailed interactions of the charged particles and magnetic fields that drive auroral substorms are still remained unsolved, for rendering purposes we approximate their effect. There are several ways to simulate the curtains, such as sine wave with Bezier's curve folds[1], 2D fluid dynamics simulation[2] shown in Figure 2. We represent an auroral curtain's path using sine waves, then applying low frequency noise curve to remap the density field and create our final footprint.

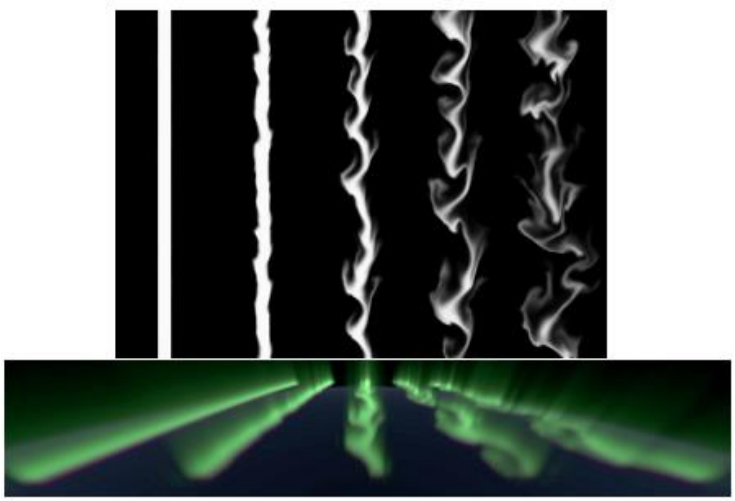

Figure 2: *2D fluid dynamics simulations to model curtain complexity.*

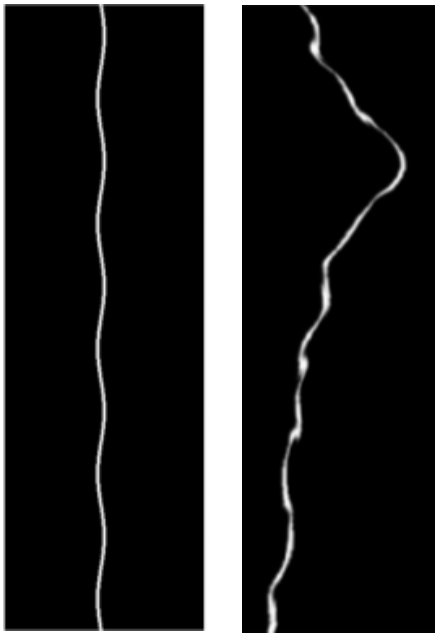

Figure 3: *Sine wave and our final footprint*

2.2 Photons Generating with Electron Beams

Vector ***v***, is defined as the electron beam's velocity. The angle between the electron's velocity vector and the geomagnetic field vector ***B*** is called the pitch angle, α (Figure 4a). A "loss cone" of pitch angles is bounded at an altitude **h** by an angle α_D. Electrons with **α <= α_D** are in the loss cone and are precipitated (lost) into the atmosphere. The boundaries of this loss cone are somewhat diffuse since the value of α_D decreases with altitude. The length of the path is given by a parameter **L** which is associated with the height chosen for the modeled auroral display. The threshold **dt** is an input parameter which depends on the initial energy of the electrons. We use **dt** = 0.004 according to [1]:

$$P^{new'} = P^{old} + (L\ d_t\ \nu)\ \frac{\vec{B}}{|\vec{B}|}$$

In practice, for each starting point **P**, we first move the electron along the geomagnetic vector **B**, whose distance is determined by a random step **s**, then we add some displacement in the plane perpendicular to the geomagnetic vector according to a uniformly sampled angle β:

$$\mathbf{p}_{new} = \mathbf{p} + s\mathbf{B} + t\mathbf{u}\cos(\beta) + t\mathbf{v}\sin(\beta)$$

and **α** is uniformly sampled between α_D and $\alpha_D - \Delta\alpha$. We iterate this procedure multiple times to generate hundreds of deflection points along the electron beam, and we place a photon at each deflection point. The radiance of the photon is interpolated from a pre-defined color map. Instead of applying the color map in [1], we built up our own color maps (Figure 5) for better visual effects.

