4K-2

Web Workersを使用した対話型アニメーションソフトウェアの並列化

渡部　裕太[†]　岡本　秀輔[†]　小花　聖輝[‡]　鎌田　賢[§]　米倉　達広[§]

成蹊大学理工学部情報科学科[†]　成蹊大学理工学研究科理工学専攻[‡]　茨城大学工学部情報工学科[§]

1. はじめに　本研究で使用している対話型アニメーションソフトウェアは、Web ブラウザ上で動作する JavaScript で書かれていたが、アニメーション内にあるオブジェクトの数が増加すると、アニメーションの描画間隔が間延びする問題があった。この問題を改善するために、Web Workers という機能に注目した。Web Workers を使用すると、JavaScript で複数の処理を並列に実行できる。これによってソフトウェアの処理を並列化することで、間延びする問題が改善できると考えた。

2. Web Workers の概要　HTML5 の機能である Web Workers を使用すると、ユーザインタフェースの操作などを行うメインスレッドとは別に、計算用のワーカスレッドをバックグラウンドで実行できる。メインスレッドで行っている高負荷な繰り返し処理を、ワーカに任せることで、プログラムの効率性を高めることができる。

メインスレッドとバックグラウンドで動作するワーカは、図 1 のようにメッセージを送り合い、データの通信を行う。メインスレッドとワーカ間のメッセージ送信は、postMessage()を使用して行い、引数に送りたいデータを指定する。メッセージの受信は、onmessage を使用して行う。このときスレッド内で event.data を参照すると、送られてきたデータを受け取れる。

一方で、Web Workers には、注意点がいくつかある。ワーカ内では利用できる機能やオブジェクトに制限があり、直接ユーザインタフェースを操作することや、DOM に関する処理ができない。ワーカはメインスレッドとデータを共有することができず、その代わりにメッセージを送り合い、データの通信を行う。

3. ソフトウェアの並列化　まず、本研究で使用している対話型アニメーションソフトウェアの概要を説明する。このソフトウェアは、多数のオブジェクトが動くアニメーションを表示する。それぞれのオブジェクトは、あらかじめ設定された状態遷移図に従って動作し、他のオブジェクトとの衝突などの要因で動きが変化する。並列化前のプログラムの処理手順は、図 2 のようになる。

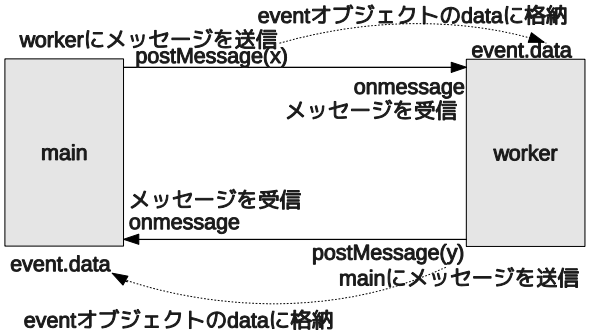

図 1:メインスレッドとワーカの通信概要

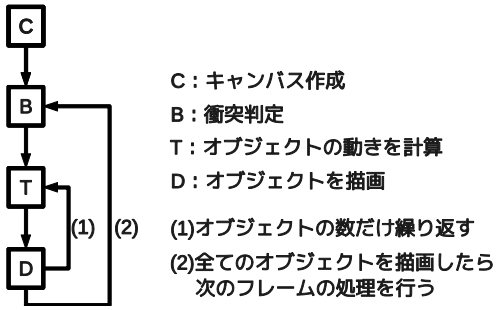

図 2: 逐次プログラムの処理手順

1. アニメーション描画用のキャンバス作成。
2. 全てのオブジェクトの衝突判定を行う。
3. オブジェクトの動きを計算する。
4. オブジェクトをキャンバスに描画する。
5. 3,4 の処理をオブジェクト数だけ繰り返す。
6. 1 フレーム分の全てのオブジェクトを描画したら 2 の処理に戻り、次のフレームの処理を行う。

JavaScript で書かれたこのプログラムは、逐次的に処理を行っているため、オブジェクトの数が多くなると、その動きの計算処理に多くの時間がかかってしまう。これにより、アニメーションの描画間隔が長くなるという問題点があった。この問題点を解決するために、Web Workers を使用して処理の並列化を行った。

