JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 237–242
Received: 22 June 2021
Accepted: 2 July 2021

Graphics display capabilities in web browsers

Możliwości wyświetlania grafiki w przeglądarkach internetowych

Damian Piotr Sołtysiuk *, Maria Skublewska-Paszkowska

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The article discusses the issue of displaying graphics in web browsers. A couple of methods related to its display can be distinguished. The methods discussed are: SVG, HTML5 Canvas and the WebGl graphics engine. The research was done using a dedicated web application written in Angular and TypeScript language along with the help of Two.js library for displaying 2D graphics. It concerned the analysis of the rendering time and frame rate of simple and complex elements. The frame rate animation also had two types of complexity. After analyzing all the results it concluded that, guided by the rendering time of the elements, HTML5 Canvas turned out to be the best method. On the other hand, the best method which achieves the highest number of FPS for animation is WebGl.

Keywords: graphics; web browser; graphics engine; comparison

Streszczenie
Artykuł dotyczy wyświetlania grafiki w przeglądarkach internetowych. Można wyróżnić parę metod związanych z jej wyświetlaniem. Omawianymi metodami w tym artykule są: SVG, HTML5 Canvas oraz silnik graficzny WebGl. Badania wykonane zostały przy użyciu dedykowanej aplikacji webowej napisanej w Angularze oraz języku TypeScript wraz z pomocą biblioteki Two.js służącej do wyświetlania grafiki 2D. Dotyczyły przeanalizowania czasu renderowania i liczby klatek na sekundę elementów prostych i złożonych. Animacja badająca liczbę klatek na sekundę także miała dwa typy złożoności. Po przeanalizowaniu wszystkich wyników stwierdzono, że kierując się czasem renderowania elementów, najlepszą metodą jest HTML5 Canvas. Natomiast najlepszą metodą, osiągającą największą liczbę FPS przy animacji jest WebGl.

Słowa kluczowe: grafika; przeglądarka internetowa; silnik graficzny; porównanie

*Corresponding author
Email address: damian.soltysiuk@pollub.edu.pl (D. P. Sołtysiuk)

1. Wstęp

W dzisiejszych czasach zaobserwować można gwałtowny rozwój sieci internetowych. Dużą część przesyłanych przez Internet informacji stanowi grafika [1, 2]. Obsługa wyświetlania i przesyłania grafiki w Internecie wymaga użycia przeglądarki internetowej wraz z dedykowanymi metodami do jej obsługi. Wybór odpowiedniej metody jest bardzo ważny, gdyż determinuje szybkość działania strony i tym samym liczbę odwiedzających ją internautów. Można wyróżnić trzy główne metody obsługi grafiki: SVG (ang. Scalable Vector Graphic), HTML5 Canvas oraz silnik graficzny WebGl (ang. Web Graphics Library).

Większość stron, czy też aplikacji internetowych posiada wiele elementów graficznych. Wymagana jest optymalizacja ich wyświetlania, gdyż przy wolno działających aplikacjach internetowych użytkownicy mogą zrezygnować z jej korzystania. Jest to o tyle ważne, że internauci, zamiast korzystać z wolno działającej strony wolą znaleźć taką, która będzie optymalna pod względem szybkości i walorów wizualnych. Kolejnym ważnym punktem jest tworzenie szybko działających biznesowych aplikacji internetowych, które opierają się na wykorzystywaniu grafiki. Między innymi mogą być to aplikacje giełdowe wyświetlające grafy i wykresy, które bez płynnego działania mogą powodować straty pieniężne dla użytkowników. Ważnym więc jest dobór odpowiedniej metody przy tworzeniu tego typu aplikacji, tak aby takie sytuacje nie miały miejsca. Warto tutaj także dodać, że nie każda aplikacja wymaga maksymalnej optymalizacji dla swoich przypadków użycia. Mimo że jedna z metod może być najszybsza, nie oznacza, że jest także optymalna pod względem zajmowanego miejsca, zużycia zasobów komputera, czy też początkowego czasu ładowania strony.

