1C-7

ウェブブラウザ向けページ切替え時間短縮手法

羽藤　淳平、 虻川　雅浩
三菱電機株式会社　情報技術総合研究所

1. はじめに

本稿では Web ブラウザのページ切替え時間の短縮方式について述べる。

Web ブラウザのページ切替え時間の主要要素にリソースロード時間があり、リソースの先読みでページ切替え時間を短縮する方式が従来よりある。しかし、WebGL[1]を用いたコンテンツの場合、リソースロード後の処理時間が長く、従来方式だけでは十分な結果が得られない。

そこで先読み後に WebGL 初回描画までをページ切替え前に実行する方式で高速化を実現した。本方式を実際のコンテンツを用いて評価し、従来より最大で 10 倍の高速化する結果を得た。

2. 背景

HTML5 は 2011 年 5 月に W3C によって Last Call が宣言され、技術的な仕様がほぼ確定した。それをうけデスクトップ PC 向けのウェブアプリケーションだけではなく、組込み機器の HMI への HTML5 適用検討を各社が進めている。組込み機器向け OS である Firefox OS[2]は HTML5 によって HMI を実現するなど、今後も HTML5 の組込み機器への展開は進むと予測される。

しかし HTML5 はネイティブ実装と比較して処理パフォーマンスが低い課題があり、主要な要因にランタイム環境によるオーバーヘッドがある。この課題については JavaScript 実行速度がネイティブ実行時間の 1.5 倍にまで短縮[3]されるなど、主に Web ブラウザベンダーによって解消されつつあり、本稿では取り扱わない。

3. 課題

Web ブラウザの処理パフォーマンスに関する他の要因にページ切替え処理があげられる。

Web ブラウザはページ切替え開始直後にリソースロードを開始するため、ページ切替え時間にリソースロード時間が含まれる。必要リソースサイズが大きい場合、ページ切替え時間が無視できなくなる。この課題に対し、ページ切替え前にリソースを先読みし、キャッシュ化する事で、リソースロード時間を短縮する方式がある(図 1)。

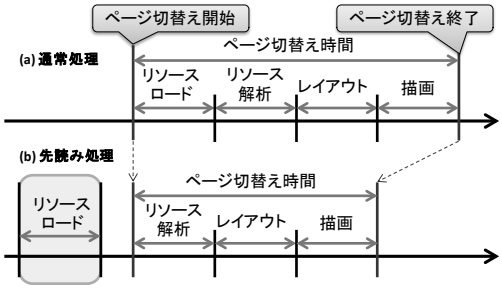

図 1 リソース先読み方式

しかし WebGL を用いたコンテンツの場合、リソースロード後に JavaScript で更にリソースをロードし、画面を構築する。その処理負荷が高い場合、不完全な画面が数秒に渡り表示され、先読みだけでは効果が十分でない（図 2)。

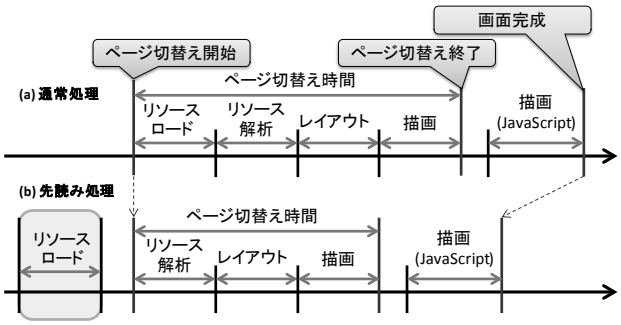

図 2 WebGL コンテンツの先読みの課題

4. 提案方式

4.1. ページ切替え前描画

上記課題を解決するため、ページ切替え前に JavaScript による画面構築処理を別フレームで実行し、ページ切替えはフレームの切替えのみで対応する方式を提案する(図 3)。

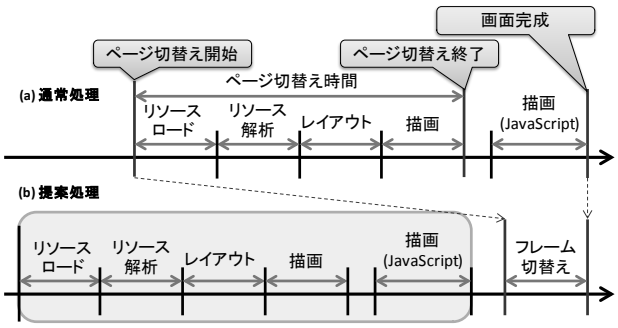

図 3 提案方式によるページ切替え処理

Shortening Page Transition Time Method for Web Browser
Jumpei Hato, Masahiro Abukawa, Information Technology R&D Center, Mitsubishi Electric Corporation