A Parallelization of Interactive Animation Software with Web Workers

† 「Science and Technology, Seikei University」

‡ 「Graduate School of Science and Technology, Seikei University」

§ 「Engineering, Ibaraki University」

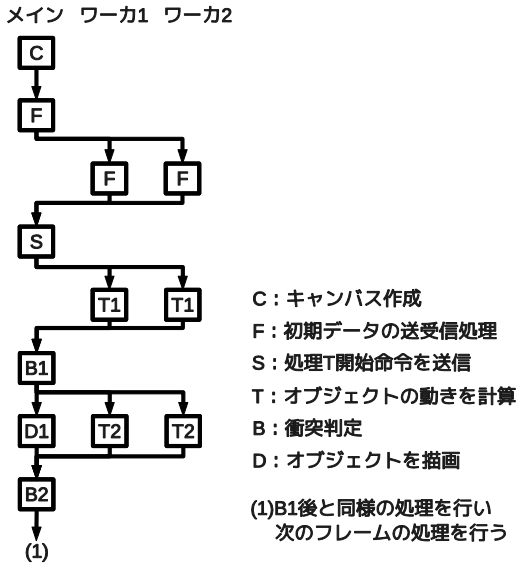

図 3:並列プログラムの処理手順(ワーカ 2 個)

並列化したプログラムの処理手順は、図 3 のようになる。

1. アニメーション描画用のキャンバス作成。
2. メインスレッドからワーカに、並列化に伴い必要になる初期データを送信する。
3. 各ワーカで初期データのセットを行い、メインスレッドに完了報告をする。
4. 全てのワーカから完了報告を受け取り後、ワーカで最初の動きの計算を行わせる。
5. 各ワーカで担当のオブジェクトの動きの計算を行い、全ての計算が終わったら、メインスレッドに描画用のデータを送る。
6. 全てのワーカから描画用のデータが送られてきたら、それを元にメインスレッドで衝突判定を行う。
7. 衝突判定結果を各ワーカに送り、各ワーカで次のオブジェクトの動きの計算を行わせる。メインスレッドではオブジェクトの描画を行う。
8. 1 フレーム分の全てのオブジェクトの描画が終了し、各ワーカから描画用のデータが送られてきたら、再び衝突判定を行う。
9. 7、8 の処理を繰り返し、次のフレームの処理を行う。

このプログラムでは、オブジェクトの描画に、HTML5 の機能の一つである Canvas[1]を使用しているので、オブジェクトの描画はメインスレッドで行わせることにした。オブジェクトの数が多くなると、その動きの計算処理に時間がかるので、この部分の処理を複数のワーカを使用して並列化した。また、メインスレッドでオブジェクトの描画を行っている間に、ワーカで次のオブジェクトの動きの計算を行わせ、オブジェクトの描画と動きの計算処理を並列化した。衝突判定を行うには、全てのオブジェクトのデータが必要になるので、各ワーカからの描画情報を元に、メインスレッドで行わせた。

表 1: 1 フレームの計算と描画時間(単位:秒)

追加したオブジェクト数	ワーカ未使用	ワーカ 2 つ使用
0	0.022	0.024
100	0.035	0.042
200	0.055	0.083
400	0.113	0.134
800	0.251	0.217
1600	0.712	0.394

4. 評価と考察 並列化したプログラムを評価するために、実験を行った。並列化前のプログラムと、ワーカを 2 つ使用して並列化したプログラムにおいて、アニメーション 1 フレームの計算と描画にかかる時間を調査した。このとき、アニメーション内のオブジェクトを 100、200、400、800、1600 と増やしていき、オブジェクト数によって、どのようにその時間が変化するか調査した。今回実験に使用したマシンの仕様は、OS が fedora16、CPU が Intel Core i7-2630QM CPU 2.00GHz、メモリが 8GB である。

結果を表 1 に示す。並列化したプログラムは、オブジェクト数が多くなっても、アニメーション 1 フレームの計算と描画にかかる時間が、並列化前と比べてあまり増加しないことが確認でき、オブジェクトが増加したときの描画間隔の間延びが、少なくなることが確認できた。

また、ワーカを 4 つ使用して並列化したプログラムでも同様の実験をしてみたところ、ワーカを 2 つ使用して並列化したプログラムとほぼ同じ結果となった。これは、ワーカで行っているオブジェクトの動きの計算が、メインスレッドで行っている描画処理よりも早く終了してしまったからだと考えられる。描画処理はメインスレッドで行うしかないため、現時点ではこれ以上効率を良くするのは、難しいと思われる。

5. まとめ 本研究では、Web Worker を使用して対話型アニメーションソフトウェアの並列化を行った。その結果、アニメーション内のオブジェクト数が多くなると、並列化前よりも効率が良くなることが確認できた。

今後の課題としては、メインスレッドで行う必要があるオブジェクトの描画処理を、より効率よく行う手法がないか調査し、この部分の効率性を良くすることがあげられる。

参考文献

[1] Canvas リファレンス - HTML5.JP, “http://www.html5.jp/canvas/ref.html”