シミュレータ教材開発に関する一提案

1532040　岡本　悠祐　　指導教員　須田 宇宙 准教授

1　はじめに

シミュレータ教材は不可視現象を可視化する教材であり，イメージが困難な事象の理解を促す手法として有用である．e-Learning の普及に伴いシミュレータ教材の活用の場が増加し，様々な分野に対応したシミュレータ教材が求められ，処理速度だけでなく，効率的な開発手法が必要とされている．

本研究室では昨年，高負荷なシミュレータ教材にプログラマブルシェーダを利用し，GPU で動作された，CPU との処理速度を比較する研究が行われた [1]．その結果，教材の処理速度は大幅に向上したが，ソースコードの変更箇所が多く，複雑なため，容易に実装できないという問題点が浮上した．

この問題点は WebWorker によるマルチスレッド化と，サブスレッドから直接描画を行う OffscreenCanvas を利用することで緩和できると考られる．

そこで本研究では，OffscreenCanvas を利用したシミュレータ教材の開発を行い，ソフトウェアの生産性と処理速度の観点から有用性の検証を行うことを目的とする．

2　比較に用いるシミュレータ教材

OffscreenCanvas の有用性の検証を行うにあたり，昨年度の卒業研究で使われた複数の音源から発せられる音場シミュレータを参照した．昨年度に開発されたプログラムの内，JavaScript のみで書かれたプログラムを手法 1，GLSL（OpenGL Shading Language）で書かれたプログラムを手法 2 とする．

手法 1 のフローチャートを図 1 に示す．1 ピクセルごとに音源から発せられる音圧を求め，この処理を 1 フレームごとに繰り返し演算を行い，音圧を明暗で表現する．

手法 1 をマルチスレッド化したものを手法 3 とする．手法 3 のフローチャートを図 2 に示す．演算をサブスレッドで実行するため，サブスレッドを多数作成することで高速化が期待できる．しかし，サブスレッドでは描画を行えないため，メインスレッドに演算結果を送信する必要が生じる．

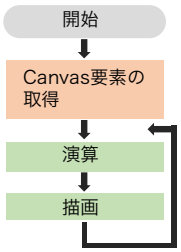

図 1: 手法 1 のフローチャート

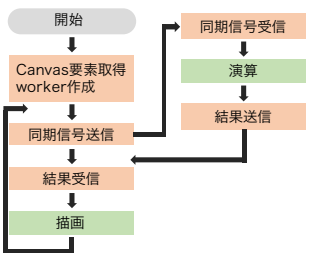

図 2: 手法 3 のフローチャート

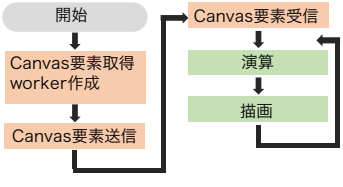

図 3: 手法 4 のフローチャート

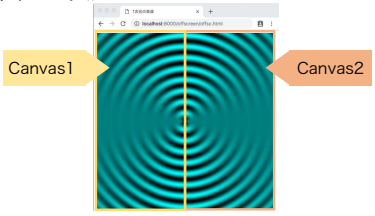

図 4: 開発したシミュレータ

手法 3 に OffscreenCanvas を取り入れ，サブスレッドから直接描画する手法を手法 4 とする．手法 4 のフローチャートを図 3 に示す．

手法 4 を用いて開発したシミュレータを図 4 に示す．Canvas 要素を縦に 2 分割し，それぞれ別スレッドで計算・描画を行っている．

3　検証

4 つの手法の処理時間と，プログラムの追加・変更箇所の比較を行った．処理時間は CPU：Intel Core i5 3.2GHz，GPU：NVIDIA GeForce GT 775M を搭載したパソコンを利用し，10,000 回演算を行う時間を計測した．その結果を表 1 に示す．

手法 3，手法 4 は複数のスレッドを用いて演算処理を行っているため処理速度が向上した．

図 1〜3 より手法 3 はメインスレッドからサブスレッドへ演算データの送信とスレッドへ結果を送信する処理を追加した．

一方手法 4 はサブスレッドへ Canvas 要素の送信を行い，演算・描画処理を行った．演算と描画が同じサブスレッド内で完結するため，繰り返しデータの送受信を行う必要がなく，処理 1 のソースコードの原型を崩さずに移植が行えた．

プログラムの変更箇所は HTML でキャンバス要素を追加，描画範囲の指定を行うことで実装できた．

表 1: 各種法の処理時間と時間比

手法	処理時間 (s)	時間比
1	206.6	1
2	128.6	0.62
3	164.3	0.8
4	98.0	0.47

4　おわりに

本研究ではシミュレータ開発における OffscreenCanvas の有用性を生産性と処理速度の観点から検証した．手法 3，手法 4 共に処理速度が向上し，サブスレッドで演算だけ行うことより容易に実装が可能であることがわかった．今後は OffscreenCanvas を利用したシミュレータ教材の開発が増加することが予測される．

参考文献

[1] 中嶋　大貴:“GPU 利用によるシミュレータ教材の演算速度”, 平成 29 年度卒業論文,(2018)