スクリーン空間を用いたオーロラの高速なシミュレーション

-Fast Simulation and Rendering of Aurora Using Screen Space-

関口智大 †　藤澤誠 †　三河正彦 †
Tomohiro SEKIGUCHI†, Makoto FUJISAWA† and Masahiko MIKAWA†

† 筑波大学図書館情報メディア研究科
† Faculty of Library, Information and Media Science, University of Tsukuba

E-mail: †{sekiguchi, fujis, mikawa}@slis.tsukuba.ac.jp

1　はじめに

物理シミュレーションを用いた CG の生成では，クリエイターが望む色や形状を得るために，パラメータを細かく調整する必要がある．現在，このような細かな調整を，炎や煙の物理シミュレーションに対してインタラクティブかつ直感的に行うことができる手法の研究 [1] がおこなわれているが，これらの手法では，クリエイターが行った調整が即座にシミュレーションに反映される必要があるため，リアルタイムにシミュレーションを行うことが求められる．

北極や南極などで観測されるオーロラ現象のシミュレーションの研究 [2] も行われている．オーロラのシミュレーションでは，荷電粒子の動きを求めるフェーズと，荷電粒子と大気粒子との衝突から発生する発光をレンダリングするフェーズの 2 つを考える必要がある．動きを求めるフェーズでは，主に 2 次元的な動きのみ考えればよいので，非常に高速な計算が可能である．それに対してレンダリングフェーズでは，衝突地点でパーティクルを発生させ，それをスクリーン空間に投影する方法 [3] などが提案されているが，未だ 1 フレーム生成には数十秒から数分の時間がかかる．

本論文では，小島らの手法 [3] をベースに，生成する動画像の品質を保ちつつより高速なオーロラのシミュレーションを可能とする方法を提案する．従来手法ではシミュレーション空間上で行っていた荷電粒子と大気粒子との衝突地点の算出を，提案手法ではスクリーン空間上に投影してから計算することで，衝突地点の投影処理を省略する．また，輝度値の積算処理に関しては，事前計算によるテーブル化を行うことで高速化を図る．さらに，スクリーン空間上での落下計算と輝度値の積算処理に対して，GPU を用いた並列化を行うことで，最終的に約 10〜15 倍の高速化を実現した．

2　提案手法

2.1　全体の流れ

オーロラは，太陽から放射される荷電粒子が地球の磁場に沿って落下し，地球上の大気粒子と衝突することで起こる発光現象である．提案手法では毎フレーム荷電粒子の落下のシミュレーションを行い，衝突位置を決定する．提案手法の全体的な流れを図 1 に示す．まず，前計算で 2 次元平面上での荷電粒子の初期位置を決定し，その位置からの荷電粒子の落下シミュレーションを 3 次元空間中で行い，相対的な落下終了位置を得る．その後，平面上に配置された荷電粒子に 2 次元平面上の電磁場シミュレーションを行いローレンツ力を算出し，それを用いて毎フレーム荷電粒子の位置をこの平面上で更新することでオーロラの運動を表現する．次に，運動計算により求められた荷電粒子の位置と，前計算で求めた相対的な落下終了位置をスクリーン空間上に投影し，スクリーン空間上で荷電粒子の落下シミュレーション及び大気粒子との衝突判定を行い，スクリーン上での衝突位置を算出する．算出された衝突位置の近傍画素に，発光によって生じる輝度値を積算していくことで，最終的なカラーマップを生成する．

2.2　荷電粒子の初期配置

オーロラの運動は，地球の磁場方向に垂直な平面上での動きと捉えることができるため，図 2 のように，磁場に垂直な 2 次元平面状に荷電粒子を配置することで，オーロラ形状を表現する．荷電粒子は，ベジェ曲線と正弦曲線を用いて作成した曲線に対して，法線方向に幅を設定し，その内部の領域にランダムに配置する．

