令和98年度 プロジェクトデザインIII

WebAssemblyを用いたグラフィクスレンダリングの高速化

4EP4-04
あもうたいき
天羽大樹

令和6年09月25日

KIT 金沢工業大学

1. はじめに – 背景と目的 –

- 将来、Webアプリケーションとして、リアルタイムに変化するxRコンテンツを作成し、動作させる際、グラフィクスレンダリングの速度が遅くなってしまうと、没入感を損なってしまう問題があると考えられる
- そのような問題に対処するために、Webブラウザ上でグラフィクスのレンダリングを早く処理するために、どのような手法を用いれば、より早く処理できるのかという取り組みを行う必要がある
- 本プロジェクトにおいては、WebAssemblyという仕組みを用いたバイナリフォーマットプログラムを使って、グラフィクスレンダリングを早く処理させるプログラムを作成する

発表の流れ

1. はじめに – 背景と目的 –
2. 用語などの簡潔な説明
3. プログラム概要
4. 性能比較評価・考察
5. むすび

2. 用語の簡潔な説明

- WebAssembly
  Webブラウザに搭載されている仮想マシンで
  実行できるバイナリコードとそれを処理するシステム全体[1]のこと。
  wasmと略して呼称されることが多いため、本スライドでも
  その略称を以後使用する。特定のプログラミング言語で記述した
  プログラムをそのバイナリコードにコンパイルしたものを
  主に指す。そのプログラムを保存したファイルは、
  wasmモジュールと呼称されることが多い。
  C,C++,Rustなどで記述したプログラムをこのWebAssemblyに
  コンパイルすることで、動作させることが可能になる。[2]

- WebGL
  Web上でグラフィクスを扱うために、ブラウザに組み込まれているAPI
  後述するWebGPUより歴史が古く関連ライブラリが豊富で、
  Three.jsやBabylon.jsといったライブラリが有名。
- WebGPU
  WebGLより高度にGPUの性能を活かし、高速なグラフィクス
  レンダリングを行うことが可能であるとされている
  ブラウザ組み込みのAPI。まだ公開されてから日が浅いため、
  著名な関連ライブラリなどはまだあまり見受けられない

2-1. なぜWebAseemblyを扱うのか

扱うメリット

WebAssemblyはWebブラウザ上でネイティブコードに近い速度で実行される[3]ため、JavaScriptでは時間がかかるであろう処理をWebAssemblyでの処理に置き替えることによる高速化が望める

WebAssemblyが得意とする処理

基本的に、WebAssemblyそのものは、整数・実数の計算しかできない[4]が、WebAssemblyが既にコンパイルされた仮想マシンコードであることを活かし、高速な演算処理を得意とする。

発表の流れ

1. はじめに – 背景と目的 –
2. 用語などの簡潔な説明
3. プログラム概要
4. 性能比較評価・考察
5. むすび

3. プログラム概要

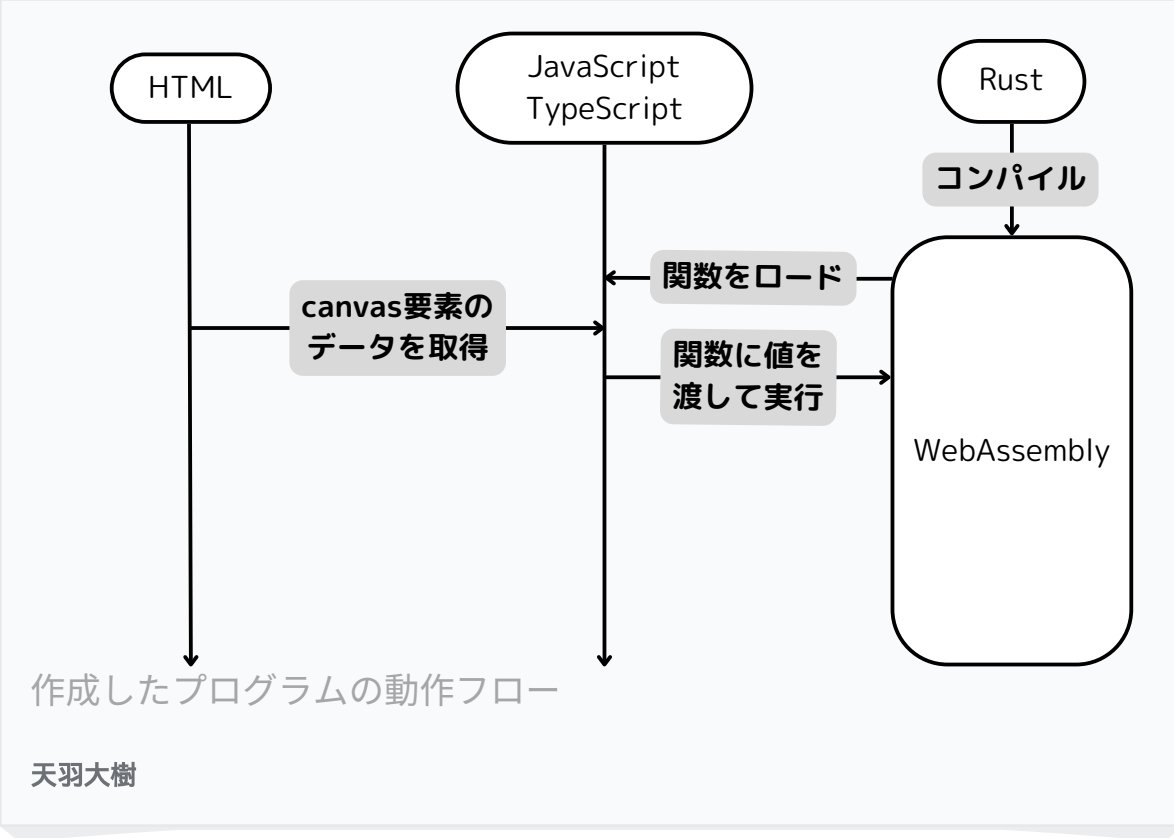
Rustでブラウザに搭載されているAPIにアクセスするプログラムを記述し、
そのプログラムをWebAssemblyにコンパイルする。
コンパイルされたプログラムをJavaScriptでロードし、実行させる。
WebAssembly側で処理するプログラムの対象として、

