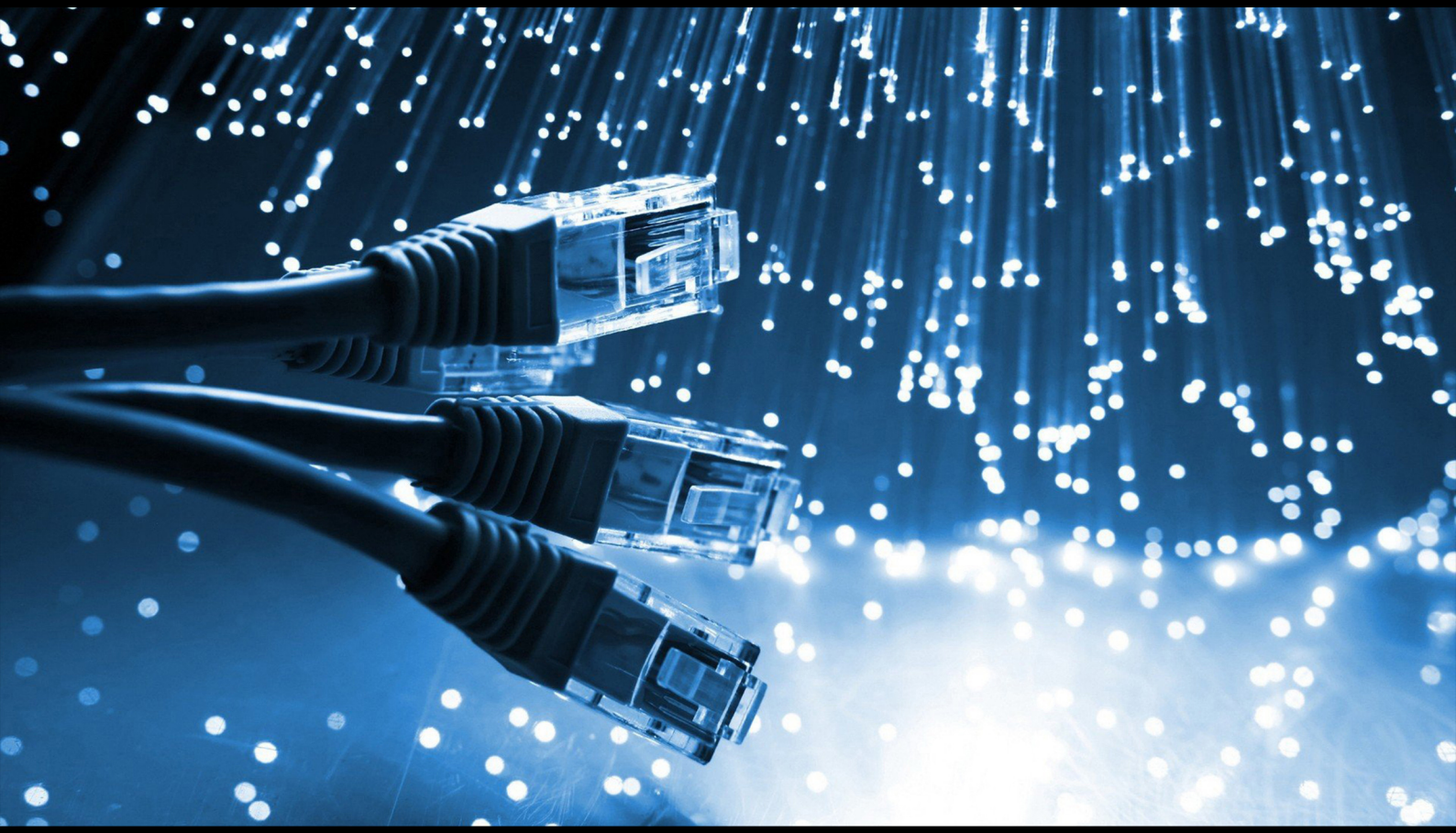
JCSI

Journal of Computer Sciences Institute

Volume 3/2017



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 inż. Dariusz Gutek
dr Mariusz Dzieńkowski
dr inż. Jakub Smołka
dr inż. Tomasz Szymczyk
dr hab. inż. Dariusz Czerwiński, prof. PL
dr inż. Marek Miłosz
dr inż. Jacek Kęsik
dr inż. Maria Skublewska-Paszkowska

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:
Dariusz Gutek
Mariusz Dzieńkowski
Jakub Smołka
Tomasz Szymczyk
Dariusz Czerwiński
Marek Miłosz
Jacek Kęsik
Maria Skublewska-Paszkowska

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

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JCSI 3 (2017) 1-5
Journal of Computer Sciences Institute

Wysłane: 2016-12-08
Przyjęte: 2016-12-14

Bezpieczeństwo aplikacji internetowych

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

Streszczenie. W artykule poruszono temat bezpieczeństwa aplikacji internetowych. Opisane zostały najpopularniejsze rodzaje ataków. Analizie poddana została autorska aplikacja Internetowy Notatnik. Przedstawiono rozwiązania mające na celu poprawę bezpieczeństwa.

Słowa kluczowe: ataki na serwisy internetowe; bezpieczeństwo; sql injection; xss

*Corresponding author.

Adres e-mail: michal.furtak1@pollub.edu.pl

Security of Web Applications

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

Abstract. This article is about web application security. Describes the most common types of web attacks. The analysis was subjected to authoring application Internetowy Notatnik. Provides solutions to improve safety.

Keywords: web attack; sql injection, xss

*Corresponding author.

E-mail address: mfurtak@gmail.com

1. Wstęp

Początki Internetu sięgają końca lat sześćdziesiątych ubiegłego wieku. W 1969 roku, na Uniwersytecie Kalifornijskim rozpoczęły się prace nad połączeniem pierwszych węzłów sieci ARPANET (ang. Advanced Research Projects Agency Network). Sieć ta obecnie uznawana jest za bezpośredniego przodka współczesnego Internetu [1]. Projekt ten pierwotnie wykorzystywany miał być do celów naukowych oraz wojskowych. Prawdopodobnie niewielu z ówcześnie żyjących zdawało sobie sprawę jak wielki wpływ wywarty zostanie przez ten projekt na życie i funkcjonowanie późniejszych pokoleń.

Obecnie niewielu z nas wyobraża sobie życie bez dostępu do światowej sieci. Według badań Netcraft liczba serwisów internetowych wynosi 863 miliardy [2]. Dla porównania warto dodać że w roku 2000 było to tylko 17 miliardów. Przyczyn tak szybkiego rozwoju można dopatrywać się w upowszechnieniu dostępu do sieci. Obecnie, dzięki postępowi technicznemu, jest on możliwy z prawie każdego miejsca na ziemi przy pomocy smartphonu. Szacuje się, że 43% ludności Ziemi jest użytkownikami Internetu.

Patrząc na historie Internetu nie sposób nie dostrzec zmian w udostępnianych treściach. Przejście z ery Web 1.0 do Web 2.0 wiąże się z licznymi zmianami. Duża rolę zaczęły odgrywać serwisy, w których to użytkownicy są twórcami treści [3]. Witryny internetowe przestały być jedynie bazą wiedzy, którą można przeczytać i przejść dalej. Autorzy stron sprawili, że użytkownicy stali się częścią Internetu. Dobrze znanymi przykładami serwisów Web 2.0 są Facebook, czy YouTube.

Internet ułatwia życie współczesnemu człowiekowi: umożliwia robienie zakupów, wykonywanie przelewów bankowych, szybką komunikację wideo-głosową oraz inne. Pomimo dobrodziejstw jakie ze sobą niesie nie należy zapominać o potencjalnymi niebezpieczeństwie związanym z jego wykorzystaniem.

Serwisy internetowe narażone są na ataki, które przyczynić się mogą do naruszenie integralności oraz poufności danych użytkowników. Skutecznie przeprowadzony DDOS (ang. distributed denial of service) może unieruchomić serwer, co skutkuje przerwą w dostępie do udostępnianiach przez niego usług [4].

W niniejszym artykule opisane zostały najczęściej wykorzystywane metody ataków. Na przykładzie analizy przepadku aplikacji *Internetowy Notatnik* wskazano przyczyny luk w zakresie bezpieczeństwa oraz metody obrony przed atakami.

2. Klasyfikacja ataków

Dwie organizacje podjęły się problemu klasyfikacji ataków: OWASP [5] oraz WASC [6]. Porównanie klasyfikacji zostało przedstawione w tabeli 1.

Istnieje wiele artykułów naukowych poruszających tematykę bezpieczeństwa serwisów internetowych. Po wpisaniu pojęcia „web attack” w wyszukiwarce serwisu IEEE Xplore uzyskano niespełna trzy tysiące rezultatów. Próbę usystematyzowania tych informacji podjęli Jeongseok Seo i inni w swoim artykule [7]. Dowiedli oni, że do pełnej

charakterystyki ataku powinna zostać wykorzystana główna przyczyna ataku oraz jego lokalizacja.

Tabela 1. Porównanie klasyfikacji OWASP oraz WASC

	OWASP	**WASC**
Nazwa	OWASP Top 10	The WASC Threat Classification
Autor	OWASP Foundation	Web Appliaction Security Consortium
Forma reportu	Strona www	Strona www Plik pdf
Częstotliwość publikacji	Ukazały się 3 edycje, co 3 lata	Jednorazowa publikacja
Ilość opisanych zagrożeń	10 najpopularniejszych	49 zagrożeń
Informacje dodatkowe	• przykłady • metody zabezpieczeń	• przykłady • metody zabezpieczeń

2.1. Injection

Jednym z najpopularniejszych atakiem z grupy Injection jest atak SQL Injection. Polega on na wstrzyknięciu w zapytanie SQL (ang. Structured Query Language) kodu mogącego wywołać nieoczekiwane działanie [8].

Przykładowym celem może stać się formularz logowania, w którym użytkownik podając hasło i login zyskuje dostęp do chronionych zasobów. Aplikacja w celu weryfikacji poprawności hasła odwołuje się do bazy danych i pobiera dodatkowe informacje o użytkowniku. Wykonane zostaje zapytanie bazodanowe z Przykładu 1.

Przykład 1. Zapytanie pobierające dane o użytkowniku

```
SELECT * from t_user where login = '[f_login]' and passwod = '[f_password]'
```

gdzie f_login i f_password to wartość pola login i hasło z formularza logowania. Dla uproszczenia przykładu pominięto kwestię hashowania hasła.

Jeśli zapytanie zwróci niepusty rezultat aplikacja może uznać, że hasło jest poprawne i użytkownik zostanie zalogowany. W przeciwnym wypadku wyświetlony zostanie komunikat błędu. Atak SQL Injection polegać może na wpisaniu do formularza w polu hasło wartości: ' or '1'='1. Poprzez zastosowanie tautologii możliwe jest manipulowanie wynikami zwracanymi przez aplikację. Innym sposobem wykorzystywanym przez atakujących może być złączenie rezultatu zapytania z wynikiem innego zapytania (operator UNION). Kluczem do powodzenia jest wtedy dopasowanie ilości oraz typu kolumn zwracanych przez obydwa zapytania. W niektórych przypadkach niemożliwe jest uzyskanie wyniku zapytania. Jeśli rezultatem ataku jest informacja binarna mamy do czynienia z atakiem SQL Blind Injection.

Przyczyn niebezpieczeństwa tego typu należy doszukiwać się w niewystarczającej filtracji danych przekazywanych do zapytań SQL [9].

Do grupy Injection zaliczamy także Command i Log Injection [10]. Z pierwszym z tech zagrożeń mamy do czynienia podczas wykonywania polecań systemu operacyjnego przez aplikację. Drugi dotyczy mechanizmu logowania informacji. Zmodyfikowane logi aplikacji mogą utrudnić zidentyfikowanie problemów związanych z jej działaniem.

2.2. XSS

Atak XSS (ang. Cross-Site-Scripting) nierozłącznie związany jest z wykorzystaniem języków skryptowych. Obecnie trudno wyobrazić sobie stronę internetową niewykorzystującą dobrodziejstw technologii JavaScript. Zagrożenie to znalazło się na 3 miejscu na liście OWASP. Polega ono na wykonaniu niepożądanego kodu skryptowego w przeglądarce klienta. Rezultatem działania takiego kodu może być wykradzenie ciasteczek (w tym identyfikatorów sesji) oraz modyfikacja zawartości treści prezentowanych na stronie [11]. Istnieje klasyfikacja tego ataku pod względem miejsca zlokalizowania niebezpiecznego kodu. Rodzaje ataków:

- stored XSS – niebezpieczny kod przechowywany jest przez aplikację (np. w bazie danych),
- reflected XSS – niebezpieczny kod przekazywany jest poprzez parametry GET w spreparowanym przez atakującego adresie URL.

Ponownie zagrożenie zostanie zobrazowane przykładem. Popularnym elementem współczesnych serwisów internetowych są wyszukiwarki. Dane wpisywane w pola wyszukiwania po zatwierdzeniu kryteriów wyszukiwania znajdują się w parametrach GET. Strona prezentująca rezultat wyszukiwania odczytuje te dane oraz prezentuje wartość parametru. W standardowym scenariuszu nie może tu dojść do niebezpieczeństwa. Problem pojawić się może w wypadku gdy atakujący przygotuje URL zawierający szkodliwy kod JavaScript. Kod taki może przesłać dane o atakowanym bezpośrednio na serwer atakującego (Przykład 2).

Przykład 2. Zapytanie pobierające dane o użytkowniku

```
document.write('<img src="http://strona_atakujacego.pl/?stolen_cookie=' + document.cookie + '" />')
```

W powyższym przykładzie zaprezentowano kod, który umożliwia przeprowadzenie ataku. Gdy kod wykona się w przeglądarce zostanie wysłany request GET na zewnętrzny serwer, a w parametrach zostaną przekazane wrażliwe dane.

Zapobieganie atakowi możliwe jest w dwojaki sposób. Pierwszy polega na filtrowaniu danych mających zostać zapisane do bazy danych. Filtracja powinna uniemożliwić (lub zneutralizować) zapis kodu skryptowego. Inną możliwością jest filtrowanie danych na poziomie wyświetlania. W tym celu mogą zostać wykorzystane dodatkowe biblioteki (np. ESAPI [12]) lub mechanizmy oferowane w standardowej bibliotece języka programowania (np. funkcja htmlspecialchars w php).

Najpopularniejszą udaną próbą wykonania ataku tego typu jest historia Samyego Kamkara [13]. Przygotowany przez niego skrypt został umieszczony w serwisie Myspace. Powodował on wysłanie zaproszenia do znajomych w imieniu zaatakowanego użytkownika. Atakujący w krótkim czasie zyskał ponad milin znajomych w serwisie. Luka w zabezpieczeniach została szybko zauważona i naprawiona.

2.3. Uwierzytelnianie

Jest to proces polegający na potwierdzeniu przez serwer tożsamości klienta. W przypadku aplikacji internetowych odbywa się zazwyczaj przy pomocy hasła i loginu. Niekiedy wykorzystywane są także dodatkowe mechanizmy:

- tokeny,
- weryfikacja SMS,
- listy haseł jednorazowych.

Strona OWASP opisuje szereg dobrych praktyk związanych z tym procesem. Obejmują one zagadnienia przechowywania, częstotliwości zmian, zasad przypominania oraz złożoności haseł. Autorzy zwracają uwagę na ataki związane z sesją, czyli z mechanizmem odpowiedzialnym za zapamiętanie stanu aplikacji w ramach każdego z klientów. Jednym z zagrożeń jest Session fixation. Scenariusz takiego ataku może polegać na zmuszeniu użytkownika do wykorzystania znanego atakującemu identyfikatora sesji, poprzez przekazanie go w adresie URL. Nie wszystkie serwery aplikacyjne przechowują id sesji w parametrach GET. W momencie gdy ofiara ataku wejdzie na stronę poprzez taki link i zaloguje się atakujący zyska dostęp do jego uprawnień. Rozwiązanie tego ataku zostało wprowadzone w niektórych serwerach aplikacyjnych (np. Apache Tomcat 6.0.23). Polega ono na automatycznym tworzeniu nowej sesji w momencie logowania.

Odrębną kwestę stanowi sposób przechowywania haseł użytkowników. Niedopuszczalne jest przechowywanie plain textu. Obecnie standardem jest wykorzystanie algorytmu bcrypt. Od wersji 5.5.0 języka php istnieją funkcje dające możliwość hashowania oraz weryfikacji poprawności hasła przy pomocy wspomnianego algorytmu (password_hash, password_verify [14]).

2.4. Niezabezpieczone bezpośrednie odniesienia do obiektów

Zazwyczaj zasoby przechowywane w tabelach bazy danych identyfikowane są przez klucze główne. Stanowią one ich identyfikatory. W przypadku gdy część zasobów nie powinna być dostępna dla wszystkich użytkowników serwisu może dojść do ataku tego typu. Polega on na nieautoryzowanym dostępie do treści. Identyfikatory zasobów mogą być przekazywane do aplikacji poprzez parametry GET adresu. Np.:

zasob.php?id=12

Atakujący może zmienić wartość parametru id z zamiarem uzyskania dostępu do zasobów innych użytkowników. Aplikacja powinna wykryć próbę takiej manipulacji. Weryfikacja uprawnień do odczytu danego zasobu może odbyć się poprzez zaimplementowanie mechanizmu ACL (ang. access control list).

3. Badania

Przedmiotem badań była aplikacja *Internetowy Notatnik* autorstwa Michała Furtaka. Zaimplementowana została ona w języku php. Daje możliwość rejestracji użytkowników oraz tworzenia notatek. Zapisywane są one w bazie MySql. Notatki domyślnie widoczne są tylko dla ich autorów. Dodatkowo udostępniona została możliwość dzielenia się z innymi swoimi notatkami poprzez wysłanie linków prowadzących do nich.

3.1. Środowisko testowe

W badaniach wykorzystano aplikację uruchomioną na środowisku lokalnym. Zastosowano serwer aplikacji Apache 2.4 oraz serwer baz danych MySql 5.6.25. Do przygotowania aplikacji testowych użyto IDE NetBeans. Wykorzystano dodatkowe aplikacje:

- sqlmap,
- Fiddler 4,
- przeglądarka Firefox z pluginem Firebug.

Całość pracowała pod kontrolą systemu Windows 7 Professional.

3.2. Przebieg badań

Przeprowadzono szereg testów mający na celu wykluczyć lub potwierdzić podatność aplikacji na ataki.

Testy SQL Injection

Przeprowadzono próbę ataku przy pomocy narzędzia sqlmap. Następnie na podstawie badań literaturowych określono cztery rodzaje wstrzyknięć, które spowodować mogą naruszenie bezpieczeństwa aplikacji:

- tautologie,
- usuwanie zawartości tabeli (polecenia delete, truncate),
- operator UNION,
- test na Blind SQL Injection.

Dodatkowo przygotowane zostało pięć aplikacji pomocniczych, które pozwoliły na przetestowanie mechanizmów obrony, których nie wykorzystano w *Internetowym Notatniku*. Przetestowano skuteczność:

- mechanizmu Prepared Statements (w wersjach: poprawnie zaimplemetowanej (I) i błędnej (II)),
- frameworka Hibernate (Przykład 3),
- frameworka Medoo,
- biblioteki ESAPI.

