ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

Дата звіту 6/3/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗздм-23-1_Понікаровська_Н_А_скорочений
Автор Науковий керівник / Експерт
Понікаровська Наталія АндріївнаОлена Олійник
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1

КЦ

25	9978	76054
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	¶	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	12

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://blog.csdn.net/csdn122345/article/details/148242441	24 0.24 %
2	https://www.bomberbot.com/data-visualization/building-a-gantt-like-chart-with-d3-js-a-step-by-step-guide/	22 0.22 %
3	https://blog.csdn.net/csdn122345/article/details/148242441	20 0.20 %
4	http://www.dut.edu.ua/uploads/p_86_96944839.pdf	20 0.20 %
5	https://en.ittrip.xyz/c-sharp/csharp-blazor-d3-dashboards	18 0.18 %

6	https://sinequa.github.io/sba-angular/tipstricks/d3-angular.html	17 0.17 %
7	https://www.genezum.org/library/vizualizaciya-informacii-na-urokah-istorii	14 0.14 %
8	https://en.ittrip.xyz/c-sharp/csharp-blazor-d3-dashboards	12 0.12 %
9	https://www.freecodecamp.org/news/d3js-tutorial-data-visualization-for-beginners/	12 0.12 %
10	https://www.freecodecamp.org/news/d3js-tutorial-data-visualization-for-beginners/	10 0.10 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.10 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Тринцолин_Дудка.pdf [redacted] 11/15/2023 Vasyl Stefanyk Precarpathian National University (VSPNU) (Факультет математики та інформатики)	10 (1) 0.10 %

з Інтернету (2.19 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://www.freecodecamp.org/news/d3js-tutorial-data-visualization-for-beginners/	48 (6) 0.48 %
2	https://blog.csdn.net/csdn122345/article/details/148242441	44 (2) 0.44 %
3	https://en.ittrip.xyz/c-sharp/csharp-blazor-d3-dashboards	30 (2) 0.30 %
4	https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B1%D0%BE%D0%BB%D0%B0	24 (4) 0.24 %
5	https://www.bomberbot.com/data-visualization/building-a-gantt-like-chart-with-d3-js-a-step-by-step-guide/	22 (1) 0.22 %
6	http://www.dut.edu.ua/uploads/p_86_96944839.pdf	20 (1) 0.20 %
7	https://sinequa.github.io/sba-angular/tipstricks/d3-angular.html	17 (1) 0.17 %
8	https://www.genezum.org/library/vizualizaciya-informacii-na-urokah-istorii	14 (1) 0.14 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

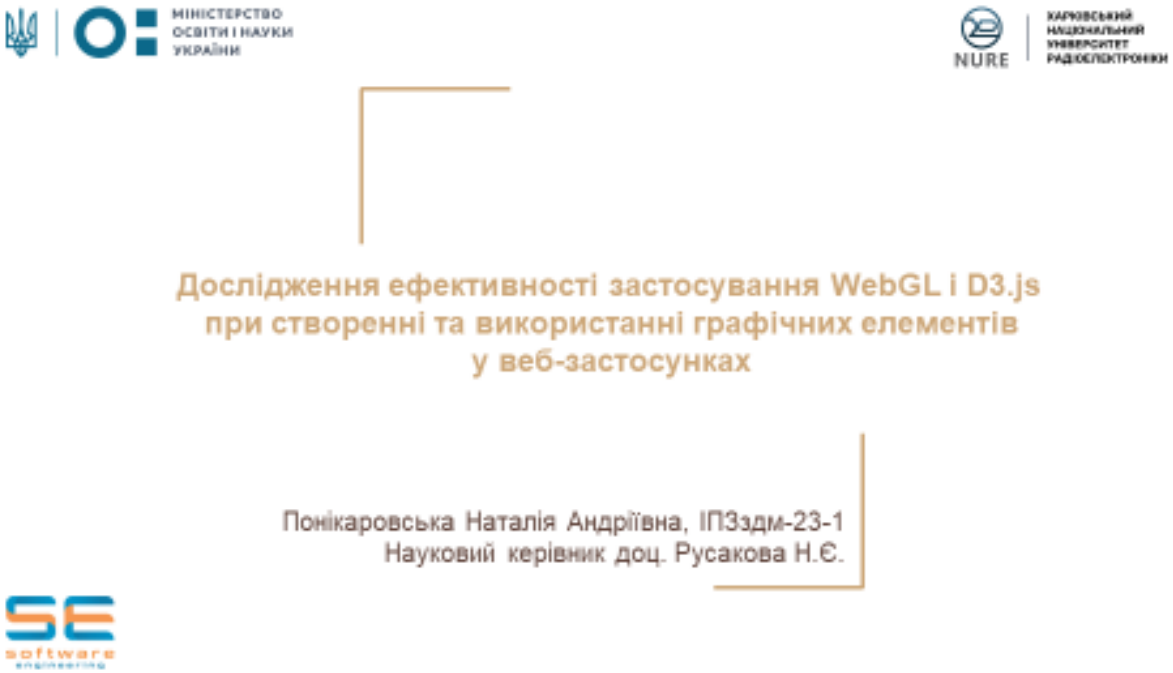
ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

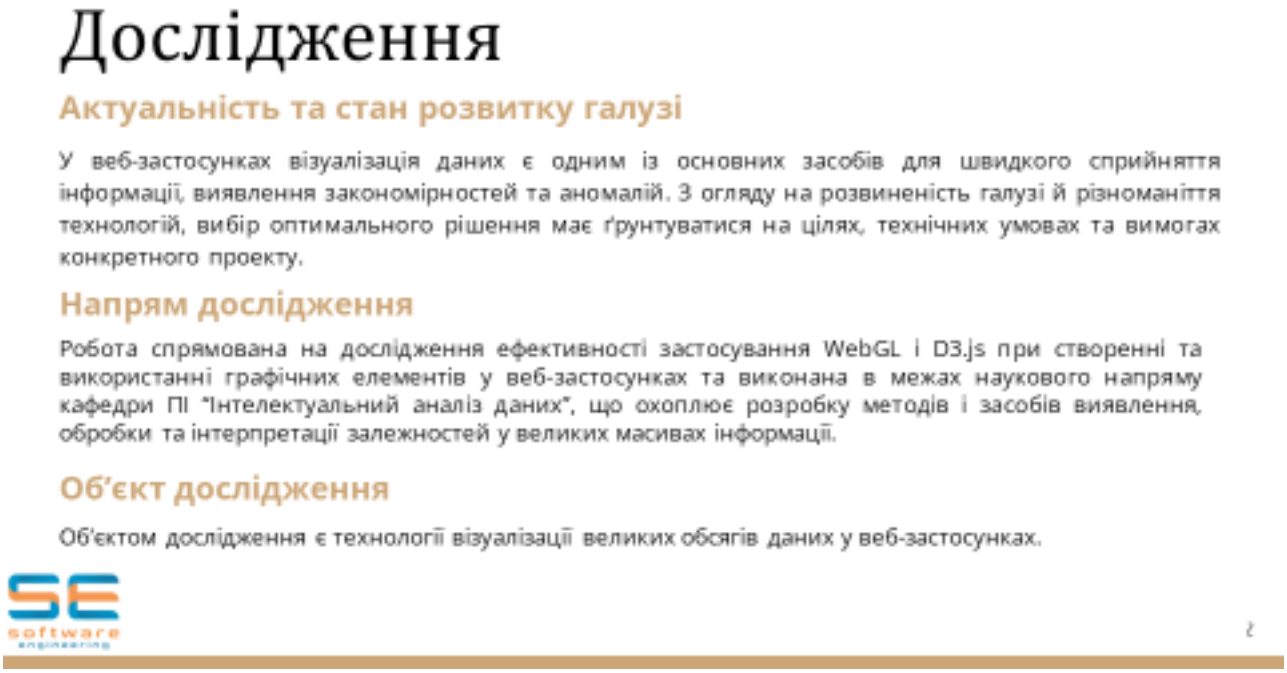
10
ВСТУП

Візуалізація у веб-застосунках є одним із інструментів аналізу великих масивів даних, який дозволяє виявляти закономірності та аномалії, спрощує інтерпретацію результатів. Цю функцію чітко окреслено у визначенні мети візуалізації, запропонованому Майком Бостоком, автором бібліотеки D3.js: "The purpose of visualization is insight, not pictures" [1]. У сфері веб-розробки актуальною є задача вибору технології. Наразі існує велика кількість рішень, що ґрунтуються на різних технологічних підходах, але їхню ефективність складно

