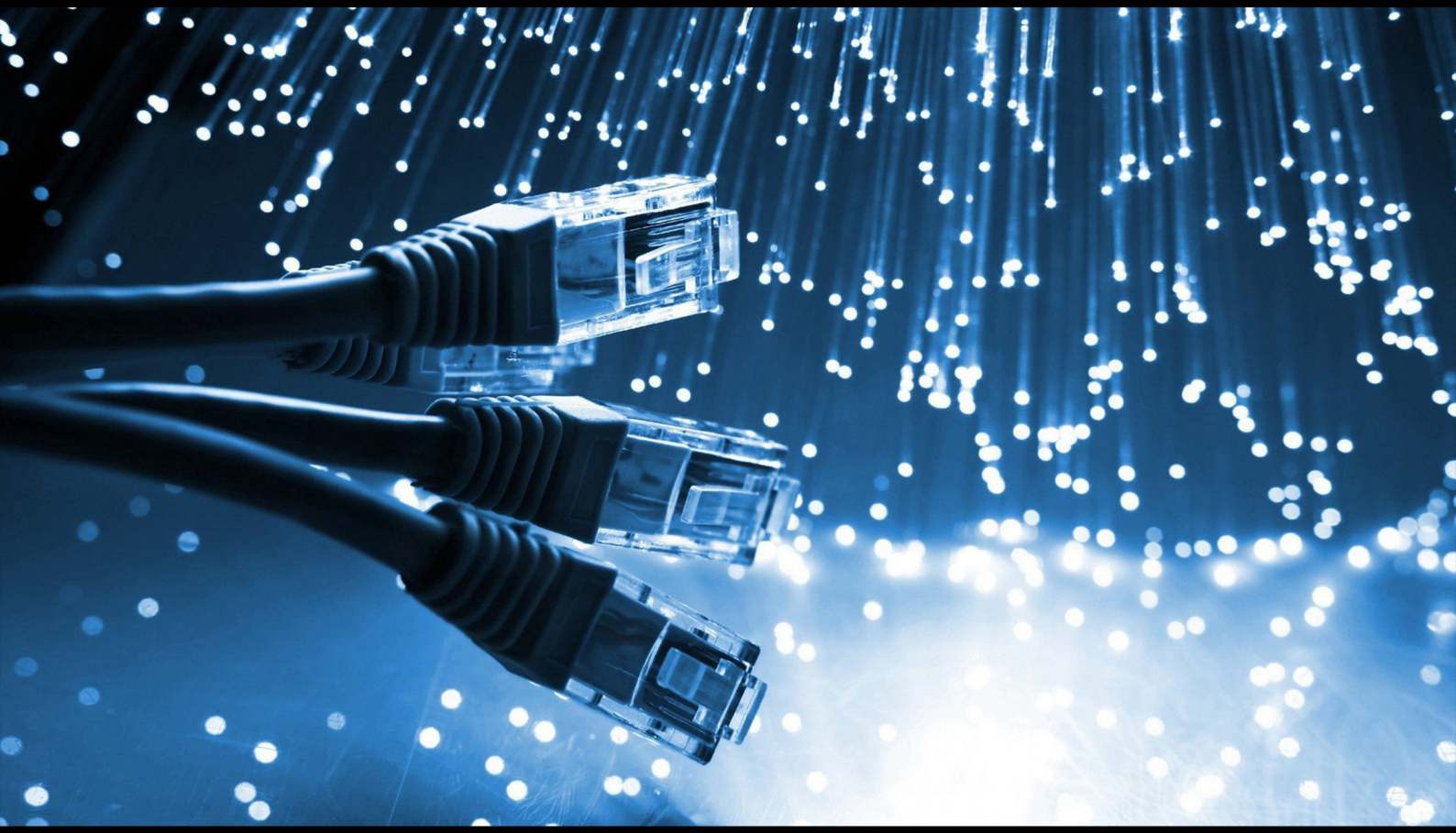
JCSI

Journal of Computer Sciences Institute

Volume 2/2016



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. Elżbieta Miłosz
dr hab. inż. Dariusz Czerwiński, prof. PL
dr inż. Jacek Kęsik
dr Edyta Łukasik
dr inż. Dariusz Gutek
dr inż. Maria Skublewska-Paszkowska
dr Beata Pańczyk
dr Mariusz Dzieńkowski
dr inż. Tomasz Szymczyk
dr inż. Maciej Pańczyk
dr inż.. Grzegorz Kozieł
dr inż. Jakub Smołka

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:
Elżbieta Miłosz
Dariusz Czerwiński
Jacek Kęsik
Edyta Łukasik
Dariusz Gutek
Maria Skublewska-Paszkowska
Beata Pańczyk
Mariusz Dzieńkowski
Tomasz Szymczyk
Maciej Pańczyk
Grzegorz Kozieł
Jakub Smołka

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

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JCSI 2 (2016) 60-63
Journal of Computer Sciences Institute

Wysłane: 2016-10-10
Przyjęte: 2016-10-13

Opracowanie i testowanie aplikacji utworzonych z użyciem platform Node.js oraz Laravel

Vasyl Bomba*, Edyta Łukasik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem pracy była analiza i badanie programowej platformy Node.js i frameworka Laravel, a także porównanie wydajności serwisów utworzonych za pomocą tych technologii. Testy wydajności aplikacji przeprowadzono za pomocą oprogramowania Apache Benchmark i Apache JMeter. Analiza obejmuje teoretyczne i praktyczne aspekty ich funkcjonowania.

Słowa kluczowe: Node; Node.js; Laravel; PHP; MongoDB; benchmark

*Autor do korespondencji.

Adres e-mail: bombavasyl@gmail.com

Development and testing of applications created with the platforms Node.js and Laravel

Vasyl Bomba*, Edyta Łukasik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of this work was the analysis and research of software platform Node.js and the framework Laravel, and comparing the performance of the created sites using Node.js and PHP. The tests were carried out with Apache Benchmark and Apache JMeter. The analysis includes theoretical and practical aspects of their functioning.

Keywords: Node; Node.js; Laravel; PHP; MongoDB; benchmark.

*Corresponding author.

E-mail address: bombavasyl@gmail.com

1. Wstęp

Celem artykułu jest analiza i badanie platformy programowej Node.js i frameworka Laravel oraz porównanie wydajności serwisów utworzonych za pomocą tych narzędzi. Analiza obejmuje teoretyczne aspekty ich funkcjonowania, a także badania wydajności aplikacji.

Do tworzenia i testowania aplikacji internetowych używane są następujące narzędzia:

- platforma programowa Node.js;
- framework Laravel;
- serwer MySQL;
- MongoDB;
- Bootstrap;
- Apache Benchmark;
- Apache JMeter.

2. Zastosowane technologie

Node (lub Node,js) – to nowoczesna platforma, która wyświetla język JavaScript poza przeglądarką i pozwala na stosowanie go w aplikacjach serwerowych. Platforma jest oparta o wyjątkowo szybki silnik JavaScript, zapożyczony z przeglądarki Chrome, V8, do której dodana została szybka i niezawodna biblioteka asynchronicznego sieciowego wejścia/wyjścia. Główny nacisk w Node położony jest na tworzenie wysokiej jakości, dobrze skalowalnych klienckich i serwerowych aplikacji webowych [1].

Node.js działa na zasadzie asynchronicznego modelu, który zbudowany jest jak kolejka zdarzeń (event loop). W przypadku wystąpienia jakiegoś zdarzenia (przyjście żądania, odczyt pliku, odpowiedź na zapytanie z bazy danych), zostanie ono umieszczone na końcu kolejki. Strumień, który obsługuje tę kolejkę, bierze zdarzenie z początku kolejki i wykonuje związany z tym wydarzeniem kod. Dopóki kolejka nie jest pusta procesor jest zajęty pracą [2].

Laravel jest to darmowy framework, przeznaczony dla tworzenia projektów z wykorzystaniem architektury modelu MVC (ang. Model View Controller) — model-widok-kontroler).

Personal HyperText Processor (PHP) jest to skryptowy język programowania, używany po stronie serwera do dynamicznego generowania stron HTML. PHP różni się od JavaScript tym, że skrypty PHP są wykonywane na serwerze i generują kod HTML, który jest wysyłany do klienta. Zawiera on wszystkie funkcje potrzebne do pracy po stronie serwera. PHP to jeden z niewielu języków programowania, stworzonych specjalnie do wytwarzania aplikacji internetowych. Użycie języka PHP wraz ze środowiskiem Apache jako serwerem WWW oraz bazą danych MySQL określa się nazwą platforma AMP. Są to elementy niezbędne do stworzenia funkcjonalnej aplikacji internetowej[4].

W wyniku ankiety sitepoint.com w grudniu 2013 roku o najbardziej popularnych frameworkach PHP Laravel zajął miejsce najbardziej obiecującego projektu na 2014 rok.

W 2015 roku w wyniku ankiety sitepoint.com dotyczącej wykorzystania frameworków PHP wśród programistów, PHP zajął pierwsze miejsce w kategoriach:

- Framework klasy korporacyjnej;
- Framework dla projektów osobistych [3].

3. Opis aplikacji

Zostały utworzone dwie aplikacje realizujące wymianę informacji tekstowej pomiędzy osobami. Jedna z nich używa Node.js, a druga frameworka Laravel i języka programowania skryptowego PHP. Mają one bardzo podobną funkcjonalność:

- zapisywanie wiadomości w bazie danych;
- autoryzacja użytkownika;
- przesyłanie wiadomości;
- powiadomienie podłączonych klientów, gdy ktoś wyszedł z czatu lub wszedł na czat;
- lista użytkowników online czatu;
- połączenie z serwerem za pomocą WebSocket.

Każda z nich dostosowuje się wyglądem strony do rozmiaru okna przeglądarki, w której jest uruchamiana.

Aby wejść na czat konieczne jest przejście przez narzędzie autoryzacji. Zrzut ekranu tego okna pokazano na rys. 1. Należy w nim wypełnić 2 pola: login i hasło. W przypadku, gdy użytkownik pozostawi jedno z pól puste, pojawia się komunikat o błędzie. Jeśli użytkownika o podanym loginie nie istnieje w bazie danych, zostanie on automatycznie rejestrowany w niej z wprowadzonymi danymi.

Rys. 1. Autoryzacja na czacie Node.js

Po tym, jak użytkownik pomyślnie przeszedł autoryzację pojawia się okno czatu pokazane na rys. 2.

W oknie czatu po lewej stronie dostępny jest spis otrzymanych wiadomości, które uporządkowane są chronologicznie według pojawiania się. Po stronie prawej okna znajduje się lista aktywnych w danym momencie czasu użytkowników. Oprócz tego wyświetlana jest informacja z licznikiem użytkowników zalogowanych do aplikacji w danej chwili. W dolnej części okna aplikacji jest miejsce na wpisanie informacji tekstowej, która ma być przesłana. Wysłanie odbywa się po wciśnięciu klawisza „*enter*”. W prawym dolnym rogu znajduje się przycisk umożliwiający zamknięcie czatu.

Rys. 2. Główne okno czatu Node.js

W stworzonych aplikacjach pojawia się dźwięk informujący o otrzymaniu wiadomości.

4. Badania wydajności

Głównym zadaniem każdego testowania wydajności witryny jest poznanie jego odporności na obciążenia, które mogą pojawiać się nie tylko z powodu dużej ilości użytkowników podłączonych jednocześnie, ale i być konsekwencją nieprawidłowej konfiguracji serwera, nieprawidłowego działania skryptów lub ataków typu DOS.

Do testów został użyty komputer posiadający:

- Procesor: Intel Core i5-3210M @ 3.10 GHz (4 Cores);
- RAM: 8 GB.

Aby przeprowadzić testy z taką dużą ilością użytkowników należy zmienić ustawienia w systemie operacyjnym Ubuntu:

- zwiększenie limitu deskryptorów plików. Zmiana wartości opcji nofile w pliku limits.conf, co zostało pokazane na listingu 1;
- zmiana wartości parametru *net.ipv4.netfilter.ip_conntrack_max* w pliku *sysctl.conf,* co pokazano na listingu 2;
- zmiany ograniczeń systemowych na otwarte pliki - poprzez edycję parametru *ulimit -n*.

Listing 1. Zwiększenie limitu deskryptorów plików w pliku limits.conf

```
#/etc/security/limits.conf
nofile 1048576
```

Listing 2. Zwiększenie limitu deskryptorów plików w pliku sysctl.conf

```
#/etc/sysctl.conf
net.ipv4.netfilter.ip_conntrack_max = 1048576
```

Apache Benchmark jest to jednowątkowa aplikacja wywoływana z wiersza poleceń służąca do pomiaru wydajności HTTP serwerów WWW. Pierwotnie opracowany został on do testowania serwera HTTP Apache, ale w zasadzie nadaje się do testowania dowolnego serwera WWW. Narzędzie to jest dostarczane ze standardową dystrybucją

Apache i jak sam Apache, jest darmowe i rozpowszechniane na Licencji Apache [5].

Do zbierania informacji o ustawieniach serwera takich jak CPU i RAM, był zainstalowany moduł node-millenium.

Serwery były testowane przez 20 sekund dla różnej liczby HTTP GET żądań. Żądania te były generowane poprzez Apache Benchmark. Liczba użytkowników oraz czas testu były podawane jako parametry wejściowe dla przeprowadzanych testów. Wyniki testów z żądaniami przedstawiono w tabeli 1 i 2.

Tabela 1. Testy aplikacji napisanej z użyciem Node.js

Liczba użytkowników	Liczba żądań	Liczba żądań / sekunda	Średnie CPU [%]	RAM [MB]
1	135037	6751.85	27	24
5	325700	16284.9	84.6	29
10	340912	17045.6	91.7	37
100	321004	16049.2	92.4	59
300	305202	15256.9	91.4	91
500	301954	15093.9	90.3	105
1000	300325	15012.9	90.1	107
5000	289842	14489.4	78.9	168
10000	310211	15508.9	81.5	215
20000	308017	15399.7	86	232

Tabela 2. Testy aplikacji napisanej z użyciem PHP i Laravel

Liczba użytkowników	Liczba żądań	Liczba żądań / sekunda	Średnie CPU [%]	RAM [MB]
1	94678	4733.9	33	26
5	261759	13087.9	95.7	31
10	263042	13152.1	97.7	80
100	255957	12797.2	97.6	86
300	248093	12404.1	94.6	140
500	239068	11952	93.3	151
1000	246103	12304.8	95.4	169
5000	228845	11441	92	194
10000	234412	11719.3	93.7	299
20000	191469	9573.42	92	330

Na podstawie przeprowadzonych badań zostały zbudowane porównawcze wykresy przedstawione na rys. 3. Na podstawie wyników można stwierdzić, że bardziej wydajny okazał się Node.js w zakresie zużywanych zasobów procesora oraz pamięci. Liczba obsłużonych żądań na sekundę dla każdej liczby użytkowników była większa dla aplikacji napisanej z użyciem Node.js. Dla liczby użytkowników od 1 do 5 ta różnica była niewielka, natomiast dla zakresu od 5 do 10000 wynosiła ona nie mniej niż 4000. Po przekroczeniu liczby 10000 użytkowników różnica ta zaczęła się powiększać, a dla 20000 użytkowników osiągnęła 6000.

Rys. 3. Liczba żadań dla HTTP GET żądań Node.js i PHP

Program Apache JMeter jest to narzędzie do przeprowadzenia testowania obciążenia, stworzone przez Apache Software Foundation. Choć początkowo program JMeter został zaprojektowany jako narzędzie do testowania aplikacji WWW, obecnie jest on w stanie przeprowadzić testy dla JDBC połączeń, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP i TCP [6].

Celem skorzystania z tego oprogramowania było zasymulowanie wysyłania wiadomości do wielu użytkowników jednocześnie. W ten sposób można było zbadać działanie aplikacji z bazą danych.

Ponieważ obciążenie serwera zapytaniami i równoległe przetwarzanie ich wyników programem Apache JMeter wymagała znacznych zasobów systemu, do tego celu użyto komputera o parametrach:

- CPU: Intel Core i7-4790K @4.00 GHz;
- RAM: 16 GB.

Wiadomości wysłane za pośrednictwem gniazd miały przekazywane parametry ustawione odpowiednio na: "type": "message" i "message": "test message". Czynności wykonywane były jednocześnie dla różnej ilości wirtualnych użytkowników w ciągu 20 sekund. Takie warunki maksymalnie zbliżyły testy do rzeczywistych warunków pracy aplikacji realizujących czaty.

Wyniki testów dla generowanych automatycznie wiadomości wysyłanych do wszystkich podłączonych użytkowników czatu przedstawiono w tabeli 3 i 4. Zarówno

liczba użytkowników jak i liczba wysyłanych wiadomości były ustawiane w programie Apache JMeter.

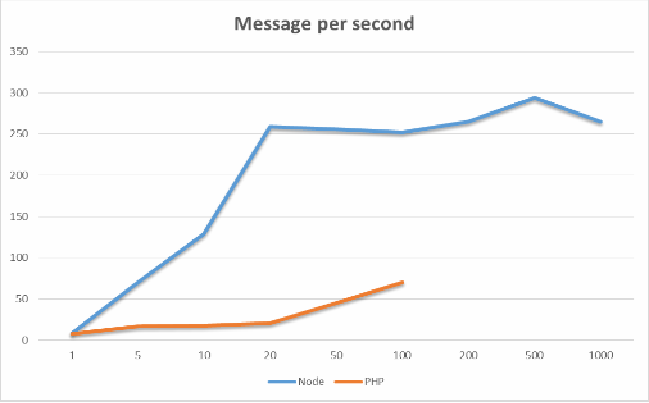
Tabela 3. Test wysyłania wiadomości - aplikacja z użyciem Node.js

Liczba użytkowników	Liczba wiadomości	Liczba żądań / sekunda	Błąd [%]	Średnie CPU [%]
1	184	9.2	0	11
5	1406	70.3	0.4	21
10	2586	129.3	1.7	26
20	5185	259.3	14.3	40
50	5137	256.9	23	42
100	5043	252.2	25.3	50
200	5301	265	24.7	58
500	5876	293.8	23.66	60
1000	5301	265	25.37	81

Tabela 4. Test wysyłania wiadomości - aplikacja z użyciem Lavarem

Liczba użytkowników	Liczba wiadomości	Liczba żądań / sekunda	Błąd [%]	Średnie CPU [%]
1	161	8.1	0	29
5	340	17	12.6	41
10	362	18.1	19	45
20	429	21.5	27	51
50	904	45.2	35.8	61
100	1399	70	46	63

Na podstawie wyników przeprowadzonych badań zostały zbudowane porównawcze wykresy, przedstawione na rys. 4.



Rys. 4. Liczba żądań na sekundę dla aplikacji w Node.js i PHP

Wynik testu pokazał, że aplikacja realizująca czat napisana z użyciem Node.js radzi sobie z wiadomościami kilka razy szybciej niż aplikacja napisana w wykorzystaniem frameworka Laravel i języka PHP. Przy ilości użytkowników powyżej 100, serwer aplikacji, napisany przy użyciu PHP i Laravel, nie radzi sobie z takim obciążeniem. Serwer Node.js wytrzymuje obciążenie ponad 1000 równoległych strumieni.

W każdym z przeprowadzonych testów widoczna jest różnica w wydajnościach stworzonych aplikacji. W testach pustymi zapytaniami różnica ta wynosiła ok. 25%, natomiast w testowaniu wysyłania wiadomości znaczną przewagę miała aplikacja stworzona z pomocą Node.js. Była ona około 4 razy szybsza.

5. Wnioski

Przeprowadzone zostały badania dotyczące wydajności dwóch bardzo zbliżonych funkcjonalnie aplikacji stworzonych w życiem Node.js oraz frameworka Laravel i PHP. Z pomocą obu technologii utworzono 2 czaty. Wielowątkowy czat na frameworku Laravel z bazą danych MySQL oraz asynchroniczny czat na platformie Node.js z bazą danych MongoDB.

Przeprowadzona analiza teoretyczna związana z funkcjonowaniem badanych technologii programistycznych wskazała podobieństwa oraz różnice w działaniu obu szkieletów projektowych. Omawiając równocześnie mocne i słabe strony każdej z technologii w sytuacjach typowych dla działania aplikacji internetowych, w analizie zwrócono uwagę na najważniejsze aspekty związane z działaniem badanych technologii.

Dla obu aplikacji przeprowadzono badania za pomocą specjalnych programów: Apache Benchmark (test pustymi zapytaniami) i Apache JMeter (test wysyłania wiadomości). W każdym z testów lepiej wypadła aplikacja stworzona z użyciem Node.js.

Przedstawione porównanie powinno ułatwić programiście dokonanie wyboru między dwoma alternatywnymi rozwiązaniami.

Literatura

[1] Пауэрс Ш.: Изучаем Node.js, O'REILLY, 2012
[2] https://habrahabr.ru/post/150788/ [1.9.2016]
[3] https://laravel.ru/[8.9.2016]
[4] http://php.net/manual/ru/intro-whatis.php [12.9.2016]
[5] http://httpd.apache.org/docs/current/programs/ab.html [18.9.2016]
[6] http://www.sitepoint.com/sitepoint-smackdown-php-vs-node-js/ [21.9.2016]

JCSI 2 (2016) 64-69
Journal of Computer Sciences Institute

Wysłane: 2016-09-19
Przyjęte: 2016-10-03

Interfejs mózg-komputer wykorzystujący sygnały EEG

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

Streszczenie. Artykuł opisuje test aplikacji interfejs mózg-komputer z wykorzystaniem paradygmatu SSVEP. Przy realizacji projektu dokonano przeglądu dostępnych metod badania aktywności mózgu oraz wybrano odpowiednie urządzenie do akwizycji. Kolejne etapy działania interfejsu, czyli przetwarzanie oraz klasyfikacja, opracowano i zaprezentowano w środowisku OpenViBE. Ostatecznie, ocenę użyteczności i sprawności zaprezentowano na zaprojektowanej aplikacji.

Słowa kluczowe: Brain-Computer Interface; BCI; SVEEP; CSP; EEG

Adres e-mail: leszek0marek@gmail.com

Brain-Computer Interface based on EEG signals

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

Abstract. The aim of the article is to test the brain-computer interface application using the SSVEP paradigm. During the realization of the project various methods of recording brain activity were tested, and the suitable acquisition device was chosen. Consecutive stages of the interface operation, which are data processing and classification, were presented in the OpenVibe environment. Finally, the usefulness and efficiency were estimated using a designed application.

Keywords: Brain-Computer Interface; BCI; SVEEP; CSP; EEG

E-mail address: leszek0marek@gmail.com

1. Wstęp

Na przestrzeni ostatnich lat zaobserwować można dynamiczny rozwój informatyki, medycyny, a także współdziałanie obu tych nauk. Badanie elektroencefalogramem jest idealnym przykładem, gdzie informatyka ma wspierać, rozwijać i analizować czynności wykonywane przez mózg. Dzięki nowoczesnym komputerom i zaawansowanemu oprogramowaniu otwierają się nowe możliwości poznania funkcjonowania nie tylko mózgu, a także całego organizmu człowieka. Poruszone zagadnienia dotyczą obszaru biopomiarów związanych z interfejsami mózg-komputer, bazujących na sygnałach elektroencefalograficznych i ich własnościach. Ze względu na rozległą tematykę w artykule ostały przedstawione tylko wybrane interfejsy oparte na pomiarze sygnałów elektroencefalograficznych z uwzględnieniem sposobów analizy takich sygnałów. W literaturze badania prowadzone są głównie pod kątem medycznym do określenia schorzeń, bądź ułatwieniu komunikacji osoby z zaburzeniami ruchowymi. Trudność tej analizy spowodowana jest głównie niskimi wartościami osiąganych potencjałów, które często są niższe od wartości pojawiających się szumów. W artykule przedstawiono opis właściwości i parametrów sygnałów EEG, technik pomiaru oraz metod ich przetwarzaniai analizy. Zaprezentowano również praktyczną analizę wybranych parametrów sygnału elektroencefalograficznego na potrzeby systemu BCI (ang. Brain Computer Interface). Interfejs mózg-komputer został wykorzystany u zdrowej osoby do obsługi zbudowanej aplikacji. Celem artykułu jest sprawdzenie użyteczności BCI do sterowania aplikacją, co jest alternatywnym sposobem wykorzystania sygnałów EEG.

2. Elektroencefaligrafia

Elektroencefalografia (EEG) to badanie elektrofizjologiczne polegające na zebraniu z powierzchni głowy sygnałów bioelektrycznych emitowanych przez mózg. Wnikliwa analiza zapisów EEG może mieć wpływ na podejmowanie właściwych decyzji diagnostycznych o podłożu neurologicznym [1].

2.1 Budowa mózgu

Mózg jest najważniejszą częścią układu nerwowego człowieka. Do badań elektroencefalografem wykorzystuje się cześć mózgowia- kresomózgowie. Jest to zewnętrzna część mózgu podzielona na dwie półkule- prawą i lewą odpowiadające odpowiednio za lewą i prawą stronę ciała.

W mózgu wyodrębniamy kilka płatów, które odpowiadają za przypisane im funkcje życiowe.

Mózg zbudowany jest z elektrycznie pobudliwych komórek nerwowych zwanych neuronami. Neurony komunikują się ze sobą poprzez synapsy budując sieci neuronowe. Przekazywanie informacji odbywa poprzez tworzenie i przewodzenie potencjałów elektrycznych. Potencjał spoczynkowy waha się od około -65 mV do -90 mV, a gdy neuron zostanie pobudzony przez bodziec, tworzy się potencjał czynnościowy o wartości ok 40 mV. Elektroencefalograf rejestruje te różnice prądowe przy pomocy elektrod znajdujących się na głowie badanego.

Uwzględniając budowę anatomiczną mózgu w badaniu ograniczono ilość elektrod. Zostały one rozmieszczone tylko z tyłu głowy ze względu na powstające tam potencjały elektryczne związane z bodźcami wzrokowymi. Bodźce te będą odpowiedzialne za sterowanie zbudowanej aplikacji.

2.3 Przetwarzanie sygnałów EEG na użytek BCI

Wszystkie elementy przetwarzania sygnału EEG muszą się odbywać w czasie rzeczywistym ze względu na rodzaj badania. Dla poprawnego odczytania i interpretacji zapisanych sygnałów EEG zachodzi potrzeba dokonania ekstrakcji i selekcji sygnałów, a także ich klasyfikacja. Selekcja jest wykorzystywana przy dużych zbiorach danych, a jej odpowiednie opracowanie jest znaczące dla skuteczności klasyfikacji. Obecnie występuje wiele metod selekcji cech, które są optymalizowane pod kątem:

1) ograniczenia nadmiarowości danych,
2) zmniejszenia ilości zapisywanych danych,
3) zmniejszenia nakładów obliczeniowych,
4) zwiększenia skuteczności klasyfikacji.

Akwizycja sygnału EEG jest rzeczą skomplikowaną ze względu na to, że odebrane sygnały z elektrod charakteryzują się niskim napięciem (do 100 μV) i występują w całym szeregu zakłóceń oraz szumów. W badaniu rozważa się tylko bezinwazyjną metodę umieszczania elektrod na skórze głowy. Zakłócenia nazywane są artefaktami, ich źródłem są procesy fizjologiczne np. ruch mięśni, praca serca, ruch oczu jak i wywołane przez zjawiska techniczne np. sieć elektroenergetyczna. Wykorzystywane w badaniu pasmo sygnałów EEG zawiera się w zakresie częstotliwości od 0,5Hz do 100Hz wraz z dominującą częstotliwością 50 Hz.

Dla wyników badań EEG kondycjonowanie sygnału ma kluczowe znaczenie, ze względu na duże zakłócenia sygnałów oraz potrzebę ich analizy on-line. W tym procesie następuje wzmocnienie sygnału, korekcja artefaktów, dolnopasmowa filtracja sygnału.

Nieodłącznym elementem badań są powstające artefakty, które zawsze wpływają na jakość badań. Można go ograniczyć na 2 sposoby. Pierwszy z nich to zastosowanie wysokiej klasy sprzętu, który będzie generował możliwie mało zakłóceń. Drugi to odpowiedni instruktarz osoby badanej, uświadomienie powstawania artefaktów i podanie ich źródeł oraz sposobów ograniczenia ich powstawania. Po wyeliminowaniu artefaktów, wzmocnieniu i korekcie sygnału zostaje on przetworzony do postaci cyfrowej i przesłany do komputera gdzie może się zacząć tzw. przetwarzania wstępne (ang. preprosessing).

Przetwarzanie wstępne ma prowadzić to wyodrębnienia pożądanych fragmentów Sygnału EEG i polega ono na zastosowaniu odpowiednich filtrów. W sygnale EEG, który obejmuje swoją częstotliwością zakres od 0,5 Hz do 100 Hz, mieści się cześć artefaktów, których nie można wyeliminować całkowicie. Zastosowanie filtru częstotliwościowego pozwala na usunięcie z otrzymanych fali np. zakłóceń powstałych w wyniku sieci elektroenergetycznej, której częstotliwość (50 Hz) mieści się w zakresie użytecznych fal EEG.

Następnym działaniem mającym na celu odseparowanie charakterystycznych dla mózgu fal jest zastosowanie filtrów przestrzennych. Pojedyncza elektroda przymocowana do głowy zbiera nie tylko sygnały z obszarów mózgu, nad którymi się znajduje ale także zakłócenia fizjologiczne i techniczne.

Najpopularniejszą metodą filtracji przestrzennej jest filtr Laplace'a, który ma zastosowanie głównie przy wykorzystaniu potencjału P300, ze względu na jego lokalizację na czubku głowy. Metoda filtru Laplace'a polega na przypisaniu wag sygnału sąsiednim równomiernie rozmieszczonym elektrodom przy generowaniu fal z badanej elektrody.

3. Interfejs mózg-komputer

Brain Computer Interface - BCI tworzy nowy, sztuczny kanał do komunikacji pomiędzy człowiekiem a komputerem opartym na rejestracji specyficznych form aktywności mózgu. Czynność zanim zostanie wykonana przez człowieka najpierw powstają impulsy elektryczne w mózgu, skąd są przekazywane za pomocą układu nerwowego do mięśni powodując ich działanie. Zadaniem interfejsu mózg-komputer jest komunikacja alternatywną drogą. Początkowo zadaniem BCI było sterowanie protezami- mózg „uczył się" odczytywać sygnały emitowane przez protezy oraz generował sygnały potrzebne do sterowania nimi [2]. Współcześnie BCI wykorzystywane są przez osoby sparaliżowane lub chore do komunikacji ze światem zewnętrznym. Schemat działania tego procesu opiera się na zbieraniu sygnału pochodzącego z mózgu przez urządzenia, jego przetworzenie i finalnie przekazanie do urządzenia lub aplikacji (Rys. 1) Dzięki temu użytkownik komunikuje się ze światem zewnętrznym alternatywną drogą.

Rys. 1. Schemat interfejsu mózg-komputer [3]

W każdym BCI można wyróżnić cztery podstawowe elementy:

1) Rejestracja sygnału odbieranego przez elektrody, polegająca na odebraniu sygnału analogowego pochodzącego z aktywności mózgu przy danych czynnościach. Jest to sygnał wejścia do BCI.
2) Zamiana sygnału z analogowego na cyfrowy i jego ekstrakcja polegająca na wyodrębnieniu swoistych cech, za pomocą dobranych procedur m.in. filtrowania przestrzennego, analizy widmowej czy pomiaru amplitudy napięcia.
3) Dopasowanie otrzymanego sygnału do wzorców aktywności neuronalnej dla określenia jaką czynność umysłową wykonuje osoba badana.

4) Przesłanie polecenia do urządzenia o czynności użytkownika, a także przekazanie informacji zwrotnej dla osoby posługującej się BCI.

3.1. Potencjały wywołane

Potencjały elektryczne rejestrowane na powierzchni głowy noszą nazwę potencjałów wywołanych (Evoked Potentials - EP) Podczas obserwacji sygnału EEG, można zaobserwować charakterystyczne przebiegi, które są wywołane aktywnością mózgu związaną z procesami myślowymi, bądź koncentracją uwagi na danym obiekcie [4].

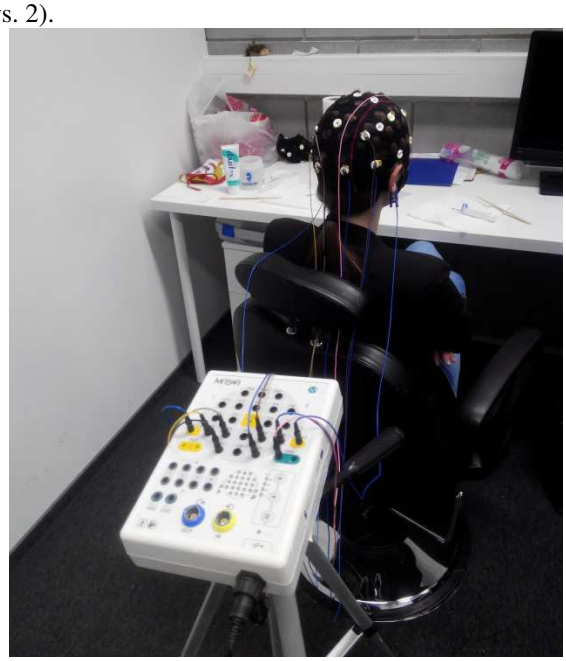
Potencjał P300 - nazywany również potencjałem sukcesu, jest jednym z najpopularniejszych potencjałów wykorzystywanych przez BCI. Powstaje w ciemieniowo-potylicznym płacie mózgu. Najprostszym sposobem na wykorzystanie potencjału P300 jest wyświetlanie liter na monitorze, a w chwili gdy pokarze się litera oczekiwana przez badanego wystąpi u niego potencjał. W ten sposób można tworzyć słowa i następnie zdania. Ponieważ potencjał P300 ma małą amplitudę, jest słabszy niż aktywność mózgu, a także może mieć różne cechy w zależności od badanego wskazane jest przeprowadzenie sesji treningowej, która pozwala na kalibrację interfejsu pod konkretnego badanego [5][6] .

Kolejnym wartym uwagi potencjałem są wyobrażenia ruchowe, a dokładnej synchronizacja/desynchronizacja powiązana z bodźcem. Jego istota polega na zmianie mocy sygnału, który pojawia się w świadomym ruchu, bądź bezpośrednio przed jego wyobrażeniem. Dzięki temu, że aktywność w mózgu powstaje na samą myśl o danej czynności rozwój interfejsu kierowany jest na np. sterowanie wózkiem przez osobę niepełnosprawną [7][8].

Stabilny potencjał wywołany wzrokowo (SSVEP - Steady State Visually Evoked Potential) - jak nazwa wskazuje jest potencjałem wywołanym przy pomocy zmysłu wzroku [9]. Powstaje w korze mózgowej w odpowiedzi na bodźce świetlne o częstotliwości w zakresie od 3,5 Hz do 75 Hz. Sygnał odbierany przez siatkówkę oka rejestrowany jest przez elektrody umieszczone w obszarze kory wzrokowej. Sygnał ten charakteryzuje się taką samą lub zwielokrotnioną częstotliwością co odbierany bodziec, ma dobry stosunek sygnału do szumu, a ponadto jest odporny na artefakty[10]. Badanie przy potencjale SSVEP wymaga od użytkownika możliwości poruszania oczyma. Osobie badanej przedstawia się specjalnie przygotowaną matrycę, na elemencie, której użytkownik skupia wzrok powodując wywołanie sygnału o pożądanej częstotliwości. Do określenia na co patrzył badany określane jest widmo sygnału, a tym samym o jakim działaniu myślał [12].

4. Badania

Badanie opiera się na opracowaniu, implementacji i przetestowaniu interfejsu mózg-komputer w oparciu o paradygmat SSVP. Zostanie wykorzystana metoda wspólnych wzorców przestrzennych (Common Spatial Pattern - CSP), która służy do zbierania sygnału pochodzącego z sąsiednich elektrod w celu uzyskania wyników wysokiej jakości. Sygnały z elektroencefalogramu zbierane zostaną poddane analizie w czasie rzeczywistym, co pozwoli poruszać się po opracowanej aplikacji. Do badań został użyty wzmacniacz Mitsar EEG 201. Jest to sprzęt obsługujący maksymalnie 25 kanałów, które przesyłają sygnał do komputera. Ze względu wybrany potencjał SSVEP podczas badań zostało użytych sześć elektrod skupionych przy części potylicznej, ponieważ znajduje się tam obszar mózgu odpowiedzialny za funkcje wzrokowe (Rys. 2).

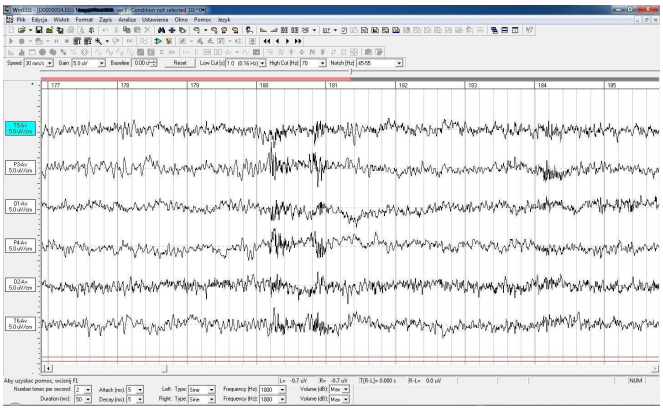


Rys. 2. Zdjęcie przedstawiające badanego przed rozpoczęciem eksperymentu

4.1. Zastosowane programy i sprzęt

Aby badanie mogło się odbyć niezbędny jest odpowiedni dobór sprzętu i urządzeń współpracujących. Jest to kluczowe dla powodzenia eksperymentu, gdyż jeden źle dobrany element może decydować o powodzeniu badania. Całość można podzielić na część programową i sprzętową. Programy i języki użyte w eksperymencie:

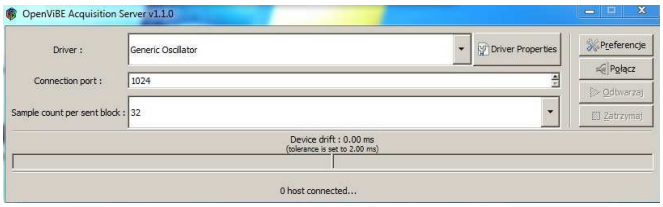
WinEEG- Jest to oprogramowanie, które umożliwia przetwarzanie odebranego sygnału EEG z uwzględnieniem istotnych cech dotyczących badania tj. analizy widmowej i koherencji, korekcji artefaktów i mapowanie mózgu. Narzędzie prezentuje w trybie on-line sygnały EEG w trybie ruchomego papieru, a dane są zapisywane na dysku twardym komputera.