2.3　シミュレーション空間での荷電粒子の落下

前節で求めた荷電粒子の初期位置を荷電粒子の落下開始位置とみなし，シミュレーション空間上での落下計算を行

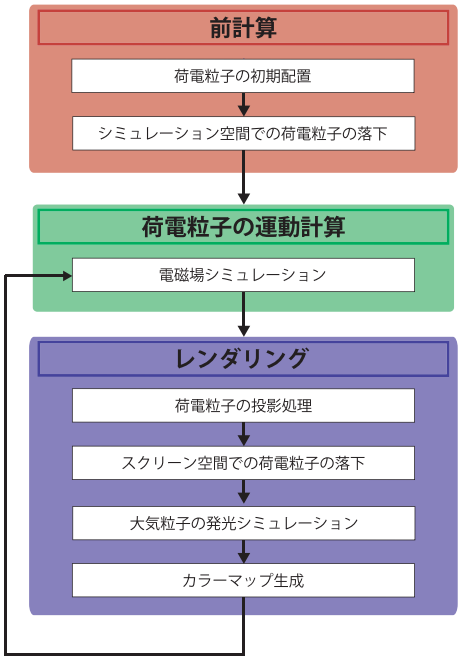

図 1　提案手法の流れ

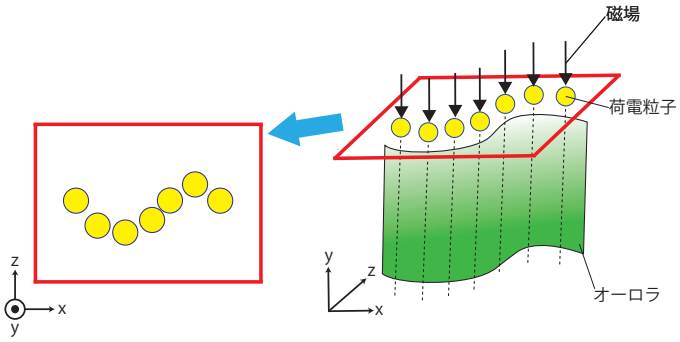

図 2　荷電粒子を配置する 2 次元平面

う．荷電粒子は，大気粒子との衝突を繰り返しやがて停止するが，最終的に落下する距離は荷電粒子の初期エネルギーに依存する．大気粒子と衝突するたびに荷電粒子の持つエネルギーを減算し，エネルギーが尽きた地点を荷電粒子の落下終了位置として導出する [4]．

まず，落下開始位置から磁場方向に微小タイムステップ幅 $\Delta\tilde{t}$ で進んだ距離を求め，落下位置を求める．現在の位置 $\tilde{\boldsymbol{P}}_{\boldsymbol{0}}$ から，落下位置 $\tilde{\boldsymbol{P}}$ を求める式は，次の式 (1) となる．

$$\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{P}}_{\boldsymbol{0}} + v_f \frac{\boldsymbol{B} + \boldsymbol{\omega}}{|\boldsymbol{B} + \boldsymbol{\omega}|} \Delta\tilde{t} \tag{1}$$

ここで，v_f は荷電粒子が地球の磁場方向に落下する速さ，$\boldsymbol{B}$ は地球の磁場，$\boldsymbol{\omega}$ は荷電粒子と大気粒子との衝突時に生じる微小なぶれを表す乱数である．

この位置を衝突判定位置とみなし，高層大気の密度分布 [5] を用いて荷電粒子が大気粒子と衝突するかを判定する．

衝突確率 P_c は次の式 (2) で求める

$$P_c = n\pi r^2 v_f \Delta\tilde{t} \tag{2}$$

n は高層大気の密度分布から求めた単位体積あたりの大気粒子の数密度，r は大気粒子の半径である．

座標 $\tilde{\boldsymbol{P}}$ において，式 (2) の P_c の値を計算し，生成した乱数との比較を行い衝突判定を行う．衝突したと判定された場合は，予め荷電粒子に設定しておいたエネルギーを減算し、衝突しなかった場合はそのまま落下のシミュレーションを継続し，次の衝突判定地点を求める．上記の処理を，荷電粒子のエネルギーが 0 になるまで行い，エネルギーが尽きた地点を落下終了位置とする．

2.4　電磁場シミュレーション

荷電粒子は，地球の電場と磁場によるローレンツ力を受け運動する．2 次元平面状での荷電粒子の分布から，平面状での電位及び電場を算出し，電場と磁場から各荷電粒子にかかるローレンツ力を求める．求めたローレンツ力を用いて，各荷電粒子の位置を更新する．上記の処理を毎フレーム行うことで，オーロラの動きを表現する．本論文では小島らの手法 [3] を用いて，このシミュレーションを行った．

