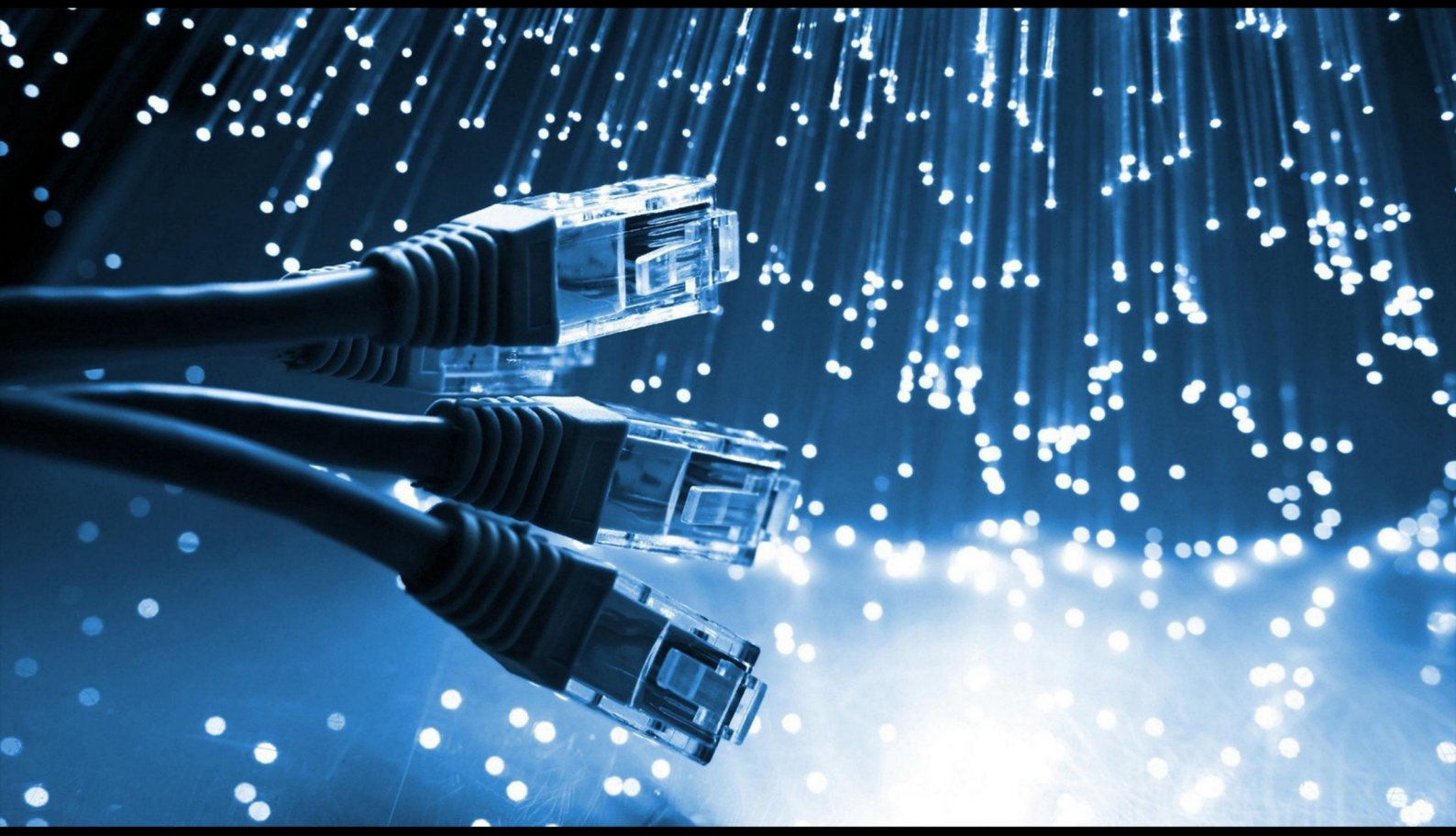
JCSI

Journal of Computer Sciences Institute

Volume 20/2021



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

Redakcja JCSI
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Katedra Informatyki
Wydział Elektrotechniki i Informatyki

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

Redaktor naczelny:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Redaktor techniczny:
Beata Pańczyk,
e-mail: b.panczyk@pollub.pl

Recenzenci numeru:
dr Marcin Barszcz
dr Mariusz Dzieńkowski
dr inż. Dariusz Gutek
dr Edyta Łukasik
dr inż. Tomasz Szymczyk
dr inż. Marcin Badurowicz
dr inż. Sylwester Korga
dr inż. Maria Skublewska-Paszkowska
dr inż. Krzysztof Dziedzic
dr inż. Jacek Kęsik
dr inż. Małgorzata Plechawska-Wójcik
dr inż. Jakub Smołka
dr inż. Grzegorz Kozieł

Skład komputerowy:
Anna Sałamacha
e-mail: a.salamacha@pollub.pl

Projekt okładki:
Marta Zbańska

JCSI Editorial
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Department of Computer Science
Faculty of Electrical Engineering and Computer Science
Lublin University of Technology
ul. Nadbystrzycka 36 b
20-618 Lublin, Poland

Editor in Chief:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Assistant editor:
Beata Pańczyk,
e-mail: b.panczyk@pollub.pl

Reviewers:
Marcin Barszcz
Mariusz Dzieńkowski
Dariusz Gutek
Edyta Łukasik
Tomasz Szymczyk
Marcin Badurowicz
Sylwester Korga
Maria Skublewska-Paszkowska
Krzysztof Dziedzic
Jacek Kęsik
Małgorzata Plechawska-Wójcik
Jakub Smołka
Grzegorz Kozieł

Computer typesetting:
Anna Sałamacha
e-mail: a.salamacha@pollub.pl

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 165–169
Received: 29 April 2021
Accepted: 26 May 2021

Tools for analysis of business processes – a comparative analysis

Narzędzia do analizy procesów biznesowych – analiza porównawcza

Jakub Janicki*, Ernest Wójcik

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

Abstract

This article discusses topics in the field of business analysis, which is the basic element of the operation of modern organizations. The aim of this publication is to present and evaluate four selected tools for modeling business diagrams in BPMN notation. As part of the introduction to the topic, business processes were characterized, their analysis was defined, and the reader was introduced to the BPMN notation and the tested tools. The research methodology and its course were also defined. Then, the research was carried out in accordance with the adopted methodology. The conclusion of this work was the result of the research together with the analysis. Finally, the conducted experiments were summarized together with conclusions.

Keywords: Business processes; BPMN; business diagrams

Streszczenie

Niniejszy artykuł porusza tematy z zakresu analizy biznesowej, będącej podstawowym elementem funkcjonowania współczesnych organizacji. Celem tej publikacji jest przedstawienie i ocena czterech wybranych narzędzi do modelowania diagramów biznesowych w notacji BPMN. W ramach wprowadzenia do tematu scharakteryzowano procesy biznesowe, zdefiniowano ich analizę, a także przybliżono czytelnikowi notację BPMN i badane narzędzia. Zdefiniowano też metodykę badań i ich przebieg. Następnie przeprowadzone zostały badania zgodnie z przyjętą metodyką. Rezultatem tych prac były wyniki badań razem z analizą. Na koniec podsumowano przeprowadzone pomiary łącznie z wnioskami.

Słowa kluczowe: Procesy biznesowe; BPMN; diagramy biznesowe

*Corresponding author:

Email address: jakub.janicki@pollub.edu.pl (J. Janicki)

1. Wstęp

Procesy biznesowe są nieodłącznym elementem każdej większej firmy. Do ich analizy stosowane są programy, które pozwalają w mniej lub bardziej efektywny sposób z nimi pracować. Liczba narzędzi służących do analizy procesów biznesowych wciąż się powiększa, w związku z czym nie jest łatwo znaleźć optymalną aplikację realizującą wszystkie potrzeby użytkowników.

W celu rzetelnego i skutecznego przeprowadzania badań, zaznajomiono się z wieloma materiałami naukowymi. Istotną pozycją na liście literatury jest książka o tytule "Fundamentals of Business Process Management" [1], wprowadzająca czytelnika w temat cyklu życia procesów biznesowych. Niezwykle ważne było też zaznajomienie się z dokumentacją ocenianych narzędzi. Umożliwiło to poznanie pełnych funkcji programów i naukę korzystania z nich.

2. Cel i zakres pracy

Celem badań, będących tematem tego artykułu, jest porównanie jakości konkretnych narzędzi służących do analizy procesów biznesowych. Analiza będzie wykonana z wykorzystaniem następującej metodyki: zdefiniowanie sytuacji stosowania narzędzi, opracowanie kryteriów porównawczych i ich parametrów (zmiennych niezależnych), wybór metody porównania z wykorzystaniem zdefiniowanych kryteriów, określenie (doświadczalne) wartości zmiennych niezależnych i przeprowadzenie finalnych obliczeń. Porównanie będzie poprzedzone przeglądem narzędzi i wyborem czterech z nich do analizy porównawczej.

Zakres pracy obejmuje przegląd istniejących narzędzi do analizy procesów biznesowych, przegląd metod porównywania z uwzględnieniem wielu kryteriów i dobór jednej z nich, przeprowadzenie eksperymentu w celu określenia parametrów każdego z wybranych narzędzi, przeprowadzenie obliczeń oraz przedstawienie wniosków.

3. Procesy biznesowe i ich analiza

3.1. Procesy biznesowe

Procesy biznesowe, zwane również metodami biznesowymi, stanowią powiązane ze sobą zadania zmierzające ku osiągnięciu zamierzonego efektu. Niezależnie od nazwy funkcjonują w każdej firmie, a najistotniejszym celem każdego z nich jest zrozumienie potrzeb klienta, możliwości zespołu dostawcy oraz słabych i mocnych stron procesu [2].

3.2. Analiza procesów

Metodą, która pozwala przeanalizować, czy istniejące procesy spełniają założone im cele nazywa się analizą procesów. Dzięki tej metodzie zespół jest w stanie zlokalizować niekorzystne fragmenty procesu, znaleźć ich przyczynę, a także określić w jaki sposób rozwiązać problem [3].

Przy użyciu narzędzi, a także metodologii, możliwe jest spojrzenie na procesy w różny sposób oraz śledzenie produktywności i wydajności. Jest to istotne podczas analizy procesów, ze względu na ich złożoność z zadań, które charakteryzuje określony cel końcowy. Przy analizie koncentracja skierowana jest na obserwację sposobów zachodzących w cyklu życia, od początku do końca procesu [4].

Dzięki szerokiemu zakresowi, analiza procesów wykorzystywana jest w szerszym gronie organizacji oraz przedsiębiorstwach o coraz szerszej specjalizacji. Stosowana jest przede wszystkim do organizacji pracy, wprowadzania nowych pracowników, planowania obciążenia stanowiska oraz ustalania poziomu osiągnięć w organizacji [5].

3.3. Business Process Model and Notation

W skrócie BPMN, czyli "Notacja i Model Procesu Biznesowego". Jak sama nazwa wskazuje, to notacja graficzna przedstawiająca proces biznesowy. Notacja BPMN utrzymywana jest obecnie przez OMG (Object Management Group), konsorcjum, w którego skład wchodziły firmy takie jak Apple czy IBM. Stworzona została natomiast dzięki Business Process Management Initiative (BPMI), organizacji non-profit, która istnieje aby promować standaryzację popularnych procesów biznesowych, jako środek wspierania rozwoju e-biznesu [6].

4. Przegląd narzędzi analizy procesów biznesowych

4.1. Bizagi Modeler

Bizagi BPMN Modeler to aplikacja umożliwiająca tworzenie graficznych diagramów, dokumentowanie, a także symulowanie procesów w notacji BPMN wytworzona przez firmę Bizagi w 2008 roku. Z wykorzystaniem modelera Bizagi, diagramy procesów mogą być publikowane między innymi w Wordzie, PDF, a także Sharepoint. Pozwala on na pracę w chmurze symultanicznie z innymi członkami organizacji [6].

4.2. Lucidchart

Lucidchart to aplikacja internetowa służąca do rysowania i udostępniania wykresów i diagramów wydana przez Lucid Software Inc. w 2008 roku [6]. Łączy w sobie elementy aplikacji Microsoft Excel, Adobe Photoshop oraz Aplikacji Google. Zbudowana została w oparciu o technologię HTML5 oraz JavaScript, co oznacza że aplikacja ta nie wymaga aktualizacji oprogramowania stron trzecich takiego jak Adobe Flash [7].

4.3. Aris express

ARIS Express to narzędzie przeznaczone do modelowania, analizy i zarządzania procesami biznesowymi stworzona przez Software AG w 2009 roku. Wydane jest na licencji freeware, co oznacza że można rozprowadzać ją za darmo, bez ujawnienia kodu źródłowego. Wspiera wiele notacji takich jak EPC, krajobrazy procesów, ERM i BPMN 2. Jedną z funkcjonalności wyróżniających ARIS Express na rynku narzędzi modelujących procesy biznesowe są fragmenty logiki, które można przechowywać i ponownie używać ich w innych modelach [8]

4.4. Gliffy

Gliffy to osadzona w chmurze aplikacja internetowa służąca do tworzenia diagramów wydana w 2007 roku przez dwóch programistów - Chrisa Kohlhardta i Clinta Dicksona [9]. Pozwala na tworzenie między innymi diagramów UML i Vienna, a także schematów pomieszczeń. Narzędzie to jest zintegrowane z serwisami JIRA i Confluence, co jest dużą zaletą tego rodzaju rozwiązań. Podobnie jak Lucidchart, Gliffy opiera się na technologii HTML5 [10].

5. Metodyka badań porównawczych

5.1. Hipotezy i kryteria

W niniejszym artykule zdecydowano przedstawić odpowiedzi na następujące pytania:

- Który program prezentuje najbardziej przejrzysty interfejs dla użytkownika?
- Który program zapewnia największą liczbę poradników?
- Który program zapewnia największą liczbę możliwości eksportu plików?

Starając się odnaleźć odpowiedzi na pytania badawcze zdefiniowano dwie hipotezy badawcze:

- H1. Program Lucidchart stanowi najbardziej kompleksowe rozwiązanie dla zespołu projektu.
- H2. Program Aris Express zapewnia największą liczbę możliwości importu oraz eksportu plików.

Przy ocenie użyteczności badanych programów, zastosowano następujące kryteria:

- przejrzystość interfejsu – intuicyjność i prostota modelowania procesów biznesowych wartościowana w punktach;
- wygląd aplikacji – prezentacja pola roboczego, zakładek z narzędziami oraz prezentacja ogólna aplikacji wartościowana w punktach;
- łatwość rejestracji – proces rejestracji nowego użytkownika wartościowany w punktach;
- jakość poradników w postaci dokumentacji – ocena wartościowości oraz przydatności poradników w postaci dokumentacji wartościowana w punktach;
- jakość poradników w postaci filmów online – ocena wartościowości oraz przydatności poradników w postaci filmów online wartościowana w punktach;

- liczba poradników w postaci dokumentacji – ocena ilości dostępnych poradników w postaci dokumentacji wartościowana w punktach;
- liczba poradników w postaci filmów online – ocena ilości dostępnych poradników w postaci filmów online wartościowana w punktach;
- dostępne funkcjonalności – ocena zero jedynkowa, czy aplikacja posiada konkretną funkcjonalność.

Podczas przeprowadzania scenariuszy w programach, przyznawano 1 punkt za każdą dostępną (posiadaną) funkcjonalność, natomiast 0 punktów za jej brak.

W celu uzyskania większej ilości informacji, podkategorie: import oraz eksport plików, zostały zawarte w odrębnym zestawieniu niż dostępne funkcjonalności, natomiast kryterium również przewiduje 1 punkt za posiadaną funkcjonalność oraz 0 punktów za jej brak.

Kolejne kryteria również zostały objęte skalą punktową, programom można było przyznać od 0 do 5 punktów za przejrzystość interfejsu, wygląd aplikacji, łatwość rejestracji, jakość poradników w postaci dokumentacji, jakość poradników w postaci filmów online, ilość poradników w postaci dokumentacji oraz ilość poradników w postaci filmów online.

Ocena każdego z kryteriów była wyższa proporcjonalnie do liczby zdobytych punktów.

5.2. Scenariusze badawcze

W celu zbadania wymienionych kryteriów, w każdym z programów zostały wykonane analizy następujących procesów:

- wybór samochodu do zakupu dla rodziny 6 osobowej;
- obsługa zamówienia w sklepie internetowym;
- produkcja ołówków w zakładzie przemysłowym;
- kontrola stanu samochodu przed użytkowaniem.

6. Rezultaty badań i ich analiza

Niniejszy rozdział ma na celu prezentację uzyskanych rezultatów, które są wynikiem przeprowadzonej analizy porównawczej badanych programów na podstawie eksperymentów modelowania scenariuszy badawczych. Rezultaty w postaci zbiorczej zostały zaprezentowane na Tabeli 1.

Program Lucidchart posiada najwyższą liczbę dostępnych funkcjonalności wytypowanych do przeprowadzenia badań (ex aequo z Bizagi Modeler). Kolejne w kolejności programy Aris Express i Gliffy mają wyraźnie mniejszą liczbę punktów w tym kryterium co przedstawiono w formie wizualnej na Rysunku 1.

Kolejnym kryterium branym pod uwagę podczas badań był import i eksport plików za pośrednictwem programów. W tym kryterium również najwyższą liczbę punktów zdobyły programy Lucidchart oraz Bizagi Modeler, gdzie ten drugi miał większą liczbę punktów w kategorii eksportu plików. Omawiane kryterium jest istotne jeżeli zespół pracowników pracuje na nie ujednoliconym środowisku i konieczne jest przesyłanie plików z modelami w celu dalszej pracy. Programy Aris Express oraz Gliffy ponownie wyraźnie odstają od pierwszych dwóch programów. Program Gliffy nie zdobył żadnego punktu zarówno przy imporcie jak i eksporcie plików, co bardzo zmniejsza jego atrakcyjność. Wyniki omawianego kryterium zostały przedstawione w formie graficznej na Rysunku 2.

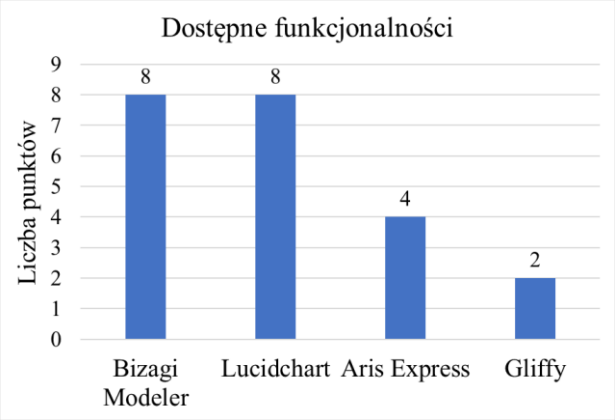
Tabela 1: Zbiorcze zestawienie punktów przyznanych programom

Program / Kryterium	Bizagi Modeler	Lucid-chart	Aris Express	Gliffy
Dostępne funkcjonalności	8	8	4	2
Import plików	3	3	1	0
Eksport plików	5	4	1	0
Wygląd aplikacji	3	4	2	2
Przejrzystość interfejsu	3	4	2	2
Łatwość rejestracji	4	5	4	5
Jakość poradników w postaci dokumentacji	4	5	3	4
Jakość poradników w postaci filmów online	4	4	3	4
Ilość poradników w postaci dokumentacji	4	5	3	5
Ilość poradników w postaci filmów online	4	5	3	4
Suma	42 / 59	47 / 59	27 / 59	30 / 59

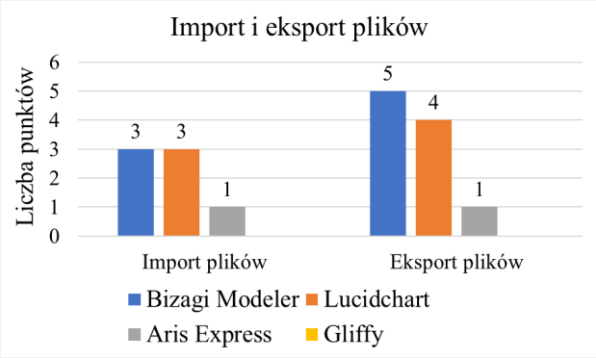
Ze względu na prostotę oraz otwartość na niewielką możliwość konfiguracji przestrzeni roboczej Lucidchart zdobył najwyższą liczbę punktów w kategorii wyglądu aplikacji oraz kategorii przejrzystości interfejsu, gdzie taką samą liczbę punktów (4 punkty) zdobył program Gliffy.

Rejestracja we wszystkich programach przebiegła w dość podobny sposób, nie powodując nieoczekiwanych problemów, dlatego programy uzyskały niemalże identyczna liczbę punktów w kategorii prostoty rejestracji. Programy Bizagi

Modeler i Aris Express zostały pozbawione punktu za konieczność nieco dłuższego przeszukania strony producenta w celu znalezienia formularza rejestracyjnego. W formie graficznej przedstawione zostały omówione trzy kryteria jako cechy niemierzalne na Rysunku 3.



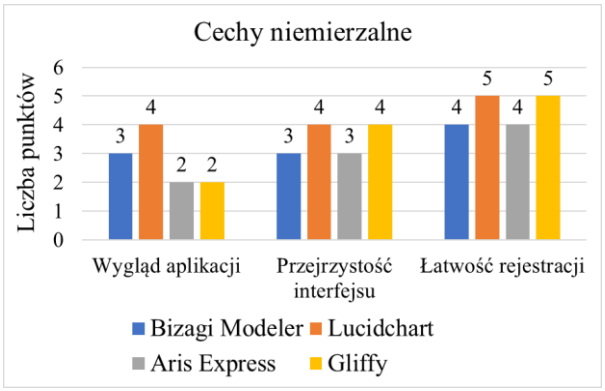
Rysunek 1: Wykres przedstawiający liczbę punktów zdobytych przez programy w zakresie dostępnych funkcjonalności.



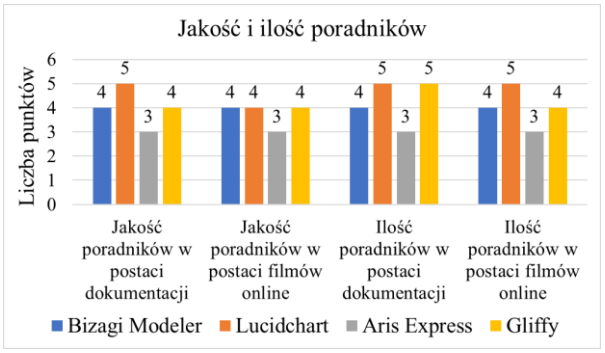
Rysunek 2: Wykres przedstawiający liczbę punktów zdobytych przez programy w zakresie importu i eksportu plików.

Zaskakująco każdy z badanych programów posiada nie małą ilość poradników w postaci dokumentacji oraz w postaci filmów online zarówno na stronach producentów programów jak i forach oraz darmowych stronach do zamieszczania filmów online. W kategoriach ilości poradników w postaci dokumentacji oraz filmów online największą ilość punktów zdobył program Lucidchart, przy czym dwa razy otrzymał maksymalną liczbę punktów, 5. Zaraz po nim uplasował się niewiele gorszy Gliffy, Bizagi Modeler oraz na ostatnim miejscu Aris Express.

W przypadku badanych programów ilość poradników została oceniona niemalże identycznie jak ich jakość, która w naszej ocenie jest ważniejsza od ilości. W przypadku kryterium jakości poradników w postaci dokumentacji i filmów online ponownie największą ilość punktów zdobył Lucidchart, na drugim miejscu ex aequo Bizagi Modeler oraz Gliffy, a na ostatnim miejscu Aris Express z trzema punktami w każdym z kryteriów. Wyniki w formie graficznej zbioru kryteriów dotyczących jakości oraz ilości poradników w postaci dokumentacji oraz filmów online przedstawiono na Rysunku 4.



Rysunek 3: Wykres przedstawiający liczbę punktów zdobytych przez programy w zakresie cech niemierzalnych.



Rysunek 4: Wykres przedstawiający liczbę punktów zdobytych przez programy w zakresie jakości i ilości poradników.

7. Wnioski

Celem niniejszego artykułu było przedstawienie wybranych narzędzi do modelowania diagramów BPPMN, razem z oceną ich funkcjonalności. Zadanie to zostało zrealizowane poprzez zastosowanie scenariuszy testowych badających konkretne możliwości każdego z badanych narzędzi. Ostateczny rezultat każdego z programów składa się z dwóch części składowych – oceny właściwości mierzalnych i niemierzalnych.

Z pośród badanych programów - Bizagi Modeler, Lucidchart, Aris Express i Gliffy – narzędziem które zdobyło najwięcej punktów jest Lucidchart, z przewagą 5 punktów nad następnym programem, czyli Bizagi Modeler. Warto zaznaczyć, że w części mierzalnej oba te narzędzia zdobyły dokładnie taką samą liczbę punktów, jednak łatwość w obsłudze i przyjemny interfejs sprawiły że Lucidchart jest wyżej klasyfikowanym oprogramowaniem. Na trzecim miejscu plasuje się Gliffy, jednak jego przewaga nad narzędziem z najniższą ilością punktów, Aris Express, wynosi jedynie 3 punkty.

Warto zauważyć że „zwycięski” program – Lucidchart – zdobył niespełna 80% wszystkich możliwych punktów, co oznacza że niestety nie jest to

narzędzie idealne, niemniej jednak będzie dobrze sprawdzać się jako modeler dla diagramów BPMN.

W ramach badań przedstawiono pierwszą hipotezę, że program Lucidchart stanowi najbardziej kompleksowe rozwiązanie dla zespołu projektu. Program ten zdobył największą liczbę punków w sumie biorąc pod uwagę każde kryterium, dlatego należy przyjąć pierwszą hipotezę.

Druga hipoteza zdefiniowana, program Aris Express zapewnia największą liczbę możliwości importu oraz eksportu plików. Zwycięski sumarycznie program Lucidchart oraz drugi w zestawieniu program Bizagi Modeler zapewniają większą liczbę możliwości importu oraz eksportu plików. W efekcie tego należy odrzucić drugą hipotezę, ponieważ istnieją programy zapewniające większa liczbę możliwość importu oraz eksportu plików.

Podsumowując zaprezentowane wyniki badań powinno się stwierdzić, że pierwsza postawiona hipoteza jest prawdziwa, natomiast druga postawiona hipoteza okazałą się nieprawdziwa.

8. Literatura

[1] S. Bechhofer, M. Hauswirth, J. Hoffmann, M. Koubarakis: The Semantic Web: Research and Applications. Springer, 2008.

[2] Podstawy procesu biznesowego, https://www.mfiles.pl/pl/index.php/Proces_biznesowy [06.05.2021].

[3] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers: Fundamentals of Business Process Management. Springer, 2013.

[4] Informacje o Aris Community, https://www.ariscommunity.com/, [06.05.2021].

[5] Przewagi programu Gliffy, https://www.cnet.com/news/gliffy-the-online-visio-killer/, [06.05.2021].

[6] Eksport oraz publikowanie w Bizagi Modeler, https://help.bizagi.com/bpm-suite/en/index.html?where_to_share.htm, [06.05.2021].

[7] Przegląd usług dla program Gliffy, https://www.gliffy.com/pricing, [06.05.2021].

[8] Podstawy analizy procesów, https://www.mfiles.pl/pl/index.php/Analiza_proces%C3%B3w, [06.05.2021].

[9] Przegląd usług dla program Lucidchart, https://lucid.app/pricing/lucidchart#/pricing, [06.05.2021].

[10] Podstawy o Business Process Management Initiative, https://searchcio.techtarget.com/definition/Business-Process-Management-Initiative-BPMI, [06.05.2021]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 170–174
Received: 14 May 2021
Accepted: 26 May 2021

Comparative analysis of UIKit and SwiftUI frameworks in iOS system

Analiza porównawcza szkieletów programistycznych UIKit i SwiftUI w systemie iOS

Piotr Wiertel*, Maria Skublewska-Paszkowska

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

Abstract
The paper concerns a comparison of SwiftUI and UIKit frameworks, used in iOS application development. SwiftUI was introduced in 2019 as a successor to UIKit as a main tool for creating application views. The purpose of this article is to compare the time performance of these two frameworks. Four testing applications have been implemented for the research, 2 in each framework. The software was complementary. The defined thesis "SwiftUI is more time efficient for applications with data collection and many filled text fields" was proved.

Keywords: SwiftUI; UIKit; time performance

Streszczenie
Artykuł dotyczy porównania szkieletów programistycznych SwiftUI i UIKit, wykorzystywanych przy tworzeniu aplikacji na system iOS. SwiftUI został zaprezentowany w 2019 jako następca UIKit dla tworzenia widoków aplikacji. Celem artykułu jest wykonanie analizy porównawczej tych szkieletów programistycznych w celu określenia ich wydajności czasowej. Na potrzeby pracy zostały wykonane cztery aplikacje testowe w obydwu badanych technologiach. Opracowane programy są komplementarne. Postawiona teza: „SwiftUI jest bardziej wydajny czasowo dla aplikacji obsługujących kolekcję danych i wiele pól tekstowych" została udowodniona.

Słowa kluczowe: SwiftUI; UIKit; wydajność

*Corresponding author
Email address: **piotr.wiertel1@pollub.edu.pl** (P. Wiertel)

1. Wstęp

Wraz z popularyzacją urządzeń mobilnych na świecie zapotrzebowanie na aplikacje rośnie i jednocześnie proces ich efektywnego wytwarzania jest stale usprawniany. Jednym z dwóch czołowych systemów operacyjnych na urządzenia mobilne jest system iOS, którego udział w rynku jest mniejszy niż konkurencyjnego Androida, aczkolwiek cieszącym się większymi dochodami ze sprzedawanych aplikacji [1]. Firma Apple prezentując pierwszego iPhona w 2007 roku przedstawiła również pierwszy system operacyjny iOS (początkowo nazwanym iPhone OS) wzorowany na macOS, stosowanym w komputerach stacjonarnych. Kolejne aktualizacje systemu wydawane każdego roku dodają nowe funkcjonalności i zwiększają użyteczność urządzeń mobilnych, umożliwiając twórcom aplikacji usprawnianie tworzonych produktów.

Razem z rozwojem aplikacji mobilnych jakość narzędzi oferowanych programistom wytwarzającym aplikacje również stale ewoluuje, jest możliwość wyboru pomiędzy różnymi zintegrowanymi środowiskami programistycznymi. Powstają także nowe języki programowania. W przypadku systemu iOS pierwszy język platformy Objective-C został w znacznej mierze zastąpiony nowszym - językiem – Swift [2]. Tworzone przez Apple środowisko XCode również jest wzbogacone o nowe dodatki, a błędy, które często utrudniały tworzenie aplikacji są stale eliminowane.

W ostatnim czasie istotną zmianą w kontekście programowania na iOS jest prezentacja nowego szkieletu programistycznego SwiftUI, deklaratywnego rozwiązania promowanego przez firmę Apple jako nowoczesne narzędzie do tworzenia aplikacji [2]. Ma ono zostać następcą dotychczas stosowanego narzędzia UIKit, który jako dojrzały produkt na przestrzeni lat stał się standardem każdego programisty działającego w obrębie technologii Apple. Jako nowe rozwiązanie SwiftUI jest obecnie w niewielkim stopniu wykorzystywany w celu wytwarzania oprogramowania w celach komercyjnych, w szczególności w aplikacjach o większym stopniu złożoności [3]. Z biegiem czasu jednak sytuacja może ulec zmianie, dlatego w niniejszej pracy przedstawiono badanie porównawcze obydwu technik.

Celem niniejszego artykułu jest porównanie szkieletów programistycznych UIKit i SwiftUI. Postawiono następującą tezę badawczą: „SwiftUI jest bardziej wydajny czasowo dla aplikacji obsługujących kolekcję danych i wiele pól tekstowych". W obydwu porównywanych technologiach wykonano możliwie podobne aplikacje testowe. Badania polegały na zmierzeniu czasu tworzenia widoków, dla każdego z testowanych urządzeń.

Na potrzeby badań wykonany został przegląd literatury. W artykułach [4, 5] porównywane były języki Swift oraz Objective-C pod względem wydajności. W pierwszym przypadku badana była szybkość wykonywania algorytmów sortowania i struktur danych. W drugiej pracy poddano analizie m.in. czas przejścia

pomiędzy widokami. W kolejnym artykule [6] poddano analizie antywzorce wydajności dotyczące aplikacji na system iOS, bez wyróżnienia zastosowanych szkieletów programistycznych. W artykule [7] dokonano przeglądu literatury, dotyczącego testowania aplikacji iOS, gdzie wyróżniono różne aspekty wpływające na wydajność testowanych urządzeń takie jak animacje, obsługa gestów czy zarządzanie danymi.

W obecnym czasie nie znaleziono artykułów naukowych poruszających tematykę porównawczą przedmiotowych szkieletów programistycznych, natomiast pomimo krótkiego czasu istnienia SwiftUI napisanych zostało wiele książek na jego temat.

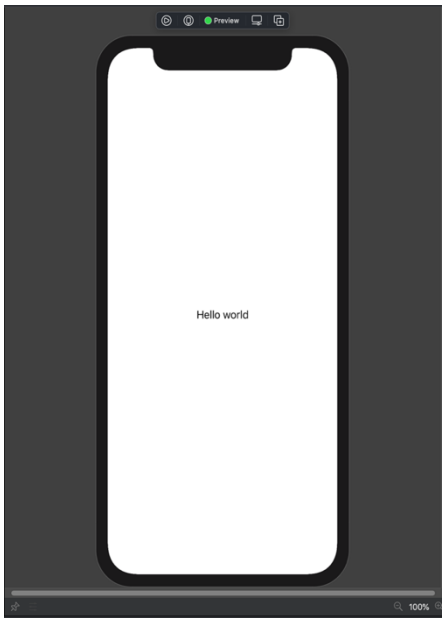
2. Implementacja aplikacji testowych

Na potrzeby badań wydajnościowych porównywanych szkieletów programistycznych zostały wykonane po dwie aplikacje o tej samej funkcjonalności, dla każdego z testowanych scenariuszy badawczych. Pierwsza z nich wykorzystuje UIKit w celach przedstawiania widoków aplikacji, a druga SwiftUI.

Projekt w SwiftUI wykorzystuje dostępny od wersji iOS 14.0 nowy cykl życia aplikacji składający się z protokołu App. Projekt zawiera 4 zaimplementowane widoki oraz początkowy, dodany przy utworzeniu projektu. Kod aplikacji w SwiftUI nie zawiera plików XML, interfejs jest definiowany tylko i wyłącznie w kodzie. Widoki są modyfikowalne w celu przeprowadzenia testów na konfigurowalnej liczbie elementów.

Projekt w UIKit wykorzystuje standardowy cykl życia aplikacji wykorzystujący AppDelegate, widoki definiowane są w większości przypadków w graficznym interfejsie. Komórki tabel i widoków kolekcji zawarte są w plikach o rozszerzeniu .xib. Graficzne reprezentacje posiadają swoje odpowiedniki w postaci klas dziedziczących po standardowych komponentach UIKit [8].

Pierwszym z testowanych widoków (rys. 1) jest pojedyncze nieedytowalne pole tekstowe na środku ekranu, jest to stan domyślnie utworzonego projektu w obydwu badanych szkieletach programistycznych.



Rysunek 1: Widok ekranu początkowego w teście aplikacji z pojedynczym widokiem.

Drugim z badanych widoków jest lista składająca się z komórek z pojedynczym polem tekstowym. Na rysunku 2 po lewej stronie znajduje się widok utworzony w UIKit, a po prawej SwiftUI. Różnice pomiędzy nimi są znikome.

Rysunek 2: Widok aplikacji wyświetlającej listę.

Kolejnym z zaimplementowanych widoków był widok pola tekstowego, który podlegał częstym zmianom w celu weryfikacji optymalnego mechanizmu rysowania widoków. Widok zaimplementowanych aplikacji przedstawiono na rysunku 3. Po lewej stronie znajduję się aplikacji wykorzystująca UIKit, a po prawej SwiftUI.

Rysunek 3: Widok aplikacji z jednym zmiennym polem tekstowym.

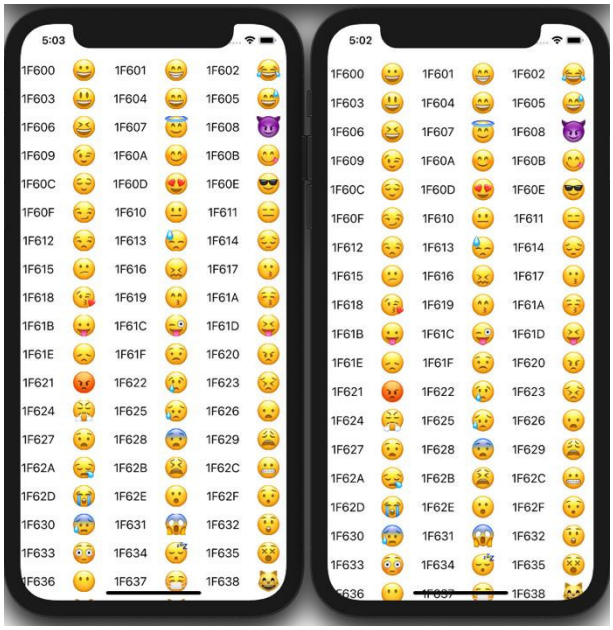
W ramach testów przygotowany został również wariant posiadający 40 pól tekstowych ułożonych jeden po drugim, przedstawiony na rysunku 4. Pola są również aktualizowane z częstotliwością wynoszącą 100Hz, więc zwiększona liczba obiektów na ekranie generuje więcej zasobów.

Rysunek 4: Widok z wieloma zmiennymi polami tekstowymi.

Następnie zaimplementowany został przykładowy widok pokazujący aktualnie wybrane miejsce na mapie. Składa się on z MapView, obrazka i pól tekstowych opisujących daną lokalizację. Wygląd został przedstawiony na rysunku 5, po lewej stronie znajduje się aplikacja wykorzystująca UIKit, a po prawej SwiftUI.

Rysunek 5: Widok z mapą.

Ostatnim z zaimplementowanych widoków jest kolekcja. W przypadku UIKit wykorzystywany jest gotowy komponent UICollectionView, który zarządza wyświetlaniem obiektów w różnorodnych układach kolumn i wierszy [9]. SwiftUI nie posiadał w czasie tworzenia pracy takiej funkcjonalności, szczególnie biorąc pod uwagę możliwości tworzenia zaawansowanych widoków. Zamiast tego, do utworzenia kolekcji zastosowano komponent ScrollView. Utworzone aplikacje przedstawiono na rysunku 6, po lewej stronie UIKit, a po prawej SwiftUI.



Rysunek 6: Widok kolekcji.

3. Metody wykonania pomiarów

Badania wydajnościowe porównywanych szkieletów programistycznych polegały na zmierzeniu czasu wygenerowania widoków i przejść między nimi. Czas został zmierzony wykorzystując wbudowaną w zintegrowane środowisko programistyczne Xcode funkcję measure [10], pozwalającą na określenie czasu wykonania bloku kodu. Metoda wykonywania pomiaru czasu startu aplikacji, wykorzystana przy przeprowadzaniu testów jest przedstawiona na listingu 1. Rezultatem uruchomienia funkcji są wartości czasów w sekundach, dla poszczególnych uruchomień testowych z zawartym błędem pomiaru.

Listing 1: Metoda wykorzystywana przy wykonywaniu pomiarów czasu start aplikacji

```
func testLaunchPerformance() throws {
    if #available(macOS 10.15, iOS 13.0, tvOS 13.0, *) {
        let options = XCTMeasureOptions()
        options.iterationCount = 20
        measure(metrics: [XCTApplicationLaunchMetric(
            waitUntilResponsive: true)], options: options) {
            XCUIApplication().launch()
        }
    }
}
```

W celu przeprowadzenia badań wydajnościowych wykorzystane zostało również wbudowane w Xcode narzędzie Instruments pozwalające na monitorowanie stanu aplikacji, umożliwiając podgląd użycia procesora czy czasu wykonania poszczególnych metod [10].

Testy przeprowadzono na następujących urządzeniach:

- iPhone 12 Mini (wersja systemu iOS 14.5),
- iPad Pro 11 2020 (wersja systemu iPadOS 14.5).

Przed rozpoczęciem testów urządzenia były uruchamiane ponownie w stanie pełnego naładowania baterii, bez aplikacji działających w tle, z wyłączonym Internetem. Odstęp czasowy pomiędzy każdą z serii testów wynosił 15 minut.

Testy uruchomieniowe przeprowadzone zostały dla pięciu różnych widoków dla każdego z testowanych urządzeń:

- widok początkowy,
- widok listy,
- widok wielu pól tekstowych,
- widok mapy, obrazka i listy,
- widok kolekcji.

W badaniach wykonane zostało po 20 prób dla każdego testowanego wariantu.

4. Wyniki badań

Badania wydajnościowe przeprowadzone dla pięciu różnych widoków, na dwóch testowanych urządzeniach przedstawione zostały w tabelach 1 - 10.

Widok początkowy

Pierwszym z badanych przypadków był widok początkowy.

Tabela 1: Średni czas uruchomienia aplikacji z widokiem początkowym na urządzeniu iPad Pro 11 2020

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**1,348 ± 2,67**
UIKit	**1,304 ± 2,12**

Tabela 2: Średni czas uruchomienia aplikacji z widokiem początkowym na urządzeniu iPhone 12 mini

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**0.965 ± 2,01**
UIKit	**0,816 ± 1,91**

Widok listy

Następnym spośród wykonywanych testów było sprawdzenie widoku listy.

Tabela 3: Średni czas uruchomienia aplikacji z widokiem listy na urządzeniu iPad Pro 11 2020

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**1,358 ± 2,67**
UIKit	**1,338 ± 2,07**

Tabela 4: Średni czas uruchomienia aplikacji z widokiem listy na urządzeniu iPhone 12 mini

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**0,986 ± 0,31**
UIKit	**1,006 ± 1,71**

Widok wielu pól tekstowych

Kolejnym spośród wykonywanych testów było sprawdzenie przypadku, w którym obiekt widoku jest aktualizowany często w krótkim czasie. W celu zmierzenia parametrów wykorzystane zostało narzędzie Profiler, które przedstawia wiele parametrów związanych z uruchamianą aplikacją.

Tabela 5: Średni czas uruchomienia aplikacji z widokiem wielu pól tekstowych na urządzeniu iPad Pro 11 2020

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**1,347 ± 4,34**
UIKit	**1,281 ± 0,66**

Tabela 6: Średni czas uruchomienia aplikacji z widokiem wielu pól tekstowych na urządzeniu iPhone 12 mini

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**1.031 ± 1,98**
UIKit	**1,021 ± 3,33**

Widok mapy, obrazka i listy

Kolejnym z przeprowadzonych testów było utworzenie przykładowego widoku składającego się z widoku mapy, obrazka i listy.

Tabela 7: Średni czas uruchomienia aplikacji z widokiem mapy na urządzeniu iPad Pro 11 2020

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**2,015 ± 2,71**
UIKit	**1,983 ± 2,01**

Tabela 8: Średni czas uruchomienia aplikacji z widokiem mapy na urządzeniu iPhone 12 mini

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**1,206 ± 4,33**
UIKit	**1,118 ± 0,42**

Widok kolekcji

Następnie przeprowadzony został test widoku kolekcji, składającego się z układu sześciu kolumn z emotikonami i odpowiadającym im kodem Unicode.

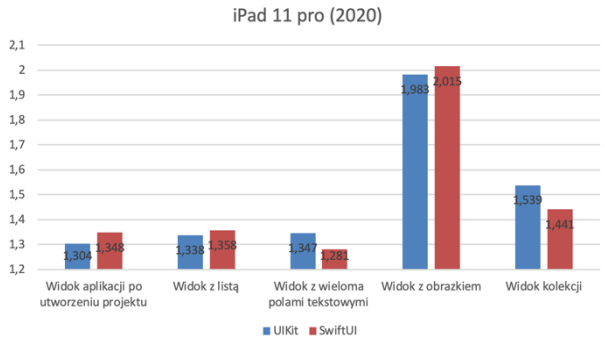
Tabela 9: Średni czas uruchomienia aplikacji z widokiem kolekcji na urządzeniu iPad Pro 11 2020

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**1,441 ± 2,24**
UIKit	**1,539 ± 2,03**

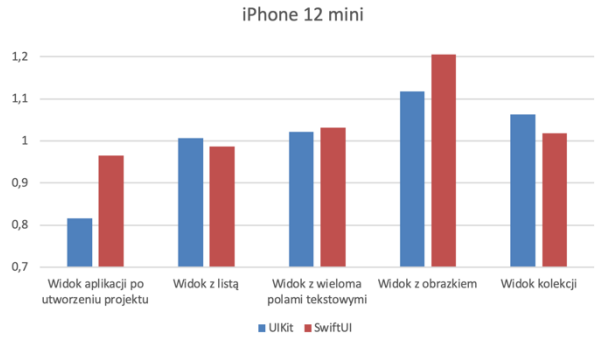
Tabela 10: Średni czas uruchomienia aplikacji z widokiem kolekcji na urządzeniu iPhone 12 mini

Liczba prób: 20	Średni czas uruchomienia [s] ± odchylenie standardowe
SwiftUI	**1,018 ± 1,28**
UIKit	**1,063 ± 0,80**

Zestawienie wyników badań wydajnościowych testowanych szkieletów programistycznych dla poszczególnych widoków zostało przedstawione na rysunkach 7 i 8.



Rysunek 7: Testy wydajnościowe badanych widoków dla urządzenia iPad 11 Pro 2020.



Rysunek 8: Testy wydajnościowe badanych widoków dla urządzenia iPhone 12 mini.

5. Wnioski

W artykule wykonana została analiza porównawcza przedmiotowych technologii: UIKit i SwiftUI. Na potrzeby badań stworzono aplikacje testowe dla obydwu szkieletów programistycznych. Wykonane badania wydajnościowe obejmowały zmierzenie czasów uruchomienia aplikacji składających się z różnych ekranów. Przetestowany również został przypadek skrajny, w którym obydwa szkielety programistyczne mogą powodować inne, znacznie różniące się wyniki spowodowane różnicami w mechanizmie rysowania widoków i ich reagowania na zmiany przy częstym przeładowywaniu widoku.

W przypadku badań na urządzeniu iPhone 12 mini największą różnicę czasu uruchomienia aplikacji widać w przypadku widoku początkowego na korzyść UIKit. Natomiast w przypadku testów na urządzeniu iPad 11 Pro, największe różnice w czasach uruchomienia widać pomiędzy widokami z wieloma polami tekstowymi oraz widokiem kolekcji, przy czym w pierwszym przypadku dla SwiftUI, pomiary są bardziej rozproszone względem średniej.

Przeprowadzone testy wydajnościowe nie wykazały znaczących różnic pomiędzy badanymi szkieletami programistycznymi. Na podstawie badań można stwierdzić, że są one równie wydajne w przypadku czasu generowania widoków. W porównaniu do UIKit, SwiftUI jest szybszy i bardziej przystępny co do możliwości generowania widoków w przypadku podstawowych aplikacji (tabela 1 - 10, rysunek 7 - 8). Na podstawie otrzymanych wyników, można stwierdzić, że postawiona teza: „SwiftUI jest bardziej wydajny czasowo dla aplikacji obsługujących kolekcję danych i wiele pól tekstowych" została udowodniona.

Pomimo wielu korzyści SwiftUI nie jest na obecny moment alternatywą dla UIKit, zwłaszcza przy bardziej rozbudowanych programach. Jest on narzędziem, które wymaga dalszego udoskonalania i musi zostać wzbogacony o wiele dodatkowych komponentów. Oprócz tego, SwiftUI nie jest jeszcze tak obszernie udokumentowany jak UIKit. Jego gotowość na ten moment jest jeszcze niewystarczająco atrakcyjna w przypadku programów produkcyjnych, aczkolwiek do zadań hobbystycznych doskonale sprawdzi się w aplikacjach o mniejszym stopniu złożoności. Do osiągnięcia statusu aplikacji gotowej do użytku SwiftUI musi zostać jeszcze zaktualizowany i ulepszony w wielu aspektach.

Literatura

[1] App Revenue Data (2021), https://www.businessofapps.com/data/app-revenues/, [02.05.2021].

[2] B. Cahill, UI Design for iOS App Development: Using SwiftUI, Apress, 2021.

[3] J. deVilla, E. Ganim, M. Hollemans, iOS Apprentice (Eighth Edition): Beginning iOS Development with Swift and UIKit, Razeware LLC, 2019.

[4] K. Gut, M. Skublewska-Paszkowska, E. Łukasik, J. Smołka, Comparison of programming languages on the iOS platform in terms of performance, IAPGOŚ 7(3) (2017) 33-36, https://doi.org/10.5604/01.3001.0010.5211.

[5] K. Banach, M. Skublewska-Paszkowska, Comparison of Objective-C and Swift on the example of a mobile game, Journal of Computer Sciences Institute 16 (2020) 305-308, https://doi.org/10.35784/jcsi.2058.

[6] S. S. Afjehei, T. P. Chen, N. Tsantalis, iPerfDetector: Characterizing and detecting performance anti-patterns in iOS applications, Empirical Software Engineering 24 (2019) 3484-3513, https://doi.org/10.1007/s10664-019-09703-y.

[7] I. Kulesovs, iOS Applications Testing, Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference 3 (2015) 138-150, https://doi.org/10.17770/etr2015vol3.187.

[8] UIKit, https://developer.apple.com/documentation/uikit, [02.05.2021].

[9] F. Farook, M. Hollemans, UIKit Apprentice, Razeware LLC, 2020.

[10] Xcode, https://developer.apple.com/xcode/, [02.05.2021].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 175–182
Received: 05 June 2021
Accepted: 21 June 2021

Comparison of selected view creation technologies in applications using the Laravel framework

Porównanie wybranych technologii tworzenia widoków w aplikacjach wykorzystujących szkielet programistyczny Laravel

Albert Wiktor Woś*, Beata Pańczyk

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

Abstract
This article presents the result of a comparison of selected view creation technologies in applications using the Laravel framework. Using the multi-criteria analytic hierarchy process method the most efficient template system was selected from Blade, Twig, and Smarty. Presented template systems were used with Laravel framework to create a landing page on the basis of which the experiment was conducted.

Keywords: template engines; Laravel; comparative analysis; AHP method

Streszczenie
W niniejszym artykule przedstawiono rezultat porównania wybranych technologii widoków w aplikacjach wykorzystujących szkielet programistyczny Laravel. Za pomocą wielokryterialnej metody analitycznego procesu hierarchicznego wyłoniony został najbardziej wydajny system szablonów spośród Blade, Twig oraz Smarty. Przedstawione silniki szablonów wykorzystane zostały razem z platformą programistyczną Laravel w celu utworzenia stron typu "landing page", na podstawie których przeprowadzono eksperyment.

Słowa kluczowe: systemy szablonów; Laravel; analiza porównawcza; metoda AHP

*Corresponding author
Email address: **albert.wiktor.wos@gmail.com** (A. W. Woś)

1. Wstęp

Istnieje wiele metod tworzenia aplikacji internetowych. Z każdym dniem powstają nowe technologie ułatwiające ten proces. Innowacje te są odpowiedzią na coraz większe zapotrzebowanie, by zaistnieć w sieci i poszerzyć swój zasięg na nowe grupy odbiorców. Z tego powodu coraz większą popularnością cieszą się szkielety programistyczne, które pozwalają programistom tworzyć rozbudowane strony szybciej oraz w bezpieczniejszy sposób, tym samym dewaluują praktykę pisania stron internetowych jedynie za pomocą własnego kodu.

Jednym ze szkieletów programistycznych, ułatwiających pracę z tworzeniem aplikacji internetowych jest Laravel. Framework ten napisany został w języku PHP, który posłużył między innymi do napisania platform programistycznych Symfony czy CakePHP, systemu zarządzania treścią Wordpress, a także najbardziej popularnego serwisu społecznościowego, jakim jest Facebook. Laravel jako jeden z najbardziej popularnych szkieletów programistycznych, bazujących na architekturze MVC, dostarcza autorski system szablonów Blade. Pozwala on, podobnie jak inne znane silniki szablonów Twig oraz Smarty, na separację logiki aplikacji od warstwy widoku i na ograniczenie redundancji kodu dzięki ponownemu wykorzystaniu go w innych widokach, w których wygląd posiada elementy wspólne.

1.1. Przegląd literatury

W celu uzasadnienia wyboru technologii, kryteriów oraz metodyki badań dokonano przeglądu literatury. Wiele prac dotyczy wyodrębnienia odpowiedniej technologii na podstawie różnych kryteriów. Najczęściej są to: popularność, wydajność oraz uniwersalność. Przykładem takiej pracy jest artykuł „Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems" [1]. Zamieszczono w nim zestawienie języków programowania działających po stronie serwera, gdzie najwyższy wynik osiągnął język PHP, który jest wykorzystywany w przypadku 87.5% aplikacji internetowych, co argumentuje wybór tego języka do badań.

Kolejne zestawienie popularności użycia technologii przedstawiono w artykule na stronie „morioh.com" [2]. Na podstawie trendów wyszukiwań haseł, tagów oraz wzmianek na innych stronach wskazano platformę programistyczną Laravel jako tę najbardziej popularną spośród innych platform programistycznych wykorzystujących język PHP.

Natomiast w publikacji „Pro PHP MVC" [3] opisano zalety architektury Model-Widok-Kontroler, a także jako przykład najpopularniejszych systemów szablonów wskazano Twig oraz Smarty. Systemy te również pojawiają się najczęściej w zestawieniu zaprezentowanym na stronie „socialcompare.com" [4], gdzie przedstawiono dane dotyczące najbardziej popularnych szkieletów programistycznych napisanych w języku PHP.

Potwierdza to popularność tych silników szablonów oraz ich uniwersalność, a także motywuje zasadność porównania ich z systemem Blade dostarczanym przez Laravel.

W celu porównania technologii niezbędny jest wybór kryteriów, które staną się wyznacznikiem danego procesu. Ich wyodrębnienia dokonano po przeanalizowaniu wyników badań i argumentacji zaprezentowanej w literaturze przedmiotu.