Rys. 3. Okno główne programu OpenVibe z wykresami generowanymi przez podłączane elektrody

Program pozwala śledzić wirtualizację każdego podłączonego kanału (Rys. 3). Dzięki temu zabiegowi można sprawdzić jakość podłączenia poprzez prosty eksperyment: Badany na polecenie zamyka oczy a na wykresach powinien być zauważony rytm alfa, ponieważ do oka nie docierają bodźce świetlne. Wprawiony technik bez problemu
w otrzymanym sygnale wskaże odpowiedni rytm. Program jest dedykowany do współpracy z urządzeniem Mitsar 200. Ważną możliwością programu jest możliwość sprawdzenia rezystancji podłączonych elektrod.

Acquisition Server - oprogramowanie wchodzące w skład środowiska OpenVibe (Rys. 4). Jego zadaniem jest przekazanie sygnału za pomocą protokołu TCP/IP do narzędzia Designer. Lista sterowników do wyboru jest pokaźna i pozwala wybrać najodpowiedniejszy do dalszych badań. Do każdego sterownika można wybrać jego właściwości za pomocą przycisku Driver Properties. W eksperymencie wybrano sterownik Generic Oscillator.



Rys. 4. Okno główne narzędzia Acquisition Server

OpenVibe Designer - Druga cześć zestawu środowiska OpenVibe. Jest to rozbudowany program służący do projektowania i testowania scenariuszy z wykorzystaniem sygnału EEG. Aplikacja umożliwia wykorzystanie wbudowanych scenariuszy, a także tworzenie własnych przy pomocy języka programowania LUA.

Matlab - szeroko rozpowszechniony program do wyliczeń naukowych i inżynierskich , a także tworzenia symulacji komputerowych. W badaniu wykorzystany ze skryptem BCILAB służącą do projektowania, testowania i badania interfejsów BCI.

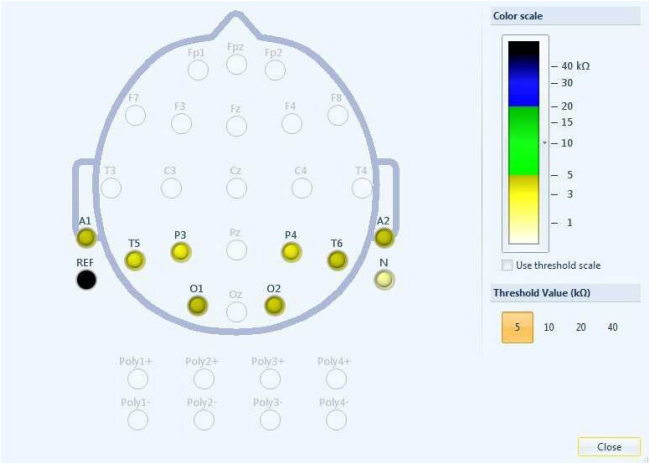
Język LUA- jest to język zaimplementowany jako biblioteka języka C. Składnia przypomina Pascala. Swoją konstrukcję opiera na tablicach asocjacyjnych i rozszerzalnej semantyce. Za jego pomocą napisane są scenariusze używane przez OpenVibe Designer.

Klasyfikator LDA (ang. Linear discriminat analusis) - |z matematycznego punku widzenia jest to liniowa analiza dyskryminacyjna. Używa one uczenia maszynowego do wyszukania liniowej kombinacji cech, a ich zadaniem jest rozróżnienie klas obiektów. W zaprezentowanym badaniu klasyfikator uczy się trzech rodzajów kombinacji, dzięki czemu poszczególne obiekty zostają rozróżnione. Poszczególne figury odpowiadają obiektom w badaniu, a zadaniem klasyfikatora jest nauczenie się intencji badanego, na których figurach skupia wzrok, a tym samym odseparowanie ich od siebie.

Metoda Common Spatial Pattern - jest to metoda wspólnych wzorców przestrzennych wykorzystywana do biofeedback'u. Zaprojektowana z myślą o skutecznej klasyfikacji sygnału EEG związanego z potencjałem ruchowym i wzrokowym. Metoda ta opiera się na zbieraniu sygnału z sąsiednich elektrod i tworzy nowe sygnały poprzez filtrację przestrzenną z doborem wag z wykorzystaniem maksymalizacji wariancji. W danym badaniu są trzy różne klasy (wyobrażenia ruchu). Na tej podstawie dzięki algorytmowi wag możliwe jest wykrycie elektrod wnoszących najwięcej do dyskryminacji klas. Dzięki wybraniu pozycji elektrod znajdujących się w części głowy, w której powstają sygnały wywołane bodźcami wzrokowymi można maksymalnie wykorzystać tą metodę [10].

4.2. Przygotowanie do badań

Mając wyznaczone punkty charakterystyczne na głowie zakładany jest czepek, tak aby miejsce montażu elektrod odpowiadało ich położeniu na głowie pacjenta (Rys. 5):



Rys. 5. Rozmieszczenie elektrod na głowie pacjenta i wyniki pomiaru oporu w użytych elektrodach

T5 - Położona jest na płaszczyźnie poziomej głowy w miejscu oddalonym o 70% długości lewego wymiaru wieńcowego od punktu nr 1, a także 30% długości lewego wymiaru wieńcowego od pkt nr 2.
P3 - elektroda P3 położona jest na środku łuku, który tworzą elektrody T5 i Pz
O1 - Położona jest na płaszczyźnie poziomej głowy w miejscu oddalonym o 90% długości lewego wymiaru wieńcowego od punktu nr 1, a także 10% długości lewego wymiaru wieńcowego od pkt nr 2.

Po uprzednim sprawdzeniu podłączenia można uruchomić wzmacniacz i sprawdzić opór pomiędzy elektrodą a skórą, który w przypadku urządzenia Mitsar 201 powinien być mniejszy niż 5kΩ. W połączeniach, w których ta wartość jest większa należy je sprawdzić i poprawić. Sytuacja idealna jest wtedy gdy wszystkie elektrody mają jednakowy opór, jednak z przyczyn technicznych niemożliwa do osiągniecia. Po zakończeniu tego etapu można przystąpić do wykonywania eksperymentu, który składa się z 6 etapów i został powtórzony 4 krotnie w celu zebrania większej ilości materiału, a także porównania ze sobą danych z poszczególnych prób.

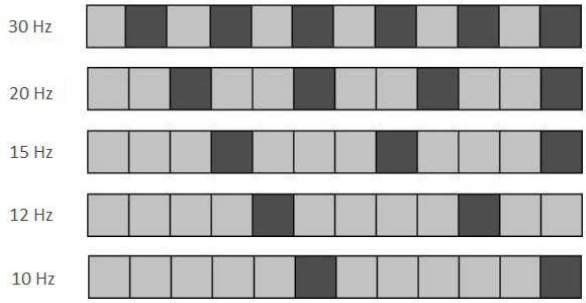
4.3 Akwizycja sygnału EEG

Jest to pierwsza cześć właściwego badania zatytułowana Ssvep-bci-0-acquisition-test. Scenariusz ten służy do zbierania danych wejściowych, uruchomiony po podłączeniu osoby badanej. Pozwala monitorować na bieżąco wszystkie sygnały z elektrod używanych w badaniu

SSVEP, a także dwa filtry przestrzenne. Celem tego etapu jest upewnienie się o prawidłowym podłączeniu sprzętu, a także o prawidłowej pracy programów współdziałających z OpenVibe. Scenariusz składa się z dziesięciu bloków, a każdy ma przypisaną funkcję.

4.4 Konfiguracja potencjału SSVEP

Ssvep-bci-1-ssvep-configuration - Scenariusz służący do wczytania danych składający się z dwóch części: ustawienia peryferyjne i ustawienia eksperymentu, które wpływają na uzyskane efekty. Ustawienia peryferyjne polegają na dobraniu częstotliwości odświeżania z zależności od monitora. Pracownia wyposażona jest w ciekłokrystaliczny wyświetlacz o maksymalnej częstotliwości odświeżania wynoszącej 60 Hz. Kolejną istotną rzeczą jest aby migoczący cel miał kontrastowe kolory, co podnosi sprawność działania interfejsu. Uwagę należy zwrócić na doborze odpowiedni dobór częstotliwości migoczących elementów. Ważne jest aby wartości były liczbami całkowitymi podzielnymi przez 60 (Rys. 6).

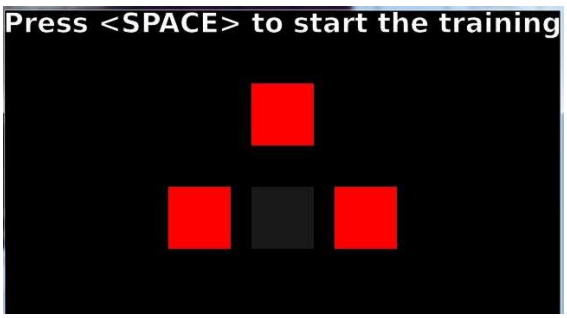


Rys. 6. Schemat przedstawiający prawidłowy dobór częstotliwości dla odświeżania 60 Hz. Opracowanie własne na podstawie [13]

W ustawieniach eksperymentu wybiera się: kolory wyświetlanych elementów, kolor tła, częstotliwość wyświetlania obrazów (iloraz musi być liczbą całkowitą), a także inne dane dotyczące obrazu wyświetlanego na monitorze podczas badań - czas trwania, interwał epoki i tolerancja błędu. Sygnał dzielony jest na epoki o ustalonej wcześniej długości trwania i odstępie pomiędzy nimi dzięki czemu powstaje odpowiedź SSVEP.

4.5 Trening Badanego

Ssvep-bci-2-training-acquisition- na tym etapie program „uczy się" na co badany skupia swoją uwagę. Na ekranie zostaje zaznaczony element, na którym należy skupić uwagę. W ten sposób program przypisuje do odpowiedniego elementu na ekranie daną wartość z zapisu EEG (Rys. 7).



Rys. 7. Rys. 7. Okno treningu użytkownika

Cały scenariusz trwa ok. 15 min, a wynikiem jest procentowa wartość z jaka program był wstanie rozpoznać elementy, na których skupiana była uwaga badanego. Wartościami oczekiwanymi są wyniki < 80%.

4.6 Trening CSP

Ssvep-bci-3-CSP-training - scenariusz wykorzystywany jest do uczenia doboru parametrów dla algorytmu CSP (Common Spatial Pattern - CSP). Metoda polega na przyporządkowaniu danych wycinków czasowych z sygnału EEG do poszczególnych elementów, na których badany skupiał swój wzrok. Dzięki temu tworzone są nowe, bardziej wartościowe sygnały po transformacji CSP poprzez zebranie z każdego z kanałów części informacji. Pożądany rezultat osiąga się poprzez dobór odpowiednich wag dla sąsiadujących elektrod.

4.7 Trening klasyfikatora BCI

Ssvep-bci-4-classifierd-training - jest ostatnim etapem przygotowań do przetestowania aplikacji w użytkowaniu przez badanego, a zarazem jej najważniejszym elementem. W tym miejscu klasyfikator interfejsu BCI ma nabyć umiejętność rozpoznania czy użytkownik skupia wzrok na danym elemencie, czy nie. Odbywa się to na podstawie wcześniej otrzymanych amplitud dla każdego z elementów. Poprawnie sklasyfikowanie zachowania osoby badanej jest kluczowe dla powodzenia całego badania. Od poprawności interpretacji przez program docierających sygnałów będzie zależało co się stanie na ekranie komputera - czy odpowiednie elementy zostaną wprawione w ruch.

Wyniki nauki klasyfikatora wyświetlane są w oknie programu w postaci informacji, z których można odczytać szanse na powodzenie badania (Tab. 4.1). Jak można zaobserwować wyniki oscylują w granicach wymaganych 80% co pozwala optymistycznie zacząć test. We wszystkich próbach blok trzeci wyraźnie ma mniejszą wartość od pozostałych, ale podczas wykonywania badań nie udało się znaleźć powodu tego stanu rzeczy, dlatego badania kontynuowano na otrzymanych wynikach.

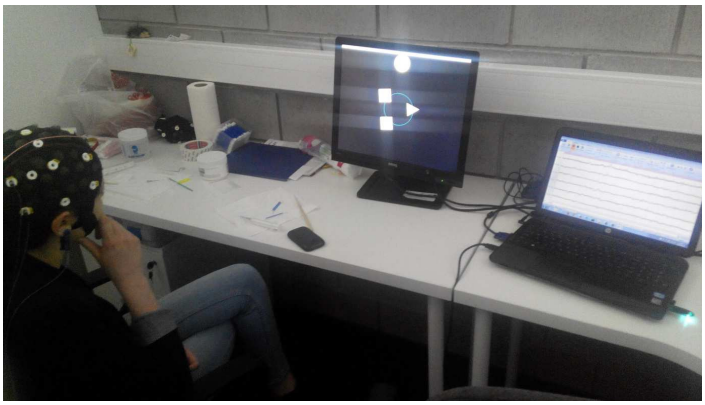
Tabela 1. Wyniki uczenia się klasyfikatora

Migoczący element	Wartość % nauki	Prognoza	Wartość sigma
Próba druga			
Blok pierwszy	80%	Optmistic	6,09083%
Blok drugi	80%	Optmistic	7,44742%
Blok trzeci	62,424%	Optmistic	7,17101%
Próba druga			
Blok pierwszy	78,5981%	Optmistic	6,09083%
Blok drugi	84,9542%	Optmistic	7,43845%
Blok trzeci	69.652%	Optmistic	7,00819%
Próba trzecia			
Blok pierwszy	71,9210%	Optmistic	6,66017%
Blok drugi	84,9201%	Optmistic	7,59601%
Blok trzeci	63,1682%	Optmistic	7,00014%

4.8 Test działania BCI

Ssvep-bci-5-online-test-shooter - jest to zwieńczenie prac opisanych w poprzednich rozdziałach poprzez przetestowanie interfejsu w praktyce. Wdrożona aplikacja pozwala na obracanie elementów po obwodzie koła (Rys.8). Klasyfikator odróżnia, które polecenie wydaje badany, poprzez skupienie wzroku na jednym z kwadratów bądź trójkącie.

Rola użytkownika BCI polega na skupianiu wzroku na jednym z kwadratów lub trójkącie, a tym samym wywołaniu potencjału SSVEP, który zostanie rozpoznany przez aplikacje i sklasyfikowany co wywoła ruch elementów na ekranie. Figury migają z różną częstotliwości dzięki czemu aplikacja potrafi rozpoznać, na którym elemencie jest skupiany wzrok. Koncentrując uwagę na jednym z kwadratów, elementy przedstawione na okręgu obracają się wokół jego środka. Trójkąt jest odpowiedzialny za strzał. Celem jest koło.



Rys. 8. Zdjęcie przedstawiające użycie protokołu SSVEP na potrzeby zbudowanej aplikacji

5. Wnioski

Opisany w artykule test interfejsu mózg - komputer został zrealizowany, chociaż nie udało uzyskać się pełnej sprawności interfejsu. Pomimo wyuczenia się przez klasyfikator cech na poziomie dochodzącym do 80% nie wszystkie elementy działały prawidłowo. Powodów częściowego niepowodzenia może być wiele, a najbardziej prawdopodobnymi mogą być:

1) użycie monitora LCD zamiast dedykowanej lampy
2) dane ze wzmacniacza przechodziły od wzmacniacza przez kilka współdziałających programów (EEGStudio, Matlab, LabStreaming i OpenVibe) co mogło zniekształcić dane, generować zakłócenia, opóźnienia.
3) parametry początkowe zostały źle dobrane do osoby badanej- każda osoba ma indywidualne predyspozycje i unikalne generowanie fal mózgowych.
4) klasyfikator nie był w stanie dobrze się nauczyć procedur pochodzących od osoby badanej.
5) użyto sześć elektrod, które mogły okazać się niewystarczające. Większa ilość elektrod zebrałaby dane lepszej jakości.

W przeprowadzonych badaniach ważnym elementem wypływającym na przebieg badania był dobór częstotliwości i kolorów migoczących elementów. Wraz ze wzrostem kontrastu poprawiała się praca klasyfikatora.

Przy odpowiednim dobraniu parametrów zadanie można wykonać z pełnym sukcesem, co jest czynnością pracochłonną i wymagającą specjalistycznej wiedzy zarówno z informatyki jak i medycyny.

Literatura

[1] A. Cichocki, S. Amari, Adaptive Blind Signal and Image Processing, ISBN:0470845899, (2002)
[2] J. Rowan, E. Tolunsky: Podstawy EEG z mini atlasem, Elsevier Urban & Partner, Wrocław, 2004.
[3] M. Jukiewicz: Praca magisterska pt. Klasyfikacja i analiza sygnału EEG na potrzeby interfejsu mózg-komputer, Wydział Elektryczny, Politechnika Poznańska, Poznań, 2012.
[4] J. Daly, J. Wolpaw, (2008). Brain-computer interfaces in neurological rehabilitation. Lancet Neurological, 7, 1032-1043
[5] E. Donchin, K.M. Spencer, R. Wijesinghe, The mental prosthesis: assessing the speed of a P300-based brain-computer interface, IEEE Trans Rehabil Eng, 8 (2000)
[6] J. Polich, Updating P300: an integrative theory of P3a and P3b. Clin Neurophysiol, 118 (2007)
[7] Hyekyoung Lee, A. Cichocki, Seungjin Choi, Kernel nonnegative matrix factorization for spectral EEG feature extraction, Neurocomputing 72 (2009)
[8] D. Regan, Steady-state evoked potentials. J Opt Soc Am 1977
[9] R. Rak, M. Kołodziej, Zastosowanie analizy częstotliwościowej sygnału EEG w interfejsach mózg-komputer, Przegląd Elektrotechniczny Nr 5 (2008).
[10] J. Ding, G. Sperling, R. Srinivasan, Attentional modulation of SSVEP power depends on the network tagged by the flicker frequency, Cereb Cortex. 16 (2006)
[11] M.M. Jackson, R. Mappus, (2010). Applications for Brain-Computer Interfaces. W: D.S. Tan, A. Nijholt (red.), Brain-Computer Interfaces. Applying our Minds to Human-Computer Interaction (s.89-104). Londyn: Springer
[12] M. Kołodziej, Przetwarzanie, analiza i klasyfikacja sygnału EEG na użytek interfejsu mózg-komputer , Warszawa, 2011
[13]SSVEP: Steady-State Visual-Evoked Potentials, posted 2011, http://openvibe.inria.fr/steady-state-visual-evoked-potentials/ [12.09.2016]

JCSI 2 (2016) 70-75
Journal of Computer Sciences Institute

Wysłane:2016-10-05
Przyjęte: 2016-10-15

Testowanie dźwięku w smartfonach w warunkach domowych

Marcin Ozimek

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

Streszczenie. W artykule opisano testy, przeprowadzone na trzech modelach smartfonów: Galaxy S6, iPhone 6S i Lumia 950. Testy miały na celu wybranie modelu, który najlepiej sobie radzi z przetwarzaniem dźwięku. Badania odbyły się w warunkach domowych, przy użyciu stosunkowo prostego sprzętu. W pracy opisano sposób przeprowadzenia testów, a następnie omówiono wnioski, decydując, który ma najlepsze parametry, jeśli o przetwarzanie dźwięku.

Słowa kluczowe: smartfony; pomiary dźwięku; dźwięk

Adres e-mail: marton026@gmail.com

Smartphone's Sound Testing in Home Setting

Marcin Ozimek

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

Abstract. Quality of sound is one of the most important factors while choosing the perfect smartphone model. This article concerns about domestic sound testing of smartphones. Both ways of examining and the equipment used for it are discussed. These tests have been performed on popular models of smarphones: Galaxy S6, iPhone 6S and Lumia 950 to collect recent data and to achieve the essential aim: establishing the fact which model is the best in terms of sound processing.

Keywords: smartphones; audio measurements; sound

E-mail address: marton026@gmail.com

1. Wstęp

Trudno wyobrazić sobie życie we współczesnym świecie bez telefonów komórkowych. Obecnie w krajach rozwiniętych zdecydowana część ludzi posiada smartfony, a z roku na rok urządzenia te stają się coraz doskonalsze, posiadają coraz więcej użytecznych funkcji i wykorzystywane są do coraz bardziej zaawansowanych celów.

Najrozmaitsze aplikacje ułatwiają życie człowieka na wiele sposobów: monitorują na przykład miejsca aktualnego pobytu użytkowania (jest to szczególnie ważne m.in. dla rodziców małych dzieci lub opiekunów osób chorych), ułatwiają podróż dzięki mapom oraz wskazówkom werbalnym (funkcja GPS-u), mierzą poziom hałasu w otoczeniu, ostrzegając przed jego nadmiernym natężeniem [1]. Coraz częściej smartfony wykorzystujemy w medycynie głównie do przekazywania danych o naszym zdrowiu. Jest to tzw. system WBAN (Wireless Body Area Network). Każdy pacjent ma swoje IP czyli nr identyfikacyjny [2]. Dane te są odbierane z urządzeń zewnętrznych podłączonych do smartfona i przesyłane przez Internet do bazy, do której wgląd ma nasz lekarz. Większość tych czujników przesyła przetworzone dane na telefon, jednak niektóre wykorzystują smartfony do rejestrowania sygnału dźwiękowego. Przykładem tego są czujniki akustyczne, służące do odsłuchiwania i rejestrowania szmerów tchawicy u pacjenta [3] lub aplikacje służące do wychwytywania nadchodzącego migotania przedsionków[4]. Trudno wymieniać wszystkie możliwości współczesnych smartfonów, gdyż jest ich bardzo wiele we wszystkich dziedzinach życia, a ich skuteczność bywa rozmaita. Przykładowo, podczas badań opisanych w artykule pt. „Testing the accuracy of smartphones and sound level meter applications for measuring environmental noise”[5] stwierdzono, że aplikacja służąca do mierzenia hałasu (na platformach Android i iOS) nie jest aktualnie lepsza od tradycyjnych urządzeń, choć dla przeciętnego użytkownika byłaby bardzo użyteczna. Jednakże badacze zasugerowali jednocześnie, że szybki rozwój technologiczny może wkrótce obalić prawdziwość wcześniejszego stwierdzenia, gdyż nowsze modele telefonów są coraz bardziej wydajne i posiadają coraz lepsze mikrofony.

W niniejszym artykule uwagę skupiono na możliwościach smartfonów w zakresie przetwarzania dźwięku, sprawdzając czułość ich mikrofonów, zdolność do rejestracji i odtwarzania mowy ludzkiej oraz muzyki.

Współczesne telefony wyposażone są bowiem w bardzo czułe mikrofony (najnowsze urządzenia posiadają co najmniej dwa), dzięki czemu można przy ich użyciu wyeliminować szumy ze ścieżki audio czy nastroić instrumenty muzyczne (służą do tego aplikacje GStrings na Androida lub Cteartune w systemie iOS). Można również wykorzystywać smarftony do tłumaczenia fragmentó tekstu – na przykład stosując aplikację iTranslate Voice (Android, iOS).

Warto jednak wspomnieć, że przeciętny użytkownik, zastanawiając się nad zakupem nowego smartfona, stoi przed nie lada wyzwaniem, gdyż mnogość dostępnych urządzeń oraz rozmaitość ich możliwości stanowi dla nich często problem trudny do rozwikłania. Zakładając, że użytkownik potrzebuje smartfona doskonałego pod kątem przetwarzania dźwięku i gdy to jest głównym kryterium jego wyboru, nadal może on mieć trudności w decyzji, gdyż nie ma gotowej odpowiedzi, który telefon spełni jego oczekiwania.

2. Metody badawcze

Zdecydowano przeprowadzić serię testów w warunkach domowych, na ogólnodostępnym sprzęcie, by wskazać wady

i zalety smartfonów pod kątem przetwarzania dźwięków. Do badań wybrano metodę eksperymentalną. Testowano trzy wybrane urządzenia mobilne, każde wyposażone w inny systemy operacyjny: Samsung Galaxy S6 (Android), Nokia Lumia 950 (Windows Mobile) oraz iPhone 6S (iOS). Były to najnowsze modele dostępne podczas testów.

Przeprowadzone badania skupiały się wokół rozpoznawania mowy (zapis mowy do postaci tekstu), rejestracji ścieżki dźwiękowej (tu brane pod uwagę były: dyktafon, rejestrator multimedialny), odczytu dźwięku (odtwarzacz multimedialny) oraz komunikacji głosowej (rozmowa telefoniczna).

W zaproponowanej metodologii posłużono się odpowiednio dobranymi miarami, do których należą m.in.: wielkość opóźnienia, skuteczność rozpoznawania mowy oraz parametry dźwięku: nagrywanych utworów muzycznych, odtwarzanych utworów na wyjściu słuchawkowym oraz głosu generowanego podczas rozmowy telefonicznej. W ramach metodologii przeprowadzono szereg testów, które miały pokazać, że możliwe jest przeprowadzenie dokładnych pomiarów parametrów dźwiękowych w urządzeniach mobilnych nawet w warunkach domowych, niewielkim nakładem środków i przy użyciu prostego sprzętu.

Przygotowując się do badań, postawiono sobie następujące hipotezy badawcze:

- Skuteczność rozpoznawania mowy ukazuje potencjał urządzenia w zakresie przetwarzania dźwięku.
- Wartości parametrów dźwięku i ich charakterystyki podczas testów w typowych operacjach przetwarzania dźwięku takich jak nagrywanie, odtwarzanie, komunikacja między rozmówcami ułatwiają porównanie urządzeń pod względem jakości rejestrowanego przez mikrofony dźwięku, jakości generowanego dźwięku przez głośniki oraz jakości dźwięku generowanego, przekazywanego i odtwarzanego podczas rozmów telefonicznych.

Poziomy parametrów i ich charakterystyki dają ogólną informację o jakości dźwięku, a więc o jakości jego przetwarzania przez urządzenia, a tym samym o zastosowanych w nim rozwiązaniach technicznych i użytych podzespołach oraz zaimplementowanych rozwiązaniach w systemie operacyjnym. Jakość wybranych parametrów dźwiękowych może świadczyć o przydatności urządzenia do zadań przetwarzania dźwięku.

Warto zdać sobie sprawę, że jakość dźwięku w różnych modelach smartfonów różni się, gdyż ma na nią wpływ oprogramowanie, sterowniki oraz użyte podzespoły audio. Współcześnie w telefonach umieszcza się tzw. audio huby, czyli zintegrowane i kompletne rozwiązania audio, w skład których wchodzą: przetwornik DAC, wzmacniacz słuchawkowy i głośnikowy oraz inne komponenty. Zaletą wykorzystywania audio hubów są oszczędność energii oraz miniaturyzacja podzespołów. W smartfonach występują one w postaciach dedykowanych lub zintegrowanych, te dedykowane (zastosowane w Samsung Galaxy S6 oraz w iPhone 6S) montuje się jako dodatkowy układ scalony. Zintegrowane audio huby natomiast zlokalizowane są w chipsecie głównym. Odnajdujemy je np. w Lumii 950. Jakość dźwięku zależy także, jak już wcześniej wspomniano, od jakości wbudowanych głośników.

System audio w Galaxy S6 zbudowany jest z następujących podzespołów: Wolfson Microelectronics 1840E Audio Codec, wzmacniacz audio Max98505, dwa głośniki oraz dwa mikrofony. Kodek audio w tym smartfonie wyprodukowała firma Wolfson. Wzmacniacz audio tajwańskiej firmy Maxim-Integrated jest wzmacniaczem mono o wysokiej sprawności klasy DG z wbudowanym przetwornikiem analogowo-cyfrowym ADC (Analog to Digital Converter). Generuje on czystsze i głośniejsze brzmienie oraz wydłuża żywotność baterii dzięki stosowaniu różnych napięć zasilania w zależności od poziomu sygnału.

W iPhone 6S jako systemu audio użyto natomiast innowacyjnych układów scalonych dostarczonych przez Cirrus Logic, amerykańskiego producenta wysokiej jakości podzespołów i systemów audio. w skład wyposażenia sprzętowego tego smartfona wchodzą:

- 2 x Apple/Cirrus Logic 338S1285 Audio Codecs
- Apple/Cirrus Logic 338S00105 Audio IC
- 2 x głośniki mono oraz stereo
- 3 x mikrofony firmy Knowles 5280 KSM2
- 1 x mikrofon Goertec 529 GWM1

W iPhone 6S głównym układem elektronicznym jest Smart Codec (wielordzeniowy procesor sygnału audio). Ma on bardzo wysoką wydajność, bo aż 600 MIPS (milion instruction per second) czyli 600 milionów instrukcji na sekundę. Steruje pracą czterech mikrofonów i w zależności od tego, jaki sygnał do nich dochodzi, wykrywa szumy i usuwa je ze strumienia audio. Proces ten zwany ANC (Active Noise Control) czyli aktywna redukcja hałasu, polega na dodaniu do ścieżki oryginalnej ścieżki o tej samej amplitudzie, ale z odwróconymi fazami, pochodzącej z mikrofonu odbierającego szumy. Dzięki temu fale się wzajemnie znoszą i na wyjściu mamy sygnał bez zakłóceń. Wewnątrz tego układu znajduje się 6/8 kanałowy 24-bitowy kodek HiFi o parametrach: 6 ADC(Analog Digital Converter) 100 dB SNR wejście mikrofonowe, 8 DAC(Digital Analog Converter) 115 dB SNR wyjście HP (high power) [6]. Dodatkowo układ ten obsługuje próbkowanie do 192 kHz oraz wspiera ultradźwiękowe akcesoria. Wzmacniacze (Boosted amplifier) użyte w tej architekturze są klasy D i mają wysoką sprawność. Kolejnym podzespołem użytym w Cirrus Logic jest HiFi D/A Converter, czyli konwerter umożliwiający konwersję sygnału cyfrowego na analogowy. Ma on zakres dynamiki 120 dB, również posiada filtry cyfrowe o małej latencji czyli opóźnieniu [7].

System audio w Lumii 950 nie należy do najnowocześniejszych. Zastosowano w nim procesor dźwięku z serii Qualcomm WCD9330 Audio Codec [8], procesor sygnałowy Qualcomm Hexagon QDSP V56, sprzętowy wzmacniacz mono klasy D serii TPA zasilający głośniki, cztery mikrofony oraz głośnik mono i stereo. W smartfonie tym do obsługi dźwięku użyto technologii: "Lumia Rich Recording" rozwijanej od 2007 przez firmę Nokia. Technologia ta ma na celu wyeliminowanie problemu

podczas nagrywania dźwięku o dużej głośności ok. 120 dB, z którą nie radzą sobie inne smartfony, mające mikrofony z małymi membranami. Lumia Rich Recording polega na użyciu specjalnych mikrofonów, składających się na system HAAC (High-Amplitude Audio Capture). System ten wykorzystuje cztery mikrofony, działające w parach. Pierwszy z jednej pary odpowiada za nagrywanie dźwięku jak standardowe mikrofony, czyli do 120 dB. Natomiast drugi przechwytuje dźwięk o natężeniu nawet do 140 dB. w systemie HAAC ważny jest również algorytm odpowiedzialny za łączenie powstałych dwóch ścieżek, redukcję szumów i niechcianych odgłosów, zarejestrowanych z pomocą drugiej pary mikrofonów. Membrana mikrofonu Lumii 950 ma dwa wewnętrzne przedwzmacniacze, mające różną wrażliwość i wykorzystuje jeden z nich w zależności od natężenia dźwięku.

3. Przebieg i wyniki testów

Smartfony testowano pod względem jakości audio na różne sposoby. Poszczególne testy telefonów były wykonywane w identycznych warunkach i takich samych ustawieniach. Pierwszy test miał na celu określenie jak urządzenia radzą sobie z rozpoznawaniem mowy. Kolejny test polegał na nagraniu fragmentu utworu przez smartfony, używając systemowych aplikacji i domyślnych ustawień. W trzecim teście analizowano nagrania z wyjścia słuchawkowego poszczególnych modeli. Następny test miał pokazać różnice jakości rozmów telefonicznych mierzoną na wyjściu słuchawkowym. Do testów użyto zestawu kina domowego (amplituner Onkyo TX-SR702 7.1ch, głośniki PolkAudio) oraz rejestratora wielościeżkowego Korg D888. Wykorzystano również kilka aplikacji takich jak: RightMark Audio Analyzer, Audacity, Spek, SpectraLAB, Praat, MediaInfo oraz Total Commander.

Rozpoznawanie mowy jest ściśle powiązane z wieloma dziedzinami wiedzy: akustyką, lingwistyką, matematyką, statystyką, a także sztuczną inteligencją. Jest ono najczęściej wykorzystane do komunikacji człowieka z komputerem czy robotem. Na rozpoznawanie mowy wpływ ma bardzo dużo czynników, przez co jest to proces dość złożony, dlatego też jego działanie na urządzeniach elektronicznych nie zyskało jeszcze stuprocentowej skuteczności. Tylko nieznaczna część operacji matematycznych przeprowadzana jest na urządzeniu rozpoznającym mowę, a reszta na serwerach działających w chmurze. Nowe smartfony są coraz wydajniejsze, dlatego też światowy potentat z branży IT - Google ciągle szuka lepszych rozwiązań. Deklaruje, że ich system rozpoznaje mowę z 92 procentową precyzją. Od kilku miesięcy zaczął testować rozpoznawanie mowy offline, czyli bez konieczności połączenia z Internetem. Ten system rozpoznawania mowy ma być 7 razy szybszy od dotychczasowego[9].

Do testów wykorzystano fragmenty dwóch książek w formacie audiobooków: "Apokalipsa" Dean Koontz - po polsku dla Galaxy S6 i iPhone 6S oraz "Odd and the frost giants" Neil Gaiman - po angielsku dla Lumia 950, gdyż w trakcie testowania smartfonów okazało się, że Lumia 950 nie obsługuje rozpoznawania mowy w języku polskim.

Test miał następujący przebieg. Urządzenie ustawiono 20 cm od głośnika centralnego kina domowego, wartość głośności amplitunera ustawiono na 50 jednostek. Telefon połączono z siecią WiFi a w notatniku uruchomiono rozpoznawanie mowy. Do zobrazowania wyników użyto funkcji "porównaj wg zawartości" w programie Total Commander.

Rys. 1. Test rozpoznawania mowy

Okazało się, że Galaxy S6 najlepiej radzi sobie z rozpoznawaniem mowy. Zamieniona mowa na tekst posiada niewiele literówek czy niewłaściwie podanych wyrazów. Dwa zdania rozpoznane były bezbłędnie. Po policzeniu wszystkich całych nierozpoznanych wyrazów oraz wszystkich wyrazów w tekście poddanym rozpoznawaniu została obliczona skuteczność rozpoznawania i w przypadku tego smartfona wyniosła 97%. Na kolejnym miejscu uplasował się natomiast iPhone 6S, gdyż przetworzony przezeń tekst już ciężko jest zrozumieć. Oprócz tego podczas testu telefon co chwilę zatrzymywał rozpoznawanie mowy i trzeba było go ponownie uruchamiać. W tym przypadku skuteczność rozpoznawania wyniosła 77%. Ostatnie miejsce zajęła Lumia 950. Podczas tego testu prawie połowa tekstu nie została rozpoznana. System rozpoznawania mowy jeszcze częściej zatrzymywał się niż w iPhonie i wymagał ponownego uruchomienia. Na uwagę jednak zasługuje wstawianie interpunkcji.

Badaniom podlegało także sprawdzenie jakości nagrań. Rejestrowanie fragmentu utworu muzycznego odbywało się z użyciem domyślnych aplikacji oraz ustawień. Następnie analizowano nagrane ścieżki dźwiękowe za pomocą programu: RightMark Audio Analyzer. Do testu wykorzystano utwór: "Mindfields" - Toto. Smartfon umieszczony był 2 metry od centralnego głośnika kina domowego. Głośność amplitunera ustawiona była na poziomie 50 jednostek.

Rys. 2. Test rejestrowania muzyki

Testowane urządzenia posiadały mikrofony rozmieszczone w różnych miejscach. Galaxy S6 posiada 2 mikrofony, jeden na górnej a drugi na dolnej ściance telefonu. Natomiast iPhone 6S i Lumia 950 mają po 4 mikrofony. W iPhonie dwa z nich umieszczone są na dolnej ściance, kolejne obok obiektywów aparatów z przodu i tyłu telefonu. W Lumii 950 mamy 2 mikrofony na przednim panelu u góry i u dołu oraz kolejne 2 u góry i u dołu na panelu tylnym.

Do testów użyto RightMark Audio Analyzer (RMAA) w wersji 6.4.1. Jest to darmowy program do użytku niekomercyjnego, przeznaczony do testowania urządzeń audio (karty dźwiękowe, amplitunery, głośniki, mikrofony, odtwarzacze multimedialne) oraz analogowych i cyfrowych ścieżek audio. Za pomocą programu możemy testować częstotliwość dźwięku, składową harmoniczną dźwięku, stosunek poziomu sygnału do szumów, czy poziom zakłóceń dźwięku. Otrzymane wyniki są porównywalne z tymi, które powstają przy użyciu profesjonalnego specjalistycznego sprzętu.

Smartfony testowano poprzez zaimportowanie plików audio do programu RMAA. Pliki audio posiadały rozdzielczość 32 bitów a ich częstotliwość próbkowania wynosiła 48 kHz. Należało przy tym pamiętać, aby ścieżki audio nie były przesterowane lub nagrane zbyt cicho. Program umożliwia wygenerowanie raportu porównania do ośmiu ścieżek audio.

W badaniu jakości nagrań najlepiej spośród wszystkich urządzeń wypadł iPhone. Na wynik ten składają się rozmaite parametry, które przedstawia tabela poniżej.

Tabela 1. Wyniki testu jakości nagrań z użyciem programu RMAA

Galaxy S6
iPhone 6S
Lumia 950

RightMark Audio Analyzer test

Testing chain: Test jakosci nagran
Sampling mode: 32-bit, 48 kHz

Summary

Test	Galaxy S6	iPhone 6S	Lumia 950
Frequency response (from 40 Hz to 15 kHz), dB:	+24.63, -14.03	+35.60, -17.64	+26.17, -15.46
Noise level, dB (A):	-23.8	-30.6	-16.0
Dynamic range, dB (A):	19.9	29.6	21.5
THD, %:	4.480	1.871	7.033
IMD + Noise, %:	97.696	98.594	99.096
Stereo crosstalk, dB:	-20.6	-27.5	-28.3

Pod względem testowania częstotliwościowej charakterystyki przenoszenia okazało się, że Galaxy S6 jest najlepszy. Zakłada się, że cały zakres częstotliwości akustycznych występuje między 20Hz a 20kHz, jednak w rzeczywistości z praktycznego punktu widzenia mieści się on w przedziale od 40Hz do 15kHz. W tym też zakresie charakterystyka powinna być jak najbardziej płaska i jak najmniej odbiegać od wartości 0 dB. Wartości takie zaobserwowano dla Galaxy S6 (+24,63, -14,03), podczas gdy największe odchylenia ma iPhone 6S.