2.5　荷電粒子の投影とスクリーン空間での落下シミュレーション

2.4 節で更新した荷電粒子の落下開始位置と，2.3 節の前計算で求めた相対的な落下終了位置から，スクリーン空間上で落下シミュレーションを行うことで，スクリーン上での大気粒子の発光地点を求める．3 次元空間で落下処理を行う従来手法と異なり，この方法では投影処理を大幅に削減できる．

処理の模式図を図 3 に示す．まず，シミュレーション空間上での落下開始位置と終了位置を発光点とみなし，光が届く範囲を半径としたパーティクルを生成する．このパーティクルをスクリーン空間上に投影することで，スクリーン空間における落下開始位置と落下終了位置が求められる (図 3 の黄色の丸)．落下開始位置から，2.3 節で行った落下計算と同じような処理を行うため，スクリーン空間上での荷電粒子の落下方向と，微小タイムステップ幅 $\Delta\tilde{t}$ ごとに落下する距離を求める．落下方向は，投影後の落下開始位置から落下終了位置へのベクトルを正規化することで算出する．落下距離は，シミュレーション空間とスクリーン空間での，微小タイムステップ幅ごとの落下距離と最終的な落下距離の比から求める．スクリーン空間上での落下開始位置を $\boldsymbol{P_s}$，落下終了位置を $\boldsymbol{P_e}$，シミュレーション空間上での落下する速さを v_f，最終的な落下距離を D とすると，スクリーン空間

上での微小タイムステップ幅 $\Delta\tilde{t}$ ごとの落下距離 $\tilde{v_f}$ は次式 (3) で表すことができる．

$$\tilde{v_f} = v_f \Delta\tilde{t} \frac{|\boldsymbol{P_e} - \boldsymbol{P_s}|}{D} \tag{3}$$

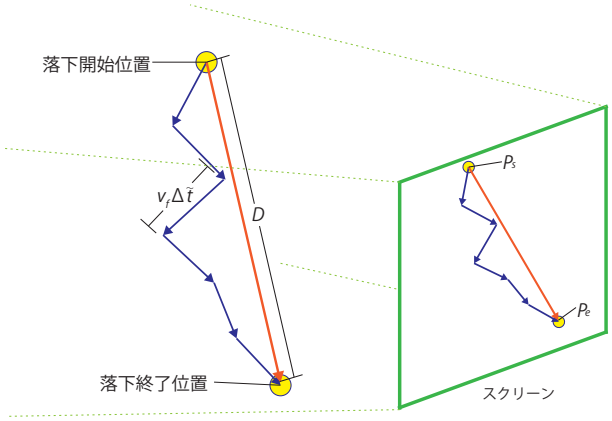

図 3　スクリーン空間上での落下方向

落下開始位置から，微小タイムステップ幅 $\Delta\tilde{t}$ ごとにスクリーン空間上での荷電粒子の位置を更新していき，大気粒子と衝突するか否かを判定する．荷電粒子のスクリーン上での位置 $\tilde{\boldsymbol{P}}$ の更新は，現在の位置を $\tilde{\boldsymbol{P_0}}$ として次の式 (4) を用いて行う．

$$\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{P_0}} + \tilde{v_f} \frac{\boldsymbol{P_e} - \boldsymbol{P_s} + \tilde{\boldsymbol{\omega}}}{|\boldsymbol{P_e} - \boldsymbol{P_s} + \tilde{\boldsymbol{\omega}}|} \tag{4}$$

ここで，$\tilde{\boldsymbol{\omega}}$ は荷電粒子と大気粒子との衝突から生じる微小なずれをあらわすベクトルである．$\tilde{\boldsymbol{\omega}}$ の各要素を，大気粒子半径 r と乱数 $\theta \in [0, 2\pi]$，$k \in [-r, r]$ から，$(r\cos\theta, r\sin\theta, k)$ とすることで，式 (1) で利用した $\boldsymbol{\omega}$ と分布がほぼ同じになるようにした．

この位置を衝突判定位置とし，荷電粒子と大気粒子が衝突するかを判定する．衝突判定は式 (2) で求めた P_c と乱数を用いて行う．衝突したと判定された場合，その地点を発光点とみなし大気粒子の次節で説明する発光シミュレーションを行い，荷電粒子のエネルギーを減算していく．衝突しなかった場合はそのまま次の衝突判定位置を求める．荷電粒子のエネルギーが 0 になるまで落下シミュレーションを行い，全ての発光点を求める．