Przykładem pracy porównującej technologie jest artykuł „RWD jako narzędzie optymalizacji stron internetowych" [5]. Zaprezentowano w nim badania czynników mających wpływ na wydajność projektów responsywnych. W tym przypadku jednym z kryteriów był czas ładowania oraz rozmiar strony.

W artykule „Analiza porównawcza wydajności frameworków Angular oraz Vue.js" [6] porównano szkielety programistyczne, wykorzystujące Javascript Angular i Vue.js za pomocą kryteriów dotyczących czasów renderowania, wymiany danych, poszczególnych elementów działania badanych aplikacji, a także rozmiaru plików końcowych.

Oprócz ustalenia kryteriów niezbędny jest także wybór metody zastosowanej do porównania. Przykład tejże znajduje się w publikacjach „Analiza wydajnościowa i możliwościowa narzędzia Laravel do tworzenia nowoczesnych aplikacji webowych" [7], a także „Zbadanie wydajności aplikacji internetowych utworzonych z wykorzystaniem Spring MVC oraz JavaServer Faces" [8]. W obu pracach badane platformy programistyczne oraz technologie wykorzystane zostały do utworzenia aplikacji o analogicznych funkcjonalnościach, które następnie posłużyły do testów wydajności. Dodatkowo w pracy porównującej technologie Spring MVC oraz JavaServerFaces testy wykonane zostały dzięki narzędziu JMeter, który pozwolił na przetestowanie aplikacji na serwerze lokalnym oraz eksport wyników.

Kolejnym etapem po przeprowadzeniu testów jest zestawienie wyników oraz ich porównanie, a następnie przedstawienie z wykorzystaniem odpowiedniej metody. Ponadto w przypadku wielu kryteriów niezbędna jest ich optymalizacja. W artykule „Comparison the processing speed between PHP and ASP.NET" [9] porównano wydajność szybkości przetwarzania dla języka PHP oraz ASP.NET. Każdej z technologii przyznawano punkty dla poszczególnych kryteriów. Dodatkowo wyniki badań przedstawiono w postaci wykresów i w tabelach, gdzie zestawiono punkty oraz technologię, która je uzyskała. Następnie punkty dla każdego z języków zsumowano i wyłoniono ten najlepszy.

Podobny w założeniu, choć bardziej miarodajny sposób porównania danych został przedstawiony w pozycji „Analiza jakości wybranych systemów e-learningowych za pomocą wielokryterialnej metody analitycznego procesu decyzyjnego AHP" [10]. W publikacji tej zaprezentowano badania dotyczące jakości systemów MOODLE oraz LAMS. Do analizy tych systemów wykorzystano metodę analitycznego procesu hierarchicznego – AHP. Metoda ta pozwoliła autorowi ustalić zależności między poszczególnymi kryteriami, które zostały zestawione, a wynikami dla porównywanych systemów. Dodatkowo w publikacji tej obliczono współczynnik niezgodności CR, który w tej metodzie pozwala określić spójność wag każdego z kryterium względem pozostałych. Wyznaczone na podstawie dziewięciostopniowej skali, przedstawionej w tej pozycji, wagi oraz eksperyment pozwoliły na obliczenie wyników dla poszczególnych systemów. Wyniki te dodano dla każdego z przedmiotów badań i wskazano system e-learningowy o największej sumie punktów.

Przedstawione w przeglądzie literatury pozycje uzasadniają wybór technologii, jakimi są Laravel oraz systemy szablonów Blade, Twig oraz Smarty, a także wybór metodyki badań i sposobu analizy wyników metodą AHP.

1.2. Cel badań

Celem badań było porównanie wybranych systemów szablonów umożliwiających tworzenie widoków w aplikacjach internetowych wykorzystujących szkielet programistyczny Laravel. W tym celu na podstawie przeglądu literatury zostały wybrane trzy systemy szablonów: Blade, Twig oraz Smarty. Spośród tych technologii, z wykorzystaniem trzech kryteriów: „objętość kodu wynikowego", „objętość kodu, który twórca musi napisać, by uzyskać dany efekt" oraz „szybkość ładowania strony", wyznaczony został najbardziej wydajny silnik szablonów. W ten sposób uzyskano odpowiedź na pytanie badawcze: „Blade, Twig czy Smarty – który z wymienionych systemów szablonów jest najbardziej wydajny do tworzenia widoków w aplikacjach typu »strona lądowania«, wykorzystujących platformę programistyczną Laravel?".

1.3. Zakres badań

Zakres badań objął utworzenie trzech aplikacji za pomocą szkieletu programistycznego Laravel. Każda z aplikacji wykorzystywała inny z systemów szablonów pozwalających na wygenerowanie widoku i odpowiednich im kontrolerów umożliwiających zwracanie widoku wraz z danymi. Wygląd aplikacji w każdym przypadku był identyczny i obejmował tę samą funkcjonalność. W ramach badań przeprowadzono testy każdej z aplikacji, a także ankietę pozwalającą na wyznaczenie wag kryteriów.

2. Obszar badań

Przed rozpoczęciem badań rozpoznano badane technologie i odpowiednio zaplanowano eksperyment.

2.1. Badane technologie

Przedmiot badań stanowią systemy szablonów Blade, Twig oraz Smarty, które, podobnie jak Laravel, zostały napisane w języku PHP. Systemy te pozwalają użytkownikowi na oddzielenie warstwy logiki biznesowej od logiki prezentacji. Umożliwia to łatwiejsze zarządzanie aplikacją oraz organizację kodu. Dodatkowo odgraniczenie poszczególnych warstw projektu pozwala na ponowne wykorzystanie kodu, który odpowiada za widok aplikacji dzięki powtórnemu użyciu tego samego wyglądu razem z inną treścią, dostarczaną z innej warstwy. Systemy szablonów umożliwiają ten proces poprzez wykorzystanie warunków oraz pętli, a także dzięki możliwości rozszerzania danego widoku przez dołą-

czenie wcześniej utworzonych bloków widoków czy całych plików. Kolejną zaletą silników szablonów jest umożliwienie użytkownikowi wyświetlania różnego rodzaju dostarczanych przez model danych, które dodatkowo mogą być wykorzystane lub nie na podstawie zaimplementowanych wewnątrz widoku warunków. Zasada działania systemów szablonów opiera się na tokenizacji danych źródłowych, które następnie segmentowane, zostają uporządkowane. Następstwem tego jest przekonwertowanie tych danych do kodu wynikowego dzięki parserom – programom analizującym składnię oraz ją interpretującym na podstawie wcześniej określonej gramatyki.

System szablonów Blade jest silnikiem dostarczonym przez twórców szkieletu programistycznego Laravel. Wykorzystuje on dyrektywy oraz tagi, których użycie przyśpiesza pracę z widokiem aplikacji. Blade kompiluje szablony do kodu PHP, który przechowywany jest w pamięci podręcznej do momentu, gdy zostanie ponownie zmodyfikowany [11]. W badaniach wykorzystano system Blade w wersji 7.0.

Kolejnym badanym silnikiem szablonów jest Twig, który jest domyślnym systemem szablonów dla Symfony, jednakże możliwe jest użycie go w innych platformach programistycznych. Autorzy tego silnika wymienili szybkość jako jedną z cech tej technologii, ponieważ szablon kompilowany jest do zoptymalizowanego kodu PHP. Kolejną cechą wymienioną przez twórców silnika Twig jest bezpieczeństwo, zapewnione użytkownikom poprzez tryb piaskownicy, a także elastyczność dzięki możliwości wykorzystania parserów oraz lekserów, które pozwalają użytkownikowi tworzyć własne filtry oraz flagi. W badaniach użyty został Twig w wersji 3.2.1 [12].

Ostatnim z badanych systemów szablonów jest Smarty, który wydany został w 2002 roku i dalej jest rozwijany przez twórców. Podobnie jak pozostałe systemy kompiluje kopię szablonów do skryptu PHP. Smarty został opisany przez twórców jako szablon cechujący się szybkością, łatwością użycia a także elastycznością i bezpieczeństwem. W eksperymencie wykorzystany został Smarty w wersji 3.1.36 [13].

Wszystkie wymienione systemy szablonów zaimplementowane zostały w szkielecie programistycznym Laravel. Framework ten napisany został w języku PHP i wydany na licencji MIT. Bazuje on na wzorcu architektonicznym Model-Widok-Kontroler (MVC), co pozwala użytkownikom na tworzenie aplikacji internetowych w bardziej zorganizowany sposób dzięki oddzieleniu warstw aplikacji.

Dodatkowo szkielet programistyczny Laravel ułatwia tworzenie bardziej rozbudowanych aplikacji internetowych dzięki wbudowanym komponentom, takim jak system autentykacji, linia komend Artisan, a także Eloquent Model, który wspomaga projektowanie bazy danych wykorzystywanej w aplikacji. W badaniu wykorzystano Laravel w wersji 7.0 [11].

Opisane systemy szablonów charakteryzują się podobnymi cechami oraz podobną metodą działania, jednakże zauważalnie różnią się pod względem składni, której przykłady dla poszczególnych systemów przedstawiono w Tabela 1.

Tabela 1: Przykładowe zestawienie elementów składni systemów szablonów Blade, Twig i Smarty

	Blade	Twig	Smarty
Dołączenie pliku o rozszerzeniu html:	@include('lorem')	{{include('lorem') }}	{include file = "lorem.html"}
Pętla for	@for ($i=0; $i<10;$i++) current value: {{ $i }} @endfor	{% for i in 0..10 %} current value: {{ i }} {% endfor %}	{for $i=1 to 10} current value: {$i} {/for}
Warunek if	@if (1< 18) @endif	{% if 1 < 18 %} {%endif%}	{if (1< 18)} {/if}
Komentarz	{{-- comment --}}	{# comment #}	{* comment *}
Rozszerzenie pliku layout na przykładzie sekcji o nazwie „body”	@yield ('body')	{% block body %} {% endblock %}	{block name = "body"} {/block}
Oznaczenie, że dany plik rozszerza plik o nazwie „layout”	@extends ('layout')	{%extends "layout"%}	{extends 'layout.tpl'}
Oznaczenie sekcji o nazwie „body”	@section ('body') @endsection	{% block body %} {% endblock %}	{block name = "body"} {/block}
Użycie zmiennej o nazwie „var”	{{$var}}	{{ var }}	{$ var }

Na podstawie Tabela 1. widać, że składnia użycia systemu Blade charakteryzuje się wykorzystaniem symbolu „@”. W przypadku systemu Smarty najczęściej wykorzystanym znacznikiem są nawiasy klamrowe, natomiast dla Twig jest to dodatkowo znak „%”. Kolejną widoczną różnicą są rozszerzenia widoków dla poszczególnych systemów. Dla Blade jest to „blade.php”, dla Twig - „twig”, natomiast dla Smarty - „tpl”.

2.2. Kryteria badań

Wybór kryteriów został uargumentowany w przeglądzie literatury i przedstawia się następująco:

- najkrótszy czas ładowania;
- najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści;
- najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści.

Czas ładowania aplikacji określa czas w sekundach, jaki upływa od momentu zainicjowania połączenia do

otrzymania odpowiedzi i pełnego obsłużenia żądania przez serwer.

Rozmiar kodu jest parametrem przedstawiającym liczbę użytych do napisania kodu znaków, które wykorzystują dany system szablonów oraz związane z nim elementy, takie jak dołączenie wymaganych bibliotek. Podczas zliczania znaków wzięto pod uwagę kontroler zwracający widok, a także pliki „layout" oraz „welcome", które zawierały składnię wykorzystywaną przez dany silnik w widokach oraz kod HTML użyty do utworzenia struktury strony.

Rozmiar plików określa liczbę bajtów pobranych przez użytkownika w celu wyświetlenia widoku dostarczonego przez aplikację. Parametr ten wskazuje, jak dobrze zoptymalizowany został kod wykorzystany do utworzenia widoku dla poszczególnych systemów szablonów. Dodatkowo wartość ta przekłada się na czas renderowania treści przez przeglądarkę, ale także jakość konwersji plików szablonów użytych przez badaną technologię.

2.3. Procedura przygotowania badań i ich realizacji

Przed przystąpieniem do eksperymentu przygotowane zostało środowisko badawcze. W celu utworzenia aplikacji użyto PhpStorm w wersji 2019.3.2 – zintegrowanego środowiska programistycznego wydanego przez firmę JetBrains, natomiast do zasymulowania działania serwera wykorzystano pakiet serwera XAMPP w wersji 2.4.46.

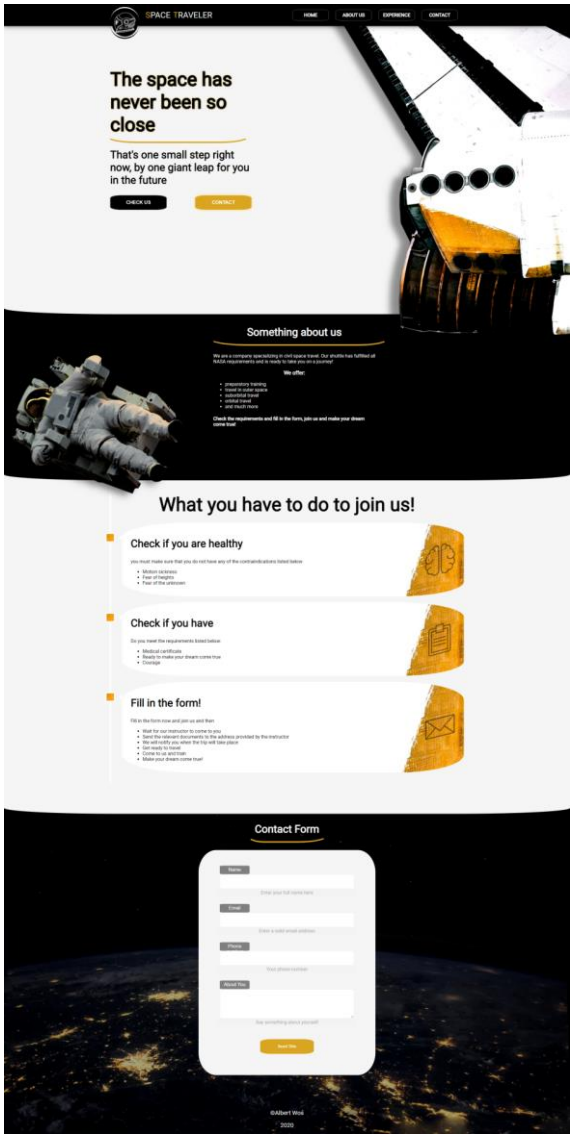
W celu instalacji szkieletu programistycznego Laravel oraz dodatkowo systemów szablonów Twig oraz Smarty wykorzystano system zarządzania bibliotekami - Composer w wersji 2.0, dzięki któremu możliwe było wydanie następujących komend:

- utworzenie projektu wykorzystującego framework Laravel - "composer create-project laravel/laravel nazwa-projektu 7.0.0";
- zainstalowanie systemu szablonów Twig – "composer require "twig/twig:^3.0";
- zainstalowanie systemu szablonów Smarty – „composer require smarty/smarty".

Dodatkowo zwiększony został limit pamięci w pliku php.ini do 2048 MB.

Do badań przygotowano także widok aplikacji typu „landing page" (strona lądowania). Widok ten został następnie wykorzystany w testowanych aplikacjach. Funkcją tego typu strony jest przedstawienie informacji na temat danego produktu czy usługi, a także przechwycenie uwagi użytkownika odwiedzającego stronę [14]. Dodatkowo strona lądowania charakteryzuje się węższym zakresem funkcjonalności niż strona główna, by zminimalizować możliwe rozproszenie uwagi użytkownika innymi opcjami dostępnymi na stronie [15]. Kolejnym charakterystycznym aspektem strony docelowej jest zwięzły tekst oraz treści strony uzupełnione obiektami graficznymi skompresowanymi i zapisanymi w odpowiednich formatach, a także odpowiednie wykorzystanie znaczników HTML.

Elementy charakterystyczne dla strony typu „landing page" zostały wykorzystane w utworzonych do badań aplikacjach, posiadających ten sam wygląd i te same funkcjonalności (Rysunek 1).



Rysunek 1: Wygenerowany widok aplikacji.

Do utworzenia aplikacji i skonfigurowania środowiska badawczego oraz testowania wykorzystano komputer z zainstalowanym systemem operacyjnym Windows 10 Home Edition o następującej specyfikacji:

- procesor: Intel Core i5-9300H, 2.40 GHz;
- pamięć RAM: 16 GB;
- karta graficzna: NVIDIA GeForce GTX 1650;
- dysk SSD: The Western Digital PC SN730.

Testy natomiast przygotowano za pomocą aplikacji Apache JMeter w wersji 5.3. Aplikacja ta napisana została w języku JAVA i wydana przez firmę Apache na licencji otwartego oprogramowania [16]. JMeter służy w głównej mierze do testowania aplikacji korzystających z serwerów HTTP i FTP oraz mierzenia parametrów, takich jak:

- przepustowość,
- czas odpowiedzi,
- czas przeprowadzanego testu,
- liczba żądań oraz bajtów, które serwer obsługuje.

Aplikacja ta pozwala również tworzyć testy, które ustawiają grupy wątków. Grupy te mogą zawierać opcje uwzględniające metodę obsługi wyjątku w razie niepowodzenia testu, liczbę wątków - liczbę przeprowadzonych testów w jednym momencie, a także, ile razy należy powtórzyć dany test. Ponadto Apache JMeter umożliwia skonfigurowanie żądań, które określają cel testów. Aplikacja ta dzięki słuchaczom może również przetworzyć wysyłane żądania oraz wyświetlić wyniki testów, które następnie można wyeksportować.

Do badań w aplikacji utworzono test, w którym następnie dodano grupy wątków dla każdego z badanych systemów szablonów. Dla każdej grupy wykonano 100 powtórzeń przy pojedynczym wątku. Dodatkowo do każdej z grup dodano próbnik „HTTP Request" z wartością dla pola o tej samej nazwie równą „GET" i nazwą serwera w polu „Server Name or IP" – „localhost". Dla każdego z próbników pola „Path" różniły się i zostały uzupełnione wartościami przedstawionymi w Tabela 2.

Tabela 2: Grupy wątków wraz z konfiguracją próbnika

	Path
Blade	comparison/comparison-blade/public/
Twig	comparison/comparison-twig/public//
Smarty	comparison/comparison-smarty/public/

Do testu dodany został także słuchacz „View Results Tree", który umożliwił obserwację wyników oraz ich eksport do pliku CSV. Ostatnim krokiem konfiguracji było zapisanie planu testu do pliku o rozszerzeniu jmx. Plik taki można następnie wczytać do programu, by wykonać tak skonfigurowane testy.

2.4. Scenariusz badań

Konfiguracja środowiska badawczego umożliwiła przeprowadzenie badań, których poszczególne etapy wykonane zostały w następującej kolejności:

1. Uruchomienie pakietu serwera WWW XAMP razem z modułem Apache;
2. Konfiguracja wirtualnych adresów badanych aplikacji;
3. Uruchomienie narzędzia JMeter;
4. Wczytanie skonfigurowanych wcześniej testów w narzędziu JMeter;
5. Podział wyników ze względu na badany system szablonów.

3. Metodyka badań

Podczas porównania przedmiotów badawczych wykorzystując więcej niż jedno kryterium, niezbędna jest ich optymalizacja. W tym celu wykorzystuje się metody optymalizacji wielokryterialnej. Umożliwiają one merytorycznie poprawne wyłonienie przedmiotu badań, który osiągnął najlepszy wynik. Jedną z tego typu metod jest metoda analitycznego procesu hierarchicznego opracowana przez Thomasa L. Saaty'ego [17]. Metoda ta uwzględnia następujące punkty krytyczne:

1. Przedstawienie problemu w postaci struktury hierarchicznej;
2. Ustalenie decydentów, którzy pomogą w ustaleniu wag kryteriów (do ustalenia wag kryteriów, które będą porównywane, przeprowadzone zostały badania ankietowe, opisane w rozdziale 3.1);
3. Porównanie parami w celu wyznaczenia priorytetów;
4. Obliczenie współczynnika niezgodności.

Następnie, wykorzystując metodę trywialną, wyznaczone zostało kryterium łączne na podstawie wyliczonych wcześniej wag.

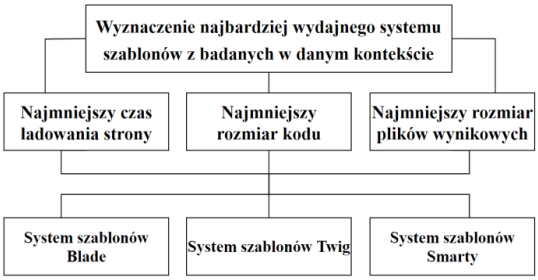
3.1. Ankieta

W celu określenia wag kryteriów wykorzystana została ankieta przygotowana na platformie Google Forms. Ankieta zatytułowana „Waga kryterium wyboru systemu szablonów do tworzenia widoków aplikacji internetowych" zawierała pytania dotyczące ankietowanego oraz porównań parami wag kryteriów. W celu przygotowania ankiety poprawnej merytorycznie wykorzystano wskazówki z artykułu „Design a questionnaire" [18]. Pytania zostały sformułowane w sposób jednoznaczny, a odpowiedzi wzajemnie się wykluczały. Pierwsze sześć pytań dotyczyło ankietowanego, jego wieku, płci, informacji, czy respondent jest studentem, czy też pracuje, najczęściej używanego przez ankietowanego języka programowania, posiadanego doświadczenia w branży IT oraz tego, czy respondent kiedykolwiek używał system szablonów. Ostatnie trzy pytania dotyczyły porównania parami dwóch kryteriów oraz przewagi jednego nad drugim w skali od -4 (przewaga kryterium Y nad X) do 4 (przewaga kryterium X nad Y).

Grupą docelową badania ankietowego byli programiści - pracownicy i studenci posiadający doświadczenie w branży IT oraz w korzystaniu z systemów szablonów. Z tego powodu w badaniach wzięte pod uwagę zostały jedynie te odpowiedzi, które zawierały odpowiedź twierdzącą na pytanie dotyczące doświadczenia w branży IT.

3.2. Struktura hierarchiczna

W badaniach wykorzystano trzy wcześniej opisane kryteria, które zostały usytuowane w strukturze hierarchicznej wraz z przedmiotami badań oraz celem badań (Rysunek 2).



Rysunek 2: Struktura hierarchiczna problemu badawczego.

3.3. Wyznaczenie priorytetów

Porównane zostały kryteria zestawione parami. Pozwoliło to uzyskać współczynnik wagowy, który określa priorytety wykorzystane do zoptymalizowania kryteriów. Do analizy tej użyta została dziewięciostopniowa skala zaproponowana przez Thomasa L. Saaty'ego [17],

której wartości zostały zestawione ze skalą użytą w ankiecie. Wyniki pochodzące z ankiety i przekształcone odpowiednio dzięki tej skali pozwoliły utworzyć macierz porównań o liczbie wierszy oraz kolumn równej liczbie badanych kryteriów. Pola w tej macierzy powyżej przekątnej uzupełnione zostały danymi dotyczącymi porównania kryteriów parami. Przekątną macierzy stanowią jedynki, które oznaczają zestawienie kryteriów odnoszących się do samych siebie. Dodatkową własnością tej macierzy jest to, że jest ona odwrotnie symetryczna. Oznacza to, że poniżej przekątnej pola uzupełnione zostały odwrotnościami wag priorytetów.

Tabela 3: Utworzona macierz porównań

	A	B	C
A	1	3	2
B	0,33	1	1
C	0,5	1	1

Następnie na podstawie macierzy porównań obliczone zostały priorytety wykorzystując wzory opisane w kolejnych trzech krokach:

1. Obliczono średnie geometryczne r_x dla każdego z trzech wierszy według wzoru:

$$r_x = \sqrt[n]{p_{x,y_1} * p_{x,y_2} * p_{x,y_3}} \quad (1)$$

gdzie *p* jest elementem macierzy porównań, natomiast indeksy *x* oraz *y* odpowiednio oznaczają numer wiersza i kolumny, a *n* liczbę porównywanych kryteriów;

2. Zsumowano obliczone w poprzednim kroku średnie geometryczne;
3. Wyznaczono priorytety w_x za pomocą następującego wzoru:

$$w_x = \frac{r_x}{\sum_{i=1}^{n} r_i}. \quad (2)$$

3.4. Współczynnik niezgodności

Spójność wyliczonych wag względem siebie przedstawiona została za pomocą współczynnika niezgodności *CR*. Thomas L. Saaty zasugerował, że, aby wagi były poprawne, wartość współczynnika *CR* nie może przekroczyć 10% [17].

W badaniach próg ten wyliczony został następująco:

1. Najpierw obliczono wartość własna λ_{max} według wzoru:

$$\lambda_{max} = \sum_{x=1}^{n} \sum_{y=1}^{n} p_{x,y} * w_x \quad (3)$$

gdzie *n* oznacza liczbę kryteriów, które są porównywane. Znaczenia symboli *p*, *w*, *x* oraz *y* wyjaśnione zostały w rozdziale 3.33.);

2. Następnie obliczony został indeks niezgodności *IC* za pomocą wzoru:

$$IC = \frac{\lambda_{max} - n}{n - 1} \quad (4)$$

3. Wykorzystując obliczony indeks niezgodności *IC*, a także wartość stablicowanego indeksu losowanego *RI*, podanego przez twórcę metody AHP (W przypadku 3 kryteriów *RI* wynosi 0,52), obliczono współczynnik niezgodności *CR* za pomocą wzoru:

$$CR = \frac{IC}{RI} \quad (5)$$

3.5. Kryterium łączne

Ostatnim krokiem niezbędnym do wyznaczenia wartości zoptymalizowanych i możliwych do porównania jest wykorzystanie metody trywialnej wyznaczania kryterium łącznego. W tym celu do obliczenia zastosowano następujący wzór:

$$K_i = \sum_{j=1}^{k} w_j \left(\frac{K_{min}}{K_{i,j}} \right) \quad (6)$$

gdzie symbolem K_i oznaczono wynik dla kryterium. Symbol w_j przedstawia wagę j-tego kryterium, K_{min} – najmniejszą wyznaczoną wartość dla danego kryterium, natomiast $K_{i,j}$ wyznaczoną wartość dla danej technologii i obliczanego kryterium.

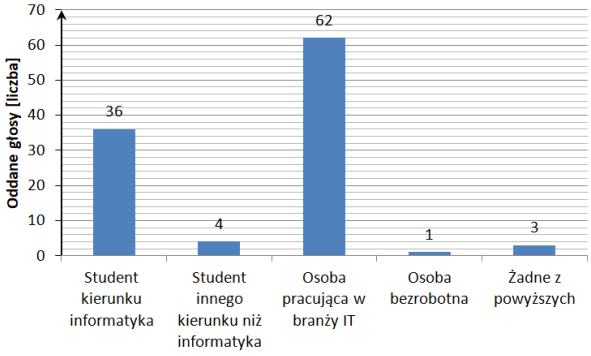
4. Wyniki

Rezultaty badań obejmują dane liczbowe wynikające z przeprowadzanego eksperymentu, a także odpowiedzi respondentów uzyskane z ankiety przeprowadzonej w dniach 22.02.2021 - 22.03.2021 r.

4.1. Ankieta

W ankiecie opisanej w rozdziale 3.11. uzyskano 113 odpowiedzi, z czego 85 zakwalifikowało się do badań ze względu na twierdzącą odpowiedź na pytanie dotyczące posiadania doświadczenia w branży IT. Większość ankietowanych - 84.71% zadeklarowała odpowiedź „20 – 40 lat”. Wiek pomiędzy 41 a 60 lat wskazało 12.94% respondentów. Jedynie 2.35% ankietowanych zaznaczyło odpowiedź „poniżej 20 lat”. Żaden z respondentów nie wybrał odpowiedzi „powyżej 60 lat”.

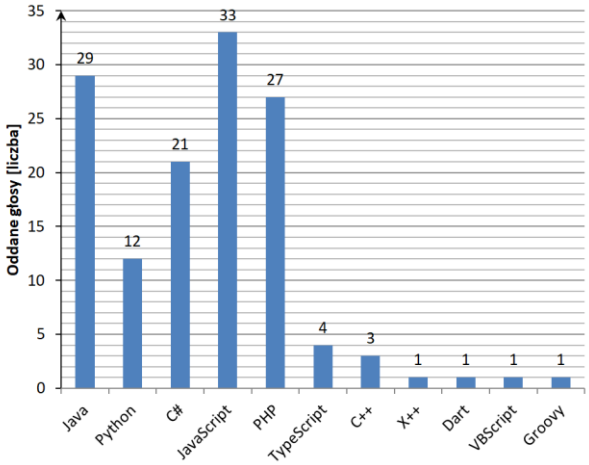
Odpowiedzi ankietowanych na pytanie „Wybierz opcję/opcje najlepiej Ciebie opisującą/opisujące.” przedstawione zostały na wykresie (Rysunek 3).



Rysunek 3: Rozkład odpowiedzi dotyczących związku ankietowanego z informatyką.

Odpowiedzi na pytanie dotyczące najczęściej wykorzystywanego języka przez respondenta przedstawiono na wykresie (Rysunek 4). Widać, że najczęściej wybie-

ranymi językami są JavaScript z 33, Java z 29 oraz PHP z 27 odpowiedziami.



Rysunek 4: Wybór języka najczęściej używanego przez respondenta.

Głównym celem ankiety było zebranie odpowiedzi dotyczących porównań parami wag kryteriów. Kryterium „A" oznaczało „najkrótszy czas ładowania strony", „B" – „najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści", natomiast kryterium „C" – „najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści". W pytaniu dotyczącym porównania kryterium „A" z „B" średnia z odpowiedzi wyniosła -0.96, co oznacza niewielką przewagę kryterium „A" nad „B". W porównaniu kryterium „A" z „C" średnia ta wyniosła -0.5. Oznacza to przewagę kryterium „A" nad „C". W przypadku zestawienia kryterium „B" z „C" średnia z odpowiedzi wyniosła 0,05 i oznacza to minimalną przewagę kryterium „C" nad „B".

4.2. Wagi kryteriów

Na podstawie wyników przeprowadzonej ankiety możliwe było wyznaczenie docelowych priorytetów dla każdego z kryteriów. Dodatkowo w tym celu przeprowadzono konwersję odpowiedzi respondentów do dziewięciostopniowej skali zaproponowanej przez twórcę metody AHP. Wynikiem tej operacji są dane w Tabela 4., przedstawiającej wagi kryteriów zestawionych parami. Następnie, na podstawie metodyki opisanej w rozdziale 3.3. oraz wag przedstawionych w tejże tabeli, obliczone zostały priorytety dla poszczególnych kryteriów. Wynik tego działania został przedstawiony w Tabela 5.

Tabela 4: Tabela wag kryteriów zestawionych parami

Para	Wynik
Przewaga ważności kryterium A nad kryterium B	3
Przewaga ważności kryterium A nad kryterium C	2
Przewaga ważności kryterium B nad kryterium C	1

Wyliczone priorytety umożliwiły obliczenie współczynnika niezgodności *CR*. W procesie tym wykorzystano metodykę przedstawioną w rozdziale 3.4. Obliczony *CR* wyniósł 1,34 %. Wynik ten nie przekroczył zaproponowanego przez twórcę progu - zatem obliczone wagi są poprawne.

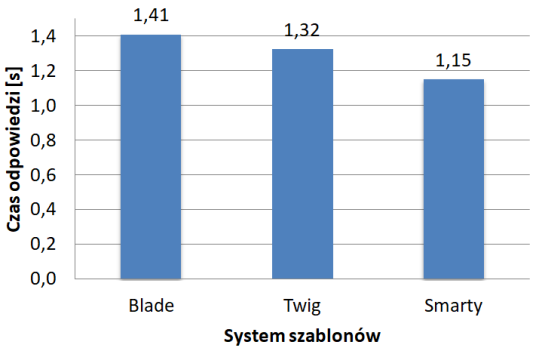
Tabela 5: Zestawienie kryteriów z ich priorytetami

Kryterium	Priorytet
Najkrótszy czas ładowania strony	0,56
Najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści	0,20
Najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści	0,24

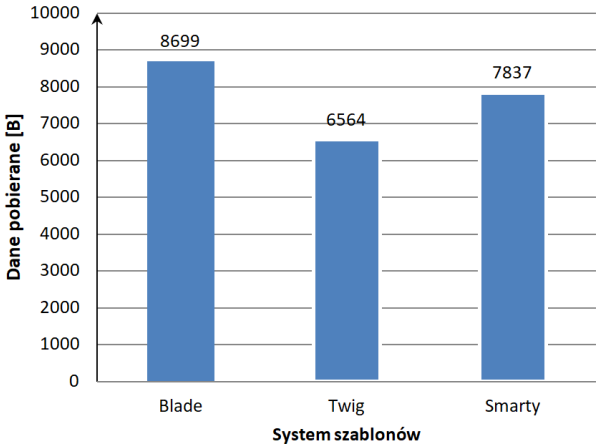
4.3. Wynik eksperymentu

Wynikiem opisanego w rozdziałach 2.3-2.4 eksperymentu jest 300 wierszy danych – po 100 dla każdej z badanych technologii. Dane te dotyczą między innymi czasu odpowiedzi aplikacji oraz odebranych przez użytkownika bajtów. Podczas eksperymentu 10 testów (po 4 w przypadku systemów Twig i Blade oraz 2 w przypadku Smarty) zakończyło się błędem spowodowanym zbyt dużym obciążeniem serwera liczbą zapytań. Dane wynikłe z testów zakończonych niepowodzeniem nie zostały wzięte pod uwagę w celu uniknięcia zaburzeń rezultatu badań.

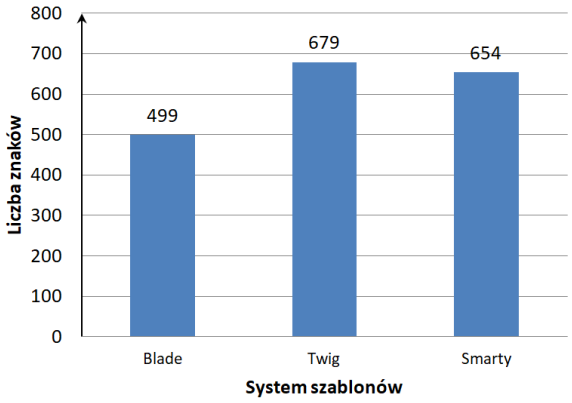
Wyniki eksperymentu przedstawione zostały na wykresie (Rysunek 5). Wyniki dotyczące liczby odebranych bajtów ukazuje następny wykres (Rysunek 6), natomiast ostatni z wykresów (Rysunek 7) prezentuje rozkład liczb użytych podczas korzystania z poszczególnych silników szablonów.



Rysunek 5: Rozkład czasu odpowiedzi poszczególnych aplikacji.



Rysunek 6: Rozkład liczby bajtów pobieranych przez użytkownika.



Rysunek 7: Rozkład liczby znaków użytych podczas korzystania z poszczególnych silników szablonów.

4.4. Wynik końcowy

Posiadając wyniki eksperymentu oraz obliczone wagi kryteriów, wyznaczono wartości punktowe zdobyte przez badane technologie w poszczególnych kryteriach. Dane przedstawione zostały w Tabela .

Tabela 6: Ocena punktowa

	Blade	Twig	Smarty
Najkrótszy czas ładowania strony	0,45	0,48	0,55
Najmniejszy rozmiar kodu, który został napisany w celu wygenerowania ustalonej treści	0,21	0,15	0,16
Najmniejszy rozmiar plików końcowych, które użytkownik pobiera w celu wyświetlenia treści	0,18	0,24	0,20
Suma punktów	0,84	0,87	0,91
Miejsce	3	2	1

5. Wnioski

Odpowiedzią na postawione w rozdziale 1.2. pytanie o to, który z badanych silników szablonów jest najbardziej wydajny w określonych warunkach - jest silnik Smarty. System ten okazał się również silnikiem, który pozwolił na najszybsze załadowanie strony. Jednakże pozostałe systemy przodowały w innych kryteriach. W przypadku najmniejszego rozmiaru plików końcowych, które użytkownik pobiera w celu wyświetlenia treści - Twig okazał się najlepszą technologią. W przypadku aplikacji wykorzystującej Blade potrzeba było z kolei najmniejszej liczby znaków w celu użycia tego silnika szablonów.

Literatura

[1] N. Prokofyeva, V. Boltunova, Analysis and practical application of PHP frameworks in development of web information systems, Procedia Computer Science 104 (2017) 51–56, http://dx.doi.org/10.1016/j.procs.2017.01.059.

[2] A. Jain, Najlepsze platformy programistyczne 2019, https://morioh.com/p/69f2267777df, [22.11.2020].

[3] C. Pitt, Pro PHP MVC, Apress, Berkeley, CA, (2012) 1–7, http://dx.doi.org/10.1007/978-1-4302-4165-2_1.

[4] Porównanie platform programistycznych wykorzystujących PHP, https://socialcompare.com/en/comparison/php-frameworks-comparison, [22.11.2020].

[5] H. Ochim, B. Pańczyk, RWD jako narzędzie optymalizacji stron internetowych, Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska 4 (2016) 81–86, http://dx.doi.org/10.5604/01.3001.0009.5196.

[6] R. Baida, M. Andriienko, M. Plechawska-Wójcik, Analiza porównawcza wydajności frameworków Angular oraz Vue.js, Journal of Computer Sciences Institute 14 (2020) 59–64, http://dx.doi.org/10.35784/jcsi.1577.

[7] P. Mincewicz, M. Plechawska-Wójcik, Analiza wydajnościowa i możliwościowa narzędzia Laravel do tworzenia nowoczesnych aplikacji webowych, Journal of Computer Sciences Institute 7 (2018) 108–115.

[8] N. Kovalchuk, E. Łukasik, Zbadanie wydajności aplikacji internetowych utworzonych z wykorzystaniem Spring MVC oraz JavaServer Faces, Journal of Computer Sciences Institute 1 (2016) 34–37.

[9] K. Bounnady, K. Phanthavong, S. Pathoumvanh, K. Sihalath, Comparison the processing speed between PHP and ASP. NET, 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) (2016) 1–5, http://dx.doi.org/10.1109/ECTICON.2016.7561484.

[10] A. Stecyk, Analiza jakości wybranych systemów e-learningowych za pomocą wielokryterialnej metody analitycznego procesu decyzyjnego AHP, Informatyka Ekonomiczna 49(3) (2018) 78–88, http://dx.doi.org/10.15611/ie.2018.3.07.

[11] Dokumentacja platformy programistycznej Laravel oraz systemu szablonów Blade, https://laravel.com/docs/7.x, [27.10.2020].

[12] Dokumentacja systemu szablonów Twig, https://twig.symfony.com/doc/3.x/, [27.10.2020].

[13] Dokumentacja systemu szablonów Smarty, https://www.smarty.net/docs/en/, [27.10.2020].

[14] I. Teodorescu, V. Vasile, Landing Pages Features to Attract Customers, Ovidius University Annals, Economic Sciences Series 15(2) (2015) 360–363.

[15] Edytor służący do tworzenia stron docelowych, https://landingi.com/pl/projektuj/landing-page/, [04.01.2021].

[16] Dokumentacja Apache JMeter, https://jmeter.apache.org/, [04.01.2021].

[17] A. Prusak, J. Strojny, P. Stefanow, Analityczny Proces Hierarchiczny (AHP) na skróty - kluczowe pojęcia i literatura, Humanities and Social Sciences 4(19.21) (2014) 179-192, http://dx.doi.org/10.7862/rz.2014.hss.66.

[18] D. H. Stone, Design a questionnaire, British Medical Journal 307(6914) (1993) 1264–1266.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 183–188
Received: 08 June 2021
Accepted: 20 June 2021

Comparison of web application state management tools

Porównanie narzędzi do zarządzania stanem aplikacji internetowych

Kacper Szymanek*, Beata Pańczyk

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

Abstract
Modern web applications require flow of large amounts of data. To maintain order in code, a state manager was invented. With manager all data can be retrieved from and goes to one place. In this paper, four libraries for state management (NgRx, Ngxs, Redux, Vuex) were analyzed. Five criteria were used for the study: code metrics, solution structure, availability of ready-made implementations, community support, and performance testing. Results showed that there is not the best tool in every criterion, but when comparing the results obtained, the most universal solution is Vuex.

Keywords: management state; FLUX; store

Streszczenie
Nowoczesne aplikacje internetowe wymagają przepływu dużej ilości danych. Do utrzymania porządku w kodzie wykorzystuje się zarządcę stanu. Dzięki menadżerowi wszystkie dane mogą być pobrane z oraz trafiają w jedno miejsce. W niniejszym artykule analizie poddano cztery biblioteki do zarządzania stanem – NgRx, Ngxs, Redux, Vuex. Do badania wykorzystano pięć kryteriów: metryki kodu, strukturę rozwiązania, dostępność gotowych implementacji, wsparcie społeczności oraz przetestowano wydajnościowo. Wyniki wykazały, że nie ma narzędzia najlepszego w każdym kryterium, jednak zestawiając ze sobą otrzymane rezultaty najbardziej uniwersalnym rozwiązaniem jest Vuex.

Słowa kluczowe: zarządzanie stanem; FLUX; store

*Corresponding author
Email address: **kacper.szymanek1@pollub.edu.pl** (K. Szymanek)

1. Wstęp

Nowoczesne systemy informatyczne wymagają przepływu coraz większej ilości danych, nad którymi ciężko zapanować tak, aby nie tworzyć nadmiernych ilości kodu oraz aby aplikacja była łatwa w rozbudowie. Coraz bardziej na popularności zyskują wzorce projektowe, które pomagają zapobiec takiemu zjawisku. Wzorcem, który porusza problem przetrzymywania i przepływu danych jest FLUX.

Na rynku narzędzi programistycznych jest dużo gotowych implementacji tego wzorca. W tej pracy porównane zostały cztery najpopularniejsze biblioteki tj. NgRx, Ngxs, Redux, Vuex. Każde narzędzie przystosowane jest do innego szkieletu programistycznego.

2. Przegląd literatury

W artykule [1] autorzy przedstawili model i zasady rozwoju "inteligentnego" systemu mieszkaniowego z wykorzystaniem struktury modułowej czujników i wzoru architektonicznego strumienia danych Redux. Wykazali, że wzorzec ten może być wykorzystywany nie tylko jak zakładali autorzy do aplikacji internetowych, ale również w tworzeniu inteligentnych domów.

Artykuł [2] próbuje odpowiedzieć na pytanie jak poradzić sobie z dużymi systemami informatycznymi. Biblioteka Redux została tam wykorzystana do zarządzania stanem aplikacji internetowej. Autorzy udowodnili, że w obecnych czasach zastosowanie menadżera to standard w przypadku rozbudowanych aplikacji internetowych.

Autorzy artykułu "Nowoczesne technologie tworzenia graficznego interfejsu użytkownika w aplikacjach internetowych" [3] badali trzy szkielety programistyczne tj. Angular, React, Vue. Skupili się na szybkości ładowania strony oraz obciążeniu pamięci. Wyniki wskazują, że najgorszym szkieletem jest Angular. Nie wykorzystano tam jednak zarządcy stanu, który mógłby znacznie zmienić wyniki badania.

Artykuły [4,5] skupiają się wokół implementacji wzorca menadżera stanu dla dwóch bibliotek tj. MobX oraz Redux. Oba artykuły pobieżnie sprawdzają możliwość wykorzystania rozwiązań do aplikacji. Lepsza technologia została wybrana na podstawie subiektywnych odczuciach autorów i jest to Redux.

Podsumowując przeprowadzoną analizę nie istnieje zbyt dużo badań prowadzonych na narzędziach zarządzania stanem. MobX oraz Redux są głównymi badanymi technologiami, gdzie ostatecznie lepsza okazuje się druga biblioteka. Wybór ten opierany jest głównie ze względu na lepszą dokumentację, możliwość większego dostosowania implementacji wzorca. Wyniki wskazują również, iż MobX naruszą zasadę "pojedynczego źródła prawdy" [6] i pozwala na deklarowanie wielu źródeł z danymi.

3. Cel badania

Celem badania w niniejszym artykule jest porównanie bibliotek do zarządzania stanem w aplikacjach internetowych oraz wybranie najlepszej możliwej biblioteki. Analizę przeprowadzono dla czterech menadżerów stanu w trzech różnych szkieletach programistycznych

tj. NgRx oraz Ngxs dla Angular, Redux dla React oraz Vuex dla Vue. Redux to biblioteka, która jest przystosowaną do wielu narzędzi, natomiast pozostałe implementacje są dedykowanymi rozwiązaniami do przypisanych im szkieletów.

4. Wzorzec projektowy Flux

Wzorzec projektowy to schemat rozwiązania powtarzających się problemów programistycznych. Wzorzec, który dotyczy zarządzaniem danych w aplikacjach internetowych to Flux. Komponenty widoku zostają uzupełnione o jednokierunkowy przepływ danych oraz jedno źródło z którego dane mogą być pobierane oraz zapisywane. Można wyróżnić trzy główne elementy wzorca:

- magazyn danych – miejsce w którym przechowywany jest stan;
- dyspozytor (ang. dispatcher) – pełni rolę centralnego węzła w aplikacji;
- akcje – elementy pomocnicze przy komunikacji dyspozytor – magazyn danych.

Wzorzec ten powstał aby rozwiązać główny problem wzorca MVC (ang. Model, View, Controller), gdzie przekazywanie danych między widokami jest bardzo problematyczne. Główne elementy MVC to model, widok oraz kontroler. Kontroler odpowiada za manipulację modelem, który aktualizuje widok [6-8]. Oba wzorce Flux i MVC umożliwiają tworzenie aplikacji internetowych, ale Flux to nowa architektura aplikacji, która koncentruje się na **jednokierunkowym przepływie danych.** Przepływ ten rozpoczyna akcja, na którą reaguje dyspozytor, który zmienia wartości w magazynie z danymi. Pojedynczy kontener pozwala na jedną zmianę bez konieczności uaktualniania tych samych danych rozproszonych po całym systemie [9]. Flux jako architektura może być implementowany dowolnie, co powoduje różnorodne nazewnictwo w dostępnych bibliotekach z menadżerem stanu np. dla narzędzia Redux oraz Ngrx za zmiany w magazynie odpowiada reduktor (ang. reducer)[10], natomiast w Vue mutacje (ang. mutation). Oba mechanizmy są sterowane przy pomocy wbudowanego w narzędzie dyspozytora.

5. Aplikacje testowe

W celu zbadania postawionego problemu zaprogramowano cztery aplikacje w odpowiednich szkieletach programistycznych. Każda aplikacja wykorzystuje odpowiednio przystosowaną do narzędzia bibliotekę do zarządzania stanem. Ma to na celu zbadanie przepływu dużych i małych danych (obiekt JSON zawierający zagnieżdżenia, oraz listę 100 obiektów) przez zarządcę stanu oraz jego wpływ na wydajność systemu. Aplikacje realizują funkcjonalności zarządzania zajezdnią autobusową i posiadają tylko wycinek całego rozwiązania. Skupiono się na realizacji przepływu danych pomiędzy komponentami oraz serwerem.

Pierwszą aplikacje zaimplementowano w szkielecie Angular w wersji 10, w której zastosowano rozwiązanie sterowania stanem przygotowane przez bibliotekę NgRx w wersji 9.2.0.

Druga aplikacja również została zaimplementowana w szkielecie Angular w wersji 10. Zastosowano tam narzędzie do zarządzania stanem o nazwie Ngxs w wersji 3.6.2.

Trzecia aplikacja wykorzystuje szkielet programistyczny React w wersji 16.13.1 wraz z biblioteką Redux w wersji 7.2.1. Asynchroniczne zapytania do serwera realizowane są za pomocą biblioteki Redux-Thunk w wersji 2.3.0. Pozwoli ona na porównanie pełnego przepływu danych jakie narzucają pozostałe narzędzia.

Ostatnią aplikację zaimplementowano w Vue w obecnie najnowszej wersji 3 z wykorzystaniem kompozycji API (ang. Composition API), które pozwala na szybszą implementacje komponentów. Menadżer stanu jaki został wykorzystany to dedykowane narzędzie o nazwie Vuex w wersji 4.

Po za aplikacjami klienckimi zaimplementowano prostą aplikację serwerową, z którą komunikują się wszystkie aplikacje klienckie. W tym celu wykorzystano technologię ASP .Net Core w wersji 3.1. Serwer ma za zadanie zwracać obiekty w formacie JSON oraz listy z obiektami. Dane nie są zwracane z bazy danych lecz są wpisane w kod aplikacji. Spowodowane jest to tym, aby czasy odpowiedzi serwera były możliwie najkrótsze.

6. Metody przeprowadzenia badań

Badanie ma na celu znaleźć odpowiedź na pytanie: ***Które narzędzie pozwala na najprostsze, najszybsze i najwydajniejsze zaimplementowanie rozwiązania informatycznego przy użyciu mechanizmu zarządzania stanem?***

W tym celu wybrano pięć **kryteriów** do przeprowadzenia analizy porównawczej.

1. Pierwszym kryterium jest metryka kodu. W celu zbadania, zestawiono magazyn danych, dyspozytor, akcje oraz reduktor czyli moduł mechanizmu zarządzania stanem. Porównaniu poddano liczbę linii jaka jest potrzebna do zaimplementowania poszczególnych elementów wzorca. Następnie pliki zostały zestawione pod względem ich wielkości. Wykorzystano w tym celu jednostkę bajt.
2. Drugim kryterium jest architektura rozwiązania. W tym fragmencie porównano możliwości rozmieszczenia poszczególnych fragmentów kodu, możliwość ich rozdzielenia na odrębne pliki oraz dołożenia dodatkowej abstrakcji w formie fasady.
3. Kolejnym etapem jest badanie dostępności gotowych implementacji. Ma to na celu zbadanie dostępności istniejących już w bibliotekach rozwiązań oraz metod i możliwość pełnego dostosowania ich do potrzeb własnych. Do zbadania zostało zaimplementowane rozwiązanie wykorzystujące to samo żądanie do API przy maksymalnym wykorzystaniu gotowych metod i funkcji bibliotek, bez konieczności tworzenia własnych.
4. Następnym etapem badania jest przeanalizowanie wsparcia społeczności. Porównaniu zostały poddane słowa kluczowe – tagi dla badanych technologii oraz wybrana została najbardziej wyszukiwane narzędzie

w serwisie StackOverflow. Jest to obecnie największy portal zrzeszający ludzi ze świata IT na świecie. [11].

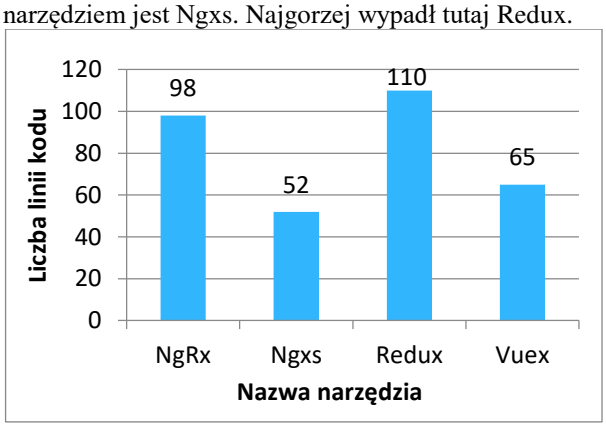
5. Ostatnim kryterium są testy wydajnościowe. Wykorzystano tutaj moduły z zaimplementowanych aplikacji i wykonano trzy scenariusze. Pierwszy to pobranie obiektu JSON z API, zapisanie go oraz pobranie z menadżera stanu. Drugi to pobranie z API stu elementowej listy oraz zapisanie i pobranie do komponentu z magazynu. Ostatni to dodatkowo przekazanie danych z informacją o liście 100 elementowej z magazynu do komponentu dziecka. Od wszystkich czasów potrzebnych na przetworzenie danych odjęto czas żądania API. Wykonano 15 prób dla każdego ze scenariuszy.

7. Wyniki badań

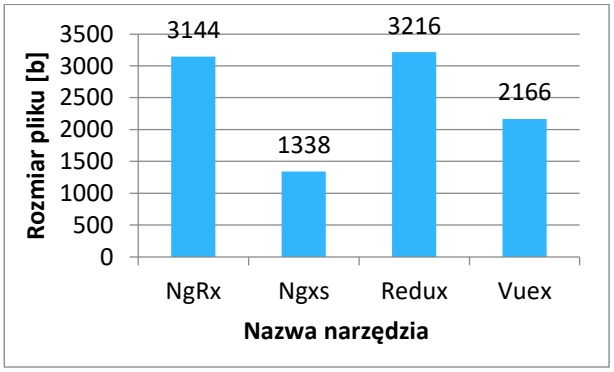
7.1. Metryka kodu

Aby zaprezentować wyniki porównania liczby linii kodu oraz wielkości plików jakie zostały zaimplementowane w badanych narzędziach, wykorzystano moduł menadżera stanu który pobiera obiekt JSON z aplikacji serwerowej. Następnie zapisuje go w magazynie, z którego zostaje pobrany do komponentu.

Na Rysunku 1 zaprezentowano sumaryczne zestawienie linii kodu wszystkich narzędzi, natomiast rysunek 2 prezentuje sumaryczne zestawienie rozmiarów wszystkich plików badanych narzędzi. Najlepszym narzędziem jest Ngxs. Najgorzej wypadł tutaj Redux.

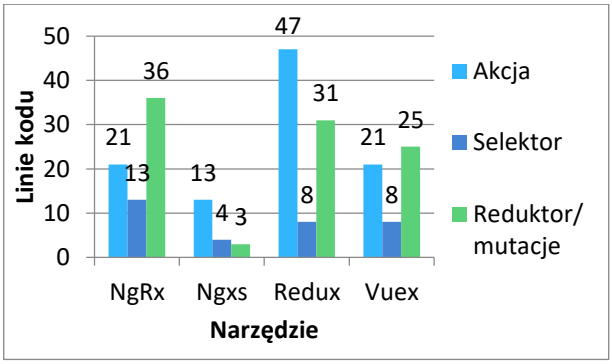


Rysunek 1: Sumaryczna liczba linii kodu wszystkich elementów zarządcy stanu.



Rysunek 2: Rozmiar wszystkich plików zarządcy stanu.