本方式はページ切替え前コンテンツと並行して次コンテンツを表示する事に等しい。そのためページ切替え完了後のイベントや JavaScript 処理も同様に行われ、WebGL に関する処理も非表示フレーム上で適切に実行可能である。

しかしコンテンツを複数個同時に実行するため、コンテンツ数に応じたメモリが必要である。加えてイベント、JavaScript 処理が可能なため、CPU 使用率の増加と、ユーザの認知できない状況で処理が進む。メモリ量と CPU 使用率の課題は将来における H/W 性能向上で軽減する事が見込めるが、認知できない状況下の処理は対処が必要である。

4.2. イベント処理の遅延実行

認知できない状況下で動作してはいけない処理はアプリケーションのロジック処理である。

一般的な Web アプリケーションの処理フローを想定した場合、アプリケーションロジック処理はページ切替え後にイベント駆動する。

そこで先読み中の JavaScript 処理開始前にイベント関連の組込み関数を、実行された事を引数と共に時系列順に記録する独自定義関数に差替え(機能差替え処理)、ページ切替え開始直後に差替えた機能を元に戻し(機能復元処理)、記録からイベント処理を実行(遅延実行処理)する事でユーザの認知できない状況下でのアプリケーションロジック動作を回避する(図 4)。

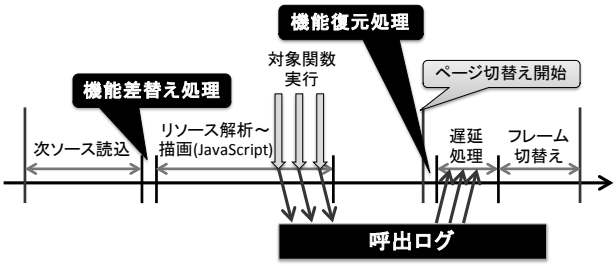

図 4 繰返し処理の遅延実行方式

機能差替え対象はイベント発行関数 dispatchEvent()、イベントリスナー登録関数 addEventListener()、タイマー起動関数 setInterval()、setTimeout()および、Web ブラウザ描画頻度に同期したタイマーの起動関数 requestAnimationFrame()とした。

ただし requestAnimationFrame()の差替え関数は初回実行のみ、requestAnimationFrame()を実行し、2 回目以降の処理を記録する制御とし、初回描画までをページ切替え開始前に完了させる。

5. 実装

Google Chrome を使用 Web ブラウザとして本方式を実装した。

本実装は Chrome の拡張機能とコンテンツに埋め込む JavaScript ライブラリとして実装し、既存コンテンツであってもライブラリ追加のみで適用可能とした。

機能差替え処理、機能復元処理などの Web ブラウザの挙動に依存して駆動すべき処理は拡張機能側でイベントを捕捉し処理を開始させる。先読み処理はページ切替え直後にライブラリ側で a タグの href 属性値から html ファイルの URL の値のみを抽出して自動先読みする方法と、明示的に先読みページ URL を登録する方法の 2 通りを用意した。

6. 評価

本実装の性能計測を行った。実行環境は Windows7 PC(Intel Core i7 3.4GHz)とし、計測用コンテンツは HelloRacer[4]を用いた。本方式適用前は Chrome Developer Tools の Timeline で、Send Request から最終 Finish Loading 後の Animation Frame Fired までの時間を計測した。適用後はリソースロードを計測に利用できないため拡張機能の BeforeRequest イベント受信からフレーム切替え完了までを計測した。その結果を図 5に示す。

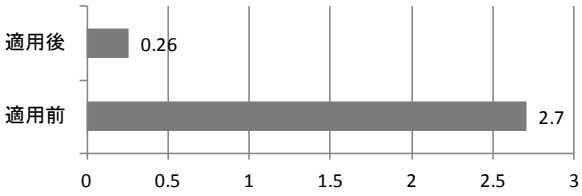

図 5 ページ切替え時間比較(sec)

この結果は 10 回の計測結果の平均値である。ページ切替え時間が約 1/10 になっており、本方式が有効である事を示している。

7. おわりに

本稿では Web ブラウザのページ切替え速度高速化方式ついて提案し、その有効性を示した。今後、組込み機器での評価・最適化進める。

参考文献

[1] KHRONOS GROUP, "WebGL -OpenGL ES 2.0 for the Web", http://jp.khronos.org/webgl/
[2] Mozilla, "Firefox OS", http://www.mozilla.jp/firefoxos/
[3] Alon, Robert, "Gap between asm.js and native performance gets even narrower with float32 optimizations", https://hacks.mozilla.org/2013/12/gap-between-asm-js-and-native-performance-gets-even-narrower-with-float32-optimizations/
[4]HelloRacer™ WebGL http://helloracer.com/webgl/