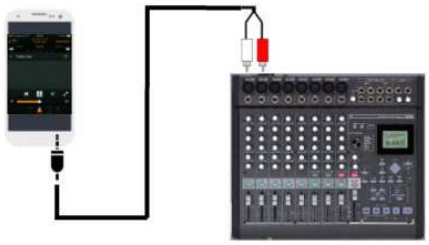
W czasie badań zwracano uwagę na poziom szumów. Największy wpływ na ocenę poziomu szumów istotny jest dla częstotliwości z zakresu od 1-3kHz. w tym teście to iPhone osiągnął najlepszy wynik. Rezultaty testu pokazują, że nagrane ścieżki dźwiękowe za pomocą smartfonów są mocno zaszumione. Dopiero wartości poziomu szumu poniżej -70dB gwarantują słuchanie utworu bez odczucia zaszumienia. Dla standardu CD poziom szumów oscyluje poniżej -90 dB.

Zakres dynamiki (dynamic range) to charakterystyka mierzona z wykorzystaniem sygnału o małej mocy. W idealnym przypadku powinna być ona zbliżona do poziomu szumów. W tym teście wyższa wartość oznacza lepszy wynik, a zatem najlepiej wypadł iPhone 6S.

Total Harmonic Distortion (THD) jest to test polegający na badaniu poziomu zniekształceń harmonicznych sygnału. Powinien być on jak najniższy. Najlepszy wynik uzyskał iPhone (1,871%). Przy czym zadowalający efekt dla ucha uzyskuje się dla wartości pomiarów na poziomie 1% lub niższych. Składowe harmoniczne powstają podczas odtwarzania sygnału o częstotliwości F. Pojawiają się wtedy słabsze sygnały, których częstotliwość jest wielokrotnością F. Sygnały harmoniczne parzyste (2F, 4F,...) są bardziej przyjemne dla ucha niż nieparzyste.

Zniekształcenia intermodulacyjne powstają w momencie kiedy różne sygnały przenoszone są jednocześnie przez układ nieliniowy. Różnią się one znacznie częstotliwością i poziomem mocy. Zniekształcenia te mogą być odbierane przez słuchacza jako dysonanse. Wartości zniekształceń poniżej 1% - 0,1% nie są dokuczliwe dla ucha. Sygnał o częstotliwości większej modulowany jest sygnałem o częstotliwości mniejszej. Wartość poziomu tych zniekształceń powinna być jak najmniejsza. Wyniki testów smartfonów są bardzo zbliżone. Najlepsze uzyskał Galaxy S6.

Test jakości dźwięku na wyjściu słuchawkowym polegał natomiast na nagraniu fragmentu utworu muzycznego odtwarzanego na smartfonie przez rejestrator wielościeżkowy Korg D888. Następnie analizowanie powstałych nagrań stereo za pomocą RMAA. Do testów posłużył utwór "Mad About You" - Toto. Smartfony podłączono przez wyjście słuchawkowe do rejestratora do dwóch oddzielnych kanałów. Głośność w telefonach ustawiono na maksymalną wartość.



Rys. 3. Test nagrywania na wyjściu słuchawkowym

Podczas tego testu liderem okazał się Galaxy S6, jednak wyniki pozostałych urządzeń były dość zbliżone. Jeśli wziąć pod uwagę pasma przenoszenia na wyjściu słuchawkowym, to najwyższe noty uzyskał iPhone 6S, gdyż jego wynik najmniej odbiega od wartości 0 dB. Natomiast najlepszy wynik poziomu szumów, wyliczany przez program na podstawie krzywej ważonej wynosi -29 dB dla urządzenia Galaxy S6. W kolejnych testach zakresu dynamiki oraz całkowitych zniekształceń harmonicznych Galaxy również

uzyskuje najlepsze noty. Tabela poniżej przedstawia dokładne wyniki tego badania.

Tabela 2. Wyniki testu jakości dźwięku na wyjściu słuchawkowym

Galaxy S6
iPhone 6S
Lumia 950

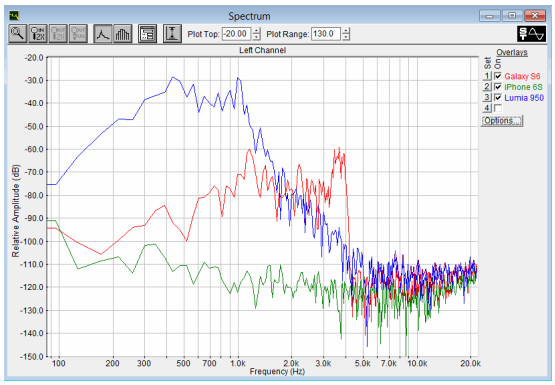
RightMark Audio Analyzer test

Testing chain: Jakosc wyjscia sluchawkowego
Sampling mode: 16-bit, 44 kHz

Summary

Test	Galaxy S6	iPhone 6S	Lumia 950
Frequency response (from 40 Hz to 15 kHz), dB:	+28.53, -16.23	+17.49, -14.74	+27.00, -17.13
Noise level, dB (A):	-29.0	-25.5	-25.1
Dynamic range, dB (A):	27.9	20.6	22.4
THD, %:	4.113	12.356	11.286
IMD + Noise, %:	97.995	94.236	92.020
Stereo crosstalk, dB:	-20.1	-15.5	-18.3

Jakość rozmów telefonicznych uzależniona jest od kilku czynników m.in. od jakości połączenia, lokalizacji, czasu, w którym ustanowiono połączenie, modelu urządzenia, operatora sieci komórkowej, pogody. Natomiast, aby uzyskać jak największą zrozumiałość mówionego głosu należy wzmocnić odpowiednie jego pasma częstotliwości. Do zilustrowania wzmocnienia poszczególnych pasm częstotliwości skorzystano z aplikacji SpectraLab a wyniki znajdują się na rysunku 4.



Rys. 4. Porównanie zakresu głośności poczczególnych częstotliwości

SpecrtaLab jest to darmowa aplikacja, uznawana za najlepszy analizator widma, oscyloskop i miernik fazy. Wyróżnia się trzy charakterystyczne pasma częstotliwości w ludzkim głosie. Pierwsze pasmo podstawowe mieści się w zakresie 85-250 Hz. Drugie samogłoskowe mieści się mniej więcej w zakresie 350 Hz - 2 kHz. Pasmo to zawiera większość mocy i energii głosu. Po przeanalizowaniu wyników widać, że najwyższą amplitudę w tych dwóch zakresach osiąga Lumia 950. Z kolei trzecie pasmo spółgłoskowe jest najistotniejsze ze względu na zrozumiałość mowy i mieści się w zakresie od ok. 1,5 kHz do 4 kHz. Tutaj można stwierdzić wyższość smartfona Galaxy S6.

Kolejny test polegał na nagraniu fragmentu audiobooka: "Apokalipsa" Koontz Deana na rejestratorze wielościeżkowym. Jeden telefon umieszczono 10 cm od centralnego głośnika i nawiązano połączenie telefoniczne ze smartfonem, podłączonym do rejestratora przez wyjście słuchawkowe.

Rys. 5. Test jakości rozmów telefonicznych

Na rejestratorze nagrywano te same fragmenty audiobooka, następnie przeanalizowano je wykorzystując program Praat. Jest to darmowy program, służący do zaawansowanej analizy i syntezy mowy z uwzględnieniem fonetyki. Pozwala on na analizę mowy pod kątem sprawdzania tonacji, odchyleń sygnału, intensywności, załamań drgań oraz przerw w głosie. Praat umożliwia również edytowanie plików audio oraz generowanie wykresów oraz raportu w postaci pliku tekstowego. Rozpoczęcie pracy z programem sprowadza się do zaczytania dowolnego pliku dźwiękowego stereo lub mono w jednym z popularnych formatów.

Rezultatem tego testu może być stwierdzenie, że Samsung Galaxy S6 osiąga najlepsze wyniki pod względem jakości głosu, a biorąc pod uwagę parametr Noise-To-Harmonics Ratio (NHR), określający zawartość szumu w podanym sygnale akustycznym, iPhone 6S wyprzedza konkurencję.

4. Wnioski

W pracy zostały postawione hipotezy. Pierwsza z nich wiązała skuteczność rozpoznawania mowy z potencjałem urządzenia mobilnego w zakresie przetwarzania dźwięku. W badaniach korzystano z domyślnych aplikacji służących do tego celu. Przeprowadzone badania nie potwierdzają tej hipotezy. Z testów wynika, że z rozpoznawaniem mowy najlepiej radził sobie Samsung Galaxy S6. Kluczową rolę w tym procesie odgrywa jednak jakość aplikacji, ich poziom zaawansowania, zastosowane algorytmy i rozwiązania. Podejście producenta do zagadnień związanych z problematyką dźwiękową skutkuje niewielkim zainteresowaniem inżynierów i programistów zajmujących się tą dziedziną. Efektem tego jest mały postęp w tym zakresie. W przeprowadzanych badaniach założono sobie, że rozpoznawanie mowy dotyczyć będzie polskiego języka. Okazało się jednak, że na urządzenie mobilne z systemem

Windows nie istniało na tamtą chwilę oprogramowanie rozpoznające język polski.

Ostatnia hipoteza odnosiła się do kwestii, czy dzięki parametrom dźwiękowym określającym jakość dźwięku można ocenić przydatność urządzenia do zadań przetwarzania dźwięku. Na podstawie przeprowadzonych badań trudno jednoznacznie to stwierdzić. Badania zostały przeprowadzone na najpopularniejszych urządzeniach mobilnych jakimi są smartfony. Analiza porównawcza dotyczyła trzech urządzeń, reprezentujących różne systemy operacyjne. Do testów wybrano smartfony o zbliżonych parametrach technicznych, reprezentujących trzy najpopularniejsze mobilne systemy operacyjne. System Android reprezentował Samsung Galaxy S6, iOS - iPhone 6S oraz system Windows 10 Mobile zainstalowany w Lumii 950. Wnioski jakie się nasuwają po badaniach jakości dźwięku w tych smartfonach to takie, że Galaxy i iPhone prześcigają się wynikami w poszczególnych testach. Jednak są dwie rzeczy, które przemawiają za wyborem iPhona jako najlepszego smartfona wśród testowanych. Są to po pierwsze wartość opóźnienia audio na poziomie 10 ms, uzyskiwany już w pierwszych modelach Apple. Drugą istotną rzeczą jest wsparcie sprzętowe audio do obsługi MIDI. Dzięki tym dużym różnicom smartfony te można używać do profesjonalnych zastosowań związanych z miksowaniem, czy obróbką dźwięku w czasie rzeczywistym. Dlatego też producenci profesjonalnego sprzętu muzycznego jak Apogee produkują urządzenia tylko dla produktów Apple.

Literatura

[1] E. Murphy i E. A. King, „Smartphone-based noise mapping: Integrating sound level meter app data into the strategic noise mapping process”, *Sci. Total Environ.*, t. 562, ss. 852–859, 2016.

[2] N. Bradai, L. C. Fourati, i L. Kamoun, „Investigation and performance analysis of MAC protocols for WBAN networks”, *J. Netw. Comput. Appl.*, t. 46, ss. 362–373, 2014.

[3] N. Reljin, B. A. Reyes, i Ki H. Chon, „Tidal Volume Estimation Using the Blanket Fractal Dimension of the Tracheal Sounds Acquired by Smartphone”, *Sens. 14248220*, t. 15, nr 5, ss. 9773–9790, 2015.

[4] J. Lee, B. A. Reyes, D. D. McManus, O. Maitas, i K. H. Chon, „Atrial Fibrillation Detection Using an iPhone 4S”, *IEEE Trans. Biomed. Eng.*, t. 60, nr 1, ss. 203–206, 2013.

[5] E. Murphy i E. A. King, „Testing the accuracy of smartphones and sound level meter applications for measuring environmental noise”, *Appl. Acoust.*, t. 106, ss. 16–22, 2016.

[6] „Android Marshmallow edges closer to «Pro Audio» with lower audio latency”, *Android Authority*, 2016. http://www. androidauthority.com/android-6-0-marshmallow-audio-latency-670727/. [11.06.2016].

[7] „Flagship Smartphone” https://www.cirrus.com/en/ applications/app/detail/APP17.html. [26.06.2016].

[8] „Snapdragon 808 and 810 unveiled: 20-nm, 64-bit beasts coming early next year”, *Android Authority*, 2014. ttp://www. androidauthority.com/snapdragon-808-810-specs-features-365530/. [25.02.2016].

[9] A. Li, „Google has created offline voice recognition that is 7x faster than an online system”, *9to5Google*, 11-mar-2016. https://9to5google.com /2016/03/11/google-accurate-offline-voice-recognition/. [10.07.2016].

JCSI 2 (2016) 76-80
Journal of Computer Sciences Institute

Wysłane: 2016-10-12
Przyjęte: 2016-10-18

Analiza porównawcza narzędzi RAD do wizualnego programowania w języku C++

Tetiana Pasikova*, Elżbieta Miłosz
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono porównanie wybranych narzędzi RAD do wizualnego programowania w języku C++. Do porównania zostały wybrane środowiska programistyczne: C++ Builder, Visual Studio, Qt Creator. Wielokryterialna analiza porównawcza i wybór najlepszego środowiska do nauki wizualnego programowania w C++ zostały przeprowadzone na podstawie danych producenta i przeprowadzonego eksperymentu ze studentami.

Słowa kluczowe: wizualne programowanie; C++; C++ Builder; Visual Studio; Qt Creator.

*Autor do korespondencji.

Adres e-mail: pasikova.t@gmail.com

Comparative analysis of visual programming RAD tools for C++ language.

Tetiana Pasikova*, Elżbieta Miłosz
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents a comparison of visual programming RAD tools for C++ language. For comparison were chosen following development environments: C ++ Builder, Visual Studio, Qt Creator. Multi-criteria comparative analysis and selection of the best learning environment visual programming in C ++ were carried out on the basis of the manufacturer and the experiment with students.

Keywords: Visual programing; C++; C++ Builder; Visual Studio; Qt Creator.

*Corresponding author.

E-mail address: pasikova.t@gmail.com

1. Wstęp

Celem badań przedstawionych w artykule było porównanie wybranych narzędzi RAD (*Rapid Application Development*) do wizualnego programowania w języku C++ i wyłonienie najlepszego z nich na podstawie wybranych kryteriów. Do porównania zostały wybrane następujące środowiska programistyczne: C++ Builder, Visual Studio, Qt Creator. Analiza obejmowała teoretyczne aspekty funkcjonowania ww. narzędzi oraz wykorzystanie ich w praktyce.

Analiza porównawcza środowisk do wizualnego programowania w języku C++ obejmowała następujące etapy:

- badania literaturowe z zakresu programowania wizualnego,
- opracowanie wybranych środowisk RAD – szybkiego wytwarzania aplikacji,
- sformułowanie hipotezy badawczej,
- dobór kryteriów dla porównania narzędzi,
- zaplanowanie i przeprowadzenie eksperymentu ze studentami,
- opracowanie wyników badań,
- wybór najlepszego środowiska i weryfikacja hipotezy badawczej.

Kluczowymi problemami badań są:

1) Jakie środowisko programowania wizulanego w języku C++ jest najlepsze?
2) Czy wcześniejsza znajomość środowiska przez studentów skraca czas stworzenia programu?
3) W jakim zakresie helpy i autopodpowiedzi środowiska ułatwiają napisanie programu?
4) Czy łatwo jest dostosować się do innego środowiska, jeśli ma się doświadczenie pracy w podobnym?

Szukanie odpowiedzi na postawione pytania pozwoli zweryfikować następującą hipotezę badawczą:

Najlepszym środowiskiem dla wizualnego programowania jest QT Creator.

2. Badania literaturowe

Języki używane w programowaniu wizualnym można klasyfikować w zależności od rodzaju i stopnia ekspresji wizualnej na następujące rodzaje [1]:

- języki wykorzystujące obiekty, gdy wizualne środowisko programistyczne zapewnia elementy graficzne lub symboliczne, które mogą być manipulowane interaktywnie zgodnie z pewnymi zasadami;
- języki schematów opartych na idei "kształty i linie", gdzie kształty (prostokąty, owale etc.) są traktowane jako przedmioty i są połączone liniami (strzałki, łuki), np. UML (*Unified Model Language*).

Programowanie wizualne wykorzystuje narzędzia typu RAD, które [2]:

- są ukierunkowane na zminimalizowanie czasu tworzenia aplikacji,
- umożliwiają szybkie opracowanie prototypu w celu udoskonalenia wymagań klienta,
- pozwalają na cykliczny rozwój oprogramowania: każda nowa wersja produktu opiera się na ocenach wyników prac poprzedniej wersji klienta, wykorzystanie gotowych modułów skraca czas tworzenia nowych wersji. Zespół korzystający z narzędzi RAD musi ściśle współpracować, każdy uczestnik musi być gotowy do wykonywania wielu zadań.

Programowanie wizualne umożliwia programowanie (a właściwie projektowanie) interfejsu aplikacji metodą umieszczania na pustych formularzach wymaganych elementów (komponentów). Komponenty wizualne (z interfejsem GUI) umieszczone są na formularzach metodą drop ciągnij i upuść, po skompilowaniu aplikacji wyglądają identycznie (WYSIWYG). Powiązania komponentów też mogą być realizowane wizualnie – można wykorzystać diagramy klas (dla aplikacji bazodanowych), często bazujące na wcześniej zaprojektowanym modelu UML. Programowanie wizualne wykorzystuje narzędzia typu CASE (*Computer Aided Software Engineering*) – zmiany dokonywane wizualnie automatycznie są zamieniane na odpowiedni kod często w kilku plikach jednocześnie, np. w specyfikacji klasy dodawana jest deklaracja metody, natomiast w ciele szablon definicji. Cechy obiektów (własności, metody i zdarzenia) dostępne są za pomocą wygodnych tabelarycznych lub drzewiastych struktur oraz podpowiadane podczas tworzenia kodu (*code completion*). Konfiguracja wartości złożonych i powiązań wspierana jest za pomocą licznych kreatorów (wizards). Składowe projektu zarządzane są za pomocą łatwo konfigurowalnych struktur hierarchicznych [3,4].

Programowanie wizualne jest ściśle powiązane z programowaniem zdarzeniowym. Przebieg realizacji programu zależy od obsługi zdarzeń jakie zachodzą na komponentach aplikacji typu naciśnięcie klawisza, wprowadzanie danych do okienek edycyjnych, przesuwanie myszy, zamykanie formularzy itp.

Programowanie zdarzeniowe jest najczęściej zaliczane do metodyk typu imperatywnego. Do zalet programowania zdarzeniowego można zaliczyć [5]:

- ograniczenie kodu do opisu tylko obsługiwanych zdarzeń,
- łatwość opisania interakcji z otoczeniem,
- posługiwanie się inną metodyką podczas samego opisu funkcji obsługi zdarzeń.

Wady programowania zdarzeniowego[6]:

- przy programowej implementacji wykorzystanie ukrytej, lecz *de facto* istniejącej części programu, odpowiadającej za obsługę pętli komunikatów,
- potrzeba znajomości możliwych zdarzeń i sposobu dostarczenia ich parametrów
- skomplikowanie budowy systemu – potrzeba implementacji sposobu obsługi nadchodzących przerwań lub komunikatów, wyszczególnienie, zdefiniowanie i udokumentowanie zdarzeń.

3. Charakterystyka wybranych środowisk RAD

Do analizy wybrano 3 środowiska szybkiego wytwarzania aplikacji w C++: Qt Creator, C++ Builder, Visual Studio.

Qt jest zespołem bibliotek umożliwiających tworzenie wszechstronnych aplikacji w języku C++. Stanowią one doskonałą bazę dla wieloplatformowych aplikacji z zaawansowanym graficznym interfejsem użytkownika. Wieloplatformowość w tym kontekście należałoby rozumieć jako zdolność do kompilacji kodu źródłowego danego rozwiązania na każdej obsługiwanej platformie bez wprowadzania jakichkolwiek zmian. W chwili obecnej wśród obsługiwanych systemów operacyjnych jest Linux, BSD, Windows, MacOS oraz wiele mniejszych mobilnych platform typu SymbianOS, MeeGo czy Windows CE. Qt jest dostępny pod dwoma licencjami: komercyjną, wymagającą wykupienia ale umożliwiającą produkcję zamkniętego oprogramowania oraz w wersji wolnej opartej na licencji GPL. Na ten moment, chyba najbardziej rozbudowanym projektem ukazującym możliwości biblioteki Qt jest K Desktop Environment (*KDE*) [7].

Borland C++ Builder jest zintegrowanym środowiskiem programistycznym, stanowiącym zbiór niezbędnych narzędzi pomocnych w szybkim tworzeniu aplikacji. Zawiera rozbudowane edytory tekstowe i graficzne, kompilator, linker oraz inne narzędzia pomocnicze. Przy wykorzystaniu C++ Buildera możliwe jest tworzenie aplikacji graficznych, bibliotek dll, kontrolek ActiveX. Z racji tego, że kompilator C++ Builder jest zgodny z standardami ANSI/ISO języka C++ możliwe jest też budowanie aplikacji konsolowych [8].

Wersja Personal zawiera bibliotekę z ponad 100 wizualnymi komponentami wielokrotnego użytku, które są przeciągane myszą na formularz. Oprócz dobrze znanych komponentów sterujących Windows (przyciski, paski przewijania, pola edycyjne, proste i połączone listy, itp) biblioteka udostępnia nowe komponenty wspierające dialog, usługi bazy danych oraz wiele innych.

Visual C++ to część bezpłatnego środowiska programistycznego Visual Studio. Program posiada edytor wizualny z wieloma wbudowanymi kontrolkami, rozbudowany edytor kodu i możliwością tworzenia własnych, system inteligentnych podpowiedzi, debugger, a także oferuje pełną integrację z Microsoft SQL Server 2008 i 2008 R2. Aplikacja posiada także opcję instalacji pakietu MSDN Library - zintegrowanej, również kontekstowej wersji offline pomocy dostępnej na stronach internetowych MSDN. Do instalacji programu wymagane jest połączenie z Internetem – instalator pobiera odpowiednie komponenty online [9].

4. Dobór kryteriów dla porównania narzędzi

Metodą weryfikującą postawioną hipotezę będzie wielokryterialna analiza porównawcza.

Wybrane środowiska zostały przetestowane według poniższych kryteriów:

1) *Licencja*. Cel: czy program jest płatny lub bezpłatny.
2) *Wieloplatformowość*. Cel: czy program jest dostępny dla Windows, Linux, MacOS.
3) *Zajętość pamięci na dysku*. Cel: zbadać ile miejsca potrzebuje program na dysku twardym.
4) *Wspomaganie programisty*. Cel: jak przydatne są helpy i autopodpowiedzi środowiska.
5) *Przyjazność środowiska – łatwość pisania aplikacji.* Cel: napisać program z GUI dla rozwiązania równania kwadratowego i ocenić stopień łatwości jego tworzenia.

Dla każdego kryterium będą wystawione oceny w odpowiedniej skali. Każdemu kryterium będzie przypisana odpowiednia waga, określająca jego ważność w procesie wyboru. Dla podjęcia decyzji będzie zastosowana metoda rozwiązywania problemów decyzyjnych – ELECTRE I. Metoda ELECTRE I zakłada, że porównując dwa warianty decyzyjne a i b możemy mieć do czynienia z jedną i tylko jedną z następujących sytuacji: a jest uznawane za równoważne b, a jest preferowane w stosunku do b, b jest preferowane w stosunku do a.

Ostatnie kryterium: przyjazność środowiska zostanie oceniona za pomocą następującego eksperymentu: wybrani studenci (10 osób) napiszą aplikacje graficzne rozwiązujace równanie kwadratowe w każdym środowisku i ocenią procesy ich tworzenia. Dla studentów przygotowano instrukcje opracowania aplikacji oraz kwestionariusze samooceny znajomości środowisk i oceny procesu tworzenia aplikacji pod kątem przyjazności środowiska.

Metody ELECTRE stosowane są w przypadkach szczególnie złożonych sytuacji decyzyjnych. Metody te wymagają zastosowania co najmniej trzech lub więcej kryteriów oceny. Są użyteczne zwłaszcza w sytuacjach, gdy zbiór dobranych kryteriów ma charakter heterogeniczny, tzn. gdy brane pod uwagę cechy różnią się istotnie między sobą [10].

Metoda ELECTRE I (***El**imination **e**t **C**hoice **T**ranslating **Re**ality)* – wykorzystuje koncepcję relacji przewyższenia A_k -> A_l, która mówi, że nawet jeśli dwa warianty nie dominują się wzajemnie, to decydent akceptuje ryzyko traktowania wariantu A_k jako prawie na pewno lepszego od wariantu A_l. Metoda ta opiera się na porównaniach parami wszystkich wariantów decyzyjnych, co ma doprowadzić do ujawnienia częściowego uporządkowania wariantów zgodnie z preferencjami decydenta [11].

Metoda wykorzystuje test zgodności i niezgodności:

- **zgodność** – czyli stopień w jakim wagi preferencji są w zgodzie z relacją dominacji par;
- **niezgodność** – czyli stopień w jakim obliczenia wagowe różnią się między sobą.

Metoda do utworzenia końcowego rankingu wykorzystuje dwie macierze:

- **macierz zgodności** *(concordance matrix)* **–** macierz wartości obliczanych dla każdej pary kryteriów, informujących w jakim stopniu jedna alternatywa jest co najmniej tak dobra jak druga.
- **macierz niezgodności** *(discordance matrix)* **–** macierz wartości obliczanych dla każdej pary kryteriów, informujących w jakim stopniu jedna alternatywa jest gorsza od drugiej.

Badane środowiska (alternatywy) a także kryteria oceny środowisk i ich wagi w skali od 1 (mała) do 9 (duża) przedstawiono w tabeli 1.

Tabela 1. Alternatywy i kryteria oceny środowisk

Alternatywa:		Kryterium:		waga
A1	Visual Studio	Q1	Licencja	6
A2	QT Creator	Q2	Wieloplatformowość	3
A3	C++ Builder	Q3	Zajętość pamięci	3
		Q4	Wspomaganie programisty	7
		Q5	Przyjazność środowiska	8
			suma	27

Kryterium Q5 - Przyjazność środowiska będzie ocenione podczas eksperymentu ze studentami na podstawie wypełnionych kwestionariuszy według kryteriów przedstawionych w tabeli 2.

Tabela 2. Alternatywy i kryteria oceny przyjazności środowiska

Alternatywa:		Kryterium:	
A1	Visual Studio	Q1	Znajomość języka C++
A2	QT Creator	Q2	Czas stworzenia formularza
A3	C++ Builder	Q3	Całkowity czas pisania programu
		Q4	Łatwość pisania aplikacji w danym środowisku
		Q5	Trudność adaptacji do składni kodu
		Q6	Intuicyjność interfejsu

5. Wyniki badań

Cztery pierwsze kryteria oceny wybranych środowisk wizualnego programowania w języku C++ (licencja, wieloplatformowość, zajętość pamięci, wspomaganie programisty) oceniono na podstawie danych producenta. Piąte kryterium – przyjazność interfejsu na podstawie eksperymentu – badania przeprowadzonego na grupie 10 studentów. Każdemu studentowi dano za zadanie napisać prosty program dla rozwiązania równania kwadratowego w trzech środowiskach programistycznych: Visual Studio, QT Creator, C++ Builder. Każdy student otrzymał szczegółową instrukcję w każdym środowisku niezbędną do napisania programu. Po wykonaniu zadania studenci

wypełniali kwestionariusz odpowiadając na pytania o łatwości użycia każdego środowiska.

Wyniki każdego studenta przeanalizowane zostały za pomocą metody wielokryterialnego wyboru – ELECTRE I. Dzięki niej oceniono jakie środowisko każdy student uważa za najlepsze. Na podstawie ponownego wykorzystania metody ELECTRE I dla 5 wybranych kryteriów wybrano środowisko programistyczne które najbardziej pasuje dla wizualnego programowania w języku C++.

Tabela 3. Podsumowanie wyników eksperymentu ze studentami

Nr studenta	Znajomość języka C++			Znajomość środowiska			Najlepsze środowisko
	zaawansowana	średni	podstawowa	Visual studio	QT Creator	C++ Builder	
1		X		X			Visual studio
2		X		X			Visual studio
3			X			X	C++ Builder
4			X		X		C++ Builder
5			X		X		QT Creator
6		X		X			Visual studio
7		X				X	C++ Builder
8			X		X		QT Creator
9	X					X	C++ Builder
10	X			X			QT Creator

W tabeli 3 przedstawiono wybór każdego studenta. Środowisko C++ Builder otrzymało najwięcej punktów – 4. Środowiska QT Creator i Visual Studio po 3 punkty.

Jak wynika z rezultatów badań – większość studentów zaznacza, że najlepszym środowiskiem dla wizualnego programowania jest to, z którego korzystali oni wcześniej.

Zgodnie z metodą ELECTRE I policzono współczynniki macierzy zgodności (tabela 4) i niezgodności (tabela 5). Zgodnie z zasadami indeksu zgody oraz indeksu niezgody ustawiono progi współczynnika zgodności c1 na 0,48 i współczynnika niezgodności d1 na 0,7.

Tabela 4. Macierz zgodności

Cij	A1	A2	A3
A1	–	0,2592592593	0,4814814815
A2	0,7407407407	–	0,3333333333
A3	0,5185185185	0,6666666667	–

Tabela 5. Macierz niezgodności

Cij	A1	A2	A3
A1	–	0,6666666667	0,6666666667
A2	0,6666666667	–	0,6666666667
A3	0,6666666667	0,6666666667	–

Tabela zalet alternatyw odnośnie innych dla badania na podstawie danych producenta i wyników eksperymentu ze studentami (tabela 6) zbudowana została w następujący sposób: jeżeli indeks zgodności Aij > c1 i indeks niezgodności Aij < d1 to alternatywa A1 jest lepsza od alternatywy A2 i ma znaczenie „+” w innym przypadku ma znaczenie „–”. Najlepszą alternatywą jest ta, która ma najwięcej „+”.

Tabela 6. Zalety alternatyw odnośnie innych

Cij	A1	A2	A3
A1	*	–	+
A2	+	*	-
A3	+	+	*

Najwięcej zalet ma alternatywa A3 – środowisko C++ Builder.

Zgodnie z wynikami badań analizy na podstawie przyjętych kryteriów i ich wag najlepszym środowiskiem dla wizualnego programowania okazało się środowisko C++ Builder. Przyjęta hipoteza ***nie została potwierdzona.***

6. Wnioski

Narzędzia do programowania wizualnego znacznie ułatwiają tworzenie aplikacji. Środowiska te mają na celu maksymalizację wydajności programisty. Można administrować programem w trakcie jego tworzenia za pomocą dwóch widoków: zakładki kodu i zakładki widoku obiektów graficznych.

W przebiegu badań miały znaczenie wagi kryteriów, co pozwoliło zaznaczyć, na ile ważne jest każde kryterium w stosunku do innych. Po testach zostały wystawione oceny dla każdej charakterystyki. Biorąc pod uwagę małą różnicę między ocenami i różnymi wagami kryteriów do podjęcia decyzji była wybrana metoda rozwiązywania problemów decyzyjnych – ELECTRE I, która pozwoliła policzyć wyniki uwzględniając wagi kryteriów.

Na podstawie przeprowadzonych badań najlepszym środowiskiem dla wizualnego programowania w języku C++ jest C++ Builder.

Studenci z podstawową znajomością języka C++ najczęściej wybierają środowisko C++ Builder. Dla studentów z lepszą znajomością języka C++ nie jest trudne korzystanie z żadnego środowiska i wybierają te z którego korzystali już wcześniej. Ośmiu z 10 studentów stwierdziło, że najlepszym środowiskiem jest to z jakiego korzystali wcześniej, dwu z 10 studentów zmienilo swoje zdanie.

Literatura

[1] К.А. Хайдаров, 4GL-Технологии. Основы визуаль-ного программирования, http://bourabai.kz/einf/4gl. htm, 02.05.2016

[2] К.А. Хайдаров, RAD – технология быстрого программирования, http://bourabai.kz/C-Builder/RAD. htm, 14.04.2016

[3] S. Vohra, Object Oriented Programming with C++: OOPC, bookrent.in Impression, 2015.

[4] K. Kuczmarski, Kurs C++, Helion, Xion, 2004.

[5] J. Matulewski, Podstawy programowania RAD z użyciem narzędzi Borland Delphi/C++ Builder, ćwiczenia, Helion, Gliwice, 2003.

[6] А.Я. Архангельский, Программирование в Borland C++, Москва, Бином, 2001.

[7] У.Р. Алексеев, Г.Г. Злобин, Программирование на языке C++ в среде QT Creator, Москва, ALT Linux, 2015.

[8] О. Вальпа, Borland C++ Builder - Экспресс-курс, Петербург, БХВ, 2006.

[9] R. Simiński, Programowanie w środowiskach RAD, http://programowanie.siminskionline.pl/resource/pwsz/qt_w02.pdf, [30.09.2016].

[10] ELECTRE, http://brasil.cel.agh.edu.pl/~13sustrojny/ electre-iii/, [30.09.2016].

[11] О.И. Ларичев, Метод ELECTRE, http://bugabooks.com/book/264-teoriya-i-metody-prinyatiya-reshenij-a-takzhe-xronika-sobytij-v-volshebnyx-stranax/86-4-metod-electre-i.html, [30.09.2016].

JCSI 2 (2016) 81-84
Journal of Computer Sciences Institute

Wysłane: 2016-10-20
Przyjęte: 2016-10-26

Metody interpolacji liniowej oraz Lagrange'a do uzupełniania trajektorii ruchu 3D

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

Streszczenie. Podczas rejestracji ruchu z zastosowaniem pasywnego systemu akwizycji ruchu, w trajektorii danego markera może powstać luka, którą w procesie obróbki danych należy uzupełnić. Stosuje się w tym celu różne algorytmy interpolacji. Artykuł przedstawia trzy różne metody interpolacji: liniową, wielomianem Lagrange'a drugiego stopnia oraz piątego stopnia. W oryginalnych danych 3D zostały utworzone dziury, które następnie uzupełniono trzema metodami. Każda metoda została oceniona za pomocą miary odległości Euklidesa dla trzech wymiarów danych oryginalnych oraz po uzupełnieniu trajektorii. W artykule przedstawiono także program do uzupełniania trajektorii, zaimplementowany w języku C++. Przedstawiona analiza dotyczy uzupełnianie trajektorii wybranych markerów z modelu Plug-in Gait podczas chodu osoby.

Słowa kluczowe: interpolacja liniowa; wielomian Lagrange'a; uzupełnianie trajektorii 3D

*Autor do korespondencji.

Adres e-mail: mateusz91.pollub@gmail.com

Methods of linear and Lagrange interpolation to fill in 3D motion trajectory

Mateusz Pęzioł*', Maria Skublewska-Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Abstract. During the motion registration using passive motion capture system in the marker's trajectory a gap may appear, which in the data post-processing should be filled in. Various interpolation algorithms are used for this purpose. This article presents three different interpolation methods: linear, polynomial of Lagrange of second degree and fifth degree. In the original 3D data the holes have been created, which are then filled in by mentioned methods. Each method was evaluated by measuring the Euklides distance for three dimensions of the original and interpolated data. The article presents the program to interpolate gaps in the trajectory, implemented in C ++ language. The analysis concerns the interpolation of selected marker's trajectory which belongs to the Plug-in Gait model while performing walking.

Keywords: linear interpolation; Lagrange polynomial; filling in the 3D trajectory

*Corresponding author.

E-mail address: mateusz91.pollub@gmail.com

1. Wstęp

Postęp techniczny w dużej mierze wpływa na świat rozrywki. Gry komputerowe oraz filmy, szczególnie te science fiction, są pełne efektów specjalnych oraz fikcyjnych postaci, których sposób poruszania się zadziwia ludzi na całym świecie. Taki ruch powstaje przy pomocy technologii Motion Capture. Jest to technologia, która polega na nagrywaniu ruchów człowieka i zapisywaniu ich do komputera w postaci różnych plików, np. C3D. Nagrywanie ruchu dobywa się w specjalnym studiu, a nagrywana osoba ma przyczepione markery, albo bezpośrednio na skórę, albo do kostiumu [1]. Kamery znajdujące się w studiu przechwytują ruch markerów, a następnie zapisują go do pliku w postaci współrzędnych (x, y, z). Często zdarza się, że podczas nagrywania ruchów (prostych, jak także skomplikowanych), któryś z markerów może zostać przysłonięty przez jakąś część ciała aktora i w zapisanym ruchu powstaje dziura, którą należy uzupełnić. Konieczne jest zatem uzupełnienie takiej przerwy, przy pomocy odpowiednich metod interpolacji [2].

Celem artykułu jest analiza wybranych metod obróbki danych ruchu 3D do uzupełniania trajektorii ruchu, zapisanych w plikach C3D. Porównywane metody interpolacji to: liniowa i wielomianowa Lagrange'a przy użuciu trzech oraz sześciu węzłów. Uzupełnianie trajektorii zostało zautomatyzowane poprzez program napisany w języku C++. Uzyskane wyniki (wartości współrzędnych interpolowanych markerów) zostały porównane z oryginalnymi wartościami. Na podstawie miary odległości Euklidesa została oceniona skuteczność każdej metody. Badania zostały przeprowadzone na danych ruchu – chodu. Ruch został nagrany w Laboratorium Analizy Ruchu i Ergonomii Interfejsów na Politechnice Lubelskiej z użyciem pasywnego, optycznego systemu motion capture. Osoba badana miała umieszczone retro-refleksyjne markery, rozmieszczone według schematu odpowiadającego modelowi biomechanicznym Plug-in Gait [3].

