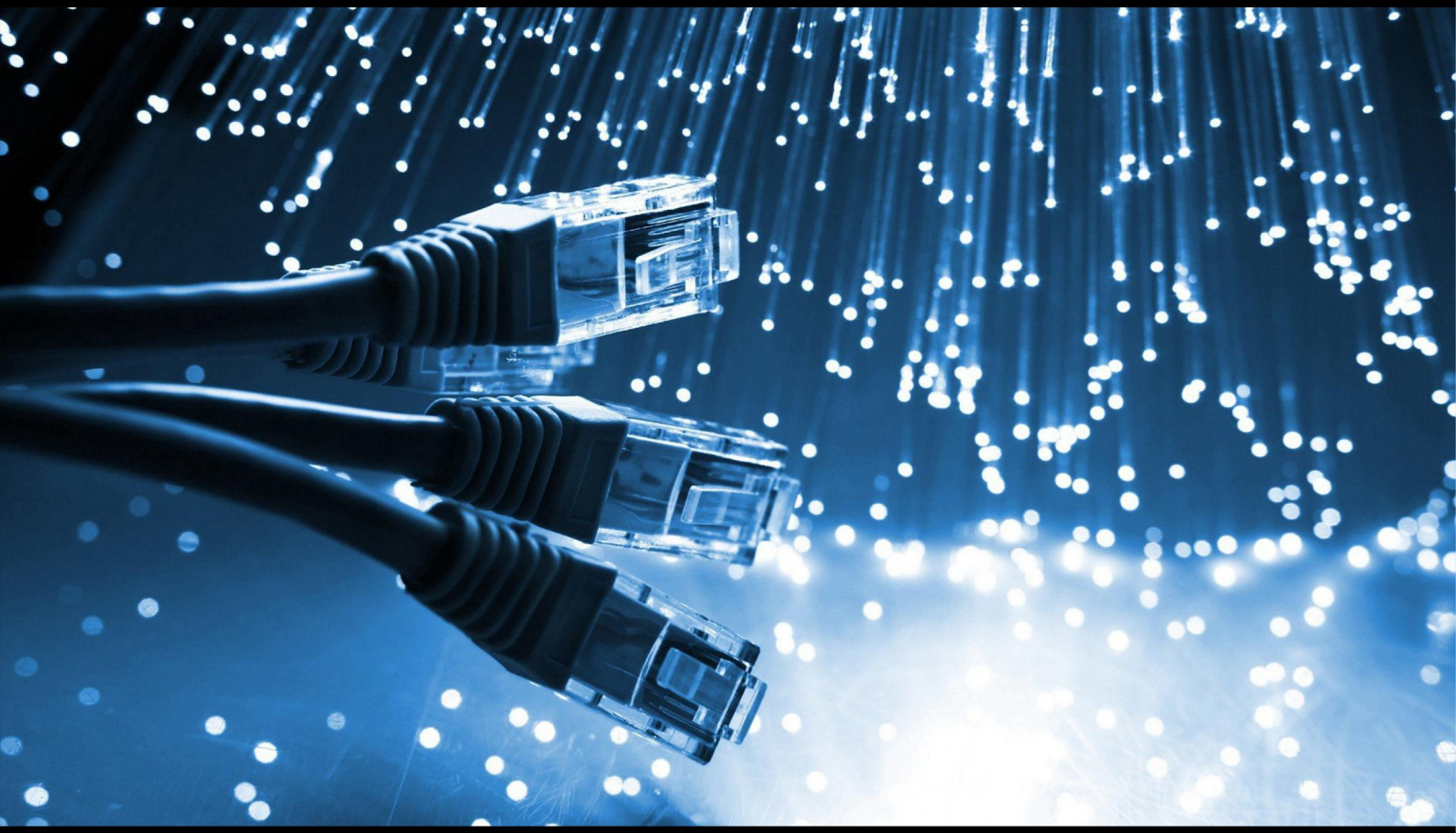
JCSI

Journal of Computer Sciences Institute

Volume 10/2019



Institute of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

Redakcja JCSI
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Instytut 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 Mariusz Dzieńkowski
dr inż. Maria Skublewska-Paszkowska
dr Edyta Łukasik
dr Paweł Powroźnik
dr inż. Małgorzata Plechawska-Wójcik
dr inż. Grzegorz Kozieł
dr inż. Jakub Smołka
dr inż. Jacek Kęsik
dr inż. Piotr Kopniak
dr inż. Elżbieta Miłosz
dr inż. Tomasz Szymczyk
dr inż. Maciej Pańczyk

Skład komputerowy:
Piotr Misztal
e-mail: p.misztal@pollub.pl

Projekt okładki:
Marta Zbańska

JCSI Editorial
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Institute 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:
Mariusz Dzieńkowski
Maria Skublewska-Paszkowska
Edyta Łukasik
Paweł Powroźnik
Małgorzata Plechawska-Wójcik
Grzegorz Kozieł
Jakub Smołka
Jacek Kęsik
Piotr Kopniak
Elżbieta Miłosz
Tomasz Szymczyk
Maciej Pańczyk

Computer typesetting:
Piotr Misztal
e-mail: p.misztal@pollub.pl

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 1-4
Wysłane: 2018-11-04
Przyjęte: 2018-11-17

Porównanie narzędzi do tworzenia aplikacji typu SPA na przykładzie Angular2 i React

Jadwiga Kalinowska*, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest porównanie dwóch, najczęściej stosowanych narzędzi do wytwarzania aplikacji webowych typu SPA (Single Page Application). Analiza została przeprowadzona dla biblioteki ReactJS i dla frameworka Angular. Do badań wykorzystano aplikacje testowe o takiej samej funkcjonalności, zaimplementowane w obu technologiach. W porównaniu uwzględniono strukturę i wydajność aplikacji, wybrane metryki kodu, jakość dokumentacji oraz wsparcie społecznościowe.

Słowa kluczowe: Angular; React; SPA

*Autor do korespondencji.

Adres e-mail: Jadwiga.kalinowska@pollub.edu.pl

Comparison of tools for creating SPA applications using the examples of Angular2 and React

Jadwiga Kalinowska*, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of the article is to compare two, most commonly used tools for the Single Page Application development. The analysis was carried out for the ReactJS library and for the Angular framework. Two applications were implemented for research purposes. The React and Angular test application has the same functionality. The comparison included the structure and performance of applications, selected code metrics, quality of documentation and social support.

Keywords: Angular; React; SPA

*Corresponding author.

E-mail address: Jadwiga.kalinowska@pollub.edu.pl

1. Wstęp

Wzrost zapotrzebowania na portale internetowe wymusił stworzenie nowego podejścia do wytwarzania aplikacji jednostronicowych typu SPA (ang. Single Page Application). Architektura ta pozwala na integrację z użytkownikiem z wykorzystaniem dynamicznego nadpisywania treści, co wpływa znacząco na jej szybkość działania. Cała aplikacja ładowana jest jedynie przy pierwszym uruchomieniu w przeglądarce [4]. Ważnym aspektem a jednocześnie wadą tego rozwiązania jest ingerencja Java Script w strukturę DOM (ang. Document Object Model) [1].

Single Page Application powstała jako alternatywa dla wielostronicowych aplikacji typu MPA (Multi Page Application). MPA cechuje się tym, iż każda strona wysyła żądanie do serwera po czym aktualizowane są wszystkie dane. SPA jest idealnym rozwiązaniem dla platform typu Software as a Service (SaaS), czyli oprogramowanie jako usługa oraz dla platform społecznościowych. MPA zaleca się używać dla sklepów internetowych, stron biznesowych itp. [2].

Architektura SPA posiada gotowe biblioteki i frameworki, pozwala na separację warstwy serwerowej od klienckiej. Istotną cechą, która wpływa na popularność tego rozwiązania jest możliwość ponownego wykorzystania kodu z aplikacji internetowej do tworzenia aplikacji mobilnych [2].

Narzędzia do wytwarzania aplikacji typu SPA dzielą się na dwa rodzaje: fameworki oraz biblioteki. Do najpopularniejszych można zaliczyć : AngularJS, Angular, Ember.js, Knockout.js, Meteor.js, ExtJS, Vue.js i React [3]. W niniejszym artykule porównano bibliotekę React i framework Angular.

Każde rozwiązanie SPA modyfikuje DOM. W tym celu, podczas aktualizacji aplikacji, tworzony jest własny, wirtualny DOM:

- React - kiedy do komponentu dostarczone są nowe właściwości (ang. props) lub zmienił się stan (ang. State) komponentu to React zostaje o tym poinformowany. React monitoruje różnice pomiędzy wstępnym, a obecnym wirtualnym DOM. Następnym krokiem jest wprowadzenie zmian w prawdziwym DOM, ale tylko w zmodyfikowanych węzłach [4].
- Angular 2 – proces ten odbywa się za pomocą biblioteki Zones.js. Biblioteka ta jest w stanie wykrywać zmiany zdarzenia przeglądarki (np. i - kliknięcie, najechanie kursorem na element, wprowadzenie tekstu z klawiatury) oraz żądania Ajax. Przy tworzeniu aplikacji do wszystkich

komponentów przywiązany zostaje detektor zmian. Po wykryciu zmian w szablonie komponentu, dokonane zostaje porównanie obecnej i poprzedniej wartości. W przypadku, gdy wartości nie są takie same - dokonywana jest aktualizacja DOM [4].

Oba rozwiązania mają szerokie zastosowania:

- React wykorzystywany jest w takich serwisach jak Facebook, Instagram, Netflix, Alibaba, Yahoo, E-Bay, Khan-Academy, AirBanB, Sony oraz Atlassian [5];
  Angular wykorzystywany jest do tworzenia narzędzi dostarczanych przez Google.

2. Aplikacja testowa

Jednym z głównych zalet aplikacji typu SPA jest wysoka wydajność, która pozwala płynnie korzystać z aplikacji, bez opóźnień. Do wykonania analizy zostały stworzone bliźniacze aplikacje, posiadające identyczne funkcjonalności. Aplikacje wykorzystują nierelacyjną bazę danych czasu rzeczywistego Firebase oraz magazyn danych Firebase przechowujący pliki graficzne.

Główną funkcjonalnością aplikacji jest galeria zdjęć, wyświetlana w postaci wierszy, po 4 zdjęcia w każdym. Dodawanie zdjęć możliwe jest za pomocą odpowiedniego formularza, dostępnego dla zalogowanego użytkownika.

3. Analiza porównawcza

Do przeprowadzenia porównania obu narzędzi, wybrano 5 kryteriów. Są to:

- struktura aplikacji - obejmuje komponenty interfejsu użytkownika oraz komponenty organizacyjne umożliwiające dostęp do nich;
- metryki kodu - porównano liczbę linii kodu oraz rozmiary plików;
- dokumentacja - ważnym aspektem w analizie narzędzi typu jest dostępność i jakość dokumentacji;
- wsparcie społecznościowe - wzrost popularność programowania sprawił, iż powstały grupy społeczności, które dzielą się swoją wiedzą i pomagają innym rozwiązywać pojawiające się problemy. Największą społeczność można znaleźć na platformie StackOverflow [6];
- testy wydajnościowe – aplikację zostały przetestowane pod względem prędkości pobierania danych potrzebnych do działania aplikacji.

Do testów wykorzystano następujące przeglądarki:

- Google Chrome, w wersji 68.0.3440.106,
- Mozilla Firefox, w wersji 61.02 (64 bity),
- Microsoft Edge, w wersji 42.17134.1.0.

W obydwu aplikacjach stosowana jest wersja ECMAScript 6 i narzędzia deweloperskie udostępniane przez każdą z przeglądarek. W testach wykorzystano m.in. 570 KB obraz o rozdzielczości 1366x768 px. Testy zostały przeprowadzone dla 1, 5, 10, 25, 50 i 75 plików w komponencie galerii.

3.1. Struktura aplikacji

W obydwu przypadkach ciężko jest określić typ modelu architektonicznego. Angular2 oraz ReactJS nie wykorzystuje typowej architektury MVC lub MVVM. W przypadku Angular2 wynika to z tego iż jest on „platformą oraz frameworkiem do budowania aplikacji klienckich w HTML i TypeScript” [14]. Ogólnie architektura Angular2 składa się z modułów, komponentów (widoki) oraz serwisów.

Architektura React oparta jest jedynie na komponentach, ponieważ React jest jedynie biblioteką do tworzenia widoków. Zazwyczaj aplikacja musi zostać wzbogacona o router nawigujący po komponentach oraz Redux do kontrolowania stanów aplikacji JavaScript. Nie jest ważne umiejscowienie plików. Jednak ze względu na lepszą czytelność struktury, zachęca się programistów do przechowywania komponentów w folderze komponent.

W obydwu przypadkach pliki z bibliotekami znajdują się w folderach node_modules. Rozmiar bibliotek dla ReactJS to 240 MB, a dla Angular2 - 339 MB. W obydwu przypadkach biblioteki muszą znajdować się w folderze projektu.

3.2. Metryki kodu

W celu porównania liczby linii kodu wymaganych do poprawnego działania konkretnego komponentu, wykorzystano funkcję dodania zdjęć.

Podczas implementacji aplikacji React dla widoku dodawania zdjęć został zdefiniowany plik Upload.js. Plik jest klasą, która zawiera funkcję render(), renderującą widok komponentu, funkcję dodawania zdjęć i funkcję wyświetlającą pasek postępu. Plik zawiera 103 linie kodu.

W przypadku Angular należało stworzyć dwa pliki komponentu (ts, html), usługę (serwis) i model danych. Pierwszy plik upload.component.html, zawiera definicję wyświetlania komponentu (16 linii kodu). Drugi plik upload.component.ts (31 linii kodu), definiuje funkcję komponentu oraz wywołuje funkcję z serwisu upload.service.ts (48 linii kodu). Funkcja dodaje plik graficzny do bazy danych i magazynu danych Firebase. Ostatnim plikiem jest plik modelu danych upload.model.ts będący klasą Upload. Plik użytkowany jest w serwisie, ma 12 linii. Suma linii kodu plików w aplikacji Angular wynosi 107 linii (Tabela 1). Istotny jest także rozmiar bibliotek wykorzystywanych w obu projektach (Tabela 1).

Tabela 1. Porównanie metryk kodu dla pojedynczego komponentu

		React	**Angular**
Komponent dodawania plików graficznych	Rozmiar plików	3,43 KB	3,16 KB
	Liczba plików	1	4
	Liczba linii kodu	103	107
Rozmiar biblioteki		240MB	339MB
Rozmiar całego projektu		**241MB**	**340MB**

3.3. Dostępność dokumentacji

Zarówno Angular [7] jak i React [8] posiadają obszerne dokumentacje dostępne w języku angielskim. Twórcy zarówno Angular, jak i React zadbali o to, aby w zrozumiały sposób przedstawić podstawy korzystania z nich (np. klarownie opisano wymagane narzędzia, poszczególne kroki tworzenia oraz uruchomienia aplikacji). Obydwa narzędzia posiadają proste, lecz dokładne tutoriale, dzięki którym można zaznajomić się z podstawowymi zasadami oraz możliwościami narzędzi.

Angular2 i React są popularnym tematem artykułów, publikacji, tutoriali w formie wideo i wpisów na stronach związanych z programowaniem. Należy zwrócić uwagę, że w obydwu przypadkach większość materiałów dostępna jest jedynie w języku angielskim. Z powodu trudności w ustaleniu liczby pozycji w języku angielskim, a także polskim dla narzędzi typu SPA, w pracy do porównania dostępności książek skorzystano z internetowej księgarni informatycznej Helion.pl [9] oraz księgarni Google Books [10]. Google Books posiada około 20 milionów wyników dla React, natomiast dla Angular około 15 milionów. Dokładniejsze liczby dostarcza Helion.pl - posiada 21 pozycji dotyczących Angular oraz 19 w przypadku React w wersji Epub/Mobi. Helion w swojej ofercie dysponuje kursami video w liczbie – 5 pozycji dla Angular i 1 związaną z React. Kolejnym aspektem oceny jest liczba udostępnionych tutoriali w internecie – wykorzystano tu wyszukiwarkę Google. Dla React istnieje około 5 500 000 wyników, natomiast dla Angulara zaledwie niecałe 700 tysięcy. Zawężając wyszukiwanie do samego YouTube otrzymano około 4 miliony wyników dla React i blisko 300 tysięcy dla Angular (Tabela 2).

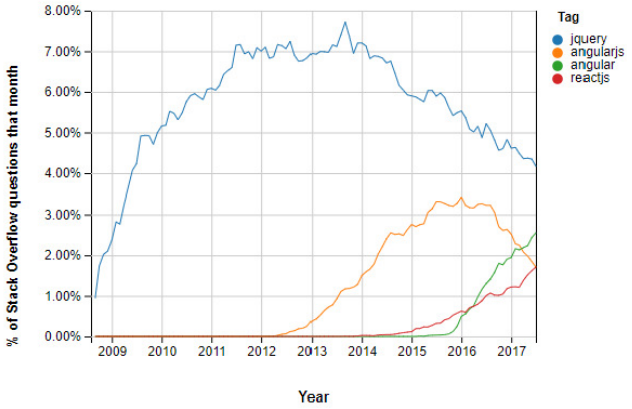
Tabela 2. Porównanie liczby dostępnych materiałów dla obu narzędzi

	Angular2	React
Google Books	15 000 000	20 000 000
Helion – Epub/Mobi	21	19
Helion – kurs video	5	1
Google – tutorial	700 000	5 500 000

3.4. Wsparcie społecznościowe

Największą społeczność można znaleźć na platformie StackOverflow [6]. StackOverflow udostępnia statystykę z wykorzystaniem tagów – słów kluczowych. Tagi zazwyczaj związane są z technologią, której dotyczy pytanie. Serwis posiada narzędzie, które sprawdza trendy zapytań zadawanych na StackOverflow [11]. Tag angular posiada 128099 zapytań [12] z czego 140 z dnia 25.08.2018, 1338 z okresu czasu 7 dni (od 19.08.2018 do 25.08.2018). Tag angular-cli posiada 4843 zapytań. Inne popularne tagi dla Angular2 to : angular5, angular-material, angular2-routing, angular6, angular-ui-bootstrap, angular2-template, angular-ui. Tag reactjs posiada 99538 zapytań [13]. Na dzień 25.08.2018 liczba zapytań wyniosła 159. W przeciągu 7 dni (19.08.2018 – 25.08.2018) liczba zapytań 1219. Popularne słowa kluczowe to: react-native, react-outer, react-redux. Tag create-react-app posiada 1338 zapytań.

Rys. 1 przedstawia zapytania dla najbardziej popularnych narzędzi JavaScript wg procentowych zapytań na StackOverflow w latach 2009-2018. Wykres pokazuje, iż Angular2 i Reactjs są topowymi rozwiązaniami do tworzenia aplikacji.



Rys. 1. Procent pytań StackOverflow w danym miesiącu [14]

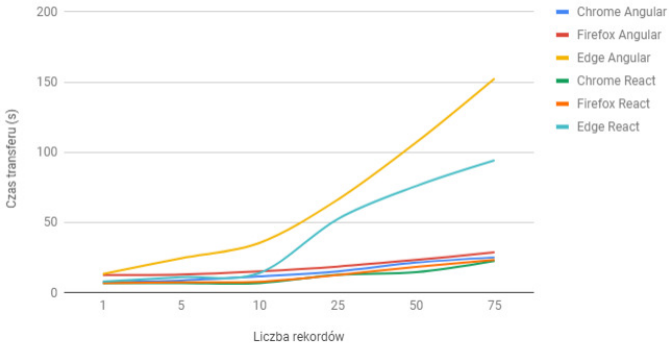
3.5. Wydajność aplikacji

W tabeli 3 przedstawiono czasy transferu danych dla obu aplikacji mierzone, narzędziami dostępnymi w analizowanych przeglądarkach.

Tabela 3. Porównanie czasów pobrania danych z bazy

	Angular			React		
	Czas transferu [s]			Czas transferu [s]		
Liczba rekordów	Google Chrome	Mozilla Firefox	Microsoft Edge	Google Chrome	Mozilla Firefox	Microsoft Edge
1	7,74	12,79	13,48	6,96	7	8,17
5	8,96	13,22	24,73	7,03	7,53	11,35
10	11,78	15,44	35,66	7,08	7,99	14,26
25	15,4	18,75	66,47	12,94	12,88	52,71
50	21,63	23,49	107,15	14,87	18,59	76,1
75	25,34	28,97	152,6	22,88	23,49	94,36

Najlepsze czasy zostały uzyskane przez Google Chrome, najgorsze - odnotowano w przypadku przeglądarki Microsoft Edge. Przeglądarka Mozilla Firefox uzyskała wyniki zbliżone do Google Chrome. W każdym z przypadków aplikacja Angular była nieznacznie wolniejsza w stosunku do React. Biorąc pod uwagę jedynie przeglądarki Google Chrome i Mozilla Firefox różnice w czasu transferu nie przekraczają 10s. W przypadku testów w przeglądarce Microsoft Edge dla 75 rekordów, aplikacja Angular potrzebowała aż 152,6 s, podczas gdy w Chrome rekordy załadowały się w 25 sekund. Na rysunku 2 przedstawiono czasy pobierania różnych liczb rekordów przez aplikacje w poszczególnych przeglądarkach.



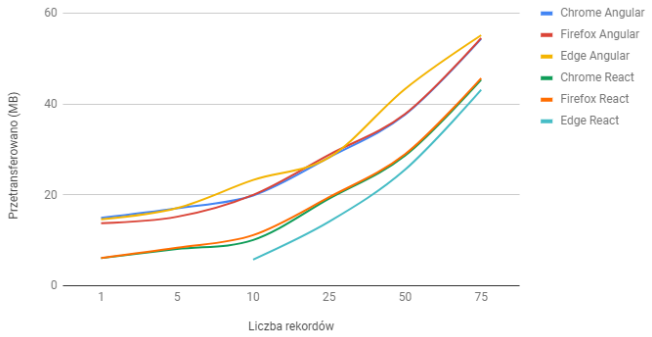
Rys. 2. Czas pobierania różnej liczby rekordów z bazy danych w przeglądarkach

Tabela 4. Analiza zużycia pamięci przy pobieraniu rekordów z bazy w przeglądarkach

	Angular2			ReactJS		
	Przetransferowano [MB]			Przetransferowano [MB]		
Liczba rekordów	Google Chrome	Mozilla Firefox	Microsoft Edge	Google Chrome	Mozilla Firefox	Microsoft Edge
1	15	13,79	14,64	6,1	6,15	0.751
5	17,1	15,22	17,12	8,1	8,38	3.01
10	19,9	20,04	23,31	10,09	11,17	5,79
25	28,3	28,9	28,27	19,2	19,53	14,14
50	37,7	37,86	43,39	28,7	29	25,61
75	54,4	54,57	55,2	45,4	45,71	43,17

W przypadku rozmiaru pamięci - aplikacja React osiąga lepsze wyniki niż aplikacja Angular. Potrzebuje też mniej pamięci do poprawnego działania. Zaskakujące jest jednak to, iż aplikacja React najlepsze wyniki osiągnęła w przeglądarce Microsoft Edge. Natomiast aplikacja Angular potrzebuje podobnego rozmiaru pamięci do transferu rekordów we wszystkich przeglądarkach.

Rysunek 3 porównuje rozmiar zużytej pamięci potrzebnej do pobrania rekordów z bazy danych.



Rys. 3. Graficzna analiza zużycia pamięci danych przy pobraniu rekordów z bazy w przeglądarkach

Tabela 5 przedstawia czas jakiego aplikacja potrzebuje do dodania pliku graficznego do bazy danych i magazynu danych Firebase, a także pobrania go oraz wyświetlenia w aplikacji. React w większości przypadków ma lepsze wyniki. Jedynie w przeglądarce Google Chrome aplikacja Angular okazała się szybsza. Różnica między tymi czasami jest niewielka. Obie aplikacje najwięcej czasu na dodanie zdjęcia potrzebowały w przeglądarce Mozilla Firefox.

Tabela 5. Czas dodawania rekordu z plikiem graficznym do bazy danych

	Angular	React
Przeglądarka	Czas transferu [s]	Czas transferu [s]
Google Chrome	4,16	4,26
Mozilla Firefox	5,74	5,11
Microsoft Edge	4,88	3,76

4. Wnioski

Angular2 jest lepszym narzędziem dla bardziej doświadczonego programisty. W tym przypadku nauka od podstaw jest dosyć trudna i wymagająca głębszej znajomości tematu. Ponieważ Angular2 jest frameworkiem - ciężko jest wdrożyć go do istniejącego projektu. Projekt Angular2 trzeba tworzyć od początku.

React jest łatwiejszym do nauki narzędziem niż Angular2. Ma on przewagę nad Angular ponieważ jest to tylko biblioteka. Pozwala to na wdrożenie komponentów React do istniejących już projektów.

Struktura aplikacji Angular2 jest idealna do dużych serwisów, natomiast struktura React dla mniejszych aplikacji. Komponent React opiera się tylko na jednym pliku. Natomiast komponenty Angular posiada 3 pliki, co pozwala na lepszy podział kodu.

Tabela 6 przedstawia ocenę punktową porównywanych narzędzi ze względu na wskazane kryteria. Skala ocen należy do przedziału 1-10, gdzie 1 jest najniższą oceną, a 10 oceną najwyższą.

Tabela 6. Ocena punktowa porównywanych narzędzi

	Angular2	React
Struktura aplikacji	8	7
Metryki kodu	7	9
Dokumentacja	9	9
Społeczność	9	7
Testy wydajnościowe	7	10
Sumaryczna ocena	**40**	**42**

Narzędzia zdobyły zbliżony wynik w ostatecznej ocenie punktowej. Na podstawie przedstawionych tabel i wykresów można wywnioskować, że React lepiej wypada w testach wydajnościowych oraz ma mniejsze rozmiary plików aplikacji. Oba narzędzia posiadają dokumentację na podobnym poziomie. Angular2 jest z kolei narzędziem z większą społecznością.

Literatura

[1] R. Nowacki, M. Plechawska-Wójcik Analiza porównawcza narzędzi do budowania aplikacji Single, 2016

[2] Single Page Application (SPA) vs Multi Page Application (MPA): Pros and Cons, http://merehead.com/blog/single-page-application-vs-multi-page-application/, [03.08.2018]

[3] "Single-page_application", https://en.wikipedia.org/wiki/Single-page_application [04.08.2018]

[4] Angular vs. React - A comparison, http://work.haufegroup.io/Angular-VS-React/ [11.08.2018]

[5] Vipul A M, Prathamesh Sonpatki, ReactJS by Example - Building Modern Web Applications with React, 2016

[6] StackOverflow Overview 2018, https://insights.stackoverflow.com/survey/2018/#overview [25.08.2018]

[7] Angular Docs Architecture, https://angular.io/guide/architecture [10.08.2018]

[8] React. https://reactjs.org/ [18.08.2018]

[9] https://helion.pl [29.08.2018]

[10] https://books.google.pl/ [29.08.2018]

[11] Stack Overflow Trends, https://insights.stackoverflow.com/trends [25.08.2018]

[12] Stack Overflow angular, https://stackoverflow.com/questions/tagged/angular [25.08.2018]

[13] Stack Overflow reactjs, https://stackoverflow.com/questions/tagged/reactjs [25.08.2018]

[14] Introducing Stack Overflow Trends, StackOverflow, https://stackoverflow.blog/2017/05/09/introducing-stack-overflow-trends/ [25.08.2018]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 5-11
Wysłane: 2018-11-15
Przyjęte: 2018-11-20

Wykonywanie nagrań za pomocą systemu akwizycji ruchu oraz urządzenia mobilnego z synchronizacją wyzwalania nagrań

Karol Walczyna*, Bartosz Jasiński*, Jakub Smołka, Mateusz Miziołek
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Głównym problemem podczas przeprowadzania badania z użyciem systemu motion capture wraz z urządzeniem mobilnym jest brak możliwości jednoczesnego startu nagrywania z systemu akwizycji ruchu, jak i z urządzenia przenośnego typu smartfon. Na potrzeby rozwiązania problemu zostały opracowane dwie metody synchronizacji nagrań. Pierwsza z nich to metoda rozpoczynająca nagrywanie po stronie urządzenia mobilnego, natomiast druga to metoda wychwytująca rozpoczęcie nagrywania w systemie akwizycji ruchu. Obie metody zostały porównane pod kątem opóźnień oraz korelacji zebranych danych pochodzących z urządzenia mobilnego oraz systemu Vicon Nexus.

Słowa kluczowe: Vicon Nexus; przechwytywanie ruchu; synchronizacja

*Autor do korespondencji.

Adresy e-mail: karolwalczyna@outlook.com*, bartosz.jasinski@outlook.com, jakub.smolka@pollub.pl

Recording using a motion capture system and a mobile device with synchronization of the recording triggering

Karol Walczyna*, Bartosz Jasiński*, Jakub Smołka, Mateusz Miziołek
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The main problem during research with a motion capture system and mobile device is lack of possibility to simultaneously begin recording on the motion capture system and a mobile device. Two methods were developed to solve this problem. In the first method the recording is started on the mobile phone and the motion capture system is notified, in the second method the motion capture system starts recording and notifies the mobile device. Both of them were compared for lags and correlation of data form the mobile device and the motion capture system.

Keywords: Vicon Nexus; motion capture; synchronization

*Corresponding author.

E-mail addresses: karolwalczyna@outlook.com*, bartosz.jasinski@outlook.com, jakub.smolka@pollub.pl

1. Wstęp

Laboratorium Analizy Ruchu i Ergonomii Interfejsów oferuje możliwość przeprowadzania badań pod kątem ruchu człowieka oraz innych obiektów, dzięki umieszczeniu na nich specjalnych markerów. Są one rejestrowane przez system optyczny składający się ze specjalistycznych kamer nagrywających obraz w bliskiej podczerwieni oraz dzięki wykorzystaniu współpracującego oprogramowania Nexus w wersji 2.0 firmy Vicon.

Głównym problemem podczas przeprowadzania badania jest brak możliwości jednoczesnego rozpoczęcia nagrywania z systemu akwizycji ruchu, jak i z urządzenia przenośnego typu smartfon. Przy nagrywaniu z wielu źródeł jest ważna ich synchronizacja.

Problem synchronizacji jest na tyle poważny, że uniemożliwił przeprowadzenie kliku projektów oraz prac naukowo-dydaktycznych w Laboratorium Analizy Ruchu i Ergonomii Interfejsów. Dane pochodzące z tych projektów były mocno stochastyczne. Próby korelacji danych pochodzących z systemu akwizycji ruchu i systemu mobilnego nie pozwalały na ustalenie, czy dane były przesunięte w czasie, czy były całkowicie niezwiązane.

2. Przegląd istniejących rozwiązań

Podczas wyszukiwania istniejących rozwiązań wybrano trzy rozwiązania synchronizacji związane z systemem motion capture i innymi urządzeniami zewnętrznymi np. instrumenty biomechaniczne, urządzenia mobilne itp.

Pierwsze rozwiązanie prezentuje zastąpienie standardowego systemu akwizycji ruchu poprzez sieć urządzeń mobilnych, które działają jak czujniki pobierające dane [1]. Dane wysyłane są na serwer, który czeka na wszystkie dane pochodzące od wszystkich czujników. Dane są synchronizowane, a następnie wysyłane z powrotem do urządzeń mobilnych w celu przeprowadzenia ich obróbki. Wszystkie urządzenia składające się na sieć urządzeń biorących udział w nagrywaniu muszą zostać uruchomione zdalnie poprzez wyzwalacz. Ręczne uruchamianie jest nieefektywne ze względu na odstępy czasowe uruchomień kolejnych urządzeń. Zdalnym wyzwalaczem w tym przypadku jest usługa sieciowa. Ta usługa wysyła zdarzenia do wszystkich urządzeń jednocześnie, zarówno do rozpoczęcia, jak i zakończenia nagrania. Na potrzeby takiego rozwiązania

autorzy opracowali protokół do transmisji danych w czasie rzeczywistym poprzez sieć Wi-Fi lub 3G. Opracowany protokół pozwala na szybki i bezstratny transfer danych, które przesyłane są w formie pakietów JSON (ang. JavaScript Object Notation). Problem opóźnień wyzwalania może wystąpić w przypadku braku stabilności oraz odpowiedniej szybkości połączenia internetowego danego urządzenia wykorzystanego w sieci. W celu uzyskania jak najmniejszych opóźnień czasowych zastosowano serwer czasu systemu Android. Podczas uruchomienia każdego z urządzeń czas ustawia się według serwera. Czas rozpoczęcia oraz ukończenia nagrywania jest zapisywany. Dane są dopasowywane na serwerze w zależności od różnicy czasowej pomiędzy wyzwoleniem nagrywania, a momentem rzeczywistego rozpoczęcia nagrywania przez urządzenie.

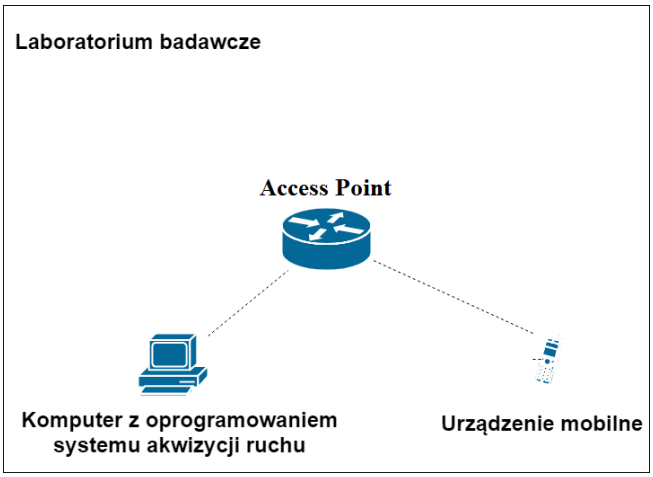
Kolejne rozwiązanie polega na synchronizacji sprzętowej z systemem akwizycji ruchu firmy Vicon. Tematyką pracy jest badanie chodu człowieka przy wykorzystaniu bezprzewodowego wyzwalacza odbierającego i wysyłającego oraz czujnika Shimmera [2]. Czujnik Shimmera wykrywa aktywność chodu i pobiera parametry z czujników inercyjnych. Dane z czujnika Shimmera, zamontowanego na pięcie badanego, dostarczane są do komputera za pomocą protokołu Bluetooth. Dodatkowo dane zostały pobierane osobno z systemu Vicon. Urządzenia nagrywające były wyzwalane poprzez pojedynczy krok. Zbadane opóźnienie wyniosło 2 ms przy 900 próbach. Korelacja danych pochodzących z systemu Vicon oraz z czujnika Shimmera wyniosła 99%. Świadczy to o bardzo dobrej poprawności tej metody synchronizacji.

Ostatnie rozwiązanie zawiera metodę synchronizacji danych pomiędzy systemem akwizycji ruchu, a różnymi urządzeniami biomechanicznymi [3]. Metoda oparta jest na sygnałach analogowych. Wykorzystane zostało oprogramowanie QUARC służące do sterowania w czasie rzeczywistym. Jest ono zintegrowane z programem MATLAB. Jako system akwizycji ruchu wykorzystano system Cortex. Dane zbierane są przez program QUARC znajdujący się na głównym komputerze, który również odpowiada za rozpoczynanie oraz kończenie nagrywania. Następnie obróbka danych odbywa się w programie MATLAB. Komputer główny posiada wyjście analogowe, które jest podłączone z wejściem systemu akwizycji ruchu. Badania zostały przeprowadzone na dwóch systemach akwizycji ruchu, jeden znajduje się na oddzielnym komputerze, natomiast drugi jest zlokalizowany na głównym komputerze, na którym jest zainstalowany program QUARC. W przypadku drugiego urządzenia wymagany jest dodatkowy port analogowy. Sygnał analogowy określa dokładny czas zegara programu QUARC w momencie zbierania danych systemu akwizycji ruchu. Opóźnienie przeprowadzonych badań wynosiło 3,5 ms

3. Opracowane metody synchronizacji nagrań

3.1. Metoda kliencka

Istotą działania metody klienckiej jest uruchomienie nagrania na telefonie, który wyzwala nagrywanie w systemie Vicon Nexus. Utworzono prostą sieć bezprzewodową, w której znajduje się urządzenie mobilne i komputer z oprogramowaniem systemu akwizycji ruchu. Pogląd tej sieci zaprezentowany jest na rysunku 1.



Rys. 1. Schemat sieci

Przesyłanie informacji odbywa się za pomocą protokołu UDP (ang. User Datagram Protocol). Jest to protokół bezpołączeniowy, który pozwala na szybkie dostarczanie informacji [4]. W ramach tego protokołu nie występuję żadna walidacja, czy dany pakiet został dostarczony lub, czy dane są poprawne. Zastosowanie protokołu TCP/IP spowodowałoby znaczne opóźnienie wyzwalania nagrania.

Aby rozpocząć nagranie, do systemu akwizycji ruchu, należy przesłać odpowiedni XML z następującymi informacjami:

- nazwa nagrania;
- notatki;
- opis;
- miejsce zapisu nagrania;
- możliwe opóźnienie;
- techniczne pole identyfikujące pakiet.

Wymagane jest dodatkowo, aby długość przesłanej wiadomości zawierała minimum 500 znaków. Najlepsze rozwiązanie to wypełnienie spacjami do określonej liczby znaków w dowolnym miejscu, poza definicjami atrybutu tak, aby nie została naruszona poprawność schematu XML. Przykładowy schemat XML rozpoczynający nagranie znajduję się w listingu 1.

Przykład 1. Schemat XML rozpoczynjacy nagranie

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<CaptureStart>
    <Name VALUE="nazwa_nagrania"/>
    <Notes VALUE="aa"/>
    <Description VALUE="bb"/>
    <DatabasePath VALUE="D:\NagraniaVicon"/>
    <Delay VALUE=""/>
    <PacketID VALUE="0"/>
</CaptureStart>
```

3.2. Metoda serwerowa

Celem metody serwerowej jest zarejestrowanie zdarzenia rozpoczęcia nagrania przez urządzenie mobilne. Komunikat

o rozpoczęciu nagrania jest wysyłany przez system akwizycji ruchu. Schemat sieci pozostaje taki sam jak na rysunku 1. Tak samo jak w metodzie klienckiej wysyłanie odbywa się z wykorzystaniem protokołu UDP. W momencie kończenia nagrania system Vicon wysyła kolejny komunikat o zakończeniu nagrania. Przykładowy odbierany komunikat znajduje się na listingu 2.

Przykład 2. Odebrany komunikat z systemu akwizycji ruchu

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<CaptureStart>
    <Name VALUE="nazwa_nagrania"/>
    <Notes VALUE=""/>
    <Description VALUE=""/>
    <DatabasePath VALUE="D:\NagraniaVicona\"/>
    <Delay VALUE=""/>
    <PacketID VALUE="0"/>
</CaptureStart>
```

4. Opis badań

W ramach badań metod synchronizacji przeprowadzone zostały wspólne badania wraz z Mateuszem Miziołkiem tworzącym prace dyplomową o tytule „Badanie dokładności pomiaru kątów za pomocą czujników inercyjnych i systemu akwizycji ruchu”.

Badanie metod synchronizacji odbyło się w laboratorium wyposażonym w niezbędny sprzęt potrzebny do działania systemu akwizycji ruchu. Politechnika Lubelska oferuje w zasobach Laboratorium Analizy Ruchu i Ergonomii Interfejsów wymagany sprzęt i oprogramowanie. System akwizycji ruchu został wyprodukowany przez firmę Vicon.

Do działania metody badania kątów użyto następujących narzędzi:

- statyw 3D WT-3130;
- mysz bezprzewodowa na Bluetooth;
- urządzenie mobilne Huawei P10;
- aparatura Systemu Vicon Nexus 2.0.

System motion capture składa z następujących elementów:

- 8 kamer nagrywających w bliskiej podczerwieni (MX-T40S);
- 2 kamer wideo nagrywających obraz rzeczywisty (Bonita 720C);
- urządzenia kalibracyjne (Active Wand);
- MX Giganet;
- komputera PC z oprogramowaniem Nexus.

Skonstruowana sieć składała się ze switcha i punktu dostępowego. Aby telefon mógł znaleźć się w izolowanej sieci lokalnej korzystającej z Wi-Fi, do switcha został podłączony punkt dostępowy. W celu ułatwienia komunikacji zostały ustawione statyczne adresy IP na urządzeniach. Zdecydowano się na wybranie tego typu rozwiązania z dwóch powodów. Pierwszym z nich jest brak zmiany adresu przy utraceniu połączenia z siecią. Drugim jest utrzymanie stałej konfiguracji aplikacji badawczej.

W celu zbadania metod synchronizacji należało użyć wzorca badawczego. Istotą jego jest zbadanie pewnego problemu naukowego przy użyciu urządzeń mobilnych i systemu motion capture. Służył on do otrzymania danych, na podstawie, których następnie ustalono opóźnienia charakteryzujące metody synchronizacji.

Scenariusze wzorcowe zostały opracowane przez Mateusza Miziołka.

Każdy scenariusz wzorcowy został przeprowadzony oddzielnie dla każdej metody synchronizacji nagrań. W ramach danego scenariusza przeprowadzono po 3 próby ze względu na możliwe niepowodzenie nagrania zarówno po stronie systemu mobilnego jak i systemu motion capture. Wszystkich scenariuszy wzorcowych jest siedem, łącznie dla obu metod przeprowadzono 42 próby.

Oznakowanie skrótów scenariuszy wygląda w następujący sposób. Pierwsza litera skrótu S oznacza scenariusz. Następny znak to numer scenariusza. Pierwszy scenariusz podzielony został na 3 odrębne scenariusze. Te scenariusze badają konkretną oś obrotu. W tych scenariuszach następna litera oznacza oś wokół, której odbywa się obrót. Litera R oznacza obrót wokół osi y, P oznacza obrót wokół osi x, a Y oznacza obrót wokół osi z. Scenariusz S1R został dodatkowo podzielony na dwie części oznaczone jako CZ1 oraz CZ2

Nazwy scenariuszy oraz badane kąty obrotu wokół osi znajdują się w tabeli 1.

Tabela 1. Scenaiusze badawcze

Nazwa scenariusza	Badany obrót wokół osi
S1P	OX
S1Y	OZ
S1RCZ1	OY
S1RCZ2	OY
S2	OX, OY, OZ
S3	OX, OY, OZ
S4	OX, OY, OZ

Wszystkie scenariusze zbadano najpierw dla metody klienckiej, a następnie wszystkie dla metody serwerowej. Nie były badane naprzemiennie lecz po sobie.

5. Obróbka nagrnia

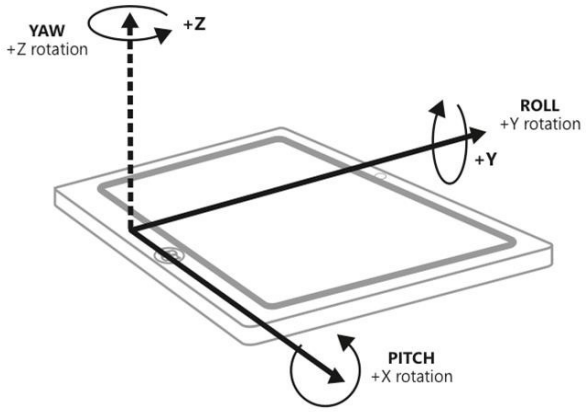
Każde nagranie musiało zostać poddane obróbce polegającej na wykonaniu następujących czynności:

1) Wczytanie pliku nagrania do programu Vicon Nexus.
2) Stworzenie lub wczytanie obiektu (ang. subject) urządzenia mobilnego.
3) Rekonstrukcja trójwymiarowych trajektorii markerów w nagraniu poprzez wybranie opcji Reconstruct.
4) Oznaczenie każdego z markerów odpowiednią ustaloną etykietą.
5) Utworzenie segmentu na podstawie wszystkich markerów.
6) Uzupełnienie luk (ang. gaps) nagrania.
7) Eksportowanie wyników.

Wyeksportowany plik wynikowy zawiera pozycje każdego markera w czasie nagrania.

6. Obliczanie kątów na podstawie punktów

Wszystkie współrzędne markerów zamieniono na odpowiadające im kąty. Kąty wyznaczono zgodnie z artykułem doktora Stevena M. LaValle [5]. Utworzono macierz obrotu ze współrzędnych trzech markerów. Wybrano markery o najmniejszej liczbie luk. Macierz obrotu należy odbierać jako złożenie trzech obrotów wokół osi X, Y, Z [6]. Graficzne przestawienie osi znajduję się na rysunku 2.



Rys. 2. Przedstawienie obrotu wokół osi X, Y i Z względem telefonu [7].

Pierwszy obrót to obrót o kąt γ wokół osi Z. Drugi obrót to obrót o kąt α wokół osi X. Trzeci obrót to obrót o kąt β wokół osi Y. Otrzymana macierz ma postać taką daną wzorem 1.

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{1}$$

gdzie: r_{xy} – jeden z punktów wektora powstałych po utworzeniu macierzy obrotu, x – jeden z trzech wektorów, y – jedna z trzech składowych wektora [8].

Dane pochodzące z urządzenia mobilnego przyjmują wartości w przedziale od -180 do 180 stopni. Do obliczenia wartości kątów wykorzystana została funkcja atan [9], która zwraca wartość arcus tangens z podanego parametru. Ta funkcja zwraca wartości tylko z zakresu od -90 do 90 stopni. Utrudnia to porównywanie danych z obu źródeł. Aby rozwiązać ten problem, wzory z [5] zostały zmodyfikowane tak, aby użyć funkcji atan2 [10], która przyjmuje wartości z zakresu -180 do 180 stopni. Funkcja atan2 zwraca wartość arcus tangens dla dwóch podanych koordynat. Wspomniane wzory przyjmą następującą postać (równania 2-4).

$$\alpha = \mathrm{atan2}(r_{21}, r_{11}) \tag{2}$$

$$\beta = \mathrm{atan2}(-r_{31}, \sqrt{{r_{32}}^2 + {r_{33}}^2}) \tag{3}$$

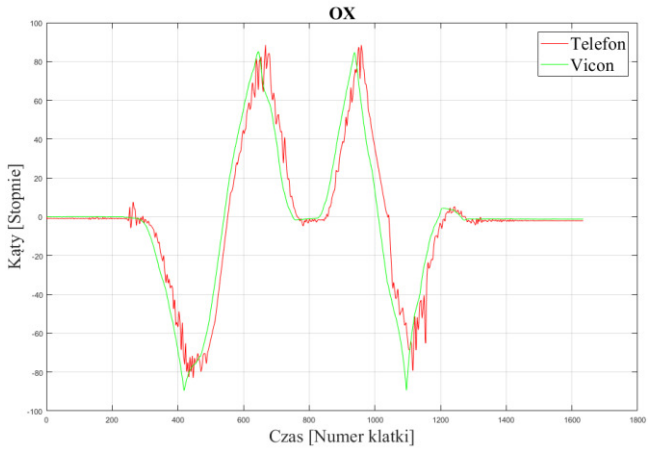
$$\gamma = \mathrm{atan2}(r_{32}, r_{33}) \tag{4}$$

7. Wyniki

Wyniki każdego z badań zostały umieszone w podrozdziałach dla konkretnego scenariusza. Każdy scenariusz posiada tabele z wyliczonymi korelacjami i opóźnieniami dla udanych prób.

7.1. S1P

Na rysunku 3 znajduje się przykładowy przebieg obrotu wokół osi OX.



Rys. 3. Przebieg nagrania dla scenariusza S1P.

W tabeli 2 znajdują się korelacje i opóźnienia dla tego scenariusza.

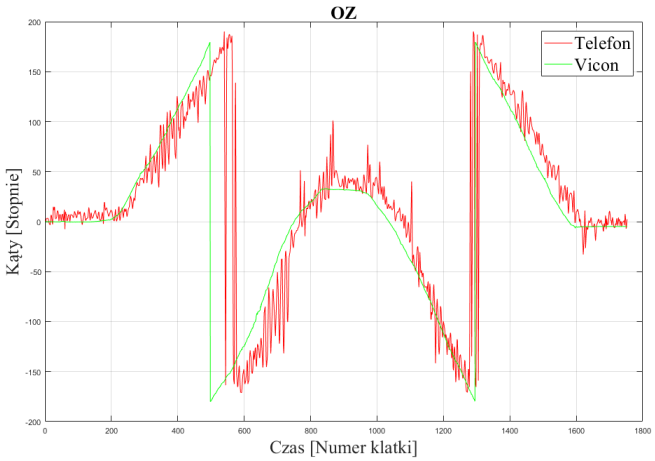
Tabela 2. Opóźnienie oraz korelacja badanych metod dla scenariusza S1P.

Metoda	Próba	OX	
		Opóźnienie [ms]	Korelacja [%]
1	1	260	99,43
	3	230	98,22
2	1	230	99,22
	2	190	99,03
	3	100	97,61

Najmniejsze opóźnienia występują w metodzie serwerowej. Współczynnik korelacji jest na podobnym poziomie.

7.2. S1Y

Na rysunku 4 znajduje się przykładowy przebieg obrotu wokół osi OZ.



Rys. 4. Przebieg nagrania dla scenariusza S1Y.

W tabeli 3 znajdują się korelacje i opóźnienia dla tego scenariusza.

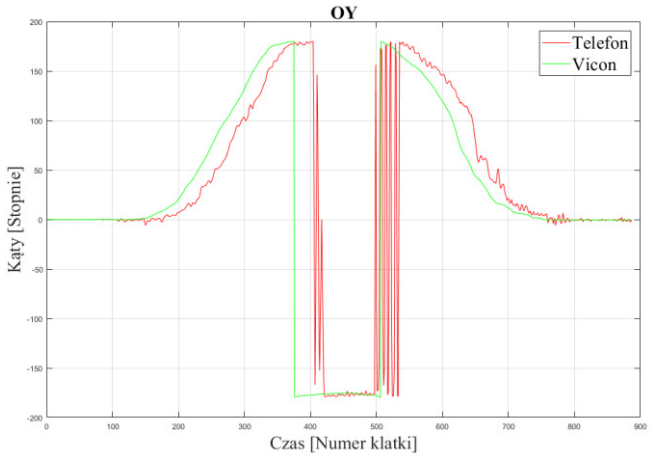
Tabela 3. Opóźnienie oraz korelacja badanych metod dla scenariusza S1Y.