Przykład 3. Program testujący framework Hibernate

```
public static void main(String[] args) {

Session session =
   SessionFactory.getSessionFactory().openSession();
```

```
    doTest(session, "17 OR true=true --");
    doTest(session, "17 OR 'test' = 'test'");
    doTest(session, "17 ; drop table test");
    doTest(session, "17 ; truncate table test");
    doTest(session, "20 and VERSION( ) = '5.6.25'");
    doTest(session, "17 UNION select id, null, name as
 'title', PASSWORD as 'content', null, null, null from
 user");
    session.close();
}

public static void doTest(Session session, Object param) {
    System.out.println("=====  Test: " + param + "
 =====");
    try {
        List<Note> notes = session.createQuery("FROM
 Note where id = :id ")
                .setParameter("id", param).list();
        for (Note note : notes) {
            System.out.println(note);
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

Testy XSS

Wytypowane zostały miejsca potencjalnego ataku: strona tworzenia nowego użytkownika oraz dodawania notatki. Przeprowadzono test przy pomocy udostępnionych skryptów JavaScript. Dodatkowo wykonano test mechanizmu blokującego ataki XSS w przeglądarkach internetowych. Do tego celu przygotowano stronę testową wyświetlającą zawartość parametru GET. Dokonano próby przekazania w parametrze kodu JavaScript.

Testy mechanizmu uwierzytelniania

Metodą eksperymentalną sprawdzono w jaki sposób aplikacja weryfikuje złożoność haseł podczas rejestracji. Dokonano próby przeprowadzenia ataku Session Fixation. Zweryfikowano w jaki sposób przechowywane są hasła w bazie danych.

Testy podatności na ataki związane z bezpośrednim odwołaniem do obiektu

Przygotowano aplikację testową wysyłającą dużą ilość requestów do aplikacji. Celem było odnalezienie identyfikatorów notatek innych użytkowników.

3.3. Prezentacja rezultatów badań

Testy SQL Injection

Testy wykazały ze mechanizmy obronne wykorzystane w aplikacji Internetowy Notatnik spełniły swoje cele. Nie udało się przeprowadzić skutecznego ataku na tej aplikacji.

Wyniki dotyczące porównania mechanizmów obrony zostały przedstawione w tabeli 2.

Tabela 2. Porównanie mechanizmów obrony przed SQL Injection

	Prepared Statements (wersja I)	Prepared Statements (bwersja II)	Hibernate	Medoo	ESAPI
tautologie	□	□	□	□	□
usuwanie zawartości tabeli	□	□	□	□	□
operator UNION	□	□	□	□	□
Blind SQL Injection	□	□	□	□	□

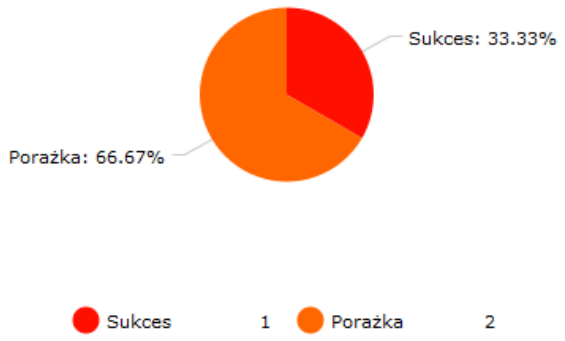
Wszystkie poprawnie wykorzystane metody dają gwarancję bezpieczeństwa. Nieprawdą jest, że sama metoda przesądza o tym, że aplikacja jest bezpieczna. W przypadku testu aplikacji gdzie w niepoprawny sposób wykorzystano Prepered Statemants przyczyny potencjalnego ataku należy dopatrywać się w błędach programisty implementującego serwis.

Testy XSS

Testy wykazały, że strona dodawania notatki umożliwia skuteczne przeprowadzenie ataku XSS typu pierwszego. Odpowiednio preparując request możliwe okazało się wyświetlenie komunikatu zawierającego identyfikator sesji użytkownika:

/add.php?title="%2F><script>+alert(document.cookie%3B+<
%2Fscript>&content=&submit=

Łącznie przeprowadzono trzy próby wykonania ataku XSS. Tylko powyższa zakończyła się sukcesem (Rys. 1).



Rys. 1. Skuteczność przeprowadzonych ataków XSS

Testy mechanizmu uwierzytelniania

Analiza kodu pozwoliła stwierdzić, że wykorzystywany jest hash md5. W celu poprawy bezpieczeństwa

przygotowano poprawkę. Obejmuje ona zmianę sposobu hashowania na taką, która wykorzystuje algorytm bcrypt.

Dodatkowo zaproponowano ulepszenie polegające na wprowadzeniu validatora weryfikującego złożoność hasła. Nowy mechanizm uwzględnia nie tylko długość wprowadzonego ciągu ale także zawartość znaków z różnych grup (małe i wielkie litery, cyfry, znaki specjalne).

Atak Session Fixation zakończył się sukcesem. Wykazano, że możliwe jest przejęcie sesji innego użytkownika. Zaproponowano rozwiązanie tworzące nową sesję bezpośrednio przed zalogowaniem użytkownika. Rozwiązanie to przyczyniło się do wyeliminowania tej nieprawidłowości.

Testy podatności na ataki związane z bezpośrednim odwołaniem do obiektu

Testy nie wykazały nieprawidłowości. Aplikacja weryfikuje uprawnienia dostępu do obiektu.

4. Wnioski

Badania wykazały, że nawet tak prosta aplikacja, jaką jest *Internetowy Notatnik* może zawierać błędy. Wykazane zostały nieprawidłowości z zakresu sposobu przechowywania hasła, zarządzania sesją oraz podatności na ataki XSS. Dla wszystkich tym problemów zostało opracowane rozwiązanie, które zostało wdrożone w aplikację.

Przyczyn podatności aplikacji na ataki doszukiwać się można w nieprawidłowym wykorzystaniu narzędzi mającym im zapobiegać (lub ich nie wykorzystaniu). Osobną kwestią, na którą należy zwrócić uwagę jest środowisko produkcyjne aplikacji. Konieczne jest instalowanie aktualizacji serwera aplikacyjnego oraz wszystkich innych komponentów systemu.

Najsłabszą rolę w systemie stanowi użytkownik. Na nic nie zdadzą się zabezpieczenia jeśli ten nie będzie świadomy czekających na niego zagrożeń. Rolą developera jest wymuszenie pewnych zachowań takich jak okresowa zmiana hasła.

Literatura

[1] R. Cohen-Almagor, Internet History, International Journal of Technoethics, 2011
[2] Netcraft, Web Server Survey, 2014
[3] R. Krzyżaniak, Web 2.0 w Polsce, 2007
[4] J. Mirković i inni, Attacking DDoS at the Source, 2002
[5] https://www.owasp.org/ [20.11.2016]
[6] http://projects.webappsec.org/ [20.11.2016]
[7] S. Jeongseok, Web server attack categorization based on root causes and their locations, 2004
[8] K. Navdeep, Modeling a SQL injection attack, 2015
[9] M. Qbea'h, Detecting and Preventing SQL Injection Attacks: A Formal Approach, 2016
[10] http://www.jtmelton.com/2010/09/21/preventing-log-forging-in-java/ [20.11.2016]
[11] M. K. Gupta i inni, Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications, 2015
[12] A. Singh i inni, A Survey on XSS web-attack and Defense Mechanisms, 2014
[13] L. Franceschi-Bicchierai, The MySpace Worm that Changed the Internet Forever, 2015
[14] http://php.net/manual/en/function.password-hash.php [20.11.2016]

JCSI 3 (2017) 6-10
Journal of Computer Sciences Institute

Wysłane: 2016-11-23
Przyjęte: 2016-11-30

Wykorzystanie Node.js w tworzeniu aplikacjach sterowanych zdarzeniami

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

Streszczenie. W niniejszej publikacji omówiona została nowa technika tworzenia oprogramowania – programowanie sterowane zdarzeniami. Została ona porównana z innymi popularnymi technikami tworzenia serwisów w celu ujawnienia słabych punktów oraz sprawdzeniu, w jakich obszarach aplikacji internetowych nadaje się do stosowania.

Słowa kluczowe: node.js; programowanie zdarzeniowe

*Autor do korespondencji.

Adres e-mail: vladyslav.gryn@gmail.com

Using of Node.js in creating application based on event-driven architecture

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

Abstract. This paper discusses a new programming method - event driven programming. This method is compared with other popular ways of implementing web services to find its weak points and discover in which areas of modern web applications it could be implemented.

Keywords: node.js; event-driven programming

*Corresponding author.

E-mail address: vladyslav.gryn@gmail.com

1. Wstęp

W związku z dynamicznym rozwojem Internetu oraz wykorzystywanych w nim technologii, aplikacje internetowe dostępne w sieci zaznały również dużych zmian. Wraz z ich rozwojem powstało wiele mechanizmów służących do tworzenia oprogramowania [1]. Jednym z takich mechanizmów jest programowanie sterowane zdarzeniami, na którym bazuje architektura platformy Node.js wykorzystywana w niniejszej publikacji [2].

Na dzień dzisiejszy istnieje wiele aplikacji internetowych wykorzystujących nową technikę tworzenia oprogramowania – programowanie sterowane zdarzeniami. Jej działanie polega na uruchomieniu programu obsługi zdarzeń w chwili wystąpienia odpowiedniego zdarzenia. Zdarzenia te wywoływane są asynchronicznie w stosunku do działania programu [3].

Platforma Node.js wykorzystywana jest w wielu produktach wytwarzanych i wspieranych przez znane firmy takie jak Yahoo czy Google. Osiągnięcie takiej popularności stało się możliwe dzięki jej wydajności oraz systemu paczek npm (*ang. node package manager)* [4]. Platforma ta stosowana jest zazwyczaj w serwisach z wysokim obciążeniem tzn. dużą liczbą odsłon strony. Na dzień dzisiejszy platforma programistyczna Node.js jest jedną z najbardziej popularnych platform tworzenia oprogramowania na runku i jej popularność wciąż rośnie [5].

W niniejszej publikacji przedstawione zostało porównanie dwóch technologii tworzenia aplikacji internetowych: PHP oraz Node.js. Każda z tych technologii ma swoje zastosowanie w różnych obszarach i bazuje na różnych paradygmatach programowania. Głównym celem jest zbadanie zachowania stworzonej aplikacji i w jej oparciu przeprowadzenie różnych testów, co pozwoli na ujawnienie słabych punktów, a następnie dokonanie analizy porównawczej. Po przeprowadzeniu analizy będzie można odpowiedzieć na pytanie, czy platforma Node.js oparta o technikę tworzenia oprogramowania „sterowanie przez zdarzenia" jest wydajna i czy nadaje się do serwisów z dużym przepływem danych oraz dużą liczbą użytkowników.

2. Opis aplikacji testowych

Aby móc dokonać analizy porównawczej powstały dwie małe testowe aplikacje internetowe napisane w popularnych technologiach tworzenia oprogramowania – PHP i Node.js [6]. Aplikacje zostały zaimplementowane w czystym kodzie PHP oraz JavaScript bez użycia żadnych frameworków, które łączą się ze wspólną bazą danych postawioną na PostgreSQL. Baza danych zawiera trzy tabele, które zostały wypełnione danymi. Po stworzeniu takich aplikacji przeprowadzone zostały różne testy.

Każda z zaprojektowanych aplikacji posiada listę funkcjonalności, które posłużą do przeprowadzenia badań.

Główną funkcjonalnością aplikacji jest posiadanie prostego interfejsu graficznego umożliwiającego wykonanie różnych operacji. Poniżej przedstawiona została lista dostępnych akcji.

Wyświetlanie danych – przykładowa aplikacja łączy się z bazą danych w celu pobrania oraz wyświetlenia różnej ilości informacji. Dla danego przypadku stworzone zostały trzy proste akcje:

- wyświetlanie listy użytkowników – stworzona została tabela zawierająca tysiąc rekordów;
- wyświetlanie stanowiska prac użytkowników – tabela zawiera 10000 rekordów;
- wyświetlanie listy artykułów użytkowników – w tym celu powstała tabela posiadająca 50000 rekordów.

Wyszukiwanie danych – akcja polega na wyszukaniu najbardziej popularnego artykułu

Filtrowanie danych – wyświetlanie użytkownika z pensja większą od pensji średniej wśród wszystkich użytkowników

Wstawianie danych – zadaniem akcji jest wstawienie stu rekordów do tabeli zawierającej informacje o użytkownikach

Operacje na tablicach – w tym przypadku wykorzystany został algorytm sortowania bąbelkowego, który przyjmuje tablice posiadającą 10000 losowych liczb i sortuje ją rosnąco

Operacje na plikach – aplikacja odwołuje się do API systemu operacyjnego w celu pobrania zawartości przykładowego folderu, a następnie wyświetlenie rozmiarów plików znajdujących się w tym katalogu. Dana operacja wywoływana została tysiąc razy

Operacja na tekście – zadaniem aplikacji jest wygenerowanie 10000 unikatowych ciągów znaków za pomocą różnych metod dostępnych w obu technologiach np. długość tekstu czy generowanie znaku.

3. Realizacja badania

W przedstawionej publikacji przeprowadzone zostały dwa eksperymenty, a następnie poddane analizie. Pierwszy eksperyment polegał na zbadaniu szybkości działania aplikacji napisanej w języku programowania PHP. Drugi eksperyment zawierał identyczne czynności jak w pierwszym przypadku, tylko że aplikacja została zaimplementowana przy użyciu platformy Node.js. Wszystkie testy wykonano na serwerze działającym lokalnie.

Każdy eksperyment polegał na wykonaniu dziewięciu różnych akcji. Pierwsze trzy akcje pobierały oraz wyświetlały różną ilość informacji z bazy danych. Kolejne trzy akcje polegały na wykonaniu różnych obliczeń oraz odwołań do funkcjonalności operacyjnego systemu. Ostatnie trzy akcje miały do czynienia z bazą danych - wykonane zostały takie działania jak: wyszukiwanie, filtrowanie oraz wstawianie danych do bazy.

Dla każdej akcji zostały wykonane trzy pomiary mierzenia czasu oczekiwania odpowiedzi z serwera, a następnie obliczona została wartość średnia. Warto zwrócić uwagę na przechowywanie danych w przeglądarkach – może to mieć duży wpływ na otrzymane wyniki, dlatego przy każdym teście czyszczona była pamięć podręczna przeglądarki.

4. Analiza badań

Porównane zostały dwie technologie tworzenia stron internetowych PHP oraz Node.js i przeprowadzono analizę otrzymanych wyników. Porównanie dotyczy takich aspektów aplikacji jak:

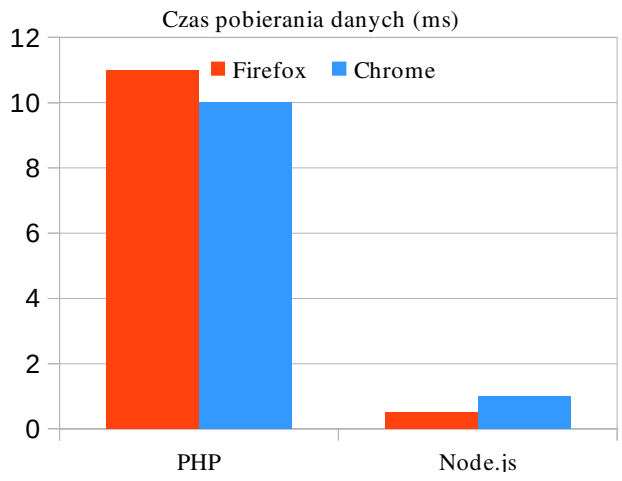
- szybkość pobierania żądania przez serwer, tworzenia odpowiedzi oraz przesłania danych w odpowiednim formacie;
- wykorzystanie zasobów fizycznej maszyny.

Celem przeprowadzenia analizy jest uzyskanie zalet oraz wad wykorzystanych technologii oraz sprawdzeniu, która z nich jest lepszy i przynosi większe korzyści w serwisach, w których głównym aspektem jest ich wydajność.

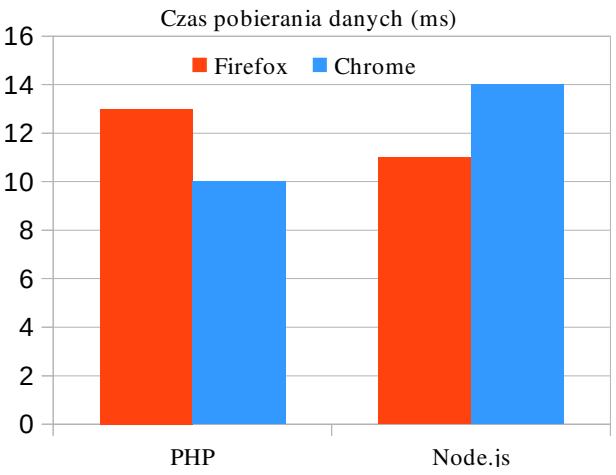
Poniżej został zaprezentowany szereg różnych testów. Testy polegały na zmierzeniu czasu odpowiedzi serwera. W celu przeprowadzenia analizy i otrzymania wyników, testy zostały podzielone na osiem etapów.

W pierwszym etapie przeprowadzono test wyświetlenia strony głównej, która zawiera listę dostępnych akcji. Jak widać na poniższym rysunku (Rys. 1) najlepszy czas osiągnięty został przez platformę Node.js. Czas potrzebny na wczytanie strony głównej jest w dziesięć razy mniejszy niż jest to w przypadku PHP.

W drugim etapie aplikacja testowa miała do czynienia z bazą danych. Testy zostały przeprowadzone przy użyciu trzech różnych akcji. Pierwszy test dotyczył wyświetlenia listy użytkowników (Rys. 2). W otrzymanych wynikach widać, że obie technologie są na podobnym poziomie, natomiast wydajność operacji związanych z pobieraniem danych z bazy przy użyciu platformy Node.js nie są zbyt szybkie i przy większym przepływie danych PHP sprawuje się dużo lepiej, co zostało zaobserwowane w kolejnych testach.



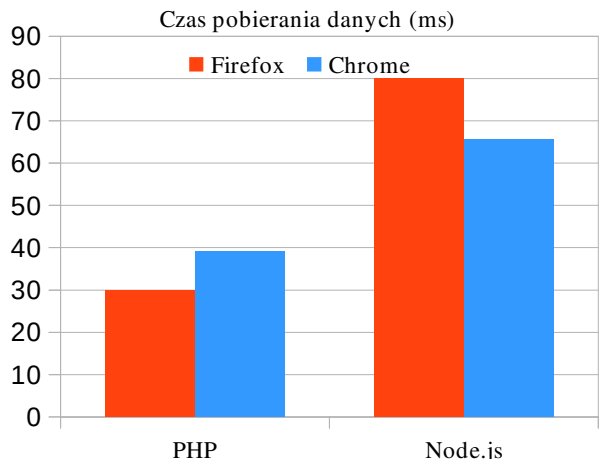
Rys. 1. Szybkość wczytywania strony głównej



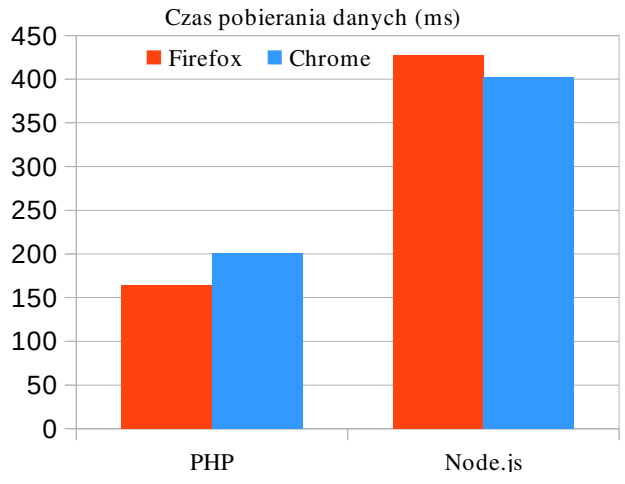
Rys. 2. Szybkość wczytywania listy użytkowników

W kolejnych testach można zauważyć, że przy większej ilości informacji pobieranych z bazy danych Node.js radzi sobie dużo gorzej niż PHP (Rys. 3 oraz Rys. 4). Dzieje się tak, ponieważ wykorzystywane biblioteki oraz połączenia z bazą danych mogą znacznie obniżyć wydajność platformy Node.js. Każda z nich mogła zostać zaimplementowana na różne sposoby lub przy tworzeniu przykładowej aplikacji mogły zostać popełnione różnego rodzaju błędy.

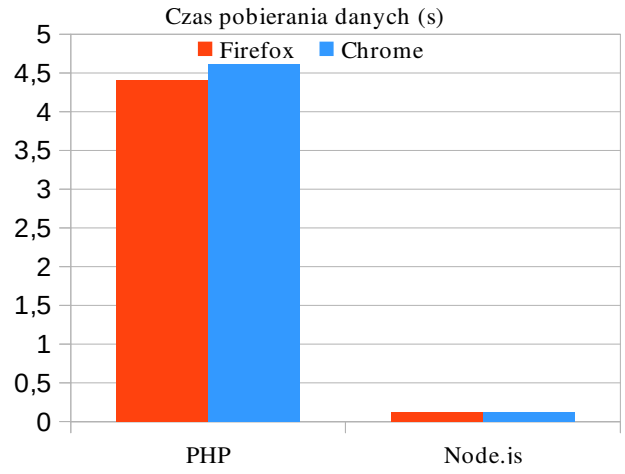
W trzecim etapie aplikacja testowa miała do czynienia z sortowaniem bąbelkowym. Taki typ testu potrafi mocno obciążyć procesor, co z kolei ma wpływ na wydajność aplikacji. Obserwując poniżej zamieszczony rysunek (Rys. 5) można stwierdzić, że Node.js potrafi dużo szybciej przetworzyć ciężkie operacje i sprawniej wykorzystuje moc obliczeniową procesora. Różnica w czasie jest ogromna – Node.js dokonał sortowania tablicy w 0,13 ms, a PHP prawie w 5s.



Rys. 3. Szybkość wczytywania listy stanowisk prac



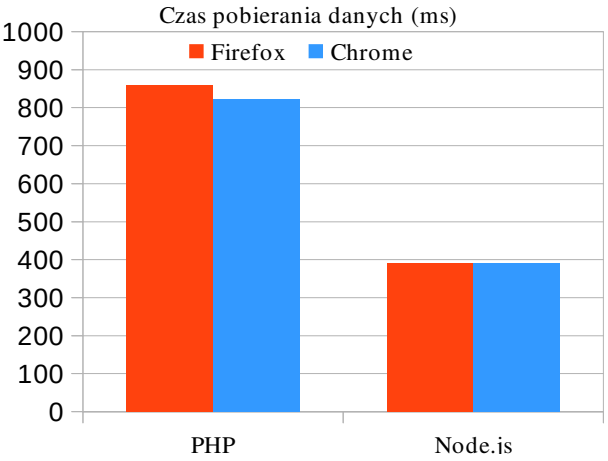
Rys. 4. Szybkość wczytywania listy artykułów



Rys. 5. Szybkość porządkowania tablicy liczb losowych

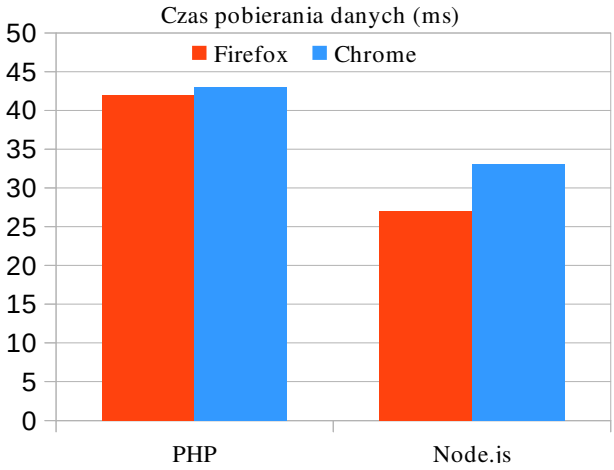
Czwarty etap polegał na odwołaniu się do udostępniających usług systemu operacyjnego – pobranie listy plików z przykładowego folderu, a następnie uzyskanie

rozmiaru dla każdego pliku. Obserwując rysunek (Rys. 6) można wnioskować, że platforma Node.js potrafi w szybki sposób obsłużyć odwołania do systemu plików, chociaż realizacja tej operacji w języku programowania PHP też jest na wysokim poziomie (różnica czasów wynosi około 400 ms na korzyść platformy Node.js).



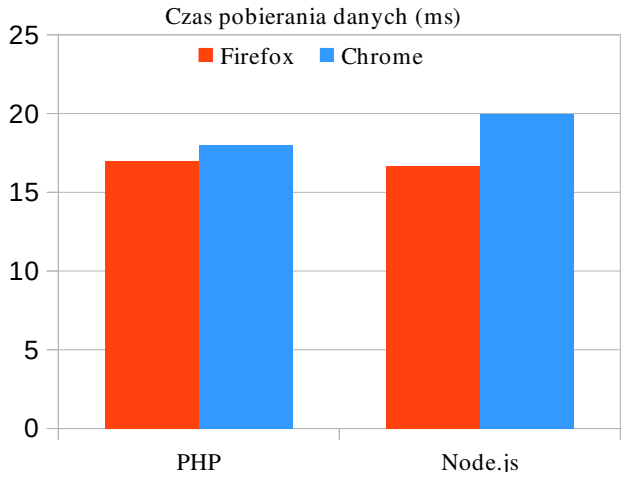
Rys. 6. Szybkość pobrania listy plików

W kolejnym piątym etapie badania obie technologie zostały przetestowane względem szybkości wykonania operacji na tekście. Rysunek 7 świadczy o tym, że aplikacje wygenerowały tekst w bardzo krótkim czasie i śmiało można stwierdzić, że działania takiego typu nie są specjalnie trudne dla obu technologii tworzenia stron internetowych.

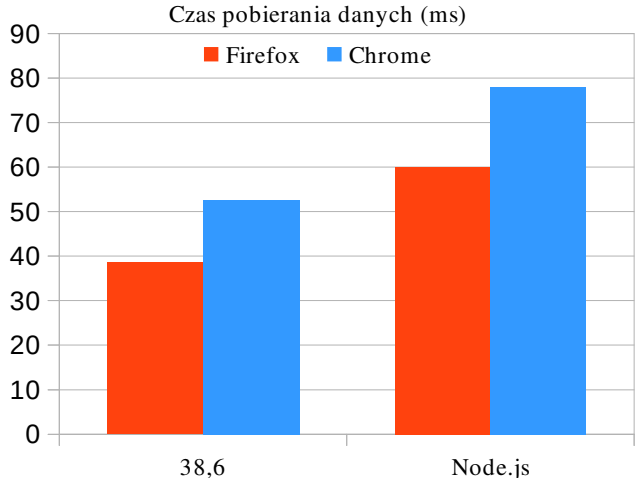


Rys. 7. Szybkość wykonania operacji na tekście

W kolejnych dwóch etapach aplikacje znowu miały do czynienia z bazą danych. Tym razem zadanie polegało na wyszukiwaniu danych – szukanie najpopularniejszego artykułu oraz filtrowanie danych pod kątem pensji użytkowników. Z otrzymanych rezultatów widać, że w przypadku mniejszej ilości danych pobieranych z bazy, czas odpowiedzi z serwera dla obu technologii jest podobny (Rys. 8). Ciekawa sytuacja występuje podczas pobierania większej ilości danych. Aplikacja oparta o platformę Node.js ma duży spadek wydajnościowy, co można zauważyć na rysunku 9. Jak wspomniałem wcześniej spowodowane to może być wieloma faktorami.



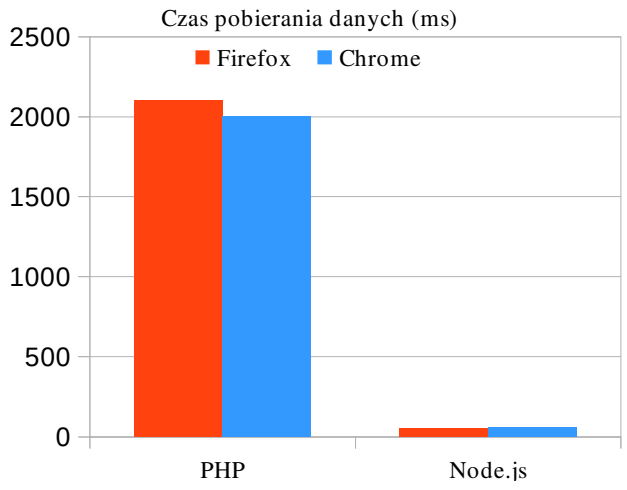
Rys. 8. Szybkość wyszukiwania artykułu



Rys. 9. Szybkość filtrowania użytkowników

Ostatni etap polegał na wstawianiu danych do tabeli. Zapytanie SQL wykonane zostało sto razy. Na przedstawionym poniżej rysunku widać, że czas osiągnięty przez platformę Node.js jest 52 razy mniejszy niż w przypadku PHP (Rys. 10). Dzięki asynchronicznemu modelowi Node.js nie czeka na wykonanie zapytania, tylko wykonuje następne [7]. Wszystkie takie operacje wykonywane są równolegle. W przypadku aplikacji napisanej w PHP wywołanie kolejnego zapytania nastąpi w chwili zakończenia poprzedniego [8]. Właśnie dlatego Node.js jest zalecany do stosowania w serwisach z dużą liczbą użytkowników.

Przy przeprowadzeniu wyżej opisanych testów ważnym aspektem było wykorzystanie dostępnych zasobów maszyny fizycznej. Zbyt duże ich zużycie potrafi spowolnić działania systemu, a czasami nawet do zatrzymania działania aplikacji.



Rys. 10. Szybkość wstawiania danych

Ważnym podzespołem serwera jest procesor. Szybkość działania operacji zależy od charakterystyk procesora oraz jego wykorzystania. Jak widać na zamieszczonej poniżej tabeli 1, serwer platformy Node.js przy 10000 jednoczesnych odwołaniach do strony głównej wykorzystuje tylko jeden z czterech dostępnych rdzeni i zużywa około 15% dostępnej mocy obliczeniowej procesora, natomiast serwer Apache przy identycznym teście wydajnościowym obciąża wszystkie cztery rdzenia procesora i wykorzystuje około 70% dostępnej mocy.

Kolejnym ważnym podzespołem serwera jest pamięć RAM. Obserwując tabele 1 można zauważyć, że platforma Node.js posiada nieco gorsze wyniki niż PHP. W przypadku technologii PHP wykorzystanie pamięci RAM w trakcie wykonywania testu nie zmieniło się, natomiast Node.js wykorzystał około 4% dodatkowej pamięci RAM. Pomimo to, ze względu na wydajność nowoczesnych serwerów (ilość dostępnej pamięci RAM), tak małe różnice nie będą miały dużego wpływu na wydajność aplikacji.

Tabela 1. Zużycie zasobów maszyny fizycznej

Serwer	**Obciążenie procesora %**				**Wykorzystanie pamięci RAM %**
	1 CPU	**2 CPU**	**3 CPU**	**4 CPU**	
Apache	68	65	68	65	35
Node.js	18	5	6	5	39

5. Wnioski

Platforma Node.js zdobywa coraz większą popularność wśród programistów. W ciągu ostatnich kilku lat platforma została zaakceptowana przez znane firmy i wykorzystywana w ich produktach [9].

Po analizie przeprowadzonych testów śmiało można stwierdzić, że Node.js jest naprawdę wydajny. W porównaniu z serwerem Apache, Node.js działa znacznie szybciej, ale dzięki tak dużej wydajności serwer platformy Node.js zużywa więcej zasobów komputera niż serwer Apache, a dokładnie pamięci RAM. Warto dodać, że Node.js nie jest prosty w nauczeniu, ponieważ architektura platformy bazuje na asynchronicznych działaniach i początkujący programista może popełnić wiele błędów, które spowodują duże zużycie zasobów maszyny fizycznej, a nawet doprowadzić zmniejszenia wydajności zaprojektowanej aplikacji.

Jeśli chodzi o wybór między tymi platformami to warto pamiętać, że Node.js nie będzie potrafił zastąpić PHP we wszystkich przypadkach, rozwiązać wszelkie problemy z architekturą aplikacji lub poprawić szybkość działania aplikacji przy dużym obciążeniu. Jeżeli wybór padł na użycie platformy Node.js w nowym projekcie lub na przepisanie starego, najpierw należy zastanowić się czy wybrana platforma będzie spełniać wszelkie wymagania projektu i czy będzie ona stanowić dobre rozwiązanie. Również w zależności od planowanych funkcjonalności aplikacji, wykorzystywanych bibliotek, połączeń z bazą danych, wydajność platformy Node.js może znacznie się różnić.

Podsumowując można stwierdzić, że architektura ma bardzo duże znaczenie dla wydajności całego systemu. Potencjał platformy Node.js i jej zastosowań dobrze pokazuje lista dostępnych modułów. Dostarczenie serwera z asynchronicznym I/O sprawia, że wielokrotne odpytywanie bazy nie prowadzi do tworzenia kolejnych operacji, ponieważ wykonywane są one równolegle [10]. W czasie oczekiwania na odpowiedź z bazy danych lub innych usług, serwer platformy Node.js potrafi bez żadnego problemu przyjmować kolejne połączenia i generować dla nich odpowiedzi. Dzięki temu Node.js potrafi ekonomicznie wykorzystywać zasoby komputera, a przede wszystkim daje duży przyrost wydajności. Warto również dodać, że używanie platformy Node.js w projekcie zależy od wiele czynników: specyfikacji projektu, doświadczenia programisty, a użycie platformy nie zawsze rozwiąże wszelkie problemy.

Literatura

[1] D. M. Simmonds, The Programming Paradigm Evolution, 2012
[2] C. Gackenheimer, Understanding Node.js, 2013
[3] C. J. Ihrig, Pro Node.js for Developers, 2013
[4] B. Syed, Beginning Node.js, 2014
[5] A. Mardan, Practical Node.js Building Real-World Scalable Web Apps, 2014
[6] K. Lei, Y. Ma, Z. Tan, Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js, 2014
[7] D. Vohra, Using Node.js, 2015
[8] S. Nakajima, An Architecture of Dynamically Adaptive PHP-based Web Applications, 2011
[9] S. Tilkov, S. Vinoski, Node.js: Using JavaScript to Build High-Performance Network Programs, 2010
[10] A. Ojamaa, K. Düüna, Assessing the security of Node.js platform, 2012

JCSI 3 (2017) 11-17
Journal of Computer Sciences Institute

Wysłane: 2016-12-11
Przyjęte: 2016-12-14

Analiza skuteczności metod wprowadzania tekstu z użyciem mobilnego komunikatora internetowego

Rafał Kacprzak*, Piotr Kaniewski*, Maria Skublewska-Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Tematyką badaną w pracy jest porównanie metod wprowadzania tekstu w mobilnym komunikatorze sieciowym. Analizowano szybkość wprowadzania tekstów, a także liczbę błędów popełnianych przez użytkowników za pomocą badanej metody. Porównywano metody wprowadzania tekstu przy użyciu klawiatury QWERTY, techniki Swype, pisma odręcznego oraz poleceń głosowych. Badania przeprowadzono wśród dwóch grup respondentów, z podziałem na wiek uczestników. Dokonano charakterystyki wybranych metod wpisywania tekstu w komunikatorze internetowym. Na potrzeby artykułu został opracowany mobilny komunikator.

Słowa kluczowe: analiza wprowadzania tekstu w urządzeniu mobilnym; QWERTY; metoda Swype; głosowe wprowadzanie tekstu; odręczne wprowadzanie tekstu

*Autor do korespondencji.

Adresy e-mail: rafal1254@gmail.com, piotr.kaniewski1@gmail.com

Analysis of the effectiveness of text input methods using the mobile network communicator

Rafał Kacprzak*, Piotr Kaniewski*, Maria Skublewska-Paszkowska
[a] Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The topics being considered in the study is to compare the text input methods in the mobile network communicator. Analyzed the speed of text entry as well as the number of mistakes made by users via the tested methods. Compared the methods of entering text using the QWERTY keyboard, Swype Technology, Handwriting and voice commands. The study was conducted among the two groups of respondents, by age of the participants. There have been characteristics of the selected text input methods in the mobile network communicator. On the needs of the article was developed mobile communicator.

Keywords: analysis of text input on a mobile device; QWERTY; Swype method; voice input text; handwritten text entry

*Corresponding author.

E-mail addresses: rafal1254@gmail.com, piotr.kaniewski1@gmail.com

1. Wprowadzenie

Na światowym rynku technologii urządzenia mobilne odgrywają coraz większe znaczenie. Jedną z podstawowych funkcji, do których używa się smartphone'y, to pisanie wiadomości SMS, e-mail lub innych komunikatów sieciowych. Nowoczesne technologie prezentowane przez producentów oprogramowania umożliwiają coraz szybsze i wygodniejsze, a tym samym przyjemniejsze, użytkowanie urządzeń mobilnych. Czołowe firmy w świecie IT cały czas ulepszają swoje produkty. Dostosowują swoje metody wprowadzania tekstu zarówno dla początkującego, jak i zaawansowanego użytkownika.

2. Metody wprowadzania tekstu z użyciem smartfonów

Technologia ekranu dotykowego została zaimplementowana w smartphone'ach przez Simon IBM w roku 1993 [1]. Od tego czasu, wraz z wynalezieniem oprogramowania multitouch (ang. wielopunktowe rozpoznawanie dotyku na ekranie), wydaje się, że możliwości interakcji urządzeń mobilnych są znacznie zwiększone. Wprowadzanie tekstu za pomocą ekranu dotykowego pozwala na efektywniejsze wykorzystanie urządzenia. Typ wejścia i orientacji interfejsu wejściowego można łatwo dostosowywać do preferencji użytkownika. Dodatkowo metody ekranowe zazwyczaj zawierają dużą ilość sygnalizacji wizualnej, w tym zmiany w wyglądzie podczas użycia. Taki stan zauważyć można na przykład, gdy pojawiają się powiększone okienka jako powiadomienie zaznaczenia bądź wizualny "szlak" jako odzwierciedlenie najazdu palca na wirtualne przyciski.

Istnieją pewne wady metod ekranowych. Zarówno wibracyjne powiadomienia i dźwiękowe sygnały mogą być włączone w ustawieniach urządzenia, ale mogą być nieakceptowalne jako substytuty fizycznych przycisków. Inna wada tego rozwiązania to problem wyboru małych przycisków dla osób posiadających duże palce. Również zjawisko paralaksy, czyli odczuwalnej zmiany w położeniu obiektu, która może być spowodowana przez kąt widzenia użytkownika i rozbieżności pomiędzy wyświetlanym, a faktycznym miejscem matrycy dotykowej ekranu, może zmniejszyć dokładność wpisywania. Na szczęście, pojemnościowe ekrany dotykowe zminimalizowały kwestię paralaksy.

Celem artykułu jest porównanie metod wprowadzania tekstu przy użyciu klawiatury QWERTY, techniki Swype, pisma odręcznego oraz poleceń głosowych w urządzeniach mobilnych. Porównywano poprawność wprowadzanego tekstu, jak także czas każdej z metod. Badania zostały przeprowadzone na utworzonej aplikacji mobilnego komunikatora internetowego, dedykowanej na platformę Android. Badania zostały przeprowadzone wśród dwóch grup użytkowników.

2.1. Klawiatura QWERTY

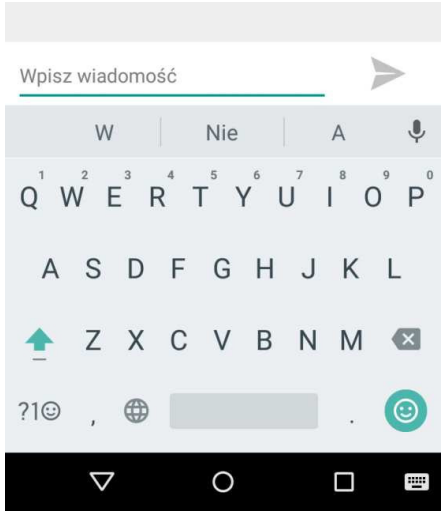
W XXI wieku pisanie na klawiaturze fizycznej, czy to klawiaturze ekranowej, nie tworzy większych trudności dla większości z użytkowników. Zazwyczaj młodzi ludzie piszą na nich nieco sprawniej. Wynika to z większego doświadczenia w pracy na klawiaturze i większej sprawności fizycznej.

Historia tworzenia układu klawiatury sięga roku 1714, kiedy to Henry Mill stworzył pierwszy działający model maszyny do pisania [2]. Kolejnym konstruktorem użytecznej maszyny pozwalającej zapisywać tekst był Christopher Latham Sholes, dokonując tego w roku 1867 [2]. Stał się on także pomysłodawcą układu QWERTY. Układ ten zbudował na podstawie własnych badań odnoszących się do częstotliwości występujących obok siebie liter w słowach języka angielskiego. Pomimo późniejszych prób ulepszenia tego układu (Dvorak, Colemak) [3] układ QWERTY okazał się najlepszym rozwiązaniem rozmieszczenia liter na klawiaturze. W niektórych krajach układ QWERTY został dostosowany do określonych wymagań językowych. W czasach dawniejszych pisanie na maszynie przypisane było do kompetencji profesjonalnych maszynistek lub sekretarzy, natomiast obecnie jest to czynność wykonywana przez miliardy użytkowników urządzeń mobilnych i komputerów na świecie.

Znajdująca się na ekranie klawiatura QWERTY jest graficznym odwzorowaniem standardowej klawiatury QWERTY renderowanej na ekranie dotykowym. Urządzenia mobilne są tworzone z własną domyślną klawiaturą QWERTY, zwykle zależną od systemu operacyjnego w którym działa [4]. Klawiatura pojawia się tylko wtedy, gdy pole tekstowe jest zaznaczone. Zaraz po pojawieniu się pierwszy raz, wszystkie litery są duże. Gdy użytkownik wybierze pierwszy klawisz, interfejs graficzny na pozostałą część zdania zmienia wygląd klawiszy na małe litery. Ponadto, na niektórych klawiaturach (domyślnie Android i iOS), gdy klawisz jest zaznaczony, użytkownik otrzymuje informację zwrotną w postaci pojawiającej się grafiki (wybrany przycisk), nałożonej na pozostałą część klawiatury. W celu wybrania cyfry lub znaku specjalnego, użytkownicy mogą przełączać tryby za pomocą klawisza przeznaczonego do przełączenia na klawiaturę numeryczną lub klawiaturę ze znakami. Ponadto systemy do wprowadzania tekstu zostały wyposażone w możliwość włączenia autokorekty. W funkcji tej można włączyć wyświetlanie propozycji proponowania kolejnego słowa, wstawianiu autokorekty poprzez wstawianie spacji i znaków przystankowych oraz automatycznie wstawianie wielkich liter na początku zdania. Od razu, gdy telefon domyśli się, co użytkownik piszący chce napisać, wyświetli na ekranie odpowiednią podpowiedź. Poprzez częste używanie niektórych słów, nawet po wstawieniu kilku pierwszych liter, telefon powinien zaproponować podpowiedź.

Oprócz domyślnej klawiatury właściciele smartphone'ów mogą pobrać inne aplikacje pełniące rolę klawiatury za darmo lub za niewielką opłatą ze sklepów, takich jak Google Play lub App Store firmy Apple. Klawiatura ekranowa (rys. 1), która jest wyświetlana na ekranie, posiada swoje zalety, ale też wady. W związku z brakiem sprzętowych (fizycznych) klawiszy urządzenie mobilne może być mniejsze, a ponadto lżejsze. W wyniku tego umożliwia także zastosowanie większego ekranu, który może usprawnić pracę z urządzeniem. Większe ekrany pozwalają na wyświetlanie większych elementów w większej rozdzielczości oraz komfortową pracę z urządzeniem mobilnym.

Ważną zaletą ekranu dotykowego jest możliwość wprowadzania tekstu bez konieczności dzielenia uwagi wzrokowej pomiędzy klawiaturą, a wpisywanym tekstem. Pozwala to na wprowadzenie większej ilości tekstu z mniejszą liczbą błędów w mniejszej jednostce czasu.



Rys. 1. Układ mobilnej klawiatury QWERTY

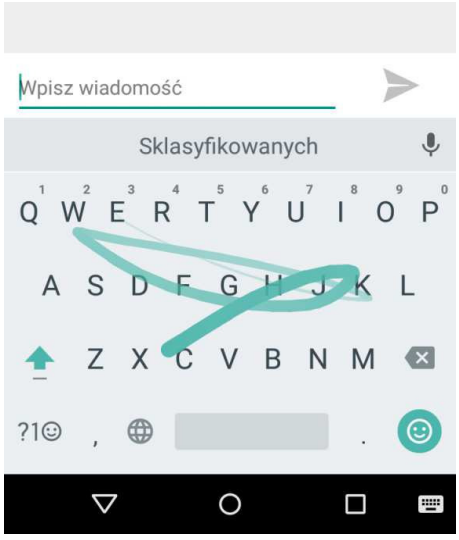
2.2. Swype

Swype jest rewolucyjną metodą, dzięki której znika problem pisania dwoma dłońmi na ekranie dotykowym. "Swype" jest również znany jako "Tracing" i został po raz pierwszy wprowadzony przez Zhai i Kristensson w 2003 roku [5]. Jest to metoda umożliwiająca szybkie pisanie nie poprzez ręczne wpisywanie pojedynczych liter, jak to występuje przy tradycyjnej klawiaturze QWERTY, lecz poprzez wprowadzania tekstu za pomocą gestów. Technika ta polega na przesuwaniu palcem po klawiaturze z litery do litery (standardowo w układzie QWERTY) w celu utworzenia słowa bez odrywania palca. Swype w miarę upływu czasu zwiększa zasoby słownikowe i uczy się stylu pisania użytkownika umożliwiając dokonanie wyboru dokładniejszej podpowiedzi. Dodatkowo uczenie się to można przyspieszyć korzystając z analizy treści wpisywanych w różnych aplikacjach służących komunikowaniu się, takich jak Facebook, Gmail czy też Twitter. Metoda Swype pojawiała się na przestrzeni lat w coraz to nowocześniejszych urządzeniach.

Swype składa się z trzech głównych elementów, które przyczyniają się do dokładności i szybkości: analizator toru wejściowego, wyszukiwarka słów z odpowiednią bazą danych i konfigurowalny interfejs [6]. Z technicznego punktu widzenia w Swype użytkownik wytycza ścieżkę. Wygląd i reagowanie linii zmienia się w różnych oprogramowaniach, a także może być dostosowywana przez użytkownika (rys. 2). Kiedy ruch po linii jest skończony i palec jest zdjęty z ekranu, oprogramowanie porównuje wzorzec wytyczony przez użytkownika z szablonami geometrycznymi zlokalizowanych w leksykonie programu [4]. Najlepsze dopasowanie jest następnie pobierane i wyświetlane jako wynik końcowy. Spacja pomiędzy wyrazami wstawia się bez ingerencji użytkownika. Podobnie jest w przypadku zmiany pierwszej litery rozpoczętego zdania – wprowadzana jest automatycznie wielka litera. W przypadku, gdy słowo zawiera "podwójną" literę, użytkownik robi gest pętli nad przyciskiem. Na przykład w słowie "passa" użytkownik musi zrobić pętlę nad przyciskiem "s" .

Do głównych zalet tej metody należy zaliczyć przede wszystkim możliwość przewidywania przez nią następnego słowa, a także opcję kopiowania oraz synchronizowania słowników osobistych. Ponadto metoda ta umożliwia wybór wielu języków pisania oraz korzystanie z gestów.

Główną wadą tej metody jest występowanie faktu, że dłoń użytkownika ułożona jest w ten sposób, że zazwyczaj podczas wprowadzania tekstu zasłania część ekranu. Może to być dość problematyczne dla użytkowników mniej zaznajomionych z układem QWERTY. Kolejna wada metody uwydatnia się w sytuacji, gdy palec zostanie zdjęty z ekranu zanim użytkownik dokończy słowo. Swype zwraca słowo najbliższe częściowo wprowadzonemu słowu. Ponadto w klawiaturze Swype brakuje bezpośredniego dostępu do wielu znaków interpunkcyjnych oraz polskich znaków. Pomimo bezpośredniego braku polskich znaków na klawiaturze, słowa są na bieżąco poprawiane. Kolejną wadą uwidocznioną w klawiaturze Swype jest brak możliwości pisania cyfr. Aby wpisać cyfry trzeba wyklikać je w standardowy sposób.



Rys. 2. Ekran swype w trakcie pisania

2.3. Wprowadzanie głosowe

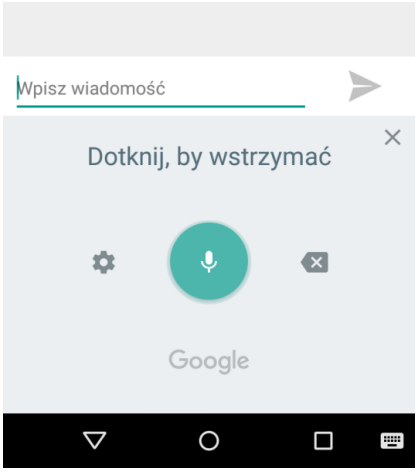
Mowa jest naturalnym sposobem komunikowania się i ma potencjał, aby być o wiele szybszym niż którykolwiek sposobem wprowadzania tekstu. Wraz z postępem technologii, mowa staje się źródłem informacji dla ludzi, a także dla urządzeń przetwarzających te sygnały. Pierwsze badania nad rozpoznawaniem mowy sięgają połowy XX wieku.

W laboratoriach Bella w 1952 roku Balashek, Davis i Biddulph przygotowali system umożliwiający rozpoznanie pojedynczych liczb [7]. Wadą tego systemu było początkowo rozpoznawanie głosu tylko jednego mówcy. Od czasu pierwszych badań do dziś technika ta ewoluowała do poziomu, w którym rozpoznawany jest głos każdego mówcy. Dzięki rozpoznawaniu głosu użytkownik nie musi być związany z klawiaturą, czy też innymi urządzeniami pozwalającymi na wprowadzanie danych. Zaletą metody wprowadzania głosowego jest brak przeciwwskazań wprowadzania w warunkach słabo oświetlonych lub też w całkowitej ciemności. Wygoda oraz naturalność mowy pozwalają na wprowadzanie tekstu w warunkach, w których użytkownik nie jest skupiony na urządzeniu. Dzięki temu powstały systemy i urządzenia, które pozwalają wydawać polecenia sterujące urządzeniami. Kolejną wadą wprowadzania głosowego jest brak możliwości użycia tej metody podczas, gdy nie ma połączenia z siecią internetową. Szybkość sieci internetowej ma więc duży wpływ na szybkość analizy nagranej mowy.

Także do niedawna nie było technicznie możliwe rozpoznawanie głosu w telefonach komórkowych. Oprogramowanie służące do przetwarzania sygnałów mowy potrzebowało więcej mocy obliczeniowej niż było dostępne. Pomimo, że technologia ta rozwijała się od połowy ubiegłego wieku, nawet najbardziej dokładne oprogramowania nie są w stanie osiągnąć 100% dokładności przetwarzania wprowadzanego głosowo tekstu [8].

Na dokładność oprogramowania do rozpoznawania mowy ma wpływ wiele istotnych zmiennych, takich jak: określone cechy użytkownika (wiek, płeć, język ojczysty, indywidualne cechy mowy, akcent), zachowanie użytkownika (wieloletnie wytężanie strun głosowych lub palenie), charakterystyka środowiska (rodzaj środowiska i poziom hałasu otoczenia) oraz parametry techniczne mikrofonu [9]. Rozpoznawanie mowy pomimo wielu czynników, które wpływają na wydajność, jest alternatywną opcją dla tych, którzy potrzebują szybkiego wprowadzenia tekstu lub dla osób z wadami wzroku. Metoda ta jest szczególnie korzystna dla osób ze zmniejszoną sprawnością manualną. Na przykład, ekspert pisząc na klawiaturze jest w stanie pisać w tempie 80 WPM (słów na minutę), natomiast tempo mowy może osiągnąć prędkość nawet 200 WPM [10]. Interfejs głosowy ma szeroki zakres funkcjonalności. Dzięki przetwarzaniu dźwięku na mowę można komponować wiadomości e-mail lub SMS oraz wpisywać polecenia dla wyszukiwarki internetowej. Zazwyczaj oprogramowanie do rozpoznawania głosu jest dostępne za pośrednictwem graficznego interfejsu użytkownika telefonu, wyświetlanego na ekranie dotykowym (rys. 3). Istnieją jednak przypadki, w których może być także

dostępny w postaci sprzętowej (np HTC myTouch "Genius" przycisk). Gdy użytkownik jest gotowy do rozpoznawania mowy, wystarczy dotknąć ekranu dotykowego lub przycisku, aby rozpocząć fazę przetwarzania. Przetwarzanie zwykle zawiera pewien rodzaj wizualnego paska postępu lub inny graficzny postęp odwzorowujący informacje zwrotną. Po przetworzeniu przez oprogramowanie danych wejściowych, wyświetlone zostaną informacje zwrotne. Niektóre oprogramowania, po zakończeniu przetwarzania, używają wibracji lub też dźwięku jako potwierdzenie ukończenia przetwarzania (na przykład Google Voice). Producenci oprogramowania w dzisiejszych czasach starają się, by ich produkt był najlepszy na rynku. Dlatego też informacje szczegółowe w jaki sposób dany mechanizm działa oraz z jakich systemów korzysta dana technologia są ukrywane. Badane wprowadzanie głosowe firmy Google obecnie rozpoznaje ponad 80 języków, z czego można wybrać też język w konkretnym dialekcie [11]. Wadą wprowadzania głosowego jest możliwość dodania znaków interpunkcyjnych tylko dla języka angielskiego, francuskiego, hiszpańskiego, niemieckiego, rosyjskiego oraz włoskiego. Mechanizm ten jeszcze nie jest dopracowany dla reszty języków w tym dla języka polskiego.



Rys. 3. Ekran wprowadzania głosowego

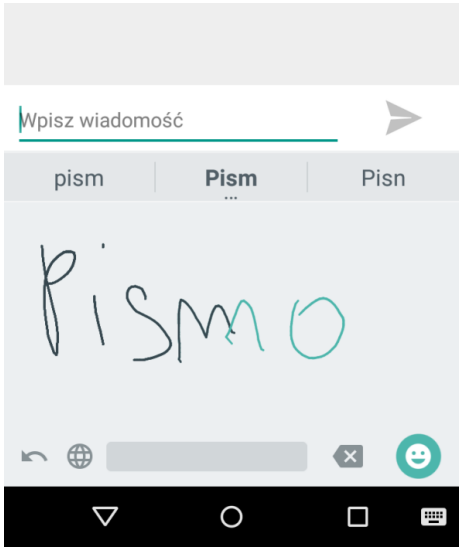
2.4. Wprowadzaniem pismem odręcznym

Kolejnym sposobem umożliwiającym wprowadzanie tekstu jest pismo odręczne. Technologia ta wykorzystywana jest w różnego rodzaju urządzeniach dotykowych. Rozpoznawanie pisma wprowadzanego przez użytkownika zostaje w określony sposób przetworzony przez system do tego służący. Wprowadzony tekst przechodzi przez kolejne procesy przetwarzania wstępnego, do których należy "przetwarzanie cyfrowe i binarne, eliminowanie szumów, zmniejszanie grubości znaków, normalizacja oraz segmentacja" [12]. Ponadto w systemie zawiera się podsystem przetwarzania wstępnego, podsystem logiki rozmytej, podsystem analiz geometrycznych, podsystem sieci neuronowych dla wyizolowanych znaków oraz podsystem sieci neuronowych dla słowników i wiedzy językoznawczej. Przetwarzanie wstępne ma w swojej właściwości zasadniczy cel polegający na zmniejszeniu wielkości wejść sieci neuronowej, a także zwiększeniu odporności na niejednoznaczność interpretacji obrazu.

Tradycyjnie systemy komputerowego pisma odręcznego wykorzystywały rysik. Metoda ta była często używana do interakcji z palmtopami (PDA) od początku 1990 roku i istniała jako opcja na niektórych urządzeniach "smartphone" od czasu Ericsson R380 w 2000 roku [13]. Metoda "Unistroke" wprowadza pojęcie pisma każdej litery alfabetu jednym pociągnięciem [14]. Zaletą metody jest to, że użytkownik może pisać bez konieczności martwienia się o kolejność pociągnięć.

Na dzisiejszych smartphone'ach, użytkownicy wyrysowują litery za pomocą palca. Zazwyczaj oprogramowanie zapewnia kolorowy ślad pozostawiany za palcem. Do odstępów między słowami, użytkownik może nacisnąć spację lub użyć gestu takiego jak przeciągnięcie od lewej do prawej [15]. Podobnie, gest od prawej do lewej w niektórych programach (np FlexT9, Pismo odręczne Google (rys. 4)) spowoduje usunięcie litery [4].

Choć pismo jest znanym sposobem, aby skomponować tekst, wpisywanie jest powolne w porównaniu do innych metod wprowadzania tekstu. Bailey [16] wykazał, że pismo osiąga około 20 WPM. Biorąc pod uwagę znajomość pisma odręcznego, użytkownicy mogą spodziewać się osiągnąć te same wyniki wprowadzania na urządzeniach przenośnych [17], to jednak większość oprogramowania do rozpoznawania pisma odręcznego jest często wolniejsza. Aczkolwiek znajomość pisma odręcznego może wypaść korzystniej dla początkujących użytkowników w stosunku do innych metod wprowadzania. W badaniu z udziałem początkujących użytkowników, pismo odręczne było początkowo szybsze (21,5 WPM) niż za pomocą klawiatury (19,6 WPM). Nawet po 250 minutach praktyki, wydajność pomiędzy tymi dwoma metodami była podobna, a także ilość popełnianych błędów [18].

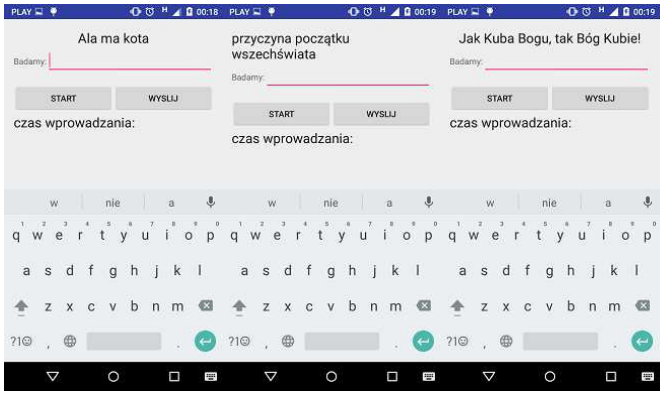


Rys. 4. Ekran wprowadzania pisma odręcznego

3. Metoda badań

3.1. Internetowy komunikator mobilny

Aplikacja dla platformy mobilnej bazuje na architekturze klient-serwer. Centralnym modułem systemu jest część programistyczna, która obejmuje implementację serwera komunikującego się z bazą danych. Kolejnym wykonanym elementem jest aplikacja mobilna dedykowana na platformę Android, pełniąca funkcję klienta wcześniej wspomnianego serwera. Głównym zadaniem aplikacji było udostępnienie możliwości komunikacji pomiędzy użytkownikami. Na potrzeby badań do aplikacji dodany został moduł zbierający wyniki (rys. 5).



Rys. 5. Ekrany badanych zdań

3.2. Grupa badawcza

Grupy wiekowe ustalone zostały na młodszych w wieku od 20 do 30 lat i starszych od 30 do 60 lat. Obie grupy liczyły po 20 osób. Wszystkie badania udało się przeprowadzić w sposób prawidłowy.

3.3. Zbieranie wyników

Celem badania było sprawdzenia umiejętności korzystania z technik wprowadzania tekstu w nowoczesnych urządzeniach mobilnych. Podczas badania mierzone były następujące parametry: szybkość i poprawność wprowadzonego tekstu. Pomiar pierwszego z nich został zapisany w milisekundach Poprawność wprowadzania oceniana była na podstawie autorskiego algorytmu porównującego wszystkie znaki analizowanego tekstu z tekstem wprowadzonym i zatwierdzonym do wysłania przez użytkownika. Metoda ta wylicza liczbę prawidłowo wprowadzonych kolejno znaków.

W badaniu zostały ustalone 3 zdania o różnej długości i stopniu trudności wprowadzania:

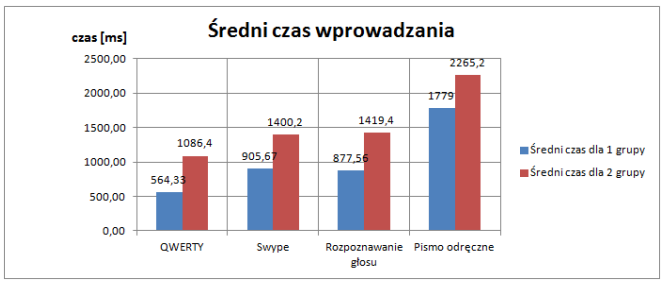
- Ala ma kota;
- Przyczyna początku wszechświata;
- Jak Kuba Bogu, tak Bóg Kubie!

Wyniki badań przesyłane były do specjalnie przygotowanej do tego celu bazy danych. Przechowuje ona takie informacje jak: wiek uczestnika, metoda za pomocą której przeprowadzane było badanie a także zadanie które było wprowadzane, czas jego wprowadzania, liczba poprawnie wprowadzonych znaków.

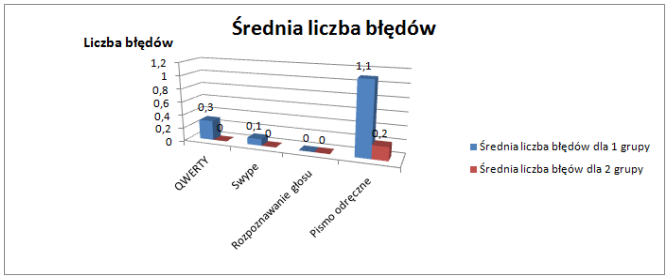
Badanie przeprowadzone zostało na terenie miasta Radom w okresie od 08.2016 r do 10.2016 r. za pomocą specjalnego modułu w komunikatorze internetowym. Badanie uwzględniało podanie wieku osoby podejmującej się zadania, metody za pomocą której zostanie wykonane, a także przesłanie 3 wiadomości. Badania zostały przeprowadzone na smartphone'ie marki Motorola model moto G 1st gen 2015 r. o przekątnej ekranu 4,5 cala. W przypadku wprowadzania głosowego szybkość połączenia internetowego miała znaczny wpływ na wynik badania, dlatego też badania przeprowadzone były w miejscach z dobrym zasięgiem sieci telefonicznej. W przypadku wprowadzania QWERTY, swype oraz pisma odręcznego szybkość połączenia internetowego ani inne parametry techniczne nie miały wpływu na wynik badania. Aby wykorzystywać wprowadzanie głosowe w języku polskim na dzień dzisiejszy konieczne jest połączenie z Internetem. Każdy badany miał możliwość wprowadzenia dowolnego zdania testowego przed podjęciem badania za pomocą każdej z metod. Badania przeprowadzane były w miejscu zamieszkania uczestników, aby czuli się oni komfortowo i mogli wykonać całe badanie unikając zbędnych bodźców nowego otoczenia. Badanie miało charakter jednorazowy i osoba podejmująca je nie miała możliwości powtórzenia go, czy poprawienia wyniku w którejkolwiek części badania. Uczestnik od początku znał treści wiadomości, o przesłanie których został poproszony, a także wyświetlane były nad ekranem wprowadzania tak, aby mógł się z nimi spokojnie zapoznać zanim rozpocznie jej wprowadzenie.

4. Analiza wyników

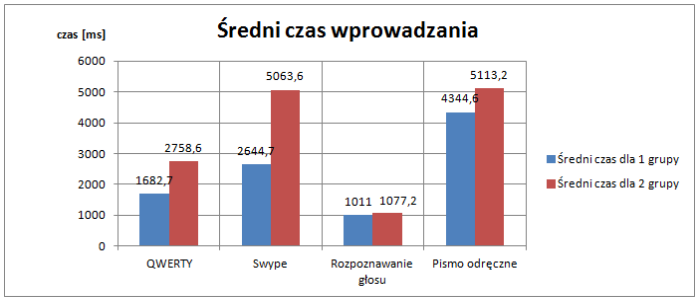
Na rys. 6, i 7 przedstawiono wyniki badań dotyczących wprowadzania zdania "Ala ma kota". Rysunki 8, i 9 dotyczą wprowadzania zdania "Przyczyna początku wszechświata" natomiast rys. 10 i rys. 11 odzwierciedlają wyniki dla zdania "Jak Kuba Bogu, tak Bóg Kubie". Pierwszy rysunek z każdej pary przedstawia porównanie średnich czasów wprowadzania pomiędzy grupami wiekowymi dla każdej metody, natomiast kolejny rysunek z każdej pary jest konfrontacją średniej liczby błędów popełnionych podczas wprowadzania pomiędzy grupami wiekowymi dla każdej metody.



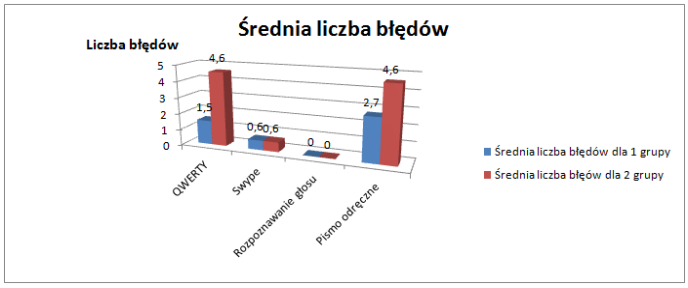
Rys. 6. Średni czas wprowadzania dla zdania "Ala ma kota"



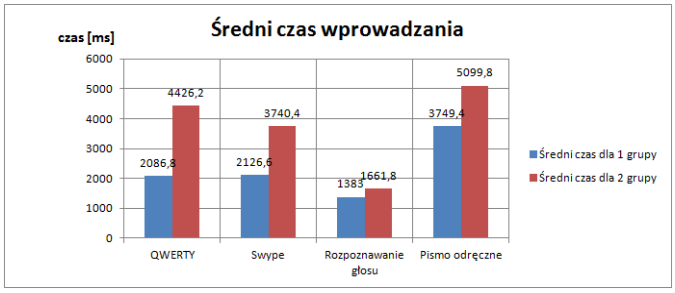
Rys. 7. Średnia liczba błędów wprowadzania dla zdania “Ala ma kota”



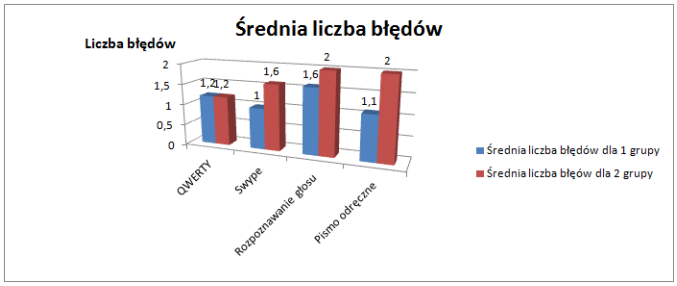
Rys. 8. Średni czas wprowadzania dla zdania “Przyczyna początku wszechświata”



Rys. 9. Średnia liczba błędów wprowadzania dla zdania “Przyczyna początku wszechświata”



Rys. 10. Średni czas wprowadzania dla zdania “Jak Kuba Bogu, tak Bóg Kubie!”



Rys. 11. Średnia liczba błędów wprowadzania dla zdania “Jak Kuba Bogu, tak Bóg Kubie!”

Na podstawie rys. 6, 8 oraz 10 zauważyć można, iż grupa pierwsza czyli młodsi użytkownicy systemu, radzą sobie z wprowadzaniem zdecydowanie szybciej w każdej z metod wprowadzania tekstu. Wynikać może to z tego, że są oni zorientowani w nowinkach technologicznych, albowiem wszyscy przedstawiciele grupy pierwszej posiadają urządzenia mobilne. W grupie drugiej wielu z uczestników badania posiada do codziennego użycia jedynie telefon komórkowy z fizyczną klawiaturą numeryczną.

Biorąc pod uwagę również wykres przedstawiony na rys. 7, a także własne subiektywne obserwacje użytkowników można śmiało stwierdzić, że błędy we wprowadzaniu pojawiały się głównie z chęci konkurencji i jak najszybszym wprowadzeniu założonego tekstu. Badani z grupy 2 byli zdecydowanie bardziej dokładni podczas wprowadzania. Widać to na wykresach przedstawionych na rys. 9 i rys. 11, podczas wpisywania dłuższych i nieco trudniejszych zdań. Niestety nawet skrupulatność nie pozwoliła im się ustrzec przed popełnieniem niewielkich błędów od 2-5. Wyjątkiem jest tu rozpoznawanie głosu, które dla obu grup podczas wprowadzania zdań, nie wymagających użycia znaków interpunkcyjnych, spowodowało zminimalizowanie występujących błędów do zera (rys. 7 oraz 9). Metoda ta okazała się także bezkonkurencyjna w przypadku pomiaru szybkości, podczas wprowadzania zdań dłuższych, oraz o większej złożoności uzyskała dla zdania drugiego odpowiednio dla pierwszej i drugiej grupy 1011 ms i 1077,2 ms, natomiast w przypadku krótkiego zdania pierwszego uzyskała ona bardzo zbliżone wyniki do najszybszych w zestawieniu 877,57 ms i 1419,4 ms (rys. 6, 8 oraz 10). Biorąc pod uwagę wszystkie wyniki badań można wysunąć wniosek, że wszystkie metody wprowadzania tekstu są dosyć łatwe oraz intuicyjne w użyciu nawet dla osób starszych i niekoniecznie korzystających z najnowocześniejszych rozwiązań, a także zapewniają podobny współczynnik błędów popełnianych przy wprowadzaniu.

5. Podsumowanie

W wyniku przeprowadzonych badań można zauważyć że grupa w której znajdują się młodsi użytkownicy systemu, radzą sobie zdecydowanie sprawniej we wszystkich metodach wprowadzania tekstu, a także liczba błędów nie przekroczyła w tej grupie 2. Wprowadzanie głosowe nie posiada w pełni dopracowanego modułu rozpoznawania znaków interpunkcyjnych dla języka polskiego i wielu innych. Skutkiem tego jest zwiększona liczba błędów podczas pisania we wszystkich grupach dla zdania trzeciego. Metoda rozpoznawania głosowego jest bezkonkurencyjna pod względem szybkości w porównaniu do innych metod dla zdania drugiego i trzeciego osiągając najniższe czasy w zestawieniu odpowiednio dla obu grup 1011 i 1077,2 oraz 1383 i 1661,8.

Literatura

[1] Urodziny pierwszego smartfona, IBM Simon – 20 lat minęło jak jeden dzień, http://android.com.pl/news/29895-urodziny-pierwszego-smartfona-ibm-simon-20-lat-minelo-jak-jeden-dzien [Dostęp: 28.11.2016]

[2] Schafer, E. D.: Mill Patents the Typewriter. Salem Pres Encyclopedia, January, 2015.

[3] Kaufman J.: The First 20 Hours: How to Learn Anything... Fast!. Wordly Wisdom Ventures LLC. 2013.

[4] Smith A.: Smartphone Text Input Method Performance, Usability, and Preference with Younger and Older Adults, 2013.

[5] Zhai S. & Kristensson P. O.: Shorthand writing on stylus keyboard. CHI '03 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. s.97-104.

[6] Kacalak W., Majewski M.: Wybrane problemy efektywnego rozpoznawania pisma odręcznego.Pomiary Automatyka Kontrola. PAK vol. 57, nr 5/2011 s. 480

[7] Davis H., Biddulph R., Balashek S.: Automatic recognition of spoken digits. Journal of the Acoustical Society of America, no. 24(6). 1952. s.637–642.

[8] Nielsen J.: Budiu R.: Funkcjonalność aplikacji mobilnych. Nowoczesne standardy UX i UI. Helion 2013. s 167.

[9] Benzeghiba M., De Mori R., Deroo O., Dupont S., Erbes T., Jouvet D., Fissore L., Laface P., Mertins A., Ris C., Rose R., Tyagi V., and Wellekens C.: Automatic speech recognition and speech variability: a review. Speech Communication, 49, 763-786

[10] Cox A. L., Cairns P. A.: Walton A. & Lee S.: Tlk or txt? Using voice input for SMS composition. Personal and Ubiquitous Computing. 2008. s.567-588.

[11] Powerful Speech Recognition,
https://cloud.google.com/speech [Dostęp :28.11.2016]

[12] Kacalak W., Majewski M.: Wybrane problemy efektywnego rozpoznawania pisma odręcznego.Pomiary Automatyka Kontrola. PAK vol. 57, nr 5/2011 s. 480

[13] Stonemetz J., Ruskin K.: Anesthesia Informatics, Springer,

[14] MacKenzie I. S. & Zhang S.: The immediate usability of Graffiti. In Proceedings of Graphics Interface. Toronto: Canadian Information Processing Society.1997. s.129-137

[15] Google Handwriting Input in 82 languages on your Android mobile device, https://research.googleblog.com/2015/04/google-handwriting-input-in-82.

[16] Bailey R. W.: Human performance engineering: Designing high quality, professional user interfaces for computer products, applications, and systems (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

[17] MacKenzie I. S. & Soukoreff R. W.: Text entry for mobile computing: Models and methods, theory and practice. Human-Computer Interaction. 2002. s.147-198.

[18] Kistensson P. O. & Denby L. C.: Text entry performance of state of the art unconstrained handwriting recognition: a longitudinal user study. In Proceedings of the ACM Symposium on Human Factors in Computing Systems (CHI). Boston. 2009. s.567-570.

JCSI 3 (2017) 18-22
Journal of Computer Sciences Institute

Wysłane: 2016-11-24
Przyjęte: 2016-11-30

Wpływ rzeczywistości wirtualnej na stan człowieka

Szymon Kołażyk*, Konrad Maciąg*, Dariusz Gutek

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

Streszczenie: Celem artykułu było ukazanie wpływu rzeczywistości rozszerzonej na stan człowieka. Przytoczono wyniki badań udowadniające wpływ wirtualnej rzeczywistości na stan emocjonalny człowieka, wpływ na przyswajanie wiedzy czy też możliwość wpływania na poziom doczuwanego bólu. Do celów badawczych stworzona została aplikacja z wykorzystaniem Unity 3D. Wyselekcjonowano grupę aplikacji oraz przygotowany został zestaw dwóch badań ankietowych. Przeprowadzone zostały badanie na grupie 10 osób.

Słowa kluczowe: wirtualna rzeczywistość; Unity 3D, Oculus Rift; stan człowieka

*Autor do korespondencji.

Adresy e-mail: szymon.kolazyk@pollub.edu.pl, konrad.maciąg@pollub.edu.pl

Analysis and evaluation of impact virtual reality on human state

Szymon Kołażyk*, Konrad Maciąg*, Dariusz Gutek

Institue of Computer Science, Lublin Univeristy of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract: The main purpose of this article is to show impact of virtual reality on human state. Results of study shows impact virtual reality on human state, for example impact on human emotions, impact on learning or impact on pain perception. Additionally, this article describes virtual reality as a tool in phobias treatment. For the research application in Unity 3D was created. Study conducted on a group of 10 people confirmed impact of virtual reality on human state.

Keywords: virtual reality; Unity 3D; Oculus Rift; human state

*Corresponding author.

E-mail addresses: szymon.kolazyk@pollub.edu.pl, konrad.maciąg@pollub.edu.pl

1. Entry

Virtual Reality is the most popular science and technology thing nowadays. VR can give people a lot of opportunities in top of fields of knowledge. Oculus Rift DK2 are one of goggles created for Virtual Reality by Oculus Company. They can be used not only for games and fun, but a lot of things to help people in normal life. For example, people can use VR goggles to, learn other languages or learn about medicine, explore new things or even learn about construction and appearance of objects [1]. What is more we can also train our behavior in emergency situation [2]. It can be also used to reduce stress [3]. VR goggles gives opportunity to view objects in all sides of view like in real life. By using VR goggle with controller called Leapmotion person who used VR goggles can also touch virtual objects and move them because Leapmotion offer possibility to view real hands and moving them in virtual world (Fig. 1).

Goggles with Virtual Reality can be used also by scientists in medicine or building for learn about human anatomy or testing some objects and reactions for physical aspects. Medicine and surgery can be easier by development of Virtual Reality goggles. Students can simulate surgery accidents and make virtual surgery to improve his skills while experienced doctors can surgery with VR remotely by using goggles. Technology of Virtual Reality and controllers using VR gives big chance to human development.

Fig. 1. Leapmotion controller in VR application [4]

In this all opportunities about Virtual Reality and fact that VR googles and controllers can trick human labyrinth there is one important question to answer – Can Virtual Reality manipulate human and may impact to human behavior?

2. Purpose of Work

The main purpose of work was to find answer for question about Virtual Reality. Analysis and evaluation of impact virtual reality on the human state.

New application called "Arachnofobia" using virtual reality goggles (Oculus Rift DK2) was created also for research. "Arachnofobia" was created in Unity 3D and Blender for research to check and possibly cure fear against spiders.

3. Range of Work

Range of work was related in main research of impact to human behavior, creating application for VR goggles using platforms called Unity 3D and Blender, exploring and choose application for Oculus Rift DK2 to analysis, creating questionnaire for research and conclusion of research results.

4. Virtual Reality History

The beginnings of Augmented Reality were year 1838 when Charles Wheatstone discovered machine called stereoscope. Stereoscope uses two pictures of the same photography in different points of view [5]. The same method was implement in Google Cardboard. Next innovations discovered in 1959 was "Sensorama" where user using this machine could watch some stereoscopic television with synchronized sound, smell, vibrations and even wind. Next innovations in this field leading to discovered Virtual Reality goggles which were called HMD (Head Mounted Display). Head Mounted Display since 1965 evolve to first goggles for Virtual Reality called Sega VR in 1990. Crucial for Virtual Reality technology was XXI age where one of companies created Oculus Rift – virtual reality goggles with two lens (Fig. 2) dedicated to virtual reality applications.

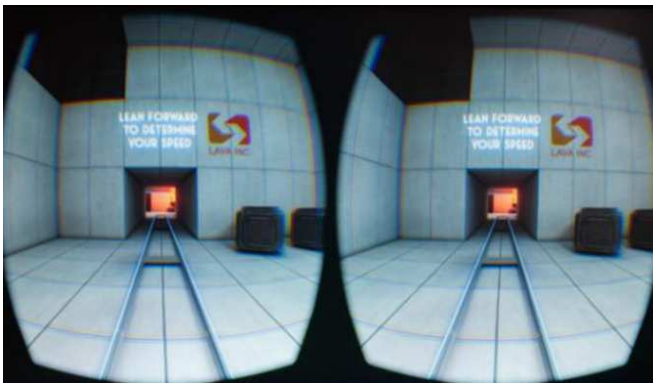


Fig. 2. View divided into two lens - Oculus Rift DK2 [6]

Nowadays Oculus Rift has got version called DK2 and consument version CK1 without a lot unnecessary wires and with special controllers for the users. More companies offer his goggles for virtual reality for example: Oculus Rift, HTC Vive. Popular were also VR goggles for smartphones which using view from android applications not from computer. Examples of mobile goggles for VR are Google Cardboard, Samsung Gear VR. For the best feelings while using VR goggles companies created additional controllers, for example virtual track called Virtuix Omni (Fig. 3) which allow users to better running in virtual reality world without moving around in reality at the same time.

Fig. 3. Presentation of VR platform called Virtuix Omni [7]

5. Impact of virtual reality to emotions

First of all, we focused on human psychology. If it is possible to manipulate of human's emotional state? This same question was asked on Vienna University [8]. Scientists create 5 virtual park scenarios. Then they examine if these scenarios can have impact of feelings like happiness, sadness, tiredness, anger or violence. They take a group of 120 students. Students were divide to 5 groups. Each of them was assigned to one of the park scenario. Next, everyone take part in virtual reality sessions which take 5 minutes. After that participants were asked to complete surveys. The research has shown that almost all of the parks were able to achieve their goals. Only scenario which should induce sadness did not cause significant increase of sadness feeling. The greatest increase of sadness was recorded on tiredness scenario.

6. Virtual reality in arachnophobia treatment

Virtual reality can be great tool for medicine. It can be used to cure a lot of phobias [9,10], for example arachnophobia. From statistics, we knew that 11% of United States society have some phobias at certain point of life. 40% of them can be assigned to group called bugs [11]. Arachnophobia is a part of this group.

Arachnophobia is a medical term for the fear of spiders. One of the most popular ways of treatment this disease is treatment by exposure. Exposure therapy involves exposure of the patient to the feared object. Of course, there is no dangerous for the patient. This therapy is intended to overcome patient anxiety.

In 2011 some study took place by Garcia-Palacios, Hoffman, Kwong, See, Tsai and Botella. They gave to group of 777 students leaflet about exposure therapy. Next, they asked them which way of treatment they prefer, the conventional or with using virtual reality. Almost 80% students preferred therapy supported by virtual reality. Researchers asked also students about taking part to 3 hours' treatment sessions. Only 8% answered "definitely no" in context of treatment in virtual reality. This same answer was chosen by 32% investigated in context of conventional therapy.

To the proper research selected 23 persons. They were divided to two groups. One of was control group. Second group took part in virtual reality sessions. The main aim of virtual reality session was to encourage patients to take to hands big virtual spider. The average number of session to achieve that aim was 4. The smallest number of session to achieve result was 3 and the biggest number was 4. Virtual reality session greatly reduced the fear of spiders. People who participated in virtual reality treatment achieved better results in conventional therapy than members of control group. 83% of members of VR group showed clinical progress. Additionally, it should be mentioned that no one from this group discontinued treatment.

7. Virtual reality and pain reduce

We can find a lot of proofs that virtual reality can have impact on human state. One of them are study with burn victims [12]. They are suffering from great pain. This pain accompanies them during daily activities, treatment and rehabilitation. Very often their pain is so great that they refuse rehabilitation, because it.

Researchers Hoffman, Patterson and Carrougher are creators of virtual reality system that should reduce feeling of pain. Feeling pain is strongly connected with attention of it. Pain signals entering the brain can be interpreted as more or less painful. It depends on what patient think about it. The more patient is focused on his pain the more he feels the pain. During a few control studies in the burn center Hoffman and his team noted that using virtual reality signified decrease patients pain. Hoffman to determine the strength of pain used a 100-point scale. In one of his group virtual reality reduced pain strength from 60 point to only 14 points. Patterson drew attention to the high potential of this observation. It is believed that the burns are among the most painful injuries what a person can endure. So we can conclude that if presented results are so good in this case we also can use virtual reality to reduce pain during painful procedures for the treatment of cancer and in other situations where for some reasons we cannot use a general anesthesia.

Hoffman and Paterson created virtual reality application called "SnowWorld". In this application patient take control on fighter and fly above snow-covered canyon. "SnowWorld" was used also in other studies. Two version of "SnowWorld" were prepared. One technically more advanced, and second simpler. Conclusion on this studies is that more realistic simulation is, the pain is more reduced.

8. Impact virtual reality to society

Most of people assume family. It is one of the most important elements all cultures in the world. What will happen when this experience will be moved to virtual reality. Nowadays, we do not have any long-term study in this matter [13]. Despite this we should considers such interesting case. Popularity of computer game "The sims" shows as that people like try alternative, virtual simulation of live and relationships.

In future, such type of simulation probably will be more advanced. That can be very dangerous for society. We should consider situation when in addition to standard equipment we add private virtual reality set. People after work will launch their VR and will move from their small apartments to big houses with Olympic swimming pools. In virtual world people can have partners and children. Simulation can be connected to Internet and other players can have impact to shape of simulation.

We should consider consequences of simulation like this. First of all, it can have negative impact to number of marry and new families. Starting family is not only a pleasure, but above all a responsibility for its members. On the other hand, virtual reality can provide simulation of no-stress life. Our partner can be programmed in a way that perfect meets our needs. Our virtual children will be smart, polite, nice, distinguished by intelligence or sport achievements. Virtual life like this could tempt a lot of people. In long-term such situation can be a great danger for social relations.

9. Research

Group of ten volunteers were using VR googles from Oculus – Oculus Rift DK2. Every one volunteer will able to watch all three applications prepared before for research.

Fig. 4. View of virtual carousel called Cyber space [14]

Virtual reality applications for Oculus Rift DK 2 used in research are “Oculus Rift Demo scene”, “Arachnofobia”, “Ocean Rift” and “Cyber Space” (Fig. 4). Every one of them is different application using different potential of Virtual Reality and googles for VR. Application “Oculus Rift Demo scene” is default application offered together with Oculus software. In this application user can see desk with plant, lamp and other office tools in different points of view. Next application “Ocean Rift” is application where user can swim in the ocean and look up to the underwater nature – fish, underwater plants. In next application called “Arachnofobia” created for research and to learn about development VR application. In this application volunteer was inside classrooms of building Lublin University of Technology. Inside rooms there is a lot of spiders so user can interact with them. It is also good for people who are scared of spiders to get used to this animal. Another and the best application for research was “Cyber Space” – virtual carousel with semi graphic but good dynamic application. Research show human behavior on virtual reality. During study, there was few physical aspects that should be important to find answer for question about impact to human behavior like pulse before and after (Fig. 5), discomfort (Fig. 6), dizziness (Fig. 7), sickness, headache and stomach discomfort (Fig. 10).

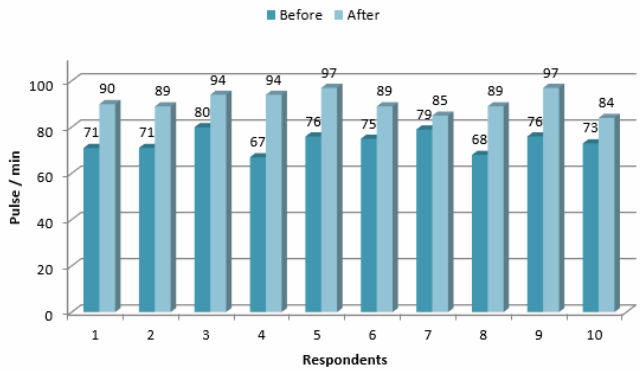


Fig. 5. Pulse before and after using Oculus Rift

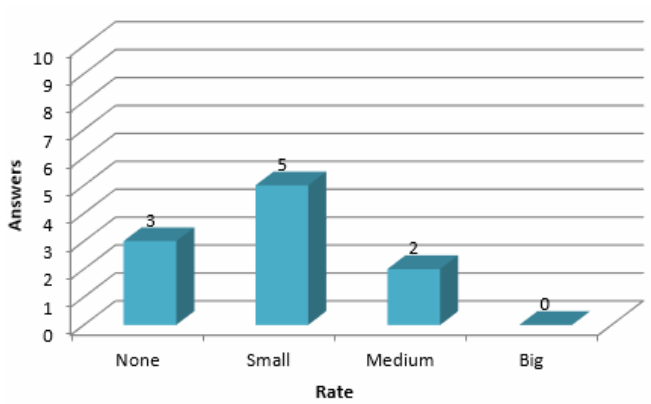


Fig. 6. Level of discomfort after Oculus Rift - “Cyber space”

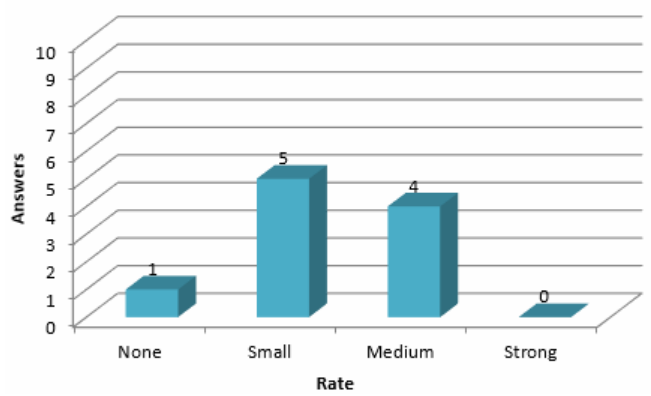


Fig. 7. Level of dizziness after using Oculus Rift

Another important aspect was opportunities inside Virtual Reality world like level of exploring world (Fig. 9), level of exploring things, method of getting around in VR, level of disorientation and similarity between real and virtual world (Fig. 8).

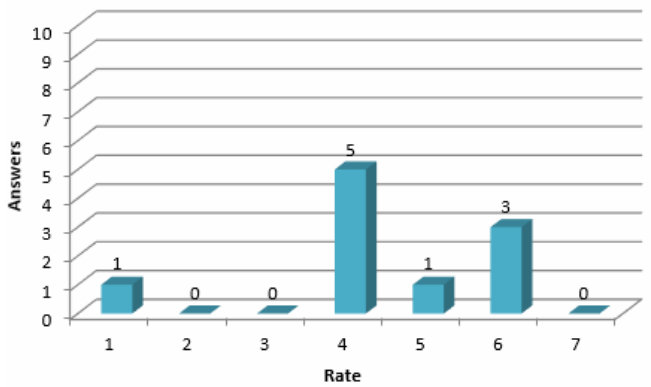


Fig. 8. Similarity level between VR world and reality

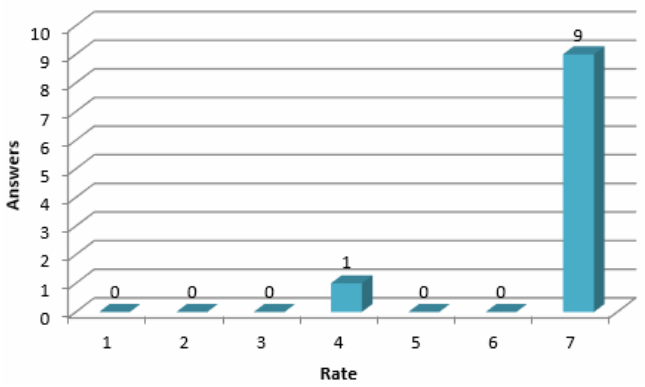


Fig. 9. Possibility level of exploring world in VR

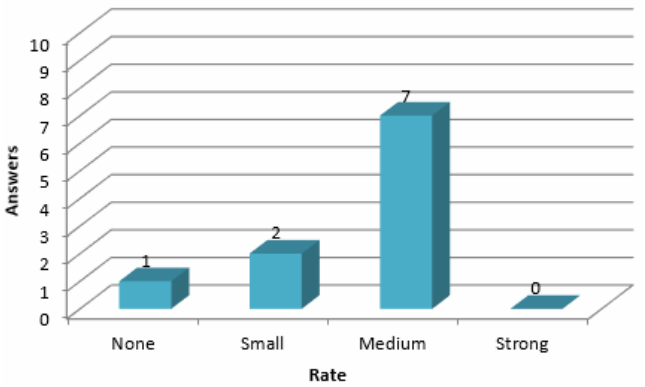


Fig. 10. Level of stomach discomfort after VR session

All those things getting answer to question about Virtual Reality and possibilities to using VR for manipulating people and impact to human state or human psychology.

10. Conclusion

Virtual Reality can be very important technology XXI century because it could be helpful in every kind of fields – medicine, learning, prototyping, fun, history. Research show that Virtual Reality may impact to human state. It is very helpful information because scientists can include inside people psyche and cure diseases and fears. It also might be dangerous if it will be used in negative things.

Bibliography:

[1] Fengfeng K., Sungwoong L., Xinhao X.; Teaching training in a mixed-reality integrated learning environment, Computers in Human Behavior 62, 2016

[2] Gamberini L., Chittaro L., Spagnolli A., Carlesso C.; Psychological response to an emergency in virtual reality: Effects of victim ethnicity and emergency type on helping behavior and navigation, Computers in Human Behavior 48, 2016

[3] Crescentini C., Chittaro L., Capurso V., Sioni R., Fabbro F.; Psychological and physiological responses to stressful situations in immersive virtual reality: Differences between users who practice mindfulness meditation and controls, Computers in Human Behavior 59, 2016

[4] http://indiecade.tumblr.com/post/107696116381/leap-motion-announces-the-winners-of-the-3d-jam

[5] http://www.vrs.org.uk/virtual-reality/history.html

[6] http://blog.pclab.pl/yosi89/Oculus.Rift.DK2.Recenzja.i.wra%C5%BCenia.z.u%C5%BCywania.okular%C3%B3w.wirtualnej.rzeczywisto%C5%9Bci,487

[7] http://virtualrealityhq.nl/

[8] Felnhofer A., Kothgassner O. D., Schmidt M., Heinzle A. K., Beutl L., Hlavacs H., Kryspin-Exner I.; Is virtual reality emotionally arousing? Investigating five emotion inducing virtual park scenarios, Int. J. Human-Computer Studies, 2015

[9] Parson T. D., Rizzo A. A.; Affective outcomes of virtual reality exposure therapy for anxiety and specific phobias, Journal of Behavior Therapy and Experimental Psychiatry 39, 2008

[10] Wrzesien M., Botella C., Bretón-López J., Río-González E., Burkhardt J. M., Alcañiz M., Pérez-Ara M. A; Treating small animal phobias using a projective augmented reality system: A single-case study, Computers in Human Behavior 49, 2015

[11] Garcia-Palacios A., Hoffman H., Carlin A., Furness T. A., Botella C.; Virtual reality in the treatment of spider phobia: a controlled study, Behaviour Research and Therapy 40, 2012

[12] Hoffman G. H., Chambers G. T., Meyer III W.J., Arceneaux L. L., Russell W. J., Seibel E. J., Richards T. L., Sharar S. R., Patterson D. R.;Virtual Reality as an Adjunctive Non-pharmacologic Analgesic for Acute Burn Pain During Medical Procedures, The Society of Behavioral Medicine, 2011

[13] Koltko-Rivera M. E.; The Potential Societal Impact of Virtual Reality

[14] https://ejmudrak.wordpress.com/tag/vr/

JCSI 3 (2017) 23-27
Journal of Computer Sciences Institute

Wysłane: 2016-11-23
Przyjęte: 2016-12-04

Symulacja zachowań typu BOIDs w środowisku Unity

Taras Lypovyi, Jerzy Montusiewicz*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W pracy opisano i scharakteryzowano zachowania typu BOID (bird-oid object) polegające na wspólnym przemieszczaniu się gromady obiektów o tych samych właściwościach. Zaprezentowano model Reynolds'a uwzględniający 3 reguły: separacja, wyrównanie i spójność oraz procedury sterowania gromadą obiektów zaproponowaną przez Parkera uwzględniająca takie wielkości, jak: wiatr, cel, prędkości, kolejność oraz występujące siły. Metoda badawcza polegała na przeprowadzeniu eksperymentów symulacyjnych przy różnych konfiguracjach współczynników sił sterowania modelem. Dla każdej symulacji był wymierzony czas ruchu od punktu początkowego do punktu końcowego. W pracy przeprowadzono sto symulacji dla każdej poszczególnej grupy współczynników, a następnie wykorzystując opisane metody statystyki wyznaczono uogólnione wartości czasu, które umożliwiały porównywanie uzyskanych wyników. Przeprowadzone symulacje numeryczne zrealizowano w środowisku Unity. Obliczanie czasu potrzebnego na przebycie identycznej drogi przeprowadzano przy zmianie wartości siły separacji, spójności, wyrównywania oraz unikania. Z uzyskanych wartości wynika, że największy wpływ na wzrost czasu przemieszczania obiektów typu BOID-s ma zwiększanie wartości współczynnika sił separacji oraz wyrównywania. Środowisko Unity dobrze nadaje się do prowadzenia takich symulacji, ponieważ umożliwia otrzymanie zarówno wartości liczbowych jak i wizualizacji procesu w postaci obrazu 3D. Oprócz tego Unity zezwala na tworzenie własnych skryptów do zarządzania symulacją we własnym IDE, a także przedstawia dokumentację, co upraszcza ich pisanie.
Słowa kluczowe: BOID; symulacja zachowań BOID-s; środowisko Unity

*Autor do korespondencji.

Adres e-mail: j.montusiewicz@pollub.pl

Simulation of BOID type behaviours in Unity environment

Taras Lypovyi, Jerzy Montusiewicz*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The study describes and characterises BOID (bird-oid object) type behaviours, consisting of joint movement of a cluster of objects with the same properties. Authors presented Reynolds' model, which takes into account 3 rules: separation, alignment and consistency, as well as the control procedures of a cluster of objects suggested by Parker, considering such variables as wind, aim, speed, order and the occurring forces. The test method was to conduct simulation experiments with different configurations of coefficients of the forces controlling the model. For each simulation the time of moving from the start point to the end point was measured. A hundred simulations were carried out for each individual group of coefficients, and then, using the described statistics methods, generalised time values were determined. This allowed a comparison of the results and made a conclusions. The numerical simulations carried out were implemented in Unity environment. Calculating the time required for travelling the same route was done by changing the value of the separation force, cohesion, alignment and avoidance. From the values obtained, it can be seen that the biggest influence on the increase of the time of moving BOID objects, is increasing value of the coefficient of separation and levelling forces. Unity environment is well suited to conduct such simulations, since it allows to obtain both numerical values and process visualization as a 3D image. In addition, Unity allows to create individual scripts to manage simulation in individual IDEs, and consists reliable documentation, which simplifies their writing.

Keywords: BOID; simulation of BOID behaviour; Unity environment

*Corresponding author.

E-mail address: j.montusiewicz@pollub.pl

1. Wstęp

Skrót "BOID" to skrócona wersja terminu "bird-oid object" („bird-like object") czyli obiekty *ptakopodobne*. W świecie przyrody w ten sposób poruszają się ławice ryb lub stada małych ptaków (nie dotyczy to jednak ptaków wędrownych przemierzających duże dystanse tworzące w powietrzu tzw. klucze).

Pierwsza praca, w której były opisane zachowania takich obiektów ujrzała światło dzienne w 1986 r. i był to program sztucznego życia Craiga Reynoldsa [1]. Autor symulował w niej zachowania grupy ptaków na potrzeby fotorealistycznie wyglądających obiektów realizowanych przy zastosowaniu grafiki komputerowej oraz przy tworzeniu naturalnie przemieszczających się grup ptaków, ryb i innych zwierząt. Obecnie algorytm zaproponowany przez Reynoldsa używany jest nie tylko w grafice czy grach komputerowych, ale także może być adaptowany do potrzeb kontroli i stabilizacji ruchu zespołów prostych bezzałogowych pojazdów naziemnych, powietrznych czy podwodnych oraz wizualizacji danych i rozwiązywanych zadań optymalizacji.

Metodę ciągłej optymalizacji funkcji nieliniowych nazwaną algorytmem roju cząstek podano w pracy [2] w 1995 r. Koncepcja algorytmu polegała na symulacji systemu wielokrotnego agenta, w którym cząstki poruszają się do optymalnego rozwiązania wymieniając przy tym informacje ze swoimi sąsiadami. W 1998 r. w pracy [3] zaproponowano zmianę polegającą na wprowadzeniu równowagi między dokładnością badania przestrzeni poszukiwania i szybkością zbieżności algorytmu. Z kolei w 2002 r. w pracy [4] wprowadzono modyfikacje algorytmu roju cząstek (algorytm ten uważany jest obecnie za kanoniczny). Zastosowany sposób

wyliczania wektorów prędkości cząstek wzbogacono o nowy czynnik, zapewniający zbieżność algorytmu bez konieczności bezpośredniego kontrolowania prędkości cząstek. W pozycji [5] autorzy wprowadzili do modelu wszystkie cząstki nazwane jako „w pełni poinformowane", czyli takie, które otrzymywały informację od wszystkich cząstek sąsiednich. W pracy [6] zaproponowano proste rozszerzenie idei BOID-ów poprzez możliwość zmiany lidera. Zmiana przywództwa jest oceniana w widocznym obszarze każdego BOID-a, czyli tylko lokalne cząstki brane są pod uwagę. Użytkownik posiada parametr, który jest atrybutem przywództwa. Szansę zostania liderem mają BOID-y znajduje się na granicy stada. W pracy [7] pokazano metodę do rozwinięcia parametrów modelów opartych na agentach, która prowadzi do powstania tzw. krytycznego zachowania. Właściwości takich modelów były badane przy użyciu algorytmów ewolucyjnych stosując pięć parametrów dla każdego BOID-a: unikanie (separacja), spójność, jednolitość, pęd i sąsiedztwo. Wykorzystanie algorytmów genetycznych do optymalizacji wektora ruchu poprzez znalezienie optymalnych współczynników opisano w pozycji [8]. Autorzy w pracy [9] opisywali jednorodny rój z uwzględnieniem komunikacji wszyscy-do-wszystkich, stosując mieszaną konfiguracja rzeczywistości, w której mała liczba robotów fizycznych współdziała z większym liczebnie rojem wirtualnym. Takie podejście pozwalało na obserwowanie powstawania wzorca ruchu dla dużych rojów w ograniczonej przestrzeni laboratoryjnej.

Celem pracy było wykonanie symulacji zachowań roju BOID-ów przy różnych wartościach sił sterowania opracowanym modelem. Poszukiwano tych sił i wartości, które najbardziej wpływają na czas przemieszczania się roju oraz jego wygląd.

2. Opis modelu BOIDs

Grupa BOID-ów to grupa obiektów wykonujących płynny ruch. Zgodnie z badaniami ruch grupy jest wynikiem działania poszczególnych obiektów, działających wyłącznie na podstawie własnego lokalnego spostrzegania świata. Stado jest więc formacją powstałą na skutek oddziaływania pomiędzy zachowaniami pojedynczych ptaków. Opracowanie symulowania ruchu stada może być zrealizowanie przez stworzenie modeli kilku ptaków i wprowadzenie dodatkowych interakcji między nimi [10]. Przykładowa struktura programu symulacji zachowań grupy obiektów może być opisana pseudokodem zaprezentowanym w Listingu 1.

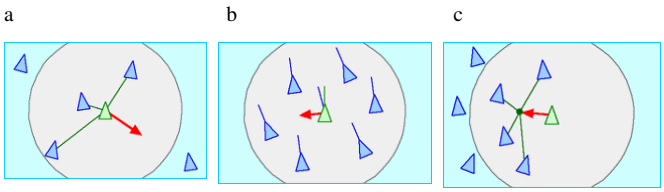
Listing 1. Struktura programu symulacji [11]