Porównując ze sobą elementy które odpowiadają za tą samą czynność w badanych narzędziach, odczytać można, że najmniej kodu jest w plikach Ngxs. Największy fragment menadżera to akcje zaimplementowane w Redux – 47 linii, natomiast najmniejszy to selektor z narzędzia Ngxs – 3 linie. Obsługa magazynu w narzędziu Vuex nosi nazwę mutacji i jest tożsama z reduktorami w pozostałych bibliotekach. Pełne zestawienie danych przedstawia Rysunek 3.



Rysunek 3: Sumaryczne zestawienie liczby linii kodu dla takich samych elementów wszystkich badanych narzędzi.

7.2. Architektura rozwiązania

W tabeli 1 zestawiono wszystkie narzędzia i możliwości podziału na pliki. Ngxs jako jedyne z badanych narzędzi nie pozwala na dowolne manipulowanie kodem. Redux jako jedyne narzędzie można zaimplementować w dowolnym szkielecie programistycznym.

Tabela 1. Porównanie dowolności architektury w badanych narzędziach

Opis / Narzędzie	Ngrx	Ngxs	Redux	Vuex
Możliwość podziału na pliki	Tak	Tak	Tak	Tak
Możliwość wydzielenia fragmentów kodu odpowiadających elementom wzorca flux	Tak	Nie	Tak	Tak
Możliwość podziału na moduły	Tak	Tak	Tak	Tak
Możliwość dodania dodatkowej warstwy dostępu – fasady	Tak	Tak	Tak	Tak
Możliwość implementacji w dowolnym szkielecie programistycznym bez większych ingerencji w bibliotekę	Nie	Nie	Tak	Nie

7.3. Dostępność gotowych implementacji

Tabela 2 prezentuje porównanie narzędzi pod względem zgodności z wzorcem projektowym Flux.

Wszystkie założenia wzorca tj. pojedyncze źródło prawdy oraz jednokierunkowy przepływ danych zostały spełnione w badanych narzędziach. Posiadają one także niezbędne elementy tj. akcje, dyspozytor, magazyn danych. Tylko Ngxs wymaga klasy jako akcji, w pozostałych badanych narzędziach akcja jest implementowana jako metoda.

Tabela 2. Porównanie prawidłowego dostosowania do wzorca Flux

Opis / Narzędzie	Ngrx	Ngxs	Redux	Vuex
Prawidłowe przygotowanie narzędzia do wzorca Flux	Tak	Tak	Tak	Tak
Akcje	Tak	Tak	Tak	Tak
Dyspozytor	Tak	Tak	Tak	Tak
Magazyn danych	Tak	Tak	Tak	Tak
Jednokierunkowy przepływ danych	Tak	Tak	Tak	Tak
Sposób implementacji Akcji	Funkcja	Klasa	Funkcja	Funkcja

7.4. Wsparcie społeczności

Najbardziej znanym oraz popularnym narzędziem zrzeszającym programistów oraz osoby związane ze światem Informatyki jest obecnie StackOverFlow. Jest to publiczna platforma do zadawania pytań programistycznych oraz do dzielenia się wiedzą [9]. Platforma pozwala na analizę wsparcia społeczności narzędzi. Udostępnia statystykę związaną ze słowami kluczowymi czyli tagami. Tagi są związane z technologią, której dotyczy pytanie. Do sprawdzenia największego wsparcia społeczności należy zliczyć liczbę tagów występujących przy pytaniach. Oprócz tego strona pozwala sprawdzić w jakich miesiącach oraz latach dany tag jest najczęściej wyszukiwaną frazą.

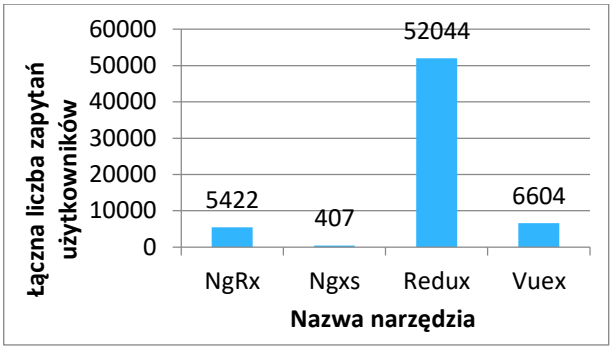
Najpopularniejszy sposób na zbudowanie słowa kluczowego to nazwa taga taka sama jak narzędzia. Często można także spotkać dodanie specjalistycznej nazwy problemu do nazwy biblioteki w ten sposób tworząc słowo kluczowe.

Sumując liczbę dostępnych tagów po jakich można wyszukiwać zapytania, narzędzie Redux posiada największą ich liczbę spośród wszystkich badanych. Drugi w kolejności jest NgRx, następnie Vuex. Najmniej słów kluczowych z badanych narzędzi posiada Ngxs – 2. Wyniki łącznej liczby słów kluczowych zestawiono w Tabeli 3.

Tabela 3. Liczba słów kluczowych na portalu StackOverFlow

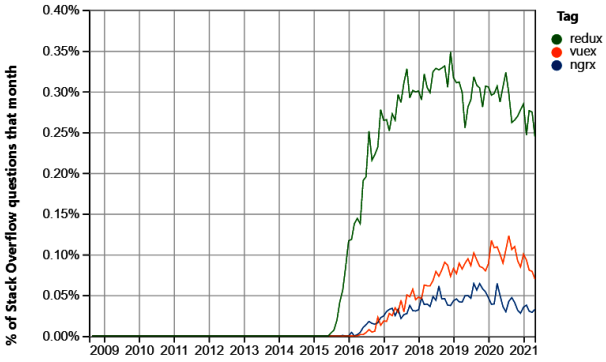
Nazwa narzędzia	Liczba słów kluczowych
NgRx	14
Ngxs	2
Redux	35
Vuex	7

Wykres zestawiający łączną liczbę zapytań użytkowników o badane biblioteki zaprezentowano na Rysunku 4. Wyniki wskazują, że najczęściej zadawane są pytania o technologię Redux. Vuex oraz NgRx plasują się na podobnym poziomie zapytań. Najmniej wyszukiwanym narzędziem jest Ngxs.



Rysunek 4: Liczba zapytań o narzędzie według tagów w serwisie StackOverFlow.

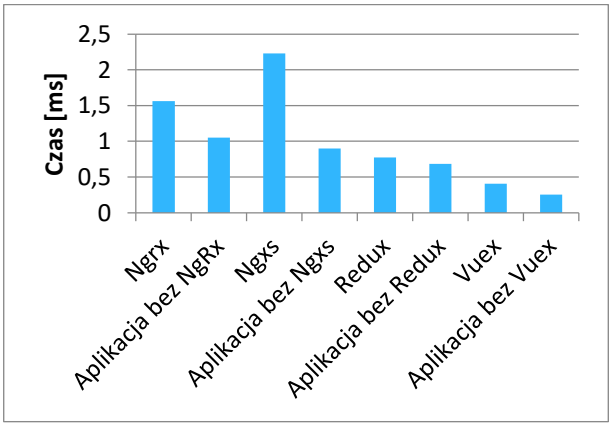
Rysunek 5 przedstawia procent wszystkich zapytań zadanych w całym portalu StackOverFlow, w poszczególnych latach. Redux widocznie przoduje nad pozostałymi narzędziami. Istotną informacją jaką można odczytać z tego rysunku jest to, że wszystkie narzędzia zaczęły być wyszukiwane w podobnym czasie tj. na przełomie 2015 oraz 2016 roku. Ngxs jest tak mało popularny, że nie występuje na wykresie udostępnionym przez portal StackOverFlow. Ngrx jest drugim najmniej popularnym narzędziem.



Rysunek 5: Procent pytań StackOverflow w danym roku [12]

7.5. Testy wydajnościowe

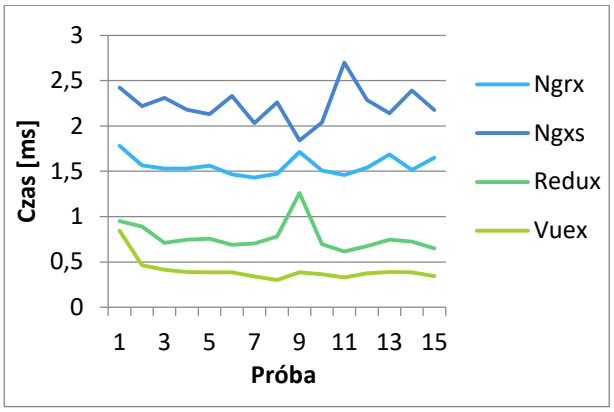
Średni czas potrzebny do przetworzenia obiektu JSON przez narzędzia przedstawiono na Rysunku 6. Najlepszą biblioteką jest Vuex. Najgorszy czas przetworzenia potrzebny jest implementując rozwiązanie w Ngxs.



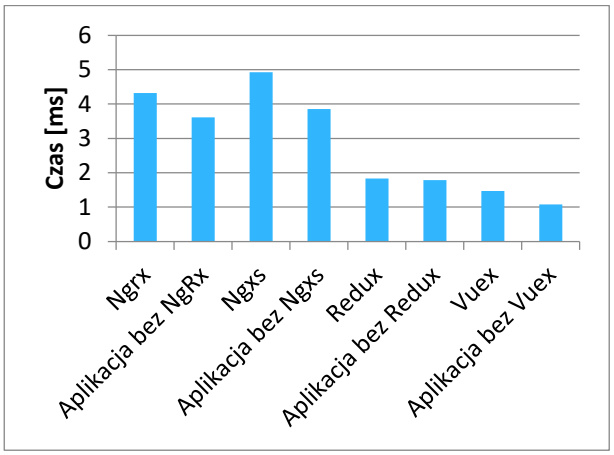
Rysunek 6: Wykres średniego czasu potrzebnego na przetworzenie obiektu JSON przez aplikacje.

Rysunek 7 przedstawia wykres z wynikami pomiarów z 15 prób przetworzenia danych. Vuex posiada najbardziej zbliżone wyniki w każdym przypadku. Najgorzej wypada Ngxs. Poza największym czasem potrzebnym na przetworzenie danych w magazynie, posiada także największe rozbieżności w wynikach co może wskazywać na małą stabilność tego rozwiązania.

Najlepszy czas przetworzenia 100 elementowej listy ma narzędzie Vuex (Rysunek 8). Jest to duża przewaga nad najgorszym wynikiem dla Ngxs Drugim najlepszym narzędziem jest Redux.



Rysunek 7: Wykres 15 prób z czasem potrzebnym na przetworzenie danych.

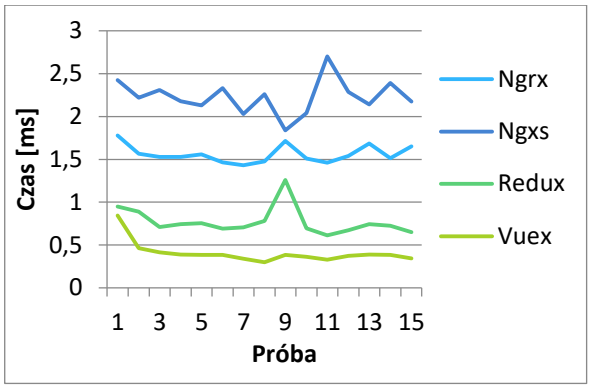


Rysunek 8: Wykres średniego czasu potrzebnego na przetworzenie 100 elementowej listy przez aplikacje.

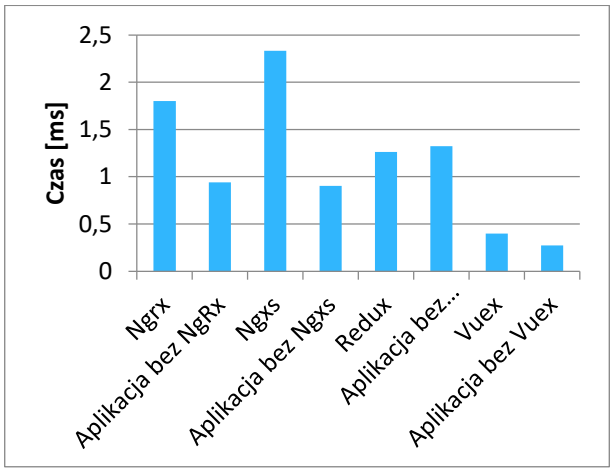
Vuex tak jak w przypadku pojedynczego obiektu, tak i w przypadku listy zachowuje się bardzo liniowo i stabilnie bez żadnych odchyłów. Najgorzej prezentuje się Ngxs (Rysunek 9).

Rysunek 10 prezentuje wykres średniego czasu potrzebnego do przetworzenia oraz przekazania 100 elementowej listu do komponentu dziecka. Wyniki wskazują, iż Ngxs również tutaj wypada najgorzej. Najlepszym narzędziem tak jak w poprzednich scenariuszach jest Vuex.

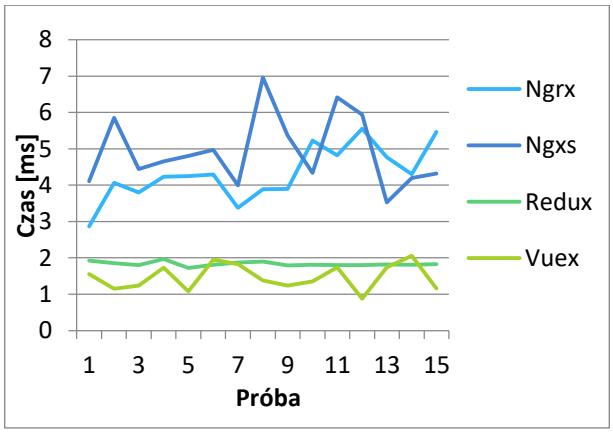
Zestawiając zebrane dane na wykresie liniowym (Rysunek 11) zauważyć można, że tym razem to Redux posiada najbardziej zbliżone do siebie wyniki



Rysunek 9: Wykres 15 prób z czasem potrzebnym na przetworzenie 100 elementowej listy obiektów.

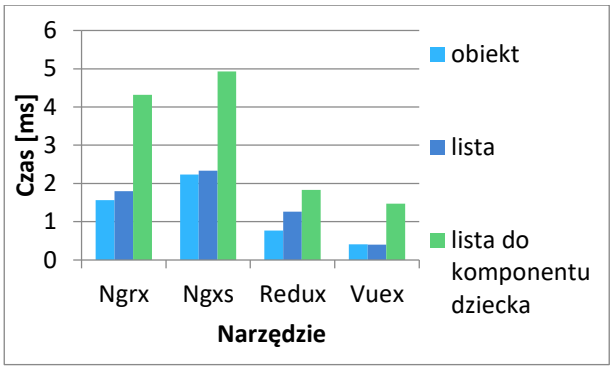


Rysunek 10: Wykres średniego czasu potrzebnego na przetworzenie 100 elementowej listy przez aplikacje i przekazanie jej do komponentu dziecka.



Rysunek 11: Wykres 15 prób z czasem potrzebnym na przetworzenie 100 elementowej listy obiektów i przekazanie jej do komponentu dziecka.

Podsumowując wszystkie scenariusze badania wydajności (Rysunek 12), najwięcej czasu zajmowało przetworzenie i przekazanie 100 elementowej listy do komponentu. Biblioteki przystosowane do szkieletu programistycznego Angular tj. Ngrx oraz Ngxs poradziły sobie najgorzej w tym zadaniu. Najlepiej we wszystkich scenariuszach poradził sobie Vuex.



Rysunek 12: Średnie czasy badanych bibliotek w 3 różnych scenariuszach.

8. Wnioski

Celem artykułu było porównanie bibliotek do zarządzania stanem w aplikacjach internetowych.

Do wskazania najlepszego narzędzia, wykorzystano punktację, która była przyznawana w skali od 1 do 4, gdzie 4 oznacza najlepsze narzędzie a 1 najgorsze. Dodatkowo przyznano 1 punkt gdy narzędzie spełniało zadane kryterium i 0 gdy nie.

Najlepszym narzędziem według tego badania jest Vuex. Na drugim miejscu są biblioteki przystosowane do szkieletu programistycznego Angular tj. NgRx oraz Ngxs. Na ostatnim miejscu znajduje się Redux. Wyniki porównania przedstawiono w Tabeli 4.

Tabela 4: Wyniki punktowe zebrane z kryteriów badanych narzędzi.

Opis / Narzędzie		Ngrx	Ngxs	Redux	Vuex
Metryka kodu	Najmniejszy rozmiar plików	2	4	1	3
	Najmniej linii kodu do implementacji	2	4	1	3
Architektura rozwiązania	Możliwość podziału na pliki według wzorca flux	1	0	1	1
	Możliwość implementacji w dowolnym szkielecie programistycznym	0	0	1	0
Dostępność gotowych implementacji	Dostępność największej ilości gotowych metod	4	3	1	2
	Zgodność ze wzorcem FLUX	1	1	1	1
Wsparcie społeczności	Największe zainteresowanie społecznością	2	1	4	3
Testy wydajnościowe	Najszybsze narzędzie	2	1	3	4
Suma punktów		**14**	**14**	**13**	**17**

Literatura

[1] A. Kazarian, V. Teslyuk, I. Tsmots, J. Greguš, Development of a «smart» home system based on the modular structure and architectural data flow pattern Redux, Procedia Computer Science. 155 (2019), 35-42.

[2] S. Mukhiya1, K. Hung. An Architectural Style for Single Page Scalable Modern Web Application, International Journal of Recent Research Aspects, 5(4) (2018), 6-13.

[3] M. Kaproń, B. Pańczyk. Nowoczesne technologie tworzenia graficznego interfejsu użytkownika w aplikacjach internetowych, Journal of Computer Sciences Institute 15 (2020), 139-152.

[4] D. Holmstedt, Analyzing and implementing a third-party state machine library for FriendlyReader and TeCST, Linköping University, Department of Computer and Information Science, Bachelor's thesis (2019).

[5] W. Wenhao, React Native vs Flutter, Cross-platforms mobile application frameworks, Metropolia University of Applied Sciences, Bachelor of Engineering Information technology Thesis (2018).

[6] Porównanie flux oraz mvc https://madasamy.medium.com/flux-vs-mvc-design-pattern-de134dfaa12b [12.04.2021].

[7] Opis czym jest flux https://jerzywickowski.pl/flux/co-to-jest-flux/, [25.03.2021].

[8] Wprowadzenie do wzorca architektonicznego Flux, https://www.freecodecamp.org/news/an-introduction-to-the-flux-architectural-pattern-674ea74775c9/, [25.03.2021].

[9] A. Boduch, Flux architecture, Packt Publishing Ltd (2016).

[10] D. Bugl, Learning Redux, Packt Publishing Ltd (2017).

[11] StackOverFlow – portal dla programistów, https://stackoverflow.com, [12.03.2021].

[12] Trendy zapytań w latach 2009-2021 w serwisie StackOverFlow, https://insights.stackoverflow.com/trends?tags=ngrx%2Cvuex%2Credux, [11.04.2021].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 189–196
Received: 09 June 2021
Accepted: 23 June 2021

Comparative analysis of the methods of watermarking X-ray images

Analiza porównawcza metod znakowania wodnego obrazów RTG

Weronika Zofia Kulbaka* , Paulina Paluch*, Grzegorz Kozieł

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

Abstract

This paper is devoted to the comparative analysis of watermarking algorithms for X-ray images. The techniques based on discrete wavelet transform (DWT), singular value decomposition (SVD) and DWT-SVD hybrid were compared. Transparency, resistance to graphical transformations, and performance were investigated. The watermarked images were visually evaluated and quality tested. SVD showed the highest resistance to attacks, and the embedded watermarked images were of better quality in the comparison to the other algorithms. The DWT technique was the fastest, but not resistant to graphical transformations. In DWT-SVD labeled images, the watermark is indistinguishable, but the resistance to attacks is low. The SVD was found to be the most suitable method for watermarking of X-ray images.

Keywords: digital watermarks; security; steganography

Streszczenie

Artykuł poświęcono analizie porównawczej algorytmów znakowania wodnego obrazów RTG. Porównano technikę opartą na dyskretnej transformacie falkowej (DWT), metodę rozkładu wartości osobliwych (SVD) oraz hybrydę DWT-SVD. Zbadano przezroczystość, odporność na przekształcenia graficzne, a także szybkości. Oznakowane obrazy poddano ocenie wizualnej oraz badaniu jakości. SVD wykazała największą odporność na ataki, a obrazy z osadzonym znakiem wodnym były lepszej jakości w porównaniu do obrazów oznakowanych za pomocą pozostałych algorytmów. Technika DWT była najszybsza, jednak nieodporna na przekształcenia graficzne. Na obrazach oznakowanych DWT-SVD znak wodny jest niedostrzegalny, jednak odporność na ataki jest niska. Najodpowiedniejszą metodą do znakowania obrazów RTG okazała się metoda SVD.

Słowa kluczowe: cyfrowe znaki wodne; bezpieczeństwo; steganografia

*Corresponding author

Email address: weronika.kulbaka@pollub.edu.pl (W. Z. Kulbaka), paulina.paluch1@pollub.edu.pl (P. Paluch)

1. Wstęp

Obecnie możliwe jest przesyłanie cyfrowych obrazów i danych medycznych przez Internet w celu rozpoznawania chorób i zdalnej diagnostyki pacjentów. W szybkim tempie wzrasta liczba przypadków fałszowania, manipulowania, usuwania i nieuprawnionego wykorzystywania danych medycznych. Obrazy medyczne, takie jak zdjęcia z diagnostycznych badań radiologicznych są istotnymi plikami, które należy dokładnie chronić. Istnieją różne metody znakowania wodnego obrazów RTG. Jednak niektóre z nich mogą skutkować utratą jakości lub znacząco wpłynąć na możliwość odczytania wyniku badań ze zdjęcia poprzez widoczność znaku wodnego. Zdjęcia rentgenowskie powinny być jak najlepszej jakości, a cyfrowe znaki wodne nie mogą wpływać na interpretację badania [1]. W tym celu zostanie przeprowadzona analiza wybranych trzech metod znakowania cyfrowego. Badanymi metodami będą: DWT (ang. Discrete Wavelet Transform - Dyskretna Tranformata Falkowa), SVD (ang. Singular Value Decomposition - Rozkład Wartości Pojedynczej) oraz hybryda DWT-SVD. Ponadto wyszczególniono następującą tezę i hipotezę badawczą:

T1: Istnieją metody znakowania wodnego obrazów medycznych pozwalające na uzyskanie odporności na popularne przekształcenia graficzne.

H1: Dobór odpowiedniego algorytmu znakowania wodnego pozwala na uzyskanie wysokiej przezroczystości osadzonego znaku wodnego.

Postawiono również następujące pytania badawcze:

1. Czy istnieje metoda znakowania wodnego, która umożliwia oznakowanie obrazu RTG bez utraty jakości?
2. Która z popularnych metod znakowania wodnego jest najbardziej odporna na popularne przekształcenia graficzne?

2. Analiza literatury

Bezpieczeństwo i ochrona praw autorskich treści cyfrowych jest obecnie bardzo trudną i istotną kwestią. Wynika to z dynamicznego rozwoju technologii multimedialnych umożliwiających łatwe przechowywanie, powielanie oraz rozpowszechnianie plików cyfrowych [1]. Obecnie występuje coraz więcej kradzieży tożsamości, nielegalnego kopiowania i dystrybucji danych, a także naruszeń prywatności. Autentyczność obrazów jest szczególnie ważna w dziedzinie medycyny. Obecnie możliwe jest przesyłanie cyfrowych obrazów i danych medycznych przez Internet w celu rozpoznawania chorób i zdalnej diagnostyki pacjentów. W szybkim tempie wzrasta liczba przypadków fałszowania, manipulowania, usuwania i nieuprawnionego wykorzystywania danych medycznych. Opisana sytuacja wpływa na duże

zainteresowanie skutecznymi i bezpiecznymi metodami zabezpieczania przesyłanych w sieci cyfrowych danych oraz ochronę praw autorskich.

W ostatnich latach powstało wiele prac naukowych w dziedzinie steganografii oraz znakowania wodnego obrazów. Jedni uczeni analizują istniejące techniki steganograficzne, inni identyfikują ograniczenia i braki tych metod oraz sprawdzają odporność na przekształcenia. Niektórzy weryfikują, które metody najlepiej sprawdzają się w zastosowaniu do obrazów medycznych. Jeszcze inni przedstawiają aktualne trendy oraz wyzwania, które będą podejmowane w przyszłości przez ludzi nauki.

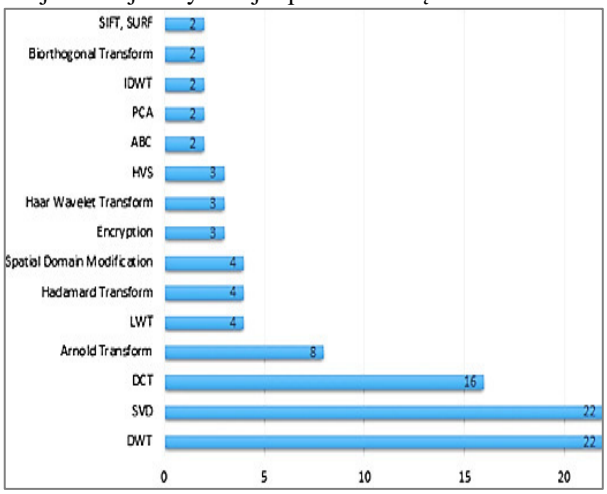
W pracy naukowej „Digital image watermarking Techniques : a review" [4] autorzy przedstawili techniki znakowania wodnego obrazów (m. in. medycznych), takie jak LSB (ang. Least Significant Bit - Najmniej Znaczacy Bit), lub DCT (ang. Discrete Cosine Transform - Dyskretna Transformata Cosinusoidalna) i inne. Zweryfikowali oferowane przez nie właściwości, takie jak: niewykrywalność, bezpieczeństwo, niezawodność oraz pojemność steganograficzna wykonując różnego rodzaju przekształcenia oznakowanych obrazów medycznych. Przydatne okazały się również obliczenia wskaźników jakości zdjęć takie jak PSNR (ang. Peak Signal to Noise Ratio), SSIM (ang. Structural Similarity) czy też MSE (ang. Mean Square Error). Każda z badanych metod miała swoje mocne i słabe strony. Metoda DWT okazała się najlepsza pod względem wysokiej jakości znakowanego obrazu. Znak wodny był niezauważalny, jednak obraz nie był odporny na niektóre znane ataki. Metoda LSB była najszybsza, jednak nie zapewniała odporności na popularne zniekształcenia. Autorzy ocenili, że w sektorze zdrowia, najlepsze okazały się metody DWT, DCT oraz SVD.

Artykuł „A DWT-SVD based Robust Digital Watermarking for Digital Images" [3] w całości został poświęcony hybrydowej technice znakowania obrazów poprzez metodę DWT oraz SVD. Wyniki badań pokazały, że na obrazach oznakowanych techniką DWT-SVD znaki wodne były wizualnie niedostrzegalne. Algorytm ten zapewnia również wysoką odporność na ataki związane z przetwarzaniem obrazu.

Porównanie różnych technik zabezpieczania obrazów znakami wodnymi zawarto również w artykule „Watermarking techniques for medical data authentication: a survey" [2]. Autorzy przeanalizowali różne metody znakowania wodnego skupiając się przede wszystkim na obrazach medycznych np. zdjęciach z diagnostycznych badań radiologicznych. Opisali cechy charakterystyczne, najnowsze zastosowania oraz koncepcje osadzania i odzyskiwania znaku wodnego. Omówili również potencjalne wyzwania związane z osadzaniem znaków wodnych na obrazach medycznych. Autorzy podkreślili cechy, jakie powinny spełniać skuteczne techniki znakowania wodnego obrazów medycznych:

- bezpieczeństwo,
- niewykrywalność,
- odporność na uszkodzenia,
- niezawodność,
- szybkość.

Autorzy artykułu „Recent trends in image watermarking techniques for copyright protection: a survey" [11] opracowali wyniki dotyczące różnych technik znakowania wodnego obrazów, określili najnowsze praktyki z nimi związane oraz ich ograniczenia. Przedstawili również procentowy udział najczęściej używanych metod znakowania wodnego (Rys. 1). Najpopularniejsze okazały się metody DWT oraz SVD. Na trzecim miejscuw tej klasyfikacji uplasowała się technika DCT.



Rysunek 1: Najczęściej używane metody znakowania wodnego obrazów [11]

Inne porównanie technik znakowania wodnego opisali w swoim artykule Garg P. oraz Kishore R. Rama [6]. Przedstawili oni informacje na temat znaków wodnych pod względem ich charakterystyki, obszarów zastosowania, typów i różnych ataków na nie przeprowadzanych.

Ze względu na popularność metod zdecydowano się na przebadania dwóch najpopularniejszych z nich oraz ich połączenie - metodę DWT-SVD, która według literatury charakteryzuje się dobrymi właściwości znakowania. W prezentowanych badaniach użyte zostały zdjęcia RTG różnych części ciała osób dorosłych i dzieci.

3. Metodyka badawcza

Przygotowano zbiór 100 źródłowych obrazów RTG pochodzących z ogólnodostępnych baz danych obrazów medycznych [7, 8] oraz znak wodny w postaci logo Politechniki Lubelskiej w skali szarości. Na podstawie analizy literatury opracowano trzy wybrane algorytmy znakowania wodnego oraz wyodrębniania znaku wodnego z obrazów. Zaimplementowano również funkcje służące do realizacji określonych ataków na oznakowane obrazy takie jak obrót, rozmycie oraz wyostrzenie.

Cały zbiór obrazów źródłowych oznakowano za pomocą każdej z metod. Zestaw obrazów wynikowych zapisano do folderu znajdującego się w plikach źródłowych. Obrazy oznakowane za pomocą każdej z metod poddano badaniom oceny jakości za pomocą miar względnych oraz bezwzględnych takich jak:

- MSE – błąd średniokwadratowy, wartość oczekiwana podniesionej do kwadratu różnicy pomiędzy es-

tymatorem, a wartością estymowaną [12]. Określany jest wzorem:

$$MSE = \frac{1}{N \cdot M} \sum_{i=1}^{N} \cdot \sum_{j=1}^{M} \cdot ([f(i,j) - f'(i,j)]^2) \quad (1)$$

gdzie N, M to wymiary obrazu w pikselach, $f(i,j)$ to wartość reprezetująca odcień piksela o współrzędnch (i,j) obrazu oryginalnego, $f'(i,j)$ to wartość reprezentująca odcień piksela o współrzędnych (i,j) obrazu skompresowanego,

- PSNR – stosunek maksymalnej mocy sygnału do mocy zakłócającego szumu wyrażany w decybelach [12]. Używany jest do określenia jakości obrazu [13]. Określany jest wzorem:

$$PSNR = 10 \cdot \log_{10} \frac{\left[\max\left(f(i,j)\right)\right]^2}{MSE} \quad (2)$$

gdzie $\max\left(f(i,j)\right)$ to wartość maksymalna danego sygnału,

- SSIM – miara prawdopodobieństwa strukturalnego służąca do pomiaru podobieństwa pomiędzy dwoma obrazami [12]. Miara między dwoma oknami x oraz y rozmiaru $N \times N$ jest wyznaczana wzorem:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (3)$$

gdzie μ_x to średnia z x, μ_y to średnia z y, σ_x^2 to wariancja z x, σ_y^2 to wariancja z y, σ_{xy} to kowariancja z x oraz y, $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$, gdzie k_1 oraz k_2 – stałe, L – rozpiętość totalna wartości pikseli,

- MS-SSIM (ang. Multi-Scale SSIM) - popularna odmiana indeksu SSIM. Wykorzystuje podstawowy algorytm SSIM działając na kilku skalach,
- 3-SSIM (ang. Three-component SSIM) - zapewnia wyniki zgodne z ludzką subiektywnością przy określaniu jakości rozmytych i zaszumionych obrazów [12],
- NIQE (ang. Natural Image Quality Evaluator) - naturalna miara jakości obrazu. Pozyskuje zestaw lokalnych cech obrazu, a następie dopasowuje wektory cech do modelu MVG (ang. Multivariate Gaussian) [10],
- PIQE (ang. Perception-based Image Quality Evaluator) - percepcyjny wskaźnik oceny jakości obrazu dla obrazów świata rzeczywistego [10],
- BRISQUE (ang. Blind/Referenceless Image Spatial Quality Evaluator) - niewidoczna/nieodnosząca się miara jakości obrazu przestrzennego. Wykorzystuje statystykę sceniczną (ang. scene statistic) lokalnie znormalizowanych współczynników luminacji do ilościowej oceny możliwych start "naturalności" obrazu spowodowanych obecnością zniekształceń [5].

W każdym przypadku wykonywano te same ataki z takimi samymi parametrami. Obrazy przekształcone zostały poddane analizie w celu określenia stopnia uszkodzenia dołączonych danych oraz możliwości odczytania dołączonego znaku wodnego i liczby przekłamanych bitów znaku wodnego. Uzyskane w wyniku znakowania obrazy również oceniono wyżej wymienionymi metrykami, a także wskaźnikiem liczby utraconych bitów BER (ang. Bit Error Rate).

Badania wykonywane były na laptopie Dell Inspiron z czterordzeniowym procesorem Intel Core i7-1065G7 oraz dyskiem SSD M.2 PCIe o pojemności 1TB. Laptop posiadał 16GB pamięci RAM oraz system operacyjny Microsoft Windows 10 Home w wersji 64-bitowej. Podczas przeprowadzania badań szybkości technik znakowania wodnego na komputerze działał tylko system operacyjny oraz oprogramowanie do znakowania wodnego.

W celu zapewnienia porównywalnych warunków prowadzenia badań wyłączono oprogramowanie antywirusowe, a komputer odłączono od sieci. Badania powtórzono 20 razy.

Wybrane do badań metody zaimplementowano w środowisku Matlab, zgodnie z algorytmami przedstawionymi przez ich autorów [1, 6, 9]. Opracowanych implementacji użyto do przeprowadzenia badań.

4. Ocena jakości oznakowanych obrazów

W procesie znakowania wodnego obrazów medycznych i innych bardzo ważne jest zachowanie odpowiedniego poziomu przezroczystości ukrywanego znaku wodnego. Wybierając odpowiednią metodę należy zwrócić uwagę na pogorszenie się jakości obrazu oraz występowanie zmian, które mogą wskazywać na istnienie dołączonego znaku [14]. W celu obiektywnego zbadania jakości obrazów z umieszczonymi znakami wodnymi wykorzystano metryki względne i bezwzględne, które oceniają w różny sposób jakość obrazu badając wielkość zakłóceń konkretnego typu [15].

Wartości poszczególnych wskaźników prawie nie zmieniają się w przypadku znaków wodnych o różnych rozmiarach. Zauważone różnice wartości mieszczą się w granicach błędów zaokrągleń. Spowodowane jest to implementacją algorytmów, które zawsze umieszczają znak wodny o maksymalnym dostępnym rozmiarze [16]. W przypadku zastosowania znaku wodnego o mniejszym rozmiarze jest on skalowany do rozmiaru obrazu znakowanego. Zdecydowano się na takie rozwiązanie, ponieważ w niniejszej pracy nacisk położono na badanie odporności znaku wodnego na uszkodzenia oraz określeniu maksymalnej dostępnej pojemności steganograficznej.

W tabelach 1, 2, 3 oraz 4 przedstawiono wartości średnie wybranych metryk dla zaimplementowanych technik znakowania wodnego oraz porównanie wizualne. Do badania użyto 40 obrazów o rozmiarze około 100 KB. Znakowano je znakiem wodnym o rozmiarze 100 KB. Przed obliczeniem wartości średnich posortowano dane i odrzucono po 5% wyników najmniejszych i największych.

Najlepszy wynik dla pomiaru PSNR uzyskała metoda SVD - 42,5dB, a najgorszy metoda DWT – 31,4dB. Wyższa wartość wskaźnika PSNR oznacza mniejszy poziom zakłóceń wprowadzony do obrazu [12]. Wynika stąd, że oznakowanie obrazu RTG techniką SVD powo-

duje najmniejszy spadek jakości zdjęcia w stosunku do pozostałych zaimplementowanych algorytmów.

Tabela 1: Średnie metryki względne oceny jakości obrazu oznakowanego różnymi metodami w stosunku do obrazu oryginalnego

	PSNR [dB]	SSIM	MS-SSIM	3-SSIM
DWT	31,4	0,895	0,992	0,992
SVD	42,5	0,990	0,995	0,995
DWT-SVD	31,9	0,962	0,997	0,997

Wartości wskaźników SSIM, MS-SSIM oraz 3-SSIM przyjmują wartości [0,1], gdzie 0 oznacza całkowicie różne obrazy, a 1 - identyczne obrazy [10]. Dla wszystkich trzech metod znakowania wyniki wskazują na bardzo duże podobieństwo obrazu oznakowanego w stosunku do oryginału. W badaniu SSIM najgorzej wypadła metoda DWT, a najlepiej SVD. Natomiast w badaniach wskaźników MS-SSIM oraz 3-SSIM najlepszy wynik osiągnęła technika DWT-SVD. Przeprowadzono również ocenę wizualną. Na obrazach oznakowanych algorytmem SVD znak wodny nie jest widoczny, natomiast w przypadku zastosowania metody DWT dostrzegalne są kontury osadzonego logo, szczególnie na obszarach o ciemnym tle. Na większości obrazów oznakowanych techniką DWT-SVD znak wodny jest niewidoczny.

Tabela 2: Średnie metryki bezwzględne obrazów oryginalnych

NIQE	PIQE	BRISQUE
4,2	15,3	19,8

Tabela 3: Metryki bezwzględne obrazu oznakowanego wybranymi metodami

	NIQE	PIQE	BRISQUE
DWT	4,0	15,3	18,8
SVD	4,1	12,6	22,4
DWT-SVD	4,1	16,3	19,3

Badając jakość obrazów medycznych po umieszczeniu znaku wodnego można zastosować miary bezwzględne, ponieważ dostępne są zarówno obrazy oryginalne, jak i zniekształcone. Jedną z często stosowanych metryk jest NIQE (ang. Naturalness Image Quality Evaluator), czyli wskaźnik oceny tego, jak bardzo obraz zniekształcony jest podobny do oryginalnego [9]. Najlepszy wynik osiągnęła metoda DWT, jednak rezultaty dla pozostałych algorytmów różnią się od niej zaledwie o wartość 0,1. Wskaźnik PIQE (ang. Perception based Image Quality Evaluator) działa podobnie jak NIQE. Wynik PIQE to nieujemny skalar liczbowy z zakresu 1-100. Jeśli wynik zawiera się w przedziale 0-20, jakość obrazu jest bardzo dobra, natomiast wynik z zakresu 81-100 oznacza obraz bardzo słabej jakości [10]. Wynik dla obrazu oryginalnego wynosi 15,3. Wyniki uzyskane przez wszystkie analizowane metody są zbliżone i wskazują, że każda z nich zapewnia bardzo dobrą jakość obrazu. Wynik metryki BRISQUE (ang. Blind/Referenceless Image Spatial Quality Evaluator) jest również skalarem liczbowym z zakresu 1-100. Im niższa wartość, tym obraz jest lepszej jakości. Najbardziej jakość obrazu RTG spadła po umieszczeniu znaku wodnego algorytmem DWT, natomiast najwyższą jakość mają obrazy oznakowane metodą SVD, choć i tutaj różnica jest niewielka.

Analiza uzyskanych wyników wykazała, że najmniejsze zniekształcenia wprowadziła metoda SVD. Uzyskała ona najlepsze wartości wszystkich analizowanych wskaźników. Oznacza to, że obrazy oznakowane tą metodą będą zawierały najmniej zniekształceń a co za tym idzie będą najbardziej wartościowe w diagnostyce medycznej spośród obrazów oznakowanych pozostałymi badanymi metodami.

5. Ocena wizualna oznakowanych obrazów

Jednym ze sposobów oceny jakości oznakowanych obrazów jest przeprowadzenie subiektywnego testu, w którym tester ocenia każdy obraz z osadzonym znakiem wodnym w ustalonej skali. W niniejszej pracy ocenie poddano zbiór stu obrazów źródłowych porównanych do obrazów oznakowanych trzema badanymi algorytmami znakowania wodnego. Wyniki zaprezentowano
w tabeli 4. Oceniono zniekształcenia wprowadzone podczas znakowania wodnego w każdym z obrazów. Użyto następującej skali:

1. zniekształcenia zupełnie niewidoczne,
2. zniekształcenia prawie niewidoczne,
3. zniekształcenia widoczne, nie przeszkadzające,
4. zniekształcenia widoczne, przeszkadzające,
5. obraz zniszczony.

Tabela 4: Wyniki oceny wizualnej obrazów oznakowanych poszczególnymi metodami

	Procent obrazów, które otrzymały poszczególne oceny		
Ocena	DWT [%]	SVD [%]	DWT-SVD [%]
1	5	100	100
2	39	0	0
3	55	0	0
4	1	0	0
5	0	0	0

W wizualnej ocenie widoczności znaku wodnego najlepiej wypadła metoda rozkładu według wartości osobliwych oraz hybryda metod DWT i SVD. Ludzkie oko nie było w stanie zauważyć artefaktów znaku wodnego w obrazach oznakowanych tymi metodami. Natomiast w obrazach oznakowanych metodą DWT trudno było znaleźć przykłady, gdzie znak wodny byłby całkowicie niewidoczny. Wyniki uzyskane podczas oceny wizualnej odzwierciedlają wyniki uzyskane podczas oceny jakości za pomocą takich miar jak SSIM czy PSNR.

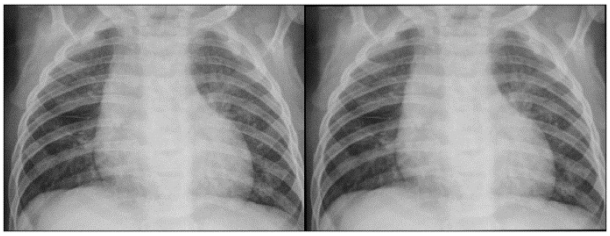
6. Odporność na przekształcenia

Technika znakowania wodnego stosowana do ochrony praw autorskich obrazów medycznych powinna być odporna na przekształcenia graficzne. Wybrano kilka najpopularniejszych przekształceń i przetestowano, jak radzą sobie z nimi zaimplementowane metody.

W tym celu znakowano zdjęcia RTG, a następnie wyodrębniano obraz będący znakiem wodnym i badano wartość wskaźnika BER. Rezultaty badania odporności zaimplementowanych algorytmów na obrót oraz rozmycie zaprezentowano na wykresach i wyciągnięto wnioski.

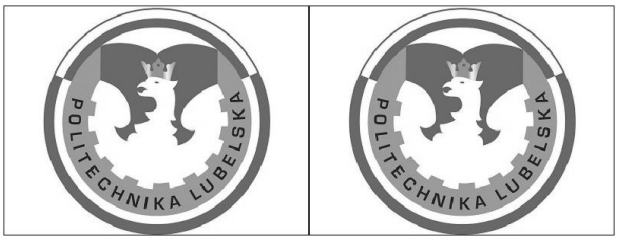
6.1. Badanie odporności na obrót

Sprawdzono jak każda z technik radzi sobie z wyodrębnieniem znaku wodnego z obróconego oznakowanego obrazu medycznego (ang. rotation attack). Metodą prób i błędów wyszczególniono zakres dopuszczalnego kąta obrotu. Badanie przeprowadzono dla następujących wartości kąta obrotu obrazu: 0,01 stopnia, 0,1 stopnia, 0,5 stopnia, 10 stopni, 45 stopni, 90 stopni. Na rysunku 2 zaprezentowano oznakowany obraz oraz ten sam obraz obrócony o kąt 0,5 stopnia, natomiast na rysunkach 3, 4, 5 znajduje się znak wodny oryginalny porównany ze znakami wodnymi wyizolowanym z obróconych obrazów oznakowanych za pomocą badanych metod.

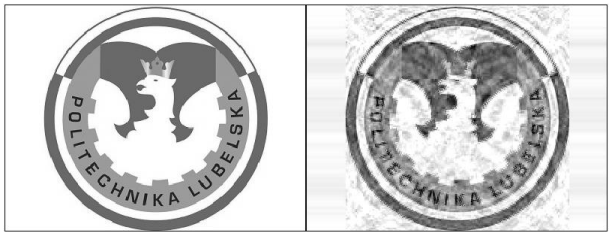


Rysunek 2: Oznakowany obraz prześwietlenia klatki piersiowej (po lewej) oraz to samo zdjęcie obrócone o 0,5 stopnia (po prawej).

Rysunek 3: Znak wodny (po lewej) oraz wyizolowany znak wodny z obrazu (po prawej) oznakowanego metodą DWT i obróconego o 0,5 stopnia.

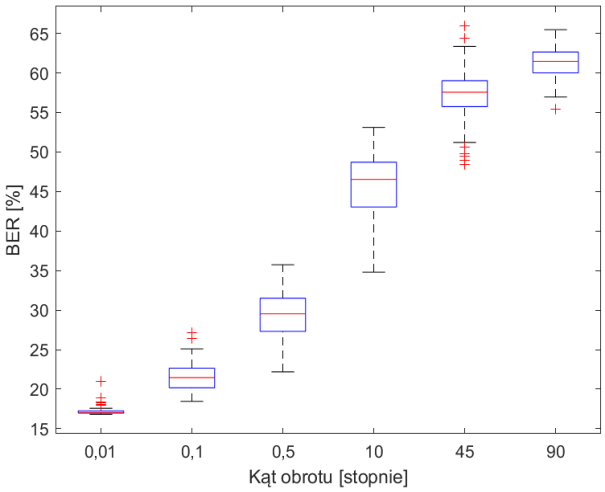


Rysunek 4: Znak wodny (po lewej) oraz wyizolowany znak wodny z obrazu (po prawej) oznakowanego metodą SVD i obróconego o 0,5 stopnia.

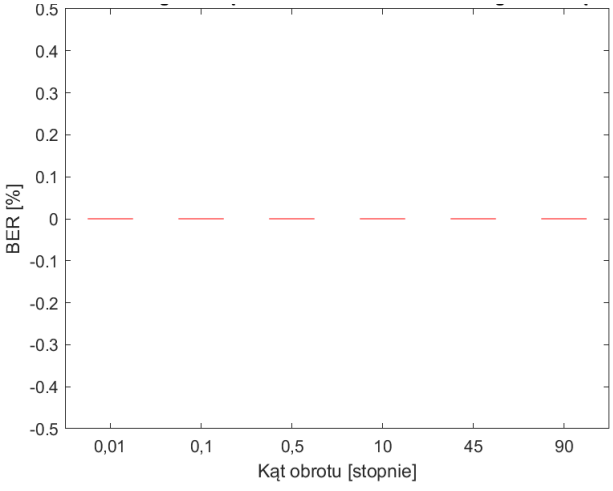


Rysunek 5: Znak wodny (po lewej) oraz wyizolowany znak wodny z obrazu (po prawej) oznakowanego metodą DWT-SVD i obróconego o 0,5 stopnia

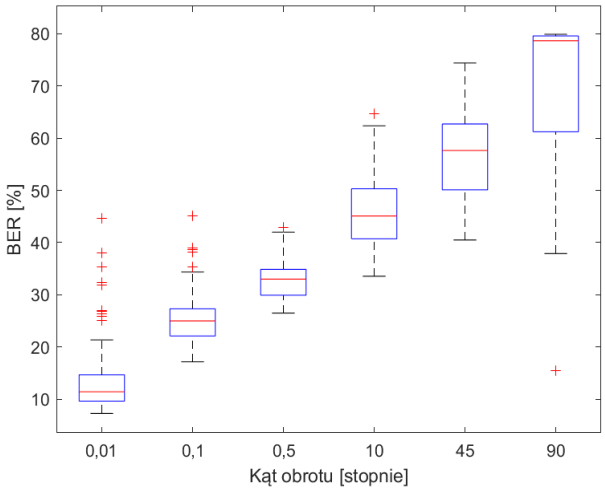
Na logo Politechniki Lubelskiej wyodrębnionym z obróconego obrazu oznakowanego metodą DWT widoczne są żebra z prześwietlenia klatki piersiowej. Wyizolowanie znaku wodnego z obrazu RTG oznakowanego techniką SVD nie spowodowało utraty danych, natomiast w przypadku algorytmu DWT-SVD znak wodny jest lekko rozmazany. Wykresy zależności wskaźnika BER wyizolowanego znaku wodnego od kąta obrotu obrazu zaprezentowano na rysunkach 6, 7 i 8.



Rysunek 6: Wykres zależności wartości wskaźnika BER od kąta obrotu obrazu dla metody DWT.



Rysunek 7: Wykres zależności wartości wskaźnika BER od kąta obrotu obrazu dla metody SVD.

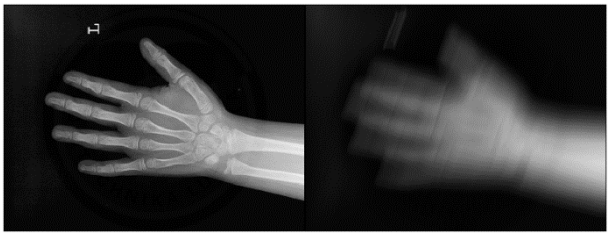


Rysunek 8: Wykres zależności wartości wskaźnika BER od kąta obrotu obrazu dla metody DWT-SVD.

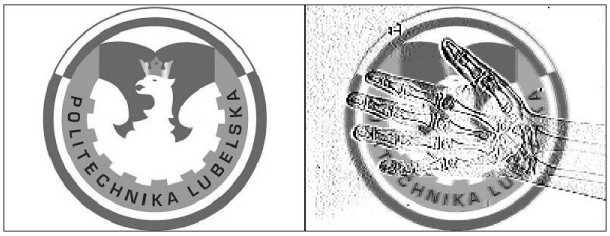
6.2. Badanie odporności na rozmycie

W celu zbadania odporności na atak rozmycia (ang. motion attack) sprawdzono przy jak dużym rozmazaniu uda się wyodrębnić wizualnie niezniszczony znak wodny. Wykorzystana funkcja programu Matlab przyjmuje parametry: kąt przesunięcia (theta) oraz długość ruchu rozmycia (len). Przyjęto długość ruchu len=10. Rysunek 9 przedstawia porównanie obrazu oznakowanego dłoni z obrazem zaatakowanym rozmyciem, natomiast na rysunkach 10, 11 i 12 porównania znaku wodnego ze znakiem wodnym wyodrębnionym z zaatakowanego obrazu.

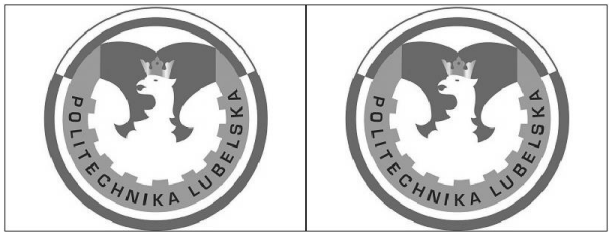
Zależność wartości wskaźnika BER od liczby utraconych bitów prezentują wykresy pokazane na rysunkach 13, 14 i 15. Najwyższą odporność na przekształcenie rozmycia wykazała technika SVD, natomiast metody DWT oraz DWT-SVD nie są odporne na ten atak, o czym świadczą zarówno zniekształcone znaki wodne, jak i znaczna utrata danych widoczna na wykresach.



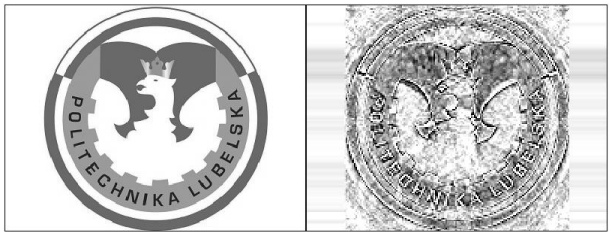
Rysunek 9: Oznakowany obraz prześwietlenia dłoni (po lewej) oraz to samo zdjęcie zaatakowane przekształceniem rozmycia (po prawej).



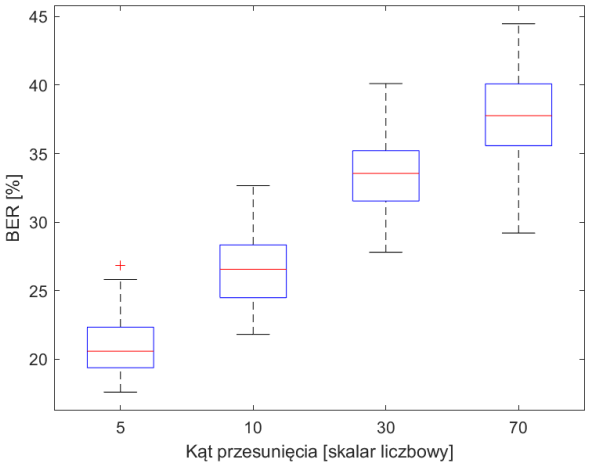
Rysunek 10: Znak wodny (po lewej) oraz wyizolowany znak wodny z rozmytego obrazu (po prawej) oznakowanego metodą DWT.



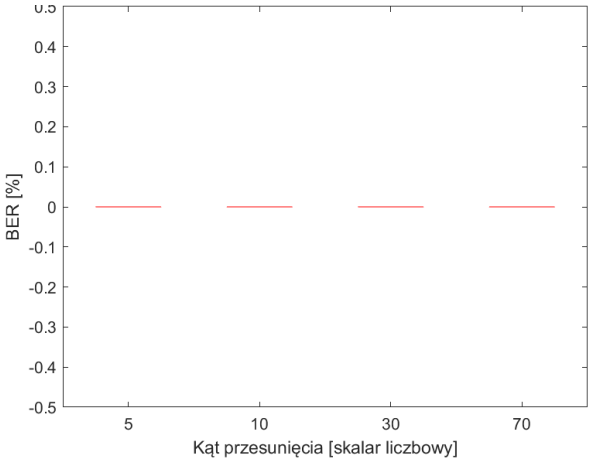
Rysunek 11: Znak wodny (po lewej) oraz wyizolowany znak wodny z rozmytego obrazu (po prawej) oznakowanego metodą SVD.