ДОДАТОК Б

Слайди презентації





Огляд літератури

Перелік основних джерел та теорій у галузі

Теоретичною основою стали роботи, присвячені декларативному підходу в D3.js (Майк Бостоk, Елайджа Мікс, Маркос Ігліесіас, Адам Рінінсланд), інтерактивності та концепції "візуалізація як історія" (Нан Чжу, Скотт Мюррей), а також імперативному програмуванню у WebGL, концепції низькорівневого управління графікою та питанням безпеки (Кенджі Мацуда, Тоні Парізі, Ной Ґелернтер).

Незважаючи на велику кількість публікацій, що висвітлюють використання окремих технологій або прикладні кейси, вибір засобів візуалізації залишається складним через велику кількість доступних рішень і необхідність врахування численних технічних вимог. Різні потреби проектів і специфіка технологій роблять це завдання багатофакторним. У дослідженні запропоновано формалізований підхід на основі методології MCDA для комплексного оцінювання технологій візуалізації у веб-середовищі.

Постановка задачі

Проблема дослідження

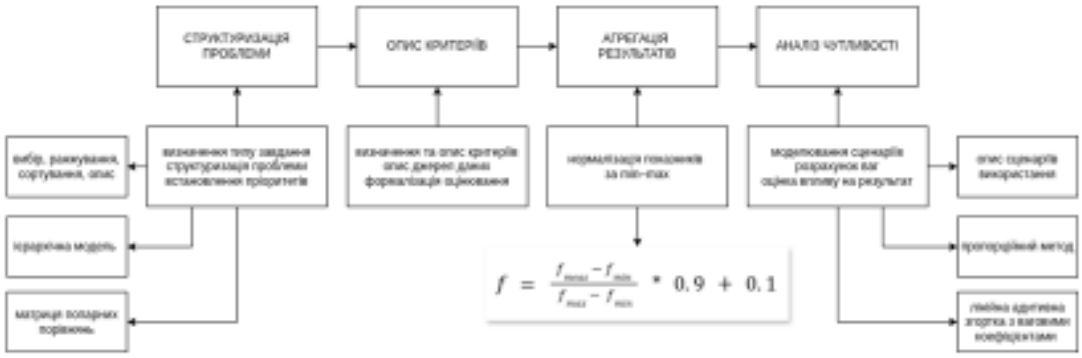
Різноманіття технологій веб-візуалізації, відмінності у їхній архітектурі, продуктивності та рівні складності впровадження ускладнюють вибір інструмента, що відповідатиме вимогам конкретного проекту. Формалізований підхід до оцінювання ефективності таких технологій за узгодженими критеріями може стати ефективним інструментом підтримки рішень у процесі проектування.

Очікувані результати дослідження:

– розроблена формалізована модель багатокритеріального оцінювання ефективності WebGL і D3.js з використанням методології MCDA;
– практичні рекомендації для розробників і керівників проектів щодо вибору технологій візуалізації залежно від вимог, цілей та контексту веб-застосунку.

Методи дослідження

MCDA (Multi-Criteria Decision Analysis, багатокритеріальний аналіз) - це методологія прийняття рішень, що дозволяє системно оцінювати та порівнювати альтернативи за кількома критеріями, які можуть мати різну природу, вагу та вплив на результат.



Інструментарій та технології

Операційна система: Ubuntu (18.04.6 LTS).
Браузер: Google Chrome (127.0.6533.119).
Мова програмування: JavaScript.
Технології: D3.js, WebGL.
Для збору метрик продуктивності: Chrome DevTools
Для збору якісних метрик: GitHub, експертна оцінка, сервіс "Can I Use", статистика з Hostmaster.ua, аналіз ARIA-атрибутів.
Опис тестових наборів даних: координати (x, y) у діапазоні [0,1], збережені у форматі JSON.
Програмне забезпечення: графічні модулі (D3.js і WebGL) для побудови графіка розсіювання.

Схема архітектури розробленої системи

Опис компонентів

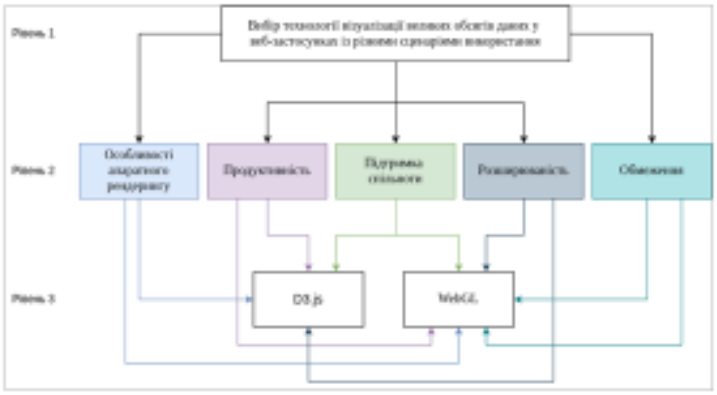
Графічні модулі

Інтерфейс модулів D3.js і WebGL включає заголовок, панель керування сценаріями та область побудови графіка. Реалізовано три сценарії запуску: візуалізація 1000 точок, 10000 точок та поступове додавання 10000 точок з інтервалом у 5 секунд.

У модулі на D3.js візуалізація реалізована через SVG-елементи з використанням декларативного підходу та прив'язки даних до DOM.
У модулі на WebGL для відображення використано графічний конвеєр GPU

Ієрархічна структура

Завдання дослідження класифікується як **вибір**, оскільки основною його метою є визначення оптимальної технології для візуалізації великих обсягів даних у веб-застосунках із різними сценаріями використання.



Тестування метрик продуктивності

Результати показують, що D3.js не задіює графічний процесор (GPU time = 0), тоді як WebGL активує апаратне прискорення, водночас не створюючи значного навантаження на систему, що свідчить про ефективність і придатність WebGL для масштабних візуалізацій навіть у ресурсообмежених середовищах.

WebGL демонструє кращу продуктивність у порівнянні з D3.js. У сценарії з динамічним оновленням точок FPS був на 49% вищим (119 проти 80), а час відгуку — на 80% меншим (179 мс проти 919 мс), ніж у D3.js. Отже, WebGL краще відповідає вимогам високонавантажених і динамічних графічних задач.

Технологія	RT	FPS
Сценарій 1 (статична діаграма, 1000 точок)		
D3.js	165	120.2
WebGL	149	139
Сценарій 2 (статична діаграма, 10 000 точок)		
D3.js	271	115.4
WebGL	111	120.2
Сценарій 3 (динамічна діаграма, + 10 000 точок кожні 5 секунд)		
D3.js	919	80.1
WebGL	179	139.3

	Мін. D3.js	Середнє D3.js	Макс. D3.js	Мін. WebGL	Середнє WebGL	Макс. WebGL
GPU time (мс)	0	0	0	19.0	-	92.0
Total time (мс)	-	-	-	860.0	-	12193.0
GPU share (%)	0	0	0	0.7	1.61	3.46

Тестування якісних метрик

Згідно з GitHub метриками D3.js значно переважає WebGL за популярністю серед розробників: кількість зірок більша у 40 разів, форків - у 30 разів, а підписників - у 34 рази

D3.js реалізує гнучкий конструкторський підхід із використанням готових механізмів бібліотеки, але не дозволяє виходити за межі передбачених методів. Натомість WebGL, як низькорівневе API, забезпечує повну свободу реалізації графіки будь-якої складності.