```
initialise_positions()
LOOP
    draw_boids()
    move_all_boids_to_new_positions()
END LOOP
```

Procedura `initialise_positions()` umieszcza każdy BOID na początkowej pozycji, zwykłe jest to losowa lokalizacja na ekranie. Na początku symulacji obiekty ruszają prosto przed siebie [11]. Funkcja `draw_boids()` odpowiada za rysowanie wszystkich BOIDów na swoich pozycjach [11]. Procedura `move_all_boids_to_new_positions()` zawiera właściwy algorytm przemieszczania się BOID-sów.

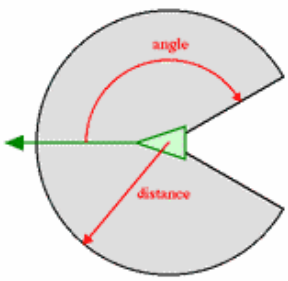
Podstawowy model stada obiektów tworzących rój składa się z trzech prostych zachowań rządzących ich ruchem. Opisują one w jaki sposób poszczególne manewry BOID-a są realizowane na podstawie położenia i prędkości innych obiektów [1]:

- Separacja (rys. 1a) – odpowiada za unikanie tłoku lokalnych członków stada, dzięki czemu nie występuje kolizja z pobliskimi obiektami;
- Wyrównanie (rys. 1b) – żąda aby przemieszczanie następowało w odniesieniu do średniej pozycji lokalnych członków stada;
- Spójność (rys. 1c) – żąda aby obiekty poruszały się w kierunku średniej pozycji lokalnych członków stada, co prowadzi do próby utrzymania się blisko sąsiednich członków stada.



Rys. 1. Podstawowe siły modely grupy BOID-ów [1]

Każdy BOID ma bezpośredni dostęp do całego opisu geometrycznej sceny, ale udział w stadzie powoduje, że obiekt reaguje tylko na członków z pewnego ograniczonego obszaru wokół siebie (rys. 1.). Sąsiedztwo opisane jest przez odległość (mierzona od środka pojedynczego BOID-a) oraz kąt kierunku ruchu BOID-a. Członkowie stada spoza tego sąsiedztwa są ignorowani [1].



Rys. 2. Obszar sąsiadów [1]

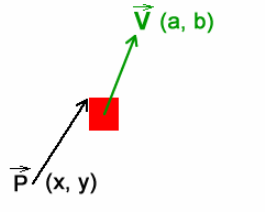
W symulacji istnieje również możliwość wprowadzenia procedury ograniczającej prędkości obiektów. Bez takiego ograniczenia rozpatrywane obiekty musiałyby latać bardzo szybko, co byłoby w sprzeczności z rzeczywistymi ptakami, które przemieszczają się z prędkościami zależnymi od danego gatunku, Listing 2.

Listing 2. Kolejność wywoływania procedur [12]

```
b.velocity = b.velocity + v1 + v2 + v3 + ...
```

```
limit_velocity(b)
b.position = b.position + b.velocity
```

Implementację wszystkich sił, które wpływają na ruch obiektu, można osiągnąć wykorzystując zapis wektorowy. Na rysunku 3 pokazano obiekt w punkcie $\vec{P}(x,y)$ o prędkości $\vec{V}(a,b)$.



Rys. 3. Obiekt, jego pozycja i prędkość w postaci wektorowej [12]

Następna pozycja obiektu P_1 jest wyliczana jako *następna_pozycja = pozycja + prędkość*, wzór (1).

$$P_1(x_1,y_1) = \begin{Bmatrix} x_1 = x + a, \\ y_1 = y + b \end{Bmatrix} \quad (1)$$

Zwrot wektora wskazuje kierunek ruchu BOID-a, długość – wartość o ile ma się przesunąć obiekt względem poprzedniej pozycji. Im większą wartość osiąga drugi parametr tym szybszy jest ruch obiektu. Do tego, aby ruch modelowanego BOID-a był bardziej podobny do ruchu realnych obiektów wykorzystywane są siły sterowania, czyli wektory, które są dodawane do wektora prędkości. W zależności od tych sił, obiekt BOID-a będzie poruszać się w okreslonym kierunku. Uzyskane zachowanie BOID-a wynika więc z żądanej prędkości oraz występujących sił sterowania.

Żądana prędkość jest to siła, która wskazuje obiektowi cel osiągany po najkrótszej możliwej trajektorii. Tym samym czas siły sterowania, to różnica między żądaną prędkością i aktualną prędkością, która także wskazuje na cel. Siły są wyliczane przez procedury jak w Listingu 3.

Listing 3. Zastosowane procedury [11]

```
desired_velocity = normalize(target - position) *
max_velocity
steering = desired_velocity - velocity
```

Po wyliczeniu siły sterowania niezbędne jest dodanie jej do składowej prędkości obiektu. Dodawanie siły sterowania do prędkości w każdej ramce symulacji spowoduje stopniową zmianę kierunku ruchu w stronę celu, co można w łatwy sposób wykreślić na rysunku przedstawiającym tą sytuację, Listing 4.

Listing 4. Zastosowane procedury [11]