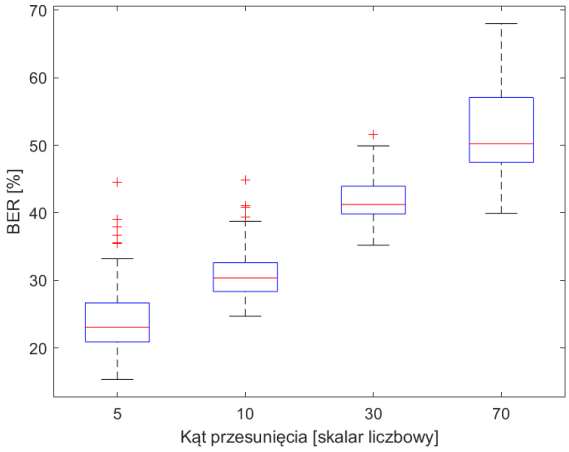
Rysunek 12: Znak wodny (po lewej) oraz wyizolowany znak wodny z rozmytego obrazu (po prawej) oznakowanego metodą DWT-SVD.



Rysunek 13: Wykres zależności wartości wskaźnika BER od kąta obrotu obrazu dla metody DWT.



Rysunek 14: Wykres zależności wartości wskaźnika BER od kąta obrotu obrazu dla metody SVD.



Rysunek 15: Wykres zależności wartości wskaźnika BER od kąta obrotu obrazu dla metody DWT-SVD.

7. Szybkość

Ostatnim kryterium oceny była szybkość wykonywania operacji znakowania wodnego oraz ekstrakcji. Jako znak wodny za każdym razem zastosowano logo Politechniki Lubelskiej o rozmiarze 100KB. Wybrano 30 obrazów
o rozmiarze 100KB i 20KB. Badania szybkości powtórzono 20 razy. Odrzucono 10% wyników najniższych oraz 10% wyników najwyższych, a następnie obliczono średnie czasy odpowiednich operacji w aplikacji. Rezultaty zaprezentowano w tabeli 5.

Tabela 5: Wyniki badania szybkości wybranych metod znakowania wodnego

	DWT		SVD		DWT-SVD	
Rozmiar obrazów [KB]	20	100	20	100	20	100
Średni czas znakowania [s]	0,09	0,12	0,06	0,10	0,13	0,18
Średni czas izolowania znaku [s]	0,05	0,05	0,08	0,08	0,06	0,08

Najkrótszy średni czas procesu znakowania obrazu osiągnęła metoda SVD. Zarówno w przypadku obrazu o rozmiarze 20 KB, jak i 100KB średnie czasy były krótsze niż pozostałych badanych technik. Najwolniejsza okazała się metoda DWT-SVD, ponieważ średni czas znakowania obrazu wynosi 0,13 dla obrazów o rozmiarze 20KB oraz 0,18s dla obrazów o rozmiarze 100KB. Czasy osadzania znaku wodnego są około dwukrotnie większe niż techniki SVD. Najszybciej proces izolacji znaku wodnego wykonuje się algorytmem DWT, a najdłużej – SVD.

8. Wnioski

Postawiona teza badawcza "Istnieją metody znakowania wodnego obrazów medycznych pozwalające na uzyskanie odporności na popularne przekształcenia graficzne" została potwierdzona. Ataki przekształcenia przeprowadzone na obrazie oznakowanym metodą SVD nie zniekształciły znaku wodnego zamieszczonego w obrazie źródłowym. Z każdego zaatakowanego obrazu udało się wyekstrahować nienaruszony znak. Niską skutecznością wykazywało się połączenie metod DWT-SVD. Metoda DWT w każdym z przebadanych obrazów nie była odporna na popularne przekształcenia graficzne. Uzyskane wyniki nie potwierdziły więc doniesień znalezionych w artykułach naukowych. W przeprowadzonych badaniach wykazano największą odporność znaków wodnych umieszczonych za pomocą metody SVD.

Teza "Dobór odpowiedniego algorytmu znakowania wodnego pozwala na uzyskanie wysokiej przezroczystości osadzonego znaku wodnego", również została potwierdzona. Jak wykazano przeźroczystość znaku wodnego zależy od zastosowanego algorytmu znaku wodnego. Najlepsze wyniki uzyskała metoda SVD oferując wysoki stopień przezroczystości dołączonych danych co potwierdziły zarówno testy numeryczne jak i ocena wizualna. Na obrazach oznakowanych metodą DWT można zauważyć kontury znaku wodnego. Jest to szczególnie widoczne na ciemnym tle obrazów rentgenowskich. Natomiast obrazy, w których został umieszczony znak wodny metodą SVD, wizualnie nie różniły się od obrazów źródłowych. Niemniej jednak każda z metod wprowadziła zniekształcenia do znakowanych obrazów, co potwierdzają uzyskane wyniki numeryczne.

Podsumowując, SVD wykazuje się znakomitą odpornością na popularne ataki przekształcenia. Metodę cechuje najkrótszy średni czas procesu znakowania, jednak jest najmniej wydajna pod względem izolacji znaku wodnego z obrazu. Wizualnie ciężko jest dostrzec różnice pomiędzy obrazami znakowanymi metodą SVD, a źródłowymi. Na podstawie wyników wskaźników miar jakości zdjęć również cechuje się najlepszą jakością obrazu spośród wszystkich trzech metod. Natomiast metoda DWT uzyskała najlepszy czas izolacji znaku wodnego. Jednak nie była odporna na popularne ataki przekształcenia oraz jakość obrazu oznakowanego tą metodą była najgorsza. Hybryda DWT-SVD otrzymywała zazwyczaj uśrednione wyniki. Nie była odporna na duże kąty obrotu, czy duże rozmycie – jednak przy małych wartościach była w stanie odtworzyć znak wodny.

Literatura

[1] A. Anand, A. K. Singh, An improved DWT-SVD domain watermarking for medical information security. Computer Communications, 152 (2020) 72-80.

[2] A. Anand, A. K. Singh, Watermarking techniques for medical data authentication: a survey. Multimedia Tools and Applications 313 (2020) 1-33.

[3] S. M. Arora, A DWT-SVD based robust digital watermarking for digital images. Procedia computer science, 132 (2018) 1441-1448.

[4] M. Begum, M. S. Uddin, Digital Image Watermarking Techniques: A Review. Information, 11(2) (2020) 110.

[5] B. V. Bhalerao, R. R. Manza, Y. M. Rajput, Use of quality measures for rural indian fingerprint image database enhancement and improve the recognition rate. International Journal of Computer Applications, 70(18) (2013) 13-15.

[6] P. Garg, R. R. Kishore, Performance comparison of various watermarking techniques. Multimedia Tools and Applications, 79(35) (2020) 25921-25967.

[7] Internetowa baza zdjęć medycznych: https://medpix.nlm.nih.gov/home, [22.03.2021r.]

[8] Internetowa społeczność naukowców i praktyków uczenia maszynowego: https://www.kaggle.com, [22.03.2021r.]

[9] F. Kahlessenane, A. Khaldi, R. Kafi,, S. Euschi, A DWT based watermarking approach for medical image protection. Journal of Ambient Intelligence and Humanized Computing, 12(4) (2020) 1-8.

[10] Y. H. Liu, K. F. Yang, H. M. Yan, No-Reference Image Quality Assessment Method Based on Visual Parameters. Journal of Electronic Science and Technology, 17(2) (2019) 171-184.

[11] A. Ray, S. Roy, Recent trends in image watermarking techniques for copyright protection: a survey. International Journal of Multimedia Information Retrieval, 9(4) (2020) 1-22.

[12] U. Sara, M. Akter, M. S. Uddin, Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study. Journal of Computer and Communications, 7(3) (2019) 8-18.

[13] A. Shehab, M. Elhoseny, K. Muhammad, A. K. Sangaiah, P. Yang, H. Huang, G. Hou, Secure and robust fragile watermarking scheme for medical images. IEEE Access, 6 (2018) 10269-10278.

[14] R. Mironov, S. Kushlev, Medical Images Watermarking using Wavelet Transform and DCT. In LIII International Scientific Conference on Information, Communication and Energy Systems and Technologies, Sozopol , 2018.

[15] M. S. El_Tokhy, Development of optimum watermarking algorithm for radiography images. Computers & Electrical Engineering, 89 (2021) 106932.

[16] H. Lala, Digital image watermarking using discrete wavelet transform. International Research Journal of Engineering and Technology, 4(1) (2017) 1682-1685.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 197–204
Received: 14 June 2021
Accepted: 28 June 2021

Analysis of the possibilities for using machine learning algorithms in the Unity environment

Analiza możliwości wykorzystania algorytmów uczenia maszynowego w środowisku Unity

Karina Litwynenko*, Małgorzata Plechawska-Wójcik

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

Reinforcement learning algorithms are gaining popularity, and their advancement is made possible by the presence of tools to evaluate them. This paper concerns the applicability of machine learning algorithms on the Unity platform using the Unity ML-Agents Toolkit library. The purpose of the study was to compare two algorithms: Proximal Policy Optimization and Soft Actor-Critic. The possibility of improving the learning results by combining these algorithms with Generative Adversarial Imitation Learning was also verified. The results of the study showed that the PPO algorithm can perform better in uncomplicated environments with non-immediate rewards, while the additional use of GAIL can improve learning performance.

Keywords: reinforcement learning; imitation learning; Unity

Streszczenie

Algorytmy uczenia ze wzmocnieniem zyskują coraz większą popularność, a ich rozwój jest możliwy dzięki istnieniu narzędzi umożliwiających ich badanie. Niniejszy artykuł dotyczy możliwości zastosowania algorytmów uczenia maszynowego na platformie Unity wykorzystującej bibliotekę Unity ML-Agents Toolkit. Celem badania było porównanie dwóch algorytmów: Proximal Policy Optimization oraz Soft Actor-Critic. Zweryfikowano również możliwość poprawy wyników uczenia poprzez łączenie tych algorytmów z metodą uczenia przez naśladowanie Generative Adversarial Imitation Learning. Wyniki badania wykazały, że algorytm PPO może sprawdzić się lepiej w nieskomplikowanych środowiskach o nienatychmiastowym charakterze nagród, zaś dodatkowe zastosowanie GAIL może wpłynąć na poprawę skuteczności uczenia.

Słowa kluczowe: uczenie ze wzmocnieniem; uczenie przez naśladowanie; Unity

*Corresponding author
Email address: karina.litwynenko@pollub.edu.pl (K. Litwynenko)

1. Wstęp

Na zaobserwowany w ostatnim czasie duży postęp w dziedzinie algorytmów uczenia ze wzmocnieniem wpływ miało istnienie odpowiednich do badań nad nimi narzędzi [1]. Zapewniają one dostęp do środowisk o coraz większym stopniu złożoności, jak i o bardziej realistycznej grafice, czy też dokładniejszej fizyce. Są też one dla badaczy głównym narzędziem umożliwiającym projektowanie, a także testowanie algorytmów. Z tego powodu, jakość oraz możliwości tych platform są bardzo często istotnym czynnikiem postępu w dziedzinie uczenia ze wzmocnieniem. Niniejszy artykuł dotyczy platformy Unity, przedstawiając jego możliwości w kontekście badań nad algorytmami uczenia ze wzmocnieniem. Unity to narzędzie pozwalające na tworzenie gier i symulacji zawierających dokładną fizykę, a także realistyczną oprawę graficzną. Platforma ta pozwala na tworzenie rozgrywki posługując się w dużej mierze wygodnym, graficznym interfejsem użytkownika. Połączenie jej z otwartoźródłową biblioteką Unity ML-Agents Toolkit sprawia, że staje się ona rozbudowanym i elastycznym narzędziem do pracy z algorytmami uczenia maszynowego. W ramach pakietu ML-Agents Toolkit dostępne są dwa algorytmy uczenia ze wzmocnieniem — Proximal Policy Optimization (PPO) [2] oraz Soft Actor-Critic (SAC) [3]. Narzędzie wspiera również uczenie przez naśladowanie, a wśród udostępnianych algorytmów można wskazać Generative Adversarial Imitation Learning (GAIL) [4, 5].

Pomimo że początki algorytmów uczenia ze wzmocnieniem sięgają już lat 50. [6], to właśnie w ciągu ostatnich kilku lat zaobserwowano znaczny postęp w tej dziedzinie. Przejawia się on w szczególności powstawaniem nowych algorytmów, takich jak PPO oraz SAC. Pierwszy z nich powstał na podstawie równie niedawno przedstawionego algorytmu Trust Region Policy Optimiza-

tion (TRPO) [7]. Metoda PPO okazała się nie tylko wydajniejsza od swojego poprzednika, ale też o wiele prostsza w implementacji [2]. Z kolei algorytm SAC został porównany z kilkoma innymi współczesnymi algorytmami uczenia ze wzmocnieniem, w tym PPO. W rezultacie, SAC wykazał się wyższą wydajnością oraz szybkością uczenia w stosunku do pozostałych badanych metod [3]. Dostępne są też wyniki uczenia obu algorytmów PPO i SAC na platformie Unity. Obie metody zastosowano w siedemnastu przykładowych środowiskach wchodzących w skład biblioteki ML-Agents Toolkit [1]. Nie podjęto jednak próby analizy tych rezultatów. Unity wykorzystywano również do treningu agentów w innych, indywidualnych projektach środowisk [8]. Jednak nie zostały one szczegółowo opisane, ani nie wskazano zastosowanych algorytmów uczenia. Platforma Unity oraz biblioteka Unity ML-Agents Toolkit posłużyła również do badania skuteczności sposobów definiowania przestrzeni akcji agenta [9]. Wśród narzędzi pozwalających na testowanie algorytmów uczenia ze wzmocnieniem w symulowanych środowiskach, obok Unity, można wyróżnić platformę Arcade Learning Environment [7, 10–13] oraz OpenAI Gym [14]. Algorytm uczenia przez naśladowanie GAIL okazał się skuteczniejszy od kilku innych porównywanych metod, takich jak Behavioral Cloning (BC), Feature Expectation Matching (FEM), czy Game-Theoretic Apprenticeship Learning (GTAL) [4]. Jako podsumowanie aktualnego stanu wiedzy warto zwrócić uwagę na to, że dotychczas wykonane porównania PPO z SAC opierały się na eksperymentach przeprowadzonych w predefiniowanych środowiskach, bez możliwości wprowadzania w nich zmian. Trudno jest odnaleźć prace dotyczące zależności pomiędzy specyfiką środowiska agenta, a skutecznością algorytmów PPO oraz SAC. Brakuje też badań nad możliwościami łączenia algorytmów uczenia ze wzmocnieniem z uczeniem przez naśladowanie.

Celem przedstawionego w niniejszej pracy badania było porównanie algorytmów uczenia maszynowego udostępnianych w ramach biblioteki Unity ML-Agents Toolkit, współpracującej z platformą Unity. Badanie skupiło się na dwóch współczesnych algorytmach uczenia ze wzmocnieniem — PPO oraz SAC. Weryfikacji poddano również hipotezę o możliwości poprawy skuteczności uczenia poprzez łączenie tych algorytmów z metodą GAIL. Uczenie z wykorzystaniem badanych metod przeprowadzono w środowisku zaprojektowanym za pomocą platformy Unity.

Kolejny rozdział artykułu zawiera opis metody badań. Dalsza część przedstawia projekt wykonanego w Unity środowiska agentów sztucznej inteligencji. Ostatnie dwa rozdziały zostały poświęcone na wyniki badania, a także wnioski, w których zawarto krótkie podsumowanie pracy oraz wynikłe konkluzje. W końcowej treści wskazano również możliwe dalsze kierunki prac.

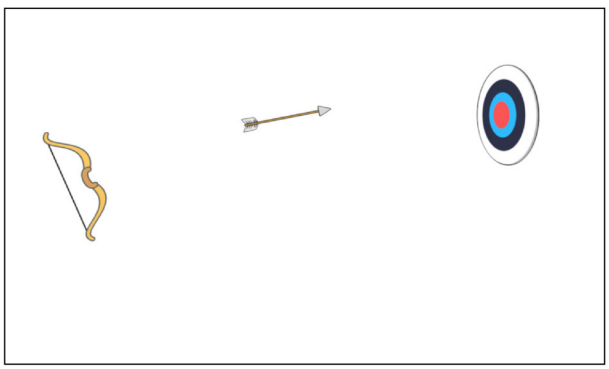
2. Metoda badań

Porównanie algorytmów zostało przeprowadzone na podstawie rezultatów uczenia agentów w grze 2D zaprojektowanej za pomocą platformy Unity wykorzystującej bibliotekę ML-Agents Toolkit. Przygotowana gra składała się z trzech wariantów, które charakteryzowały się innym poziomem trudności rozgrywki. Głównym zadaniem agenta było strzelanie z łuku do tarczy treningowej, zdobywając przy tym jak największą sumę punktów (Rysunek 1). Poszczególne warianty środowiska można scharakteryzować następująco:

1. Siła oddawanego strzału była zawsze stałą, ponadto tarcza nigdy nie zmieniała swojego położenia.
2. Siła strzału była stałą, jak w przypadku wariantu pierwszego, jednak po każdym celnym strzale tarcza zmieniała swoją pozycję, pojawiając się losowo w jednym z 25 możliwych miejsc.
3. Siła strzału była nadawana przez agenta, kontrolującego naciągnięcie cięciwy łuku. Tarcza zmieniała też swoje położenie, jak w wariancie drugim.

Algorytmy zostały więc porównane na podstawie rezultatów eksperymentu, zakładającego zwiększanie złożoności środowiska agenta, rozpoczynając od zadania najbardziej podstawowego. Każda z wersji przygotowanego środowiska charakteryzowała się również tym, że zdobywane przez agenta nagrody nie były nigdy natychmiastowe, a wymagały wykonania korzystnej sekwencji kroków. Porównanie opierało się na wynikach przebiegu uczenia, jak również nauczonego przez agentów zachowania. Wytrenowane modele sprawdzono w rozgrywce uwzględniającej liczbę zdobytych punktów, celność (stosunek strzałów trafionych w cel do wszystkich oddanych), a także czas ukończenia zadania przy ograniczonym zasobie strzał.

Możliwości łączenia algorytmów PPO i SAC z GAIL w celu poprawy wyników uczenia zbadano w trzecim, najbardziej złożonym wariancie środowiska. Zastosowane demonstracje pochodziły z rozgrywki człowieka, trwającej 8 i 48 minut. Rezultaty uczenia łączonych metod zostały porównane w ten sam sposób, co samodzielne algorytmy PPO oraz SAC — na podstawie przebiegu uczenia i rozgrywki wytrenowanych agentów.



Rysunek 1: Widok z rozgrywki przygotowanej na potrzeby badania gry.

3. Projekt środowiska

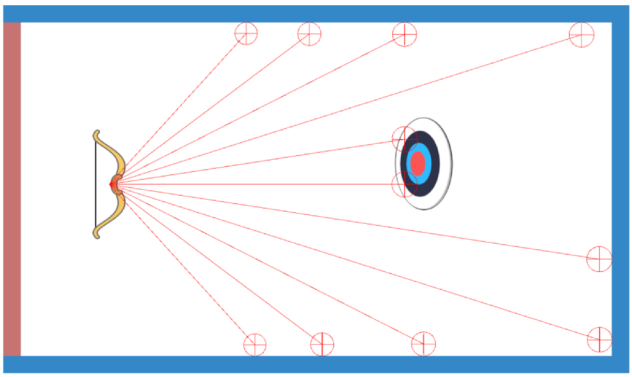
Zadaniem agenta w przygotowanym na potrzeby eksperymentu środowisku jest strzelanie z łuku do tarczy treningowej. Każdy celny strzał jest nagradzany odpowiednią liczbą punktów, zależną od koloru trafionego okręgu tarczy — od 0,7 do 1 (Rysunek 2). Sterowanie dla gracza jest realizowane za pomocą klawiatury. Użycie strzałki lewej lub prawej obraca łuk, zaś sterowanie cięciwą odbywa się za pomocą spacji. W przypadku dwóch prostszych wariantów gry do oddania strzału wystarczy pojedyncze wciśnięcie tego klawisza, zaś w najtrudniejszym trybie rozgrywki możliwe jest jego przytrzymanie w celu nadania większej siły wystrzału.

Rysunek 2: Punktacja dla gracza, a także wartości wzmocnienia zdobywane przez agenta w zależności od trafionego okręgu tarczy.

Obserwacje agenta, zbierane w każdym kroku uczenia, składają się z następujących elementów:

- gotowość łuku do oddania strzału (1 dla prawdy, 0 dla fałszu),
- kąt obrotu łuku,
- wartości zarejestrowane przez 10 promieni komponentu *RayPerceptionSensor3D*, wykrywające tarczę oraz ściany otaczające agenta (Rysunek 3),
- wartość naciągu cięciwy (tylko w przypadku 3. wariantu środowiska).

Ponadto obserwacje te są kumulowane dwukrotnie, co oznacza, że agent podejmuje decyzje na podstawie bieżącego oraz poprzedniego stanu środowiska.



Rysunek 3: Rozmieszczenie promieni komponentu *RayPerceptionSensor3D* z widocznymi ścianami otaczającymi agenta.

Akcje agenta zaimplementowano jako dyskretne, stosując gałęzie akcji (z ang. *action branching*), które pozwalają uprościć przestrzeń akcji, rozdzielając ją na grupy, z których agent może wybierać wartości jednocześnie [9,15]. Ponadto akcje podejmowane są co 5 kroków środowiska i są one automatycznie powtarzane pomiędzy kolejnymi decyzjami. Maksymalna długość epizodu została ustawiona na 1000 kroków.

Środowisko przewiduje następujące wartości wzmocnienia:

- od 0,7 do 1 w zależności od koloru trafionego okręgu tarczy (Rysunek 2),
- - 0,01 za trafienie w ścianę za agentem (czerwona na rysunku 3)
- - 0,001 za trafienie w ścianę boczną lub przednią,
- - 0,001 za zwłokę, gdy została podjęta akcja inna niż oddanie strzału, gdy łuk jest w stanie gotowości lub napięcie cięciwy osiągnęło maksymalny poziom.

Przy projektowaniu systemu nagród, zdecydowano, aby nie zniechęcać agenta do prób oddawania strzałów. Dlatego zastosowano niewielkie wartości kar za trafianie w otaczające ściany. Należy też dodać, że agent nie otrzymuje przedstawionych nagród i kar natychmiastowo. Co oznacza, że nie następuje to od razu po podjęciu korzystnej akcji, lecz dopiero po wielu kolejnych krokach, w których również podejmowane są decyzje. Najdłuższa sekwencja sprzyjających działań agenta wymagana jest w najtrudniejszym wariancie środowiska, co wynika z dodatkowej konieczności naciągnięcia cięciwy łuku w celu oddania strzału.

Dobór hiperparametrów uczenia został przeprowadzony w sposób eksperymentalny, ręcznie regulując ich wartości i obserwując rezultaty uczenia. W badaniu wykorzystano następnie te parametry, dla których uczenia osiągnęło najwyższy wynik skumulowanej wartości nagrody, przy zachowaniu stabilności uczenia. W celu oceny skuteczności testowanych hiperparametrów wykorzystano dodatkowo funkcję strat polityki, która pozwoliła określić jak bardzo w trakcie uczenia zmieniała się polityka agenta, a także wyznaczyć odpowiedni moment zakończenia treningu. Maksymalną liczbę kroków uczenia ustalono na podstawie najmniejszej wartości wspólnej dla obu algorytmów, dla której nastąpiło ustalenie się maksymalnej skumulowanej wartości nagrody. Zakres testowanych wartości hiperparametrów dostosowano na podstawie zaleceń umieszczonych w dokumentacji projektu ML-Agents Toolkit [16]. Najistotniejsze hiperparametry zastosowane w badaniu zebrano w tabelach 1 (PPO) oraz 2 (SAC). Ich nazwy pozostawiono zgodnie z definicją pliku konfiguracyjnego *yaml*. W przypadku algorytmu GAIL zastosowano stałą uczenia (parametr *learning_rate*) równą 0,0006. Wagę sygnału wzmocnienia (parametr *strength*) ustawiono na 0,01, co pozwoliło zapobiec bezpośredniemu kopiowaniu przez agenta zachowania przedstawionego w demonstracjach.

Trening w przypadku każdego wariantu gry uruchomiono na dziesięciu instancjach środowiska. Zrealizowa-

Tabela 1: Hiperparametry uczenia zastosowane dla algorytmu PPO

	Wariant środowiska		
	1.	2.	3.
learning_rate	0,001	0,0006	
batch_size	32		
buffer_size	5000		12000
beta	0,005		
epsilon	0,2		
lambd	0,95		
num_epoch	3		
normalize	true		
hidden_units	32	64	128
num_layers	2		
max_steps	504000	900000	1008000

Tabela 2: Hiperparametry uczenia zastosowane dla algorytmu SAC

	Wariant środowiska		
	1.	2.	3.
learning_rate	0,0008	0,0006	
batch_size	32		
buffer_size	50000		100000
tau	0,005		
steps_per_update	10		
init_entcoef	0,5		
normalize:	true		
hidden_units	32	64	256
num_layers	2		
max_steps	504000	900000	1008000

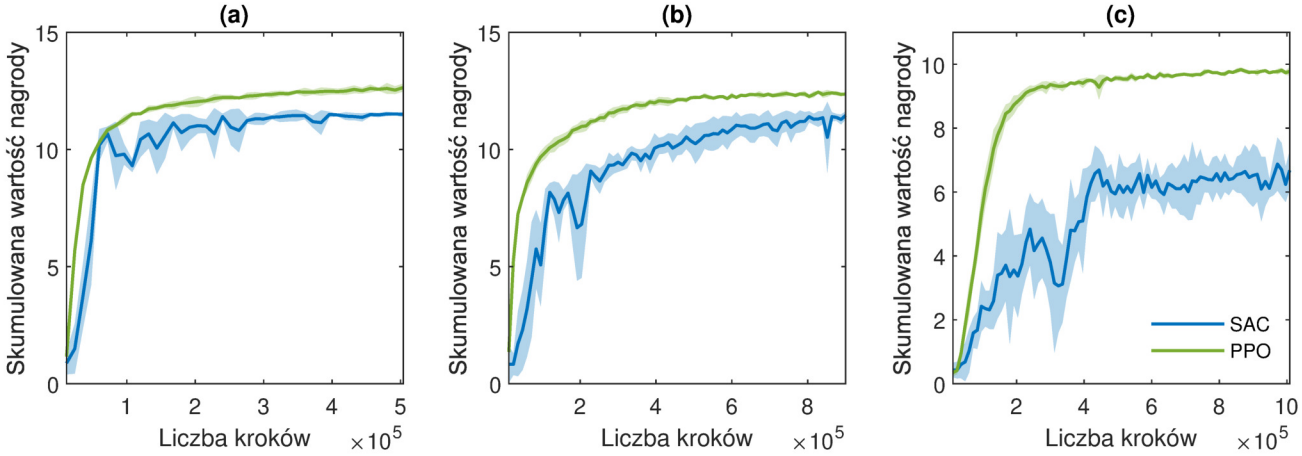
no to poprzez powielenie obiektu agenta oraz środowiska w pojedynczej scenie. Każda z instancji agenta podejmuje niezależne decyzje na podstawie lokalnych obserwacji, ale wykorzystuje i przyczynia się do aktualizowania wspólnej polityki. Zabieg ten wpływa na szybsze zapełnianie bufora doświadczeń, a tym samym na skrócenie czasu oczekiwania na rezultat uczenia.

4. Wyniki

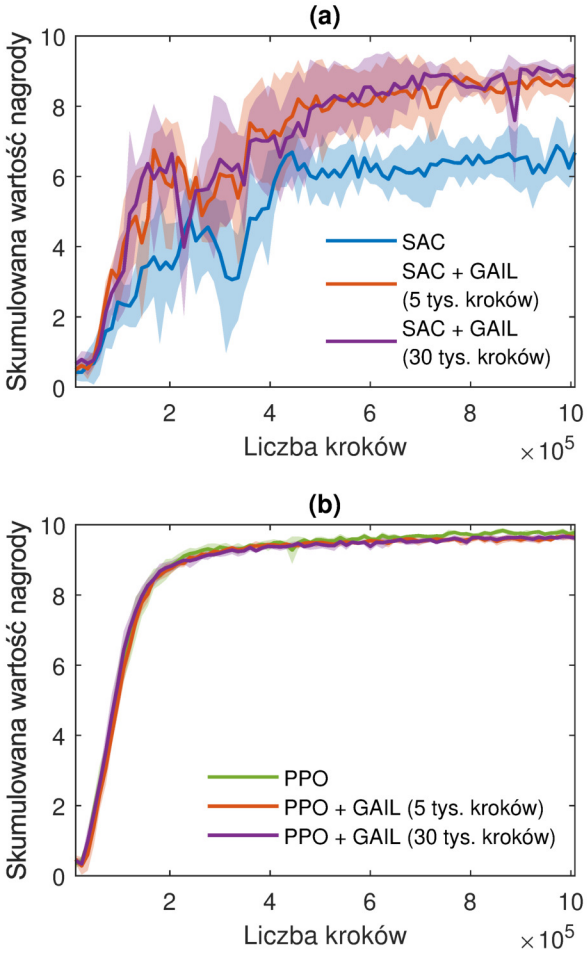
Wyniki badań przedstawiające skumulowaną wartość nagrody zdobywaną przez agenta opracowano na podstawie średniej z pięciu prób pochodzących z niezależnych procesów uczenia z przedziałem ufności 95%. Przebiegi treningów w poszczególnych wariantach środowiska z wykorzystaniem metod PPO oraz SAC przedstawiono na rysunku 4. Na ich podstawie można zaobserwować, że algorytm PPO uzyskał bardzo dobry rezultat w każdym z testowanych środowisk, czyniąc duże podstępy już na samym początku uczenia. Chociaż dalszy trening przebiegał mniej intensywnie, skumulowana wartość nagrody rosła konsekwentnie, aby osiągnąć swoją największą wartość w pobliżu maksymalnej liczby kroków. Wynik ten był też za każdym razem wyższy niż w przypadku uczenia z wykorzystaniem SAC. Algorytm PPO okazał się też bardzo stabilny. W przypadku uczenia z zastosowaniem SAC, zauważyć można zdecydowanie większe wahania wartości nagrody, które stają się coraz bardziej widoczne wraz ze wzrostem złożoności środowiska. Największe różnice pomiędzy algorytmami są widoczne w przypadku najtrudniejszego trybu gry, w którym występują zarówno największe dysproporcje skumulowanej wartości nagrody pomiędzy dwoma algorytmami, jak i bardziej widoczna niestabilność algorytmu SAC.

Przebiegi uczenia algorytmów PPO oraz SAC wspomaganych GAIL w trzecim wariancie środowiska zebrano na rysunku 5. Rozgrywka trwająca 8 minut odpowiada 5 tysiącom kroków demonstracji, z kolei ta o długości 48 minut dostarczyła 30 tysięcy przykładów. Dodatkowe zastosowanie uczenia przez naśladowanie pozwoliło znacząco poprawić wynik uczenia w przypadku algorytmu SAC. Maksymalna skumulowana wartość nagrody zwiększyła się z 6,5 do zakresu w okolicy 8,5. Pomimo że zastosowanie GAIL nie wpłynęło na stabilizację przebiegu, zyskał on bardziej wzrostową tendencję, gdy samodzielny algorytm SAC posiadał skłonność do zahamowania wzrostu skumulowanej wartości nagrody już w połowie treningu. Warto też zauważyć, że zastosowanie GAIL mogłoby pozwolić na otrzymanie lepszego rezultatu w mniejszej liczbie kroków. Pomimo pozytywnego wpływu wykorzystania GAIL w połączeniu z SAC, nie jest to już widoczne w przypadku algorytmu PPO. Przebiegi uczenia samodzielnej metody PPO, jak i tych łączonych z GAIL, okazały się być do siebie bardzo podobne, co sprawia, że trudno jest wskazać poprawę wyniku. Zastosowanie większej liczby demonstracji nie wpłynęło na skuteczność uczenia w przypadku algorytmu PPO. Niewielką różnicę można zaś zaobserwować w przebiegu uczenia SAC przy wykorzystaniu 30 tysięcy przykładów dla GAIL, w którym nieznacznie zwiększyła się maksymalna skumulowana wartość nagrody (o mniej niż ok. 0,5).

Wyniki rozgrywki agentów z modelami wytrenowanymi za pomocą algorytmów PPO oraz SAC w poszczególnych wariantach środowiska przedstawiono na rysunku 6. Rezultat poszczególnych algorytmów otrzymano na podstawie 500 rozgrywek w danym trybie gry. Pomimo słabszego przebiegu uczenia algorytmu SAC, agenci z wynikowym modelem zachowania osiągnęli bardzo wyrównany rezultat w stosunku do PPO w przypadku pierwszego wariantu gry, a nawet zdobywali średnio prawie dwa punkty więcej. Różnice zaczęły się jednak pojawiać w drugim wariancie środowiska, w którym SAC posiadał przewagę jedynie w postaci krótszego średniego czasu ukończenia zadania. Algorytm PPO z kolei osiągnął wyższą średnią celność strzału, jak i większą średnią liczbę zdobytych punktów. Największe dysproporcje pomiędzy dwiema badanymi metodami można zaobserwować jednak w przypadku najtrudniejszego wariantu środowiska. Agenci z modelami wytrenowanymi za pomocą SAC kończyli zadanie średnie o prawie 4 sekundy później, a także posiadali niską celność, trafiając w cel tylko w 80% strzałów. Algorytm PPO okazał



Rysunek 4: Przebieg uczenia algorytmów SAC oraz PPO w poszczególnych wariantach środowiska: (a) pierwszym, (b) drugim, (c) trzecim.



Rysunek 5: Przebieg uczenia algorytmów (a) SAC oraz (b) PPO, wspomaganych GAIL dla różnej liczby wykorzystanych demonstracji.

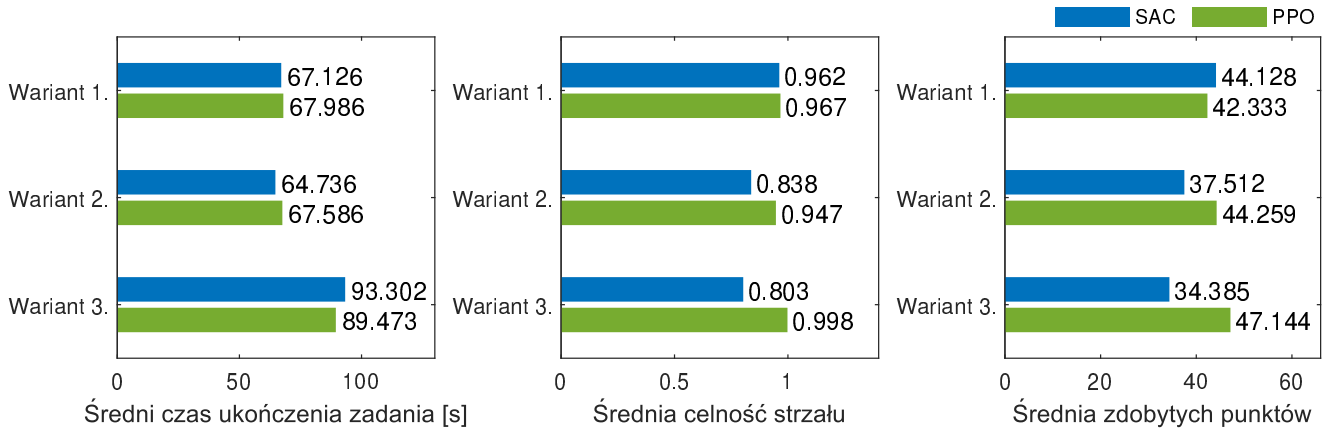
się bardzo skuteczny, uzyskując niemal idealną celność (0,998), a także zdobywając wysoką średnią liczbę punktów (47,144). Metoda SAC uzyskała w tym czasie średnio jedynie 34,385 punktów.

Wyniki rozgrywki agentów z zachowaniem nauczonym z wykorzystaniem łączenia metod PPO oraz SAC z GAIL w trzecim wariancie środowiska przedstawiono w tabeli 3. Podobnie jak w przypadku samego przebiegu uczenia, tak i wynik pochodzący z gry agentów nie zyskał wyraźnych różnic po zastosowaniu GAIL z PPO. Wykorzystanie demonstracji przyniosły jednak poprawę rezultatów w przypadku SAC w każdej z ocenianych kategorii, skracając czas ukończenia zadania, a także zwiększając celność i liczbę zdobytych punktów. Nie wystarczyło to jednak na wyrównanie do wyników osiąganych przez PPO.

Tabela 3: Wyniki rozgrywki agentów w trzecim wariancie środowiska trenowanych za pomocą samodzielnych algorytmów PPO i SAC oraz tych samych metod łączonych z GAIL dla zastosowanej różnej liczby demonstracji

	Średni czas ukończenia [s]		Średnia celność strzału		Średnia suma punktów	
	PPO	SAC	PPO	SAC	PPO	SAC
Bez GAIL	89,473	93,302	0,998	0,803	47,144	34,385
GAIL (5000 kroków)	90,557	92,118	0,995	0,958	47,162	43,824
GAIL (30000 kroków)	90,577	90,568	0,995	0,961	47,043	44,294

Średni czas trwania treningu z wykorzystaniem algorytmów PPO, SAC oraz tych metod łączonych z GAIL zebrano w tabeli 4. Jak można zauważyć, czas ten wydłużał się wraz ze wzrostem złożoności środowiska, czego przyczyną była konieczność stosowania coraz to większej sieci neuronowej, jak i stopniowego zwiększania liczby kroków uczenia. Algorytm PPO charakteryzował się krótszym czasem treningu niż SAC w każdym z wariantów gry. Różnica ta rosła wraz ze zwięk-



Rysunek 6: Wyniki rozgrywki w poszczególnych wariantach środowiska uzyskane przez agentów trenowanych za pomocą algorytmów SAC i PPO.

szaniem trudności zadania wykonywanego przez agenta. W najtrudniejszym wariancie środowiska SAC potrzebował średnio około 6 minut więcej na ukończenie uczenia w przewidzianej liczbie kroków. W przypadku PPO łączonego z GAIL dla 5 tysięcy kroków czas uczenia został wydłużony o prawie 19%. Zwiększenie liczby przykładów wpłynęło zaś na wzrost tego czasu o blisko 22%. Większą różnicę można zaobserwować jednak przy zastosowaniu GAIL z SAC. Dla małej liczby demonstracji średni czas uczenia zwiększył się o 36%, wydłużając trening z 37,5 do około 51 minut. Wykorzystanie przykładów odpowiadających 30 tysiącom kroków poskutkowało wzrostem czasu uczenia do niewielu ponad 50 minut.

Tabela 4: Średnia oraz odchylenie standardowe czasu uczenia algorytmów PPO oraz SAC w trzecim wariancie środowiska oraz wynik tych samych algorytmów łączonych z metodą GAIL

	PPO [s]	σ	SAC [s]	σ
Wariant 1.	927,6	26,1	968,2	37,2
Wariant 2.	1719,4	14,6	1789,2	33
Wariant 3.	1881,4	19,9	2250,6	70,6
Wariant 3. z GAIL (5000 kroków)	2233	27,6	3065,4	70,7
Wariant 3. z GAIL (30000 kroków)	2300,8	32,9	3003,6	37,7

Przy podsumowaniu rezultatów badania, należy podkreślić istotny wpływ wyboru hiperparametrów na wyniki uczenia. Metodę ręcznej ich regulacji nie można uznać za najskuteczniejszą, a bardziej wynikającą z konieczności. Pakiet ML-Agents Toolkit nie posiada, jak dotąd, narzędzi umożliwiających automatyzację tego procesu. Zawodność ręcznego doboru hiperparametrów mogła mieć tym bardziej istotny wpływ na wynik badania ze względu na większe odchylenia przebiegów uczenia z SAC. Mogło to wpłynąć na niekorzystną interpretację wyników podczas doboru konfiguracji parametrów wykorzystanych we właściwym badaniu. Dodatkowym czynnikiem wpływającym na znaczenie sposobu wyboru hiperparametrów, a wskazującym konieczność stosowania metod zautomatyzowanych, jest zaobserwowana czułość algorytmu SAC na zmianę ich wartości. W ostatecznej ocenie wyników badania, należy więc uwzględniać możliwość istnienia bardziej odpowiednich hiperparametrów, których wykorzystanie mogłoby potencjalnie przynieść poprawę skuteczności metody SAC.

5. Wnioski

Przeprowadzone badanie pozwoliło dokonać porównania dwóch współczesnych algorytmów uczenia ze wzmocnieniem — PPO oraz SAC, a także weryfikacji możliwości poprawy skuteczności uczenia poprzez łączenie tych metod z algorytmem uczenia przez naśladowanie GAIL. Zaprojektowane w tym celu środowisko powstało przy użyciu platformy Unity oraz biblioteki Unity ML-Agents Toolkit. Składało się ono z kilku wariantów, sprawdzających skuteczność algorytmów w zależności od stopnia złożoności prezentowanego zadania. Celem agenta w utworzonej grze było strzelanie z łuku do tarczy treningowej, przy czym poszczególne tryby rozgrywki stawiały przed agentem większe wymagania. Algorytmy PPO i SAC zostały porównane na podstawie przebiegu uczenia, jak i skuteczności nauczonego zachowania. W przypadku sprawdzenia możliwości łączenia uczenia ze wzmocnieniem z metodą GAIL zbadano dodatkowo wpływ liczby demonstracji na wynik uczenia.

Na podstawie wyników przeprowadzonego badania, algorytm PPO okazał się skuteczniejszy od SAC. Pomimo wyrównanych rezultatów obu metod w przypadku środowiska o najniższym poziomie trudności zadania, znaczące różnice pomiędzy algorytmami zaczęły występować w dwóch trudniejszych wersjach przygotowanej gry. Agenci wytrenowani za pomocą PPO osiągali znacznie wyższy rezultat podczas rozgrywki, zwłaszcza w najbardziej złożonym wariancie środowiska. Ana-

liza przebiegu procesu uczenia pozwoliła zaobserwować znacznie większą niestabilność skumulowanej wartości nagrody w przypadku algorytmu SAC, występującą najsilniej w najtrudniejszym trybie rozgrywki. Analiza czasu uczenia wykazała, że PPO był nieznacznie szybszy w przypadku dwóch pierwszych wariantów środowiska. Okazał się on jednak o wiele wydajniejszy czasowo w najtrudniejszym trybie rozgrywki, ale przyczyną takiego rezultatu była najprawdopodobniej konieczność zastosowania większej sieci neuronowej dla SAC.

Uzyskane wyniki badania pozwoliły potwierdzić hipotezę o możliwości poprawy rezultatu treningu poprzez łączenie algorytmów uczenia ze wzmocnieniem z GAIL. Efekt ten udało się osiągnąć w przypadku SAC, dla którego otrzymano znacznie lepsze wyniki zarówno przebiegu, jak i rozgrywki, przy zastosowaniu jedynie kilkuminutowej demonstracji. Poprawa rezultatów nie była jednak wystarczająca, aby osiągnąć skuteczność porównywalną z algorytmem PPO. Zaobserwowano też, że zastosowanie GAIL znacznie wydłużyło czas uczenia. W przypadku PPO nie udało się zaobserwować istotnej zmiany wyników treningu, ale należy wziąć pod uwagę to, że już samodzielny algorytm charakteryzował się dużą skutecznością, a zastosowane przykłady odpowiadały zachowaniu suboptymalnemu. Wykorzystanie większej liczby demonstracji nie przyniosło istotnej poprawy wyników treningu w przypadku obu algorytmów.

Zebrane rezultaty badania pozwoliły wskazać, że algorytm PPO jest metodą bardziej uniwersalną, zdolną sprawdzić się w większości środowisk, posiadającą przy tym sporą stabilność i skuteczność uczenia. Poprzez ręczny dobór hiperparametrów zauważono, że jest to metoda prosta w konfiguracji, znacznie mniej wrażliwa na zmiany wartości hiperparametrów niż SAC. Wniosek ten może mieć większe znaczenie biorąc pod uwagę przeznaczenie wykorzystanych narzędzi, czyli platformy Unity i pakietu ML-Agents Toolkit, które są dedykowane dla szerszej grupy użytkowników, nie tylko badaczy, ale i programistów. Dodatkową przeszkodą w przeprowadzeniu skutecznego uczenia z wykorzystaniem SAC może być brak dostępnych w Unity rozwiązań pozwalających na automatyzację poszukiwania najbardziej korzystnych wartości hiperparametrów. Możliwe jest, że wybór parametrów posiadał istotny wpływ również na wynik przeprowadzonego badania i istnieje konfiguracja, która przyniosłaby lepszy rezultat, zwłaszcza w przypadku algorytmu SAC. Okazało się też, że niezadowalający wynik uczenia można skutecznie poprawić wykorzystując udostępniany przez pakiet algorytm uczenia przez naśladowanie GAIL. Pamiętać jednak należy, że wybór tej metody wiąże się z koniecznością dodatkowej rejestracji demonstracji oraz możliwym dłuższym czasem uczenia.

Zaprezentowane w niniejszej pracy badanie może być początkiem dalszych eksperymentów, wykorzystujących bardziej skomplikowane środowiska, o większym zapotrzebowaniu zasobów sprzętowych, czy czasu uczenia. Ciekawą hipotezą, wartą weryfikacji może być to, że algorytm SAC mógłby sprawdzić się lepiej w bardzo powolnym środowisku. Dalsze badania mogą dotyczyć również pozostałych możliwości oferowanych przez pakiet ML-Agents Toolkit, czego przykładem może być moduł *Curiosity*, zalecany w środowiskach z rzadkimi nagrodami, czy zastosowanie rekurencyjnej sieci neuronowej, która może być wykorzystywana jako pamięć agentów.

Literatura

[1] A. Juliani, V. P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, D. Lange, Unity: A General Platform for Intelligent Agents, arXiv preprint arXiv:1809.02627v2 (2020).

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[3] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, In Proceedings of Machine Learning Research 80 (2018) 1861–1870.

[4] J. Ho, S. Ermon, Generative Adversarial Imitation Learning, Advances in neural information processing systems (2016) 4565–4573.

[5] A. Hussein, M. M. Gaber, E. Elyan, C. Jayne, Imitation Learning: A Survey of Learning Methods, ACM Computing Surveys 50(2) (2017) 1–35, https://doi.org/10.1145/3054912.

[6] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, Cambridge, 2018.

[7] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, In International conference on machine learning (2015) 1889–1897.

[8] M. Urmanov, M. Alimanova, A. Nurkey, Training Unity Machine Learning Agents using reinforcement learning method, In 2019 15th International Conference on Electronics, Computer and Computation (2019) 1–4, https://doi.org/10.1109/ICECCO48375.2019.9043194.

[9] M. Pleines, F. Zimmer, V. Berges, Action Spaces in Deep Reinforcement Learning to Mimic Human Input Devices, In 2019 IEEE Conference on Games (2019) 1–8, https://dx.doi.org/10.1109/CIG.2019.8848080.

[10] V. Mnih, K. Kavukcuoglu, D. Silver et al., Human-level control through deep reinforcement learning, Nature 518(7540) (2015) 529–533.

[11] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The Arcade Learning Environment: An Evaluation Platform for General Agents, Journal of Artificial Intelligence Research 47 (2013) 253–279.

[12] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, C. Blundell, Agent57: Outperforming the Atari Human Benchmark, In International Conference on Machine Learning (2020) 507–517.

[13] A. Defazio, T. Graepel, A Comparison of learning algorithms on the Arcade Learning Environment, arXiv preprint arXiv:1410.8620 (2014).

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI Gym, arXiv preprint arXiv:1606.01540 (2016).

[15] A. Tavakoli, F. Pardo, P. Kormushev, Action branching architectures for deep reinforcement learning, In Proceedings of the AAAI Conference on Artificial Intelligence 32(1) (2018).

[16] Dokumentacja biblioteki ML-Agents Toolkit — opis i zalecany zakres wartości hiperparametrów uczenia, https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md, [04.05.2021].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 205–209
Received: 16 June 2021
Accepted: 29 June 2021

Comparative analysis of the Angular 10 and Vue 3.0 frameworks

Analiza porównawcza szkieletów programistycznych Angular 10 i Vue 3.0

Jarosław Kyć*, Piotr Lipski*, Beata Pańczyk

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

Abstract

The aim of this article is to perform a comparative analysis of the Angular v10 and Vue v3.0 frameworks. The basis of the comparison is the performance tested with two applications that are similar in terms of functionality. The view with a variable number of displayed elements was examined, and the time was measured from the moment the number of components was indicated to the end of rendering. The amount of disk space occupied by the final applications and application segments was also compared in relation to the method of implementing their functionality. The results of the research allowed to formulate the conclusions that Vue is more efficient than Angular and additionally the Vue application takes up less disk space.

Keywords: Angular; Vue.js; efficiency; comparison

Streszczenie

Celem artykułu jest przeprowadzenie analizy porównawczej dwóch szkieletów programistycznych Angular w wersji 10 oraz Vue w wersji 3.0. Podstawą porównania jest wydajność zbadana za pomocą dwóch podobnych co do funkcjonalności aplikacji. Badaniu poddano widok ze zmienną liczbą wyświetlanych elementów, a czas mierzono od momentu wskazania liczby komponentów do zakończenia renderowania. Porównano także ilość przestrzeni dyskowej jaką zajmują finalne aplikacje oraz modułów aplikacji względem sposobu realizacji ich funkcjonalności. Wyniki badań pozwoliły sformułować wnioski, że Vue jest bardziej wydajny od Angulara a dodatkowo aplikacja Vue zajmuje mniej przestrzeni dyskowej.

Słowa kluczowe: Angular; Vue.js; wydajność; porównanie

*Corresponding author

Email address: jaroslaw.kyc@pollub.edu.pl (J. Kyć), piotr.lipski2@pollub.edu.pl (P. Lipski)

1. Wstęp

Środowisko przeglądarek i języki programowania zorientowane na aplikacje internetowe rozwijały się na przestrzeni lat. Szkielet strony internetowej w postaci plików HTML w połączeniu z kaskadowymi arkuszami stylów i językiem JavaScript dają coraz więcej możliwości. Dzięki rozwojowi technologii aplikacje już nie tylko przedstawiają treści, ale wchodzą w interakcje z użytkownikiem.

Aplikacje internetowe można podzielić ze względu na wielkość. Z reguły mniejsze aplikacje są aplikacjami statycznymi [1]. Mniejsze aplikacje, które nie posiadają skomplikowanej logiki mogą zostać zbudowane od podstaw przy użyciu czystego kodu JavaScript, plików HTML oraz arkuszy stylów. Przy rozbudowanych projektach bardziej opłaca się skorzystać z pomocy szkieletów programistycznych.

Korzystanie z gotowych szkieletów stało się powszechną praktyką w branży IT. Szkielety znacząco ułatwiają i przyspieszają początkowe prace nad projektem. Oczywiście z każdym nawet najprostszym szkieletem programistycznym należy się zaznajomić, poznać jego budowę i zasady działania. Należy jednak zastanowić się jak taki pakiet gotowych rozwiązań wpływa na proces tworzenia aplikacji i jej końcowy wynik. Czym różnią się szkielety programistyczne i którego użyć do danego projektu?

Technologie rozwijają się w bardzo szybkim tempie i pojawiają się coraz to nowsze ich wersje Nasuwa się jednak pytanie jak pokrewne rozwiązania wypadają na tle wydajności oraz procesu tworzenia aplikacji internetowych.

2. Cel i zakres badań

Celem badań przeprowadzonych w niniejszym artykule jest analiza wydajności szkieletów programistycznych Vue.js w wersji 3.0 oraz Angular w wersji 10.0 i wybranie korzystniejszego rozwiązania. W tym celu stworzona została identyczna aplikacja internetowa za pomocą obydwu szkieletów programistycznych. Porównanie wydajności będzie polegać na zmierzeniu czasu rysowania elementów od momentu wyboru ich liczby, aż do zakończenia rysowania. Zostanie to zmierzone za pomocą narzędzi deweloperskich dostępnych w przeglądarce Google Chrome i Firefox.

W artykule postawiono następującą tezę badawczą: Szkielet programistyczny Vue.js jest bardziej wydajny niż Angular.

3. Obiekty badań

Testowane szkielety programistyczne zostały wybrane ze względu na swoją popularność. Są to dwa szkielety programistyczne, które plasują się w ścisłej czołówce popularności.

Wedle ankiety [2] przeprowadzonej na jednym z najbardziej znanych serwisów społecznościowych programistów StackOverflow - Vue i Angular znajdują się wśród czołówki najczęściej używanych sieciowych szkieletów programistycznych przez deweloperów z całego świata.

Zarówno Vue.js jak i Angular są potężnymi narzędziami zdolnymi do tworzenia zaawansowanych stron typu SPA (ang. Single Page Application). Oba rozwiązania stosują podobny styl pisania komponentów, które można użyć w różnych kontekstach. Jednak mimo podobieństw mają kilka znaczących różnic.

3.1. Vue

Vue to progresywny szkielet programistyczny do budowania interfejsów użytkownika. W przeciwieństwie do innych monolitycznych szkieletów programistycznych, Vue jest projektowany od podstaw tak, aby można go było stopniowo adaptować. Podstawowa biblioteka skupia się tylko na warstwie widoku i jest łatwa do pobrania i zintegrowania z innymi bibliotekami lub istniejącymi projektami. Z drugiej strony, Vue jest również przystosowany do zasilania zaawansowanych jednostronicowych aplikacji, gdy jest używany w połączeniu ze wspierającymi bibliotekami [3].

3.2. Angular

Angular jest szkieletem programistycznym do projektowania aplikacji i platformą programistyczną do tworzenia wydajnych i zaawansowanych aplikacji typu SPA. Angular w przeciwieństwie do Vue i Reacta jest rozwiązaniem kompletnym i nie polega na zewnętrznych bibliotekach. Implementuje on podstawowe i opcjonalne funkcje jako zestaw bibliotek TypeScript, które są importowane do aplikacji. Podstawowymi elementami konstrukcyjnymi szkieletu programistycznego Angular są komponenty, które są zorganizowane w moduły. Moduły zbierają powiązany kod do zestawów funkcyjnych. Aplikacja Angular jest zdefiniowana przez zestaw modułów. Zawsze posiada co najmniej główny moduł i zazwyczaj wiele innych modułów funkcjonalnych [4].