Metoda	Próba	OZ	
		Opóźnienie [ms]	Korelacja [%]
1	1	260	87,88
	2	430	67,99
	3	90	66,23
2	1	540	68,43
	2	540	61,33
	3	570	58,57

Tylko w pierwszej próbie metody pierwszej korelacja była wysoka przy zachowaniu dobrego opóźnienia. Ze względu na niską korelacje te dane nie były brane pod uwagę do porównania metod.

7.3. S1RCZ1

Na rysunku 5 znajduje się przykładowy przebieg obrotu wokół osi OY.



Rys. 5. Przebieg nagrania dla scenariusza S1RCZ1.

W tabeli 4 znajdują się korelacje i opóźnienia dla tego scenariusza.

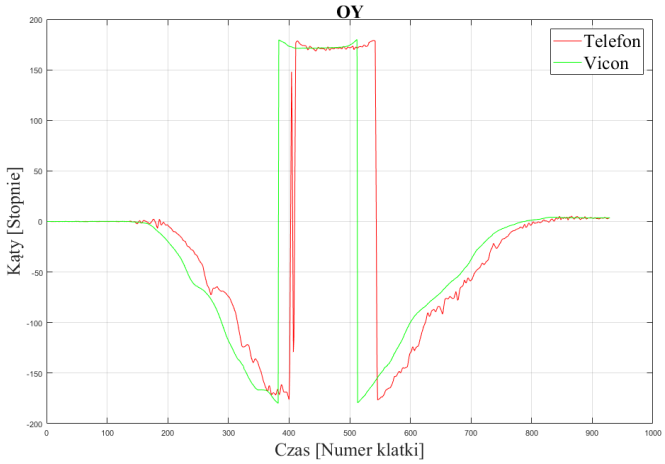
Tabela 4. Opóźnienie oraz korelacja badanych metod dla scenariusza S1RCZ1.

Metoda	Próba	OY	
		Opóźnienie [ms]	Korelacja [%]
1	1	290	61,08
	2	330	71,89
	3	280	87,62
2	1	180	58,13
	2	260	98,91
	3	200	63,11

Korelacje są niskie ze względu na wahania odczytów z urządzenia mobilnego.

7.4. S1RCZ2

Na rysunku 6 znajduje się przykładowy przebieg obrotu wokół osi OY.



Rys. 6. Przebieg nagrania dla scenariusza S1RCZ2.

W tabeli 5 znajdują się korelacje i opóźnienia dla tego scenariusza.

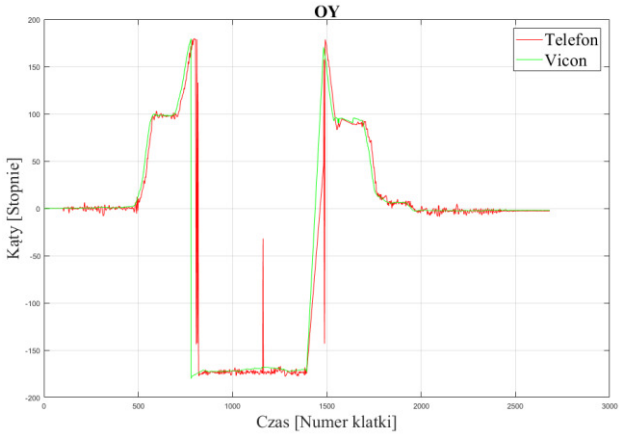
Tabela 5. Opóźnienie oraz korelacja badanych metod dla scenariusza S1RCZ2.

Metoda	Próba	OY	
		Opóźnienie [ms]	Korelacja [%]
1	1	280	92,82
	2	340	85
	3	300	95,78
2	1	220	94,44
	2	230	97,56

We wszystkich przypadkach współczynnik korelacji jest wysoki. Najmniejsze opóźnienia są w metodzie drugiej.

7.5. S2

Na rysunku 7 znajduje się przykładowy przebieg obrotu wokół osi OY.



Rys. 7. Przebieg nagrania dla scenariusza S2.

W tabeli 6 znajdują się korelacje i opóźnienia dla tego scenariusza.

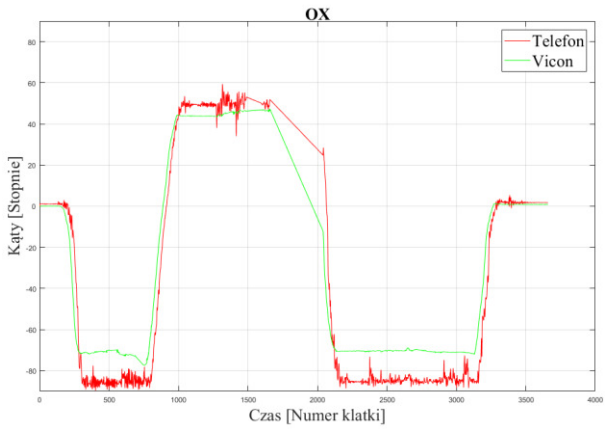
Tabela 6. Opóźnienie oraz korelacja badanych metod dla scenariusza S2.

Metoda	Próba	OY	
		Opóźnienie [ms]	Korelacja [%]
1	2	290	93,38
	3	320	45,65
2	1	210	89,64
	2	250	53,33
	3	240	93,43

W próbie 3 metody pierwszej oraz w próbie 2 metody drugiej jest niższa korelacja ze względu na wahania wartości pomiędzy granicami przedziałów.

7.6. S3

Na rysunku 8 znajduje się przykładowy przebieg obrotu wokół osi OX.



Rys. 8. Przebieg nagrania dla scenariusza S3.

W tabeli 7 znajdują się korelacje i opóźnienia dla tego scenariusza.

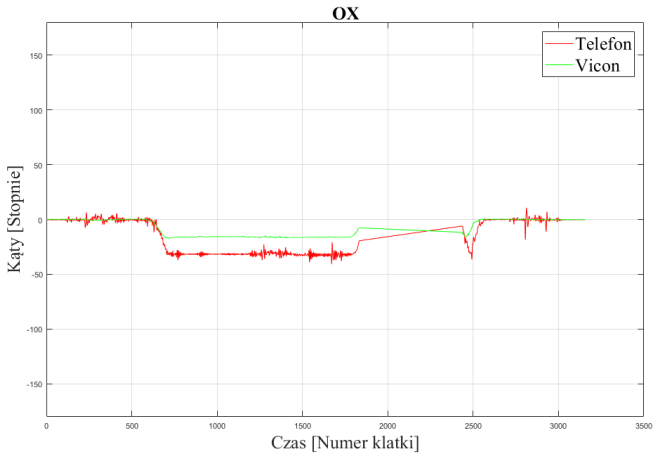
Tabela 7. Opóźnienie oraz korelacja badanych metod dla scenariusza S3.

Metoda	Próba	OY	
		Opóźnienie [ms]	Korelacja [%]
1	3	240	99,72
2	1	190	99,32
	2	180	99,34

Opóźnienia są niskie przy zachowaniu wysokiego współczynnika korelacji.

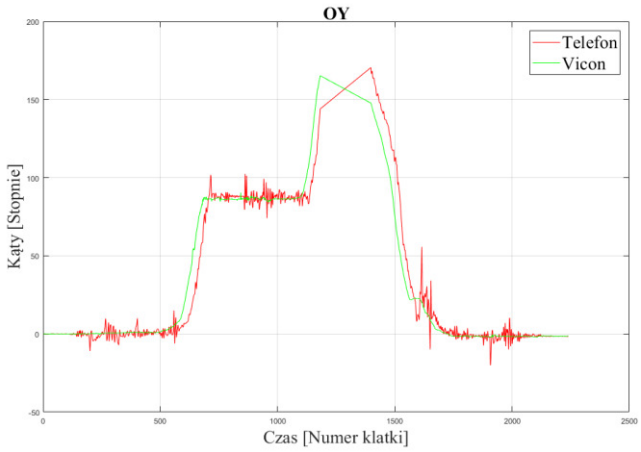
7.7. S4

Na rysunku 9 znajduje się przykładowy przebieg obrotu wokół osi OX.



Rys. 9. Przebieg nagrania osi OX dla scenariusza S4.

Na rysunku 10 znajduje się przykładowy przebieg obrotu wokół osi OY.



Rys. 10. Przebieg nagrania osi OY dla scenariusza S4.

W tabeli 8 znajdują się korelacje i opóźnienia dla tego scenariusza.

Tabela 8. Opóźnienie oraz korelacja badanych metod dla scenariusza S4.

Metoda	Próba	OX		OY	
		Opóźnienie [ms]	Korelacja [%]	Opóźnienie [ms]	Korelacja [%]
1	1	280	96,49	270	99,77
	2	12250	2,98	240	99,33
	3	270	99,24	290	98,72
2	1	180	99,51	260	99,42
	2	30	99,24	0	99,32

W tym scenariuszu zarówno oś X jak i oś Y miały wysoki współczynnik korelacji przy zachowaniu niskiego opóźnienia. W drugiej próbie metody drugiej zerowe opóźnienie świadczy prawdopodobnie o błędzie w nagraniu.

8. Porównanie

Przyjęto następujące kryteria porównawcze opracowanych metod synchronizacji nagrań:

- wygoda użytkowania;

- wykonane próby badania;
- opóźnienia;
- korelacja wyników.

Wygoda użytkowania określa łatwość pracy tzn., czy metoda wymaga dużo pracy przy konfiguracji i przystosowaniu metody. Wykonane próby badania określają efektywność bezawaryjności metody. Im mniej było przeprowadzonych powtórzeń prób tym lepiej. Opóźnienie określa przesuniecie wyników w czasie. Korelacja wyników świadczy o poprawności metody badawczej.

Wygoda użytkowania metody klienckiej polega na tym, że nie trzeba fizycznie rozpoczynać nagrywania przy komputerze zawierającym oprogramowanie Vicon Nexus. Osoba przeprowadzająca badanie może być w tym wypadku również obiektem badanym. Awaryjność metody klienckiej polegała na tym, że raz na jakiś czas nie rozpoczynało lub nie kończyło się nagrywanie po wciśnięciu przycisku w aplikacji mobilnej.

Metoda serwerowa okazała się również łatwa w użytkowaniu. Ta metoda okazałą się metodą bezawaryjną. Wszystkie próby nagrań udały się za pierwszym razem.

W tabeli 9 przedstawiono średnią i medianę dla każdej z metod. Uwzględniono tylko próby, w których korelacja wyniosła powyżej 90%. Aby wyeliminować wysokie wartości opóźnienia spowodowane niską jakością danych z telefonu.

Tabela 9. Średnia i mediana opóźnień dla każdej z metod.

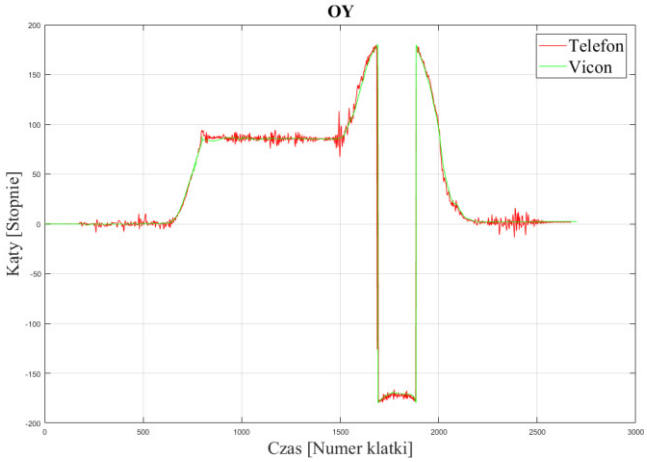
		Kliencka	**Serwerowa**
Opóźnienie	Średnia[ms]	268	187
	Mediana[ms]	270	180

Na podstawie powyższych kryteriów porównawczych najlepszą opracowaną metodą synchronizacji nagrań okazała się metoda serwerowa. Cechuje się głównie bezawaryjnością działania oraz mniejszym poziomem opóźnień w stosunku do metody klienckiej. Patrząc pod kątem użytkowym lepsza wydaję się metoda kliencka głównie ze względu na łatwą obsługę zdalnego rozpoczynania i kończenia nagrań.

9. Wnioski

Metody zostały opracowane i zaimplementowane. Wszystkie metody wykorzystują protokół sieciowy UDP do komunikacji pomiędzy obydwoma systemami. Podstawową różnicą pomiędzy nimi jest podmiot wyzwalający start nagrania. Przeprowadzone badania wykazały, że metody synchronizacji się sprawdziły. Metoda kliencka jest łatwiejsza do implementacji oraz w prowadzeniu badań. W tej metodzie występowały częste awarie startu i stopu nagrywania. Metoda serwerowa przyniosła mniejsze opóźnienie. Zaletą tej metody jest bezawaryjność.

Efektem pracy jest możliwość szybkiego porównania wyników nagrania pochodzących z systemu Vicon Nexus, a innym urządzeniem mobilnym. Znając średnie opóźnienie każdej z metod można zastosować je w pracach i badaniach naukowych np. takich jak przedstawione scenariusze wzorcowe. Przykładowy wygląd dopasowania danych pochodzących z systemu motion capture i urządzenia mobilnego został przedstawiony na rysunku 11.



Rys. 11. Po zastosowaniu dowolnej metody i przesunieciu ze znanym średnim opóźnieniem.

Literatura

[1] Pascu T., White M., Patoli Z.: Motion capture and activity tracking using smartphone-driven body sensor networks, Third International Conference on Innovative Computing Technology (INTECH 2013), 2013.

[2] Kugler P., Schlarb H., Blinn J., Picard A., Eskofier B.: A Wireless Trigger for synchronization of Wearable Sensor to External System during Recording of Human Gait, Conf Proc IEEE Eng Med Biol Soc, 2012.

[3] Komisar V., Novak A.C., Haycock B. A novel method for synchronizing motion capture with other data sources for milliseconds-level precision, Gait Posture, 2017.

[4] Krzysiak K.: Sieci komputerowe. Kompendium. Wydanie II Helion, 2015.

[5] LaValle S.: Planning Algorithms, Cambridge University Press, 2006.

[6] Bułka D., Świder P.: Model pojazdu zastosowany w programie V-SIM do symulacji ruchu i zderzeń pojazdów samochodowych, Zeszyty Naukowe Politechniki Świętokrzyskiej. Mechanika, 2004.

[7] https://stackoverflow.com/questions/32466314/apple-watch-cmdevicemotion-is-not-giving-me-good-readings [18.11.2018]

[8] Evans P.: Rotations and rotation matrices, Acta Crystallographica Section D Biological Crystallography, 2001.

[9] http://www.cplusplus.com/reference/cmath/atan/ [18.11.2018]

[10] http://www.cplusplus.com/reference/cmath/atan2/?kw=atan2 [18.11.2018]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 12-17
Wysłane: 2018-11-14
Przyjęte: 2018-11-20

Dokładność pomiaru kątów z wykorzystaniem czujników inercyjnych w trójwymiarowym układzie współrzędnych

Mateusz Miziołek

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł dotyczy dokładności pomiaru kątów z wykorzystaniem czujników inercyjnych w trójwymiarowym układzie współrzędnych. Jako system odniesienia wykorzystano system akwizycji ruchu firmy Vicon. W artykule opisano przebieg badania, aplikację, która posłużyła do zbierania danych ze smartfona. W artykule zawarto także metody, wzory oraz algorytm, których użyto, aby porównać uzyskane dane.

Słowa kluczowe: czujniki inercyjne; smartfon; akcelerometr; magnetometr; vicon nexus

Adres e-mail: mateusz.miziolek94@gmail.com

Angle measurement accuracy assessment using inertial sensors in three-dimensional coordinate system

Mateusz Miziołek

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

Abstract. This article discusses angle measurement accuracy assessment using inertial sensors in three-dimensional coordinate system. Vicon's acquisition system was used as a reference system. The article describes conduct of the study and application that was used for collecting data from the smartphone. The article also contains methods, formulas and algorithms that were used to compare obtained data.

Keywords: inertial sensors; smartphone; accelerometer; magnetometer; vicon nexus

E-mail address: mateusz.miziolek94@gmail.com

1. Wstęp

Sprzedaż telefonów komórkowych w roku 2018 zanotowała spadek wynoszący 3% [1]. Mimo to, badania firmy IDC dowodzą, że niedługo można się spodziewać wzrostu sprzedaży na poziomie około 2,8% [2]. Dzisiejszy świat miniaturyzacji dotyka każdej dziedziny życia. Miniaturyzacja nie ominęła również medycyny. Jest obecna w e-rehabilitacji. Istnieją aplikacje wspomagające użytkownika w powrocie do pełnej sprawności po złamaniu ręki lub nogi [3]. Takie aplikacje wykorzystują czujniki inercyjne telefonów. Zalet takiego rozwiązania jest mnóstwo. Jednym z nich jest niewątpliwie ogromna wygoda. Sposobów wykorzystania czujników jest więcej, Zależą tylko od pomysłu i potrzeb użytkownika. Można tworzyć zróżnicowane aplikacje. Jedną z nich może być aplikacja do rozpoznawanie rodzaju ruchów na podstawie odczytów sensorów urządzenia mobilnego.

W niniejszym artykule przedstawiono badanie dokładności pomiaru kątów za pomocą czujników inercyjnych. Jako system odniesienia obrano system akwizycji ruchu firmy Vicon. Jest to system stacjonarny. Przeznaczony do pracy w laboratorium. Dzięki wykorzystaniu czujników inercyjnych w telefonie komórkowym można przenieść małą część laboratorium i obliczać kąty w trójwymiarowej przestrzeni w razie takiej potrzeby.

2. Przegląd istniejących zastosowaniach czujników w smartfonie

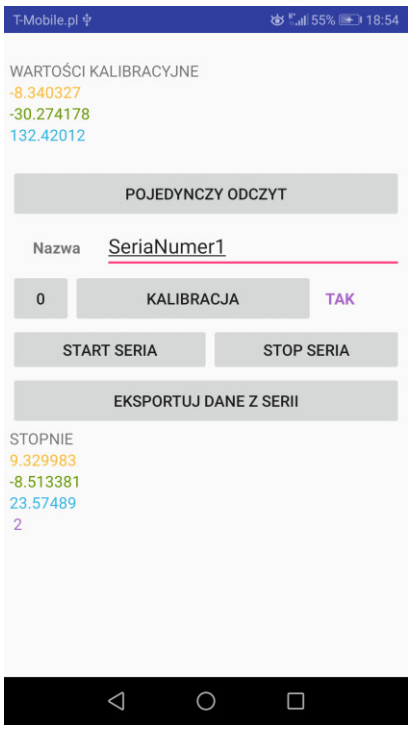
Dzisiejszy standardowy telefon komórkowy to nie tylko aparat do komunikacji, ale też ogromne źródło wielu danych. Smartfon w obecnych czasach posiada bardzo dużo czujników, w tym żyroskop, akcelerometr, czujnik światła, itp. Profesjonalne wykorzystanie takich czujników odkrywa zachowanie użytkownika smartfona. Najlepszym przykładem są aplikacje do joggingu. Telefony komórkowe mogą zastąpić również inne urządzenia takie jak kompas lub GPS. Dzisiaj osoba, która trafia do nowego miasta, nie musi już zaopatrywać się w mapę lub chociażby pytać o drogę. Wystarczy jej odpowiednia aplikacja. Tak samo odpowiednie oprogramowanie w praktycznie takim samym stopniu zastępuje nawigację samochodową. Do zastosowania czujników można podejść bardziej naukowo. Aplikacja łącząca działanie kilku czujników może rozpoznać styl jazdy samochodem i sklasyfikować kierowcę. Do tego typu badań wykorzystać można żyroskop oraz akcelerometr. Dzięki żyroskopowi określane jest położenie kątowe. Natomiast atutem akcelerometru jest udostępnienie danych określających przyspieszenie liniowe. Na podstawie takich danych można określić, jak szybko auto jedzie, jak gwałtownie przyspiesza oraz hamuje. Zinterpretowanie takich danych może dać odpowiedzieć, jakim jest się kierowcą [4,5].

3. Apikacja do pomiarów

Aplikacja mobilna została opracowana dla systemu operacyjnego Android. Korzysta z dwóch czujników - akcelerometru oraz magnetometru [6]. Łączy ona wyniki ich działania za pomocą metody *getRotationMatrix()* [7]. Wynikiem działania tej metody jest macierz obrotu. Dzięki metodzie *getOrientation(),* na podstawie macierzy obrotu, można obliczyć szukane kąty obrotu [8]. Są to:

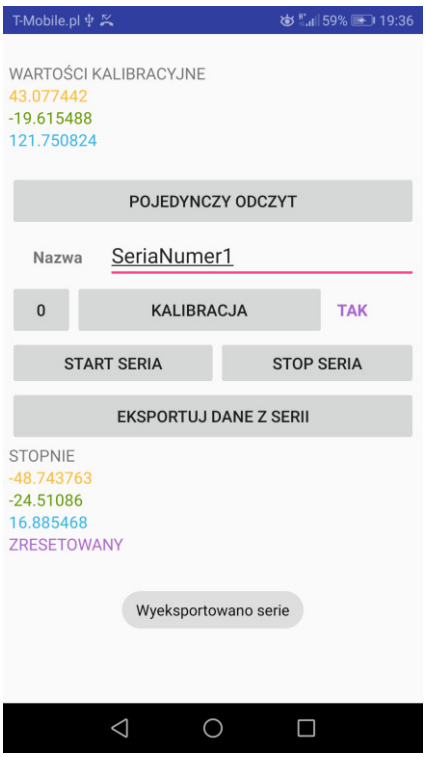
- azymut (ang. Azimuth), czyli kąt obrotu wokół osi -z. Wartość ta reprezentuje kąt pomiędzy osią y urządzenia i biegunem północnym magnesu.
- skok (ang. Pitch), czyli kąt obrotu wokół osi x. Wartość ta reprezentuje kąt pomiędzy płaszczyzną równoległą do ekranu urządzenia, a płaszczyzną równoległą do ziemi.
- rolka (ang. Roll), czyli kąt obrotu wokół osi y. Wartość ta reprezentuje kąt pomiędzy płaszczyzną prostopadłą do ekranu urządzenia, a płaszczyzną prostopadłą do podłoża.

Aplikacja posiada możliwość odczytu pojedynczego, jak i serii odczytów. Są one eksportowane do pliku tekstowego. Istnieje również możliwość kalibracji, czyli zmiany punktu odniesienia. Punkt odniesienia określa ustawienie punktu telefonu w przestrzeni. Natomiast punkt zerowy to punkt, gdy wszystkie kąty obrotu wynoszą zero stopni. Na rysunkach 1 oraz 2 przedstawiono interfejs aplikacji. Kolor jasno pomarańczowy określa kąt obrotu wokół osi y, kolor zielony określa kąt obrotu wokół osi x, a kolor jasnoniebieski określa obrót wokół osi z.



Rys. 1. Interfejs aplikacji

liki, jakie można uzyskać za pomocą tej aplikacji, to plik z pojedynczym zapisem, kalibracją oraz serią zapisów. Kilka rekordów zaprezentowano na przykładzie 1.



Rys. 2. Interfejs aplikacji po zebraniu serii danych

Przykład 1. Seria danych

```
1;18;-4.7431893;-40.2664;20.557674;||;-1.4575933;-
40.2664;-48.070034;
2;42;-1.7075988;-42.43523;31.480942;||;1.5779971;-
42.43523;-37.146767;
3;59;-4.544641;-42.18836;26.97633;||;-1.2590452;-
42.18836;-41.65138;
4;76;-8.535869;-42.960033;20.504326;||;-5.2502728;-
42.960033;-48.123383;
5;93;-1.3106618;-40.74677;31.838531;||;1.9749341;-
40.74677;-36.789177;
6;109;-1.2624013;-41.606133;33.46056;||;2.0231946;-
41.606133;-35.16715;
7;126;-2.963713;-43.51383;32.199604;||;0.32188302;-
43.51383;-36.428104;
8;148;-4.324525;-40.69429;24.223255;||;-1.0389287;-
40.69429;-44.404453;
```

W roli separatora występuje średnik, natomiast jako dodatkowy separator wykorzystano „||”. Pierwsza kolumna to indeks, czyli numer zapisu. Druga wartość to czas w milisekundach. Kolejne trzy wartości to obliczane kąty. Od lewej są to kąty obrotu wokół osi y, x oraz z. Po dodatkowym separatorze znajdują się trzy wartości skalibrowane.

4. Badanie

W celu zbadania dokładności kątów uzyskiwanych za pomocą czujników inercyjnych i systemu akwizycji ruchu w trójwymiarowym układzie współrzędnych przeprowadzono badania. W celach komunikacyjnych skonstruowano sieć lokalną, której zadaniem było umożliwienie komunikacji między telefonem a komputerem, na którym pracował system Vicon. Sam telefon został umieszczony na statywie, a na jego krańcach umieszczono markery w taki sposób, aby utworzyły bryłę imitującą telefon. Całość zaprezentowano na rysunku 3.

Rys. 3. Statyw z telefonem

Badanie przeprowadzono zgodnie z opracowanym wcześniej scenariuszem. Zakładał on badanie każdej osi oddzielnie. Zatem jest podzielony na kilka części. Punktem początkowym był punkt w, którym wartości kątów obrotowych wynosiły 0 stopni.

Dla obrotu wokół osi y określono następujące czynności:

1. Ustawienie telefonu w postaci leżącej tak, aby wartość Roll wynosiła 0 stopni.
2. Przekręcenie telefonu w prawą stronę zgodnie z ruchem wskazówek zegara według osi y, aż do zmiany położenia wartości Roll na wartość 180 stopni.
3. Wykonanie punktu 2 w odwrotnej kolejności. Z kąta o wartości 180 stopni do 0 stopni.
4. Wyeksportowanie pliku S1RCZ1.txt.
5. Obrót telefonu tak, aby zamienić jego krawędź górną z dolną. Ponowne umieszczenie na statywie.
6. Wykonanie punktów od 1 do 3 Różnicą jest to, że wartość Roll zamiast 180 stopni będzie wynosić - 180 stopni
7. Wyeksportowanie pliku „S1RCZ2.txt".

Wynikiem były serie danych o nazwach S1RCZ1 oraz S1RCZ2.

1. Dla obrotu wokół osi x określono następujące czynności:
2. Ustawienie telefonu w postaci leżącej tak, aby wartość Pitch wynosiła 0 stopni.
3. Obrót telefonu w stronę „do siebie". Przekręcając aż do momentu początku (punktu początkowego).
4. Odczekanie sekundy.
5. Obrót telefonu w stronę „od siebie". Przekręcając, aż do momentu początku (punktu początkowego).
6. Wyeksportowanie plik „S1P.txt".

Wynikiem była seria danych o nazwie S1P.

Dla obrotu wokół osi z określono następujące czynności:

1. Ustawienie telefonu w postaci leżącej tak, aby wartość Azimuth wynosiła 0 stopni.
2. Patrząc od góry. Przekręcanie telefonu w prawą stronę zgodnie z ruchem wskazówek zegara, aż do momentu początku (punktu początkowego).
3. Odczekanie sekundę.
4. Patrząc od góry. Przekręcanie telefonu w lewą stronę przeciwnie do ruchu wskazówek zegara, aż do momentu wyjścia.
5. Wyeksportowanie plik „S1Y.txt".

Wynikiem była seria danych o nazwie S1Y.

4.1 Obliczanie wartości kątowych na podstawie nagrań z systemu akwizycji ruchu

Dane uzyskane z systemu Vicon to pozycje markerów w trójwymiarowym układzie współrzędnych. Należało je przeliczyć na kąty obrotowe, aby można było je porównać do kątów zebranych przez smartfon. W tym celu wykorzystano wzory i metody z artykułu Planning Algorithms profesora Stevena M. LaValle [9]. Na początku należało obliczyć macierz obrotu z trzech markerów, które były przymocowane do ekranu. Jako, że operowano w trójwymiarowym układzie współrzędnych macierz wynikowa ma rozmiar trzy na trzy. Co widać we wzorze 1.

$$macierz = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{32} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{1}$$

Na podstawie powyższej macierzy obliczono kąty rotacji wokół osi X (kąt α), Y (kąt β) oraz Z (kąt γ). Wzory dane są równaniami 2,3,4.

$$a = a\tan 2(r_{21}, r_{11}) \tag{2}$$

$$\beta = a\tan 2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \tag{3}$$

$$\gamma = a\tan 2(r_{32}, r_{33}) \tag{4}$$

4.2 Wyznaczanie opóźnień

Kolejnym krokiem było określenie opóźnienia między zebranymi seriami danych. Niestety długości nagrań danych przez system Vicon oraz telefon komórkowy różnią się. Czas w danych pochodzących z urządzenia mobilnego zapisany jest w milisekundach, natomiast w systemie Vicon są to klatki. Jedna klatka to 10 milisekund. Aby wyrównać dane wykorzystano interpolację liniową. Następnie do obliczenia opóźnienia wykorzystywano korelacje. Przykładowy algorytm dla osi Y zaprezentowano na listingu 2. W tym skrypcie zmienna tablicowa OY zawiera serię danych, która

zawiera kąty rotacji pochodzące z systemu Vicon. Zmienna tablicowa OYm przechowuje dane pochodzące z telefonu. Zmienna tablicowa Frame zawiera czasy z nagrania z systemu Vicon. Natomiast t_org odpowiada czasom zarejestrowanym przez telefon. Skrypt jest kompatybilny ze środowiskiem Matlab.

Przykład 2. Skrypt wyliczający opóźnienie

```
len_vicon = size(OY,1);
len_mobile = size(OYm,1);
t = t_org/10;

%Wyrównanie przedziałów danych-POCZĄTEK
for i=len_vicon:-1:1
   if Frame(i) > t(len_mobile)
   ;
   else
      Frame = Frame(1:i);
      OY = OY(1:i);
      break;
   end
end
len_vicon = i ;
%Wyrównanie przedziałów danych-KONIEC

%Przepisanie i interpolacja danych-POCZĄTEK
OYminter = zeros(len_vicon,1);
for i=2:len_vicon
   for j=1:(len_mobile-1)
      if t(j) == Frame(i)
         OYminter(i) = OYm(j);
         break;
      elseif t(j+1) == Frame(i)
         OYminter(i) = OYm(j+1);
         break;
      elseif t(j)< Frame(i) && Frame(i) < t(j+1)
OYminter(i) =OYm(j) +
((OYm(j+1)-OYm(j))/(t(j+1) - t(j))) * (Frame(i)-t(j));
         break;
      end
   end
end
%Przepisanie i interpolacja danych-KOENIEC

%obliczanie opóźnienia-POCZĄTEK
shift = zeros(2);
[r,lag]=xcorr(OY,OYminter);
[max_r, max_r_index] = max(r);
shift(1,2) = -1*lag(max_r_index);
shift(2,1) = lag(max_r_index);

%obliczanie opóźnienia-KONIEC
```

4.3 Przesunięcie rozkładu oraz obliczenie średniego błędu

Najważniejszymi wynikami skryptu zamieszczonego w przykładzie 2 jest seria danych OYminter oraz przesunięcie. OYminter to seria danych z kątami obrotowymi wokół osi Y, która powstała z serii danych OYm po użyciu na niej algorytmu interpolacji liniowej. Przesunięcie należało wykorzystać, aby dopasować serie danych ze smartfona do serii danych z systemu Vicon. Dla tak przygotowanych serii danych obliczano średnią różnicę kątów. Jako, że są to kąty, a ich zasięg waha się od -180 stopni do 180 wykorzystano funkcję atan2 [10]. Równanie 5 przedstawia sposób obliczenia średniej wartości kąta.

$$a\tan 2\left(\sum_{j=1}^{n}\sin\alpha_j, \sum_{j=1}^{n}\cos\alpha_j\right) \tag{5}$$

Kąt α to różnica kątów między poszczególnymi próbkami w seriach z systemu Vicon oraz Smartfona.

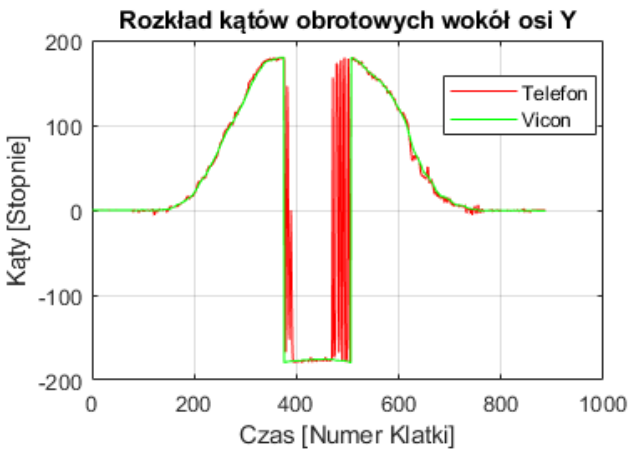
5. Wyniki

Badanie przeprowadzono zgodnie ze scenariuszem zamieszczonym w punkcie 4. Dla każdej części wykonano dwa podejścia. W każdym z nich na nowo ustawiano stanowisko badawcze. W każdym podejściu wykonano trzy powtórzenia scenariusza. Pierwsza część badała kąty obrotu wokół osi Y.

Tabela 1. Wyniki dla pierwszej części scenariusza

Numer Podejścia	Numer Próby	Średnia Różnica [°]
1	1	0,0499
	2	-0,2262
	3	-0,2759
2	1	-0,2856
	2	-0,1362
	3	0,0422

Rozkład kątów obrotu wokół osi Y próby trzeciej w podejściu pierwszym zaprezentowano na rysunku 4.



Rys. 4. Rozkład kątów rotacyjnych OY - podejście 1 – próba 3

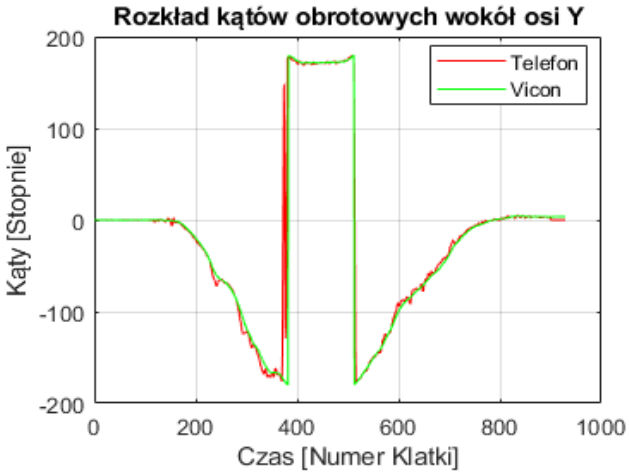
Analizując wyniki zamieszczone w tabeli 1 stwierdzono, że średnia różnica kątów waha się od -0,2856 do 0,2759 stopnia. Najbliżej zera jest wynik ostatniej próby podejścia drugiego i wynosi on 0,0422 stopnia. Jest to najlepszy wynik. W tym przypadku stwierdzono, że dokładność kątów jest bardzo wysoka.

Wyniki dla drugiej części scenariusza zamieszczono w tabeli 2. One również przedstawiają wyniki dla obrotu wokół osi Y.

Tabela 2. Wyniki dla drugiej części scenariusza

Numer Podejścia	Numer Próby	Średnia Różnica [°]
1	1	0,0396
	2	0,2897
	3	-0,2432
2	1	0,0803
	2	0,0997
	3	brak danych

Rozkład kątów obrotu wokół osi Y dla podejścia pierwszego przy pierwszej próbie zamieszczono na rysunku 5.



Rys. 5. Rozkład kątów rotacyjnych OY - podejście 1 – próba 3

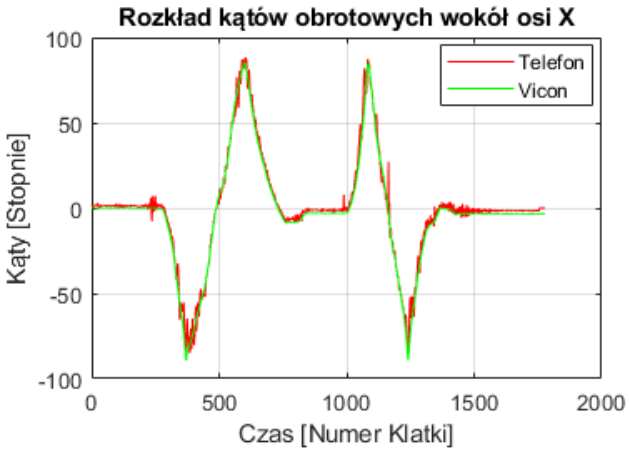
Analizując tabelę 2 w której zamieszczono wyniki dla drugiej części scenariusza stwierdzono, że zakres średniej dokładności waha się od -0,2432 do 0,2897 stopnia. Najbliżej zera jest wynik próby pierwszej w podejściu o tym samym numerze. Sama dokładność w tym przypadku wynosi 0,0396 stopnia. Dokładność kątów w tej części scenariuszu oceniono na bardzo dobrą.

Wyniki dla trzeciej części scenariusza zamieszczono w tabeli 3. Trzecia część przedstawia wyniki dla obrotu wokół osi X.

Tabela 3. Wyniki dla trzeciej części scenariusza

Numer Podejścia	Numer Próby	Średnia Różnica [°]
1	1	0,1981
	2	brak danych
	3	1,174
2	1	1,677
	2	-1,285
	3	-0,848

Rozkład kątów obrotu dla osi X w podejściu drugim dla próby pierwszej zamieszczono na rysunku 6.



Rys. 6. Rozkład kątów rotacyjnych OX - podejście 2 – próba 1.

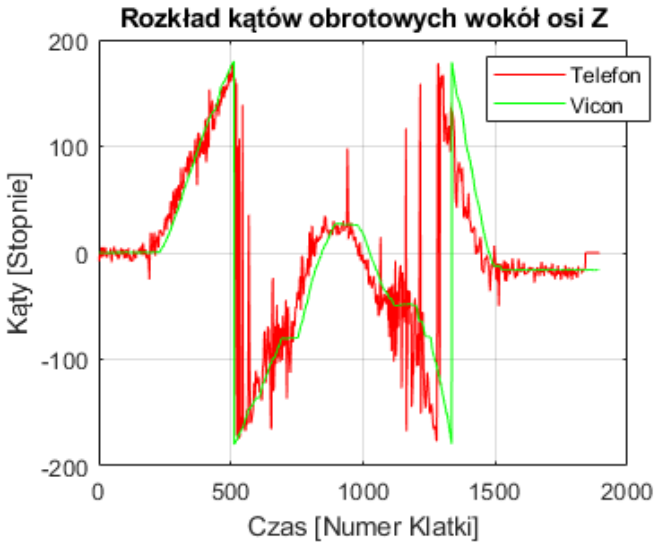
Analizując wyniki zamieszczone w tabeli 3 stwierdzono, że średnia różnica kątów waha się od -1,285 do 1,677 stopni. Najlepszym wynikiem jest wynik uzyskany w podejściu numer jeden w pierwszej próbie. Wynosi on 0,1981 stopnia. Dokładność w trzech przypadkach jest gorsza o ok 1 stopnień niż w poprzednich częściach scenariusza. Mimo to dokładność jest dobra.

Wyniki dla czwartej części scenariusza zamieszczono w tabeli 4. Czwarta część przedstawia wyniki dla obrotu wokół osi Z.

Tabela 4. Wyniki dla drugiej części scenariusza

Numer Podejścia	Numer Próby	Średnia Różnica [°]
1	1	2,9883
	2	2,0259
	3	-0,7042
2	1	-2,6156
	2	-1,4688
	3	2,4699

Rozkład kątów obrotu dla osi Z w podejściu drugim dla próby pierwszej zamieszczono na rysunku 7.



Rys. 7. Rozkład kątów rotacyjnych OZ - podejście 2 – próba 1

Analizując wyniki, które zamieszczono w tabeli 4. Średni wynik różnicy kątów waha się od -2,6156 do 2,9883 stopni. Najlepszy wynik też jest gorszy o ok 0,6 stopnia w stosunku do poprzednich wyników. Wynosi on -0,7042 stopnia. Badając wykres zamieszczony na rysunku 7 stwierdzono, że jest poszarpany. Wyniki jak i rozkład prawdopodobnie spowodowany jest tym, że badanie było przeprowadzone po zbyt małym łuku.

6. Wnioski

Analizując wszystkie wyniki widać, iż wahają się one od -2,6156 do 2,9883 stopni. Natomiast najlepszy wynik to 0,0396 stopnia. Wpływ na wyniki z pewnością miała częstotliwość próbkowania telefonu komórkowego, która wahała się od 51 do 67. Początkowe wyniki ze smartfona miały charakter stochastyczny. Natomiast Vicon pracował z częstotliwością 100 próbek na sekundę. Jego wyniki były deterministyczne, czyli stałe. Jedna próbka co 10 milisekund. Drugim istotnym czynnikiem, który miał miejsce był czynnik ludzki. Telefon obracano ręcznie za pomocą uchwytu na statywie. Mimo iż nogi statywu były wsparte całość lekko zmieniła swoje położenie. Promień był jedynie kilkucentymetrowy, co z pewnością powiększyło wahania wartości kątów. Lepsze wyniki można było uzyskać wydłużając promień obrotu. Dlatego, aby wyniki były bardziej wiarygodne wykonywano badania kilkukrotnie. Średnia dokładność wyniosła około jednego stopnia. Oceniono ten wynik jako bardzo dobry.

Literatura

[1] S. Srivastava, Global Smartphone Market Declined YoY For Second Successive Quarter, 2018.

[2] A. Scarsella, M. Chau, R. Reith, M. Shirer, Smartphones Hit Pivotal Stage as Worldwide Shipment Volumes Decline 0.5% in 2017, But Return to Growth is Expected, According to IDC, 2018.

[3] https://play.google.com/store/apps/details?id=pl.ortomedsport.mobile (dostęp marzec 2018).

[4] M. Van, M. Sujitha, M. Trivedi, Driver Classification and Driving Style Recognition using Inertial Sensors, IEEE Intelligent Vehicles Symposium (IV) Australia, 2013.

[5] D. Johnson, M. Trivedi, Driving Style Recognition Using a Smartphone as a Sensor Platform, International IEEE Conference on Intelligent Transportation Systems Washington, DC, USA., 2011.

[6] https://developer.android.com/reference/android/hardware/SensorManager [03.2018].

[7] https://developer.android.com/reference/android/hardware/SensorManager#getRotationMatrix(float[],%20float[],%20float[],%20float[]) [03.2018].

[8] https://developer.android.com/reference/android/hardware/SensorManager.html#getOrientation(float[],%20float[])[03.2018].

[9] S. LaValle, Planning Algorithms, Cambridge University Press 2006.

[10] https://en.wikipedia.org/wiki/Mean_of_circular_quantities [03.2018].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 18-23
Wysłane: 2018-11-14
Przyjęte: 2018-11-20

Modyfikacje algorytmów planowania trasy uwzględniające ograniczenia czasowe i odległościowe

Mateusz Wolanin*, Klaudia Korniszuk, Jakub Smołka
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia modyfikacje algorytmów wyszukiwania ścieżki w grafie mające na celu wprowadzenie ograniczeń: czasowych lub odległościowych do znalezionej trasy. Zmodyfikowane zostały dwa algorytmy: A* oraz BFS. Zaproponowana została również modyfikacja algorytmu A*, która łączy atuty tych dwóch algorytmów – wygenerowanie najkrótszych tras o jak najmniejszej liczbie wierzchołków. Zmodyfikowane algorytmy umożliwią stworzenie aplikacji pozwalającej na łatwiejsze i bardziej oszczędne poruszanie się z wykorzystaniem usług typu rowerem miejski.

Słowa kluczowe: wyznaczanie trasy; rower miejski; algorytm A*; algorytm BFS

*Autor do korespondencji.

Adresy e-mail: mat.wol20@outlook.com, klaudia714@gmail.com, jakub.smolka@pollub.pl

Modification of path-finding algorithms introducing time and distance limitations

Mateusz Wolanin*, Klaudia Korniszuk, Jakub Smołka
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This paper describes modifications of path-finding algorithms. The modifications add time and distance constraints to generated paths. A* and BFS algorithms are modified. Additionally, A* algorithm modification which combines the advantages (generating the shortest routes with the smallest number of vertices) of A* and BFS is presented.. This allows for creating a route planning app that enables users of bike sharing services to travel more easily and economically.

Keywords: route planning; bike sharing system; algorithm A*; algorithm BFS

*Corresponding author.

E-mail addresses: mat.wol20@outlook.com, klaudia714@gmail.com, jakub.smolka@pollub.pl

1. Wstęp

Podróżowanie zawsze było istotne dla człowieka. Niezależnie czy podróż jest krótka lub długa, największym problemem jest wyznaczenie tak trasy, aby w optymalnym czasie i koszcie dotrzeć w docelowe miejsce. W dzisiejszych czasach dostępnych jest wiele środków transportu: samochody, samolotów lub rowery. Ten artykuł skupia się na planowaniu trasy dla tej ostatniej opcji podróży.

Istnieją systemy wypożyczalni rowerów (z ang. BSS – bike sharing system), gdzie użytkownik może wypożyczyć z jednej stacji rower, by oddać go w dowolnej innej stacji. Dzięki takiemu rozwiązaniu użytkownik może poruszać się po mieście szybciej i sprawniej, niż jakby to robił pieszo.

Zazwyczaj firma udostępniająca rowery miejskie posiada następujący model biznesowy: z góry określony czas od wypożyczenia jest darmowy. Wypożyczenie roweru na dłużej niż wyżej wymieniony czas kosztuje określoną w cenniku ilość gotówki. Dzięki takiemu rozwiązaniu istnieje możliwość przejechania od stacji do stacji bez wydania pieniędzy na wypożyczenie roweru. Takie rozwiązanie ma zapewne na celu zachęcenie ludzi do korzystania z ich środka transportu.

Celem badań autorów było stworzenie takich algorytmów, które umożliwią podróżowanie wyżej wymienionym środkiem transportu w określonym darmowym czasie. Można to wykonać wyznaczając pomiędzy stacją początkową, a końcową stacje pośrednie, na których można wymienić rower. Dzięki temu czas się resetuje i ponownie użytkownik ma do dyspozycji darmowy czas na przejazd do następnej stacji. Przy sprawnym nawigowaniu można bezproblemowo przejechać na miejskim rowerze prawie całe miasto.

W tym artykule zostaną przedstawione algorytmy umożliwiające wyznaczenie drogi od stacji początkowej do stacji końcowej poprzez stacje pośrednie. W tym celu zostaną uwzględnione ograniczenia czasowe lub odległościowe. Zaproponowane algorytmy zostaną porównane ze względu na liczbę wyznaczonych przystanków, koszt trasy oraz czas działania.

2. Przegląd istniejących rozwiązań

Podczas wyszukiwania istniejących rozwiązań nie znaleziono artykułów poruszających ten problem. Badania o zbliżonej tematyce zostały przestawione poniżej,

W pierwszej pracy autorzy publikacji [1] opisują problem rozmieszczenia rowerów na stacjach tak, aby użytkownicy

mogli wypożyczyć rower z wybranej, najczęściej najbliższej stacji. W tej pracy zostały zaproponowane dwa algorytmy wyznaczające trasę z od punktu startowego do najbliższej stacji początkowej gdzie znajdują się rowery, następnie od stacji początkowej do stacji końcowej z wolnymi miejscami dokującymi oraz od stacji końcowej do punktu końcowego. Algorytmy dla *m* stacji oraz *n* użytkowników wyznaczały dla każdego użytkownika trasę. Według autorów tego artykułu, im mniejszy czas podróży, tym większe z niej zadowolenie (jakość podróży). Dlatego ten współczynnik stał się podstawową miarą do wyznaczenia trasy przez algorytm. Pierwszy z nich o nazwie GTP (Greedy Trip Planning) automatycznie przypisuje do użytkownika trasę, która w zbiorze wycieczek ma aktualnie najwyższą jakość pasującą do wymagań użytkownika. Przy tych założeniach można wywnioskować, że tylko pierwsi użytkownicy rowerów miejskich będą mieli zapewnioną najwyższą jakość trasy. Następni po nich użytkownicy będą musieli się liczyć z tym, że na niektórych stacjach startowych nie ma już możliwości wypożyczenia roweru, natomiast na stacjach końcowych może zabraknąć miejsca do zostawienia nowoprzybyłych rowerów. Drugim zaproponowanym algorytmem jest HTP (Humble Trip Planning). W przeciwieństwie do poprzedniego algorytmu GTP, HTP najpierw eliminuje konflikty wśród użytkowników poprzez przypisywanie tras o najwyższej jakości do użytkowników, które mają najwyższy współczynnik zmiany jakości podróży po zmianie przystanku początkowego i\lub końcowego. Dopiero po przydzieleniu użytkownikom najbardziej problematycznych tras reszta użytkowników otrzymuje najlepsze, pod względem jakości, trasy.

Kolejny artykuł [2] opisuje wyznaczenie trasy od punktu początkowego, przez stację początkową i końcową roweru miejskiego, do punktu końcowego dla jednego użytkownika. Jako ograniczenia w wyznaczaniu powyższej trasy przyjęto czas podróży lub odległość do pokonania, bezpieczeństwo na drodze oraz poziom zanieczyszczenia powietrza. Dopiero po uwzględnieniu tych trzech warunków obliczają całkowity koszt podróży na wybranych odcinkach na mapie i na podstawie tego algorytm Dijkstry ma za zadanie wyznaczyć końcową trasę rowerzysty. Na potrzeby przetestowania algorytmu autorzy publikacji [2] stworzyli testowe miasto które zawiera prawie 700 wierzchołków oraz 2620 łuków. W ten sposób odtworzono typowe środowisko miejskie z blokami, drogami oraz parkami. Zaprezentowany algorytm bada wszystkie możliwe trasy pod kątem długości, zanieczyszczenia powietrza oraz bezpieczeństwa i na tej podstawie wyznacza trasę rowerową dla użytkownika. Oprócz tego algorytm również zwraca alternatywne trasy różniące się między sobą tymi trzema współczynnikami.