Попри підтримку в більшості браузерів, WebGL має низку обмежень: блокування через апаратні або безпекові налаштування, а також повну відсутність підтримки доступності, що обмежує його використання
в інклюзивних веб-застосунках.

Технологія	Fork	Watch	Star
D3.js	3609	22900	109000
WebGL	117	670	2700

Технологія	Підтримка в браузерах	Доступність
D3.js	100%	100%
WebGL	89.89%	0%

	D3.js	WebGL
Розширюваність	0.1	1

Агрегація результатів

Для забезпечення порівнюваності показників із різними одиницями виміру та масштабами, усі значення було приведено до єдиного діапазону шляхом мін-макс нормалізації.

На цій основі виконано перший етап аналізу чутливості за допомогою лінійної адитивної згортки з рівнозначними ваговими коефіцієнтами (по 0.2). WebGL отримав вищу оцінку завдяки високій продуктивності та розширюваності, тоді як загальний результат D3.js виявився нижчим, попри його переваги у сфері доступності та підтримки спільноти.

Критерій	D3.js	WebGL
Особливості апаратного рендерингу	0.1	1
Продуктивність	0.63	0.97
Підтримка спільноти	1	0.1
Розширюваність	0.1	1
Обмеження	1	0.1

Технологія	TS
D3.js	0.566
WebGL	0.634

Аналіз чутливості

Другий етап аналізу чутливості передбачав оцінювання технологій у двох сценаріях із контекстно зваженими пріоритетами.

Результати аналізу показують, що зміна пріоритетів критеріїв залежно від контексту використання призводить до різної вагомості показників і, відповідно, до різних результатів оцінювання. У першому сценарії (високонавантажений аналітичний графік) WebGL отримує вищі оцінки за критеріями продуктивності та розширюваності. У другому сценарії (статичні графіки для освітньої платформи) кращі результати демонструє D3.js завдяки більшій доступності, сумісності з браузерами та популярності серед розробників.

Критерій	1 сценарій	2 сценарій
Особливості апаратного рендерингу	0.267	0.067
Продуктивність	0.333	0.267
Підтримка спільноти	0.133	0.200
Розширюваність	0.200	0.133
Обмеження	0.067	0.333

Технологія	1 сценарій	2 сценарій
D3.js	0.353	0.733
WebGL	0.813	0.433

Аналіз отриманих результатів

- WebGL рекомендовано використовувати в проектах, що потребують обробки великих обсягів даних у реальному часі, складної анімації або високої частоти оновлення, оскільки технологія ефективно задіює GPU.
- D3.js рекомендовано для завдань, що потребують гнучкої конфігурації, швидкої інтеграції та високої доступності, зокрема у звітах, освітніх проектах та дашбордах із помірним навантаженням.
- Гібридний підхід рекомендований для змішаних сценаріїв, в яких D3.js доцільно використовувати для створення вісей, тексту та контролерів, WebGL - для реалізації складних графічних компонентів.
- Архітектурні рішення слід приймати з урахуванням підготовки команди, бюджету і планів підтримки. D3.js має нижчий поріг входження, натомість WebGL потребує глибоких знань у графіці.
- Слабкі пристрої та застарілі браузери можуть обмежити роботу WebGL, тому його використання слід перевіряти на сумісність та стабільність.

Публікація результатів

Основні положення роботи було апробовано під час виступів на конференціях "Problems of Infocommunications. Science and Technology (PIC S&T 2024)" та 9th Open International Conference "Electrical, Electronic and Information Sciences" (eStream 2025)


Comparative Analysis of WebGL and D3.js


Research on the efficiency of applying WebGL and D3.js in the creation of graphical elements in web applications

Підсумки

Корисність отриманих результатів

Отримані результати показують, що поставлену мету - дослідити ефективність використання WebGL та D3.js у побудові графічних елементів для веб-застосунків - реалізовано в повному обсязі. Розроблена аналітична модель є дієвим інструментом для обґрунтованого вибору технологій візуалізації з урахуванням проєктних вимог. Її гнучка структура забезпечує можливість адаптації до різних сценаріїв застосування та подальшого розширення відповідно до змін контексту або критеріїв оцінювання.

Подальші дослідження

Подальші дослідження доцільно орієнтувати на підвищення доступності WebGL для типових прикладних завдань. Перспективним напрямом є розробка бібліотек або інструментальних засобів, які поєднують високу продуктивність низькорівневого рендерингу з перевагами декларативного підходу.

ДОДАТОК В

Апробація результатів роботи

Comparative Analysis of WebGL and D3.js

Natalia Rusakova, Nataliia Ponikarovska, Mariya Shirokopetleva
Department of Software Engineering
Kharkiv National University of Radio Electronics
Ukraine, Kharkiv
0000-0001-6991-9582
nataliia.ponikarovska.cpe@nure.ua, marija.shirokopetleva@nure.ua

***Abstract* — This article conducts a comparative analysis of WebGL and D3.js technologies, utilizing Multi-Criteria Decision Analysis (MCDA) to assess their effectiveness in various usage scenarios. WebGL is a JavaScript API that allows for the rendering of interactive 3D and 2D graphics within any compatible web browser without plug-ins. It operates directly with the GPU, enabling high-performance graphics and visual effects. In contrast, D3.js is a JavaScript library that uses web standards such as HTML, SVG, and CSS to render complex, data-driven visualizations. It allows developers to attach data to DOM elements, applying data-driven transformations to the document.**

By examining the primary characteristics of each technology, the paper explores how WebGL and D3.js perform specific tasks in data visualization, focusing on their strengths and weaknesses. The analysis is structured around three main criteria: Performance, Complexity, and Limitations. Each criterion encompasses several indicators. The study of each technology occurs within the context of their approach to solving specific tasks, highlighting key factors to consider when choosing between them. The use of MCDA facilitates an objective evaluation and ensures a balanced decision, considering all necessary aspects of technology selection for specific project needs. This research has formulated a hypothesis regarding the optimal use of these technologies.

Keywords—WebGL, D3.js, GPU acceleration, data visualization, performance, rendering, dynamic data, FPS, large datasets, criteria, MCDA.

I. INTRODUCTION

Analytics is a key component in many fields, including business, scientific research, and the technology industry, and infocommunication systems. This necessitates tools that are capable of processing and visualizing large volumes of data efficiently. Visualization not only facilitates the interpretation of analytical results but also allows for quick and effective insights. In this context, the choice of visualization technology is crucial as the quality of visualization directly impacts the level of understanding and the insights that can be drawn from the data.

D3.js is a powerful tool for creating dynamic visualizations [1, 2, 3], but its performance is limited by the CPU when handling large and continuously growing datasets. WebGL [4, 5, 6], by utilizing the GPU, offers hardware-accelerated rendering and significantly improves processing speed through parallel computing.

The GPU consists of thousands of small cores specialized in parallel processing of simple operations, making it ideal for pixel rendering, image processing, and visualizing large datasets [7]. WebGL uses a shader system for computations, allowing parallel data processing and greatly accelerating graphic rendering.

Unlike the GPU, the CPU is optimized for sequential data processing, which creates limitations for tools like D3.js when data volumes exceed the processor's capabilities. With WebGL, browsers can directly harness the power of the GPU for fast graphic rendering.

Choosing the right technology for data visualization involves more than just understanding the data; it requires a clear definition of criteria that will guide the selection process and a robust methodology to justify the choice.

This article aims to compare two prominent visualization technologies—WebGL and D3.js—using a multi-criteria decision-making approach. By systematically evaluating these technologies against defined criteria, we provide a structured analysis that assists in making an informed decision about which tool best meets specific visualization needs.