```
position = position + velocity
steering = truncate (steering, max_force)
steering = steering / mass
velocity = truncate (velocity + steering , max_speed)
```

3. Metodyka badawcza

3.1. Opis badań i ich cel

Dla badania zachowań typu BOIDs wybrano metodę eksperymentalną, która polegała na przeprowadzeniu symulacji i porównaniu czasów pokonywania zadanej drogi przez grupę obiektów. Badano wpływ poszczególnych sił na zachowanie grupy BOID-ów, oraz wpływ jednoczesnej zmiany wartości wszystkich sił na zachowanie całej grupy obiektów. W symulacji uwzględniono 3 siły podstawowego modelu Reynolds'a: *separacja*, *spójność*, *wyrównanie* oraz dodatkowo siłę *unikanie* przeszkód. Dla każdego z analizowanych przypadków wykonano po sto symulacji zapisując czas ruchu przejścia od pozycji start do pozycji cel. Na podstawie wartości uzyskanych czasów, a także ogólnego wyglądu grupy BOID-ów, tzn. wzajemnego położenia poszczególnych członków grupy, przeprowadzono analizę dotyczącą wpływu każdej z badanej siły na szybkość ruchu grupy obiektów podążającej do celu.

Celem symulacji było więc zbadanie jak zmiany wartości poszczególnych sił wpływają na ruch grupy obiektów. A także czy dobór odpowiedniej konfiguracji i wartości badanych sił może zdecydowanie skrócić czas przemieszczania się grupy obiektów oraz czy ma to wpływ na ogólny wygląd grupy.

Badania były dwuetapowe. W pierwszym etapie przeprowadzono wstępne symulacje zakładając, że wszystkie siły przyjmują takie same wartości współczynników wpływu (wartość 1). W drugim etapie wykonano badania, w których dokonywano zmian wartości poszczególnych sił – wzrost dwukrotny, przy jednoczesnym zachowaniu wartości pozostałych. Wykonano również symulacje, w których wyłączano kolejne siły oddziaływujące na grupę BOID-ów (wartość takiej siły wynosiła 0).

3.2. Środowisko Unity

Do badań został opracowany model w środowisku Unity służącym do tworzenia trójwymiarowych i dwuwymiarowych gier komputerowych, wizualizacji oraz animacji tzw. bytów interaktywnych. Zaletą środowiska Unity jest to, że jest ono bezpłatne oraz może działać w różnych systemach operacyjnych, np. Windows, OS X, i zezwala na tworzenie zarówno aplikacji internetowych, komputerowych, jak i komórkowych. Oprócz tego możliwe jest pisanie skryptów do zarządzania obiektami. Do tego celu można wykorzystać jeden z trzech języków programowania wspieranych przez środowisko Unity: C#, Unity Script i Boo. Silnik Unity posiada również możliwość importu bibliotek dynamicznych (DLL). Ponadto w tym środowisku można stosować kompatybilne hełmy rzeczywistości wirtualnej, np. Oculus Rift i Gear VR.

3.3. Model badawczy

Utworzony wirtualny model badawczy składał się z miejsca generowania BOID-ów, to jest miejsca gdzie zaczynały one swój ruch, przeszkód stojących na ich drodze

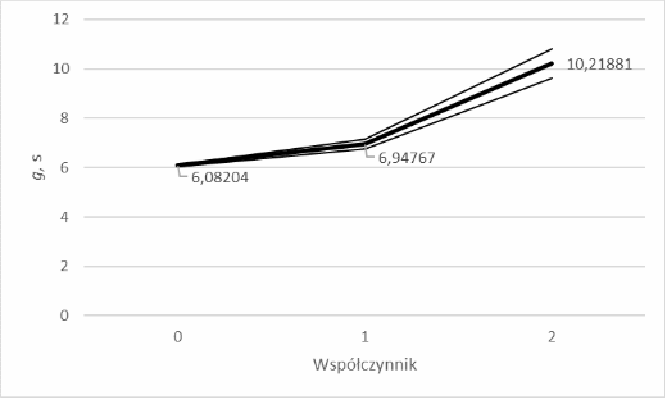
oraz obiektu będącego celem – czyli miejsca, gdzie BOID-y kończą ruch. Do zbudowanego modelu należy również zaliczyć same BOID-y.

Na podstawie opisów algorytmów przedstawionych w pozycjach [1] i [12] oraz własnych modyfikacji zostały napisane skrypty w języku C#. Zmiany polegały na tym, że do procedur dotyczących sił separacji i unikania dodano nowe współczynniki c1 i c2. Współczynnik c1 przyjmował wartości odwrotnie proporcjonalne do odległości między BOID-em a jego sąsiadem. Został wykorzystywany do tego, aby największy wpływ na siłę separacji mieli najbliżsi sąsiedzi. Do wyznaczenia przeszkody najbardziej niebezpiecznej z punktu widzenia ruchu całej grupy wykorzystano współczynnik c2. Współczynnik ten przyjmował wartości odwrotnie proporcjonalne do odległości między centrem przeszkody, a końcem wektora ahead. Wektor ten jest tzw. linią „wzroku" BOID-a. Powstaje jako kopia wektora prędkości, ale ma inną wartość.

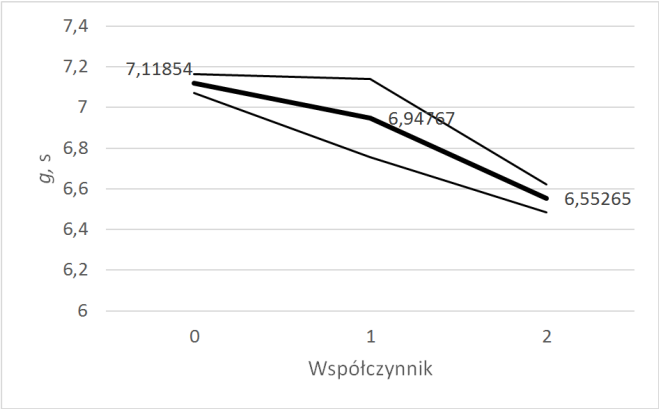
W zastosowanym skrypcie występują również tzw. publiczne zmienne: maksymalna prędkość, promień sfery sąsiadów, odległość separacji, współczynnik separacji, współczynnik spójności, współczynnik wyrównania, współczynnik unikania, odległość widoczności. W badanym modelu te wielkości były potraktowane jako parametry, tzn., że ich wartości nie podlegały zmianom w procesie symulacji. Wielkości te posłużyły do skonfigurowania badanego modelu (rysunek 3), czyli określały ogólne warunki względem których wyliczane były badane siły wpływu na zachowanie BOID-a.

4. Wyniki badań

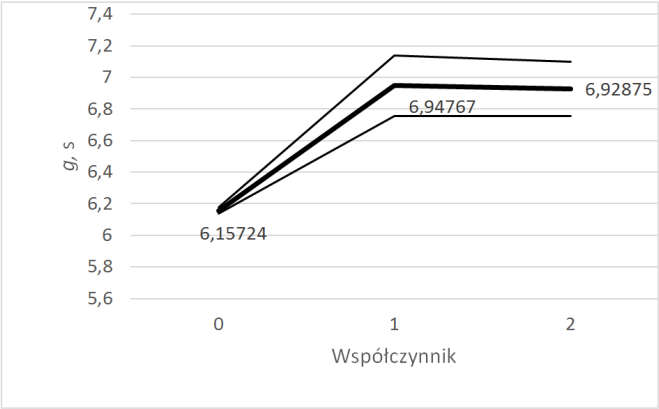
Wyniki wykonanych symulacji w postaci wykresów średnich wartości czasu potrzebnych na pokonanie zadanego dystansu oraz ich średnich odchyleń przy zmianie wartości poszczególnych współczynników sił oddziaływujących na grupę pokazano na rysunkach 4-8.



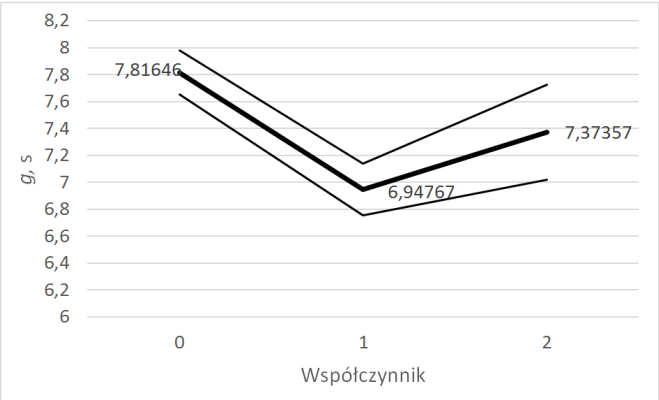
Rys. 4. Średnia geometryczna i średnie odchylenie przy zmianach współczynnika siły separacji



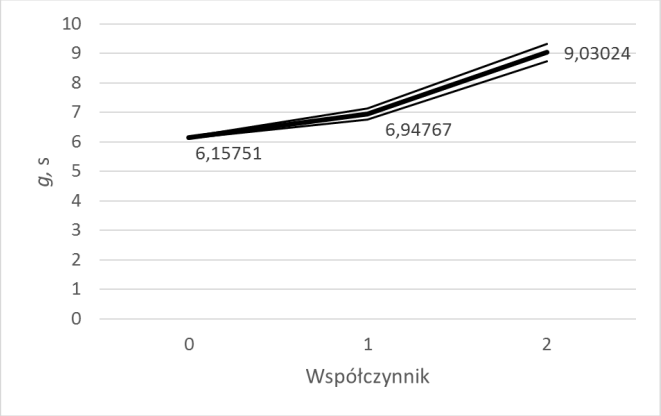
Rys. 5. Średnia geometryczna i średnie odchylenie przy zmianach współczynnika siły spójności



Rys. 6. Średnia geometryczna i odchylenie średnie przy zmianach współczynnika siły wyrównywania

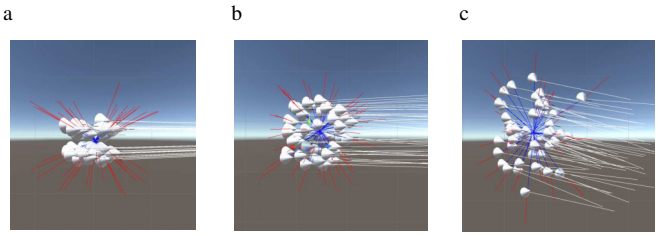


Rys. 7. Średnia geometryczna i odchylenie średnie przy zmianach współczynnika siły unikania



Rys. 8. Średnia geometryczna i odchylenie średnie przy zmianach współczynnika przy każdej sile

Na rysunku 9 pokazano wizualizację położenia grupy BOID-ów zrealizowaną w środowisku Unity.



Rys. 9. Widok grupy BOID-ów w środowisku Unity przy wartościach współczynnika każdej z sił a) – 0, b) – 1, c) – 2

5. Dyskusja i wnioski

Przeprowadzone badania i uzyskane wyniki pozwalają wyciągnąć następujące wnioski.

1) Zwiększenie współczynnika siły separacji z wartości 1 na 2 powoduje ponad 50% wzrost średniego czasu pokonania zadanej drogi przez grupę BOID-ów. Wartość 0 powoduje, że obiekty poruszają się w szyku „gęsiego".
2) Zwiększenie współczynnika siły spójności z wartości 1 na 2 powoduje, że czas pokonania drogi przez grupę obiektów zmniejsza się o około 6%.
3) Zwiększenie współczynnika siły wyrównania z wartości 1 na 2 w zasadzie nie powoduje zmiany czasu średniego na pokonania drogi przez grupę BOID-ów. Natomiast wyłączenie działania tej siły (wartość = 0) zmniejsza średni czas o około 13%.
4) Zmiana wartości współczynnika siły unikania powoduje, że przebieg wykresu jest niemonotoniczny. Najdłuższy średni czas pokonania drogi przez grupę obiektów uzyskujemy przy braku tej siły (wartość 0), przy wartości 1 średni czas zmniejsza się o około 11%, zaś przy wartości 2 zwiększa się o około 6%.
5) Gdy analizujemy jednocześnie identyczny wzrost wszystkich sił (od wartości 1 do 2) to okazuje się, że wzrost średniej wartości czasu na przebycie przez grupę BOID-ów zaplanowanej drogi wzrasta z około 6,95 s do 9,03 s, co stanowi około 30%. Przy pominięciu tych wszystkich sił (wartości = 0) średni czas jest krótszy o około 11%, ale w takiej sytuacji poszczególne obiekty BOID wpadają na siebie, co jest niedopuszczalne.
6) Środowisko Unity dobrze nadaje się do przeprowadzenia symulacji zachowań typu BOIDs oferując przy tym możliwość wykonania wizualizacji, co znacząco ułatwia obserwację poszczególnych etapów przemieszczania się grupy obiektów i zrozumienia działania poszczególnych sił na ich ruch.

Literatura

[1] C. W. Reynolds: Boids. Background and Update: www.red3d.com/cwr/boids, dostęp: 2016.10.10.

[2] Clerc M., Kennedy J.: The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space [W]: IEEE Transactions on Evolutionary Computation. Volume 6, 1, Feb 2002

[3] R. Mendes, J. Kennedy, J. Neves: The Fully Informed Particle Swarm: Simpler, Maybe Better [W]: IEEE Transactions on Evolutionary Computation, Volume 8, 3 June 2004

[4] C. Hartman, B. Benes: Autonomous boids: Computer Animation and Virtual Worlds, 2006, Volume 17

[5] M. Wagner, W. Cai, M. H. Lees: Emergence by Strategy: Flocking Boids and Their Fitness in Relation to Model Complexity: Simulation Conference (WSC), 8-11 Dec 2013

[6] S. Alaliyat, H. Yndestad, F. Sanfilippo: Optimisation of Boids Swarm Model Based on Genetic Algorithm and Particle Swarm Optimisation Algorithm (Comparative Study): files.matlabsite.com/docs/papers/sp/pso-paper-126.pdf, dostęp: 2016.10.10.

[7] K. Szwaykowska and I. B. Schwartz, L. Mier-y-Teran Romero, C. R. Heckman, D. Mox and M. Ani Hsieh: Collective Motion Patterns of Swarms with Delay Coupling: Theory And Experiment: arxiv.org/pdf/1601.08134.pdf, dostęp: 2016.10.10.

[8] C. W. Reynolds: Steering Behaviors for Autonomous Characters: www.red3d.com/ cwr/steer/gdc99/, dostęp: 2016.10.10.

[9] Reynolds C. W.: Flocks, Herds, and Schools:A Distributed Behavioral Model. Computer Graphics, 1987, Nr 21(4)

[10] C. Parker: Boids Pseudocode: www.kfish.org/boids/ pseudocode.Html, dostęp: 2016.10.10.

JCSI 3 (2017) 28-32
Journal of Computer Sciences Institute

Wysłane: 2016-12-10
Przyjęte: 2016-12-14

Porównanie technologii tworzenia aplikacji internetowych JEE na przykładzie JavaServer Faces i Spring Boot

Michał Marcin Kizeweter*, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono wyniki porównania efektywności wytwarzania aplikacji internetowych na platformie Java Enterprise Edition z zastosowaniem JavaServer Faces i Spring Boot. Analiza porównawcza została przeprowadzona na bazie specjalnie przygotowanych aplikacji testowych, zaimplementowanych w obu technologiach.

Słowa kluczowe: Spring Boot; JavaServer Faces; aplikacje internetowe

*Autor do korespondencji.

Adres e-mail: michalkizeweter@gmail.com

Comparison of JEE platform web applications development using JavaServer Faces and Spring Boot example

Michał Marcin Kizeweter*, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the results of the web applications development effectiveness on the Java Enterprise Edition platform using JavaServer Faces and Spring Boot. The comparative analysis was performed using the specially prepared test applications, implemented in both technologies.

Keywords: Spring Boot; JavaServer Faces; web application development

*Corresponding author.

E-mail address: michalkizeweter@gmail.com

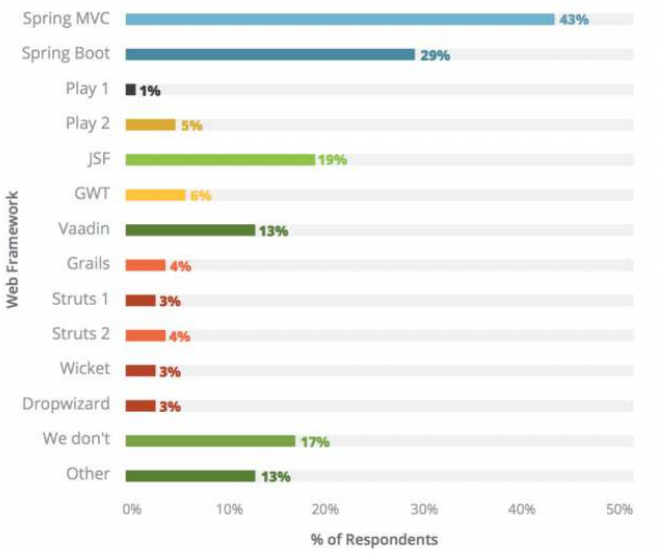
1. Wstęp

Stworzenie nawet prostej aplikacji internetowej w Javie wymaga zwykle wielu plików konfiguracyjnych *xml*, uruchomienia kontenera aplikacji, wdrożenia i wykonania wielu innych powtarzalnych czynności. Wszystko to powoduje, że wytwarzanie aplikacji web na platformie JEE nie jest proste, pomimo od dawna istniejących technologii wspomagających jak Spring czy JavaServer Faces (JSF).

Nowym rozwiązaniem, które ułatwia ten proces jest Spring-Boot. Projekt powstał w celu przyspieszenia i uproszczenia startu z popularnym szkieletem programistycznym Spring (rysunek 1). Spring Boot wprowadził automatyczną konfigurację dla Spring i wyeliminował całkowicie pliki *xml.* Tym samym projekt Spring Boot idealnie nadaje się do projektów studenckich i szybkiego prototypowania aplikacji. Jedynym wymaganiem stawianym przed użytkownikiem jest znajomość struktury projektu Maven, popularnego narzędzia automatyzującego budowę oprogramowania na platformie Java.

Z uwagi na dużą popularność jaką osiągnął Spring Boot w roku 2016 (rysunek 1 - 29% popularności), autorzy przeanalizowali jego konkurencyjność w stosunku do starszego frameworka JSF (rysunek 1 - 19% popularności), który nie jest oparty na założeniach Spring.

Najważniejsze cechy Spring Boot i JSF zostały zestawione w tabeli 1.



Rys. 1. Popularność wybranych frameworków Java w roku 2016 [1]

W artykule przedstawiono wyniki porównania efektywności wytwarzania aplikacji internetowych w oparciu o JSF i Spring Boot. Analiza porównawcza została przeprowadzona na bazie specjalnie przygotowanych aplikacji testowych, zaimplementowanych w obu technologiach.

Tabela 1. Najważniejsze cechy Spring i JSF

	Spring	JSF
Twórca	Pivotal Software	Oracle
Pierwsza wersja	Czerwiec 2003 (Spring 0.9)	Marzec 2004 (JSF 1.0)
Wersja finalna	Lipiec 2016 (Spring 4.3.2, Spring Boot 1.4.0)	Maj 2013 (JSF 2.2)
Język progr.	Java	
SDK	Java EE 7 SDK	
IDE	NetBeans (Open source), Eclipse (Open source), IntelliJ IDEA (komercyjne)	
Technologia widoku	Brak	Facelet
Inne możliwe tech. widoku	Facelet, JSP, Groovy, AngularJS	JSP
Serwery open-source	GlassFish, Apache Tomcat (wbudowany w Spring Boot)	GlassFish, Apache Tomcat
Wsparcie	Bogate wsparcie i dokumentacja, duża baza użytkowników	
Licencja	Open Source	
Standard stron widoku	Brak	XHTML1.1, HTML5
Niezależność platformy systemowej	Windows, Mac, Linux	
Inne	Struktura aplikacji MVC, Spring nie posiada natywnej technologii widoku, Spring Boot posiada wbudowany kontener aplikacji	Struktura aplikacji MVC, podstawą działania są komponenty zarządzane, generowanie widoków za pomocą facelet

2. Cel, teza i metody badań

Celem badań było porównanie efektywności tworzenia prostej aplikacji internetowej w wybranych frameworkach.

W artykule postawiono następującą tezę:

Framework Spring Boot jest bardziej efektywnym narzędziem wytwarzania aplikacji JEE w porównaniu do JavaServer Faces.

Dla potwierdzenia tej tezy wykorzystano metodę badań opartą na analizie porównawczej obu frameworków.

W tym celu stworzono dwie, funkcjonalnie identyczne, testowe aplikacje internetowe wykorzystując następujące narzędzia i technologie:

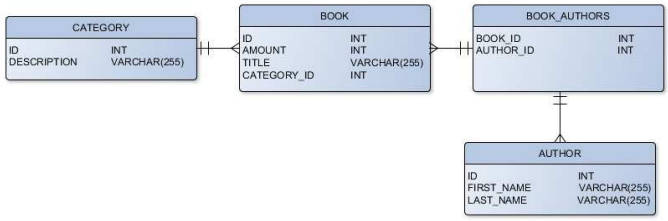
- szkielety programistyczne Spring-Boot oraz JavaServer Faces;
- środowisko developerskie Eclipse [2];
- serwer bazy danych PostgreSQL [3];
- Maven - narzędzie do automatyzacji budowy projektów Java [4];
- AngularJS - framework JavaScript [5];
- Hibernate – framework służący do realizacji warstwy dostępu do danych.

3. Aplikacje testowe

Proste aplikacje testowe (do obsługi biblioteki) zostały zaprojektowane w oparciu o pozycje [6, 7].

Obie aplikacje (Spring Boot i JSF) posiadają identyczne funkcjonalności: dodawanie nowych, edycję, usuwanie istniejących kategorii książek, autorów i książek. Są to typowe funkcjonalności aplikacji typu CRUDS (ang. Create, Read, Update, Delete, Search).

Schemat wykorzystanej bazy danych przedstawiono na rysunku 2. Obie aplikacje korzystają z tej samej bazy danych na serwerze PostgreSQL.



Rys. 2. Schemat bazy danych

Interfejs obu aplikacji umożliwia pełny dostęp do wszystkich, wcześniej opisanych funkcjonalności (Rys. 3).

Rys. 3. Przykładowa strona aplikacji testowej

4. Analiza porównawcza

W tabeli 2 przedstawiono parametry sprzętowe komputera, na którym wykonywano testy.

Tabela 2. Parametry sprzętowe zestawu pomiarowego

Element	Stan/Wersja
System operacyjny	Windows 8.1 64-bit
Procesor	Intel Core i7-4700MQ (2,40-3,40 GHz)
Pamięć RAM	8 GB (SO-DIMM DDR3)
PostgreSQL	9.6
ApacheTomcat	8.5
Java	Java EE7

W celu porównania obu aplikacji analizowano 4 kryteria:

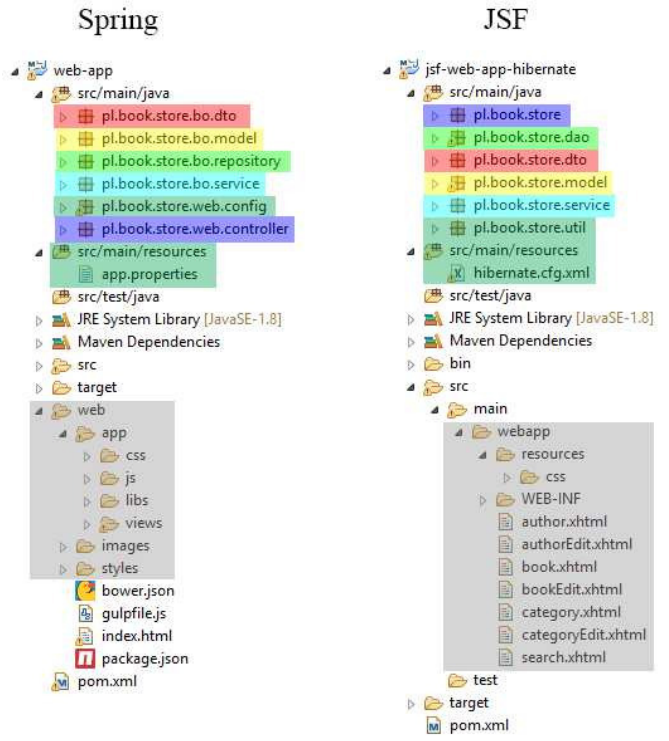
- ogólną strukturę aplikacji;
- efektywność pracy z danymi;
- efektywność wczytywania zasobów aplikacji przez przeglądarkę internetową;
- podstawowe metryki kodu.

4.1. Struktura aplikacji

Rysunek 4 przedstawia strukturę obu aplikacji, opartą na projekcie Maven. Na rysunku wyróżniono kolorem elementy funkcjonalnie wspólne dla obu aplikacji:

- czerwonym zaznaczono pakiet klas **dto**, które mapują encje bazy danych na obiekty w aplikacji;
- żółty wskazuje pakiety dla **modeli**, zawierające klasy mapujące encje z bazy danych za pomocą adnotacji JPA;

- jasno zielony oznacza pakiety klas **repozytoriów** dostępu do danych w bazie, zawierające deklaracje zapytań;
- jasno niebieski oznacza pakiety usług (**service**), zawierające klasy łączące kontrolery z repozytoriami (warstwa logiki biznesowej);
- ciemno zielonym zaznaczono pliki konfiguracyjne obu aplikacji, m.in. połączenia z bazą danych;
- ciemno niebieskim oznaczono **kontrolery** obu aplikacji (klasy odbierające żądania użytkownika);
- szary wskazuje pliki odpowiedzialne za **widoki.**.



Rys. 4. Struktura projektu Spring Boot i JSF

Dzięki zastosowaniu struktury Maven, elementy obu aplikacji są zbliżone. Główne różnice to pliki konfiguracyjne Hibernate (w Spring Hibernate jest głębiej zintegrowany), oraz obecność AngularJS, generującej widoki dla Spring.

4.2. Efektywność pracy z bazą danych

W celu porównania wydajności pracy obu aplikacji, zmierzono czas pracy z danymi. Do testu przygotowano 5 scenariuszy (Tabela 3). Każdy ze scenariuszy powtarzano minimum 10 razy a wynik pomiaru uśredniono.

Tabela 3. Scenariusze testowe

Scenariusz	Opis
1	Zapis 94 autorów do tabeli Author.
2	Zapis 100 książek w tabeli Book.
3	Odczyt wszystkich książek z tabeli Book.
4	Wyszukanie książek w tabeli Book (losowo wybranymi słowami kluczowymi).
5	Zapis wielu książek jednocześnie do tabeli Book.

Testowy zbiór książek i autorów do bazy danych pobrano ze strony [8]. Czasy mierzono w milisekundach [ms]. Operacje na bazie danych realizowano za pomocą Hibernate.

Fragment kodu z pomiarem czasu operacji według scenariusza 1 dla Spring i JSF, pokazują odpowiednio przykłady 1 i 2. W pozostałych scenariuszach, pomiar odbywał się analogicznie.

Przykład 1. Pomiar czasu dodania nowego autora - Spring

```
Author newAuthor = new Author();
newAuthor.updateAuthor(dto.getFirstName(),
dto.getLastName());
long startMillis = System.currentTimeMillis();
repository.save(newAuthor);
log.info("Add author {}", (System.currentTimeMillis() -
startMillis));
```

Przykład 2. Pomiar czasu dodania nowego autora- JSF

```
public void insertAuthor(AuthorDto dto) {
Long start = System.currentTimeMillis();
AuthorDao dao = new AuthorDao();
Author author = new Author();
author.setId(dao.getId());
author.setFirstName(dto.getFirstName());
author.setLastName(dto.getLastName());
author.setBooks(Sets.newHashSet());
dao.save(author);
System.out.println("Dodaj autora : " +
(System.currentTimeMillis() - start));
```

W przypadku scenariusza 5 dodawano jednocześnie 10, a następnie 100 książek. Każdą operację wykonano 10 razy a wyniki uśredniono.

W tabelach 4 i 5 przedstawiono rezultaty pomiarów dodawania jednocześnie 10 i 100 książek

Tabela 4. Czasy dodawania jednocześnie 10 książek

	Spring	JSF
Nr. Pomiaru	**Czas [ms]**	
1	89	226
2	84	214
3	85	272
4	70	235
5	72	227
6	84	262
7	71	230
8	75	245
9	71	238
10	69	221
Średni czas	**77**	**237**

Tabela 5. Czasy dodawania jednocześnie 100 książek w obu aplikacjach testowych

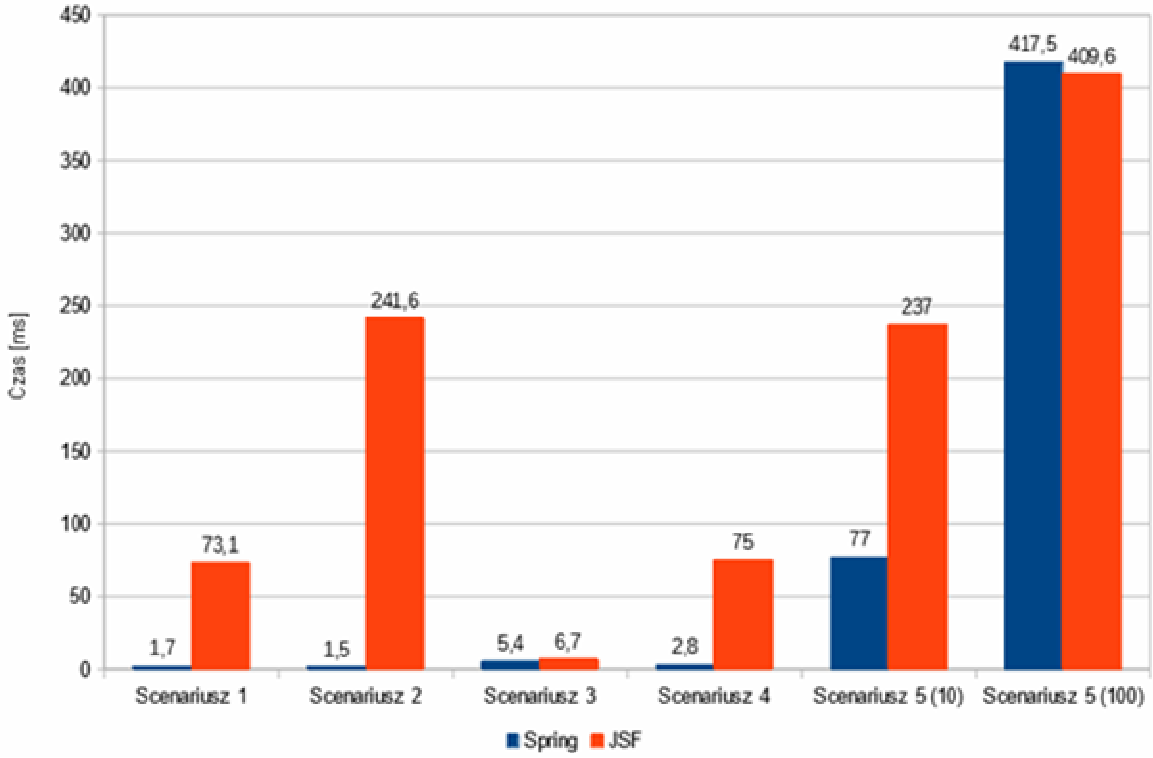
	Spring	JSF
Nr. Pomiaru	**Czas [ms]**	
1	475	397
2	397	448
3	431	399
4	404	395
5	415	420
6	382	416
7	467	390
8	392	392
9	431	437
10	381	402
Średni czas	**417,5**	**409,6**

Na rysunku 5 zestawiono średnie czasy pomiarów dla scenariuszy 1-5. Porównując obie aplikacje, w kategorii dodawania nowych rekordów (scenariusz 1 i 2) , Spring-Boot jest zdecydowanie szybszy od JSF. W przypadku pobierania rekordów (scenariusz 3) - Spring jest nieznacznie szybszy. Dla scenariusza 4 (wyszukiwanie po słowach kluczowych), Spring jest szybszy średnio o 70 milisekund od JSF.

Przy dodaniu większej liczby rekordów jednocześnie (scenariusz 5), Spring jest szybszy przy 10 rekordach, jednak dla 100 rekordów nieznacznie efektywniejszy okazuje się JSF.

4.3. Efektywność wczytywania zasobów aplikacji w przeglądarce internetowej

Kolejnym ważnym pomiarem wydajności aplikacji webowych jest czas mierzony od momentu wystąpienia żądania użytkownika, do momentu załadowania wszystkich potrzebnych zasobów. Czas ten został zmierzony za pomocą konsoli deweloperskiej Google Chrome w odpowiedzi na 20 żądań użytkownika. Wyniki pomiarów przedstawia tabela 6.



Rys. 5. Średnie czasy operacji dla wszystkich scenariuszy testowych dla Spring Boot i JSF

Tabela 6. Czas pobierania zasobów przez przeglądarkę Chrome

	Spring	JSF			Spring	JSF
Nr. Pomiaru	**Czas [ms]**			**Nr. Pomiaru**	**Czas [ms]**	
1	386	261		**11**	30	268
2	35	253		**12**	34	250
3	32	248		**13**	30	273
4	33	228		**14**	29	259
5	30	256		**15**	33	251
6	30	226		**16**	31	277
7	31	248		**17**	27	260
8	32	231		**18**	30	264
9	35	265		**19**	27	240
10	29	227		**20**	31	247
Średni czas:	48,75	251,6				

Pierwszy pomiar w aplikacji Spring, jest niewspółmiernie długi w porównaniu do reszty pomiarów. Jest to spowodowane tym iż przeglądarka pierwszy raz otwierając aplikację musi pobrać wszystkie skrypty Java Script i zasoby AngularJS potrzebne do jej prawidłowego działania. Ponieważ jest to Single Page Application (SPA), przy kolejnej nawigacji pobierane jest tylko minimum danych.

Patrząc na wyniki pomiarów aplikacji JSF, trzeba wziąć pod uwagę zaobserwowane w poprzednim teście długie czasy dostępu do bazy danych. Hipotetycznie, gdyby można je skrócić, prędkość pobierania stron aplikacji JSF byłaby bardziej zbliżona do czasów aplikacji Spring.

4.4. Porównanie metryk aplikacji

Ostatnim elementem porównania obu aplikacji jest analiza metryk kodu. Do tego pomiaru wykorzystano aplikację LocMetrics [9]. Zlicza ona linie kodu z pominięciem linii pustych oraz komentarzy. Wyniki pomiarów przedstawiono w tabeli 7.

Liczba linii kodu dla Spring Boot jest mniejsza od aplikacji JSF, ponieważ Spring automatycznie konfiguruje oraz samodzielnie zarządza wieloma aspektami aplikacji (np. mapowaniem, Hibernate, aspektami MVC). W JSF należy ręcznie to wszystko skonfigurować oraz samodzielnie implementować metody do pracy z danymi.. Widać to w tabeli 7 na przykładzie liczby bibliotek używanych przez obie

aplikacje i różnicy długości kodu kontrolerów. Kod kontrolerów jest zawarty w tej samej liczbie plików w obu aplikacjach testowych.

W przypadku Spring Boot liczba linii kodu widoku oraz liczba plików, w których jest on zawarty, jest znacznie większa od tej dla widoku JSF. Spring nie posiada własnego mechanizmu tworzenia widoku a tylko gotowe interfejsy, które należy podłączyć. W tym przypadku zastosowano bibliotekę AngularJS.

Liczba linii kodu modelu jest zbliżona, ponieważ obie aplikacje mają taką samą funkcjonalność.

Aplikacja Spring Boot zajmuje znacznie większą przestrzeń dyskową ponieważ posiada wbudowany kontener aplikacji Tomcat oraz bibliotekę AngularJS.

Tabela 7. Wybrane metryki kodu obu aplikacji

	Liczba linii kodu	
	Spring	**JSF**
Kod Java	599	731
Kod html	203	-
Kod JavaScript/Angular	471	-
Kod xhtml	-	378
Inne pliki konfiguracyjne	13	59
Razem linijek kodu	1286	1168
Liczba bibliotek	90	36
. Kontroler – linie kodu	97	235
Kontroler – liczba plików	3	3
Widok – linie kodu	674	378
Widok – liczba plików	23	8
Model – linie kodu	502	496
Model – liczba plików	19	13
Waga projektu (KB)	**2524**	**96**

5. Wnioski

Na podstawie przeprowadzonych badań można sformułować następujące wnioski:

- zastosowanie zarówno Spring Boot jak i JSF wymaga znajomości podstaw technologii Java oraz JEE;
- Spring jest stabilną i stale rozwijającą się platformą developerską, oferującą dość bogate możliwości przystosowania do potrzeb własnych projektów oraz predefiniowanych konfiguracji, które ułatwiają start początkującym programistom;
- JSF jest zdefiniowanym standardem platformy JEE i cieszy się oficjalnym wsparciem ze strony firmy Oracle, ale wymaga aby programista samodzielnie zadbał o każdy aspekt aplikacji (konfiguracja serwletu, konfiguracja bazy danych);
- wokół obu frameworków istnieje bogata społeczność developerska, zrzeszająca zarówno amatorów jak i profesjonalnych developerów, obie posiadają bogatą dokumentację techniczną;
- na podstawie. przeprowadzonych badań, Spring Boot, dzięki gotowym wbudowanym metodom CRUD, znacznie szybciej przeprowadza operacje na bazach danych;
- aplikacja JSF jest lżejsza, ale oferuje ograniczony potencjał rozbudowy;
- do obsługi widoków w aplikacji Spring Boot, należy wykorzystać dodatkowe technologie widoków (np. AngularJS);
- Spring Boot dostarcza gotowy kontener aplikacji w postaci wbudowanej dystrybucji Apache Tomcat.

Wyniki badań pozwalają potwierdzić postawioną tezę - Spring Boot jest korzystniejszym wyborem zarówno dla początkujących, jak i zaawansowanych programistów. Daje developerom gotowe elementy kodu, które należy zastosować do tworzonego projektu, a nie budować je od podstaw. Aplikację zbudowaną na bazie Spring Boot można dość swobodnie łączyć z innymi rozwiązaniami opartymi nie tylko o język Java.

Literatura

[1] https://zeroturnaround.com/rebellabs/most-popular-java-frameworks-tools-and-libraries-2016/ [10.11.2016]

[2] https://www.eclipse.org/ [07.12.2016]

[3] https://www.postgresql.org/ [07.12.2016]

[4] https://maven.apache.org/ [07.12.2016]

[5] https://angularjs.org/ [07.12.2016]

[6] Vishal Layka, *Java Projektowanie aplikacji WWW*, przeł. Lech Lachowski, Gliwice, Helion, 2015

[7] Vishal Layka, *Learn Java for Web Development*, Apress, 2014

[8] http://booklips.pl/zestawienia/100-ksiazek-ktore-trzeba-przeczytac-wg-polskich-dziennikarzy-i-pisarzy/ [08.11.2016]

[9] http://www.locmetrics.com/ [08.11.2016]

JCSI 3 (2017) 33-37
Journal of Computer Sciences Institute

Wysłane: 2016-11-29
Przyjęte: 2016-11-30

Zwiększenie efektywności procesu tworzenia aplikacji internetowych poprzez połączenie frameworków Meteor JS i Angular JS

Viacheslav Nishtuk*, Elżbieta Miłosz[a]
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu było przeprowadzenie analizy porównawczej frameworków AngularJS i MeteorJS, oraz próba ich połączenia w procesie tworzenia aplikacji internetowej Analiza porównawcza została wykonana na podstawie dokumentacji technicznej wybranych narzędzi oraz na podstawie oceny procesu tworzenia przykładowej aplikacji internetowej z ich udziałem. Wybrane zostały mocne strony każdego z narzędzi, zostały one wykorzystane w procesie tworzenia aplikacji internetowej, w którym połączono frameworki przydzielając każdemu z nich okresloną funkcjonalność aplikacji. Połaczenie frameworków miało na celu zwiększenie efektywności procesu wytwarzania aplikacji internetowej.

Słowa kluczowe: MeteorJS; AngularJS; JS - frameworki; połączenie frameworków.

*Autor do korespondencji.

Adres e-mail: viacheslav@ideaua.com

Increasing an efficiency of the web-applications developing the process through the combine of frameworks MeteorJS and AngularJS

Viacheslav Nishtuk*, Elżbieta Miłosz[a]
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The comparative analysis was made on the basis of technical documentation selected tools and on the process of creating a sample web application. Strengths of each tool were selected and have been used in the process of developing a web application that combines frameworks by assigning each of them a specific application functionality. Combining of frameworks aimed increasing the efficiency of the developing process of web application.

Keywords: MeteorJS; AngularJS; JS - frameworks; an unification frameworks.

*Corresponding author.

E-mail address: viacheslav@ideaua.com

1. Wstęp

Celem artykułu było przeprowadzenie analizy porównawczej dwóch obecnie popularnych frameworków do tworzenia stron internetowych w języku Java Script: AngularJS i MeteorJS, wyjaśnienie ich wad i zalet na podstawie oceny procesów tworzenia z ich pomocą aplikacji internetowej oraz próba ich połączenia tj. wykorzystania obydwu frameworków w jednej aplikacji. Połączenie miało na celu zwiększenie efektywności procesu wytwarzania aplikacji.

2. Metody i narzędzia tworzenia stron internetowych

Pierwszym ważnym zadaniem w procesie tworzenia strony internetowej jest planowanie jej układu zawartości i wyglądu . Proces ten można podzielić na kilka etapów: generowanie pomysłów, tworzenie struktury projektu, wybór technologii i badanie układu projektu. Przykładowy szkicowy projekt serwisu przedstawiono na rys.1.

Metody tworzenia strony WWW są podzielone na dwie grupy: ręczne pisanie kodu strony internetowej i automatyczne metody tworzenia strony internetowej z wykorzystaniem narzędzi typu framework [1]. Do drugiej grupy należą takie narzędzia jak Joomla i Wordpress, które z kolei pozwalają szybko stworzyć prostą stronę internetową bez znajomości programowania.

Technologie sieci Web, które umożliwiają tworzenie stron internetowych czy aplikacji sieci Web to [2]:

- Języki znaczników (HTML, XML);
- Kaskadowe arkusze stylów (CSS, SCSS, SASS, LESS);
- Język skryptowy JavaScript;
- Język skryptowy PHP;
- Obiektowy język programowania JAVA;
- Technologia NodeJS;

Narzędzia typu frameworki, są dostępne w różnych językach programowania, najczęściej wykorzystywane są js-frameworki lub php-frameworki.

3. Cel, hipoteza i metody badań

Celem badań było porównanie efektywności tworzenia prostej aplikacji internetowej w 2 frameworkach, ocena słabych i silnych stron frameworków oraz próba połączenia

tych frameworków w jednej aplikacji w celu. zwiększenia efektywności procesu wytwarzania aplikacji.

W artykule postawiono następujące *pytania badawcze*:

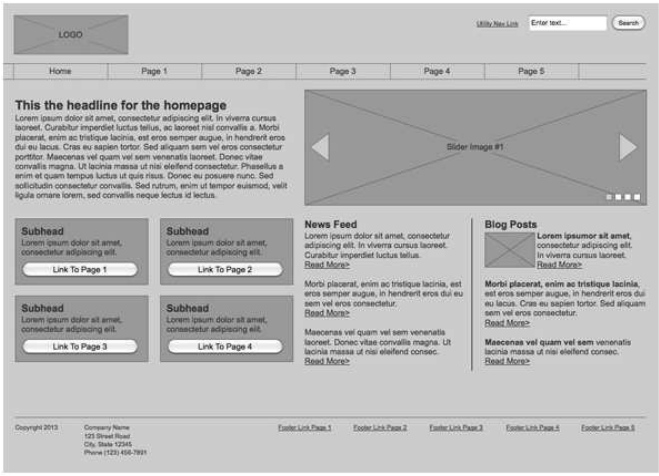
1. Który z frameworków łatwiej realizuje wybrane funkcje aplikacji?
2. Czy możliwe jest połączenie 2 frameworków w celu zwiększenia efektywności tworzenia aplikacji?

Dla uzyskania odpowiedzi na postawione pytania badawcze sformułowano następującą *hipotezę badawczą*:

Możliwe jest połączenie frameworków: Meteor JS i Angular JS w procesie tworzenia aplikacji internetowej.

Dla potwierdzenia hipotezy wykorzystano następujące *metody badań*:

- Analiza porównawcza frameworków
- Eksperyment połączenia frameworków.



Rys. 1. Szkicowy zarys serwisu [3]

4. Analiza porównawcza frameworków

Porównanie frameworków jest istotne dla społeczności programistów podczas podejmowania decyzji: "Jaki framework mam wybrać dla następnego projektu?". Każdy programista przed tym, jak ma zacząć korzystać z nowego frameworka, porównuje go z tym, który wcześniej używał.

Analizie porównawczej poddano dwa najbardziej popularne obecnie frameworki. Pierwszym był MeteorJS, który przyciąga dewelopera dostępnością, powłoką i logiką biznesową NodeJS. Drugim został wybrany AngularJS, który jest aktualnie najbardziej popularnym frameworkiem [4]. Wyniki porównania frameworków przedstawiono w tabeli 1.

W rzeczywistości wybór frameworka zależy od potrzeb programisty i zadań do realizacji, jedne frameworki mogą radzić sobie lepiej z określonym zadaniem, a inne, odpowiednio lepiej z innym zadaniem. Nie ma idealnego frameworka, wszystkie one koncentrują się na określonych sprofilowanych zadaniach.

5. Eksperyment połączenia frameworków

Eksperyment połączenia frameworków wykonano dla przykładowej aplikacji internetowej, w której użytkownik mógł samodzielnie tworzyć notatki, a przy tym uzyskać do nich dostęp z dowolnego urządzenia w dowolnym miejscu z dostępem do Internetu. Aplikacja powinna zapewnić możliwość ustawienia przypomnienia dla użytkownika, tworzenia notatek, udostępniania ich innym użytkownikom w postaci określonych grup notatek, albo konkretnych wybranych notatek.

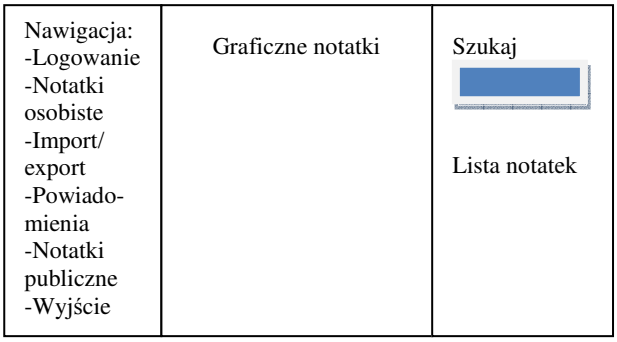
Do projektowania witryny sieci Web wybrano szereg kryteriów, a mianowicie:

- prosty trójkolumnowy układ strony,
- zapewnienie logiki biznesowej (back-end) – szybka realizacja zapytań do bazy danych.

Tabela 1. Porównanie frameworków AngularJS i MeteorJS [5, 6, 7]

Kryterium porównania	**AngularJS**	**MeteorJS**
Full-stack framework	Nie, tylko MVC w kliencie	Tak
Logika biznesowa (Back-end)	Dowolny	NodeJS
Preprocesor	Nie	Tak
Dynamiczne połączenie html z danymi klienta	Tak	Tak
Renderowanie html na serwerze	Nie	Tak
Społeczność	Npm, Bower	Atmospherejs, Npm, Bower
Reaktywność	Działa tylko w $scope	Wszystko
Systememy szablonów	Oficjalnego wsparcia nie ma, ale można użyć innych	Oficjalne wsparcie Blaze, Handerbars, Jade
Baza danych	Wszystko zależy od Back-end (dowolna)	NodeJS obsługuje mongoDB, ale można użyć dowolnej
Synchronizacja danych między klientami	Nie	Są (Optimistic UI)
Kanał synchronizacji danych	Nie	DDP protokół (web-sockets)
Aktualizacja aplikacji bez konieczności ponownego uruchamiania (wygoda rozwoju)	Nie	Tak — html, css, js

Projekt układu strony przedstawiono na rys.2.



Rys. 2. Projekt układu strony do eksperymentu

Do realizacji badania przedstawiony projekt został napisany w każdym z dwóch frameworków, a następnie utworzono aplikację, w której połączono 2 frameworki, które realizowały w optymalny dla siebie sposób wybrane funkcje. W aplikacjach zaimplementowano następujące funkcjonalności:

- autoryzację i rejestrację użytkowników (logowanie)
- ustawienia notatek osobistych
- import/export notatek
- powiadomienia o notatkach
- ustawienia notatek publicznych
- wyjście z aplikacji.

Realizację usługi autoryzacji i rejestracji do strony internetowej zrealizowaną w AngularJS, można zobaczyć na listingu 1. Logowanie i rejestracja działa na "websocket", który jest bezpośrednio podłączony do back-end. Tak jak większość funkcji witryny potrzebują oprogramowania logiki biznesowej, więc do AngularJS był podłączony NodeJS. AngularJS w odróżnieniu od MeteorJS ma wbudowany back-end (NodeJS synchroniczny). W MeteorJS podobną funkcjonalność realizowano w łatwiejszy sposób, ponieważ MeteorJS ma wbudowane moduły, które pracują z identyfikacją i rejestracją użytkownika (przykład 2).

Przykład 1. Usługa odpowiedzialna za autoryzację i rejestrację użytkownika w AngularJS [8]