Następny artykuł [3] przedstawia rozwiązanie problemu podobnego do przedstawionego w tym artykule - wyznaczyć trasę pomiędzy punktami, tak aby była najkrótsza. Różnicą jest to, że w tym rozwiązaniu wyznaczono trasę pomiędzy atrakcjami turystycznymi w danym mieście. Rozwiązują oni problem znalezienia trasy pomiędzy atrakcjami turystycznymi tak, aby wygenerowana droga była najkrótsza oraz aby przedstawiała różnorodne atrakcje turystyczne. W tym celu wprowadzili oni miarę *typeS*. Przyjmuje ona coraz to niższe wartości, gdy kolejna atrakcja jest podobna do poprzedniej. Jako algorytm zastosowali oni algorytm kolonii mrówek (ant colony alghoritm). Algorytm stworzyli Dorigo i Gambardella na podstawie obserwacji kolonii mrówek szukającej najkrótszej trasy od mrowiska do pożywienia. [4] Polega on na tym, że każda mrówka zostawia po sobie ślad tzw. feromon. Gdy mrówka znajdzie pożywienie, wraca do mrowiska tą samą drogą wzmacniając ślad pozostawiony w drodze do pożywienia. Kolejne mrówki wyczuwają ten ślad i kierują się tam gdzie prowadzi. Jako że feromon ulatnia się wraz z powietrzem w czasie, to długie ścieżki lub te, które nie nigdzie nie prowadzą zostają zapomniane. W ten sposób mrówki znajdują najkrótszą ścieżkę do pożywienia nie komunikując się ze sobą tylko pozostawiając ślady.

Ostatni artykuł [5] opisuje problem wyszukiwania trasy w dużym mieście bądź kraju. Według autorów publikacji [5] algorytmy Dijkstry oraz A* działają dobrze w przypadkach, kiedy mapa, na której działają jest względnie niewielka. Natomiast w przypadku wyszukiwania ścieżki pomiędzy dwoma punktami w dużym mieście lub w kraju, z powodów ograniczonej pamięci urządzenia lub ograniczonych zasobów procesora te dwa algorytmy mogą zwracać wyniki w zbyt długim czasie. By rozwiązać ten problem w artykule [5] zaproponowano użycie algorytmu Hierarchical A* (HAS), który jak sama nazwa wskazuje, dodaje mechanizm hierarchii do tradycyjnego algorytmu A*. Algorytm ten dzieli mapę na części, a w miejscu przecięcia krawędzi grafu z granicami sektora wstawia punkty. W przypadku gdy algorytm ma wyznaczyć trasę pomiędzy punktami w różnych sektorach, to najpierw wyznacza trasę z wierzchołka początkowego do wierzchołka końcowego przez punkty znajdujące się na krawędziach sektorów. Dopiero po tym kroku jest generowana trasa pomiędzy wierzchołkami, a punktami na granicach sektorów. Dzięki takiemu rozwiązaniu algorytm może działać na ograniczonej liczbie wierzchołków, generując trasę tylko dla jednej części mapy na raz. Według testów przeprowadzonych przez autorów opisywanej pracy, wraz ze wzrostem odległości pomiędzy punktem startowym, a końcowym od pewnego momentu HAS zaczął zwracać wyniki szybciej niż algorytm A*.

3. Opis badań

Zbadane zostaną trzy algorytmy, które zostały zmodyfikowane, aby uwzględniały zadane ograniczenia czasowe lub odległościowe:

- BFS – jeden z najprostszych algorytmów, służy za podstawę do bardziej złożonych algorytmów [6]. Autorzy zmodyfikowali ten algorytm, dodając uwzględnianie wag krawędzi oraz selekcję wierzchołków na podstawie zadanego ograniczenia. Schemat tej modyfikacji przedstawiony jest na poniższym przykładzie 1.

Przykład 1. Pseudokod algorytmu BFS uwzględniającego zadane ograniczenia

```
Wybierz punkt początkowy i końcowy
Ustal wartość ograniczenia
Dodaj punkt początkowy do kolejki
Dopóki kolejka nie jest pusta
    Pobierz i usuń pierwszy element E z kolejki
    Ustaw E jako odwiedzony
```

```
    Jeśli E jest punktem końcowym
        Zwróć wygenerowaną trasę
    Koniec warunku
    Dla każdego nieodwiedzonego sąsiada S elementu E, dla
    których waga krawędzi N – S jest mniejsza niż zadane
    ograniczenie
        Dodaj S do kolejki
    Koniec pętli
Koniec pętli
Zwróć Null
```

- A* - jeden z najlepszych algorytmów pod względem efektywności wyszukiwania najkrótszej ścieżki w grafie [7,8]. Wykorzystuje on funkcję kosztu, składającej się z sumy kosztu dotarcia do danego wierzchołka oraz przewidywanego kosztu dotarcia z danego punktu do wierzchołka końcowego. Podobnie jak w przypadku algorytmu BFS, ten algorytm również został zmodyfikowany, aby uwzględniał zadane ograniczenia czasowe lub odległościowe. Schemat działania algorytmu A* jest przedstawiony na poniższym przykładzie 2.

Przykład 2. Pseudokod algorytmu A* uwzględniającego zadane ograniczenia

```
Wybierz punkt początkowy i końcowy
Ustal wartość ograniczenia
Dodaj punkt początkowy do kolejki
Ustaw funkcję kosztu = 0 dla punktu początkowego
    Dopóki kolejka nie jest pusta
    Pobierz i usuń element E z kolejki, który posiada
    najmniejszą funkcję kosztu
    Ustaw E jako odwiedzony
    Jeśli E jest punktem końcowym
        Zwróć wygenerowaną trasę
    Koniec warunku
    Dla każdego nieodwiedzonego sąsiada S elementu E, dla
    których waga krawędzi N – S jest mniejsza niż zadane
    ograniczenie
        Oblicz P koszt dotarcia z punktu początkowego do S
        Oblicz K przewidywany koszt dotarcia z S do punktu
        końcowego
        Zapisz funkcję kosztu P+K dla S
    Koniec pętli
Koniec pętli
Zwróć Null
```

- Mod A* - zmodyfikowany przez autorów algorytm A*, który wraz z uwzględnianiem zadanego ograniczenia, oblicza minimalną liczbę stacji pośrednich oraz minimalny koszt pomiędzy stacjami. Na podstawie tych danych algorytm ogranicza liczbę przesiadek na danej trasie, poprzez wybieranie stacji pośrednich do których koszt dojazdu jest większy niż obliczony minimalny koszt.
  Minimalną odległość między stacjami jest obliczana następującym wzorem:

$$MD = d - n \times OD \tag{1}$$

  gdzie: d – odległość pomiędzy stacją aktualną, a stacją końcową, n – minimalna liczba stacji pośrednich, OD – wartość zadanego ograniczenia.

Schemat działania algorytmu jest przedstawiony na poniższym *przykładzie 3*.

Przykład 3. Pseudokod modyfikacji algorytmu A*

```
Wybierz punkt początkowy i końcowy
Ustal wartość ograniczenia
Dodaj punkt początkowy do kolejki
Oblicz minimalną liczbę przesiadek MP, dzieląc koszt trasy
z punktu początkowego do końcowego przez ograniczenie
Ustaw funkcję kosztu = 0 dla punktu początkowego
Dopóki kolejka nie jest pusta
    Jeśli kolejka zawiera stację końcową
        Pobierz i usuń stację końcową E z kolejki
    W przeciwnym wypadku
        Pobierz i usuń element E z kolejki, który posiada
        najmniejszą funkcję kosztu
    Koniec warunku
    Ustaw E jako odwiedzony
    Jeśli E jest punktem końcowym
        Zwróć wygenerowaną trasę
    Koniec warunku
    Oblicz liczbę stacji potrzebną do dotarcia do E
    Jeśli E jest większe lub równe MP
        Zwiększ MP o 1
        Wyczyść kolejkę
        Dodaj element początkowy do kolejki
        Ustaw funkcję kosztu = 0 dla punktu początkowego
        Rozpocznij następną iterację pętli
    W przeciwnym wypadku
        Oblicz minimalny dystans pomiędzy E, a stacją
        sąsiadującą MD za pomocą wzoru 1
    Koniec warunku
    Dla każdego nieodwiedzonego sąsiada S elementu E, dla
    których waga krawędzi N – S jest mniejsza niż zadane
    ograniczenie i większa niż MD
        Oblicz P koszt dotarcia z punktu początkowego do S
        Oblicz K przewidywany koszt dotarcia z S do punktu
        końcowego
        Zapisz funkcję kosztu P+K dla S
    Koniec pętli
    Jeśli istnieje nieodwiedzony sąsiad S elementu E, dla
    których waga krawędzi N – S jest mniejsza niż zadane
    ograniczenie oraz kolejka jest pusta
        Zwiększ MP o 1
        Wyczyść kolejkę
        Dodaj element początkowy do kolejki
        Ustaw funkcję kosztu = 0 dla punktu początkowego
    Koniec warunku
Koniec pętli
Zwróć Null
```

Algorytmy działały na testowych grafach ważonych. Zostały wygenerowane za pomocą żądań sieciowych do [9]. Budowane były na podstawie współrzędnych stacji początkowej oraz końcowej i zwracały plik JSON, zawierający maksymalnie trzy trasy. Z tych danych były wybierane te trasy, które zawierały jak najmniejszą odległość lub czas trwania podróży. W ten sposób zapewniono wagi dla każdego możliwego połączenia w grafie. Każdy testowy graf posiadał 86 wierzchołków.

Algorytmy wygenerowały 7310 tras. Są to wszystkie możliwe połączenia pomiędzy wierzchołkami w testowych grafach. Trasy te zostaną ocenione pod względem trzech kryteriów:

1) Liczba przesiadek na trasie
2) Koszt trasy pomiędzy stacją początkową, a końcową
3) Czas wyszukiwania trasy

Pierwszym głównym kryterium porównawczym algorytmów jest liczba przesiadek między stacjami

wymaganymi by dotrzeć do celu podróży. Trasa wygenerowana przez algorytmy ma być najmniej problematyczna dla użytkownika. Częste i niepotrzebne przesiadki mogą zniechęcić i sfrustrować rowerzystów korzystających z aplikacji, która ma zaimplementowany algorytm nieprzystosowany do tego typu zadania.

Drugim głównym kryterium porównawczym algorytmów jest długość wygenerowanej trasy. Koszt będzie mierzony w metrach w przypadku, gdy jako źródło danych będzie wykorzystany graf ważony z wagami krawędzi równymi dystansowi pomiędzy stacjami. Natomiast, gdy źródłem danych będzie graf, który jako wagi krawędzi będzie miał czas wymagany do przebycia odległości pomiędzy dwoma stacjami, koszt trasy będzie mierzony w sekundach.

Porównanie algorytmów na podstawie kosztu podróży pozwoli określić, który algorytm zwraca najkrótszą, czasowo lub odległościowo, trasę pomiędzy dwoma zadanymi punktami. Za długa trasa ponownie może zniechęcić użytkowników, którzy chcą przejechać od punktu A do punktu B w jak najkrótszym czasie.

Ostatnim, pobocznym kryterium porównawczym będzie średni czas wymagany wygenerowania jednej trasy. Pomimo tego, że komputery osobiste jak i telefony są coraz bardziej wydajne i generowanie trasy w grafie posiadającym 86 wierzchołków nie powinno stanowić wyzwania, autorzy chcą przedstawić jak ich modyfikacje wpływają na szybkość działania algorytmu. Kryterium to będzie badane na komputerze o następujących podzespołach:

- procesor: Intel i7 – 4710HQ, 2,5 GHz,
- 8 GB pamięci RAM,
- dysk SSD Samsung 860 EVO 500 GB.

W przypadku grafu z wagami krawędzi równymi odległościom pomiędzy stacjami wartość ograniczenia wynosi 5000 metrów. Jest to średnia odległość, którą rowerzysta przejedzie z prędkością 16 km/h w 20 minut [10]. Natomiast w przypadku grafu z wagami krawędzi równymi czasowi wymaganym do przejechania pomiędzy dwoma stacjami wartość ograniczenia wynosi 1200 sekund, czyli 20 minut.

4. Wyniki badań

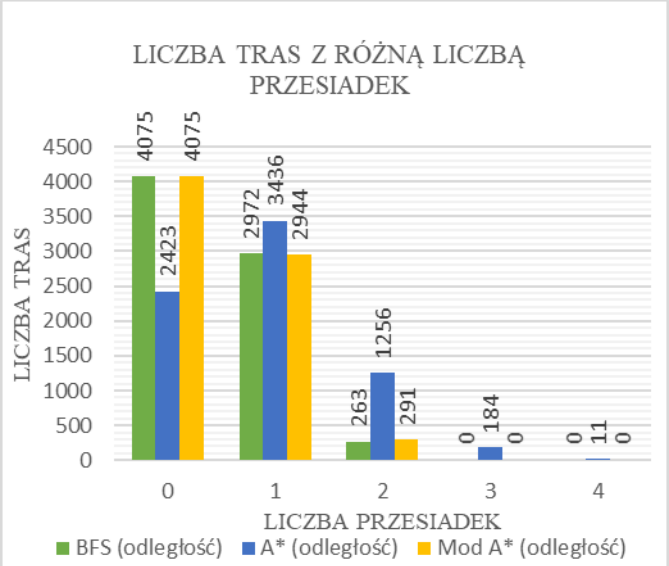
W tym rozdziale zostaną przedstawione porównania wyników opisanych w *rozdziale 3* algorytmów. Najpierw zostaną przedstawione wyniki dla algorytmów działających na grafie z wagami krawędzi odpowiadającymi odległości pomiędzy dwoma stacjami. Następnie zostaną zaprezentowane wyniki dla tych samych algorytmów, które działały na grafie z wagami krawędzi odpowiadającymi czasowi wymaganemu na pokonanie odległości pomiędzy dwoma stacjami.

4.1. Algorytmy uwzględniające ograniczenia odległościowe

Wykorzystane w tym podrozdziale algorytmy zostały zaprezentowane pod poniższymi nazwami:

- BFS (odległość), BFS (odl.) – algorytm BFS uwzględniający zadane ograniczenie,
- A* (odległość), A* (odl.) – algorytm A* uwzględniający zadane ograniczenie,
- Mod A* (odległość), Mod A* (odl.) – zmodyfikowany algorytm A*.

Te trzy algorytmy przyjmują jako ograniczenie odległość podaną w *rozdziale 3*.



Rys. 1. Porównanie liczby przesiadek – algorytmy uwzględniające ograniczenia odległościowe.

Tabela 1. Liczba tras, w danym przedziale kosztu, wygenerowanych przez algorytmy uwzględniające ograniczenie odległościowe.

	Liczba tras		
Koszt (metry)	**BFS (odl.)**	**A* (odl.)**	**Mod A* (odl.)**
<= 5000	4091	4292	4282
> 5000	3219	3018	3028
> 6250	2700	1830	1855
> 7500	1797	929	1006
> 8750	840	400	478
> 10000	248	148	154
Średni koszt trasy	5205,56	4700,58	4772,76

4.2. Algorytmy uwzględniające ograniczenia czasowe

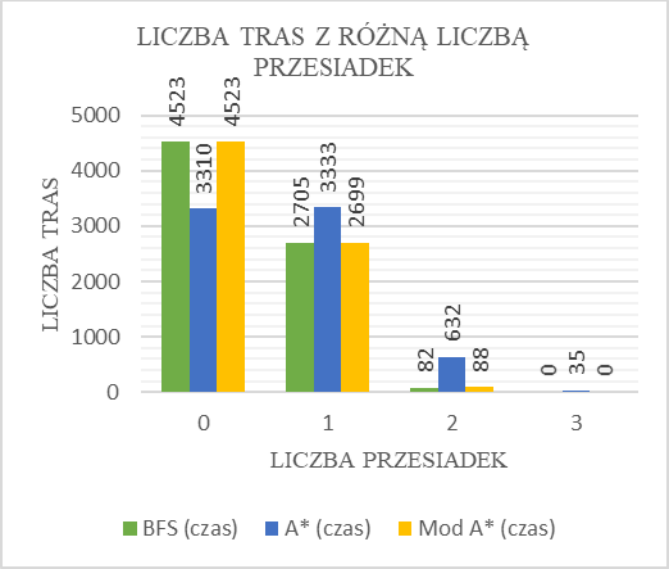
Wykorzystane w tym podrozdziale algorytmy zostały zaprezentowane pod poniższymi nazwami:

- BFS (czas) – algorytm BFS uwzględniający zadane ograniczenie,
- A* (czas) – algorytm A* uwzględniający zadane ograniczenie,
- Mod A* (czas) – zmodyfikowany algorytm A*.

Te trzy algorytmy przyjmują jako ograniczenie czas podany w *rozdziale 3*.

Tabela 2. Liczba tras, w danym przedziale kosztu, wygenerowanych przez algorytmy uwzględniające ograniczenie czasowe.

	Liczba tras		
Koszt (sekundy)	**BFS (czas)**	**A* (czas)**	**Mod A***
<= 1200	4531	4625	4622
> 1200	2779	2685	2688
> 1500	2254	1412	1421
> 1800	1378	602	628
> 2100	499	184	217
> 2400	77	38	40
Średni koszt trasy	1151,64	1055,76	1064,17



Rys. 2. Porównanie liczby przesiadek – algorytmy uwzględniające ograniczenia czasowe.

4.3. Czasy wygenerowania tras

Tabela 3. Czasy wygenerowania 7310 tras przez dany algorytm

Algorytm	Czas (ms)
BFS z zadanymi ogr.	297
A* z zadanymi ogr.	923
Modyfikacja A*	339

5. Wnioski

W przypadku algorytmów uwzględniających ograniczenia odległościowe, zarówno BFS jak i Mod A* wygenerowały podobną liczbę tras o zadanej liczbie stacji pośredniej. Ich wyniki różnią się w przypadku tras posiadających więcej niż jedną przesiadkę, na korzyść algorytmu BFS. Algorytm A* wygenerował najmniej tras nieposiadających przesiadek. Jako jedyny też wygenerował trasy prowadzące przez trzy lub cztery stacje pośrednie. Mogłoby się wydawać, że najkrótsza trasa powinna prowadzić bezpośrednią drogą od punktu początkowego do końcowego zawierająca ewentualne stacje pośrednie znajdujące się na tej drodze jednak algorytm A* zwrócił takie wyniki, ponieważ trasy wygenerowane przez [9], nie są zawsze możliwie najkrótsze. Spowodowane jest to algorytmem zwracającym trasy zaimplementowanym w [9]. Faworyzuje on ścieżki rowerowe i jeśli do punktu trasa może prowadzić przez ścieżkę rowerową, to zostanie ona zwrócona nawet w przypadku, gdy istnieją o wiele krótsze trasy.

Na podstawie wyników zawartych w tabeli 1, algorytm A* zwrócił najkrótsze trasy z pośród wszystkich algorytmów. Średni koszt w przypadku tego algorytmu wyniósł około 4701 metrów. Niewiele dłuższe trasy zwróciła modyfikacja A*, gdzie średnia długość tras wyniosła około 4773 metry. Porównując te dwa algorytmy na podstawie liczby tras w danym przedziale kosztu, to oba algorytmu posiadają zbliżone do siebie wyniki. Większa rozbieżność znajduje się w przypadku tras o koszcie większym niż 7500 oraz 8250 metrów. Algorytm BFS wygenerował najmniejszą liczbę tras o koszcie mniejszym lub równym 5000 metrów. Równocześnie w innych przedziałach, algorytm ten wygenerował największą liczbę tras.

W przypadku algorytmów uwzględniających ograniczenia czasowe, wyniki są podobne do tych wygenerowanych przez analogiczne algorytmy uwzględniające ograniczenia odległościowe. Tutaj też BFS oraz modyfikacja A* wygenerowały najmniejszą liczbę tras posiadających przynajmniej jedną przesiadkę. Natomiast różnica pomiędzy wynikami tych dwóch algorytmów jest znacząco mniejsza. Modyfikacja algorytmu A* wygenerowała o sześć więcej tras z dwoma przesiadkami niż BFS. A* wygenerował najgorsze wyniki z analogicznych powodów opisanych w przypadku algorytmów uwzględniających ograniczenia odległościowe.

Na podstawie wyników zawartych w tabeli 2, trasy o najmniejszym koszcie wygenerował algorytm A*, gdzie średni koszt trasy wyniósł 1056 sekund. Natomiast najdłuższe trasy zwrócił algorytm BFS. W tym przypadku średni koszt jest o prawie 100 sekund większy niż średni koszt algorytmu A*. Modyfikacja A* wygenerowała trasy o średnim koszcie równym 1064 sekundy, niewiele większym niż w przypadku algorytmu A*. Również, algorytmy te zwróciły podobną liczbę tras w poszczególnych przedziałach kosztu dla pojedynczej trasy. Algorytm BFS ponownie wygenerował najmniej tras o koszcie mniejszym lub równym 1200 sekund. Stworzył on również ponad dwa razy więcej tras posiadających koszt większy niż 1800 oraz 2100 sekund niż konkurencyjne algorytmy.

W tabeli 3 zawarto czasy wygenerowania wszystkich możliwych tras w grafach testowych przez poszczególne algorytmy. Najkrócej trasy tworzył algorytm BFS. Potrzebował na to mniej niż 300 milisekund. Zaproponowana przez autorów modyfikacja wygenerowała trasy w trzykrotnie mniejszym czasie niż podstawowa wersja algorytmu A*. Powodem tego jest mniejsza ilość badanych wierzchołków przy generowaniu pojedynczej trasy. Dzięki temu modyfikacja czasowo zbliżyła się do algorytmu BFS. A* potrzebował najwięcej czasu na wygenerowanie poszczególnych tras. Wyniósł on 923 milisekund.

Z pośród przetestowanych algorytmów najlepsza okazała się zaproponowana przez autorów modyfikacja A* (Mod A*). Generowała ona trasy o zbliżonej liczbie przesiadek tak jak w przypadku algorytmu BFS w krótkim czasie. Dodatkowo trasy posiadały niewiele większy koszt niż algorytm A*, który generował najkrótsze trasy, ale za to z największą liczbą przesiadek. Dodatkowo algorytm A* potrzebował najwięcej czasu do wygenerowania wszystkich tras spośród zaprezentowanych algorytmów

Literatura

[1] L GZhi Li, Jianhui Zhang, Jiayu Gan, Pengqian Lu, Fei Lin, Large-Scale Trip Planning for Bike-Sharing Systems, IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems, 2017

[2] Leonardo Caggiani, Rosalia Camporeale, Michele Ottomanelli, A real time multi-objective cyclists route choice model for a bike-sharing mobile application, Politecnico di Bari, 2017

[3] Wen Ouyang, Chang Wu Yu, Pei-Ju Huang, Huai-Tse Chang, Non-commutative path planning for tours with diversified attractions, Chung Hua University, 2017.

[4] Jing Luan, Zhong Yao, Futao Zhao, XinSong, A novel method to solve supplier selection problem: Hybrid algorithm of genetic algorithm and ant colony optimization, Mathematics and Computers in Simulation, Elsevier, 2019

[5] Haifeng Wang, Jiawei Zhou, Guifeng Zheng, Yun Liang, HAS:Hierarchical A-Star algorithm for big map navigation in special areas, 2014 International Conference on Digital Home, IEEE, 2014

[6] K. Khantanapoka, K. Chinnasarn: Pathfinding of 2D & 3D Game Real-Time Strategy with Depth Direction A*Algorithm for Multi-Layer, Eighth International Symposium on Natural Language Processing, IEEE, 2009

[7] W. Lu: Beginning Robotics Programming in Java with LEGO Mindstorms, Rozdział 9, 2016

[8] A. Chaudhari, M. Apsangi, A. Kudale: Improved A-star Algorithm with Least Turn for Robotic Rescue Operations, Computational Intelligence, Communications, and Business Analytics, Springer Nature, 2017

[9] Mapy Google, https://www.google.pl/maps/dir/ [dostęp 10.10.2018]

[10] dr inż. Tadeusz Kopta, mgr Aleksander Buczyński, Marcin Hyła, mgr inż. Bartłomiej Lustofin, Konkurencyjność roweru w zakresie czasu podróży, Warszawa-Kraków, czerwiec 2012

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 24-27
Wysłane: 2018-11-25
Przyjęte: 2018-12-04

Analiza wydajności metod tworzenia aplikacji w technologii Salesforce

Damian Radosław Miącz*

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W poniższym artykule przedstawiono wyniki analizy porównawczej metod tworzenia aplikacji w technologii Salesforce. Zestawiono ze sobą język obiektowy Apex, szkielet programistyczny Visualforce oraz tworzenie aplikacji bez ingerencji w kod, metodą Wskaż-i-kliknij. Analiza polegała na badaniu wydajności stron utworzonych za pomocą każdej z metod. Parametry poddane analizie to czas załadowania strony, wielkość pobranych danych oraz wpływ liczby wyświetlanych rekordów na wydajność strony.

Słowa kluczowe: salesforce; apex; visualforce

*Autor do korespondencji.

Adres e-mail: damian.miacz@pollub.edu.pl

Performance analysis of methods for building applications on the Salesforce platform

Damian Radosław Miącz*

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

Abstract. The presented article describes the results of a comparative analysis of methods for building applications on the Salesforce platform. The following methods were confronted with each other: object-oriented programming language Apex, Visualforce framework and Point-and-click method, which does not require writing any code. The analysis consisted in examining the performance of pages created using each method. The analyzed parameters include page load time, the amount of data that have been downloaded and the impact of the number of records displayed on the page performance.

Keywords: salesforce; apex; visualforce

*Corresponding author.

E-mail address: damian.miacz@pollub.edu.pl

1. Wstęp

Systemy wsparcia sprzedaży i zarządzania relacjami z klientami nieodzownie towarzyszą korporacjom i większym przedsiębiorstwom. Inwestorzy są świadomi jak bardzo ważnym elementem jest kontakt z potencjalnym klientem, przekształcenie go w aktywnego klienta oraz utrzymanie go na stałe. Kolejnym istotnym zagadnieniem jest znaczna automatyzacja procesów, wiąże się to ze zwiększoną wydajnością oraz oszczędnością czasu, co później przekłada się na zwiększone zyski i zwrot kosztów utrzymania i wdrożenia systemu do wsparcia sprzedaży. Jednym z systemów, który usprawnia wcześniej wymienione kwestie jest Salesforce.

Odpowiednia konfiguracja platformy, za pomocą różnych metod tworzenia aplikacji w systemie Salesforce, pozwala na przekształcenie standardowych mechanizmów i budowanie mocno spersonalizowanych rozwiązań. Sprawia to, że aplikacje wykonane na platformie Salesforce mogą znaleźć wiele różnych zastosowań. Daje to na tyle bogate możliwości, że Salesforce może zostać wykorzystywany w wielu różnych branżach. Nie zmienia to faktu, że standardowe aplikacje nadal są w stanie zapewnić szereg funkcjonalności, a są to między innymi:

- Generowanie raportów
- Definiowanie procesów biznesowych
- Tworzenie struktury danych i relacji między nimi
- Zabezpieczenia danych

2. Charakterystyka platformy Salesforce

Platforma Salesforce to system do zarządzania relacji z klientami. Pozwala na przechowywanie danych klientów, udostępnia procesy, które umożliwiają pozyskiwanie potencjalnych klientów. Zrzesza pracowników oraz użytkowników systemu i pozwala im ze sobą współpracować [7]. Jako system do zarządzania relacji z klientami pozwala na śledzenie współpracy z aktualnymi jak i potencjalnymi, przyszłymi klientami. Platforma Salesforce wykorzystywana jest również do nadzorowania kompletnego cyklu życia klienta i uzyskiwania przy tym informacji potrzebnych do odpowiedniego rozwoju i dopasowywania ofert [3].

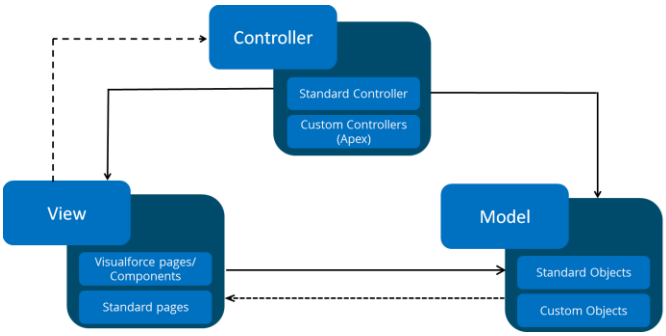
Na platformie Salesforce, wszystkie informacje są przechowywane w chmurze. Dane uporządkowane są w postaci obiektów i rekordów. Obiekt można porównać do karty w arkuszu kalkulacyjnym, natomiast rekord jest jak pojedynczy wiersz danych [3].

Aplikacja to w platformie Salesforce zbiór narzędzi i podstron. Można porównać ją do kontenera, posiadającego swoje logo,

nazwę i podstrony. Wszystkie obiekty, klasy czy wyświetlane strony są niezależne od aplikacji. Aplikacja je tylko tematycznie grupuje.

2.1. Wzorzec architektoniczny

Tworzenie aplikacji na platformie Salesforce opiera się na wzorcu MVC (model-widok-kontroler, ang. model-view-controller). Trzy warstwy w systemie są wyraźnie od siebie oddzielone, co pozwala na łatwe utrzymanie i możliwość modułowości. Model zdefiniowany jest jako obiekty standardowe i niestandardowe lub klasy języka Apex. Warstwa widoku składa się ze stron Visualforce i ich komponentów, które są generowane na serwerze i wyświetlane użytkownikowi w przeglądarce internetowej. Warstwa kontrolera może być zdefiniowana programowo za pomocą klas Apex, lub też może wykorzystywać standardowe kontrolery generowane przez platformę force.com [2]. Graficzna reprezentacja wzorca MVC na platformie Salesforce widoczna jest na rys. 1.



Rys. 1. Wykorzystanie wzorca MVC na platformie Salesforce [12]

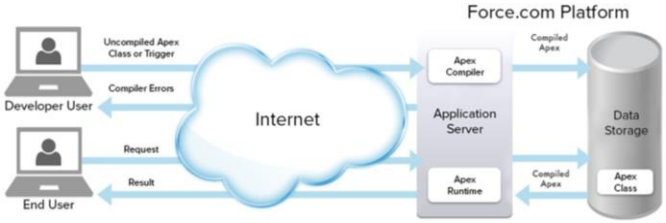
3. Metody tworzenia aplikacji

3.1. Apex

Język programowania Apex jest językiem obiektowym i ma zbliżoną składnię do języków programowania takich jak na przykład Java. Innymi podobieństwami, oprócz składni, są również typy danych, występowanie zmiennych, pętli, interfejsów, klas i obiektów. Na tle innych języków programowania, Apex wyróżnia się przystosowaniem do łatwego tworzenia kokpitów menadżerskich, generowania raportów oraz zarządzania bazą danych [4].

Tworzenie aplikacji w technologii Salesforce jest możliwe dzięki platformie force.com (zwanej również Lightning Platform), która pozwala programistom tworzyć aplikacje wielozadaniowe oparte na rozwiązaniach chmurowych. Wspólne zasoby wykorzystywane są przez wielu użytkowników [8].

Platforma force.com wykorzystuje język Salesforce Object Query Language (SOQL), jest to dedykowany język przeznaczony do pobierania danych z bazy danych, mocno upodobniony do języka SQL, przeznaczonego do obsługi relacyjnych baz danych [2]. Rys. 2 przedstawia strukturę zapytania SOQL.



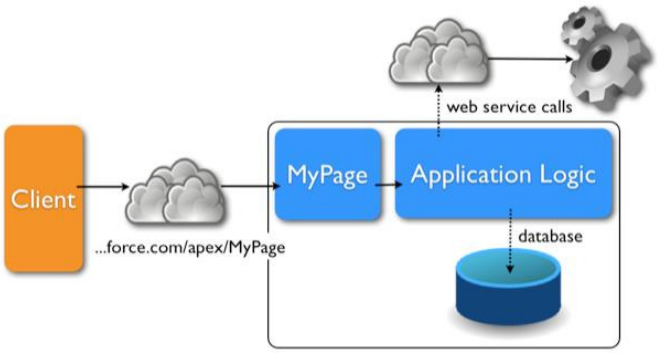
Rys. 2. Struktura zapytania SOQL **Błąd! Nie można odnaleźć źródła odwołania.**

3.2. Visualforce

Visualforce to szkielet programistyczny. Pozwala tworzyć interfejs zarówno na urządzenia mobilne, jak i komputery stacjonarne. Zapewnia dostęp do wbudowanych funkcji platformy Salesforce, ale również umożliwia rozszerzenie bądź wprowadzenie nowej lub niestandardowej funkcjonalności [1]. Ze względu na to, że niektóre standardowe strony są przepełnione informacjami, Visualforce bywa wykorzystywany do tworzenia stron, które mają na celu zebranie najważniejszych informacji dla konkretnych potrzeb pracowników [6].

Składnia kodu obejmuje język oparty na znacznikach, podobny do HTML. Każdy znacznik w Visualforce odpowiada składnikowi interfejsu użytkownika, takim jak sekcja strony lub pole. Visualforce udostępnia wbudowane komponenty oraz mechanizm, dzięki któremu można tworzyć własne komponenty. Visualforce korzysta z automatycznie generowanych kontrolerów dla obiektów bazy danych, zapewniając integrację z bazą danych. Kontrolery Visualforce zapewniają obsługę zdarzeń po wykonaniu określonej interakcji ze stroną wykonaną przy pomocy szkieletu programistycznego Visualforce. Jest to zestaw instrukcji, które aktywują się gdy użytkownik wywoła interakcję, na przykład kliknie w przycisk lub link [9].

Visualforce może zostać zintegrowany z innymi narzędziami, takimi jak JavaScript, jQuery, AngularJS, Bootstrap. Pozwala to na tworzenie atrakcyjnych dla użytkownika stron. Każda utworzona strona za pomocą szkieletu programistycznego Visualforce posiada swój unikalny adres URL. W momencie otworzenia adresu, serwer wyświetla stronę, która dalej może łączyć się z bazą danych i/lub wywoływać usługi WWW [13]. Proces ten, w sposób graficzny, ilustruje rys. 3.

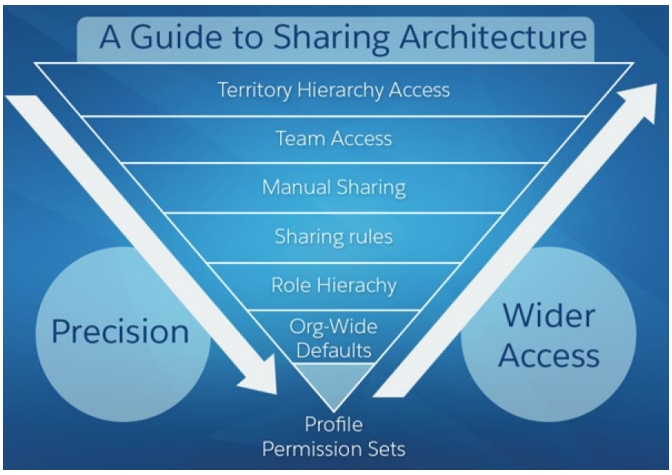


Rys. 3. Proces uruchomienia strony wykonanej za pomocą Visualforce [13]

3.3. Wskaż-i-kliknij

Platforma Salesforce została stworzona w taki sposób, aby jej konfiguracja była w jak największym stopniu edytowalna metodą Wskaż-i-kliknij, czyli całkowicie pozbawiona ingerencji w kod. Wśród opcji można znaleźć konfigurację strony startowej i obiektów dla każdej aplikacji oraz ich układu graficzny.

Platforma Salesforce zapewnia pełną kontrolę nad użytkownikami systemu. Mogą oni zostać podzielni według pełnionych ról lub specyficznych grup. Podział użytkowników jest wykorzystywany do ograniczenia dostępu do danych dla konkretnej grupy odbiorców. Salesforce pozwala na zadeklarowanie ograniczeń dostępu na siedmiu różnych płaszczyznach. Model udostępniania został przedstawiony na Rys. 4.



Rys. 4. Model udostępniania rekordów w zależności właściwości użytkownika [10]

Każda z siedmiu opcji pozwala na bardzo szczegółowe sprezycowanie jacy użytkownicy mają dostęp do jakich danych.

Kolejną funkcjonalnością możliwą do skonfigurowania za pomocą metody Wskaż-i-kliknij jest prognozowanie. Pomaga ono oszacować możliwe przychody w danym okresie. Prognozowanie może zostać wykonane za pomocą zdefiniowanych przez klienta kwartałów fiskalnych lub standardowego roku obrotowego. Prognoza może dotyczyć potencjalnych klientów lub na przykład rodzin produktów [5].

Metoda Wskaż-i-kliknij pozwala na spersonalizowanie standardowych obiektów, które dostępne są na domyślnie skonfigurowanej platformie. Przechowują i porządkują dane użytkowników sytemu, a także pozwalają na generowanie raportów oraz kokpitów menadżerskich.

Dla nietypowych potrzeb platforma Salesforce udostępnia również tworzenie niestandardowych obiektów. Mogę one posiadać własne pola, układ strony oraz powiązania do innych obiektów.

4. Metoda badań

W celu porównania metod tworzenia aplikacji na platformę Salesforce zostały stworzone trzy identyczne moduły: całkowicie za pomocą języka programowania Apex, metodą Wskaż-i-kliknij oraz z wykorzystaniem szkieletu programistycznego Visualforce.

Każda z metod pozwoliła na implementację następujących funkcjonalności:

- Wyświetlenie listy kontaktów w tabeli
- Wyświetlenie imienia, nazwiska, numeru telefonu oraz adresu e-mail w osobnych kolumnach
- Zawężenie wyników do jednego konta
- Przesortowanie wyników

Pierwszy etap testów to porównanie wydajności wyświetlania listy kontaktów z czterema rekordami. Miał on na celu porównanie wydajności wyświetlania tabeli, tekstu i grafik. Druga część testu to porównanie wydajności wyświetlania listy kontaktów, składającej się ze stu wierszy. Ten etap miał na celu porównanie wydajności zapytań bazodanowych.

Do przeprowadzenia testów zostało wykorzystane narzędzie wbudowane w przeglądarkę – Chrome Developer Tools. W celu osiągnięcia jak najbardziej rzetelnych wyników, w narzędziu testowym została zablokowana pamięć podręczna oraz ograniczenie przepustowości łącza do standardu 3G. Tym sposobem, wyniki testów nie były naruszone nieprzewidywalnymi skokami prędkości łącza internetowego. Zablokowana pamięć podręczna spowodowała również, że za każdym przeładowaniem strony przeglądarka od nowa pobierała całość plików potrzebnych do prawidłowego wyświetlenia strony.

5. Wyniki testów

Każda strona, utworzona za pomocą innej metody, została przeładowana pięciokrotnie. Cały proces ładowania został zarejestrowany za pomocą narzędzia Chrome Developer Tools. W drugim etapie testów baza kontaktów w platformie Salesforce została powiększona o dodatkowe rekordy.

Mierzonymi parametrami podczas porównywania stron była liczba zapytań, wielkość pobranych plików, czas odpowiedzi oraz czas całkowitego załadowania stron.

Tabela 1 prezentuje średnią liczbę zapytań wygenerowanych podczas ładowania strony.

Tabela 1. Średnia liczba zapytań wykonana podczas ładowania strony

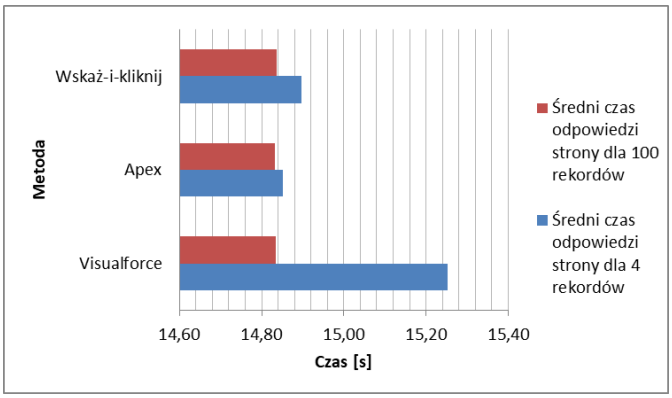
	Średnia liczba zapytań	
	4 rekordy	100 rekordów
Visualforce	57	58
Apex	41	42
Wskaż-i-kliknij	41	40

Tabela 2 przedstawia średnią pobraną wielkość plików. Liczba rekordów miała nieznaczący wpływ na tę wartość. Metoda Visualforce potrzebowała najwięcej danych do pobrania.

Tabela 2. Średnia wielkość plików pobrana podczas ładowania strony

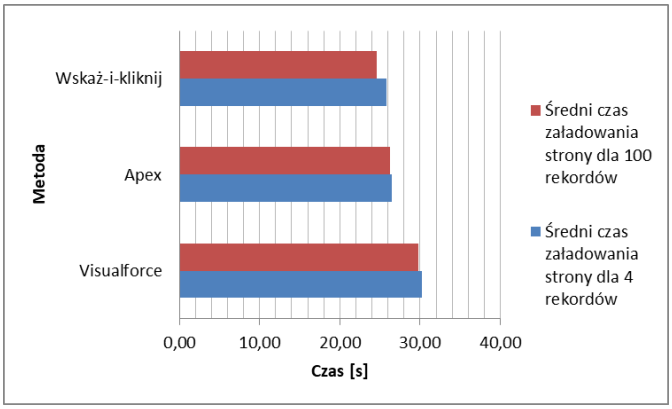
	Średnia wielkość pobranych plików [MB]	
	4 rekordy	100 rekordów
Visualforce	3,4	3,4
Apex	2,8	2,8
Wskaż-i-kliknij	2,7	2,7

Rys. 5 prezentuje porównanie średniego czasu odpowiedzi strony stworzonej za pomocą każdej z metod.



Rys. 5. Porównanie średniego czasu odpowiedzi stron

Rys. 6. prezentuje porównanie średniego czasu ładowania strony stworzonej za pomocą każdej z metod.



Rys. 6. Porównanie średniego czasu ładowania stron

6. Wnioski

Proces badawczy zagadnienia umożliwił zaobserwowanie działania każdej z metod tworzenia aplikacji na platformie Salesforce. Każda z metod pozwoliła na zapewnienie oczekiwanych funkcjonalności i zachowała przy tym stosunkowo zbliżoną wydajność.

Zwiększona liczba rekordów nie pogorszyła wydajności żadnej z metod. Dowodzi to jak dobrze platforma Salesforce przystosowana jest do przetwarzania i analizowania dużych ilości danych. Jednym ze sposobów optymalizacji wyświetlania danych wykazała się strona zaprojektowana za pomocą metody Wskaż-i-kliknij. Wczytywała ona tylko pierwsze rekordy, które użytkownik mógł zobaczyć bez przewijania strony. Kolejne rekordy były ładowane dopiero, gdy użytkownik zdecydował się je zobaczyć. Strona utworzona za pomocą języka Apex wczytywała wszystkie rekordy w jednym momencie.

Przeprowadzanie testów z wykorzystaniem ograniczenia łącza internetowego oraz mocy obliczeniowej procesora wykazało jak bardzo istotny wpływ na wyświetlanie stron w platformie Salesforce mają te czynniki. Czas kompletnego załadowania strony wyświetlającej listę kontaktów, zależnie od metody i liczby rekordów wahał się w granicach 25-30 sekund. Dezaktywowanie ograniczenia łącza internetowego oraz mocy obliczeniowej procesora skutkowało zredukowaniem tego czasu do przedziału 8-15 sekund. Dowodzi to jak bardzo platforma Salesforce uzależniona jest od prędkości łącza internetowego.

Literatura

[1] S. K. Datta, Salesforce for Beginners (Apex And Visualforce): Salesforce Interview Questions 2017 (included) (Edycja Kindle), 2017

[2] A. Poniszewska-Maranda, R. Matusiak, N. Kryvinska, Use of Salesforce Platform for Building Real-Time Service Systems in Cloud, IEEE International Conference on Services Computing (SCC), 2017

[3] A. Manchar, A. Chouhan , Salesforce CRM: A new way of managing customer relationship in cloud environment, Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2017

[4] J. Patel, A. Chouhan, An approach to introduce basics of Salesforce.com: A cloud service provider, International Conference on Communication and Electronics Systems (ICCES), 2016

[5] M. Shrivastava, Salesforce Essentials for Administrators, Packt Publishing, 2014

[6] K. Bowden, Visualforce Development Cookbook : Over 75 Recipes to Help You Create Powerful Custom Pages, Simplify Data-entry, and Enrich the Salesforce User Interface, Packt Publishing, 2013

[7] https://trailhead.salesforce.com/trails/force_com_admin_beginner/modules/starting_force_com/units/starting_intro [22.11.2018]

[8] https://en.wikipedia.org/wiki/Salesforce.com [12.11.2018]

[9] https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_intro_what_is_it.htm [19.11.2018]

[10] http://resources.docs.salesforce.com/206/14/en-us/sfdc/pdf/sharing_architecture.pdf [19.11.2018]

[11] https://trailhead.salesforce.com/en/content/learn/modules/apex_database/apex_database_intro [24.11.2018]

[12] https://www.edureka.co/blog/salesforce-developer/ [23.11.2018]

[13] https://www.janbasktraining.com/blog/create-visualforce-page-salesforce/ [23.11.2018

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 28-31
Wysłane: 2018-11-26
Przyjęte: 2018-12-02

Analiza możliwości testowania aplikacji typu SPA na przykładzie narzędzi Selenium i Protractor

Mateusz Szpinda*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Testy automatyczne mają na celu weryfikowanie funkcjonalności systemu na poziomie interfejsu użytkownika końcowego. W artykule porównane zostały narzędzia Selenium WebDriver oraz Protractor. Do przeprowadzenia analizy porównawczej narzędzi ustalono pięć kryteriów porównawczych na podstawie, których przeprowadzony został proces analitycznej hierarchizacji. Na podstawie otrzymanych obliczeń zostały wyciągnięte wnioski dotyczące tego, która platforma jest lepszym wyborem pod względem testowania aplikacji Single Page Application.

Słowa kluczowe: testy automatyczne; Selenium; Protractor; Single Page Application; Angular

*Autor do korespondencji.

Adres e-mail: mateusz.szpinda94@gmail.com

Analysis of the possibilities of testing SPA applications on the example of Selenium and Protractor tools

Mateusz Szpinda*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Automated tests are designed to verify the functionality of the system at the level of the end-user interface. The article compared popular tools Selenium WebDriver and Protractor. To conduct a comparative analysis tools have been established five benchmarks under which it was carried out the analytical hierarchy proces. Based on the obtained calculations, conclusions have been drawn regarding which platform is a better choice in terms of testing the application of the Single Page Application.

Keywords: automated tests; Selenium; Protractor; Single Page Application; Angular

*Corresponding author.

E-mail address: mateusz.szpinda94@gmail.com

1. Wstęp

Testowanie oprogramowania jest nieodzownym elementem podczas procesu wytwarzania jak najwyższej, jakości oprogramowania. Testy umożliwiają wykrycie błędów we wczesnej fazie rozwoju oprogramowania [1]. Próby zminimalizowania ilości błędów występujących w wersjach produkcyjnych systemów informatycznych od lat skutkowały ewolucją procesu testowania. Szczególnym rodzajem testów, na które w coraz większym stopniu kładzie się nacisk są testy automatyczne, które znacząco przyspieszają proces testowania. Głównym celem stosowania testów automatycznych jest zmniejszenie nakładu pracy, jaką trzeba wykonać by przetestować tworzoną aplikacje [2].

2. Przegląd literatury

Porównanie Selenium i Protractor nie było jak dotąd poruszane w literaturze naukowej. Główną tego przyczyną jest fakt, że Protractor jest nowym narzędziem na rynku służącym do testowania aplikacji internetowych. Można natomiast znaleźć pozycje przedstawiające tematykę testowania aplikacji opartych na frameworku Angular oraz porównania między innymi Selenium WebDriver z innymi technologiami wspierającymi testy automatyczne.

Tobias Palmer oraz Markus Waltre przedstawiają architekturę aplikacji Single Page Application oraz opisują testowanie SPA na przykładzie narzędzia Protractor [3].

Harsh Bajaj w swoim artykule pisał na temat wyboru technologii służących do wykonywania testów automatycznych [4]. Szukając najlepszego rozwiązania brał pod uwagę takie kryteria jak wspierane systemy i przeglądarki, łatwość wdrożenia w technologie oraz łatwość tworzenia skryptów testowych.

Badanie zostanie przeprowadzone przy pomocy jednej z metod analizy wielokryterialnej zwanej procesem analitycznej hierarchizacji. Paweł Cabała w swoim artykule przedstawia założenia teoretyczne tej metody [5]. Dodatkowo został przedstawiony praktyczny przykład zastosowania procesu.

3. Procedura badawcza

3.1. Cel

Celem pracy jest analiza możliwości testowania aplikacji internetowych typu Single Page Application (SPA) na przykładzie narzędzi Selenium WebDriver oraz Protractor. Oba narzędzia służą do przeprowadzania testów automatycznych aplikacji internetowych. Analiza zostanie

przeprowadzona przy pomocy procesu analitycznej hierarchizacji [5]. W tym celu zostały określone kryteria porównawcze takie jak dokumentacja techniczna, wspierane platformy, możliwości testowania SPA, wyszukiwanie elementów oraz szybkość wykonywania testów.

3.2. Technologie

3.2.1. Selenium

Selenium jest jedną z najpopularniejszych platform służących do wykonywania testów automatycznych stron internetowych. Framework powstał w 2004 roku, którego twórcą był Jason Huggins. Narzędzie umożliwia pracę w wielu popularnych językach programowania takich jak Java, C#, JavaScript, Python itp. W skład Selenium wchodzą następujące narzędzia: Selenium IDE, Selenium RC, Selenium WebDriver oraz Selenium Grid [6].

