粒子描画ライブラリ particles.js の GPU オフロードによる高速実行

1140059 木山 拓弥（密山研究室）

1. 序論

Web サイト上でアニメーション処理による演出事例が増える中、開発コストを抑えた上で表現力が高く CPU 負荷の低い描画ライブラリが求められている。JavaScript で動作する幾何学粒子描画ライブラリの一つである particles.js は、開発コストの低く、表現力に優れたライブラリであるが、CPU 負荷が高いという課題がある。本研究では、CPU 負荷を抑えるために、WebGL API を用いた GPU オフロードによる particles.js の互換ライブラリの開発を行い、評価をする。

2. particles.js ライブラリ

paritcles.js を用いた実行例を図 1 に示す。

図 1 particles.js の実行例

particles.js には、3 つの課題がある。Canvas API を使用していることによるパフォーマンスの課題、読み込み時にグローバルプロパティを 3 種類用意する必要があることによるポータビリティの課題、描画･粒子計算ロジックが同一関数内で混在していることによるメンテナビリティの課題である。

3. 互換ライブラリ partixi.js の開発

パフォーマンスの改善のために、WebGL API を利用する。現在、WebGL API の仕様は 1.0 から 2.0 へのバージョン移行期にあるため、直接採用は結合度が高くなるリスクがある。そこで、WebGL API のラッパーライブラリである Pixi.js を使用したレンダラロジック実装する。

ポータビリティ向上のために、ECMAScript 2015 Modules に準拠したモジュールシステムを採用する。これによりグローバルプロパティが不要となる。

メンテナビリティの向上ために、図 2 に示す処理フローに従って、オブジェクト指向プログラミングにより開発する。これにより、処理毎の凝集度を高め、処理間の結合度を下げることが出来た。一方で、オブジェクト指向プログラミングを行うことにより、ファイル容量の肥大や処理時間の増加が新たな課題になった。

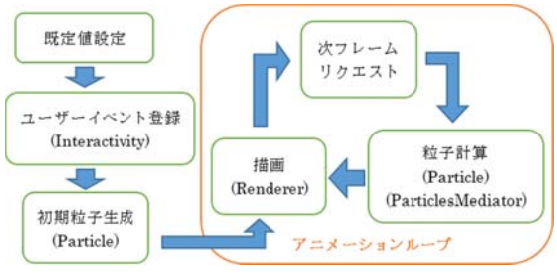

図 2 互換ライブラリの簡易処理フロー

4. 性能評価

一部の Web ブラウザにはパフォーマンスモニタが用意されており、CPU や GPU の性能評価を行うことが出来る。しかし、パフォーマンスモニタには規格や仕様が存在しないため、統一的な検証が困難である。そこで、Stats.js を用いて JavaScript 上からフレームレートを計測する方法により、性能評価を行った。

CPU/GPU の条件が異なる Windows OS の PC を 3 台用意し、ブラウザは Internet Explorer、Firefox、Google Chrome を対象として、ベンチマーク時のフレームレートを計測した。ベンチマークは粒子のみの描画と粒子＆線の描画のものを使用した。結果を図 3~図 5 に示す。Google Chrome と Firefox は傾向が似ており、性能が向上したケースが多い。一方で、IE はフレームレートが低下するケースが多い。また、PC2 においては GPU の性能が低いため速度があまり向上しなかった。

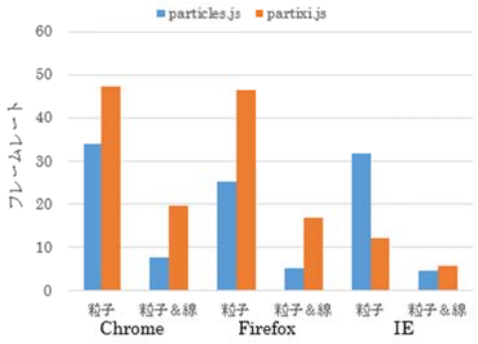

図 3 PC1 でのブラウザ毎の速度変化

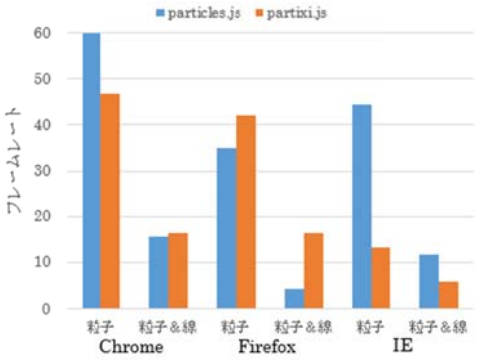

図 4 **PC2 でのブラウザ毎の速度変化**

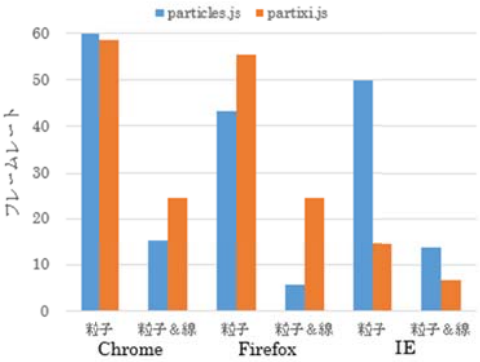

図 5 PC3 でのブラウザ毎の速度変化

5. 結論

particles.js の GPU オフロードによる高速実行のための互換ライブラリを開発した。まず particles.js の調査をし、パフォーマンス、ポータビリティ、メンテナビリティでの課題を明らかにした。互換ライブラリでは、これらの課題を克服する。性能評価比較では、ブラウザ毎で傾向が異なる点と、GPU オフロードの有効性が GPU の性能に大きく依存する点がそれぞれ明らかになった。また、オブジェクト指向プログラミングを採用したことにより、ファイル容量の肥大や処理時間の増加が新たな課題になった。新たに明らかになった課題を踏まえ、引き続き互換ライブラリの開発を行い、完成を目指す。