2. Metody interpolacji

2.1. Interpolacja liniowa

Interpolacja liniowa jest prostą i szybką metodą interpolacji. Polega ona na uzupełnianiu nieznanych wartości z użyciem wielomianu pierwszego stopnia dla znanych wartości. Do wyznaczenia pustych klatek korzysta się z funkcji liniowej. Ponieważ funkcja liniowa jest jednoznacznie określona przez dwa punkty, więc interpolację przeprowadza się dla każdej pary kolejnych punktów z poszukiwanego przedziału. Sprawdza się ona tylko w przypadkach, gdy tor ruchu jest zbliżony do linii prostej, natomiast podczas rejestracji bardziej skomplikowanego ruchu jej użycie nie jest wskazane. Wartość funkcji interpolującej w punkcie *x* leżącym pomiędzy dwoma punktami (x_0; y_0) oraz (x_1; y_1) można obliczyć ze wzoru 1 [2]:

$$L(x) = y_0 + \frac{y_1 - y_0}{h}\left(x - x_0\right) \tag{1}$$

gdzie:

L(x) – wartość y szukanego punktu;

h – różnica współrzędnych x_1 oraz x_0.

2.2. Interpolacja Lagrange'a

Interpolacja Lagrange'a (wzór 2) [2] jest interpolacją wielomianową, można ją liczyć wielomianem różnego stopnia. Im więcej znanych punktów początkowych, tym większy stopień wielomianu, a to powinno generować dokładniejsze wyniki podczas interpolacji.

$$w_f(x) = \sum_{j=0}^{n} f(x_j) l_j(x) \tag{2}$$

gdzie:

$w_f(x)$ – współrzędna y szukanego punktu;

$f(x_j)$ – wartości funkcji w punkcie xj;

x_j – współrzędne x dla kolejnych punktów;

x – znana współrzędna x szukanego punktu, która należy do przedziału (x_0, x_1);

$l_j(x)$ – dany jest wzorem 3 [2]:

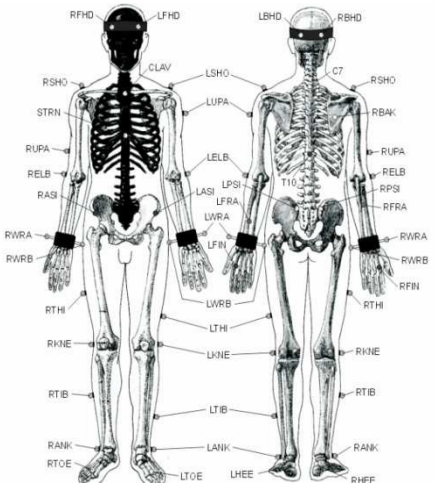
$$l_i(x) = \prod_{0 < j \le n,\ j \ne i} \frac{x - x_j}{x_i - x_j} \tag{3}$$

3. Metodyka badawcza

Analiza została przeprowadzona na danych ruchu 3D zawierających chód. W pierwszej kolejności w oryginalnych plikach zostały utworzone dziury w trajektoriach dla wybranych markerów z różnych część ciała. Analizowano markery: 1)LBHD (umieszczony po lewej stronie na głowie), 2) LRWA (umieszczony na lewym nadgarstku), 3) LANK (umieszczony na lewej kostce). Następnie, pliki zostały poddane trzem interpolacjom, w wyniku czego otrzymano nowe interpolowane wartości usuniętych pozycji markerów. Metody interpolacji zostały ocenione miarą Euklidesa.

3.1. Plug-in Gait Model

Liczba i rozmieszczenie znaczników na aktorze wpływa na jakość nagrania. Markery przyczepione w odpowiednich miejscach pozwalają na odwzorowanie naturalnych ruchów postaci. Model biomechaniczny Plug-in Gait Model (rysunek 1) pokazuje, w którym miejscu należy umocować znaczniki, aby otrzymać realistyczny ruch.



Rys. 1. Schemat rozmieszczenia markerów w modelu biomechanicznym Plug-in Gait Model [3]

3.2. Wygenerowanie dziur w trajektoriach

Współrzędne ruchu markerów są generowane podczas nagrywania aktora w studiu Motion Capture. Nagrywany ruch, użyty do badań, to spacer. Warto podkreślić, że dane podczas nagrywania aktora, są zapisywane do pliku trójwymiarowego (C3D), który można otworzyć i edytować w programie Mokka [4]. Plik ten można następnie eksportować do pliku tekstowego, z którego korzysta program przeprowadzający interpolacje. Po pobraniu danych z pliku, program wyszukuje linie, które nie zawierają żadnych współrzędnych. Należy zwrócić uwagę na fakt, iż wyżej opisane „puste" linijki to inaczej klatki, w których doszło do zaniku markera, w wyniku czego współrzędne nie zostały przechwycone. Często zanik taki jest spowodowany przysłonięciem markera przez którąś z części ciała aktora lub jakimś innym działaniem zewnętrznym, najczęściej podczas skomplikowanych ruchów.

3.3. Interpolowanie dziur

Program, napisany w języku C++ wczytuje dane z pliku, które przedstawiają ruch aktora w przestrzeni trójwymiarowej, w postaci pliku tekstowego. Po znalezieniu dziur, czyli klatek nie zawierających współrzędnych, program pobiera numery klatek pustych oraz skrajnych zawierających współrzędne. Następnie program pobiera z tych skrajnych współrzędnych punktu kolejno współrzędne x, y, z i na ich podstawie, przy pomocy wybranej interpolacji, oblicza kolejno szukane współrzędne x, y, z.

3.4. Miara odległości Euklidesa

Różnice odległości między punktami macierzystymi a punktami po interpolacjach są obliczane ze wzoru na odległość między punktami w przestrzeni trójwymiarowej, według miary Euklidesa [5]. Odległość została przedstawiona na wzorze 4.

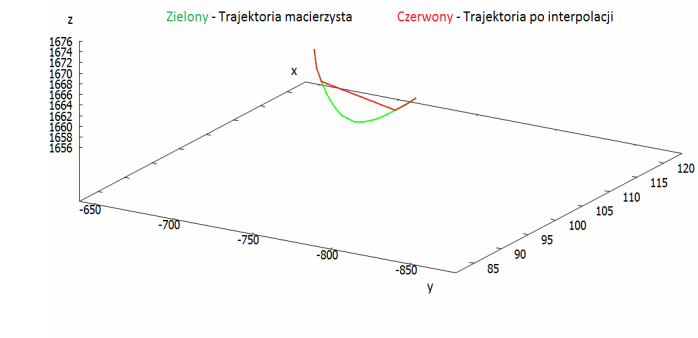
$$|AB| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2} \qquad (4)$$

gdzie: *x, y, z* – współrzędne punktów.

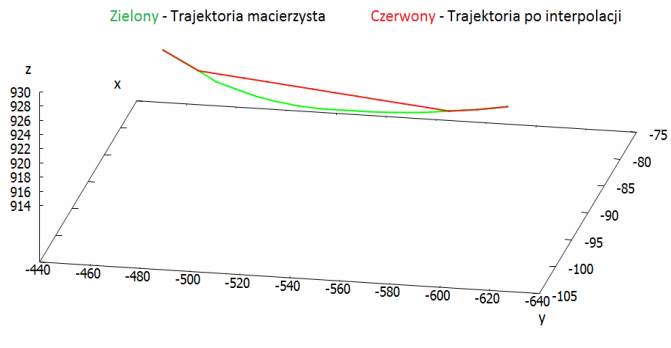
4. Wyniki

4.1. Interpolacja liniowa

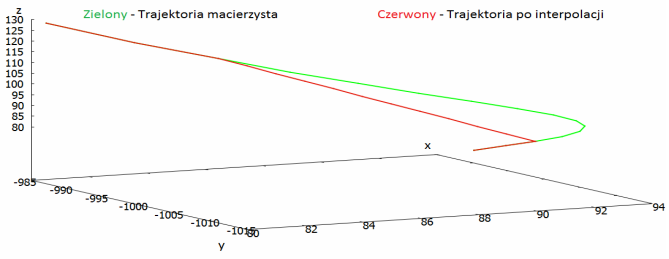
Wyniki interpolacji liniowej dla trzech markerów (LBHD, LRWA, LANK) są przedstawione na rysunkach 2,3,4. Na zielono przedstawiono wartość po interpolacji. Kolor czerwony ilustruje oryginalne dane.



Rys. 2. Wykres interpolacji liniowej dla markera LBHD



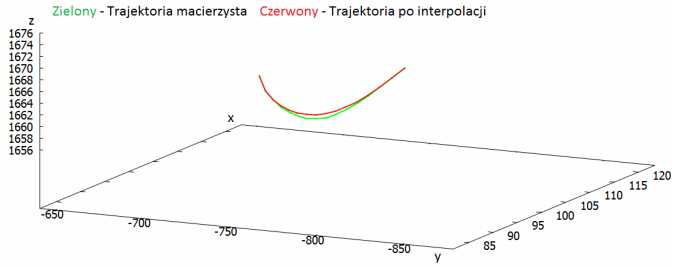
Rys. 3. Wykres interpolacji liniowej dla markera LWRA



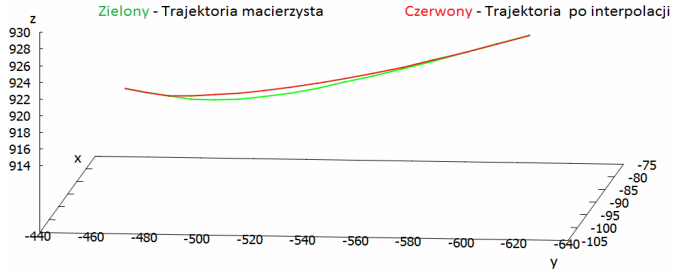
Rys. 4. Wykres interpolacji liniowej dla markera LANK

4.2. Interpolacja Lagrange'a wielomianem drugiego stopnia

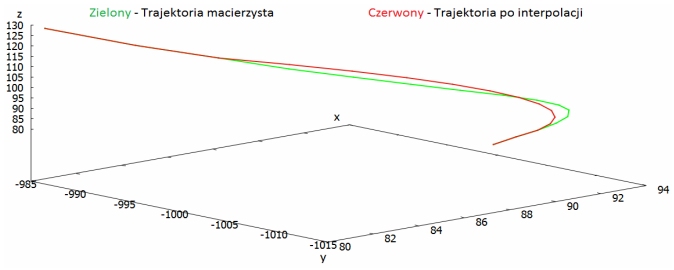
Wyniki interpolacji Lagrange'a wielomianem drugiego stopnia dla trzech markerów (LBHD, LRWA, LANK) są przedstawione na rysunkach 5,6 oraz 7.



Rys. 5. Interpolacja Lagrange'a wielomianem drugiego stopnia – LBHD



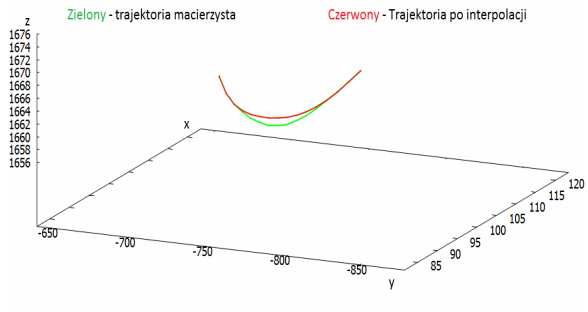
Rys. 6. Interpolacja Lagrange'a wielomianem drugiego stopnia – LWRA



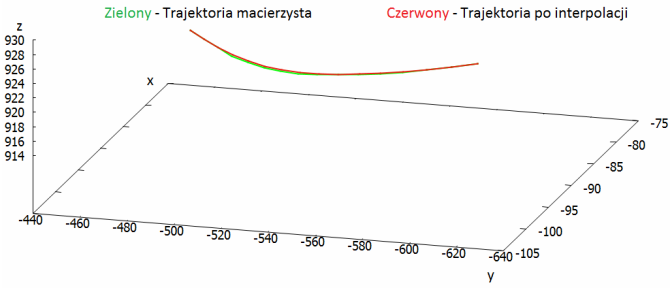
Rys. 7. Interpolacja Lagrange'a wielomianem drugiego stopnia – LANK

4.3. Interpolacja Lagrange'a wielomianem piątego stopnia

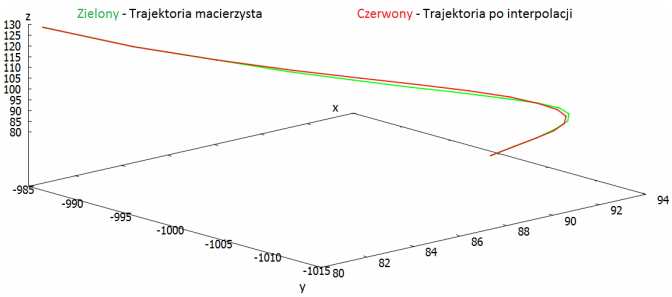
Wyniki interpolacji Lagrange'a wielomianem piątego stopnia dla trzech markerów (LBHD, LRWA, LANK) są przedstawione na rysunkach 8,9 oraz 10.



Rys. 8. Interpolacja Lagrange'a wielomianem piątego stopnia - LBHD



Rys. 9. Interpolacji Lagrange'a wielomianem piątego stopnia - LWRA



Rys. 10. Interpolacji Lagrange'a wielomianem piątego stopnia - LANK

4.4. Błąd odległości

Średni błąd odległości markerów macierzystych, a markerów po wyliczonych interpolacjach, przy użyciu wybranych metod jest przedstawiony w tabeli 1.

Tabela 1. Średni błąd metod interpolacji, dla poszczególnych markerów

Marker	Interpolacja Liniowa	Interpolacja Lagrange'a wielomianem drugiego stopnia	Interpolacja Lagrange'a wielomianem piątego stopnia
LBHD	3,18	0,53	0,71
LWRA	3,27	0,56	0,35
LANK	6,45	1,15	0,36

5. Wnioski

Za pomocą przedstawionych rysunków oraz wyników ukazanych w poszczególnych tabelach można wyciągnąć następujące konkluzje: 1) Interpolacja liniowa jest najmniej efektywna i nie sprawdza się dla trajektorii, której torem nie jest linia prosta; 2) Interpolacja Lagrange'a wielomianem drugiego stopnia sprawdza się najlepiej dla trajektorii o ruchu zbliżonym do parabolicznego, ponieważ interpolacja jest obliczana dla trzech znanych punktów. Przykładem takim jest marker LBHD, którego ruch cyklicznie zbliżony jest do paraboli, w wyniku czego interpolacja w tym przypadku okazała się najdokładniejsza; 3) Interpolacja Lagrange'a wielomianu piątego stopnia okazała się najskuteczniejsza w ruchu dla wszystkich markerów, poza LBHD.

Nie można pominąć faktu, że im więcej punktów jest użytych do interpolacji, tym większy jest stopień wielomianu interpolacyjnego, dzięki czemu obliczenia są dokładniejsze. Należy wtedy pamiętać, że czas obliczeń jest dłuższy. Dlatego powinno się odpowiednio dobierać interpolację do potrzeb. Na przykład, w przypadku ruchu po linii prostej, nie należy od razu wybierać interpolacji wielomianowej, lecz wskazane jest skorzystanie z interpolacji liniowej, w efekcie czego wyniki zostaną obliczone, w krótszym czasie, nie zajmując cennej pamięci komputera. Natomiast w przypadku trajektorii, w której ruch cyklicznie lub w całości przypomina parabolę, odpowiednim wyborem jest interpolacja Lagrange'a wielomianem drugiego stopnia, który da wyniki identyczne, a może nawet lepsze niż wielomian o wyższym stopniu.

Literatura

[1] R. Parent, Animacja Komputerowa – Algorytmy i Techniki, Warszawa 2012, s. 210, 213-214.

[2] Z. Fortuna, B. Macukow, J. Wąsowski: Metody numeryczne. Warszawa: Wydawnictwa Naukowo-Techniczne, 1993. ISBN 83-204-1551-9.

[3] Plug-in Gait Model, www.irc-web.co.jp/vicon_web/news_bn/PIGManualver1.pdf

[4] MOKKA Motion Kinematic and Kinetic Analyzer, http://biomechanical-toolkit.github.io/mokka/

[5] W. Rudin: Podstawy analizy matematycznej. Warszawa: 2005, s. 18-19

JCSI 2 (2016) 85-88
Journal of Computer Sciences Institute

Wysłane: 2016-10-15
Przyjęte: 2016-10-17

Analiza możliwości zastosowania środowiska Unity 3D w tworzeniu symulacji postaci

Aleksandra Agnieszka Woźniak*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Niniejsza praca przedstawia analizę możliwości zastosowania środowiska Unity 3D w kontekście symulacji postaci. Jej celem jest analiza porównawcza procesu animacji postaci w wyżej wymienionym środowisku. Głównym aspektem badawczym jest proces symulacji ruchu, który zrealizowano w projekcie na dwa sposoby. Dostrzeżono przewagę procesu symulacji ruchu opierającego się na blend tree, który cechował się większą płynnością i jak się okazało, także większą wydajnością niż animacja stanowa.

Słowa kluczowe: Unity 3D, symulacja postaci; blend tree; animacja stanowa

*Autor do korespondencji.

Adres e-mail: wozniak.aleksandra92@gmail.com

Performance analysis of Unity 3D environment in development of a character simulation

Aleksandra Agnieszka Woźniak*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This work presents the performance analysis of Unity 3D environment character simulation development. Ihe purpose of this work is a comparative analysis of character animation processes in the aforementioned environment. The analysis is based on motion simulation process, which has been shown in the two different ways. The results showed advantage of motion simulation process based on the blend tree, which was characterized by greater fluency and as it also turned out higher efficiency than the state animation.

Keywords: Unity 3D; character simulation; blend tree; state animation

*Corresponding author.

E-mail address: wozniak.aleksandra92@gmail.com

1. Wstęp

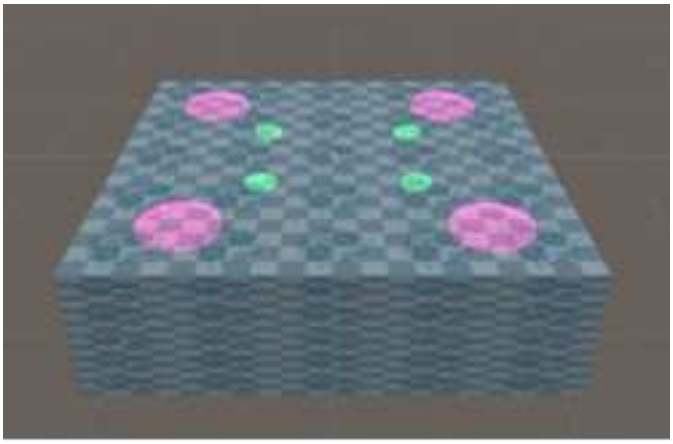
Trójwymiarowe postacie humanoidalne są powszechnie zauważalne w grach, filmach, aplikacjach mobilnych jak i webowych. Zastosowanie modeli 3D w grze komputerowej lub symulacji środowiska (tzw. terrarium) wymaga odpowiedniej interwencji w celu skojarzenia takiego modelu z odpowiednim zestawem ruchu i mechanizmów kontrolnych. Włączenie takiego modelu w system animacji jest procesem złożonym, dającym wiele możliwości poczynając od lokomocji, manipulacji obiektem a kończąc na syntezie mowy i synchronizacji ruchu warg. Wytwarzanie trójwymiarowej postaci humanoidalnej składa się z wielu etapów, w tym wykonania modelu postaci, określenia szkieletu dla tego modelu, manipulacji modelem w zależności od ruchu szkieletu, dołączenia algorytmów ruchu oraz kontroli, a ostatecznie instruowania postaci do wykonywania ruchów. Ze względu na popularność i szeroki zakres zastosowania modeli 3D powstaje coraz więcej rozwiązań przeznaczonych do ich obsługi. Takim rozwiązaniem jest również zintegrowanie środowisko oparte o silnik do tworzenia gier Unity 3D.[1,2,3,5]

1.1. Założenia projektu

Celem pracy jest analiza możliwości zintegrowanego środowiska Unity 3D w kontekście symulacji postaci. Analizę oparto na projekcie symulacji postaci w wyżej wymienionym środowisku. Głównym aspektem badawczym jest proces symulacji ruchu, który zrealizowano w projekcie na dwa sposoby: animacja stanowa oraz blend tree. Następnie typy animacji poddano wnikliwej analizie oraz rzetelnemu porównaniu, na których oparto wnioski niniejszej pracy. Założono, że animacja opierająca się o blend tree, ze względu na znacznie większe możliwości tzw. miksowania ruchu będzie w sposób bardziej realistyczny oraz płynniejszy ukazywała ruch postaci humanoidalnej, a niżeli animacja stanowa. Kolejną hipotezą wynikającą ze złożoności animacji blend tree stała się jej mniejsza wydajność w stosunku do animacji stanowej. Na potrzeby wyżej postawionych tez skorzystano z gotowego modelu postaci typu rigged and skinning, posiadającego gotowy zestaw animacji.

2. Projekt symulacji postaci oraz jej animacji

Badanie oparte zostało na dwóch oddzielnie stworzonych w Unity 3D projektach symulacji ruchu. Obie symulacje zostały przeprowadzone w takich samych warunkach. Na potrzeby projektu została stworzona plansza utworzona z obiektu typu cube, po której poruszać się będzie analizowana postać. Plansza zawiera również obiekty, które wymuszają na postaci ściśle określone zachowania w momencie kolizji. Kolizja z obiektem goal (różowa kapsuła) – symulacja animacji zwycięstwa. Obiekt checkpoint (zielona kapsuła) – po spadnięciu postaci z planszy powraca ona w miejsce ostatniej kolizji z obiektem tego typu przed upadkiem.



Rys. 1. Widok planszy po której poruszają się postacie.

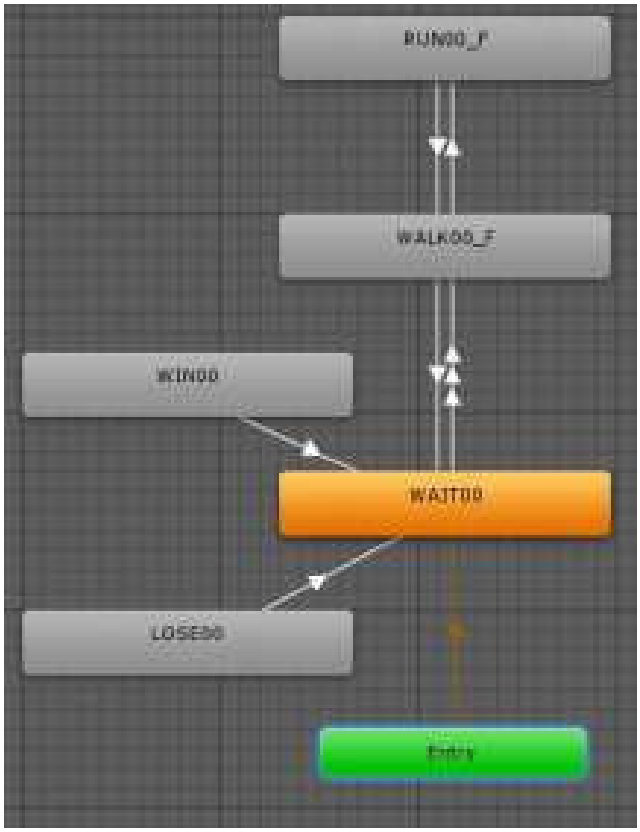
Dla pełnego wykorzystania symulacji ruchu, wybrano model humadoidalny (model postaci Unity Chan dostępny pod licencją Unity Technologies Japan G.K)[9], posiadający rigged oraz skinned typ siatki (ang. mesh). Termin rigged oznacza proces tworzenia hierarchii połączeń inaczej zwanego szkieletem. Definiuje on położenie oraz przemieszczanie się względem siebie kości znajdujących się w siatce reprezentującej obiekt 3D. Proces ten pozwala na pełną kontrolę ruchu postaci. Jeżeli chodzi natomiast o termin skinned jest to proces łączenia szkieletu z siatką postaci. Wybrana postać zawiera również zestaw gotowych animacji, które wykorzystano na potrzeby opisanego w niniejszej pracy projektu.[1,4,6].

.

Rys. 2. Wykorzystany model postaci Unity Chan dostępny pod licencją Unity Technologies Japan G.K

2.1. Animacja stanowa (ang. state)

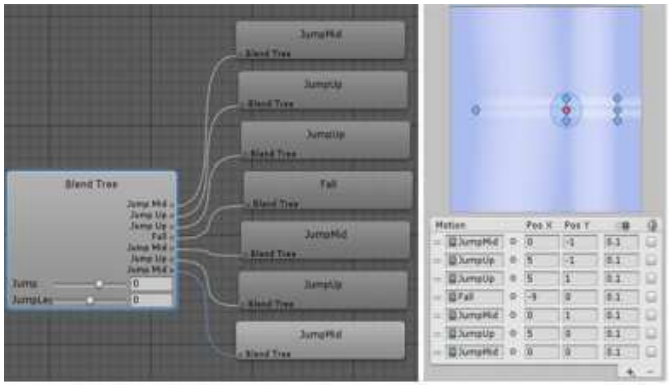
Przedmiotem badania jest analiza porównawcza dwóch rozwiązań symulacji ruchu postaci. W pierwszym projekcie animacja stworzona została poprzez animacje stanową. Główną ideą animacji stanowej jest określona hierarchia wykonywanych ruchów. Postaci można nadać animacje, które będzie wykonywała w zadanej kolejności oraz pod zdefiniowanymi warunkami. Mówiąc ogólnie postać posiada restrykcje przejścia z jednego stanu w drugi i nie jest możliwe jej bezpośrednie przejście z dowolnego stanu w inny. Dla przykładu, jeżeli chcemy uzyskać animację skoku podczas biegu to możliwość przejścia do niej występuje tylko w przypadku, gdy postać jest już w stanie biegu, a nie w stanie spoczynku.



Rys. 3. Schemat animacji stanowej stworzonej na potrzeby projektu symulacji

2.2. Blend tree

W drugim projekcie animacja stworzona została poprzez blend tree. Blend tree pozwala na tzw. miksowanie animacji. Umożliwia płynne mieszanie wielu animacji poprzez wcielanie ich części między siebie w różnym stopniu. Udział poszczególnych części animacji w efekcie końcowym kontrolowany jest przez parametry mieszające (ang. blending parameter).[7]



Rys. 4. Schemat blend tree animacji skoku (po lewej) oraz jego parametry (po prawej) stworzony na potrzeby projektu symulacji

2.3. Analiza porównawcza względem parametrów wizualnych

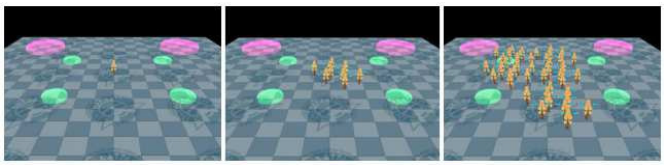
Pierwszy etap analizy miał na celu porównanie obu typów animacji pod względem wizualnym. W obu

przypadkach przedmiotem był obiekt postaci poruszający się po planszy. Porównania dokonano dzięki stworzonym na potrzeby analizy parametrom wizualnym. Wśród wyróżnionych kryteriów znalazły się: płynność ruchu, łączenie animacji, manipulacja zachowania obiektu.

Płynność ruchu jest parametrem opisującym płynność przejść animacji z jednej w drugą oraz stopień, w jakim ruch przypomina realistyczne zachowanie postaci humanoidalnej. Łączenie animacji opisuje w jaki sposób animacje zostały połączone ze sobą, natomiast manipulacja zachowania obiektu świadczy jakie elementy miały wpływ na kontrolę ruchu postaci

2.4. Analiza porównawcza względem parametrów wydajnościowych

Kolejny etap analizy miał na celu porównanie obu typów animacji pod względem wydajnościowym, przy pomocy wbudowanego w środowisko Unity Profilera. Jako główny parametr wzięto pod uwagę zużycie procesora. Analiza wydajnościowa składała się z trzech etapów. W pierwszym etapie wyświetlono jeden obiekt, w następnym 10, a w ostatnim 50. Podczas każdego z nich zebrane zostały dane dotyczące zużycia procesora w trakcie działania poszczególnych animacji.



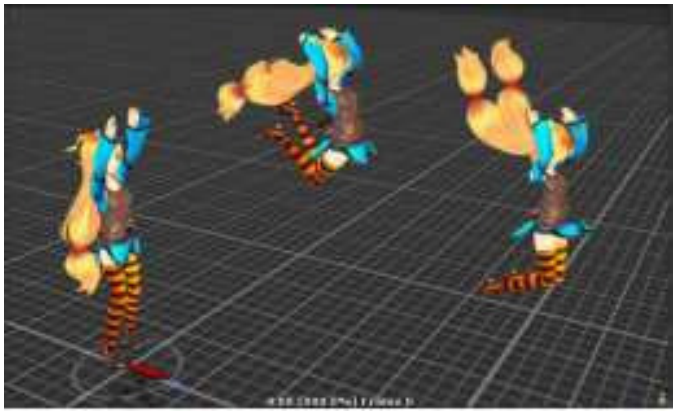
Rys. 5. Etapy analizy wydajnościowej (kolejno): etap I, etap II, etap III

3. Otrzymane wyniki analizy

3.1. Analiza względem parametrów wizualnych

Dla animacji stanowej przejścia okazały się być ostre. Na pierwszy rzut oka ruch jest zbliżony do realistycznego i wydaje się być całkiem naturalny, jednak przy dłuższej obserwacji zauważono wyraźne przejścia z jednego stanu w drugi. Jako przykład warto podać animacje chodu oraz biegu. Postać w pewnym momencie zaczyna bieg i nie wykazuje dla tego elementu przyspieszenia, jakie pojawia się w ruchu naturalnym podczas przechodzenia ze stanu chodu do biegu. Co za tym idzie zauważono, że animacje następują jedna po drugiej nie przenikając się w żadnym stopniu. Jeżeli chodzi o manipulacje zachowania obiektu stwierdzono, że na postać działa tylko fizyka. Dla animacji skoku działająca na postać siła grawitacji sprawia, że postać porusza się w górę i w dół, ale nie wykonuje animacji skoku tylko ciągle biegnie.

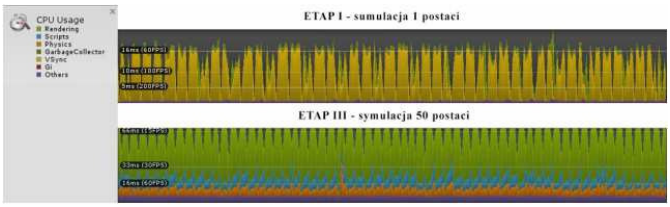
Dokonując analizy animacji blend tree stwierdzono, że symulacja w pełni przypominała poruszanie się obiektu w sposób realistyczny. Postać płynnie przechodziła z chodu w bieg co powodowało naturalność ruchu z jakim się poruszała. Zauważono również, że animacje miksują się między sobą. Postać z chodu poprzez przyspieszanie zaczyna biec. Na podstawie obserwacji skoku stwierdzono, że postać nie tylko porusza się w górę i w dół (co sprawia siła grawitacji), lecz również wykonuje animację skoku. Głównym efektem, na którego podstawie wysunięto wniosek był ruch włosów oraz rąk. Podczas wznoszenia się postaci w górę włosy oraz ręce płynnie się unosiły, natomiast w końcowej fazie skoku płynnie opadały.



Rys. 6. Przebieg ruchu podczas animacji skoku dla blend tree

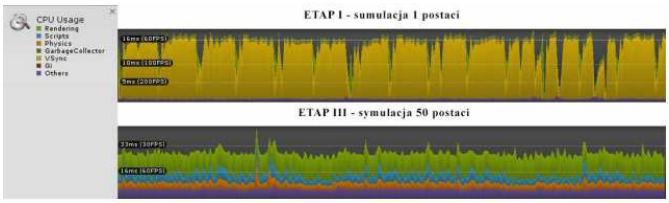
3.2. Analiza względem parametrów wydajnościowych

Na potrzeby niniejszej pracy zostaną przedstawione wykresy zużycia procesora dla obu animacji, dla etapu I oraz etapu III.



Rys. 7. Zużycie procesora dla animacji stanowej

Po przeanalizowaniu wykresów animacji stanowej, dla pierwszego etapu ilość oscyluje w granicy 60 FPS natomiast dla trzeciego 15 FPS.



Rys. 8. Zużycie procesora dla animacji blend tree

Jeżeli chodzi o animacje blend tree to podczas symulacji jednej postaci ilość oscyluje w granicy 60 FPS, natomiast dla 50 postaci 30 FPS..

4. Wnioski

Jak założono we wstępie animacja opierająca się o blend tree ukazywała ruch postaci humanoidalnej w sposób bardziej realistyczny oraz płynniejszy, aniżeli animacja stanowa.

Spowodowane jest to złożonością symulacji ruchu opartej o blend tree, której główną zaletą okazało się być miksowanie animacji. Jak już wspomniano w rozdziale 3.1. niniejszej pracy postać płynnie przechodziła z jednej animacji w drugą. Taki efekt próbowano również uzyskać poprzez animację stanową. Niestety, jak się spodziewano, w każdej sekwencji ruchu widać wyraźnie moment przejścia z jednego stanu w drugi, za co odpowiada tylko jeden parametr lub kilka warunków. Powoduje to, że ruch jest ostrzejszy niż w przypadku blend tree. Główną zaletą blend tree jest sposób manipulacji zachowania obiektu, na który wpływa miksowanie oraz fizyka, co nie występuje w animacji stanowej. Animacja stanowa opiera się bowiem wyłącznie na fizyce, za którą odpowiedzialny jest Rigidbody

Zaskoczeniem okazały się natomiast wyniki analizy wydajnościowej obu typów animacji. Założono, że ze względu na większą złożoność, blend tree będzie mniej wydajny niż animacja stanowa. Po analizie wykresów przedstawionych w rozdziale 3.2., blend tree okazało się być nie tylko wydajniejszym, ale również stabilniejszym rozwiązaniem niż animacja stanowa. O ile przy symulacji jednego obiektu wyniki w postaci liczby FPS są zbliżone (60 FPS) dla obu badanych rozwiązań, to podczas symulacji 50 obiektów wyniki diametralnie się różnią. W obu przypadkach zauważono spadek wydajności. Porównując jednak liczby dla blend tree zauważono spadek o 30 FPS natomiast w animacji stanowej o 45 FPS, co stanowiło aż 75%. Różnice zaobserwowano także wizualnie. Aplikacja przestała reagować w odpowiedni sposób na przesyłane sygnały sterujące, a co za tym idzie obraz zaczął klatkować. Ponieważ w wykresach animacji opartej o blend tree nie zauważono skoków wydajności jak w przypadku animacji opartej o stany, stwierdzono kolejną przewagę blend tree jaką jest stabilność rozwiązania

Na podstawie przeprowadzonych analiz i testów, należy stwierdzić, że Unity 3D okazało się doskonałym rozwiązaniem w zakresie symulacji postaci. W pełni pozwala na obróbkę modelu 3D dzięki rigged and skinned system. Środowisko umożliwia nie tylko symulacje modelu 3D, ale również daje wiele możliwości i rozwiązań w kontekście symulacji ruchu

Literatura

[1] Miller, Christian, Okan Arikan, and Don Fussell. "Frankenrigs: building character rigs from multiple sources." Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. ACM, 2010.

[2] Feng, Andrew, et al. "Fast, automatic character animation pipelines." Computer Animation and Virtual Worlds 25.1 (2014): 3-16

[3] Haas, John. A History of the Unity Game Engine. Diss. WORCESTER POLYTECHNIC INSTITUTE.

[4] Unity 5.4 documentation. "Preparing your own character". https://docs.unity3d.com/Manual/Preparingacharacterfromscratch.html. [14.10.2016]

[5] Murdoch, Steven. "Agent-oriented modelling in the production of 3D character animation." Studies in Australasian Cinema 10.1 (2016): 35-52.

[6] Suma, Evan A., et al. "Rapid generation of personalized avatars." 2013 IEEE Virtual Reality (VR). IEEE, 2013.

[7] Unity 5.4 documentation. "Blend Trees". https://docs.unity3d.com/Manual/AnimationStateMachines.html. [14.10.2016]

[8] Unity 5.4 documentation. "Animation State Machines" https://docs.unity3d.com/Manual/class-BlendTree.html [14.10.2016]

[9] Unity Chan – official website. "Unity-Chan License Terms – Summarized Version 2.00" http://unity-chan.com/contents/guideline_en/ [14.10.2016]

[10] Unity – Game Engine. https://unity3d.com/ [14.10.2016]

JCSI 2 (2016) 89-92
Journal of Computer Sciences Institute

Wysłane: 2016-10-15
Przyjęte: 2016-10-18

Analiza porównawcza biblioteki jQuery Mobile i frameworka Bootstrap w wytwarzaniu stron responsywnych

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

Streszczenie. Artykuł przedstawia analizę porównawczą technologii jQuery Mobile oraz frameworka Bootstrap w procesie wytwarzania stron responsywnych. Porównanie odbywa się na podstawie dwóch stron wykonanych w tych obu technologiach. Kryteriami analizy są: kompatybilność z urządzeniami mobilnymi, ułatwienia i gotowe komponenty, szybkość ładowania na różnych urządzeniach, wielkość kodu oraz zgodność ze standardami W3C i Google. Każda z technologii w dużym stopniu ułatwia proces tworzenia stron, aczkolwiek jQuery Mobile jest narzędziem znacznie bardziej rozbudowanym.

Słowa kluczowe: Bootstrap; jQuery Mobile; responsywność

*Autor do korespondencji.

Adres e-mail: martaawronska@gmail.com

Comparative analysis of jQuery Mobile library and Bootstrap framework in responsive websites development

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

Abstract. Article presents comparative analysis of jQuery Mobile library and Bootstrap framework in responsive site development. The comparative based on two websites, made in these technologies. Criterions of analysis are: compatibility of mobile devices, prepared components, loading speed on different devices, code size and compatibility with W3C and Google standards. Both technologies in a large degree helps in process of websites development, although jQuery Mobile is more expanded.

Keywords: Bootstrap; jQuery Mobile; Responsive Web Design

*Corresponding author.