3.2.2. Protractor

Protractor jest darmowym narzędziem służącym do tworzenia testów automatycznych. Został opracowany przez zespół programistów firmy Google Inc. Protractor jest napisany w języku JavaScript oraz jest oparty o Selenium WebDriver. Zaletą narzędzia jest fakt, że oferuje dedykowane funkcje służące do testowania aplikacji Angular.js oraz Angular [3]. Protractor współpracuje z takimi frameworkami jak Jasmine, Mocha itp.

3.3. Aplikacja testowa

W celu przeprowadzenia procedury porównawczej powstała aplikacja internetowa symulująca działanie sklepu internetowego. Część serwerowa została napisana w środowisku uruchomieniowym Node.js z wykorzystaniem frameworka Express oraz nierelacyjnej bazy danych MongoDB zgodnie z wzorcem REST (ang. Reprezentational State Transfer). Za warstwę klienta odpowiada framework Angular 5, Angular Material oraz bootstrap 4. System można podzielić na dwa moduły: część przeznaczoną dla użytkownika oraz dla administratora systemu. Część użytkownika umożliwia korzystanie z następujących funkcjonalności: rejestracja, logowanie, edycja danych osobowych, dodawanie produktów do koszyka, dodawanie produktów do listy życzeń. Część administracyjna udostępnia następujące funkcjonalności: zmiana statusów zamówień, funkcje CRUD (ang. create, read, update and delete) na katalogach, produktach oraz użytkownikach.

3.4. Kryteria

3.4.1. Dokumentacja

Istotnym punktem podczas wyboru nowej technologii, z której programista ma korzystać jest dokumentacja techniczna. Dobrze sporządzona dokumentacja cechuje się tym, że zawiera prawidłowo opisane funkcjonalności oraz praktyczne przykłady ich użycia. Dokumentacja powinna być przygotowana w sposób uporządkowany, intuicyjny, aby programista korzystający z niej mógł znaleźć poszukiwane informacje przy jak mniejszym nakładzie czasu.

3.4.2. Wspierane platformy i języki

Niewątpliwie wsparcie wielu platform bądź języków programowania jest istotnym elementem podczas wyboru odpowiedniego narzędzia [7]. Istnieją narzędzia, które wymuszają na programiście korzystanie z danego języka z powodu braku wsparcia innych platform. W przypadku projektów o dużej skali, w której wykorzystywane są różne języki programowania istotne jest, aby narzędzia testowe były dokładnie w tym języku, co dana część projektu.

3.4.3. Możliwości testowania SPA

W aplikacjach webowych typu Single Page Application przepływ danych miedzy klientem a serwerem funkcjonuje w sposób asynchroniczny za pomocą protokołu http [8]. Narzędzia służące do testowania tego typu aplikacji mają za zadanie obsłużyć asynchroniczne funkcjonowanie aplikacji. Głównym problemem obsługi tego typu zdarzeń jest nieprzewidywalność komponentów aplikacji. Czas od momentu wysłania żądania do serwera, w celu pobrania bądź wysłania danych, do momentu otrzymania odpowiedzi jest nieznany i zależny między innymi od takich czynników jak odległość serwera, ilość danych oraz poziom zaawansowania ich przetworzenia.

3.4.4. Wyszukiwanie elementów

Wyszukiwanie elementów na stronie jest istotnym kryterium podczas badania oraz korzystania z narzędzi służących do tworzenia testów automatycznych. Są one podstawowymi funkcjonalnościami, jakie oferują narzędzia, które są wykorzystywane podczas każdego scenariusza testowego. Za ich pomocą symulowane są interakcje ze stroną naśladujące zachowanie użytkownika. Ponadto znajdowanie elementów na stronie może służyć kontrolowaniu poprawności wyświetlanych danych, sprawdzaniu czy dane się pojawiły bądź oczekiwaniu na element.

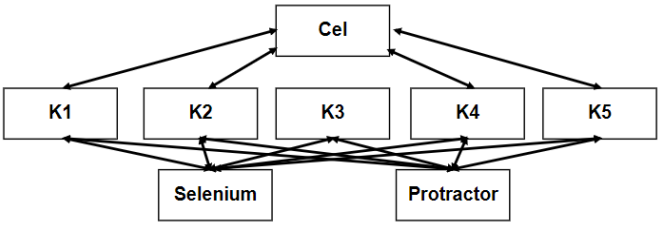
3.4.5. Szybkość

Czas, w jakim testy zostaną wykonane jest jednym z kluczowych kryteriów porównawczych narzędzi służących do testowania aplikacji SPA [9]. Testy automatyczne mają za zadanie symulować zachowanie użytkownika aplikacji, a co za tym idzie, wykonanie ich może być bardzo czasochłonne, szczególnie w przypadku systemów o dużej skali.

3.5. Proces hierarchizacji

Pierwszy etap polega na utworzeniu hierarchicznej struktury procesu decyzyjnego. Na najwyższym poziomie struktury hierarchicznej powinien znajdować się główny cel, który zostaje rozkładany na niezależne od siebie kryteria oceny, wyznaczone przez decydenta. Kryteria te znajdują się na następnym poziomie hierarchii. Na najniższym poziomie struktury hierarchicznej znajdują się rozpatrywane warianty

decyzyjne. Schemat struktury przedstawiony został na rysunku 1.



Rys. 1. Struktura hierarchiczna problemu

Kryteriami wyboru odpowiedniego narzędzia do testowania są następujące pozycje:

- dokumentacja techniczna (K1);
- wspierane platformy programistyczne (K2);
- praca z asynchronicznym działaniem aplikacji (K3);
- szybkość wykonywanych testów (K4);
- łatwość wyszukiwania elementów na stronie (K5).

Tabela 1. Skala Saaty`ego 0

Wartość	**Stopień ważności (ocena obiektu A względem B)**
1	Jednakowa ważność (A jest jednakowo preferowane z B)
3	Nieznaczna ważność (A jest bardziej preferowane niż B)
5	Wyraźna ważność (A jest dużo bardziej preferowane niż B)
7	Bardzo wyraźna ważność (A jest bardzo silnie preferowane w porównaniu z B)
9	Absolutna ważność (A jest ekstremalnie preferowane w porównaniu z B)
(2,4,6,8)	Wartości pośrednie

Kolejnym etapem procesu jest porównywanie parami kryteriów oceny. W tabeli 1 przedstawiono dziewięciostopniową skalę preferencji na podstawie, której dokonano porównań ważności ustalonych kryteriów. Ustalono, że K1 jest najmniej istotne spośród wszystkich kryteriów. Określono, że K2 ma większą wagę od K1, a zdecydowanie mniejsze od pozostałych. Uznano, że K3 ma większą wagę od pozostałych opcji. W dalszej kolejności porównywano K4 z K1 (ocena 6), K2 (ocena 3), K3 (ocena 1/4), oraz K5 (ocena 1/3). Kryterium K5 okazało się mniej ważne tylko od K3 (ocena 1/4). Powyższe oceny pozwoliły na budowę macierzy porównań parami (A), która została przedstawiona w tabeli 2.

W następnym kroku macierz A jest normalizowana w macierz B. Do wykonania tego kroku skorzystano z poniższego wzoru:

$$b_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \qquad (1)$$

gdzie: n- liczba porównywanych elementów.

Tabela 2. Wyznaczenie wag i wskaźników spójności dla kryteriów

Macierz **A**

	K1	K2	K3	K4	K5
K1	1	1/3	1/7	1/6	1/5
K2	3	1	1/7	1/5	1/3
K3	7	6	1,	4	4
K4	6	5	1/4	1	1/3
K5	5	3	1/4	3	1

Macierz **B**

	K1	K2	K3	K4	K5
K1	0,0455	0,0217	0,0800	0,0199	0,0216
K2	0,1364	0,0652	0,0800	0,0239	0,0576
K3	0,3182	0,3913	0,5600	0,4781	0,6906
K4	0,2727	0,3261	0,1400	0,1195	0,0576
K5	0,2273	0,1957	0,1400	0,3586	0,1727

	w	**Aw**	**Aw/w**
K1	**0,0377**	0,1895	5,0210
K2	**0,0726**	0,3651	1,1232
K3	**0,4876**	2,7955	16,5540
K4	**0,1832**	0,9675	4,1567
K5	**0,2188**	1,2968	10,0137
	1,0000		

L	7,3737
CI	0,5934
CR	8,6%

Macierz B umożliwia wyznaczenie wektorów preferencji badanych elementów. W tym celu skorzystano z poniższego wzoru:

$$w_i = \frac{1}{n}\sum_{j=1}^{n} b_{ij} \qquad (2)$$

gdzie : n- liczba porównywanych elementów.

Z powyższych rezultatów widać, że największą wagę otrzymało kryterium K3 (0.4876). Następnie znalazły się K5 (0,2188), K4 (0,1832), K2 (0,0726). Najniższą wagę posiada kryterium K1 (0,0377).

Metoda AHP (ang. Analytic Hierarchy Process) umożliwia weryfikację spójności porównań parami. W tym celu należy obliczyć maksymalną wartość własnej macierzy, która została obliczona za pomocą poniższego wzoru:

$$\lambda_{max} = \frac{1}{n}\sum_{i=1}^{n} \frac{(Aw)_i}{w_i} \qquad (3)$$

gdzie: λ_{max}- maksymalna własność macierzy, n- liczba porównywanych elementów, w_i- wektor preferencji, Aw- wektor preferencji macierzy A

Wartość wskaźnika konsekwentności (CR, ang. consistency ratio) nie przekracza 10 %, co oznacza, że przy porównywaniu ze sobą elementów zachowano logikę ocen.

Kolejnym etapem procesu analitycznej hierarchizacji jest porównanie parami narzędzi Selenium oraz Protractor biorąc pod uwagę każde z kryteriów osobno. Końcowe wyniki oraz obliczone wektory preferencji oznaczone literą v zostały zaprezentowane w tabeli 3. Względem K1 lepsze okazało się narzędzie Protractor (0,67). Biorąc pod uwagę K2 wyższą wagę otrzymało Selenium WebDriver (0,86). W następnych kryteriach narzędziem o wyższej wadzę okazał się Protractor, dla K3 (0,8), dla K4 (0,67) oraz K5 (0,75).

Tabela 3. Ocena narzędzi ze względu na poszczególne kryteria

K1-dokumentacja					
	I	II	macierz B		**v**
I	1,00	1/2	0,33	0,33	**0,33**
II	2,00	1,00	0,67	0,67	**0,67**
K2-wspierane platformy					
	I	II	macierz B		**v**
I	1	6	0,86	0,86	**0,86**
II	1/6	1	0,14	0,14	**0,14**
K3-asynchroniczność					
	I	II	macierz B		**v**
I	1	1/4	0,20	0,20	**0,20**
II	4	1	0,80	0,80	**0,80**
K4-szybkość					
	I	II	macierz B		**v**
I	1	1/2	0,33	0,33	**0,33**
II	2	1	0,67	0,67	**0,67**
K5-wyszukiwanie elementów					
	I	II	macierz B		**v**
I	1	1/3	0,25	0,25	**0,25**
II	3	1	0,75	0,75	**0,75**

4. Wyniki

W tabeli 4 przedstawiano dane związane z wyznaczeniem ostatecznych wag oraz wariantów. Końcowo wyższą ocenę otrzymał Protractor, którego waga wyniosła 0,61. Ponadto suma wag poszczególnych kryteriów oraz suma ostatecznych wag wyniosła 1, co potwierdza poprawność obliczeń.

Tabela 4. Wyznaczenie ostatecznych wag i wariantów

Kryteria	K1	K2	K3	K4	K5	Suma
Wagi	0,0377	0,0726	0,4876	0,1832	0,2188	1,0000
Warianty	Wagi wariantów względem kryteriów					Ostateczne wagi
	K1	K2	K3	K4	K5	
Selenium	0,33	0,86	0,20	0,33	0,25	**0,39**
Protractor	0,67	0,14	0,80	0,67	0,75	**0,61**
Suma	1,00	1,00	1,00	1,00	1,00	1,00

5. Podsumowanie

Biorąc pod uwagę wyniki przeprowadzonego badania oraz wybrane kryteria porównawcze uznano, że lepszym wyborem pod względem testowania aplikacji napisanej przy użyciu frameworka Angular będzie Protractor. Ponadto biorąc pod uwagę każde z kryteriów osobno, wykazano, że we wszystkich zestawieniach nieznacznie bądź wyraźnie lepszym wyborem jest Protractor. Głównym powodem tego jest fakt, że Protractor jest dedykowanym narzędziem do aplikacji Single Page Application dającym większe możliwości oraz posiadającym wbudowane mechanizmy ułatwiające testowanie aplikacji.

Literatura

[1] Jureczko M.: Testowanie oprogramowania, Politechnika Wrocławska, 2011

[2] Test Automation Tool Comparison – HP UFT/QTP vs. Selenium, https://www.aspiresys.com/WhitePapers/QTPvsSelenium.pdf [20.11.2018]

[3] Palmér T., Walter M.: Automated end-to-end user testing on single page web applications

[4] Bajaj H.: Choosing the right automation tool and framework is critical to project success

[5] Cabała P.: Proces analitycznej hierarchizacji w ocenie wariantów rozwiązań projektowych, Quarterly Journal–No, 2018, 24 [20.11.2018]

[6] Umesh N., Saraswat A: Automation Testing: An Introduction to Selenium. International Journal of Computer Applications, 119(3), 2015

[7] Smilgin R.: Zawód tester, Wydawnictwo Naukowe PWN, 2016

[8] Scott E.: SPA design and architecture: understanding single page web applications. Manning Publications Co., 2015

[9] Drążek K., Skublewska-Paszkowska M.: Porównanie wydajności testów automatycznych napisanych w technologii SeleniumWebDriver i HP UFT

Saaty T. L.: How to make a decision: The Analytic Hierarchy Process, European journal of operational research, 1990, 48.1: 9-26

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 32-35
Wysłane: 2018-11-25
Przyjęte: 2018-11-29

Wykorzystanie CPU i GPU do obliczeń w Matlabie

Jarosław Woźniak*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule zostały przedstawione wybrane rozwiązania wykorzystujące procesory CPU oraz procesory graficzne GPU do obliczeń w środowisku Matlab. Porównywano różne metody wykonywania obliczeń na CPU, jak i na GPU. Zostały wskazane różnice, wady, zalety oraz skutki stosowania wybranych sposobów obliczeń.

Słowa kluczowe: CPU; GPU; Matlab

*Autor do korespondencji.

Adres e-mail: jaroslaw.wozniak@pollub.edu.pl

The use of CPU and GPU for calculations in Matlab

Jarosław Woźniak*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents selected solutions using CPU processors and GPUs for calculations in the Matlab environment. Various methods of performing calculations on the CPU as well as on the GPU were compared. Differences, disadvantages, advantages and effects of using selected calculation methods have been indicated.

Keywords: CPU; GPU; Matlab

*Corresponding author.

E-mail address: jaroslaw.wozniak@pollub.edu.pl

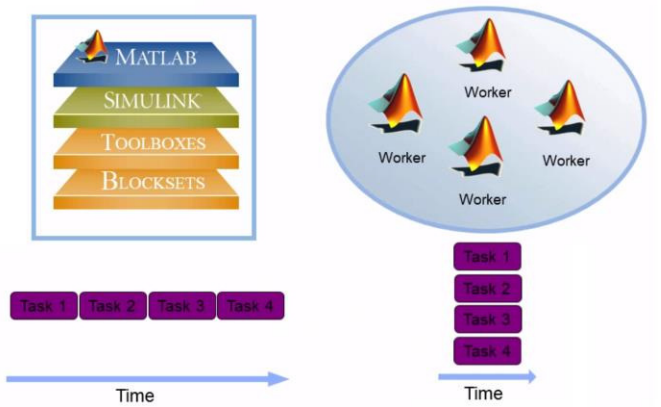
1. Wstęp

Matlab jest to język programowania, przeznaczony do obliczeń technicznych. Jest to również interaktywne środowisko, do którego wbudowano operacje na wektorach, macierzach i tablicach, które tworzą matematyczną podstawę do obliczeń naukowych i technicznych. Dzięki temu możliwe jest szybsze tworzenie i wykorzystywanie algorytmów obliczeniowych w porównaniu do języków tradycyjnych (C, Fortran), gdyż nie jest koniecznym deklarowanie zmiennych ich typów i adresów [1]. W tak popularnym i atrakcyjnym środowisku jakim jest Matlab istotne jest wykorzystanie nowoczesnych technologii takich jak obliczenia równoległe [2]. Kilka wybranych metod implementacji obliczeń równoległych w Matlabie prezentuje niniejszy artykuł.

2. Obliczenia równoległe w Matlabie

Do obliczeń równoległych w Matlabie służy dodatek Parallel Computing Toolbox. Pozwala on na rozwiązanie obliczeń dużej ilości danych przy użyciu wielordzeniowych procesorów, kart graficznych oraz klastrów komputerowych. Wspiera również konstrukcje wysokiego poziomu, równoległe pętle for, specjalne typy tablic i spersonalizowane algorytmy numeryczne. Używając tych funkcjonalności, nie jest konieczne programowania CUDA i MPI. Można skorzystać z tego dodatku wraz z dodatkiem Simulink do równoległego uruchomienia wielu symulacji modeli [3].

Narzędzie to pozwala wykorzystać pełną moc obliczeniową komputerów wielordzeniowych uruchamiając aplikację na tzw. *workerach*, czyli silnikach obliczeniowych Matlaba, które działają lokalnie. Bez zmian kodu można uruchamiać te same aplikacje w klastrze komputerowym lub usługach gridowych (za pomocą *MATLAB Distributed Computing Server*). Aplikacje można uruchamiać interaktywnie lub wsadowo [4].



Rys. 1. Podział na zadania i ich realizacja w czasie dla jednego rdzenia (z lewej) i wielu rdzeni – workerów (z prawej) [5]

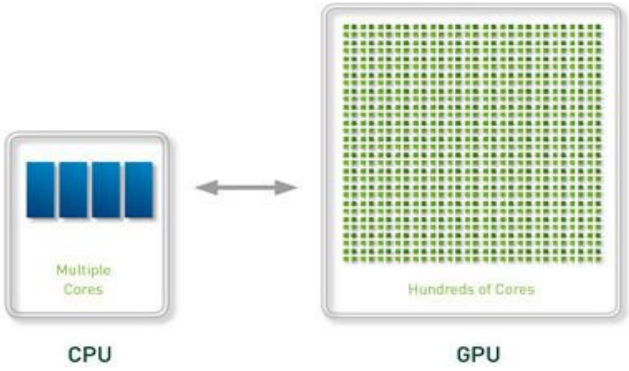
Równoległe przetwarzanie danych pozwala na jednoczesne wykonywanie wielu obliczeń. Duże problemy można często podzielić na mniejsze, które są następnie rozwiązywane w tym samym czasie, co prezentuje powyższy rysunek.

Przedstawia on również rozłożenie zadań w czasie – w przypadku szeregowego wykonywania zadań na jednym rdzeniu oraz ich równoległego wykonania przy wielu workerach – związanych z liczbą rdzeni w procesorze. Główne powody, dla których warto rozważyć przetwarzanie równoległe, to:

1) oszczędność czasu poprzez rozprowadzenie zadań i wykonanie ich równolegle,
2) rozwiązanie problemów z dużą ilością danych poprzez ich dystrybucję,
3) pełniejsze wykorzystanie zasobów komputera i skalowanie do klastrów i chmury obliczeniowej [6, 7].

3. Procesory GPU

Obliczenia na GPU rozpoczęto ponad dekadę temu, kiedy programiści zaczęli wykorzystywać jednostki przetwarzania graficznego (GPU) do zadań obliczeniowych, wymagających duże ilości danych. Na przestrzeni lat naukowcy zajmujący się komputerami wraz z badaczami w dziedzinach takich jak np. obrazowanie medyczne dostrzegli ogromny potencjał wykorzystania procesorów graficznych w obliczeniach wysokiej wydajności (HPC) do ogólnych celów. Termin GPGPU (Komputery ogólnego przeznaczenia GPU) został szybko ustanowiony [8].

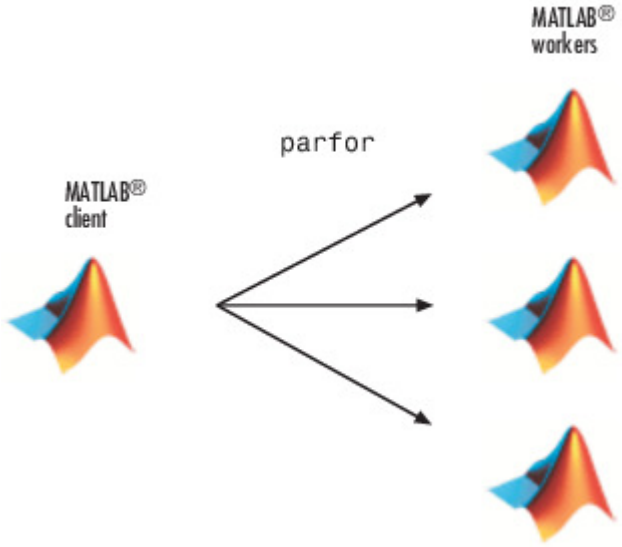


Rys. 2. Porównanie ilości rdzeni w CPU oraz w GPU [8].

Procesory CPU i procesory graficzne GPU przetwarzają zadania na różne sposoby. CPU składa się z zaledwie kilku rdzeni, które są zoptymalizowane do sekwencyjnego przetwarzania szeregowego. Natomiast GPU jest zbudowany z setek lub tysięcy rdzeni, które mogą obsługiwać tysiące wątków jednocześnie [9].

4. Przykład wykorzystania pętli *parfor*

W tym przykładzie pokazane jest działanie wolnej pętli for, a następnie obliczenia są przyspieszone za pomocą pętli parfor, która rozdziela wykonanie iteracji pętli pomiędzy wątki w puli równoległej.



Rys. 3. Zobrazowanie działania pętli parfor [10].

Ten przykład oblicza promień spektralny macierzy i przekształca pętlę for w pętlę parfor. Dodatkowo dodane są polecenia mierzące czas wykonania poleceń, by zmierzyć wynikowe przyspieszenie [11].

1. W edytorze Matlab wprowadzona jest pętla for oraz dodane polecenia tic i toc, aby zmierzyć czas, który upłynął, co prezentuje przykład 1. Wszystkie instrukcje są umieszczone w skrypcie o nazwie *parforTest.m*.

Przykład 1. Pomiar czasu funkcjami tic i toc.

```
tic
n = 200;
A = 500;
a = zeros(n);
for i = 1:n
 a(i) = max(abs(eig(rand(A))));
end
toc
```

2. Uruchomienie skryptu i zanotowanie upływu czasu.

```
>> parforTest

Elapsed time is 27.883551 seconds.
```

3. W skrypcie pętla for zastąpiona jest przez pętlę parfor – przykład 2.

Przykład 2. Wykorzystanie pętli parfor.

```
tic
n = 200;
A = 500;
a = zeros(n);
parfor i = 1:n
 a(i) = max(abs(eig(rand(A))));
end
toc
```

4. W tym momencie zalecane jest włączenie parallel pool z narzędzia Parallel Computing Toolbox, które uruchamia workery i udostępnia im kod (ta procedura nie jest niezbędna, ponieważ kompilator przy sprawdzeniu frazy parfor, automatycznie uruchomi parallel pool i zezwoli wszystkim

workerom na pracę z kodem, lecz czas tego uruchomienia zakłóciłby rzeczywiste różnice pomiarowe dla sprawdzanego przykładu).

5. Uruchomienie skryptu i zanotowanie czasu wykonania.

```
>> parforTest

Elapsed time is 10.102277 seconds.
```

Pętla parfor działa na czterech workerach. Jej wykonanie jest około trzy razy szybsze niż odpowiadającej jej pętli for. Przyspieszenie jest mniejsze niż idealne przyspieszenie dla czterech wątków. Jest to spowodowane równoległym obciążeniem, w tym czasem wymaganym do przesłania danych z klienta do workera i z powrotem. Ten przykład pokazuje dobre przyspieszenie ze stosunkowo małym dodatkowym narzutem i korzyści z konwersji do pętli parfor. Nie wszystkie iteracje pętli mogą zostać przekształcone w szybsze pętle parfor. Jednym z kluczowych wymagań używania pętli parfor jest to, że poszczególne iteracje muszą być od siebie niezależne [11].

5. Porównanie obliczeń na CPU i GPU w Matlabie

W tym punkcie zostanie przedstawiony przykład porównujący obliczenia wykonywane na CPU oraz na GPU w środowisku Matlab. Przedstawione jest wykorzystanie transformaty Fouriera do analizy spektralnej.

W tym przykładzie wykorzystany jest Parallel Computing Toolbox, aby wykonać szybką transformatę Fouriera (FFT) na GPU. Zastosowaniem FFT jest znalezienie składowych częstotliwości sygnału, ukrytego w sygnale czasowym.

Przedstawiony jest przykład wykonujący obliczenia na jednym wątku CPU oraz wykorzystujący wszystkie dostępne wątki procesora. Następnie pokazane jest wykonanie tego zadania na procesorze graficznym GPU. Na końcu przedstawione są wyniki czasu realizacji przykładu i porównanie ich ze sobą [12].

Początkowo zostaje zasymulowany sygnał. Zostają wzięte pod uwagę dane próbki o częstotliwości 1000 Hz i utworzona oś czasu przyjętego przykładu dla dużej liczby próbek. Ponadto utworzone są dwie sinusoidy o częstotliwościach *freq1* i *freq2*. Dla wszystkich trzech sposobów, implementacja pozostaje taka sama, przedstawia ją przykład 3.

Przykład 3. Implementacja utworzenia danych próbki.

```
sampleFreq = 1000;
sampleTime = 1/sampleFreq;
numSamples = 2^23;
freql = 2 * pi * 50;
freq2 = 2 * pi * 120;
```

Sygnał składa się z dwóch komponentów harmonicznych. Pierwszym krokiem jest utworzenie wektora czasu *timeVec* i obliczenie sygnału *signal* jako połączenia dwóch wyżej utworzonych sinusoidalnych częstotliwości.

```
timeVec = (0:numSamples-1) * sampleTime;
signal = sin( freql .* timeVec ) + sin( freq2 .* timeVec );
```

Dodane są losowe zakłócenia dla sygnału.

```
signal = signal + 2 * randn ( size ( timeVec ) );
```

Określenie częstotliwości składowych sygnału umożliwia dyskretna transformata Fouriera w postaci funkcji *Fast Fourier Transform*.

```
transformedSignal = fft( signal );
```

Dodatkowo obliczona jest gęstość widmowa mocy mierząca energię na różnych częstotliwościach.

```
powerSpectrum = transformedSignal .* conj
(transformedSignal) ./ numSamples;
```

W drugim przypadku – przy wykorzystaniu wszystkich wątków, uruchamiana jest równoległa pula w dodatku Parallel Computing Toolbox, pozwalającą na wykorzystanie wszystkich wątków procesora.

Implementacja tego sposobu jest zrealizowana w bloku *spmd* (single program, multiple data). Jego ogólną postać instrukcji prezentuje przykład 4.

Przykład 4. Szkielet bloku *spmd.*

```
spmd
   instrukcje
end
```

W środku bloku są umieszczone instrukcje do wykonania. Zadania są automatycznie rozmieszczone na wszystkie dostępne wątki. Poza tą zmianą, kod nie różni się niczym od tego, który był wykonywany na jednym wątku.

Realizacja zadania dla GPU to przede wszystkim zmiana deklaracji wektora czasu poprzez użycie funkcji *gpuArray* do przesłania danych do procesora GPU w celu dalszego przetwarzania.

```
timeVec = gpuArray( (0:numSamples-1) * sampleTime );
```

W przykładzie 5. jest przedstawiona zawartość skryptu, który jest utworzony do realizacji i prostego porównania wszystkich sprawdzanych rozwiązań wykonywania obliczeń.

Przykład 5. Zawartość wykonanego skryptu.

```
%% DATA INITIALIZATION
sampleFreq = 1000;
sampleTime = 1/sampleFreq;
numSamples = 2^23;
freql   = 2 * pi * 50;
freq2   = 2 * pi * 120;

%% CPU
tic;
timeVec = (0:numSamples-1) * sampleTime;
signal = sin( freql .* timeVec ) + sin( freq2 .* timeVec );
signal = signal + 2 * randn ( size ( timeVec ) );
transformedSignal = fft( signal );
powerSpectrum = transformedSignal .* conj
(transformedSignal) ./ numSamples;
tCPU = toc;
disp(['tCPU = ' num2str(tCPU)]);

%% CPU 4 WORKERS
tic;
spmd
timeVec = (0:numSamples-1) * sampleTime;
```

```
signal = sin( freql .* timeVec ) + sin( freq2 .* timeVec );
signal = signal + 2 * randn ( size ( timeVec ) );
transformedSignal = fft( signal );
powerSpectrum = transformedSignal .* conj
(transformedSignal) ./ numSamples;
end
tCPU_all = toc;
disp(['tCPU_all = ' num2str(tCPU_all)]);

%% GPU
tic;
timeVec = gpuArray( (0:numSamples-1) * sampleTime );
signal  = sin ( freql .* timeVec ) + sin( freq2 .* timeVec );
signal = signal + 2 * randn ( size ( timeVec ) );
transformedSignal = fft ( signal );
powerSpectrum = transformedSignal .* conj
(transformedSignal) ./ numSamples;
tGPU = toc;
disp(['tGPU = ' num2str(tGPU)]);

%% SUMMARY
tDIFF1 = tCPU_all / tCPU;
disp(['tCPU_all/tCPU = ' num2str(tDIFF1)]);
tDIFF2 = tCPU / tGPU;
disp(['tCPU/tGPU = ' num2str(tDIFF2)]);
tDIFF3 = tCPU_all / tGPU;
disp(['tCPU_all/tGPU = ' num2str(tDIFF3)]);
```

Skrypt wykorzystywał polecenia tic oraz toc, dzięki którym zostały zapisane czasy wykonania operacji dla procesora CPU i GPU. Wartości zwrócone po wykonaniu skryptu zostały przedstawione w przykładzie 6.

Przykład 6. Otrzymane wyniki.

```
tCPU = 1.3864
tCPU_all = 4.4697
tGPU = 0.36826
tCPU_all/tCPU = 3.2239
tCPU/tGPU = 3.7648
tCPU_all/tGPU = 12.1372
```

Przeprowadzenie stu symulacji pozwoliło otrzymać średnio około czterokrotnie krótsze czasy wykonania operacji dla procesora graficznego względem sposobu z jednym wątkiem oraz ponad 12-krotne przyspieszenie względem uruchomienia rozwiązania z blokiem *spmd*. które jest ponad trzykrotnie wolniejsze od pracującego na jednym wątku.

6. Wnioski

Prowadząc badania na czterordzeniowym procesorze, w próbie jego pełnego wykorzystania – zastosowaniu pętli *parfor* odnotowano około trzykrotnie lepsze wyniki czasowe przeprowadzonych działań. Wyniki nie były idealnie czterokrotnie lepsze, czego można się było teoretycznie spodziewać. Przygotowanie bloku równoległego, przesyłanie danych z pamięci RAM do pamięci karty graficznej i synchronizacja wątków wprowadza dodatkowy czas potrzebny do wykonania części równoległej kodu.

Rozwiązanie z blokiem *spmd* jest najwolniejsze dla badanego przykładu, ponieważ więcej czasu zajmuje komunikacja między wątkami niż same obliczenia.

Procesory graficzne nie zastępują architektury CPU. Są raczej potężnymi akceleratorami dla istniejącej infrastruktury. Procesory przyspieszane przez GPU odciążają część aplikacji intensywnie wykorzystującą obliczenia na GPU, podczas gdy pozostała część kodu nadal działa na procesorze. Z perspektywy użytkownika aplikacje działają znacznie szybciej. Podczas gdy przetwarzanie ogólnego przeznaczenia nadal jest domeną CPU, procesory graficzne są szkieletem sprzętowym prawie wszystkich intensywnych aplikacji obliczeniowych [13].

Jednak procesory są bardziej elastyczne niż GPU, mają większy zestaw instrukcji, dzięki czemu mogą wykonywać szerszy zakres zadań. CPU pracuje również z wyższymi maksymalnymi prędkościami zegara i jest w stanie zarządzać wejściami i wyjściami wszystkich komponentów komputera.

Literatura

[1] MATLAB Product Description - MathWorks Documentation, *https://www.mathworks.com/help/matlab/learn_matlab/product-description.html* [01.08.2018]

[2] K. Banasiak, Algorytmizacja i programowanie w MATLABIE, Wydawnictwo BTC, 2017.

[3] Parallel Computing Toolbox - Documentation, *https://www.mathworks.com/help/distcomp/* [01.08.2018].

[4] J. W. Sut, Y. Kim, MATLAB and Parallel Computing Toolbox, 2014, 99-125.

[5] Obliczenia równoległe w środowisku Matlab - MathWorks Video and Webinars, *https://www.mathworks.com/videos/parallel-computing-in-matlab-116769.html* [01.08.2018].

[6] B. Mrozek, „Obliczenia równoległe w Matlab-ie," Pomiary Automatyka Robotyka, tom R. 15, nr 2, pp. 285-294, 2011.

[7] I. Azzini, R. Muresano, M. Ratto, Dragonfly: A multi-platform parallel toolbox for MATLAB/Octave, 2018, 21-42.

[8] What is GPU computing?, *https://www.boston.co.uk/info/nvidia-kepler/what-is-gpu-computing.aspx* [01.08.2018].

[9] M. Sourouri, J. Langguth, F. Spiga, S. B. Baden. X. Cai, CPU+GPU Programming of Stencil Computations for Resource-Efficient Use of GPU Clusters, 2015, 17-26.

[10] Using parfor-loop - MathWorks Documentation,, *https://www.mathworks.com/help/distcomp/interactively-run-a-loop-in-parallel.html#responsive_offcanvas* [01.08.2018].

[11] What Is Parallel Computing?, *https://www.mathworks.com/help/distcomp/what-is-parallel-computing.html* [01.08.2018].

[12] Using FFT on the GPU for Spectral Analysis MathWorks – Documentation, *https://www.mathworks.com/help/distcomp/examples/using-fft-on-the-gpu-for-spectral-analysis.html* [01.08.2018].

[13] H. Anzt, M. Gates, J. Dongarra, M. Kreutzer, G. Welling, M. Kohler, Preconditioned Krylov solvers on GPUs, 2017, 32-44.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 36-43
Wysłane: 2018-11-22
Przyjęte: 2018-12-03

Analiza porównawcza gogli do VR

Łukasz Pełka*, Łukasz Podstawka*, Tomasz Szymczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł dotyczy porównania dwóch typów okularów do wirtualnej rzeczywistości – z wbudowanym ekranem oraz różnych konstrukcji okularów, z wyjmowanym ekranem, którym najczęściej jest smartfon. Opracowano scenariusze badawczo – testowe, których celem było porównanie tych technologii w zakresie jakości wyświetlanego obrazu, ergonomii urządzeń oraz wygody ich użytkowania.

Słowa kluczowe: wirtualna rzeczywistość; gogle projekcyjne; HMD; Oculus Rift

*Autor do korespondencji.

Adresy e-mail: lukasz.pelka94@wp.pl, wookee94@gmail.com

Comparative analysis of VR goggles

Łukasz Pełka*, Łukasz Podstawka*, Tomasz Szymczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Article deals with comparison of two goggle types for virtual reality – with built-in screen and a variety of constructions with removable screen, which generally was a smartphone. For this reason research and test scenarios has been made. Its purpose was to compare those technologies in terms of quality of displaying image, ergonomics and comfort of usage.

Keywords: virtual reality; projection goggles; HMD; Oculus Rift

*Corresponding author.

E-mail addresses: lukasz.pelka94@wp.pl, wookee94@gmail.com

1. Wstęp

Wirtualną rzeczywistość można w skrócie opisać jako obraz imitujący rzeczywistość- świat realny lub też fikcyjny, wygenerowany za pomocą technologii informatycznych. Powszechnie uważa się, że twórcą pojęcia wirtualnej rzeczywistości jest amerykański informatyk Jaron Zepel Lanier.

Obecnie wirtualna rzeczywistość uzyskiwana jest najczęściej za pomocą generowanych obrazów i dźwięków. Obrazy możemy obserwować za sprawą różnych ekranów, najczęściej komputerowych. Mogą być one wielkopowierzchniowe a także miniaturowe, umieszczone w specjalnych goglach (np. Oculus Rift DK2).

Wirtualna rzeczywistość ma szerokie zastosowanie w dziedzinach użytkowych i rozrywkowych. Dziedziny użytkowe wykorzystują środowisko podobne do rzeczywistego świata. Tworzone są różnego rodzaju symulatory dla pilotów, astronautów, żołnierzy, lekarzy. Dzięki temu można w kontrolowanych warunkach przeprowadzać treningi i testować nowe rozwiązania [1]. Z kolei dziedziny rozrywkowe dają większe pole do wykorzystania wyobraźni. Generowane są różne światy rodem z filmów science-fiction, które wnoszą rozrywkę na nowy, dotąd nieosiągalny poziom. Ponadto popularne stało się ostatnio wykorzystywanie wirtualnej rzeczywistości do tworzenia tak zwanych „wirtualnych spacerów". Za pomocą specjalnych kamer skanuje się np. wnętrza budynków, następnie generuje się model, który możemy oglądać za pomocą specjalnych gogli lub smartfonów.

W czerwcu 2012 roku firma Oculus ogłosiła opracowanie Oculus Rift, czyli specjalnych gogli, z wbudowanym wyświetlaczem, umożliwiających rozrywkę w wirtualnym świecie. W celu sfinansowania produkcji skorzystano z portalu Kickstarter. W lipcu 2014 do sprzedaży trafiła druga wersja deweloperska gogli- Oculus Rift DK2, którą posiada Politechnika Lubelska.

2. Okulary projekcyjne

Wirtualna rzeczywistość stała się powszechną technologią w ciągu ostatnich kilku lat. Jednak jej historia sięga o wiele dalej. W roku 1935 wydano książkę zatytułowaną *Pygamalion's Spectacles* autorstwa Stanleya G. Weinbauma, która zawierała prawdopodobnie pierwszą wzmiankę o goglach do wirtualnej rzeczywistości. Cały system pozwalał doświadczać obrazu, smaku, zapachu, dotyku obiektów w wirtualnej rzeczywistości. Ponadto użytkownik mógł komunikować się z postaciami z przedstawionego świata [2].

2.1. Okulary historyczne

W latach 50. XX wieku Morton Heilig wydał artykuł zatytułowany *Experience Theatre,* w którym przedstawił swoją wizję maszyny pozwalającej na wyświetlanie przed oczami widza obrazów, co dawałoby złudzenie przebywania w innej rzeczywistości. Prototyp tej maszyny został

zaprezentowany w roku 1962. Heilig nazwał ją *Sensorama* [3]. Było to mechaniczne urządzenie zawierające stereoskopowy, kolorowy wyświetlacz, krzesło oraz wiatraki. *Sensorama* wydzielała także zapachy oraz posiadała system dźwiękowy stereo.

Rys. 1. Widok urządzenia- Sensorama [4]

W roku 1966 technologię wirtualnej rzeczywistości wprowadzono do armii, za sprawą Thomasa A. Furnessa III, który opracował symulator lotu. Chciał on by piloci mogli trenować w bezpiecznych warunkach.

W roku 1968 Ivan Sutheraland zaprezentował światu *Ultimate Display* [4]. To urządzenie można uznać za pierwsze w historii urządzenie HMD (head-mounted display), czyli wyświetlacz nagłowny. Sprzęt ten był jednak bardzo ciężki. Dlatego też został on podwieszony do sufitu. *Ultimate Display* wyświetlał użytkownikowi proste szkielety obiektów.

Rys. 2. Widok urządzenia Ultimate Display [5]

Kolejne lata to ciągły rozwój urządzeń wirtualnej rzeczywistości bardziej przypominających obecne rozwiązania. W latach osiemdziesiątych VPL Research opracowuje DataGlove oraz EyePhone

Rys. 3. Widok urządzeń- DataGlove oraz EyePhone (VPL Research) [6]

Sega w roku 1994 prezentuje urządzenie *Sega VR-1*. Skorzystać z niego można było wyłącznie w salonach gier sieci *SegaWorld* [7]. W specjalnych goglach wyświetlano grafikę 3D. Ponadto sprzęt umożliwiał śledzenie ruchów głowy. *Sega VR-1* nie odniosło sukcesu, ze względu na niezbyt dopracowaną oprawę graficzną [3].

W roku 1995 zaprezentowano także inne urządzenie do VR- *VFX1*. Były to gogle połączone ze słuchawkami [7]. Prace nad goglami zaczęły się już 5 lat wcześniej. Były one wyposażone w 2 wyświetlacze o rozdzielczości 263×230 px każdy, które mogły wyświetlać do 256 kolorów. Ruchy głowy były śledzone za pomocą zewnętrznych czujników.

Rys. 4. Wygląd urządzenia VFX1 [3]

2.2. Okulary współczesne

W 2012 roku firma Oculus VR zbiera pieniądze w crowdfuncingowej kampanii na stronie Kickstarter, by stworzyć okulary do wirtualnej rzeczywistości eliminujące dotychczas problemy związane z urządzeniami tego typu – nudności i bóle głowy. Ciągle udoskonaląc swoje rozwiązania prezentuje wersję deweloperską gogli – Development Kit 1. Po dwóch latach zaczyna się sprzedaż drugiej wersji deweloperskiej oznaczonej symbolem DK2 [8]. W ciągu 7 miesięcy od uruchomienia sprzedaży ogłoszono, że nabywców znalazło 100 000 urządzeń [9].

Rys. 5. Widok urządzenia - Oculus Rift DK2 [10].

W roku 2016 na rynek trafia pierwsza wersja konsumencka (CV1), charakteryzująca się znacznie poprawionymi parametrami względem wszystkich poprzednich odsłon.

Rys. 6. Widok urządzenia - Oculus Rift CV1 [11].

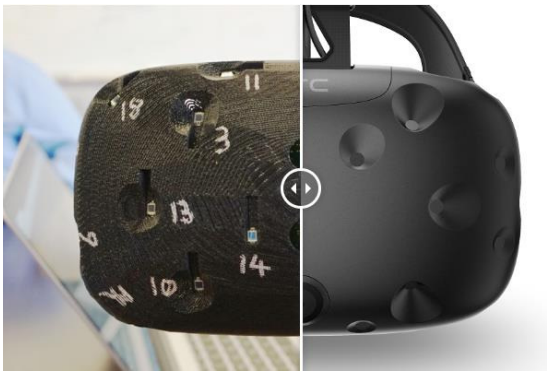
Za następcę Oculus Rift można uznać Oculus Go zapowiedziany w październiku 2017 roku. Firma Oculus rozpoczęła produkcję tego urządzenia we współpracy z chińskimi firmami Qualcomm oraz Xiaomi. Oculus Go w odróżnieniu od swoich poprzedników jest samodzielną platformą, niezależną od komputera, przez co nie wymaga zastosowania ograniczających przewodów, jednak tym samym pozbawionym jest zewnętrznego zasilania, co wymusza zastosowanie wbudowanej baterii, co z kolei ogranicza czas pracy [11].

W poniższej tabeli przedstawione zostały główne różnice pomiędzy czterema wersjami gogli Oculus udostępnionymi do ogólnej sprzedaży [12, 13, 14, 15, 16].

Tabela 1. Porównanie wersji Oculus DK1, DK2, CV1 oraz Go

Wersja	**DK1**	**DK2**	**CV1**	**Go**
Typ ekranu	LED	OLED	OLED	LCD
Rozdzielczość [px]	1280×800	1920×1080	2160×1200	2560×1440
Częstotliwosć odświeżania [Hz]	60	60, 72, 75	90	60, 72
Waga [g]	380	440	470	470
Pole widzenia [°]	90	90	110	ok 101
Cena w dniu premiery [$]	300	350	400	200

Głównym konkurentem Oculusa w dziedzinie gogli VR podłączanych do komputera jest HTC Vive. W 2012 roku, kiedy temat wirtualnej rzeczywistości stawał się coraz bardziej popularny firmy Valve oraz HTC zaczęły interesować się tym tematem. Obie firmy próbowały swoich sił osobno, jednak dopiero wspólna praca nad projektem dała owocne skutki. 5 kwietnia 2016 roku premierę ma urządzenie HTC Vive. Gogle stosują podobną metodę śledzenia ruchów głowy względem konkurencji, jednak z zastosowaniem promieni laserów obok zwykłych diod LED. Ponadto do zestawu dołączane są dwa kontrolery, których położenie wykrywane jest na tej samej zasadzie działania [17].



Rys. 7. HTC Vive wewnątrz i na zewnątrz [18]

Rys. 8. Wnętrze i wygląd kontrolera HTC Vive [19]

8 stycznia 2018 roku, HTC ujawniło ulepszoną wersję Vive, znaną jako HTC Vive Pro. Zawiera ona wyświetlacze o wyższej rozdzielczości, dołączane słuchawki, mikrofon stosowany do aktywnego tłumienia dźwięków z zewnątrz, odświeżony wygląd z bardziej zbalansowaną formą oraz niższą wagą. Zamiast połączenia portem USB-A używa USB-C [20].

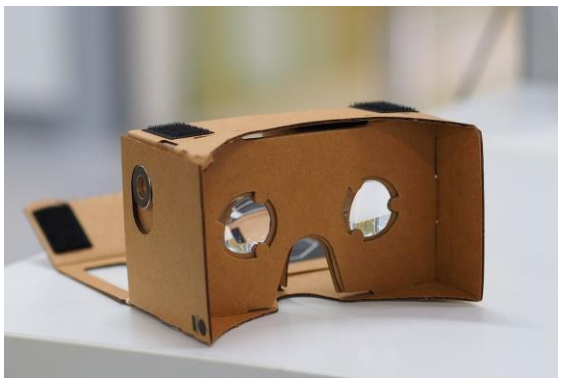
W poniższej tabeli przedstawione zostały główne różnice pomiędzy goglami Oculus Rift CV1, HTC Vive a HTC Vive Pro [14, 24, 26]

Tabela 2. Porównanie Oculus Rift CV1, HTC Vive i HTC Vive Pro

Wersja	**Oculus Rift CV1**	**HTC Vive**	**HTC Vive Pro**
Typ ekranu	OLED	OLED	AMOLED
Rozdzielczość [px]	2160×1200	2160×1200	2880×1600
Częstotliwosć odświeżania [Hz]	90	90	90
Waga [g]	470	555	470
Pole widzenia [°]	110	110	110
Cena w dniu premiery [$]	400	800	1100

Przedstawione wyżej rozwiązania są niewątpliwie efektywne – wyświetlają obraz z dużą płynnością, w wysokiej rozdzielczości, pozwalają na dokładne odwzorowanie ruchów głowy w wirtualnym świecie. Jednak urządzenia te mają swoje wady. Każde z nich musi być podłączone kilkoma przewodami do komputera, który z kolei musi dysponować bardzo dużą mocą obliczeniową, by zdołać wygenerować 90 klatek obrazu na sekundę w tak dużych rozdzielczościach. To wszystko generuje wysokie koszty, a nie każdy użytkownik chcący poznać świat VR może pozwolić sobie na wydatek rzędu kilku lub kilkunastu tysięcy złotych na komputer i odpowiednie gogle. Alternatywą dla tego rozwiązania jest zakup telefonu z wyższej półki (nawet za niecały tysiąc złotych) oraz odpowiedniej ramki, do której można włożyć telefon i cieszyć się wirtualną rzeczywistością za dużo mniejszą kwotę, a ponadto telefon może służyć nie tylko temu jednemu, konkretnemu celowi.

Najprostsza a zarazem najtańsza z tych ramek – Google Cardboard została zaprezentowana podczas konferencji Google I/O w czerwcu 2014 roku. Był to debiut tego typu akcesorium dla telefonów z systemem Android. Projekt powstał, by pokazać, że nie trzeba przepłacać za Oculus Rift, by mieć dostęp do wirtualnej rzeczywistości. Cały zestaw opiera się wyłącznie na kawałku tektury połączonym w odpowiedni sposób oraz parze soczewek, gdyż wszystkie obliczenia i wyświetlanie obrazu dokonywane są za pomocą umieszczonego wewnątrz ramki telefonu. Cena takiego rozwiązania zaczyna się od około 15$, niecałe 55 zł [18]



Rys. 9. Google Cardboard [19]

Różne firmy podjęły pomysł Google i zaczęły produkować własne wersje ramek do telefonów. Jednym z najpopularniejszych jest Samsung Gear VR, wypuszczony do sprzedaży w listopadzie 2015 roku. Działa on z wybranymi modelami smartfonów Samsunga, począwszy od modelu Galaxy S6, przez wersje S7, S8 do S9 włącznie [28].

Rys. 10. Samsung Gear VR [29]

W obecnych czasach na rynku istnieje wiele firm produkujących różne ramki umożliwiające odtwarzanie obrazu wirtualnej rzeczywistości na ekranach smartfonów. Głównymi różnicami w ich konstrukcji są materiały, z których wykonano soczewki, możliwość i sposób dostosowania ostrości widzenia dla osób z wadą wzroku oraz proporcje obrazu, który można wyświetlać. Niektóre z ramek pozwalają na wyświetlanie obrazu w formacie 4:3, inne dają obraz panoramiczny. Różne są również ceny, od najtańszych, za 10 zł, do tych droższych z ceną sięgającą nawet kilkuset dolarów.

3. Analiza porównawcza

Badanie przeprowadzone na każdym z 16 respondentów zostało podzielone na 4 etapy, w trzech pierwszych prezentowane były inne treści, zawarte w odrębnych scenach, przygotowanych na potrzeby badania. Czwarty etap polegał na odpowiedzi na pytania o ogólne wrażenia z użytkowania urządzeń.

Badanie w etapie pierwszym polegało na prezentacji modeli 3D zwierząt. Obiekty te obracały się wokół własnej osi. Zaprezentowano 8 modeli zwierząt rozmieszczonych w przestrzeni. Autorzy ułożyli modele zwierząt w kolejności od najmniejszego, do największego. Pierwszym był szczur, następnym kura, kot, koza, wilk, krowa, żyrafa a ostatnim mamut. Ich odległości od wzroku obserwatora wynosiły od 11,8 metra do 217,1 metra. Celem eksperymentu było wykazanie, czy użytkownik przy ograniczonej rozdzielczości ekranu był w stanie dostrzec modele dobrze znanych zwierząt. Wynikiem badania jest procentowa ilość poprawnie rozpoznanych zwierząt.