II. THEORETICAL FOUNDATIONS

Multi-Criteria Decision Analysis (MCDA) is a critical component in evaluating competing technologies where multiple criteria must be considered. MCDA provides a systematic approach for weighing and integrating diverse measures into a single composite index that can guide decision-making.

The correct choice of technology at the development stage is crucial for the project's short- and long-term success. Each decision impacts system performance, compatibility, scalability, and maintenance, as well as development costs and future updates. Underestimating the importance of technology selection can lead to functional limitations and the need for substantial rework in the future. Thus, MCDA serves not only as a tool for assessing current needs but also as a forecasting mechanism that helps provide a reliable foundation for scaling, support, and further project development at subsequent stages.

This MCDA approach allows us to provide a balanced and nuanced comparison between WebGL and D3.js, making the decision-making process transparent and justifiable "at the design stage" [8]. By applying this methodology, stakeholders can understand the trade-offs involved and make an informed choice based on a comprehensive analysis of the options available.

The research methodology is based on Multi-Criteria Decision Analysis (MCDA) using the steps described by Ishizaka and Nemery [9].

The first step is problem structuring [9, ch. 1.2, p. 3], which involves clearly defining the type of task: selection, sorting, ranking, or description. This helps establish evaluation criteria and alternatives. In our case, it is a choice between technologies, allowing us to identify the critical aspects for assessing the effectiveness of WebGL and D3.js.

The second step is evaluating of criteria and determination of weights [9, ch. 2.2.2, p. 16]. This stage involves defining the significance of each criterion for decision-making. Criteria weights can be established through a pairwise comparison method, where the importance of one criterion is assessed relative to another. This approach ensures that each criterion is considered proportionally to its impact on the final result.

The third step is the aggregation of results [9, ch. 1.3, p. 4; ch. 1.6, p. 8]. This step combines criterion values to create an overall score for each alternative. In our study, this is achieved by calculating a cumulative score for WebGL and D3.js, which helps identify the optimal solution for specific project tasks. Complete aggregation allows low scores in one criterion to be offset by high scores in another.

The final step is sensitivity analysis [9, ch. 2.2.4, p. 19]. This crucial part of the methodology checks the robustness of results by adjusting criterion weights. Sensitivity analysis enables an assessment of how priority changes impact the final choice, ensuring the reliability and validity of the conclusions.

This methodology enables an objective comparison of WebGL and D3.js, providing a systematic approach to selecting a technology based on essential performance, complexity, and limitation metrics. In the analysis of WebGL and D3.js:

- Performance scenarios: testing on static data with 1,000 points, extensive data with 10,000 points, and dynamic updates every 5 seconds, because "One of the important problems ... is the processing of large data sets" [10].
- Performance results: recording rendering time and FPS.
- Complexity and compatibility: evaluating ease of learning and technical limitations for users with varying levels of expertise.

III. IDENTIFICATION AND DEFINITION OF CRITERIA

Our chosen approach to Multi-Criteria Decision Analysis (MCDA) involves defining key metrics [7] such as rendering speed, ease of learning, scalability, and technological limitations. These criteria were selected due to their critical importance in the context of real-world applications of technologies across various usage scenarios, where each aspect can play a decisive role in choosing the optimal solution for a specific project.

A. Performance.

This criterion assesses the technology's ability to process and visualize data quickly and efficiently, especially with large datasets. Indicators are Rendering Time, Frames per second, Complexity, Limitations. Rendering Time is the time required to display the visualization on the screen.

TABLE I. RENDERING TIME INDICATOR

Name	Description	Rating
Excellent	Up to 100 ms	40
Good	100-200 ms	30
Acceptable	200-300 ms	20
Unsatisfactory	More than 300 ms	10

Frames per second (FPS). The rate at which images are refreshed on the screen influences the smoothness of the visualization.

TABLE II. FPS INDICATOR

Name	Description	Rating
Excellent	100 FPS and above	40
Good	65-100 FPS	30
Acceptable	30-65 FPS	20
Unsatisfactory	Less than 30 FPS	10

The methodology employed for evaluating these indicators involves the use of advanced performance profiling tools, such as Chrome DevTools, to measure rendering times and frame rates precisely during controlled testing scenarios.

B. Complexity.

This criterion evaluates the ease of learning and using the technology, including the required level of technical knowledge.

Indicators are Technical Complexity, Learning Time, Limitations.

Measures the depth of knowledge required for effective use of the technology.

TABLE III. TECHNICAL COMPLEXITY INDICATOR

Name	Description	Rating
Low	Basic understanding of common functionalities	20
Moderate	Good understanding of both common and advanced functionalities	30
High	Deep knowledge of complex functionalities and customization	50

Learning time assesses the amount of time needed for a new user to master the basics of the technology [11].

TABLE IV. LEARNING TIME INDICATOR

Name	Description	Rating
Short	Less than a week to become proficient	50
Moderate	One to three weeks to become proficient	30
Long	More than three weeks to become proficient	20

The assessment of each indicator relies on a thorough analysis of available documentation, discussions within technical forums, and a variety of resources. This comprehensive approach enables an accurate evaluation of

the overall complexity and the requisite skill level necessary to effectively utilize the technology, particularly within the constraints of project timelines.

C. Limitations.

This criterion involves analyzing the limitations and potential issues that users may encounter while using the technology. Indicators are Compatibility.and Scalability.

Compatibility is the degree of support across various devices and platforms.

TABLE V. COMPATIBILITY INDICATOR

Name	Description	Rating
High	Supported on all major platforms and devices without issues	50
Moderate	Supported on most major platforms with some minor issues	30
Low	Limited support, significant issues on multiple platforms	20

Scalability is the technology's ability to handle increasing amount of data efficiently.

TABLE VI. SCALABILITY INDICATOR

Name	Description	Rating
High	Can handle significant increases in data volumes without performance degradation	50
Moderate	Can handle moderate increases in data volumes but may show signs of strain under heavy loads	30
Low	Struggles with any significant increase in data volume, leading to performance degradation	20

Indicators are assessed through a meticulous review of technical documentation. This process is designed to ascertain both the compatibility and scalability of the technology. Furthermore, this criterion will take into account the results of the performance evaluation to ensure a comprehensive analysis.

IV. WEIGHTING AND SCORING

A. Performance.

The measurements were conducted using Chrome DevTools - a tool for analyzing web page performance integrated into the Google Chrome browser. It was used to record rendering times and assess frame rates (FPS) during the display and interaction with the charts.

All performance measurements were conducted under controlled conditions using Google Chrome (version 127.0.6533.119), with consistent hardware and browser settings to ensure reliable results. Multiple trials were performed and several scenarios were executed [12] to measure the Rendering Time and FPS, with the results presented the representing averages of these trials. The measurements were carried out on a test dataset to maintain controlled experimental conditions and assess system performance under predefined parameters. To achieve this goal, two charts were created: one using D3.js and the other using WebGL.

Scenario 1: Rendering Static Data (1,000 points).

In this scenario, the performance of rendering static data in WebGL and D3.js was tested. The task involves constructing a chart with 1,000 points and measuring the rendering time and FPS for each technology.

TABLE VII. PERFORMANCE TESTING. RESULTS OF SCENARIO 1

technology	RT	threshold	exceeding RT	FPS	threshold	exceeding FPS
D3.js	165	good	no	120.2	excellent	no
WebGL	149	good	no	119	excellent	no

The results indicate that for small static datasets, both technologies exhibit good performance with WebGL having a slight advantage in rendering speed (149 ms versus 165 ms in D3.js).

Scenario 2: Rendering Large Datasets (10,000 points).

This scenario aims to compare the performance of rendering a larger number of points (10,000) in WebGL and D3.js. The rendering time and FPS for each technology are measured.