E-mail address: martaawronska@gmail.com

1. Wstęp

Rozwój współczesnej technologii oraz wzrost ilości urządzeń mobilnych na rynku przyczynił się w ciągu ostatniego dziesięciolecia do potrzeby budowania interaktywnych i responsywnych witryn internetowych. Pojawiła się zatem technika Responsive Web Design (RWD), która aktualnie wiedzie prym. [1] Jej popularność sprawiła, że jest tematem poruszanym w wielu książkach i artykułach, między innymi w takich publikacjach jak: „Responsive Web Design" autorów : Harb, Kapellari, Luong i Spot (2011), w książce autorów Jaya Bryana i Mike'a Jones'a „Pro HTML5 Performance" (2012), gdzie jest jej poświęcony osobny rozdział oraz w książce „HTML5 and CSS3 Responsive Web Design Cookbook" Benjamina LaGrone (2013). W związku z powyższym powstają gotowe narzędzia mające na celu ułatwić deweloperom proces wytwarzania stron zgodnych z techniką RWD. Popularnymi tego typu technologiami są między innymi: jQuery Mobile oraz framework Bootstra i to one są głównym tematem niniejszej pracy.

jQuery Mobile jest to biblioteka wywodząca się i stworzona przez autorów biblioteki jQuery do wytwarzania aplikacji webowych. Aplikacje tworzone w jQuery Mobile są kompatybilne ze wszystkimi urządzeniami mobilnymi, w przeciwieństwie to aplikacji natywnych.[2] Tworzone są, podobnie jak zwykłe strony internetowe, do działania w środowisku przeglądarek internetowych. jQuery Mobile daje również zestaw gotowych rozwiązań dla interfejsu użytkownika, takich jak widgety, animacje czy elementy nawigacji. Twórcy tej biblioteki działali zgodnie z zasadą *Progressive Enhancement*, czyli techniką narzucającą oddzielenie warstwy treści i semantyki od warstwy prezentacji. [3]

Bootstrap natomiast to lekkie narzędzie do tworzenia responsywnych stron internetowych, rozwijane przez programistów Twittera. Podobnie jak jQuery Mobile, Bootstrap zawiera zestaw gotowych narzędzi ułatwiających tworzenie graficznego interfejsu użytkownika.[4]

Obie te technologie warte są porównania pod wieloma względami – współczesny programista może więc wybrać najbardziej odpowiednią dla siebie opcję. Celem artykułu jest stworzenie dwóch stron – jednej przy użyciu biblioteki jQuery Mobile, drugiej przy użyciu frameworka Bootstrap, a następnie porównane ich ze sobą na podstawie kilku kryteriów, opisanych szerzej w rozdziale nr 2. W celu ułatwienia, w dalszej części artykułu, stroną nr 1 będzie nazywana strona stworzona przy użyciu jQuery Mobile, natomiast stroną nr 2, strona stworzona przy użyciu Bootstrapa.

2. Kryteria porównania

Strony stworzone na potrzeby tej analizy zostały poddane badaniom na podstawie kilku kryteriów. Są to: wygląd stron na urządzeniach mobilnych, czas ładowania stron na komputerach stacjonarnych i na różnych urządzeniach mobilnych, zgodność ze standardami Google i W3C oraz wielkość kodu. Użyte narzędzia to:

- http://ami.responsivedesign.is – wirtualne narzędzie pokazujące wygląd stron na różnych urządzeniach,
- http://webspeed.intensys.pl – narzędzie analizujące stronę pod różnymi kątami, między innymi wielkość kodu, czas ładowania itp.,
- https://developers.google.com/speed/pagespeed/insights – strona dla deweloperów, służąca do testowania, w tym przypadku sprawdzające szybkość strony na komputerze i urządzeniu mobilnym oraz wygodę użytkownika (tzw. *User Experiance*), podczas korzystania z interfejsu graficznego na urządzeniu mobilnym,
- autorski skrypt napisany w języku JavaScript, liczący czas ładowania strony na rzeczywistych urządzeniach.

3. Porównanie stron

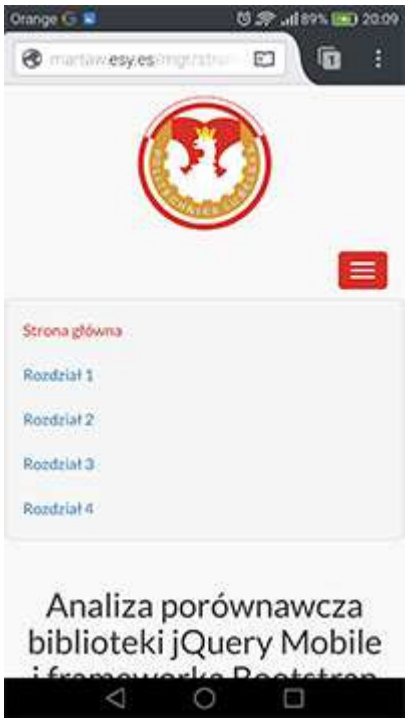
Wizualnie strony są do siebie bardzo podobne, szczególnie na rozdzielczościach powyżej 1200px. Z powodu różnych punktów zmiany szerokości elementów ich wygląd będzie się zmieniał wraz ze zmniejszaniem rozdzielczości. Dla przykładu na rysunkach nr 1 i nr 2 przedstawione jest menu wyświetlone na monitorze o rozdzielczości 1366px dla obu stron.

Rys. 1. Wygląd strony nr 1 na ekranie monitora

Rys. 2. Wygląd strony nr 2 na ekranie monitora

Natomiast rysunki nr 3 i nr 4 przedstawiają strony na urządzeniach mobilnych.

Rys. 3. Wygląd strony nr 1 na urządzeniu mobilnym



Rys. 4. Wygląd strony nr 1 na urządzeniu mobilnym

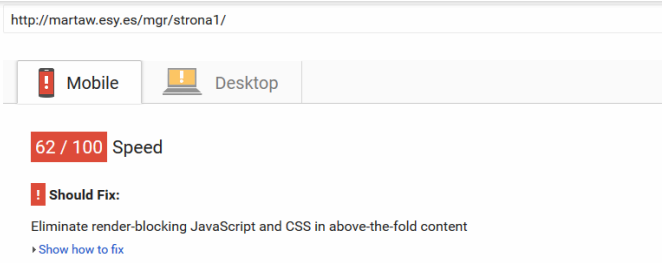
Jak można zauważyć wygląd stron na urządzeniu mobilnym różni się od siebie, co w tym przypadku jest spowodowane tym, że Bootstrap w przeciwieństwie do jQuery Mobile posiada komponent, który automatycznie tworzy responsywne zwijane menu [5].

4. Analiza

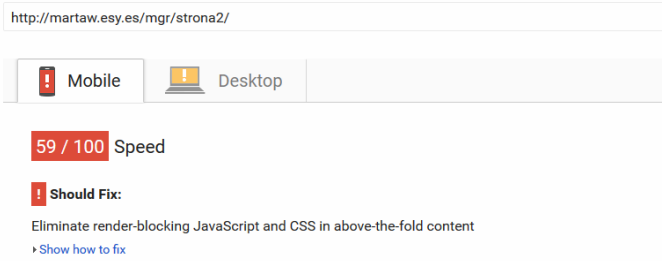
W fazie implementacji można zauważyć pewne różnice i podobieństwa. W przypadku frameworka Bootstrap, można mówić o jego „wyższości" nad jQuery Mobile w kwestiach tworzenia szkieletu całej strony, tak zwanego systemu siatkowego. Jest on bardziej intuicyjny i daje większe możliwości. Pozwala w dowolny sposób zmieniać szerokości elementów w danych punktach rozdzielczości. jQuery Mobile natomiast posiada stały punkt tzw. *breakpoint* i wynosi on 560px. [6] Jest to jedyny punkt, w przypadku Bootstrapa jest ich cztery. Ponadto Bootstrap posiada wbudowane responsywne menu, tzw. *hamburger,* czego nie zapewnia jQuery Mobile, a co jest istotne w przypadku większych witryn z dużą ilością pozycji w menu. Jednakże obie technologie są podobne pod względem ilości kodu jaki trzeba napisać, aby uzyskać dany efekt.

Po zakończeniu implementacji, strony zostały poddane szczegółowej analizie pod kątem szybkości, wydajności i zgodności ze standardami Google oraz organizacji W3C. Jednak najważniejszym testem był test responsywności przeprowadzony za pomocą symulatora on-line na stronie http://ami.responsivedesign.is, który obie strony przeszły bezbłędnie.

Kolejną badaną kwestią jest zgodność obu stron ze standardami Google dla komputerów i urządzeń mobilnych. Analiza została przeprowadzona na stronie https://developers.google.com/speed/pagespeed/insights. [7]. W przypadku strony nr 1 zgodność wyniosła 62%, dla strony nr 2 - 59%. Taki wynik spowodowany jest między innymi brakiem kompresji skryptów, kodu HTML oraz CSS. Wyniki analizy są przedstawione na rysunkach 5 i 6.



Rys. 5. Analiza zgodności z urządzeniami mobilnymi dla strony nr 1



Rys. 6. Analiza zgdoności z urządzeniami mobilnymi dla strony nr 2

Kolejną przeprowadzoną analizą od Google Developers jest zgodność stron z urządzeniami mobilnymi. Dotyczy to tzw. *User Experiance,* czyli wygody użytkownika. W przypadku strony nr 1 wynik tej analizy to 99%, a uwagi jakie zostały zaproponowane to zwiększenie wielkości czcionki. Dla strony nr 2 wynik to 100%, czyli pełna zgodność.

Następnym testem jest czas ładowania i powtórnego wczytywania strony, przeprowadzony przy pomocy narzędzia on-line na stronie https://webspeed.intensys.pl. Wyniki okazały się być bardzo podobne, jak strony nr 1 czasy wyniosły odpowiednio 1.427s oraz 1.46s, natomiast dla strony nr 2 czasy te wyniosły 1.467s oraz 1.5s. Innym porównaniem jest wielkość kodu, tj. skryptów JavaScript, kodu HTML i CSS. Dla strony nr 1 obliczono 884KB, a dla strony nr 2 538KB. W przypadku stron otwieranych na urządzeniach mobilnych ważne jest jak najmniejsze obciążenie.

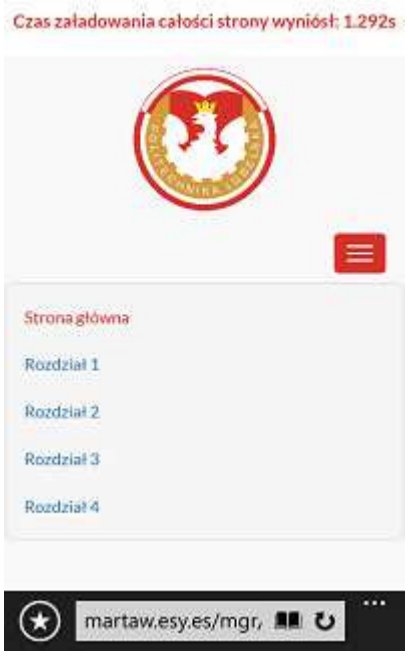
Kolejnym kryterium porównawczym jest również szybkość ładowania stron, jednakże test został przeprowadzony na rzeczywistych urządzeniach. W celu przeprowadzenia tej analizy został napisany autorski skrypt napisany w języku JavaScript liczący czas ładowania strony. Pierwsza część skryptu znajduje się zaraz po otwarciu sekcji *head*, przed załadowaniem całej strony i liczy ona aktualny czas. Natomiast druga część skryptu umieszczona na samym końcu dokumentu, liczy czas całkowitego ładowania strony. Kod progrmu znajduje się na listingu nr 1.

Przykład 1. Kod programu

```
<script type="text/javascript">
    var timeNow = Date.now();
 </script>
    [...]
<script type="text/javascript">
    $(window).load(function() {
          time = (Date.now()-timeNow)/1000;
    });
</script>
```

Test został przeprowadzony na czterech urządzeniach mobilnych, posiadających rożne systemy operacyjne oraz przeglądarki. Dla ułatwienia odczytania wyniku został on wyświetlony w części front-endowej, na górze strony. Przykładowo na rysunku nr 7 i nr 8 pokazane są wyniki działania skryptu dla obu stron na telefonie Nokia Lumia 520 z systemem operacyjnym Windows Phone w wersji 8.1.

Rys. 7. Czas ładowania strony nr 1 na urządzeniu Nokia Lumia 520 z systemem operacyjnym Windows Phone 8.1



Rys. 8. Czas ładowania strony nr 2 na urządzeniu Nokia Lumia 520 z systemem operacyjnym Windows Phone 8.1

Liczone czasy zostały podane po pierwszym załadowaniu strony, aby uniknąć przekłamania wartości po powtórnym przeladowaniu. Jak widać, strona stworzona na frameworku Bootstrap wczytała się szybciej.

Z przeprowadzonej analizy wynika, że na każdym badanym urządzeniu czas wczytywania strony zbudowanej na frameworku Bootstrap jest mniejszy niż czas ładowania strony zbudowanej na podstawie biblioteki jQuery Mobile. Tabela nr 1 przedstawia wyniki całościowej analizy, czyli czasy ładowania stron na poszczególnych urządzeniach.

Tabela 1. Wyniki czasów ładowania stron na poszczególnych urządzeniach

	Strona nr 1	Strona nr 2
Huawei P9 - Android 6.0	0.940s	0.730s
Nokia Lumia - Windows Phone 8.1	3.118s	1.292s
Samsung - Android 4.4.2	3.153s	0.982s
Iphone 5s - IOS 10.0.2	0.325s	0.858s

5. Wnioski

Po przeprowadzeniu dokładnej analizy, można wysnuć pewne wnioski. Na etapie impementacji istnieje wiele podobieństw, między innymi ilość kodu jaką trzeba napisać, aby uzyskać podobny efekt. Strona pisana w jQuery Mobile minimalnie szybciej wczytuje się na laptopie lub komputerze stacjonarnym, natomiast wolniej na urządzeniach mobilnych. Rozmiar plików tej biblioteki jest prawie dwa razy większy niż Bootstrapa, co może mieć wpływ na czas ładowania np. na starszym typie smartfona. W przypadku dużych witryn te parametry mogą być prawie niewidoczne dla użytkownika. Należy również pamiętać, że jQuery Mobile to narzędzie bardzo rozbudowane i w niniejszej pracy zostały ukazane jedynie jego pewne aspekty. Jeśli chodzi o aspekt wizualny, strony dobrze wyglądają na wszelkich urządzeniach. Podsumowując, obie technologie są warte przetestowania, a ich możliwości pomagają zmniejszyć czas poświęcony na kodowanie strony. Pozwalają one programiście zapomnieć o zajmujących czas problemach, a skupić na konkretnych funkcjonalnościach.

Literatura

[1] Plechawska-Wójcik, Malgorzata, Sergio Luján-Mora, and Lukasz Wójcik: Assessment of User Experience with Responsive Web Applications using Expert Method and Cognitive Walkthrough-A Case Study. ICEIS (3). 2013.

[2] Jon Reid: jQuery Mobile, Helion, 2012.

[3] Adrian de Jonge, Phil Duston: jQuery, jQuery UI oraz jQuery Mobile, receptury, Helion, 2013.

[4] Syed Fazle Rahman: Bootstrap. Tworzenie interfejsów stron WWW. Technologia na start!, Helion, 2015.

[5] http://getbootstrap.com/components/ [20.09.2016]

[6] Maximiliano Firtman: jQuery Mobile: Up and Running, O'Reilly Media, 2013.

[7] https://developers.google.com/ [20.09.2016]

JCSI 2 (2016) 93-97
Journal of Computer Sciences Institute

Wysłane: 2016-11-15
Przyjęte: 2016-11-18

Budowa i animacja realistycznego modelu 3D mięśnia dwugłowego ramienia

Sebastian Poleszak* , Piotr Kopniak
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł dotyczy opracowania sposobu budowy modelu trójwymiarowego mięśnia dwugłowego ramienia człowieka tak aby mięsień wyglądał i zachowywał się realistycznie podczas animacji. Dane do animacji są pozyskiwane przez systemy akwizycji ruchu - Motion Capture. Do zaprojektowania modelu bryły wykorzystano oprogramowanie Blender v.2.76. Na jego podstawie możliwe jest przeprowadzanie badań dotyczących analizy ruchu ramienia z wykorzystaniem danych zapisanych w plikach BVH.

Słowa klucze: modelowanie 3D; animacja komputerowa; model biomechaniczny; analiza ruchu.

*Autor do korespondencji.

Adres e-mail: hisolmex@gmail.com

Construction and animation of realistic biceps 3D model

Sebastian Poleszak*, Piotr Kopniak
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article concerns the development of three-dimensional model of how to build biceps human with the muscle looked and behaved realistically when animation. Data for animation are obtained by acquisition systems Motion - Motion Capture. Blender software v.2.76 was used to design a solid model. On this basis it is possible to carry out research on the analysis of the movement of the arm using data stored in BVH files.

Keywords: 3D modeling; computer animation; biomechanical model; motion analysis.

*Corresponding author.

E-mail address: hisolmex@gmail.com

1. Wstęp

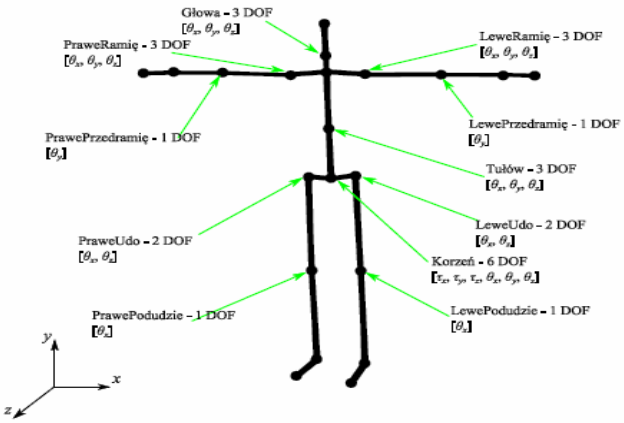
Modele trójwymiarowe postaci ludzkiej (tzw. modele biomechaniczne) stosowane są przede wszystkim w śledzeniu ruchu postaci ludzkiej. Opisują tym samym nie tylko kinematyczne właściwości ludzkiego ciała, jak chociażby model szkieletowy, ale również jego kształt, czyli ciało oraz skórę. Jest to wykonywane przeważnie przy zastosowaniu modelu dwuwarstwowego, w którym można wyróżnić model kinematyczny, jak również model kształtu. Dzięki temu model kinematyczny odpowiada w zasadzie za poruszanie, natomiast model kształtu zapewnia odpowiednią reprezentację kształtu poszczególnych części ludzkiego ciała.

Mówiąc o modelu kinematycznym trójwymiarowej postaci ludzkiej warto zauważyć, że zdecydowana większość modeli do opisu właściwości kinematycznych ludzkiego ciała wykorzystuje przeważnie kinematyczne drzewo składające się w szczególności z segmentów, zwanych inaczej kośćmi, które są połączone stawami. Jak wiadomo każdy ze stawów posiada określoną ilość stopni swobody, które określają sposób, w jaki dany staw może się poruszać. Wspomniany parametr, bardzo często jest określany mianem DOF(Degrees of Freedom), zaś stopnie swobody każdej z części modelu umożliwiają dokładne sprecyzowanie ruchów możliwych do wykonania przez ludzką postać. Zarówno orientacja jak i pozycja każdego z poszczególnych segmentów jest zdecydowanie najczęściej określana względem segmentu nadrzędnego w hierarchii drzewa przy wykorzystaniu macierzy transformacji albo też kwaternionu. Jest to równoznaczne z tym, że do wyznaczenia w pełni absolutnej orientacji, jak również i samego położenia segmentu w przestrzeni konieczne jest przeprowadzenie rekursywnej transformacji wzdłuż całej hierarchii, czyli począwszy od korzenia aż do danego segmentu [1].

Model kinematyczny może być tak samo dwu jak i trójwymiarowy, z tym, że model dwuwymiarowy wykorzystywany jest w większości przypadków w sytuacjach, kiedy śledzenie odbywa się na podstawie obrazów pochodzących z jednej kamery, natomiast śledzona postać porusza się równolegle do płaszczyzny obrazu. Pomimo tego, że modele dwuwymiarowe idealnie sprawdzają się w sytuacjach, kiedy śledzony jest ruch postaci ludzkiej, skierowanej całkiem na wprost albo też równolegle do kamery, to jednak brak możliwości uwzględnienia kątów w przestrzeni trójwymiarowej oraz dość dokładnego oszacowania głębokości jak też i wzajemnych przesłonięć powoduje, iż nie mogą być wykorzystywane z powodzeniem w śledzeniu bardziej złożonych czy tez skomplikowanych ruchów. W trójwymiarowych modelach postaci ludzkiej poszczególne segmenty są bardzo sztywne, zaś odpowiadające im stawy mają maksymalnie trzy stopnie swobody. Tego typu modele zdecydowanie najczęściej są wykorzystywane w systemach wielokamerowych, z uwagi na to, że analizując obraz pochodzący z wielu odpowiednio ustawionych kamer, możliwe jest jak najbardziej ustalenie parametrów, które określają konfigurację kątową poszczególnych stawów. Zatem chcąc dokonać dość dokładnego opisu pozy postaci ludzkiej niezbędne jest wykorzystanie swobody ruchu, która posiada przeszło 50 wymiarów, co tym samym daje olbrzymią liczbę hipotetycznych konfiguracji [2].

Istotnym zagadnieniem jest również odwzorowanie kształtu postaci ludzkiej. W tworzeniu modeli

trójwymiarowych mają zastosowanie przeważnie bryły geometryczne, a także modele powierzchniowe złożone z siatek wielokątów. Toteż w modelach, które powstały z brył geometrycznych niemal każdy segment jest reprezentowany przez pojedynczą bryłę, natomiast do ich budowy zdecydowanie najczęściej stosowane są cylindry, stożki cięte, stożki eliptyczne oraz superkwadryki jak też i inne. W przypadku modeli, które zostały oparte w dużej mierze na siatce wielokątów model przeważnie złożony jest z pojedynczej powierzchni, która przedstawia całe ciało postaci ludzkiej. W tej sytuacji ruch postaci ludzkiej jest wykonywany przy współudziale modelu kinematycznego oraz animacji szkieletowej. Warto zauważyć, że wykorzystywane są nie tylko modele złożone wyłącznie z siatek wielokątów, ale też i te charakteryzujące się niewielką liczbą wielokątów [3].



Rys. 1. Struktura kinematyczna trójwymiarowego modelu postaci ludzkiej [4]

Pierwszym etapem projektu było założenie, iż model biomechaniczny składać się będzie z układu kostnego oraz układu mięśniowego. Bryły tych poszczególnych obiektów musiały być zgodne z anatomią człowieka, której opis oparto o literaturę medyczną. Drugi etap projektowy związany był z wykonaniem realistycznego wyglądu kości oraz mięśni tego modelu biomechanicznego w wirtualnym środowisku przy pomocy odpowiedniego oprogramowania. Na ten etap przypada również proces związany z tworzeniem realistycznych tekstur danego obiektu bryły przestrzennej wraz z tworzeniem określonych map UV.

Kolejny etap dotyczył przygotowania modelu przystosowanego do animacji komputerowej. Warto tutaj dodać iż model ten musiał mieć pewne przygotowania związane z animacją komputerową . Odpowiednio ustawione limity zakresu ruchu dla poszczególnych brył , tak aby bryły te nie nachodziły na siebie , oraz mieć odpowiednią armaturę własną jak i ustawione poszczególne wagi dla obiektów. Po wykonanej analizie dostępności i właściwości istniejących modeli trójwymiarowych ciała ludzkiego okazało się , iż jest brak takich rozwiązań, które spełniałyby wszystkie kryteria. Dlatego też wystąpiła konieczność stworzenia nowego modelu na potrzeby pracy badawczej[5,6].

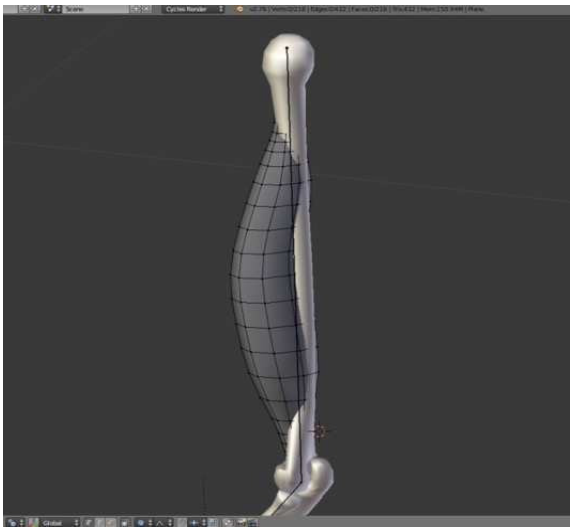
Pod kątem porównawczym z innymi darmowymi modelami [7,8] okazało się, iż modele te nie spełniają wszystkich niezbędnych wymogów do przeprowadzenia testów. Niektóre z nich nie nadawały się do przetestowania ruchu z systemu Motion Capture, ponieważ proces "armaturowania" nie był całkowicie wykonany. W modelach porównawczych wystąpiły błędy w postaci złej topologii siatki oraz nieprawidłowo wykonanego teksturowania obiektów, gdzieniegdzie zauważono pewne rozmycia co skutkowało ustawieniem słabej jakości tekstury. Jedynie modele z programu MakeHuman były odpowiednie do przeprowadzenia testów na nagranej sesji. Mimo wszystko, bryły obiektów nachodziły na siebie w pewnych momentach ruchu, co skutkowało brakiem przygotowania wag obiektów. Na tej podstawie określono hipotezę badawczą.

Hipoteza badawcza - jest możliwe opracowania trójwymiarowego modelu mięśnia, który będzie wyglądał i zachowywał się podczas ruchu zgodnie z opisem anatomicznym przy wykorzystaniu edytora Blender.

2. Budowa modelu mięśnia

Normal Map (na przykładzie projektowania "mięśni") Do tworzenia wypukłości na powierzchni brył służą Normalne. Podczas animacji komputerowej czy też symulacji ruchu w czasie rzeczywistym duże znaczenie ma liczba wierzchołków ponieważ liczba elementów geometrii znacząco wpływa na czas renderingu. Dlatego też należy pamiętać, aby tworzyć detal danego obiektu (w tym przypadku dla obiektu mięśni) wypalaniem danej geometrii i wykorzystaniem mapy normalnych (ang. normal map) [9,10].

Rys. 2. Obiekt kość ramienna człowieka



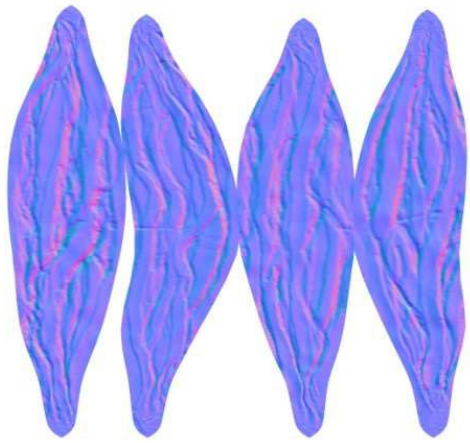
Rys. 3. Struktura kinematyczna trójwymiarowego modelu mięśnia dwugłowego.

Rys. 4. Model mięśnia dwugłowego - tryb rzeźbienia cyfrowego

Do wymodelowania obiektu tej bryły posłużono się dodaniem dwóch vertexów, a następnie dodano modyfikator „skin", który umożliwia szybkie tworzenie tzw. "base-mash'a", a na sam koniec dodano modyfikator "subdivice surface" służący do wygładzania obiektu jak również zmieniono tryb wyświetlania dla ścian na "smooth". W celu bardziej precyzyjnego dostosowania kształtu modelu mięśnia do referencji posłużono się opcją edycji proporcjonalnej. Następnie zduplikowano obiekt i wykonano detal dla kopii tego obiektu. W celu wykonania detalu dla tej bryły posłużono się opcją "sculp mode", a następnie z listy dostępnych narzędzi w postaci "pędzli" służących do rzeźbienia cyfrowego wykonano bardziej precyzyjny wygląd obiektu. Etap ten jest również niezwykle pracochłonny. Dlatego też na potrzeby danej pracy został tylko pokazany sposób projektowy.

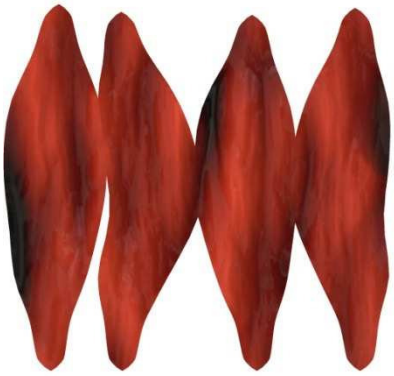
W programie takim jak Blender v.2.76 nie ma tak dopracowanej opcji służącej do tego typu projektowania jak w płatnym oprogramowaniu dedykowanym głównie do sculptinu jakim jest Z-Brush. Tryb „sculpt mode" w programie Blender opiera się na tworzeniu dodatkowej geometrii dla danego obiektu. Istnieje możliwość rzeźbienia dynamicznego czyli takiego, że w konkretnym miejscu, gdzie znajduje się obszar zaznaczany przez kursor myszki przy wciśniętym przycisku jest tworzona dodatkowa geometria. Jest to nieco problematyczny sposób, ponieważ przy dużym detalu to oprogramowanie zaczyna tracić swoją stabilność i znacząco zwalniać działanie programu. W tym przypadku skorzystano z dodatkowego modyfikatora "multiresolution", którego działanie opiera się na równomiernym tworzeniu dodatkowej geometrii dla całej powierzchni tej bryły. Jest on nieco bezpieczniejszy, ponieważ mając podzieloną siatkę dwu lub trzy krotnie można tworząc detal nie martwić się, iż w niektórych obszarach ta geometria będzie niezwykle zagęszczona jak to bywa w przypadku opcji "dyntopo" dla dynamicznego rzeźbienia.

Ważne jest, aby dla modyfikatora "multires" w opcji "preview" była wartość 0, ponieważ istotne jest, aby te nierówności wypaliły się na pierwotnej siatce. Powinny być zaznaczone następujące wartości: "bake from Mutires", czyli sczytywanie z modyfikatora multires, "bake mode: normals", czyli proces dla wektorów normalnych oraz zaznaczona opcja "clear" i 12px dla "margin". Jest to margines odstępu dla wypalonej normal mapy. Ważne jest, aby wypalać normal-mapę w trybie „obiekt mode". Następnym krokiem jest utworzenie nowej UV Map, po to aby nie było skalowania tekstury tylko ona automatycznie się powieli. Na tym etapie można mieć kilka UV MAP w celu lepszej precyzji. Pierwsza UV Map odnosić się będzie do wypukłości i nią należy skalować. Kolejna UV Map musi być identyczna co do wielkości tak, aby pokrywała się z normal-mapą dla tego obiektu.



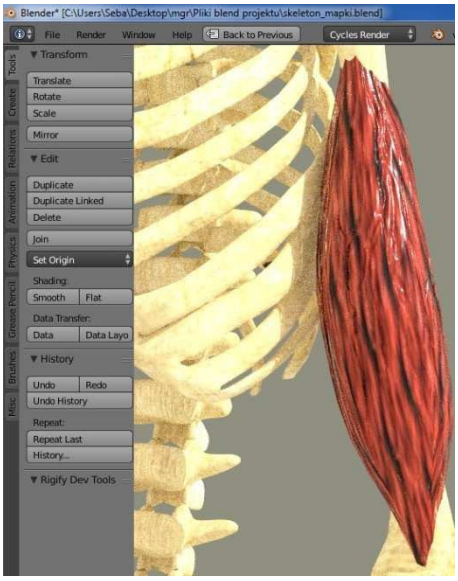
Rys. 5. Mięśnie normal-map

Powyżej (Rys. 5) przedstawiono prawidłowo przygotowaną normal-mapę w środowisku Blender. Warto dodać, iż nie każda normal - mapa występująca w tonacjach kolorów jasno fioletowych będzie prawidłowo interpretowana. Wynika to z faktu różnych algorytmów tworzących normal-mapy w programach graficznych. Dlatego też, warto jest wykonywać ten proces w tym samym programie, w którym wykonywany jest proces modelowania brył. Kolory te przechowują wartości, które reprezentują wysokości wgłębień.



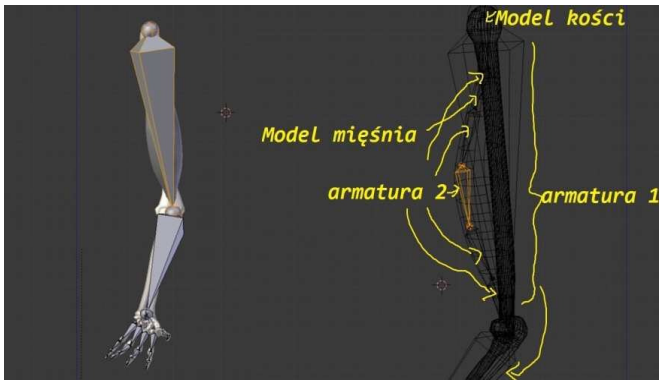
Rys. 6. Mięśnie textura-seamless

Powyżej (Rys. 6) przedstawiono jak wygląda prawidłowo wykonana tekstura dla obiektu "mięśnie". W celu lepszego odwzorowania skorzystano tu z tekstury bezszwowej. Jest to taki rodzaj tekstur, który jest powtarzalny dla danego obiektu. Jak widać obiekt wygląda realistycznie. Poprzez wykonaną normal-mapę można realnie odzwierciedlić wypukłości panujące na danym obiekcie i to bez konieczności interwencji w dodatkową geometrię tej bryły. Jest to technika, którą również stosuje się w animacji komputerowej czy projektowaniu modeli do silników gier [10,11].



Rys. 7. Mięśnie z texturą

Aby pokazać ruch uginania się mięśni należało przygotować zarówno armaturę dla całej bryły "mięśnia" (czyli rodzaj animacyjnego kośćca) jak i wewnątrz zrobić dodatkową armaturę, która wpływać będzie na proces uginania tej bryły w czasie ruchu. Jest to dość skomplikowana i pracochłonna operacja, w przypadku gdy chce się wiernie odzwierciedlić ruch, dlatego też na potrzeby tej pracy została pokazana metoda w nieco uproszczony sposób.



Rys. 8. Mięśnie - dodawanie armatury

Armatura 2 - nazwa tego obiektu została opisana jako: "biceps_kontroller" (składa się z 5 obiektów). Armatura 1 - nazwa tego obiektu została opisana jako: "actual_biceps". Ważnie jest, aby usunąć opcję "Parent" (usunięcie rodzica), czyli kość nadrzędną, a następnie dodać "offset" , czyli ta linia przerywana związana z kinematyką ruchu, w tym przypadku trzeba było skorzystać z odwrotnej kinematyki ruchu (Add IK). To tłumaczą prawa fizyczne i procesy zachodzące w realnym świecie. Następnie trzeba było przypisać taką samą rotację armaturze nr2 ("copy rotation"). Proces ten wykonuje się po to, aby przy rotacji ruchu kości ramiennej nie było deformacji siatki bryły (mesh).

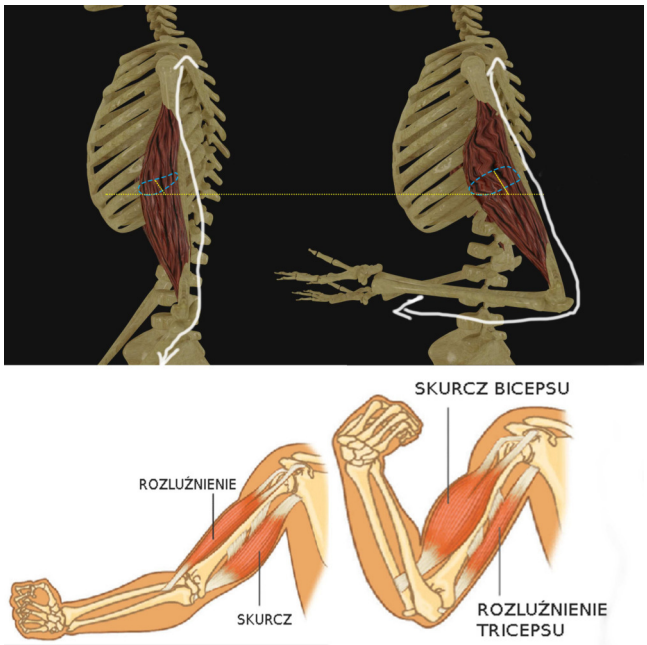
Następnie trzeba było połączyć te armatury (opcja join).

Aby mieć dokładną kontrolę nad deformacją bryły "mięśni" przy procesie ruchu tworzy się grupę vertexów. Będzie ona odpowiedzialna za ruch podczas, gdy poruszona będzie dana grupa kości (ang. rigg biceps). Tą grupę ustawia się dla obiektu "mięśni" (w trybie edit mode).

Następnie należy przygotować animację ruchu jako akcję zapisaną w programie Blender. Do tego posłużyło ustawianie klatek kluczowych oraz tryb „pose mode”, w którym ustawia się dokładny ruch obiektu. Również należy powtórzyć ten proces dla obiektów poszczególnych kości. Wartości wag dla obiektów kości należy ustawić na maksymalną (wartość 1), ponieważ kości się nie wyginają i są statyczne. Tak, aby poruszać tylko zaznaczonym obszarem. Wtedy uniknie się wszelkich błędów, przy wczytywaniu nagranych animacji z plików BVH. Ważna jest spójność nazewnictwa, tak aby grupy vertexów odpowiadały nazwom riggu dla kości.

3. Testy zbudowanego rozwiązania

Na potrzeby analizy ruchu z systemu akwizycji ruchu zaprojektowany model został poddany testom związanym z animacją komputerową. Ruch ugięcia mięśnia jest zgodny z ruchem rzeczywistym człowieka i nie zastąpi tego żadna animacja wykonana komputerowo. Do przetestowania rozwiązania wykorzystano wtyczkę importująca nagrane pliki BVH do programu Blender 2.76. Wszystko przebiegało pomyślnie i mięsień kurczył się prawidłowo.



Rys. 9. Prawidłowe reakcje mięśnia

Udało się wykonać realistyczny model mięsień, który działa anatomicznie na dowód tego przedstawiony został powyższy rysunek. Niezbędne było dodanie armatury dla całej bryły modelu "mięśnia " jako rodzaj kośćca animacyjnego oraz armatury dodatkowej która definiowała proces uginania się tej bryły w czasie na podstawie odpowiednich klatek kluczowych.

