Vol.2014-AVM-84 No.6
2014/2/21

HTML5 Canvas および SVG における 自動選択アルゴリズムを用いた描画パフォーマンスの最適化

山本愛佑子[†1]　　渡辺裕[†2]

HTML5 では，ビットマップ形式のデータを扱う Canvas，ベクター形式のデータを扱う Inline SVG がサポートされ，JavaScript ベースで HTML 内に図形や画像を直接記述できる．しかし，Flash，Java などのプラグインを使用した場合と比べ，描画パフォーマンスが劣るという問題点がある．本研究では，Canvas および SVG における従来の描画パフォーマンス高速化手法の比較を行った．さらに，描画する要素に対して最適な手法を自動選択し，適切な手法で描画するアルゴリズムを提案する．
キーワード　HTML5，canvas，SVG，レンダリング，JavaScript

Optimizing Rendering Performance Using Automatic Selection of HTML5 Canvas or SVG

AYUKO YAMAMOTO[†1]　HIROSHI WATANABE[†2]

In HTML5, Canvas dealing with data of the bitmap and Inline SVG dealing with Data for vector are supported. Thus we can directly describe image or graphic in HTML, by the JavaScript. However, drawing performance is not good compared with the plugins such as Flash or Java. In this paper, we sinvestigate the performance of traditional approaches to increase drawing speed. Finally, we propose an algorithm to select the best approach automatically.
keywords：　HTML5, Canvas, SVG, rendering, JavaScript

1. はじめに

World Wide Web Consortium（W3C）は，2008 年に草案を公開した HTML5 について，2012 年に仕様策定を完了し，2014 年半ばに最終勧告を行うとしている．勧告を見据え，既に多くの開発者が，HTML5 を積極的に採用している[1]．HTML5 において，注目されている機能の一つとして「Canvas」が挙げられる[2]．これにより，Flash や Java などのプラグインを使わずに，JavaScript ベースで直接ブラウザ上にビットマップ形式の図を描くことができるようになった．また，同じく画像関連技術として「inline SVG」もサポートされ，HTML 内にベクター形式の図を直接記述可能となった．この二つにはアニメーション機能はない．しかし，JavaScript と組み合わせて移動，拡大及び縮小，回転，画像フィルタ，再描画などを行えば，動きを表現することも可能となる．プラグインを使わず HTML5 での記述となるため，クロスブラウザ対応であることが特徴である．特に iPhone，iPad などは Flash をサポートしていないため，Canvas での描画は今後も積極的に採用されていくと考えられる．その一方で，HTML5 の描画のパフォーマンスは従来の Flash には劣り，JavaScrip もスクリプト言語であるためにレスポンスが処理が遅い．そこで本研究では，Canvas および SVG における従来の描画パフォーマンスの比較を行い，高速化手法を体系化している．その上で描画する要素に対して最適な描画手法を自動選択し，適切な手法で描画するアルゴリズムを提案する．

2. 従来ライブラリの比較

2.1 代表的なライブラリ

Canvas 及び SVG でよく使われる代表的なライブラリについて，特徴をまとめ，ベンチマークテストを行い性能を比較した．

(1) **create.js(easel.js)**
Create.js は，Flash コンテンツ制作者として著名な Grant Skinner 氏が開発した，HTML5 でのリッチコンテンツ制作用の JavaScript ライブラリである[3]．Canvas に描画するための Easel.js, Canvas に描画されたオブジェクトのアニメーションを行う Tween.js, 素材のロード処理をサポートする Preload.js, Audio を扱う Sound.js の 4 つのライブラリから成る．Flash の ActionScript に似た API となっており，Flash からの書き出しにも対応している．

(2) **enchant.js**
enchant.js は，ユビキタスエンターテインメント(UEI)が開発した，スマートフォンゲーム向けのゲームエンジンライブラリである[4]．元々は DOM ベースで作られていたが，バージョン 0.6.0 からはよりリッチな表現ができるよう Canvas ベースに変更された．

(3) **arctic.js**
arctic.js は，DeNA が開発した，スマートフォン向けブラウザゲーム開発支援ライブラリである[5]．Ajax 対応となっており，リロードせずに JavaScript の通信機能で処理を進めることができる．

(4) **Pixi.js**
pixi.js は Mat Groves 氏が開発した，WebGL と Canvas に対応した 2D ゲームレンダリングライブラリである[6]．上述のライブラリほど有名ではないが，WebGL 対応のため GPU でのレンダリングが可能である．

2.2 ベンチマークテスト

2.1 で挙げた 4 つのライブラリと Flash の性能比較を行う．次に挙げる条件を複合したデモで検証を行う．

- 移動
- 回転
- 拡大，縮小
- 親子構造
- 透明度
- ビットマップ画像の表示

†1 早稲田大学基幹理工学部情報理工学科
Department of Computer Science and Engineering, Waseda University
†2 早稲田大学 大学院 国際情報通信研究科
Graduate School of Global Information and Telecomunication studies,Waseda University

