2022 年 2 月 8 日

令和３年度　情報・ネットワーク工学専攻　CS コース　修士論文発表会

Unity によるマイクロマグネティクスシミュレーションのリアルタイム可視化

I 専攻　成見研究室　学籍番号：2031160　　LI JIAQING

1 はじめに

シミュレーションの可視化は自然現象の解析や分析，工業製品の開発に欠かせないものであり，日頃から盛んに行われている．また，リアルタイムに可視化しながらシミュレーションできれば，解析サイクルを早く出来る．

画像処理を担当する Graphics Processing Unit(GPU)に汎用計算を行わせる GPGPU が近年流行しており[1]，シミュレー ションした結果をそのまま描画にも使用することで可視化しながらシミュレーションできる．

さらに，近年ではモバイル端末が普及しており，性能も高くなっている．これまで高性能な PC でしか不可能だったリアルタイムシミュレーションがスマートフォンでも可能になってきている．

そこで，本研究では，Unity によるマイクロマグネティクスシミュレ ーションのリアルタイム可視化システムを開発する．Unity の機能を使って，Compute shader(cs)を用いてシミュレーションを高速化し，Shader プログラムを用いて描画する． NVIDIA の GPU だけではなく，他のブランドの GPU，あるいは CPU 内蔵 GPU でシミュレーションを実行することができる．

2 マイクロマグネティクスシミュレーション

マイクロマグネティクスとは，磁石内部に現れる原子磁気モーメント[式(1)]によって作られる磁化構造や，その動的な変化を扱う分野である．

$$\dot{\vec{M}} = -|\gamma|\vec{M} \times \vec{H} + \frac{\alpha}{M}(\vec{M} \times \dot{\vec{M}}) \qquad (1)$$

ここで$\vec{M}$は磁気モーメント，$\dot{\vec{M}}$は$\vec{M}$の時間微分，$\vec{H}$は磁界，γとαは定数である．

$\vec{H}$ は外部磁界($\overrightarrow{H^{EXT}}$)，一軸異方性エネルギーによる磁界($\overrightarrow{H^{A}}$)，交換エネルギーによる磁界($\overrightarrow{H^{E}}$)，静磁界($\overrightarrow{H^{D}}$)を用いて以下のように表される．

$$\vec{H} = \overrightarrow{H^{EXT}} + \overrightarrow{H^{A}} + \overrightarrow{H^{E}} + \overrightarrow{H^{D}} \qquad (2)$$

このうち静磁界の計算量は計算点の数を N とすると$O(N^2)$であり，計算量が膨大である．そのため，リアルタイムでシミュレーションするには GPU での加速が不可欠である．

3 先行研究

CUDA でマイクロマグネティクスシミュレーションを行い，グラフィック API(OpenGL)を用いて可視化を行っている研究は存在 している[2]が OpenGL で可視化プログラムを構築するのは手数がかかる．また，CUDA は NAVIDA の GPU だけをサポートし，別のブランドの GPU を使ったり，スマートフォンを使うと動 作しないという問題がある．

また，Unity と Compute shader でクロスシミュレーションを行う研究が存在している[3]．さらに，Compute shader は CUDA に負けない速度をもち，ボリュームレイキャスティングで CUDA より速いと報告されている[4]．本研究では，グラフィック API の Compute shader を用いて計算を加速している．

4 シミュレーションシステムの概要

図 1 は本システムでの処理の流れを示している．図 2 はアプリの外観である．本システムは主にシミュレーション計算プログラムと可視化プログラムの二つの部分がある．

シミュレーションプログラムでは，C#スクリプトで CPU ベースのプログラムあるいは Compute shader プログラムでシミュレーションプログラムを実行する．Compute shader を選ぶと GPGPU により，シミュレーションを高速に行う．

可視化プログラムでは，前述のシミュレーショプログラムで計算された結果を Unity のリアルタイムレンダリングパイプラインに送り，shader プログラムを実行し，それぞれの色と方向が付けられた矢印ポリゴンを描画する．