4. Wnioski

Podczas przeprowadzonego przeglądu, nie odnaleziono dostępnego darmowo modelu spełniającego wszystkie wymagania niezbędne do przeprowadzenia analizy ruchu z systemów motion capture. W związku z tym podjęto próbę zaprojektowania takiego modelu w programie Blender. Przy dodawaniu armatury do modelu należało pamiętać, iż ustawianie wag dla poszczególnych brył jest niezbędne do prawidłowo funkcjonującej animacji. W projekcie należało również wykonać poszczególne limity ruchu tak, by nie wystąpiły nieprawidłowości podczas odbywania danego ruchu postaci np.: przenikanie przez siebie różnych brył. Do obsługi plików MoCap niezbędna była instalacja dodatkowej wtyczki tak, aby umożliwić podgląd w programie Blender 2.76.Wczytywanie plików BVH przebiegło prawidło i udało się przeprowadzić analizę ruchu na zaprojektowanym modelu zgodnie z oczekiwaniami.

Literatura

[1] T. Krzeszowski, Śledzenie ruchu postaci ludzkiej w systemie wielokamerowym, Politechnika Śląska, (2013), 26.

[2] T. Pięciak, R. Pawłowski, Wizualizacja ruchu człowieka (Motion Capture), Inżynierowie dla Biologii i Medycyny: kwartalnik wykładowców i studentów inżynierii biomedycznej (2009), nr 5, 22–27

[3] E. Lech, K. Wadas, Ocena animacji szkieletowych postaci wirtualnych, Studia Informatica (2010), nr 1(88), vol. 31, 85-99.

[4] H. Li, D. Simpson, W. Tang, Bahaviour based motion simulation for fire evacuation procedures. Theory and Practice of Computer Graphics, (2004), 112-118.

[5] H. Huang, Ch. Liang, Strategy-based decision making of a soccer robot system using a realtime self-organizing fuzzy decision tree. Fuzzy Sets and Systems, (2002), 49-64.

[6] L. Goncalves, M. Kalman, D. Thalmann, Defining behaviors form autonomous agents based on local perception and smart objects. Computers and Graphics, (2002), t. 26, 887-897.

[7] E. Hernandez, J. Kogler, F. Mirand, M. Netto, An artificial life approach for the animation of cognitive characters. Computer and Graphics, (2001), 955-964.

[8] P. Kopniak, Budowa modeli 3D w edytorze Blender dla silnika gier JMonkeyEngine, [w:] Prace Instytutu Elektrotechniki, Warszawa (2011), z. 249, 81-94.

[9] K. Kuklo, J. Kolmaga, Blender. Kompendium., Helion, Gliwice, 2007.

[10] T. Mullen, Blender. Mistrzowskie Animacje 3D., Helion, Gliwice, 2010.

[11] P. Zradziński, Zasady modelowania zagrożeń elektromagnetycznych. Modelowanie ciała pracownika., Bezpieczeństwo Pracy 10/2006, Centralny Instytut Ochrony Pracy – Państwowy Instytut Badawczy, Warszawa 2006

JCSI 2 (2016) 98-103
Journal of Computer Sciences Institute

Wysłane: 2016-11-15
Przyjęte: 2016-11-18

Analiza porównawcza narzędzi do budowania aplikacji Single Page Application – AngularJS, ReactJS, Ember.js

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

Streszczenie. Tematem artykułu jest analiza porównawcza jednych z aktualnie najpopularniejszych frameworków języka JavaScript, służących do budowania aplikacji typu Single Page Application. Podczas analizy przyjęto cztery kryteria porównawcze: trudność nauki biblioteki, dodatkowe narzędzia charakterystyczne dla danego frameworka, wydajność na urządzeniach stacjonarnych oraz wydajność na urządzeniach mobilnych.

Słowa kluczowe: Angular; React; Ember; Single Page Application

*Autor do korespondencji.

Adres e-mail: radoslaw.nowacki@pollub.edu.pl

Comparative analysis of tools dedicated to building Single Page Applications – AngularJs, ReactJS, Ember.js

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

Abstract. The subject of this article is a comparative analysis of some of the most popular frameworks, of JavaScript programming language, dedicated to creating Single Page Applications. There were four criteria accepted for the purpose of the research. The criteria are: difficulty of learning the given library, additional tools specific for given framework, performance on desktop devices and performance on mobile devices.

Keywords: Angular; React; Ember; Single Page Application

*Corresponding author.

E-mail address: radoslaw.nowacki@pollub.edu.pl

1. Wstęp

Dynamiczny rozwój technologii webowych, takich jak HTML, CSS oraz w szczególności JavaScript wpłyną na ogromne zmiany, zarówno w procesie tworzenia, jak i w samej strukturze stron internetowych. Dodatkowe możliwości jakie oferują te technologie pozwoliły na znaczne poszerzenie zastosowania tych właśnie stron. Jeszcze niedawno służyły one głównie jako wizytówki firm zawierające bardzo dużo tekstu, i kilka obrazków. Obecnie wygląda to zupełnie inaczej. Tradycyjne strony internetowe zawierają obecnie w większości grafikę, a tylko najważniejsze informacje są umieszczone na stronie.

Zmiany wizualne nie są jednak największą zmianą jakie spowodował dynamiczny rozwój tych technologii. Przyczynił się on również do ogromnego wzrostu sposobów wykorzystania stron internetowych [1]. Ewoluowały one do ogromnych serwisów społecznościowych, serwisów informacyjnych i rozrywkowych, ale są także szeroko wykorzystywane do zarządzania całymi strukturami firm. Mogą to być przykładowo systemy klasy CRM (Customer Relationship Service), a nawet klasy HIS (Hospital Information System) służące do obsługi szpitali.

Taka zmiana w zastosowaniu spowodowała, że coraz częściej używana jest nazwa aplikacja internetowa lub aplikacja webowa, zamiast strona internetowa. Nazwa ta dużo lepiej odzwierciedla, że aktualnie takie aplikacje mają szerokie zastosowanie [2].

Jednym z najpopularniejszych rodzajów takich aplikacji są aplikacje typu Single Page Aplication (SPA). Są to serwisy, które po pierwszym wczytaniu nigdy nie odświeżają przeglądarki internetowej użytkownika, a wszystkie dane potrzebne do działania takiej strony są pobierane dynamicznie. Dzięki takiemu podejściu reakcja na akcję użytkownika następuje natychmiastowo, bez zbędnego oczekiwania na przeładowanie strony [1].

Oczywiście podejście to wiąże się z pewnymi trudnościami. Jest ono znacznie trudniejsze w implementacji, wymaga częstej ingerencji JavaScript w strukturę drzewa DOM. Dodatkowo, aby początkowe założenia aplikacji SPA, czyli płynne działanie i natychmiastowe akcje, były spełnione, bardzo ważna jest wydajność takich aplikacji.

Implementacja takich aplikacji jest wymagająca, ale zostaje ona znacznie ułatwiona dzięki zastosowaniu dostosowanych do tego celu frameworków języka JavaScript [3].

W tym artykule opisany został proces analizy trzech bibliotek: AngularJS, React oraz Ember.js. Zostały one

wybrane, ponieważ wszystkie są narzędziami open-source z licencją MIT, oznacza to, że ich kod źródłowy jest dostępny na platformie GitHub dla każdego użytkownika, a biblioteki mogą być wykorzystywane komercyjnie bez żadnych opłat. Kolejnym kryterium wyboru tych frameworków jest ich ogromna popularność oraz sukces prestiżowych aplikacji napisanych za ich pomocą. Ember został wykorzystany w takich projektach jak Playstation Now oraz Apple Music. Za pomocą React napisane zostały portale Airbnb oraz Instagram. Angular natomiast został wykorzystany przy tworzeniu stron: Ryanair oraz Google Cast.

2. Kryteria analizy

2.1. Trudność nauki biblioteki

Pierwszym kryterium jest trudność nauki frameworka. Jest to bardzo ważne kryterium podczas wybierania technologii do projektu. W tym artykule miarą trudności biblioteki będzie tzw. próg wejścia.

Próg wejścia jest określeniem opisującym zakres wiedzy jaki należy przyswoić w celu stworzenia prostej, w pełni działającej aplikacji za pomocą badanej biblioteki. Trudność nauki jest wartością, której nie da się zmierzyć, w związku z tym na potrzeby tej pracy przyjęto, że jej wyznacznikiem będzie liczba zagadnień, które muszą zostać przyswojone w celu stworzenia prostych, ale w pełni funkcjonalnych, aplikacji demonstracyjnych.

2.2. Wydajność na urządzeniach stacjonarnych

Wydajność jest jednym z najważniejszych aspektów aplikacji SPA. Każde opóźnienie negatywnie wpływa na odczucia użytkownika i może potencjalnie wpłynąć na porażkę strony.

Na wydajność składają się dwa elementy, którymi są czas ładowania aplikacji, oraz czas reakcji na akcję użytkownika. Aby zbadać ten aspekt, stworzone aplikacje demonstracyjne zostały zapełnione dużymi zestawami danych, a następnie zostały zmierzone wspomniane wcześniej czasy. Zestawy danych zawierały dane osobowe fikcyjnych użytkowników, zawierające sześć pól: imię, nazwisko, miasto, telefon, pesel oraz email. Przeprowadzono po pięć prób dla zestawów danych składających się kolejno z: 50, 100, 500 oraz 1500 rekordów.

W celu przeprowadzenia analizy dla frameworków Angular oraz React, wykorzystano narzędzie webpack, posiadające wbudowany serwer: webpack-dev-server umożliwiający uruchomienie aplikacji [4]. W przypadku Embera skorzystano z wbudowanego narzędzia, ember-server, dającego podobne możliwości.

2.3. Wydajność na urządzeniach mobilnych

W dzisiejszych czasach aplikacje internetowe są użytkowane równie często na urządzeniach mobilnych co na urządzeniach stacjonarnych. W związku z tym utrzymanie wydajności stało się jeszcze większym wyzwaniem, gdyż bardzo często przenośne urządzenia nie posiadają tak dobrej specyfikacji technicznej jak tradycyjne komputery [5].

Analiza wydajności na tych urządzeniach została przeprowadzona w sposób analogiczny do analizy na urządzeniach stacjonarnych. Aplikacje testowe zostały poddane stresorowi w postaci dużych zestawów danych, następnie zbadano czas ładowania strony oraz czas reakcji na zmiany modelu aplikacji.

Zarówno webpack-dev-server jak i ember-server dają programiście możliwość przekazania parametru –host, którego ustawienie na wartość 0.0.0.0 umożliwia uruchomienie lokalnie uruchomionej aplikacji internetowej na urządzeniu mobilnym połączonym z tą samą siecią.

Aplikacja była uruchamiana na przeglądarce mobilnej Google Chrome, ponieważ daje ona możliwość badania aplikacji internetowej za pomocą Chrome DevTools po podłączeniu narzędzia mobilnego do komputera stacjonarnego lub laptopa. Chrome DevTools zawierają natomiast zakładkę linii czasu, która umożliwia wykonanie potrzebnych pomiarów.

Wszystkie testy zostały przeprowadzone na smartphone HTC One m8, z systemem Android w wersji 6.0.

2.4. Dodatkowe narzędzia

Każdy z frameworków oferuje pewien zestaw narzędzi ułatwiających lub przyspieszających proces tworzenia aplikacji z ich wykorzystaniem. Mogą być to zarówno narzędzia deweloperskie, ułatwiające pisanie i analizowanie kodu oraz wykrywanie błędów, ale również zestawy gotowych rozwiązań, a nawet platformy umożliwiające tworzenie natywnych aplikacji mobilnych używając danego frameworka języka JavaScript.

W celu analizy frameworków pod tym kątem, zostały wybrane po dwa najważniejsze narzędzia wpierające pracę programistów. Następnie wartość jaką one wnoszą została oceniona w celu wyłonienia najlepszego pod tym względem frameworka.

3. Przedstawienie aplikacji demonstracyjnych

Wydajność jest jednym z najważniejszych aspektów aplikacji SPA. Każde opóźnienie negatywnie wpływa na odczucia użytkownika i może potencjalnie spowodować porażkę komercyjną strony. Wykonanie analizy wymagało stworzenia wersji demonstracyjnych dla każdego z badanych frameworków. Aby wyniki analizy były wiarygodne, wszystkie z aplikacji posiadają identyczne funkcjonalności. Ponadto, przedmiotem badań są biblioteki JavaScript, ważne było więc również aby kod CSS i HTML był jak najbardziej zbliżony w każdej z wersji demonstracyjnych.

Jako aplikacje demonstracyjne, stworzone zostały strony typu CRUD (Create Read Update Delete), posiadające możliwość tworzenia, odczytywania, aktualizowania

i usuwania rekordów. Rekordami były dane osobowe fikcyjnych użytkowników.

Dane te zostały wyświetlone w tabeli. W jej drugim wierszu osadzony został formularz dodawania nowego użytkownika. Od trzeciego wiersza zaczyna się lista fikcyjnych użytkowników. Dodatkowo aplikacje musiały umożliwiać modyfikowanie użytkowników, jednakże wymaganiem było, aby zmiany były aktualizowane na bieżąco, bez konieczności akceptacji, ponieważ ciągłe aktualizowanie treści strony jest charakterystyczne dla aplikacji SPA. Ponadto, aby sprawdzić jakie możliwości daje framework, formularz edycji każdego z wierszy jest początkowo ukryty, dopiero przejście w tryb edycji danego wiersza sprawia, że ten formularz jest widoczny dla użytkownika. Ukrywanie treści również jest bardzo ważnym elementem aplikacji tego typu.

Ostatnią funkcjonalnością była wyszukiwarka użytkowników. Wymaganiem było filtrowanie listy po każdej zmianie w polu wyszukiwania, dodatkowo wyszukiwarka powinna działać bez względu na wielkość wpisanych liter.

4. Analiza porównawcza

4.1. Trudność nauki bilbioteki

Ocena progu wejścia dla każdego z frameworków została wykonana na podstawie procesu tworzenia aplikacji demonstracyjnej. Głównym wyznacznikiem była obszerność materiałów, które należało przyswoić w celu stworzenia aplikacji. Ponadto negatywnie wpływały na nią nieoczekiwane problemy napotkane podczas tworzenia danej aplikacji.

Początki z AngularJS mogą wydawać się trudne, framework ten zawiera bowiem bardzo wiele terminów dla niego specyficznych takich jak digest cycle, scope lub watchers, jednakże jest to wrażenie mylne, ponieważ ich zrozumienie jest konieczne dopiero w późniejszych etapach nauki frameworka. Początki okazały się bardzo łatwe, doskonałą pomocą była dokumentacja techniczna. Odtwarzając kroki z dokumentacji, należało stworzyć moduł, a następnie zdefiniować w nim komponenty [6].

W przypadku aplikacji demonstracyjnej były to dwa komponenty – lista użytkowników oraz użytkownik. Następnie do każdego z nich należało zadeklarować szablon HTML, co również okazało się bardzo proste. Listę udało się wyświetlić za pomocą dyrektywy ng-repeat doskonale opisanej w dokumentacji technicznej. Podczas wyświetlania listy za pomocą komponentu użytkownika pojawił się jednak problem. Wykorzystanie komponentu użytkownika jako wiersz tabeli zaburzyło strukturę tabeli HTML, ponieważ wewnątrz znacznika *tbody* dopuszczalne są tylko kolejne wiersze *tr*. Aby rozwiązać ten problem, do wyświetlenia użytkownika wykorzystano dyrektywę atrybutową. Następnie należało stworzyć kontrolery dla danych komponentów, odpowiedzialne za sterowanie zachowaniem aplikacji. Kontrolery są jednak zwykłymi klasami, w podstawowym przypadku ich stworzenie nie wymaga więc żadnej wiedzy na temat frameworka. Ostatnim krokiem było dodanie filtrowania listy. Efekt ten osiągnięto za pomocą dyrektywy ng-model oraz zastosowania filtra dyrektywy ng-repeat.

W przypadku AngularJS stworzenie aplikacji wymagało więc podstawowej wiedzy na temat komponentów i wbudowanych dyrektyw. Dodatkowo z racji wspomnianego problemu również wiedza na temat tworzenia własnych dyrektyw była wymagana.

React okazał się na początku dużo trudniejszy. Sam framework nie zapewnia dobrej obsługi modelu aplikacji, dlatego do stworzenia aplikacji demonstracyjnej wykorzystano bibliotekę Redux. Jest to implementacja narzędzia do zarządzania modelem zgodnie z architekturą Flux. Użycie tego narzędzia wymaga więc znajomości tej architektury, ale wymaga również umiejętności budowania czystych funkcji, czyli takich, które nie posiadają efektów ubocznych i nie mutują stanu aplikacji. Opanowanie tych podstaw okazało się dużo trudniejsze niż w przypadku AngularJS. Stworzenie aplikacji wymagało znajomości komponentów frameworka React, ale również umiejętności tworzenia funkcji redukujących, akcji, kontenerów oraz zarządzaniem obiektem store. Dodatkowym utrudnieniem może być początkowo przyzwyczajenie do JSX, które jest rozszerzeniem do języka JavaScript, dodającym do niego składnię przypominającą XML. JSX umożliwia pisanie kodu w pliku JavaScript, podobnego do kodu HTML, który jest następnie kompilowany do odpowiednich funkcji – w przypadku React jest to funkcja *createElement*.

W przypadku Ember, wydawać by się mogło, że dzięki zastosowaniu Ember-CLI stworzenie aplikacji demonstracyjnej będzie bardzo proste, jednakże próg wejścia jest dość wysoki. O ile tworzenie elementów takich jak komponenty czy szablony sprowadza się do wpisania jednej linijki w terminalu, to zrozumienie jak one ze sobą współgrają jest dużo trudniejsze [7]. Do stworzenia badanej aplikacji wymagana była znajomość narzędzia Mirage, Ember-CLI, komponentów oraz wiedzy na temat działania obiektu store, odpowiedzialnego za operacje na rekordach. Dodatkową trudnością jest konieczność znajomości handlebars, czyli narzędzia do tworzenia szablonów. Jest ono podobne do klasycznego HTML, jednakże mylące może być przykładowo definiowanie elementów takich jak input wewnątrz podwójnych nawiasów klamrowych zamiast tradycyjnego znacznika.

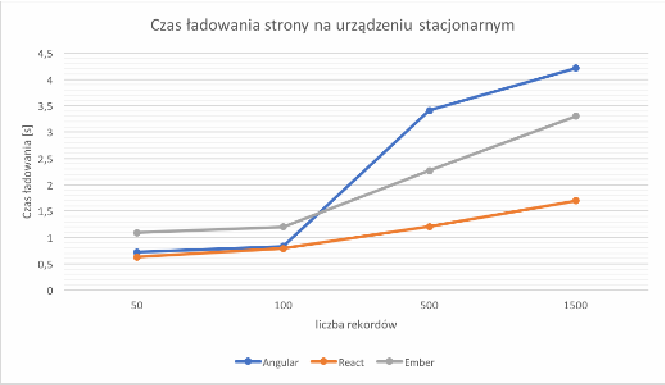
Angular jest więc zdecydowanie frameworkiem o najniższym progu wejścia. Nieco trudniejszy jest Ember, gdyż wymaga znajomości większej liczby elementów charakterystycznych dla niego, ale jednocześnie umożliwia ich łatwiejszą implementację za pomocą Ember-CLI. Najtrudniejszy okazał się React, ponieważ poza znajomością samej biblioteki, należało zapoznać się z Reduxem. Wymagał on również umiejętności pisania funkcji niemutujących stanu aplikacji.

4.2. Wydajność na urządzeniach stacjonarnych

W celu zbadania wydajności na urządzeniach mobilnych, aplikacje zostały zasilone fikcyjnymi zestawami danych.

Następnie aplikacje te zostały uruchomione za pomocą webpack-dev-server lub ember-server. Do mierzenia czasu reakcji aplikacji użyte zostały narzędzia developerskie przeglądarki Google Chrome.

Pierwszym badanym elementem był czas ładowania strony w zależności od liczby rekordów wyświetlanych jednocześnie na stronie. Wyniki przedstawione zostały na rysunku 1. Na osi X znajduje się liczba rekordów na stronie, natomiast na osi Y – badana wartość, czyli czas ładowania strony.

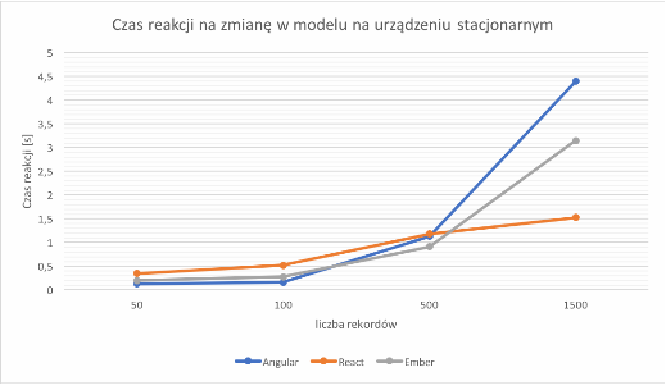


Rys. 1. Czas ładowania strony na urządzeniu stacjonarnym.

Najszybszym frameworkiem pod tym względem okazał się React. Różnica przy 50 i 100 wierszach tabeli jest znikoma, jednakże przy próbach przeprowadzonych dla 500 i 1500 elementów łatwo można zauważyć dużą różnicę w wydajności tej biblioteki pod względem prędkości ładowania strony.

Przy niewielkim zestawie danych Ember poradził sobie najgorzej, jednakże jest on dużo bardziej wydajny niż AngularJS przy większym zestawie danych. Dla 1500 rekordów był on o sekundę szybszy.

Podobnie wyglądała sytuacja w przypadku czasu reakcji na zmianę w modelu aplikacji. Charakterystyczne dla aplikacji typu SPA jest ciągłe aktualizowanie treści na stronie po wykryciu, dlatego wydajność pod tym względem jest tak istotna. Wyniki analizy przedstawiono na rysunku 2.

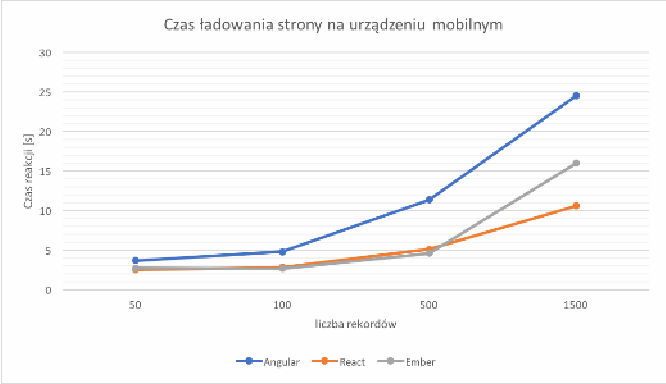


Rys. 2. Czas reakcji na zmianę w modelu na urządzeniu stacjonarnym.

Dla małego zestawu danych najszybszy okazał się Angular, jednakże jego skalowalność pozostawia dużo do życzenia. Już przy 500 elementach na stronie Ember znacznie szybszy od pozostałych. Jednakże podobnie jak w przypadku czasu ładowania strony, różnice w wydajności najłatwiej jest ocenić przy bardzo dużym zestawie danych. Można więc zauważyć, że React był znacznie szybszy od rywali, gdyż przy 1500 rekordach na stronie, zareagował na zmianę w czasie o ponad połowę krótszym od Ember, a niemalże trzy razy krótszym niż w przypadku Angular.

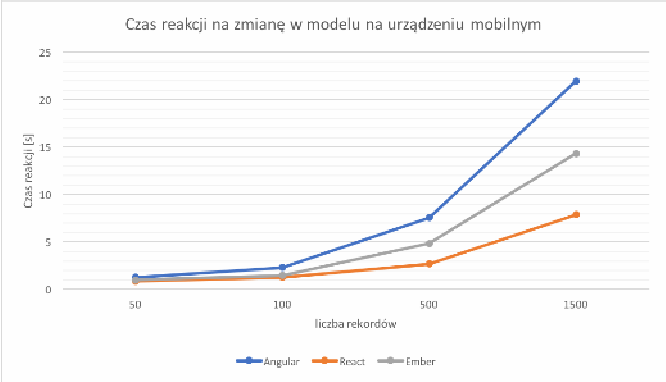
4.3. Wydajność na urządzeniach mobilnych

W celu zbadania wydajności na urządzeniach mobilnych, wykonano próby analogiczne do tych, wykonanych na urządzeniu stacjonarnym. Wyniki pomiarów czasu ładowania strony zostały przestawione na rysunku 3, natomiast reakcji na zmianę w modelu na rysunku 4.



Rys. 3. Czas ładowania strony na urządzeniu mobilnym.

Wyniki analizy na urządzeniu mobilnym wyglądają podobnie jak w przypadku urządzenia stacjonarnego. Pomimo tego, że Ember jest minimalnie szybszy przy małych zestawach danych, to po ostatnim zestawie danych łatwo możemy zaobserwować, że React jest dużo lepiej skalowalny. Angular ponownie został zdeklasowany przez rywali. Jego czas już przy 500 użytkownikach na liście był zdecydowanie zbyt duży, aby strona była oddana do użytku.



Rys. 4. Czas reakcji na zmianę w modelu na urządzeniu mobilnym.

Warto również zauważyć, że czasy ładowania wszystkich stron są znacznie większe niż w przypadku urządzenia stacjonarnego. Właśnie dlatego, pisząc aplikację dostosowaną do urządzeń mobilnych tak ważne jest badanie jej wydajności.

W przeciwieństwie do wyników analizy czasu ładowania strony, wyniki pomiarów czasu reakcji na zmianę w modelu są jednoznaczne. Już w przypadku najmniejszego zestawu danych React uzyskał najlepszy czas. Angular natomiast ponownie okazał się najwolniejszy, co widać w szczególności przy ostatnim zestawie danych.

4.4. Wnioski z analizy wydajności

Z powyższych badań można wywnioskować, że wszystkie frameworki doskonale radzą sobie z niewielkimi oraz średnimi zestawami danych. Różnice między nimi są dla takich zestawów znikome, a wydajność jest wystarczająca do zapewnienia płynnego działania aplikacji. Wraz ze wzrostem liczby wyświetlanych elementów, różnice między badanymi bibliotekami stają się bardziej wyraziste. Najbardziej wydajny dla dużych zestawów danych, czyli tym samym najlepiej skalowalny jest React. Dzieje się tak ponieważ został w nim zaimplementowany mechanizm Virtual DOM, działający tak, że po wykryciu zmian wymagających modyfikacji drzewa DOM, React tworzy coś w rodzaju wirtualnej kopii tego drzewa, a następnie bardzo wydajne mechanizmy znajdują najlepszą akcję pozwalającą osiągnąć pożądaną strukturę strony [8]. Przykładowo zamiast usuwać elementu z środka listy, zostanie usunięty ostatni wiersz, a zaktualizowana zostanie tylko treść pozostałych tak aby zaoszczędzić czas potrzebny na wykonanie kosztownych manipulacji.

Nieco wolniejszy okazał się Ember, jednakże jego wyniki również były stosunkowo zadowalające. Pomimo tego, że przy 1500 elementach aplikacja stawała się dość trudna w użytku a czas oczekiwania na zmiany przekroczył 3s, co jest niedopuszczalne, to przy nieco mniejszym zestawie danych radził on sobie podobnie, a nawet nieco lepiej od React.

Pomimo bardzo dobrych wyników dla bardzo małych zestawów danych, najgorzej wypadł Angular. Aplikacje posiadające 50 elementów nie są zbyt częstym zjawiskiem w realnym świecie, dlatego największe znaczenie miały późniejsze próby. Angular nie jest zbyt dobrze skalowalny i wymaga uważnego projektowania komponentów. Dzieje się tak ponieważ wykrywanie zmian w tej bibliotece jest zaimplementowane jako tzw. dirty checking. Oznacza to, że jeśli jakakolwiek z wartości obserwowanych, czyli tzw. watcher, zmieni się, Angular rozpocznie cykl digest, który sprawdzi wszystkie pozostałe watchery i jeśli to konieczne, zaktualizuje je. Jest to mechanizm bardzo niewydajny, i powoduje, że już przy 2000 takich obserwatorów strona może działać mniej płynnie [9. 10].

4.5. Dodatkowe narzędzia

W celu analizy frameworków pod tym kątem, z każdego z nich wybrano po dwa najważniejsze narzędzia ułatwiające pracę z daną biblioteką.

W przypadku Angular, wybrane narzędzia to ui-bootstrap oraz AngularJS Batarang. Pierwsze z nich to zestaw dyrektyw, napisany z zastosowaniem bardzo popularnego frameworka CSS – Bootstrap. Jest to zestaw uniwersalnych rozwiązań, które mogą być w bardzo łatwy sposób zaimplementowane w danym projekcie. Dodatkowo są one łatwo modyfikowalne, dlatego można je dostosować na potrzeby większości projektów. Jest to świetne narzędzie, gdyż pozwala zaoszczędzić bardzo dużo czasu na implementację często powtarzających się w projektach funkcjonalności, jak np. Accordion.

AngularJS Batarang to wtyczka do przeglądarki Google Chrome, pozwalająca na głębszą inspekcję aplikacji. Po zainstalowaniu wtyczki, pojawia się dodatkowa zakładka w narzędziach deweloperskich, zawierająca informacje charakterystyczne dla Angular, czyli przykładowo strukturę drzewa $scope, ich zawartość oraz liczbę aktywnych funkcji obserwujących.

W przypadku React wybranymi narzędziami są React Native oraz Redux devtools. React Native zasługuje na miano oddzielnego frameworka, jednakże jego ideą jest pozwolenie programiście na tworzenie natywnych aplikacji mobilnych za pomocą React. Różni się on od obecnych do tej pory na rynku narzędzi, ponieważ powstała aplikacja ma strukturę identyczną jak w przypadku standardowych aplikacji iOS i Android, przez co jest bardzo wydajna.

Drugim narzędziem są Redux DevTools. Podobnie jak AngularJS Batarang jest to wtyczka do Google Chrome, udostępniająca dodatkową zakładkę w narzędziach deweloperskich. Narzędzie daje to w połączeniu z aplikacją napisaną w Redux daje ogromne możliwości. Wyświetlane są tam wszystkie wykonane w aplikacji akcje użytkownika oraz ich wpływ na stan aplikacji. Dzięki temu, że w Redux stan ten jest przechowywany w jednym obiekcie store, Redux DevTools umożliwia dowolne cofanie, przyspieszanie, odtwarzanie wykonywanych akcji. Znacznie ułatwia to proces tworzenia aplikacji i przyspiesza odnajdywanie błędów.

Wybranymi narzędziami dla Ember są Ember-CLI oraz Mirage. Pierwsze z nich służy do wspomagania tworzenia aplikacji poprzez udostępnienie programiście dodatkowych komend w konsoli lub terminalu. Pozwala ono na stworzenie bazowej struktury projektu oraz dodanie do aplikacji każdego dostępnego w Ember elementu, czyli m.in. komponentów, ścieżek i szablonów.

Drugim narzędziem jest Mirage. Umożliwia ono definiowanie w łatwy sposób sztucznych odpowiedzi serwera. Jest to bardzo duże ułatwienie podczas tworzenia aplikacji typu front-end, kiedy strona serwerowa nie została jeszcze zaimplementowana. Wbrew pozorom jest to sytuacja bardzo częsta i bywa bardzo uciążliwa. Mirage umożliwia zwracanie różnych kodów odpowiedzi, danych i bardzo wspomaga pracę programisty.

Wyniki analizy narzędzi nie są jednoznaczne. Wszystkie badane narzędzia okazały się bardzo pomocne. Mimo to, za najlepszy można uznać React, ponieważ React Native jest technologią pionierską, oferującą tworzenie aplikacji natywnych w JavaScipt, bez utraty wydajności, jak było to w przypadku rozwiązań hybrydowych takich jak np. Cordova. Również Redux DevTools okazało się imponujące, możliwość dowolnego cofania i przywracania akcji użytkownika to coś,

z czym ciężko spotkać się w przypadku innych technologii. Walkę o drugie miejsce między Angular i Ember można uznać za remis, gdyż wszystkie narzędzia oferowane przez te biblioteki są na bardzo wysokim poziomie.

5. Wnioski

W celu łatwiejszej analizy otrzymanych wyników, stworzona została skala punktowa. Najlepsza biblioteka w danej kategorii otrzymała 2 punkty, druga w kolejności biblioteka 1 punkt, a najgorsza 0 punktów. Wyniki zostały przedstawione w tabeli 1.

Tabela 1. Zestawienie wyników analizy

	Angular	**React**	**Ember**
Stopień trudności przyswajania wiedzy	2	0	1
Wydajność aplikacji	0	2	1
Wydajność aplikacji na urządzeniach mobilnych	0	2	1
Dodatkowe narzędzia	1	2	1
Suma	3	6	4

Jak widać najlepszym z frameworków według niniejszej analizy okazał się React. Jest to zasługa głównie świetnej wydajności, która była głównym elementem tej analizy. Dodatkowym atutem są udostępniane narzędzia.

Nieco gorszy okazał się Ember. We wszystkich kategoriach zajął on drugie miejsce. Jego wydajność jest nieco słabsza od React, jednak znacznie lepsza od Angular. Ponadto początki w tym frameworku są znacznie łatwiejsze od React. Oferuje on również przydatne narzędzia deweloperskie.

Najgorszym według niniejszej analizy okazał się Angular. Przyczyną tak rozczarowującego wyniku okazała się słaba wydajność. Angular ma jednak swoje bardzo silne strony. Przede wszystkim jest on bardzo przyjazny początkującym programistom, Znacznie łatwiej zacząć pracę z nim pracę niż z pozostałymi bibliotekami. Dodatkowo oferuje on równie dobre narzędzia deweloperskie co Ember.

Wyniki analizy są jednoznaczne, jednak warto podkreślić, że cztery kryteria nie są wystarczające, aby jednoznacznie określić który z frameworków jest najlepszy. Mają jedynie podkreślić silne i słabe cechy badanych frameworków w porównaniu do ich konkurencji. React jest więc bardzo wydajny i oferuje najlepsze narzędzia, kosztem jest jednak duży próg wejścia. Ember oferuje umiarkowaną wydajność i niezłe narzędzia, ale również wymaga sporej wiedzy nawet przy tworzeniu małych aplikacji. Pierwsze kroki są najłatwiejsze z Angular, oferuje on także dobre dodatkowe narzędzia, kosztem jest jednak wydajność, dlatego dobrze sprawdza się on podczas do budowania serwisów, które wyświetlają jednocześnie niewielką ilość danych wymagających ciągłej aktualizacji.

Literatura

[1] Beda B.: Single Page Web Applications Security, Bucharest University of Economic Studies, ISSN: 2067-4074 (Print), 2015.

[2] Minović M., Vesic S.: Single Page Applications: Trend or Future, Info, 2015.

[3] Enache M. C.: Web Application Frameworks, Dunarea de Jos University of Galati ISSN: 1584-0409, 2015.

[4] Vepsäläinen J.: Webpack and React from apperentice to master, Survivejs, 2015.

[5] Weyl E.: HTML5 : strony mobilne, Wydawnictwo Helion, 2014.

[6] Darwin B. P., Kozłowski P.: Mastering Web Applications Development with AngularJs, Packt Publishing, 2013.

[7] Brady T., Cravens J.: Ember.js dla webdeveloperów, Wydawnictwo Helion, 2015.

[8] Fedosejev A.: React.js Essentials, Packt Publishing, 2015.

[9] Freeman A.: AngularJS : profesjonalne techniki, Wydawnictwo Helion, 2015.

[10] Green B., Seshadri S.: AngularJS, Wydawnictwo Helion, 2014.

JCSI 2 (2016) 104-107
Journal of Computer Sciences Institute

Wysłane: 2016-11-14
Przyjęte: 2016-11-21

Optymalna alokacja zasobów dźwiękowych urządzenia mobilnego na potrzeby gier

Arkadiusz Wrzos*, Jakub Smołka
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Urządzenia mobilne dysponują ograniczonymi zasobami. Szczególnie gry potrzebują specjalnego podejścia od programisty. Większość wzorców projektowych pozwala osiągnąć statyczne zużycie pamięci i niskie użycie procesora, jednak ciągle jest wiele sytuacji gdzie wytworzenie płynnie działającej funkcjonalności jest wymagającym zadaniem, wymagającym przeprowadzenia badań. W tej pracy autor bada optymalną alokację zasobów dźwiękowych do odtwarzania ich z minimalnym opóźnieniem w języku Java na systemie Android.

Słowa kluczowe: android; audio; java

*Autor do korespondencji.

Adres e-mails: arkadiusz.wrzos@pollub.edu.pl

Optimal allocation of mobile device sound resources for game programming purposes

Arkadiusz Wrzos*, Jakub Smołka
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Android is a mobile device platform, whose resources are limited. Games, and high-performance applications need special approach. Most of programming patterns allow for static memory allocation or maintaining low processor usage. Regardless, there are still many situations, in which development is demanding task, and needs much research. In this work author studied and research for efficient allocation of audio resources to play them with minimal latency in non deterministic way.

Keywords: android; audio; java

*Corresponding author.

E-mail address/addresses: arkadiusz.wrzos@pollub.edu.pl

1. Wstęp

Aktualnie dostęp do urządzeń mobilnych wysokiej wydajności jest powszechny. Prace nad architekturą ARM i układami dla smartfonów i tabletów pozwalają na dostarczanie coraz wydajniejszych urządzeń, które już teraz są w stanie realizować większość zadań typowych dla komputerów osobistych, takich jak komunikacja, multimedia, rozrywka itp.. Popularność zyskują interfejsy służące do przyłączania ich do telewizorów oraz monitorów, dzięki czemu już teraz z powodzeniem można ich używać zamiast komputera stacjonarnego w codziennych zadaniach, takich jak przeglądanie internetu, obsługa poczty czy kina domowego [1]. Statystycznie mieszkaniec Polski spędza dziennie 6,3 godzin przy komputerze, z czego częściej używa urządzeń przenośnych niż komputerów osobistych [2]. Pozwalają na to coraz wydajniejsze ogniwa elektryczne oraz procesory heterogeniczne (np. big.LITTLE [3]) z coraz mniejszym zapotrzebowaniem na prąd. Intensywny rozwój tej branży komputerowej stawia wiele nowych wymagań przed programistami. Mimo wysokiej wydajności, trzeba mieć świadomość, że zasoby ciągle są ograniczone. Zmusza to twórców oprogramowania do odpowiedniego balansowania pomiędzy maksymalnym obciążeniem procesora a minimalnym zużyciem baterii. W niektórych dziedzinach, takich jak przetwarzanie dźwięku, system android jest ciągle intensywnie rozwijany, i wymaga specjalnych podejść by osiągnąć zadowalające rozwiązania.