3.3. Podobieństwa i różnice

W tabeli 1 porównano cechy obu szkieletów programistycznych [5, 6].

Tabela 1: Porównanie frameworków Vue i Angular

	Angular	Vue.js
Podobieństwa	Wykorzystanie podwójnych nawiasów klamrowych do interpolacji kodu w szablonach.	
	Szkielety posiadają zaawansowane narzędzie cli służące do generacji kodu/projektu.	
	Analogiczny jest sposób korzystania z techniki data binding (np. v-bind vs ng-bind).	
Różnice	Modyfikacje realizowane na rzeczywistym obiekcie DOM.	Wykorzystanie virtual-DOM do modyfikacji dokumentu.
	Wykorzystuje komponenty klasowe.	Wsparcie dla komponentów obiektowych oraz funkcyjnych.
	Konieczność korzystania z języka TypeScript,	Dowolność wyboru między językiem
	który jest nadzbiorem JavaScript.	JavaScript i TypeScript
	Konieczność korzystania z rxjs.	Szerokie wsparcie dla zewnętrznych bibliotek.

4. Przegląd literatury

W artykule [7] zostało przeprowadzone badanie szkieletów Vue.js w wersji 2 oraz Angular 6 pod kątem wydajności wyrażonej jako czas wysyłania żądania i renderowania komponentów. Autorzy stwierdzili, że szkielet programistyczny Vue.js lepiej nadaje się dla początkującego programisty niż Angular, ponieważ łatwiej jest się go nauczyć. Dodatkowym wymienionym powodem jest wydajność przemawiająca za szkieletem programistycznym Vue.js pod warunkiem, że tworzona aplikacja ma średnią złożoność. W podsumowaniu autorzy doszli do wniosku, że na podstawie przeprowadzonych badań szybkość pracy obu szkieletów programistycznych jest do siebie zbliżona.

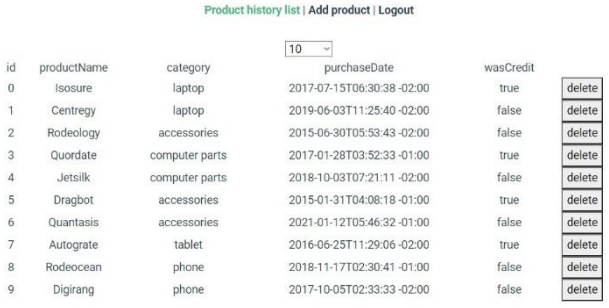
Artykuł [8] dotyczy szkieletów aplikacyjnych Vue 2, Angular 4 i React 16 oraz metodyki tworzenia jednostronicowych SPA i wielostronicowych MPA (ang. Multi Page Application) aplikacji internetowych. Najlepszy wynik w przeprowadzonych tam badaniach uzyskał Vue (w kategorii SPA oraz MPA), potem React i Angular.

5. Metoda badań

Na bazie takich samych co do funkcjonalności aplikacji (typu SPA) zaimplementowanych w obu szkieletach programistycznych, przeprowadzono badanie wydajności.

Podstawą porównania był sposób przygotowania aplikacji, implementacja jej komponentów w odniesieniu do dwóch szkieletów oraz finalna wydajność, czyli czas renderowania głównego komponentu.

Obiektem prowadzonych badań był widok z tabelą (Rys. 1), która zawierała szereg danych do wyświetlenia. Na stronie znajduje się rozwijana lista z możliwością wyboru: 10, 100, 1000 i 10000 wierszy do wyświetlenia w tabeli. Zmierzono czasy między kliknięciem w konkretną liczbę z listy, a momentem wyrenderowania danych w tabeli. Dla każdej opcji z listy, testy powtórzono po 10 razy.



Rysunek 1: Interfejs graficzny aplikacji testowych.

Aplikacje zostały uruchomione na dwóch przeglądarkach: Google Chrome w wersji 90.0.4430.93 (64-

bitowa) i Firefox Developer w wersji 89.0b6 (64-bitowa). Przeglądarki zostały wybrane ze względu na różne silniki renderowania co mogło wpływać na wyniki. Obie przeglądarki posiadają własne narzędzia deweloperskie. Google Chrome został zbudowany na podstawie projektu chromium, który korzysta z silnika blink [9]. Na jego podstawie zaimplementowane są także takie przeglądarki jak Opera, Edge, czy Vivaldi. Firefox z kolei korzysta z autorskiego silnika Gecko [10].

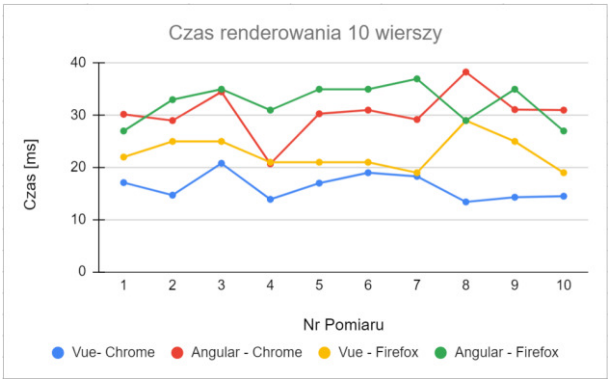
6. Platforma testowa

Do uruchomienia aplikacji został użyty komputer klasy PC z następującą specyfikacją:

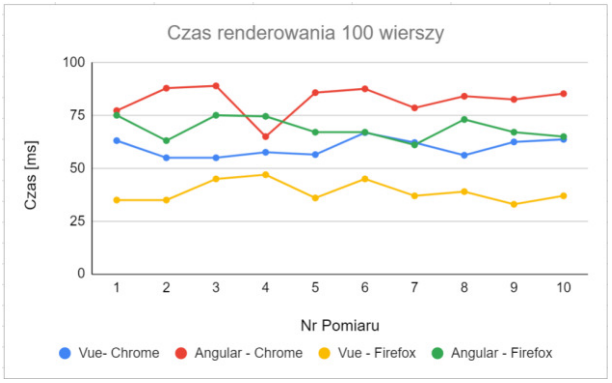
- procesor: AMD Ryzen 5 1600 AF 3.2GHz, 6 rdzeni, 12 wątków,
- pamięć: 16384MB DDR4 3000MHz,
- dysk: SSD 512GB M.2,
- system operacyjny: Windows 10 Pro,
- rozdzielczość ekran: 1920x1080px,
- układ graficzny: NVIDIA GeForce RTX 2060.

7. Wyniki

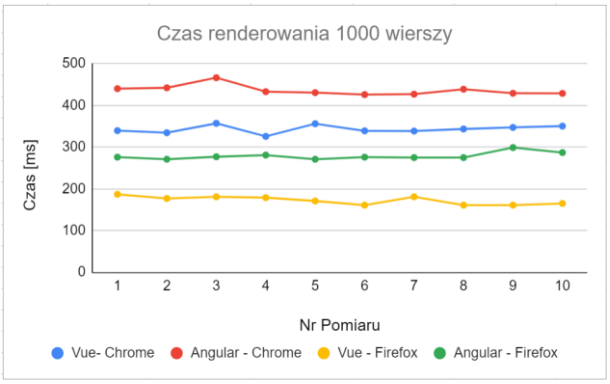
Wyniki przeprowadzonych testów przedstawiono na rysunkach 2-5.



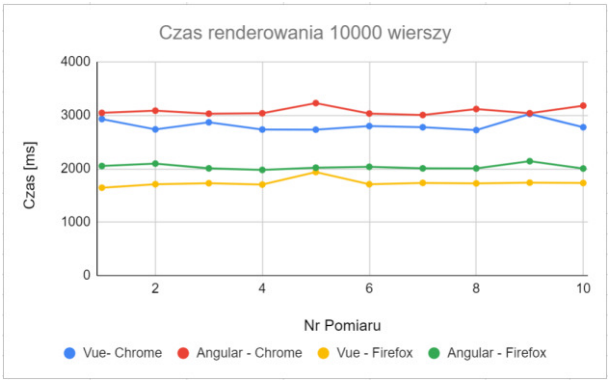
Rysunek 2: Wyniki pomiarów dla 10 wierszy.



Rysunek 3: Wyniki pomiarów dla 100 wierszy.

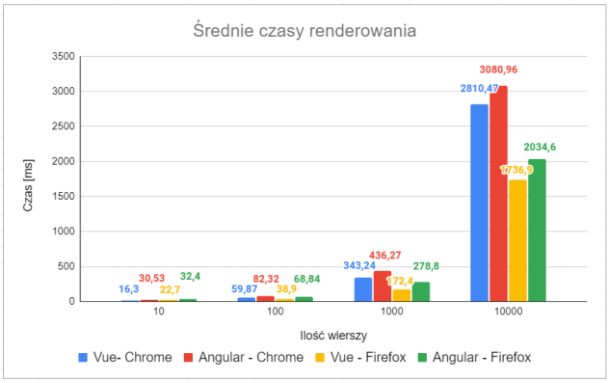


Rysunek 4: Wyniki pomiarów dla 1000 wierszy.



Rysunek 5: Wyniki pomiarów dla 10000 wierszy.

Średnie czasy przedstawia rysunek 6.



Rysunek 6: Średnie czasy pomiarów.

8. Analiza

Wyniki dla 10 wierszy (Rys. 2) przedstawiają niższe czasy renderowania dla Vue niż w przypadku Angular. Takie wyniki otrzymano zarówno w przeglądarce Google Chrome jak i Firefox. Wyniki pojedynczych pomiarów wykazywały największe wahania i odstępstwa od średniej właśnie dla wyświetlania 10 wierszy.

Najniższe czasy renderowania dla 100 wierszy (Rys. 3) uzyskał ponownie Vue, jednak najlepszy czas odnotowano tym razem dla przeglądarki Firefox (55ms). Angular w najlepszym pomiarze potrzebował 61ms. Wahania pojedynczych pomiarów zmalały względem pomiarów dla 10 wierszy.

W przypadku 1000 wierszy (Rys. 4) można zaobserwować wzrost przewagi aplikacji Vue nad Angular.

Najniższe czasy uzyskał Vue w przeglądarce Firefox (161ms). Dla porównania najniższy czas Angular osiągnął również w przeglądarce Firefox 271ms. Jest to aż 160% czasu uzyskanego przez Vue.

Dla 10000 wierszy (Rys. 5) wyniki są zbliżone do przypadku z 1000 wierszy. Najkrótszy czas uzyskał Vue w Firefox - 1645ms w jednym pomiarze. W przypadku aplikacji Angular najlepszy wynik to 1978 ms. Przewaga Vue jest widoczna na obydwu przeglądarkach.

Wszystkie wyniki przedstawiają dość spójny obraz wydajności porównywanych szkieletów programistycznych. W każdym przypadku to Vue jest szybszy w renderowania widoku z tabelą. Dodatkowo widać, że Vue uzyskał lepsze wyniki w przeglądarce Firefox.

W tabeli 2 porównano rozmiary aplikacji i czasy ich kompilacji.

Tabela 2: Porównanie rozmiarów aplikacji

	Angular	**Vue**
Rozmiar aplikacji	482 MB	239 MB
Czas kompilacji	26539 ms	2729 ms

9. Wnioski

Oba badane szkielety programistyczne są zaawansowanymi narzędziami tworzenia aplikacji typu SPA. Wykazują podobieństwa i różnice, które przedstawiono w tabeli 1.

Pod względem czasu renderowania się komponentów to Vue okazał się lepszy niż Angular (Rys. 2-5). Wyniki dla dużej liczby wierszy, przedstawione w niniejszym artykule mogą nie znaleźć odzwierciedlenia w rzeczywistym użyciu, gdyż wtedy stosowane są ograniczenia co do maksymalnej liczby wyświetlanych rekordów oraz stronicowanie wyników. Mimo tego dla celów badawczych zostały one poddane testom i również wykazały lepszą wydajność Vue, co potwierdza postawioną tezę badawczą iż Vue.js jest bardziej wydajny niż Angular.

Dodatkowym atutem przemawiającym po stronie Vue.js jest finalny rozmiar aplikacji na dysku. Jest on dwukrotnie mniejszy niż aplikacji Angular. Dla szkieletu programistycznego Vue.js zostały wykorzystane pokrewne biblioteki do gotowych elementów graficznych i należy wziąć to pod uwagę. Wybór różnych bibliotek wynika z tego, że Vue material nie obsługuje wersji Vue 3.0.

Jeśli chodzi o łatwość przyswajania danego szkieletu programistycznego oraz preferencje programistów to Angular może wydawać się bardziej przyjazny programistom części serwerowych aplikacji (back-end) z powodu zastosowania podobnych konwencji programowania, na przykład techniki wstrzykiwania zależności. Często spotykanym zjawiskiem na stanowisku fullstack developer jest programowanie w Angular części klienckiej i programowanie w Spring (Java) w części serwerowej aplikacji.

Z kolei Vue może wydawać się bardziej przystępny dla nowych programistów, bez doświadczenia programistycznego po stronie serwera, ze względu na możliwość pisania w czystym języku JavaScript oraz ze względu na konieczność przyswojenia mniejszej liczby technologii. Z tych powodów wydajność nie powinna być jedynym kryterium wyboru. Różnice rzędu kilkudziesięciu milisekund będą niezauważalne dla użytkownika końcowego, zaś wybór odpowiednich narzędzi dla zespołu programistycznego może znacznie zwiększyć jego komfort oraz szybkość pracy.

W artykule [7] stwierdzono, że wydajności aplikacji Angular i Vue są bardzo zbliżone. Natomiast w wyniku badań prezentowanych w niniejszym artykule Vue.js uzyskał zdecydowanie lepsze wyniki. Wnioski dotyczące przystępności szkieletów programistycznych są natomiast zbliżone do tych przedstawionych w publikacji z artykułu [7].

Różnice w wydajności w stosunku do badań z pozycji [7] mogą być spowodowane konkretną funkcjonalnością aplikacji testowych oraz różnymi wersjami szkieletów programistycznych, na których realizowano testy wydajnościowe.

Literatura

[1] Porównanie stron statycznych z dynamicznymi, https://about.gitlab.com/blog/2016/06/03/ssg-overview-gitlab-pages-part-1-dynamic-x-static/ , [25.06.2021]

[2] Ankieta dotycząca popularnych technologii, https://insights.stackoverflow.com/survey/2020#technology-web-frameworks-all-respondents2 [31.05.2021].

[3] Dokumentacja Vue.js, https://vuejs.org/v2/guide/ , [30.11.2020].

[4] Dokumentacja Angular, https://angular.io/docs , [30.11.2020].

[5] M. Frisbie, Angular 2 Cookbook, Packt Publishing, 2017.

[6] A. Passaglia, Vue.js 2 Cookbook, Packt Publishing, 2017.

[7] R. Baida, M. Andriienko, M. Plechawska-Wójcik, Perfomance analysis of frameworksAngular and Vue.js, Journal of Computer Sciences Institute, 14 (2020) 59-64.

[8] M. Kaluža, K. Troskot, B. Vukelić, Comparison of front-end frameworks for webapplications development. Zbornik Veleučilišta u Rijeci, 2018.

[9] Silnik renderowania Blink, http://www.chromium.org/blink , [25.06.2021].

[10] Silnik renderowania Gecko, https://developer.mozilla.org/en-US/docs/Glossary/Gecko , [25.06.2021].

[11] M.S. Mikowski, J.C. Powell, Single Page Web Applications, Manning Publications, 2013.

[12] TypeScript Notes for Professionals, GoalKicker.com, 2018.

[13] Angular Notes for Professionals, GoalKicker.com, 2018.

[14] JavaScript Notes for Professionals, GoalKicker.com, 2018.

[15] Dokumentacja MDN web docs, https://developer.mozilla.org/en-US/docs/Learn/Performance , [30.11.2020].

[16] Dokumentacja biblioteki Vuex, https://vuex.vuejs.org/ , [30.11.2020].

[17] Dokumentacja biblioteki vue-router, https://router.vuejs.org/ , [30.11.2020].

[18] Dokumentacja biblioteki Ngrx, https://ngrx.io/docs , [30.11.2020].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 210-216
Received: 12 June 2021
Accepted: 6 August 2021

Immersion analysis during gameplay in VR and on a PC

Analiza immersji podczas rozgrywki w wirtualnej rzeczywistości oraz na komputerze stacjonarnym

Karol Jakub Moniuszko*, Tomasz Szymczyk

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

Abstract

This article concerns the study of immersion and human behavior while in a virtual reality (VR) world. It was compared with the behavior of participants during the traditional process of watching the image displayed on a LCD monitor. The study was conducted on the original test stand and the following vital signs were measured: pulse rate and stress. Additional data about the participants were collected by conducting a questionnaire concerning, among others, feelings, and well-being during the game. The obtained results clearly show that VR gameplay provides much more emotions to the participants. It also makes them able to forget that they are in a virtual world regardless of whether they have used a VR generating device before or not. These results determined the impact of immersion on gameplay appeal and player engagement.

Keywords: virtual reality; immersion; VR games; pulse monitoring

Streszczenie

Artykuł dotyczy badania immersji oraz zachowań ludzi podczas przebywania w świecie wirtualnej rzeczywistości (VR). Porównano je z zachowaniami uczestników podczas tradycyjnego procesu oglądania obrazu wyświetlanego na monitorze LCD. Badania przeprowadzono na autorskim stanowisku i wykonano pomiary następujących parametrów życiowych: pulsu oraz stresu. Dodatkowe dane o uczestnikach zostały zebrane poprzez przeprowadzenie ankiety dotyczącej m.in. odczuć czy samopoczucia podczas rozgrywki. Uzyskane wyniki wskazują jednoznacznie, że rozgrywka VR dostarcza znacznie większych emocji uczestnikom. Sprawia również, że potrafią zapomnieć, iż znajdują się w wirtualnym świecie bez względu na to, czy korzystali już wcześniej z urządzenia generującego VR, czy nie. Wyniki te pozwoliły określić wpływ immersji na atrakcyjność rozgrywki i zaangażowanie gracza.

Słowa kluczowe: wirtualna rzeczywistość; immersja, gry VR; badanie pulsu

*Corresponding author

Email address: karol.moniuszko@pollub.edu.pl (K. J. Moniuszko)

1. Wstęp

Rzeczywistość wirtualna (VR) to symulowane środowisko stworzone przy wykorzystaniu technologii komputerowej. W przeciwieństwie do tradycyjnych interfejsów użytkownika, VR umieszcza użytkownika wewnątrz doświadczenia. Zamiast patrzeć na ekran przed nim, zanurzony jest on w wirtualnym świecie i ma możliwość interakcji z trójwymiarowymi światami [1]. Podstawowym założeniem tej technologii jest wywołanie jak najbardziej realnych i naturalnych dla człowieka doznań. Jest to tak zwane zjawisko immersji [2]. Określane jest ono często przez graczy, projektantów czy badaczy jako niezwykłe przeżycie lub zanurzenie. Koncepcja zanurzenia najczęściej używana jest właśnie w odniesieniu do gier w VR. W recenzjach aplikacji często wspomina się o zanurzeniu związanym z realizmem świata gry oraz dźwiękami i muzyką podsycającą atmosferę gry. Stworzenie immersji jest zatem jednym z głównych celów twórców gier i potrafi ono zrujnować lub znacząco uatrakcyjnić rozgrywkę. Poprzez stymulację jak największej ilości zmysłów, takich jak wzrok, słuch, dotyk, a nawet zapach, gra może całkowicie „pochłonąć" użytkownika, przez co może on zapomnieć, że znajduje się w wirtualnym świecie. Obecnie głównym ograniczeniem dla niemal rzeczywistych wrażeń VR jest niedostateczna moc obliczeniowa urządzeń generujących VR [1, 3].

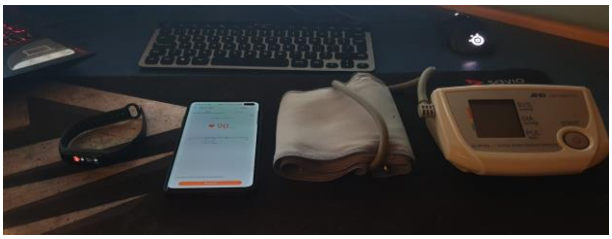
Motywacją do przeprowadzenia badania było zjawisko gwałtownie rozwijającej się technologii VR oraz to, że przyciąga ona coraz większą rzeszę użytkowników na całym świecie. Ikony branży gier twierdzą, że technologia ta wywoła rewolucję w sektorze rozrywkowym, a nawet ma szansę stworzyć silną konkurencję smartfonom [4]. Artykuł pomoże lepiej zrozumieć tę stosunkowo nową technologię oraz to jak zjawisko immersji wpływa na odbiór danej aplikacji przez graczy. Dodatkowo w artykule poruszany jest temat najbardziej naturalnego interfejsu dla użytkownika oraz próba odpowiedzi na pytanie, czy istnieje szansa, że myszka i klawiatura odejdą w zapomnienie już w niedalekiej przyszłości.

2. Metodyka badań

Badanie zostało przeprowadzone na 20 osobach w wieku od 16 do 58 lat, zarówno na kobietach jak i mężczyznach. Badane osoby znajdowały się w pozycji siedzącej, aby zapewnić im jak największe bezpieczeństwo, a ponadto zminimalizować efekt choroby wirtualnej [5].

Rysunek 1: Widok stanowiska badawczego.

W badaniu wykorzystane zostały okulary VR Oculus Quest 2, oraz komputer Lenovo Legion Y720 z kartą graficzną GTX 1060 oraz procesorem Intel i7-7700k.



Rysunek 2: Widok interfejsów służących do pomiaru pulsu oraz stresu.

Do pomiaru pulsu użyto elektronicznego aparatu do pomiaru ciśnienia krwi firmy A&D Medical, Samsunga s10+ oraz opaski Xiaomi Mi Band 5. Do pomiaru stresu wykorzystano dwa ostatnie urządzenia.

Badanie uwzględnia porównanie immersji podczas rozgrywki na VR oraz PC. Podczas badania zostały wzięte pod uwagę: puls, stres oraz potliwość badanych osób, a także ich prywatne odczucia związane z grą, zebrane poprzez ankietę. Miarą pulsu była liczba uderzeń serca na minutę (BPM, ang. beats per minute). Przed pomiarem pulsu każdy z uczestników znajdował się 3 minuty w pozycji siedzącej i starał się uregulować oddech. Następnie został im zmierzony puls trzema różnymi urządzeniami, aby zapewnić jak najdokładniejszy odczyt wybranych funkcji życiowych. Wyniki z opaski oraz telefonu nie odbiegały znacznie od pomiaru ciśnieniomierzem. Błąd względny pomiaru wyniósł 2,5%. Opaska mierzyła ciśnienie badanym osobom co minutę podczas badania, co pozwoliło określić uśredniony oraz maksymalny pomiar. Po każdym z testów zbierano dodatkowy pomiar z telefonu, aby sprawdzić, czy opaska daje nadal porównywalne wyniki. Zanotowane różnice pomiarów z obu urządzeń nie odbiegały od wcześniej wyznaczonego błędu względnego.

W badaniu założono 2 możliwe błędy pomiarowe, związane z niedoskonałością urządzenia do pomiarów, a także błędem ludzkim podczas odczytywania pomiarów z urządzenia w określonym momencie gry. Warto również nadmienić, że ze względu na sytuację pandemiczną panującą na świecie, grupa badawcza musiała zostać mocno ograniczona, co może mieć dodatkowy wpływ na wyciągnięte z badań wnioski.

3. Rezultaty badań

Oprócz głównego celu pracy, czyli zbadania immersji na podstawie pulsu oraz stresu, udało się również sformułować interesujące wnioski dotyczące zaleceń korzystania z VR. W pracy zostały również przedstawione najważniejsze pytania i odpowiedzi z ankiety przeprowadzonej po badaniu.

Rysunek 3: Udokumentowany przebieg badania.

3.1. Pomiar pulsu

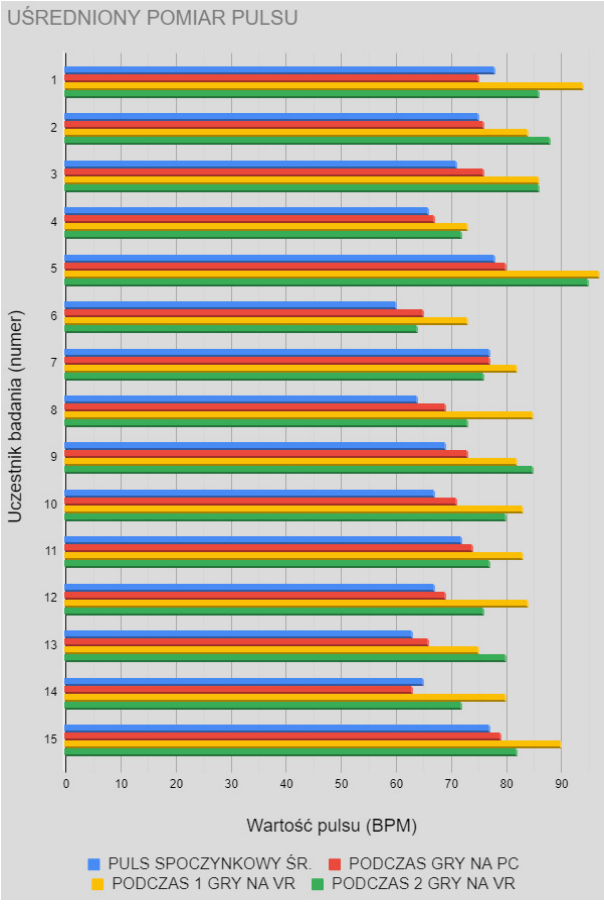
Jak można zauważyć w Tabelach 1 i 2, u każdego z uczestników nastąpił znaczący wzrost pulsu podczas rozgrywki w wirtualnej rzeczywistości. Film wyświetlany na zwykłym monitorze LCD miał znikomy wpływ na wzrost pulsu. Mimo zastosowanych w aplikacji podobnych zabiegów, takich jak jazda tyłem, czy nagły atak na gracza z towarzyszącymi temu dźwiękami. Na te wydarzenia, wszyscy uczestnicy reagowali tylko znajdując się w wirtualnej rzeczywistości. Bardzo jednoznacznie potwierdzają to wyniki pokazane na wykresie (Rysunek 4).

Tabela 1: Uśrednione wyniki pomiaru pulsu

L.p.	Śr. puls spoczynkowy (BPM)	Śr. puls podczas gry na PC (BPM)	Śr. puls podczas 1 gry na VR (BPM)	Śr. puls podczas 2 gry na VR (BPM)
1	78	75	94	86
2	75	76	84	88
3	71	76	86	86
4	66	67	73	72
5	78	80	97	95
6	60	65	73	64
7	77	77	82	76
8	64	69	85	73
9	69	73	82	85
10	67	71	83	80
11	72	74	83	77
12	67	69	84	76
13	63	66	75	80
14	65	63	80	72
15	77	79	90	82
16	67	67	79	78
17	77	80	94	88
18	70	69	81	87
19	69	69	80	82
20	64	71	94	89

Tabela 2: Maksymalne odczyty pulsu badanych osób

L.p.	Maks. puls spoczynkowy (BPM)	Maks. puls podczas gry na PC (BPM)	Maks. puls podczas 1 gry na VR (BPM)	Maks. puls podczas 2 gry na VR (BPM)
1	79	82	99	91
2	75	78	89	90
3	71	76	89	87
4	67	68	75	74
5	82	91	109	99
6	62	67	81	69
7	77	79	89	95
8	65	71	95	82
9	70	79	86	91
10	68	72	87	83
11	73	75	83	79
12	67	71	86	82
13	65	70	79	80
14	66	64	88	76
15	77	82	93	87
16	68	69	83	79
17	80	81	97	92
18	70	70	89	91
19	71	73	87	83
20	68	71	101	95



Rysunek 4: Przedstawienie wyników pomiaru pulsu badanych osób.

3.2. Pomiar stresu

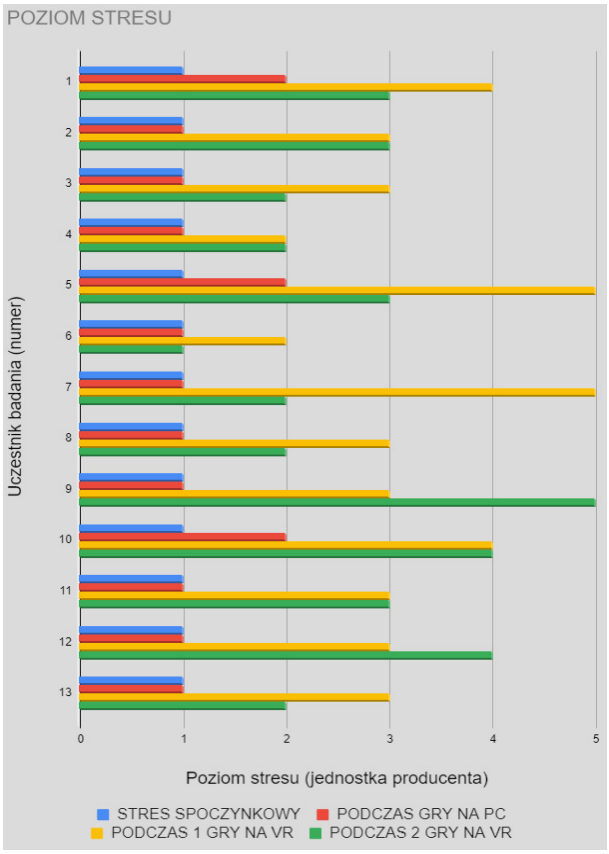
Tabela 3: Uśrednione odczyty stresu badanych osób

L.p.	Poziom stresu spoczynkowego (wartość podana przez producenta)	Poziom stresu podczas gry na PC (wartość podana przez producenta)	Poziom stresu podczas 1 gry na VR (wartość podana przez producenta)	Poziom stresu podczas 2 gry na VR (wartość podana przez producenta)
1	1	2	4	3
2	1	1	3	3
3	1	1	3	2
4	1	1	2	2
5	1	2	5	3
6	1	1	2	1
7	1	1	5	2
8	1	1	3	2
9	1	1	3	5
10	1	2	4	4
11	1	1	3	3
12	1	1	3	4
13	1	1	3	2
14	2	2	4	4
15	1	1	2	3
16	1	1	5	4
17	1	1	3	2
18	1	1	2	2
19	1	2	4	3
20	1	1	3	3

Pomiar stresu odbył się w analogiczny sposób jak pomiar pulsu. Im wyższa liczba (Rysunek 5, Tabela 3), tym większy stres u uczestnika badania. Jest to jednostka podana przez producenta urządzenia do badania stresu.

Stres zbadany został na podstawie zmiany tętna i miał odzwierciedlać faktyczne samopoczucie użytkownika. Sam producent urządzenia podkreśla jednak, że wyniki, w pełni mogą nie odzwierciedlać rzeczywistości [6, 7]. Otrzymane wyniki odpowiadają jednak wzrostowi pulsu, a także zgadzają się z personalnymi odczuciami badanych osób, dzięki czemu można wziąć je pod uwagę.

Wykres (Rysunek 5) jednoznacznie wskazuje, że u wszystkich uczestników nastąpił wzrost stresu podczas rozgrywki na okularach VR. Potwierdzały to również krótkie podsumowania uczestników po badaniu oraz wyniki z ankiety. U większości uczestników można było również zauważyć zwiększoną potliwość podczas przebywania w wirtualnej rzeczywistości. Mogło wynikać to zarówno z ruchów wykonywanych podczas rozgrywki jak i samego stresu oraz strachu, co również zostało potwierdzone przez badane osoby.



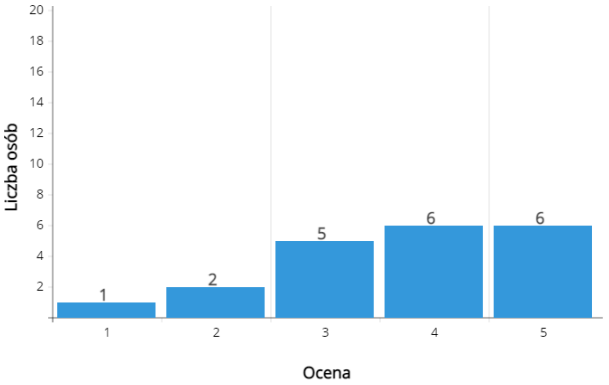
Rysunek 5: Przedstawienie wyników pomiaru stresu badanych osób.

4. Wyniki ankiety

W badaniu brało udział 13 mężczyzn oraz 7 kobiet. Połowa badanych nie miała wcześniej do czynienia z okularami VR i nigdy nie korzystała z tej technologii. Jedna osoba nie grała wcześniej w żadne gry komputerowe, natomiast 35% osób gra w nie praktycznie codziennie. W celu zebrania opinii, przeprowadzono ankietę. Zawierała ona 7 pytań. Odpowiedzi uzyskane na nie, zostaną omówione w kolejnych podrozdziałach.

4.1. Ankieta uporządkowane i nieuporządkowane

W poniższym histogramie im wyższa ocena tym lepsze samopoczucie użytkownika. Oznacza to, że ocena 5 to brak jakichkolwiek dolegliwości podczas korzystania z VR, natomiast ocena 1 to bardzo złe samopoczucie.



Rysunek 6: Histogram odpowiedzi ankietowanych na pytanie „Czy czuł/a się Pan/Pani niekomfortowo lub odczuwał/a jakieś inne dolegliwości podczas rozgrywki w okularach VR?”

Wyniki dotyczące tego pytania oraz własne obserwacje podczas badania pozwalają sformułować wytyczne dotyczące ograniczeń dla uczestników w korzystaniu z tej technologii. Są to:

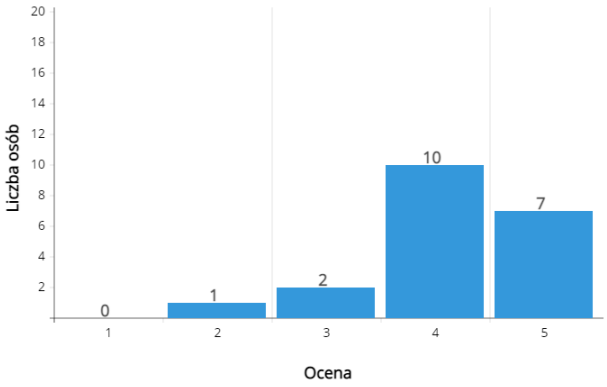
- ograniczenie czasu rozgrywki. Mimo zaleceń producenta okularów VR dotyczących maksymalnie 30-minutowej sesji grania, większość uczestników poczuła dyskomfort już po pierwszych minutach gry,
- zachowanie szczególnej ostrożności w przypadku istniejących chorób, jak np. choroba lokomocyjna [5]. U osoby posiadającej taką dolegliwość musiało nastąpić przerwanie badania, ze względu na bardzo złe samopoczucie – wystąpienie choroby symulatorowej,
- zalecenie, aby przy osobie, która odczuwa nawet lekkie dolegliwości podczas korzystania z VR, czuwała osoba towarzysząca. Podczas badania wystąpiły zawroty głowy u dwóch uczestników.

Należy jednak wziąć pod uwagę fakt, że na rynku istnieje wiele modeli okularów VR i każdy z nich posiada wady oraz zalety konstrukcyjne, które w znaczny sposób wpływają na komfort ich użytkowania. Okulary użyte w badaniu, czyli Oculus Quest 2 są jednymi z najnowszych okularów na rynku, jednak znajdują się w średnim przedziale cenowym. Mimo wysokiej rozdzielczości wyświetlanego obrazu (1832x1920 pikseli na każde oko) oraz wysokiej częstotliwości jego odświeżania (do 90Hz), producent nie dał dużych możliwości dostosowania okularów do użytkownika. Ograniczony rozstaw soczewek, waga oraz słabe dostosowanie samych okularów do kształtu głowy mogą nasilić niektóre dolegliwości oraz zwiększyć poziom zmęczenia oczu. Na rynku nadal nie ma idealnego urządzenia, jednak wciąż pojawiają się nowe rozwiązania, które w pewnym stopniu eliminują występujące problemy [8].

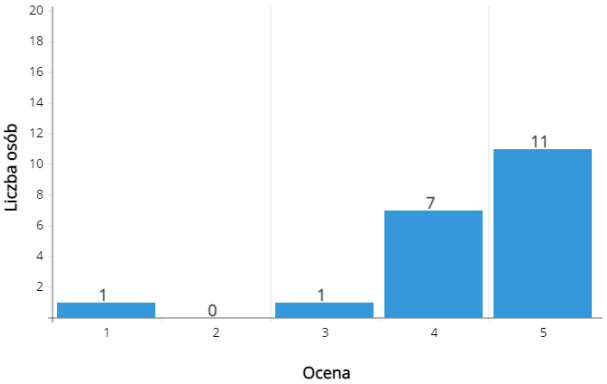
4.2. Ankieta – Pytania dotyczące wpływu grafiki i dźwięku na immersję aplikacji

Analogicznie do poprzedniego pytania z ankiety, im wyższa ocena w powyższych histogramach tym większy wpływ na immersję (1 – brak wpływu, 5 – ogromny wpływ). Większość uczestników uznała, że dźwięk ma większy lub tak samo ważny wpływ na immersję gracza jak oprawa wizualna. Nie jest to może tak oczywiste, ale dźwięk jest niezwykle ważny w odbiorze filmu lub gry. Dobrym przykładem jest tutaj znany gatunek filmowy jakim jest horror. Utwory tego gatunku składają się zwykle z komponentu wizualnego oraz dźwiękowego, w którym oprawa muzyczna i dźwiękowa często odgrywa ważniejszą rolę niż dialogi. Niektórzy twórcy filmowi twierdzą, że horror jest przede wszystkim dźwiękowym medium. Wykorzystane w filmie dźwięki tj. krzyki, skrzypienie, powolne kroki, narastająca w tle muzyka, czy jej nagły brak nie są w żaden sposób przypadkowe. Mają one wyzwolić u widzów poczucie strachu i napięcia, które będzie towarzyszyło im do końca seansu [9]. Oczywiście nie można też pominąć oprawy wizualnej. Pełni ona również niezwykle ważną rolę w odbiorze medium, co również wynika z badania. Można zatem stwierdzić, że estetyczna grafika, zaawan-

sowana fizyka gry (możliwe interakcje ze światem gry), a także wykorzystanie i dobranie odpowiednich dźwięków przestrzennych ma niezwykle duży wpływ na zanurzenie gracza w wirtualnym świecie.



Rysunek 7: Histogram odpowiedzi ankietowych dotyczących wpływu oprawy wizualnej gier na immersję aplikacji.

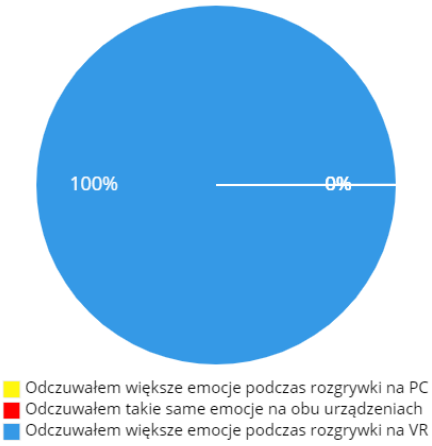


Rysunek 8: Histogram odpowiedzi ankietowych dotyczących wpływu dźwięku i muzyki gier na immersję aplikacji.

4.3. Ankieta – Pytania dotyczące zaangażowania i emocji podczas rozgrywki

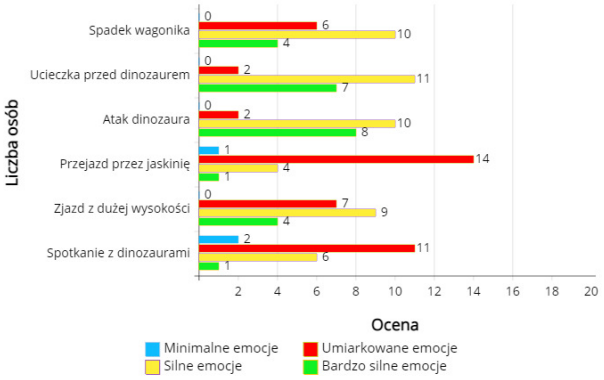
Rysunek 9: Wykres przedstawiający odpowiedzi na pytanie „Czy odczuwał/a Pan/Pani większe zaangażowanie podczas gry VR niż w przypadku gry na PC?”

Wszyscy uczestnicy badania odczuwali większe emocje podczas gry w VR. Podobnie było z zaangażowaniem – 90% uczestników udzieliło twierdzącej odpowiedzi. Mowa tutaj o odruchach oraz wymaganych interakcjach podczas rozgrywki. Uczestnik, który odczuwał identyczne zaangażowanie w przypadku obu urządzeń był najmłodszym oraz najbardziej doświadczonym graczem, co może sugerować, że przy częstszym korzystaniu z okularów ekscytacja nie jest już tak duża, a użytkownik przyzwyczaja się do wymagań gry.



Rysunek 10: Wykres przedstawiający odpowiedzi na pytanie „Czy odczuwał/a Pan/Pani większe emocje podczas gry VR niż w przypadku gry na PC?”

Największe emocje wywołały etapy wiążące się z nagłym przyspieszeniem wagonika lub atakiem dinozaurów, który powodował dyskomfort u badanych. Ciekawym zjawiskiem było wywołanie u uczestników wrażenia przeciążeń występujących podczas jazdy, jedynie przy pomocy dźwięku oraz efektów wizualnych. Podczas ostrych zakrętów czy wybicia się wagonika w powietrze większość badanych wychylała się na różne strony, jakby miało to miejsce w realnym świecie.



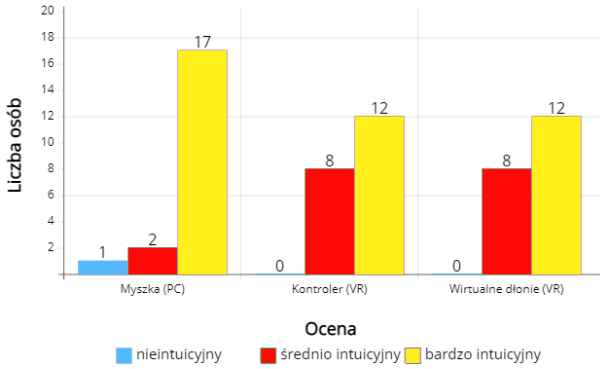
Rysunek 11: Histogram przedstawiający wyniki pytania, jak silne emocje wywołały u uczestników poszczególne etapy gry.

4.4. Ankieta – Pytanie dotyczące najbardziej intuicyjnego interfejsu

Najnowsza technologia pozwala na odzwierciedlenie rzeczywistej pozycji i rotacji rąk użytkownika w wirtualnym świecie. Z racji tego, że połowa badanych nie miała do czynienia z VR, wyniki dotyczące łatwości nawigowania po aplikacji za pomocą kontrolerów czy technologii wirtualnych dłoni są bardzo pozytywne. Rozwój technologii wykrywania dłoni i używania ich podczas rozgrywki może zwiększyć znacząco immersję graczy. Jest to najbardziej naturalny interfejs dla czło-

wieka. Zmniejsza to tzw. próg wejścia w korzystaniu z tej technologii.

Samo widzenie własnych dłoni w wirtualnym świecie przełamuje kolejną barierę między światem rzeczywistym, a wirtualnym i powoduje jeszcze większe zanurzenie się w grze. W aplikacjach możemy spotkać różne awatary dłoni, jak np. kończyny potwora, skrzydła, czy oczywiście dłonie ludzkie, co również ma wpływ na immersję. W badaniu opisanym w artykule pt. "These are not my hands!": Effect of Gender on the Perception of Avatar Hands in Virtual Reality, u kobiet zaobserwowano mniejsze poczucie zanurzenia w aplikacji, gdy używały awatarów męskich dłoni, natomiast u mężczyzn ten sam problem występował przy użyciu awatarów nie będących ludzkimi dłońmi [10].



Rysunek 12: Histogram odpowiedzi do pytania „Który interfejs jest dla Pana/Pani najbardziej intuicyjny?”

5. Wnioski

W artykule przedstawiono wyniki badania immersji podczas przebywania osób w wirtualnej rzeczywistości. Ukazują one jednoznacznie jak duży wpływ na wrażenia z gry ma zanurzenie gracza w jej świecie. U każdego z badanych można było zaobserwować wyraźny wzrost pulsu oraz stresu podczas przebywania w wirtualnym świecie w stosunku do pulsu spoczynkowego, czy tego podczas gry na standardowym komputerze. Oznacza to, że VR dostarczył im znacznie większych emocji, co również potwierdził każdy uczestnik w ankiecie. Objawy choroby symulatorowej, występujące u badanych, nie zniechęciłyby ich do ponownego zanurzenia się w wirtualnym środowisku. W krótkim czasie badania technologia ta zdołała zaciekawić każdego z uczestników, bez względu na jego wiek, czy wcześniejsze doświadczenia z VR. Jednym z czynników takiego zachowania mogło być wystąpienie imersji podczas rozgrywki, zwłaszcza w czasie najbardziej ekscytujących momentów, takich jak ucieczka przed atakującym dinozaurem, dynamiczna zmiana prędkości wagonika, czy zaawansowana fizyka i możliwości interakcji z obiektami gry. Istotny wpływ na zadowolenie z rozgrywki oraz faktyczne zanurzenie się w świecie gry, mają kontrolery, które pozwalają użytkownikowi na stosunkowo proste i intuicyjne sterowanie. Większość badanych nie miała problemu z nawigowaniem po aplikacji, a także doceniła ciągle rozwijającą się technologię wirtualnych dłoni, która ich zdaniem w przyszłości może znacznie poprawić wrażenia z gry.

Gry komputerowe czy konsolowe mogą uzależnić użytkownika, a znaczny na to wpływ ma właśnie immersja. Mogą również spowodować zatracenie się w ich świecie w pozytywnym tego słowa znaczeniu. Jest to osiągnięte dzięki ogromnej pracy i zaangażowaniu ich twórców. Grafika, dźwięki oraz mechaniki gry muszą znajdować się na najwyższym poziomie i wymagają często wielu lat pracy, aby sprostać wymaganiom graczy [11].

W przypadku VR, można powiedzieć, że użytkownik dosłownie znajduje się w grze. Pomimo uproszczonej grafiki, która często odbiega od flagowych gier na PC, gracze odczuwają silniejsze emocje. Dużo częściej czują, że oni sami są zagrożeni, a granica między światem rzeczywistym, a wirtualnym zatraca się, gdy interakcje w grze wymagają od użytkowników faktycznej aktywności fizycznej [12]. Wykorzystane jest to również w branży filmowej. Podobnie jak podczas seansu w kinie 3D, samo wrażenie głębi podczas przebywania w VR robi na użytkowniku niesamowite wrażenie. Wiele platform takich jak Netflix, YouTube, czy BigScreen znając możliwości wirtualnej rzeczywistości, oferuje obejrzenie filmu w zaciszu domowym, w jakości nie odbiegającej od uzyskiwanej podczas oglądania seansów w znanych multipleksach kinowych [13]. VR wraz z rozwojem technologii ma wręcz nieograniczony potencjał i z pewnością możemy spodziewać się, coraz większej rzeszy jego użytkowników.

Małym zaskoczeniem może być łatwość poruszania się i nawigowania w wirtualnym świecie nawet przez osoby starsze, które nigdy wcześniej nie miały do czynienia z okularami VR. Można również założyć, że następne generacje urządzeń generujących VR nie będą wykorzystywane jedynie do celów rozrywkowych. Technologia wirtualnych dłoni otwiera ścieżki w branży biznesowej czy badawczej, wykorzystując najbardziej naturalny interfejs dla człowieka. Obecnie już można zaobserwować wzrost powstawania aplikacji mających na celu utworzenie wirtualnego środowiska pracy czy spotkań biznesowych lub towarzyskich. Technologia ta jest jednak dalej na wczesnym etapie rozwoju i z pewnością, w niedalekiej przyszłości rozwiązanie to nie zastąpi dotychczasowej pracy biurowej, czy korzystania z interfejsu w postaci myszki i klawiatury.

Literatura

[1] Czym jest wirtualna rzeczywistość, https://www.marxentlabs.com/what-is-virtual-reality/ , [26.03.2019].

[2] E. Brown, P. Cairns, A grounded investigation of game immersion, Association for Computing Machinery, New York, 2004.

[3] S. Jayaram, H. I. Connacher, K. W. Lyons, Virtual assembly using virtual reality techniques, Computer-Aided Design 29 (1997) 575-584.

[4] Możliwe kierunki rozwoju technologii VR w przyszłości, https://www.forbes.pl/technologie/vr-w-przyszlosci-jak-rozwinie-sie-wirtualna-rzeczywistosc/m3hb9fs , [27.10.2017].

[5] I. Plakhotniuk, M. Popko, T. Szymczyk, Health Aspects of Immersion in VR. Virtual Disease - Factor or Myth?, INTED 2019: 13th International Technology, Education and Development Conference (2019) 543-551.

[6] Sposób mierzenia stresu pulsometrem z telefonu, https://www.wirtualnemedia.pl/artykul/pulsometr-w-smartfonie-galaxy-s5-sprawdzi-tez-poziom-stresu , [28.05.2014].

[7] Informacje od producenta, dotyczące mierzenia stresu, https://www.samsung.com/pl/support/mobiledevices/jak-korzystac-z-samsung-health-przy-uzyciu-galaxy-watch/ , [2020].

[8] Ł. Pełka, Ł. Podstawka, T. Szymczyk, Analiza porównawcza gogli do VR, Journal of Computer Sciences Institute 10 (2019) 36-43.

[9] G. N. Martin, (Why) Do You Like Scary Movies? A Review of the Empirical Research on Psychological Responses to Horror Films, Fronitiers in Psychology 10 (2019) Article 2298.

[10] V. Schwind, P. Knierim, C. Tasci, P. Franczak, N. Haas, N. Henze, "These are not my hands!": Effect of Gender on the Perception of Avatar Hands in Virtual Reality, CHI '17: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (2017) 1577-1582.

[11] P. Świątek, Immersja w grach MMO, czyli o „farmieniu expa" słów kilka, Media i Społeczeństwo 2 (2012) 94-100.

[12] J. T. Lin, Fear in virtual reality (VR): Fear elements, coping reactions, immediate and next-day fright responses toward a survival horror zombie virtual reality game, Computers in Human Behavior 72 (2017) 350-361.

[13] W. Siwak, Matrix i pół-Matrix, czyli rzeczywistość wirtualna i rzeczywistość rozszerzona jako wyzwania dla tożsamości, kultury, sztuki, Rocznik Naukowy Kujawsko-Pomorskiej Szkoły Wyższej w Bydgoszczy, Transdyscyplinarne Studia o Kulturze (i) Edukacji 11 (2016) 355-388.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 217–224
Received: 21 June 2021
Accepted: 29 June 2021

Comparative analysis of the proprietary navigation system and the built-in Unity engine tool

Analiza porównawcza autorskiego systemu nawigacji i wbudowanego narzędzia silnika Unity

Maciej Kempny*, Marcin Barszcz

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

Abstract
Navigation systems are advanced tools that allow you to create characters intelligently moving around the game world. The purpose of this thesis was to conduct a comparative analysis of the proprietary navigation system "AlchemyNavigation" and the built-in tool of the Unity engine - "NavMesh". An proprietary application created with the Unity engine was used to conduct the research, under which identical scenarios based on the tested systems were implemented. Finally, on the basis of the collected results and documentation of the research objects, a comparative analysis was carried out and proved the thesis that own solutions can match the default solutions.

Keywords: navigation systems; unity; navmesh; alchemynavigation

Streszczenie
Systemy nawigacyjne to zaawansowane narzędzia, które pozwalają tworzyć postacie inteligentnie poruszające się po świecie gry. W ramach niniejszej pracy przeprowadzono analizę porównawczą autorskiego systemu nawigacji o nazwie AlchemyNavigation oraz wbudowanego narzędzia silnika Unity - systemu NavMesh. Do przeprowadzenia badań wykorzystano autorską aplikację zbudowaną przy pomocy silnika Unity, w ramach której zaimplementowano tożsame scenariusze badawcze bazujące na porównywanych systemach. Finalnie, na podstawie zebranych wyników oraz dokumentacji badanych systemów zrealizowana została analiza porównawcza, która dowiodła tezy że własne rozwiązania mogą dorównać rozwiązaniom domyślnym, a także, że pozwalają wprowadzać nowe funkcjonalności.

Słowa kluczowe: systemy nawigacji; unity; navmesh; alchemynavigation

*Corresponding author
Email address: maciej.kempny@pollub.edu.pl (M. Kempny)

1. Wstęp

W dzisiejszych czasach twórcy gier oraz symulacji bardzo rzadko decydują się na tworzenie własnych rozwiązań dla problemów, które pojawiają się w większości tego typu produkcji. Kierunek rozwoju technologii sprawił, że na rynku pojawiły się gotowe narzędzia ogólnego przeznaczenia, które niższym kosztem i nakładem pracy pozwalają na osiągnięcie zamierzonych (bądź zbliżonych do zamierzonych) efektów. Przykładem takiego zjawiska jest Unity - silnik gier, który między innymi dostarcza rozwiązań dla problemów takich jak: proces renderowania, symulacja fizyki oraz nawigacja agentów. Poza wykorzystaniem wbudowanych elementów Unity pozwala również na tworzenie własnych, które mogą kontrolować, uzupełnić bądź zastąpić gotowe rozwiązania.

Systemy nawigacji to zaawansowane narzędzia pozwalające na tworzenie postaci, które mogą, w inteligentny sposób, poruszać się po świecie gry bądź symulacji. Są to rozwiązania wielopoziomowe, otóż sprawny system powinien zapewniać następujące funkcjonalności: określenie powierzchni nawigacyjnej, wytyczanie optymalnych ścieżek oraz naturalny ruch agentów. NavMesh to wbudowany system nawigacji silnika Unity. W niniejszej pracy zostanie on porównany pod kątem wydajności i uniwersalności z autorskim rozwiązaniem o nazwie AlchemyNavigation.