TABLE VIII. PERFORMANCE TESTING. RESULTS OF SCENARIO 2

technology	RT	threshold	exceeding RT	FPS	threshold	exceeding FPS
D3.js	271	acceptable	no	115.4	excellent	no
WebGL	111	Good	no	120.2	excellent	no

When working with large datasets, WebGL outperforms D3.js. The rendering time for WebGL is only 111 ms, while D3.js requires 271 ms. Additionally, the FPS in D3.js drops to 115, whereas WebGL maintains a steady FPS at almost the same level (119/120).

Scenario 3: Dynamically Updating Data (10,000 points).

This scenario analyses the performance of rendering dynamic data that updates every 5 seconds three times (adding 10,000 new points to the chart each time). The rendering time for new points and the FPS during updates are evaluated.

TABLE IX. PERFORMANCE TESTING. RESULTS OF SCENARIO 3

technology	RT	threshold	exceeding RT	FPS	threshold	exceeding FPS
D3.js	919	unsatisfactory	no	80.5 80.5 79.3	good	no
WebGL	179	Good	no	119 119 120	excellent	no

WebGL is significantly more efficient when dealing with dynamically updating data. The update time in WebGL [10] is shorter than in D3.js (179 ms versus 919 ms). Additionally, WebGL maintains a stable FPS even as the number of points increases, while D3.js shows a gradual degradation in performance.

B. Complexity.

D3.js [11, 12] offers a higher-level approach to visualization with a manageable learning curve, though it requires knowledge of JavaScript and data manipulation, while its strong community and well-structured documentation help accelerate learning. In contrast, WebGL [13, 4] is a low-level API that demands expertise in computer graphics, including shaders and matrix math, making it more complex to learn, with steep learning requirements despite thorough documentation.

TABLE X. COMPLEXITY TESTING

technology	Technical Complexity	Learning Time	Technical Complexity Rating	Learning Time Rating
D3.js	moderate	moderate	good understanding of both common and advanced functionalities	one to three weeks to become proficient
WebGL	high	long	in-depth knowledge of complex functionalities and customization	more than three weeks to become proficient

C. Limitations.

D3.js [8, 11], compatible with modern browsers using web standards like SVG, performs well with small to moderate datasets but struggles with scalability due to DOM limitations. WebGL [12, 3], while dependent on hardware capabilities, excels in scalability by leveraging GPU acceleration for handling large datasets and complex visualizations.

TABLE XI. LIMITATIONS TESTING

technology	Compatibility	Scalability	Compatibility Rating	Scalability Rating
D3.js	high	moderate	supported on all major platforms without significant issues	handles moderate data volumes but struggles with very large datasets
WebGL	moderate	high	supported on most platforms but may have limitations on older hardware	effectively handles large increases in data volume with hardware acceleration

D. Aggregation

The data from various criteria is consolidated to form a comprehensive evaluation matrix. Aggregation aims to synthesize individual scores into an overall assessment that provides clear guidance on selecting the most suitable technology for specific project needs.

TABLE XII. RESULTS AGGREGATION

Technology	Performance	Complexity	Limitations
D3.js	53.2	60	80
WebGL	70	70	80

If there are multiple indicators (as is the case with the performance criterion), the results in the matrix are aggregated using the arithmetic mean.

V. ANALYSIS AND DECISION

Further work should be carried out according to the formula provided, which represents a weighted sum calculation used to evaluate the overall performance of the technologies (1):

$$Total\ Score = (P \times Wp) + (C \times Wc) + (L \times Wl) \quad (1)$$

Where:

P = Performance score

Wp = Weight of the performance score

C = Complexity score

Wc = Weight of the complexity score

L = Limitations score

Wl = Weight of the limitations score

Examples of use: an infocommunication systems project involves creating multiple small graphs that do not require interactive updates. Given that the developers are not familiar with either D3.js or WebGL.

TABLE XIII. WEIGHTING FACTORS

Performance	Complexity	Limitations
20	40	40

D3.js: $(53.2 \times 0.20) + (60 \times 0.40) + (80 \times 0.40) = 66.64$

WebGL: $(70 \times 0.20) + (70 \times 0.40) + (80 \times 0.40) = 74$

D3.js is the better choice due to its lower complexity and superior capabilities for static graphs, which are crucial for developers new to these technologies.

VI. DISCUSSION AND RECOMMENDATIONS

Based on the results of this study, key points for discussion are outlined below.

WebGL is adept at managing vast data sets, thanks to the GPU's hardware acceleration capabilities. This makes WebGL an ideal choice for complex 3D scenes and dynamic visualizations with large numbers of data points. Even when processing thousands of elements, WebGL can maintain high performance, ensuring smooth interactivity.

Since D3.js relies on the CPU and operates through the DOM, its performance significantly decreases when handling large datasets. Rendering a large number of elements can result in a noticeable drop in FPS and overall web application performance. D3.js is more suitable for small to medium-sized datasets.

WebGL is a low-level API that requires a significant understanding of graphic processes, shader programming, memory management, and buffer usage. Creating even simple scenes requires more code and technical expertise.

This makes the entry barrier to WebGL quite high, especially for beginners in graphics programming.

D3.js has a lower entry barrier due to its higher level of abstraction. The library provides convenient methods for manipulating DOM elements and creating visualizations. Developers can quickly create interactive charts without deep knowledge of graphics programming, making D3.js more accessible for beginners.

The main limitation of WebGL lies in its complexity of learning. Additionally, not all browsers and devices may support WebGL equally well, especially lower-powered devices. Incorrect optimization of computations can also lead to memory overload or performance issues.

D3.js is limited by its use of the CPU and rendering through the DOM, which creates performance issues when dealing with large datasets. Additionally, D3.js does not support 3D graphics rendering and is limited in interactive capabilities compared to WebGL.

Despite its capabilities, WebGL presents challenges in cross-platform accessibility and requires extensive optimization to avoid memory bottlenecks. Conversely, D3.js is constrained by its reliance on CPU-bound DOM manipulations, limiting its performance on modern multi-core processors.

The experimental data obtained in this study allow for the formulation of recommendations regarding the application areas of each technology.

WebGL's capabilities make it an ideal choice for the gaming industry, especially for creating 3D scenes and real-time visualizations. It is also highly recommended for scientific research that requires visualizing large datasets and for projects that demand high performance and interactivity, such as in high-performance scientific visualizations, WebXR, and gaming applications that require real-time 3D graphics. Due to its GPU acceleration, WebGL demonstrates significantly better performance in rendering large datasets, making it particularly effective for dynamic and complex visualizations.

On the other hand, D3.js is recommended for creating analytical visualizations, charts, graphs, and interactive dashboards in web applications. It is well-suited for projects in data journalism, for infocommunication systems, business analytics, finance, and education, where the focus is on visualizing two-dimensional data in a user-friendly format. While D3.js provides a higher level of abstraction that is beneficial for simpler tasks, it should be noted that its performance may be limited when dealing with large amounts of data.

VII. CONCLUSIONS AND FUTURE WORK

This article presents a comprehensive comparison of WebGL and D3.js technologies for data visualization tasks using a Multi-Criteria Decision Analysis (MCDA) methodology. The study determined that WebGL, through hardware acceleration and GPU utilization, has significant advantages in handling large dynamic datasets, maintaining high performance even with extensive data volumes. Conversely, D3.js proved more effective for simpler tasks where high performance is not critica, but has a lower entry barrier.

The novelty of this research lies in applying a structured MCDA approach to objectively evaluate and select visualization technologies based on specific project requirements, enabling developers to make more informed decisions. The findings of this study provide guidance when choosing visualization tools for tasks of varying complexity.

Future research directions may include a more detailed study of WebGL integration with other GPU-accelerated technologies and an expanded analysis of other JavaScript visualization libraries. The research could also be extended to various data types and visualization scenarios to evaluate the potential applications of these technologies in a broader range of projects.

REFERENCES

[1] Ae. Rininsland and S. Teller, *D3.Js 4.x Data Visualization: Learn to Visualize Your Data with JavaScript*. Birmingham, UK: Packt Publishing, 2017

[2] E. Meeks, *D3.Js in Action: Data Visualization with JavaScript*, 2nd Edition. Shelter Island, NY: Manning Publications, 2018.

