OffscreenCanvas を用いたシミュレータ教材の軽量化

1532040　岡本　悠祐　　指導教員　須田 宇宙 准教授

1　背景

シミュレータ教材は不可視現象を可視化する教材である．そのため，イメージが困難な事象への理解促進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躍の場が増加し，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．

磁場や音場などの波形を可視化するシミュレータにおいて FTDT 法が広く用いられているが演算回数が多いことから処理速度は決して速くない．

本研究室では昨年，FTDT 法を用いたシミュレータ教材に GPU を適用し処理速度を比較する研究を行った．その結果，大幅な処理速度向上が得られたが，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．

その問題点は JavaScript をマルチスレッド化，描画を行う OffscreenCanvas を利用することで緩和されると考える．

本研究では OffscreenCanvas を利用した FTDT 法を用いたシミュレータ教材の制作を行い，生産性と処理速度を昨年制作したシミュレータ教材と比較することで，有用性の検証を行うことを目的とする．

2　WebWorker と OffscreenCanvas

2.1　WebWorker

WebWorker は JavaScript でマルチスレッド処理を可能にする API である．Worker を利用するには処理ごとに Worker を分け，実行するにはサーバーを利用する必要がある．また，Worker 上で描画を行うことができないため描画処理を行いたい場合は，Worker を定義するメインのスレッドに対して描画処理を渡す必要がある．

2.2　OffscreenCanvas

OffscreenCanvas は WebWorker を利用して Worker 上で描画処理を可能にする API である．

基本ルールは WebWorker と同様，サーバーを建てる必要がある．事前に Canvas タグを HTML ないし JavaScript で生成し,Warker に処理を譲渡，描画まで行うため，Worker より必要な処理が少なく手軽に実装できる．

3　検証方法

平成 29 年度本学卒業論文である「GPU 利用によるシミュレータ教材の演算速度」を拡張し昨年のシミュレータと演算速度を比較，変更箇所を比較することで OffscreenCanvas の実用性の検証を行う．

１つのキャンバスを左右に分割することで処理，描画の負荷を分散させ，1,000 回計算するごとに時間を計測し，それを 10,000 回まで行った．計測では研究室で使用している Mac を用いた．また，OffscreenCanvas の基である Worker の処理速度も計測した．

3.1　制作したシミュレータ教材

図 1 は複数の音源からから発生する音場を可視化したシミュレータ教材である．

図 1: 制作したシミュレータ

4　検証結果

CPU を基準とした処理速度を式 1 に当てはめて求めた．

キャンバスを分割した WebWorker と OffscreenCanvas は処理速度が遅い方を参照する．その結果は表 1 である．

$$\frac{\text{各手法 } 10{,}000 \text{ 回の処理速度}}{CPU\,10{,}000 \text{ 回の処理速度}} = \text{処理効率} \qquad (1)$$

表 1: 各種法の処理速度比較

GPU	1:	0.810988
OffscreenCanvas	1:	0.80789763
WebWorker	1:	62.52562982

OffscreenCnavas の変更箇所は分割したキャンバスの範囲指定と，キャンバスの生成元である HTML のキャンバスを増やすだけで，ソースコードを変更することはほとんどなかった．GPU の処理速度と遜色ない処理速度を実現できたため，OffscreenCanvas を用いたシミュレータ教材は FTDT 法において有用であるといえるのではないのだろうか．

5　終わりに

本研究では OffscreenCanvas の FTDT 法を用いた音場シミュレータへの有用性を検証した．Worker の処理速度の低さと，GPU の処理速度と遜色ない結果が得られたのは正直以外だった．しかし，実装するだけであれば容易だが，シミュレータとして成立させるため，リアルタイムで数値を変更する場合は，追加箇所が多くなった．したがって，今後は追加箇所の知識の共有が必要になるだろう．

参考文献

[1] 川勝 研太郎:“オフスクリーンキャンバスを使った JS のマルチスレッド描画 – スムーズなユーザー操作実現の切り札”,https://ics.media/entry/19043,2018/12/23 参照