Wymienione poniżej pozycje literaturowe dowodzą że tematy związane z nawigacją w grach komputerowych i robotyce nie są obce w zbiorach prac naukowych. Nie mniej jednak są to tematy bardzo złożone, i z tego powodu większość z dostępnych materiałów skupia się na badaniu i ulepszaniu tylko konkretnych elementów wchodzących w skład tego zagadnienia. Natomiast temat porównania wydajności i funkcjonalności całych systemów jest poruszany bardzo rzadko.

W artykule [1] omawiany jest problem kooperacyjnego wyszukiwania ścieżek - wyszukiwania i planowania ścieżek dla wielu agentów, w taki sposób aby ograniczyć zachodzące między nimi kolizje. Autor zwraca uwagę na to jak ważna, w kontekście nowoczesnych gier, jest funkcjonalność unikania kolizji oraz redukcja generowanego przez nią obciążenia podzespołów stacji komputerowych. Prezentowane jest nowe podejście – CA* (Cooperative A*), które pozwala na efektywne rozwiązywanie tego problemu, oraz badania poddające analizie jego funkcjonalności. Ostatecznie udowodniono tezę, że zaproponowane przez autora rozwiązanie jest efektywne i pozwala na lepsze wybieranie ścieżki niż inne analizowane algorytmy.

W artykule [2] prezentowana jest metoda polegająca na przedstawieniu przestrzeni nawigacyjnej w oparciu o ograniczoną triangulację Delaunay (ang. constrained Delaunay triangulation), dzięki której możliwe jest znaczące zmniejszenie złożoności procesu wyszukiwania ścieżek. Autor w swoich badaniach skupia się na udowodnieniu przewagi przedstawianej metody nad innymi znanymi sposobami, oraz potwierdzeniu jej skuteczności i przetestowaniu jej w różnych scenariuszach badawczych.

Waga problemu wygładzania ścieżek tworzonych przez algorytmy wyszukiwania trasy jest bardzo duża w kontekście systemów nawigacji. Autor artykułu [3] motywuje tą tezę oraz tłumaczy w jaki sposób powinna wyglądać ścieżka, aby agenci mogli z niej poprawnie korzystać. Jednak, głównym celem artykułu i badań jest omówienie metody, która bazuje na kwadratowej interpolacji i pozwala na uzyskanie odpowiedniego efektu. W toku pracy działanie algorytmu i jego atuty zostają dokładnie przedstawione.

Artykuł [4] został poświęcony przeglądowi i analizie zagadnienia wyszukiwania ścieżek. Autor podejmuje próbę przedstawienia, w prostej formie, elementów oraz sekwencji operacji składających się na proces wytyczania trasy. Szczegółowo omówione są różne etapy procesu, logika stojąca za najbardziej popularnymi algorytmami (takimi jak: algorytm A* lub algorytm Dijkstry). Przeglądowi poddane zostały również algorytmy rozszerzające, które zapewniają funkcjonalności takie jak: płynny ruch agentów i unikanie kolizji. Finalnie, dla wybranych rozwiązań, przedstawione są również sposoby dzięki którym możliwe jest uzyskanie większej efektywności.

W artykule [5] autorzy dokonują analizy działania i sposobów wykorzystywania systemu nawigacji dostarczonego przez silnik gier Unity. Szczegółowo omawiane są zastosowane algorytmy, sposób w jaki są wykorzystywane oraz jaki wpływ ma to na rozwój gier komputerowych

Finalne na rynku dostępnych jest również wiele książek odnoszących się do tematyki nawigacji [6]. Pojawiają się tytułu dotyczące sztucznej inteligencji - ruchu i wyszukiwania ścieżek w grach [7], zestawy gotowych do implementacji rozwiązań dla konkretnych platform [8], oraz teorii stojącej za popularnymi rozwiązaniami [9]. Ostatecznie często spotykane są również tytuły przeznaczone dla nowicjuszy wprowadzające w tajniki wykorzystywania silników gier w ramach, których rozwijany jest temat nawigacji [10], bądź zestawy wiedzy dla bardziej zaawansowanych użytkowników [11].

2. Lista cech uniwersalnego systemu nawigacji

Lista cech uniwersalnego systemu nawigacji (tabela 1) została opracowana na podstawie przeglądu gier dostępnych aktualnie na rynku.

Tabela 1: Lista cech uniwersalnego systemu nawigacji

Lp.	Cecha systemu nawigacji
1	Możliwość tworzenia predefiniowanych części przestrzeni nawigacyjnej
2	Możliwość tworzenia i modyfikowania przestrzeni nawigacyjnej w czasie wykonania programu
3	Możliwość wprowadzenia podziału przestrzeni nawigacyjnej na mniejsze elementy (ang. chunks) i płynnego włączania i wyłączania ich
4	Możliwość tworzenia agentów przemieszczających się w naturalny sposób po przestrzeni nawigacyjnej
5	Możliwość ustalania parametrów ruchu dla różnych agentów
6	Możliwość przypisywania wag decydujących o częstości wybierania określonych fragmentów przestrzeni nawigacyjnej w procesie poszukiwania ścieżki
7	Możliwość definiowania alternatywnych powierzchni nawigacyjnych dla różnego typu agentów
8	Zapewnienie funkcji unikania kolizji dla agentów
9	Wsparcie dla dużej liczby (rzędu setek) agentów jednocześnie
10	Możliwość kontrolowania przemieszczania się agentów pomiędzy oddzielonymi fragmentami przestrzeni nawigacyjnej

3. Obiekt badań

Ten rozdział został poświęcony opisowi badanych systemów, który został opracowany w oparciu o ich dokumentacje techniczne [12, 13].

3.1. AlchemyNavigation

AlchemyNavigation to autorski system nawigacji, który może zostać dołączony do projektu Unity za pomocą narzędzia Package Manager. Charakteryzuje się rozbudowanymi funkcjonalnościami, które pozwalają na tworzenie i modyfikowanie przestrzeni nawigacyjnej w czasie wykonania programu oraz przystosowanymi do nadpisywania modułami.

Najważniejszym elementem systemu jest komponent AlchemyNavigationSystem, który zapewnia działanie wszystkich kluczowych funkcji, oraz pozwala na dostosowanie ustawień nawigacji, w tym przedstawia podział przestrzeni nawigacyjnej na 32 powierzchnie i warstwy.

System oferuje trzy sposoby tworzenia przestrzeni nawigacyjnej, są to:

1. Tworzenie w oparciu o skrypty - użytkownik może pisać skrypty, które w dowolnym momencie rozszerzają przestrzeń nawigacyjną o kolejne trójkąty. Każda operacja dodania zwraca unikalny uchwyt, który można wykorzystać do usunięcia dodanego trójkąta.
2. Tworzenie z poziomu komponentu FacesHolder - komponentu wysokiego poziomu, który ukrywa złożoność tworzenia przestrzeni nawigacyjnej w oparciu o skrypty - pozwala na wizualne modyfikowanie trójkątów za pomocą edytora. Dodatkowo komponent wspiera wszystkie domyślne funkcje obiektów gry Unity, to znaczy w dowolnym momencie może być włączany, wyłączany, przemieszczany, rotowany i skalowany, a dokonane zmiany zostaną odzwierciedlone na aktywnej powierzchni nawigacyjnej.
3. Wypalanie na podstawie geometrii sceny - w procesie wypalania, użytkownik może tworzyć przestrzeń nawigacyjną w oparciu o geometrię znajdującą się

na scenie gry. Proces ten może być realizowany statycznie lub dynamicznie. Statycznie wypalanie wykonywane jest za pomocą narzędzi edytora, a jego rezultatem jest zestaw obiektów z dodanymi komponentami FacesHolder (które mogą być dalej modyfikowane przez użytkownika). Natomiast wypalanie dynamiczne (w czasie działania programu) wykonywane jest za pomocą skryptów, a jego rezultatem są zestawy trójkątów, które mogą zostać dodane do przestrzeni nawigacyjnej.

Na etapie tworzenia agentów, system pozwala na obranie jednej z dwóch dróg: tworzenie i wykorzystanie własnych niestandardowych agentów, lub wykorzystanie wbudowanych prostych agentów. Prości agenci zapewniają standardowy ruch, który jest odpowiedni dla większości postaci spotykanych w grach komputerowych. Dodatkowo ich zachowanie można modyfikować poprzez dodawanie specjalnych komponentów nazywanych modyfikatorami ruchu. Ta funkcja może zostać wykorzystana do implementacji zachowania polegającego na unikaniu kolizji.

3.2. NavMesh

NavMesh to system nawigacji dostępny domyślnie w każdym projekcie rozwijanym przy pomocy Unity. Charakteryzuje się bardzo wysokim poziomem integracji ze środowiskiem i uniwersalnością. Sam system składa się czterech głównych elementów, są to:

1. NavMesh - struktura danych opisująca powierzchnię, po której mogą poruszać się agenci.
2. NavMesh Agent - komponent pozwalający na tworzenie postaci, które mogą poruszać się po powierzchni nawigacyjnej oraz unikać kolizji z innymi postaciami i przeszkodami.
3. NavMesh Obstacle - komponent pozwalający na tworzenie dynamicznych przeszkód, które powinny być unikane przez agentów.
4. Off-Mesh Link - komponent pozwalający na tworzenie skrótów pomiędzy miejscami znajdującymi się na powierzchni nawigacyjnej.

Poza głównymi składnikami, Unity pozwala również na skorzystanie z wysokopoziomowych narzędzi, które dostarczają dodatkowej kontroli nad procesem budowania powierzchni nawigacyjnej (w czasie wykonania programu oraz w edytorze), są to:

1. NavMesh Surface - komponent pozwalający na budowanie powierzchni nawigacyjnej dla jednego typu agenta oraz przełączanie jej aktywności.
2. NavMesh Modifier - komponent pozwalający na oznaczanie typu obszaru powierzchni nawigacyjnej w oparciu o hierarchię obiektów na scenie.
3. NavMesh Modifier Volume - komponent pozwalający na oznaczanie typu obszaru powierzchni nawigacyjnej w oparciu o zdefiniowany obszar.
4. NavMesh Link - komponent pozwalający na dynamiczne tworzenie połączeń między dwiema lokacjami znajdującymi się na powierzchniach określanych przez NavMesh Surface.

W ramach systemu NavMesh powierzchnia nawigacyjna tworzona jest w procesie wypalania, który pobiera komponenty renderujące ze wszystkich obiektów oznaczonych na scenie jako statyczne nawigacyjnie, a następnie przetwarza zawarte w nich dane dotyczące geometrii, i na tej podstawie aproksymuje powierzchnie po których mogą poruszać się agenci.

4. Metodyka badań

Badania zostały przeprowadzone w sposób identyczny na trzech różnych stacjach, które różnią się podzespołami i parametrami. Specyfikacja maszyn została przedstawiona w tabeli 2.

Tabela 2: Specyfikacja stacji wykorzystanych do przeprowadzenia badań

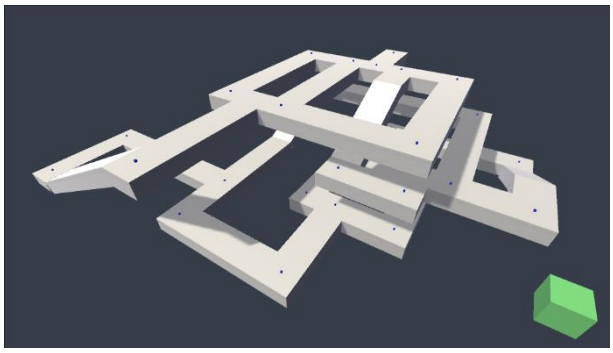
Numer stacji	Procesor	Taktowanie procesora (GHz)	Karta graficzna	Pamięć RAM (GB)
1	Intel Core i7-6700K	4,00	Nvidia GeForce GTX 1660 SUPER	32
2	Intel Core i7-8750H	2,20	Nvidia GeForce GTX 1050Ti	16
3	Intel Core i7-7200U	2,50	Nvidia GeForce 940MX	4

4.1. Badanie obciążenia procesora i pamięci RAM

Badania obciążenia procesora i pamięci RAM zostały przeprowadzone w oparciu o autorską aplikację, która wykorzystuje badane systemy oraz nie wykorzystuje systemów nawigacji (skala odniesienia). Kryteria porównawcze stanowią:

1. Średni czas trwania klatki liczony w sekundowych odstępach w dwudziestominutowym okresie działania aplikacji.
2. Średnia ilość pamięci RAM wykorzystywanej przez aplikację liczona w sekundowych odstępach w dwudziestominutowym okresie działania aplikacji.

Na potrzeby badań utworzony został projekt unity w ramach którego przygotowana została scena bazowa (rysunek 1), która posiada cechy opisane w tabeli 3. Przygotowane zostały również warianty sceny bazowej (tabela 4) różniące się aktywnym systemem nawigacji i jego konfiguracją.



Rysunek 1: Scena bazowa autorskiej aplikacji testowej.

Tabela 3: Cechy sceny bazowej

Lp.	Cecha sceny
1	Scena jest zbudowana ze stu prostopadłościennych obiektów.
2	Obiekty na scenie umieszczone są na różnych poziomach względem osi pionowej, w taki sposób że tworzą piętra.
3	Piętra są ze sobą połączone, w taki sposób że obiekty tworzą większą nie prymitywną figurę.

Tabela 4: Warianty sceny bazowej

Lp.	Wariant sceny
1	Brak aktywnego systemu nawigacji.
2	Aktywny system AlchemyNavigation (bez unikania kolizji).
3	Aktywny system AlchemyNavigation (z unikaniem kolizji).
4	Aktywny system NavMesh (bez unikania kolizji).
5	Aktywny system NavMesh (z unikaniem kolizji).

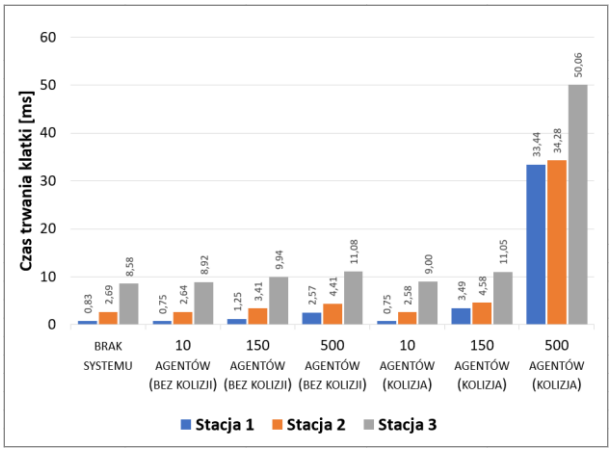
Na każdym z wariantów sceny bazowej wykonano pomiary dla 10, 150 i 500 agentów, na trzech różniących się parametrami maszynach. Do wykonania pomiarów wykorzystano skrypt bazujący na narzędziach do profilowania, które są dostępne w domyślnych bibliotekach Unity - Unity.Profiling [14].

4.2. Analiza porównawcza

Ta część badań skupia się na przeprowadzaniu analizy porównawczej cech badanych systemów w oparciu o ich dokumentację oraz wykonane pomiary. Szczególna uwaga została poświęcona cechom przedstawionym na liście cech uniwersalnego systemu nawigacji (tabela 1).

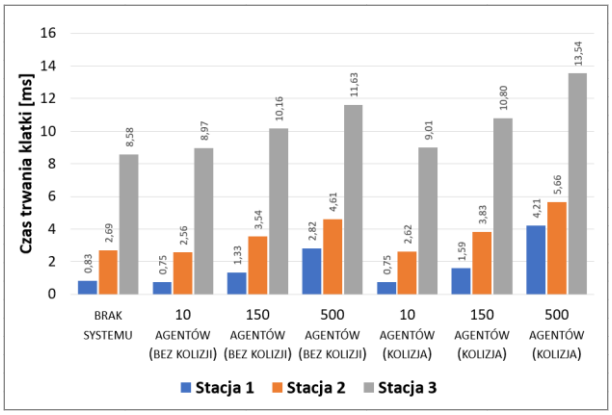
5. Wyniki badań i dyskusja

Na rysunkach 2 i 3 przedstawiono średnie wyniki pomiarów wykonanych dla badanych systemów pod kątem czasu trwania klatki.



Rysunek 2: Średni czas trwania klatki dla systemu AlchemyNavigation.

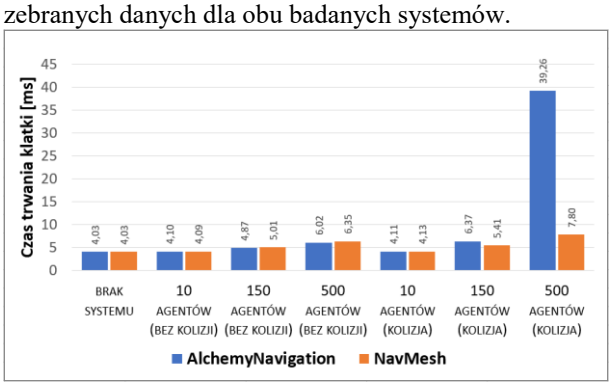
Na podstawie otrzymanych wyników badań dla systemu AlchemyNavigation można stwierdzić, że podzespoły maszyny i liczba agentów mają zauważalny wpływ na czas trwania klatki. Wartym uwagi jest również fakt, że w przypadku większych liczb aktywnych agentów, włączona funkcja unikania kolizji powoduje gwałtowny wzrost wykorzystania procesora. Dla stu pięćdziesięciu agentów największa różnica wyniosła 2,24 ms, natomiast dla pięciuset agentów aż 38,98 ms.



Rysunek 3: Średni czas trwania klatki dla systemu NavMesh.

W oparciu o wyniki dla systemu NavMesh można stwierdzić, że liczba agentów i podzespoły maszyny mają widoczny wpływ na czas trwania klatki. Wartym uwagi jest również fakt, że wzrost obciążenia procesora generowany poprzez włączenie funkcji unikania kolizji jest zauważalny, ale stabilny i niewysoki (w najgorszej próbie ~50%).

Na rysunku 4 przedstawiono uśrednione porównanie zebranych danych dla obu badanych systemów.



Rysunek 4: Porównanie uśrednionych czasów trwania klatki pobranych ze wszystkich stacji dla badanych systemów.

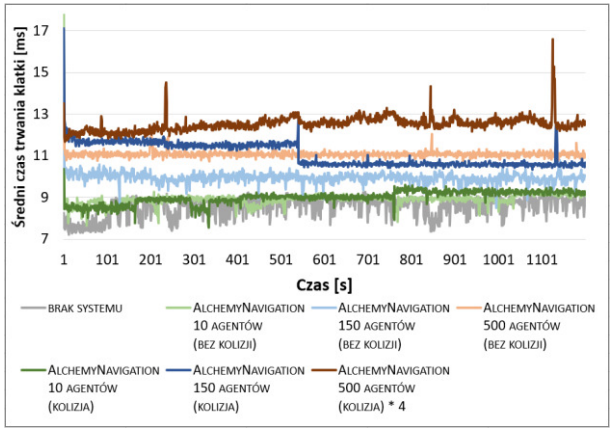
Pierwszym nasuwającym się wnioskiem jest fakt, że w przypadku niskiej liczby agentów (10), niezależnie od systemu i tego czy funkcja unikania kolizji jest włączona, wzrost czasu klatki jest niemalże niezauważalny - maksymalnie ~2%. System AlchemyNavigation jest nieznacznie szybszy, gdy funkcja unikania kolizji jest wyłączona. W przeciwnej sytuacji, system NavMesh zużywa mniej zasobów procesora i jest bardziej stabilny, w szczególności kiedy liczba aktywnych agentów jest wysoka (różnica 31,46 milisekund w próbie 500 agentów z aktywnym unikaniem kolizji).

Na rysunkach 5 i 6, przedstawiono średnie przebiegi czasowe prób, odpowiednio dla systemu AlchemyNavigation i NavMesh.

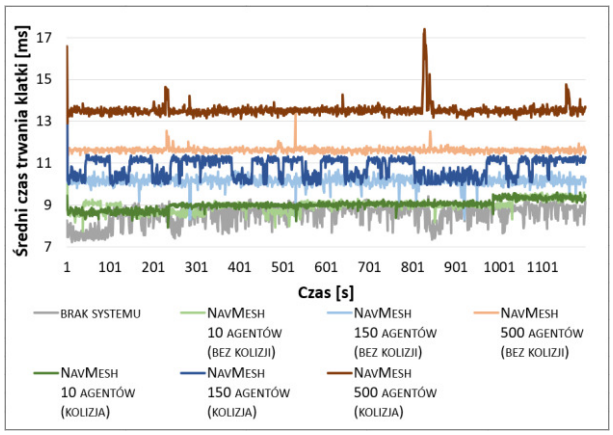
Na podstawie przebiegów czasowych obu badanych systemów, można zauważyć następującą zależność - próby z wyłączoną funkcją omijania kolizji są stabilniejsze - zauważalne zmiany występują rzadziej i są mniejsze. W przypadku kiedy unikanie kolizji jest włączone i liczba agentów wynosi 150 lub 500, można również dostrzec tendencje do okresowych zmian długości czasu trwania klatki. Na przykład, w próbie stu

pięćdziesięciu agentów systemu AlchemyNavigation wartości początkowo stabilizują się na poziomie 11,5 ms, a następnie na poziomie 11 ms. Natomiast w tożsamej próbie systemu NavMesh wartości naprzemiennie stabilizują się na poziomie 9,5 ms i 11,5 ms. Tendencja ta może wynikać z faktu, że na wąskich korytarzach mapy tworzą się zatory i tego jak systemy radzą sobie z kontrolą tłumu.

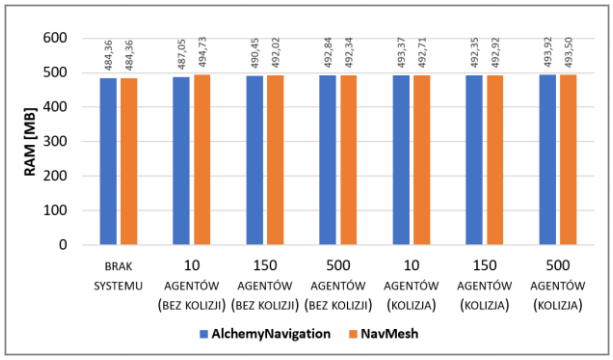
Na rysunku 7 porównano średnie wartości użycia pamięci RAM uzyskane we wszystkich próbach.



Rysunek 5: Średnie wartości czasu trwania klatki (ms) prób systemu AlchemyNavigation.



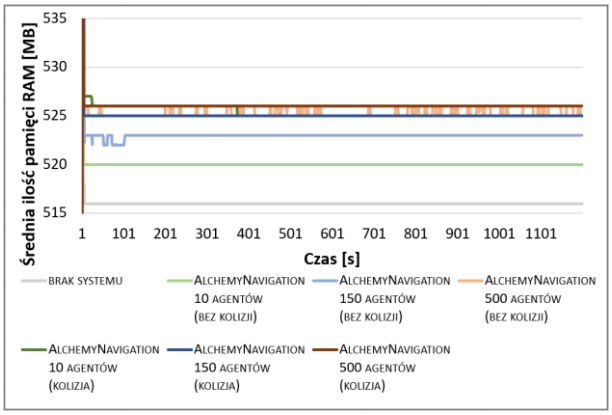
Rysunek 6: Średnie wartości czasu trwania klatki (ms) prób systemu NavMesh.



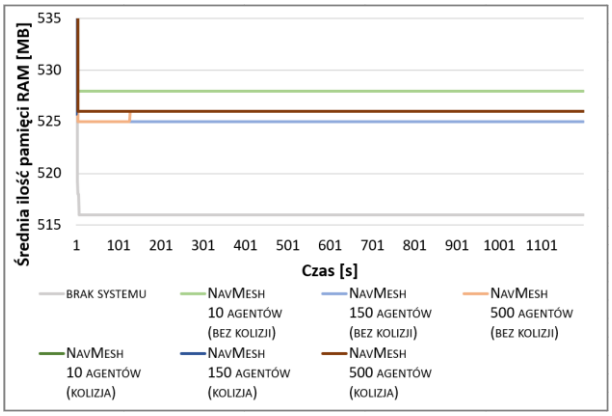
Rysunek 7: Porównanie uśrednionego wykorzystania pamięci RAM wszystkich stacji dla badanych systemów.

W przypadku obu badanych systemów, z rezultatów badań wynika, że nie ma zauważalnej relacji pomiędzy liczbą agentów, a wykorzystaniem pamięci RAM. Największa odnotowana różnica wyników wyniosła 1,57%. Jest to wynik na tyle niski, że można uznać różnice w wykorzystaniu pamięci przez badane systemy za nieistotną.

Rysunki 8 i 9 przedstawiają średnie wartości wykorzystania pamięci RAM wykonanych prób dla obu systemów.



Rysunek 8: Średnie wartości wykorzystania pamięci RAM dla prób systemu AlchemyNavigation.



Rysunek 9: Średnie wartości wykorzystania pamięci RAM dla prób systemu NavMesh.

W oparciu o przebiegi czasowe można dostrzec, że w przypadku obu badanych systemów wykorzystanie pamięci RAM jest bardzo stabilne - występujące zmiany nie przekraczają wartości jednego Megabajta. Należy również zauważyć, że w pierwszych sekundach użycie pamięci RAM zmienia się gwałtownie, może to wynikać z faktu, że w tym okresie inicjalizowane są przestrzenie nawigacyjne i wszyscy agenci żądają ścieżek w jednym momencie. Wartym spostrzeżenia jest fakt, że w przypadku prób systemu AlchemyNavigation z mniejszą liczbą agentów wartości stabilizują się na nieznacznie niższych poziomach (520 MB dla dziesięciu agentów i 523 MB dla stu pięćdziesięciu agentów). Natomiast, w przypadku systemu NavMesh wszystkie wartości stabilizują się przy podobnym zużyciu pamięci ~526 MB.

6. Analiza porównawcza cech badanych systemów

W tym rozdziale, badane systemy porównano pod kątem posiadanych cech w oparciu o wyniki badań i dokumentacje [12, 13]. Szczególną uwagę poświęcono

cechom znajdującym się na liście cech uniwersalnego systemu nawigacji (tabela 1).

6.1. Tworzenie predefiniowanych części przestrzeni nawigacyjnej

Dla systemu AlchemyNavigation tworzenie predefiniowanych części przestrzeni nawigacyjnej jest czynnością domyślną. Przestrzeń nawigacyjna jest budowana poprzez sekwencyjnie dodawanie trójkątów, które mogą być przechowywane w formie serializowanej (za pomocą komponentu FacesHolder lub innych typów zdefiniowanych przez użytkownika). Z kolei serializowane dane mogą być przechowywane na scenie lub w formach prefabrykatu (ang. prefab).

W przypadku systemu NavMesh funkcjonalność ta nie jest domyślnie dostępna, a cała przestrzeń zapisywana jest na scenie. Żeby korzystać z opcji tworzenia predefiniowanych części przestrzeni nawigacyjnej w formie prefabrykatów należy pobrać paczkę narzędzi wysokiego poziomu i użyć komponentu NavMesh Surface.

6.2. Tworzenie i modyfikowanie przestrzeni nawigacyjnej w czasie wykonania programu

System AlchemyNavigation pozwala na swobodne modyfikowanie przestrzeni nawigacyjnej w czasie wykonania programu, poprzez dodawanie i usuwanie należących do niej trójkątów. Operacje te wykonywane są z użyciem wątków, a potrzebne dane mogą być przechowywane w dowolnej serializowanej formie (np. plik lub komponent) lub generowane proceduralnie. Użytkownicy mogą również wykorzystać komponent FacesHolder w sposób opisany w punkcie 6.1.

W tej dziedzinie system NavMesh posiada mniejsze możliwości. Użytkownicy mogą wykorzystywać sposób opisany w punkcie 6.1 do dynamicznego przełączania aktywności wybranych elementów przestrzeni nawigacyjnej oraz dodatkowo rozmieszczać komponenty NavMesh Obstacle, które definiują kształt, w obrębie którego zdefiniowana przestrzeń zostanie wyłączona.

6.3. Podział przestrzeni nawigacyjnej na mniejsze elementy (ang. chunks)

Żaden z badanych systemów nie posiada dedykowanego narzędzia, które służy do dynamicznego przeładowywania fragmentów przestrzeni i zapewnia czyszczenie pamięci podręcznej. Aczkolwiek takie narzędzie może zostać łatwo opracowane w oparciu o sposób bazujący na wykorzystaniu prefabrykatów opisany w punkcie 6.1 (i innych form serializacji w przypadku systemu AlchemyNavigation) i systemu zasobów adresowalnych [15] (Addressables), który jest jednym z udostępnionych przez Unity pakietów.

6.4. Tworzenie agentów przemieszczających się w naturalny sposób po przestrzeni nawigacyjnej

Oba badane systemy posiadają możliwość tworzenia agentów, którzy symulują naturalny ruch w trakcie przemieszczania się po przestrzeni nawigacyjnej. W przypadku systemu AlchemyNavigation odpowiedzialny za tą funkcję jest komponent SimpleAgent, a w przypadku systemu NavMesh - NavMesh Agent.

6.5. Ustalanie parametrów ruchu dla różnych agentów

Oba badane systemy posiadają możliwość ingerowania w sposób, w jaki przemieszczają się agenci. System AlchemyNavigation posiada trzy możliwe scenariusze w ramach których takie zmiany są możliwe:

1. W oparciu o dziedziczenie po abstrakcyjnej klasie BasicAgent możliwe jest definiowanie nowych typów agentów, którzy w sposób całkowicie zależny od programisty przemieszczają się po obliczonych przez system ścieżkach.
2. Wykorzystując komponent SimpleAgent, który zapewnia naturalny ruch agentów i pozwala dostosować takie parametry jak: maksymalna prędkość, maksymalne przyspieszenie i prędkość rotacji.
3. Wykorzystując komponent SimpleAgent i podsystem MovementModifier, który pozwala definiować dodatkowe modyfikatory wpływające na sposób w jaki porusza się agent.

W przypadku systemu NavMesh, możliwe jest dostosowanie parametrów: prędkość, prędkość kątowa i przyspieszenie.

6.6. Przypisywanie wag decydujących o częstości wybierania określonych fragmentów przestrzeni nawigacyjnej w procesie poszukiwania ścieżki

Analizowane systemy nawigacji implementują bardzo podobne rozwiązania w kwestii przypisywania wag wybranym fragmentom przestrzeni nawigacyjnej. W obu przypadkach użytkownik może zdefiniować do 32 powierzchni (ang. areas). Powierzchnie posiadają wagę (AlchemyNavigation) lub koszt (NavMesh), które modyfikują częstość wyszukiwania prowadzących przez nie ścieżek. Oba systemy również pozwalają na definiowanie masek wskazujących na powierzchnie po których mogą przemieszczać się agenci.

6.7. Definiowanie alternatywnych powierzchni nawigacyjnych dla różnego typu agentów

Oba badane systemy posiadają opcje definiowania niezależnych przestrzeni nawigacyjnych, które można wykorzystać do budowania zaawansowanej logiki przemieszczania się agentów lub do celów optymalizacyjnych. W przypadku systemu AlchemyNavigation jest to podział na warstwy (ang. layers), a dla systemu NavMesh - typy agentów.

6.8. Unikanie kolizji

System AlchemyNavigation implementuje zachowanie unikania kolizji przy pomocy modyfikatora ruchu - AvoidanceModifier. Nawiązując do dokumentacji jest to bardzo prosta implementacja algorytmu, która jest odpowiednia tylko dla sytuacji, w których liczba agentów nie jest wysoka lub są oni odpowiednio podzieleni na grupy. Informację tą potwierdzają również wyniki badań, otóż średni wzrost czasu trwania klatki dla agentów wykorzystujących modyfikator względem tych

którzy go nie wykorzystywali wynosił: 0,23% dla dziesięciu, 31,80% dla stu pięćdziesięciu i 552,16% dla pięciuset agentów. Rekomendowanym alternatywnym sposobem, jest wykorzystanie klasy MovementModifier do przygotowania własnego modyfikatora dostosowanego do potrzeb użytkownika.

System NavMesh posiada wbudowany system unikania kolizji, który oferuje prace w pięciu trybach jakości: żaden (unikanie kolizji wyłączone), niski, średni, dobry i wysoki. Nawiązując do wykonanych badań średni wzrost czasu trwania klatki po przełączeniu trybu z "żaden" na "wysoki" wynosił odpowiednio: 0,98% dla dziesięciu, 7,98% dla stu pięćdziesięciu i 22,83% dla pięciuset agentów. Co oznacza że oferowany moduł jest znacznie wydajniejszy niż ten, który jest wykorzystywany przez system AlchemyNavigation.

6.9. Wsparcie dla dużej liczby (rzędu setek) agentów jednocześnie

Wsparcie dla dużej liczby agentów oznacza, że aplikacje wykorzystujące systemy nawigacji powinny działać ze stabilną liczbą klatek na sekundę [16], w tabeli 5 przedstawiono zestawienie standardowych progów liczby klatek na sekundę w grach.

Tabela 5: Standardowe progi liczby klatek na sekundę w grach komputerowych

Próg	Liczba klatek na sekundę	Przybliżony czas trwania klatki (ms)	Opis
Minimalny	30	33	Minimalna liczba klatek na sekundę uznawana za płynną w grach komputerowych
Standardowy	60	17	Liczba klatek na sekundę odpowiadająca najczęściej spotykanej częstotliwości odświeżania monitorów na rynku
Wysoki	144	7	Liczba klatek na sekundę odpowiadająca częstotliwości monitorów przeznaczonych dla graczy

Dla badanej aplikacji próg minimalny nie został osiągnięty jedynie w przypadku próby dotyczącej pięciuset agentów systemu AlchemyNavigation (z włączoną funkcją unikania kolizji). Wszystkie pozostałe próby stacji trzeciej osiągnęły poziom standardowy, a stacji jeden i dwa – wysoki.

6.10. Kontrolowanie przemieszczania się agentów pomiędzy oddzielonymi fragmentami przestrzeni nawigacyjnej

Przemieszczanie agentów pomiędzy oddzielonymi fragmentami przestrzeni nawigacyjnej w systemie AlchemyNavigation może zostać wykonane poprzez dynamiczne dodanie do przestrzeni trójkątów, które łączą wybrane fragmenty. Sposób ten jest naturalny i logiczny, ale może okazać się czasochłonny, ponieważ wymaga dodatkowego rozplanowania predefiniowanych połączeń lub przygotowania algorytmów odpowiedzialnych za ten proces.

Podejście prezentowane przez system jest prostsze - polega na wskazaniu dwóch punktów, pomiędzy którymi agenci będą się przemieszczać. Operacji tej dokonuje się za pomocą komponentu należącego do paczki narzędzi wysokiego poziomu - NavMesh Link.

6.11. Dostęp do kodu źródłowego, dokumentacji oraz forów społeczności

Kod źródłowy systemu AlchemyNavigation został udostępniony na warunkach licencji MIT [17], tym samym jest on dostępny do wglądu, modyfikacji i komercyjnego użycia. Dostępna jest również zawierająca przykłady i samouczki dokumentacja. System nie posiada jednak aktywnych forów, które mogły by posłużyć do wymiany wiedzy przez użytkowników. Taki układ może być preferowany przez średnio-zaawansowanych i zaawansowanych programistów, którzy bazując na swoim doświadczeniu są w stanie szybciej uzyskiwać odpowiedzi na pytania poprzez analizę dokumentacji i kodu.

Natomiast w przypadku systemu NavMesh dostępna jest bardzo szczegółowa dokumentacja i fora społeczności, a kod jest zamknięty. W tym wypadku grupą faworyzowaną są użytkownicy początkujący, którzy szukają gotowych rozwiązań lub samouczków.

7. Wnioski

Systemy nawigacyjne to złożone i wielopoziomowe narzędzia, które wciąż mogą być rozwijane ze względu na oferowane funkcjonalności oraz wykorzystywane algorytmy.

AlchemyNavigation i NavMesh stanowią przykład w pełni rozwiniętych systemów nawigacji, które oferują szeroką gamę funkcji i gotowych rozwiązań. W zakresie silnika Unity stanowią dobry punkt początkowy do implementacji sztucznej inteligencji agentów, których celem jest przemierzanie świata gry.

Po dokonanej analizie rezultatów i cech badanych systemów należy stwierdzić, że oba systemy nawigacji posiadają podobny zestaw funkcji i generują zbliżone obciążenie. Posiadają również cechy, dzięki którym można uznać je za rozwiązania uniwersalne.

System AlchemyNavigation znajduje lepsze zastosowanie w sytuacjach, kiedy potrzebna jest duża swoboda ingerencji w przestrzeń nawigacyjną w czasie wykonania programu. Mogą być to sytuacje, kiedy gracz posiada opcje budowania środowiska gry lub kiedy jest ono generowane proceduralnie. Ten wniosek motywowany jest faktem, że system opiera się na dynamicznym modyfikowaniu przestrzeni nawigacyjnej (z dokładnością do trójkątów). Natomiast dostarczane komponenty, za pomocą abstrakcji, ukrywają złożoność systemu i pozwalają na wykonywanie uproszczonych operacji.

Kolejną ważną cechą systemu AlchemyNavigation jest dostępny do wglądu i edycji kod źródłowy oraz domyślna możliwości tworzenia i zastępowania poszczególnych modułów (np. możliwość stworzenia własnego typu agentów i budowania modyfikatorów

agenta domyślnego). Takie właściwości w dużym stopniu pozwalają dostosować system do wymogów produktu końcowego, ale znajdą lepsze zastosowanie u lepiej doświadczonych użytkowników.

Ważnym odnotowanym dla systemu AlchemyNavigation wynikiem jest wysoki wzrost obciążenia zasobów stacji testowych, kiedy aktywnych jest wielu agentów oraz funkcja unikania kolizji. System udostępnia możliwości optymalizacji, takiej jak dzielenie agentów na grupy lub nadpisanie modułu kolizji, ale dla niedoświadczonych programistów takie podejście może stanowić poważną przeszkodę.

System NavMesh posiada nieznacznie mniejszą liczbę funkcji, które dodatkowo zostały podzielone na pakiet domyślny i rozszerzający. Takie podejście, wspierane przez szczegółową dokumentację oraz aktywne fora, skutkuje znacznie większą przystępnością narzędzia. Przez co system jest odpowiedni zarówno dla użytkowników doświadczonych, jak i początkujących.

Kluczową cechą systemu NavMesh jest również jego stabilność. Liczba klatek na sekundę jest wysoka, nawet kiedy aktywne są setki agentów, a różnica obciążenia generowane przez aktywność funkcji unikania kolizji nie jest duża.

Finalnie, przedstawione badania i analiza dowodzą tezy, że własne rozwiązania mogą dorównywać rozwiązaniom domyślnym, a także, że pozwalają wprowadzać nowe ciekawe funkcjonalności.

Literatura

[1] D. Silver, Cooperative Pathfinding, In Proceedings of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainmen (2005) 117-122.

[2] D. Demyen, M. Buro, Efficient Triangulation-Based Pathfinding, In Proceedings of the 21st National Conference on Artificial Intelligence (2006) 942-947.

[3] U. Huh, S. Chang, A G² Continuous Path-smoothing Algorithm Using Modified Quadratic Polynomial Interpolation, International Journal of Advanced Robotic Systems 11 (2014) 224-234.

[4] P. Lester, A* Pathfinding for Beginners, https://www.gamedev.net/reference/articles/article2003.asp, [20.06.2021].

[5] Z. He, M. Shi, C. Li, Research and application of path-finding algorithm based on unity 3D, In Proceedings of the 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS) (2016) 1-4.

[6] S. Rabin, Game AI Pro 2: Collected Wisdom of Game AI Professionals, CRC Press, Boca Raton, 2015.

[7] S. Rabin, Game AI Pro 360: Guide to Tactics and Strategy, CRC Press, Boca Raton, 2019.

[8] J. Palacios, Unity 2018 Artificial Intelligence Cookbook: Over 90 recipes to build and customize AI entities for your games with Unity, 2nd Edition, Packt Publishing, Birmingham, 2018.

[9] A. Thorn, Geometric and Discrete Path Planning for Interactive Virtual Worlds, Morgan & Claypool Publishers, San Rafael, 2016.

[10] D. Aversa, A. S. Kyaw, C. Peters, Unity Artificial Intelligence Programming: Add powerful, believable, and fun AI entities in your game with the power of Unity 2018!, 4th Edition, Apress Publishing, Birmingham, 2018.

[11] R. Barrera, Unity 2017 Game AI Programming - Third Edition: Leverage the power of Artificial Intelligence to program smart entities for your games, Packt Publishing, Birmingham, 2018.

[12] Dokumentacja systemu AlchemyNavigation, https://kempnymaciej.github.io/alchemy-navigation/index.html, [20.06.2021].

[13] Dokumentacja systemu NavMesh, https://docs.unity3d.com/Manual/Navigation.html, [20.06.2021].

[14] Dokumentacja pakietu Unity.Profiling, https://docs.unity3d.com/ScriptReference/Unity.Profiling.ProfilerRecorder.html, [20.06.2021].

[15] Dokumentacja pakietu Unity.Addressables, https://docs.unity3d.com/Packages/com.unity.addressables@0.4/manual/index.html, [20.06.2021].

[16] F. Metzger, A. Rafetseder, C. Schwartz, T. Hoßfeld, Games and Frames: A Strange Tale of QoE Studies, In Proceedings of the International Conference on Quality of Multimedia Experience (2016) 1-2.

[17] Licencja MIT, https://choosealicense.com/licenses/mit/, [20.06.2021].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 225–229
Received: 21 June 2021
Accepted: 4 July 201

Comparison of the compilation speed of the SCSS and LESS preprocessors

Porównanie szybkości kompilowania kodów preprocesorów SCSS oraz LESS

Andrii Berkovskyy*, Kostiantyn Voskoboinik, Marcin Badurowicz

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

Abstract
This article compares the compilation speed of the SCSS and LESS preprocessor codes. Each preprocessor has its own syntax, which is transpiled into the CSS stylesheet language in the further development of the web page. These technologies serve the same purpose - to simplify and speed up the writing of page views, but are based on different programming languages - Ruby (SCSS) and JavaScript (LESS). As part of the research, the compilation times of the codes in the SCSS and LESS format with sizes of 10, 50, 100 and 200 kB were measured, the obtained results clearly showed that LESS turned out to be a faster tool for processing the code into CSS syntax. The size of the CSS result code was also taken into account in the analyzes.

Keywords: CSS preprocessors; compilation speed comparison; SCSS; LESS

Streszczenie
Ten artykuł jest poświęcony porównaniu szybkości kompilowania kodów preprocesorów SCSS oraz LESS. Każdy preprocesor ma swoją składnię, która w ciągu dalszym tworzenia strony webowej jest transpilowana do składni języka arkuszy stylów CSS. Technologie te służą temu samemu celowi – uproszczeniu i przyśpieszeniu pisania widoków stron, ale bazują się na różnych językach programowania – języku Ruby (SCSS) oraz JavaScript (LESS). W ramach przeprowadzonych badań dokonano pomiarów czasów kompilacji kodów w formacie SCSS i LESS o rozmiarach 10, 50, 100 i 200 kB, otrzymane wyniki jednoznacznie pokazały, że szybszym narzędziem do przetwarzania kodu na składnię CSS okazał się LESS. W analizach pod uwagę również wzięto wielkość kodu wynikowego CSS.

Słowa kluczowe: preprocesory CSS; porównanie szybkości kompilowania; SCSS; LESS

*Corresponding author
Email address:

1. Wstęp

Coraz więcej ludzi korzysta z usług webowych, a zaprogramowanie stron internetowych jest jednym z popularniejszych rodzajów działalności programistów. Tworzenie efektywnych widoków stron może odbywać się na kilka sposobów, np., wykorzystywanie konstruktorów stron webowych z gotowymi szkicami, które twórca tylko wypełnia treścią, lub pisanie unikalnych kaskadowych arkuszy stylów CSS (Cascading Style Sheets) co zapewni stronie unikalność w porównaniu do innych.

Żeby ułatwić pracę programistom i wnieść do programowania widoków stron cechy właściwe językom obiektowym zostały wymyślone preprocesory CSS. Transpilator kodu korzystając z preprocesora przetwarza kod napisany w składni preprocesora na kod CSS. Korzystanie z tych technologii zaoszczędza czas, ponieważ główną zaletą preprocesorów jest wykorzystywanie zmiennych z możliwością przypisania do nich cech potrzebnych do realizacji widoku. Później napisany kod zostanie transpilowany do zwykłego CSS, który jest zrozumiały dla przeglądarek internetowych.

Aby skrócić czas programowania w tych technologiach jeszcze bardziej, można posłużyć się preprocesorem, który szybciej przetwarza kod na składnie CSS. Autorzy niniejszej pracy pragną znaleźć odpowiedź pytanie dotyczące wydajności tego typu narzędzi.

Do realizacji badań dotyczących sprawdzenia szybkości przetwarzania kodów źródłowych został stworzony internetowy transpilator rozróżniający gramatykę obu preprocesorów SCSS (Sassy CSS) oraz LESS (Leaner CSS), pozwalający mierzyć czas transpilacji oraz pobierać pliki końcowe w formacie CSS. Jako wyniki pracy preprocesorów pozwalające wskazać bardziej efektywne narzędzie, będą brane pod uwagę czasy transpilacji. Preprocesor, który szybciej przetworzy kod źródłowy na kod CSS, będzie określany szybszym.

2. Przegląd prac o podobnej tematyce

Istniejące prace na czas obecny przedstawiają sobą ogólny opis badanych preprocesorów, ich zalety i funkcjonalności. Żadna z przejrzanych przez autorów prac nie przedstawia w sposób eksperymentalny wyników do wskazania preprocesora, który szybciej przetwarza kod swojej składni na składnie CSS. Do takich prac należą książki oraz prace naukowe [1-3].

Wymienione wyżej prace nie przedstawiają eksperymentów nad czasem kompilacji, które będą wykonane przez autorów w ciągu dalszym danej pracy.

Istnieją również artykuły i książki naukowe prezentujące dane eksperymentalne biorące pod uwagę funkcjonalności badanych preprocesorów, a nie czasu ich pracy. Można tu wymienić pracę D. Mazinanian & N. Tsantalis pod tytułem "An empirical study on the use of CSS preprocessors." [4], w której można znaleźć tabele i wykresy danych dotyczących zbiorów funkcjonalności preprocesorów SCSS i LESS. Tu mogą być znalezione dane statystyczne wykorzystywania mixinów (fragmenty kodu, które mogą być wywołane wielokrotnie) w tych preprocesorach lub np., o ile w tym czy innym preprocesorze częściej jest stosowane zagnieżdżanie. Ale jak i w przypadku wyżej wymienionych prac, nie ma tu eksperymentów i otrzymanych danych opisujących czasy kompilacji.

Prace, w których były przedstawione czasy kompilacji - „A survey on CSS preprocessors." [5] oraz „CSS preprocessing: Tools and automation techniques" [6] autorstwa Queirós, R., gdzie autor wykorzystuje tabele pokazującą czas kompilacji preprocesorów w zależności od rozmiaru pliku wejściowego bez żadnych opisów zawartości treści kodów oraz bez przedstawienia dodatkowych informacji o plikach wejściowych. Pozostała część treści wyżej wymienionych artykułów danego autora również jest poświęcona opisowi możliwości, funkcjonalności i wykorzystaniu danych preprocesorów.

3. Technologie

Niniejszy artykuł opisuje badania na preprocesorach CSS – SCSS oraz LESS. Są to dwie popularniejsze technologie na rynku w czasie obecnym do wykorzystania w swojej dziedzinie. Każdy z nich posiada własną składnię do napisania kodów źródłowych, które w przyszłości są transpilowane na składnie CSS. Kod CSS jest produktem końcowym działania obu preprocesorów.

3.1. Język arkuszy stylów CSS

CSS jest językiem arkuszy stylów opisującym widok strony internetowej. Język ten jest stosowany do definicji kolorów, czcionki, stylów, pozycjonowania oddzielnych bloków oraz innych elementów prezentacji widoku stron webowych. Głównym celem rozwoju CSS było oddzielenie opisu struktury strony webowej (za pomocą HTML lub innych języków) od opisu widoku tej strony, który teraz jest realizowany za pomocą arkuszy stylów CSS [7]. Takie rozdzielenie zwiększa dostępność dokumentu oraz przedstawia dużą elastyczność oraz możliwość sterowania widokiem.

Oprócz tego, CSS pozwala wyświetlać ten sam dokument w różnych stylach lub metodach wyświetlania, takich jak prezentacja ekranowa, prezentacja do druku, czytanie głosowe lub prezentacja na urządzeniach wykorzystujących Alfabet Braille'a.

3.2. Preprocesor SCSS

SCSS – dialekt języka SASS ze składnią zbliżoną do składni języka CSS przeznaczony do uproszczenia tworzenia kodu CSS. Składnia języka jest bardzo elastyczna, biorąca pod uwagę wiele drobiazgów, które są tak pożądane w CSS. Język ten posiada logikę (*@if*, *@each*), matematykę (można dodawać liczby, wiersze i kolory). SCSS jest łatwym do zrozumienia programistom pracującym ze zwykłym CSS i jest lepszym wyborem w przejściu na programowanie widoków stron z wykorzystaniem preprocesorów [8].

3.3. Preprocesor LESS

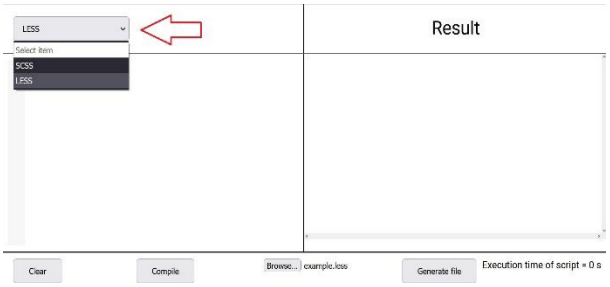
LESS również jest preprocesorem arkuszy stylów CSS. On jest nowszym preprocesorem w odniesieniu do SCSS i jest często z nim porównywany. Jak i SCSS ma on korzenie w języku Ruby, ale w czasie obecnym bazuje się na JavaScript. Wyróżniającą cechą tego preprocesora jest kompilacja kodu na kod CSS przez przeglądarkę w czasie rzeczywistym.

Składnia LESS pozwala na korzystanie z następujących elementów preprocesora [9]:

- zmienne – jednostki z przypisanymi wartościami;
- mixins – wielokrotnie wywoływany fragment kodu;
- zagnieżdżanie – hierarchia wierszy;
- operacje matematyczne.

4. Środowisko badawcze

W celu przeprowadzenia serii testów został stworzony transpilator kodów badanych preprocesorów w postaci aplikacji webowej, dostępnej z przeglądarki internetowej.



Rysunek 1: Interfejs aplikacji testowej.

Na rysunku 1 przedstawiony jest interfejs aplikacji testowej. Można tu zobaczyć dwa główne pola tekstowe. Pole w lewej części służy do wstawiania kodu poprzez opcję wklejania lub za pomocą przycisku „Browse…". Do wyboru preprocesora służy pole w lewej górnej części z etykietą „Select", w którym są dwie opcje – „SCSS" oraz „LESS". Pole tekstowe w prawej części interfejsu wyświetla wynik kompilacji w postaci kodu formatu CSS. Dolną część aplikacji zajmują przyciski „Clear" – do czyszczenia kodu preprocesora, przycisk „Compile" – do rozpoczęcia kompilacji, „Browse" – do wyboru pliku z kodem źródłowym preprocesora oraz „Generate file" – do wygenerowania pliku formatu CSS z kodem wyniku. W tym miejscu również znajduje się pole z wyświetlającym się czasem bieżącej kompilacji.

Do kompilacji kodów preprocesorów napisane oddzielne metody z wykorzystaniem gotowych bibliotek ze źródeł otwartych do transpilacji kodów preprocesorów SCSS [10] oraz LESS [11]. Podczas wyboru preprocesora jednocześnie jest wybierana metoda z odpowiednią biblioteką do transpilacji kodu wybranego preprocesora.

Do funkcji transpilatora należą:

- wybór preprocesora do transpilacji;
- wczytanie fragmentów kodu poprzez wstawianie kodu w odpowiednie pole lub poprzez plik wejściowy z danym kodem;
- wyświetlenie wyników transpilacji kodu preprocesora w formacie CSS;
- pomiar czasu wykonywanej transpilacji;
- generowanie i pobieranie pliku wyjściowego z wynikiem transpilacji.

Funkcjonalność środowiska testowego nie jest obciążona zaawansowanym widokiem czy niewykorzystywanymi funkcjami, które mogą skutkować na niepoprawne wyniki testów. Transpilator tylko i wyłącznie otrzymuje kod od użytkownika i przetwarza ten kod wybranym preprocesorem na składnie CSS.

5. Scenariusze badań

Badania zostały przeprowadzone na zbiorach kodów o różnych rozmiarach plików wejściowych dla każdego z preprocesorów. Takie podejście pozwoli zachować lepsze podobieństwo danych wejściowych, w przeciwieństwu, np., liczby linijek kodu. Spowodowano to różnicą w składni preprocesorów oraz możliwością wypełnienia linijek mniej obciążoną składnią, co w konsekwencji może doprowadzić do mniej prawdopodobnych badań.

Zbiory kodów poddanych eksperymentom są przedstawione plikami formatów SCSS oraz LESS o rozmiarach 10KB, 50KB, 100KB oraz 2000KB dla każdego preprocesora. Pliki o rozmiarze 200KB będą górną granicą pozwalającą zobaczyć różnicę pracy preprocesorów z dużą objętością danych wchodzących.

W środowisku badawczym do każdego z preprocesorów po kolei zostaną wczytane pliki o przedstawionych rozmiarach. Dla uwiarygodnienia wyników, badania zostały powtórzone 5 razy, a następnie uśrednione. Według autorów pracy, ta liczba kompilacji każdego pliku jest wystarczającą, ponieważ czasy kompilacji są o dużym stopniu spójne i w pełni ilustrują zachowywanie preprocesorów z odpowiednim fragmentem kodu. Po każdym kolejnym eksperymencie jest zapisywany czas kompilacji z pola „Execution time of script" oraz wygenerowany plik wyjściowy formatu CSS do sprawdzenia jego rozmiaru.

W wyniku opisanych badań otrzymano zbiory danych do obróbki w postaci 5-ciu czasów kompilacji dla każdego rozmiaru pliku każdego preprocesora oraz pliku wyjściowego formatu CSS.