[3] M. Iglesias, *Pro D3.Js: Use D3.Js to Create Maintainable, Modular, and Testable Charts*. Berkeley, CA: Apress, 2019. DOI: https://doi.org/10.1007/978-1-4842-5203-1

[4] "WebGL fundamentals," WebGL Fundamentals, https://webglfundamentals.org/ (accessed Sep. 20, 2024).

[5] K. Matsuda and R. Lea, *WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL*. Upper Saddle River, NJ: Addison-Wesley, 2013.

[6] A. Belkin, N. Gelernter, and I. Cidon, "The risks of WebGL: Analysis, Evaluation and detection," *Lecture Notes in Computer Science*, pp. 545–564, 2019. doi:10.1007/978-3-030-29962-0_26

[7] T. Akenine-Möller, *Real-Time Rendering Tomas Akenine-Möller, Eric Haines, Naty Homan, Angelo Pesce, Michael Iwanicki, Sébastien Hillaire*, 4th ed. Boca Raton: CRC Press, Taylor & Francis Group, 2018

[8] I. Gruzdo, I. Kyrychenko, G. Tereshchenko, and N. Shanidze, "Metrics applicable for evaluating software at the design stage," *5th International Conference on Computational Linguistics and Intelligent Systems* (COLINS), Kharkiv, Ukraine, 2021, vol. 2870, pp. 916-936.

[9] A. Ishizaka and P. Nemery, *Multi-Criteria Decision Analysis: Methods and Software.* Hoboken, NY: Wiley, 2013/ DOI: https://doi.org/10.1002/9781118644898

[10] K. Smelyakov, D. Pribylnov, V. Martovytskyi, and A. Chupryna, "Investigation of network infrastructure control parameters for effective intellectual analysis," *14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering*, TCSET 2018 - Proceedings, April 2018, pp. 983–986.DOI: https://doi.org/10.1109/TCSET.2018.8336359

[11] N. Q. Zhu, *Data Visualization with D3 4.x Cookbook: Discover over 65 Recipes to Help You Create Breathtaking Data Visualizations Using the Latest Features of D3*, 2nd Edition. Birmingham, UK: Packt Publishing, 2017.

[12] "D3," by Observable | The JavaScript library for bespoke data visualization, https://d3js.org/ (accessed Sep. 24, 2024).

[13] P. Cozzi, *WebGL Insights*. Boca Raton: CRC Press/Taylor & Francis Group, 2016.

[14] S. Murray, Interactive Data Visualization for the Web: An Introduction to Designing with D3, 2nd ed. Sebastopol, CA: O'Reilly Media, 2017.

[15] T. Parisi, *WebGL: Up and Running*. Sebastopol, Calif: O'Reilly Media, 2012.

Research on the efficiency of applying WebGL and D3.js in the creation of graphical elements in web applications

Nataliia Rusakova
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
0000-0002-1242-6602

Nataliia Ponikarovska
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
0009-0000-5243-3287

Mariya Shirokopetleva
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
0000-0002-7472-6045

***Abstract*— With the rising demand for data visualization in web applications, choosing an efficient tool has become critically important for developers. However, as data volumes grow, challenges related to performance and ensuring seamless interactive user experience arise, complicating the choice of the optimal visualization approach. This study compares two popular technologies—D3.js and WebGL—used for rendering graphical elements. The analysis is based on key criteria, including hardware rendering specifics, performance, community support, extensibility, and limitations. Recommendations are provided regarding the appropriate use of visualization technologies depending on task-specific requirements and application conditions, formulated through a multi-criteria analysis.**

Keywords*— *WebGL, D3.js, visualization, performance, rendering, dynamic data, large datasets, criteria, multi-criteria analysis.

I. INTRODUCTION

Visualization plays a crucial role in identifying data patterns in modern information systems, which focus on analyzing large datasets. Big data serves as a novel source of insights into the surrounding world and the dynamics of social processes, making it a valuable foundation for research [1]. As Mike Bostock, the author of the D3.js library, said: "The purpose of visualization is insight, not pictures" [2], emphasizing that the primary goal of graphical data representation is not merely to create visually appealing images, but to enable deeper insight into information and support the creation of an intuitive, well-designed interface [3]. Visualization enables the representation of complex relationships in an accessible form. However, as data volumes increase, the question of efficiency arises [4]. Not all visualization approaches can ensure high performance and smooth interactivity as the number of elements grows.

One of the key challenges in this field is choosing an appropriate visualization tool. Given the wide range of solutions based on different technological approaches, evaluating their effectiveness is difficult without comparison. Some libraries offer high-quality rendering but may be less efficient when handling large datasets, while others prioritize speed at the expense of visual complexity. To make an informed choice, it is recommended to use multi-criteria analysis to evaluate visualization tools based on various parameters.

One of the most widely used solutions for visualization is D3.js [5, 6]. This library provides extensive capabilities for creating flexible and interactive graphical elements by utilizing a declarative approach and DOM manipulations through SVG. Due to its large developer community, well-documented API, and numerous examples, D3.js is actively used across various fields, from academic research to business analytics. However, given the nature of D3.js, the question of its performance scalability when handling large datasets remains open.

An alternative approach to visualization is WebGL [7, 8], a low-level technology that enables direct interaction with the graphics processor (GPU) for fast rendering of complex scenes. WebGL serves as the foundation for many modern visualization solutions, particularly those requiring the processing of large data points, 3D graphics, or real-time animations. However, the use of WebGL often demands additional computational resources and involves a steeper learning curve, which can present challenges for its adoption in standard web-based analytical systems.

Prior research [9] demonstrates that mastering WebGL requires significant effort, involving understanding shaders, buffers, and GPU memory management. WebGL follows a low-level, imperative approach, requiring developers to manually manage rendering pipelines and optimize performance, which demands not only technical proficiency but also a deep understanding of computer graphics principles. D3.js abstracts much of this complexity by providing a declarative paradigm, where developers can focus on data transformations rather than the intricacies of GPU operations. This fundamental difference in learning approaches impacts adoption: WebGL necessitates a steep learning curve with an emphasis on graphics programming, while D3.js allows for a more intuitive entry point. Building on this, the current study uses multi-criteria analysis to assess the efficiency of D3.js and WebGL in different visualization scenarios, considering not only the above but also other criteria.

Thus, this paper presents a multi-criteria study of the efficiency of D3.js and WebGL to evaluate their suitability for various visualization scenarios. The analysis will be based on criteria such as hardware rendering specifics, performance, community support, extensibility, and limitations. This approach will provide an objective assessment of the strengths

and weaknesses of each technology and help determine their applicability in modern web application.

II. Theoretical Foundations

Decision-making in complex multifactorial contexts requires the integration of multiple criteria with varying characteristics and levels of significance. The use of a formalized methodology enables a structured approach to consolidating all aspects of the problem, transforming subjective assessments into quantitative indicators, and minimizing bias, thereby ensuring a well-founded choice. This study proposes the application of the MCDA methodology [9], which allows for the aggregation of diverse indicators into a single integral index.

The methodology is implemented in four main stages:

- Problem structuring – classification of the task by type.
- Definition and evaluation of criteria – identification and qualitative assessment of each criterion according to the specified requirements.
- Aggregation of results – integration of criteria into a single generalized indicator.
- Sensitivity analysis – assessment of the impact of changes in weight coefficients on the final result.

At each stage, additional data processing methods are employed. The distribution of weight coefficients is carried out using the proportional method, which ensures a balanced representation of all criteria. Normalization of values is performed using the min-max normalization method, allowing for comparing criteria measured on different scales. To integrate the normalized values into a comprehensive index, a linear additive aggregation with weight coefficients is applied, ensuring accurate data aggregation and interpretability. Building on these theoretical foundations, the study applies the MCDA methodology to address the practical challenge of selecting visualization tools.