W pracy wykonano badania optymalnego wykorzystania wybranych narzędzi dostarczanych przez system Android, dla osiągnięcia wydajnej prezentacji multimediów. Wykorzystano w tym celu autorską aplikację rozrywkową, w której zadaniem osadzonego w kosmicznej scenerii gracza jest unikanie pocisków przeciwnika, oraz stukania w część ekranu z przeciwnikami zgodnie z rytmika podkładu muzycznego.

Rys. 1. Zrzuty ekranu z badanej gry

W powyższej aplikacji użytkownik kieruje bohaterem przesuwając go ku przeciwległym brzegom ekranu, oraz stuka w niebieskich przeciwników, którzy wylatują z przeciwnej strony.

2. Badania alokacji zasobów

Dosyć podobnym problemem zajmują się producenci narzędzi dla sekwencerów i aplikacji muzycznych SuperPowered, jednak ich rozwiązania, podobnie jak Android NDK są niskopoziomowe [4], wymagają dużego nakładu pracy przy implementacji.

Na konferencji Google I/O 2016 przedstawiono narzędzia Android Professional Audio znacznie obniżające opóźnienia dźwiękowe na wybranych urządzeniach z systemem Android [5]. Teoretycznie umożliwi to osiągnięcie opóźnień sprzętowych w odtwarzaniu dźwięku na poziomie 3,7ms co jest bardzo dobrym wynikiem, biorąc pod uwagę, że komputerowe karty dźwiękowe sięgają 2ms. Technologia jest w trakcie intensywnego rozwoju i najpewniej w kolejnych wersjach systemu stanie się standardem.

3. Badanie alokacji zasobów dźwiękowych

Gry towarzyszą komputerom od bardzo dawna, dzięki czemu programiści zdążyli już opisać bazę optymalnych podejść do najpopularniejszych problemów [6]. W aplikacji przygotowanej na potrzeby artykułu wykorzystano wiele spośród tych rozwiązań [7]. Zgodnie z nimi Aktualizacja stanu gry realizowana jest w pętli gry, która manipulując czasem potrzebnym na wykonanie cyklu reguluje liczbę klatek tak, by osiągnąć zadowalające efekty. Detekcja kolizji jest wykonana przez analizę części wspólnych prostokątnych instancji obiektów ruchomych. Animacje wykorzystują atlasy tekstur i przesuwają się pośród klatek dla aktualizowania wyświetlanej klatki. Te i inne problemy są doskonale znane programistom gier. Problem, który wymagał specjalnej analizy w tym przypadku polegał na potrzebie uzyskania ciągłej linii melodycznej złożonej z krótkich wycinków, które odgrywane byłyby w różny sposób w zależności od powodzenia gracza. Dla przykładu: odtwarzanie zapętlonej linii perkusyjnej miałoby być zastąpione odtwarzaniem zapętlonymi liniami perkusyjną i gitarową. Analiza wystąpienia dotknięcia ekranu w odpowiednim momencie powinna zapisywać sukces, który mechanizm odtwarzania muzyki powinien zinterpretować i przedstawić odpowiednie dźwięki w tle w czasie możliwie szybkim, najlepiej niezauważalnym. Poszukiwania wzorców, metod i wskazówek nie przyniosły skutku, prawdopodobnie ze względu na bardzo specyficzne zastosowanie mechanizmów odtwarzania dźwięku.

3.1. Plan badań

Wykonano serię badań porównującą wydajność podstawowych klas systemowych dostarczanych z Android SDK pozwalających na odtwarzanie dźwięku. Każda z nich została zaimplementowana i następowało dwudziestosekundowe badanie sygnału wyjściowego audio, polegające na stawianiu markerów na początku i końcu każdej próbki i mierzeniu przerwy między nimi. Badano klasy SoundPool i MediaPlayer. Odtwarzanie dźwięków było wykonywane w cyklicznych odstępach czasu, a ich zmiany również były cykliczne. Program wykonywał zadanie w osobnym wątku, i nie posiadając spowalniającego sprzężenia z wątkiem głównym mógł być oddelegowany do osobnego rdzenia procesora [8]. Dla porównania wykonano sprawdzenie odtworzenia pojedynczej próbki w zapętleniu. Badanie przeprowadzono na urządzeniach OnePlus One i Sony E4g. Próby miały sprawdzić, czy istnieje możliwość wysokopoziomowego wykorzystania klas SoundPool i MediaPlayer do naprzemiennego odtwarzania czterech próbek dźwiękowych z niską latencją.

Badania przewidywały sześć scenariuszy, w których każdy był wykonywany w sześciu krokach:

1) Wygenerowaniu dźwięku na podstawie przebiegu sinusoidalnego
2) Implementacji wyzwalania kolejnych próbek dźwiękowych w wątku powtarzającym odtwarzanie w odstępie czasu równym długości próbki.
3) Instalacji i uruchomieniu programu na urządzeniu testowym
4) Przechwyceniu sygnału wyjściowego audio z urządzenia na którym uruchomiony był program
5) Oznaczenie markerami czasów początków i końców odtwarzania próbek
6) Wprowadzenie danych do arkusza kalkulacyjnego i pomiar długości opóźnień i ich sumy i średniej oraz liczby wystąpień próbek i liczby straconych próbek.

Zbadano następujące mechanizmy:

1) Użycie klasy SoundPool do odtwarzania próbek w cyklu czasowym
2) Użycie klasy SoundPool do odtwarzania próbek w cyklu czasowym, z optymalizacją. Optymalizacja polegała na ustawieniu odpowiednich flag w sterowniku urządzenia, informujących o priorytecie i sposobie wykorzystania sterownika. Dodatkowo zoptymalizowano próbki dźwiękowe tak, by miały zgodną z urządzeniem częstotliwość, tak, by nie występowała konieczność resamplingu strumienia do częstotliwości stosowanej wewnętrznie przez system i układ dźwiękowy.
3) Użycie klasy SoundPool do odtworzenia pojedynczej próbki w zapętleniu. Udostępniona jest jedynie metoda odtwarzania pojedynczej próbki w zapętleniu, nie można zmieniać ich w czasie trwania.
4) Użycie wielu instancji klasy SoundPool. Każda instancja była przeznaczona do odtwarzania pojedynczej próbki w zapętleniu. Wszystkie obiekty zostały wstępnie zatrzymane i uruchamiane naprzemiennie, tak żeby uniknąć wewnętrznego ładowania bufora dla każdej z nich występującego przy podmianie pliku-jak w punktach 1. i 2.
5) Użycie klasy MediaPlayer do odtwarzania próbek w cyklu czasowym.
6) Użycie klasy MediaPlayer do odtworzenia połączonych w jeden plik próbek, a następnie wykorzystywanie mechanizmu przesunięcia znacznika odtwarzania do czasu odpowiadającego kolejnym próbkom.

Wyniki badań przedstawiono w tabelach 1-6.

Tabela 1. Wyniki próby SoundPool - próbki wyzwalane w cyklu czasowym

Próba	1
Pożądana liczba próbek	66,667
Wystąpiło próbek	56,737
Straconych	9,93
Suma opóźnień w ms	2979
Średnie opóźnienie w ms	52,26
Długość próbki w ms	300
Długość próby w ms	20000

Tabela 2. Wyniki próby SoundPool z optymalizacją- próbki wyzwalane w cyklu czasowym

Próba	2
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	65,81666667
Straconych	0,85
Suma latencji w ms	255
Średnia latencja w ms	18,21428571
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 3. Wyniki próby SoundPool – pojedyncza próbka wyzwalana w zapętleniu

Próba	3
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	66,66666667
Straconych	0
Suma latencji w ms	0
Średnia latencja w ms	0
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 4. Wyniki próby SoundPool – kontrolowane przełączanie zapętlonych strumieni

Próba	4
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	66,65
Straconych	0,01666666667
Suma latencji w ms	5
Średnia latencja w ms	0,07575757576
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 5. Wyniki próby MediaPlayer – próbka wyzwalana w zapętleniu

Próba	5
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	63,31333333
Straconych	3,353333333
Suma latencji w ms	1006
Średnia latencja w ms	20,95833333
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 6. Wyniki próby MediaPlayer – przesuwanie znacznika odtwarzania

Próba	6
Pożądana liczba sampli	40
Wystąpiło sampli	38,696
Straconych	1,304
Suma latencji w ms	652
Średnia latencja w ms	40,75
Długość sampla w ms	500
Długość próby w ms	20000

4. Wnioski

W pracy przedstawiono zestawienia opóźnień i przerw w odgrywaniu dźwięku występujących przy dynamicznym zmienianiu jego źródła/pozycji. Do pomiarów wykorzystano różne implementacje klas systemowych.

Najgorsze wyniki napotkano przy klasie MediaPlayer, która mimo iż zgodnie z dokumentacją nie służy do odtwarzania dźwięku z niskimi opóźnieniami [9], dostarcza możliwość przesuwania markera aktualnie odtwarzanego dźwięku. Umożliwia to zmienianie dźwięków przetrzymując wszystkie w jednym pliku. Rozwiązanie okazało się bardzo niewydajne i powodowało stosunkowo duże opóźnienia zarówno przy wyzwalaniu próbki w zapętleniu (Tabela 5.), jak i podczas przesuwania markera znacznika czasowego (Tabela 6.).

Lepsze wyniki osiągnięto korzystając z klasy SoundPool, gdzie opóźnienia były niskie (Tabela 1.), a po dodatkowej konfiguracji dopasowującej parametry dźwięku do parametrów sprzętowych opóźnienia zostały praktycznie wyeliminowane (Tabela 2.). Odstępy między dźwiękami były znikome, a czasem nie występowały wcale. Kluczowym jest przygotowanie próbek spójnych z parametrami urządzenia. Interesujące jest też to, że odtwarzanie pojedynczej próbki nie tworzy opóźnień (Tabela 3.).

Metoda realizująca przełączanie zatrzymanych strumieni okazała się nieprzydatna, ponieważ nie dawało możliwości synchronizacji strumieni ze sobą. Każde opóźnienie powodowało utratę synchronizacji, a suma opóźnień spowodowała, że na koniec próby dźwięk był nie do zaakceptowania (Tabela 4.) .

Najlepszym i skutecznym sposobem było się podejście drugie, czyli „SoundPool z optymalizacją - próbki wyzwalane w cyklu czasowym„. Dostarczało ono najbardziej stabilne opóźnienia na dosyć niskim poziomie. Po wykonaniu prób zamieniono dźwięki wygenerowane na podstawie przebiegu sinusoidalnego na fragmenty linii melodycznej i według empirycznej oceny to rozwiązanie dostarczyło zadowalające wrażenia słuchowe.

Należy wziąć pod uwagę dużą fragmentację rynku urządzeń z systemem Android i pamiętać, że różnice w wydajności urządzeń poszczególnej klasy mogą dawać niejednakowe wyniki przy powyższych metodach, dlatego okazuje się, że są one wydajne na nowszych modelach. Testy odbywały się na wydajnych urządzeniach z systemem w wersji 5 (Lollipop) i wielordzeniowymi procesorami.

Ze względu na niezwykle dynamiczny rozwój systemu android i narzędzi z nim związanych większość wiedzy opiera się o internetową dokumentację techniczną, oraz nieliczne pozycje w literaturze.

Nie znaleziono publikacji naukowych powiązanych z pracą.

Literatura

[1] https://liliputing.com/2016/07/andromiums-99-superbook-turns-android-phone-laptop-crowdfunding.html Andromium's $99 Superbook turns your Android phone into a laptop (crowdfunding) [04.11.2016]

[2] http://qz.com/214307/mary-meeker-2014-internet-trends-report-all-the-slides/ - Mary Meeker's 2014 internet trends report: all the slides plus highlights [04.11.2016]

[3] https://www.arm.com/products/processors/technologies/biglittleprocessing.php - Presenting the Next Generation of Mobile Processing [04.11.2016]

[4] http://superpowered.com/superpowered-audio-sdk-for-ios-and-android About Superpowered Audio SDK for iOS, OSX and Android [04.11.2016]

[5] https://www.codechannels.com/video/Google/android/android-high-performance-audio-google-io-2016/ Android high-performance audio – Google I/O 2016 [04.11.2016]

[6] Ernest Adams: Projektowanie gier. Podstawy.Helion, Gliwice 2011

[7] Robert Nystom: Game Programming Patterns ISBN: 978-0-9905829-2-2

[8] Anders Göransson: Android. Aplikacje wielowątkowe. Techniki przetwarzania. Helion 2015

[9] https://developer.android.com/reference/android/media/MediaPlayer.html - MediaPlayer [04.11.2016]

JCSI 2 (2016) 104-107
Journal of Computer Sciences Institute

Wysłane: 2016-11-14
Przyjęte: 2016-11-21

Optymalna alokacja zasobów dźwiękowych urządzenia mobilnego na potrzeby gier

Arkadiusz Wrzos*, Jakub Smołka
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Urządzenia mobilne dysponują ograniczonymi zasobami. Szczególnie gry potrzebują specjalnego podejścia od programisty. Większość wzorców projektowych pozwala osiągnąć statyczne zużycie pamięci i niskie użycie procesora, jednak ciągle jest wiele sytuacji gdzie wytworzenie płynnie działającej funkcjonalności jest wymagającym zadaniem, wymagającym przeprowadzenia badań. W tej pracy autor bada optymalną alokację zasobów dźwiękowych do odtwarzania ich z minimalnym opóźnieniem w języku Java na systemie Android.

Słowa kluczowe: android; audio; java

*Autor do korespondencji.

Adres e-mails: arkadiusz.wrzos@pollub.edu.pl

Optimal allocation of mobile device sound resources for game programming purposes

Arkadiusz Wrzos*, Jakub Smołka
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Android is a mobile device platform, whose resources are limited. Games, and high-performance applications need special approach. Most of programming patterns allow for static memory allocation or maintaining low processor usage. Regardless, there are still many situations, in which development is demanding task, and needs much research. In this work author studied and research for efficient allocation of audio resources to play them with minimal latency in non deterministic way.

Keywords: android; audio; java

*Corresponding author.

E-mail address/addresses: arkadiusz.wrzos@pollub.edu.pl

1. Wstęp

Aktualnie dostęp do urządzeń mobilnych wysokiej wydajności jest powszechny. Prace nad architekturą ARM i układami dla smartfonów i tabletów pozwalają na dostarczanie coraz wydajniejszych urządzeń, które już teraz są w stanie realizować większość zadań typowych dla komputerów osobistych, takich jak komunikacja, multimedia, rozrywka itp.. Popularność zyskują interfejsy służące do przyłączania ich do telewizorów oraz monitorów, dzięki czemu już teraz z powodzeniem można ich używać zamiast komputera stacjonarnego w codziennych zadaniach, takich jak przeglądanie internetu, obsługa poczty czy kina domowego [1]. Statystycznie mieszkaniec Polski spędza dziennie 6,3 godzin przy komputerze, z czego częściej używa urządzeń przenośnych niż komputerów osobistych [2]. Pozwalają na to coraz wydajniejsze ogniwa elektryczne oraz procesory heterogeniczne (np. big.LITTLE [3]) z coraz mniejszym zapotrzebowaniem na prąd. Intensywny rozwój tej branży komputerowej stawia wiele nowych wymagań przed programistami. Mimo wysokiej wydajności, trzeba mieć świadomość, że zasoby ciągle są ograniczone. Zmusza to twórców oprogramowania do odpowiedniego balansowania pomiędzy maksymalnym obciążeniem procesora a minimalnym zużyciem baterii. W niektórych dziedzinach, takich jak przetwarzanie dźwięku, system android jest ciągle intensywnie rozwijany, i wymaga specjalnych podejść by osiągnąć zadowalające rozwiązania.

W pracy wykonano badania optymalnego wykorzystania wybranych narzędzi dostarczanych przez system Android, dla osiągnięcia wydajnej prezentacji multimediów. Wykorzystano w tym celu autorską aplikację rozrywkową, w której zadaniem osadzonego w kosmicznej scenerii gracza jest unikanie pocisków przeciwnika, oraz stukania w część ekranu z przeciwnikami zgodnie z rytmika podkładu muzycznego.

Rys. 1. Zrzuty ekranu z badanej gry

W powyższej aplikacji użytkownik kieruje bohaterem przesuwając go ku przeciwległym brzegom ekranu, oraz stuka w niebieskich przeciwników, którzy wylatują z przeciwnej strony.

2. Badania alokacji zasobów

Dosyć podobnym problemem zajmują się producenci narzędzi dla sekwencerów i aplikacji muzycznych SuperPowered, jednak ich rozwiązania, podobnie jak Android NDK są niskopoziomowe [4], wymagają dużego nakładu pracy przy implementacji.

Na konferencji Google I/O 2016 przedstawiono narzędzia Android Professional Audio znacznie obniżające opóźnienia dźwiękowe na wybranych urządzeniach z systemem Android [5]. Teoretycznie umożliwi to osiągnięcie opóźnień sprzętowych w odtwarzaniu dźwięku na poziomie 3,7ms co jest bardzo dobrym wynikiem, biorąc pod uwagę, że komputerowe karty dźwiękowe sięgają 2ms. Technologia jest w trakcie intensywnego rozwoju i najpewniej w kolejnych wersjach systemu stanie się standardem.

3. Badanie alokacji zasobów dźwiękowych

Gry towarzyszą komputerom od bardzo dawna, dzięki czemu programiści zdążyli już opisać bazę optymalnych podejść do najpopularniejszych problemów [6]. W aplikacji przygotowanej na potrzeby artykułu wykorzystano wiele spośród tych rozwiązań [7]. Zgodnie z nimi Aktualizacja stanu gry realizowana jest w pętli gry, która manipulując czasem potrzebnym na wykonanie cyklu reguluje liczbę klatek tak, by osiągnąć zadowalające efekty. Detekcja kolizji jest wykonana przez analizę części wspólnych prostokątnych instancji obiektów ruchomych. Animacje wykorzystują atlasy tekstur i przesuwają się pośród klatek dla aktualizowania wyświetlanej klatki. Te i inne problemy są doskonale znane programistom gier. Problem, który wymagał specjalnej analizy w tym przypadku polegał na potrzebie uzyskania ciągłej linii melodycznej złożonej z krótkich wycinków, które odgrywane byłyby w różny sposób w zależności od powodzenia gracza. Dla przykładu: odtwarzanie zapętlonej linii perkusyjnej miałoby być zastąpione odtwarzaniem zapętlonymi liniami perkusyjną i gitarową. Analiza wystąpienia dotknięcia ekranu w odpowiednim momencie powinna zapisywać sukces, który mechanizm odtwarzania muzyki powinien zinterpretować i przedstawić odpowiednie dźwięki w tle w czasie możliwie szybkim, najlepiej niezauważalnym. Poszukiwania wzorców, metod i wskazówek nie przyniosły skutku, prawdopodobnie ze względu na bardzo specyficzne zastosowanie mechanizmów odtwarzania dźwięku.

3.1. Plan badań

Wykonano serię badań porównującą wydajność podstawowych klas systemowych dostarczanych z Android SDK pozwalających na odtwarzanie dźwięku. Każda z nich została zaimplementowana i następowało dwudziestosekundowe badanie sygnału wyjściowego audio, polegające na stawianiu markerów na początku i końcu każdej próbki i mierzeniu przerwy między nimi. Badano klasy SoundPool i MediaPlayer. Odtwarzanie dźwięków było wykonywane w cyklicznych odstępach czasu, a ich zmiany również były cykliczne. Program wykonywał zadanie w osobnym wątku, i nie posiadając spowalniającego sprzężenia z wątkiem głównym mógł być oddelegowany do osobnego rdzenia procesora [8]. Dla porównania wykonano sprawdzenie odtworzenia pojedynczej próbki w zapętleniu. Badanie przeprowadzono na urządzeniach OnePlus One i Sony E4g. Próby miały sprawdzić, czy istnieje możliwość wysokopoziomowego wykorzystania klas SoundPool i MediaPlayer do naprzemiennego odtwarzania czterech próbek dźwiękowych z niską latencją.

Badania przewidywały sześć scenariuszy, w których każdy był wykonywany w sześciu krokach:

1) Wygenerowaniu dźwięku na podstawie przebiegu sinusoidalnego
2) Implementacji wyzwalania kolejnych próbek dźwiękowych w wątku powtarzającym odtwarzanie w odstępie czasu równym długości próbki.
3) Instalacji i uruchomieniu programu na urządzeniu testowym
4) Przechwyceniu sygnału wyjściowego audio z urządzenia na którym uruchomiony był program
5) Oznaczenie markerami czasów początków i końców odtwarzania próbek
6) Wprowadzenie danych do arkusza kalkulacyjnego i pomiar długości opóźnień i ich sumy i średniej oraz liczby wystąpień próbek i liczby straconych próbek.

Zbadano następujące mechanizmy:

1) Użycie klasy SoundPool do odtwarzania próbek w cyklu czasowym
2) Użycie klasy SoundPool do odtwarzania próbek w cyklu czasowym, z optymalizacją. Optymalizacja polegała na ustawieniu odpowiednich flag w sterowniku urządzenia, informujących o priorytecie i sposobie wykorzystania sterownika. Dodatkowo zoptymalizowano próbki dźwiękowe tak, by miały zgodną z urządzeniem częstotliwość, tak, by nie występowała konieczność resamplingu strumienia do częstotliwości stosowanej wewnętrznie przez system i układ dźwiękowy.
3) Użycie klasy SoundPool do odtworzenia pojedynczej próbki w zapętleniu. Udostępniona jest jedynie metoda odtwarzania pojedynczej próbki w zapętleniu, nie można zmieniać ich w czasie trwania.
4) Użycie wielu instancji klasy SoundPool. Każda instancja była przeznaczona do odtwarzania pojedynczej próbki w zapętleniu. Wszystkie obiekty zostały wstępnie zatrzymane i uruchamiane naprzemiennie, tak żeby uniknąć wewnętrznego ładowania bufora dla każdej z nich występującego przy podmianie pliku-jak w punktach 1. i 2.
5) Użycie klasy MediaPlayer do odtwarzania próbek w cyklu czasowym.
6) Użycie klasy MediaPlayer do odtworzenia połączonych w jeden plik próbek, a następnie wykorzystywanie mechanizmu przesunięcia znacznika odtwarzania do czasu odpowiadającego kolejnym próbkom.

Wyniki badań przedstawiono w tabelach 1-6.

Tabela 1. Wyniki próby SoundPool - próbki wyzwalane w cyklu czasowym

Próba	1
Pożądana liczba próbek	66,667
Wystąpiło próbek	56,737
Straconych	9,93
Suma opóźnień w ms	2979
Średnie opóźnienie w ms	52,26
Długość próbki w ms	300
Długość próby w ms	20000

Tabela 2. Wyniki próby SoundPool z optymalizacją- próbki wyzwalane w cyklu czasowym

Próba	2
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	65,81666667
Straconych	0,85
Suma latencji w ms	255
Średnia latencja w ms	18,21428571
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 3. Wyniki próby SoundPool – pojedyncza próbka wyzwalana w zapętleniu

Próba	3
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	66,66666667
Straconych	0
Suma latencji w ms	0
Średnia latencja w ms	0
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 4. Wyniki próby SoundPool – kontrolowane przełączanie zapętlonych strumieni

Próba	4
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	66,65
Straconych	0,01666666667
Suma latencji w ms	5
Średnia latencja w ms	0,07575757576
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 5. Wyniki próby MediaPlayer – próbka wyzwalana w zapętleniu

Próba	5
Pożądana liczba sampli	66,66666667
Wystąpiło sampli	63,31333333
Straconych	3,353333333
Suma latencji w ms	1006
Średnia latencja w ms	20,95833333
Długość sampla w ms	300
Długość próby w ms	20000

Tabela 6. Wyniki próby MediaPlayer – przesuwanie znacznika odtwarzania

Próba	6
Pożądana liczba sampli	40
Wystąpiło sampli	38,696
Straconych	1,304
Suma latencji w ms	652
Średnia latencja w ms	40,75
Długość sampla w ms	500
Długość próby w ms	20000

4. Wnioski

W pracy przedstawiono zestawienia opóźnień i przerw w odgrywaniu dźwięku występujących przy dynamicznym zmienianiu jego źródła/pozycji. Do pomiarów wykorzystano różne implementacje klas systemowych.

Najgorsze wyniki napotkano przy klasie MediaPlayer, która mimo iż zgodnie z dokumentacją nie służy do odtwarzania dźwięku z niskimi opóźnieniami [9], dostarcza możliwość przesuwania markera aktualnie odtwarzanego dźwięku. Umożliwia to zmienianie dźwięków przetrzymując wszystkie w jednym pliku. Rozwiązanie okazało się bardzo niewydajne i powodowało stosunkowo duże opóźnienia zarówno przy wyzwalaniu próbki w zapętleniu (Tabela 5.), jak i podczas przesuwania markera znacznika czasowego (Tabela 6.).

Lepsze wyniki osiągnięto korzystając z klasy SoundPool, gdzie opóźnienia były niskie (Tabela 1.), a po dodatkowej konfiguracji dopasowującej parametry dźwięku do parametrów sprzętowych opóźnienia zostały praktycznie wyeliminowane (Tabela 2.). Odstępy między dźwiękami były znikome, a czasem nie występowały wcale. Kluczowym jest przygotowanie próbek spójnych z parametrami urządzenia. Interesujące jest też to, że odtwarzanie pojedynczej próbki nie tworzy opóźnień (Tabela 3.).

Metoda realizująca przełączanie zatrzymanych strumieni okazała się nieprzydatna, ponieważ nie dawało możliwości synchronizacji strumieni ze sobą. Każde opóźnienie powodowało utratę synchronizacji, a suma opóźnień spowodowała, że na koniec próby dźwięk był nie do zaakceptowania (Tabela 4.) .

Najlepszym i skutecznym sposobem było się podejście drugie, czyli „SoundPool z optymalizacją - próbki wyzwalane w cyklu czasowym„. Dostarczało ono najbardziej stabilne opóźnienia na dosyć niskim poziomie. Po wykonaniu prób zamieniono dźwięki wygenerowane na podstawie przebiegu sinusoidalnego na fragmenty linii melodycznej i według empirycznej oceny to rozwiązanie dostarczyło zadowalające wrażenia słuchowe.

Należy wziąć pod uwagę dużą fragmentację rynku urządzeń z systemem Android i pamiętać, że różnice w wydajności urządzeń poszczególnej klasy mogą dawać niejednakowe wyniki przy powyższych metodach, dlatego okazuje się, że są one wydajne na nowszych modelach. Testy odbywały się na wydajnych urządzeniach z systemem w wersji 5 (Lollipop) i wielordzeniowymi procesorami.

Ze względu na niezwykle dynamiczny rozwój systemu android i narzędzi z nim związanych większość wiedzy opiera się o internetową dokumentację techniczną, oraz nieliczne pozycje w literaturze.

Nie znaleziono publikacji naukowych powiązanych z pracą.

Literatura

[1] https://liliputing.com/2016/07/andromiums-99-superbook-turns-android-phone-laptop-crowdfunding.html Andromium's $99 Superbook turns your Android phone into a laptop (crowdfunding) [04.11.2016]

[2] http://qz.com/214307/mary-meeker-2014-internet-trends-report-all-the-slides/ - Mary Meeker's 2014 internet trends report: all the slides plus highlights [04.11.2016]

[3] https://www.arm.com/products/processors/technologies/biglittleprocessing.php - Presenting the Next Generation of Mobile Processing [04.11.2016]

[4] http://superpowered.com/superpowered-audio-sdk-for-ios-and-android About Superpowered Audio SDK for iOS, OSX and Android [04.11.2016]

[5] https://www.codechannels.com/video/Google/android/android-high-performance-audio-google-io-2016/ Android high-performance audio – Google I/O 2016 [04.11.2016]

[6] Ernest Adams: Projektowanie gier. Podstawy.Helion, Gliwice 2011

[7] Robert Nystom: Game Programming Patterns ISBN: 978-0-9905829-2-2

[8] Anders Göransson: Android. Aplikacje wielowątkowe. Techniki przetwarzania. Helion 2015

[9] https://developer.android.com/reference/android/media/MediaPlayer.html - MediaPlayer [04.11.2016]

JCSI 2 (2016) 114-119
Journal of Computer Sciences Institute

Wysłane: 2016-11-20
Przyjęte: 2016-11-25

Biblioteki AngularJS i ReactJS - analiza wydajnościowa

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

Streszczenie. Artykuł zawiera porównanie wydajności bibliotek języka JavaScript. Analizie podlegają programy napisane z wykorzystaniem AngularJS oraz ReactJS. Badanie wydajności zostało zrealizowane poprzez implementację aplikacji korzystających z języka JavaScript, z użyciem obu bibliotek. Na każdej z nich wykonano pomiary wydajności, na ich podstawie sporządzono analizę wyników. Analizę wykonano przy użyciu narzędzi developerskich przeglądarek internetowych oraz poprzez zawarty w kodzie mechanizm badawczy.

Słowa kluczowe: Angular; React; wydajność

*Autor do korespondencji.

Adres e-mail: karol.kowalczyk@pollub.edu.pl

AngularJS and ReactJS libraries - performance analysis

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

Abstract. The article contains a comparison of the performance of selected JavaScript libraries. The analyzed programs have been written using AngularJS and ReactJS. Performance testing was carried out through the implementation of several applications using JavaScript, with a usage of both libraries. For each of them performance measurement was made and on its basis the results analysis was prepared. The analysis was performed using the development tools and by a mechanism contained in the code.

Keywords: Angular; React; performance

*Corresponding author.

E-mail address: karol.kowalczyk@pollub.edu.pl

1. Wstęp

Aplikacje internetowe z biegiem czasu zyskują coraz większą popularność. Ich twórcy starają się jak najbardziej ułatwić sobie pracę nad nimi. Z tego powodu powstaje wiele różnorodnych frameworków oraz bibliotek. Pisanie aplikacji z ich wykorzystaniem staje się coraz mniej skomplikowane, niemniej ważna jest ich wydajność. Głównym czynnikiem wpływającym na zapewnienie komfortu pracy jest kod wykonywany po stronie przeglądarki internetowej. Wiadomym jest, że aplikacja może dostawać dane z pewnym opóźnieniem. Dobrze napisany interfejs użytkownika (ang. Frontend) może wprowadzić odczucie płynności działania strony, wczytując dane partiami na bieżąco odświeżając wyświetlaną użytkownikowi treść[1].

Analizując współczesne trendy tworzenia aplikacji, można stwierdzić, że największą popularnością cieszą się tak zwane „single-page application", czyli aplikacje zawarte na jednej stronie z dynamicznie zmieniającą się zawartością. W każdej z takich aplikacji najważniejszą technologią od strony użytkownika jest Java Script. Niestety przy nowoczesnych aplikacjach pisanie w czystym JS nie jest już opłacalne. Stosowane są różnego rodzaju frameworki, z których wyróżnić można dwa najbardziej popularne, znacząco różniące się koncepcją. Są to Angular oraz React.

Obie biblioteki oferują programiście automatyczne odświeżanie widoku po zmianie modelu. Takie udogodnienie musi wiązać się z ciągłym obserwowaniem modelu danych oraz przystosowaniem do częstej konieczności odświeżania różnych fragmentów strony internetowej. React wprowadza innowacyjne podejście opierające się na manipulowaniu na wirtualnym DOM (Document Object Model) i odświeżaniu tylko tego fragmentu strony, który rzeczywiście uległ zmianie, podczas gdy Angular wprowadza zmiany bezpośrednio[2]. Czy takie podejście rzeczywiście niesie za sobą wzrost płynności działania aplikacji? Technologia ta z powodzeniem sprawdza się w aplikacjach takich jak Facebook czy Instagram. Natomiast Angular jest wykorzystywany do tworzenia narzędzi dostarczanych przez Google [3].

Celem testów jest zbadanie obydwu bibliotek pod kątem wydajności. W Internecie można znaleźć wiele sprzecznych informacji na ten temat. Przeprowadzając testy postaram się rozwiać wszelkie wątpliwości i odpowiedzieć na pytanie „Korzystanie z której biblioteki jest korzystniejsze pod kątem wydajności?". Dodatkowo artykuł zawiera porównanie bibliotek z wykorzystaniem różnych przeglądarek internetowych.

Wnioski z analizy wyników testów Angular oraz React z pewnością przyczynią się do lepszego doboru technologii. Zbadane zostanie, która z nich lepiej się sprawdzi przy wykonaniu małych aplikacji, a która przy większych. Jaką technologię należy wybrać chcąc rozpocząć tworzenie bardziej statycznej strony, a jaką dla dynamicznej. Dzięki tej wiedzy będzie można tworzyć nową aplikację w jednej

z badanych bibliotek, bez wątpliwości czy ta druga nie byłaby lepsza [4, 5].

2. Realizacja testów

2.1. Sposób testowania

Testy wydajnościowe przeprowadzone zostały na podstawie trzech aplikacji, z których każda bada inny aspekt wykorzystania bibliotek. Podejście to daje możliwość przetestowania obu technologii w różnych, niezależnych sytuacjach. W pierwszej kolejności zbadano aplikację prezentującą na ekranie losowy ciąg wyrazów. Aplikacja została wykonana w obu technologiach. Zbadano szybkość ładowania oraz zużycie pamięci. Następnie wykonane zostały testy podstawowych funkcji Java Script, a na koniec testom poddana została aplikacja prezentująca tabelę danych o różnych rozmiarach. Zasymulowano duże obciążenie danymi, które w kolejnych testach poddano dynamicznej zmianie [6]. Poniżej przedstawiono konfigurację sprzętową maszyny testującej, na której wykonane zostały testy:

- procesor: Intel Core i7-4710HQ (4 rdzenie, od 2.50 GHz do 3.50 GHz, 6 MB cache),
- pamięć: 8 GB (SO-DIMM DDR3, 1600 MHz)
- dysk: 1000 GB SATA 7200 obrotów,
- grafika: + Intel HD Graphics 4600, NVIDIA GeForce GTX 860M,
- system operacyjny : Windows 8.1 Pro 64-bit (6.3, Build 9600) .

Wszystkie przypadki rozpatrzone zostały przy użyciu trzech najbardziej popularnych przeglądarek:

- Google Chrome wersja 54.0.2840.71,
- Mozilla Firefox wersja 49.0.2,
- Internet Explorer wersja 11.0.9600.16384.

Pomiary wykonane zostały przy użyciu narzędzi deweloperskich oraz biblioteki JSLitmus.

Narzędzia developerskie – to zestaw narzędzi dostarczany wraz z przeglądarką internetową, pozwalające programistom na testowanie oraz debugowanie elementów strony internetowej. Niektóre z nich pozwalają na dynamiczną zmianę kodu HTML, JS czy też CSS. Aktualnie każda znana przeglądarka jest w nie wyposażona.

JSLitmus - jest to narzędzie typu open source dające możliwość nie tylko badania wydajności funkcji, ale też tworzenia na tej podstawie wykresów. Jego zaletą jest niewielki rozmiar kodu źródłowego, co pozwala na instalację jedynie poprzez dodanie jednego pliku do aplikacji. Testy wykonane tą biblioteką pokazują ile razy na sekundę przeglądarka może wykonać badany kod. Takie podejście bardzo dobrze sprawdza się w charakteryzowaniu wydajności funkcji.

2.2. Badane parametry

Czas odpowiedzi – jest to główny czynnik odpowiadający za wydajność. Znacząco wpływa na komfort korzystania z aplikacji. Na czas odpowiedzi strony internetowej składają się między innymi czasy wczytywania, wykonywania skryptów oraz czasy rysowania czy też renderowania strony. Do badania tych parametrów zastosowane zostanie narzędzie „timeline” oraz opisany wcześniej JSLitmus[7].

Zużycie zasobów – zbadano pamięć, jaką zajmuje aplikacja, z wyróżnieniem pamięci potrzebnej na przechowanie obiektów Java Script oraz elementów DOM. Parametr ten jest mniej odczuwalny dla użytkownika, jednakże nie pozostaje bez znaczenia, szczególnie w przypadku dużych aplikacji. Do badania tego parametru wykorzystana została zakładka „profiles”.

Rozmiar skryptu – to liczba linii kodu. Parametr bardzo ważny z punktu widzenia twórcy, ponieważ mniej linii kodu zwykle niesie za sobą mniejszy wkład programisty w implementację. Przy badaniu tego parametru pomijane są puste linie.

Płynność działania przy dużym obciążeniu danymi – z użyciem wspomnianej wcześniej aplikacji, zbadano jak radzą sobie one z prezentacją dużej liczby danych tekstowych oraz wizualnych na ekranie. Wykonane zostaną testy dla tabeli danych o rozmiarach 10x10, 20x20, 30x30 oraz 40x40. Tabela zawierać będzie losowe dane numeryczne wraz z losowym kolorem wypełnienia. Uwzględniona zostanie ich modyfikacja. Umożliwi to określenie komfortu pracy z taką aplikacją.

Działanie przy dynamicznie zmieniających się danych – dane z poprzedniego punktu zostały przegenerowane z losowym opóźnieniem o maksymalnej wartości 1000ms. Zasymulowało to pracę aplikacji przy dużej liczbie zmieniających się danych[8].

3. Badania

Pierwsza zaprezentowana tabela przedstawia analizę liczby linii kodu potrzebnych na wykonanie poszczególnych aplikacji.

1) Aplikacja wyświetlająca losowy ciąg wyrazów o wybranej przez użytkownika liczbie.
2) Aplikacja pozwalająca na sortowanie oraz wyszukiwanie elementów w tablicy o wybranym rozmiarze.
3) Aplikacja prezentująca tablicę wypełnioną losowymi danymi, umożliwiająca ich dynamiczną zmianę wraz ze zmianą koloru tła komórki.

Tabela 1. Zestawienie liczby linii kody potrzebnych na implementację

Aplikacja	Angular	React
1	36	41
2	69	77
3	134	151

Pod tym względem zdecydowanie lepszy okazał się AngularJS. Jego przewaga zwiększa się wraz ze wzrostem złożoności aplikacji.

3.1. Testy aplikacji do wyświetlania tekstu

W pierwszej kolejności wykonano testy aplikacji wyświetlającej losowy tekst na ekranie. Czas jaki przedstawia tabela, to czas jaki upłynął od momentu przeładowania strony, do czasu pojawienia się na niej tekstu.