2.6　大気粒子の発光シミュレーション

米山らの手法 [4] を基に、荷電粒子と大気粒子との衝突地点において、衝突した大気粒子の種類と放出する波長を決定する．

また，大気粒子が発光する光は視点との間にある空気層により減衰するため，その輝度値は，放射する波長ごとに設定する輝度値と光の減衰率を用いて求める．発光地点のスクリーン空間上での z 座標を z_a，減衰係数を d とし，光の減衰率 c は平方指数式を用いて次の式 (5) で表すことができる．

$$c = \exp(-(d\, z_a)^2) \tag{5}$$

発光点の z 座標 z_a は，落下開始位置と終了位置のスクリーン空間上での z 座標を線形補間することで得られる．

2.7　カラーマップの生成

各発光点における波長の種類と輝度値を用いて，最終的なカラーマップ (出力画像) を生成する．発光の届く範囲を半径，発光点を中心とした発光パーティクルを生成し，そのパーティクルの半径内に存在する画素に輝度値を積算していく．このとき，発光の届く範囲の大きさは各波長ごとに半径として予め設定しておく．スクリーン空間上での発光パーティクルの半径は，透視投影により，シミュレーション空間上での発光点の位置によって変化する．本手法では，落下開始位置と終了位置以外での投影処理を省略しているため，透視投影による縮尺率を，落下開始位置と落下終了位置におけるパーティクルの縮尺率から線形補間で求めることで近似する．

半径が求まったら，それをサイズとしたガウシアンフィルタを用いることで，発光点に近い画素ほど積算される輝度値が大きく，遠い画素ほど小さくなるように格納する．標準偏差を σ，発光点と画素中心との距離を b とし，ガウシアンフィルタの式を以下の式 (6) に示す．

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{b^2}{2\sigma^2}\right) \tag{6}$$

また，ガウシアンフィルタの計算を各パーティクル及び各画素ごとに毎回行うのは計算コストが高いため，パーティクル中心と画素中心との距離をキーとしたテーブル化を行った．全ての発光点における輝度値を格納し終えたら，各画素の波長と輝度値を RGB 値に変換することで，最終的なカラーマップを生成する．

3　実行結果と考察

オーロラの 1 フレームあたりの静止画生成にかかる時間を計測し，従来手法と提案手法の速度比較実験を行った．実行環境は CPU: Intel Core i7-6700K 4.00GHz,GPU: NVIDIA GeForce GTX 980 Ti である．従来手法として，小島ら [3] の手法を参考としたパーティクルベースのシミュレーション手法を実装し，荷電粒子の落下シミュレーション及び衝突地点での発光シミュレーションにかかる時間を計

測する．CPU による実装では，OpenMP を用いた並列化を行った．更に同手法を NVIDIA CUDA を用い GPU によって並列化し，同様に提案手法との速度比較を行う．実験方法は，図 4 のようなオーロラを 10 種類生成し，それぞれの落下する荷電粒子数と発光パーティクルの半径倍率 k を変化させ，各条件において 1 フレームあたりにかかった時間の平均値を比較した．レンダリング時間のグラフは図 5 のようになった．