```
angular.module('noteApp').service('AuthService',
function($rootScope, $cookies, $routeSegment, $location, mySocket)
{
    $rootScope.bool.isLoading    = true;
    this.status = {
   authorized: false,
  };
  this.check = function(data, callback){
    mySocket.emit('authCheck', $cookies.get('token'), function(data)
{
    if(data){
        $rootScope.hideBlocks = true;
            callback(self.status.authorized = true);
        }else{
            $rootScope.hideBlocks = false;
            callback(self.status.authorized = false);
        }
            $rootScope.bool.isLoading    = false;
        });
  }
  this.logIn = function(key, callback){
    mySocket.emit('authLogin', {key: key}, function(data) {
            if(data.token){
                $cookies.put('token', data.token);
                $rootScope.hideBlocks = true;
                callback(self.status.authorized = true);
            }else{
                $rootScope.hideBlocks = false;
                callback(self.status.authorized = false);
            }
            $rootScope.bool.isLoading    = false;
        });
  }
  this.register = function(key, callback){
    mySocket.emit('authRegister', {key: key}, function(data) {
            if(data.token){
                $cookies.put('token', data.token);
                $rootScope.hideBlocks = true;
                callback(self.status.authorized = true);
            }else{
                $rootScope.hideBlocks = false;
                callback(self.status.authorized = false);
            }
            $rootScope.bool.isLoading    = false;
        });
  }
})
```

Po wykonaniu aplikacji w dwóch frameworkach podsumowano doświadczenie z procesu tworzenia aplikacji, oceniono, który framework pozwalał w łatwiejszy sposób realizować funkcje aplikacji. Duża różnica między AngularJS i MeteorJS - to obecność gotowego „back-end-a" w MeteorJS - NodeJS (synchroniczny). Dlatego napisanie tej funkcji z wykorzystaniem AngularJS zajmuje więcej czasu niż z wykorzystaniem MeteorJS. MeteorJS dobrze współpracuje z bazą danych z frond-end ' em i dlatego napisanie funkcjonalności do tworzenia czy edycji notatek w MeteorJS jest łatwiejsze.

W kolejnym kroku podjęto próbę połączenia frameworków rozdzielając funkcje aplikacji między dwa frameworki. MeteorJS zajmie się pracą z bazą danych i wszystkim co dotyczy logiki biznesowej (back-end), a AngularJS będzie działał z warstwą widoku i prezentacją danych (front-end)).

Przykład 2. Fragment kodu rejestracja / logowanie użytkownika w MeteorJS

