

令和98年度 プロジェクトデザインIII

WebAssemblyを用いた グラフィッ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99年99月99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、グラフィッ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上でグラフィッ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、グラフィッ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. 評価
5. むすび

2. 用語の簡潔な説明

- WebAssembly

Web上で動作するバイナリフォーマットプログラムのこと。
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フォーマット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。

- WebGL

Web上でグラフィックスを扱うために、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、高速なグラフィックス
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

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

3-1. 作成したWebAssemblyプログラムの概要

引数

- ブラウザを経由して受け取ったGPUの総合的なデータ (GPU型の値)
- ブラウザを経由して受け取ったGPU装置のデータ (GPUDevice型の値)
- ブラウザが対応しているWebGPU上でレンダリングを行うにあたって利用するフォーマット (GPUTextureFormat型の値)

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

3-2. ○○○○○○処理の方法

① Javascript

あああああああああああああああああああ
いいいいいいいいいいいいいいいいいいい
うううううううううう

② Python+Tornado

あああああああああああああああああああ
いいいいいいいいいいいいいいいいいいい
うううううううううう

③ pigpio

あああああああああああああああああああああいいいいいいいいいいいいいい

いいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいい
うううううううううううう

4. 評価・考察

- このスライドでは何をどのような方法で評価したかを明記し，結果をグラフで示すこと（表よりグラフのほうが良い）．
- システムが動いている様子が見えるようにデモ映像を流すこと（デモ映像には字幕をつけたりするなどしてわかりやすくすること）．
- 評価の際は，改良の前後でどうなったかを示す．あるいは他の手法などと比較してどうなのかを示すことも必要．
- 結果について考察も示すこと．

5. むすび

- 何のために何を作成したかを改めて書く．
- 現時点での評価結果，考察を簡潔に書く．
- 来月の報告までに何をするか計画を書く．

ここからおまけ

📺 PDF ファイルと同じフォルダに demo002.mp4 があれば再生できる.

📺 YOUTUBE で再生

📺 <https://youtu.be/74agBeJxdFI>

リスト 1: test2.c

```
1 #include <avr/io.h>
2 #include <avr/wdt.h>
3 int main(void)
4 {
5     DDRC = 0x30; // PC5/4を出力ピンに設定
6     PORTC = 0x10; // PC5/4の出力をL/Hに設定
7     for (;;) {
8         wdt_reset(); // ウォッチドックタイマをリセット
9     }
```

```
10     return 0;
11 }
```

リスト 2: test2.py

```
1 from time import sleep
2 from random import randint
3
4 while True:
5     input('push ENTER key')
6     r = randint(1,6)
7     print( r )
8     sleep(0.5)
```

UNIXv1におけるタスク切り替えが行われるタイミング

① みなさん

② こんにちは

- まんじゅう
- りんご

③ お元気で
またあうひまで

```
$ gcc test.c ↵  
(*_*)  
(*_*)
```

ここで **CTL+C** を押す

謝辞 本研究はJSPS科研費21Kxxxxxxxxxx助成を受けた

文献

- [1] K.Thompson, D.M.Ritchie, "**The UNIX Time-Sharing System**", Communications of the ACM, Vol.17, No.7, 1974.
- [2] Digital Equipment Corporation: **PDP11/20-15-r20 Processor Handbook**, 1971.
- [3] T.R. Bashkow, "**Study of UNIX: Preliminary Release of Unix Implementation Document**", http://minnie.tuhs.org/Archive/Distributions/Research/Dennis_v1/PreliminaryUnixImplementationDocument_Jun72.pdf, Jun. 1972.
- [4] simh, "**The Computer History Simulation Project**", <https://github.com/simh/simh>, 参照 Mar.14, 2022.

- [5] W.Toomey, "**First Edition Unix: Its Creation and Restoration**", IEEE Annals of the History of Computing, 32 (3), pp.74-82, 2010.
- [6] Diomidis.Spinellis, "**unix-history-repo**", <https://github.com/dspinellis/unix-history-repo/tree/Research-V1>, 参照 Mar.14, 2022.
- [7] Digital Equipment Corporation: **PDP11 Peripherals HandBook**, 1972.