III. Problem Structuring

Within the MCDA methodology, choice is one of the fundamental decision problem types, alongside ranking, sorting, and description. It focuses on selecting the most suitable option from a given set, ensuring that the chosen alternative best aligns with the decision-maker's objectives and constraints. Unlike ranking, which orders all alternatives, or sorting, which groups them into predefined categories, the choice problem requires a final selection based on a structured evaluation of multiple criteria.

The task of this study falls into the choice category, as its primary goal is to identify the optimal technology for web visualization of large datasets across various application scenarios. Multiple factors, such as performance, scalability, etc., influence this decision; the choice must balance trade-offs rather than maximize a single criterion. MCDA provides a structured approach to this process, allowing for a transparent and justifiable selection based on a comprehensive evaluation of competing alternatives.

IV. Definition and Evaluation of Criteria

A. *Hardware rendering specifics*

The browser utilizes both the CPU and GPU for rendering The browser utilizes both the CPU and GPU for rendering graphics, but their roles differ. The CPU handles primary computations, processes the DOM structure, and manages the rendering of vector graphics, while the GPU accelerates specific tasks, such as processing transformations (transform, opacity, filter). In the case of SVG, which is rendered through the DOM, the primary load falls on the CPU. It processes all vector paths, performs geometry calculations, and draws elements. The GPU may be involved only in the final image rendering if the browser applies hardware acceleration for style processing or visual effects.

For simple visualizations, the difference between CPU and GPU usage is negligible. However, when handling interactive time series charts with thousands of data points, the CPU becomes a bottleneck. Therefore, when evaluating this criterion, it is crucial to determine whether a technology leverages GPU acceleration or relies solely on the CPU.

Evaluating a technology based on this criterion is important, as the efficient workload distribution between the CPU and GPU determines how stable and predictable visualization performance will be under different conditions. Computational resources are limited, and improper allocation can lead to inefficient device power use. Understanding which part of the system handles key calculations allows for consideration of the technical capabilities of target devices and ensures optimal performance with minimal resource waste.

TABLE I. Hardware Rendering Specifics Testing

Technology	GPU load share
D3.js	0.1
WebGL	0.95

B. *Performance*

This criterion evaluates the technology's ability to ensure fast rendering and minimize latency. Performance can be assessed using relevant metrics.

Rendering Time refers to the time required to display the visualization on the screen. High values can cause delays in data representation, which is critical for interactive elements.

FPS (frames per second) measures the number of frames rendered per second, determining the smoothness of updates. A drop below 30 FPS results in noticeable stuttering, while at 100+ FPS, interactions appear instantaneous.

High performance is achieved through rapid data processing and minimising unnecessary computations. It is crucial that only the required elements are updated during visualization changes, animations run without delays, and the time between user interaction and result display remains minimal.

Evaluating a technology based on this criterion is important, as rendering speed directly affects usability and data accuracy.

Rendering delays can significantly impact the user experience and overall interaction with the visualization and data.

The measurements were conducted using Chrome DevTools, a built-in tool for analyzing web page performance in the Google Chrome browser. Tests were performed under controlled conditions (version 127.0.6533.119). The evaluation of technology performance in different scenarios is a fundamental aspect of its effective use [11]. Therefore, measurements were performed across three use cases with varying levels of computational load.

TABLE II. PERFORMANCE TESTING

Technology	RT	FPS
Scenario 1 (static chart, 1,000 points)		
D3.js	165	120.2
WebGL	149	119
Scenario 2 (static chart, 10,000 points)		
D3.js	271	115.4
WebGL	111	120.2
Scenario 3 (dynamic chart, + 10,000 points each 5 sec.)		
D3.js	919	80.1
WebGL	179	119.3

C. Community support

In this context, a community is understood as a group of individuals who actively create, use, support and develop a specific technology. It includes developers, contributors, users, and those who generate educational materials, build plugins, or provide support. The community functions as an ecosystem that facilitates knowledge exchange, contributes to problem-solving, and continuously enhances the tool itself.

Community support indicators can be tracked through GitHub repository metrics [12, 13]. Metrics such as watch, fork, and star provide insight into the popularity and reach of the community. The number of forks reflects the project's popularity among developers who intend to improve the code or use it as a foundation for their own projects. The number of watches indicates the level of interest in the project, while the number of stars represents the overall popularity of the repository.

Evaluating a technology based on this criterion is essential, as an active community provides support, facilitates knowledge sharing, and drives tool development. A broad network of developers, users, and contributors simplifies problem-solving, accelerates solution discovery, and promotes continuous technology improvement.

TABLE III. COMMUNITY SUPPORT TESTING

Technology	Fork	Watch	Star
D3.js	3609	22900	109000
WebGL	117	670	2700

D. Extensibility

Since project requirements may evolve over time — such as increasing data volumes, adding complexity to functionality, or incorporating interactive components — extensibility is a crucial criterion. A flexible solution allows adaptation to changes without significant time and resource expenditures, ensuring longevity and minimizing the risk of technical limitations. D3.js follows a constructor-based approach, where the interface is built from individual elements rather than configured through a parameterized object. This provides high flexibility, as developers have direct control over the structure and logic of the visualization. WebGL, as a low-level API, offers virtually unlimited extensibility, with its only constraints being the developer's expertise and creative vision.

Evaluating a technology based on this criterion is important, as it determines how easily a solution can adapt to evolving project requirements. Limited extensibility may lead to significant costs for adjusting the approach or even switching to a different tool. A flexible solution enables scalability, supports the addition of interactive elements, and handles larger data volumes without requiring fundamental architectural changes.

TABLE IV. EXTENSIBILITY TESTING

Technology	Approach to interacting with the interface
D3.js	design approach
WebGL	low-level API

E. Limitations

The limitations of a technology define its suitability for widespread use. One key factor is browser support. Some graphical technologies may have limited compatibility or exhibit differences across environments. For example, WebGL relies on hardware acceleration and graphics drivers, which can cause issues on older devices in environments where WebGL is disabled or in browsers with partial support. In contrast, SVG is the standard for vector graphics on the web, ensuring stable compatibility across all modern browsers. The choice of technology must consider these limitations, especially when targeting a broad user base.

Another critical factor is accessibility. Visualizations should be understandable to users with visual impairments or specific cognitive and sensory needs, necessitating support for alternative text descriptions, navigation, and screen reader compatibility. SVG integrates with the DOM and supports attributes such as aria-label and title, making it inherently more accessible. WebGL, on the other hand, lacks built-in accessibility mechanisms, as its graphical content is rendered within a Canvas element, making it inaccessible to assistive technologies. This can present additional challenges for developers aiming to ensure an inclusive user experience.

Evaluating a technology based on this criterion is essential, as technical limitations determine how stable and accessible a solution will be in real-world conditions. Choosing a tool that does not account for browser compatibility or the needs of different user groups may lead to unforeseen challenges, requiring additional resources for adjustments or workarounds.

TABLE V. LIMITATIONS TESTING

Technology	Support in browsers	Accessibility
D3.js	1	1
WebGL	0.9	0

V. AGGREGATION

Evaluating technologies across different criteria requires comparing metrics with varying units of measurement and value ranges. For instance, CPU and GPU load share is represented as a relative fraction (0–1), whereas community popularity is measured in thousands of stars, forks, and watchers on GitHub. To accurately compare these criteria within a unified system, they must be converted to a common scale, eliminating the influence of absolute values and focusing on relative differences between technologies.

The standard normalization approach, which transforms diverse criteria into a unified scale from 0 to 1, is fundamental for objective comparison and subsequent weight-based analysis. However, when decision-making requires preserving the influence of each criterion, conventional normalization, which may result in zero values, can become problematic. Therefore, the method was adapted to prevent the occurrence of zero values, ensuring that all criteria retain their significance in the evaluation process:

$$f = \frac{f_{meas} - f_{min}}{f_{max} - f_{min}} \times 0.9 + 0.1a \quad (1)$$

The calculations were performed using min-max normalization of the indicators, followed by scaling to prevent zero values. When necessary, the results were adjusted to their arithmetic mean.