Badanie w etapie drugim polegało na prezentacji użytkownikowi dwóch tablic, pokrytych regularnymi pionowymi i poziomymi liniami, dalej zwanymi siatką. Każda z tablic ma taki sam rozmiar. Odpowiada on w rzeczywistości 20 m szerokości i 10 m wysokości. Znajdują się one w odległości 15,5 m oraz 37,5 m od obserwatora i są ustawione pod kątem prostym. Na planszach znajdują się siatki złożone z ciemnych linii o rzeczywistej szerokości 19,6 cm. Respondentom w tym etapie badania zadano następujące pytania: Czy widzisz siatkę na bliższej oraz dalszej tablicy? Czy widzisz efekt mory na bliższej oraz dalszej tablicy? Wynikiem badania jest procentowa liczba osób, które

widziały/nie widziały siatki oraz procentowa liczba osób, które widziały/nie widziały mory.

Etap trzeci badania polegał na zaprezentowaniu respondentom trzech tablic zawierających znaki alfanumeryczne – cyfry, litery oraz znaki specjalne, takie jak znak zapytania, wykrzyknik czy podkreślnik. Podobnie jak w etapie drugim, tutaj tablice również miały rozmiar wynoszący w rzeczywistości 20 m szerokości i 8 m wysokości. Tablice znajdowały się w odległości odpowiednio 36 m, 90,1 m oraz 155,2 m od oczu obserwatora. Wynikiem badania jest procentowa liczba poprawnie odczytanych znaków dla każdej z tablic.

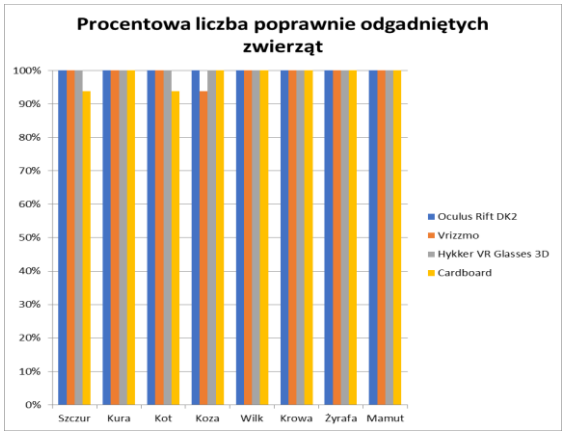
Etap czwarty badania polegał na zadaniu użytkownikom urządzeń szeregu pytań odnośnie ich odczuć z korzystania z okularów. Ocena polegała na przyznaniu goglom punktów z zakresu 1 – 5, gdzie 1 to najniższa ocena, a 5 to najwyższa ocena, dla następujących kategorii:

1. Waga urządzenia
2. Wygoda umocowania na głowie
3. Mobilność
4. Ogólna jakość wyświetlanego obrazu
5. Ostrość krawędzi wyświetlanego obrazu
6. Płynność obrazu
7. Zasięg dokładnego widzenia wirtualnego świata
8. Poziom zmęczenia oczu
9. Ogólne odczucia z użytkowania
10. Ogólna ocena dla urządzenia

Każdy z respondentów przeszedł przez cztery etapy. Po każdym etapie badani przydzielali punkty dla każdej z kategorii w skali 1 – 5 (1 to najniższa ocena, 5 to najwyższa) bądź odpowiadali na zadane pytania używając słów tak lub nie.

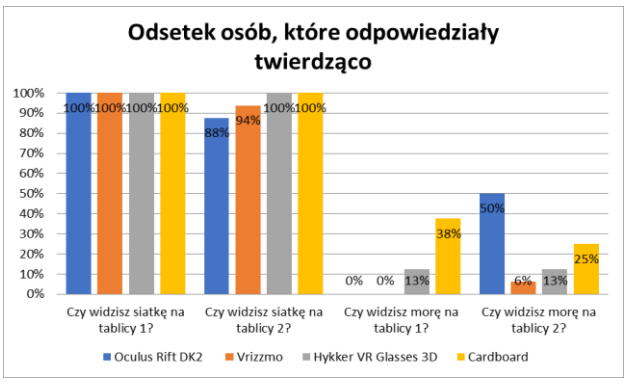
Do celów badawczych starano się wybrać różne osoby pod względem wieku, płci czy znajomości technologii wirtualnej rzeczywistości. Wiek badanych wahał się w przedziale od 24 do 80 lat, jego średnia wyniosła 35,19 lat. Wśród badanych znalazło się 75 % mężczyzn i 25 % kobiet. 44 % badanych stwierdziło, że nigdy nie miało styczności z wirtualną rzeczywistością. 56 % pozostałych osób zapytano, jak długo używały VR.

W etapie pierwszym badania znaczna większość respondentów poprawnie zidentyfikowała wyświetlane modele zwierząt. Wśród wszystkich badanych na wszystkich urządzeniach zdarzyły się jedynie trzy błędnie odczytane obiekty, gdzie jeden z badanych używając kartonowych okularów pomylił kota z pumą oraz kozę z osłem; drugi natomiast używając okularów Vrizzmo uznał model szczura za model wiewiórki. Wszystkie błędy w odczytywaniu zwierząt wystąpiły w trzech z czterech najbliżej położonych modeli.



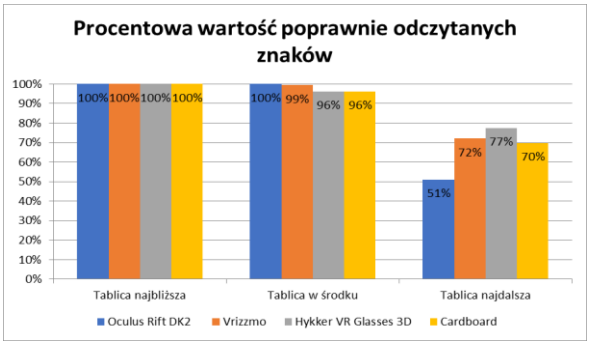
Rys. 11. Prezentacja procentowego rozkładu poprawnie odczytanych modeli zwierząt w pierwszym etapie badania

Drugim etapem badania było wyświetlenie plansz pokrytych siatkami i zbadanie, czy występuje efekt mory. Wszyscy z badanych widzieli siatkę na najbliższej tablicy, jednak na tej dalszej, już tylko w przypadku okularów Hykker oraz Cardboard wszyscy użytkownicy widzieli siatkę. Należy zwrócić uwagę, że w przypadku dwóch ostatnich zapytań niższy odsetek oznacza lepszą ocenę.



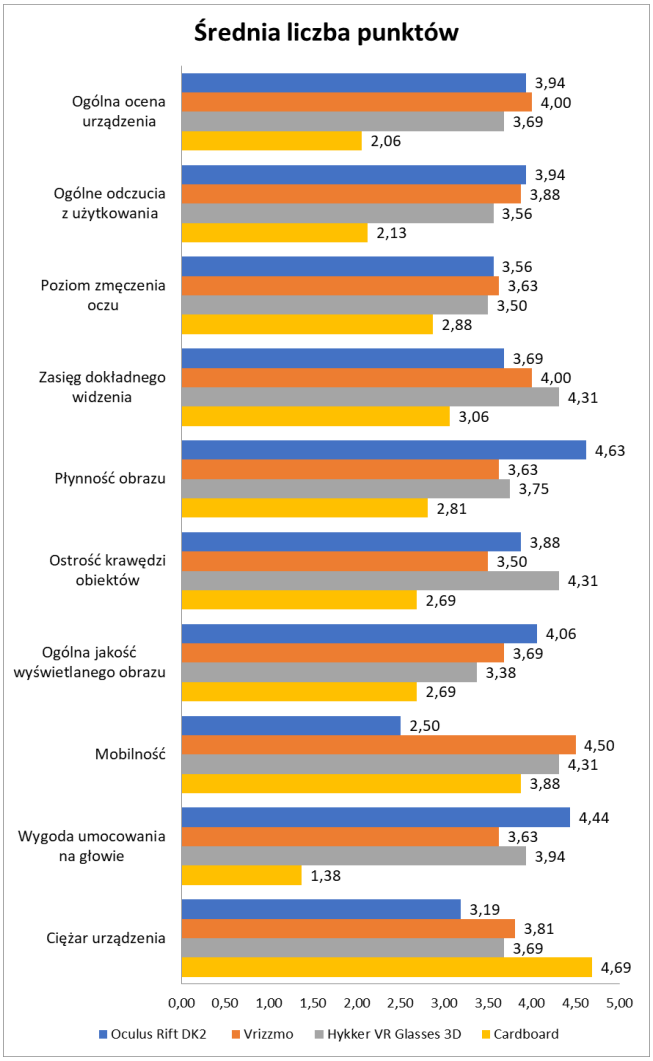
Rys. 12. Prezentacja procentowego rozkładu osób, które odpowiedziały twierdząco na poszczególne pytania dotyczące wyświetlanych tablic

Etap trzeci badania to odczytywanie znaków alfanumerycznych z trzech tablic położonych w różnych odległościach od obserwatora. Bliższa z tablic, znajdująca się 15,5 m od oczu badanego nie była problemem dla respondentów. Dla wszystkich testowanych urządzeń ilość poprawnie odczytanych znaków wyniosła 100 %. Druga tablica odległa o 37,5 m okazała się możliwa do odczytania dla wszystkich badanych jedynie na goglach Oculus Rift, jednak na wszystkich urządzeniach liczba poprawnie odczytanych znaków wyniosła minimum 96%. Trzecia i tym samym najdalsza tablica okazała się trudnością dla wszystkich urządzeń, jednak największą dla Oculusa Rift, który uzyskał najniższy wynik w tej grupie – 51 % poprawnie odczytanych znaków. Znacznie wyżej bo na poziomie 70 % znalazły się dotychczas najgorzej wypadające okulary Cardboard. Najlepiej poradziły sobie natomiast gogle Hykker VR Glasses 3D z wynikiem 77 % poprawnie odczytanych znaków.



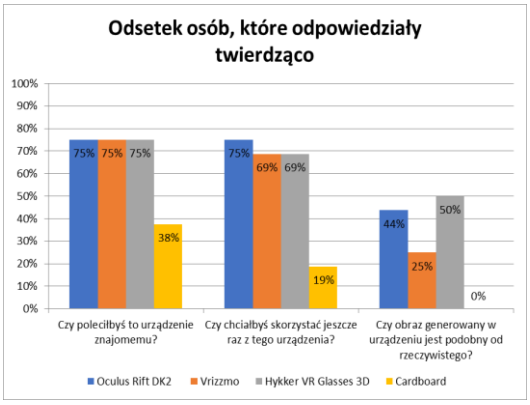
Rys. 13. Prezentacja procentowego rozkładu poprawnie odczytanych znaków z poszczególnych tablic

Etapem czwartym było opisanie swoich wrażeń i odczuć z użytkowania testowanych urządzeń. Zadano 13 pytań, z czego w pierwszych 10 badani oceniali w skali punktowej od 1 do 5 (gdzie 1 to najgorsza ocena, a 5 to najlepsza), natomiast na trzy pozostałe pytania odpowiadali tak lub nie. Średnie wyniki punktowe po zsumowaniu wszystkich odpowiedzi, prezentuje poniższy wykres.



Rys. 14. Prezentacja średniej liczby punktów otrzymanych w ankiecie dla danych urządzeń

Na kolejne trzy pytania dopuszczone odpowiedzi brzmiały tak lub nie. Tutaj również im wyższy wynik, tym lepiej oceniane urządzenie.

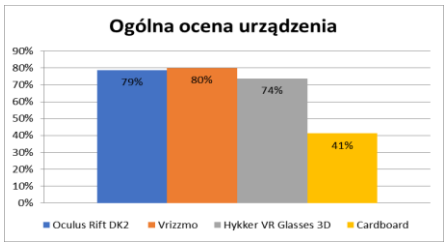


Rys. 15. Prezentacja procentowego rozkładu osób, które odpowiedziały twierdząco na poszczególne pytania dotyczące urządzeń

Wyznaczenie najlepszych okularów do wirtualnej rzeczywistości nie jest łatwe. Konieczne było zadanie wielu pytań i przebadanie dużej liczby osób w tym celu. Mimo to każde urządzenie było najlepsze chociaż w jednej rozpatrywanej kategorii, co jeszcze bardziej utrudnia znalezienie najlepszego urządzenia. Autorzy opracowali cztery metody ostatecznego porównania ze sobą okularów:

- Porównanie ogólnej oceny urządzenia, nadanej przez badanych
- Porównanie liczby kryteriów, w których urządzenie było najlepsze
- Przyznanie punktów za zajęte miejsca w każdej z kategorii i wyciągnięcie średniej z tych punktów
- Przyznanie punktów za zajęte miejsca w każdej z kategorii i wyliczenie mediany.

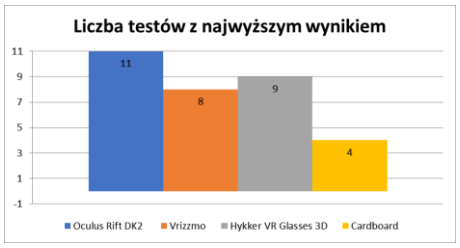
Według pierwszej metody najlepszym urządzeniem są okulary Vrizzmo, jednak ten sposób może nie być do końca wiarygodny. Okulary te były najlepsze jedynie w dwóch kategoriach spośród 8 wcześniej badanych, ponadto na przykładzie okularów Cardboard – ich ostateczna ocena niekoniecznie musi równać się z wcześniej uzyskanymi wynikami. Okulary Vrizzmo uzyskały w 7 z 8 poprzednich testów noty niższe lub równe ogólnej ocenie.



Rys. 16. Prezentacja procentowego rozkładu maksymalnej ilości punktów w ogólnej ocenie urządzeń

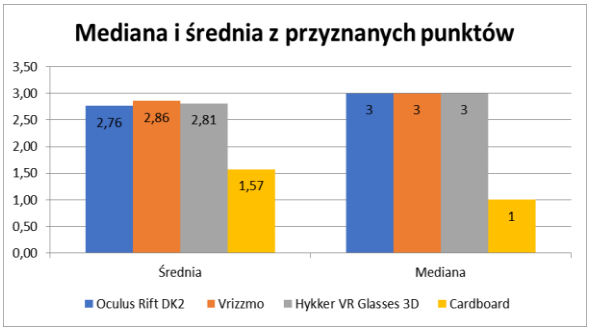
Drugi sposób oceniania wskazuje, że najlepsze są gogle Oculus Rift. Za najwyższy wynik w każdym z pomiarów, w każdym z etapów, przyznawany był jeden punkt. Jeśli najwyższy wynik osiągnęło jednocześnie więcej niż jedno

urządzenie, każde z nich dostawało punkt. Następstwem tego jest fakt, że po przeprowadzeniu 21 różnego rodzaju testów, ilość punktów na wykresie nie sumuje się do 21.



Rys. 17. Prezentacja liczby testów, w których dane urządzenie osiągnęło najwyższy wynik

Trzecia i czwarta metoda wymaga odrobiny przygotowań i wyliczeń. Zakładając, że za najwyższy wynik w teście przyznane zostają 4 punkty, za drugi najwyższy 3, następnie 2, a za najniższy 1 punkt, można przypisać ilość punktów do każdego z testów, dla każdego z urządzeń. W przypadku, gdy więcej niż jedno urządzenie zajmuje któreś z miejsc, przyznawana jest mu średnia z ilości dostępnych punktów dla tego miejsca, np. dwa urządzenia z najwyższym wynikiem otrzymują średnią z dwóch najwyższych not punktowych – 3,5 punktu. Po dokonaniu wszystkich wyliczeń, wyniki prezentują się następująco:



Rys. 18. Mediana i średnia z przyznanych punktów

W trzecim sposobie, podobnie jak w pierwszym, najlepszymi okularami okazują się być Vrizzmo, osiągając 2,86 punktu na 4 możliwe (71 % punktów). Blisko za nimi znalazły się okulary Hykker z wynikiem 2,81 punktu (70 %), dalej Oculus Rift osiągając średnią 2,76 punktu (69 %). Jak widać różnica między najwyższą ilością punktów a drugą najniższą wynosi tylko 2 punkty procentowe, co należy uznać za bardzo zbliżony wynik. Ostatnie miejsce zajmują okulary Cardboard z wynikiem 1,57 z 4 punktów (39 %).

W przypadku mediany zbliżenie wyników jest jeszcze większe. Oculus Rift, Vrizzmo oraz Hykker osiągnęły medianę na poziomie 3 punktów, natomiast Cardboard z medianą wynoszącą 1 znalazł się na końcu.

4. Wnioski

Na podstawie przeprowadzonych badań można stwierdzić, że użytkownicy cenią sobie mobilność urządzeń. W tym aspekcie Oculus Rift DK2 zdecydowanie przegrał, ze względu na kable oraz komputer niezbędne do poprawnego działania. Pozostałe okulary do poprawnego działania wymagają jedynie smartfona. Dzięki temu użytkownik posiada nieograniczony zakres ruchów i może dowolnie się przemieszczać oraz zabrać urządzenie w każdą podróż. Kolejnym niezwykle ważnym aspektem jest wygoda umocowania na głowie. Daje się zauważyć, że użytkownicy cenią sobie ten aspekt, gdyż pozwala on cieszyć się wirtualnym światem, bez użycia rąk. Dlatego też w tej dziedzinie zdecydowanie przegrywa Cardboard, ze względu na brak mocowania- urządzenie należy trzymać w dłoniach.

Według badanych Hykker VR Glasses 3D wyświetla obraz, w szczególności krawędzie obiektów, najostrzej oraz pozwala na dostrzeżenie szczegółów znajdujących się najdalej.

Badani wskazali urządzenie Oculus Rift DK2 jako to, które wyświetla obraz najpłynniej. Komputer, do którego podłączono Oculus posiada kartę graficzną NVIDIA GeForce GTX 1080, czterordzeniowy procesor Intel Core i7 oraz 16 GB pamięci RAM. Smartfon użyty do wyświetlenia obrazów w pozostałych goglach to LG G6 wyposażony w czterordzeniowy procesor Qualcomm Snapdragon 821, 4 GB pamięci RAM oraz układ graficzny Adreno 530. W momencie przeprowadzenia badania smartfon ten oferuje czołowe osiągi w swojej kategorii. Można zatem stwierdzić, że urządzenia mobilne nie są jeszcze w stanie dorównać pod względem płynności działania komputerom. Ciekawą kwestią są znaczne różnice w ocenie płynności między trzema goglami, które do wyświetlania obrazu używają tego samego smartfona. Na taki wynik mogą się składać inne elementy, którymi charakteryzują się konkretne okulary: ostrość wyświetlanego obrazu, zasięg dokładnego widzenia, czy poziom zmęczenia oczu. Kilku respondentów przyznało, że obraz w okularach Hykker VR Glasses 3D jest nienaturalnie wyostrzony, dodatkowo zauważyli, iż jest on wyświetlany w proporcjach 4:3, przez co obraz jest węższy. Powodem takiego rezultatu może być materiał oraz jakość wykonania soczewek, co może również przekładać się na uczucie gorszej lub lepszej płynności wyświetlanego obrazu. Podobnie kilku respondentów skarżyło się na zmęczenie oczu po skorzystaniu z niektórych modeli. Warto podkreślić, że każde oko postrzega inaczej dany obraz oraz inaczej reaguje na bodźce zewnętrzne. Efektem tego są rozbieżności w ocenie, jednak próba 16 osób w różnym wieku, korzystających z 4 modeli gogli VR, pozwala wyciągnąć pewne wnioski i znaleźć okulary, które w najmniejszym stopniu wpływają na zmęczenie oczu.

Można uznać, że respondenci nie kierowali się ceną, gdyż w celu uwiarogodnienia wyników, nie zostali o niej poinformowani przed wykonaniem badań. Zostali oni poinformowani o aktualnych cenach dopiero po zakończeniu ankiety. Ceny przedstawiają się następująco (stan na listopad 2018):

- Oculus Rift DK2 około 1000 zł
- Vrizzmo około 150 zł
- Hykker VR Glasses 3D około 40 zł
- Cardboard około 10 zł

Analizując uzyskane wyniki można dojść do wniosku, że nie ma urządzenia idealnego, które spełnia wszelkie oczekiwania. Przed zakupem gogli VR należy zastanowić się nad oczekiwaniami wobec sprzętu i w zależności od potrzeb wybrać najodpowiedniejsze. Jak wynika z przeprowadzonego badania najdroższe urządzenie niekoniecznie jest najlepsze.

Podsumowując, mimo, że Oculus Rift DK2 osiągnął najlepsze oceny w największej liczbie kategorii, w większości przypadków jego przewaga nad Hykker VR Glasses 3D nie była znacząca. Biorąc pod uwagę 25 razy wyższą cenę i niską dostępność urządzenia, warto wziąć pod uwagę okulary Hykker, jako poważną alternatywę dla tego typu rozwiązania. Nie tylko cena jest argumentem przemawiającym za tym wyborem. Gogle te są nieporównywalnie bardziej dostępne w Polsce, jak również osiągnęły wyższe wyniki w kategorii mobilności, ostrości krawędzi, zasięgu widzenia oraz wagi urządzenia od gogli Oculus Rift.

Literatura

[1] *Wykorzystanie rozszerzonej rzeczywistości we współczesnych systemach informatycznych*, Szymczyk T. Prace Instytutu Elektrotechniki, zeszyt 261, 2013.

[2] http://www.historyofinformation.com/expanded.php?id=4543 [marzec 2018]

[3] http://www.miastogier.pl/wiki,strona-2592,historia_vr.html [marzec 2018]

[4] http://histografy.pl/wp-content/uploads/2017/01/sensorama.jpg [listopad 2018]

[5] https://pbs.twimg.com/media/CUyQ_SqWIAAq7Yi.jpg [listopad 2018]

[6] https://flashbak.com/wp-content/uploads/2014/11/PA-9197076-1024x678.jpg [listopad 2018]

[7] https://www.wired.com/2009/09/augmented-reality-the-ultimate-display-by-ivan-sutherland-1965/ [marzec 2018]

[8] https://www.oculus.com/blog/dk2s-now-shipping-new-0-4-0-sdk-beta-and-comic-con/ [maj 2018]

[9] https://twitter.com/brendaniribe/status/565888922362728449 [maj 2018]

[10] https://et.wikipedia.org/wiki/Oculus_Rift#/media/File:Oculus_Rift_development_kit_2.jpg [listopad 2018]

[11] https://en.wikipedia.org/wiki/Oculus_Rift#/media/File:Oculus-Rift-CV1-Headset-Front.jpg [listopad 2018]

[12] https://www.polygon.com/virtual-reality/2018/5/1/17284454/oculus-go-review [maj 2018]

[13] A Review Paper on Oculus Rift - A Virtual Reality Headset - Parth Rajesh Desai, Pooja Nikhil Desai, Komal Deepak Ajmera, Khushbu Mehta, *U.G. students, Electronics and Telecommunication Department, DJSCOE, Vile Parle (W), Mumbai 400056,India. International Journal of Engineering Trends and Technology (IJETT) –Volume 13 Number 4 – Jul 2004*

[14] https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/ [maj 2018]

[15] https://riftinfo.com/oculus-rift-specs-dk1-vs-dk2-comparison [maj 2018]

[16] http://www.benchmark.pl/aktualnosci/oculus-go-samodzielne-gogle-vr-za-199-dolarow.html [maj 2018]

[17] https://arstechnica.com/gaming/2018/05/oculus-go-review-the-wireless-vr-future-begins-today-for-only-199/ [maj 2018]

[18] Souppouris A., How HTC and Valve built the Vive, https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/ [maj 2018]

https://pl.wikipedia.org/wiki/Google_Cardboard#/media/File:Assembled_Google_Cardboard_VR_mount.jpg [listopad 2018]

[19] Souppouris A., How HTC and Valve built the Vive, https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/ [maj 2018]

[20] *How HTC and Valve built the Vive,* Aaron Souppouris, www.engadget.com, https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/ [maj 2018]

[21] *HTC's Vive Pro will add more pixels to an otherwise familiar-looking VR system*, Sam Machkovech, www.arstechnika.com, https://arstechnica.com/gaming/2018/01/htcs-vive-pro-will-add-more-pixels-to-an-otherwise-familiar-looking-vr-system/ [maj 2018]

[22] *Oculus Rift vs. Vive Pro*, Jon Martindale, www.digitaltrends.com, https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-vive-pro/ [maj 2018]

[23] https://en.wikipedia.org/wiki/HTC_Vive [maj 2018]

[24] *New HTC Vives Weigh 15% Less Than They Did at Launch,* Ben Lang, www.roadtovr.com, https://www.roadtovr.com/htc-vive-weight-15-percent-lighter-than-original-headset-vs-oculus-rift-comparison/ [maj 2018]

[25] https://vr.google.com/intl/pl_pl/cardboard/get-cardboard/ [maj 2018]

[26] https://news.samsung.com/global/gear-vr-how-samsung-makes-virtual-reality-a-reality [maj 2018]

[27] https://eazlblog.com/tag/customer-interviews/ [listopad 2018]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 44-48
Wysłane: 2018-11-25
Przyjęte: 2018-12-02

Ocena metod obliczania zużycia energii sportowca zaimplementowanych na urządzeniach z systemem Android

Sylwester Muzyka*, Piotr Wójcik*, Jakub Smołka
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł prezentuje dwie metody obliczania energii zużytej podczas wysiłku. Wykorzystują one dane uzyskane przy użyciu sensorów dostępnych na urządzeniach mobilnych z systemem Android. Metody te zostały porównane ze sobą jak i z wynikami spalonych kalorii uzyskanymi z dwóch aplikacji dostępnych w sklepie Google Play. Aplikacje te to Endomondo oraz Sports Tracker.

Słowa kluczowe: zużyta energia; spalone kalorie; Android; sensor

*Autor do korespondencji.

Adresy e-mail: sylwester.muzyka@pollub.edu.pl, piotr.wojcik3@pollub.edu.pl

Evaluation of methods for computing athlete's energy expenditure implemented on Android devices

Sylwester Muzyka*, Piotr Wójcik*, Jakub Smołka
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents two methods of computing calories burned during exercise. They use data obtained using sensors available on mobile devices running Android OS. These methods were compared with each other and with the results obtained from two applications available on the Google Play store. These applications are Endomondo and Sports Tracker.

Keywords: consumed energy; calories burned; Android; sensor

*Corresponding author.

E-mail addresses: sylwester.muzyka@pollub.edu.pl, piotr.wojcik3@pollub.edu.pl

1. Wstęp

Aktywność fizyczna jest bardzo ważną częścią życia wielu osób. Jak wiadomo jest też w większości przypadków domeną ludzi prowadzących zdrowy tryb życia. Już w czasach starożytnych rywalizowano między sobą o miano najlepszych w wielu różnych dyscyplinach. Płynący czas powodował wymyślanie coraz to nowszych i ciekawszych dyscyplin. Z jego biegiem zaczęto zapisywać rekordy z wielu sportowych konkurencji, dzięki czemu rywalizacja stawała się bardziej emocjonująca.

Rozwój techniki spowodował powstanie pierwszego smartfonu pod koniec ubiegłego wieku. Wraz z rozpowszechnieniem tego wynalazku w obecnym wieku zaczęły powstawać coraz to bardziej ciekawe i funkcjonalne aplikacje przeznaczone właśnie na urządzenia mobilne. Obecnie wielu użytkowników ma dostęp do aplikacji związanych z uprawianiem sportu. Pozwalają one na przechowywanie danych na temat swoich osiągnięć, czy też udostępnienie ich znajomym. Aplikacje takie zwiększają motywację ich użytkowników dzięki chęci bicia rekordów własnych lub też cudzych. Najczęściej pozwalają one na zliczanie dystansu czy też spalonych kalorii.

Aktualnie na rynku systemów wykorzystywanych na urządzeniach mobilnych przewodzą dwa, Android oraz iOS. Zdecydowaną przewagę ma pierwszy z nich. Może się on pochwalić instalacją na około 75% wszystkich smartfonów [1].

Urządzenia takie posiadają często po kilka czy nawet kilkanaście różnego rodzaju sensorów. Dwa z nich to GPS, będący czujnikiem lokalizacji, oraz akcelerometr, pozwalający na wykrycie ułożenia telefonu w przestrzeni [2]. Przy ich użyciu aplikacje sportowe mogą zliczać dystans pokonany przez użytkownika, a na jego podstawie spalone przez niego kalorie. Warto zaznaczyć, że nie istnieje żaden oficjalny sposób na ich liczenie, stąd też każda aplikacja może generować różne wyniki dla tych samych parametrów. Z tego też powodu powstało Kompendium Aktywności Fizycznej na Uniwersytecie Stanforda badające tego typu zależności [3].

Celem artykułu jest porównanie dwóch metod liczących zużycie energii podczas wysiłku na podstawie danych zebranych z sensorów urządzeń mobilnych. Pomocna w tym będzie aplikacja stworzona na system Android. Wyniki uzyskane przy wykorzystaniu tych sposobów zostały zestawione z rezultatami otrzymanymi z dwóch istniejących aplikacji jakimi są Endomondo oraz Sports Tracker [4, 5].

2. Przegląd metod liczących kalorie

Kaloria jest to jednostka energii równa około 4,1868 J. Określenie spalone kalorie oznacza natomiast ilość energii jaką zużywa człowiek w czasie wysiłku. Najczęściej stosuje się do tego jednostkę wielokrotną jaką jest kilokaloria [6]. Jak do tej pory nie wyznaczono dokładnego wzoru liczącego kalorie zużyte podczas wysiłku. Tym samym nie istnieje żadna oficjalna metoda ich zliczania. Powodem jest fakt, że na liczbę spalonych kalorii wpływa wiele czynników. Są to dane na temat sportowca jak i aktywności jaką wykonał między innymi jego płeć, wiek, waga, wzrost, poziom zaawansowania, metabolizm oraz intensywność treningu wyrażana na przykład czasem i pokonanym dystansem [3, 7, 8]. Wybrane metody przedstawiono w podpunktach 2.1 i 2.2.

2.1. Metoda pierwsza – korzystająca z pokonanego dystansu

Pierwsza z nich korzysta jedynie z masy sportowca oraz pokonanego przez niego dystansu. Do wyliczenia zużytej energii wykorzystywane są dwa wzory w zależności od prędkości poruszania się. Wykorzystują one masę sportowca wyrażoną w funtach oraz pokonany przez niego dystans podany w milach. Zaprezentowano je poniżej [9].

Wzór 1 jest używany dla prędkości marszowych, nie większych niż 3,7 mili na godzinę.

$$bkc = 0{,}57 \cdot m \cdot s \tag{1}$$

gdzie: *bkc* – liczba spalonych kalorii [kcal], *m* – masa sportowca [lbs], *s* – pokonany dystans [miles].

Natomiast wzór 2 jest wykorzystywany dla prędkości biegowych, powyżej 3,7 mili na godzinę.

$$bkc = 0{,}72 \cdot m \cdot s \tag{2}$$

gdzie: *bkc* – liczba spalonych kalorii [kcal], *m* – masa sportowca [lbs], *s* – pokonany dystans [miles].

W dalszej części artykułu sposób ten będzie nazywany również metodą pierwszą.

2.2. Metoda druga – korzystająca z pułapu tlenowego i odpowiednika metabolicznego

Druga metoda wykorzystuje dodatkowo czas aktywności. Opiera się na pułapie tlenowym, który identyfikuje się jako zdolność organizmu do pochłaniania tlenu, czyli liczby mililitrów tlenu jaką jest w stanie pobrać jeden kilogram masy człowieka przez minutę wykonywanej aktywności [10]. Dodatkowo wykorzystywany w tym sposobie jest odpowiednik metaboliczny (MET), będący wskaźnikiem wyrażającym zużycie energii podczas różnych aktywności sportowych. 1 MET jest definiowany jako 1 kilokaloria na kilogram na godzinę, jak również jako 3,5 mililitry na kilogram na minutę [11]. Na tej podstawie podobnie jak wyżej posłużono się dwoma wzorami zależnymi od prędkości [12, 13, 14].

Wzór 3 jest używany dla prędkości marszowych, nie większych niż 3,7 mili na godzinę.

$$bkc = \frac{((0{,}1 \cdot \frac{s}{t}) + (1{,}8 \cdot \frac{s}{t} \cdot g) + 3{,}5) \cdot m}{210} \cdot t \tag{3}$$

gdzie: *bkc* – liczba spalonych kalorii [kcal], *s* – pokonany dystans [m], *t* – czas aktywności [min], *g* – nachylenie terenu [%].

Natomiast wzór 4 jest wykorzystywany dla prędkości biegowych, powyżej 3,7 mili na godzinę.

$$bkc = \frac{((0{,}2 \cdot \frac{s}{t}) + (0{,}9 \cdot \frac{s}{t} \cdot g) + 3{,}5) \cdot m}{210} \cdot t \tag{4}$$

gdzie: *bkc* – liczba spalonych kalorii [kcal], *s* – pokonany dystans [m], *t* – czas aktywności [min], *g* – nachylenie terenu [%].

W dalszej części artykułu sposób ten będzie nazywany również metodą drugą.

3. Aplikacja badawcza

W celu zbadania opisanych wyżej metod stworzono aplikację na urządzenia z systemem Android. Zaimplementowano w niej oba te sposoby wykorzystując do tego język obiektowy Java oraz środowisko programistyczne Android Studio. W przykładach 1 oraz 2 pokazano dwie funkcje będące odzwierciedleniem pokazanych w punkcie 2 wzorów zliczających liczbę spalonych kalorii.

Przykład 1. Fukcja obliczająca spalone kalorie na podstawie pokonanego dystansu

```
public double getBurnedCalories1Way(double distanceInMeters, double timeInSeconds)
{
   double distanceInKilometers = distanceInMeters / 1000;
   double timeInMinutes = timeInSeconds / 60;
   double speedInMetersPerMinute = distanceInMeters / timeInMinutes;
   double kilocaloriesBurnedInKilometer;

   if(speedInMetersPerMinute <= 99.243)
      kilocaloriesBurnedInKilometer = (0.57 * mCalorieCalculatorModel.getWeightInPounds()) / 1.609344;
   else
      kilocaloriesBurnedInKilometer = (0.72 * mCalorieCalculatorModel.getWeightInPounds()) / 1.609344;

   return kilocaloriesBurnedInKilometer * distanceInKilometers;
}
```

Przykład 1 przedstawia implementację pierwszej metody zliczającej spalone kalorie. Listing ten jest odzwierciedleniem wzorów 1 oraz 2.

Przykład 2. Fukcja obliczająca spalone kalorie na podstawie pułapu tlenowego oraz odpowiednika metabolicznego

```
public double getBurnedCalories2Way(double distanceInMeters, double inclinationInPercents, double timeInSeconds)
{
   double timeInMinutes = timeInSeconds / 60;
   double speedInMetersPerMinute = distanceInMeters / timeInMinutes;
   double inclinationInFraction = inclinationInPercents / 100;
   double oxygenVolumeUsageInMillilitersPerKilogramPerMinute;

   if(speedInMetersPerMinute <= 99.243)
      oxygenVolumeUsageInMillilitersPerKilogramPerMinute = (0.1 * speedInMetersPerMinute) + (1.8 * speedInMetersPerMinute * inclinationInFraction) + 3.5;
   else
      oxygenVolumeUsageInMillilitersPerKilogramPerMinute = (0.2 * speedInMetersPerMinute) + (0.9 * speedInMetersPerMinute * inclinationInFraction) + 3.5;

   double oxygenVolumeUsageInMillilitersPerMinute = oxygenVolumeUsageInMillilitersPerKilogramPerMinute * mCalorieCalculatorModel.getWeightInKilograms();
   double kilocaloriesBurnedPerMinute = oxygenVolumeUsageInMillilitersPerMinute / 210;

   return kilocaloriesBurnedPerMinute * timeInMinutes;
}
```

Przykład 2 przedstawia implementację drugiej metody zliczającej spalone kalorie. Listing ten jest odzwierciedleniem wzorów 3 oraz 4.

4. Porównanie metod

W punkcie tym zestawiono ze sobą wyniki badań z obu omawianych metod. Dodatkowo porównano je z rezultatami uzyskanymi przy pomocy aplikacji Endomondo oraz Sports Tracker. Wyniki zebrano na odległości 611 metrów uzyskanej z uśrednienia długości trasy wyznaczonej w Google Maps oraz MapFactor Navigator. Co więcej, badania były wykonywane przy dwóch różnych prędkościach, marszowej oraz biegowej. Poniżej w tabelach 1, 2 oraz 3 zaprezentowano wyniki otrzymanych pomiarów. Odległości różnią się, ponieważ były wyznaczane za pomocą dwóch metod zliczania dystansu przy użyciu sensorów, jakimi są wcześniej wspomniany GPS oraz krokomierz korzystający z akcelerometru.

Tabela 1. Wyniki pomiarów na dystansie 611 metrów

Spalone kalorie – dystans 611m			
Czas (m)	Zmierzony dystans (m)	Spalone kalorie (**1. metoda**) (kcal)	Spalone kalorie (**2. metoda**) (kcal)
06:55,42	563,24	35,18	30,69
06:58,37	571,46	35,7	31,07
07:03,39	555,77	34,72	30,58
07:29,37	547,55	34,2	30,84
07:25,13	583,41	36,44	32,12
07:02,98	549,05	34,3	30,31
07:00,36	561,74	35,09	30,74
06:51,53	555,77	34,72	30,32
04:03,80	548,25	43,26	47,19
06:06,57	553,35	34,57	29,22
08:10,38	617,02	38,54	34,4
08:00,40	612,54	38,26	34,01
07:11,98	604,46	37,76	32,62
07:16,99	665,22	41,55	35,05
07:01,05	625,7	39,09	33,19
06:56,16	627,01	39,17	33,13
07:05,16	619,65	38,71	33,05
04:14,36	587,34	46,34	50,4
04:27,17	597,85	47,17	51,49
08:16,06	630,29	39,37	35,03
06:49,93	473,55	29,58	27,15
07:26,90	505,46	31,57	29,19
04:07,15	554,94	43,79	47,77
04:08,99	571,26	45,08	49,06
04:08,35	560,15	44,2	48,2
04:09,15	552,5	43,6	47,63
06:46,27	571,46	35,7	30,8
07:06,53	581,17	36,3	31,62
04:07,34	533,8	42,12	46,17
07:20,09	572,2	35,74	31,58
04:12,29	560,15	44,2	48,28
06:47,33	613,88	38,35	32,44
06:57,98	634,29	39,62	33,45

Tabela 1 zawiera wyniki uzyskane z autorskiej aplikacji podczas poruszania się na dystansie 611 metrów.

Tabela 2. Wyniki pomiarów na dystansie 611 metrów z aplikacji Endomondo

Endomondo – spalone kalorie – dystans 611m		
Czas (m)	Zmierzony dystans (m)	Spalone kalorie (kcal)
07:06	570	36
06:57	590	40
04:01	590	52
08:03	640	40
03:00	630	50
04:03	610	53
07:21	580	37
03:58	590	52

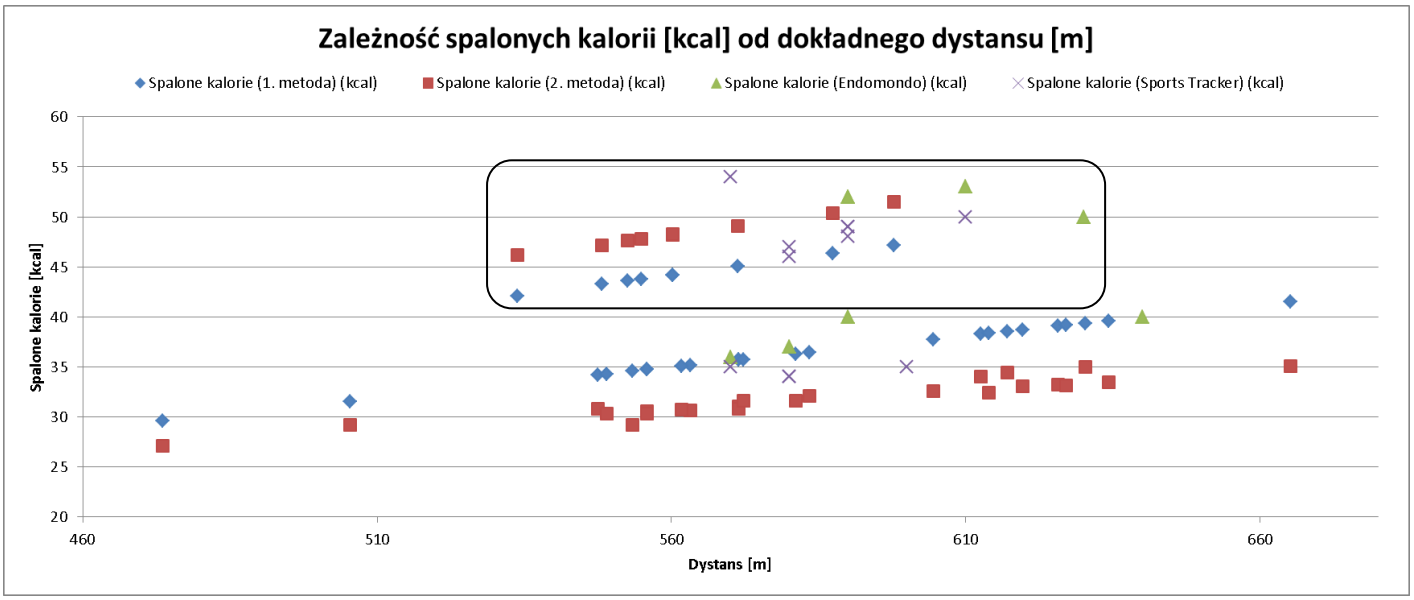
Tabela 2 zawiera wyniki uzyskane z aplikacji Endomondo podczas poruszania się na dystansie 611 metrów.

Tabela 3. Wyniki pomiarów na dystansie 611 metrów z aplikacji Sports Tracker

Sports Tracker – spalone kalorie – dystans 611m		
Czas (m)	Zmierzony dystans (m)	Spalone kalorie (kcal)
03:59	580	47
08:07	600	35
03:59	590	49
03:41	570	54
06:40	610	50
06:24	630	61
04:08	580	46
07:13	570	35
04:01	590	49
07:25	580	34
03:57	590	48
07:15	580	34

Tabela 3 zawiera wyniki uzyskane z aplikacji Sports Tracker podczas poruszania się na dystansie 611 metrów.

Poniżej natomiast wszystkie dane z tabel 1, 2 oraz 3 zaprezentowano na wykresie (rys. 1) w celu łatwiejszej ich oceny oraz porównania.



Rys. 1. Zależność spalonych kalorii od dokładnego dystansu (obszar zaznaczony – wyniki uzyskane podczas biegu, pozostały obszar – wyniki uzyskane podczas marszu)

Rysunek 1 przedstawia wykres będący zestawieniem danych zebranych w tabelach 1, 2 oraz 3. Zaznaczony obszar to wyniki uzyskane podczas biegu, natomiast pozostałe reprezentują energię zużytą podczas marszu. Wykres ten pokazuje, że w przypadku badanych metod na obu prędkościach wraz ze wzrostem dystansu liczba spalonych kalorii rośnie. Podczas chodu wyższe wyniki zwraca sposób korzystający jedynie z masy sportowca oraz dystansu jaki pokonał. Bieg natomiast powoduje odwrócenie tej relacji, w związku z czym większa ilość zużytej energii jest wskazywana przez metodę opierającą się na pułapie tlenowym oraz odpowiedniku metabolicznym. Sposób ten dodatkowo korzysta z czasu aktywności.

Wyniki uzyskane z aplikacji Endomondo oraz Sports Tracker korzystają z większej ilości danych przez co, jak mawiają twórcy tej pierwszej, powinny być bardziej dokładne [8]. Obie oprócz masy sportowca, pokonanej przez niego odległości oraz czasu aktywności wykorzystują dodatkowo płeć użytkownika oraz jego wiek. Powyższy wykres pokazuje, że zużyta energia uzyskana za pomocą omawianych w tym artykule metod pokrywa się z wynikami otrzymanymi z tych aplikacji. Jest to potwierdzeniem tego, że badane sposoby są porównywalne z tymi użytymi w aplikacjach Endomondo oraz Sports Tracker.

Aby możliwe było dokładniejsze określenie, która z omawianych metod jest bardziej precyzyjna wyliczono względne błędy pomiarów. Skorzystano z poniższego wzoru.

$$\delta = \left(\frac{|x - x_o|}{x} \right) \cdot 100\% \qquad (5)$$

gdzie: δ - względny błąd pomiaru, x - wartość dokładna, x_o – wartość zmierzona.

Jako, że nie ma oficjalnego wzoru na spalone kalorie, za wartość dokładną uznano uśrednione wyniki z aplikacji Endomondo oraz Sports Tracker wyznaczone dla dokładnej odległości, to znaczy 611 metrów, podanej ręcznie w obu tych aplikacjach. Za wartość zmierzoną uznano natomiast liczbę spalonych kalorii uzyskaną przez podstawienie tego dystansu do wzorów 1, 2, 3 oraz 4. Zabiegi te wykonano w celu osiągnięcia prawidłowej wartości błędów pomiarów. Gdyby zamiast 611 metrów podstawić uśrednione odległości, to błędy zostałyby wyliczone na różnych dystansach. Nie byłyby one więc miarodajne. Za czas w aplikacjach oraz metodzie korzystającej z pułapu tlenowego podstawiono uśredniony czas uzyskany z badań opisywanych metod oddzielnie dla marszu oraz biegu. W ten sposób otrzymano wyniki, które prezentuje tabela 4.

Tabela 4. Dane potrzebne do wyliczenia błędów pomiarów

Rodzaj aktywności	Dystans (m)	Uśredniony czas (min)	1. metoda (kcal)	2. metoda (kcal)	Endomondo (kcal)	Sports Tracker (kcal)
marsz	611	07:10,12	38,17	32,84	40	34
bieg	611	04:10,54	48,21	52,13	53	50

Na podstawie tabeli 4 za pomocą wzoru 5 wyznaczono poniżej zebrane błędy pomiarów. Pod x podstawiono uśrednione zużycie energii z aplikacji Endomondo oraz Sports Tracker dla badanej odległości (611 metrów). Za x_o uznano natomiast spalone kalorie wyliczone za pomocą obu metod (punkt 2.1 oraz 2.2) również dla badanej odległości. Względne błędy pomiarów przedstawiono w tabeli 5.

Tabela 5. Względne błędy pomiarów

Rodzaj aktywności	Dystans (m)	Metoda	x_o (kcal)	x (kcal)	Względny błąd pomiaru
marsz	611	1	38,17	37	3,16%
marsz	611	2	32,84	37	11,24%
bieg	611	1	48,21	51,5	6,39%
bieg	611	2	52,13	51,5	1,22%

Tabela 5 przedstawia względne błędy pomiarów. Według nich metodą bardziej dokładną jeśli chodzi o liczenie kalorii podczas marszu była ta korzystająca jedynie z masy sportowca i dystansu jaki pokonał. Błąd pomiaru jaki został wyliczony w jej przypadku to około 3,16%. Świadczy to o tym, że podczas chodu czas jest mało znaczącym czynnikiem [15]. Podczas biegu natomiast bardziej dokładne dane zwracała metoda korzystająca z pułapu tlenowego oraz współczynnika metabolicznego. Wynik błędu pomiaru to jedynie około 1,22%, co świadczy o jeszcze bardziej dokładnych rezultatach zużycia energii oraz o tym, że podczas biegu czas jest ważniejszy niż w przypadku marszu.

5. Wnioski

Celem artykułu było porównanie dwóch metod liczących zużycie energii podczas wysiłku na podstawie danych zebranych z sensorów urządzeń mobilnych. Został on zrealizowany poprzez skonfrontowanie ze sobą dwóch metod wykorzystujących dystans wyznaczony przy pomocy dostępnych czujników. Jako, że nie istnieje oficjalny sposób zliczania kalorii to za punkt odniesienia uznano dobrze znane aplikacje ze sklepu Google Play, Endomondo oraz Sports Tracker. Na ich podstawie możliwe okazało się stwierdzenie, że podczas marszu czas nie jest znaczącym czynnikiem, ponieważ dokładniejszą metodą jest sposób nie uwzględniający go w swoim wzorze. Odwrotne wyniki uzyskano w przypadku biegu. W tym przypadku to właśnie metoda wykorzystująca czas zwraca dokładniejsze wyniki pomiarów zużycia energii przez sportowca. Wyniki tego porównania świadczą o tym, że znalezienie odpowiedniego wzoru zliczania spalonych kalorii jest rzeczą bardzo trudną, zwłaszcza, że każdy człowiek jest inny, posiada inną przemianę materii oraz zaawansowanie w uprawianiu sportu, a wykonywane przez niego czynności mają różną intensywność.