Aby uniknąć problematycznych sytuacji związanych z optymalizacją stron, deweloperzy muszą dobrać odpowiednią metodę do zastosowania w tworzonych przez siebie aplikacjach. Zostało to przebadane przy pomocy dedykowanej aplikacji pozwalającej na indywidualne przetestowanie wszystkich omawianych metod. Dzięki opracowanym wynikom można dobrać odpowiednią metodę do swoich potrzeb.

2. Cel i zakres

Głównym celem badań jest porównanie wydajności metod wyświetlania grafiki 2D w wybranej przeglądarce internetowej. Aby porównać te metody zbadany zostanie czas renderowania i liczba FPS (ang. Frames Per Second) na określonej liczbie elementów za pomocą dedykowanej aplikacji. Dodatkowo określona zostanie wydajność na elementach prostych i złożonych.

Zostały sporządzone dwie hipotezy badawcze i są one następujące:

1. *Metoda WebGl osiąga najlepsze wyniki czasu renderowania elementów.*
2. *Metoda WebGl osiąga najlepsze wyniki liczby FPS przy animowaniu elementów.*

3. Przegląd literatury

Na rzecz artykułu zostały znalezione odpowiednie prace naukowe opisujące badania nad technologiami wyświetlania grafiki w przeglądarkach internetowych. Badania w źródłach opierają się głównie na zbadaniu liczby FPS dla poszczególnych metod. Przedstawiają także dobrze opisane wnioski oraz wykresy ukazujące wyniki w dobrze zrozumiały sposób.

W artykule [3] autorzy zbadali różne metody wyświetlania grafiki, wliczając w nie także biblioteki wykorzystujące renderowanie grafiki poprzez te metody. Badania opierały się na przeanalizowaniu liczby FPS podczas wywoływania zdarzeń przez urządzenie wskazujące, takie jak na przykład mysz komputerowa. Z badanych metod najlepiej działającą techniką okazały się metody łączone. WebGl połączony z SVG osiągnął najlepsze wyniki liczby FPS. Można także zauważyć, że wynik liczby FPS metody HTML5 Canvas połączonej z SVG nie wyróżniał się znacząco od najlepszego wyniku badania. Najsłabszą metodą okazała się być metoda renderowania poprzez SVG. W artykule przedstawione są także wyniki badań nad liczbą linii kodu potrzebnych do użycia danych metod. Najmniej linii kodu wymaga tutaj metoda SVG. Można więc uznać, że przy aplikacjach statycznych, gdzie nie występują animacje i liczy się jak najmniejsza liczba linii kodu, bardzo dobrze sprawdzi się metoda SVG. Natomiast dla aplikacji wymagających jak najlepszego działania, najlepiej będzie wybrać WebGl lub HTML5 Canvas oraz metody łączone.

Artykuł [4] przedstawia podobne badania. Autorzy zbadali liczbę FPS dla animacji wykresu drzewa poprzez interakcje przesuwania i powiększania. W badaniach została przedstawiona także liczba węzłów z elementami, gdzie jeden taki węzeł miał około dwudziestu elementów graficznych. Z przeprowadzonych badań wynika, że dla dwustu węzłów wszystkie badane metody nie różniły się znacząco w liczbie FPS. Różnicę można było już zauważyć od czterystu węzłów gdzie zaczynał się spadek w liczbie FPS dla metod SVG oraz HTML5 Canvas. Metody te osiągały niemal identyczne wyniki od około ośmiuset węzłów. Natomiast metoda WebGl osiągała o wiele lepsze wyniki, a metoda ta bez elementów tekstowych osiągała około sześćdziesięciu FPS dla każdej badanej liczby węzłów. Autorzy proponują także zastosowanie metod łączonych, ponieważ taka technika może znacząco przyspieszyć działanie wizualizacji.

Badane metody zostały także przedstawione w pracy [5]. Autor przeprowadził badanie trzech metod takich jak SVG, HTML5 Canvas oraz WebGl. Tak jak w innych publikacjach, tak i tutaj WebGl osiągnął najlepsze wyniki, zaraz po nim znalazł się HTML5 Canvas. Natomiast SVG osiągnął najsłabszy wynik liczby FPS. Warto tutaj także wspomnieć o tym, że takie wyniki były osiągnięte bez żadnych dodatkowych filtrów. Gdy filtr rozmycia gaussowskiego został uaktywniony można było zaobserwować lekką przewagę metody HTML5 Canvas w liczbie FPS. Autor tutaj wspomniał także o tym, że można by było zaimplementować dany filtr wydajniej, co spowoduje uzyskanie innych wyników.