TABLE VI. NORMALIZED VALUES

Criterion	D3.js	WebGL
Hardware rendering specifics	0.1	1
Performance	0.63	0.97
Community support	1	0.1
Extensibility	0.1	1
Limitations	1	0.1

The normalized results are essential for the next stage of analysis – sensitivity assessment.

VI. SENSITIVITY ANALYSIS

As the first step, a linear additive aggregation is applied with weight coefficients, assuming all criteria are equally significant. In this case, each weight coefficient is set to 0.2.

$$TS = H \times W_H + P \times W_P + C \times W_C + E \times W_E + L \times W_L \quad (2)$$

WebGL achieved a higher overall score (0.634) due to its superior performance and extensibility. D3.js received a lower score (0.566) but excels in community support and accessibility.

TABLE VII. ANALYSIS WITH EQUAL WEIGHTS

Technology	TS
D3.js	0.566
WebGL	0.634

The choice of technology always depends on the context, specifically the project requirements. Therefore, two usage scenarios are considered to determine which technology is best suited for specific tasks.

Next, let's consider two usage scenarios. The first is the development of a dynamic, high-performance chart for a closed analytical platform designed for web visualization of large datasets (100,000 data points). The timeframe for learning and MVP development is limited to six months.

The second of the proposed usage scenarios is the development of ten static charts (pie chart, histogram, scatter plot) with minimal interactive elements for an open educational platform.

Next, weighting factors are determined according to the proposed usage scenarios and the priority of tasks using the proportional method.

TABLE VIII. WEIGHTS OF SCENARIOS

Criterion	Weight for sc.1	Weight for sc.2
Hardware rendering specifics	0.267	0.067
Performance	0.333	0.267
Community support	0.133	0.200
Extensibility	0.200	0.133
Limitations	0.067	0.333

Here are the results of linear additive aggregation with defined weights for both use cases.

TABLE IX. RESULT OF SCENARIOS

Technology	TS of scenario 1	TS of scenario 2
D3.js	0.353	0.733
WebGL	0.813	0.433

The results of the linear additive aggregation with the defined weights indicate that WebGL is the most suitable choice for the first scenario.

And for usage scenario 2, the results show that D3.js is the most suitable choice.

VII. DISCUSSION AND RECOMMENDATIONS

The analysis of the obtained results demonstrates that the choice of web visualization technology should be based on the project's specific requirements. WebGL had a slight advantage over D3.js when weight coefficients were evenly distributed, indicating its versatility across different scenarios. However, the sensitivity analysis revealed that final scores are significantly influenced by project priorities.

In the first scenario, where the key criteria were performance and rendering efficiency, WebGL achieved a significantly higher final score. This is explained by the fact that the technology is optimized for handling large datasets and fully utilizes the graphics processor. While D3.js offers a convenient API and robust capabilities for working with SVG, it cannot compete with WebGL in terms of visualization speed. Therefore, WebGL is the more suitable choice if a project involves intensive data processing and requires high performance.

The second scenario produced the opposite result: when accessibility, community support, and ease of integration were prioritized, D3.js emerged as the preferred option. This is due to its strong popularity among developers, the availability of a vast number of ready-made examples, and its adaptability to a wide range of tasks without imposing significant graphical rendering requirements.

Based on these findings, several recommendations can be made. WebGL provides superior performance for projects that require high interactivity and efficient processing of large datasets. In cases where accessibility, ease of deployment, and fast implementation are more important, D3.js should be the preferred choice. The final selection of technology should be guided by a clear definition of priorities, as they directly influence the weight coefficients of the criteria and, consequently, the overall evaluation.

VIII. CONCLUSIONS AND FUTURE WORK

The proposed methodology has demonstrated the effectiveness of a structured approach to selecting web visualization technologies for solving multifactorial challenges. The study confirmed that WebGL is a highly efficient solution for rendering large datasets due to its superior performance and scalability. Its ability to leverage GPU acceleration allows for real-time processing of complex visualizations, making it particularly suitable for applications that require high interactivity and the handling of extensive data volumes. However, the technical complexity of WebGL remains a significant limitation. The necessity for lower-level programming, the absence of built-in abstractions, and the steep learning curve may hinder its adoption in projects with tight development schedules or limited resources. In contrast, D3.js provides a more accessible alternative, offering a well-documented API, an extensive community, and a broad range of ready-to-use visualization components.

Future research should focus on mitigating the barriers associated with WebGL development while enhancing its practical applicability. Refining high-level abstraction layers, optimizing rendering pipelines, and automating routine tasks could significantly improve WebGL's usability. By addressing these challenges, WebGL could become more accessible to a wider range of developers while maintaining its advantages in high-performance rendering. At the same time, continued advancements in declarative visualization frameworks like D3.js will ensure that developers have flexible and efficient tools for creating scalable web-based visualizations. Future efforts should aim at bridging the gap between these two technologies by exploring hybrid solutions that integrate the performance of WebGL with the usability of D3.js, thereby creating more versatile and adaptive visualization frameworks suitable for diverse application needs.

REFERENCES

[1] O. Kyslova, "Big Data in the Context of Studying Problems of Modern Society," SSMS, vol. 42, pp. 59-68, Dec. 2019, DOI:10.26565/2227-6521-2019-42-06.

[2] M. Bostock, "Keynote - CSVConf 2017," Jan, 2018. [Online]. Available: https://www.youtube.com/watch?v=aT4JvF7sglg&ab_channel=csvconf

[3] I. Gruzdo, I. Kyrychenko, G. Tereshchenko, and O. Shanidze, "Analysis of Models Usability Methods Used on Design Stage to Increase Site Optimization," 7th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2023), vol. 3403, pp. 387– 409, 2023.

[4] T. Akenine-Möller, E. Haines, and N. Hoffman, Real-Time Rendering, 4th ed. Boca Raton, FL, USA: CRC Press, 2018.

[5] E. Meeks, D3.js in Action: Data Visualization with JavaScript, 2nd ed. Shelter Island, NY, USA: Manning, 2018.

[6] M. Iglesias, Pro D3.js: Use D3.js to Create Maintainable, Modular, and Testable Charts. Berkeley, CA, USA: Apress, 2019, DOI: 10.1007/978-1-4842-5203-1.

[7] G. Niemeyer, "WebGL Fundamentals," Jan. 2025. [Online]. Available: https://webglfundamentals.org. [Accessed: Jan. 28, 2025].

[8] P. Cozzi, WebGL Insights. Boca Raton, FL, USA: CRC Press, 2015.

[9] A. Ishizaka and P. Nemery, Multi-Criteria Decision Analysis: Methods and Software. New York, NY, USA: Wiley, 2013, doi: 10.1002/9781118644898.

[10] N. Rusakova, N. Ponikarovska, and M. Shirokopetleva, "Comparative analysis of WebGL and D3.js," in Problems of Infocommunications. Science and Technology (PIC S&T′ 2024), in press.

[11] K. Smelyakov, A. Chupryna, D. Darahan, and S. Midina, "Effectiveness of modern text recognition solutions and tools for common data sources," in Proc. Int. Conf. Computational Linguistics and Intelligent Systems (COLINS-2021), vol. 2870, 2021, pp. 154–165. [Online]. Available: https://ceur-ws.org/Vol-2870/paper15.pdf.

[12] M. Bostock, "D3.js", [Online]. Available: https://github.com/d3/d3. [Accessed: Jan. 28, 2025].

[13] Khronos Group, "WebGL", [Online]. Available: https://github.com/KhronosGroup/WebGL. [Accessed: Jan. 28, 2025].

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗздм-23-1 (група)

Понікаровська Наталія Андріївна

(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

06.06.2025