Literatura

[1] http://gs.statcounter.com/os-market-share/mobile/worldwide/2018, Mobile Operating System Market Share Worldwide [22.11.2018]

[2] Greg Milette, Adam Stroud, Professional Android Sensor Programming, John Wiley & Sons, 2012

[3] https://sites.google.com/site/compendiumofphysicalactivities, Compendium of Physicial Activities [22.11.2018]

[4] https://play.google.com/store/apps/details?id=com.endomondo.android&hl=pl, Endomondo - Bieganie & Rower [22.11.2018]

[5] https://play.google.com/store/apps/details?id=com.stt.android&hl=pl, Sports Tracker – bieganie i jazda na rowerze [22.11.2018]

[6] https://pl.wikipedia.org/wiki/Kaloria, Kaloria [22.11.2018]

[7] Edward J. Coates, Measured Success!, Trafford Publishing, 2005

[8] https://support.endomondo.com/hc/en-us/articles/201861383-Calories, Calories [22.11.2018]

[9] http://www.bmi-calculator.net/how-many-calories-do-you-burn-walking-or-running-a-mile, How Many Calories Do You Burn Walking or Running a Mile? [22.11.2018]

[10] https://en.wikipedia.org/wiki/VO2_max, VO_2 max [22.11.2018]

[11] https://en.wikipedia.org/wiki/Metabolic_equivalent, Metabolic equivalent [22.11.2018]

[12] Alex A. Lluch, Easy Fat, Carb, and Calorie Counter, WS Publishing Group, 2009

[13] Nicholas Robertson, Comparing Calorie Expenditure and Rating of Perceived Exertion between the Curve and a Motorized Treadmill, Eastern Washington Univeristy, 2014

[14] https://www.livestrong.com/article/34973-calculate-treadmill-calories, How to Calculate Treadmill Calories [22.11.2018]

[15] Karen J Nolan, Jo-Ann Heslin, The Calorie Counter, 6th Edition, Simon and Schuster, 2012

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 49-53
Wysłane: 2018-11-22
Przyjęte: 2018-12-02

Programowanie wielowątkowe w językach strukturalnych i obiektowych

Mateusz Łukasz Wiśniewski
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono sposoby programowania wielowątkowego w wybranych językach programowania takich jak: C podejście strukturalne oraz obiektowe C++ przy użyciu bibliotek Posix Threads dla języka C oraz bibliotekę Boost dla języka C++. Opisano również charakterystykę wybranych bibliotek. Przedstawiono przykładowe rozwiązania typowych problemów programistycznych wykorzystujących wątki. Starano odpowiedzieć na pytanie czym kierować się przy wyborze języka do nauki programowania wielowątkowego oraz programowania w ogólnym tego słowa znaczeniu.

Słowa kluczowe: programowanie; wielowątkowość; pthreads; boost

Adres e-mail: wmmateusz@gmail.com

Multithreaded programming in structural and object-oriented languages

Mateusz Łukasz Wiśniewski
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents multithreaded programming in selected programming languages such as: C structural and C ++ object oriented approach using Posix Threads libraries for C language and Boost library for C ++ language. The characteristics of selected libraries are also described. Examples of typical programming problems using threads are presented. An effort was made to answer the question of what to do when choosing a language for learning multithread programming and programming in the general meaning of this word.

Keywords: programming; multithreading; pthreads; boost

E-mail address: wmmateusz@gmail.com

1. Wstęp

Programowanie wielowątkowe jest dobrym sposobem na pisanie programów, które w tym samym czasie wykonują równolegle różne operacje. Wielowątkowość jest cechą systemu operacyjnego a nie samego języka programowania. Obecne systemy operacyjne oraz produkowane procesory posiadające kilka rdzeni obsługują wiele wątków. Możliwość programowania wielowątkowego dostarcza wiele języków programowania. Wybór zależy od wielu czynników często bardzo subiektywnych. Ciekawą kwestią wydaje się być wybór środowiska do nauki programowania wielowątkowego. Prostszy strukturalny język C może okazać się lepszym rozwiązaniem. Dodatkowo wielowątkowość w C z biblioteką Posil Threads ma dłuższą historię niż w języku C++, w którym jest dostępna od wersji 11 ew. w postaci popularnej ale stosunkowo młodej biblioteki Boost.

2. Przegląd języków programowania

W tym rozdziale przedstawiono krótką charakterystykę języków programowania C oraz C++.

Język C

Język C tworzony jako język łączący języki wysokiego poziomu z niskopoziomowymi. Nowe kompilatory języka mogły być szybko tworzone na nowe platformy. Niskopoziomowa natura języka C pozwala programiście sprawować kontrolę na tym, co robi komputer, jednocześnie pozwalając na specjalne dostosowanie i optymalizacje na konkretnych platformach. Pozwala to na szybkie działanie kodu nawet na ograniczonym sprzęcie, na przykład w systemach wbudowanych. Język C nie zawiera wielu właściwości dostępnych w innych językach programowania m. in. brak bezpośredniej obsługi programowania wielowątkowego. Brak jest natomiast w nim obsługi wyjątków czy obsługi programowania obiektowego, a w szczególności polimorfizmu czy dziedziczenia.[1]

Język C++

Język C++ jest językiem wieloparadygmatowym. Znaczy to, że można w nim stosować różne style programowania: programowanie proceduralne, obiektowe czy generyczne. C++ zakłada statyczną kontrolę typów, umożliwia bezpośrednie zarządzanie wolną pamięcią.[7]

3. Przegląd użytych bibliotek

W rozdziale opisano dwie biblioteki programistyczne Posix Threads oraz Boost.

POSIX Threads

Biblioteki wątków POSIX są standardowymi API (Interfejs programowania aplikacji) interfejs programistyczny aplikacji wielowątkowych dla języka C / C ++.

Standard POSIX Threads wchodzi w skład standardu POSIX określająca implementację wielowątkowości, która obejmuje podstawowe mechanizmy zarządzania wątkami, obiektami synchronizującymi oraz definiuje jednolity interfejs programistyczny dla języka C. Standard określa pewien podstawowy zestaw funkcji oraz szereg opcji, które mogą być

udostępnione przez implementację. Rozwiązanie to jest bardziej efektywne w systemach wieloprocesorowych lub wielordzeniowych, w których przepływ procesów można zaplanować na innym procesorze, zwiększając szybkość działania dzięki przetwarzaniu równoległym lub rozproszonym. Wątki wymagają mniejszej ilości zasobów niż tworzenie osobno nowego procesu.[3]

Boost

Boost to zestaw bibliotek dla języka programowania C++, który zapewnia wsparcie dla zadań i struktur, takich jak algebra liniowa, generowanie liczby pseudolosowej, wielowątkowość, przetwarzanie obrazu, wyrażeń regularnych i testowanie jednostkowe. Zawiera ponad osiemdziesiąt pojedynczych bibliotek. Biblioteki boost są objęte licencją Boost Software License, która pozwala każdemu używać, modyfikować i dystrybuować biblioteki za darmo. Biblioteki są niezależne od platformy i obsługują najpopularniejsze kompilatory.

Za tworzenie i publikowanie bibliotek Boost odpowiedzialna jest społeczność Boost składa się ona ze stosunkowo dużej grupy programistów C ++ z całego świata koordynowanych za pośrednictwem strony internetowej www.boost.org. Serwis GitHub jest używany jako repozytorium do przechowywania kodu bibliotek. Misją tej społeczności jest rozwijanie i gromadzenie wysokiej jakości bibliotek, które uzupełniają standardową bibliotekę. Biblioteki, które okażą się wartościowe i stają się ważne dla rozwoju aplikacji napisanych w języku C ++, mają dużą szansę na włączenie do standardowej biblioteki.[6]

Tabela 1. Zestawienie bibliotek.

Biblioteka	Posix Threads	Boost
Rok powstania	1985-1990	1999
Język	C	C++
Aktualna wersja stabilna	Sierpień 2018	Sierpień 2018
Platforma sprzętowa	GNU/Linux, Windows	Wieloplatformowa
Licencja	GNU Library	Boost Software License

4. Przegląd dostępnych funkcji w bibliotekach

Biblioteka Posix Threads posiada około 100 procedur obsługujących wątki, wszystkie poprzedzone pthread_ i można je podzielić na cztery grupy:

1) Zarządzanie wątkami - tworzenie, łączenie wątków itp.,
2) Mutexy,
3) Zmienne warunku,
4) Synchronizacja między wątkami za pomocą blokad i barier odczytu / zapisu.

Biblioteka Boost C ++ udostępnia kilka klas, które mogą być używane do pisania wielowątkowych programów w języku C++. Proces pisania wielowątkowego programu jest podobny do programów pisanych chociażby w Javie.

Tabela 2. Zestawienie wybranych funkcji dostępnych w bibliotekach.[8]

Posix Threads	Boost
Tworzenie i usuwanie wątków	
pthread_create (thread,attr,start_routine,arg) pthread_exit(status) pthread_cancel(thread) pthread_attr_init(attr) pthread_attr_destroy(attr)	boost::thread::thread() boost::thread::this_thread() boost::thread::thread_group()
Dołączanie i odłączanie wątków	
pthread_join (threadid,status) pthread_detach (threadid) pthread_attr_setdetachstate (attr,detachstate) pthread_attr_getdetachstate (attr,detachstate)	boost::thread::detach() boost::thread::join() boost::thread::join_all() boost::thread::joinable() boost::thread::timed_join() boost::thread::try_join_for() boost::thread::try_join_until()
Mutexy	
pthread_mutex_init (mutex,attr) pthread_mutex_destroy (mutex) pthread_mutexattr_init (attr) pthread_mutexattr_destroy (attr) pthread_mutex_lock (mutex) pthread_mutex_trylock (mutex) pthread_mutex_unlock (mutex)	boost::mutex::mutex() boost::mutex:recursive_mutex() boost::mutex::timed_mutex() boost::mutex:: lock_guard() boost::mutex:: unique_lock() boost::mutex::try_lock() boost::mutex::lock() boost::mutex::call_one()
Zmienne warunkowe	
pthread_cond_init (condition,attr) pthread_cond_destroy (condition) pthread_condattr_init(attr) pthread_condattr_destroy (attr) pthread_cond_wait (condition,mutex) pthread_cond_signal (condition) pthread_cond_broadcast (condition)	boost::condition_variable:: wait() boost::condition_variable:: timed_wait() boost::condition_variable:: wait_for() boost::condition_variable_ any::wait() boost::condition_variable_ any::wait_for() boost::condition_variable_ notify_one() boost::condition_variable_ notify_all()
Inne	
pthread_attr_getstacksize (attr, stacksize) pthread_attr_setstacksize (attr, stacksize) pthread_attr_getstackaddr (attr, stackaddr) pthread_attr_setstackaddr (attr, stackaddr) pthread_self () pthread_equal (thread1,thread2) pthread_once (once_control, init_routine)	boost::thread::sleep() boost::this_thread::sleep_for() boost::this_thread::sleep_until() boost::this_thread:: interruption_point() boost::thread::id()

5. Przykładowe programy

Rozdział przedstawia opis oraz fragmenty rozwiązań typowych programistycznych problemów synchronizacji.

Problem producent - konsument

Problem producenta i konsumenta to przykład synchronizacji zasobów. W zagadnieniu tym występują dwa rodzaje procesów, pomiędzy którymi są współdzielone pewne zasoby (pudełko). Pierwszy proces – producent generuje dane a drugi

– konsument je pobiera. Zadaniem producenta jest dostarczenie danych do pudełka, natomiast zadaniem konsumenta jest ich pobieranie z pudełka.

Listing 1. Kod funkcji - pthreads.[10]

```
void wloz(int m) {
    pthread_mutex_lock(&mutex);
    if (iloscElem == ILOSC_ELEMENTOW) {{
        printf ("Pudelko jest pelne.\n");
    }
    while (iloscElem == ILOSC_ELEMENTOW) {;
        pthread_cond_wait(&cond, &mutex);
    }
    }
    zawartosc[p] = m;
    p = (p + 1) % ILOSC_ELEMENTOW;
    ++iloscElem;
    pthread_mutex_unlock(&mutex);
    pthread_cond_signal(&cond);
}
```

W pierwszym przykładzie (Listing 1) wykorzystano funkcję pthread_mutex_lock() przysyłając w parametrze zmienną mutex. Funkcja ta pozwala zablokować zmienną oraz dalej wykonywać na niej operację. Po wszystkim należy ręcznie odblokować zmienną używając funkcji pthread_mutex_unlock(). Należy pamiętać, że ilość wywołań funkcji otwierającej blokadę musi być taka sama jak ilosć funkcji zamykającej tę blokadę. W przeciwnym razie może dojść do tzw. zakleszczenia.

W programie użyty został obiekt typu pthread_cond_t, który udostępnia funkcję do komunikacji między wątkami. Funkcja pthread_cond_wait() jako parametr przyjmuję referencję do obieku mutex oraz cond i zatrzymuje działanie wątku do czasu, aż inny wątek wyśle powiadomienie. Po wywołaniu funkcji pthread_cond_wait(), blokada obiektu podanego jako parametr jest zdjęta. Po otrzymaniu powiadomienia od innego wątku następuje wznowienie działania wątku oraz ponowne ustawienie blokady na obiekcie mutex. Do komunikacji między wątkami służy metoda pthread_cond_signal(). Wznawia ona wszystkie wątki, które wywołały wcześniej metodę wait() danego obiektu condition.

Listing 2. Kod funkcji - boost.[2]

```
class Pudelko {
private:
    boost::mutex mutex;
    boost::condition cond;
    unsigned int p, c, iloscElem;
    int zawartosc[ILOSC_ELEMENTOW];
public:
    Pudelko()
    : p(0), c(0), iloscElem(0){}

void wloz(int m)    {
      mutex.lock();
if (iloscElem == ILOSC_ELEMENTOW) {{
    boost::mutex::scoped_lock lock(ioMutex);
    cout << "Pudelko jest pelne." << std::endl;
    }
    while (iloscElem == ILOSC_ELEMENTOW) {
    cond.wait(mutex);
    }
}
    this->zawartosc[p] = m;
    p = (p+1) % ILOSC_ELEMENTOW;
    ++iloscElem;
    cond.notify_one();
    mutex.unlock();
}
Pudelko pudelko;
```

W przykładzie drugim (Listing 2) wykorzystano funkcje lock() obiektu boost::mutex, która jest alternatywą dla funkcji pthread_mutex_lock() służy ona do zamknięcia blokady. Otwarcie blokady następuje po wywołaniu funkcji unlock() na zmiennej mutex. Również w tym przypadku należy wystrzegać się zakleszczania i pamietać o ilości wywołań funkcji lock() oraz unlock().

W programie użyty został obiekt typu boost:condition, który udostępnia funkcję do komunikacji między wątkami, podobnie jak w przypadku biblioteki pthread – obiekt pthread_cond oraz jego funkcję - wait() wywołana na obiekcie boost::conditio jako parametr przyjmuje obiekt typu boost::mutex i zatrzymuje działanie wątku do czasu, aż inny wątek wyśle stosowane powiadomienie. Po wywołaniu funkcji wait(), blokada obiektu podanego jako parametr jest zdjęta. Po otrzymaniu powiadomienia od innego wątku następuje wznowienie działania wątku oraz ponowne ustawienie blokady na obiekcie boost::mutex. Do komunikacji między wątkami służy metoda notify_one() obiektu boost::condition. Wznawia ona wszystkie wątki, które wywołały wcześniej metodę wait() danego obiektu boost::condition.

Problem ucztujących filozofów

Problem 5-filozofów to klasyczny przypadek prezentacji problemu synchronizacji pracujących współbieżnie procesów. Teoretyczne wyjaśnienie zakleszczania i uniemożliwienia innym jednostkom korzystania z zasobów, przy założeniu takim, że każdy z filozofów bierze po jednym widelcu, a następnie próbuje zdobyć drugi. Zakłada się też, że jedzenie jednym widelcem jest niemożliwe.

Listing 3. Kod funkcji - pthreads.[9]

```
void *funkcja_pieciu(int n) {
    printf ("Filozof %d mysli\n", n);
    pthread_mutex_lock(&widelec[n]);
    pthread_mutex_lock(&widelec[(n + 1) % 5]);

    printf ("Filozof %d je\n", n);
    usleep(3);

    pthread_mutex_unlock(&widelec[n]);
    pthread_mutex_unlock(&widelec[(n + 1) % 5]);

    printf ("Filozof %d skonczyl jesc\n", n);
    return(NULL);
}
```

Poza opisywaną do tej pory funkcją pthread_mutex_lock(), która w tym przypadku ma za zadanie blokować zmienną widelec (lewy oraz prawy) tak by w danym momencie filozof mógł korzystać z dwóch widelców jednocześnie, w funkcji została użyta funkcja usleep(), która umożliwia zatrzymanie działania wątku na pewien określony czas przesłany w parametrze tej funkcji w milisekundach.

Listing 4. Kod funkcji main - pthreads.[11]

```
int main() {
    int i, j;
    for( j = 1; j <= POSILKI; j++) {
        for(i = 0; i < 5; i++)
            pthread_mutex_init(&widelec[i],NULL);
        for(i = 0; i < 5; i++)
            pthread_create(&filozofowie[i],NULL,(void
*)funkcja_pieciu,(void *)i);
        for(i = 0; i < 5; i++)
            pthread_join(filozofowie[i],NULL);
        for(i = 0; i < 5; i++)
            pthread_mutex_destroy(&widelec[i]);
    printf ("Filozof %d skonczyl %d posilek\n", i, j);
 }
```

Do tworzenia wątków w bibliotece Posix Threads służy funkcja pthread_create(), która w argumencie przyjmuje referencję do zmiennej pthread_t oraz funkcję, która ma zostać wywołana w tworzonym wątku.

Listing 5. Kod funkcji - boost.[5]

```
class Filozof {
private:
    int numer;
    int lewyWidelec;
    int prawyWidelec;
    int zjedzonychPosilkow;
public:
    Filozof(int n) : numer(n), zjedzonychPosilkow(0) {
    lewyWidelec = numer;
    prawyWidelec = (numer+4) % FILOZOFOWIE;
    }
    void operator()() {
        int n = FILOZOFOWIE;
        cout << "Filozof " << numer << " mysli\n";
        widelec[n].lock();
        widelec[(n + 1) % 5].lock();
        cout << "Filozof " << numer << " je\n";
        Sleep(3);
        widelec[n].unlock();
        widelec[(n + 1) % 5].unlock();
    }
};
```

Przy pomocy języka C++ stworzona została klasa Filozof posiadająca prywatne pola i przeciążony konstruktor. Posiada również przeciążony operator w klasie wykonujący operację jak zwykła funkcja. Znajduje się tu również funkcja Sleep działająca podobnie do funkcji usleep z poprzedniego przykładu.

Listing 6. Kod funkcji main - boost.[4]

```
int main() {
    thread_group filozofowie;
    int i, j;
    for( j = 1; j <= POSILKI; j++) {
        for(i = 0; i < 5; i++)
            boost::lock_guard<boost::mutex> lock{widelec[i]};
        for(i = 0; i < 5; i++){
            filozofowie.create_thread(Filozof(i));
            filozofowie.join_all();
            cout << "Filozof " << i << " skonczyl " << j << "
posilek\n";      }
```

Wątki w bibliotece Boost są reprezentowane przez klasę boost::thread. Przeciążony konstruktor klasy wątku pobiera jako parametr adres funkcji, która ma zostać wykonywana w tworzonym wątku. Przykład utworzenia wątku: boost::thread nazwa_watku(&nazwa_funkcji);

Funkcja main() jest wykonywana w głównym wątku programu. Po utworzeniu grupy wątków przy pomocy klasy boost::thread_group, w pętli tworzy wątki przy pomocy funkcji create_thread(). W programie działa już niezależnie pięć wątków, które reprezentują 5 – filozofów.

Stworzone wątki zostają dołączone do wątku głównego w funkcji main() za pomocą funkcji boost::thread::join_all(). Wszystkie wątki kończą swoją pracę.

6. Wnioski

Przedstawione dwie biblioteki programistyczne Posix Threads dla języka C oraz Boost dla języka C++ mimo różnic w implementacji mają wiele wspólnych cech i funkcji za sprawą których możemy tworzyć, synchronizować wątki, a w konsekwencji rozwiązywać problemy informatyczne. Posiadają też własne funkcje niedostępne w drugiej bibliotece.

Zaletą języka C w stosunku do C++ jest jego prostota. Zyskał on popularność na systemach wbudowanych dzięki wydajności działania i dostępności operacji niskopoziomowych. Dodatkowo biblioteka Pthreads posiada większą ilość dostępnych funkcji.

C++ - jest językiem obiektowym ogólnego przeznaczenia, dynamicznie rozwijanym, stanowiącym rozszerzenie popularnego języka C. Podejście obiektowe jest trudniejsze, ale oferuje nowe funkcjonalności m.in. hermetyzację danych. Biblioteka Boost jest popularna w programowaniu równoległym.

Programowanie wielowątkowe jest kluczem do tworzenia programów, aplikacji działających szybko i wydajnie na urządzeniach. Wybór języka wraz z biblioteką zależy od przeznaczenia programów jak również preferencji czy doświadczenia programisty właściwie w analogiczny sposób jak odpowiedź na pytanie czy lepiej uczyć się języka C czy C++. Najlepiej znać oba standardy. Podobnie jak biblioteki Posix Threads i Boost. Warto też zauważyć, że często programowania wielowątkowego uczy się w oparciu o standardy OpenMP i MPI. Niewątpliwie obie biblioteki zawierają bardzo bogaty zestaw gotowych i podobnych funkcji pozwalających na efektywne tworzenie aplikacji wielowątkowych .

Literatura

[1] Stephen Prata, Język C. Szkoła programowania. Wydanie VI, Helion, 2016.

[2] Alex Allain, C++. Przewodnik dla początkujących, Helion, 2014.

[3] Dick Buttlar, Jacqueline Farrell, Bradford Nichols, Pthreads Programming A POSIX Standard for Better Multiprocessing, Wydawnictwo O'Reilly Media, 2013.

[4] Douglas C. Schmidt, Stephen D. Huston, C++ Network Programming Volume 1 Addison-Wesley Professional; 1 edition, 2001.

[5] Boris Sch Ling, The Boost C++ Libraries, Wydawnictwo XML Press, 2011.

[6] Bjarne Stroustrup, "Why C++ is not just an Object-Oriented Programming Language", Sigplan, 1995.

[7] LukasEinkemmer, A resistive magnetohydrodynamics solver using modern C++ and the Boost library, Elsevier B.V., 2016.

[8] https://theboostcpplibraries.com/introduction[20.11.2018]

[9] https://mortoray.com/2011/12/16/how-does-a-mutex-work-what-does-it-cost/[20.11.2018]

[10] Bil Lewis, Multithreaded Programming With PThreads Prentice Hall; 136th ed. edition (9 December 1997)

[11] David R. Butenhof Programming with POSIX Threads, Addison Wesley Longman, Inc,1997

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 54-61
Wysłane: 2018-11-28
Przyjęte: 2018-12-03

Wykorzystanie postprocesingu i jego wpływu na wydajność renderowania w silniku Unreal Engine 4

Eryk Puławski*, Marcin Tokarski
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Dzisiejszą grafikę 3D ciężko wyobrazić sobie bez efektów znanych nam z filmowych produkcji czy wysokobudżetowych gier wideo. Jednak użycie takich upiększeń wiąże się ze zwiększonym zapotrzebowaniem na moc obliczeniową. Żeby sprostać oczekiwaniom deweloperów i konsumentów na rynku aplikacji 3D powstały narzędzia do ich tworzenia w postaci kompleksowych silników. Jednym z tych rozwiązań jest silnik Unreal Engine. W poniższym artykule badano wpływ na wydajność wspomnianych efektów - postprocesów. W tym celu przygotowano testową scenę i dwa scenariusze testów.

Słowa kluczowe: postprocess; Unreal Engine; 3D

*Autor do korespondencji.

Adres e-mail: eryk.pulawski@pollub.edu.pl

The use of postprocessing and its impact on rendering performance in the Unreal Engine 4

Eryk Puławski*, Marcin Tokarski
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Today's 3D graphics are hard to imagine without the effects known to us from film productions or high-budget video games. However, the use of such embellishments is associated with an increased demand for computing power. In order to meet the expectations of developers and consumers on the 3D applications market, tools for their creation in the form of comprehensive engines were created. One of these solutions is the Unreal Engine. The following article examines the impact on the efficiency of these effects - postprocesses. For this purpose, a test stage and two test scenarios were prepared.

Keywords: postprocess; Unreal Engine; 3D

*Corresponding author.

E-mail addresses: eryk.pulawski@pollub.edu.pl

1. Wstęp

Tworzenie grafiki komputerowej 3D rozwija się nieustannie od lat 60-tych [1]. Na urządzeniach domowych zagościła już w końcówce lat 70 na przykład na komputerach Apple II [1]. Tworzenie takiej grafiki możemy podzielić na trzy podstawowe fazy:

- modelowanie – proces nadawania kształtu obiektu w postaci modelu matematycznego na komputerze,
- układ i animację – proces rozmieszczenia obiektów w przestrzeni (scenie) oraz ich poruszanie,
- renderowanie – proces komputerowego wyliczania obrazu 2D (pikseli) na podstawie ustalonych modeli 3D i kamery (projekcji) i np. typów materiałów, światła.

Rynek gier komputerowych przyczynił się do szybszego rozwoju grafiki 3D. Gracze pragnęli coraz to ładniejszej lub realistyczniejszej grafiki, szybkości działania czy dostępności produkcji w 3D. Dlatego też studia deweloperskie czy sami deweloperzy od początku istnienia tej dziedziny prześcigali się technologicznie, tworząc coraz to nowsze rozwiązania, by sprostać oczekiwaniom odbiorców. Jednak wraz z kolejnymi latami rosła też złożoność i skomplikowanie programów generujących grafikę i coraz trudniej było jednostkom nadążyć za trendami. Małe aplikacje renderujące rozrastały się w kompletne silniki. Taka postać rzeczy doprowadziła do sytuacji, w której stworzenie własnego rozwiązania jest nieopłacalne bądź nawet niekiedy niewykonalne. Na takie przedsięwzięcie decydują się już tylko nieliczni, posiadający ogromny kapitał.

Sytuacja na rynku silników 3D zdążyła się już ustabilizować. Część z największych studiów zmieniła podejście i zaoferowała darmowy dostęp do swoich produktów. Większość rynku posiadają silniki Unreal Engine, Unity i CryEngine [2].

W celu utworzenia fotorealistycznej czy ładnej oprawy graficznej kluczowym procesem są efekty nakładane po wstępnej fazie renderowania. Te metody (postprocessing) oferowane są współcześnie przez wszystkie silniki 3D.

Celem artykułu jest zmierzenie wpływu postprocesingu na wydajność renderowania wykorzystując silnik Unreal Engine 4. Analiza wykonywana jest używając narzędzi profilujących układy CPU i GPU skupiając się na czasie renderowania (wykonania) poszczególnych funkcji i obliczeń.

2. Obiekt badań

Badanie skupia się na oferowanych przez Unreal Engine 4 efektach postprocessingu. Silnik zapewnia implementację wielu różnych algorytmów i efektów, jednak w artykule skupiono się na kilku z nich opisanych w podrozdziałach poniżej.

2.1. Unreal Engine 4

Silnik Unreal Engine 4 to kompletny zestaw narzędzi deweloperskich stworzony dla wszystkich pracujących z technologiami 3D. Pozwala na uzyskanie fotorealistycznych obrazów renderowanych w czasie rzeczywistym (generowanych i wyświetlanych w tym samym czasie tworząc wrażenie animacji). Zachowuje przy tym relatywnie wysoką wydajność w porównaniu z podobnymi narzędziami.

Każdy deweloper korzystający z tego silnika ma pełny dostęp do kodu źródłowego, który może dowolnie modyfikować oraz dodawać nowe funkcjonalności wykorzystując język C++. Programowanie wizualne za pomocą szkiców, zwanych blueprintami, w Unreal Engine 4 to elastyczny i potężny interfejs oparty na węzłach do tworzenia elementów rozgrywki [3]. Ten interfejs zapewnia projektantom i artystom możliwość programowania ich własnych gier wewnątrz edytora nie pisząc ani jednej linii kodu.

Blueprinty skonstruowane są na zasadzie diagramów, które zawierają liczne węzły połączone ze sobą. Połączenia te definiują ich działanie.

2.2. Post-processing

Nazwa post-processing używana jest w branżach wideo jako metody ogólnego przetwarzania obrazu w celu polepszenia jakości, ale także w renderingu 3D jako dodatkowe efekty, które można uzyskać już po zasadniczej fazie renderowania.

W przypadku gier wideo zamiast renderowania sceny bezpośrednio na ekran, obiekty najpierw renderowane są do bufora znajdującego się w pamięci karty graficznej. Następnie shadery (pixel i czasami vertex) używane są do zastosowania filtrów na obrazie w buforze. Dopiero wtedy następuje wyrenderowanie na docelowy ekran. Niektóre z postprocesów wymagają jednego "przejścia" po obrazie, inne zaś wielokrotnego. Post-processing pozwala na wykorzystanie takich efektów, które muszą mieć informacje o całym obrazie (standardowo obiekty renderowane są w izolacji od siebie).

W branży gier używa się kilkudziesięciu zdefiniowanych efektów, jednak ich liczba jest nieograniczona i każdy może wymyślić i stworzyć swoje własne. Do najpopularniejszych zaliczają się: Anti-Aliasing, Auto-Exposure, Blending, Color Grading, Depth of Field, Lens Flare, Panini Projection, Fog, Dithering, Texture Filtering, Bloom.

2.3. Antyaliasing

Antyaliasing jest to technika zmniejszania efektów schodkowania obrazu. W renderingu rezultatem tego działania jest wrażenie gładkich krawędzi wyświetlanych na ekranie rastrowym. Problem aliasingu dotyczy krzywych, pionowe i poziome linie są renderowane bez potrzeby wygładzania. W grafice wektorowej i na odpowiednich ekranach efekt schodkowania nie występuje i nie zachodzi potrzeba wykorzystywania tego efektu.

Antyaliasing w uproszczeniu może polegać na zmianie koloru pikseli bezpośrednio sąsiadujących z krzywą na kolor proporcjonalny do ich odległości od tej krzywej. Taka linia obiektu będzie wydawać się gładka, jednak stwarzała będzie wrażenie rozmazanej. Przykład efektu antyaliasingu widoczny jest na Rys. 1.

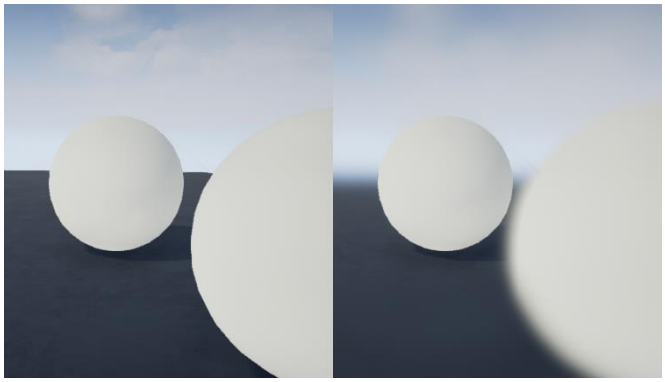
Rys. 1. Efekt antyaliasingu – wył./wł

2.4. Depth of field

W optyce głębia ostrości (ang. depth of field) jest to zakres odległości, w którym wszystkie obserwowane obiekty wydają się być ostre. Duża głębia ostrości charakteryzuje się tym, że większość elementów na zdjęciu jest ostra, natomiast mała głębia najczęściej spotykana jest podczas fotografii portretowych eksponując jeden obiekt [4].

Depth of Field w silniku Unreal Engine 4 nakłada rozmycie na obiekty, które są w określonej odległości przed lub za punktem centralnym. Ten efekt używany jest aby skupić uwagę obserwatora w jednym miejscu i sprawić żeby wyrenderowana scena przypominała fotografię lub film. Na Rys. 2 znajduje się zrzut ekranu z widocznym efektem DoF. Silnik zapewnia trzy metody do nakładania tego efektu:

- Bokeh DoF,
- Gaussian DoF,
- Circle DoF.



Rys. 2. Efekt Depth of Field – wył./wł

2.5. Lens flare

W optyce flara (ang. lens flare) odnosi się do zjawiska, w którym światło jest rozpraszane w systemie soczewek, gdy obiektyw skierowany jest na jasny punkt. Efektem wizualnym tego zjawiska są najczęściej różnokolorowe, przeźroczyste okręgi. Obiektywy z dużą liczbą elementów, takich jak zoomy, wykazują większą flarę.

We wszelkiego rodzaju produkcjach filmów animowanych, generowanych komputerowo efektach specjalnych oraz grach komputerowych powszechnie wykorzystywany jest efekt Lens Flare [5]. To technika oparta na obrazie, która symuluje rozpraszanie światła podczas oglądania jasnych obiektów z powodu niedoskonałości obiektywów aparatu. Ten efekt zobrazowany jest na Rys. 3.

Rys. 3. Efekt Lens Flare – górwa wył., dół wł

2.6. Bloom

Jest to efekt grafiki komputerowej wykorzystywany w grach wideo do odtworzenia artefaktu widzialnego na obrazie z rzeczywistych kamer. Wynikiem tego efektu jest obraz z obszarem jasnego światła, który wydaje się rozlewać poza jego faktyczne ramy. Tworzą w ten sposób iluzję wyjątkowo jasnego światła, które przytłacza aparat lub oko.

W silniku Unreal Engine 4 do uzyskania efektu Bloom wykorzystywane jest rozmycie Gaussa. Dla lepszej jakości takiego efektu łączonych jest kilka takich rozmyć z różnymi promieniami [6]. Przykład takiego efektu widoczny jast na Rys. 4.

Rys. 4. Efekt Bloom – wył./wł

3. Metodyka badawcza

W celu poprawnego zmierzenia wpływu postprocessingu na wydajność, wyniki pomiarów czasów nakładania poszczególnych efektów należy przedstawić w milisekundach, ze względu na:

- pomiary i porównywanie – ciężko porównać poszczególne efekty w FPS (ang. Frames per second);
- FPS jest końcowym wynikiem – nie nadaje się do mierzenia pojedynczych funkcji, jest odpowiedni do całościowych testów wydajności;
- trudność w wyrażeniu kosztu efektu w FPS;
- koszt niektórych efektów jest stały.

3.1. Platforma testowa

Wpływ postprocessingu na wydajność zbadany został na dwóch komputerach klasy PC działających na systemie Windows 8.1. Ich główne podzespoły ukazane zostały w Tabeli 1.

Tabela 1. Użyty sprzęt do testów

Lp.	CPU	GPU	RAM	Typ dysku
1.	i5-2500k 4 x 4,5GHz	GTX 960 4GB	8GB	SSD
2.	Pentium G4560 4 x 3,5GHz	Radeon R7 200 Series 2GB	8GB	HDD

Specyfikacja pierwszego komputera odpowiada mniej więcej przeciętnemu sprzętowi gracza w 2017-2018 roku, który określony jest w Tabeli 2. Drugą natomiast sklasyfikować można poniżej przeciętnej ze względu na jednostkę procesora graficznego.

Tabela 2. Najpopularniejsze podzespoły graczy na podstawie ankiety Steam [7]

Element	Most popular	Percentage
GPU	NVIDIA GeForce GTX 1060	11.89%
	NVIDIA GeForce GTX 1050 Ti	8.06%
	NVIDIA GeForce GTX 960	5.16%
VRAM	1024 MB	24.92%
	2047 MB	23.57%
	4095 MB	17.43%
RAM	8 GB	38.97%
	12 GB and higher	36.67%
	4 GB	11.51%
CPU speed	3.3 Ghz to 3.69 Ghz	22.75%
	3.0 Ghz to 3.29 Ghz	18.05%
	2.3 Ghz to 2.69 Ghz	15.31%
Physical CPUs	4 cpus	60.51%
	2 cpus	31.40%
	6 cpus	4.15%

Wykorzystane podzespoły nie są obecnie najwydajniejszymi podzespołami na rynku, jednak ich wydajność nie wpływa na wnioski z wyników badań.

3.2. Aplikacja testowa

W celu przetestowania wpływu wydajności przygotowano przykładową scenę przedstawiającą fragment lasu. Wykorzystane assety (modele 3D, aktorzy, tekstury itp.) przedstawiono w Tabeli 3.

Tabela 3. Liczebność assetów i ich typy

Nazwa	Typ	Liczebność
Kwiaty – Jaskier	Foliage, StaticMesh	312
Kępa trawy		488
Kwiaty - Krwawnik		65
Mirt bagienny	StaticMeshActor	9
Drzewa		14
Wielka skała - Wulkaniczna		2
Średni głaz		6
Platforma	StaticMeshActor	1
Martwe liście		39
Paproć		4
Gwiazdnik		6
Podłużna skała		9
Pionowa skała		2
Równinny głaz		2
Skała górska		27
Skała rzeczna		1
Kłoda		1
Pień		1
Światło puntowe	PointLight	6
Sfera odbić	SphereReflectionCapture	1
Kierunek wiatru	WindDirectionalSource	1

Na utworzoną scenę zostało nałożonych większość efektów postprocesowych, które oferuje silnik Unreal Engine. Łączna liczba trójkątów siatki wszystkich umieszczonych modeli na scenie wynosi 1761725. Umieszczone assety na scenie tworzą teren o wymiarach około: wysokość 50m, szerokość 30m, długość 30m.

3.3. Narzędzia pomiarowe

Profilowanie to mierzenie czasu potrzebnego do wykonania bloku instrukcji lub funkcji np. renderowania obiektów przeźroczystych. Do zmierzenia tego wpływu wykorzystać można wbudowane w Unreal Engine narzędzie GPU Visualizer oraz Session Fronted. Unreal umożliwia zmierzenie czasów wykonania poszczególnych etapów renderowania poprzez umieszczanie znaczników czasu (GPU Timestamps). Profilowanie przydatne jest podczas tworzenia każdej aplikacji 3D, ze względu na możliwość:

- porównania różnych rozwiązań;
- znalezienia wąskich gardeł;
- uniknięcia często niepotrzebnej, wczesnej optymalizacji.

GPU Visualizer pokazuje statystyki podzielone na kilka kategorii takich jak ShadowDepths (tworzenie mapy cieni), czy Lights (obliczenia światła). Jego jedynym wyraźnym ograniczeniem jest to, że nie zapewnia statystyk dla specyficznych obiektów lub niektórych świateł (niezwiązanych z cieniem). Żeby wywołać okno Visualizera można skorzystać ze skrótu klawiszowego ”Ctrl + Shift + ,”, w którym to ukazany jest przekrój pojedynczej klatki obliczanej przez GPU w postaci wykresu segmentowego pokazującego poszczególne kategorie oraz bardziej dokładnej listy kategorii wraz z ich efektami w scenie.

Narzędzie Session Frontend jest stworzone do uproszczania i przyspieszenia procesu tworzenia gier komputerowych. Pozwala na zdalne monitorowanie aktualnie aktywnych sesji gier. Wykorzystywane jest do zidentyfikowania możliwych źródeł spowolnień w grze.

3.4. Scenariusze testowe

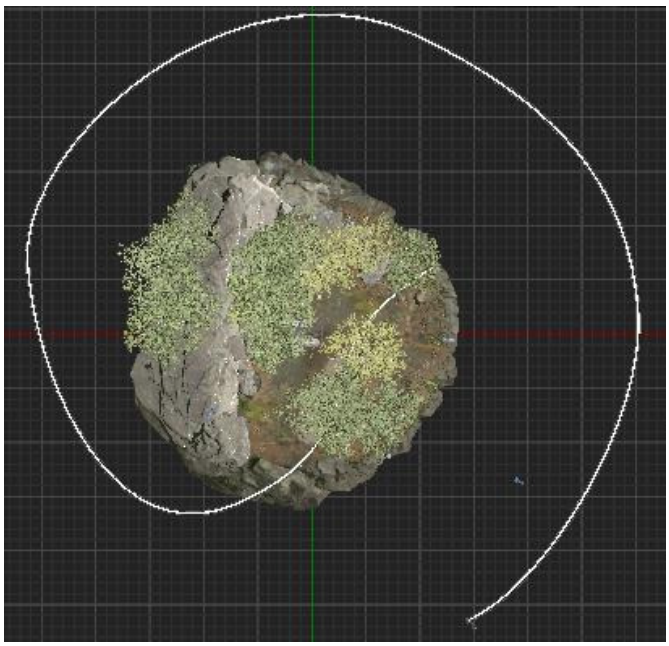
Przygotowane zostały dwa odrębne scenariusze testowe do przeprowadzenia na platformach testowych. Pierwszy nakierowany jest na zbadanie wpływu poszczególnych postprocesów na wydajność. Drugi zaś skupia się na zmierzeniu wpływu wszystkich efektów postprocesowych na długość renderowania pojedynczej klatki.

Na scenariusz nr 1 składają się następujące kroki:

1) rozstawienie kamer w różnych miejscach na scenie,
2) ustawienie aktualnego widoku na widok z n-tej kamery,
3) ustawianie rozdzielczości okna,
4) uruchomienie sceny w nowym oknie,
5) uruchomienie GPU Visualizer – pomiar z 1 klatki,
6) powtórzyć kroki od punktu 2 do 5 dla wszystkich badanych kamer i rozdzielczości,
7) zebranie danych do analizy.

W przeciwieństwie do poprzedniego testu – w którym badane są poszczególne klatki z różnych kamer – następny

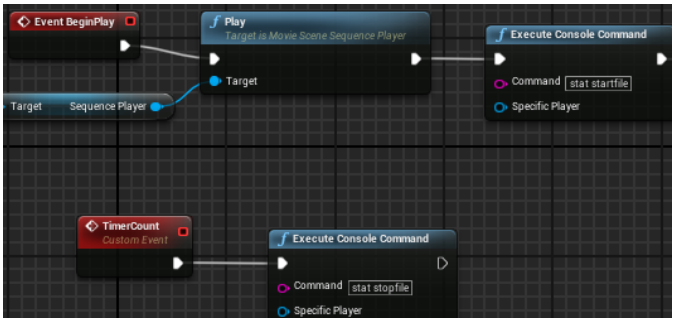
test obejmuje ciągły zapis danych przez czas działania aplikacji. W trakcie testu kamera porusza się dynamicznie po przygotowanym torze wokół sceny. Tor ruchu kamery widoczny jest na Rys. 5. Zebrane dane są w postaci pliku o rozszerzeniu UE4STATS zawierającego tekstowe wyniki dla poszczególnych klatek. Interpretacja rezultatów możliwa jest np. za pomocą narzędzia Session Frontend.



Rys. 5. Platforma testowa do scenariusza nr 2

Kroki scenariusza nr 2 przedstawiają się następująco:

1) ustawienie aktualnego widoku na widok z kamery umieszczonej na szynach,
2) ustawienie rozdzielczości okna,
3) uruchomienie sceny w nowym oknie,
4) powtórzyć poprzednie kroki dla wszystkich badanych rozdzielczości,
5) analiza plików z wynikami.



Rys. 6. Zrzut ekranu z Level Blueprint

Na Rys. 6 widoczny jest fragment blueprinta wykorzystanego w drugim scenariuszu, który po uruchomieniu aplikacji rozpoczyna zapis danych do analizy z ustalonej sekwencji ruchu kamery i po określonym czasie zatrzymuje logowanie.

4. Wyniki

Do realizacji pierwszego scenariusza wykorzystano cztery testowe kamery. Zrzuty ekranów z widoku tych kamer znajdują się na Rys. 7. Wyniki z pierwszego testu znajdują się w Tabeli 4 i Tabeli 5.

Rys. 7. Widok z czterech testowanych kamer

Wybrano następujące rozdzielczości testowe:

- HD - 1280x720 px,
- FHD - 1920x1080 px.

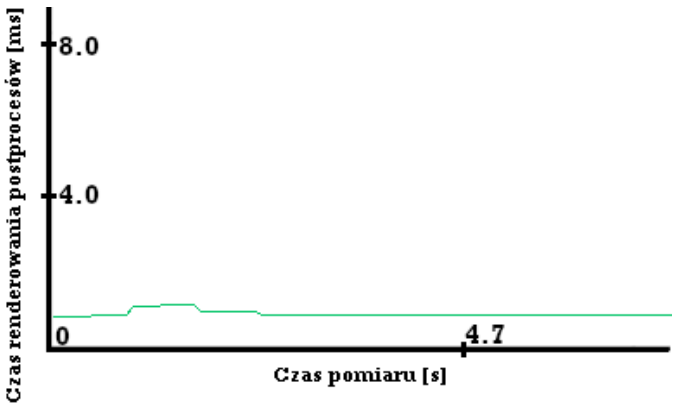
Tabela 4. Czas renderowania w rozdzielczości HD w ms – 1280x720 px

Nazwa procesu	**Kamera**							
	I	II	III	IV	I	II	III	IV
Antyaliasing	0,56	0,54	0,57	0,54	1,55	1,54	1,55	1,55
Bloom	0,03	0,03	0,03	0,03	0,08	0,09	0,09	0,09
Depth of Field	0,31	0,32	0,27	0,32	0,77	0,77	0,77	0,77
Lens Flare	0,16	0,16	0,16	0,16	0,36	0,36	0,36	0,36
Narzut postprocesów	1,56	1,56	1,56	1,6	3,79	3,8	3,79	3,79
Czas wyrenderowania klatki	25,1	26,55	16,8	24,72	46,98	54,36	35,03	48,46
	PC 1				PC2			

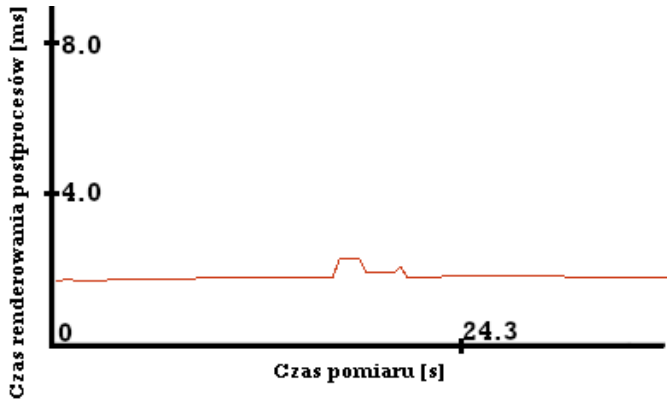
Tabela 5. Czas renderowania w rozdzielczości FHD w ms – 1920x1080 px

Nazwa procesu	Kamera							
	I	II	III	IV	I	II	III	IV
Antyaliasing	0,93	0,93	1	0,93	2	2,03	2,03	2,03
Bloom	0,04	0,04	0,04	0,04	0,12	0,13	0,12	0,12
Depth of Field	0,7	0,69	0,59	0,69	1,29	1,29	1,29	1,29
Lens Flare	0,35	0,35	0,35	0,35	0,79	0,79	0,79	0,79
Narzut postprocesów	3,5	3,53	3,52	3,6	7,12	7,17	7,17	7,15
Czas wyrenderowania klatki	39,88	41,66	25,24	38,88	76,5	82,81	51,28	73,67
	PC 1				PC2			

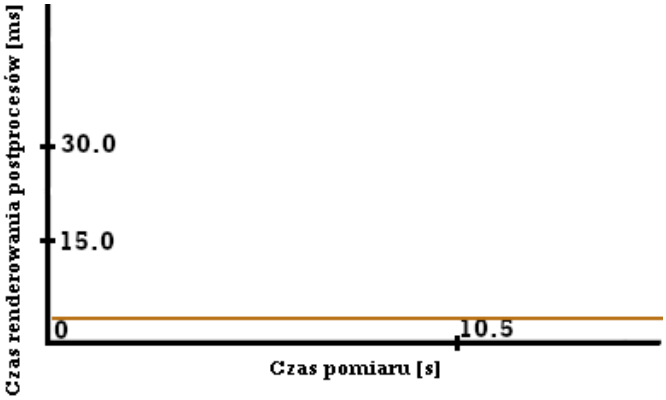
Do realizacji drugiego scenariusza wykorzystano jedną kamerę poruszającą się po określonym torze ruchu. Dane zbierane były przez 25 sekund. Wyniki z drugiego testu znajdują się w Tabeli 6 na podstawie Rys. 8, Rys. 9, Rys. 10 oraz Rys. 11.



Rys. 8. Czasy renderowania postprocesów na PC1 w rozdzielczości HD w ms – 1280x720 px



Rys. 9. Czasy renderowania postprocesów na PC1 w rozdzielczości FHD w ms – 1920x1080 px



Rys. 10. Czasy renderowania postprocesów na PC2 w rozdzielczości HD w ms – 1280x720 px

Rys. 11. Czasy renderowania postprocesów na PC2 w rozdzielczości FHD w ms – 1920x1080 px

Tabela 6. Czasy renderowania poszczególnych klatek - drugi scenariusz

	PC1		PC2	
	HD	FHD	HD	FHD
Min	0,68	1,30	2,95	6,72
Max	1,11	2,28	3,84	8,53
Średnio	0,87	1,74	3,31	7,59

Ilość zgromadzonych danych jest względnie niewielka lecz nie przeszkadzają w zauważeniu pewnych tendencji czy korelacji. Dane te pozwalają na analizę wpływu na wydajność renderowania wybranych postprocesów, która to znajduję się w kolejnym rozdziale.

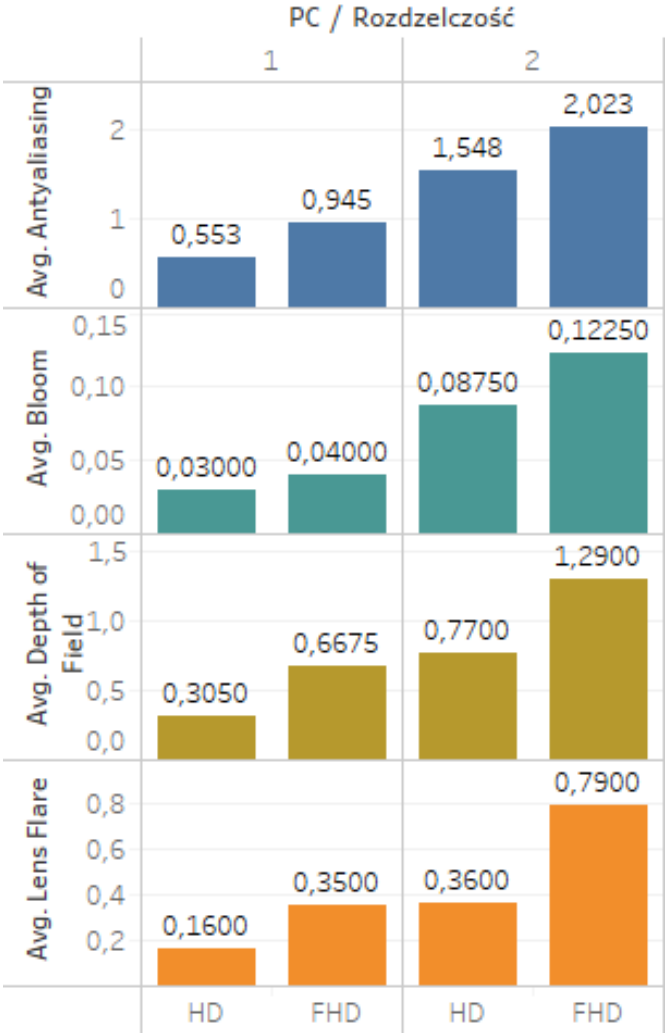
5. Analiza wyników

Dane zebrane są dla 4 wymiarów, z uwagi na chęć zbadania korelacji między nimi. Wymiary te stanowią: sprzęt testowy, rozdzielczość, kamera i postprocesy. Ich liczba powoduje trudność przedstawienia wszystkich wymiarów na jednym wykresie tak aby był on czytelny. Z uwagi na to wyniki przedstawiono na kilku wykresach różnego typu skupiając się na wybranych wymiarach.