Szkielet do budowy diagramów pod nazwą FluidDiagrams został przebadany przez autorów w artykule [6]. Szkielet ten zdolny jest do renderowania wizualizacji poprzez metody HTML5 Canvas oraz WebGl. Metoda SVG została przebadana przez bibliotekę D3.js. Z przedstawionych wniosków można stwierdzić, że najlepsze wyniki osiągnęła metoda WebGl. W badaniach metoda SVG osiągnęła lepsze wyniki niż HTML5 Canvas. Jest to całkiem zaskakujący wynik, gdyż w innych źródłach badających te metody to HTML5 Canvas osiągał lepsze wyniki. Może to być spowodowane użyciem innej biblioteki do renderowania elementów SVG.

Czas renderowania elementów został także przebadany przez autorów w artykule [7]. Jednakże zostały tutaj przebadane tylko metody SVG oraz HTML5 Canvas. Okazało się, że pomiędzy tymi dwoma metodami najlepsze wyniki osiągnęła metoda SVG. Najgorszą natomiast była metoda HTML5 Canvas.

Podsumowując informacje ze znalezionych artykułów, można stwierdzić, że najlepszą technologią do animowania grafiki jest WebGl, gdyż osiąga ona największą liczbę FPS. Kolejnymi po niej są HTML5 Canvas wraz z SVG. Natomiast w przypadku czasu renderowania najlepiej wypadła metoda SVG. Warto tutaj dodać, że wszystkie te technologie mają swoje własne zastosowanie do różnych problemów. Można tutaj wyróżnić duże aplikacje opierające się na grafice, których zapotrzebowanie na płynne działanie spełniać będzie prawdopodobnie WebGl. Z drugiej strony są także proste grafiki na stronach typu logo, ikony, proste animacje, do generowania których w zupełności wystarczy SVG. Kolejnym punktem mogą być złożone animacje, czy też interaktywne rysowanie, które zapewni HTML5 Canvas. Tak więc każda technologia ma swoje wady, jak i zalety, dzięki temu każda może zostać wykorzystana w zależności od zapotrzebowania w budowanej aplikacji. Warto tutaj także wspomnieć o tym, że czas potrzebny na początkowe wyrenderowanie elementów nie został przebadany dla wszystkich technologii. Niestety w znalezionych artykułach nie został przebadany czas renderowania elementów dla metody WebGl.

4. Metody badania

Celem badań jest porównanie wydajności metod do wyświetlania grafiki w przeglądarkach internetowych i przypisanie im odpowiednich funkcji. Jest to bardzo ważne zagadnienie, gdyż każda z metod może mieć swoje zastosowanie w różnych przypadkach. Dla

przykładu jedna może być wydajna, lecz nauczenie się jej może zająć wiele godzin, lub też kod potrzebny do jej zastosowania jest rozległy. Z kolei inna może być niewydajna, ale prosta w przyswojeniu i liczba kodu potrzebna do jej zastosowania jest mała. Dodatkowym czynnikiem może być czas potrzebny na początkowe wyrenderowanie strony, przy stronach statycznych. Dzięki tym badaniom programista będzie mógł bez problemu wybrać metodę idealnie dopasowującą się do jego celu przy tworzeniu swojej aplikacji webowej.

Badania zostały wykonane w oparciu o opracowane scenariusze badawcze. Odbyło się to za pomocą dedykowanej aplikacji, która umożliwia zbadanie różnych metod przy zastosowaniu różnych opcji w niej udostępnionych. Ważnym punktem jest tutaj umożliwienie użytkownikowi wyboru zaawansowania badań w stopniu podstawowym oraz złożonym. Oznacza to, że rysowane obrazy, jak i animacje są proste lub też złożone. Dla przykładu prostą figurą można nazwać kwadrat, koło lub też trójkąt, a złożoną mogą być grupy figur połączone w jeden obiekt wraz z nałożonym tłem. Podobnie dla animacji, prostą może być zwykłe obracanie figury, a złożoną animacją przesuwania wraz z zaimplementowanym systemem do wykrywania zderzeń obiektów z ramą kontenera. Dodatkowym czynnikiem badań jest także liczba elementów. Liczba ta została dobrana tak aby ukazać możliwości renderowania małej, średniej oraz dużej liczby obiektów. Liczba elementów wybranych do badań jest następująca: 100, 500, 1000, 1500 oraz 2000.