- 比較的、ハードウェア側に近いプログラムの処理
- 行列などを扱う演算負荷の高い処理

というような処理を指定して行う

本プロジェクトでは、2つのプログラミング言語を用いて高速化を行う

1. Rustのプログラムでグラフィクスを操作するAPIをコール
2. RustのプログラムをWebAssemblyにコンパイルする
3. WebAssemblyプログラムをJavaScript側からコールする



作成したプログラムの動作フロー

天羽大樹

3-1. 作成したWebAssemblyプログラムの概要

処理内容

WebGPU APIを通してレンダリング内容を決め、実行する内容をバッファデータとして処理し、順番に実行させる一連の処理を行う

実行される処理の結果

HTMLのcanvas要素の範囲にプログラムで指定した
矩形や図形がレンダリングされる

発表の流れ

1. はじめに – 背景と目的 –
2. 用語などの簡潔な説明
3. プログラム概要
4. 性能比較評価・考察
5. むすび

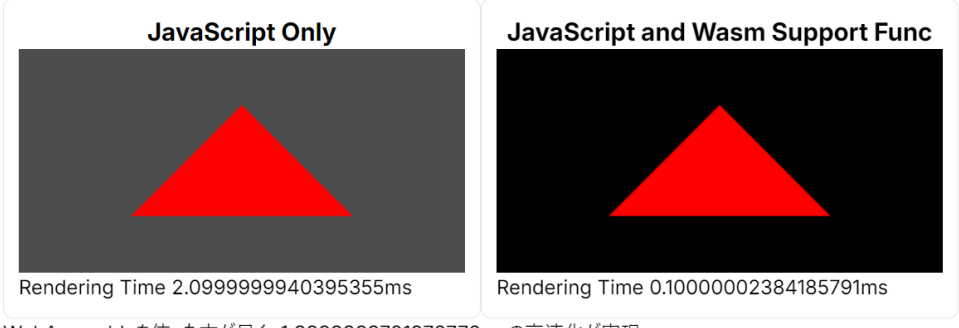
4. 性能比較評価・考察

評価方法

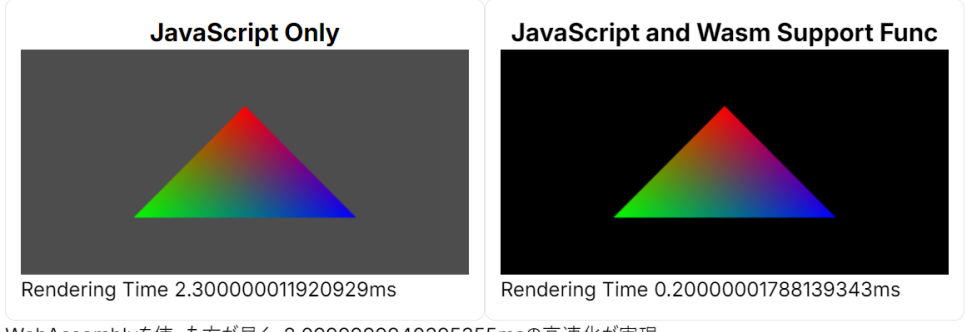
JavaScriptのみで組んだ関数と、WebAssemblyを交えたプログラムを同時に実行して、その2つの関数の実行が終了するまでの時間をJavaScriptのperformance.now関数で、実行を終了した時間から実行を開始した時間を減算することで計測し、実行速度の比較を行う

評価結果

今回の比較では、以下の結果が出た。このケースではそれほど大きな差が出なかった。
また、今回は単一の図形のレンダリングのみなので、映像ではなく画像でレンダリング時の画面を掲示する。



WebAssemblyを使った方が早く、1.9999999701976776msの高速化が実現



WebAssemblyを使った方が早く、2.0999999940395355msの高速化が実現

考察

今回は単純な単色の図形、もしくは三色のグラデーションの図形のみの描画であるため、大きな差にはならなかったと考えられる。しかし、2つの比較で、それぞれ速度差が現れているため、より内部の数値処理を増やすと、より大きな差が生まれる可能性があると考えられる

5. むすび

- Web上でのグラフィクスレンダリングを高速化するため、WebAssemblyを用いたモジュールを作成した。
- 現時点では、それほど大きな描画速度の差がないが、計算する数値が増えれば増えるほど、差が開いていく可能性があると考えられる。
- 来月の報告までに、より計算量が多くなるプログラムで比較を行う

文献

[1] 日向俊二. (2021). より速く強力なWebアプリ実現のためのWebAssemblyガイドブック (初). カットシステム.

[2] [1]と同様

[3] Gallant, G. (2022). ハンズオン WebAssembly ——Emscripten と C++を使って学ぶ WebAssemblyアプリケーションの開発方法 (初). オライリー・ジャパン.

[4] [1]と同様