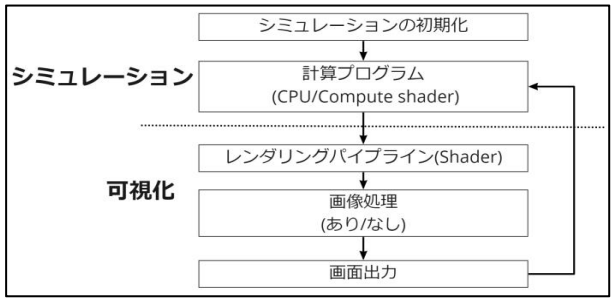

図 1:可視化システムの概要

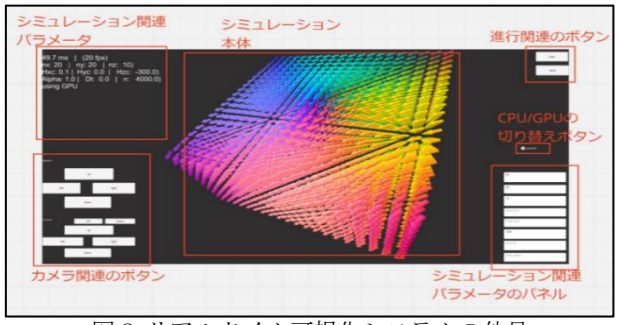

図 2:リアルタイム可視化システムの外見

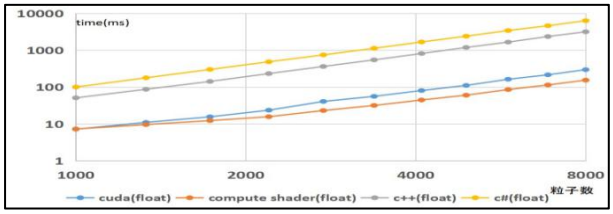

図 3：性能比較 1

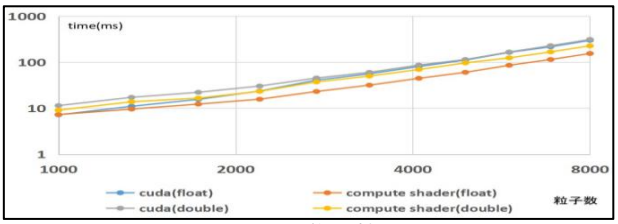

図 4：性能比較 2

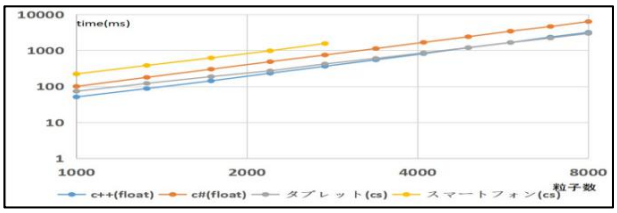

図 5：性能比較 3

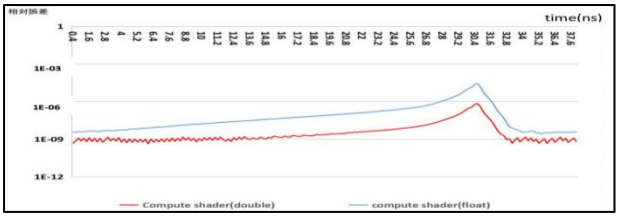

図 6：誤差解析 1

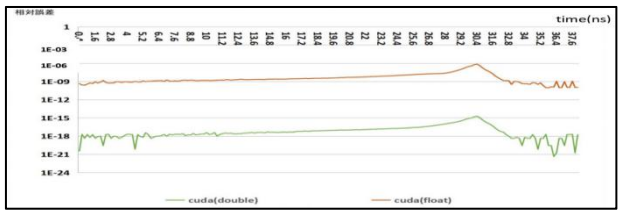

図 7：誤差解析 2

図 8：実行状況(左：パソコン，右：スマートフォン)