```
Template.formRegist.events({ //meteor method
 'click #register': function (event, template) { //register
        var user = {
        login: $(event.currentTarget).children('input#login').val(),
        password: $(event.currentTarget).children(#password').val()
        };
        if(validFormInput(user)){
            //meteor method
            Accounts.createUser(user, function(error, result) {
                if(error){
                    thow error;
                }else if(result){
                    showMessageDisplay('jestes zarejestrowany');
                    initNote(user);
                }
            });
        }else{
            showMessageDisplay('invalid data');
        }
 },
 'click #login': function(event, template){ // login
        var user = {
        login: $(event.currentTarget).children('input#login').val(),
        password:$(event.currentTarget).children('#password').val()
        };
        if(validFormInput(user)){
            // meteor method
            Meteor.loginWithPassword(user.login, user.password, f
                function(error, result) {
                    if(error){
                        thow error;
                    }else if(result){
                        showMessageDisplay('welcome');
                        initNote(user);
                    }
                });
        }else{
            showMessageDisplay('invalid data');
        }
 }
});
```

Do zadań realizowanych za pomocą MeteorJS należy:

- logika aplikacji;
- render szablonów;
- routing między szablonami;
- praca z bazą danych;
- RestAPI.

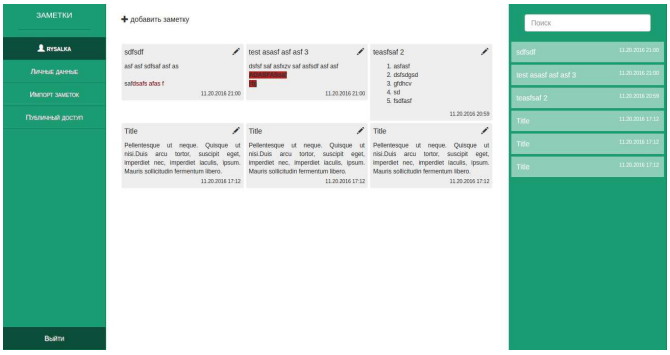
Z kolei AngularJS będzie realizował następujace funkcje:

- data binding $scope;
- traktowanie niektórych zdarzeń (zmiany tekstu w czasie rzeczywistym).

Po eksperymencie przeprowadzono analizę porównawczą stosowanych frameworków, zarówno między sobą, jak i z wynikiem ich połączenia.

Realizacja funkcji rejestracji/logowania i innych funkcji nawigacji strony najbardziej przyjazna dla programisty jest z wykorzystaniem frameworka MeteorJS. Podczas tworzenia strony internetowej w MeteorJS powstało znacznie mniej problemów niż w AngularJS, gdzie bardzo dużo problemów było z wykorzystaniem wbudowanych metod, a dokładniej ze słabą do nich dokumentacją. Do wyszukiwania i usuwania błędów przeprowadzono ponowne debugowanie całego kodu, na co poświęcono dodatkowy czas.

Wygląd końcowy aplikacji powstałej na podstawie projektu i wykonanej w 3 frameworkach przedstawia rys.3.



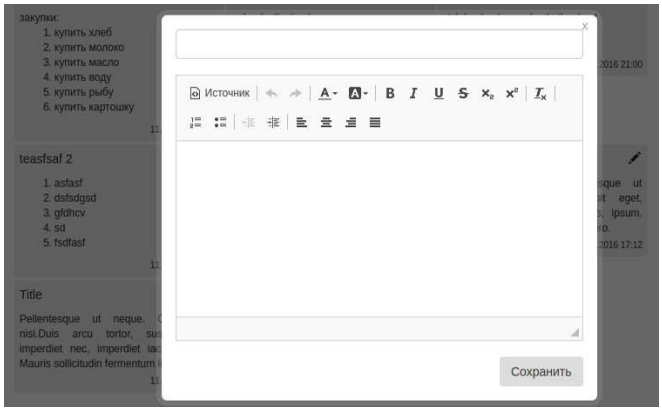
Rys. 3. Wygląd aplikacji dla notatek

Do edycji tekstu notatek był używany edytor tekstu CKEditor v4.6.0, który z kolei był nastawiony na konkretne zadania, konkretne funkcje, posiadał pasek narzędziowy do łatwiejszej edycji tekstu (Rys.4).

6. Rezultaty badań i wnioski

W wyniku badań zostały zidentyfikowane zalety i wady każdego z wybranych frameworków i funkcje, które wykonują bez zarzutu. Właśnie dla tego przeprowadzono połączenie frameworków w jednej aplikacji, aby przy minimalnym nakładzie pracy i czasu móc w pełni zrealizować wszystkie wymagane funkcje.

Hipoteza badawcza pracy: „Możliwe jest połączenie frameworków: Meteor JS i Angular JS w procesie tworzenia aplikacji internetowej" została potwierdzona.



Rys. 4. Formularz tworzenia notatek

Aby uniknąć konfliktów, podjęto decyzje podzielić kod między frameworkami. Kod AngularJS był używany do importu określonej funkcjonalności do głównego kodu MeteorJS, czyli części poszczególnych funkcjonalności znajdują się w różnych plikach, w różnych katalogach (struktura aplikacji jest podzielona między frameworkami), przy tym są całkowicie odizolowane od siebie.

Podczas realizacji scalenia frameworków okazało się dość trudne rozdzielić funkcje frameworków między sobą według łatwości i jakości wykonania. Została przeprowadzona szczegółowa praca nad dokumentacją MeteorJS, ponieważ wystąpił problem z izolacją: MeteorJS nie wymaga podania jaki plik ma w użyciu, automatycznie przetwarza wszystkie dostępne pliki na raz, tworząc z nich "bundle", w którym znajdują się jednocześnie kod MeteorJS i AngularJS. Przy tym powstały problemy z powodu braku możliwości zastosować wybrane moduły funkcjonalne określonego kodu MeteorJS. Problemy rozwiązano w postaci importu i izolacji AngularJS od zewnętrznych manipulacji. Przy prawidłowym pisaniu kodu, frameworki nie wchodziły w konflikt i nie przeszkadzały sobie nawzajem, każdy obsługiwał swoją część kodu. W rezultacie w aplikacji zostały zrealizowane wszystkie projektowane funkcje, przy tym nie ma strat w zakresie projektowania i pozycjonowania strony.

Literatura

[1] Сэмми Пьюривал, Основы разработки веб-приложений, Питер, 2015.

[2] Веб-технологии для разработчиков, https://developer.mozilla.org/ru/docs/Web

[3] Создание веб-сайта. Курс молодого бойца, https://habrahabr.ru/post/273795/

[4] ТОП 10 JavaScript фреймворков и библиотек, https://habrahabr.ru/post/305442

[5] Brad Green & Shyam Seshadri, AngularJS. O'Reilly Media, 2013.

[6] Tom Coleman, Sacha Greif, Discover Meteor: Building Real-Time JavaScript Apps. N/A, 2014.

[7] Kyle Simpson, You Don't Know JS: Up & Going, 2015

[8] David Flanagan, JavaScript: The Definitive Guide, 6th Edition. Activate Your Web Pages, 2011

JCSI 3 (2017) 38-45
Journal of Computer Sciences Institute

Wysłane: 2016-11-27
Przyjęte: 2016-12-14

Metody obróbki danych EMG

Michał Serej*, Maria Skublewska - Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule zostały przedstawione metody obróbki danych badania elektromiograficznego (EMG) oraz analiza sygnału EMG przy pomocy zaimplementowanej autorskiej aplikacji. Aplikacja służy do wczytania sygnału EMG zapisanego w pliku o rozszerzeniu .C3D. Analizę przeprowadzono pod względem największej aktywności mięśni podczas wykonywania ćwiczeń rejestrowanych przy pomocy techniki Motion Capture.

Słowa kluczowe: EMG; Motion Capture; Plik C3D

*Autor do korespondencji.

Adres e-mail: michalserej@gmail.com

The methods of EMG data processing

Michał Serej*, Maria Skublewska - Paszkowska
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents both the methods of data processing of electromyography (EMG), and EMG signal analysis using the implemented piece of software. This application is used to load the EMG signal stored in a file with the .C3D extension. The analysis was conducted in terms of the highest muscles activaton during exercise recorded with Motion Capture technique.

Keywords: EMG; Motion Capture; File C3D

*Corresponding author.

E-mail address: michalserej@gmail.com

1. Wstęp

W dzisiejszych czasach można zauważyć coraz większe znaczenie Informatyki w dziedzinie medycyny. Coraz więcej firm specjalizuje się w tworzeniu oraz rozwijaniu oprogramowania bądź urządzeń służących ludzkości w leczeniu poznanych już chorób. Dzięki nowym technologiom informatycznym nie tylko można wspomagać leczenie, ale również poznać źródło choroby, jej budowę, skład, a także jak się bronić przed nią lub zapobiegać jej powstawaniu. Rozwiązania informatyczne coraz częściej znajdują zastosowanie w medycynie. Systemy do elektromiografii pozwalają na uzyskanie danych odnośnie aktywności elektrycznej mięśni w postaci cyfrowej, co pozwala na zarządzanie nimi i ich analizowanie poprzez aplikacje, czy systemy komputerowe.

2. Obróbka danych EMG

2.1. Pojęcie EMG

Definicję EMG (elektromiografii) można znaleźć w książce wydanej przez J. Basmajian i C. De Luca pt. "Muscles Alive: Their Functions Revealed by Electromyography" a cytowanej w artykule "An automatic SSA-based de-noising and smoothing technique for surface electromyography signals" [1] : „EMG to technika wykorzystywana do wykrywania i analizowania sygnału elektrycznego, który wytwarzają mięśnie".

Jest to badanie mięśni przy pomocy sygnału elektrycznego. Tkanka mięśniowa przewodzi potencjały elektryczne podobne do nerwów. Takie sygnały nazywane są potencjałem czynnościowym mięśni. Badanie EMG prowadzi zapis informacji zawartych w tych potencjałach podczas działania mięśni. W trakcie wykrywania i rejestrowania sygnału EMG, istnieją dwa główne problemy wynikające z obawy o czystość badanego sygnału. Pierwszym z nich jest stosunek sygnału do szumu. Oznacza on stosunek energii sygnału EMG do energii sygnału szumu. Ogólnie rzecz biorąc, hałas jest definiowany jako sygnał elektryczny, który nie jest częścią pożądanego sygnału EMG. Inną kwestią jest zakłócenie sygnału, co oznacza, że względny udział dowolnego elementu częstotliwości w sygnale EMG nie powinien być zmieniany.

Do uzyskania odpowiedniego sygnału z mięśni wykorzystywane są dwa rodzaje metod – inwazyjna oraz nieinwazyjna. Metoda nieinwazyjna polega na zamontowaniu elektrod bezpośrednio na skórę, uzyskuje się wtedy składową sygnałów pochodzących ze wszystkich potencjałów czynnościowych włókien mięśniowych leżących pod skórą. Takie potencjały występują w losowych odstępach czasu, dlatego sygnał EMG może posiadać zarówno dodatnie jak i ujemne napięcie. Innym sposobem uzyskania wartości potencjałów jest metoda inwazyjna, która polega na umieszczeniu bezpośrednio w mięśniu drutu lub igłowych elektrod. Podczas badania sygnał jest odbierany na elektrodzie

i wzmacniany. Zazwyczaj do pierwszego wzmocnienia sygnału stosowany jest wzmacniacz różnicowy. Przed wyświetleniem, a następnie zapisem w odpowiednim miejscu, uzyskany sygnał EMG może ulec przetworzeniu w celu wyeliminowania hałasu lub innych możliwych zakłóceń. W konsekwencji, sygnał jest często prostowany i uśredniany wskazując amplitudę EMG. Zakres amplitudy sygnału EMG przed wzmocnieniem wynosi 0-10 mV (+5 do -5).

2.2. Szum elektryczny i czynniki wpływające na EMG

Podczas badań sygnał EMG może zostać zniekształcony poprzez różnego rodzaju zanieczyszczenia. Zakłócenia elektryczne, które mogą mieć wpływ na sygnał EMG, można podzielić na następujące rodzaje [1]:

1) Hałas w urządzeniach pomiarowych : Wszystkie urządzenia elektroniczne generują hałas, którego nie można wyeliminować; jednakże używając wysokiej jakości urządzeń zakłócenia mogą zostać zmniejszone.
2) Otaczający hałas: Przykładem takiego zakłócenia jest promieniowanie elektromagnetyczne. Powierzchnia ciała nieustannie ulega promieniowaniom magnetycznemu i elektrycznemu; praktycznie niemożliwe jest, aby uniknąć narażenia na tego typu działania promieniowania na powierzchni ziemi. Otaczający hałas może mieć amplitudę, która jest od jednego do trzech rzędów wielkości większa niż amplituda sygnału EMG.
3) Ruch elektrod: Podczas ruchu elektrod, informacja jest przekrzywiona przez co powoduje on nieprawidłowości w danych. Istnieją dwa główne źródła ruchu elektrod: 1) Interfejs elektrody i 2) Kabel elektrody. Ruch elektrod może być zmniejszony przez właściwe projektowanie obwodów elektronicznych oraz ich odpowiednią konfigurację.

Czynniki, które wpływają na sygnał EMG mogą być sklasyfikowane. Dzięki takiej klasyfikacji algorytmy analizy sygnału EMG mogą zostać zoptymalizowane, a sprzęt służący do pomiaru odpowiednio zaprojektowany i skonfigurowany. Czynniki wpływające na sygnał EMG dzieli się na trzy podstawowe kategorie [1]:

1) Czynniki sprawcze: Jest to bezpośredni wpływ na sygnał. Czynniki sprawcze można podzielić na dwie klasy:

   - Zewnętrzna : Wynika to ze struktury elektrody i miejsca docelowego. Czynniki takie jak obszar powierzchni wykrywania, kształt elektrody, odległości między elektrodą, a powierzchnią wykrywania, lokalizacja elektrody w odniesieniu do punktów mechanicznych w mięśniu.

   - Nieodłączna : fizjologiczne, anatomiczne, składu typu włókna, przepływu krwi, średnicy włókien, głębokości i rozmieszczenia włókien aktywnych i ilości tkanki pomiędzy powierzchnią mięśnia i elektrody.

2) Czynniki pośrednie: czynniki pośrednie są to zjawiska fizyczne i fizjologiczne. Przyczynami tego mogą być np.: szybkość przewodzenia potencjału czynnościowego, który rozchodzi się wzdłuż błony włókien mięśniowych.

3) Deterministyczne czynniki : Liczba aktywnych jednostek ruchowych, szybkość wypaleń mechanicznych i oddziaływania mechanicznego między włóknami mięśniowymi mają bezpośredni wpływ na informacje zawarte w sygnale EMG i zarejestrowanej siły. Amplituda, czas trwania i kształt potencjału czynnościowego jednostek ruchowych mogą być również odpowiedzialne za przepływ informacji wydobytych z sygnału EMG.

Maksymalizacja jakości sygnału EMG może być wykonana za pomocą następujących sposobów [1]:

1) Stosunek sygnału do szumu powinien zawierać największą ilość informacji z sygnału EMG oraz minimalną ilość zanieczyszczeń hałasu.

2) Zniekształcenie sygnału EMG musi być możliwie jak najmniejsze bez zbędnego filtrowania i zniekształcenia szczytu sygnału.

2.3. Przedstawienie metod obróbki danych EMG

2.3.1. Analiza Falkowa i Transformaty Fouriera

Transformata falkowa (ang. Wavelet Transform) jest skutecznym narzędziem matematycznym lokalnej analizy niestacjonarnych i szybkich nieustalonych sygnałów. Guglielminotti i Merletti postawili hipotezę, że jeżeli analiza falkowa wybierana jest tak, aby dopasować do kształtu MUAP (ang. motor unit action potential), czyli zmian elektrycznych generowanych przez badany mięsień, otrzymaną transformatę falkową to otrzymuje się najlepsze z możliwych lokalizacje energii w skali czasu [2]. W 1997 Laterza i Olmo udowodnili, że transformata falkowa jest alternatywą dla innych przedstawień częstotliwości z zaletami takimi jak liniowość, uzyskanie reprezentacji w wielu rozdzielczościach oraz brak krzyżowań składników [3]. Jest to szczególnie istotne, przy sygnałach wieloskładnikowych. W pewnych warunkach, sygnał EMG może być uważany jako suma przeskalowanych opóźnionych wersji jednego prototypu. W oparciu o teorię Guglielminotti'ego, Laterza i Olmo zastosowali analizę falkową tak, aby dopasować kształt MUAP dla jednobiegunowego zarejestrowanego sygnału stosując warunki zawarte w hipotezie przedstawionej przez Gabor'a w 1946 roku [4]. Wynik zasugerował użycie znanej falki meksykańskiego kapelusza, która w istocie jest pochodną drugiego rzędu z rozkładu Gaussa.

W 1998 roku Ismail i Asfour dowiedli, że najczęstszą metodą stosowaną w celu określenia spektrum częstotliwości EMG są szybkie i terminowe transformaty Fouriera (FFT i SFT) [5]. Stwierdzili również, że główną wadą tych metod transformacji jest to, że zakładają one, iż sygnał jest nieruchomy. Jednak sygnały EMG są niestacjonarne.

W 1999 Marios Pattichis i Constantinos Pattichis stwierdzili, że terminal transformaty falkowej może być użyty także do analizy sygnałów na różnych poziomach rozdzielczości [6]. Według teorii proces analizowania sygnałów na różnych poziomach rozdzielczości jest znany w postaci analizy wielorozdzielczej. Przeanalizowano zależność pomiędzy współczynnikami falkowymi, a płaszczyzną czasu i częstotliwości. Algorytm transformaty falkowej składa się z etapów fazy rozkładu i odbudowy. Pattichis i Pattichis krótko opisują w jaki sposób współczynniki z każdego etapu transformaty falkowej mogą być stosowane do konstruowania zbliżenia funkcjonalnego do oryginalnego sygnału. Podane próbki sygnału x_0 , x_1 , x_2 ..., uzyskują odpowiedni sygnał ciągły względem czasu, według wzoru 1 [6]:

$$\mathrm{f}^0(t) = \sum_k x_k \theta(t-k) \qquad \textbf{(1)}$$

gdzie ϴ(t-k) jest nazywana funkcją skalowania. Zakłada się, że próbkami sygnału są średnie ważone ciągłego sygnału.

W 2003 Kumar wyszedł z podobną hipotezą twierdząc, że transformata falkowa rozkłada się na kilka części składowych sygnału wielorozdzielczego zgodnie z funkcją zwaną "funkcją falkową" (WF) [7]. Funkcja falkowa jest zarówno rozszerzona i przetłumaczona w czasie podjęcia dwuwymiarowego przekroju korelacji z sygnałem w dziedzinie czasu badania aktywności elektrycznej mięśni (ang. surface electromyography (sEMG)). Ta metoda może być postrzegana jako mikroskop, to znaczy że zawiera narzędzie do wykrywania i charakteryzowania krótkiego składnika czasu w niestacjonarnym sygnale. Jest to technika, która zawiera informacje odnoszące się do zmian czasu i częstotliwości sygnału. Kumar stwierdził również, że krótka transformacja Fouriera (SFT) o stosunkowo krótkich przedziałach czasowych może próbować śledzić zmiany widmowe wraz z upływem czasu, ale nie przyjmuje optymalnego czasu oraz rozdzielczości częstotliwości dla sygnału niestacjonarnego.

W sEMG został rozłożony za pomocą transformaty falkowej ze zmienną funkcją falkową i obliczonym wyjściem mocy przekształcenia domeny i użytym jako parametr decydujący o doborze funkcji falkowje, która zapewnia najlepszy kontrast pomiędzy przypadkami sEMG. W wyniku prowadzonych badań, można stwierdzić, że za pomocą sEMG i falkowych transformacji, jest możliwe, aby określić, zmęczenie mięśni (osłabienie mięśni).

2.3.2. Podejście czasowo-częstotliwościowe

Próby uzyskania informacji ilościowej z nagrań EMG były przeprowadzane w momencie gdy sygnał był reprezentowany jako funkcja czasu (w dziedzinie czasu). Klasa transformacji Cohen'a [9], rozkład Wignera-Ville [11] i rozkład Choi-Williams [13] to tylko niektóre z metod czasowo-częstotliwościowych wykorzystywanych do przetwarzania sygnału EMG.

Piper wykazał, na początku tego wieku (1912), że podczas przedłużonego skurczu mięśnia składniki widma potencjału czynnościowego sygnału są ściskane w kierunku niższych częstotliwości [8]. Mechanizmy, które regulują to zjawisko zostały wyjaśnione w ciągu ostatnich dwóch dekad. Gdy sEMG jest rejestrowana w trakcie dynamicznych skurczów założenie o stacjonarności nie zachodzi, ponieważ zawartość częstotliwości sygnału stale zmienia się w czasie. Niestacjonarności z potencjału czynnościowego sygnału mogą być sklasyfikowane jako wolne lub szybkie [8]. Powolne niestacjonarności są głównie ze względu na nagromadzenie metabolitów, które wywołuje elektryczne objawy zmęczenia mięśni. Szybkie niestacjonarności związane są głównie z biomechaniką. Wahania siły mięśni powodują zmianę zawartości częstotliwości sygnału.

Transformacja klasy zaproponowana przez Cohena w 1995 roku cieszyła się wielkim zainteresowaniem, zwłaszcza w dziedzinie biomedycznego przetwarzania sygnału [9]. Klasa reprezentacji czasowo-częstotliwościowej jest szczególnie przydatna do analizy potencjału czynnościowego źródła sygnału zarejestrowanego podczas skurczów dynamicznych, który może być modelowany jako realizacje niestacjonarnego procesu stochastycznego. Cohen w 1995 dowiódł również, że jeżeli jądro transformacji g (θ, τ) = 1, uzyskany zostaje rozkład określany jako rozkład Wigner-Ville - jest on optymalny do analizy sygnałów, składający się z pojedynczego składnika. Jednakże, nie jest dobrze dostosowany do stosowania na sygnały wieloskładnikowe, ponieważ dwuliniowość transformaty powoduje obecność składników zakłóceń [9].

Rozkład Wignera-Ville jest czasowo-częstotliwościowy, który może wyświetlić częstotliwość jako funkcja czasu, a tym samym wykorzystanie wszystkich dostępnych informacji zawartych w sygnale EMG. Chociaż sygnał EMG często może być traktowany jako quasi-stacjonarny to zawiera jeszcze ważne informacje, które można odróżnić jedynie przez Rozkład Wignera-Ville. Ricamato w 1992 roku odkrył, że rozkład Wignera-Ville może być używany do wyświetlania zakresów częstotliwości jednostki napędowej [11]. Możliwe jest przedstawianie charakterystyki poboru w momencie wykonywanie przez mięsień trudnych zadań. Równanie rozkładu Wignera-Ville przedstawia się wzorem 2 [11]:

$$W_x(t,\omega) = \int_{-\infty}^{\infty} x(t+\frac{\tau}{x}) x^*(t-\frac{\tau}{2}) e^{-j\omega\tau} d\tau \qquad \textbf{(2)}$$

Implementacja rozkładu Wignera-Ville przy użyciu komputera wymaga dyskretnej postaci. Pozwala to na wykorzystanie szybkiej transformaty Fouriera (FFT), która wytwarza dyskretny czas oraz dyskretną reprezentację częstotliwości. Wspólnym typem rozkładu częstotliwości czasu jest krótki czas transformacji Fouriera (STFT). Według Daviesa i Reisman (1994), poważnym problemem rozkładu STFT jest to, że nie spełnia czterech istotnych właściwości, które są pożądane do rozkładu czasowo-częstotliwościowego [12]. Dwie właściwości są to czas oraz częstotliwość marginalna, pozostałe dwie są wsparciem czasu i częstotliwości. Informują one również, że wspólne widmo gęstości wyprodukowane przez rozkład Wignera-Ville jest bardzo zanieczyszczone, ale wyświetla bardzo dobre

właściwości lokalizacji i ogólnie koncentruje się wokół chwilowej częstotliwości sygnału. Sposób Choi-Williams zaproponowany w 1993 roku jest przykładem redukującym zakłócenia rozkładu [13]. Davies i Reisman, odkryli, że chociaż rozkład Choi-Williams nie spełnia wszystkich tych pożądanych właściwości rozkładu częstotliwości czasu, ale spełnia jedną ważną właściwość, czyli zmniejszenie zakłócenia. Czynnik ten wskazuje, że gdy jest brany pod uwagę przedział czasu rozkładu STFT to nie równa się on widmu gęstości mocy w tym momencie. Davies i Reisman wybrali rozkład STFT oraz rozkład Wignera-Ville, ponieważ były one szeroko stosowane w przeszłości. W ich badaniach STFT najwydatniej pokazuje kompresję widma jako zmęczenie mięśni.

2.3.3. Model autoregrsji

Model autoregresji (AR) szeregów czasowych jest wykorzystywany do badania sygnału EMG. Powierzchnia elektrody wskazuje wzrost aktywności sygnału EMG ze wszystkich aktywnych mięśni w jego pobliżu, a domięśniowe badanie sygnału EMG jest bardzo wrażliwe, ale można uzyskać minimalny wynik tylko z sąsiednich mięśni. Dlatego, aby połączyć wygodę i dokładność zaistniała wielka potrzeba opracowania techniki szacowania domięśniowego EMG i ich właściwości spektroskopowych z pomiaru powierzchni.

W 1975 Graupe i Cline po raz pierwszy przedstawili model autoregresji średniej ruchomej (ARMA) do przedstawienia sygnałów EMG [14]. Uzyskany wynik Graupe i Cline pokazują, że sygnał EMG w wystarczająco krótkich odstępach czasowych można uznać jako stały. Sherif jednak, w 1980 roku, zmienił powyższy model, ponieważ zachowanie elektryczne mięśnia naramiennego było niestałe. Sherif w swojej pracy doktorskiej podkreślił niestały charakter EMG i stosowanego modelu autoregresji [15]. Doerschuk w 1983 roku zauważył problem podobny do tego, który przedstawili Graupe i Cline, a mianowicie kontroli urządzeń protetycznych z sygnałami EMG, przy użyciu modelu autoregresji w przypadku wielu sygnałów EMG [16]. W roku 1986 Zhou zaprezentował pole EMG jako model autoregresji z opóźnionym sygnałem EMG zwany "filtrem tkanki" [17]. Można więc założyć, że dla prototypów domięśniowych i powierzchniowych sygnałów EMG dostępne są parametry modelu szeregu czasowego przekształcające domięśniowe sygnały na sygnały z pola powierzchni. Przedstawiony model jest następnie wykorzystywany do oszacowania sygnału domięśniowe
z sygnału powierzchni. Model ten jest zilustrowany za pomocą rzeczywistych przebiegów EMG. W 1992 Tohru uznał, że precyzyjne wykorzystanie modeli takich jak ARMA lub nie było konieczne dla dynamicznych ruchów mięśni [18]. Koszt obliczeń modelu ARMA jest zbyt wysoki, a określenie rzędu modelu jest skomplikowane i często trudne. Model autoregresji został wybrany przez Tohru, głównie ze względu na jego koszt obliczeniowy, który jest stanowi problemem podczas symulacji.

2.3.4. Inne metody

Oprócz powyżej opisanych metod istnieją również inne, które w skrócie zostaną przedstawione w poniższym podrozdziale.

W 1969 Rosenfalck sformułował matematyczne równanie $g(z) = 96x^3e^{-z} - 90$, które oparte jest na pracach doświadczalnych Ludin'a przeprowadzanych na mięśniach klatki piersiowej [19]. Nandedkar i Stalberg zmienili wyrażenie w 1983 roku z g(z) do e(z) = g(2z), w celu lepszego dopasowania danych eksperymentalnych, co prowadzi do równania $e(z) = 768z^3e^{-2z} - 90$ [20].

Nanderdar i Barkhaus zaproponowali w 1992 roku model, który opiera się na prostej zasadzie sumowaniu wektora [21]. Według Slawnych, Laszlo i teorii Hershler (1990), model Nandedkar'a zakłada, że amplituda MUAP dodaje algebraicznie do generowanego związku działania potencjału mięśni (CAMP) amplitudę. Ponieważ przebiegi MUAP nie występują synchronicznie, założenie to nie jest ważne. Jeśli dwie amplitudy MUAP A1 i A2 są sumowane, to amplituda fali równa się A1 + A2 . Innymi słowy, MUAP przyczynia się do zmniejszenia amplitudy fali CMAP, takie zjawisko nazywa się anulowaniem faz. Suma amplitud wyrażona jest wzorem 3 [21].

$$A_{12}^2 = A_1^2 + A_2^2 - 2A_1A_2\alpha \quad \textbf{(3)}$$

W 1994 roku , Englehart i Parker uznali dwa rodzaje międzyimpulsowego interwału (IPI) za prawdopodobieństwo funkcji gęstości modelu (PDF) [22]. Sekwencja wyładowań jako seria IPI, ocena średniej IPI, odchylenie i prawdopodobieństwo funkcji gęstości (PDF) zostały wykorzystane jako deskryptory aktywności neuronów ruchomych. Gęstość funkcji Gaussa wyrażona jest wzorem 4 [22].

$$f_x(x) = \frac{1}{\sigma_x\sqrt{2\pi}}\exp\frac{(x-\mu_x)^2}{2\sigma_x^2} \quad \textbf{(4)}$$

gdzie μ_x jest średnią, a σ_x^2 jest wariancją.

Funkcja gęstości gamma jest wyrażona wzorem 5 [22].

$$f_x(x) = \frac{1}{\beta\Gamma(\rho)}\left(\frac{x-\alpha}{\beta}\right)^{\rho-1}\exp\left(-\frac{x-\alpha}{\beta}\right) \quad \textbf{(5)}$$

gdzie α jest parametrem lokalnym, β jest parametrem skali, ρ jest parametrem kształtu i Γ () jest funkcją gamma.

Zgodnie z modelem, jeżeli dane są nieustalone to szacunki momentów i prawdopodobieństwo funkcji gęstości neuronowej sekwencji rozładowania są podatne na odchylenia. Niektórymi czynnikami, które mogą mieć wpływ

na stopień ustalenia doświadczalnych danych IPI są czas trwania skurczu, sposób produkcji siły oraz poziom skurczu.

Wyrażenie analityczne dla sygnału potencjału czynnościowego mięśnia jest pochodną użycia integralnego impulsu częstotliwości i modelu modulacji amplitudy (IPFAM) przedstawionego przez Zhang'a w roku 1995 [23]. Model ten składa się z trzech głównych elementów: modulacji amplitudy impulsu (PAM), modulacji częstotliwości impulsu (PFM) oraz systemu liniowego. PAM opisuje związek amplitudy EMG ze zmianami w mocy mięśni, PFM opisuje wahania sygnału EMG, spowodowane zmianą w nerwach wypalania i systemu liniowego, p(t) reprezentuje związek potencjału czynnościowego połączonego z efektem dyspersji propagacji i filtrowania tkanki. W tym modelu, potencjał wzrasta, dopóki wstępnie określona wartość progowa nie zostanie osiągnięta. Zatem model IPFAM zawiera najważniejsze funkcje związane z wytwarzaniem rzeczywistych sygnałów EMG.

W 1995 roku Karlsson i Nystrom przedstawili system czasu rzeczywistego do analizy sygnału EMG [24]. Celem było opracowanie systemu do użytku klinicznego z charakterystyką zwrotnej grafiki, elastyczny dobór parametrów, metody standardowej i elastycznego przetwarzania dodawania. W celu otrzymania reprezentacji czasowo-częstotliwościowej sygnału zaproponowano użycie krótkiego czasu transformacji Fouriera. Główną wadą tej metody jest to, że przyjmuje się iż sygnał jest nieruchomy.

3. Badania Motion Capture i EMG

3.1. Technologia Motion Capture

Motion capture (Mo-cap) jest techniką stosowaną przede wszystkim w filmach i grach komputerowych, polegającą na pobieraniu trójwymiarowych ruchów aktorów i zapisywaniu ich na dysku komputera. Dzięki takiej technice zarejestrowane ruchy postaci można w łatwy sposób odzwierciedlić w postaci animacji. Motion Capture spowodowało wzrost użycia grafiki komputerowej we współczesnym kinie zapewniając naturalne i realistyczne ruchy wygenerowane przez animatora w procesie animacji. Technika Mo-cap może również zostać użyta w celu przechwycenia sygnału elektrycznego wysłanego przez mięśnie podczas ruchu, czyli sygnału EMG. Jest to na przykład bardzo pomocne w dzisiejszej medycynie.

3.2. Plik .C3D

Pliki z rozszerzeniem .C3D są to pliki w binarnym formacie, wykorzystywane w biomechanice, animacji oraz laboratoriach analizy ruchu do rejestrowania danych zsynchronizowanych w 3D. Format C3D został opracowany przez dr Andrew Dainis'a i jest używany od 1987 roku, stosowany powszechnie w National Institutes of Health Biomechanics Laboratory w Bethesda, a także w wielu innych czołowych laboratoriach na świecie zajmujących się biomechaniką.

Format C3D udostępnia następujące funkcje [25]:

1) przechowuje informacje takie jak stosowane kanały EMG czy zestawy markerów,
2) informacje odnoszące się do okoliczności sesji testowej, takich jak częstotliwość próbkowania
3) przechowuje informacje pacjenta - imię i nazwisko, wiek, wagę, długości nóg,
4) wyniki analizy, takie jak synchronizacja cyklu chodu i związanych z nią informacji.
5) rozszerzalność - format C3D zapewnia możliwość przechowywania nowych informacji bez dokonywania zmian w starszych nieaktualnych.

3.3. Rejestrowanie danych

Badanie przy pomocy techniki Motion Capture zostało przeprowadzone w Instytucie Informatyki Politechniki Lubelskiej, w Laboratorium Akwizycji Ruchu i Ergonomii Interfejsów i polegało na wykonaniu kilku czynności ruchowych takich jak:

1) chód,
2) przysiad na jednej nodze,
3) przysiad na obu nogach,
4) skłon o prostych kolanach.

Na rysunku 1 został przedstawiony uczestnik podczas wykonywania ćwiczenia jakim był przysiad na jednej nodze.

Rys. 1. Badanie Motion Capture – Przysiad na jednej nodze

3.4. Analiza sygnału EMG

Uzyskany przy pomocy technologii Motion Capture sygnał EMG został zapisany w formacie .C3D. W tabelach 1. - 4. zostały przedstawione dane jakie udało się pozyskać dla pierwszych dziesięciu klatek z sygnału EMG podczas

wykonywania jednego z ćwiczeń w trakcie przeprowadzonego badania Motion Capture dla:

1) mięśnia prostego lewego uda,
2) mięśnia prostego prawego uda,
3) mięśnia dwugłowego lewego uda,
4) mięśnia dwugłowego prawego uda.

Tabela 1. Dane EMG uzyskane z mięśnia prostego prawego uda

Klatka	Pomiar 1	Pomiar 2	Pomiar 3	Pomiar 4	Pomiar 5
1	0	0	0	0	0
2	0	0	0	0	0
3	0,207	0,193	-0,073	-0,249	-0,183
4	0,256	0,279	-0,003	-0,161	0,064
5	0,209	0,18	-0,053	-0,243	-0,173
6	0,235	0,258	-0,023	-0,163	0,062
7	0,214	0,177	-0,053	-0,231	-0,177
8	0,229	0,258	-0,011	-0,17	0,055
9	0,228	0,195	-0,073	-0,253	-0,17
10	0,224	0,26	0	-0,18	0,055

Tabela 2. Dane EMG uzyskane z mięśnia prostego lewego uda

Klatka	Pomiar 1	Pomiar 2	Pomiar 3	Pomiar 4	Pomiar 5
1	0	0	0	0	0
2	0	0	0	0	0
3	0,2	0,167	-0,042	-0,248	-0,132
4	0,265	0,248	-0,035	-0,21	0,006
5	0,213	0,192	-0,063	-0,223	-0,118
6	0,24	0,261	-0,026	-0,197	0,011
7	0,196	0,188	-0,071	-0,215	-0,121
8	0,261	0,253	-0,025	-0,214	0,009
9	0,188	0,19	-0,066	-0,206	-0,131
10	0,271	0,26	-0,014	-0,232	-0,008

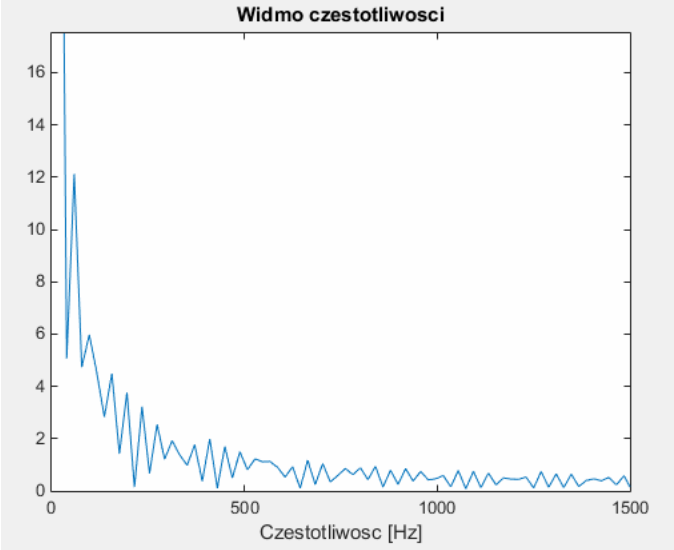
Tabela 3. Dane EMG uzyskane z mięśnia dwugłowego prawego uda

Klatka	Pomiar 1	Pomiar 2	Pomiar 3	Pomiar 4	Pomiar 5
1	0	0	0	0	0
2	0	0	0	0	0
3	0,169	0,163	-0,09	-0,212	-0,135
4	0,248	0,237	-0,026	-0,211	0,038
5	0,171	0,153	-0,09	-0,229	-0,108
6	0,264	0,25	-0,007	-0,202	-0,006
7	0,185	0,174	-0,066	-0,208	-0,115
8	0,286	0,259	-0,013	-0,163	0,033
9	0,177	0,166	-0,076	-0,229	-0,124
10	0,275	0,244	-0,011	-0,18	-0,026

Tabela 4. Dane EMG uzyskane z mięśnia dwugłowego lewego uda

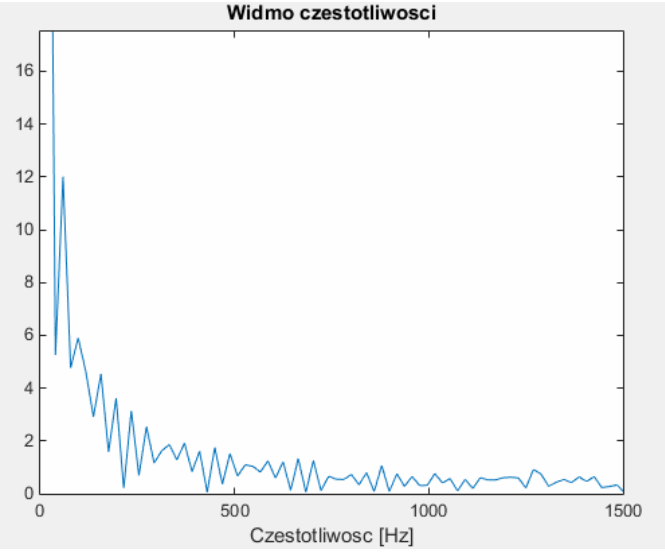
Klatka	Pomiar 1	Pomiar 2	Pomiar 3	Pomiar 4	Pomiar 5
1	0	0	0	0	0
2	0	0	0	0	0
3	0,247	0,193	-0,077	-0,249	-0,17
4	0,217	0,237	-0,027	-0,183	0,062
5	0,27	0,201	-0,071	-0,237	-0,183
6	0,226	0,229	-0,016	-0,163	0,067
7	0,254	0,189	-0,076	-0,229	-0,164
8	0,196	0,257	-0,035	-0,186	0,083
9	0,251	0,157	-0,065	-0,254	-0,182
10	0,2	0,238	-0,021	-0,179	0,059

Uzyskany sygnał EMG został poddany analizie w celu sprawdzenia jego nieliniowości za pomocą transformaty Fourier'a. Na rysunkach 2 – 5 przedstawiono uzyskane widma częstotliwościowe.

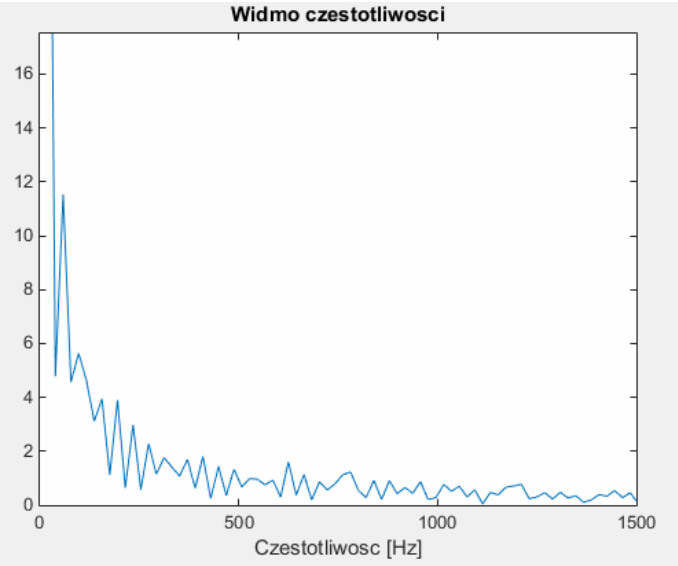


Rys. 2. Widmo częstotliwościowe dla badanego mięśnia prostego prawego uda

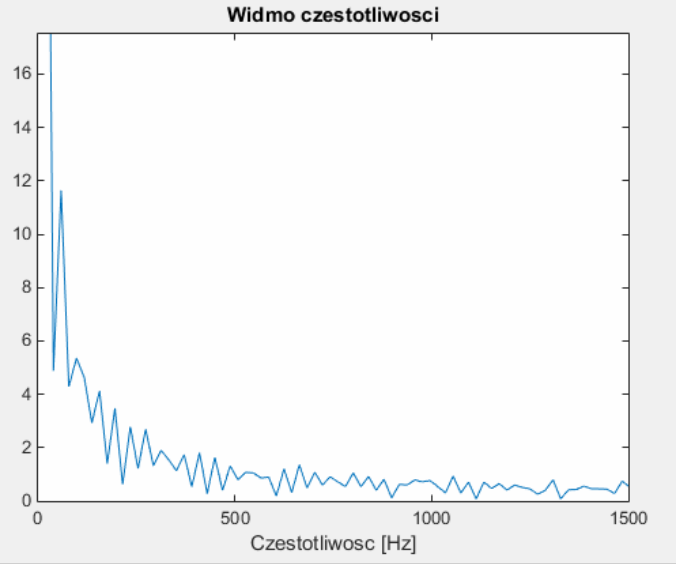
Dzięki przeanalizowaniu otrzymanych wyników z pomiaru danych EMG oraz utworzonych wykresów widma częstotliwościowego dla mięśni prostych prawego oraz lewego uda można zauważyć, iż oba te mięśnie pracowały podobnie aczkolwiek wraz ze wzrostem częstotliwości ich widma zaczynają się różnić. Różnica spowodowana jest tym, że uczestnik badania ma bardziej umięśnione prawe udo niż lewe.



Rys. 3. Widmo częstotliwościowe dla badanego mięśnia prostego lewego uda



Rys. 4. Widmo częstotliwościowe dla badanego mięśnia dwugłowego prawego uda



Rys. 4. Widmo częstotliwościowe dla badanego mięśnia dwugłowego lewego uda

Analiza widm częstotliwościowych dla mięśni dwugłowych prawego oraz lewego uda potwierdziła tylko, że badany uczestnik posiada większą siłę w prawej nodze.

4. Podsumowanie

Na przestrzeni kilkudziesięciu lat naukowcy coraz częściej przyglądają się i badają impulsy elektryczne wytwarzane w trakcie aktywności mięśni. Jak każde badanie również i te opisywane w niniejszej pracy są obarczone ryzykiem niepowodzenia. Jako niepowodzenie można rozumieć otrzymanie zanieczyszczonego sygnału. Jednakże, metody opisane w rozdziale drugim ukazują, że świat nauki już od bardzo dawna stara się w lepszy lub w gorszy sposób wyeliminować ryzyko niepowodzenia badania stosując różne sposoby zaczynając od zmiany częstotliwości badanego sygnału a kończąc na stosowaniu statystyk wyższego rzędu. Sposoby różnią się między sobą, ale dzięki tym badaniom świat dowiaduje się coraz to nowszych rzeczy na temat działania ludzkiego organizmu. Niniejszy artykuł przybliżył wartość jaką niesie ze sobą sygnał EMG i zawarte w nim informację na temat działania ludzkiego mięśnia. Dzięki uzyskaniu sygnału EMG przy pomocy coraz bardziej popularnej technologii Motion Capture, a następnie przetworzeniu go przy pomocy zaimplementowanej aplikacji stosując transformatę Fouriera, została przeprowadzona analiza badawcza dzięki, której możliwe stało się stwierdzenie jak reagują dane mięśnie na bodźce wytworzone poprzez wykonywanie ćwiczeń. Można stwierdzić, które mięśnie są bardziej rozwinięte i mocniej pracują.

Literatura

[1] Basmajian J. V., De Luca C. J.: Muscles Alive: Their Functions Revealed by Electromyography, 1985

[2] Guglielminotti P., Merletti R.: Effect of electrode location on surface myoelectric signal variables: a simulation study, 1992.

[3] Laterza F., Olmo G.: Analysis of EMG signals by means of the matched wavelet transform, 1997.

[4] Gabor D.: Theory of communication,1946.

[5] Ismail A.R., Asfour S.S.: Continuous wavelet transform application to EMG signals during human gait, 1998.

[6] Pattichis C.S., Pattichis M.S.: Time-scale analysis of motor unit action potentials, 1999.

[7] Kumar D.K., Pah N.D., Bradley A.: Wavelet analysis of surface electromyography to determine muscle fatigue, 2003.

[8] Piper H.: Electrophysillogie Muschliche Muskeln, 1912.

[9] Cohen L.: Time-frequency analysis. Englewood Cliffs, 1995.

[10] Syeed A.M, Jones D.L.: Optimal kernel for nonstationary spectral estimation, 1995.

[11] Ricamato A.L., Absher R.G., Moffroid M.T., Tranowski J.P.: A time-frequency approach to evaluate electromyographic recordings, 1992.

[12] Davies M.R., Reisman S.S.: Time frequency analysis of the electromyogram during fatigue, 1994.

[13] Amin M., Cohen L., Williams W.J.: Methods and Applications for Time Frequency Analysis,1993.

[14] Graupe D., Cline W.K.: Functional Separation of EMG signals via ARMA identification. 1975.

[15] Sherif M.H.: Stochastic Model of Myoelectric Signals for Movement Pattern Recognition in Upper Limb Prostheses, 1980.

[16] Doerschuk P.C., Gustafson D.E., Willsky A.S.: Upper extremity limb function discrimination using EMG signal analysis,1983.

[17] Zhou Y., Chellappa R., Bekey G.: Estimation of intramuscular EMG signals from surface EMG signal analysis,1986.

[18] Kiryu T., Saitoh Y., Ishioka K.: Investigation on parametric analysis of dynamic EMG signals by a muscle-structured simulation model, 1992.

[19] Rosenfalck P.: Intra- and extracellular potential fields of active nerve and muscle fibres. A physico-mathematical analysis of different models,1969.

[20] Nandedkar S.D., Stålberg E.: Simulation of single muscle fibre action potentials,1983.

[21] Nandedkar S.D, Barkhaus P.E.: Phase interaction in the compound muscle action potential: application to motor unit estimates, 1992.

[22] Englehart K.B., Parker P.A.: Single motor unit myoelectric signal analysis with nonstationary data,1994.

[23] Zhang Y.T, Herzog W., Liu M.M.: A mathematical model of myoelectric signals obtained during locomotion, 1995.

[24] Karlsson S., Nystrom L.: Real-time system for EMG signal analysis of static and dynamic contractions, 1995.

[25] https://www.c3d.org/

JCSI 3 (2017) 46-53
Journal of Computer Sciences Institute

Wysłane: 2016-11-30
Przyjęte: 2016-12-15

Metody optymalizacji wydajności silnika Unity 3D w oparciu o grę z widokiem perspektywy trzeciej osoby

Krzysztof Siarkowski*, Przemysław Sprawka*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Optymalizacja gier to jeden z najważniejszych aspektów ich tworzenia. Artykuł opisuje metody optymalizacji silnika Unity, a jako przedmiot analizy posłużyła gra z widokiem perspektywy trzeciej osoby. Zbadano jaki wpływ na wydajność gry mają metody, które polegają na odciążeniu karty graficznej, poprzez zwiększenie wykorzystania procesora oraz pamięci.

Słowa kluczowe: optymalizacja; silnik gier; Unity

*Autor do korespondencji.

Adresy e-mail: krzsiarkowski@gmail.com, przemek1992spr@gmail.com

Methods for optimizing the performance of Unity 3D game engine based on third-person perspective game

Krzysztof Siarkowski*, Przemysław Sprawka*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Game optimization is one of the most important aspects of their creation. The article describes methods to optimize Unity Engine using third person perspective game as an example. Various methods that rely on offloading graphics card, by increasing the use of CPU and memory were used in order to check how the game performance changes.

Keywords: optimization; game engine; Unity

*Corresponding author.

E-mail addresses: krzsiarkowski@gmail.com, przemek1992spr@gmail.com

1. Wstęp

W ostatnich latach pojawiło się na rynku wiele darmowych silników służących do tworzenia gier, co znacznie obniżyło barierę wejścia do środowiska twórców gier osobom, dla których komercyjne silniki były zbyt drogie. Jednakże cechy i jakość produktu finalnego, jaką oczekują od gier użytkownicy rośnie z każdym dniem. Należy oczekiwać, że każdy aspekt gry zostanie szczegółowo przeanalizowany zarówno przez graczy jak i krytyków, dlatego nie można pozwolić sobie na problemy związane z wydajnością.

Postanowiono więc wyjść naprzeciw tym problemom i w niniejszym artykule opisać zastosowanie i analizę działania wybranych metod optymalizacji. Zbadano wpływ tych metod na wykorzystanie zasobów komputera takich jak procesor i pamięć oraz głównych statystyk odpowiedzialnych za pomiar wydajności renderowania, zaś jako przedmiot analizy posłużyła gra utworzona w Unity z perspektywy trzeciej osoby. Postarano się udowodnić, że użycie metod optymalizacji odciążających jeden z zasobów komputera odpowiedzialnych za wydajność, powoduje większe wykorzystanie innego.

Wybór Unity był spowodowany tym, że silnik ten ostatnio zyskuje sporą popularność oraz posiada wbudowane narzędzia do analizowania statystyk podczas renderowania. Dodatkowo z poziomu interfejsu graficznego edytora Unity, można w prosty sposób użyć danej metody optymalizacji i skonfigurować jej parametry.

2. Renderowanie

Renderowaniem nazywamy proces zamiany informacji wejściowych na inną formę reprezentacji tych danych, adekwatną do danego medium. W grafice trójwymiarowej renderowanie to proces konwertowania trójwymiarowego, matematycznego modelu obiektu na wyświetlany obraz dwuwymiarowy w postaci pojedynczej klatki, bądź animacji [1]. W skład procesu renderowania 3D wchodzą różne elementy, takie jak kąt padania światła, fizyczne właściwości materiałów na obiekcie, odbicia oraz cienie. Dodawanie takich cech wizualnych sprawia, że modele 3D stają się bardziej realistyczne.

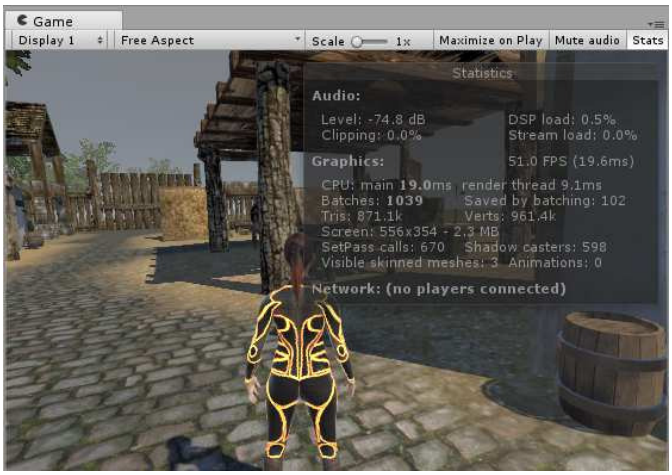
Renderowanie w czasie rzeczywistym polega na generowaniu obrazów wystarczająco szybko, aby stworzyć wrażenie płynnej animacji. Wykorzystywane jest szczególnie w grach, gdzie wszystkie modele 3D i elementy scen muszą być przedstawiane użytkownikom w określonej liczbie klatek na sekundę. Wpływa to na jakość przedstawianych modeli ponieważ, muszą być przygotowane tak, aby jak najwierniej odzwierciedlały obiekty, które mają przedstawiać, ale jednocześnie składały się z jak najmniejszej liczby wielokątów (ang. *polygon*) bez utraty swojej formy. Najczęstszym błędem przy badaniu liczby klatek na sekundę

wyświetlanych na ekranie jest przekonanie, że głównym czynnikiem na nie wpływającym jest złożoność geometrii, czyli złożoność siatek (ang. *meshes*) modeli 3D. W rzeczywistości wpływ na czas renderowania jednej klatki ma liczba świateł na scenie, złożoność materiałów, ilość pamięci zajętej przez tekstury i siatki, fizyka, animacje, cząsteczki i inne pomniejsze elementy [2].

3. Narzędzia

Unity posiada zaimplementowane w sobie dwa główne narzędzia do optymalizacji wydajności. Pierwszym z nich jest okno statystyk renderowania, zaś drugim Profiler, z którego można korzystać na licencji darmowej, wraz z wydaniem Unity 5 i udostępnieniem w niej prawie wszystkich funkcji, które wcześniej były dostępne tylko w płatnej wersji Pro.

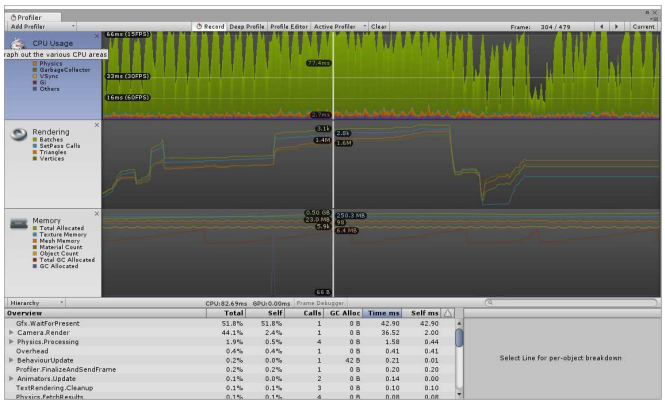
3.1. Rendering Statistics Window



Rys. 1. Okno statystyk renderowania

Statystyki renderowania widoczne są w oknie (Rys. 1), które ukazuje się po kliknięciu przycisku *Stats* w sekcji *Game View*. Statystyki te pokazywane są w czasie rzeczywistym i są bardzo przydatne do analizy oraz optymalizacji wydajności.

3.2. Rendering Profiler



Rys. 2. Okno Profilera

W górnej części okna Profilera wyświetlane są dane dotyczące wydajności, w miarę upływu czasu (Rys. 2). Po uruchomieniu gry, dane zapisywane są co każdą klatkę, a wyświetlana jest jedynie historia kilkuset ostatnich klatek. Klikając na pojedynczą klatkę, zostaną wyświetlone jej szczegóły w dolnej części okna. Różne informacje zostaną zaprezentowane w zależności, która sekcja osi czasu została wybrana. Oś czasu zawiera kilka obszarów: wykorzystanie procesora, renderowanie i użycie pamięci.

3.3. Statystyki renderowania

Statystyki mające największy wpływ na wydajność renderowania, których wartości można odczytać zarówno z okna statystyk renderowania oraz Profilera to:

Time per frame and FPS – liczba klatek na sekundę (ang. *frames per second*) oraz czas w milisekundach potrzebny w Unity do przetworzenia i renderowania jednej klatki. Klatką nazywamy finalny render złożony z pikseli przekazany z karty graficznej i wyświetlony na ekranie. Podczas przygotowywania gotowej klatki, zarówno procesor jak i karta graficzna muszą wykonać wiele obliczeń. To wszystko odzwierciedla złożoność renderowania i czas potrzebny do jego wykonania. Dłuższy czas renderowania może skutkować spadkiem FPS, ponieważ mniej klatek zostanie przetworzonych w ciągu sekundy. Ogólnie rzecz biorąc, im wyższa ich liczba, tym lepiej. Liczba klatek nie powinna być mniejsza niż 15-30 na sekundę, ponieważ spadek poniżej tej wartości powodują zauważalne dla ludzkiego oka zburzenia płynności oraz zawieszanie się ekranu.

SetPass calls – liczba odwołań procesora do karty graficznej (ang. *draw calls*). Odwołania te odnoszą się do liczby wysłanych żądań do karty graficznej, w celu pobrania danych z pośredniczących buforów, a następnie dane te utworzą ostateczny render. Dzięki temu obiekty są widoczne na ekranie, w momencie, w którym powinny. Zwykle wykonuje się jedno odwołanie na każdy renderowany obiekt. W przeciwieństwie do liczby klatek na sekundę, im mniej *draw calls* tym lepiej, ponieważ każde odwołanie pociąga za sobą znaczne zapotrzebowanie i obciążenie procesora. Oznacza to, że odwołania do karty graficznej są bezpośrednio związane z wydajnością, którą można poprawić poprzez redukcję ich liczby. W osiągnięciu tego celu, Unity posiada wbudowany system pakietowania (ang. *batching*), pozwalający grupować odwołania i przetwarzać kilka obiektów razem w jednym odwołaniu, zamiast w wielu [3].

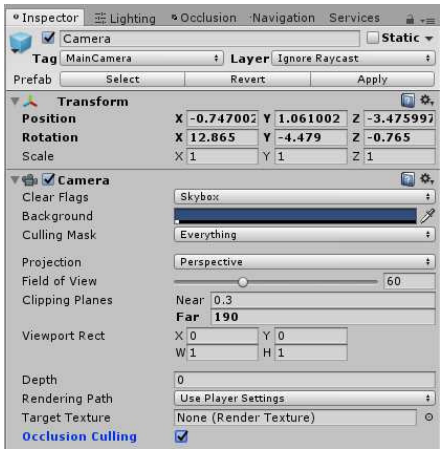
Tris and Verts – określa łączną liczbę trójkątów i wierzchołków obiektów aktualnie renderowanych, a nie wszystkich znajdujących się na scenie. W momencie przesuwania się kamery i obiektów, ta liczba zmienia się, ponieważ obiekty wychodzą poza obszar widzenia kamery, a niektóre w tym samym momencie wkraczają. Współczesny sprzęt komputerowy i systemy renderowania są w stanie bez problemu poradzić sobie z milionami trójkątów i wierzchołków, jednak nadal obowiązuje zasada, że utrzymanie jak najmniejszej ich liczby wpływa na wydajność, ponieważ mniej danych należy przetworzyć.

4. Metody optymalizacji

W rozdziale tym szczegółowo opisano działanie metod optymalizacyjnych, które zostały użyte w grze wykonanej w Unity, w celu przeanalizowania, na jakie parametry renderowania wybrane metody mają największy wpływ.

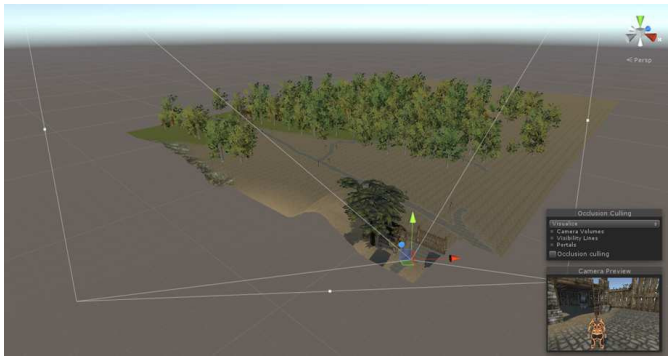
4.1. Occlusion Culling

Usuwanie niewidocznych powierzchni (ang. *occlusion culling*) pozwala uniknąć renderowania obiektów, które są zasłaniane przez inne. Nie jest to domyślna funkcja w grafice trójwymiarowej, ponieważ zazwyczaj najpierw renderowane są obiekty, które znajdują się najdalej od kamery, a dopiero na nich te bliżej.



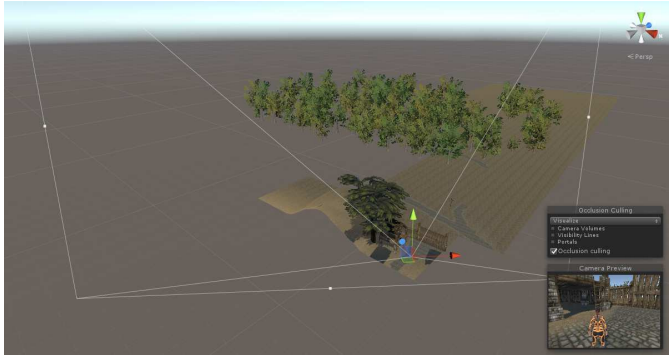
Rys. 3. Okno Inspektora obiektu kamery z zaznaczoną opcją Occlusion Culling

Usuwanie obiektów znajdujących się za innymi obiektami, nie jest tym samym co usuwanie obiektów, które wychodzą poza obszar widzenia kamery (ang. *frustum culling*). *Frustrum culling* jest wykonywane automatycznie, zaś usuwanie zasłoniętych obiektów musi zostać włączone w edytorze Unity w oknie Inspektora obiektu kamery (Rys. 3) i poprawnie skonfigurowane.



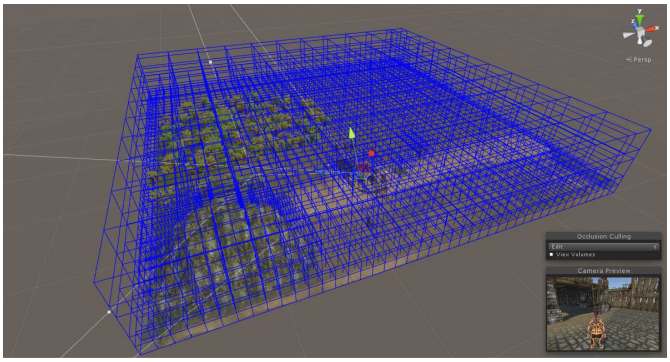
Rys. 4. Widok sceny z wyłączoną opcją usuwania zasłoniętych obiektów

Usuwanie obiektów wychodzących poza obszar kamery przedstawia rysunek 4, zaś połączenie tych dwóch technik można zaobserwować na rysunku 5, gdzie doskonale widać różnice pomiędzy nimi.



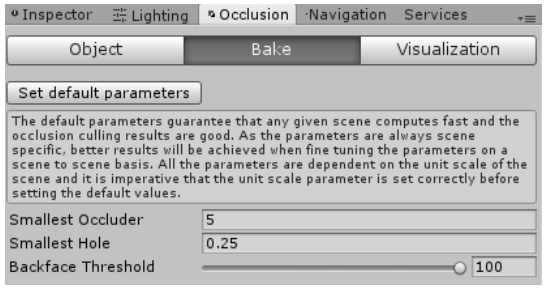
Rys. 5. Widok sceny z włączoną opcją usuwania zasłoniętych obiektów

Przy użyciu wirtualnej kamery, budowana jest hierarchia potencjalnie widzianych obiektów. Dane te następnie są wykorzystywane przez wszystkie kamery na scenie do sprawdzenia, które obiekty powinny zostać wyrenderowane, a które potraktowane jako zasłonięte przez inne obiekty [4].



Rys. 6. Podział sceny na komórki

Dane wykorzystywane do usuwania obiektów składają się z komórek, które zostają wydzielone z głównego obszaru obejmującego całą scenę. Komórki te tworzą tzw. siatkę ukrywania obiektów (Rys. 6). Wszelkie obiekty mniejsze niż wielkość komórki nie spowodują zasłonięcia innych obiektów, zaś obiekty większe niż komórka będą zasłaniać obiekty za sobą.



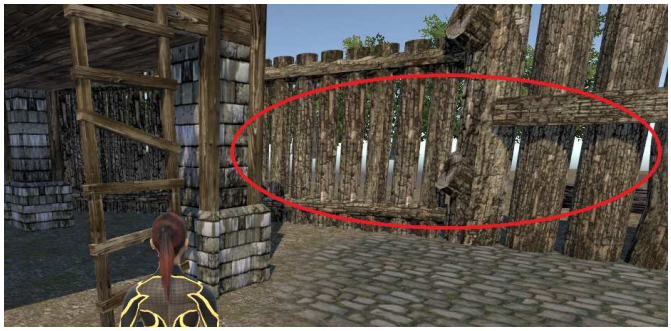
Rys. 7. Okno parametryzownia metody Occlusion Culling

Wielkość najmniejszego obiektu (ang. *smallest occluder*), odpowiada najmniejszej wielkości komórki, na jaki może zostać podzielona scena. Parametr ten jak i drugi z parametrów *occlusion culling*, jakim jest najmniejszy dopuszczalny prześwit pomiędzy siatkami obiektów, przez który obiekty powinny być renderowane (ang. *smallest hole*) można ustawić w oknie Occlusion widocznym na rysunku 7.

Rys. 8. Ustawienie kamery dla zbadania działania occlusion culling

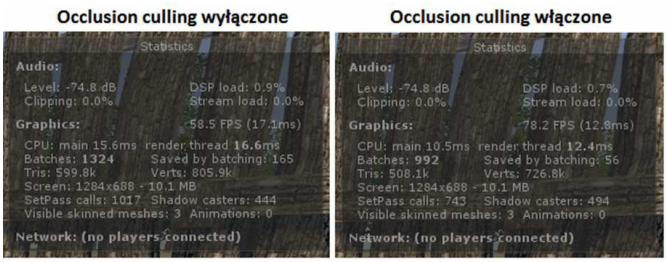
Przesuwając po scenie kamerę z włączoną wizualizacją usuwania niewidocznych powierzchni, wybrano najbardziej odpowiednie miejsce do przeprowadzenia badania wpływu na renderowanie tej metody optymalizacji. Duża część sceny została zasłonięta przez wzgórze, a dodatkowo obszar widzenia kamery obejmował prześwit pomiędzy ogrodzeniem wioski (Rys. 8).

Parametr *smallest hole* ustawiony na zbyt wysoką wartość spowodował, że włączenie opcji usuwania niewidocznych obiektów, potraktowało teren za ogrodzeniem oraz zbyt dużo drzew za zasłonięte, co zostało ukazane na rysunku 9.



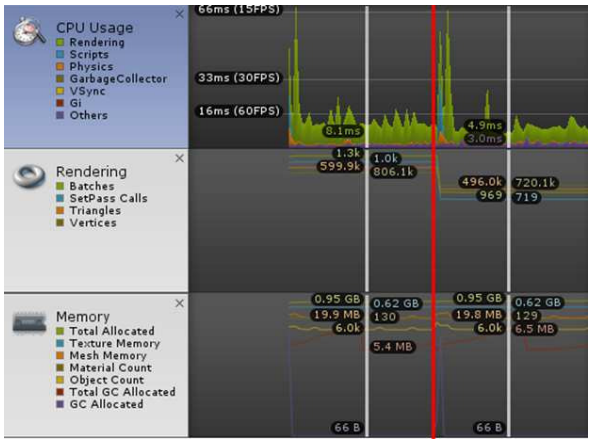
Rys. 9. Efekt zbyt dużego dopuszczalnego prześwitu między siatkami obiektów, przez który obiekty traktowane są jako zasłonięte

Wyniki



Rys. 10. Okno statystyk renderowania – Occlusion Culling

Włączenie usuwania niewidocznych powierzchni, zmniejszyło liczbę trójkątów i wierzchołków, ponieważ zasłonięte obiekty nie zostały wyrenderowane. Pozwoliło to zaoszczędzić dużą liczbę odwołań do karty graficznej, co pozytywnie odbiło się na wzroście liczby wyświetlanych klatek na sekundę (Rys. 10).



Rys. 11. Okno Profilera – Occlusion Culling

Na rysunku 11 widać linię czasu z Profilera podczas renderowania sceny z wyłączonym i włączonym *occlusion culling* – czerwona linia oznacza moment włączenia usuwania niewidocznych powierzchni. W wyraźniejszy sposób został przedstawiony spadek trójkątów i wierzchołków oraz liczby odwołań, co nie wpłynęło negatywnie na zużycie pamięci lub procesora.

4.2. Clipping Plane

Płaszczyzny odcinające (ang. *clip plane*) pozwalają zmniejszyć zasięg renderowania obiektów. Działają w ten sposób, że obiekty, które znajdują się w większej odległości od kamery niż określono, nie będą wyświetlane [5].

Rys. 12. Widok z gry z domyślnym zasięgiem renderowania

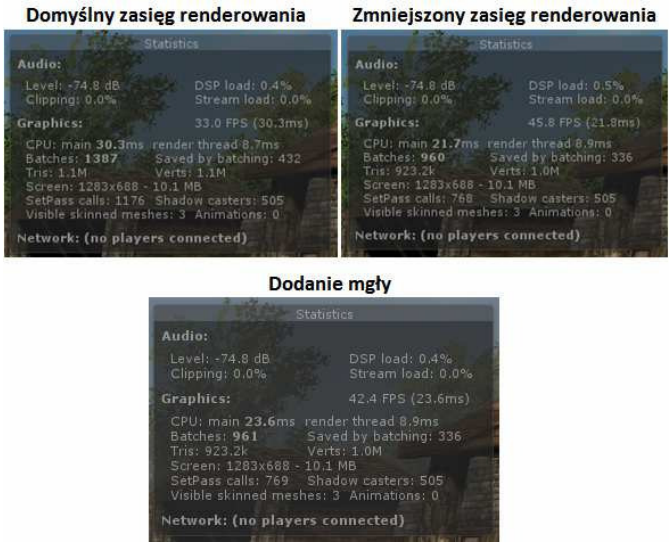
Za zmniejszenie wielkości obszaru, który powinien być renderowany odpowiada parametr *Far Clip Plane* (ustawienie tego parametru jest możliwe w oknie Inspektora kamery widocznego na rysunku 3). Widok z gry, z domyślną wartością tego parametry został przedstawiony na rysunku 12. Zmniejszenie wartości parametru spowoduje, że większa liczba obiektów nie będzie renderowana, ponieważ ograniczono obszar widoczności kamery. Powstała wolna przestrzeń (Rys. 13), która w miarę zbliżania się postaci, będzie wypełniana odciętymi wcześniej obiektami.

Rys. 13. Efekt działania płaszczyzny odcinającej

Ten niepożądany efekt można za to w łatwy sposób ukryć, używając mgły. Nie dość, że odcięte obiekty zostaną zasłonięte przez mgłę, to równocześnie zwiększą się wrażenia towarzyszące graczowi zwiedzającemu wirtualny świat (Rys. 14).

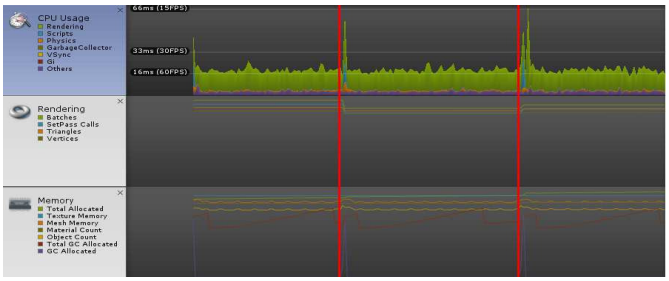
Rys. 14. Widok z gry z użyciem płaszczyzny odcinającej i mgły

Wyniki



Rys. 15. Okno statystyk renderowania – Far Clip Plane

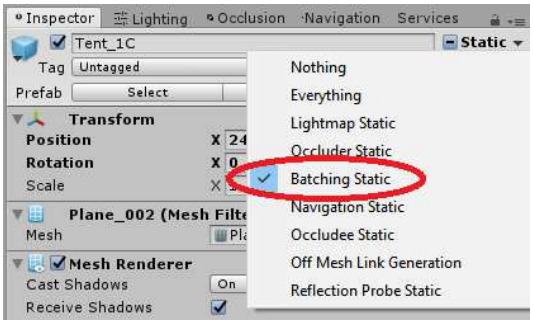
Na rysunku 15 przedstawiono statystyki renderowania w poszczególnych etapach konfigurowania na scenie płaszczyzny odcinającej oraz po dodaniu na scenę mgły. W oknie Profilera (Rys. 16), który również w tym czasie rejestrował dane, przejście do kolejnego etapu zostało zaznaczone na osi czasu czerwoną linią. Zmniejszenie zasięgu usunęło z renderowania kilka obiektów, zatem zmalała liczba trójkątów i wierzchołków, odwołań procesora do karty graficznej renderowania, co przyczyniło się do podniesienia liczby klatek wyświetlanych na sekundę. Analizując oś czasu przedstawiającą użycie procesora, można zauważyć, że kolejne etapy w momencie uruchomienia gry, chwilowo ale coraz bardziej i bardziej obciążają CPU. Może to być spowodowane tym, że procesor musi nie tylko obliczyć odległość obiektów od kamery, ale również je usunąć, żeby nie zostały wyrenderowane, a w ostatnim etapie dodatkowo wykonać skrypt odpowiedzialny za wyświetlenie mgły.



Rys. 16. Okno Profilera – Far Clip Plane

4.3. Batching

Wyróżniamy dwa rodzaje pakietowania: statyczne (ang. *static batching*) oraz dynamiczne (ang. *dynamic batching*) i obie metody mają wpływ na grupowanie obiektów w celu zmniejszenia liczby odwołań procesora do karty graficznej. Pakietowanie przede wszystkim pozwala Unity grupować wiele obiektów razem, traktując je jakby były jedną siatką lub zasobem, więc mogą być przetworzone w pojedynczym odwołaniu, zamiast w wielu [6]. *Batching* jest w głównej mierze procesem wewnętrznym i automatycznym w Unity, czyli to edytor podejmuje decyzję na własną rękę, jak zgrupować obiekty i kiedy. Decyzje te podejmowane są na podstawie ustawień obiektów w oknie Inspektora. Wymusza to pewne wymagania dotyczące przygotowanie przez nas obiektów do pakietowania. Dzięki temu mamy pewien poziom kontroli nad tym jak zadziała pakietowanie.



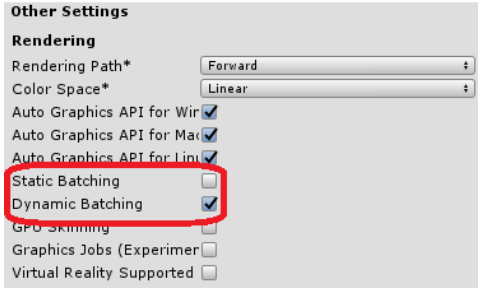
Rys. 17. Oznaczenie obiektu jako statyczny

Dla siatek modeli obiektów takich jak ściany, krzesła, wzgórze, góry, latarnie i innych nieporuszających się w czasie działania gry, należy zawsze oznaczać obiekty jako statyczne (Rys. 17). Jest to sygnał dla Unity, że siatki te mogą zostać

zgrupowane w jedną wielką siatkę, co edytor zrobi już automatycznie. Grupowanie obiektów pomiędzy pakietami zależy od różnych ustawień obiektów. Wszystkie obiekty dzielące ten sam materiał zostaną wysłane w jednym pakiecie, dlatego najczęściej wiele obiektów znajduje się w tylu grupach, z ilu korzystają materiałów [7].

Pakietowanie dynamiczne jest procesem grupowania, który wykonywany jest przez Unity automatycznie, na obiektach niebędących statycznymi, żeby jeszcze bardziej zmniejszyć liczbę *draw calls*. Wszystkie siatki o małej złożoności, które dzielą ten sam materiał i skalę zostaną połączone w pakiet, z wyjątkiem obiektów na które oddziałuje światło w czasie rzeczywistym [8].

Domyślnie w Unity włączona jest opcja pakietowania dynamicznego, lecz zarówno *static batching* jak jak i *dynamic batching* mogą być włączone lub wyłączone jednocześnie. Opcje te dostępne są w oknie Player Settings, widocznym na rysunku 18.



Rys. 18. Okno Player Settings, gdzie możliwe jest włączenie/wyłączenie pakietowania

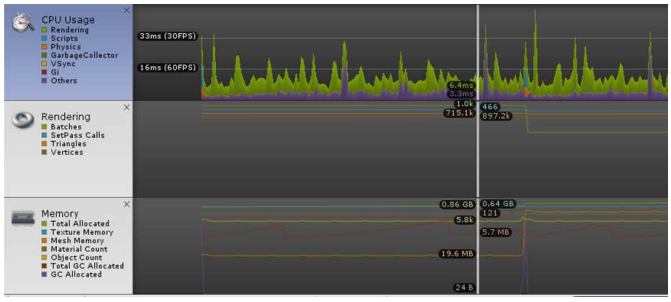
Wyniki

Rys. 19. Widok z gry użyty do zbadania działania batchingu

Do przetestowania działania poszczególnych mechanizmów pakietowania wykorzystano widok z gry, który został przedstawiony na rysunku 19.

Rys. 20. Statystyki renderowania, używając różnych metod pakietowania

Jak można zauważyć na zestawionych ze sobą oknach renderowania statystyk (Rys. 20), w zależności, który mechanizm pakietowania został włączony, znacząco różni się liczba grup. Włączając kolejno metody pakietowania, inne siatki są ze sobą grupowane, co powoduje że liczba tych grup wzrasta. Nie miało to jednak zbyt dużego wpływu na liczbę wierzchołków, trójkątów, czy klatek na sekundę, ponieważ wahały się one w jednakowych granicach przez cały proces. Wydawałoby się w takim razie, że pakietowanie nie wpłynęło na wydajność naszej gry, lecz zanim to stwierdzimy, spójrzmy co możemy zaobserwować w Profilerze.



Rys. 21. Okno Profilera – pakietowanie

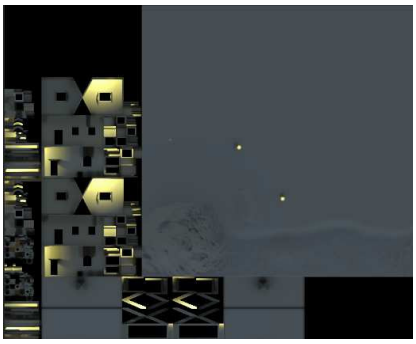
Na rysunku 21 zaznaczony został moment włączenia statycznego pakietowania. Pakietowaniu statycznemu poddanych zostało dużo więcej siatek obiektów niż pakietowaniu dynamicznemu, co doprowadziło do dwukrotnego wzrostu zużycia pamięci. Również doprowadziło to do tego, że podczas uruchomienia gry ze statycznym pakietowaniem chwilowy skok zużycia procesora, większy niż w przypadku samego pakietowania dynamicznego.

4.4. Lighting

Oświetlenie w Unity może być renderowane w czasie rzeczywistym (ang. *real-time lighting*) lub wypalone na mapach światła (ang. *baked lightmaps*). Przy dodawaniu światła na scenę, domyślnie jest ono renderowane w czasie rzeczywistym, czyli aktualizowane co klatkę. W ten sposób

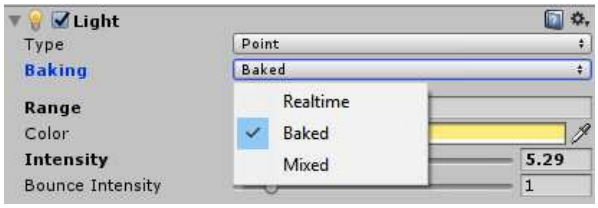
możemy oświetlać modele w ruchu oraz przemieszczać źródła światła. Niestety takie światła nie odbijają się od powierzchni obiektów, a więc nie można uzyskać bardziej realistycznych efektów, takich jak oświetlenie globalne (ang. *global illumination*), a cienie, które generują są czarne. Powinniśmy jednak unikać tego rodzaju renderowania oświetlenia, ponieważ negatywnie wpływa to na wydajność [9].

Oświetlenie wypalone (ang. *baked lighting*) sprawdza się świetnie, kiedy na scenie znajdują się nieruchome obiekty oraz gdy zależy nam na optymalizacji oświetlenia. Światła są obliczane bezpośrednio na scenie i wypalane na teksturach modeli w taki sposób, aby stworzyć efekt padającego światła. Wypalanie mapy światła (ang. *lightmap*) często jest procesem długotrwałym i głównie zależy od prędkości komputera.



Rys. 22. Tekstura lightmapy wygenerowana podczas wypalania oświetlenia

Mapa światła zawiera dane na temat jasności punktów z procesu wypalenia oświetlenia. Generowana jest jeden raz, a później tylko nakładana na teren. *Lightmapy* mogą posiadać w sobie światło bezpośrednie lub pośrednie, czyli takie które odbija się od innej powierzchni i pada na obiekt. Podczas gdy aplikacja jest uruchomiona, światła te nie mogą być zmienione, natomiast mogą na nie wpływać światła w czasie rzeczywistym i dodatkowo dodawać do nich cienie [10]. Mapa światła wygenerowana w edytorze Unity dla sceny w grze została przedstawiona na rysunku 22.



Rys. 23. Ustawianie sposobu renderowania światła

Sposób renderowania światła może zostać zmieniony w oknie Inspektora obiektu światła (Rys. 23). Odpowiada za to parametr Baking, w którym mamy możliwość ustawienia, czy wybrane oświetlenie ma być renderowane w czasie rzeczywistym czy wypalone. Sam proces wypalania lightmap można uruchomić z poziomu okna Lighting, w którym znajdują się opcje konfiguracyjne oświetlenia na całej scenie.

Wyniki

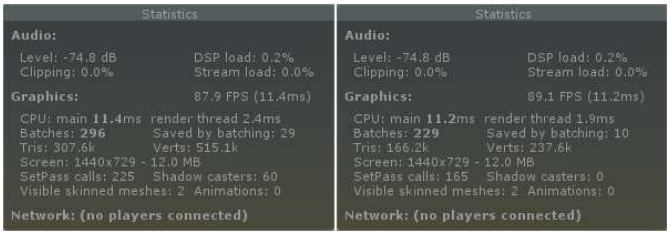
Renderowanie światła w czasie rzeczywistym i wypalanie oświetlenia, zbadano wykorzystując stworzony w programie graficznym Blender model domku oraz latarni, ze światłem punktowym.

Rys. 24. Real-time lighting

Ściana budynku w przykładzie z *real-time* ma wyraźnie podkreślone detale, a kostki brukowe posiadają zacieniowane miejsca (Rys. 24). Również światło renderowane w czasie rzeczywistym, przy tych samych ustawieniach natężenia i obszaru jest znacznie ostrzejsze.

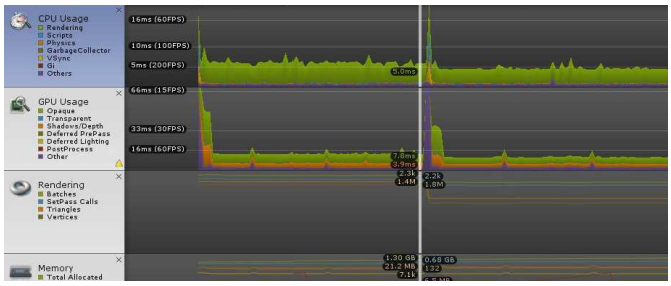
Rys. 25. Baked lighting

Jednym z mankamentów wypalania świateł jest to, że nie oświetlają obiektów dynamicznych takich jak postacie (Rys. 25). Jednak wydajnościowo znacznie lepiej one wypadają. Przy oświetleniu w czasie rzeczywistym, każdy obiekt na który pada światło, wymaga wyrenderowania go tyle razy, ile świateł na niego pada. Powoduje to mnożenie wyświetlanej geometrii (liczba trójkątów i wierzchołków), co wyraźnie widać na zrzutach ekranu z okna statystyk renderowania (Rys. 26).



Rys. 26. Okna statystyk renderowania. Po lewej: oświetlenie w czasie rzeczywistym. Po prawej: oświetlenie wypalone.

Wypalane mapy traktują lightmapy jak tekstury i nie przeliczają świateł padających na poszczególne piksele. Również znacznie zmniejsza się liczba odwołań procesora, a dodatkowo nie przeliczane są cienie, które w czasie rzeczywistym znacząco obciążają grę.



Rys. 27. Okno Profilera w real-time i baked lighting

Oświetlenie renderowane co klatkę obciąża procesor dużo bardziej, niż oświetlenie wypalone na teksturach, co potwierdza odczyt z osi czasu (Rys. 27), gdzie pionowa linia wskazuje moment, w którym zatrzymano grę i Profiler przestał zbierać dane dotyczące renderowania światła w czasie rzeczywistym. W edytorze została wypalona *lightmapa* i ponownie uruchomiono grę, co wznowiło rejestrowanie danych przez Profiler, wykorzystując do renderowania drugą z metod odwzorowania światła.

5. Wnioski

Przeanalizowanie działania wybranych metod optymalizacji uświadomiło nas, że proces optymalizacji wydajności silnika gry, wymaga dużej wiedzy na temat działania najważniejszych funkcji silnika, zanim ktokolwiek zacznie z niego korzystać. Funkcje te niewłaściwie użyte, zamiast zwiększyć wydajność aplikacji, mogą przynieść nieoczekiwane konsekwencje.

Zbadanie wpływu wybranych metod optymalizacji na wydajność wykazało, że w przypadku analizowanej gry, uzyskiwany duży spadek liczby trójkątów i wierzchołków, odwołań procesora do karty graficznej, nie wiązał się z oczekiwanym dużym skokiem wydajności pod względem liczby klatek na sekundę, ponieważ prawie zawsze odbywało się to kosztem zwiększenia zużycia pamięci i mocy obliczeniowej procesora. Wynikało to z próby przeniesienia części obliczeń wykonywanych przez cały czas trwania gry, na moment jej uruchomienia lub dokonanie wyliczeń już na scenie i późniejsze odwoływanie się do nich, tak jak to działo się w przypadku wypalonego oświetlenia,.

Należy więc zachować proporcje pomiędzy optymalizacją liczby klatek na sekundę, a zużyciem pamięci i procesora, żeby odciążając kartę graficzną nie doprowadzić do przeciążenia innych podzespołów komputera, które również składają się na wydajność aplikacji.

Zatem nie powinno się błędnie zakładać, że proces optymalizacji głównie skupia się na zmniejszeniu geometrii sceny, ponieważ w przeprowadzonych metodach optymalizacji udowodniono, że bardzo ważny wpływ na wydajność, oprócz karty graficznej mają również pamięć oraz procesor i nie uzyskamy poprawy liczby klatek na sekundę, co można było zauważyć przy pakietowaniu statycznym.

Literatura

[1] M. F. Shiratuddin, W. Thabet, Utilizing a 3D game engine to develop a virtual design review system, Journal of Information Technology in Construction – ITcon, 16, 2011, 39-68.

[2] S. Blackman, Beginning 3D Development with Unity 4. All-in-One, Multi-Platform Game Development, New York City, Apress, 2013.

[3] A. Thron, Unity 4 Fundamentals: Get Started at Making Games with Unity, Burlington, Focal Press, 2014.

[4] https://docs.unity3d.com/Manual/OcclusionCulling.html [12.11.2016].

[5] W. Goldstone, Unity 3.x Game Development Essentials, Birmingham, Packt Publishing, 2009.

[6] C. Dickinson, Unity 5 Game Optimization, Birmingham, Packt Publishing, 2015.

[7] A. Thorn, How to Cheat in Unity 5: Tips and Tricks for Game Development, Burlington, Focal Press, 2016.

[8] https://docs.unity3d.com/Manual/DrawCallBatching.html [14.11.2016].

[9] A. Thorn, Practical Game Development with Unity and Blender, Boston, Cengage Learning, 2014.

[10] https://unity3d.com/learn/tutorials/topics/graphics/introduction-lighting-and-rendering [16.11.2016].

JCSI 3 (2017) 54-58
Journal of Computer Sciences Institute

Wysłane: 2016-12-15
Przyjęte: 2016-12-16

Porównanie aplikacji mobilnej w językach Swift i Objective-C

Kacper Erwin Sienkiewicz*, Edyta Łukasik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Tematyką artykułu jest porównanie metodyk wytwarzania aplikacji mobilnych w językach Swift i Objective-C. W ramach przeprowadzonej analizy zostaną wskazane podobieństwa i różnice w implementacji aplikacji na te dwa języki programowania. Zaprojektowana została i zaimplementowana aplikacja Magic Drawing Board wykorzystująca silnik QUARTZ 2D. Stworzono dwie identyczne funkcjonalnie aplikacje. Analiza porównawcza została przeprowadzona dopiero po dokładnym objaśnieniu implementowanych widoków.

Słowa kluczowe: metody wytwarzania aplikacji; analiza implementacji widoków; silnik QUARTZ 2D.

*Autor do korespondencji.

Adres e-mail: ksienkiewicz14@gmail.com

Comparison of mobile application using Swift and Objective-C

Kacper Erwin Sienkiewicz*, Edyta Łukasik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The subject of the article is to compare the methodologies for the production of mobile applications in Swift and Objective-C languages. The similarities and differences of the implementation of applications for these two programming languages will be identified as a part of the analysis. An Magic Drawing Board application using Quartz 2D engine was designed and implemented. Two identical functional applications were created. The comparative analysis was carried out only after a thorough explanation of the implemented views.

Keywords: method of applications development; the analysis of the implementation of views; QUARTZ 2D engine.

*Corresponding author.

E-mail address: ksienkiewicz14@gmail.com

1. Wstęp

W przeciągu ostatnich dekad znacznie wzrosło znaczenie aplikacji mobilnych w technologiach informatycznych. Obecnie większość populacji w krajach dobrze rozwiniętych nie wyobraża sobie funkcjonowania w życiu codziennym bez smartfonów i zawartych w nim aplikacji. Głównie dotyczy to prostych rzeczy, takich jak: sprawdzenie rozkładu jazdy autobusu, sprawdzenia pogody, czy kontakt z innymi użytkownikami przez media społecznościowe. Za tymi postępami musi również nadążać technologia, dlatego w dziedzinie programistycznej również widać, jak z każdym rokiem pojawią się nowe języki programowanie lub modernizacje języków starszej daty.

Firma Apple, znana z renomy swoich produktów, nie mogła dłużej czekać na wypuszczenie nowego języka. Podczas konferencji organizowanej przez firmę Apple o nazwie *Worldwide Developers Conference* (WWDC), 2 czerwca 2014 został zaprezentowany język Swift. Aktualna wówczas wersja języka była to *Swift 1.0*. Ogólnym założeniem języka jest zastąpienie Objective-C. Ma on służyć do sprawniejszego niż dotąd tworzenia aplikacji pracujących pod kontrolą systemów *OS X* i *iOS*. Aktualnie Swift ciągle się rozwija i na każdej corocznej czerwcowej konferencji *WWDC* pojawia się inna wersja języka. W 2016 roku była to wersja *Swift 3.0*. Ostatnia wersja języka wprowadziła bardzo wiele zmian w składni, jeszcze bardziej podkreślając, jak jest to wysokopoziomowy język programowania [1].

Apple stworzyło Swift z myślą o obniżeniu bariery wejścia dla programistów chcących tworzyć oprogramowanie w ich technologii. Nowy język jest tym narzędziem, do którego przyzwyczajeni są młodzi programiści, z wysokopoziomowymi strukturami wbudowanymi w język i wieloma ustawieniami systemowymi.

Celem artykułu jest porównanie metodyk wytwarzania oprogramowania na stworzonej aplikacji Magic Drawing Board w językach Swift i Objective-C. Analizie porównawczej została poddana składania obu tych języków. Podstawowa funkcjonalność tej aplikacji to edytor do rysowania z możliwością zapisu obrazu w galerii zdjęć. Aplikacja została zaprojektowana na urządzenie iPad.

W celu utworzenia aplikacji mobilnej wykorzystano następujące narzędzia:

- środowisko programistyczne X-code [2];
- języki programowania Swift i Objective-C;
- silnik QUARTZ 2D;
- emulatory oraz urządzenie iPad Pro.
- Do porównania obu języków została przeprowadzona analiza teoretyczna związana z funkcjonowaniem metod implementujących widoki aplikacji. W jej wyniku wskazane zostały podobieństwa i różnice równocześnie mocne i słabe strony każdego z języków. Zwrócono głównie uwagę na najważniejsze aspekty związane z działaniem obu aplikacji, czyli:
- składania języków;

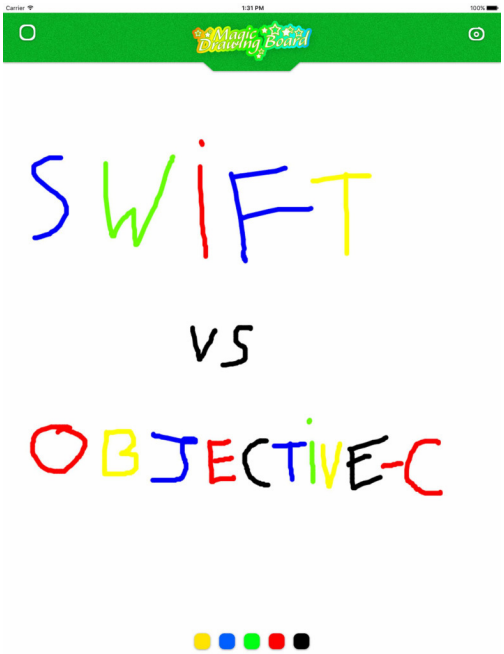
- wykorzystane metody;
- wykorzystanie silnika QUARTZ 2D.

2. Opis aplikacji

Utworzone zostały dwie identyczne aplikacje pod względem graficznym. Każda z nich posiada następujące funkcjonalności:

- w lewym górnym rogu znajduje się przycisk, który służy do wyczyszczenia wszystkiego, co zostało narysowane;
- przycisk po prawej stronie zapisuje narysowany obrazek i wysyła go do wbudowanej na urządzeniu aplikacji *zdjęcia*;
- pięć dostępnych przycisków reprezentujących kolory.

Wygląd przykładowych ekranów działających aplikacji w obu językach został pokazany na rysunkach 1 i 2. Po uruchomieniu aplikacji pojawi się główny widok aplikacji. Przykładowy efekt jej działania pokazano na rys. 1.



Rys. 1. Interfejs graficzny w aplikacji Magic Drawing Board dla języków Swift i Objective-C

Zapisany, po naciśnieciu przycisku umożliwiającego zapis aktualnego stanu ekranu, w galeri zdjęć narysowany obraz pokazano na rys. 2.

3. Opisanie składni języków Swift i Objetive-C na przykładzie listingów

3.1 Opis użytych funkcji w języku Swift

Funkcja *touchesBegan*, przedstawiona na przykładzie 1, odpowiedzialna jest za początek rysowania linii na ekranie [3]. Słowo kluczowe *override* umieszczone przez nazwą funkcji oznacza, że jest to metoda domyślna, która generuje automatycznie kod. W środku funkcji jest zainicjalizowanie zmiennej old, w której zawarte zostanie pierwsze dotknięcie oraz lokalizacja na widoku ekranu.

Rys. 2. Zapisany stan obrazka w galerii zdjęć

Przykład 1. Implementacja dotyku użytkownika na ekranie