Do badań zostały wybrane trzy możliwe metody do wyświetlania grafiki w przeglądarkach internetowych. Każda z nich ma swoje zastosowanie w budowaniu aplikacji oraz swoje mocne i słabe strony. Takimi metodami są: SVG, HTML5 Canvas oraz WebGl.

Środowisko testowe jest niezwykle ważną częścią każdego badania. Ważne jest, aby było ono jednakowe, w celu uzyskania spójnych wyników badań. Na rzecz badań zostało wybrane jedno środowisko testowe, które posiada następującą specyfikację:

- System operacyjny: Windows 10.
- Monitor: Philips 243V7QDSB/00.
- GPU: AMD Radeon 5700xt Nitro +.
- CPU: AMD Ryzen 5 3600 3,6 GHZ.
- RAM: 16 GB DDR4 HyperX 3200MHz.
- Dysk: Adata XPG SPECTRIX S40G 512GB PCIe Gen3x4 M.2.
- Przeglądarka: Brave w wersji 1.25.68.
- Chromium: 91.0.4472.77.

Plan badań opisuje sposób, w jaki przeprowadzone zostaną badania i jakie przypadki zostały wyspecyfikowane. Dane badania zostały przeprowadzone za pomocą dedykowanej aplikacji umożliwiającej użytkownikom badanie metod wyświetlania grafiki w przeglądarkach internetowych. Użytkownicy mają do wyboru opcje umożliwiające specyfikacje badania w zależności od potrzeb. Ważną kwestią są także przypadki testowe. Pomagają skonstruować plan badań oraz koncentrują je na wyspecyfikowanych celach. Na rzecz badań zostały przygotowane następujące scenariusze:

- Badanie czasu renderowania 100, 500, 1000, 1500, 2000 prostych elementów.
- Badanie czasu renderowania 100, 500, 1000, 1500, 2000 złożonych elementów.
- Badanie liczby FPS 100, 500, 1000, 1500, 2000 prostych elementów dla prostej animacji.
- Badanie liczby FPS 100, 500, 1000, 1500, 2000 złożonych elementów dla prostej animacji.
- Badanie liczby FPS 100, 500, 1000, 1500, 2000 prostych elementów dla złożonej animacji.
- Badanie liczby FPS 100, 500, 1000, 1500, 2000 złożonych elementów dla złożonej animacji.

Pierwsze dwa scenariusze badań opierają się na zbadaniu czasu renderowania elementów i polegają na wyborze typu elementu jako prosty, bądź złożony. Elementem prostym jest kwadrat ze zwykłym szarym wypełnieniem. Natomiast elementem złożonym jest koło i trójkąty zgrupowane razem w jeden element i wypełnione gradientem. Dodatkowo wybierana jest liczba elementów do wyrenderowania przez aplikację. Po wyborze danych opcji renderowanie rozpoczyna się poprzez przyciśnięcie odpowiedniego przycisku. Po wyrenderowaniu elementów wyświetlany jest czas renderowania w milisekundach (ms). Każdy przypadek badań wykonany zgodnie z tymi scenariuszami został wykonany po pięć razy. Wyniki w tabelach i wykresach zostaną ukazane jako średnia liczba z tychże pięciu prób.

Kolejne cztery scenariusze dotyczą badania liczby FPS dla animacji. Wybierany jest typ obiektu jako prosty, bądź złożony oraz typ animacji, także jako prosta, bądź złożona animacja. Wybierana jest także liczba elementów do wyrenderowania dla animacji. Następnie animacja jest rozpoczynana poprzez przyciśnięcie odpowiedniego przycisku. Po jednej minucie od rozpoczęcia animacji jest ona przerywana i liczba średnich FPS jest wyświetlana na interfejsie aplikacji.