フレームレートを計測し，30fps を下がったときのオブジェクトの個数を比較した．結果を図 1 に示す．

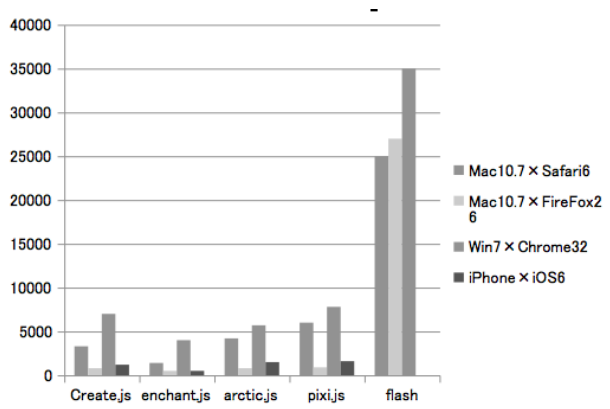

図 1　各種ライブラリのベンチマーク結果
Figure 1　Benchmark test of JavaScript library

各種 JavaScript ライブラリのスピードは Flash に劣っていることが確認できる．JavaScript ライブラリの中では pixi.js が最も高速で，続いて Create.js と arctic.js がほぼ同じ性能を示し，enchant.js が最も遅いという結果となった．ただし，この実験では pixi.js の WebGL が機能していない．WebGL を活用した場合，Flash と同程度もしくはそれ以上のフレームレートとなると考えられる．しかし，pixi.js の WebGL を機能させると生成時のコストが高く，描画を始めてから数十秒フリーズするという現象が発生する．このため，全体のパフォーマンスを他のライブラリと比較することが難しい．

3. 高速化手法

Canvas 描画を高速化するコーディング手法について以下にまとめる．

3.1 requestAnimationFrame を使う

描画を繰り返し行いアニメーションを行う場合，従来は setTimeout もしくは setInterval を利用することが主流となっていた．この二つは再描画の準備ができていなくても，必ず一定の FPS で再描画が実行される．一方で requestAnimationFrame を使うと，一定の FPS 以内で描画の準備が整ったタイミングで再描画が実行されるため，メモリの消費を抑えることができる．

3.2 プレレンダリングを行う

プレレンダリングは，頻繁に更新しない図に対して有効である．別の Canvas 要素に図を描画しておき，必要なときにメインで使う Canvas 要素から drawImage を使って呼び出しレンダリングを行う．

3.3 複数のオブジェクトを一括して呼び出す

複数のオブジェクトを描画する場合，1 つのオブジェクトに対して 1 回の描写呼び出しを行うよりも，複数のオブジェクトを含むパスを作成し，そのパスを描画した方が効率が上がる．

3.4 差分のみレンダリングを行う

レンダリング領域は少ないほど負荷が小さいため，画面全体をクリアせずに差分のみ再描画を行う．

3.5 複数のレイヤーに描画する

Canvas はレイヤーとして重ねて使用することができる．背景部分とアニメーション部分に分割することができる場合は、レイヤーを分けてレンダリングするとパフォーマンスが向上する．

4. 自動選択アルゴリズム

4.1 Canvas と SVG の比較

前章まで Canvas の JavaScript ライブラリについて述べてきたが，ここで Canvas と SVG の比較を行う．それぞれの特徴について以下の表 1 に示す．

	Canvas	SVG
ベース	ピクセル	ベクター図形
要素	HTML 要素	DOM の一部
変更	スクリプト	スクリプト／CSS
オブジェクト数	多数のとき 高パフォーマンス	少数のとき 高パフォーマンス
オブジェクトの大きさ	小さいとき 高パフォーマンス	大きいとき 高パフォーマンス
アニメーション	リアルタイム描画が得意	滑らかな動きが得意

表 1　Canvas と SVG の比較
Table 1　Comparison of SVG and Canvas

4.2 Canvas と SVG を自動的に選択するアルゴリズム

第三章で，複数の Canvas レイヤーを重ねることが可能であると述べたが，同じ様に SVG のレイヤーを重ねることも可能である．そこで，各オブジェクトに対して，以下の指標を用いて Canvas と SVG のどちらを使って描画を行うかを自動的に選択する．

- 描画する画面のサイズ
- オブジェクトの数
- オブジェクトの複雑さ
- アニメーションの有無
- フィルタ処理の有無

5. おわりに

第三章で述べた高速化手法と，第四章で述べた自動選択アルゴリズムを導入すると，従来のライブラリのメリットを全て取り入れたことになり，描画パフォーマンスの高速化を図ることができる．また，WebGL による GPU での描画のサポートが進めば，さらなる高速化に繋がると考えられる．

参考文献

[1]HTML5 Reference - W3C
http://dev.w3.org/html5/html-author/
[2]Canvas とは - Canvas - HTML5.JP
http://html5.jp/canvas/
[3]CreateJS | A suite of Javascript libraries and tools designed for working with HTML5
http://www.createjs.com/
[4]enchant.js - A simple JavaScript framework for creating games and apps.
http://enchantjs.com/
[5]Arctic.js 入門編 - Technology of DeNA
http://engineer.dena.jp/2012/04/arcticjs.html
[6]Pixi.js - 2D webGL renderer with canvas fallback
http://www.pixijs.com/