Wyniki narzutu postprocesów pokazują, że widok z kamery ma bardzo mały wpływ na czas nakładania procesów. W teorii wpływ *Antyaliasingu* powinien być zauważalny, gdy liczba assetów się zmienia.

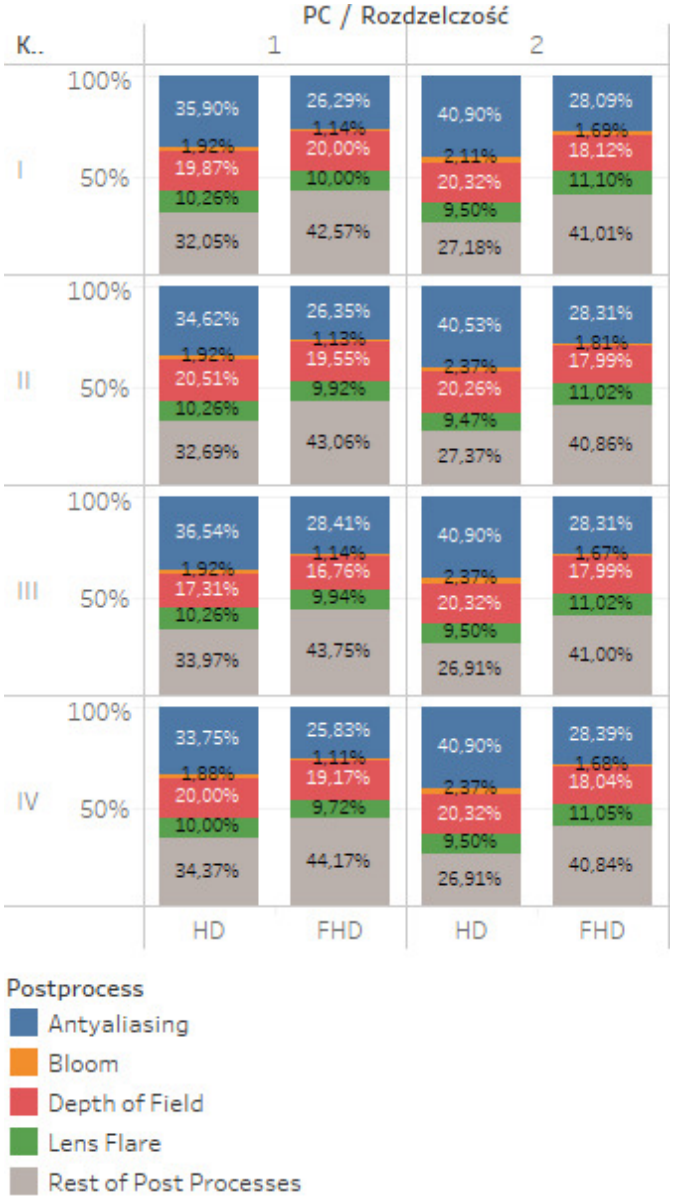
Analizując wyniki dla poszczególnych efektów faktem staje się, że ich wartości rosną wraz ze wzrostem rozdzielczości. Wynika to z liczby pikseli jakie muszą zostać przetworzone dla danego postprocesu.

Przyglądając się wykresom, będących efektem drugiego scenariusza testowego, ukazanych na Rys. 8, Rys. 9, Rys. 10 i Rys. 11 można zauważyć małe zmiany wartości czasu nakładania post procesów przez cały czas trwania. Prezentuje się to prostą linią na wykresach.



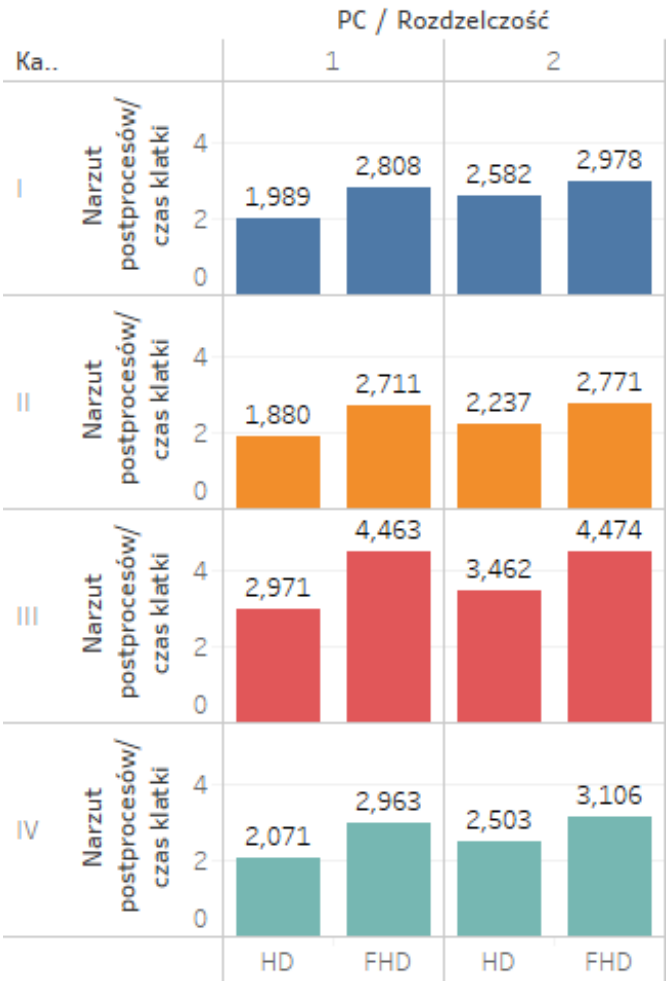
Rys. 12. Średnie czasy [ms] renderowania wybranych postprocesów

Na wykresie Rys. 12 zauważyć można wzrost średniego czasu nakładania każdego z postprocesów wraz ze wzrostem rozdzielczości. Dla *Lens Flare* i *Depth of Field* skok ten był w przybliżeniu dwa razy większy. Pozostałe dwa zanotowały mniejszy wzrost. Świadczy to o tym, że *Antyaliasing* oraz *Bloom* są mniej zależne od rozdzielczości niż pozostałe.



Rys. 13. Procentowy narzut wybranych postprocesów względem czasu renderowania wszystkich

Z wykresu na Rys. 13 można wywnioskować, że niezależnie od kamery, sprzętu testującego i rozdzielczości największy narzut na czas nakładania postprocesów ma *Antyaliasing* a najmniejszy *Bloom*. Dodatkowo tylko czas nakładania efektu *Lens Flare* nie zależy od widoku z kamery a reszta badanych zmienia się nieznacznie.



Rys. 14. Średni narzut postprocesów względem średniego czasu renderowania klatki

W procesie optymalizacji renderowania (zwiększania liczby klatek na sekundę przy zachowaniu tych samych ustawień rozdzielczości) czas poświęcony na postprocesy jest wartością szczególną, ponieważ można przyjąć iż jest stały. Oznacza to, że im więcej klatek na sekundę chcemy osiągnąć, tym większy narzut w całości będzie miał post-processing. Dla przykładu mając scenę o średnim czasie renderowania 30 FPS (co najwyżej 33,(3) ms na klatkę), gdzie post-processing zajmuje 10 ms to jego narzut wynosi około ⅓. Optymalizując scenę by działała w co najmniej 60 FPS (co najwyżej 16,(6) ms na klatkę) postprocesy będą stanowić około ⅝ całego czasu. Analogicznie dla 120 FPS postprocesy stanowią 83% potrzebnego czasu na wyrenderowanie klatki. Łatwo zauważyć, że post-processing jest elementem blokującym w procesie optymalizacji dla aplikacji dążących do działania w dużej liczbie FPS.

6. Wnioski

Na podstawie otrzymanych wyników i analizy z dwóch scenariuszy wyciągnięto następujące wnioski:

- liczba assetów widocznych z kamery nieznacznie wpływa na czas, jaki jest potrzebny na nałożenie postprocesów na poszczególną klatkę;
- im większa rozdzielczość okna tym dłuższy czas nakładania każdego z postprocesów przez GPU;
- czas nakładania postprocesów podczas działania aplikacji oscyluje wokół określonej wartości, generalizując - jest stały;
- *Antyaliasing* oraz *Bloom* są bardziej odporne na wzrost rozdzielczości. Oznacza to niewielki w stosunku do innych postprocesów wzrost czasu nakładania efektu;
- mający największy wpływ na wydajność z badanych postprocesów jest *Antyaliasing*;
- spośród badanych postprocesów najmniejszy wpływ na wydajność wykazał *Bloom*;
- post-processing jest procesem, którego nie można efektywnie zoptymalizować;
- udział postprocesów jest odwrotnie proporcjonalny do czasu renderowania klatki[ms];
- wraz ze wzrostem rozdzielczości udział czasu renderowania postprocesów w czasie całej klatki będzie zwiększał się im wydajniejsza jednostka graficzna.

Literatura

[1] Salustri F. A., „A Brief History of Computer Graphics," 30 Marzec 2018. [Online]. Available: deseng.ryerson.ca/dokuwiki/mec222:brief_history_of_computer_graphics [25.10.2018].

[2] TNW DEALS, „This engine is dominating the gaming industry right now," 24 Marzec 2016. [Online]. Available: https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/ [25.10.2018].

[3] Satheesh P.V., Unreal Engine 4 Game Development Essentials, Packt, 2016.

[4] Grey E., „Understanding Depth of Field – A Beginner's Guide," 11 Luty 2018. [Online]. Available: https://photographylife.com/what-is-depth-of-field. [22.05.2018].

[5] Underdahl K., Premiere Elements 8 For Dummies, For Dummies, 2009.

[6] Kalogirou C., „How to do good bloom for HDR rendering," 20 Maj 2006. [Online]. Available: http://kalogirou.net/2006/05/20/how-to-do-good-bloom-for-hdr-rendering/. [22.05.2018].

[7] Steam, „Ankieta dotycząca sprzętu i oprogramowania: September 2018," [Online]. Available: https://store.steampowered.com/hwsurvey/ [25.10.2018].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 62-66
Wysłane: 2018-11-30
Przyjęte: 2018-12-05

Analiza jakości interfejsu aplikacji internetowej z wykorzystaniem eye-trackingu – studium przypadku

Marcin Jusiak*, Marek Miłosz
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia przykład wykorzystania technologii eye-trackingu w badaniu jakości interfejsu aplikacji internetowej. Przedmiotem badań była część webowa systemu FastCat, wspomagającego obsługę miejskich wyścigów rowerowych. Wykorzystana metodyka badawcza polegała na realizacji eksperymentu z użytkownikami, w trakcie którego wykonywane były opracowane typowe scenariusze wykorzystania systemu. Działania użytkowników były rejestrowane z użyciem technologii eye-trackingu i zostały poddane szczegółowej analizie. Sześcioosobowa grupa badawcza posiadała profil analogiczny do potencjalnych użytkowników aplikacji. W końcowej części artykuły przedstawiono rezultaty badań ilościowych, wykaz wykrytych błędów w interfejsie, a także wnioski ukierunkowane na poprawę jego jakości.

Słowa kluczowe: eye-tracking; aplikacja; jakość interfejsu

*Autor do korespondencji.

Adres e-mail: marcin.jusiak@pollub.edu.pl

Analysis of the quality of web application interface using eye-tracking – a case study

Marcin Jusiak*, Marek Miłosz
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents an example of the use of eye-tracking technology in examining the quality of the web application interface. The subject of the research was the web part of the FastCat system, supporting the service of urban cycling races. The research methodology based on the implementation of an active experiment with users, using typical scenarios of system developed. User activities were recorded using eye-tracking technology and were subjected to detailed analysis. The six-person research group had a profile analogous to potential users of the application. The final part of the article presents the results of quantitative research, a list of detected errors in the interface, as well as conclusions aimed at improving its quality.

Keywords: eye-tracking; application; interface quality

*Corresponding author.

E-mail addresses: marcin.jusiak@pollub.edu.pl

1. Wstęp

W dzisiejszych czasach ciężko wyobrazić sobie brak możliwości korzystania z dostępu do Internetu. Szybko rosnąca popularność sieci komputerowych spowodowała możliwość ich wykorzystywania w wielu dziedzinach życia. Rozwój technologiczny, spowodowany coraz częstszym stosowaniem różnorodnych urządzeń, takich jak smartfony czy laptopy, wymusza na programistach aplikacji webowych tworzenie interfejsów responsywnych, czyli dostosowanych do różnych rozdzielczości ekranu w zależności od urządzenia [1, 2]. Główny nacisk położony jest na prostotę, przejrzystość aplikacji i ogólnie pojętą użyteczność [1, 3]. W celu sprawdzenia poprawności utworzonego interfejsu zostało opracowanych bardzo wiele metod badawczych. Metody te można podzielić wg różnych kryteriów na [4]:

automatyczne i manualne,
z udziałem użytkownika i bez takowego.

W trakcie badań jakości eksperymentu używane są różnego rodzaju techniki, takie jak [4]: testowanie, inspekcja, wywiad, modelowania analityczne czy też symulacja.

Jedną z najbardziej efektywnych metod analizy jakości interfejsu aplikacji jest eksperyment badawczy z udziałem użytkowników, realizowany metodą testowania z wykorzystaniem scenariuszy. W tej metodzie już grupa badawcza składająca się z 6. użytkowników jest w stanie wykryć większość problemów z interfejsem [3]. W metodzie tej, do rejestracji działań użytkowników coraz częściej używana jest technologia eye-trackingu [5, 6].

W niniejszym artykule zaprezentowano przypadek wykorzystania metody eksperymentu z użytkownikiem do badania jakości interfejsu autorskiej aplikacji webowej, będącej częścią systemu FastCat, wspomagającego obsługę miejskich wyścigów rowerowych. Przedstawiono metodę, rezultaty oraz omówiono szczegółowo wyniki zrealizowanego eksperymentu.

2. Opis obiektu badań

Badana aplikacja jest częścią systemu FastCat wspomagającego obsługę miejskich wyścigów rowerowych [7]. System składa się z aplikacji mobilnej dostępnej na

platformę Android oraz z aplikacji webowej. Badania zostały przeprowadzone na interfejsie aplikacji webowej.

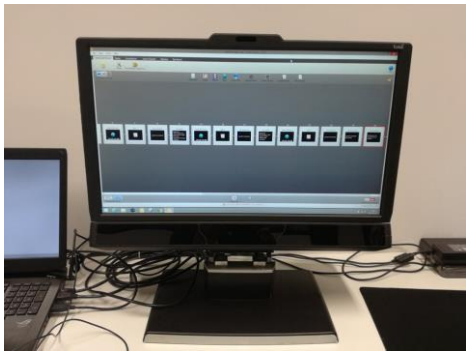
Do podstawowych funkcji, realizowanych przez webową część systemu FastCat należą:

- tworzenie, komentowanie i usuwanie postów,
- zapisywanie się na wyścig,
- organizacja nowego wyścigu,
- dodawanie punktu kontrolnego do wyścigu.

Pozostałe elementu funkcjonalności realizuje interfejs aplikacji mobilnej. System FastCat powstał w ramach dyplomowej pracy inżynierskiej [7].

3. Eye-tracking jako metoda badawcza

Okulografia (ang. eye-tracking) jest to technologia pozyskiwania danych o postrzeganiu człowieka, wykorzystująca technikę śledzenia ruchów gałki ocznej, punktów skupienia wzroku oraz rozmiaru źrenicy osoby poddanej badaniu. Technologia ta jest stosowana od ponad 100 lat w wielu dziedzinach, między innymi w psychologii, ergonomii, marketingu i interakcji człowieka z komputerem. Obecnie, eye-tracking wykorzystuje bezkontaktowe, bezpieczne i bezinwazyjne śledzenie ruchów gałki ocznej przy pomocy promieni podczerwonych [5]. Wykorzystywane są specjalne urządzenia – eye-trackery stacjonarne (rys. 1) i mobilne – oraz specjalistyczne oprogramowanie do rejestracji danych, ich przetwarzania i wizualizacji wyników.



Rys. 1. Eye-tracker Tobii TX300

4. Metodyka realizacji badań

Badania zostały wykonane zgodnie z metodyką zaprezentowaną w pracy [8]. Składa się na nią następujące etapy [8]:

- Opracowanie scenariuszy wykorzystania aplikacji i przeprowadzenia eksperymentu.
- Organizacja stanowiska badawczego.
- Dobór grupy badawczej.
- Przeprowadzenie eksperymentu i zbieranie danych.
- Obróbka rezultatów eksperymentu i ich wizualizacja.
- Analiza, dyskusja i wysnucie wniosków.

4.1. Scenariusze badawcze

W trakcie przygotowywania badań opracowane zostały następujące scenariusze (zadania do realizacji przez badanych użytkowników) wykorzystania aplikacji webowej systemu FastCat [7]:

1) Jako zalogowany użytkownik utwórz nowy post.
2) Zapisz się wyścig o nazwie: „Lublin sierpień 2018”.
3) Jako zalogowany użytkownik zorganizuj nowy wyścig o nazwie „Ulicami Lublina 2018”.
4) W wyścigu „Lublin sierpień 2018” dodaj dowolny punkt kontrolny.
5) Jako zalogowany użytkownik usuń utworzony przez Ciebie post.

Opracowany został także scenariusz badań, tj. odpowiedni tekst dla uczestników, prowadzący ich „za rękę” w trakcie realizacji eksperymentu. W ten sposób osoba nadzorująca eksperyment nie ingeruje w kolejność i sposób jego przeprowadzenia. Ma to zapewnić realizację eksperymentu w takich samych warunkach dla wszystkich uczestników.

4.2. Organizacja stanowiska badawczego

Eksperyment został przeprowadzony listopadzie 2018 r. przy wykorzystaniu eye-trackera Tobii TX300 (rys. 1) wraz z oprogramowaniem Tobii Studio Pro w Laboratorium Analizy Ruchu i Ergonomii Interfejsów (LARiEI) Instytutu Informatyki Politechniki Lubelskiej.

Na wykorzystane w badaniach stanowisko testowe składały się następujące elementy:

- Eye tracker: Tobii TX300 (Tobii AB, Sweden).
- Oprogramowanie eye-trackera: Tobii Pro Studio (wersja 3.3.2) Enterprise edition.
- Monitor TFT 23” zintegrowany z eye-trackerem,
- Urządzenie do obrazowania: Tobii User Camera v3.
- Laptop (Asus G750JX-T4191H, Procesor Intel Core i7-4700HQ, Pamięć RAM 8 GB, system Operacyjny: Windows 8.1).
- Nagrywanie dźwięku: Microphone (Realtek High Definition) – zintegrowana z płytą główną.
- Przeglądarka internetowa: Internet Explorer 11(wersja 11.0.37).

Oprogramowanie eye-trackera korzysta ze starszej wersji przeglądarki internetowej Internet Explorer, bez możliwości zmiany. W związku z tym, w celu poprawnego działania aplikacji webowej, należało poprawić niektóre moduły aplikacji, które w przeciwnym wypadku uniemożliwiałyby wykonanie zadań przez uczestników eksperymentu.

4.3. Grupa badawcza

System FastCat [7] przeznaczony jest dla młodych, aktywnych osób. Osoby te zwykle dobrze są zaznajomione z technologiami webowymi mobilnymi jako użytkownicy. Elementy te wpłynęły na dobór grupy badawczej.

W badaniu wzięło udział 6 osób w wieku 23-24 lat: 4. kobiety oraz 2. mężczyzn. Wszyscy byli studentkami i studentami Politechniki Lubelskiej.

4.4. Eksperyment, zbieranie danych i ich wizualizacja

Badania zrealizowano w pomieszczeniu laboratoryjnym LARiEI, które przypomina pomieszczenie biurowe. Poszczególni użytkownicy po krótkim przeszkoleniu realizowali samodzielnie scenariusz badań, korzystając z wydrukowanego scenariusza. Dane były rejestrowane przy pomocy stanowiska badawczego (rozdz. 4.2).

Zarejestrowane w trakcie eksperymentów dane posłużyły do późniejszej analizy, która określiła:

- Parametry ilościowe wykonania poszczególnych zadań (czas realizacji zadania, liczba fiksacji na poszczególnych fragmentach strony, liczba popełnionych błędów).
- Problemy użyteczności, wykryte przez poszczególnych użytkowników.

Pozyskane dane posłużyły także do stworzenia map cieplnych punktów fiksacji wzroku na poszczególnych podstronach serwisu, które pełnią wspomagającą role w trakcie analizy problemu jakości interfejsu.

5. Rezultaty badań

W tabeli 1 przedstawiono czasy realizacji poszczególnych zadań przez użytkowników i wyniki ich podstawowej analizy statystycznej.

Tabela 1. Czasy wykonywania zadań przez użytkowników

	Czas [s]					
Zadanie/ Użytkownik	1	2	3	4	5	Suma
1	77	10	29	30	20	166
2	857	44	313	171	45	1430
3	206	16	111	76	93	502
4	426	16	233	126	38	839
5	60	14	52	22	33	181
6	102	29	48	25	17	221
Średnia	288	21.5	131	75	41	556.5
Minimum	60	10	29	22	17	166
Maksimum	857	44	313	171	93	1430

Tabela 2 zawiera dane o liczbie fiksacji wzroku użytkowników w trakcie realizacji badań. Ekran został poddany segmentacji na trzy podstawowe części: oba menu (lewe i prawe górne) i pozostałą część ekranu.

W wyniku analizy zapisów z badań określono liczbę popełnionych błędów w trakcie realizacji zadań przez poszczególnych użytkowników oraz ich typy. Rezultaty przedstawiono w tabelach 3 i 4.

Tabela 2. Liczba fiksacji wzroku poszczególnych użytkownikach na trzech segmentach ekranu

	Liczba fiksacji		
Użytkownik	Lewe górne menu	Pozostała część strony	Prawe górne menu
1	5	20	0
2	30	84	13
3	3	31	2
4	24	70	1
5	24	36	3
6	8	23	0
Średnia	15,67	44	3,17
Suma	94	264	19

Tabela 3. Liczba zarejestrowanych błędów popełnionych przez użutkowników

Zadanie/ Użytkownik	1	2	3	4	5	Suma
1	8	0	0	0	3	11
2	14	0	6	0	1	21
3	9	0	1	0	9	19
4	17	0	6	1	3	27
5	3	0	2	0	1	6
6	5	0	2	0	1	8
Suma	56	0	17	1	18	82

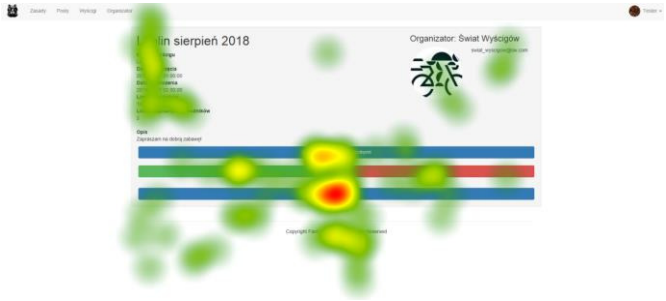
Tabela 4. Wykryte problemy użyteczności stron

Zadanie/ Użytkownik	1	2	3	4	5
1	Y				
2	X, Y, Z		X		X
3	X, Y				X, Y
4	X, Y		X		X
5					X
6	X, Y				

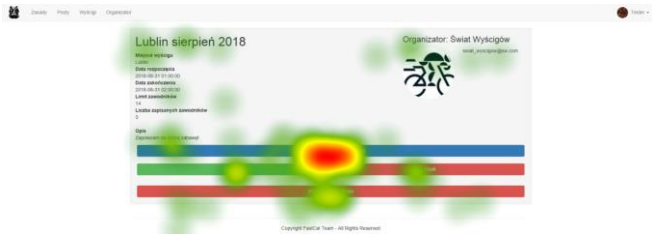
Legenda: X – problem ze znalezieniem danej funkcjonalności., Y – problem podczas wypełniania pól formularza, Z – użycie złej funkcjonalności.

Mapy cieplne, stworzone na podstawie pozyskanych danych przedstawione zostały na rysunkach 2-4.

Rys. 2. Mapa cieplna strony głównej



Rys. 3. Mapa cieplna dla zadania 2 (zapisywanie się na wyścig)



Rys. 4. Mapa cieplna dla zadania 4 (tworzenie punktu kontrolnego)

6. Analiza i dyskusja rezultatów eksperymentu

Analiza czasów wykonania poszczególnych zadań (tab. 1) wskazuje na znaczne zróżnicowanie szybkości realizacji działań w interfejsie. Różnica w łącznym czasie wykonania zadań była ponad 8. krotna, jak 166 sek. do 1430 sek.(tab. 1). Taka różnica może świadczyć o niedostosowaniu interfejsu do pewnej, niedoświadczonej grupy użytkowników.

Duże wartości średnich i maksymalnych czasów realizacji (tab. 1) dość prostych zadań (zadania numer 1 i 3) sygnalizują problemy z użytecznością podstron, realizujących te zadania. Potwierdza to analiza błędów wykrytych przez użytkowników na tych podstronach – tab. 4. Głównie były to problemy z odnalezieniem funkcjonalności na ekranie (oznaczenie X w tab. 4). Proste zadanie (numer 5, tj. usunięcie własnego postu) sprawiało również problem (tab. 4), a i czasy jego realizacji były stosunkowo duże (tab. 3).

Segmentacja ekranu pozwoliła ustalić przyczynę części błędów i zwiększonego czasu realizacji zadań. Była nią umieszczenie menu w prawej górnej części ekranu, na które to użytkownicy rzadko koncentrowali uwagę – tab. 2. Potwierdzają to mapy cieplne wykonania zadania numer 1 – rys. 2. Wyraźnie z niej można odczytać (rys. 2), że użytkownicy poszukiwali opcji „nowy post" w lewej górnej części ekranu. Zjawisko błędnego poszukiwania wystąpiło w zadaniu 2 i następnych, w których, nauczeni w trakcie realizacji poprzedniego zadania, użytkownicy poszukiwali opcji zapisywania na wyścig w różnych miejscach ekranu, rzadziej zwracając na lewy górny róg gdzie ta opcja była umieszczona – rys. 3 i 4. Wskazuje to na dość chaotyczną strukturę interfejs.

Największa liczba błędów została popełniona podczas realizacji zadania 1 (tab. 4). Błędem podczas wykonania tego zadania było m.in. wejście na złą podstronę (X), niepoprawne wypełnienie danego formularza (Y) oraz mylne przekonanie o ukończeniu zadania (Z), np. usunięcie wyścigu przy zadaniu polegającym na usunięciu postu. Sporą część popełnionych błędów stanowiło przejście na złą podstronę (X w tab. 4).

Na 92. popełnionych wszystkich błędów tylko 7. razy uczestnicy musieli ponownie wypełniać formularz. Zadania 2 oraz 4 przysporzyły najmniej problemów. Podczas ich wykonywania popełniono tylko jeden błąd. Błędy z zadania 3 wynikają z negatywnego przyswojenia/nauczenia się interfejsu strony – badani użytkownicy szukali formularza w tym samym miejscu, gdzie ostatecznie znaleźli formularz dotyczący zadania 1. Warto zwrócić uwagę na uczestnika numer 2, który mimo tego, iż najdłużej wykonywał zadania nie popełnił największej liczby błędów. Uczestnik 4 popełnił w sumie 27 błędów, uczestnik 2, tylko 21. Uczestnik 3 wykonał o 2 błędy mniej, z czasem prawie 3 razy mniejszym. Liczba popełnionych błędów ma wpływ na czas wykonania zadania. Można wywnioskować także w stosunku do użytkownika numer 2, że anomalia w czasie realizacji zadań wynika raczej ze stresu, jaki odczuwał ten uczestnik podczas realizacji eksperymentu.

7. Wnioski

Wyniki badań interfejsu aplikacji internetowej przy wykorzystaniu eye-trackingu wskazały na istnienie znaczącego problemu z brakiem ujednolicenia interfejsu aplikacji webowej systemu FastCat. Metoda ta umożliwiła zbadanie użyteczności danego interfejsu, z wykorzystaniem eksperymentu z udziałem użytkowników. Błędy, które zostały wykryte, miały znaczny wpływ na komfort korzystania (i jakość interfejsu) z aplikacji przez uczestników eksperymentu.

Podstawowym wnioskiem ze zrealizowanych badań, jest konieczność przeprojektowania interfejsu pod kątem ujednolicenia nawigacji w nim oraz usunięcia problemów z wypełnieniem formularzy (w tym, i usunięcie konieczności ponownego wprowadzania danych do formularza po wystąpieniu błędnego działania użytkownika).

Opracowana w LARiEI metodyka badań interfejsów w warunkach quasi-realnych [8] sprawdziła się w praktyce, co potwierdza duża liczba znalezionych problemów z interfejsem bardzo prostej aplikacji webowej przy stosunkowo niewielkiej próbie badawczej.

Literatura

[1] H. Bozikovic, M. Stula, Web design – Past, present and future, 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018. Proceedings, 2018, pp. 1476-1481.

[2] J. Bernacki, I. Błażejczyk, A. Indyka-Piasecka, M. Kopel, E. Kukla, B. Trawiński, Responsive Web Design: Testing Usability of Mobile Web Applications, Lecture Notes in Computer Science, vol. 9621, 2016, pp. 257-269.

[3] J. Nielsen, T. Landauer, A mathematical model of the finding of usability problems, Proceedings of ACM INTERCHI'93 Conference, April 1993, pp. 206-213

[4] M. Miłosz, Ergonomia systemów informatycznych, Poliechnika Lubelska, 2014, 126 s.

[5] J. Karn, Eye tracking in human-computer interaction and usability research: Ready to deliver the promises, The Mind's Eye: Cognitive and Applied Aspect of Eye Movement Research. Hyona, Radach & Deubel (eds.) Oxford, England, 2003, 574-576.

[6] P. Chynal, J.M. Szymanski, J. Sobecki, Using Eyetracking in a Mobile Applications Usability Testing, Lecture Notes in Computer Science, vol. 7198, 2012, pp. 178-186.

[7] M. Jusiak, J. Wrzos, B. Milaniuk, M. Garbala, System wspomagający obsługę miejskich wyścigów rowerowych, Praca dyplomowa, Politechnika Lubelska, Lublin, 2017, 94 str.

[8] M. Borys, M. Miłosz, Mobile application usability testing in quasi-real conditions a case study of a mobile eye tracker, 8th International Conference on Human System Interactions, IEEE, 2015, pp. 381-387.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 67-70
Wysłane: 2018-12-02
Przyjęte: 2018-12-07

Porównanie skuteczności wybranych algorytmów rozpoznawania twarzy w przypadku zdjęć o niskiej jakości

Jakub Gozdur*, Bartosz Wiśniewski, Piotr Kopniak
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest określenie skuteczności popularnych algorytmów rozpoznawania twarzy w przypadku zdjęć o niskiej jakości. W trakcie pracy zostały opisane podstawowe algorytmy rozpoznawania twarzy takie jak LBPH, Eigenfaces i Fisherfaces. Do przeprowadzenia badań stworzono platformę badawczą wyposażona w oprogramowanie pozwalające testować dane i zbierać wyniki. Rezultaty badań pokazała, że jedynym algorytmem nadającym się do takich rozwiązań jest LBPH. Pozostałe natomiast nie uzyskały odpowiednio wysokiego współczynnika skuteczności.

Słowa kluczowe: rozpoznawanie; twarz; LBPH; Eigenfaces; Fisherfaces

*Autor do korespondencji.

Adres e-mail: jakub.gozdur@pollub.edu.pl

Comparison of the effectiveness of selected face recognition algorithms for poor quality photos

Jakub Gozdur*, Bartosz Wiśniewski, Piotr Kopniak
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The goal of the article is to determine the effectiveness of popular face recognition algorithms for poor quality photos. Basic facial recognition algorithms such as LBPH, Eigenfaces and Fisherfaces were described during the work. A research platform equipped with software allowing to test data and collect results was created. The results of the research showed that the only algorithm suitable for such solutions is LBPH. The others, however, did not achieve a sufficiently high effectiveness factor.

Keywords: recognition; face; LBPH

*Corresponding author.

E-mail addresses: jakub.gozdur@pollub.edu.pl

1. Wstęp

W dzisiejszych czasach technologie rozpoznawania twarzy są coraz częściej wykorzystywane. Najlepszym przykładem na to są telefony komórkowe lub inteligentne domy z systemami zabezpieczeń opartymi na detekcji twarzy. Mimo to, że te technologie i narzędzia osiągnęły wysoki poziom to w dalszym ciągu nie są one doskonałe i wymagają dalszych badań.

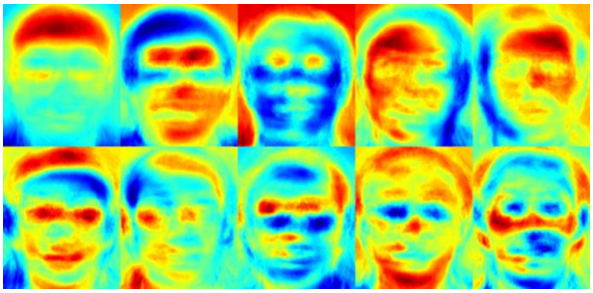
2. Metody rozpoznawania twarzy

Kierując się popularnością używanych rozwiązań wybrane zostały trzy algorytmy rozpoznawania twarzy, które jednocześnie odznaczają się stosunkową łatwością w implementacji:

- Eigenfaces;
- Fisherfaces;
- LBPH.

2.1. Eigenfaces

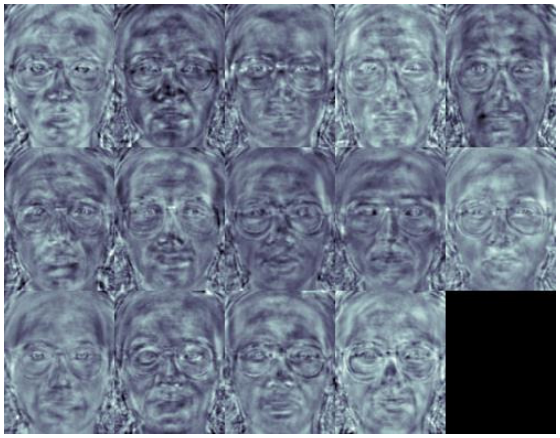
Eigenfaces bierze pod uwagę fakt, że nie wszystkie części twarzy są równie ważne lub równie przydatne przy rozpoznawaniu twarzy. Ma to odzwierciedlenie w rzeczywistości, ponieważ gdy patrzymy na kogoś, możemy tę osobę rozpoznać poprzez jej cechy szczególne, takie jak oczy, nos, policzki lub czoło; i tym jak bardzo różnią się one względem siebie. Dokładniej, koncentrujemy się na obszarach najbardziej różniących się od siebie. Na przykład, różnica pomiędzy obszarem oczu, a obszarem nosa jest zdecydowanie zauważalna. Gdy patrzymy na wiele twarzy, porównujemy je, patrząc właśnie na te regiony, ponieważ dzięki uchwyceniu maksymalnych różnic między twarzami możemy odróżnić jedną od drugiej. W ten właśnie sposób działa funkcja rozpoznawania Eigenfaces. Analizuje wszystkie obrazy treningowe wszystkich osób jako całości i próbuje wydobyć składniki, które są istotne i użyteczne, a odrzuca pozostałe. Te najbardziej znaczące cechy nazywane są głównymi komponentami [2, 4]. Rysunek 1 przedstawia cechy wyodrębnione z twarzy. Można zauważyć, że cechy charakterystyczne reprezentujące twarze są zaznaczone odpowiednimi kolorami. Stąd właśnie algorytm Eigenfaces otrzymał swoją nazwę: z jęz. niem. Eigen - swój/własny.



Rys. 1. Algorytm Eigenfaces – chechy charakterystyczne twarzy [5]

2.2. Fisherfaces

Fisherfaces uważany jest jako ulepszona wersja Eigenfaces. Poprzedni algorytm analizuje wszystkie twarze treningowe naraz i wyszukuje główne komponenty wszystkich z nich. W ten sposób nie koncentruje się na cechach, które jednoznacznie wykluczają podobieństwo jednej osoby do drugiej. Ponieważ Eigenfaces uwzględnia oświetlenie jako istotne kryterium oceny, zakwalifikuje tę zmiany jako użyteczne dla rozpoznawania twarzy, w wyniku czego może odrzucić cechy twarzy innych osób, uznając je za mało przydatne, prowadząc do błędnej klasyfikacji. Co prawda można tego uniknąć, konfigurując algorytm Eigenfaces, tak aby wyodrębnił użyteczne cechy z twarzy każdej osoby oddzielnie, zamiast ze wszystkich naraz. W ten sposób, nawet jeśli dane jednej osoby mają duże zmiany w oświetleniu, nie wpłynie to na proces wyodrębniania cech innych osób. Natomiast, algorytm rozpoznawania twarzy Fisherfaces wyodrębnia główne komponenty, które odróżniają jedną osobę od innych. W tym przypadku komponenty jednostki nie dominują (stają się bardziej użyteczne) nad innymi. Poniżej znajduje się obraz głównych komponentów wyodrębnionych za pomocą algorytmu Fisherfaces.

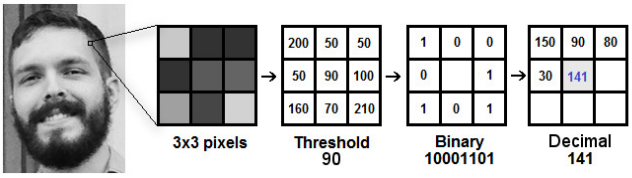


Rys.2. Algorytm Fisherfaces – cechy charatreystczne twarzy [5]

Należy zwrócić uwagę na to, że moduł FisherFaces zapobiega tylko dominacji jednej osoby, lecz jego wadą jest to, że analizując twarz brane pod uwagę jest jej oświetlenie. Wiemy natomiast, że zmienność światła nie jest użyteczną funkcją do wyodrębnienia, ponieważ nie jest częścią rzeczywistej twarzy [1, 4].

2.3. LBPH

Główna idea LBPH nie polega na oglądaniu obrazu jako całości, ale na próbie znalezienia lokalnej struktury przez porównanie każdego piksela z sąsiednimi pikselami. Proces rozpoznawania twarzy algorytmu LBPH można zademonstrować rysując okno 3 × 3 i przesuwając je po obrazie. Przy każdym przesunięciu należy porównać wartość piksela w środku z wartościami pikseli otaczających go. W przypadku gdy wartość sąsiada jest mniejsza lub równa centralnemu, przypisujemy mu 0, a w przeciwnym wypadku 1 [3]. Po posortowaniu etykiet pikseli w oknie 3 × 3 w kolejności zgodnej z ruchem wskazówek zegara, otrzymujemy wzór binarny, który jest lokalny dla określonego obszaru obrazu. Po zakończeniu procesu dla całego obrazu, otrzymujemy listę lokalnych wzorów binarnych.



Rys. 3. Local Binary Pattern Histograms Fisherfa – zasada działania [5]

Następnie, po uzyskaniu listy lokalnych wzorców binarnych, konwertuje się każdy z nich na liczbę dziesiętną a następnie tworzy się histogram wszystkich wartości dziesiętnych [4].

W celu dokładniejszego zapoznania się z zasadami działania algorytmów, odsyłamy do pozycji literaturowych znajdującymi się w bibliografii.

3. Cel i plan badań

Celem badań jest określenie skuteczności wybranych algorytmów rozpoznawania twarzy w przypadku zdjęć o niskiej jakości. Temat ten został podjęty, aby zbadać efektywność rozpoznawania twarzy w życiu codziennym gdzie kamera nie zawsze jest wysokiej jakości oraz kiedy nie można zagwarantować idealnych warunków atmosferycznych. Należy potwierdzić lub odrzucić następującą hipotezę:

Algorytmy LBPH, Fisherfaces, Eigenfaces mogą być wykorzystywane do rozpoznawania twarzy w przypadku zdjęć o niskiej jakości.

Przez zdjęcia o niskiej jakości rozumie się fotografię o niskiej rozdzielczości lub rozmytym obrazie. Do przeprowadzenia badań została stworzona dedykowana platforma badawcza wykorzystująca minikomputer Raspberry Pi Zero W wyposażoną w dedykowaną kamerę o rozdzielczości 8 Mpx. Ponadto, zostało zaimplementowane oprogramowanie wykorzystujące bibliotekę OpenCv, która umożliwia wykorzystanie ww. algorytmów.

Przykład 1. Kod programu

```
#include <stdio.h>
import cv2
import os
import numpy as np
subjects = ["", "wisnia", "kubus", "kamila", "mohi", "nati"]
def detect_face(img):
```

```
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier('opencv-
files/lbpcascade_frontalface.xml')
    faces = face_cascade.detectMultiScale(gray,
scaleFactor=1.3, minNeighbors=4);
    if (len(faces) == 0):
        return None, None
    (x, y, w, h) = faces[0]
    return gray[y:y+w, x:x+h], faces[0]
def prepare_training_data(data_folder_path):
    dirs = os.listdir(data_folder_path)
    faces = []
    labels = []
    for dir_name in dirs:
        if not dir_name.startswith("s"):
            continue;
        label = int(dir_name.replace("s", ""))
        subject_dir_path = data_folder_path + "/" + dir_name
        subject_images_names = os.listdir(subject_dir_path)
        for image_name in subject_images_names:
            if image_name.startswith("."):
                continue;
            image_path = subject_dir_path + "/" + image_name
            image = cv2.imread(image_path)
            cv2.imshow("Training on image...", cv2.resize(image,
(400, 500)))
            cv2.waitKey(100)
            face, rect = detect_face(image)
            if face is not None:
                faces.append(face)
                labels.append(label)
    cv2.destroyAllWindows()
    cv2.waitKey(1)
    cv2.destroyAllWindows()
    return faces, labels
print("Preparing data...")
faces, labels = prepare_training_data("training-data")
print("Data prepared")
print("Total faces: ", len(faces))
print("Total labels: ", len(labels))
face_recognizer = cv2.face.HaarFaceRecognizer_create()
face_recognizer.train(faces, np.array(labels))
def draw_rectangle(img, rect):
    (x, y, w, h) = rect
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
def draw_text(img, text, x, y):
    cv2.putText(img, text, (x, y), cv2.FONT_HERSHEY_PLAIN,
1.5, (0, 255, 0), 2)
def predict(test_img):
    img = test_img.copy()
    face, rect = detect_face(img)
    label, confidence = face_recognizer.predict(face)
    label_text = subjects[label]
    draw_rectangle(img, rect)
    draw_text(img, label_text, rect[0], rect[1]-5)
    return img
print("Predicting images...")
test_img = cv2.imread("test-data/test1.jpg")
predicted_img = predict(test_img)
cv2.imshow(cv2.resize(predicted_img, (400, 500)))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Napisany program odpowiedzialny jest za przekształcenie zdjęć szkoleniowych w odpowiednie dane treningowe wykorzystując funkcje prepare_treining_data(PATH) oraz detect_face(IMG), która realizuje detekcje twarzy. Kolejno algorytm jest inicjalizowany oraz uczony wykorzystując dane treningowe. Ostatnim a zarazem najważniejszym zadaniem programu jest samo rozpoznanie twarzy na dostarczonym zdjęciu. Jest to możliwe dzięki funkcji predict(IMG). Dodatkowo dołączone są dwie funkcji odpowiedziane za formatowanie zdjęcia wynikowego - draw_rectangle(IMG, RECTANGLE), draw_text(IMG, TEXT, X, Y). Stworzony program realizuje funkcjonalności rzeczywistych systemów rozpoznawania oraz może być użyty do symulacji sytuacji z życia codziennego. Dodatkowo może być z łatwością powielony i rozwijany wedle potrzeb.

W celu określenia skuteczności algorytmów przeprowadzono szereg badań. Zostały one przeprowadzone z udziałem 5 osób dla których wykonano po 10 zdjęć badawczych wykorzystując stworzoną platformę. Do wykrycia obszaru twarzy został użyty wbudowany klasyfikator Haar biblioteki OpenCV przeszkolony na tysiącach twarzy. Zdjęcia badawcze zostały zrobione obniżając rozdzielczość oraz przysłaniając obiektyw kamery.

Rys. 4. Zdjęcia o niskiej jakości

Wynikami badań są trzy wartości:

- Skuteczność – procentowa wartość pozytywnego dopasowania osoby do zdjęcia;
- Czas predykcji;
- Pewność – wartość określająca odległość do najbliższego elementu w bazie znanych twarzy, mniejsza wartość oznacza większą pewność.

4. Wyniki badań

Pierwszym algorytmem poddanym badaniom był algorytm LBPH. Uzyskane wyniki przedstawia tabela 1

Tabela 1. Uśrednione wyniki rozpoznawania twarzy z wykorzystaniem algorytmu LBPH

Skuteczność	Czas predykcji [s]	Pewność
78%	4,16	76.059

Za zadowalającą skuteczność można uznać wynik w okolicy 80%. Badania pokazują, że LBPH znalazł się na granicy tej wartości. Porównując uzyskany wynik z otrzymanymi wartościami efektywności dla zdjęć o dobrej jakości, uzyskanymi w efekcie analogicznych badań, która wynosi w okoli 92% można uznać, że LBPH zadowalająco radzi sobie z zdjęciami o niskiej jakości. Średni czas wyniósł 4,16 sekund co w przypadku użytego sprzętu jest zadowalającym wynikiem. Docelowa pewność wyznaczona na podstawie

własnych badań na ponad 300 zdjęciach dla tego algorytmu wynosi w okolicach 20. Dużo wyższa wartość uzyskana w badaniach świadczy o tym, że rozpoznanie straciło na jakości.

Drugim algorytmem poddanym badaniom był Fisherfaces, uzyskane wyniki przedstawia tabela 2.

Tabela 2. Uśrednione wyniki rozpoznawania twarzy z wykorzystaniem algorytmu Fisherfaces

Skuteczność	Czas predykcji [s]	Pewność
44%	3,93	3126

Uzyskana skuteczność jest wynikiem nie zadowalającym co oznacza, że algorytm nie może być skutecznie użyty w systemach rozpoznania twarzy obsługującym zdjęcia o niskiej jakości.

Ostatnim przetestowanym algorytmem był Eigenfaces, którego wyniki przedstawia tabela 3.

Tabela 3. Uśrednione wyniki rozpoznawania twarzy z wykorzystaniem algorytmu Eigenfaces

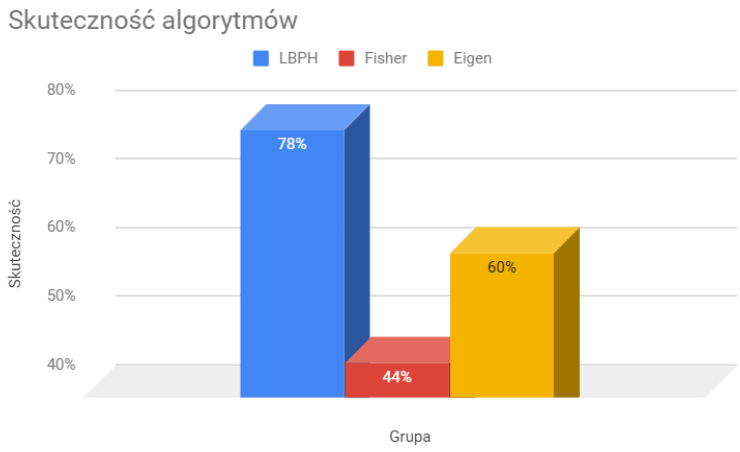
Skuteczność	Czas predykcji [s]	Pewność
60%	4,44	8894

Osiągnięta skuteczność jest również nie zadowalającym wynikiem, co świadczy o tym, że algorytm nie powinien być wykorzystywany ww. scenariuszach.

5. Wnioski

Podstawioną hipotezę, że **Algorytmy LBPH, Fisherfaces, Eigenfaces mogą być wykorzystywane do rozpoznawania twarzy w przypadku zdjęć o niskiej jakości** udało się udowodnić tylko w przypadku algorytmu LBPH. Pozostałe algorytmy nie osiągnęły zadowalających wyników.

Czas predykcji we wszystkich przypadkach okazał się być zbliżony, więc w przypadku rozwiązań gdzie szybkość działania nie jest najwyższym priorytetem nie powinna być brana pod uwagę jako czynnik decydujący. Warto zauważyć, iż mimo to, że algorytm LBPH osiągnął zadowalającą skuteczność, niekorzystna wartość pewności może świadczyć o nieprzewidywalności algorytmu z tego względu, więc nie powinien być on wykorzystywany w rozwiązaniach gdzie dominującym założeniem jest niedopuszczenie fałszywego rozpoznania. Poniższy wykres przedstawia zestawienie skuteczności algorytmów.



Rys.5. Skuteczność algorytmów

Literatura

[1] Stan Z. Li, Anil K. Jain: Handbook of Face Recognition. Springer. New York 2011.

[2] Ahonen, T., Hadid, A., and Pietikainen, M.: Face Recognition with Local Binary Patterns. Springer. Heidelberg 2004.

[3] Asit K. Datta, Madhura Datta, Pradipta K. Banerjee: Face Detection and Recognition: Theory and Practice. CRC Press. Boca Raton 2015

[4] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman: Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Transactions on Pattern Analysis and Machine Intelligence. Tom 19 Nr 7/1997

[5] Super data science, https://www.superdatascience.com/opencv-face-recognition/ [15.05.2018].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 71-76
Wysłane: 2018-11-29
Przyjęte: 2018-12-07

Porównanie nowych możliwości tworzenia aplikacji PHP na przykładzie Laravel i CodeIgniter

Daniel Drabik*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł ten przestawia wyniki porównania możliwości tworzenia aplikacji internetowych korzystając z języka PHP przy użyciu szkieletów budowy aplikacji – Laravel i CodeIgniter. Tekst wykazuje różnice dwóch implementacji między innymi w sposobie wytworzenia, aspektach architektury i wydajności.

Słowa kluczowe: laravel; codeigniter

*Autor do korespondencji.

Adres e-mail: daniel.drabik@outlook.com

Comparison of new ways of creating PHP applications using Laravel and CodeIgniter example

Daniel Drabik*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Abstract. This article presents a comparision of different ways of creating PHP web applications using Laravel and CodeIgniter frameworks. The text shows differences of two implementations including ways of creation, architecture aspects and performance.

Keywords: laravel; codeigniter

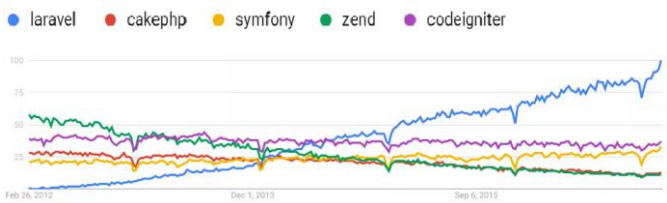
*Corresponding author.

E-mail address: daniel.drabik@outlook.com

1. Wstęp

Ideą tego artykułu jest porównanie dwóch niezależnych frameworków do tworzenia aplikacji internetowych w języku PHP – Laravel i CodeIgniter. Wybór został uwarunkowany popularnością zapytań w wyszukiwarce Google [1] (Rys. 1) oraz brakiem podobnych prac zawierających szczegółowe zestawienie tych narzędzi.

Istnieją artykuły, które starają się wskazać różnice pomiędzy obydwoma frameworkami. Nie uwzględniają one jednak badań nad wydajnością lub testów symulujących zwiększony ruch użytkowników [2-4].



Rys. 1. Wyniki zapytań poszczególnych frameworków PHP w wyszukiwarce Google na przestrzeni pięciu lat

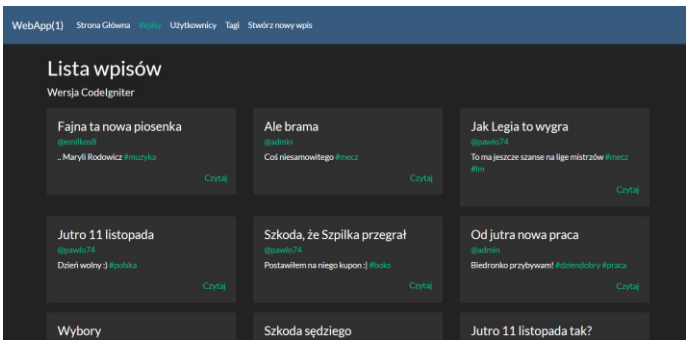
2. Cel i zakres badania

Celem badania jest porównanie wydajności oraz sposobów wytwarzania aplikacji przy użyciu frameworka Laravel i CodeIgniter. Do wykonania testów zostały utworzone dwie aplikacje testowe posiadające taką samą szatę graficzną oraz funkcjonalności. Różnicą jest szkielet budowy aplikacji. Projekty korzystają ze wspólnej bazy danych oraz środowiska testowego.

3. Aplikacje testowe

Aplikacja wykonana przy użyciu obydwu frameworków jest prostym dziennikiem internetowym – mikro blogiem. Podstawową funkcją jest tworzenie wpisów składających się z tytułu, treści oraz tagów tematycznych. Dodatkowo można wyświetlić wszystkie posty napisane przez danego autora.

Możliwe jest również pokazanie wszystkich wpisów, użytkowników oraz tagów znajdujących się w bazie danych. W każdej z podstron jest możliwość przejścia do pojedynczego elementu. Szata graficzna strony została maksymalnie uproszczona. W tym celu wykorzystano darmowy szablon Bootstrap (Rys. 2).



Rys. 2. Widok prezentujący listę wszystkich wpisów

4. Analiza porównawcza

Najważniejsze cechy Laravel i CodeIgniter zostały zaprezentowane w tabeli 1.

Tabela 1. Właściwości frameworków

	Laravel	**CodeIgniter**
Rok wprowadzenia	9 czerwca 2011	28 lutego 2006
Aktualna wersja i rok wydania	5.7 (04.09.2018)	3.1.9 (12.06.2018)
Instalacja i konfiguracja	Wymagane użycie Composera do pobrania instalatora. Konfiguracja polegająca na edycji pliku php.	Rozpakowanie archiwum. Konfiguracja polegająca na edycji pliku php.
Waga frameworka [MB]	30 MB	11 MB
Obsługiwane bazy danych	MySQL, SQLite, PostgreSQL, SqlSrv	MySQL, PostgreSQL, MS SQL, SQLite, CUBRID, Oracle, Interbase/Firebird, ODBC
Stosowany ORM	Eloquent	ActiveRecord
Wzorzec projektowy	Model-View-Controller	Model-View-Controller
Zabezpieczenia	Zabezpieczenie przed SQL Injection, filtrowanie XSS, ochrona przed atakiem CSRF, system autentykacji	Zabezpieczenia URI, filtrowanie XSS, ochrona przed atakiem CSRF
Silniki szablonów (widok)	Blade	Brak dedykowanego silnika (czysty kod PHP + HTML)
Licencja	X11 (MIT)	X11 (MIT)

4.1. Instalacja i konfiguracja

Proces instalacyjny CodeIgniter jest prosty i intuicyjny. Polega on na pobraniu najnowszej wersji ze strony twórców [5]. Następnie należy rozpakować zawartość otrzymanego archiwum do głównego katalogu projektu. Kolejnym krokiem jest zdefiniowanie adresu URL aplikacji. Odbywa się to w pliku *application/config/config.php*. Należy zmodyfikować wartość elementu tablicy *$config*:

```
$config['base_url'] = 'http://localhost/ci';
```

Kolejną czynnością jest zdefiniowanie połączenia z bazą danych. W tym celu należy uzupełnić tablicę *$db* znajdującą się w pliku *application/config/database.php* o dane takie jak host, nazwa bazy, login i hasło użytkownika oraz port. Tablica *$db* posiada domyślnie klucz *default*. Jest to identyfikator połączenia z bazą danych. Możliwe jest utworzenie wielu różnych połączeń pod innymi kluczami tablicy. Edytując wartość zmiennej *active_group* o odpowiedni identyfikator zostaje wskazane połączenie, z którego aplikacja będzie obecnie korzystać.

Laravel do instalacji wykorzystuje narzędzie Composer, czyli system zarządzania pakietami dedykowany dla języka PHP. Jest to jeden z wymogów Laravel. W przypadku braku narzędzia Composer, należy go zainstalować i skonfigurować na docelowej maszynie. Zwiększa to złożoność całego procesu instalacji w porównaniu do CodeIgniter. Posiadając skonfigurowany Composer należy pobrać instalator dla Laravel wykorzystując komendę:

```
composer global require "laravel/installer"
```

Następnie w docelowym katalogu projektu wykonać polecenie:

```
laravel new app1
```

Instalator utworzy nowy projekt, który zostanie nazwany *app1*. Laravel pozwala uruchomić serwer lokalny pod adresem *http://localhost:8000*, korzystając z interfejsu Artisan za pomocą komendy:

```
php artisan serve
```

Do obsługi aplikacji testowej nalezy ustawić połączenie połączenie z bazą danych oraz zdefiniowania adresu URL aplikacji. Konfiguracja ta przebiega poprzez edycję pliku *.env*. W odróżnieniu od CodeIgniter obie te czynności wykonywane są w tym samym miejscu.

4.2. Struktura plików

Struktura plików i katalogów obu projektów została ukazana na rysunku 3.

W CodeIgniter folder *application* jest odpowiedzialny za przechowywanie kodu logiki aplikacji. Wszystkie funkcjonalności oraz metody powinny znaleźć się właśnie w tym folderze. Katalog ten zawiera w sobie kilka podkatalogów:

- config – posiada pliki konfiguracyjne aplikacji np. ustawienia połączenia bazy danych;
- controllers – przechowuje kontrolery. Jest to podstawowa część aplikacji;
- hooks – służy do umieszczania kodu rozbudowującego oraz modyfikującego podstawowe funkcjonalności frameworka;
- libraries – przechowuje biblioteki;
- models –zawiera modeli aplikacji;

- views – odpowiada za przechowywanie widoków projektu.

Katalog *system* zawiera pliki kluczowe do działania aplikacji. Jest to rdzeń całego szkieletu frameworka.

Rys. 3. Struktura projektów

Każdy katalog w projekcie Laravel jest odpowiedzialny za inny aspekt infrastruktury aplikacji, a jego nazwa jest logicznie dopasowana do zadania:

- app – miejsce przechowywania aplikacji. Zawiera pliki obsługujące komendy konsolowe, zdarzenia, wyjątki, kontrolery, usług, modele oraz filtry i klasy żądań;
- bootstrap – skrypty uruchomieniowe frameworka;
- config – pliki konfiguracyjne aplikacji oraz klasy obsługujące bazy danych i ich migracje;
- public – katalog rdzenia projektu. Zawiera plik konfiguracji .htaccess, favicon strony, pliki JavaScript, kaskadowe arkusze stylów oraz *index.php*, który uruchamia aplikację;
- resource – katalog przechowujący pliki językowe oraz widoki projektu;
- routes – zawiera zasady routingu aplikacji;
- storage – miejsce magazynowania pamięci podręcznej, automatycznie generowanych logów oraz plików przesyłanych od strony użytkownika;
- tests – katalog zawierający przypadki testowe;
- vendor – zewnętrzny kod pobierany przy użyciu Composera.

Ważnymi elementami struktury projektu Laravel są niektóre pliki znajdujące się w głównym folderze projektu:

- .env – konfiguracja zmiennych środowiskowych projektu;
- artisan – interfejs konsolowy Artisan;
- composer.json – manifest zależności projektu;
- phpunit.xml – konfiguracja PHPUnit do obsługi testów jednostkowych;
- server.php – lekki serwer do lokalnego rozbudowywania aplikacji.

4.3. Routing

Edycja routingu w CodeIgniter polega na modyfikacji pliku znajdującego się w *application/config/routes.php*. Umiejscowiona jest w nim tablica *$route*, w której można dostosowywać zasady routingu. Klucz tablicy odpowiada ścieżce URL, która zostanie przechwycona, natomiast wartość tablicy decyduje o wywołaniu konkretnej metody kontrolera.

W Laravel, routing jest definiowany w czterech różnych plikach – *web.php*, *api.php*, *channels.php* oraz *console.php*. Znajdują się one w katalogu *routes* i są automatycznie inicjalizowane przez framework. Każdy z plików odpowiada za routing innego interfejsu aplikacji. Do obsługi aplikacji testowej został wykorzystany jedynie interfejs webowy.

Ustalanie zasad routingu polega na korzystaniu z metod obiektu klasy *Route*. Dostępne podstawowe funkcje to *get()*, *post()*, *patch()*, *put()*, *delete()* i *options()*. Każda z metod odpowiada innemu żądaniu przeglądarki. Wszystkie z nich przyjmują natomiast dwa parametry – ścieżkę *uri* i *callback* czyli wywołanie funkcji zwrotnej.

4.4. Kontroler

W CodeIgniter utworzenie kontrolera wiąże się z wygenerowaniem nowej klasy, która musi dziedziczyć po bazowej klasie *CI_Controller* i umieszczeniem jej w katalogu *application/controllers*. Ważnym krokiem jest dodanie w konstruktorze wywołania konstruktora rodzica klasy. Dodatkowo można wprowadzić w nim inicjalizację potrzebnych bibliotek i modeli.

Definiowanie kontrolerów w Laravel odbywa się w katalogu *app/Http/Controllers*. Podobnie jak w CodeIgniter, dobrą praktyką jest utworzenie klasy, która dziedziczy po wbudowanej klasie o nazwie *Controller*. Nie jest to co prawda wymagane, ale zyskuje się dostęp do różnych ciekawych

metod, takich jak na przykład *middleware()*, *validate()*, *dispatch()*.

4.5. Model

W CodeIgniter tworzenie modelu polega na wygenerowaniu nowej klasy, która dziedziczy po bazowej klasie *CI_Model*. Jest to analogiczne z procesem wytwórczym kontrolera w tym frameworku. Klasę należy umieścić w katalogu *application/models*. Modele w CodeIgniter posiadają rozbudowaną listę metod ułatwiającą otrzymywanie i wprowadzanie danych.

Istnieje możliwość nadania statusu globalnego modelowi. Oznacza to, iż będzie on inicjalizowany dla wszystkich kontrolerów podczas uruchamiania się silnika frameworka. W celu ustawienia takiej opcji należy dodać nazwę modelu do tablicy automatycznego włączenia, która znajduje się w *application/config/autoload.php*.

Model w Laravel wykorzystuje dziedziczenie po klasie *Model*. Ma to związek z dostarczanymi funkcjami zapewniającymi swobodne pobieranie informacji z bazy danych. Przykładem takiej metody jest *all()*. Funkcja ta wywołana na klasie modelu *Post* stworzonej do obsługi wpisów, pobiera wszystkie wiersze z bazy danych w tabeli *posts*. Laravel nie prosi o podanie tabeli, z której mają zostać uzyskane informacje. Przyjęto zasadę, iż nazwa tabeli w bazie danych powinna składać się z nazwy klasy modelu oraz dołączonej na końcu litery „s”.

Miejsce magazynowania modeli to katalog *app*. Operacje na bazie danych mogą być wykonywane bezpośrednio przy użyciu klasy *DB*, która pozwala w łatwy sposób zarządzać danymi.

4.6. Widok

W przeciwieństwie do innych popularnych frameworków w CodeIgniter nie wykorzystano żadnego znanego systemu szablonów, jak na przykład Twig czy Smarty. Wyświetlanie danych, tworzenie pętli raz instrukcji warunkowych przebiega przy użyciu wbudowanej składni języka PHP jaka jest dostępna w aplikacjach nie zawierających frameworków lub zewnętrznych bibliotek.

Laravel magazynuje swoje widoki w katalogu *resources/views*. Do obsługi wykorzystuje dedykowany silnik szablonów – Blade. W odróżnieniu od rozwiązań innych frameworków (na przykład Twig), pomimo posiadania własnej składni - nie istnieje zakaz wykorzystywania kodu napisanego w PHP [6]. Ponadto wszystkie widoki są kompilowane do postaci czystego kodu PHP. Dla swoich plików Blade korzysta z rozszerzenia *.blade.php*.

4.7. Porównanie wybranych metryk kodu

W tabeli 2 zostały porównane długości i wagi przykładowych komponentów obu aplikacji testowych.

Tabela 2. Metryki kodu

		Laravel	CodeIgniter
Kontroler wpisów	Liczba linii kodu	83	93
	Liczba znaków	2162	3330
	Rozmiar w KB	2.11	3.32
Model wpisów	Liczba linii kodu	49	111
	Liczba znaków	1210	3435
	Rozmiar w KB	1.18	3.74
Rozmiar aplikacji w KB		20.3	170
Rozmiar projektu w MB		37.9	14.6

5. Przebieg badania

Aplikacje testowe zostały uzupełnione o metody, które mierzą czas wykonywania danej operacji:

- uruchamiania strony powitalnej;
- wprowadzania dużej liczby wpisów;
- generowanie strony internetowej składającej się z dużej liczby elementów (wpisów).

Wszystkie pomiary czasu przeprowadzone zostały na dwóch niezależnych maszynach – komputerze stacjonarnym (PC) i laptopie, które posiadają różne wersje oprogramowania zestawione w tabeli 3.

Tabela 3. Wersje zastosowanego oprogramowania

	PC	Laptop
Serwer Apache	2.4.29	2.4.34
PHP	7.1.15	7.2.10
Serwer MySQL	5.6.36-83.0	5.6.39-83.1

Porównanie specyfikacji obu maszyn zostało zaprezentowane w tabeli 4.

Tabela 4. Specyfikacja sprzętowa

	PC	Laptop
Procesor	Intel Pentium G4560 2 x 3,50 GHz	Intel Core i5-3210M 2 x 2,50 GHz
Pamięć RAM	8 GB	8 GB
Dysk	550 MB/s dla odczytu 540 MB/s dla zapisu	555 MB/s dla odczytu 510 MB/s dla zapisu

Do mierzenia czasu została wykorzystana wbudowana funkcja PHP *microtime()*, która zwraca obecną godzinę w formie mikrosekund. Odejmując wartość przed i po wykonaniu operacji otrzymuje się liczbę sekund jaką potrzebował skrypt na jej realizację. Każdy test został powtórzony 5 razy, natomiast wynik końcowy to średnia z wszystkich prób.

5.1. Czas generowania strony

Strona powitalna obu aplikacji nie wykorzystuje żadnych informacji pochodzących z bazy danych. W tabeli 5 pokazano pomiary czasu generowania się dokumentów HTML.

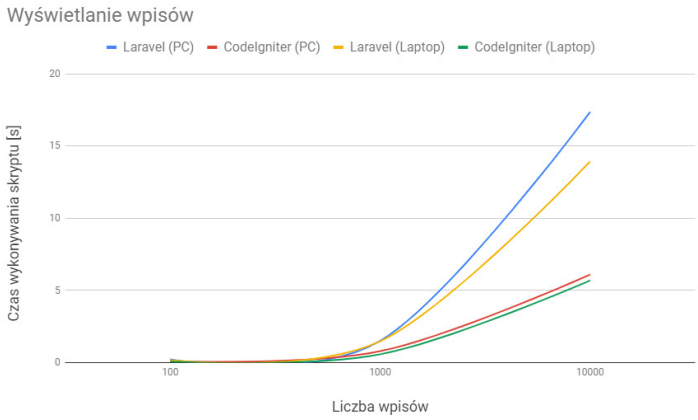
Tabela 5. Czasy generowania strony powitalnej

	CodeIgniter	Laravel
PC	0.00133 s	0.00679 s
Laptop	0.00188 s	0.01116 s

W tabeli 6 i na rysunku 4 porównano czas generowania się podstrony, która pobiera informacje z bazy danych. Przetestowano warianty wyświetlania listy wpisów w liczbie 100, 1 000 i 10 000. Obie aplikacje korzystają z tej samej bazy danych, na tym samym serwerze lokalnym.

Tabela 6. Czasy wyświetlania wpisów z bazy danych

	Liczba wpisów	CodeIgniter	Laravel
PC	100	0.09252 s	0.22297 s
	1000	0.79095 s	1.50472 s
	10000	6.0967 s	17.36832 s
Laptop	100	0.06506 s	0.16905 s
	1000	0.57473 s	1.47966 s
	10000	5.68999 s	13.93202 s



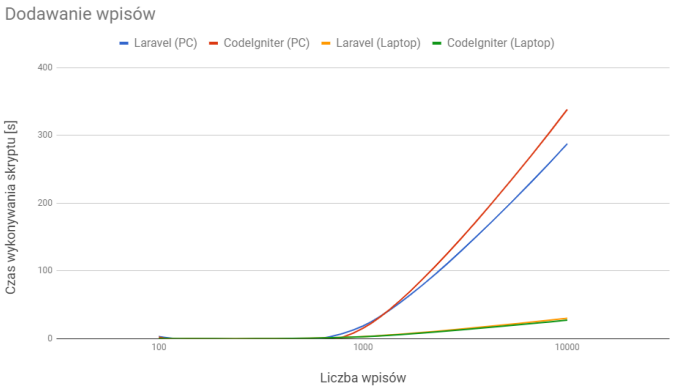
Rys. 4. Wykres zależności czasowych na podstawie danych z tabeli 6

5.2. Operacje wprowadzania wpisów

Kolejnym badaniem było sprawdzenie czasu wykonywania się metod wprowadzających nowe wpisy do bazy danych. Przed każdą operacją, tabela *posts* w bazie danych była opróżniana. Na potrzeby badania zostały stworzone odpowiednie metody, których zadaniem było generowanie wpisów w obu frameworkach. Funkcje te przyjmują jeden argument – liczbę iteracji dodawania postów jaką ma wykonać metoda. Badanie przeprowadzono w trzech konfiguracjach - 100, 1000 i 10 000 postów. Wyniki przedstawia tabela 7 oraz rysunek 5.

Tabela 7. Czasy wprowadzania wpisów do bazy danych

	Liczba wpisów	CodeIgniter	Laravel
PC	100	2.05762 s	3.10915 s
	1000	15.25737 s	18.66885 s
	10000	338.18025 s	287.47388 s
Laptop	100	0.33715 s	0.33725 s
	1000	2.74994 s	2.95962 s
	10000	27.22635 s	29.91931 s



Rys. 5. Wykres liniowy sporządzonych na podstawie danych z tabeli 7

5.3. Działanie pod obciążeniem

Do wykonania tzw. *stress testu* wykorzystano Apache JMeter [7]. Test polega na imitowaniu wejść użytkowników na stronę. JMeter pozwala wysyłać zapytania HTTP do serwera oraz kreować scenariusze zachowania użytkowników. Program jest udostępniany przy użyciu licencji Apache w wersji 2.0. Na potrzeby badania stworzono dwa scenariusze testowe. Parametry scenariuszy prezentuje tabela 8. Pierwszy test symuluje ruch 30 użytkowników, którzy w ciągu 50 sekund wchodzą na stronę i po około 30-60 s przechodzą na inne podstrony. Test został powtórzony 20 razy. Drugi test symuluje wejście 250 osób na stronę w ciągu 15 sekund. Otrzymane wyniki prezentuje tabela 9.

Tabela 8. Parametry scenariusza do testów obciążeniowych

	Test #1	Test #2
Liczba wątków (użytkowników)	30	250
Czas rozpoczęcia wszystkich wątków	50	15
Ilość powtórzeń	20	20

Tabela 9. Wyniki *stress testu*

	Test	CodeIgniter		Laravel	
		PC	Laptop	PC	Laptop
Średni czas odpowiedzi [ms]	#1	151	337	387	389
	#2	1903	9545	376	9620
Min. czas odpowiedzi [ms]	#1	94	155	166	246
	#2	93	152	11	232
Maks. czas odpowiedzi [ms]	#1	3134	2478	8661	8453
	#2	7481	44050	8068	39458
Odchylenie standardowe	#1	158.76	222.31	325.98	400.78
	#2	1443.39	13292.89	514.43	12843.94
Liczba błędów	#1	0%	0%	0.5%	0.5 %
	#2	0%	23.65%	11.33%	24.91%
Średnia przepustowość (zapytania na sekundę)	#1	0.6	0.6	0.6	0.04
	#2	4.7	4.0	4.8	4.1

5.4. Ocena frameworków

Na podstawie wyników przeprowadzonego porównania, dokonano autorskiej oceny obu frameworków (tabela 10).

Każdy element został oceniony w skali 1-5, gdzie 5 jest oceną najwyższą.

Tabela 10. Ocena frameworków

	CodeIgnitier	**Laravel**
Instalacja i konfiguracja	5	3
Struktura aplikacji	4	3
Praca na bazach danych	3	5
Testy wydajnościowe	4	2
Działanie pod obciążeniem	4	3
Metryki kodu	3	5
Stopień skomplikowania	4	3
Licencja	5	5
Sumaryczna ocena	**32**	**29**

6. Wnioski

Na podstawie przeprowadzonych badań można zauważyć, iż ważnym aspektem poprawnego i optymalnego działania każdego frameworka jest wykorzystanie najnowszej wersji języka PHP. W przypadku obu aplikacji zauważalny jest skok wydajnościowy i redukcja czasu wykonywania się operacji. Zwłaszcza w przypadku operacji dodawania nowych wpisów do bazy danych, gdzie widać nawet dziesięciokrotne zmniejszenie czasu potrzebnego do wykonania zadania.

Biorąc pod uwagę testy związane z generowaniem statycznej strony oraz wyświetlaniem listy wpisów, można zauważyć dominację CodeIgniter nad Laravel. Laravel zaczyna pokazywać swoją przewagę dopiero przy dodawaniu wielotysięcznej liczby rekordów do bazy danych. Przewaga jednak dotyczy tylko konfiguracji ze starszą wersją języka PHP (na PC). Korzystając z najnowszej wersji - Laravel nieznacznie przegrywa z lekkim CodeIgniter.

Badanie symulujące ruch użytkowników przy użyciu oprogramowania JMeter - również wskazuje na wyższość CodeIgniter. Dopiero na słabszej maszynie przy zwiększonym obciążeniu, aplikacja zaczęła pokazywać błędy. Laravel zwracał błędy nawet przy małym ruchu na każdej z maszyn.

Na podstawie informacji z tworzenia aplikacji testowych oraz przeprowadzonych badań można jednoznacznie polecić framework CodeIgniter. Jest on bardzo lekki i przyjemny. W połączeniu z najnowszą wersją PHP, również wydajny. Ponadto posiada niski próg wejścia dla programistów, którzy nie mieli wcześniej styczności z żadnym innym frameworkiem.

Literatura

[1] https://trends.google.pl/trends/explore?date=today%205-y&q=laravel,cakephp,symfony,zend,codeigniter [26.10.2017]

[2] https://www.quora.com/What-is-the-differences-between-Codeigniter-and-Laravel [27.11.2018]

[3] https://www.elsner.com/laravel-vs-codeigniter-php-framework/ [27.11.2018]

[4] https://www.codementor.io/chirilovadrian360/laravel-or-codeigniter-ilos0bfw9 [27.11.2018]

[5] https://www.codeigniter.com/download [17.11.2018]

[6] https://symfony.com/doc/current/templating.html [10.10.2018]

[7] https://jmeter.apache.org/ [27.11.2018]

[8] https://laravel.com/docs/5.7 [13.04.2018]

[9] M. Bean, Laravel 5 Essentials, Packt Publishing 2015

[10] T. Matula, Laravel, Tworzenie aplikacji, Receptury, Helion, Gliwice 2015

[11] https://www.codeigniter.com/user_guide/ [17.11.2018]

[12] R. Foster, CodeIgniter Web Application Blueprints, Packt Publishing 2015

[13] K. Suzuki, M. Whitney, CodeIgniter Testing Guide, LeanPub 2016

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 77-81
Wysłane: 2018-12-01
Przyjęte: 2018-12-07

Wdrożenie narzędzi wspomagających zarządzanie projektami w firmach IT

Radosław Albiniak*, Elżbieta Miłosz
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł prezentuje ocenę narzędzi do wspomagania zarządzania projektem informatycznym. Badania przeprowadzono za pomocą metody sondażu diagnostycznego na temat oceny wdrażania narzędzi do wspomagania zarządzania projektem informatycznym.

Słowa kluczowe: narzędzia do wspomagania zarządzania projektem; wdrożenie narzędzi do zarządzania projektem informatycznym; zarządzanie projektem informatycznym

*Autor do korespondencji.

Adres e-mail: radoslaw.albiniak@gmail.com

Implementation of management support tools projects in IT companies

Radosław Albiniak*, Elżbieta Miłosz
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the evaluation of tools to support IT project management. The research was carried out using the diagnostic survey method on the assessment of the implementation of tools to support IT project management.

Keywords: tools to support project management; implementation of IT project management tools; IT project management

*Corresponding author.

E-mail address: radoslaw.albiniak@gmail.com

1. Wstęp

Inżynieria oprogramowania to dziedzina inżynierii systemów, która zajmuje się wszelakimi aspektami w produkcji oprogramowania. Inżyniera oprogramowania ma na celu dostarczenie oprogramowania, które będzie charakteryzować się wysoką jakością. Należy koordynować oraz zarządzać bardzo wieloma czynnikami, które mają realny wpływ na końcowe zakończenie sukcesem.. Każdy krok realizacji może spowodować, że pojawią się bardzo duże komplikacje, które powodować zagrożenia, wpływające na jakość produkowanego oprogramowania. Mogą również doprowadzić do sytuacji, w której oprogramowanie nie zostanie w ogóle wytworzone. Jakość oprogramowania jest uzależniona od wielu czynników występujących, od momentu inicjacji projektu, aż do czasu oddania go w ręce użytkownika. Cykl życia produktu jest objęty poprzez zarządzanie projektami, co tym samym stanowi o wartościach czynników, które kształtują jego jakość.

Wzrost popularności projektów informatycznych - rozwój inżynierii oprogramowania - spowodował powstanie również wielu technik oraz metod, które służą do estymacji oraz planowania. Nowe, proste metody oraz techniki, które należą do zestawu narzędzi miękkich w ostatnich latach zyskały bardzo dużą popularność.

Za cel badań autor postawił sobie przedstawienie procesu wdrażania narzędzi wspomagających zarządzaniem projektami na przykładzie 10-15 osobowy firm IT, z różnych województw, zajmujących się tworzeniem oraz serwisowaniem aplikacji internetowych. Ponadto wszystkie użyte narzędzia zostaną opisane w pracy, wskazane będą ograniczenia oraz uwagi odnośnie ich stosowania.

Praca naukowa wymagała zapoznania się z nowymi technologiami informacyjnymi oraz poszerzenia wiedzy z technologiami, z którymi autor miał już jakieś doświadczenie. Wiedzę poszerzano za pomocą literatury i artykułów dostępnych w Internecie.

2. Definicja projektu informatycznego, metodyk zarządzania projektem oraz wdrażanie narzędzi do zarządzania projektem informatycznym

2.1. Definicja projektu informatycznego

Projektem można nazwać „niepowtarzalne (realizowane jednorazowo), złożone przedsięwzięcie zawarte w skończonym przedziale czasu — z wyróżnionym rozpoczęciem i końcem — realizowane zespołowo (wielopodmiotowo), w sposób względnie niezależny od powtarzalnej działalności przedsiębiorstwa, za pomocą specjalnych metod oraz technik” [7]. Przytoczoną definicję można rozszerzyć dodając, że celem projektu jest wprowadzenie założonych zmian w dotychczasowym funkcjonowaniu przedsiębiorstwa odznaczających się innowacyjnością oraz unikalnością bez przekraczania ustalonego budżetu.

Szczególnego znaczenia w zarządzaniu projektem nabierają wszelkie działania związane z planowaniem oraz estymacją ukierunkowane na zmiany, a nie sam plan projektu [5]. Odwołując się do stożka niepewności (ang. Cone of

Uncertainty) wg Barry'ego Boehma można zauważyć, że oszacowanie czasu trwania projektu w początkowej fazie jego definiowania waha się od 60% do 160% jego wartości. Niemożliwe jest więc stworzenie bezbłędnego planu realizacji projektu w fazie początkowej oraz późniejszych fazach.

Niepewności oszacowań nie można wyeliminować. Konieczne jest ciągle analizowanie zdobytej wiedzy, zmian zaistniałych w projekcie, jak i w środowisku projektu, oraz ich odpowiednie adoptowanie. Dobry proces planowania jest próbą odnalezienia optymalnej metody realizacji projektu poprzez zmniejszenie ryzyka, niepewności, lepsze wspomaganie procesów decyzyjnych i przepływu informacji. Także poprzez budowanie zaufania pomiędzy zespołem projektowym a klientem oraz zespołem projektowym a kierownikiem projektu. Założenia te są spełniane przez lekkie metodyki zarządzania projektami.

Celem projektu informatycznego jest utworzenie lub modyfikację systemu informatycznego, który ma na celu powiązanie ze sobą sprzętu, oprogramowania, zasobów ludzkich, elementów informacyjnych oraz organizacyjnych, aby przetwarzane dane przy pomocy techniki komputerowej były zgodne ze stworzoną dokumentacją techniczną dla projektu. Projekt informatyczny dotyczy głównie wdrożeń struktur IT, oprogramowania lub wytworzenia oprogramowania [2].

2.2. Metodyki zarządzania projektem informatycznym

Metodyka to całość postępowania, zbiór metod prowadzące do, z góry podjętego celu, bądź zbiór zasad określających do doprowadzenia do założonej czynności [3]. Metodyka zarządzania projektem jest spisem konkretnych kroków przeprowadzenia projektu. Nakreśla w jaki sposób planowania, zarządzania oraz kontrolowania projektu.

Jest kilka rodzajów metodyk. Różnią się one sposobem zrozumienia samej definicji projektu, określeniem celów oraz priorytetów. Dzielone są na cztery główne kategorie [6]:

- Wytwórcze: MSF (ang. Microsoft Solution Framework), RUP (ang. Rational Unified Process),
- Organizacyjne: Six Sigma, COBIT (ang. Capability Matatuirty Model Integration).
- Adaptacyjne: Programowanie zwinne (ang. Agile), w których skład wchodzą: Scrum, eXP (ang. Extremme Programming),
- Zarządcze: PMBoK (ang. Project Management Body of Knowledge), PRINCE 2 (ang. Projects In Controlled Enviroments).

Zastosowanie metodyk w pracy nad projektami pomaga w kontrolowaniu porządku w sprawach organizacyjnych. Ma to wpływ na wydajność zespołu, ale także zmniejsza wydatki, a największą zaletą jest zwiększenie szansy na końcowy sukces projektu.

SCRUM to iteracyjna metodyka prowadzenia projektów, zalicza się do zwinnych metodyk wytwarzania oprogramowania [4]. Jednym z głównych założeń oprogramowania jest praca w zazwyczaj stałych okresach czasowych, które nie trwają zazwyczaj więcej niż kilka tygodni. Efektem takiego działania jest dostarczenie kolejnej części oprogramowania. W związku z tym, podstawą zakończenia każdej iteracji jest osiągnięcie zauważalnych zmian w projekcie. W metodyce wyróżniana jest struktura organizacyjna, która wygląda następująco [8]:

- Prodcut owner - właściciel projektu, zazwyczaj klient
- Scrum master - kierownik projektu
- Team - zespół odpowiedzialny za dostarczanie kolejnych wersji produktu, aż do końcowego produktu.

Wyszczególnić można następujące etapy projektu [1]:

1. Przygotowanie projektu (ang. Project Backlog), czyli opis wszystkich zakładanych funkcjonalności z określeniem priorytetów,
2. Spotkanie określające listę zadań wykonanych podczas przebiegu (z ang. Sprint Backlog),
3. Realizacja zadań iteracji, w której nie powinno zmieniać się zakresu zadań. Codziennie powinny się odbywać spotkania, które pozwalają na omówienie przez zespół aktualnego stanu i planu realizacji zadań.
4. Prezentacja, czyli spotkanie i zaprezentowanie wykonanego produktu (ang. Sprint review). W tym etapie ustala się również termin spotkania wprowadzającego do kolejnego przebiegu.

PRINCE2 jest uniwersalną metodyką do zarządzania różnymi rodzajami projektów, nie skupia się wyłącznie na projektach informatycznych. Głównymi aspektami jest jak najlepsze zaplanowanie poszczególnych etapów projektu i stworzenie dokładnej dokumentacji [9].

Metodyka PRINCE2 składa się z podstawowych zasad, czyli procesów odpowiadających za plan pracy, tematy opisujące sposoby działania oraz poniesione koszty.

Komitet sterujący odgrywa główną rolę w strukturze organizacyjnej projektu, a więc odpowiedzialny jest za podstawowe decyzje dotyczące projektu. W metodyce tej projekt posiada [6]:

- Odpowiednią ilość zasobów (ludzie, osprzęt, pieniądze)
- Konkretnie określony początek oraz koniec
- Strukturę organizacyjną z podziałem odpowiedzialności oraz obowiązków
- Ściśle zdefiniowany i określony cel projektu

Metodyka eXP to programowanie ekstremalne, jest kolejnym paradygmatem zwinnego tworzenia oprogramowania [14]. Najważniejszym założeniem w tej metodyce jest ominięcie fazy projektowania architektury, w celu większego skupienia się na rzeczywistym celu projektu. Stworzona jest dla zazwyczaj kilkunastoosobowych zespołów, które zajmują się małymi lub średnimi projektami. Założenia metodyki :

- Planowanie - planuje się przed każdą pojedynczą iteracją,
- Relacja z klientem - kontakt z klientem rozwiązuje problemy w przypadku złych specyfikacji lub założeń,
- Iteracyjność - dostarczanie kolejnych wersji produktu,
- Testy jednostkowe - tworzone przed rozpoczęciem pracy nad projektem, sprawdza czy opracowany kod jest prawidłowy,

- Ciągła zmiana architektury - kiedy modyfikacja architektury jest potrzebna, zostaje wykonana, gdy nie zostaną popsute wyniki poprzednich iteracji,
- Programowanie w parach - praca programistów w dwuosobowych grupach, gdzie jedna osoba odpowiedzialna jest za pisanie kodu, a druga sprawdza jego poprawność. Programiści wykonują czynności naprzemiennie.

Tak jak SCRUM, metodyka eXP jest metodyką iteracyjną, która dzieli się na fazy.

2.3. Wdrożenie narzędzi do zarządzania projektem informatycznym

Wdrożenie systemu polega na dostarczaniu oraz uruchomieniu systemu, a także dostosowywaniu systemu do wymagań użytkownika. Kolejnymi zadaniami wdrożenia jest testowanie oraz migracje danych [10].

Prawidłowe wdrażanie to takie, gdzie posiadana jest kompletna dokumentacja, w skład dokumentacji wchodzi:

- dokumentacja użytkownika,
- dokumentacja programów,
- dokumentacja administratora systemu.

Wdrażanie systemu składa się z procesu wdrażania, wyszczególniamy następujące podstawowe procesy:

1) Analiza potrzeb klienta,
2) Przygotowanie projektu technicznego,
3) Przygotowanie infrastruktury sprzęto-systemowej,
4) Instalacja i dostosowanie komponentów oprogramowania,
5) Testowanie wdrażanego systemu,
6) Szkolenie administratorów oraz użytkowników.

Wyróżniamy trzy zasadnicze strategie wdrożenia systemu. Są to:

- Strategia całościowa - jest to strategia niosąca największe ryzyko. Polega na całkowitej zamianie starego systemu na nowy, nie stosując stopniowej rezygnacji ze starego systemu. Główne problemy wynikają z przyzwyczajeń do obsługi nowego systemu oraz braku możliwości powrotu do starszego systemu. Atutem jest niski koszt wdrożenia.
- Strategia cząstkowa - jest to strategia niosąca najmniejsze ryzyko, ale może przynosić większe koszty od strategii całościowej. Bezpieczeństwo implementacji przy dużych projektach, polega na wdrażaniu kolejnych części systemu. Wyróżniamy dwa podziały strategii: geograficzna oraz funkcjonalna.
- Strategia równoległa - jest to strategia najbardziej kosztowna, czasochłonna, ale jest to najbezpieczniejsze rozwiązanie. Jej podstawowym założeniem jest korzystanie ze starego systemu oraz wdrażaniu nowego. Strategia ta może powodować konflikty, ponieważ część pracowników pracuje na nowym systemie, podczas gdy reszta pracowników pracuje jeszcze na starym.

Odpowiednio dobrana procedura wdrożenia powinna być dopasowana do aspektów takich jak rodzaj oraz wielkość systemu informatycznego i stopnia ryzyka związanego z wdrożeniem. Najważniejszym elementem podczas wdrożenia według prawa jest wprowadzenie systemu do firmy, organizacji, bądź instytucji przez przekazanie praw własności, licencji oraz praw autorskich.

3. Cel badań

Celem głównym badań jest ocena wdrożeń narzędzi do wspomagania zarządzania projektem. Badane narzędzia oraz firmy IT wybrane zostały kierując się, tym aby narzędzia były aplikacjami internetowymi oraz liczba pracowników firmy znajdowała się przedziale 10 - 15 osobowym. Następnie na podstawie pytań badawczych powstała ankieta, która została rozesłana pracownikom wybranych firm IT.

Narzędzia poddane badaniu:

1) Jira,
2) Asana,
3) Redmine.

Zostały ocenione pod względem:

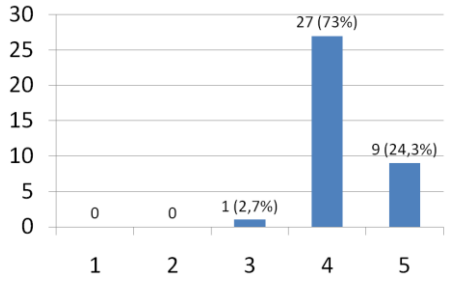
1) Zarządzania projektem przed i po wdrożeniu,
2) Korzyści jakie dostarczyło wdrożenie danego narzędzia do wspomagania zarządzania projektem.
3) Problemów, które dostarczyło podczas i po wdrożeniu danego narzędzia do zarządzania projektem.

4. Wyniki analizy

Skala ocen potrzebna do analizy:

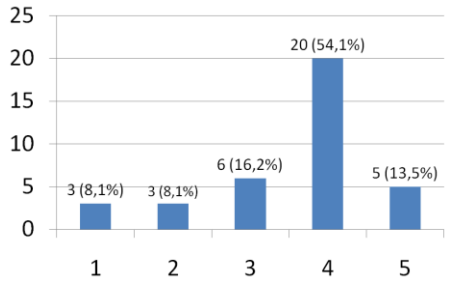
- bardzo źle,
- źle,
- średnio,
- dobrze,
- bardzo dobrze.

Pierwsza ocena dotyczy narzędzia po wdrożeniu pod względem korzyści jakie dostarczyło pracownikom firmy narzędzie do zarządzania projektem informatycznym. Tylko jeden pracownik ocenił jako średnie dostarczone korzyści. Więcej odpowiedzi, niż ocena średnia uzyskała odpowiedź bardzo dobrze, taką ocenę pod względem korzyści dostarczonych poprzez wdrożenie narzędzia do wspomagania zarządzania projektem wybrało 9 osób, czyli 24,3% osób badanych. Największa ilość osób, bo aż 27 oceniło wdrożenie narzędzia pod względem korzyści jako dobre. Była to bardzo znaczna różnica, ponieważ odpowiedź dobre wybrało aż 73% badanych pracowników. Oceny mówiące, o dostarczonych korzyściach po wdrożeniu narzędzi do wspomagania zarządzaniem projektem informatycznym jako bardzo złe lub złe, nie wystąpiły. Rysunek 1. przedstawia wykres, na którym znajduje się jak narzędzie oceniali pracownicy pod względem korzyści przyniesionych poprzez wdrożenie narzędzi.



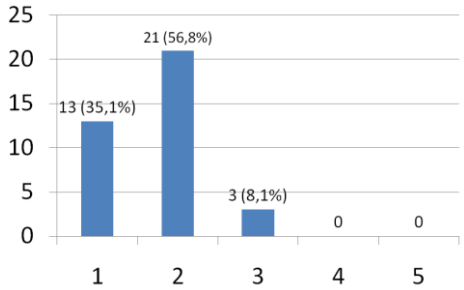
Rys. 1. Wykres przedstawiający ocenę pracowników pod względem korzyści jakie dostarczyło wdrożenie narzędzi do wspomagania zarządzaniem projektem informatycznym.

Druga ocena ma związek z narzędziem pod względem problemów jakie dostarczyło ono pracownikom podczas wdrażania i po wdrożeniu. Skala w tym pytaniu przyjmowała wartości od 1 do 5, a jej opis możemy znaleźć wyżej. Pracownicy przy odpowiedzi na pytanie wykorzystali wszystkie możliwe opcje. Trzy osoby uznały problemy podczas i przy wdrażaniu jako bardzo złe. Odnosząc się do tej informacji możemy ustalić, że według tych pracowników wystąpiły bardzo poważne problemy. Taką samą ilość jak odpowiedź bardzo źle, uzyskała odpowiedź źle, a zatem zdaniem trzech pracowników wystąpiły istotne problemy. Kolejnych 5 pracowników uznało, że nie wystąpiły problemy, albo wystąpiły problemy bardzo mało istotne podczas i przy wdrażaniu, odpowiedź tą sugeruje zaznaczenie przez nich odpowiedzi bardzo dobrze. Kolejne 6 osób zaznaczyło odpowiedź średnie. Najwięcej pracowników oceniło narzędzie pod względem problemów dostarczonych podczas i przy wdrażaniu jako dobre, było to 20 pracowników, znaczna większość, biorąc pod uwagę inne odpowiedzi Najwięcej odpowiedzi miała ocena dobra, zaznaczył ją 54,1% pracowników. Rysunek 2. przedstawia wykres oceny narzędzia do wspomagania zarządzanie projektem informatycznym podczas i przy wdrażaniu według pracowników.



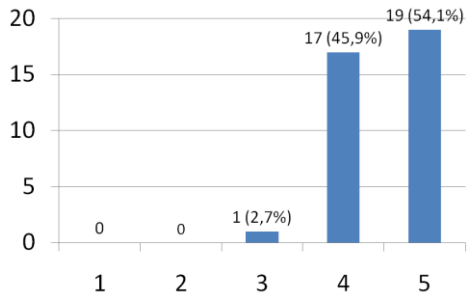
Rys. 2. Wykres przedstawiający ocenę pracowników pod względem problemów jakie dostarczyło wdrożenie i wdrażanie narzędzi do wspomagania zarządzaniem projektem informatycznym

Trzecia ocena dotyczy zarządzania projektem informatycznym przed wdrożeniem narzędzia do wspomagania. Odpowiedzią był wybór wartości od 1 do 5, gdzie opis tej skali wymieniony został wyżej. Według trzech pracowników, zarządzanie projektem informatycznym przed wdrożeniem narzędzia do wspomagania zarządzania projektem informatycznym utożsamiło się z oceną średnią. Krytycznie, bo zaznaczając odpowiedź bardzo źle, zarządzanie projektem informatycznym przed wdrożeniem narzędzia do wspomagania zarządzania projektem oceniło 13 pracowników. Najwięcej pracowników, bo aż 21, zaznaczyło odpowiedź źle, a więc według tych pracowników zarządzanie projektem informatycznym przed wdrożeniem narzędzia do zarządzania projektem informatycznym było złe. Odpowiedź złe wybrało, aż 56,8% pracowników. Analizując te dane, można stwierdzić, że zarządzanie projektem informatycznym przed wdrożeniem narzędzia do zarządzania projektem informatycznym nie zostało najlepiej ocenione. Rysunek 3 przedstawia odpowiedzi pracowników dotyczących oceniania przez nich zarządzanie projektem informatycznym przed wdrożeniem narzędzia do zarządzania projektem informatycznym.



Rys. 3. Wykres przedstawiający ocenę pracowników zarządzanie projektem informatycznym przed wdrożeniem narzędzia do wspomagania zarządzania projektem informatycznym.

Czwarta ocena ma związek z zarządzaniem projektem informatycznym po wdrożeniu narzędzia. Tak jak w poprzednich pytaniach, przy odpowiedzi pomocna jest skala, która przyjmuje wartości od 1 do 5. Pracownicy zaznaczyli tylko 3 opcje spośród wszystkich możliwych - średnio, dobrze, bardzo dobrze. Najmniej odpowiedzi uzyskała opcja średnio, taką opcję wybrał tylko jeden pracownik. Pomiędzy ocenami dobrze, a bardzo dobrze, różnica głosów była równa 2. Ocenę dobrą zaznaczyło 17 pracowników, natomiast bardzo dobrze zaznaczyło, aż 19 pracowników. Podsumowując te dane można stwierdzić, że pracownicy pozytywnie oceniają zarządzanie projektem po wdrożeniu narzędzia do wspomagania zarządzaniem projektem informatycznym. Rysunek 4. przedstawia odpowiedzi dotyczące oceny zarządzania projektem po wdrożeniu narzędzi.



Rys. 4. Wykres przedstawiający ocenę pracowników zarządzanie projektem informatycznym po wdrożeniu narzędzia do wspomagania zarządzania projektem informatycznym.

5. Wnioski

Celem badań było porównanie wdrożonych narzędzi do zarządzania projektem informatycznym w firmach IT. Cel ten został zrealizowany, a ocenione narzędzia to: Asana, Jira oraz Redmine.

W artykule przedstawiono definicję zarządzania projektem, najważniejsze metodyki zarządzania projektem oraz wdrażanie narzędzi do wspomagania zarządzania projektem informatycznym.

Najważniejszym elementem przedstawionych badań, było ocenienie narzędzi pod względem dostarczonych korzyści po wdrożeniu narzędzi do zarządzania projektem. Narzędzia zostały ocenione pozytywnie w aspekcie dostarczonych korzyści.

Kolejnym bardzo ważnym czynnikiem podczas wdrażania były problemy, które występowały podczas i po wdrożeniu. Pracownicy różnie oceniali występujące problemy, lecz najwięcej osób oceniła ten proces pozytywnie.

Następnym elementem badanym, było zarządzanie projektem informatycznym. Pracownicy oceniali zarządzanie projektem informatycznym przed wdrożeniem narzędzi do zarządzania projektem informatycznym jako złe lub bardzo złe.

Ostatnim badanym elementem było zarządzanie projektem informatycznym po wdrożeniu narzędzi do wspomagania zarządzaniem projektem informatycznym, oceny były pozytywne.

Literatura

[1] Apanowicz J., Metodologia ogólna, BERNARDINUM, Gdynia 2002.

[2] Bradley K.: Understanding PRICE2,SPOCE Project Management Ltd, Poole 2002

[3] Charvat J.: Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects. Wiley 2003.

[4] Highsmith J.: Agile Project Managment: Creating Innovative Products. The Agile Software Development Series. Addison-Wesley Professional, 2004.

[5] Kapusta M: Zarządzanie projektami krok po kroku, Edgard, 2013.

[6] Koszlajda A.: Zarządzanie projektami IT. Przewodnik po metodykach. Helion 2010.

[7] Mignus N.: Zarządzanie projektami. Wydawnictwo Helion, 2009.

[8] Schwaber K., Sutherland J.: Scrum Guide. Scrum.org 2010.

[9] The Stationery Office: PRINCE2. Skuteczne zarządzanie projektami. OGC 2006.

[10] Wachnik B, Wdrażanie systemów informatycznych wspomagających zarządzanie, Polskie Wydawnictwo Ekonomiczne, 2016.