```
var old:CGPoint?
override func touchesBegan(_ touches: Set<UITouch>, with
event: UIEvent?)
{
        old = touches.first?.location(in: self.view)
 }
```

Kolejna domyślna funkcja, *touchesMoved* określa, w którym kierunku podąża palec użytkownika. Wewnątrz metody zmienna *current*, również pełni tę samą rolę, co zmienna *old,* lecz jest wykorzystana w celu określenia, w którym kierunku zmierza rysowana linia. Następna zmienna o nazwie *ctt* reprezentuje obiekt *UIGraphics-GetCurrentContext*, który z założenia jest prostokątem. Następnie jest wpisana nazwa zmiennej *ctt* i odwołania po kropce do właściwości tej figury i podane są własne ustawienia. Funkcja ta przedstawiona została na przykładzie 2.

Przykład 2. Implementacja ruchu rysowanej linii

```
override func touchesMoved(_ touches: Set<UITouch>, with
event: UIEvent?) {
        let current = touches.first?.location(in: self.view)
        let ctt = UIGraphicsGetCurrentContext()
        ctt?.setStrokeColor(currentcolor.cgColor)
        ctt?.setLineWidth(10.0)

        ctt?.move(to: CGPoint(x: old!.x, y: old!.y))
        ctt?.addLine(to: CGPoint(x: current!.x, y: current!.y))
        ctt?.strokePath()

        let image =
UIGraphicsGetImageFromCurrentImageContext()
        (self.view as! UIImageView).image=image
}
```

W celu oprogramowania przycisku *zapisz w galerii,* wewnątrz funkcji została wykorzystana domyślna metoda *UIImageWriteToSavedPhotosAlbum.* Umożliwia ona w prosty sposób przekazanie do aplikacji *zdjęcia* tego, co

zostało narysowane na ekranie[4]. Jej zawartość przedstawiono na przykładzie 3.

Przykład 3. Implementacja przycisku zapisującego rysunek w galerii zdjęć

```
@IBAction func saveCamera(_ sender: AnyObject) {
UIImageWriteToSavedPhotosAlbum(UIGraphicsGetImageFro
mCurrentImag
eContext()!, nil,nil,nil)
}
```

W każdej funkcji jest odwołanie do zmiennej *currentcolor*, która przyjmuje wartość obiektu typu *UIColor*, Zapewnia ona dostęp do podstawowych kolorów, w tym przypadku jest to *red* (czerwony). Ustala ona jakiego koloru będzie rysowana pierwsza linia. Jej użycie pokazano na przykładzie 4.

Przykład 4. Oprogramowanie dwóch z pięciu kolorowych przycisków

```
var currentcolor=UIColor.red
 @IBAction func yellowClicked(_ sender: AnyObject) {
     currentcolor=UIColor.yellow
 }
   @IBAction func blueClicked(_ sender: AnyObject) {
      currentcolor=UIColor.blue
 }
```

Przycisk *clean* wywołuje funkcje *initContext,* której zawartość pokazano na przykładzie 5. Jest to metoda, która zawiera w sobie obiekt *UIGraphicBeginImageContext*, który sam w sobie zawiera funkcje umożliwiającą zresetowanie widoku *UIImageView*.

Przykład 5. Wywołanie funkcji initContext w implementacji przycisku cleanClicked

```
@IBAction func cleanClicked(_ sender: AnyObject) {
      self.initContext()
   }
func initContext() {
      UIGraphicsBeginImageContext(self.view.frame.size)
      let image =
UIGraphicsGetImageFromCurrentImageContext()
      (self.view as! UIImageView).image=image
   }
```

3.2 Opis użytych metod w języku Objective-C

W pliku ViewController.h, przedstawionym na przykładzie 6, dodane zostały instancje zmiennych:

- *lastPoint* – przechowuje ostatni namalowany punkt na obrazie;
- *red*, *green*, *blue* – zapisuje aktualną wartość RGB wybranego koloru;
- *brush*, *opacity* – przechowują szerokość pociągnięcia pędzla i jego krycie;
- *mouseSwiped* – identyfikuje, czy pociągnięcie pędzla *brush* będzie kontynuowane.

Przykład 6. Zawartość klasy ViewController.h

```
@interface ViewController : UIViewController {
   CGPoint lastPoint;
   CGFloat red;
   CGFloat green;
   CGFloat blue;
   CGFloat brush;
   CGFloat opacity;
}
@property (weak, nonatomic)
IBOutlet UIImageView *mainImage;
- (IBAction)pencilPressed:(id)sender;
- (IBAction)save:(id)sender;
- (IBAction)reset:(id)sender;
@end
```

Następnie zostały ustawione połączenia dla widoków utworzonych w *Interface Builder*. Obiekt typu *UIImageView* o nazwie *mainImage* to główny obraz, na nim odbywa się rysowanie.

Utworzone zostały połączenia dla pięciu kolorów, dla przycisków: *pencilPressed, save* i *reset*. Wartość początkowa koloru w notacji *RGB* jest ustawiona na czarno. Domyślne krycie jest ustawione na 1.0, a szerokość linii na 10.0. Na starcie aplikacji zostaną te ustawienia od razu zastosowane, gdyż znajdują się w metodzie *viewDidLoad*, pokazanej na przykładzie 7 [5].

Przykład 7. Metoda wywołana na starcie aplikacji

```
- (void)viewDidLoad {
   red = 0.0/255.0;
   green = 0.0/255.0;
   blue = 0.0/255.0;
   brush = 10.0;
   opacity = 1.0;
   [super viewDidLoad];
}
```

W metodzie *touchesBegan* na przykładzie 8, zmienna *lastPoint* jest inicjalizowana wartością bieżącego punktu dotyku. Metoda lokalizuje, w którym miejscu użytkownik dotknął ekranu.

Przykład 8. Implementacja metody touchesBegan

```
- (void)touchesBegan:(NSSet *)touches withEvent:
 (UIEvent *)event {
   UITouch *touch = [touches anyObject];
   lastPoint = [touch locationInView:self.view];
}
```

Metoda *touchesMoved* śledząca ruch dotyku pokazana została na przykładzie 9. Na początku metody zostanie zlokalizowana pozycja dotyku na ekranie. Pozwoli to następującym obiektom: *CGContextMoveToPoint*, *CGContextAddLineToPoint* i *CGContextSetLineCap* na narysowanie linii z ostatniego zlokalizowanego punktu dotyku do bieżącego [6].

Przykład 9. Implementacja metody touchesMoved

```
- (void)touchesMoved:(NSSet *)touches withEvent:
(UIEvent *)event {
   mouseSwiped = YES;
   UITouch *touch = [touches anyObject];
   CGPoint currentPoint = [touch locationInView:self.view];
   UIGraphicsBeginImageContext(self.view.frame.size);
   [self.mainImage.image drawInRect:CGRectMake(0, 0,
   self.view.frame.size.width, self.view.frame.size.height)];
   CGContextMoveToPoint(UIGraphicsGetCurrentContext(),
   lastPoint.x, lastPoint.y);
   GContextAddLineToPoint(UIGraphicsGetCurrentContext(),
   currentPoint.x, currentPoint.y);
   CGContextSetLineCap(UIGraphicsGetCurrentContext(),
   kCGLineCapRound);
```

Kolejne trzy obiekty: *CGContextSetLineWidth*, *CGContextSetRGBStrokeColor* i *CGContextSetBlendMode* ustawiają rozmiar pędzla, jego poziom krycia oraz kolor. Ich użycie pokazano na przykładzie 10.

Przykład 10. Inicjalizacja obiektów w metodzie touchesMoved

```
CGContextSetLineWidth(UIGraphicsGetCurrentContext(),
 brush );
CGContextSetRGBStrokeColor(UIGraphicsGetCurrentContext
(), red, green, blue, 1.0);
CGContextSetBlendMode(UIGraphicsGetCurrentContext(),
kCGBlendModeNormal);
```

Obiekt CGContextStrokePath, którego użycie pokazano na przykładzie 11, zakończy działanie metody poprzez narysowanie ścieżki na obrazie.

Przykład 11. Zakończenie metody touchesMoved

```
CGContextStrokePath(UIGraphicsGetCurrentContext());
   self.mainImage.image=
 UIGraphicsGetImageFromCurrentImageContext();
   [self.mainImage setAlpha:opacity];
   UIGraphicsEndImageContext();
}
```

Do każdego przycisku został przypisany *tag*, co umożliwi odróżnienie przycisków i wykorzystanie instrukcji *switch*. Pozwoli to także na nadanie odpowiedniego koloru ustawionego wartościami *RGB* dla dostępnych pięciu przycisków na głównym widoku interfejsu aplikacji. Na przykładzie 12 przedstawiono implementację przycisków z kolorami.

Przykład 12. Implentacja kolorowych przycisków

```
- (IBAction)pencilPressed:(id)sender {
   UIButton * PressedButton = (UIButton*)sender;
   switch(PressedButton.tag)
   {
      case 0:
       red = 255.0/255.0;  green = 255.0/255.0; blue =
       0.0/255.0;
       break;
      case 1:
       red = 0.0/255.0;  green = 0.0/255.0;  blue =
      255.0/255.0;
       break;
   }
```

Zresetowanie wszystkiego, co zostało narysowane, zrealizowane zostanie poprzez przypisanie wartości *nil* dla obiektu *mainImage*. Metoda obsługująca przycisk *reset* przedstawiona została na przykładzie 13.

Przykład 13. Implementacja przycisku reset

```
- (IBAction)reset:(id)sender {
      self.mainImage.image = nil;
}
```

Wciśnięcie przycisku *save*, zapisze we wbudowanej aplikacji *zdjęcia* na urządzeniu to, co zostało aktualnie narysowane na obiekcie *UIImageView*. Jego implementację przedstawiono na listingu 14.

Przykład 14. Implementacja przycisku save

```
- (IBAction)save:(id)sender {
 UIGraphicsBeginImageContextWithOptions(self.mainImage.
 bounds.size, NO, 0.0);
 [self.mainImage.image drawInRect:CGRectMake(0, 0,
 self.mainImage.frame.size.width,
 self.mainImage.frame.size.height)];
 UIImage *eSaveImage=
 UIGraphicsGetImageFromCurrentImageContext();
 UIGraphicsEndImageContext();
 UIImageWriteToSavedPhotosAlbum(eSaveImage,
 self,nil, nil);
 }
```

4. Podobieństwa i różnice języków w aplikacji Magic Drawing Board

Obie aplikacje wykorzystują specjalny silnik *Quartz 2D* [6], służący do tworzenia dwuwymiarowej grafiki. *Quartz* bazuje na języku C, większość wywoływanych funkcji jest napisana w tym języku, lecz dla języka Swift wszystko jest napisane już w Swift. Rysowanie zawsze odbywa się w klasie *UIView* lub w klasach dziedziczących po *UIView*. Rysowanie bazuje na tworzeniu ścieżek, na których zostają namalowane linie oraz definiowane są kształty, które zostaną odmalowane. Funkcje operują na pojęciu punktu. Punkt to współrzędne X oraz Y ekranu, gdzie punkt 0,0 znajduje się w lewym górnym rogu ekranu. Wartości X i Y są przechowane w zmiennej typu *CGFloat*. Struktura *CGPoint* użyta w obu języka przechowuje te wartości dla punktów ekranu.

Wywołanie większości funkcji w *Quartz 2D* wymaga przekazania parametru tzw. *context* (kontekstu). Każdy widok posiada swój własny kontekst, który jest odpowiedzialny za przeprowadzenie operacji rysowania. Kontekst jest przechowywany w zmiennej o typie *CGContext* [7].

Można zauważyć, że w obu językach są zawarte takie same funkcje, gdzie ich nazwy zaczynają się od słowa *CGContext*, np.:

- *CGContextSetLineWidth;*
- *CGContextSetStrokeColorWithColor*;
- *CGContextMoveToPoint*.

Język Swift i Objective-C posiadają metody o takich samych nazwach: metody *touchesBegan* i *touchesMoved*. Są one metodami domyślnymi z automatycznie generowanymi parametrami. Warto zaznaczyć, że operowanie na obiekcie *CGContext* w języku Swift odbywa się w bardziej uproszczony sposób. Dla Objective-C wywołanie tych samych metod odbywa się bardziej precyzyjnie i szczegółowo. W języku Swift wystarczy tylko utworzyć zmienną, która wywoła funkcje *UIGraphics-GetCurrentContext*, następnie należy ustawić kropce wartości takich zmiennych, jak: *setStrokeColor*, *setLineWidth*, *move, addLine* itd. (listing 1). Zatem należy stwierdzić, że główną różnicą w obu przypadkach jest składnia, która godzi się lepiej na korzyść języka Swift .

Implementacja odpowiednich kolorów dla każdego z przycisków w każdym języku wygląda inaczej. W aplikacji napisanej w Objective-C został wykorzystany jeden z modeli przestrzeni barw *RGB*. Pozwoliło to na przypisanie odpowiednich wartości liczbowych dla każdej z trzech zmiennych: *red*, *green*, *blue*. W aplikacji dla języka Swift została wykorzystana specjalna do tego klasa *UIColor,* która zawiera w sobie piętnaście podstawowych kolorów. Oczywiście takie same rozwiązania można było by zastosować dla jednego i drugiego języka, ponieważ język Objective-C również zawiera taką klasę.

5. Podsumowanie

W artykule zaprezentowano metodyki tworzenia aplikacji mobilnych na platformę iOS w językach Swift i Objective-C. Na przykładzie stworzonej aplikacji wskazano podobieństwa i różnice podczas ich projektowania. Opisane zostały różnice między językami w implementacji różnych widoków od początku tworzenia projektu do jego końca.

W wyniku przeprowadzonych analiz napisanego kodu aplikacji można zauważyć, jak bardzo różnią się od siebie jezyki Swift i Objective-C. Zdecydowanie sprawniej można wykorzystać język Swift do implementacji różnych widoków. Tworzenie kodu w tym języku programowania jest bardziej zrozumiałe i przyjemne. W większości przypadków jest go mniej, a to jest ważną rzeczą dla programisty, ponieważ zachodzi mniejsze prawdopodobieństwo wyskoczenia błędu podczas kompilacji projektu. Różnicę tą można zauważyć na dwóch metodach implementujących dotyk i ruch rysowanej linii przez użytkownika. Już samo to, że klasa nie zawiera dwóch plików: interfejsu i implementacji, jak w przypadku aplikacji napisanej w języku Objective-C, stawia Swift na lepszej pozycji. Jedyną wadę, jaką można wskazć, to fakt,iż jest on w ciągłym rozwoju i składnia języka może ulec większej lub mniejszej zmianie. Jednak zwykle tego typu zmiany wychodzą na lepsze.

Język Objective-C nadal otrzymuje wsparcie od firmy Apple, lecz z biegiem czasu popularność języka Swift zdobywa coraz większe uznania wśród programistów tworzących oprogramowanie na platformę OS X i iOS. Czynnik ten może zdecydować, że w przyszłości przestanie on być używany.

Literatura

[1] code.tutsplus https://code.tutsplus.com/pl/tutorials/an-introduction -to-swift-part-1--cms-21389 [Dostęp: 14.12.2016].

[2] developer.apple https://developer.apple.com/xcode/ [Dostęp: 14.12.2016]

[3] Ng S.: Beginning iOS 9, Programming with Swift. AppCoda Limited. 2015.

[4] Ng S.: Intermediate iOS Programming with Swift. AppCoda Limited. 2015.

[5] Kochan S.G.: Objective-C, Vademecum profesjonalisty. Helion, Gliwice. 2012.

[6] developer.apple https://developer.apple.com/library/content/documenttationn/GraphicsImaging/Conceptual/drawingwithquartz2d/Introduction/Introduction.html [Dostęp 14.12.2016]

[7] code.tutsplus https://code.tutsplus.com/tutorials/an-introduction to-quartz-2d--cms-24267 [Dostęp 14.12.2016]

JCSI 3 (2017) 59-63
Journal of Computer Sciences Institute

Wysłane: 2016-12-14
Przyjęte: 2016-12-16

Analiza wydajności silnika Unity3D w aspekcie symulacji cząsteczkowych

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

Streszczenie: Dokument ten przedstawia badania w sektorze wydajności cząsteczek utworzonych przez wbudowany w Unity3D system cząsteczek oraz tworzonych poprzez skrypt napisanym w języku C#. Autor skupił się na zbadaniu wpływu poszczególnych zadań na szybkość renderowania klatki, różnicy jakie występują podczas generowania cząsteczek przez system cząsteczek, a tych generowanych przez skrypt oraz jak duży wpływ ma złożoność siatki cząsteczki generowanej na zachwianie wydajności aplikacji. W tym celu, z użyciem Unity3D, Microsoft Visual Studio oraz Blendera, została utworzona aplikacja na komputery z systemem Windows która pozwala na dostosowanie parametrów symulacji (liczby generowanych cząsteczek, kształtu cząsteczki – kula, sześcian).

Słowa kluczowe: cząsteczka; system cząsteczek; Unity3D; wydajność; symulacja

*Autor do korespondencji.

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

Efficiency analysis of Unity3D engine in terms of particle simulation

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

Abstract. This document reviews research in efficiency analysys of particles created by Unity 3D built-in particle system compared to those created via sctript written in C# language. Author focused his research on examining the impact of the various tasks on the speed of rendering frames, a difference which occur during the generation of particle by particle system and those generated by a script, and how influential is the complexity of the mesh particles generated on the application performance. For this purpose, using Unity3D, Microsoft Visual Studio and Blender, an applications for computers that are ruunniong on Windows system was created, application allows to adjust the simulation parameters (the number of generated particles, particle shape - sphere, cube).

Keywords: Particle; Particle system; Unity3D; Efficiency; Simulation

*Corresponding author.

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

1. Wstęp

Efekty cząsteczkowe są nieodzownym elementem każdej dopracowanej gry dodając jej realizmu lub wręcz odwrotnie ujmując go. Używając systemu cząsteczek we właściwy sposób, ten element pozwala na dodanie odrobiny realizmu w tworzony świat.

Cząsteczki to małe elementy (obrazy 2D lub siatki 3D) [1], które emitowane w dużych ilościach oraz sterowane przez system cząsteczek [2], pozwalają na osiągnięcie efektów takich jak: dym, ogień, deszcz.

Przykładowo każda cząsteczka emitowana przez system cząsteczek który zajmuje się tworzeniem efektu deszczu lub śniegu, jest pojedynczą kroplą lub płatkiem. Podobnie jest w przypadku tworzenia efektu dymu, gdzie pojedyncza cząstka w odcieniu szarości, wyemitowana pojedynczo nie będzie przypominała postaci unoszącej się sadzy, natomiast wyemitowana wielokrotnie przez system cząsteczek będzie sprawiała to wrażenie.

Cząsteczka by spełniała określone zachowania występujące w naturze musi posiadać określone właściwości [3-4].

Właściwości cząsteczki która jest jednym z tysiąca o ile nie większej liczby generowanych elementów to m. in.: wektor prędkości określający kierunek i dystans jaki cząsteczka pokonuje podczas jednej klatki, rozmiar, kolor. Wspomniany kolor może ulegać zmianie zarówno przed jak i w czasie życia cząsteczki lub też zmieniając się w zależności od prędkości czy od czasu jaki ona istnieje.

Niektóre właściwości mogą ulec wpływom sił zewnętrznym takim jak grawitacja np. wektor prędkości może ulec zmianie, może zostać skierowany w dół (dla cząsteczek imitujących krople deszczu) lub w górę (dla cząsteczek imitujących ogień, dym) gdy siła oddziaływania grawitacji jest nieuwzględniana.

Wszystkie parametry określa się przy użyciu systemu cząsteczek, który z użyciem modułu emisji [5] uwolni je do utworzonego wirtualnego świata.

Te parametry to m. in. częstotliwość generowania cząsteczek, liczba cząstek generowana na raz, czas życia cząsteczek, kształt cząsteczki, oraz te parametry które wspomniane były wcześniej, czyli, kolor, wektor prędkości, wpływ grawitacji, rozmiar, a także aktywacja kolizji cząsteczek z innymi obiektami.

Wraz z wprowadzeniem w Unity 5 wydajniejszego silnika fizycznego PhysX [6-9] w wersji 3.3 którego kod źródłowy został udostępniony na licencji open-source pojawiło się pytanie, jak bardzo wydajne jest symulowanie kolizji wielu obiektów(cząsteczek) przez silnik w porównaniu do symulacji cząsteczek utworzonych przez wbudowany od wersji 3.5 Unity, system cząsteczek.

Analiza i porównanie zostało przeprowadzone z użyciem aplikacji desktopowej oraz przy użyciu narzędzi dostarczonych przez środowisko Unity3D (Profiler)[10].

Analiza będzie dotyczyć wbudowanego w Unity3D systemu cząsteczek oraz zostanie on porównany wydajnościowo ze skryptowo tworzonymi cząsteczkami (obiekty typu GameObject z takimi samymi właściwościami). Hipotezy jakie zostały podstawione przed wykonaniem symulacji to:

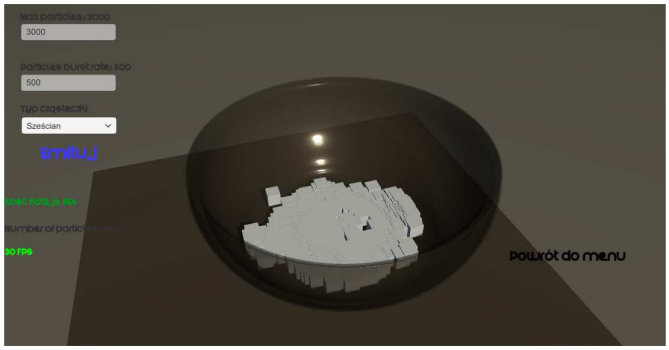
1.1. Złożoność cząsteczki emitowanej przez SC, negatywnie wpływa na wydajność aplikacji.
1.2. Liczba kolizji znacząco wpływa na spadek wydajności
1.3. Symulacja cząsteczek wyemitowanych przez system cząsteczek w większym stopniu obciąża system, niż symulacja cząsteczek wygenerowanych skryptowo.

2. Opis symulacji

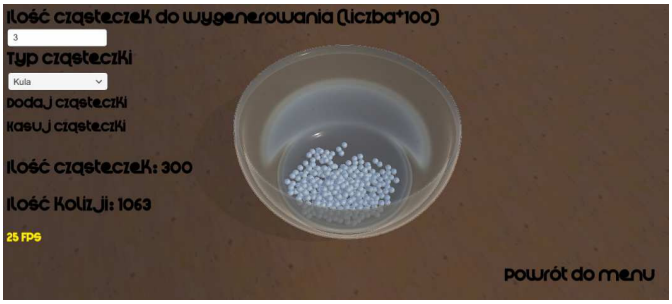
By umożliwić symulację oraz zbiór danych z wykorzystaniem narzędzia „profiler" dostępnego w środowisku Unity[11, 12], została utworzona aplikacja. Program powstał przy użyciu wspomnianego środowiska, skryptów pisanych w języku C# [13-15] w Microsoft Visual Studio, a model naczynia utworzono w programie Blender [16-18]. Aplikacja pozwala na dostosowanie parametrów niezbędnych wspomnianych w powyższym podrozdziale, a następnie emisję sparametryzowanych cząsteczek.

By umożliwić symulację program udostępniał możliwość zmiany, zdefiniowania parametrów dotyczących liczby emitowanych cząsteczek oraz kształtu cząsteczki na każdej z dostępnych (2) scen przeznaczonych do symulacji.

Widoki aplikacji są przedstawione na rysunkach poniżej (Rysunek 1 i 2).



Rys. 1. Scena symulacji systemu cząsteczek



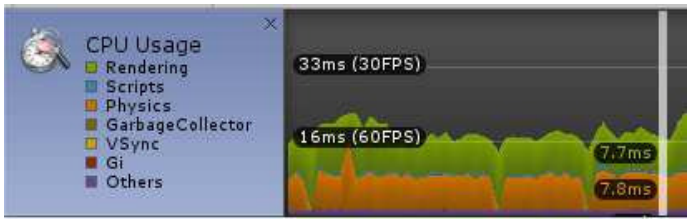
Rys.2. Scena symulacji cząsteczek tworzonych przez skrypt

Aby zbadać trafność postawionych hipotez, poszczególne symulacje przeprowadzono na 4 komputerach z systemem Windows. Parametrem, jaki był w każdej symulacji taki sam to liczba cząsteczek, która wynosiła 1000. Zmianie poddawany był kształt cząsteczki, w zależności od sposobu generowania cząsteczki.

Te przypadki to:

- Przypadek 1 - generowanie cząsteczek (sześcianów) przy użyciu systemu cząsteczek
- Przypadek 2 - generowanie cząsteczek (sześcianów) przy użyciu skryptu
- Przypadek 3 - generowanie cząsteczek (kul) przy użyciu systemu cząsteczek
- Przypadek 4 - generowanie cząsteczek (kul) przy użyciu skryptu

Odczytanie poszczególnych wyników symulacji przebiegało w każdym przypadku przez odczyt danych z *„profilera"* w następujący sposób:
W opisie rysunku „x" to numer maszyny na której została przeprowadzana symulacja, a „y" numer przypadku.



Rys.3. Użycie procesora, przypadek y, maszyna x

Overview	Total	Self	Time ms	Self ms
▶ Physics.Processing	37.3%	7.5%	6.38	1.28
Gfx.WaitForPresent	34.8%	34.8%	5.96	5.96
▶ Camera.Render	11.3%	0.3%	1.94	0.06
▶ BehaviourUpdate	4.0%	2.1%	0.68	0.37
Physics.FetchResults	3.0%	3.0%	0.52	0.52
Physics.UpdateBodies	2.9%	2.9%	0.50	0.50
▶ Physics.ProcessReports	1.7%	0.0%	0.29	0.00
Overhead	1.6%	1.6%	0.28	0.28
Profiler.FinalizeAndSendFrame	1.1%	1.1%	0.19	0.19
▶ Canvas.SendWillRenderCanvases()	0.7%	0.6%	0.12	0.11

Rys.4. CPU, procesy, przypadek y, maszyna x

Korzystając z danych dostarczonych przez to narzędzie można było określić czas generowania poszczególnej klatki, a także określić stopień obciążenia procesora oraz jakie zadanie najbardziej go obciąża. W tym wypadku czas generowania pojedynczej klatki to ~16ms, a proces w największym stopniu obciążający procesor to ten, odpowiedzialny za obliczenia związane z fizyką obiektów.

2.1. Wykorzystane konfiguracje sprzętowe

W tym podrozdziale zostaną przedstawione fizyczne parametry konfiguracji sprzętowych maszyn użytych w celach przeprowadzenia symulacji.

Maszyna 1

- Procesor: Intel Core i5-2430M
- Pamięć RAM : 6GB
- Grafika: Nvidia GeForce GT 520MX

Maszyna 2

- Procesor: Intel Core i7-5500U
- Pamięć RAM: 8GB
- Grafika: AMD Radeon R5 M330

Maszyna 3

- Procesor: Intel Core 2 Quad Q6600
- Pamięć RAM 8GB
- Grafika: Nvidia GeForce 9600 GT

Maszyna 4

- Procesor: Intel Core 2 Duo P8400
- Pamięć RAM: 4GB
- Grafika: Nvidia GeForce 9600m GT

2.2. Kod umożliwiający generowanie cząsteczek przy użyciu skryptu

By umożliwić tworzenie cząsteczek w ten sposób, musiała zostać utworzona metoda, będzie za to odpowiedzialna.

Parametrami metody są:

- Liczba cząsteczek w linii (numberOfSpheres)
- Liczba linii w płaszczyźnie (numberOfLines)
- Numer płaszczyzny (row)

Kod metody przedstawia przykład 1.

Przykład 1. Metoda odpowiedzialna za tworzenie cząsteczek przez skrypt.