5. Analiza porównawcza

Analiza porównawcza omawia wyniki uzyskane z przeprowadzonych badań. Zostaną przedstawione wyniki czasu renderowania oraz liczby FPS przy animacji.

5.1. Badania czasu renderowania

Badania polegały na sprawdzeniu inicjalnego czasu renderowania wybranej liczby elementów danego typu. Pierwsze badanie opierało się na zbadaniu tego czasu dla elementów typu prostego. Jak można zauważyć w tabeli 1, najlepszą metodą, osiągającą najmniejszy inicjalny czas renderowania jest HTML5 Canvas. Zaraz po niej znajduje się kolejna metoda pod nazwą SVG, która osiąga już wyniki pomiędzy metodami HTML5 Canvas oraz WebGl. Najgorzej wypada tutaj metoda WebGl. Takie wyniki występują dla każdej liczby elementów.

Tabela 1: Średni czas (milisekunda/ms) potrzebny na wyrenderowanie danej liczby prostych elementów

SVG (ms)	HTML5 Canvas (ms)	WebGl (ms)	Liczba elementów
14±1	10±1	30±4	100
67±2	56±8	105±14	500
138±16	120±12	210±36	1000
228±16	191±16	329±42	1500
336±36	281±23	425±53	2000

Kolejne badanie oparte na czasie renderowania dotyczyło elementów złożonych. Okazuje się, że wnioski są analogiczne jak dla elementów prostych. Jedyną różnicą jest zwiększony czas renderowania nawet do prawie trzynastu sekund dla metody WebGl i dwóch tysięcy elementów, co można zauważyć na tabeli 2.

Tabela 2: Średni czas (milisekunda/ms) potrzebny na wyrenderowanie danej liczby złożonych elementów

SVG (ms)	HTML5 Canvas (ms)	WebGl (ms)	Liczba elementów
158±19	129±24	286±22	100
1067±69	869±74	1849±38	500
2791±184	2312±81	4706±196	1000
5508±175	4527±125	8318±320	1500
9520±336	7304±233	12774±696	2000

5.2. Badania liczby FPS przy animacji

Następne cztery badania opierają się na zbadaniu liczby FPS dla dwóch typów animacji oraz dwóch typów elementów. Pierwszym badaniem jest określenie liczby FPS dla prostej animacji i prostych elementów.

Tabela 3: Liczba średnich FPS dla 60 sekundowej prostej animacji dla prostych elementów

SVG (fps)	HTML5 Canvas (fps)	WebGl (fps)	Liczba elementów
60±2	60±1	60±2	100
56±5	60±2	60±1	500
28±3	60±4	60±1	1000
19±2	46±4	60±1	1500
14±1	34±3	60±2	2000

Jak można zauważyć na tabeli 3, najlepszą metodą okazała się metoda WebGl osiągająca około 60 FPS dla każdej liczby elementów. Trochę słabiej wypadła metoda HTML5 Canvas, w której dla dwóch tysięcy elementów osiągnęła około 34 FPS, co jest całkiem dobrym wynikiem. Natomiast najgorzej wypadła metoda SVG z wynikiem 14 FPS dla dwóch tysięcy elementów.

Wyniki dla animacji złożonej i prostych elementów są analogiczne jak w animacji prostej z prostymi elementami. Jedyną różnicą jest lekko zwiększona liczba FPS dla tysiąca pięciuset oraz dwóch tysięcy elementów przy metodzie HTML5 Canvas. Natomiast nie zmienia to ostatecznych wniosków, w których metoda WebGl osiąga najlepsze wyniki. Można to zobaczyć na tabeli 4.