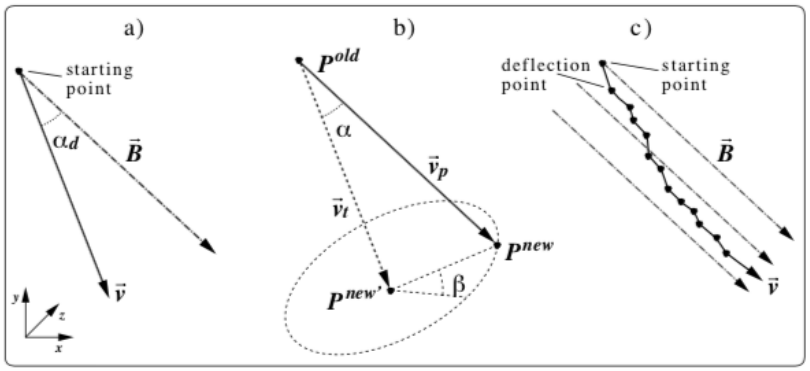

Figure 4: *Stages of the simulation of an electron beam path: a) precipitation, b) computation of deflection points and c) electron beam crossing onto other field lines.*

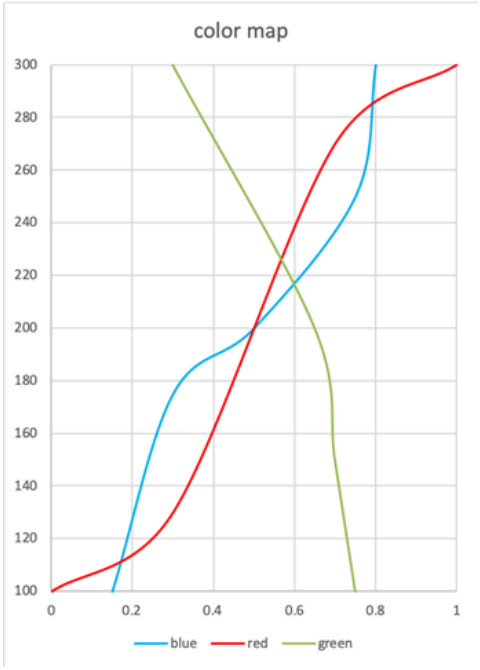

Figure 5: *New color map*

2.3 Volumetric Photon Mapping

Unlike the paper [1] which did rendering based on rasterizing photons into image films directly (Figure 6), we used volumetric photon mapping to render the final images. For each point inside the volume, we used the photons nearby to compute radiance of the emissive/in-scattering light at that specific point. The radiance comes from the weighted sum of all the photons inside a ball around the point with a predefined radius, where the weight is computed from a Gaussian kernel.

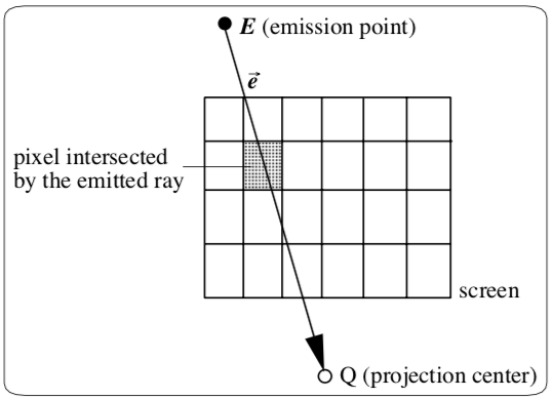

Figure 6: *Geometry for the forward mapping to the screen.*

2.4 Adding Perlin Noises

In order to make our aurora more natural, we decided to add Perlin noises during the process of generating photons and searching photons. We choose three parts of processing to add different noises to perturb:

2.4.1 Starting height of generating the electrons for different densities of photon (the energy of photon).

2.4.2 The height of color map of photons for different color changing.

2.4.3 The radius when we search for new photons for various densities of aurora curtain.

There are two types of Perlin noise we used in perturbing:

2.4.4 1D Perlin noise with Catmull Rom splines [3] for 2.4.1 and 2.4.2.
We sample each point with constant frequency and get each tangent with Catmull Rom splines method to make 1D Perlin noise.