5 評価

5.1 性能評価

図 3 は同じパソコンで計算方法を変えた時の 1 フレームの計算時間を示している．Compute shader は CUDA より約 2 倍，CPU より約 20~40 倍速い．

図 4 は CUDA と Compute shader に対し，精度を切り替えたときの性能である．CUDA の double，float 精度の性能はほぼ同じであり，Compute shader では，float の性能は double より約 2 倍速い．

図 5 は比較的性能の低いスマートフォン(OPPO　AX7)とタブレット(Fire HD 10)の計算時間を PC の CPU と比較したものである．モバイル端末でも GPU を使えばデスクトップ PC の CPU 並の性能が出る．

表 1 は粒子数が 1152 の場合にスマートフォンでの性能を比較したものである．比較的新しいスマートフォンを用いればリアルタイムのシミュレーションが可能であることが分かる．

5.2　誤差解析

図 6，7 は CPU での倍精度に対する相対誤差を示している。計算方法を表 2 に示す．本来は Compute shader の誤差は CUDA と同じだと期待しているが，少し Compute shader の方が悪い．

表 1 ：各スマートフォンでの性能(フレームレート)

機種	Pixel3a	GalaxyZ Flip3	xiaomi mi9t pro	Xperia XZ1
発売年	2019	2021	2019	2017
性能	8fps	34fps	27fps	20fps

表 2：相対誤差の計算方法

1. CPU で全粒子の平均 $\overrightarrow{V_i}$ 値 V_{avr} を求める.
2. GPU で i 番目の Vec を計算する.
3. 粒子の Vec の相対誤差 $E_i = \left\|(\overrightarrow{V_i} - \overrightarrow{U_i})/V_{avr}\right\|$ を求める
4. 全粒子の平均相対誤差 E_{avr} を求める

6　考察

Compute shader の計算精度は CUDA より少し大きい．また，数値計算するためのライブラリなどはサポートされておらず，難しい物理シミュレーションアルゴリズムの実装は未だに困難である．総合的に考えると，Compute shader で精密な科学的物理シミュレーションをするには注意が必要である．

本システムでは，マルチプラットフォーム対応しており[図 8]，スマートフォンなどの端末でも実行できる．計算プログラムで使用される Compute shader はグラフィックスドライバーの一部であるため，CUDA のように数ギガバイトの実行環境をインストールする必要がない．そのため，利用者に対する負担が低く，手軽に使用できる．

7 おわりに

本研究では Unity でマルチプラットフォーム対応のマイクロマグネティクスのリアルタイム可視化システムを開発した．本システムは他の GPGPU API(CUDA など)で実装されるシステムに負けない速度をもち，スマートフォンでスムーズに実行できる．あらかじめドライバーをインストールする必要もなく，学生はスマートフォンなどのデバイスで手軽にシミュレーションができ，教育用途に役立つ可能性がある．

参考文献

[1] Amos Egela,　Lorenzo Pattelli,　Giacomo Mazzamuto,　Diederik S.Wiersma,Uli Lemmer,”CELES: CUDA-accelerated simulation of electromagnetic scattering by large ensembles of spheres,”　Journal　of Quantitative Spectroscopy&Radiative Transfer,vol.199,pp.103-110,2017

[2] 清水陽介,マイクロマグネティクスシミュレーションの可視化,電気通信大学情報理工学研究科 修士論文,2012

[3] Hongly Va,Min-Hyung Choi,Min Hong,”Real-Time Cloth Simulation Using Compute Shader in Unity3D for AR/VR Contents”,Multidisciplinary Digital Publishing Institute,vol.11,issue.17,no.8255,2021

[4] Francisco Sans,Rhadamés Carmona,”A　Comparison between GPU-based Volume Ray Casting Implementations: Fragment Shader,Compute Shader, OpenCL,and CUDA”,CLEI ELECTRONIC JOURNAL,vol.20,no.2,2017