Tabela 4: Liczba średnich FPS dla 60 sekundowej złożonej animacji dla prostych elementów

SVG (fps)	HTML5 Canvas (fps)	WebGl (fps)	Liczba elementów
60±2	60±1	60±7	100
55±5	60±2	60±2	500
28±2	60±4	60±2	1000
19±2	57±5	60±1	1500
14±1	41±3	60±2	2000

Dla prostej animacji i złożonych elementów najlepiej wypadła metoda WebGl osiągając około 12 FPS dla dwóch tysięcy elementów. Nie jest to duża liczba, jednakże w porównaniu do innych metod można uznać, że wynik ten jest bardzo dobry. Już od tysiąca elementów nie widać znaczącej różnicy pomiędzy metodami HTML5 Canvas i SVG. Osiągają one bardzo słabe wyniki. Animacje z taką liczbą FPS wahającą się pomiędzy 1 a 7 FPS, nie sprawdzą się w żadnej aplikacji i spowodują niezamierzone negatywne reakcje użytkowników. Jedynie dla stu elementów można zobaczyć, że HTML5 Canvas osiąga o 13 FPS więcej niż SVG. Widać to na tabeli 5.

Tabela 5: Liczba średnich FPS dla 60 sekundowej prostej animacji dla złożonych elementów

SVG (fps)	HTML5 Canvas (fps)	WebGl (fps)	Liczba elementów
20±2	33±2	60±1	100
4±1	7	49±3	500
2	3	25±1	1000
1	2	16±1	1500
1	2	12±1	2000

Dla złożonej animacji i złożonych elementów wnioski są analogiczne jak w poprzednich badaniach. WebGl wypada tutaj najlepiej, a pozostałe dwie metody nie radzą sobie już od pięciuset elementów. HTML5 Canvas wypada trochę lepiej niż SVG do pięciuset elementów. Niestety dla większej liczby obiektów,

metody te osiągają od 1 do 3 FPS, co tak jak w przypadku wcześniejszego badania jest bardzo słabym wynikiem. Wyniki można zauważyć na tabeli 6.

Tabela 6: Liczba średnich FPS dla 60 sekundowej złożonej animacji dla złożonych elementów

SVG (fps)	HTML5 Canvas (fps)	WebGl (fps)	Liczba elementów
21±2	32±3	60±2	100
4±1	6	49±3	500
2	3	25±1	1000
1	2	14±1	1500
1	2	10±1	2000

6. Wnioski

Po przeanalizowaniu wyników ze wszystkich badanych przypadków można przedstawić następujące wnioski. W przypadku inicjalnego renderowania elementów najgorsze wyniki osiąga metoda WebGl. Jest to prawdopodobnie spowodowane potrzebą przetworzenia większej liczby czystego kodu do renderowania elementów poprzez tę metodę. Czysty kod oznacza kod, który znajduje się w metodach biblioteki Two.js wywoływanych przy renderowaniu elementów. Taki kod różni się w zależności od wybranego silnika renderowania. Kolejną metodą osiągającą już znacznie lepsze wyniki jest metoda SVG, a najlepiej wypada HTML5 Canvas. Jest to spowodowane prawdopodobnie tym, że każdy element graficzny stworzony poprzez metodę SVG musi posiadać odpowiedni znacznik w drzewie DOM (ang. Document Object Model). HTML5 Canvas wypadł najlepiej ze wszystkich metod, co jest prawdopodobnie spowodowane tym, że w odróżnieniu od metody SVG, grafika renderowana jest dynamicznie w elemencie języka HTML o znaczniku canvas. Między metodą SVG, a WebGl różnice wywodzą się prawdopodobnie z tego, że SVG wymaga o wiele mniej początkowego przetworzenia kodu, gdyż polega tylko na utworzeniu nowego elementu w drzewie DOM. WebGl natomiast potrzebuje o wiele więcej czasu na przetworzenie początkowego kodu, gdyż zapewnia o wiele więcej możliwości. Końcowe wyniki dla obydwu rodzajów obiektów nie różnią się końcowymi wnioskami. Jedyną różnicą jest znacznie zwiększony czas renderowania dla elementów złożonych.