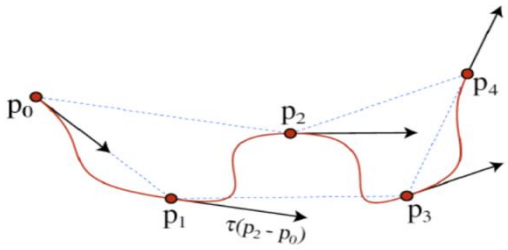

Figure 7: Catmull Rom Spline

2.4.5 2D Perlin noise with 4 near points interpolation for 2.4.3.
We sample points with constant frequency and get the altitude of each point, which multiplied by a random number and different level of persistence. Then, we interpolate 4 near points from the point we choose to get the final noise.

2.5 Aurora Casting Light on Objects

With generating process of aurora, we understand that aurora is an emitting light source to the scene, so we expect there would be light cast on other objects in the scene, which means the color of the aurora would affect the landscape's color. With this concept, we trace shadow rays along the volume, to emphasize the aurora casting light on the objects, when we trace each shadow ray, we give the radiance **Lsi** from the surface and compute with the transmittance **T** and the emission radiance results **Lai**, which is provided by the aurora. The final radiance **Li** is

$$\mathbf{Li = T * Lsi + Lai}$$

3 Result

There are a landscape model and two auroras in the final scene. The aurora simulates the footprint given in Figure 3. The aurora color is from purple to white, and at last to green, following the color map in Figure 5. The light cast on the landscape is shown with aurora's color when we trace the shadow ray along the volume.

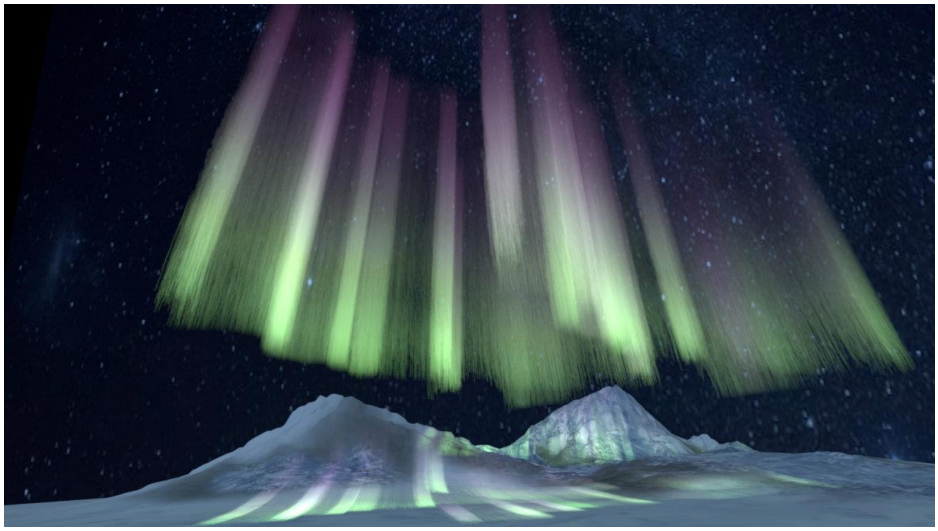

Figure 8: Final Aurora Result

References

[1] Gladimir VG Baranoski, Jon G Rokne, Peter Shirley, Trond Trondsen, and Rui Bastos. Simulating the aurora borealis. In Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on, pages 2–432. IEEE, 2000.

[2] Orion Sky Lawlor and Joe Genetti. Interactive volume rendering aurora on the gpu. 2011.

[3] C. Twigg, "Catmull-rom splines," Computer, vol. 41, no. 6, pp. 4–6, 2003.