OffscreenCanvas を用いたシミュレータ教材の開発

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と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 に適用させ，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える．

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　開発したシミュレータ教材

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．

本研究では 4 つの手法の比較を行う．Worker などの手法を用いずに，シミュレータの演算，描画を JavaScript で行ったプログラムを処理 1 とする．手法 1 の演算，描画をシェーダを用いて GPU に処理させたプログラムを処理 2 とする．処理 1 に WebWorker を利用しマルチスレッド化を行ったプログラムを処理 3 とする．処理 3 を利用しつつ，OffscreenCanvas を適用したプログラムを処理 4 とする．

図 1 は実際に開発したシミュレータである．音圧の描画を Canvas に行う．その際，描画に用いられる Canvas を 2 つに分け，負荷を分散させた．

図 2 は手法 1 のフローチャートである．演算では，1 ピクセル，1 フレームごとに音源から発せられる音圧を求める．この処理を 1 フレームごとに繰り返し行う．描画では演算で得られた数値を基に Canvas に描画を行う．Canvas に描画する際，1 フレーム前に描画された内容の削除，演算結果を基に描画を繰り返し行う．

図 3 は手法 3 のフローチャートである．main で Worker を定義することで，手法 1 をマルチスレッド化した．その際，main から Worker へデータの送信が必要となり，追加した．また，WebWorker では描画を行うことができないため Worker で行った演算の結果を main に送信し，main で描画を行う必要があった．

図 4 は手法 4 のフローチャートである．手法 3 を利用し，worker 内での描画を可能とした．そのため，Worker では演算の結果を main に送信していたが，演算結果の送信を必要とせず，Worker 内で演算と描画を完結させることができる．

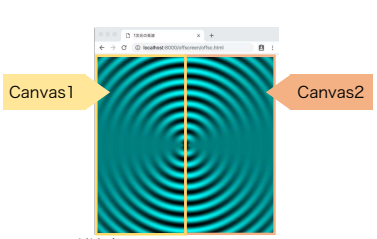

図 1: 開発したシミュレータ

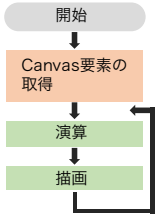

図 2: 手法 1 のフローチャート

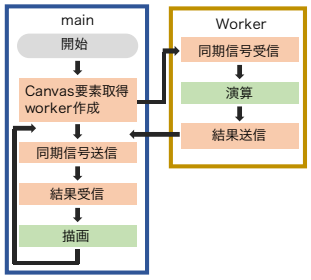

図 3: 手法 3 のフローチャート

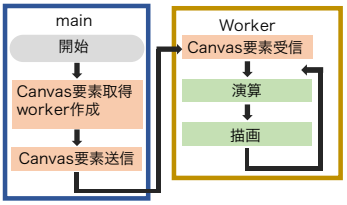

図 4: 手法 4 のフローチャート

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て，10,000 回演算を行う時間を計測した．手法 1 の処理に費やした時間を基準として時間比を求めた．その結果を表 1 に示す．

表 1: 各種法の高速化率

手法	処理時間 (s)	時間比
手法 1	206.608	1
手法 2	128.652	0.622689
手法 3	98.094	0.474786
手法 4	164.368	0.795557

手法 1 がシングルスレッドで動作するのに対し，手法 3，手法 4 は 2 つ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 2〜4 より手法 3・手法 4 のプログラムは，main から Worker へデータの送信を追加した．更に，手法 3 に関しては Worker から演算の結果を main に送信を行う処理を追加した．手法 3 は手法 1 の演算と描画が分離しているため，演算が終わる度に結果を送信する必要があり，main と Worker 間で送受信が何度も行われた．一方手法 4 は，手法 1 の演算と描画が同じ Worker 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すことなく移植が行えた．プログラムの変更箇所は HTML でキャンバスを追加，描画範囲の指定を行うことで実装できた．

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．シングルスレッドで動作するプログラムより処理速度が向上し，WebWorker より容易に実装が可能なことがわかった．今後は更なる処理速度の向上と，生産性の向上が求められる．

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)