W badanych scenariuszach związanych z animacją, metodą znacznie wyróżniającą się liczbą FPS jest metoda WebGl. W każdym badaniu metoda ta osiągała o wiele większą liczbę FPS niż metoda SVG oraz HTML5 Canvas. Dla elementów prostych WebGl osiągała niemalże stałą liczbę 60 FPS dla obydwu typów animacji. Jednakże dla obiektów złożonych liczba FPS znacząco spadała przy zwiększeniu liczby elementów. Dla maksymalnej liczby 2000 elementów liczba FPS nie przekraczała średnio nawet 12 FPS. Tak duża liczba FPS w różnych badaniach dla tej metody jest prawdopodobnie spowodowana tym, że WebGl może używać zasobów komputera, na którym uruchamiana jest dana przeglądarka. Dzięki temu WebGl może bez problemu osiągać stałe 60 FPS, gdzie inne metody nie mają takiej możliwości i znacznie odstają w liczbie FPS. Metoda SVG osiąga tutaj najgorsze wyniki, co może być także spowodowane osobnymi elementami ze znacznikami svg, w aplikacji internetowej. Natomiast metoda HTML5 Canvas znajduje się pomiędzy pozostałymi metodami i osiąga zadowalające wyniki. Można to zauważyć na animacji dla elementów prostych, w której metoda ta nie odstaje znacznie od metody WebGl w zakresie liczby FPS. Znaczącą różnicę można było zobaczyć już dla 1500 elementów, natomiast metoda SVG traciła FPS już dla 500 elementów. Dla elementów złożonych można zobaczyć ogromne spadki liczby FPS przy zwiększaniu liczby elementów. Już dla 500 elementów metody SVG oraz HTML5 Canvas osiągały liczbę od 4 do 7 FPS, co jest bardzo słabym wynikiem.

Każda z metod może mieć swoje własne zastosowanie. Gdy deweloper pracuje nad prostą aplikacją posiadającą wyłącznie grafiki takie jak logo, ikony czy też jakieś inne proste grafiki i zależy mu na responsywności strony, to znakomicie sprawdzi się tutaj metoda SVG. Metoda ta nie wymaga od użytkownika poznania dodatkowego API, czy też zastosowania różnych bibliotek. Wystarczy zwykły znacznik języka HTML, co bardzo skraca liczbę linii kodu, jak i czas tworzenia aplikacji. Dodatkowym atutem tej metody jest także grafika wektorowa. Dla aplikacji graficznie zaawansowanych, na przykład giełdowych, gdzie bardzo ważna jest płynność działania wykresów, doskonale sprawdzi się metoda WebGl, która osiągała najlepsze wyniki liczby FPS z pozostałych. Natomiast dla aplikacji interaktywnych, czy też nawet prostych aplikacji graficznych, dobrze wpasuje się metoda HTML5 Canvas. Odpowiadając na zdefiniowane hipotezy badawcze możemy stwierdzić, że:

- Metoda WebGl osiąga najlepsze wyniki czasu renderowania danej liczby elementów - dana hipoteza nie może zostać potwierdzona, najlepsze wyniki osiągnęła metoda HTML5 Canvas.
- Metoda WebGl osiąga najlepsze wyniki liczby FPS przy animowaniu elementów - dana hipoteza została potwierdzona, najlepsze wyniki liczby FPS osiąga metoda WebGl.

Literatura

[1] Statystyki użycia różnych formatów zdjęć na stronach internetowych, https://w3techs.com/technologies/overview/image_format, [25.03.2021].

[2] Statystyki użycia SVG na stronach internetowych, https://w3techs.com/technologies/details/im-svg, [25.03.2021].

[3] D.E. Kee, L. Salowitz, R. Chang, Comparing Interactive Web-Based Visualization Rendering Techniques, VisWeek 2012 Poster Program, Washington, 2012.

[4] T. Horak, U. Kister, R. Dachselt, Comparing Rendering Performance of Common Web Technologies for Large Graphs, Poster Program of the 2018 IEEE VIS Conference, Berlin, 2018.

[5] A. Lindberg, Performance Evaluation of JavaScript Rendering Frameworks, Master thesis, Linköping University, Linköping, 2020, http://www.diva-portal.org/smash/get/diva2:1411632/FULLTEXT01.pdf, [25.03.2021].

[6] K. Andrews, B. Wright, FluidDiagrams: Web-Based Information Visualisation using JavaScript and WebGl, Eurographics Conference on Visualization EuroVis (2014) 43-47.

[7] D.W. Johnson, T.J. Jankun-Kelly, A Scalability Study of Web-Native Information Visualization, Proceedings of the Graphics Interface 2008 Conference (2008) 163–168.