```
void createPlainOfSpheres(int numberOfSpheres, int
numberOfLines, int row)
{
  Rigidbody sphereRigidbody;
  int sphereNameIndex =
    GameObject.FindGameObjectsWithTag("sphere").Length;
 for (int j = 1; j <= numberOfLines; j++)
 {
   int sphereIndex = (j - 1 == 0) ? 0 : (j - 1)
   *numberOfSpheres;
    transformVector.y +=
        1.1f*sphereCollider.bounds.size.y *row;
   for (int i = sphereIndex; i < numberOfSpheres * j; i++)
   {
      //indeks dodawany do nazwy obiektu
      sphereNameIndex++;
      sphereRigidbody = new Rigidbody();
     //utworzenie obiektu o zadanym typie (kula lub sześcian)
     sphere = GameObject.CreatePrimitive(objectType);
    //do właściwości komponentu Collider
    //zostaje przypisany materiał
    //(po to by cząsteczka m.in. odbijała się)
    sphere.GetComponent<Collider>().material = pm;
    //nadanie tagu
    sphere.tag = "sphere";
    //dodanie skryptu do obiektu po to by
    //zliczać kolizje występujące na obiekcie
    sphere.AddComponent<ObjectsCollisionScript>();
   //dodanie komponentu Rigidbody
   sphereRigidbody = sphere.AddComponent<Rigidbody>();
   //nadanie obiektowi masy
   sphereRigidbody.mass = Random.Range(4.0f, 500.0f);
   //ustawienie oddziaływania siły grawitacji na obiekt
   sphereRigidbody.useGravity = true;
   //nadanie nazwy obiektowi
   sphere.name = "sphere" + sphereNameIndex;
   //określenie sposobu detekcji kolizji
    sphereRigidbody.collisionDetectionMode =
    CollisionDetectionMode.ContinuousDynamic;
    //nadanie pozycji
    transformVector.x += 1.1f*sphereCollider.bounds.size.x;
    sphere.transform.position = transformVector;
  }
 transformVector = tableCenter;
 transformVector.z += 1.1f*sphereCollider.bounds.size.z * j;
 }
 transformVector = tableCenter;
}
```

Metoda napisana w języku C#[13-15], każdemu tworzonemu prymitywowi (sześcianowi lub kuli) nadawała odpowiednie właściwości przy użyciu odpowiednich komponentów. Tak utworzony obiekt posiada własną siatkę kolizji, masę, materiał, nazwę.

3. Wyniki symulacji

W tym rozdziale zostały zebrane wyniki wszystkich przeprowadzonych symulacji na każdej z maszyn (Tabela 1), przedstawione w postaci tabeli tak, by w łatwy sposób zauważyć różnice w kwestiach liczby generowanych klatek na sekundę jak i czasu generowania klatki. Kolumny na niebiesko oznaczają dane zebrane przy okazji generacji cząstek przy użyciu systemu cząsteczek wbudowanego w Unity, kolumny zielone zaś oznaczają dane zebrane przy okazji emisji cząsteczek przy pomocy skryptu.

Tabela 1. Tabela wyników symulacji

Wyniki symulacji z poszczególnych maszyn	Przypadek 1	Przypadek 2	Przypadek 3	Przypadek 4
1	15 kl/s	80 kl/s	11 kl/s	66 kl/s
	65 ms	12 ms	86 ms	15 ms
2	60 kl/s	170 kl/s	22 kl/s	83 kl/s
	16 ms	6 ms	45 ms	12 ms
3	12 kl/s	66 kl/s	3 kl/s	70 kl/s
	78 ms	15 ms	260 ms	14 ms
4	8 kl/s	13 kl/s	3 kl/s	37 kl/s
	125 ms	74 ms	310 ms	26 ms

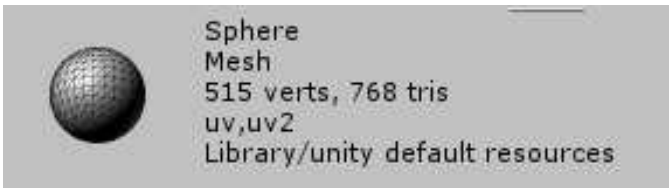
4. Analiza wyników symulacji

Jak można odczytać z tabeli 1, czasy generowania klatek podczas generowania cząsteczek przez wbudowany system cząsteczek(SC) czy czasy cząsteczek generowanych skryptowo różnią się w niektórych przypadkach nawet trzykrotnie (dla wyników symulacji na maszynie 3).

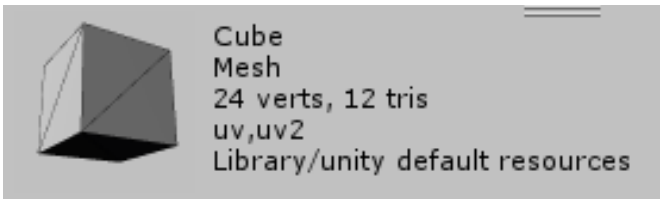
Wynika to z tego, że procesor który zajmuje się także tworzeniem, rysowaniem obiektów na scenie wymaga więcej czasu na stworzenie cząsteczki która przedstawia obiekt kuli, a mniej dla sześcianu.

Jak jest to widoczne na rysunkach poniżej, kula (Rysunek 3), składa się, aż z 515 wierzchołów i 768 trójkątów podczas gdy sześcian (Rysunek 4), zbudowany jest z jedynie 24 wierzchołków i 12 trójkątów, mimo wsparcia ze strony Unity technologią „Dynamic Batch" [20] która usprawnia działanie poprzez zaprzestanie wysyłania zadania przez procesor przy każdym rysowaniu tego samego obiektu, a wysłanie go tylko

raz by zostało wykonane określoną ilość razy dla obiektów o tej samej siatce, wolna szyna między procesorem a układem graficznym napotka problemy przy generowaniu dużej ilości kul, właśnie przez złożoność jej siatki.



Rys. 5. Obiekt kuli używany jako cząsteczka



Rys. 6. Sześcian używany jako cząsteczka

W symulacjach cząsteczek w przypadku tej aplikacji to druga przyczyna spadku liczby klatek generowanych na sekundę.

Pierwszą przyczyną spadku, są symulacje fizyki, zderzeń[21, 22] (w Unity obliczenia przeprowadzane są na procesorze z użyciem PhysX [9] dla prymitywów i bez użycia PhysX dla cząsteczek generowanych przy pomocy SC) co w wypadku generowania cząsteczek przez SC jak i tych tworzonych skryptowo, mogą doprowadzić nawet do zawieszenia aplikacji z braku dostępnych zasobów.

Biorąc pod uwagę czas generowania klatki jak i liczbę klatek na sekundę, można zauważyć, że typ cząsteczki generowanej przez SC, ma istotny wpływ na szybkość generowania z racji swojej mniej lub bardziej złożonej budowy - im mniej skomplikowana jest cząsteczka tym przy zadanej ilości cząsteczek do emisji będzie mniejszym obciążeniem dla procesora.

Podobna sytuacja wystąpiła podczas symulacji przypadków 2 oraz 4, zmiana kształtu cząsteczki emitowanej znacząco (co jest widoczne w tabeli 1) wpłynęła na wydajność aplikacji.

Wyniki we wszystkich przypadkach, są potwierdzeniem hipotezy na temat wpływu złożoności siatki cząsteczki na wydajność aplikacji.

Kolejna hipoteza stawiała pytanie, który sposób symulacji cząsteczek w Unity 3D jest wydajniejszy (czy jest to symulowanie cząsteczek wygenerowanych przez SC czy może tych wygenerowanych skryptowo).

Analizując wyniki z tabeli 1, można jednoznacznie określić która symulacja w mniejszym stopniu obciąża zasoby systemu, a co za tym idzie, czas generowania klatki jest mniejszy.

Hipoteza jest w tym przypadku częściowo prawdziwa, z tego powodu, że cząsteczki tj. prymitywy które znajdowały się w naczyniu pomimo wywołanej większej liczby kolizji, nie zmuszały one procesora do obliczania danych w sposób ciągły, dla wszystkich elementów, a jedynie dla niewielkiej liczby elementów które się poruszały wywołując nowe kolizje. Przy wstępnym zderzeniu wszystkich cząsteczek z naczyniem, czas generowania klatki zwiększa się niemal 10-krotnie, wtedy można powiedzieć że symulacja cząsteczek tworzonych przez SC jest mniej obciążająca system, z racji wywołania mniejszej liczby kolizji.

Te wnioski dają odpowiedź na ostatnią hipotezę odnośnie wpływu liczby kolizji na wydajność aplikacji. Hipoteza jest całkowicie trafna, ponieważ jak wspomniane zostało wyżej, większa liczba kolizji, która występuje w czasie wstępnej zderzeń prymitywów z naczyniem jak i między sobą w większym stopniu obciążają system niż kolizje wywoływane przez cząsteczki wygenerowane przez system cząsteczek.

5. Podsumowanie

Podczas analizy skupiono się na złożoności cząsteczki emitowanej oraz sposobie w jaki ona była emitowana (skryptowo lub przez system cząsteczek).

Jedna z hipotez (1.3) okazała się częściowo trafna , ponieważ cząsteczki (prymitywy) po usytuowaniu się w naczyniu nie powodują powstawania nowych kolizji, przez co, nie istnieje potrzeba ponownych obliczeń dla kolizji na procesorze dla każdego obiektu. Biorąc pod uwagę cząsteczki wygenerowane przez SC, sytuacja jest zupełnie odwrotna. Cząsteczki są w ciągłym ruchu, powodują kolizje w coraz to innych miejscach, stąd procesor jest bardziej obciążony podczas symulacji, niż w przypadku symulacji prymitywów, które obciążają procesor obliczeniami związanymi z fizyką, kolizjami do momentu, gdy nie ułożą się w naczyniu na scenie.

Można dość do wniosku, że emisja przez system cząsteczek wbudowany w Unity jest mniej wydajna, niż emisja cząsteczek przy użyciu skryptu.

Jak zostało wykazane, największym obciążeniem podczas symulacji były obliczenia związane z fizyką, kolizjami.

Wbudowany system daje możliwość prostej, sprawnej konfiguracji w celu uzyskania efektu takiego jak ogień, deszcz, efekty które trudno osiągnąć przy pomocy cząstek generowanych przy użyciu skryptu.

Tworząc te efekty, rzadko kiedy cząsteczki dochodzą do kolizji z innym elementem świata gry, co pozwala na utworzenie ich w większej liczbie oraz nie obciążającej procesora w znacznym stopniu, co jest wystarczające dla twórców gier, by z niego korzystać.

Cząsteczki emitowane w grach, rzadko mają postać siatek 3-wymiarowych, (które zostały użyte w celach symulacji) a są to elementy 2-wymiarowe, które nie obciążają w takim stopniu procesora.

Innymi słowy system cząsteczek daje twórcy większą elastyczność, przy mniejszym nakładzie pracy.

Literatura

[1] https://docs.unity3d.com/Manual/PartSysWhatIs.html, dostęp 11.09.2016
[2] Eric Van de Kerckhove, Introduction to Unity: Particle Systems, https://www.raywenderlich.com/113049/introduction-unity-particle-systems, dostęp 08.09.2016
[3] William T. Reeves, Particle Systems A Technique for Modeling a Class of Fuzzy Objects, „Computer Graphics”, wydanie 17, numer 3, Lipiec 1983, s. 360-363

[4] Allen Martin, Particle Systems, https://web.cs.wpi.edu/~matt/courses/cs563/talks/psys.html, dostęp 10.09.2016
[5] https://docs.unity3d.com/Manual/PartSysEmissionModule.html, dostęp 7.09.2016
[6] Adrian Boeing, Thomas Bräunl, Evaluation of real-time physics simulation systems, GRAPHITE '07, s. 284-288
[7] Sean C. Mondesire, Douglas B. Maxwell Jonathan Stevens, Steven Zielinski, Glenn A. Martin, Physics Engine Benchmarking in Three-Dimensional Virtual World Simulation, MODSIM World 2016, s. 5-8
[8] Anthony, High-performace physics in Unity 5, https://blogs.unity3d.com/2014/07/08/high-performance-physics-in-unity-5/, dostęp 8.09.2016
[9] http://physxinfo.com/wiki/PhysX_SDK_3.x, dostęp 8.09.2016
[10] https://docs.unity3d.com/Manual/ProfilerWindow.html, dostęp 8.09.2016
[11] https://unity3d.com/, dostęp 8.09.2016
[12] https://en.wikipedia.org/wiki/Unity_(game_engine), dostęp 8.09.2016
[13] Andrew Stellman, Jennifer Greene, Head First C#, O'Reilly Media, Listopad 2007
[14] Alex Okita, Learning C# Programming with Unity 3D, CRC Press, Lipiec 7, 2014
[15] Jeff W. Murray, C# Game Programming Cookbook for Unity 3D, CRC Press, Czerwiec 24, 2014
[16] Ben Simonds, Blender. Praktyczny przewodnik po modelowaniu, rzeźbieniu i renderowaniu, Helion, Maj 22, 2014, s. 49-68
[17] Thorn Alan, Unity i Blender. Praktyczne tworzenie gier, Helion, Kwiecień 3, 2015
[18] https://pl.wikipedia.org/wiki/Blender_(program), 8.09.2016
[19] https://pl.wikipedia.org/wiki/Microsoft_Visual_Studio, 8.09.2016
[20] Yorick, 4 ways to increase performance of your unity game, http://www.paladinstudios.com/2012/07/30/4-ways-to-increase-performance-of-your-unity-game/, dostęp 12.09.2016
[21] http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/RigidBodyCollision.html, dostęp 14.09.2016
[22] http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/RigidBodyDynamics.html, dostęp 14.09.2016

JCSI 3 (2017) 64-69
Journal of Computer Sciences Institute

Wysłane: 2016-12-19
Przyjęte: 2016-12-21

Analiza porównawcza narzędzi e-learningu

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

Streszczenie. W artykule przedstawiono analizę porównawczą platform e-learningowych. Do analizy wybrano dwa typy platform: open source oraz komercyjne. Szczególną uwagę zwrócono na oferowane przez nie funkcjonalności oraz sposób obsługi przez nauczycieli akademickich. Ponadto w opracowaniu uwzględniono wyniki przeprowadzonej ankiety wśród nauczycieli akademickich czterech wybranych uczelni.

Słowa kluczowe: e-learning; platformy open source; platformy komercyjne;

*Autor do korespondencji.

Adres e-mail: weronika.przadka@pollub.edu.pl

Comparative analysis of e-Learning tools

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

Abstract. The article presents a comparative analysis of e-learning platforms. For the analysis selected two types of platforms: open source and commercial. Particular attention has been paid to the functionality offered by them. In addition, the study takes into account the results of the survey among teachers in four selected schools.

Keywords: e-learning; open source platforms; commercial platforms

*Corresponding author.

E-mail address: weronika.przadka@pollub.edu.pl

1. Wstęp

Internet w dzisiejszych czasach jest nieodzownym elementem codziennego życia. Większość użytkowników nie wyobraża sobie bez niego kolejnego dnia. Jednak nie zawsze tak było. Jeszcze kilkadziesiąt lat temu nikt nie słyszał o czymś takim jak Internet, nie wyobrażał sobie także jaki będzie miał zasięg. Historia Internetu sięga końca lat 60, kiedy to nawiązano pierwsze połączenie pomiędzy dwoma punktami nowej sieci ARPANET. Miało to miejsce 29 października 1969 [1].

1.1. Historia e-learningu

Internet jest potężnym narzędziem wykorzystywanym do wielu celów, nie tylko społecznościowych ale również naukowych. Jednym z takich właśnie celów jest zdalne nauczanie zwane e-learningiem. E-learning jest starszy nawet od Internetu. Na grafice widocznej na Rys.1 przedstawiona została historia e-learningu. Już w roku 1924 Sidney Pressey wynalazł pierwszą maszynę do testów: „Testing machine". Równo 30 lat później powstała pierwsza maszyna do nauczania, która miała na celu pomoc uczniom w nauce. W 1960 roku stworzono pierwszą platformę e-learningową: PLATO (ang. Programmed Logic for Automatic Teaching Operations). Sześć lat później profesorowie z Uniwersytetu Stanforda zaczęli nauczać dzieci matematyki oraz czytania za pomocą komputerów. Jak zostało wcześniej wspomniane w roku 1969 zaczął rozwijać się ARPANET. Projekt ten miał służyć celom akademickim oraz wojskowym. Powstał wtedy znany dziś protokół TCP/IP i do ARPANET'u dołączyły lokalne sieci akademickie. Przez kolejne 10 lat miało miejsce wiele włamań do serwerów ARPANET'u, dlatego też w 1980 roku postanowiono o rozdzieleniu części akademickiej od wojskowej. Część wojskowa dalej nosiła nazwę ARPANET, natomiast część akademicka nazwana została Internetem. Istotny wpływ na rozwój e-learningu miała era komputerów osobistych oraz społeczności internetowych dzielących się informacjami (lata 80) oraz digital natives (lata 90). Od roku 2000 e-learning zaczął być używany do szkoleń pracowników. Łatwo dostępne stały się narzędzia do tworzenia kursów oraz nauki online. Od roku 2010 do nauki online wykorzystywane są portale społecznościowe, takie jak np.: YouTube, Facebook, Twitter, a także komunikatory, np. Skype.

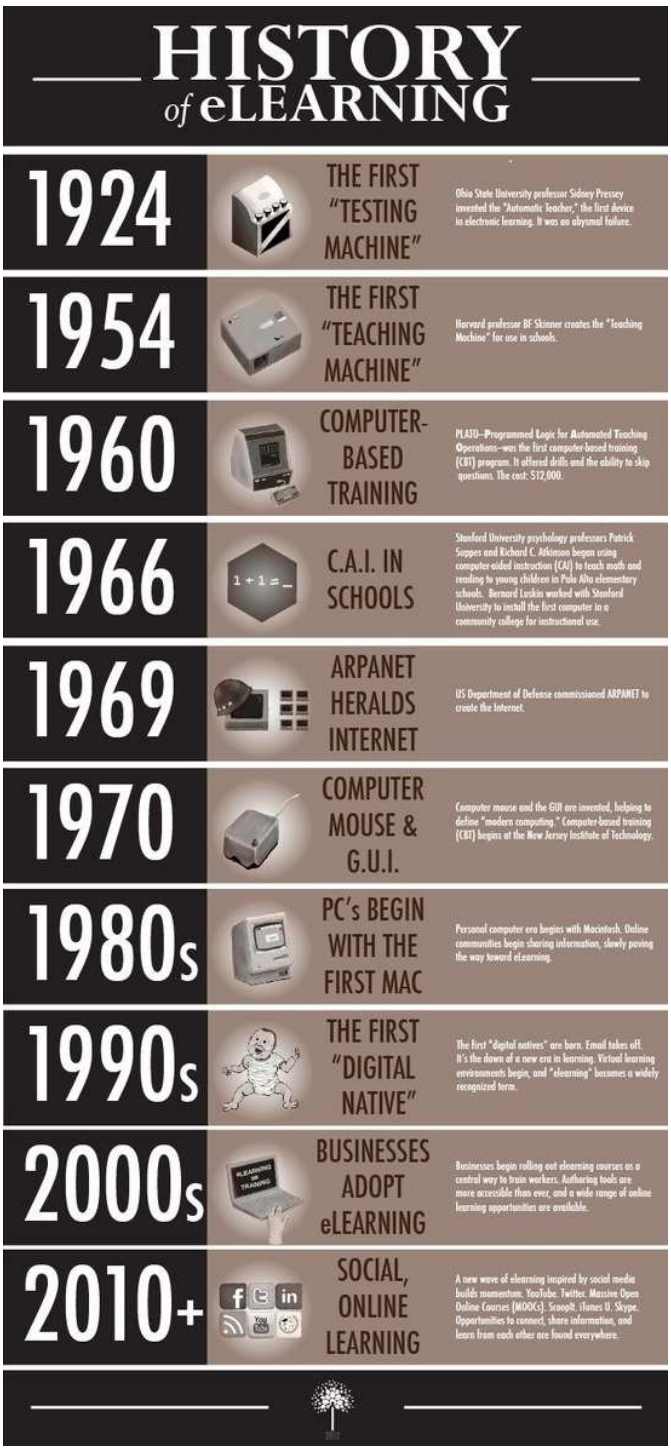
1.2. Podział oraz przegląd platform e-learningu

Obecnie na rynku dostępnych jest wiele platform e-learningowych. Dzielą się one na kilka grup. Najważniejszym z podziałów jest podział na platformy:

- płatne (komercyjne),
- darmowe (open source),
- tworzone indywidualnie [3].

Platformy komercyjne są to platformy, za wykorzystywanie których należy zapłacić. Ich kod źródłowy nie jest nigdzie udostępniony. Charakterystyczną cechą jest wsparcie techniczne producenta oraz możliwość modyfikacji platformy na potrzeby użytkownika na podstawie umowy licencyjnej. Istnieje możliwość zawężenia lub rozszerzenia funkcjonalności danej platformy, co jest zależne od wymagań

kupującego oraz budżetu. Systemy tego typu są wdrażane przez techników z firmy produkcyjnej. Jeśli chodzi o zdalne nauczanie na uczelniach wyższych prym wiodą platformy Fronter oraz Blackboard.



Rys. 1. Ikonografika przedstawiająca historię e-learningu [2]

Platformy open source są to platformy darmowe, ich kod źródłowy jest ogólnodostępny i może być dowolnie modyfikowany. Na początku były głównie wykorzystywane do zapoznania się ogólnie z e-learningiem i funkcjonalnościami oferowanymi przez platformy. Obecnie stosowane są na równi z platformami komercyjnymi. Do wdrożenia systemów tego typu przeważnie wykorzystywane są firmy zajmujące się wdrażaniem platform edukacyjnych, co wiąże się z kosztami. Do najpopularniejszych narzędzi open source zalicza się: Moodle, Ilias, eFront.

Platformy tworzone indywidualnie są to najczęściej platformy tworzone na potrzeby konkretnej uczelni. Uczelnie wykorzystują je jako wsparcie w procesie nauczania. Jednakże systemy te mogą służyć do realizacji zajęć na odległość. Przykładami platform tego typu są m.in.: platformy: „SAS" (System Administrowania Studiami) w Ośrodku Kształcenia na Odległość OKNO Politechniki Warszawskiej, „Edu" w Polsko-Japońskiej Wyższej Szkoły Techniki Komputerowych w Warszawie oraz „CENO PG" (Centrum Edukacji na Odległość Politechniki Gdańskiej) [4, 5].

1.3. Cel i obszary badań

Celem badań było porównanie wybranych platform e-learningowych. Obecnie systemów takich jest bardzo dużo. Ze względu na podobny zakres oferowanych funkcjonalności potencjalny użytkownik może mieć problem z wyborem odpowiedniego narzędzia.

Do badań wybrano platformy z dwóch najpopularniejszych grup: komercyjne oraz open source. Z każdego typu wybrano po dwie platformy. Wśród systemów komercyjnych są to: Blackboard oraz Fronter, systemy open source: Moodle, eFront. Platformy zostały zbadane pod względem funkcjonalności oferowanych dla realizacji dydaktyki na uczelniach wyższych. W tym obszarze skupiono się głównie na danych podanych przez producenta, wykorzystano dokumentacje techniczne platform oraz wersje demonstracyjne systemów, jeśli takowe istnieją i są dostępne. Kolejnym badanym obszarem była użyteczność platform dla wykładowców uczelni wyższych.

1.4. Hipotezy badawcze

Ważnym elementem w każdej pracy badawczej jest określenie hipotez badawczych. W pracy magisterskiej „Analiza porównawcza narzędzi e-learningu", na której oparty jest ten artykuł postawiono hipotezę główną. Mówiła ona, że wszystkie platformy spełniają oczekiwania nauczycieli je wykorzystujących. Do tej hipotezy postawiono dwie hipotezy dodatkowe:

1) W zakresie oferowanych możliwości oraz sposobu obsługi platformy open source i komercyjne nie różnią się znacząco między sobą.
2) Nauczyciele akademiccy ograniczają zakres wykorzystanie platform do umieszczania na nich zasobów statycznych, takich jak np.: pliki, prezentacje, testy. Rzadko wykorzystują inne oferowane funkcjonalności, jak np.: fora, czaty.

2. Obiekt badań

Obiektem badań były platformy e-learningowe komercyjne oraz open source. Z każdego typu wybrano do analizy po dwie paltformy. Platformy komercyjne: Blackboard oraz Fronter, open source: Moodle, eFront.

Platforma Blackboard jest platformą komercyjną, co oznacza, że należy wykupić licencję na użytkowanie systemu. Stworzona została w firmie Blackboard Inc. Miała za zadanie wspomóc nauczanie, obecnie wykorzystywana jest do nauczania online. W roku 2006 firma przejęła prawa do bardzo popularnej w tamtym czasie platformy WebCT i zmieniła jej nazwę na Blackboard. System może być instalowany na serwerach lokalnych, jak również na serwerach udostępnionych przez producenta – zalecane. Oparty jest na sieci Web, dzięki czemu możliwa jest integracja z innymi systemami działającymi w obrębie danej uczelni [6]. Platforma wykorzystywana jest m.in. na Uniwersytecie w Białymstoku.

Platforma Fronter jest kolejną platformą komercyjną wybraną do badania. Stworzona przez norweską firmę FRONTER AS. Obecnie system wykorzystywany jest przez około 217000 osób [7]. Użytkownicy mają możliwość wyboru modułów oraz funkcjonalności i zapłacić tylko za te wybrane, a nie jak w przypadku innych platform za całość. System może być zintegrowany z różnymi zewnętrznymi aplikacjami, jak np.: „Ephorus" czy „Creaza". Wykorzystywany jest m.in. na Wyższej Szkole Kultury Społecznej i Medialnej w Toruniu.

Platforma Moodle jest platformą darmową, działa na licencji open source. Jej kod źródłowy jest znany i ogólnodostępny. Od roku 2003 istnieje witryna poświęcona w całości temu systemowi. Można tam znaleźć m.in. wsparcie techniczne oraz listę firm specjalizujących się we wdrażaniu platformy. Dzięki czemu użytkownik, pomimo braku wsparcia technicznego producenta, ma możliwość uzyskania pomocy od doświadczonych administratorów platformy [8]. Wykorzystywana jest na uczelniach wyższych, m.in. na Politechnice Lubelskiej oraz UMCS w Lublinie [9].

Platforma eFront, jest kolejną darmową platformą wybraną do badania. Twórcy systemu podczas projektowania oraz tworzenia platformy nie skupiali się wyłącznie na funkcjonalnościach ale również na interfejsie platformy. Pragnęli aby platforma była jak najbardziej przystępna oraz łatwa w obsłudze dla użytkowników. Dostępna jest w ponad 40 językach, w tym również w polskim [10]. Stosowana jest m.in. na WSP w Łodzi oraz na Uniwersytecie Łódzkim [9].

3. Metody i przebieg badań

Do badań wybrano 4 platformy oraz 4 uczelnie wyższe. Systemy zostały wybrane losowo. Natomiast wpływ na wybór uczelni miało to z jakiego systemu e-learningowego korzysta dana uczelnia. Ważne było, aby były to systemy wybrane wcześniej oraz aby się nie powtarzały.

Przygotowania do badań rozpoczęto już w czerwcu bieżącego roku. Przez około 2 miesiące próbowano uzyskać dostęp do wersji demonstracyjnych systemu oraz zebrać jak największą ilość materiału. W październiku bieżącego roku zaczęto badania nad platformami od strony dokumentacji technicznej oraz dostępnych wersji demonstracyjnych. W grudniu rozesłana została ankieta do wykładowców. Łącznie rozesłano ankiety do około 300 osób. Przyjęto za najbardziej prawdopodobne, że z platform e-learningowych w większości korzystają wydziały, jednostki związane z kierunkiem Informatyki. Dlatego też szukano wykładowców, laborantów z tych wydziałów. Adresy e-mailowe zostały pobrane ze stron poszczególnych uczelni:

- Uniwersytet Łódzki - http://www.math.uni.lodz.pl/lista-pracownikow/
- Politechnika Lubelska - https://ehms.pollub.pl/staff.php?step=1
- Uniwersytet w Białymstoku - http://matinf.uwb.edu.pl/pl/wydzial/kadra.php
- Wyższa Szkoła Kultury Społecznej i Medialnej w Toruniu - https://www.mat.umk.pl/web/wmii/wydzial/nauczyciele-akademiccy

W celu poprawnego przeprowadzenia badań należało dobrać odpowiednie metody badawcze. Wybrane zostały dwie metody: porównawcza oraz ankietowa. Metoda porównawcza posłużyła do porównania platform komercyjnych i open source ocenianych dla realizacji dydaktyki na uczelniach wyższych. Wskazała najlepsze cechy każdej z badanych platform oraz ich wady. Metoda ankietowa posłużyła do porównania użyteczności platform dla wykładowców.

W pierwszym kroku dokonano porównania platform na podstawie danych producenta. Głównie wykorzystana została dostępna dokumentacja techniczna. Do platform darmowych wykorzystano także ich wersje demonstracyjne. W początkowych założeniach było wykorzystanie wersji demo dla wszystkich platform, jednakże uzyskanie dostępu do wersji platform komercyjnych było niemożliwe.

Porównano platformy pod kątem przede wszystkim oferowanych funkcjonalności. Zwrócono uwagę na dostępność oraz jakość dokumentacji oraz wsparcia technicznego. Ważnym punktem w badaniu było określenie jakości oraz łatwości korzystania z interfejsu, jego budowa. Ponad to zweryfikowano możliwości sprawdzania wiedzy studentów, tutaj głównie jej typy, ilość dostępnych typów pytań do wyboru, prostotę tworzenia kursów. Określono sposób oceniania uczniów, raportowanie ocen oraz wyników. Zwrócono również uwagę na elementy społecznościowe platform: fora, czaty oraz sposoby komunikacji.

Drugim krokiem było stworzenie oraz rozesłanie ankiet mających na celu zebranie jak największej ilości odpowiedzi na temat użyteczności platform dla korzystających z nich wykładowców. Rozesłano ankiety do około 300 osób z 4 wybranych uczelni. Ankietowani dostali tydzień czasu na udzielanie i odsyłanie wypełnionych prac. Stosunek wypełnionych ankiet do rozesłanych wynosi ponad 16%, co jest słabym wynikiem. Ankietowani odpowiadali na pytania w stosunku do konkretnej platformy, którą zaznaczyli w pierwszym pytaniu. W ankiecie skupiono się głównie na wykorzystaniu platformy przez nauczycieli akademickich oraz na ocenie przez nich poszczególnych elementów, jak np.: dokumentacja, wsparcie techniczne, interfejs. Większość pytań w ankiecie była typu zamkniętego, pytanie otwarte zostało zastosowane do pytania 5, w którym ankietowani mogli wpisać brakujące ich zdaniem funkcjonalności. Pytanie

to nie zostało oznaczone jako obowiązkowe, jednak zebrano tutaj kilka odpowiedzi.

Wyniki badań zostały przedstawione w następnym rozdziale.

4. Wyniki badań

W tabeli 1 przedstawione zostało zestawienie najważniejszych cech i funkcjonalności badanych platform na podstawie danych producenta.

Na podstawie zestawienia ukazanego w tabeli 1 można stwierdzić, że zarówno platformy open source jak i komercyjne oferują podobne funkcjonalności. Mogą się one różnić jedynie ilością elementów, czy sposobem obsługi. Do każdego systemu dołączona jest dokumentacja, często wzbogacona o dodatkowe materiały, jak np.: filmy szkoleniowe oraz instruktażowe, prezentacje czy zrzuty ekranów z danych modułów. Producenci platform komercyjnych oferują także wsparcie techniczne do swoich produktów. Powoli wchodzi to również do platform open source, gdzie do tej pory nie było takich udogodnień. Wsparcie to może przyjmować różne formy, od gorących linii telefonicznych po forum, czaty. Dzięki takim rozwiązaniom szybko można uzyskać pomoc. Na przykład dla systemu Moodle zostało stworzone forum skupiające wiele osób służących jako wsparcie dla użytkowników tej platformy. Co prawda nie są to osoby zatrudnione przez firmy produkujące platformę, ale zawsze jest to jakaś pomoc. Jest to duży plus dla tej platformy.

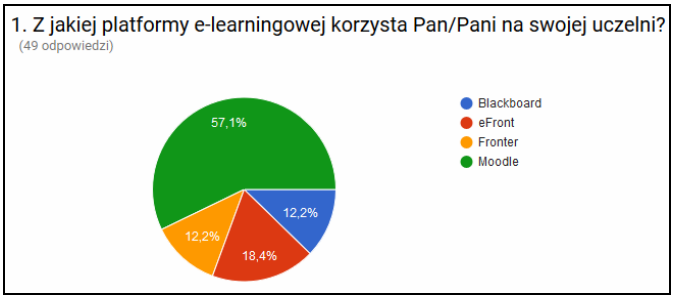
Tabela 1. Zestawienie najważniejszych cech i funkcjonalności badanych platform

Cechy/ Funkcjonalności ↓	**Platforma →**	**Blackboard**	**eFront**	**Fronter**	**Moodle**
Licencja		Komercyjna	Open source	Komercyjna	Open source
Wersja demonstracyjna systemu		Brak dostępu	Ogólnodostępna (tylko jęz. angielski)	Brak dostępu	Ogólnodostępna (także jęz. polski)
Interfejs		Brak informacji	Prosty, niewielka ilość elementów	Przejrzysty, intuicyjny	Budowa modułowa, intuicyjny
Administracja kursami		Instruktor	Instruktor	Użytkownik z uprawnieniami kreatora lub administrator	Tylko administrator
Dokumentacja		Instrukcje bez screenów, filmy instruktażowe	Instrukcje ze screenami, brak filmów	Instrukcje ze screenami, filmy instruktażowe	Instrukcje ze screenami, brak filmów
Dokumentacja w jęz. polskim		Tak	Brak	Brak	Tak
Sprawdzanie wiedzy studentów		Testy, quizy	Testy	Testy	Testy
Typy pytań (ilość)		17, np: • quiz, • tak/nie, • dopasowanie, • skala,	12, np: • tak/nie, • przeciągnij i upuść, • tekstowa,	7, np: • tak/nie, • dopasowanie, • krótka, długa odpowiedź,	8, np: • tekstowa, • dopasowanie, • jednowyborowa,
Możliwość tworzenia ankiety		Tak	Brak informacji	Tak	Tak
Oceny od razu po wypełnieniu testu		W zależności od decyzji instruktora	Tak	W zależności od typu testu	W zależności od typu testu
Raporty		Brak informacji	Tak	Tak	Tak
Fora		Tak (w tym forum tylko dla studentów)	Tak (dla każdego kursu oddzielne)	Tak (5 typów)	Tak (wspólne dla instruktorów i studentów)
Ocenianie odpowiedzi na forum		Brak	Brak	Tak, po przyznaniu praw przez administratora	Brak
Funkcjonalności dostępne dla użytkowników		W zależności do decyzji władz uczelni	Wszystkie	Wszystkie	Wszystkie
Narzędzie do wykrywania plagiatu		"SafeAssign"	Brak	Brak	Brak
Dodatkowe funkcjonalności/wtyczki (ilość)		Tak	Brak informacji	Brak	Tak

Wysłane: 2016-12-19
Przyjęte: 2016-12-21

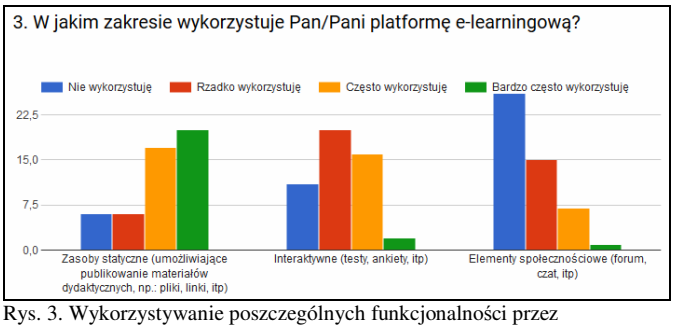
Wyniki przeprowadzonej ankiety ukazały duże zróżnicowanie jeśli chodzi o wymagania użytkowników konkretnej platformy.

Na rysunku 2 przedstawiony został wykres kołowy prezentujący procentową ilość odpowiedzi dotyczącą konkretnych platform w stosunku do wszystkich odpowiedzi.



Rys. 2. Procentowe wykorzystanie platform w stosunku do wszystkich odpowiedzi

Do najczęściej wykorzystywanych funkcjonalności platform e-learningowych według ankiety należą zasoby statyczne, które umożliwiają publikowanie materiałów dydaktycznych. Ponad połowa ankietowanych wykorzystuje je bardzo często. Drugą pod względem częstości użytkowania okazała się być funkcjonalność związana ze sprawdzaniem wiedzy studentów. Najrzadziej wykorzystywane są elementy komunikacji ze studentami, aż 26 spośród ankietowanych nie wykorzystuje tej funkcjonalności. Zestawienie wykorzystania poszczególnych funkcjonalności zostało przedstawione na rysunku 3.



Rys. 3. Wykorzystywanie poszczególnych funkcjonalności przez wykładowców

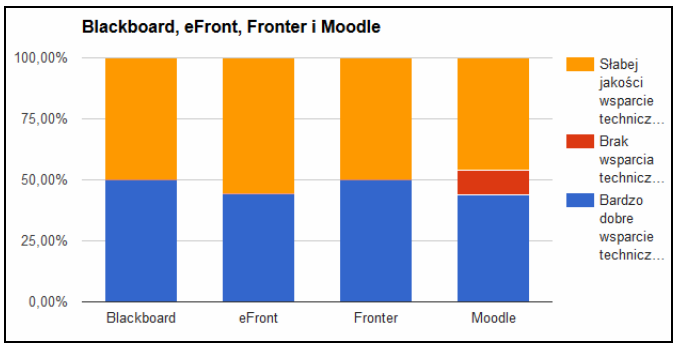
Użytkownicy platformy Moodle wskazali następujące funkcjonalności jako brakujące w systemie:

- możliwość egzekwowania czasu spędzanego przy nauce,
- kolorowanie składni zamieszczanych kodów źródłowych w różnych językach programowania,
- możliwość bezpośredniego wpisywania zadań matematycznych zawierających np. całki czy pierwiastki,
- ocenianie wielu studentów jednocześnie,
- dziennik ocen.

Użytkownicy pozostałych 3 platform nie udzielili odpowiedzi na to pytanie.

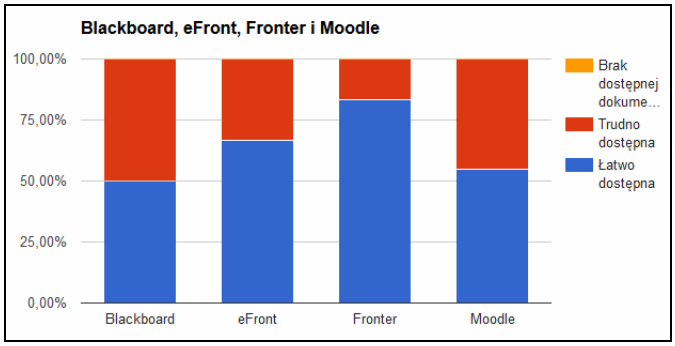
Ważnym punktem w ankiecie były pytania dotyczące wsparcia oraz dokumentacji technicznej.

Jeśli chodzi o wsparcie dla badanych platform wyniki ankiety wskazują na podział prawie pół na pół odpowiedzi pozytywnych oraz negatywnych dla każdej platformy. Tylko w przypadku platformy Moodle około 10% użytkowników wskazało na brak takiego wsparcia. Podział ten może wskazywać na to, że nie każdy użytkownik w przypadku korzystania ze wsparcia otrzymał zadowalające odpowiedzi. Wyniki zostały zaprezentowane na wykresie na rysunku 4.



Rys. 4. Ocena wsparcia technicznego przez wykładowców

Inaczej ma się sytuacja z dokumentacją techniczną, tutaj wyniki już nie są tak jednoznaczne. Żaden z ankietowanych nie wskazał braku dokumentacji. Według zestawienia zbiorczego odpowiedzi najlepiej pod tym względem wypadła platforma Fronter, gdzie ponad 75% ankietowanych wskazało jej dokumentację jako łatwo dostępną i dobrej jakości. Na drugim miejscu znalazła się platforma eFront, gdzie odpowiedzi takiej udzieliło ponad 60% pytanych. Pozostałe dwie platformy znajdują się niemalże egzekwo na ostatnim miejscu, tutaj około połowa respondentów wskazała taką odpowiedź. Zbiorcze zestawienie ocen zostało zaprezentowane na rysunku 5.



Rys. 5. Zbiorcze zestawienie ocen dokumentacji technicznej

W tabeli 2 zostały zaprezentowane funkcjonalności wykorzystywane przez wykładowców oraz procent użytkowników, którzy zaznaczyli użytkowanie danego modułu. Dane w tabeli posortowane są według procentowego użycia.

Tabela 2. Funkcjonalności oraz procent ich wykorzystania

Funkcjonalność	**Procent użytkowników**
Publikowanie materiałów dydaktycznych (pdf, doc, itp)	84%
Publikowanie materiałów dydaktycznych (linki do zasobów internetowych)	70%
Publikowanie materiałów dydaktycznych (prezentacje multimedialne)	65%
Sprawdzanie wiedzy (wrzucanie zadań)	59%
Publikowanie materiałów dydaktycznych (rysunki, zdjęcia, schematy, wykresy, itp)	55%
Sprawdzanie wiedzy (testy)	55%
Publikowanie ogłoszeń	45%
Kontakt ze studentami (forum)	39%
Publikowanie filmów dydaktycznych	29%
Ankiety	26%
Kontakt ze studentami (e-mail)	24%
Sprawdzanie wiedzy (inne)	16%
Kontakt ze studentami (czat)	8%

5. Wnioski

Celem pracy była analiza porównawcza narzędzi e-learningu. Na podstawie wybranych metod badawczych udało się potwierdzić wszystkie hipotezy postawione w pracy.

Za pomocą metody porównawczej udało się udowodnić hipotezę 1. Platformy open source nie ustępują platformom komercyjnym pod względem funkcjonalności. Często nie ma większych różnic w sposobie obsługi systemów. Producenci platform oferują bogate dokumentacje techniczne często wzbogacone o dodatkowe materiały. Coraz częściej można też spotkać wsparcie techniczne nie tylko dla platform komercyjnych, ale również darmowych.

Metoda ankietowa pozwoliła udowodnić hipotezę drugą. Wykładowcy w głównej mierze wykorzystują platformy e-learningowe do umieszczania na nich zasobów statycznych: wszelkiego rodzaju plików, filmów, zadań. Najmniej użytecznymi funkcjonalnościami okazały się być te związane z komunikacją. Nauczyciele akademiccy nie korzystają z forum, czatu czy grup dyskusyjnych. W przypadku konieczności poinformowania studentów o jakimś wydarzeniu, np.: teście bądź nieobecności korzystają z możliwości umieszczenia komunikatu na stronie głównej. Rzadko także wykorzystują oni możliwości interaktywnej nauki, czasami proces stworzenia testu na platformie jest kłopotliwy lub pracochłonny.

Przeprowadzone badania pokazały, że wszystkie platformy e-learningowe spełniają znaczną część wymagań wykorzystujących je wykładowców. Niemożliwe jest dobranie systemu tak, aby spełniał wszystkie oczekiwania wszystkich użytkowników.

Literatura

[1] Jak powstał internet? Historia prawdziwa, http://softonet.pl/publikacje/poradniki/Jak.powstal.internet.Historia.prawdziwa,1192 [16.12.2016].

[2] Gogos R.: A brief history of elearning (infographic), https://www.efrontlearning.com/blog/2013/08/a-brief-history-of-elearning-infographic.html [16.12.2016].

[3] Clarke A.: E-learning nauka na odległość. Wydawnictwo Komunikacji i Łączności, 2007.

[4] Chrabąszcz K.: Using e-learning in the process of students' education, Research Papers Collection, 2011, Nr 1(17), s. 55-66.

[5] Kopciał P.: Analiza metod e-learningowych stosowanych w kształceniu osób dorosłych. Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki, 2013, Nr 9, Rok 7, s. 79-99.

[6] Bradford P., Porciello M., Balkon N., Backus D.: The Blackboard Learning System. The Journal Of Educational Technology Systems 35:301-314, UUP Research Working Papers, 2007.

[7] O Platformie Fronter, https://www.pol.fronter.com/o-nas/ [16.12.2016].

[8] Brzózka P.: Moodle dla nauczycieli i trenerów. Helion, 2011.

[9] Kuźmicz K.: Rzecz o edukacyjnym zastosowaniu oprogramowania klasy Open Source, czyli przegląd bezpłatnych platform edukacji wirtualnej. WWW w sieci metafor: strona internetowa jako przedmiot badań naukowych (s.258–273), Wydawnictwo Naukowe Dolnośląskiej Szkoły Wyższej, 2008.

[10] Strona domowa platformy eFront, http://efront.pl/dlaczego-efront.html [16.12.2016].