6. Wyniki badań

Niżej zostaną przedstawione wyniki badań w postaci zbioru tabeli i wykresów.

Przedstawiony w tabeli 1 zbiór danych pozwoli wyliczyć średni czas kompilacji każdego z preprocesorów dla każdego rozmiaru pliku.

Tabela 1: Zbiór czasów kompilacji dla każdego rozmiaru pliku wejściowego

		Czas kompilacji (s)	
		SCSS	LESS
Rozmiary plików wejściowych (KB)	10	0,12617	0,00990
		0,07447	0,00914
		0,07518	0,00887
		0,07430	0,00914
		0,07808	0,01156
	50	1,80656	0,04972
		1,64693	0,04713
		1,78883	0,04710
		1,62705	0,04607
		1,67479	0,04797
	100	6,68259	0,10307
		5,88420	0,09813
		6,26396	0,10444
		6,54755	0,09945
		6,46393	0,09610
	200	29,2196	0,20374
		25,8104	0,22330
		25,3238	0,21979
		24,6682	0,21164
		24,4384	0,22547

Tabela 2: Średnie czasy kompilacji dla każdego rozmiaru pliku wejściowego

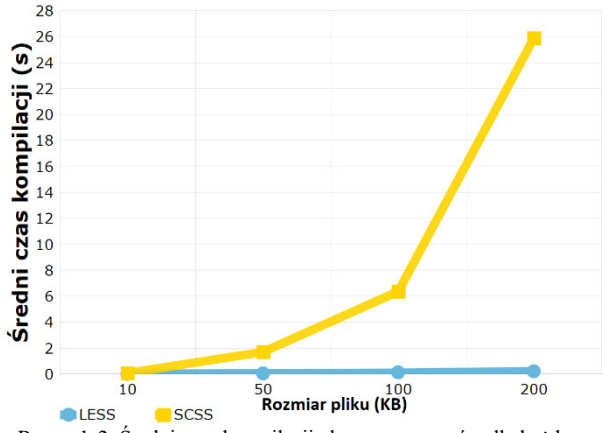
		Średni czas kompilacji (s)	
		SCSS	LESS
Rozmiary plików wejściowych (KB)	10	0,08564	0,00972
	50	1,70883	0,0476
	100	6,36845	0,10024
	200	25,89208	0,21679

Zbiór danych przedstawiony w tabeli 2 pozwala zobaczyć o ile średnie czasy kompilacji obu preprocesorów różnią się. Jednocześnie, dane przedstawione w taki sposób są łatwiejsze do obejrzenia różnicy w pracy preprocesorów niż zbiór danych składający się ze wszystkich czasów kompilacji wszystkich eksperymentów.

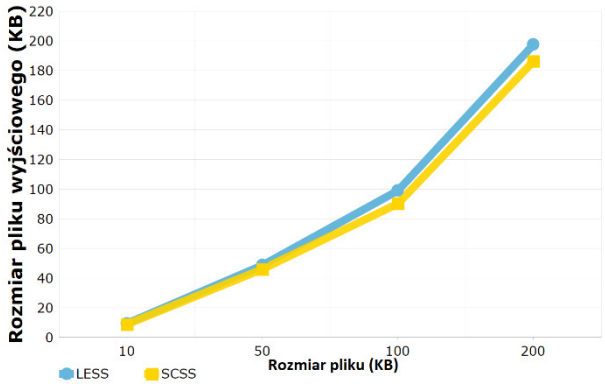
Tabela 3: Rozmiary plików wynikowych formatu CSS

		Rozmiary plików wyjściowych formatu CSS (KB)	
		SCSS	LESS
Rozmiary plików wejściowych (KB)	10	9	10
	50	45	48
	100	89	97
	200	182	193

Tabela 3 zawiera dane dotyczące rozmiarów plików wyjściowych formatu CSS, co pozwala przedstawić efektywność mechanizmów kompresji plików zawartych w preprocesorach. Dane wskazują na to, że preprocesor SCSS lepiej kompresuje rozmiar pliku wyjściowego.



Rysunek 2: Średni czas kompilacji obu preprocesorów dla każdego rozmiaru pliku wejściowego.



Rysunek 3: Rozmiary plików wyjściowych dla każdego rozmiaru pliku wejściowego

Przedstawione wyżej rysunki ilustrują wykresy, na których pokazano porównanie średnich czasów kompilacji oraz porównanie rozmiarów plików wyjściowych.

7. Wnioski

Porównując preprocesory SCSS oraz LESS w szybkości kompilacji kodów, można stwierdzić, że preprocesor LESS jest szybszy we wszystkich przedstawionych przypadkach badań dla wszystkich rozmiarów plików źródłowych. Różnica w czasie kompilacji jest widoczna już na etapie kompilacji plików o najmniejszym rozmiarze (10KB). Chociaż w takim przypadku ona jest najmniejsza i różni się o setną sekundy.

Wraz ze wzrostem rozmiaru (objętości) kodu źródłowego, a odpowiednio i rozmiaru pliku wejściowego, różnica w czasie kompilacji też znacznie rośnie, ponieważ preprocesor LESS kontynuuje przetwarzać kod nawet dla pliku o rozmiarze 200KB mniej niż za sekundę, w przeciwieństwie do preprocesora SCSS, któremu na taką pracę już potrzebne są od 24 do 29 sekund.

Jedynym aspektem, w którym preprocesor SCSS prezentował nieco lepsze wyniki, to jest rozmiar pliku wyjściowego formatu CSS. Ze względu na lepszą optymizację swojej składni preprocesora SCSS na składnie CSS, pliki wyjściowe mają mniejszy rozmiar niż przy wykorzystaniu preprocesora LESS, co ilustrują tabela 3 oraz rysunek 3. Również, różnice w poszczególnych przypadkach rozmiarów wynoszą około 10KB, co nie można nazwać wadą preprocesora LESS.

8. Podsumowanie

Ze względu na elastyczność i łatwość pracy z preprocesorami arkuszy stylów, programiści coraz częściej korzystają z tych technologii. Preprocesory wnoszą funkcjonalność zwykłych języków programowania, co ułatwia pracę i skraca czas pisania kodu w składni tych technologii.

Celem niniejszej pracy było określenie preprocesora, który szybciej skompiluje kod ze swojej składni na składnie CSS, co w przyszłości pozwoli użytkownikom tych technologii nie wybierać między dwoma, a pracować od razu na lepszej.

Z wyników eksperymentów, które były splanowane i przeprowadzone twórcami danej pracy, można powiedzieć, że preprocesor LESS jest technologią, która zapełni szybszą pracę. Preprocesor ten pokazuje czas kompilacji o wiele mniejszy niż preprocesor SCSS. Nawet dla plików w zakresie od 100 do 200 kilobajtów, czas potrzebny preprocesorowi LESS będzie mniejszy jednej sekundy.

Z drugiej strony, jeżeli twórca widoków stron zaczyna pracować z preprocesorami od „zera” i te technologie nie są mu znane dotychczas, nauczenie się składni też nie będzie decydującym czynnikiem, ponieważ trzeba będzie nauczyć się zarówno jednej tak i drugiej technologii jako całkiem nieznanej, a nie można stwierdzić, że jedna jest łatwiejsza do nauczania lub trudniejsza od drugiej.

Zatem, biorąc pod uwagę, że preprocesor LESS o wiele szybciej przetwarza kod i nie ma znaczących wad w porównaniu do preprocesora SCSS, można jego nazwać lepszą technologią do korzystania w pisaniu arkuszy widoków. Taki wniosek opiera się na eksperymentach i ich wynikach przeprowadzonych w punkcie 6 niniejszej pracy.

Literatura

[1] A. Prabhu, Introduction to Preprocessors. In Beginning CSS Preprocessors, Apress, Berkeley, CA (2015) 1-12.

[2] M. Dowden, M. Dowden, Preprocessors. In Architecting CSS, Apress, Berkeley, CA (2020) 165-180.

[3] T. Laukkanen, CSS preprocessor-Sass eller Less. Toni (2017).

[4] D. Mazinanian, N. Tsantalis, An empirical study on the use of CSS preprocessors. In 2016 IEEE 23rd international conference on Software Analysis, Evolution, and Reengineering (SANER), IEEE, Vol. 1 (2016) 168-178.

[5] R. Queirós, A survey on CSS preprocessors. SLATE (2017).

[6] R. Queirós, CSS preprocessing: Tools and automation techniques, Information (2018) 9(1) 17.

[7] M. W. Alawar, S. S. Abu-Naser, CSS-Tutor: An intelligent tutoring system for CSS and HTML (2017).

[8] Możliwości preprocesora SCSS, https://sass-lang.com/guide , [03.05.2021].

[9] Funkcjonalność preprocesora LESS, https://lesscss.org/, [03.05.2021]

[10] Biblioteka do transpilacji kodu preprocesora SCSS: https://scssphp.github.io/scssphp/, [26.06.2021]

[11] Biblioteka do transpilacji kodu preprocesora LESS: https://leafo.net/lessphp/, [26.06.2021]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 230–236
Received: 22 June 2021
Accepted: 2 July 2021

Performance analysis of machine learning libraries

Analiza wydajności bibliotek uczenia maszynowego

Ewa Justyna Kędziora*, Grzegorz Krzysztof Maksim*

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

Abstract

The paper presents results of performance analysis of machine learning libraries. The research was based on ML.NET and TensorFlow tools. The analysis was based on a comparison of running time of the libraries, during detection of objects on sets of images, using hardware with different parameters. The library, consuming fewer hardware resources, turned out to be TensorFlow. The choice of hardware platform and the possibility of using graphic cores, affecting the increase in computational efficiency, turned out to be not without significance.

Keywords: machine learning; performance; TensorFlow; ML.NET

Streszczenie

W artykule zaprezentowane zostały wyniki analizy wydajności bibliotek uczenia maszynowego. Badania oparte zostały na narzędziach ML.NET i TensorFlow. Przeprowadzona analiza bazowała na porównaniu czasu działania bibliotek podczas wykrywania obiektów na zbiorach zdjęć, przy użyciu sprzętu o różnych parametrach. Biblioteką, zużywającą mniejsze zasoby sprzętowe, okazała się TensorFlow. Nie bez znaczenia okazał się wybór platformy sprzętowej oraz możliwość użycia rdzeni graficznych, mających wpływ na zwiększenie wydajności obliczeń.

Słowa kluczowe: machine learning; wydajność; TensorFlow; ML.NET

*Corresponding authors

Email address: ewa.kedziora@pollub.edu.pl (E. J. Kędziora), grzegorz.maksim@pollub.edu.pl (G. K. Maksim)

1. Wstęp

We współczesnym świecie wykorzystanie technik uczenia maszynowego stało się niezwykle powszechne. Zgodnie z jednym z wielu przeprowadzonych na ten temat sondaży, już w 2019 roku stwierdzono, że prawie połowa szybko rozwijających się przedsiębiorstw ma w planach wdrożenie sztucznej inteligencji w przeciągu roku [1].

Nierozerwalną koncepcją mieszczącą się w ramach sztucznej inteligencji jest uczenie maszynowe (ang. machine learning), gdzie komputer wykrywa jak ma zostać wykonane zadanie dzięki określonym wzorcom i przekazanemu zbiorowi danych. Powstało wiele narzędzi oraz bibliotek wspierających programistów w tworzeniu i trenowaniu modeli, a wraz z rosnącym zapotrzebowaniem zwiększa się liczba dostępnych rozwiązań, które w znacznej części są typu open source.

Algorytmy uczenia maszynowego korzystają z parametrów, opartych na danych szkoleniowych. Dostosowanie danych w taki sposób, aby w jak najlepszy sposób reprezentowały rzeczywistość, pozwala na uzyskanie dokładniejszych wyników i użytecznych analiz, których osiągnięcie nie byłoby możliwe do zbadania przez człowieka lub wymagałoby więcej nakładów pracy [2].

W celu lepszego poznania zastosowań uczenia maszynowego i wcześniejszych badań związanych z wydajnością bibliotek uczenia maszynowego przeprowadzono analizę artykułów o podobnym temacie.

W artykule „Review and comparative analysis of machine learning libraries for machine learning” porównano dokładność czasów rozpoznawania cyfr pisanych pismem odręcznym przy pomocy bibliotek TensorFlow, PyTorch, Theano, Keras oraz SciKit-Learn. Wydajność była mierzona na podstawie czasu treningu sieci neuronowej oraz dokładności rozpoznawania danych pobranych z bazy MNIST. Czas uczenia biblioteki TensorFlow był bardzo podobny do narzędzi Scikit-learn oraz Keras i wynosił od 1 do 3 sekund, co zdecydowanie pokonywało bibliotekę PyTorch, osiągającą czas treningu na poziomie 25 sekund [3].

Autorka książki „Hands-On Machine Learning with SciKit-Learn and TensorFlow”, w swojej publikacji stwierdza, że jednym z większych problemów wydajności jest jej spadek, w przypadku dostarczenia błędnych danych. Jako rozwiązanie takiego problemu wskazane jest reagowanie na nieprawidłowości za pomocą algorytmu wykrywania anomalii, umożliwiającego śledzenie przychodzących danych [4].

Autorzy jednej z prac sugerowali skupienie największej uwagi w kierunku uczenia nadzorowanego, czyli np. na stworzeniu etykiet, prawidłowo opisujących grupę przykładowych obrazów, ponieważ niezawodność uczenia wrasta dzięki ingerencji człowieka i opisaniu danych [5]. Podobne wnioski uzyskali naukowcy realizujący doświadczenia na otwartoźródłowej bazie ImageNet stwierdzając, że rozwiązaniem sprzyjającym wydajności, jest zmierzenie dokładności etykiet w uczeniu nadzorowanym [6].

Algorytm jest nadzorowany przez człowieka aby w efekcie sam, po analizie szeregu przypadków, mógł zwracać poprawne odpowiedzi z jak największą dokładnością. To zagadnienie dotyczy nie tylko wykrywa-

nia obiektów na obrazach ale może służyć również do wielu zadań, w których szukany wynik jest liczbą. Na podstawie informacji statystycznych, warunków atmosferycznych i ograniczeń prędkości, istnieje możliwość określenia liczby możliwych wypadków na drodze lub określenia cen sprzedaży nieruchomości na podstawie wielkości, stanu lub lokalizacji [7]. Zagadnienia tego rodzaju mieszczą się w definicji techniki regresji.

Krokiem w kierunku niezawodności są techniki głębokiego uczenia (ang. deep learning) wykorzystujące sztuczne sieci neuronowe. Symulują one systemy, które odzwierciedlają aspekty działania prawdziwych neuronów. Są odpowiedzialne za wiele ostatnich postępów w uczeniu maszynowym jako całości, szczególnie w takich dziedzinach, jak rozpoznawanie obrazów i przetwarzanie języka naturalnego [8].

Architektura sieci neuronowej nadal znacznie odbiega swoją strukturą od złożonej sieci mózgowej składającej się z ogromnej ilości neuronów. Pozostaje jeszcze kwesta użycia zasobów sprzętowych oraz poboru mocy jaki jest potrzebny przy działaniu sztucznych sieci. Przy najbardziej zaawansowanych operacjach wymagane są duże zasoby energii oraz rozbudowane stacje, co powoduje kolejne problemy przy wykorzystaniu uczenia maszynowego [9].

W dokumentacji biblioteki TensorFlow, jako działanie sprzyjające doskonaleniu modelu, wskazywane jest rozwiązanie wąskich gardeł. Sprawdzenie czy model radzi sobie dobrze jest możliwe dzięki zaimplementowaniu go i profilowaniu, które pomaga w analizie zużycia zasobów sprzętowych- pamięci i czasu [10].

Rozwiązanie, które zaproponowała firma NIVIDA umożliwia wykorzystanie rdzeni graficznych w formie procesów ogólnego przeznaczenia. Narzędzia pozwalające na wykorzystanie kart graficznych pozwalają na przyspieszenie sprzętowe [11].

W artykule „Performance Analysis of Deep Learning Libraries: TensorFlow and PyTorch" porównano dwa narzędzia wykorzystujące rdzenie CUDA: PyTorch oraz TensorFlow. Po analizie wydajności tych bibliotek zauważono, że pierwsza z nich wykorzystuje mniejszy potencjał GPU niż TensorFlow. Podczas badań zwrócono uwagę również na to, że pomimo podobnego użycia jednostki CPU, przy TensorFlow temperatura procesora była kilka stopni wyższa niż przy wykorzystaniu konkurencyjnego rozwiązania. Narzędzie od Google jest również znacznie bardziej proste i intuicyjne. PyTorch natomiast wykorzystuje mniej zautomatyzowane funkcje, co skutkuje większym wkładem włożonym przez programistów [12].

Kolejne powstające platformy mają zminimalizować problemy z wydajnością, unifikacją, rozszerzalnością i skalowalnością, umożliwiając jak najlepszą wydajność przy maksymalnym wykorzystaniu zasobów sprzętowych [13].

TensorFlow jest biblioteką obecną w wielu badaniach, natomiast materiałów dotyczących biblioteki ML.NET jest zdecydowanie mniej. Pierwsza z nich, w zestawieniach z innymi platformami, wielokrotnie osiągała zbliżone wyniki, będąc jednocześnie uzależnioną od parametrów sprzętowych. Analizowane pozycje literaturowe na temat badania wydajności, nie obejmowały porównania tych dwóch platform, co motywuje do przeprowadzenia eksperymentu obejmującego oba rozwiązania, umożliwiającego wybór narzędzia realizującego scenariusz wykrywania obiektów na obrazach w sposób optymalny. Celem autorów jest zbadanie wydajności bibliotek oferowanych przez firmy Microsoft i Google, pod względem szybkości badania i użycia zasobów sprzętowych.

2. Metodyka badań

2.1. Opis badania

W celu przeprowadzenia badania utworzono dwie aplikacje bazujące na algorytmach uczenia maszynowego:

1. Aplikacja konsolowa do badania wydajności biblioteki ML.NET
2. Aplikacja stworzona w środowisku Jupyter Notebook do badania czasu analizy obiektów przez TensorFlow.

Ich działanie sprowadza się do wprowadzania przez użytkownika danych w formie zbiorów obrazów. Następnie zbiór danych ulega przetworzeniu. Aplikacja wykrywa obiekty na obrazach i zapisuje rezultaty do osobnego katalogu. Wyniki z przeprowadzonej analizy są przedstawiane w postaci prawdopodobieństwa i ramek otaczających obiekty, co przedstawione jest na Rysunku 1. Podstawą takiego działania jest użycie prawidłowo wytrenowanego modelu.

Rysunek 1: Obraz po analizie z zaznaczonymi wykrytymi obiektami.

Badanie wydajności działania aplikacji polega na zmierzeniu czasów wykonywania tych samych operacji przez dwie różne biblioteki. W kodzie źródłowym dodano funkcję mierzenia czasu wykonania kodu. Po wykonaniu określonych zadań, czas był wyświetlany na ekranie.

W celu uzyskania jak najbardziej wiarygodnych wyników zostały użyte gotowe, wyuczone modele. Badanie zostało przeprowadzone na dwóch urządzeniach posiadających różną specyfikację, przedstawioną

w Tabeli 1, celem zweryfikowania, które z nich radzi sobie lepiej z narzuconym zadaniem i jakie parametry mają wpływ na wydajność działania bibliotek.

Tabela 1: Specyfikacja urządzeń użytych w badaniu

Urządzenie	Procesor			RAM	GPU
	Typ	Liczba rdzeni	Taktowanie (MHz)	(GB)	
Komputer stacjonarny	AMD Ryzen 7	8	od 3600 do 4400	16	GeForce RTX 3070
Laptop	Intel Core i5	2	od 1900 do 2051	8	-

Pierwsze stanowisko badawcze posiada dodatkowo kartę graficzną, która umożliwiła skorzystanie ze specjalnych rdzeni CUDA. CUDA jest zastrzeżoną platformą programistyczną dostosowaną do masowej równoległości zadań obliczeniowych, a posiadanie rdzeni w liczbie 5888, umożliwia równoległe przetwarzanie 5888 wektoryzowanych jednostek wykonawczych. Za ich pomocą, dzięki możliwościom oferowanym przez język Python, realne jest zbadanie czasu, jakiego potrzebuje sama karta graficzna.

Skorzystano z metody obserwacji polegającej na świadomym obserwowaniu funkcjonowania aplikacji podczas wykrywania obiektów na zbiorze obrazów.

2.2. Środowisko badawcze

W przeprowadzanym badaniu wykorzystane zostały dwie konkurencyjne biblioteki: TensorFlow-platforma oferowana przez Google Brain Team, najbardziej spopularyzowana wśród programistów zajmujących się uczeniem maszynowym i ML.NET-biblioteka od firmy Microsoft stworzona dla języków programowania C# oraz F#.

2.3. Model

Celem implementacji, skorzystano z modelu w formacie ONNX (ang. Open Neural Network Exchange), wytrenowanego na podobnym problemie. Został on pobrany ze zbioru ONNX Model Zoo. Operatory w ONNX są podzielone na zestawy funkcji i prymitywnych operatorów. Funkcja to operator, którego obliczenia mogą być wyrażone jako podgraf innych operatorów. Do ich opisu używany jest graf. W nim znajdują się listy węzłów, wejść, wyjść oraz wartości stałe. Dzięki użyciu rozszerzalnego modelu grafu obliczeniowego możliwe jest użycie go przy wykorzystaniu wielu różnych platform [14].

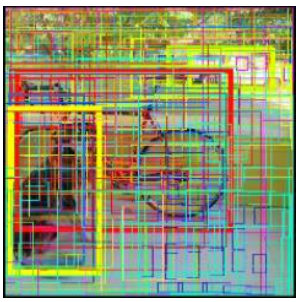
W początkowej fazie działania algorytmu uczenia maszynowego wymagane są obiekty typu DataView. Każdy model zawiera definicję danych wejściowych oraz wyjściowych, zawierającą nazwy, typy i rozmiary danych. Jeżeli któraś z danych jest niepoprawnie zdefiniowana bądź źle zaimplementowana, wówczas w trakcie działania programu mogą wystąpić wyjątki.

Obraz zostaje podzielony przez sieć neuronową na poszczególne regiony, tworząc siatkę o rozmiarach 13x13 (Rysunek 2). Następnie jest szacowane prawdopodobieństwo dla każdego regionu i na jego podstawie przewidywane są obwiednie (Rysunek 3, Rysunek 4).

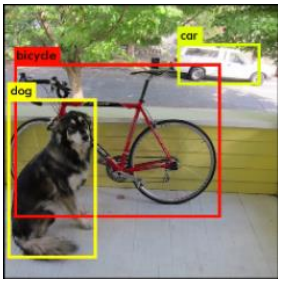
Przy trenowaniu modelu istotne jest najpierw zgromadzenie odpowiednich danych, a następnie ich załadowanie. W kolejnym etapie tworzone są potoki, które model jest w stanie przetworzyć. Następnym krokiem jest trenowanie modelu i jego ocena. Jeżeli model nie jest jeszcze odpowiednio wytrenowany, należy powtórzyć wszystkie czynności do uzyskania odpowiedniego efektu. Jeżeli model zostanie wytrenowany do odpowiedniego poziomu, zostaje zapisany. Proces uczenia go od podstaw pochłania dużo zasobów obliczeniowych i wiąże się z ogromną liczbą parametrów do skonfigurowania, dlatego też tak użyteczne są projekty społecznościowe jak ONNX, gdy samo wytrenowanie modelu nie jest tak istotne dla zakresu badań.

Rodzaje modeli ONNX można podzielić na dwa rodzaje: ONNX skupiony na sieciach neuronowych, który rozpoznaje tensory jako typy wejściowe i wyjściowe oraz ONNX-ML, pozwalający na rozpoznawanie sekwencji i map. Drugi typ wykorzystuje modele, które nie zostały stworzone w oparciu o sieci neuronowe.

Rysunek 2: Schemat działania algorytmu wykrywania obiektów na obrazie, etap 1 [15].



Rysunek 3: Schemat działania algorytmu wykrywania obiektów na obrazie, etap 2 [15].



Rysunek 4: Schemat działania algorytmu wykrywania obiektów na obrazie, etap 3 [15].

Wykorzystany w badaniach model to Tiny YOLO2. Jest to sieć neuronowa, pozwalająca na wykrywanie obiektów w czasie rzeczywistym [15][16]. W czasie testu jest analizowana cała fotografia, więc prognozy bazują na globalnym kontekście obrazu. Modele YOLO charakteryzują się bardzo dużą szybkością wykrywania obiektów oraz wysoką dokładnością. Dzięki wytrenowaniu modeli na procesorach GPU, modele te uzyskały ogromną dokładność i bardzo dobrą szybkość. Rozmiary modeli jednak są dosyć duże i często przy środowiskach z ograniczoną pamięcią nie są one w stanie pracować w zadowalający sposób [17].

2.4. Implementacja biblioteki TensorFlow

Za pośrednictwem biblioteki TensorFlow, dane są przetwarzane w postaci grafu. Działanie to wymaga przekształcenia obiektów do postaci tensora. Tensorami są krawędzie grafu łączące wierzchołki, które z kolei odzwierciedlają działania matematyczne. Podczas przetwarzania obrazu odbywa się konwersja, co skutkuje powstaniem tablicy liczb, która reprezentuje poddawany analizie obraz i na tym etapie tablica może być weryfikowana przez model. Niezbędnym krokiem jest określenie, które dane mają być podane na wyjściu modelu i przetworzone, ponieważ sam model składa się z danych wejściowych i wyjściowych. Powyższe działania zostały przeprowadzone z użyciem platformy opensource TensorFlow Object Detection API.

Implementacja tej biblioteki przeprowadzona została w środowisku do wykonywania aktywnego kodu krok po kroku tj. Jupyter Notebook w języku Python. TensorFlow jest narzędziem wszechobecnym we współczesnej technologii, dlatego też wdrożenie aplikacji, bazującej na tej platformie można wykonać na większości urządzeń m.in. komputerach lokalnych, procesorach i procesorach graficznych, klastrach, oraz na urządzeniach z systemem Android i iOS.

Celem prawidłowego przebiegu eksperymentu zostały użyte dodatkowe pakiety:

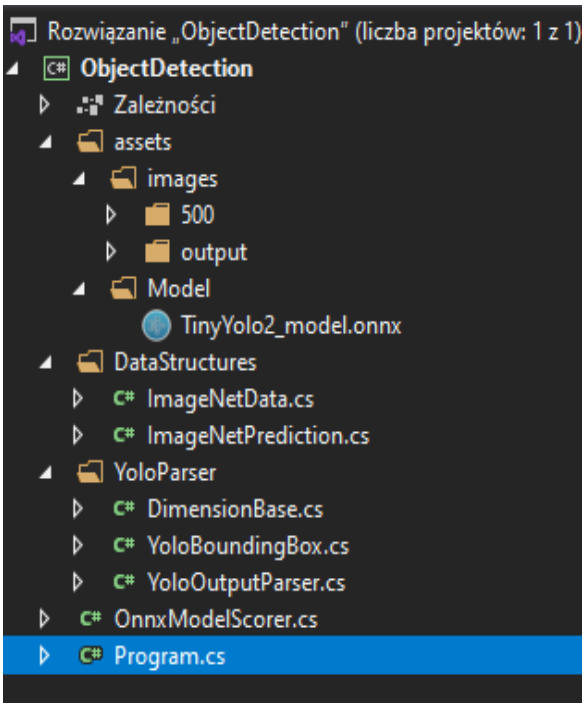
1. Pakiet pillow-umożliwiający przetworzenie obrazów do języka akceptowalnego przez język Python,
2. Pakiet jupyter-powiązany z możliwością tworzenia arkuszy do wykonywania kodu w Pythonie.
3. Pakiet lxml-umożliwiająca przetwarzanie plików w formacie xml.

Baza obrazów wykorzystanych do przeanalizowania przez algorytmy w eksperymencie badawczym to COCO. Dodatkowo użyto pakietu API umożliwiającego wykorzystanie powyższej bazy w projekcie.

2.5. Implementacja biblioteki ML.NET

Gotowa struktura katalogów projektu została przedstawiona na rysunku 5. Folder Model, zawiera wcześniej wspomniany model TinyYolo2_model.onnx. Katalog assets zawiera testowane obrazy, natomiast zdjęcia przetworzone z wykrytymi obiektami mieszczą się w katalogu output.

Zdefiniowanie zmiennych, które reprezentują lokalizację folderów z obrazami wraz z zawierającym model katalogiem zostały uwzględnione w klasie Main.



Rysunek 5: Struktura katalogów projektu w środowisku Visual Studio.

Tworzenie obiektów z klas wcześniej przedstawionych, zawarte jest w bloku try-catch. To w nim znajduje się m.in. obiekt, który przechowuje informację o prawdopodobieństwie, zadeklarowana zostaje instancja obwiedni, a także wraz z głównym kodem aplikacji następuje obliczanie czasu wykonania kodu.

Jednym z głównych obiektów tworzonych w tym miejscu jest także model punktowy dla modelu w formacie ONNX.

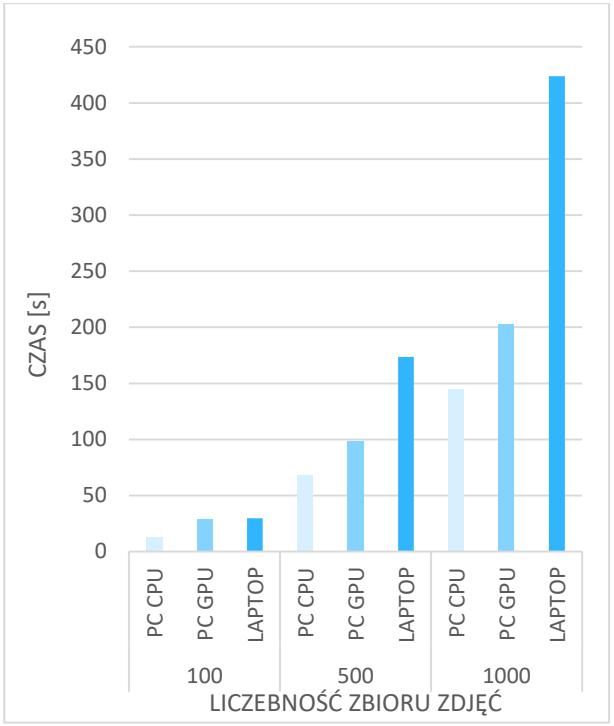
3. Wyniki badań

Metodyka badawcza, polegała na porównaniu trafności procesu wykrywania obiektów na obrazach, przez wybrane, konkurencyjne biblioteki oraz na zweryfikowaniu szybkości działania poszczególnych bibliotek na urządzeniach posiadających odmienne parametry.

Uzyskane wyniki podzielone zostały pod kątem liczebności zdjęć oraz urządzeń, na którym badanie zostało przeprowadzone. Podział ten, umożliwiał przedstawienie różnic w działaniach aplikacji i przede wszystkim w funkcjonowaniu algorytmów.

W przypadku biblioteki TensorFlow zdjęcia podzielone zostały na trzy zbiory zawierające 100, 500 i 1000 obrazów. Dla każdego ze zbiorów wyniki, wyrażone w sekundach, prezentowały ten sam trend. Czas przetwarzania obrazów w każdej z badanych grup, był najkrótszy dla CPU, wynosząc kolejno, 13, 68 i 145 sekund. Analiza obrazów z użyciem karty graficznej i wspomnianych w rozdziale 3 rdzeni CUDA wyniosła 29, 99 i 203 sekundy.

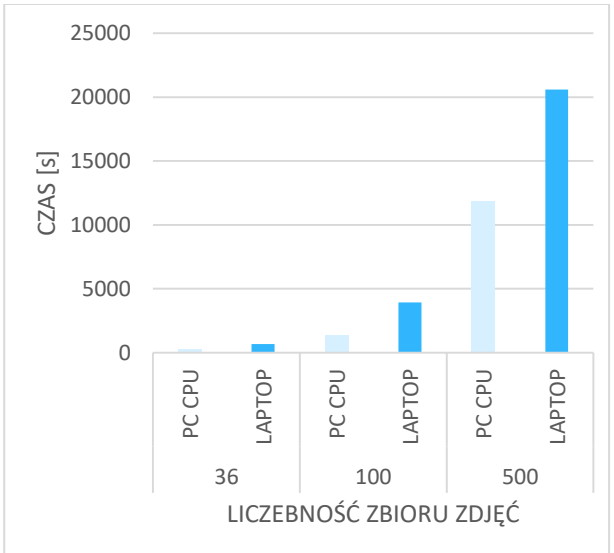
Wyniki na laptopie z wbudowaną kartą graficzną wyniosły 30, 174 i 424 sekundy. Zauważalny jest dwukrotny wzrost czasu pomiędzy GPU i laptopem na korzyść karty graficznej (Rysunek 6).



Rysunek 6: Czas analizy zbiorów zdjęć wykonanej na Laptopie, GPU i CPU z wykorzystaniem biblioteki TensorFlow.

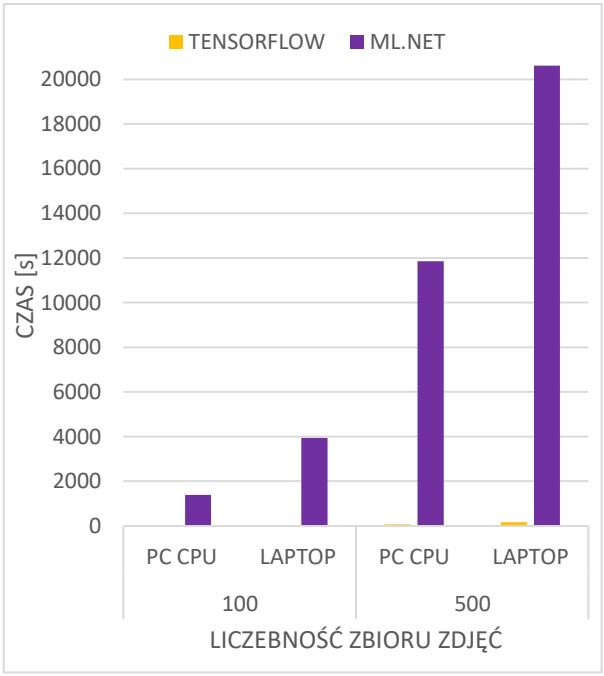
Podczas analizy biblioteki ML.NET zdjęcia podzielone zostały na trzy zbiory: 36, 100 i 500 obrazów. W czasie trwania eksperymentu prowadzono próby przeprowadzenia go na liczebniejszych zbiorach, jednak takie działanie było ograniczane przez możliwą do wykorzystania pamięć RAM.

Czas, poświęcony przez program na przetworzenie fotografii i wykrycie obiektów, cechował się podobną tendencją co poprzednia biblioteka. Najkrótsze wyniki zostały osiągnięte za pośrednictwem CPU, wynosząc kolejno: 250, 1393 oraz 11846 sekund. Mniej korzystne czasy zostały osiągnięte na laptopie, tj. drugim badanym w przypadku ML.NET urządzeniu, wynosząc 671, 3941 i 20605 sekund (Rysunek 7).



Rysunek 7: Czas analizy zbiorów zdjęć wykonanej na Laptopie i CPU z wykorzystaniem biblioteki ML.NET.

Zestawienie wyników uzyskanych z obu aplikacji wykazało ponad 100 razy dłuższy czas poświęcony na wyznaczone zadanie w badaniu przeprowadzonym za pomocą ML.NET (Rysunek 8).



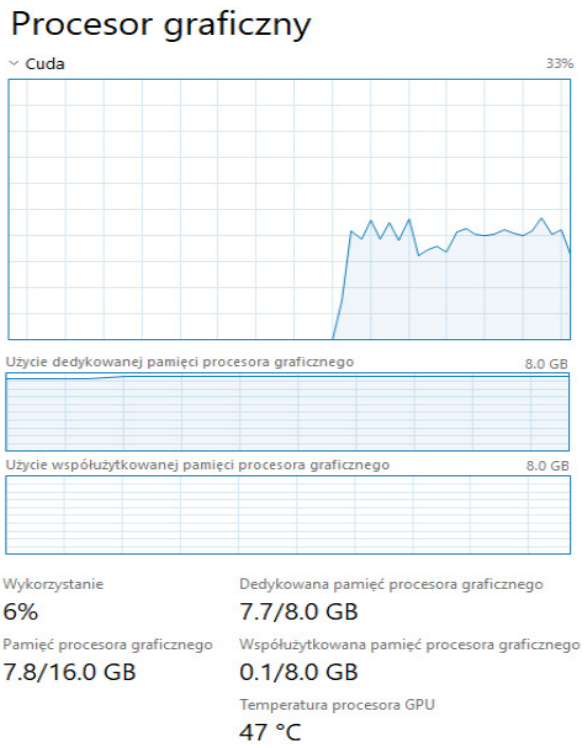
Rysunek 8: Graficzne porównanie czasu wykrycia obiektów na obrazach przez biblioteki TensorFlow i ML.NET dla różnych środowisk badawczych.

Przy próbie analizy zbioru zdjęć składającego się z 1000 zdjęć, został otrzymany komunikat o braku zasobów pamięciowych. Podobna sytuacja, jednak przy zbiorze dwukrotnie większym, nastąpiła również dla biblioteki TensorFlow. Użycie procesora było w tym przypadku znacznie niższe i tym samym nie miało to wpływu na otrzymany czas przetwarzania obrazów.

Tabela 2: Porównanie czasu wykrycia obiektów na obrazach przez biblio-teki TensorFlow i ML.NET dla różnych środowisk badawczych

Biblioteka	Środowisko badawcze	Liczba zdjęć	Czas przetwarzania (s)
TensorFlow	PC CPU	100	13
		500	68
	Laptop	100	30
		500	174
ML.NET	PC CPU	100	1393
		500	11846
	Laptop	100	3941
		500	20605

W przypadku użycia w aplikacji możliwości wykorzystania rdzeni graficznych, zaobserwowano wzrost użycia rdzeni CUDA oraz znaczne wykorzystanie dedykowanej pamięci procesora graficznego. Użycie procesora nie zmieniło się znacząco, natomiast czas działania uległ nieznacznemu wydłużeniu (Rysunek 9).



Rysunek 9: Zużycie procesora graficznego podczas analizy obrazów za pomocą TensorFlow, na komputerze stacjonarnym.

4. Wnioski

W tym artykule zostały porównane dwie biblioteki uczenia maszynowego pod kątem czasu poświęconego na wykrycie obiektu znajdującego się na obrazie, oraz zużycia zasobów sprzętowych urządzeń o odmiennej specyfikacji.

Przeprowadzona analiza wyników pozwoliła zauważyć, że biblioteka TensorFlow zdecydowanie lepiej sprawdza się w scenariuszu obejmującym wykrywanie obiektów na zdjęciach niż biblioteka ML.NET.

Na podstawie przeprowadzonych obserwacji potwierdzono tezę, że działanie biblioteki TensorFlow jest uzależnione od platformy sprzętowej. Wykorzystanie rdzenia graficznego i rdzeni CUDA pozwala na zwiększenie wydajności obliczeń. Liczba przetwarzanych grafik nie wpłynęła znacząco na długość wykonywania algorytmu rosnąc proporcjonalnie.

W badaniach zrealizowanych za pomocą biblioteki ML.NET analiza fotografii była bardzo powolna, pomimo maksymalnego wykorzystania procesora.

Zdecydowanie widoczna przewaga TensorFlow może wynikać z większej liczby użytkowników, którzy swoje działania związane z zagadnieniem uczenia maszynowego opierają na tej platformie. ML.NET pomimo gorszych osiąganych czasów w przeprowadzonym badaniu, nadal dociera do dużego grona użytkowników, z powodu bazowania na językach C# i F#.

Literatura

[1] Leading business in the age of AI, https://news.microsoft.com/europe/features/leaders-look-to-embrace-ai-and-high-growth-companies-are-seeing-the-benefits, [29.11.2020].

[2] Przewodnik po strukturze ML.NET, https://docs.microsoft.com/pl-pl/dotnet/machine-learning/how-does-mldotnet-work, [30.11.2020].

[3] M. N. Gevorkyan, A. V. Demidova, T. S. Demidova, A. Sobolev, Review and comparative analysis of machine learning libraries for machine learning, Discrete And Continuous Models And Applied Computational Science 27 (2019) 305-315, http://dx.doi.org/10.22363/2658-4670-2019-27-4-305-315.

[4] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2019.

[5] E. Brill, M. Banko, Scaling to very large corpora for natural language disambiguation, Proceedings of 39th Annual Meeting on Association for Computational Linguistics (2001) 26-33, https://doi.org/10.3115/1073012.1073017.

[6] V. Shankar, R. Roelofs, H. Mania, A. Fang, B. Recht, L. Schmidt, Evaluating Machine Accuracy on ImageNet, 37th International Conference on Machine Learning (2020) 8634-8644.

[7] E. Zuccarelli, Using machine learning to predict car accidents, https://towardsdatascience.com/using-machine-learning-to-predict-car-accidents-44664c79c942, [15.06.2021].

[8] M. Hartley, T. S. G. Olsson, dtoolAI: Reproducibility for Deep Learning, Patterns 1(5) (2020) 100099, https://doi.org/10.1016/j.patter.2020.100073.

[9] C. Deng, X. Ji, C. Rainey, J. Zhang, W. Lu, Integrating Machine Learning with Human Knowledge, iScience 23(11) (2020) 101656, https://doi.org/10.1016/j.isci.2020.101656.

[10] Optimize TensorFlow performance using the Profiler, https://www.tensorflow.org/guide/profiler, [15.06.2021].

[11] G. Nguyen, S. Dlugolinsky, M. Bobák, V. Tran, Á. García, I. Heredia, P. Malík, L. Hluchý, Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey, Artificial Intelligence Review 52 (2019) 77-124, https://doi.org/10.1007/s10462-018-09679-z.

[12] F. Florencio, E. D. M. Ordonez, T. V. Silva, M. C. Júnior, Performance Analysis of Deep Learning Libraries: TensorFlow and PyTorch, Journal of Computer Science 15 (2019) 785-799, http://dx.doi.org/10.3844/jcssp.2019.785.799.

[13] Z. Ahmed, S. Amizadeh, M. Bilenko, R. Carr, W.-S. Chin, Y. Dekel, X. Dupre, V. Eksarevskiy, E. Erhardt, C. Eseanu, S. Filipi, T. Finley, A. Goswami, M. Hoover, S. Inglis, M. Interlandi, S. Katzenber, Machine Learning at Microsoft with ML.NET, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019) 2448-2458, https://doi.org/10.1145/3292500.3330667.

[14] T. Jin, G.-T. Bercea, T. D. Le, T. Chen, G. Su, H. Imai, Y. Negishi, A. Leu, K. O'Brien, K. Kawachiya, A. E. Eichenberger, Compiling ONNX Neural Network Models Using MLIR (2020), https://arxiv.org/abs/2008.08272.

[15] J. Redmon, YOLO: Real-Time Object Detection, https://pjreddie.com/darknet/yolov2, [10.06.2021].

[16] J. Redmon, A. Farhadi, YOLO9000:Better, Faster, Stronger (2016), https://arxiv.org/abs/1612.08242.

[17] W. Fang, L. Wang, P. Ren, Tinier-YOLO: A Real-Time Object Detection, IEEE Access 8 (2019) 1935-1944, https://doi.org/10.1109/ACCESS.2019.2961959.

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.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 243–246
Received: 24 June 2021
Accepted: 9 July 2021

Comparative analysis of online stores

Analiza porównawcza sklepów internetowych

Arkadiusz Zbigniew Wójtowicz*, Marek Miłosz

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

Abstract

This article raises issues of online stores. An analysis of many aspects of sales platforms was carried out, such as awareness about existence of these platforms, frequency of purchases, interface quality, intuitiveness, safety and their popularity. Allegro, Amazon, Aliexpress, Ebay and Wish were selected from the most popular online stores. In addition, a survey was conducted and analysed, which examines the awareness and experiences of online stores users. This article allowed us to draw relevant conclusions about these sales platforms and their community.

Keywords: e-commerce; online stores

Streszczenie

Niniejszy artykuł porusza kwestie sklepów internetowych. Przeprowadzono analizę dotyczącą wielu aspektów platform sprzedażowych takich jak świadomość o istnieniu poszczególnych platform, częstotliwość dokonywania zakupów, jakość interfejsu, intuicyjność, bezpieczeństwo i ich popularność. Spośród najpopularniejszych sklepów internetowych wybrano Allegro, Amazon, Aliexpress, Ebay oraz Wish. Ponadto została przeprowadzona i odpowiednio przeanalizowana ankieta, której celem było zdobycie informacji o świadomości oraz o doświadczeniach użytkowników związanych z sklepami internetowymi. Całość pracy pozwoliła wyciągnąć odpowiednie wnioski na temat analizowanych platform sprzedażowych i społeczności ich budujących.

Słowa kluczowe: handel elektroniczny; sklepy internetowe

*Corresponding author

Email address: **arkadiusz.wojtowicz@pollub.edu.pl** (A. Z. Wójtowicz)

1. Introduction

Online stores gain more and more popularity from year to year. The first appeared in the nineties. Every day, their creators develop these platforms, from year to year, gaining new customers. Initially, the greater part of trade in goods was carried out in stationary stores, where customers personally made shopping. However, along with the development of technology, the tendency changed towards e-stores. Many websites and mobile applications have been created, from which they can directly order products to their homes or other pick-up point. Thanks to this, consumers gained the opportunity to buy objects not only in their city, but also around the country, and even worldwide with only one click. In addition, the situation caused by the coronavirus contributed to the next increase in the importance of e-commerce. When most stationary stores were closed for many months, online shopping experienced time of prosperity. According to many experts, this trend will remain with us forever. People have learned to buy online, and this is much easier than a few years ago.

2. Research of the literature

There are many literature positions, both Polish and foreign language, which concern issues related to online stores. A comparative analysis of online stores requires a specialized knowledge that will introduce in details on trade in the network. In this chapter, points of view of individual authors of books or other fields of culture will be presented. The influence of a pandemic situation for internet trade will also be proven, which will help to reduce additional conclusions. In the book by Petr Weinlich and Tereza Semerádova "Website Quality and Shopping Behavior: Quantitative and Qualitative Evidence" [1], the author analyzes the impact of web design parameters on user experience and behavior. The author analyze the concept of website quality based on the identification of user behavior patterns in the online environment, with particular emphasis on the functional and aesthetic parameters of website design and the causal relationships between them. The website design is considered to be one of the key parameters of the company's presentation on the Internet, influencing consumer attitudes and purchasing behavior.

The website "similarweb.com" [2] compares the most popular websites around the world. The analysis took into account such factors as the average visit time on a given website, the average number of subpages that were visited during one visit, or the percentage of people who visited only one page when visiting a given online store. However, the most important factors that influenced the result are the total number of visits as well as the frequency of purchases. The data on this website is updated once a day, therefore the popularity of individual stores may change on this list on a daily basis. However, the first place is invariably occupied by Amazon, which has been winning the global e-commerce market for several years.

Yuan Gao in his book describes the design of Internet systems and consumer behavior on the Internet [3]. It takes an interdisciplinary approach to the design

of systems in an online environment, providing an understanding of how consumers behave when shopping and how certain elements of system design can influence consumer perceptions, attitudes, intentions and actual behavior. He pays special attention to the time course of making purchases, which, in his opinion, should guide only the most important activities through the process, avoiding advertising elements at this point.

David Reibstein in the article "What attracts customers to online stores, and what keeps them coming back?" [4] tried to discover which tactic appeals to online shoppers the most and keeps them on the storefront. This study reveals surveys and behavioral data collected from online customers that reflect what was most important to online shoppers and compares pull and retention factors. As many believed that the Internet created better information for buyers, the question was how important the price would be in the buying process. It is clear from the analysis that what attracts customers to your site are not the same dimensions that are critical to customer retention in the long run.

Customer behavior covers a wide range of processes and activities related to sensory responses, perception, shaping attitudes, preferences, decisions, satisfaction assessment and loyalty building. Customer behavior on the Internet is influenced by exogenous and endogenous factors. External factors include attributes related to the e-merchant and the consumer's environmental influences. Endogenous factors include characteristics attributed to consumers. Of these, personality has a great influence on the behavior of customers in online stores. Costinel Dobre and Anca-Maria Milovan-Ciuta, in their article "Personality influences on online stores customers behavior" [5], draws attention to the influence of personality on important decision variables related to the customer's visit to the Internet, purchase and the post-purchase process, therefore he shows that the influence of personality on the criteria for assessing stores , the expectations of customers towards stores and the perception of the store's operation are important.

3. Research methodology and determination the size of the research sample

The purpose of the article is a comparison of online stores to learn about the differences in individual sites. Aspects such as the quality of the interface, service intuitiveness, security or popularity have been analysed using the survey method.

A survey was also carried out and awareness of individual sales platforms was checked, as well as the frequency of purchases on each of them. To select the appropriate number of people during the test, it was decided to use the sample selection calculator [6]. This is a minimum calculator, the required number of people on a sample from the population, assuming individual parameters. The given value informs how much people should investigate in the study to obtain a given minimum error value in the measurement at a fixed confidence level to a result of 95%. To calculate the result, the size of the population is given in the form of eight million four hundred and sixty thousand. This is the equivalent of the Internet users in Poland with Internet users buying in foreign e-stores. In addition to the size of the population, two parameters remain for addition. This is the size of the fraction and maximum error. The size of the fraction, i.e. the percentage of the population, which exhibits a specific feature was set to 0.9, i.e. it was assumed that the tested features occur in 90% of the population. Last parameter, i.e. the maximum error indicates what the highest value of the measuring error may occur in a given experience. A 5% value has been set, which means that the result of the actual survey may differ. The result of the sample selection calculator showed a number 138, which means that the minimum number of people in the study is required. The survey was made on an attempt of 169 people. This means that the calculator's assumptions have been met.

4. Results of surveys of online stores users

A study was conducted in which e-shops were examined. The survey was conducted among people who regularly shop online. It was created with the tool "Google Forms" [7], which was also used to process the survey. The survey was conducted on the "Facebook.com" [8] platform on groups related to online shopping. A survey consists of eight sections, each containing questions related to a specific issue. 169 respondents were examined. Table 1 shows as many as 79.3% are women. Men account for 20.7% of the respondents.

Table 1: Percentage of men and women participating in the study

Gender	Percentage
Men	20.7%
Women	79.3%

Table 2 shows the percentage of respondents by age.

Table 2: Percentage of respondents by age

Age	Percentage
18-24	69.2%
25-34	22.5%
35-44	4.1%
45-54	3.6%
55+	0.6%

Table 3 shows the results about awareness of the users about online stores. Users were asked if they had ever heard of a given store. The "known" column shows the percentage of respondents who have heard about a given store as opposed to the "unknown" column.

Table 3: Awareness of the users about online stores

Online store	Known	Unknown
Allegro	100%	0%
Amazon	97%	3%
Aliexpress	98.8%	1.2%
Ebay	90.5%	9.5%
Wish	77.5%	22.5%

Table 4 shows the results about the frequency of making purchases on sales platforms.

Table 4: The frequency of making purchases on sales platforms

Online store	Never	Less than once a month	1-3 times per month	4-6 times per month	More than 7 times per month
Allegro	4.1%	56.2%	30.8%	5.9%	3%
Amazon	87.6%	10.1%	1.2%	0.6%	0.6%
Aliexpress	45%	43.8%	8.3%	2.4%	0.6%
Ebay	94.1%	3.6%	1.2%	0.6%	0.6%
Wish	95.3%	3%	1.2%	0%	0.6%

Table 5 shows the results about the interface. Users judged the overall appearance of the interface and they gave 1-5 mark on a five-point scale. Mark 1 if it's bad and 5 mark if it's good.

Table 5: Interface on a five-point scale

Online store	1	2	3	4	5
Allegro	0%	1.2%	7.7%	46.2%	48.5%
Amazon	3.4%	8.1%	50.7%	28.4%	9.5%
Aliexpress	1.9%	11%	40%	36.1%	11%
Ebay	4.8%	8.2%	57.5%	25.3%	4.1%
Wish	6.3%	18.3%	57.7%	15.5%	2.1%

Table 6 shows the results about intuitiveness on a five-point scale. Users rated the ease of navigating the store and they gave 1-5 mark on a five-point scale. Mark 1 if it's bad and 5 mark if it's good.

Table 6: Intuitiveness on a five-point scale

Online store	1	2	3	4	5
Allegro	0.6%	1.8%	4.1%	27.8%	65.7%
Amazon	4.9%	9.2%	47.2%	26.1%	12.7%
Aliexpress	5.2%	11.8%	32%	28.8%	22.2%
Ebay	4.4%	10.2%	52.6%	24.1%	8.8%
Wish	6.6%	11.8%	55.9%	20.6%	5.1%

Table 7 shows the number of times customers have bought products and never received them.

Table 7: Number of times the product was never received

Online store	0	1	2	3 or more
Allegro	91.7%	6.5%	0.6%	1.2%
Amazon	97%	2.2%	0.7%	0%
Aliexpress	59.3%	15.3%	11.3%	14%
Ebay	97.7%	2.3%	0%	0%
Wish	96%	3.1%	0%	0.8%

Table 8 shows the percentage of respondents who most often choose the surveyed online store.

Table 8: Popularity of each online store

Online store	Percentage
Allegro	85.7%
Amazon	1.2%
Aliexpress	12.5%
Ebay	0.6%
Wish	0.6%

5. Analysis of research results

Among the respondents, the Allegro online store is the most popular, as as many as 100% of respondents indicated that they have ever heard of it. It is certainly the most popular shopping platform in Poland. Asking a question about the knowledge of other platforms, i.e. Amazon, Aliexpress, Ebay and Wish, the majority of respondents replied that they had heard about them.

It can be clearly concluded that Allegro is the most frequently chosen store. Only 4.1% of the respondents say that they do not make purchases on this website, 39.7% declare that they make purchases more than once a month. The second place is taken by the Aliexpress store. More than every second respondent, as much as 55%, make purchases on this sales platform. Despite the fact that shipping usually comes from abroad, Polish consumers are eager to buy there. The other platforms aren't so popular anymore. 87.6% of respondents declare that they do not shop on Amazon at all. Even more - 94.1% on Ebay, and Wish is the worst in the ranking, where 95.3% have never bought anything.