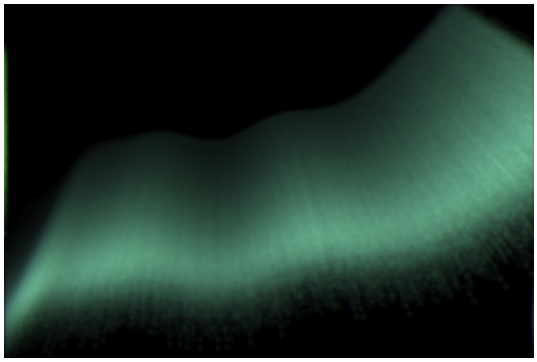

図 4　オーロラの生成結果

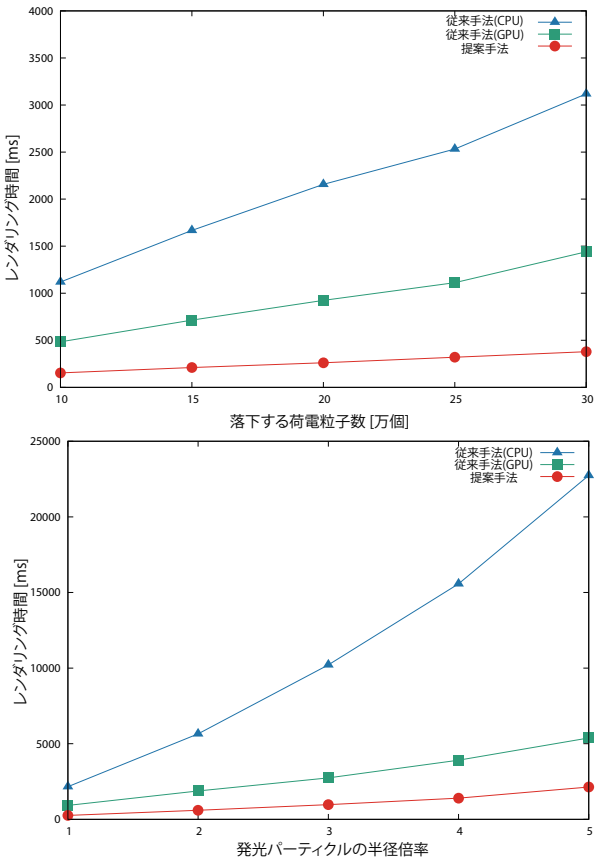

図 5　従来手法と提案手法の速度比較結果

図 5 から，投影処理が必要な発光パーティクルの数が増えるほど，従来手法と提案手法との差が開いているのが観測された．また，発光パーティクルの半径を増加させると，レンダリングにかなりの時間がかかってしまっていることから，荷電粒子の落下処理よりも，発光パーティクルの積算処理にかかる時間の割合が大きいことが考えられる．

4　まとめと今後の課題

本論文では，シミュレーション空間で行っていた荷電粒子の落下計算をスクリーン空間上で行うことに加え，発光パーティクルの積算処理のテーブル化や GPU による並列化を行うことで，オーロラのシミュレーションの高速化を実現し，速度の比較実験でそれを確かめた．

今後の課題として更なる高速化と精度の向上が挙げられる．パーティクル数が少ない小規模なシーンならば 5〜10fps でレンダリングまで可能なのだが，荷電粒子数が膨大なオーロラや，複数のオーロラが 1 枚の画像に含まれているような大規模なシミュレーションの場合，未だレンダリングに時間がかかってしまう．また，本手法ではスクリーン空間上での荷電粒子の微小タイムステップ幅毎の落下距離を，シミュレーション空間とスクリーン空間の最終落下距離の比によって求めているため，オーロラを真横から見た場合は従来手法との生成結果に差はないが，下方向から見た場合に，オーロラの磁場方向の長さが従来手法による生成結果に比べ長くなってしまう現象が観測された．これを解決するため，スクリーン空間上での荷電粒子の微小タイムステップ幅ごとの落下距離を決定する際に，磁場方向と視線方向との関係性を考慮に入れる必要があると考えている．

参考文献

[1] 渋川雄平，土橋宜典，山本強，“ガス状物体のボリュームレンダリングのための特徴量に基づく伝達関数設計手法”，映像情報メディア学会誌，vol. 68，no. 2，pp. J66-J71, 2014.

[2] G.Baranoski, J. Rokne, P. Shirley, T. Trondsen, R. Bastos. ”Simulating the Aurora”, *J. Visual. Comput. Animat.*, Vol. 14, No. 1, pp. 43-59, 2003.

[3] 小島啓史，竹内亮太，渡辺大地，三上浩司，“特徴的な動き方を考慮したオーロラのビジュアルシミュレーション”，芸術科学会論文誌，Vol. 12，No. 1， pp. 24-35，2012.

[4] 米山考史，近藤邦雄，“発光原理を考慮したオーロラのビジュアルシミュレーション”，日本図学会 2005 年度大会学術講演論文集， pp. 69-74，2005.

[5] 丸山隆，“6．電離圏プラズマ (＜小特集＞宇宙天気予報)”，プラズマ・核融合学会誌，Vol. 82，No. 11，pp.762-766，2006.