Allright, the interface is certainly very important in terms of website quality. After all, he is responsible for interacting with the user. The respondents were asked how they rate the interface of each eshop on a five-point scale from 1 to 5. Additionally the survey results show that users rate its interface most often at 5 points on a five-point scale. Not a single person decided to give this website a grade of 1. This shows that this store has a very good visual reception. For the remaining platforms, the number 3 was assigned the most frequently. This means that the respondents rated the interface of each of these stores on average. Aliexpress ranks second in the ranking, with 47.1% of respondents assessing the interface at 4 or 5 points. Wish online store is the worst in the ranking. Only three people decided to assign a score of five points. As many as 24.6% of people participating in the study assessed the borderline layout of the website below the average, because they assigned a score of one out of five points to 6.3% of the respondents and as many as 18.3% of the score two out of five. This is the worst result of all five sales platforms.

When examining intuitiveness, the author meant the ease of navigating through individual websites. The programmers have the greatest impact on intuitiveness, choosing the functionalities of websites so that the consumer can make a purchase as quickly and easily as possible. In the next compilation of the study, a five-point scale was used, as was the case with the interface quality. Thanks to this solution, it is easy to find out how the respondents assess which e-store is intuitive for them. Allegro is presented first, because it congratulates once again when comparing online stores. As many as 65.7% of respondents assessed the intuitiveness of this website as maximally and only one person gave the lowest score. The Aliexpress online store seems to be in second place again, because exactly 51% of respondents gave this service a 4 or 5 rating in terms of intuitiveness. The other websites show the same trend. Both Amazon, Ebay and Wish got more than 50% of the votes in the

form of a three-point rating. This means that consumers rate the intuitiveness of these online stores on average.

Undoubtedly, one of the most important features of any online store is its security. When examining this factor, the author focused on the effectiveness of the delivery of the purchased product by a given e-shop to the consumer. The survey asked about an unfinished delivery, which resulted in the consumer not receiving the product. In particular, the interest in the results may be aroused by the Aliexpress website. 40.7% of the respondents replied that they had not received the purchased product at least once. As many as 14% of them claim that they have not received it more than 3 times. This result raises controversy whether the store is at all a safe place to shop online. All of them are more than ninety percent effective. Allegro achieved the lowest, 91.7% efficiency in delivery. The result may be due to the fact that consumers most often shop on this portal, therefore the statistical effectiveness is lower. However, despite the fact that Allegro is more popular than Aliexpress, Aliexpress is the least secure e-store for making purchases.

The vast majority, as much as 85.7% of Internet users, do their shopping on Allegro. The result of the study confirms other online sources, which also clearly show that this e-store is the most frequently chosen online store in Poland. Aliexpress is undoubtedly second. This portal is very popular because it offers very good prices. The other three websites are less popular. But it should be noted that still 2.4% of respondents choose them most often when shopping online.

6. Conclusions

A thorough analysis of selected online stores allowed us to define the strengths and weaknesses of individual websites, which are nowadays one of the most frequently chosen ways of making purchases by consumers. The surveyed group of respondents are people who mostly shop online. Allegro turns out to be the best of the five stores selected in the survey. The largest number of respondents assessed this website positively, both in terms of its interface and intuitiveness. Also, the vast majority feel safe when making purchases on this platform. The second most popular platform is, of course, the Aliexpress service. It encourages consumers with low prices, but this does not always translate into the final quality of purchases. Most of the respondents assessed its interface and intuitiveness on average. The website is definitely the worst in terms of transaction finalization, i.e. receiving the product. Many people declare that they have never received the purchased item despite placing an order. Certainly, the platform has potential, but it does not encourage customers there to continue shopping in the future. The other three platforms are not so popular with Polish consumers anymore. The respondents who use Amazon declare good security when making purchases there, but average intuitiveness. Ebay and Wish were also considered in this work. A small number of people shop there, despite being one of the largest sales platforms in the world. Users poorly declared their interface and average intuitiveness. It can be concluded that these parameters play a key role in the selection of online stores, as in the case of Allegro, Internet users assessed them at a fairly good level.

Consumers using online stores pay attention to many factors. They appreciate the legible graphic layout of the website and its simplicity. They pay attention to aspects such as safety and intuitiveness. Allegro can be considered a winner in comparison. This service is the most popular and encourages further purchases with transparency. However, it's good to be careful about the Aliexpress store. It is very likely that the goods ordered there may not be delivered.

References

[1] P. Weinlich, Website Quality and Shopping Behavior: Quantitative and Qualitative Evidence, Springer Nature, 2020.

[2] Website that researches online stores, https://www.similarweb.com/top-websites/category/e-commerce-and-shopping, [15.05.2021].

[3] Y. Gao, Web Systems Design and Online Consumer Behavior, Idea Group Publishing, 2004.

[4] D. J. Reibstein, What attracts customers to online stores, and what keeps them coming back?. J. of the Acad. Mark. Sci. 30, Journal of the Academy of Marketing Science, (2002) 465-473.

[5] C. Dobre, A. M. Milovan-Ciuta, Personality influences on online stores customers behavior, Ecoforum Journal, 4(1) (2015) 69.

[6] Sample selection calculator website, https://www.naukowiec.org/dobor.html, [11.05.2021].

[7] Website for creating online surveys, https://docs.google.com/forms, [17.05.2021].

[8] A social network where registered users can create networks and groups, https://www.facebook.com, [19.05.2021].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 247–253
Received: 27 June 2021
Accepted: 9 July 2021

Comparative analysis of Unity and Unreal Engine efficiency in creating virtual exhibitions of 3D scanned models

Analiza porównawcza wydajności silników Unity i Unreal Engine w aspekcie tworzenia wirtualnych pokazów modeli pochodzących ze skanowania 3D

Agata Malwina Ciekanowska*, Adam Krzysztof Kiszczak – Gliński*, Krzysztof Dziedzic

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

Abstract
The main purpose of this work was to compare two game engines (Unreal Engine and Unity) in creating virtual exhibitions. The article is a scientific description of a test of their efficiency. For the needs of the research two identical test applications built on the basis of the same assets were created. Those applications enabled researchers to examine and compare the efficiency of engines in question, as well as familiarizing themselves with the workflow on each platform. The comparative analysis of gathered data let more effective solution to emerge, which happens to be Unity engine.

Keywords: virtual museum; 3D models; game engines; comparative analysis

Streszczenie
Głównym celem tej pracy było porównanie dwóch silników gier (Unreal Engine oraz Unity) w tworzeniu wirtualnych pokazów. W artykule opisano doświadczenie badające ich wydajność. Na potrzeby eksperymentu przygotowano dwie bliźniacze aplikacje testowe, zbudowane na bazie tych samych assetów. Pozwoliły one na zbadanie i porównanie wydajności rozpatrywanych silników oraz zapoznanie się z tym jak wygląda praca na każdym z nich. Analiza porównawcza zebranych danych pozwoliła wyłonić wydajniejsze rozwiązanie, którym okazał się silnik Unity.

Słowa kluczowe: wirtualne muzeum; modele 3D; silniki gier; analiza porównawcza

*Corresponding author

Email addresses: **agata.ciekanowska@pollub.edu.pl** (A. M. Ciekanowska), **adam.kiszczak-glinski@pollub.edu.pl** (A. K. Kiszczak – Gliński)

1. Introduction

The number of internet connections across the globe is growing rapidly. Thanks to this process, developing virtual entertainment and education systems is easier and faster than ever. This trend also includes virtual museums, which are quickly gaining in popularity. Considering recent restrictions related to the pandemic, the value of virtual exhibitions is even greater lately.

In "The virtual museum", F. Antinucci attempted to create a definition of virtual museum [1]. According to the author, virtual exhibition is not a real museum, as watching 3D models cannot replace the impression of watching an exhibit in real life. The author underlines the importance of experiencing a real museum space. Such visit makes perceiving sizes, volumes and textures of artefacts easier. That said, he values virtual museums, for the flexibility to create exhibitions which are difficult or even impossible to create in real life. In the discussed work, virtual museums are considered by the author more as a supplement to the existing, real life exhibitions rather than a separate entity.

J. Derwisz in "Współczesne technologie multimedialne w wirtualnej rekonstrukcji oraz prezentacji historycznych obiektów architektonicznych" indicates issues with the reconstruction of objects that changed their form or appearance over the years [2]. Further, she points out, that such exhibitions often do not have a sufficient financial support. The author suggests 3D scanning and modelling of the artefacts as a viable solution for those complications.

Possible methods of scanning objects in 3D are presented in „An approach to computer-aided reconstruction of museum exhibits" by J. Kęsik, J. Montusiewicz and R. Kayumov [3]. One of the given techniques is to use a stationary device where an object can be placed and scanned. A disadvantage of such technique is the risk of damaging the exhibit during the transfer process. On the other hand, there is a safer method available – a handheld device. Using a mobile device minimizes the risk of a destruction of the artefact.

In the „Comparison of Unity and Unreal Engine" by A. Šmíd the author examines title game engines using a game created especially for this purpose [4]. The game was deployed in both of the engines, and later examined on a desktop computer, laptop, Android smartphone and Virtual Reality system. The author thinks that Unity was easier game engine to learn, the compilation time was shorter and the final size of the project was smaller. Alternatively, the Unreal Engine editor allowed creating scripts in visual Blueprint system, and had more advanced tools to create materials or generate terrain and vegetation. The final conclusion was that Unity is better solution when creating simpler projects on mobile platforms, and Unreal Engine is better suited for develop-

ment of more complex games on desktop computers or consoles.

A study of optimization methods in Unity game engine is included in the „Metody optymalizacji wydajności silnika Unity 3D w oparciu o grę z widokiem perspektywy trzeciej osoby" by K. Siarkowski, P. Sprawka and M. Plechawska-Wójcik [5]. For the research, the application with the third-person perspective was created. Parameters affecting performance such as occlusion culling, clip planes, batching and lighting were examined. Appropriate changes to the mentioned parameters gave positive results and positively affected the game performance.

E. Puławski and M. Tokarski within „Wykorzystanie postprocesingu i jego wpływu na wydajność renderowania w silniku Unreal Engine 4" article examined the Unreal Engine performance based on parameters such as lens flare, antialiasing, bloom or depth of field [6]. Researchers used an application created specifically for this purpose. The outcome showed that antialiasing is the most aggravating effect and bloom – the least.

2. Selected game engines overview

Game engines chosen as the research subjects in this work are Unity and Unreal Engine. Unity is an engine developed by Unity Technologies and Unreal Engine was created by Epic Games [7][8]. Both of them allow users to create 2D or 3D cross-platform applications such as games, visualisations, product configurators, interactive exhibitions or even virtual museums. According to the survey made in 2020 by Unity Technologies, Unity was the most popular software used to create games on mobile devices [9]. A diagram showing survey results can be seen on Figure 1.

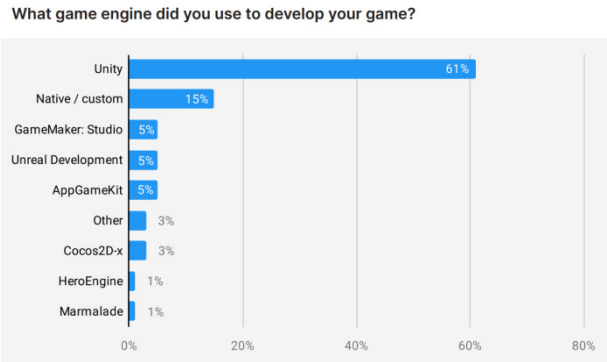


Figure 1: Game engines popularity in 2020 in creating mobile apps [9].

Based on the Figure 1, the Unity engine was selected to develop mobile apps 61% of time. Unreal Engine ranks as the second most popular solution available to general use, along with GameMaker: Studio and AppGameKit (each of them scored 5%).

Another research was made in 2019 and used a special script [10]. The script investigated games published on Steam platform for the usage of different game engines. Collected data is valuable but unfortunately it's not the most precise. To be involved in the research, the game must have had a Wikipedia page where an information about used game engine had to be filled. The results can be seen on Figure 2.

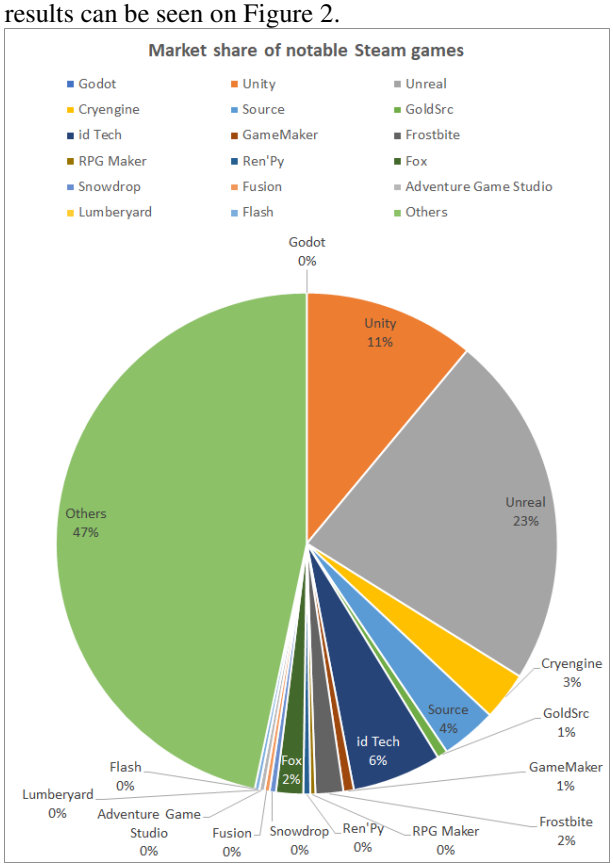


Figure 2: Market share of notable Steam games in 2019 [10].

According to this research (as can be seen on Figure 2), the most popular game engines were Unreal Engine which scored 23% and Unity with the score of 11%. The popularity aspect had a great importance in the choice of game engines to be used in our research.

It is worth adding that both of the examined engines at the moment of writing this paper had a policy which allowed free use for the non-commercial purposes. Moreover, Unity and Unreal have extensive documentation, which make application development easier and faster.

3. 3D scanned models

Artefacts used in this research originate from the Silk Road in Central Asia. The 3D models of these exhibits can be seen on Figure 3.

Figure 3: 3D models of exhibits from the Silk Road.

From the top left corner, clockwise – a lamp fragment, a camel-shaped jug, a beverage bottle and a perfume vessel.

The four scanned and textured artefacts presented on Figure 3 were used creating the virtual museum. Each model has a few variations with mesh simplified in a different degree. A 3D mesh is the structural build of a 3D model. 3D meshes use reference points in X, Y and Z axes to define shapes. Availability of multiple models in several quality levels allowed achieving a satisfying performance.

3.1. Obtaining the models

The models shown in the Figure 3 were created using Artec Space Spider handheld scanner, made by Artec 3D. The scanner is shown on the Figure 4 [11].

Figure 4: Artec Space Spider scanner [11].

The Space Spider has an ability to capture objects up to 2000 cm^3. Its maximum scanning capability is 1 million points per second with the 7.5 FPS frame rate. Moreover, it enables obtaining textures of scanned pieces with a 1.3 Mpx resolution [12].

Finished scans of the four exhibits were later processed using AutoCAD and Artec Studio software.

It is also possible to use stationary devices to scan exhibits but it poses a risk of damaging or even destroying object while moving it [13].

4. The virtual museum

The virtual museum was deployed in both Unity and Unreal Engine. Efforts were made to make the two scenes as similar to each other as possible, e.g. using the same textures, materials and 3D model of a museum or placing artefacts in an identical arrangement, etc. Alike settings for the museum were applied as well.

Figure 5: Virtual museum implemented in the Unity game engine.

Figure 6: Virtual museum implemented in the Unreal game engine.

The model of the museum area including lamps, sconces, pedestals and a front door was created using Blender software which is free to use [14]. A virtual visitor can move around in the created space from the first-person perspective. The user also can interact with each exhibit by pressing ‘E’ key on a keyboard whilst standing nearby the chosen artefact. An information about this function is displayed when appropriate for convenience.

Figure 7: Exhibit view made in the Unity game engine, allowing interaction with the model.

Figure 8: Exhibit view made in the Unreal game engine, allowing interaction with the model.

The purpose of the view presented in Figures 7 and 8, is that the person visiting the museum can rotate the chosen object and read information about it. An informative note about the object is presented at the bottom of the screen. This gives the user a chance to take a better look at every exhibit and to discover its origins.

Scripts responsible for logic of the projects were created in C# and the Blueprint system for Unity game engine and Unreal Engine respectively.

The Level Of Detail (LOD) technique was used for optimization of performance on available hardware. The LOD is a tool allowing for improvement of the performance of a scene by loading a lower quality meshes in situations where user wouldn't notice the lack of details in models. It is based on the distance between the object and a camera (Unity) or percentage of screen surface occupied by model (Unreal) [15][16]. Considering very high complexity of models in the best quality (from 508028 up to 3199707 of mesh triangles), adding the LOD was necessary to ensure smooth and correct performance of the application. The highest quality model of an artefact is only displayed when the user is standing as close as possible to the object or enables the view presented in Figures 7 and 8.

5. The experiment

The experiment was conducted on two PCs and two laptops – each of them with different hardware specifications, listed in the Table 1.

Table 1: Specifications of computers used in experiment

Computer	Processor (CPU)	Graphic card (GPU)	RAM
PC 1	AMD Ryzen 7 3700x, 3.60GHz	AMD Radeon RX580	32 GB
PC 2	Intel i5-4460, 3.20GHz	AMD Radeon R9 270x	8 GB
Laptop 1	Intel i5-8250U, 1.60GHz	NVIDIA GeForce 940MX	8 GB
Laptop 2	Intel i5-7200U, 2.50GHz	NVIDIA GeForce 940MX	8 GB

In order to carry out the comparative analysis and to verify the efficiency of both game engines, six main criteria for examination were established:

- CPU load (measured as percentage),
- GPU load (measured as percentage),
- RAM load (measured as MiB),
- FPS value - Frames Per Second,
- subjective authors opinion on working with examined game engines and intuitive of their user interfaces (measured on 0 to 5 scale),
- final project size (measured as MB).

The experiment duration was 1 minute and it was repeated 10 times on every computer for each game engine. Performance data was gathered every second. To make tests comparable, all runs involved a similar course and the same order of exhibit examination. An individual artefact was observed in a view presented in Figures 7 and 8 for about 15 seconds, and after that, a virtual visitor proceeded to another exhibit.

Data describing CPU and RAM usage was collected using Performance Monitor built in the Windows 10 operating system. It allows saving hardware usage information limited to only one process, which was a crucial functionality for the experiment. Open Hardware Monitor software was used to gather information about GPU load and FPS values were captured with the help of Fraps programme. What is important, FPS value was limited to 60 in both instances. The reason behind that is this FPS value is considered optimal. Images projected in this frame rate are smooth, and GPU load is often reduced if the graphic card is powerful enough.

5.1. Results

Data shown on every diagram presented in this subsection is an average based on averages coming from data collected in each experiment series and is used for verification of the performance of both examined engines.

Figure 9 presents a diagram containing average processor usage data.

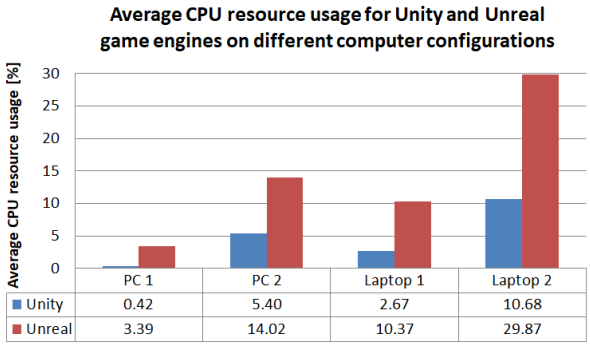


Figure 9: Average CPU usage.

The best performance for both Unity and Unreal engines was carried out by Laptop 1 in the laptops group and PC 1 in desktop computers group. An interesting outcome from this graph is the fact that PC 2 configuration has shown a worst performance than Laptop 1, despite having slightly better hardware specification. After delving into more specific information about each of the examined processors, possible explanation of this phenomenon was discovered. It is possible, that the number of threads in each unit was a decisive factor. CPU inside the PC 1 has 8 cores and 16 threads and the PC 2 has only 4 cores and 4 threads available. The processor in Laptop 1 has 4 cores alike PC 2, but 8 threads. Laptop 2 has only 2 cores and 4 threads. Quantity of cores can make a difference too, which is clear, when comparing PC 2 and Laptop 2.

Figure 10 represents collected information about average graphic card usage.

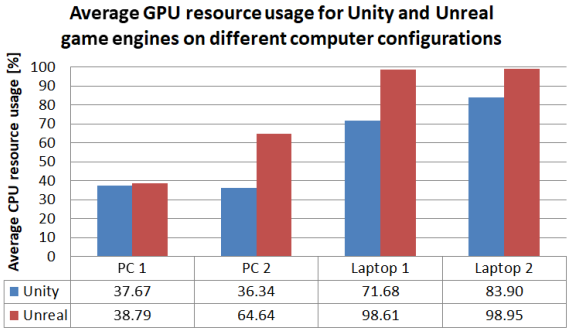


Figure 10: Average GPU usage

In most of the cases the Unity game engine achieved a better result compared to its competitor. The most significant difference between the two can be observed on the PC 2 – Unreal is nearly 2 times less efficient. For both of the laptops, the Unreal engine achieved similar result which can be rounded to approx. 99% of GPU usage. Despite the fact that both laptops have the same graphic card, the Unity engine had slightly better performance in this category on Laptop 1.

Figure 11 presents an average RAM consumption for each of the examined game engines.

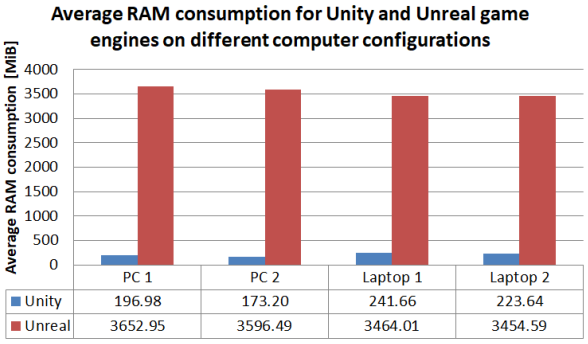


Figure 11: Average RAM consumption.

The difference between the two subjects of study is significant. Whilst testing Unity, RAM usage value has never exceeded 250MiB. For the Unreal game engine, the value fluctuated around 3500MiB.

Diagram of an average frames per second value is shown on the Figure 12. On the PC 1 both of the examined engines achieved nearly identical score which is very high. Most of the time, the scene on PC 1 was running at approx. 60FPS and that means it was stable and smooth in both examined cases. PC 2 achieved very similar score when it comes to the Unity engine. When running Unreal Engine, PC 2 performed slightly worse, but still excellent. In regard to laptops, Unity was capable of generating more FPS than Unreal by about 10 on both Laptop 1 and Laptop 2. The worst score for Unreal game engine was registered on Laptop 2 and it was 40 frames per second. It is still satisfactory result, but in this case, the quality was about ⅓ worse compared to PC 1 and PC 2.

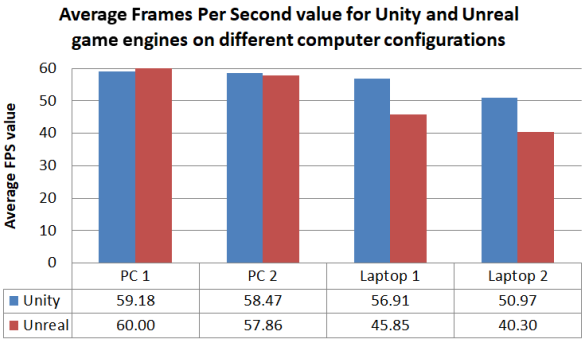


Figure 12: Average FPS value.

It is worth to compare average FPS diagram with average GPU usage graph. Unreal game engine in laptops group used about 99% of graphic card resources. That means generating more FPS than what can be seen on Figure 12 is not possible without further optimization. Alternatively, Unity did not use up all of the GPU computing power. That means improving achieved FPS values is still possible.

Figure 13 shows sizes of the final built projects. The size of the project made in Unity is about 50% smaller compared to the Unreal Engine.

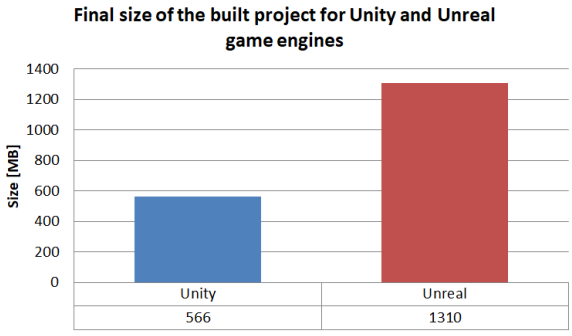


Figure 13: Final size of the built project.

The intuitiveness of engines User Interface and opinion on working with examined game engines are subjects to subjective evaluation. That is why there's no diagram with comparison results included in this category. In the opinion of the authors, both interfaces are similar. They allow the user to personalize the editor by changing positions of the individual windows in the workspace. One of the Unreal Engine biggest advantages is the possibility of programming scripts in C++ as well as in the Blueprint. The Blueprint is a visual programming system which allows users with no experience in programming as well as professionals to write full-fledged scripts. It is worth adding that Unity allows writing code in C# which is easier to learn than C++ but the user has more capabilities available using the Epic Games product. From the hardware perspective, the Unreal editor is more resource-demanding compared to Unity. Despite the fact that both of the examined engines are free to use up to a certain revenue from sales, the Unreal Engine is targeted more towards bigger game developing studios, whilst Unity engine is addressed to independent developers.

In order to compare values measured in different units (e.g. RAM and FPS), a grade is assigned to the selected criterion. The grade depends on the average score obtained during the experiment. All configurations were taken into consideration. Ratings for the ranges of results are listed in the Table 2.

Table 2: Rates for the ranges of results

Rating	CPU [%]	GPU [%]	RAM [MiB]	FPS	Size [MB]
5	0-5	0-30	-	60-55.01	-
4	5.01-10	30.01-40	-	55-50.01	-
3	10.01-15	40.01-50	-	50-45.01	-
2	15.01-20	50.01-70	500-	45-40.01	600-
1	20.01-25	70.01-90	500+	40-35.01	600+
0	25.01+	90.01+	-	35-	-

An average score for Unity engine on all of the computer specifications in the aspect of CPU usage can be rounded to 4.79% and for Unreal Engine – 14.41%. That means according to the Table 2 Unity's rating is 5 and Unreal's rating is 3.

In the GPU usage category an average result for Unity after rounding was 60.23, resulting in a grade 2. After calculations, Unreal result was 81.63, so it was rated 1.

The RAM consumption evaluation slightly varies-from previous criteria. Considering the differences in amount of RAM available in PC 1, and the fact that there is a big difference between the two compared engines in this case, there are only two grades available. The threshold was established based on the corresponding diagram. For Unity the grade is 2, and Unreal received 1.

An average score for FPS values for both of the game engines was also calculated. In this category Unity achieved an average of 56.38 FPS which equals rating 5. Unreal Engine accomplished a similar average value of FPS, which was 51, resulting in grade 4.

Due to the fact that interface intuitiveness and the simplicity of use of both of the examined engines is a subjective value, there are no ranges for the rating in this category. Based on the described advantages and disadvantages in the opinion of authors, both engines score 5 in the 0 to 5 scale.

In the final category, project size, there are only 2 grades available. As the size of project made in Unity was below 600MB, it receives rating of 2 and given the fact that the exhibition size made in Unreal Engine was above 600MB, its rating is 1.

Achieved results and given grades are presented in Table 3.

Table 3: Results and ratings given to the examined game engines

Criterion	Unity		Unreal	
	Score	Rating	Score	Rating
CPU load	4.79%	5	14.41%	3
GPU load	60.23%	2	81.63%	1
RAM load	<500 MiB	2	>500 MiB	1
FPS value	56.38	5	51	4
Overall experience	-	5	-	5
Final project size	<600 MB	2	>600 MB	1

Based on all of the information contained in this subsection, the resulting Table allowing comparative analysis was created. Listed criteria are the same as the ones mentioned in chapter 5. Weights of all of the six criteria add up to the number 1 and – more importantly – the higher the weight, the criterion had more importance for the whole project and the final rating. Unity and Unreal columns are meant for ratings of these game engines, bearing in mind the specific categories. The evaluation for a specific category and engine is derived from multiplying weight and rating from Table 3.

Table 4: Resulting table

No.	Criterion	Weight	Unity	Unreal
1	CPU load	0.1	0.5	0.3
2	GPU load	0.3	0.6	0.3
3	RAM load	0.2	0.4	0.2
4	FPS value	0.25	1.25	1
5	Overall experience	0.1	0.5	0.5
6	Final project size	0.05	0.1	0.05
	Totals:	1	3.35	2.35

As is seen in the Table 4, Unity engine has better final score (3.35) than Unreal Engine (2.35) meaning that Unity is more efficient tool for creating virtual exhibitions of 3D scanned models.

6. Conclusions

The experiment created for this work purposes has shown that the virtual museum can be made in Unreal and Unity game engines. However, achieved effects differ slightly. While both editors are similar in use from the creator's point of view and the visual side of created exhibitions is similar too, the final products aren't identical in terms of performance. In the most cases Unity has advantage over the rival engine. The biggest difference between the two, to the disadvantage of the Unreal Engine can be seen in the RAM usage comparison diagram and final size of the built project. For Unreal engine it was 3542 MiB and 1310 MB accordingly and for Unity – 209 MiB and 566 MB. Unity also wins in the CPU resource consumption – it used half as much resources as the competitor (appropriately 4.8% and 14.4%). Results in the GPU resource consumption are in favour of the Unity engine as well, which resource usage oscillated around 57%, being 18 percentage points lower than second game engine under consideration. When it comes to the amount of generated FPS, similar results can be seen on desktop computers – both Unity and Unreal accomplished an average of 59 frames per second, but on both of the laptops Unreal Engine performs worse - its mean was 43 FPS which was a score worse than Unity by 10 frames.

Final grades for Unity and Unreal Engine were 3.35 and 2.35 accordingly. These ratings beg the question - why is Unreal Engine still so popular? From the subjective point of view the created scene was looking better in Unreal with less work put into visual aspect of it. Blueprint system is definitely an advantage of this engine too. The engine also provides more advanced tools, resulting in greater popularity among professional users and game developing studios.

Considering high RAM usage compared to Unity, it is worth further examination. One of the possibilities is the potential difference between applications created using Blueprints and C++ and their RAM consumption.

References

[1] F. Antinucci, The virtual museum, Archeologia e Calcolatori, supplemento, 1 (2007) 79-86, http://www.archcalc.cnr.it/indice/Suppl_1/6_Antinucci.pdf [02.07.2021].

[2] J. Derwisz, Współczesne technologie multimedialne w wirtualnej rekonstrukcji oraz prezentacji historycznych obiektów architektonicznych, Wiadomości Konserwatorskie, 34 (2013) 82-91, http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-909a5d65-29b1-46fd-bfa5-3c76fb7efac1/c/Derwisz.pdf [02.07.2021].

[3] J. Kęsik, J. Montusiewicz, R. Kayumov, An approach to computer-aided reconstruction of museum exhibits, Advances in Science and Technology. Research Journal, 11 (2017) 87-94, http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-73ba2929-b9ff-4741-978b-bf780441dd3f/c/kesik.pdf [02.07.2021].

[4] A. Šmíd, Comparison of Unity and Unreal Engine, Bachelor Thesis, Czech Technical University, Prague, 2017, https://dcgi.fel.cvut.cz/theses/2017/smidanto [02.07.2021].

[5] K. Siarkowski, P. Sprawka, M. Plechawska-Wójcik, Metody optymalizacji wydajności silnika Unity 3D w oparciu o grę z widokiem perspektywy trzeciej osoby, Journal of Computer Sciences Institute, 3 (2017) 46-53, https://doi.org/10.35784/jcsi.592 [02.07.2021].

[6] E. Puławski, M. Tokarski, Wykorzystanie postprocesingu i jego wpływu na wydajność renderowania w silniku Unreal Engine 4, Journal of Computer Sciences Institute, 10 (2019) 54-61, https://doi.org/10.35784/jcsi.206 [02.07.2021].

[7] J. Haas, A History of the Unity Game Engine, An Interactive Qualifying Project, Worcester Polytechnic Institute, Worcester, 2014, https://digital.wpi.edu/pdfviewer/2f75r821k [02.07.2021].

[8] A. Szewczyk, Oczekiwania fanów elektronicznej rozrywki wobec grafiki w grach komputerowych, Studia Informatica Pomerania, 40 (2016) 71-85, http://wneiz.pl/nauka_wneiz/studia_inf/40-2016/si-40-71.pdf [02.07.2021].

[9] 2021 Gaming Report - Unity insights from 2020 and predicted trends for 2021, https://images.response.unity3d.com/Web/Unity/%7B4645ad28-63a3-4348-bc5b-dc09f2811419%7D_Unity_2021-Gaming-Report.pdf [02.07.2021].

[10] Research of the market share of game engines on Steam, from over 60,000 Steam games, https://www.reddit.com/r/gamedev/comments/8s20qp/i_researched_the_market_share_of_game_engines_on/ [02.07.2021].

[11] Artec Space Spider, https://skanery3d.eu/skanery-3d/artec-spider/ [02.07.2021].

[12] Artec 3D Technical specifications, https://www.artec3d.com/portable-3d-scanners/artec-spider-v2#specifications [02.07.2021].

[13] M. J. Wachowiak, B. V. Karas, 3D Scanning and Replication for Museum and Cultural Heritage Applications, Journal of the American Institute for Conservation, 48 (2019) 54-61, https://www.si.edu/content/MCIImagingStudio/papers/scanning_paper.pdf [02.07.2021].

[14] A. Salwierz, T. Szymczyk, Metody wytwarzania realistycznych pomieszczeń – skanowanie 3D oraz modelowanie 3D, Journal of Computer Sciences Institute, 14 (2020) 101-108, https://doi.org/10.35784/jcsi.1584 [02.07.2021].

[15] Unity - Level of Detail (LOD) for meshes, https://docs.unity3d.com/Manual/LevelOfDetail.html [02.07.2021].

[16] Unreal Engine - Creating and Using LODs, https://docs.unrealengine.com/4.26/en-US/WorkingWithContent/Types/StaticMeshes/HowTo/LODs/ [02.07.2021].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 20 (2021) 254–258
Received: 14 July 2021
Accepted: 19 August 2021

IoT system for remote monitoring of the mangrove forests of Sundarbans

System IoT do zdalnego monitorowania lasów namorzynowych Sundarbans

Asif Rahman Rumee*

Department of Computer Science and Engineering, Jashore University of Science and Technology, Jashore-7408, Bangladesh

Abstract

In-situ monitoring of mangrove forests is expensive, cumbersome, time consuming and error-prone, hence remote approaches are being used widely nowadays. Remote sensing using satellites, UAVs and other devices is incapable of collecting many important types of data required for processing, therefore a prototype of an IoT device is designed and built for monitoring environmental parameters of the largest mangrove forests in the world, the Sundarbans in Bangladesh. The prototype is tested for a few hours in a simulated environment where the readings are updated every 2 seconds and alert notifications are received if an emergency event occurs. The simulation results prove the effectiveness of the proposed device and the feasibility of it for low cost remote monitoring of the mangrove forests.

Keywords: remote monitoring; IoT; mangrove forests; Sundarbans

*Corresponding author

Email address: arrumee@gmail.com (A. R. Rumee)

1. Introduction

Generally remote sensing using satellites, aeroplanes and unmanned aerial vehicles etc. [1-2] are being used nowadays to capture imagery and related data for mapping and monitoring mangrove forests in different parts of the world. Before the appearance of the remote sensing in-situ monitoring [3] approach was used involving a huge amount of human resources to collect the required data from widespread adverse areas. On-site monitoring process was tedious and monotonous due to the repetitive nature of the tasks belonged to it. It was error-prone as well due to human intervention. This approach also used to take a lot of time to perform the tasks and it was a very high cost process. On the contrary the remote sensing approaches used currently are easy to conduct, more accurate and less expensive compared to in-situ approaches [4]. But this approach has some limitations to such as unable to collect all required data [5]. Therefore, people are looking for some other techniques to overcome these issues. With the advancement of embedded systems, sensors, wireless networks IoT systems have become a very common mechanism for remote monitoring and tracking of many objects and parameters in agriculture, industries, healthcare, transportation, supply-chain management sectors and so on. Some IoT devices can measure the concentration of different gases such as CO_2, CO, NO_2 present in the air as well as temperature, humidity and send it to the distant servers [6]. Sound level and noise in smart cities can be measured using some autonomous devices as well [7]. Besides air and sound quality almost all kinds of environmental variables are being measured and analysed using IoT based systems. Corn, rice, vegetables, fruits are cultivated with the help of smart IoT devices. These devices monitor the water level, soil moisture, soil nutrients etc. and then according provide required amount of water, fertilizer and necessary things [8-9].

The Sundarbans are the largest mangrove forests in the world, located at the southern region of Bangladesh on shore of Bay of Bengal and on the estuary of Ganga, Brahmutra, Meghna along with the west Bengal of India. The total area of it is about 10,000 square kilometres where 6,000 square kilometres belong to Bangladesh. It was declared one of the World Heritage Sites by UNESCO in 1997. It alone contributes 40% to the total forest land of Bangladesh. It plays a vital role for maintaining ecological and environmental balance in Bangladesh. Therefore, often she is called the lunge of our country. She protects us from natural disasters like cyclones, storms, floods etc. many times in every year. It is important for our country not only for environmental aspects but also for economic aspects. The ecosystem of the Sundarbans is degrading day by day which have become a serious threat to Bangladesh as well as the whole world. The average yearly revenue earned from provisioning services of the forest e.g. timber, fuel wood, fishes, thatching materials, honey and waxes, was 744,000 US dollars. The revenue from cultural services i.e. tourism was on an average 42,000 US dollars per year during the fiscal years 2001-2002 to 2009-2010 [10]. The economic contributions of the forest to low-income, middle-income and high-income population living adjacent to it were 74%, 48% and 74% of per capita annual income respectively [11]. Actually it is not possible to express the importance of the Sundarbans in terms of both ecological and economic perspectives by words. It has blessed the people in many ways but in response the people are causing harms to

her gradually. The water in the rivers of the forest is being contaminated by oil spillage by ships, pesticides used in near fields and farms. Industrial factories near the forest such coal power plants are continuously polluting the air quality by emitting toxic gases e.g. SO_2, NO_2 etc. Therefore, an effective monitoring system is needed to correctly measure the pollution of the environment of the forest in order to save it from degradation.

In this paper a prototype of an IoT device for low cost remote monitoring of the mangrove forests of Sundarbans is studied. The different functionalities of the device such as air, water, soil and sound quality measurement are checked as well. This research focuses on measuring the efficiency of the prototype as well as the feasibility of it for remote monitoring of the mangrove forests in a lower cost.

2. Materials and Methodology

2.1. Materials

A networking and IoT simulation tool named cisco packet tracer [12] is used to build the prototype as well as to make the test environment. From many different kinds of sensors available in the tool 8 types of sensors are used in building the prototype. One microcontroller board is used with a wireless module attached to it. In the remote station one single board computer is used as a server (see Figure 1). A list of these components is described below briefly.

- 1 MCU-PT microcontroller. Equivalent real device available in the market is Arduino Uno which costs 16 US dollars [13] and Wireless module SIM808 which costs 18 US dollars [14].
- 1 Humidity Sensor that provides analog output in range 0 to 255, representing 0 to 100% humidity in the air whose equivalent real device is DHT11 sensor.
- 1 Smoke or Gas Sensor which gives analog output in range 0 to 1023, representing 0 to 100% smoke percentage in the air that is similar to real device MQ-2 Gas Smoke sensor.
- 1 Water Sensor which gives analog output from 0 to 1023 representing 0 to 20 cm.
- 1 Sound Sensor which gives analog output as sound level in decibels.
- 1 Temperature Sensor which gives analog output in range 0 to 255, representing -100 C to 100 C temperature ranges that is equivalent to real sensor DS18B20.
- 1 Water Detector that detects presence water seems to be a rainfall detector.
- 1 Wind or Vibration Sensor that detects wind in the environment.
- 1 Fire or Flame Sensor that detects IR in the range of fire. The total cost of these 8 sensors is about 36 US dollars [15].
- 1 SBC-PT single computer board. Equivalent real device available in the market is Raspberry-pi which costs 67 US dollars.

The total cost of the proposed prototype is only 70 US dollars which is very cheap comparative to any UAVs or satellites. This cost is excluding of the cost of the Raspberry-pi as it is not part of the prototype.

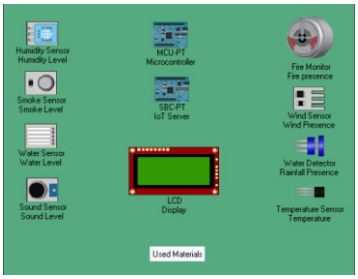


Figure 1: Materials used to build prototype and remote server.

2.2. Methodology

The overall system architecture as well as the workflow is depicted in Figure 2. The sensors grasp data from environment and the microcontroller collects the data from sensor and sends to the remote station. The single board computer i.e. the server in the remote station receives the data sent by the prototype. The mobile operator passes data from the prototype to the internet.

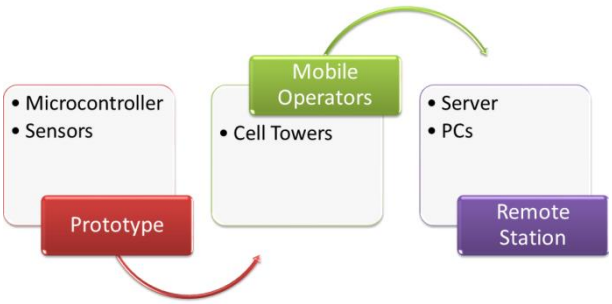


Figure 2: Overall system architecture.

The microcontroller of the prototype collects the data from the sensors connected to its ports and sends the data over a cellular network by using the wireless module PT-IOT-NM-3G/G attached to it (see Figure 3).

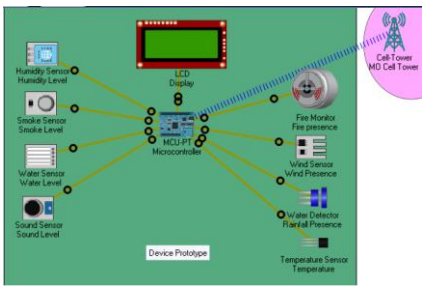


Figure 3: Prototype – first part of system architecture.

Wired broadband network infrastructure is not available in the Sundarbans area. One of the cellular phone operators named Teletalk provides network coverage in the forest area. Therefore, the cellular network is used to act

as the physical media for transmitting data from the prototype to the remote server (see Figure 4).

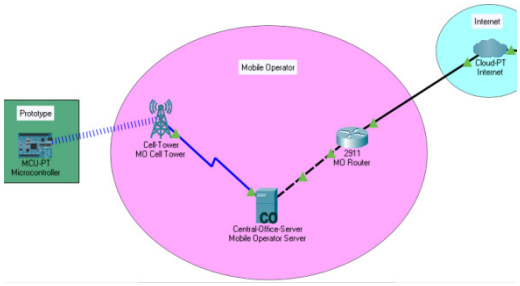


Figure 4: Mobile operator – middle part of system architecture.

A single board computer SBC-PT is used as the remote server which receives data from the distant forest, store and process the data as user requirements (see Figure 5).

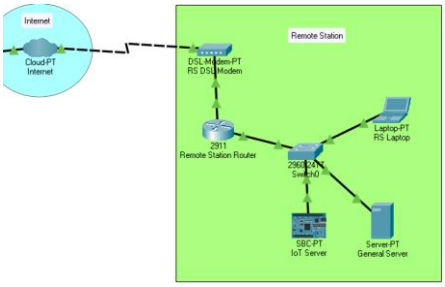


Figure 5: Remote station – last part of system architecture.

The python code that collects and sends data from the prototype is shown in Figure 6 and Figure 7. And the code which is running on the remote server SBC-PT is shown in Figure 8. The cisco packet tracer has the option to write the code in javascript as well.

```
from gpio import *
from tcp import *
from email import *
from time import *

serverIP = "100.168.2.20"
serverPort =8088
client = TCPClient()

def onTCPConnectionChange(type):
    print("connection to " + client.remoteIP() + " changed to state " + str(type))

def onTCPReceive(data):
    print(client.remoteIP() + data)

def getAnalogSensorData(analogPin):
    return float(analogRead(analogPin)*100/1023)

def getDigitalSensorData(digitalPin):
    return "NO" if digitalRead(digitalPin) == 0 else "YES"

def getWaterInCM():
    return float(analogRead(A2)*20/1023)

def getSoundInDB():
    return float(analogRead(A3))

def getTemperatureInC():
    return float((digitalRead(0)-512)*100/512)
```

Figure 6: Source code running in the microcontroller – 1.

```
def onEmailSend(status):
    print("send status: " + str(status))

def main():
    client.onConnectionChange(onTCPConnectionChange)
    client.onReceive(onTCPReceive)
    EmailClient.setup(
        "device@iotserver.com",
        "iotserver.com",
        "device",
        "device"
    )
    EmailClient.onSend(onEmailSend)
    print(client.connect(serverIP, serverPort))

    while True:
        data = "| {}    | {}     | {}     | {}     | {}     |".format(str(getAnalogSensorData(A0)), \
        str(getAnalogSensorData(A1)), str(getWaterInCM()), str(getSoundInDB()), str(getTemperatureInC()))
        for i in range(3):
            data = data + " {}  |".format(str(getDigitalSensorData(i+1)))
        data = data + ctime()
        customWrite(5, data)
        client.send(data)
        if getDigitalSensorData(3) == "YES":
            EmailClient.send("admin@iotserver.com", "Notification for fire", "A fire has been detected around me")
        sleep(2)

if __name__ == "__main__":
    main()
```

Figure 7: Source code running in microcontroller – 2.

```
from tcp import *
from time import *

port = 8088
server = TCPServer()

def onTCPNewClient(client):
    def onTCPConnectionChange(type):
        print("connection to " + client.remoteIP() + " changed to state " + str(type))

    def onTCPReceive(data):
        print(client.remoteIP() + data)
        # send back same data
        client.send(data)

    client.onConnectionChange(onTCPConnectionChange)
    client.onReceive(onTCPReceive)

def main():
    server.onNewClient(onTCPNewClient)
    print(server.listen(port))
    print("Source       |Humidity (%)|Smoke (%)|Water (cm)|Sound (db)|Temperature (c)| \
    Rainfall (y/n)|Wind (y/n)|Fire (y/n)| Time ")

    while True:
        sleep(36000)

if __name__ == "__main__":
    main()
```

Figure 8: Source code of the remote server.

3. Results

The data of eight environment parameters namely humidity, smoke, water, sound, temperature, rainfall, wind and fire are shown from second column to ninth column orderly in Figure 9. The first and last columns represent source IP address and received time respectively.

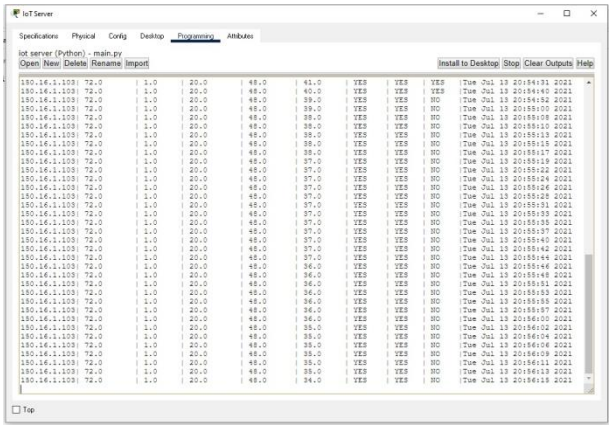


Figure 9: Environment data received in remote server.

When the prototype detects any fire, it immediately sends a notification to the administrator. A notification email sent to the remote administrator is shown in Figure 10.

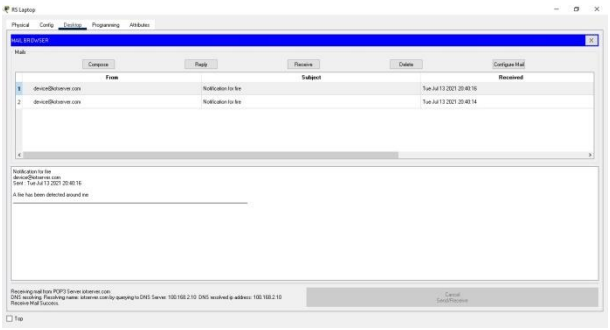


Figure 10: Email notification sent by the prototype.

4. Discussion

The proposed device is tested for 5 hours in the cisco packet tracer simulation tool where a test environment like mangrove forests is created using several actuators. It grasps data from the test environment and transmits to the remote server using TCP/IP continuously in 2 seconds interval. For the sake of simplicity custom socket is used instead of MQTT for transmitting data. The real implementation should use MQTT protocol as it is more lightweight, efficient, reliable and secure for IoT than TCP/IP. The readings captured in the remote server are updated in every 2 seconds. However, in the case of real implementation the prototype will be used to send data on a hourly or even daily basis depending on the requirements. The data received in the server proves the effectiveness of the prototype. Instead of this private server any cloud server can be used to receieve, store and process these data. The prototype also sends an alert message to the remote administrator using SMTP in any emergency situations such as fire detection. The proper functionality of the prototype proves the feasibility of it for low cost (see Table 1) remote monitoring of the mangrove forests of Sundarbans.

Table 1: Cost of the prototype

Item	Cost
Arduino Uno	$16
Wireless module	$18
Sensors	$36
Total	$70

5. Conclusions

A prototype of an IoT device for monitoring the world's largest mangrove forests of Sundarbans is designed, constructed and examined in a simulated environment in this research. The experiment shows that the proposed device is feasible to remotely monitor the forest properly. This study also concludes that the prototype is very cheap as well because it costs only 70 US dollars (see Table 1).

The functionality of the device can be extended by incorporating more advanced sensors to collect data such as gas concentration in the air, soil nutrients and so on. Solar panels can be added to the device for providing powers.

References

[1] C. Giri, Recent Advancement in Mangrove Forests Mapping and Monitoring of the World Using Earth Observation Satellite Data, Remote Sensing 13(4). (2021) 563, https://doi.org/10.3390/rs13040563.

[2] Y. Jiang, L. Zhang, M. Yan, J. Qi, T. Fu, S. Fan, B. Chen, High-Resolution Mangrove Forests Classification with Machine Learning Using Worldview and UAV Hyperspectral Data, Remote Sensing 13(8) (2021) 1529.

[3] K. Mirakhorlou, S. Teimouri, M. Abadeh, Mapping potential of mangrove forests based on site demands (Geomorphological factors and physico-chemical characteristics of soil and water), Environ. Conserv 23 (2017) 90-97.

[4] M. Ruwaimana, B. Satyanarayana, V. Otero, A. M. Muslim, M. A. Syafiq, S. Ibrahim, D. Raymaekers, N. Koedam, F. Dahdouh-Guebas, The advantages of using drones over space-borne imagery in the mapping of mangrove forests, PloS one 13(7) (2018) e0200288.

[5] T. D. Pham, N. Yokoya, D. T. Bui, K. Yoshino, D. A. Friess, Remote sensing approaches for monitoring mangrove species, structure, and biomass: Opportunities and challenges, Remote Sensing 11(3) (2019) 230.

[6] K. D. Purkayastha, R. K. Mishra, A. Shil, and S. N. Pradhan, IoT Based Design of Air Quality Monitoring System Web Server for Android Platform, Wireless Personal Communications 118(4) (2021) 2921-2940.

[7] M. Anachkova, S. Domazetovska, Z. Petreski, V. Gavriloski, Design of low-cost wireless noise monitoring sensor unit based on IoT concept, Journal of Vibroengineering 23(4) (2021).

[8] F. Akhter, H. R. Siddiquei, M. E. E. Alahi, K. Jayasundera, S. C. Mukhopadhyay, An IoT-enabled Portable Water Quality Monitoring System with MWCNT/PDMS Multifunctional Sensor for Agricultural Applications, IEEE Internet of Things Journal (2021).

[9] P. Sumathi, R. Subramanian, V. V. Karthikeyan, S. Karthik, Soil monitoring and evaluation system using EDL-ASQE: Enhanced deep learning model for IoI smart agriculture network, International Journal of Communication Systems (2021) e4859.

[10] M. S. Uddin, E. R. V. Steveninck, M. Stuip, M. A. R. Shah, Economic valuation of provisioning and cultural services of a protected mangrove ecosystem: A case study on Sundarbans Reserve Forest, Bangladesh, Ecosystem Services 5 (2013) 88-93.

[11] A. N. M. Abdullah, N. Stacey, S. T. Garnett, B. Myers, Economic dependence on mangrove forest resources for livelihoods in the Sundarbans, Bangladesh, Forest Policy and Economics 64 (2016) 15-24.

[12] A. Jesin, Packet Tracer Network Simulator, Packt Publishing Ltd, 2014.

[13] Arduino UNO R3 board with DIP ATmega328P, https://www.walmart.com/ip/Arduino-UNO-R3-board-with-DIP-ATmega328P/133534784, [09.08.2021].

[14] SIM808 Module GSM GPRS GPS Development Board IPX SMA with GPS Antenna for Arduino Raspberry Pi Support 2G 3G 4G SIM Card, https://www.aliexpress.com/item/1005001967026161.ht

ml?spm=a2g0o.productlist.0.0.5173b0dclMbfH9&algo_pvid=807751fc-8db4-4387-af70-84777f58bd1c&algo_exp_id=807751fc-8db4-4387-af70-84777f58bd1c-2, [09.08.2021].

[15] KOOKYE 16 in 1 Smart Home Sensor Modules Kit for Arduino Raspberry Pi DIY Professional, https://www.walmart.com/ip/KOOKYE-16-in-1-Smart-Home-Sensor-Modules-Kit-for-Arduino-Raspberry-Pi-DIY-Professional/392581400, [09.08.2021].