Tabela 2. Wyniki testów aplikacji do wyświetlania tekstu – analiza czasu odpowiedzi

	Liczba danych	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	100	149,8ms	83,5ms	87,3ms
	10000	185,3ms	130,5ms	96,2ms
	1000000	1930,2ms	1800,0ms	960,4ms
React	100	280,4ms	350,0ms	245,0ms
	10000	305,4ms	358,0ms	247,0ms
	1000000	2475,4ms	2320,0ms	1480,0ms

Z wyników pierwszego testu wyraźnie widać, że aplikacje napisane w Angular uruchamiają się o wiele szybciej niż React. Przyczyną wolniejszego wczytywania się strony w React jest czas wykonywania skryptów. Jak wiadomo, podczas inicjalizacji musi on bowiem stworzyć i przegenerować swój wirtualny DOM. Dodatkowym środkiem spowalniającym jest wykorzystanie typu dokumentu „text/babel". Jest to typ pozwalający na używanie HTML w plikach Java Script. Stwarza to konieczność dodatkowego parsowania kodu HTML zawartego w JS, co dodatkowo obciąża przeglądarkę podczas wczytywania.

Następnym badanym czynnikiem jest wielkość pamięci zajmowanej przez aplikację. Ponieważ w tym przypadku wielkość zajmowanej pamięci wskazana przez przeglądarkę po załadowaniu strony nie ulegała zmianie, wykonano jeden pomiar dla każdego z parametrów.

Tabela 3. Wyniki testów aplikacji do wyświetlania tekstu – analiza zużycia pamięci

	Liczba danych	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	100	4,20MB	1,22MB	0,45MB
	10000	4,25MB	1,29MB	0,93MB
	1000000	5,80MB	7,95MB	5,16MB
React	100	7,72MB	6,97MB	2,16MB
	10000	8,40MB	7,03MB	2,63MB
	1000000	9,80MB	13,71MB	6,86MB

Wielkość pamięci jaką zajmują obydwie aplikacje na poszczególnych przeglądarkach różni się znacząco. Ma na to wpływ fakt, iż różne przeglądarki biorą pod uwagę różne elementy, prezentując ile pamięci zajmuje strona. Jednakże z porównania jednoznacznie widać, że Angular zajmuje zdecydowanie mniej pamięci przeglądarki dla każdego z testowanych przypadków.

3.2. Testy sortowania oraz wyszukiwania

Kolejny program posłużył do wykonania pomiarów funkcji języka Java Script. Testowanymi algorytmami były sortowanie oraz wyszukiwanie danych w tablicy. Pierwszym badanym parametrem był czas wykonywania skryptów. Oddzielnie wykonano testy sortowania oraz wyszukiwania. Pomiary wykonano z wykorzystaniem JSLitmus.

Tabela 4. Wyniki testów aplikacji do wyszukiwania – analiza liczba wykonań na sekundę

	Liczba danych	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	100	31503	26402	35199
	1000	9312	11940	8809
	10000	1122	1089	550
	100000	47	98	35
React	100	30386	76420	28854
	1000	9252	28100	8782
	10000	1292	1033	545
	100000	46	78	34

Wyniki testów przedstawione w tabeli 4 pokazują, że lekką przewagę w czasie wyszukiwania danych zyskał AngularJS. Wynik ten można zaobserwować na każdej z przeglądarek oraz dla każdego badanego rozmiaru tablicy wejściowej. Można też zauważyć, że przeglądarka Internet Explorer wykonała obliczenia najwolniej dla obydwu technologii.

Tabela 5. Wyniki testów aplikacji do sortowania – analiza liczba wykonań na sekundę

	Liczba danych	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	100	6209	6980	7448
	1000	725	689	1210
	10000	64	59	85
	100000	4	3	5
React	100	16359	21409	26709
	1000	1885	3650	4058
	10000	168	250	317
	100000	35	59	23

Test sortowania danych przedstawiony w tabeli 5 prezentuje nieco odmienne wyniki. Okazało się, że z tym testem znacznie lepiej poradził sobie ReactJS. Uwagę w wynikach testu przykuwa duża rozbieżność w czasach wykonania na przeglądarce Mozilla Firefox. Osiągnęła ona najgorszy wśród wszystkich przeglądarek wynik z biblioteką Angular, ale okazała się być najszybsza z React. Bardzo dobry wynik z biblioteką Angular osiągnęła przeglądarka Internet Explorer.

Zastanawiać może co spowodowało, że przy sortowaniu znacząco lepszy okazał się React, natomiast przy wyszukiwaniu lekką przewagę zyskał Angular. Stało się tak ponieważ podczas implementacji sortowania w Angular wykorzystano gotowy filtr dostarczony wraz z biblioteką, natomiast do wyszukiwania wykorzystano możliwość implementacji własnego. Gotowy filtr do sortowania dostarczony przez bibliotekę pozwala na zaawansowane sortowanie wielu rodzajów danych, nie tylko tablic. Jest on więc bardziej uniwersalny, co wpływa negatywnie na jego wydajność. Aby korzystając z AngularJS, osiągnąć wysoką wydajność aplikacji, warto używać własnych filtrów. Zwykle jednak nie sortuje się tak dużych ilości danych, aby mogło to znacząco obniżyć wydajność strony, a stosowanie uniwersalnego filtra jest o wiele prostsze.

W następnej kolejności zbadano pamięć, jaką przeglądarka przeznaczyła na aplikację w każdej z technologii.

Tabela 6. Wyniki testów aplikacji do sortowania danych – analiza zużycia pamięci

	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	6,30MB	1,03MB	0,87MB
React	6,20MB	2,61MB	1,26MB

Dla Google Chrome można zaobserwować porównywalne wartości tego parametru. W przypadku pozostałych przeglądarek nieco więcej pamięci potrzebuje ReactJS. Porównując wartości zajmowanej pamięci przez aplikację z pierwszego testu z aplikacją z testu numer dwa widać, że druga aplikacja napisana w ReactJS zajmuje o połowę mniej niż pierwsza. Jest tak, ponieważ do wykonania aplikacji drugiej nie było potrzeby wykorzystania kodu html w Java Script, czyli type="text/babel". Jak widać, jego wykorzystanie znacząco wpływa na wielkość potrzebnej pamięci. Mimo to obydwie aplikacje zajmują więcej niż te pisane w AngularJS.

3.3. Testy dyrektyw

Bindowanie zmiennych jest chyba największym udogodnieniem dostarczanym przez obie biblioteki. Niegdyś programista musiał sam martwić się, aby dane przetwarzane przez kod Java Script trafiły do widoku, a widok został odświeżony. Teraz to wszystko za programistę robi biblioteka. Niestety jak wynikło z analizy, bindowanie najmocniej wpływa na wydajność aplikacji. Dodatkowo obydwie biblioteki mają różne sposoby na obsługę odświeżania widoku. Testy przeprowadzone w tym punkcie dadzą odpowiedź, która z nich lepiej sobie z tym radzi pod względem wydajności. Początkowo zostało sprawdzone działanie aplikacji obarczonej dużą liczbą powiązanych z widokiem zmiennych, następnie sprawdzono jak każda z nich zareaguje na dużą liczbę zapytań typu AJAX, z których każde zechce odświeżyć fragment widoku.

Pierwszym testem było uruchomienie aplikacji z tablicą danych 20x20, 30x30, 40x40. Zbadany został czas odpowiedzi na zmianę jednego elementu, rozmiar skryptu oraz ilość zajmowanej pamięci. Początkowo opóźnienie serwera ustawiono na 1s.

Tabela 7. Wyniki testów aplikacji do testowanie dyrektyw – analiza czasu odpowiedzi dla 1 elementu

	Rozmiar tablicy	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	10x10	14,6ms	51,1ms	35,3ms
	20x20	48,1ms	103,1ms	64,2ms
	30x30	136,3ms	147,5ms	122,5ms
	40x40	171,6ms	193,4ms	182.0ms
React	10x10	12,9ms	12,6ms	32.3ms
	20x20	37,3ms	27,3ms	59.1ms
	30x30	80,0ms	39,3ms	87.5ms
	40x40	142,3ms	57,6ms	123,4ms

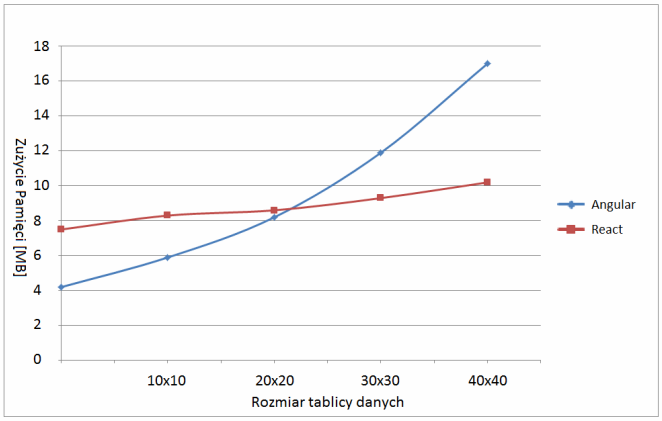
Podczas testowania zauważono, że Angular znacznie więcej czasu poświęca na wykonywanie skryptów, natomiast React na renderowanie strony. Wynika to ze wcześniej opisanych różnic w koncepcji odświeżania widoku. ReactJS najpierw operuje na wirtualnym DOM w celu obliczenia zmian w widoku, a czas poświęcony na te operacje zalicza się do czasu renderowania. Zauważyć jednak można, iż dłuższy czas renderowania komponentów przez React jest rekompensowany z nawiązką o wiele krótszym czasem wykonywania skryptów. Fakt ten potwierdza Tabela 7 przedstawiająca wyniki pomiarów czasu modyfikacji danych jednej komórki tabeli. Wyraźnie widać, że React jest pod tym względem o wiele szybszy od Angular. Można zaobserwować też jak dobrze React działa z przeglądarką Mozilla Firefox. Uzyskał on wynik ponad dwukrotnie lepszy od pozostałych przeglądarek. Potwierdza to poprzednie testy, które również stawiały tą przeglądarkę na czele szybkości działania z tą technologią. Podobnie jest ze słabszym działaniem AngularJS na tej przeglądarce. Uzyskała ona najgorszy wynik czasu odpowiedzi. Angular w tym przypadku najlepiej działał na Google Chrome. Podczas tego testu pojawił się problem z przeglądarką Internet Explorer. Aplikacja napisana w AngularJS nie uruchamiała się, co wymagało przeprowadzenia specjalnych modyfikacji kodu dla tej przeglądarki.

W kolejnym teście zbadano pamięć, jaką potrzebuje przeglądarka na wyświetlenie strony. Została ona zbadana, w momencie kiedy tablica została w pełni wypełniona danymi[9].

Tabela 8. Wyniki testów aplikacji do testowanie dyrektyw – analiza zużycia pamięci

	Rozmiar tablicy	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	10x10	5,9MB	2,6MB	2,5MB
	20x20	8,2MB	5,6MB	4,4MB
	30x30	11,9MB	11,7MB	8,5MB
	40x40	17,0MB	18,7MB	14,3MB
React	10x10	8,3MB	6,5MB	2,9MB
	20x20	8,6MB	7,7MB	3,9MB
	30x30	9,3MB	9,4MB	6,0MB
	40x40	10,2MB	11,7MB	7,5MB

Analiza wyników tych pomiarów daje bardzo ciekawe rezultaty. Można zauważyć, że dla tabeli danych 10x10 wyraźnie mniej pamięci wymaga AngularJS, natomiast wraz ze wzrostem liczby elementów na stronie traci on swoją przewagę. Dla tablicy 20x20 zaobserwowano bardziej zbliżone wyniki. Sytuacja odwróciła się dla tablicy 30x30. Aplikacja w Angular dla takich danych potrzebuje nieco więcej zasobów. Podobnie dla największej badanej tablicy. Tutaj wynik zużycia pamięci przez React jest prawie dwa razy mniejszy. Dzieje się tak na każdej z przeglądarek.



Rys. 1. Wykres zużycia pamięci przez biblioteki dla przeglądarki Google Chrome

Na Rys. 1 widać jak zmienia się zapotrzebowanie na pamięć przy wzroście liczby prezentowanych elementów dla obydwu bibliotek. React odnotowuje wyraźnie mniejszą tendencję wzrostową przy zwiększaniu liczby komponentów.

Ponieważ w kolejnym etapie testowania zmienianych było wiele elementów widoku jednocześnie, zmodyfikowany został parametr odpowiedzialny za symulowanie odpowiedzi serwera. Ustawiono go na losową wartość z przedziału 0-1000ms, następnie uruchomiony został mechanizm odświeżający wszystkie elementy. Zbadano czas, jaki aplikacja potrzebowała na ukończenie tego procesu.

Tabela 9. Wyniki testów aplikacji do testowanie dyrektyw – analiza czasu modyfikacji wszystkich elementów tabeli

	Rozmiar tablicy	Google Chrome	Mozilla Firefox	Internet Explorer
Angular	10x10	1014,3ms	1006,2ms	1014,5ms
	20x20	4290,4ms	2742,5ms	2694,5ms
	30x30	19780,2ms	13457,3ms	12802,5ms
	40x40	62371,6ms	38931,4ms	37972.6ms
React	10x10	1034,9ms	1019,3ms	1039.3ms
	20x20	1292,2ms	1121,2ms	1159.2ms
	30x30	4869,0ms	1199,5ms	1485.4ms
	40x40	10642,4ms	1622,1ms	1787,6ms

Ogromną przewagę w tym teście osiągnął ReactJS. Angular kompletnie nie poradził sobie z tablicami większymi od 20x20. Na Google Chrome przy największej badanej tablicy trzeba było czekać około minuty, w przypadku gdy React wykonał to samo w 10s. W przypadku testów na innych przeglądarkach zaobserwowano brak płynności w odświeżaniu widoku. Jedyną przeglądarką, która prawidłowo prezentowała zmianę danych przy tablicach większych od 10x10, było Google Chrome. Czasy przeglądarki Firefox oraz IE są odpowiednio niższe, ponieważ dla dużej liczby danych nie odświeżały one widoku po zmianie każdego elementu. Wiele danych zmieniających się niemal jednocześnie grupowane były w jedno zdarzenie odświeżenia, co odpowiednio zmniejszyło czasy. Pomimo tego w każdej z przeglądarek znacząco lepszy okazał się React[10].

4. Wnioski

W artykule zostały porównane biblioteki Angular oraz React pod kątem wydajności. Wykonane zostały trzy aplikacje w każdej z technologii tak, aby możliwe było zbadanie bibliotek przy różnych zastosowaniach. Testy przeprowadzone zostały z wykorzystaniem trzech najpopularniejszych przeglądarek internetowych.

Ze sporządzonych analiz można wywnioskować, że odpowiedni dobór biblioteki może okazać się kluczowy jeśli chce się aby aplikacja działała możliwie najlepiej. Ciężko jednak jednoznacznie stwierdzić, która technologia jest wydajniejsza. Można natomiast wyróżnić kilka grup aplikacji.

Chcąc wykonać małą aplikację z niewielką ilością zmieniających się danych, zdecydowanie wydajniejszy okaże się Angular. Jest to biblioteka, która według testów zajmuje mniej pamięci po uruchomieniu mniejszych aplikacji. Czas

uruchamiania się stron również przemawia za tą biblioteką. Dodatkowo kody źródłowe są mniejsze, a komfort pisania aplikacji o wiele lepszy.

W przypadku średniej wielkości aplikacji, obydwie biblioteki wydają się być odpowiednie. Zużycie pamięci będzie zbliżone. Czasy odpowiedzi obydwu aplikacji dla niedużej liczby bindowanych danych również są porównywalne. Różnica zaczyna być widoczna dopiero przy ich znaczącym wzroście.

ReactJS najlepiej sprawdzi się w przypadku aplikacji z dużą liczbą danych dynamicznie zmieniających się. Jeśli chcemy napisać aplikację, w której znajdzie się powyżej 100 zmieniających się elementów, AngularJS może sobie z nią nie poradzić i wtedy zdecydowanie lepszy okaże się React. Są to jednak bardzo specyficzne przypadki.

Jeśli chodzi o przeglądarki, to można zauważyć skrajne wyniki testów przeglądarki Mozilla Firefox, która najlepiej radzi sobie z React, natomiast słabo z Angular, co również może mieć wpływ na decyzję w wyborze biblioteki.

Biorąc pod uwagę całokształt testów, można by stwierdzić, że obie biblioteki są wydajne jeśli używa się ich adekwatnie do potrzeb.

Literatura

[1] Chugh Ravi, i inni. Staged information flow for JavaScript. (2009), Conference on Programming Language Design and Implementation, 50-62.

[2] Valente Marco Tulio, Terra Ricardo i Santos Gustavo. AngularJS in the Wild: A Survey with 460 Developers. The 7th International Workshop. Amsterdam : ACM, 2016.

[3] Fedosejev Artemij. React.js Essentials. Birmingham : Packt Publishing, 2015.

[4] Green Brad i Seshadri Shyam. AngularJS. [tłum.] Robert Górczyński. Gliwice : Helion, 2014.

[5] Chansuwath Wutthichai i Senivongse Twittie. A model-driven development of web applications using AngularJS framework. 2016. 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS).

[6] Hunt Pete. React: Facebook's Functional Turn on Writing JavaScript. New York : ACM, (2016), Queue - Web, 40-57.

[7] Weyuker Elaine J. Experience with performance testing of software systems: issues, an approach, and case study. IEEE Transactions on Software Engineering 26. 2001.

[8] Khan Rizwan Performance testing (load) of webapplications based on test casemanagement. Perspectives in Science. 8, (2016), 355–357.

[9] Stępniak Wojciech i Nowak Ziemowit. Performance Analysis of SPA Web Systems. [aut. książki] Borzemski Leszek, i inni. Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology – ISAT 2016 – Part I. (2016), 235-247.

[10] Antonio Cássio de Sousa. Architecting Applications with Components. Pro React. (2015), 51-89.

JCSI 2 (2016) 120-126
Journal of Computer Sciences Institute

Wysłane: 2016-11-18
Przyjęte: 2016-11-24

Wpływ zastosowania programowania równoległego na wydajność algorytmów kryptograficznych

Mateusz Kraska*, Piotr Kozieł*

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

Streszczenie. W artykule zostały porównane możliwości zrównoleglenia dwóch algorytmów z dziedziny kryptografii: szyfrowania XOR oraz wyszukiwania kolizji funkcji skrótu MD5. Prezentowane rozwiązania zostały zaimplementowane w taki sposób, by używały wybraną przez użytkownika liczbę rdzeni procesora. Wydajność algorytmów została sprawdzona na różnych procesorach i przedstawiona na wykresach.

Słowa kluczowe: programowanie; równoległe; synchroniczne; kryptografia.

*Autor do korespondencji.

Adresy e-mail: Mateusz.kraska@pollub.edu.pl; Piotr.koziel@pollub.edu.pl

The influence of the parallel programming on the performance of cryptographic algorithms

Mateusz Kraska*, Piotr Kozieł*

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

Abstract. Paper compares possibility to parallelize two cryptographic algorithms: Xor cipher and MD5 collision search. Presented solutions allows user to specify number of used processors. Performance of algorithms where tested on a different types of processors and visualized on graphs.

Keywords: programming; parallel; asynchronous; cryptography.

*Corresponding author.

E-mail addresses: Mateusz.kraska@pollub.edu.pl; Piotr.koziel@pollub.edu.pl

1. Wstęp

Programowanie równoległe jest stosowane niemal od początku istnienia informatyki. Zrównoleglanie obliczeń w wielu przypadkach jest jednym możliwym sposobem by skrócić czas obliczeń. Jednak dopiero początek XXI wieku spowodował popularyzację tego zagadnienia. W wieku XX, większość procesorów posiadała jeden rdzeń. Producenci procesorów zwiększali ich wydajność poprzez zwiększenie częstotliwości taktowana. Zgodnie z obserwacjami Gordona Moor'a, częstotliwość taktowania podwajała się co około 2 lata. Na początku XXI wieku, produkowano procesory osiągające częstotliwość pracy bliską 4 GHz, jednak od tego czasu wzrost częstotliwości taktowania został zahamowany. Tranzystory w układach scalonych osiągnęły na tyle małe rozmiary, że dalsza ich miniaturyzacja stała się bardzo trudna. Aby zwiększać wydajność procesorów, producenci rozpoczęli produkcję procesorów wielordzeniowych. Komputery posiadające takie procesory, pojawiły się masowo już w pierwszych latach XXI wieku. Obecnie, nawet w najtańszych komputerach, laptopach, a także urządzeniach mobilnych typu smartphone procesory jednordzeniowe są spotykane bardzo rzadko. Programiści, aby w pełni wykorzystać moc urządzeń, projektują swoje aplikacje w taki sposób by wykorzystywać wiele rdzeni procesora.

Celem opisanej w niniejszym artykule pracy badawczej była analiza wydajności zastosowania programowania asynchronicznego. Posłużono się przy tym porównaniem wyników zrównoleglenia popularnych algorytmów i języków programowania stosowanych w informatyce.

2. Stanowisko badawcze

Do wykonania wszystkich badań i obliczeń posłużono się prywatnym sprzętem autorów artykułu. Uznano, że badania te będą najbardziej wartościowe gdy zostaną wykonane na sprzęcie domowym, dostępnym dla zwykłego użytkownika.

Tabela 1. Parametry sprzętu komputerowego wykorzystanego do pomiarów

Procesor	Liczba rdzeni	Pamięć RAM	System operacyjny	Wersja JVM / .NET
Intel Core i3-4160	2 fizyczne, 4 logiczne	2 X 4GB DDR3 800Mhz	Windows 10 64 bit	.NET Framework 4.6.2
AMD Athlon 64 X2 Dual Core 4200+	2 fizyczne	2 x 2GB DDR2 800MHz	Windows 7 64 bit	.NET Framework 4.6.2
MediaTek MT6580	4 fizyczne	1 x 1GB	Android 5.1 Lollipop	Nie dotyczy

3. Zrównoleglenie algorytmu XOR

Algorytm XOR jest jednym z najprostszych i jednocześnie najmniej bezpiecznych algorytmów służących do szyfrowania danych [1,2]. Jego działanie polega na wykonaniu operacji XOR na kolejnych bajtach danych, które należy zaszyfrować z kluczem. Deszyfrowanie jest operacją identyczną

wykorzystującą ten sam klucz. Szyfr wykonany tym algorytmem można złamać w bardzo prosty sposób metodą ataku statystycznego. Ze względu na łatwość implementacji oraz niską złożoność obliczeniową, algorytm XOR jest przedmiotem badań w niniejszej pracy.

Listing 1. Przykładowy kod realizujący szyfrowanie XOR

```
void Method1(byte[] array, byte key)
{
    for (inti = 0; i<array.length; i++)
    {
     array[i] ^= key;
    }
}

(...)

byte[] dataArray = new byte[] { 2, 4, 17, 0, 255, 170, 1 }
Method1(dataArray, 170);
```

Pierwszy parametr metody Method1 służy do przekazania danych do zaszyfrowania lub deszyfrowania. Drugi parametr to klucz. Metoda nie zwraca danych. Zaszyfrowane zostają dane wejściowe. Po wykonaniu operacji przedstawionej na listingu 1, tablica dataArray zawierać będzie elementy: 168, 174, 187, 170, 0, 171. Aby przystosować algorytm do wykorzystywania wielu rdzeni należało zrównoleglić operację szyfrowania oraz deszyfrowania danych. W tym celu zmodyfikowano metodę Method1.

Listing 2. Przykładowy kod realizujący szyfrowanie XOR

```
void Method1(byte[] array, int start, int end, byte key)
{
    for (inti = start; i< end; i++)
    {
     array[i] ^= key;
    }
}
byte[] dataArray = new byte[] { 2, 4, 17, 0, 255, 170, 1 }
Method1(dataArray, 170,0,4);
```

Do metody dodano dwa parametry: start oraz end. Oznaczają one pierwszy oraz ostatni indeks, które zostaną zaszyfrowane. Dzięki nim obliczenia możemy rozdzielić na kilka wątków.

3.1. Wykorzystanie klasy Thread do równoległego wykonania obliczeń

Podstawową metodą do zrównoleglenia obliczeń na platformie .NET jest wykorzystanie klasy Thread [3]. Inicjalizacja obiektu Thread powoduje utworzenie nowego zarządzanego wątku na platformie .NET. Bazowy konstruktor tej klasy przyjmuje jako argument instancję klasy ThreadStart. Jednak dzięki zastosowaniu wyrażenia lambda nie musimy jawnie tworzyć obiektu klasy ThreadStart.

Poniższy kod (Listing 3) inicjalizuje wątek, który szyfruje 4 pierwsze elementy tablicy. Następnie wywołanie metody Start powoduje rozpoczęcie obliczeń w nowym wątku. Metoda Join() [3,4] służy do synchronizacji wątku głównego z nowo stworzonym wątkiem. Dzięki temu mamy pewność, że po wykonaniu metody Join() na obiekcie wątku, zakończył on swoje działanie.

Przedstawiony na listingu 3 przykład tworzy wątek za pomocą klasy Thread, ale obliczenia nie są zrównoleglone. Aby zrównoleglić obliczenia potrzebujemy co najmniej dwa wątki, pracujące jednocześnie.

Listing 3. Przykładowy kod realizujący szyfrowanie w osobym wątku

```
byte[] dataArray = new byte[] { 2, 4, 17, 0, 255, 170, 1 }
var thread = new Thread(() => {
    Method1(dataArray, 0, 3, 170);
});
thread.Start();
thread.Join();
```

Metoda przedstawiona na listingu 4 przyjmuje 4 parametry. Pierwszy z nich to dane do zaszyfrowania (lub odszyfrowania). Drugi parametrem jest klucz. Kolejny, trzeci parametr, to liczba wątków jaka ma być wykorzystana do wykonania zadania. Ostatni, czwarty parametr, to delegat do metody wykonującej szyfrowanie, czyli do metody Method1().

Listing 4. Metoda realizująca równoległe szyfrowanie za pomocą obiektów klasy Thread

```
void CalculateInManyThreads(byte[] dataArray, byte key,
intthreadCount, Action<byte[], int, int, byte> action){
    Thread[] threads = new Thread[threadCount];
    intelementsForThread = dataArray.Length / threadCount;
    for (inti = 0; i<threadCount; i++){
     int start = i * elementsForThread; int end;
     if (i == threadCount - 1){
          end = dataArray.Length;
     }
     else{
          end = (i + 1) * elementsForThread;
     }
     threads[i] = newThread(() =>     {
          action(dataArray, start, end, key);
     });
    }
    for (inti = 0; i<threadCount; i++){
     threads[i].Start();
    }
    for (inti = 0; i<threadCount; i++){
     threads[i].Join();
    }
}
```

Przedstawiona na listingu 4 Funkcja CalculateInManyThreads() najpierw tworzy tablicę, która przechowywać będzie instancje klasy Thread. Rozmiar tej tablicy jest określany przez parametr threadCount. Następnie, w pętli for, jest obliczany zakres tablicy, który ma być zaszyfrowany przez poszczególne wątki. Tworzone są także poszczególne instancje klas Thread. Kolejna pętla uruchamia wszystkie wątki za pomocą metody Start(). Ostatni fragment metody, to trzecia pętla for, której wykonywanie kończy się dopiero wtedy, gdy wszystkie wątki zakończą swoje działanie. Wywołanie metody Join na obiekcie klasy Thread blokuje wątek wywołujący, aż do czasu zakończenia pracy wątku reprezentowanego przez ten obiekt.

3.2. Wykorzystanie klasy Task do równoległego wykonania obliczeń

W czwartej wersji .NET Framework, wprowadzona została klasa Task [4, 5]. Podobnie jak Thread, służy ona do zrównoleglania operacji. Jej działanie opiera się na ThreadPool. Oznacza to, że stworzenie instancji obiektu klasy Task nie musi oznaczać utworzenia nowego wątku. O tym, czy zainicjowanie nowego obiektu Task powoduje również stworzenie obiektu Thread decyduje TaskScheduler. Jeśli TaskScheduler działa w domyślnej konfiguracji to liczba utworzonych wątków powinna być równa ilości rdzeni procesora. Ma to znaczenie wtedy, gdy chcemy wykonywać wiele różnych zadań jednocześnie, ponieważ inicjalizacja obiektu Thread jest operacją kosztowną.

Listing 5. Metoda realizująca równoległe szyfrowanie za pomocą obiektów klasy Task

```
void CalculateInManyTasks(byte[] dataArray, byte key,
inttaskCount, Action<byte[], int, int, byte> action){
    Task[] tasks = new Task[taskCount];
    intelementsForThread = dataArray.Length / taskCount;
    for (inti = 0; i<taskCount; i++){
     int start = i * elementsForThread;
     int end;
     if (i == taskCount - 1)    {
          end = dataArray.Length;
     }
     else{
          end = (i + 1) * elementsForThread;
     }
    tasks[i] = new Task(() => { action(dataArray, start, end,
key); });}
    for (inti = 0; i<taskCount; i++){
     tasks[i].Start();
    }
    Task.WaitAll(tasks);
}
```

Metoda CalculateInManyTasks() (Listing 5) jest bardzo podobna do poprzedniej o nazwie CalculateInManyThreads(). Podstawowa różnica polega na wykorzystaniu klasy Task zamiast Thread. Kolejną różnicą jest wykorzystanie metody Task.WaitAll() w celu poczekania na zakończenie pracy przez wszystkie Taski.

W CalculateInManyThreads() do tego celu służyła pętla, która czekała na wszystkie wątki za pomocą metody Join(). W przypadku tego algorytmu, synchronizacja wątków jest dość prosta w implementacji, co może nie przekonywać do zastosowania klasy Task. Oprócz tego klasa Task, posiada bliźniacze funkcje Task.WaitAny() oraz Task.WhenAny() [6]. Obie metody służą do oczekiwania na zakończenie pracy przez dowolny wątek, spośród kilku przekazanych w parametrze. Dodatkowo, metoda Task.WhenAny() zwraca Task, który jako pierwszy zakończył pracę. Task.WhenAny() została wykorzystana w rozdziale dotyczącym wyszukiwania kolizji md5. Aby przeprowadzić podobne działanie z wykorzystaniem klasy Thread zamiast Task, należałoby wykorzystać mechanizmy synchronizacji oparte na klasach AutomaticResetEvent, ManualResetEventlubSemaphor lub Semaphor co wymagało by bardziej złożonej implementacji, oraz czyni kod trudniejszym w interpretacji.

3.3. Równoległe wykonanie obliczeń z wykorzystaniem Simple Instruction Multiple Date (SIMD)

W wersji 4.6 platformy .NET Framework pojawiło się wiele nowych funkcjonalności. Jedną z nich jest wsparcie dla instrukcji procesorów Intel o nazwie Simple Inctruction Multiple Date (SIMD) [7]. Instrukcje SIMD, są to instrukcje wykonywane nie na pojedynczych danych, lecz na całych wektorach. Instrukcje SIMD są obsługiwane zarówno w procesorach w architekturze x86 jak i x64. Instrukcje w procesorach mogą się różnić. Procesory x86 Intel Pentium 4 oraz AMD Athlon 64 obsługują zestaw instrukcji SSE2. Zestaw instrukcji AVX posiada część procesorów Intel z serii Sandy Bridge, Ivybridge, Haswell. [7].

Poniższa metoda Method2() (Listing 6), realizuje to samo działanie co Method1(), ale korzystając z SIMD.

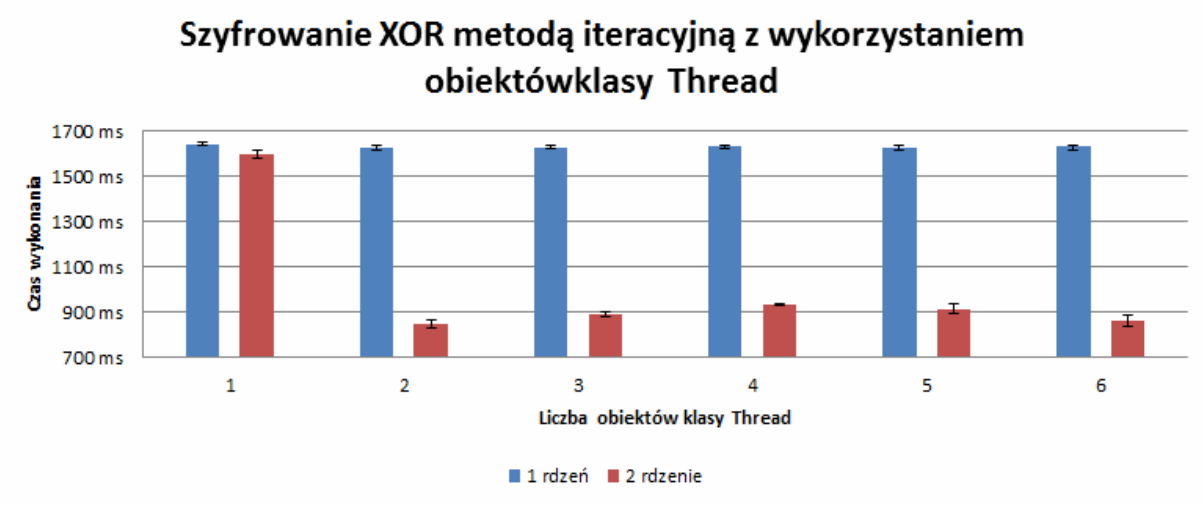
Listing 6. Metoda realizująca równoległe szyfrowanie za pomocą obiektów klasy Task

```
private static void Method2(byte[] array, int start, int end,
byte key)
{
    byte[] keysArray = new byte[Vector<byte>.Count];
    for (int j = 0; j <Vector<byte>.Count; j++){
     keysArray[j] = key;
    }
    Vector<byte>keysVector = new
Vector<byte>(keysArray);
    vartmpEnd = end - Vector<byte>.Count; inti;
    for (i = start; i<tmpEnd; i += Vector<byte>.Count){
     Vector<byte>outputVector = newVector<byte>(array, i)
^ keysVector;
     outputVector.CopyTo(array, i);
    }
    byte[] b = new byte[Vector<byte>.Count];
    Array.Copy(array, i, b, 0, end - i);
    Vector<byte>outputVector = new Vector<byte>(b) ^
keysVector;
    outputVector.CopyTo(b);
    b.Take(end - i).ToArray().CopyTo(array, i);
    }
}
```

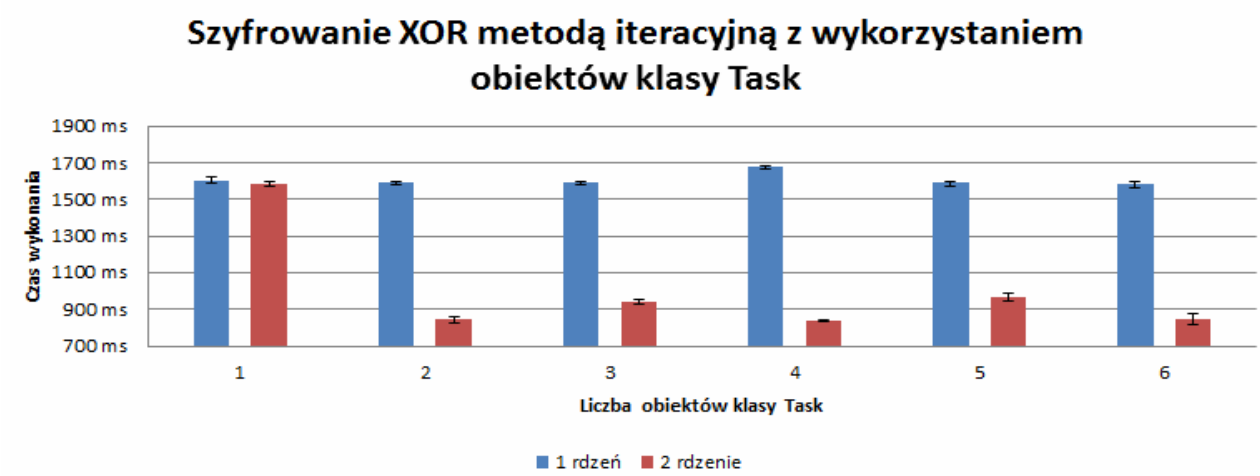
Na początku Method2 (Listing 6) tworzony jest wektor kluczy keysVector. Następnie w pętli for tworzone są wektory kolejnej porcji danych, oraz wykonywana jest operacja XOR wektora danych z wektorem klucza.

3.4. Pomiary Wydajności

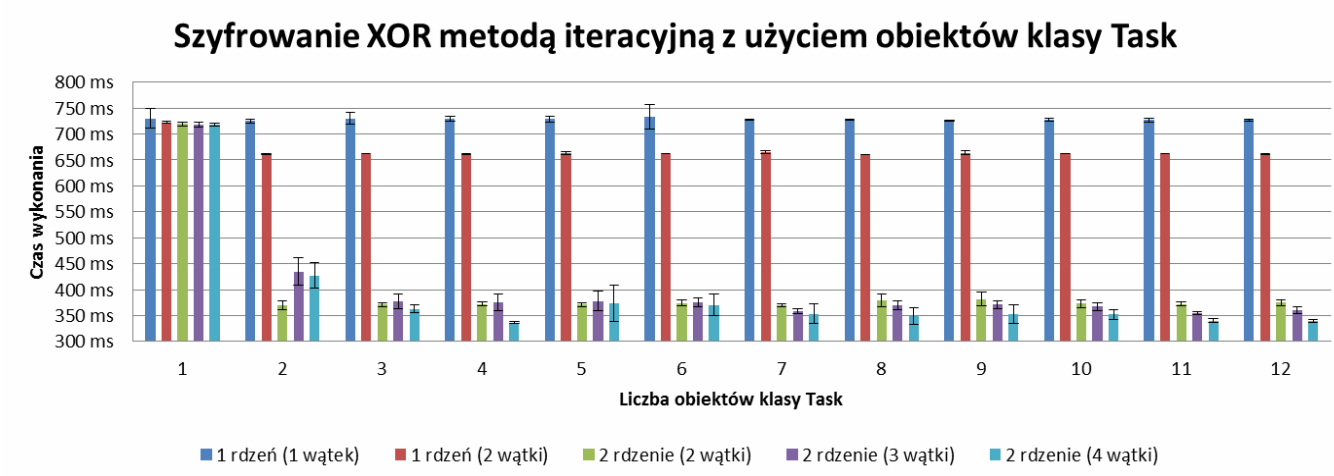
W celu porównania wydajności prezentowanych rozwiązań przeprowadzono szereg testów. Ich wyniki zaprezentowano na wykresach zamieszczonych na rysunkach 1, 2, 3 oraz 4. Na osiach rzędnych zamieszczonych wykresów przedstawiono czas szyfrowania 1GB danych, natomiast osie odciętych wskazują liczbę wątków (obiektów klasy Thread lub Task). Wykresy przedstawiają uśrednione czasy wykonywania obliczeń z 50 prób. Aby otrzymać tablicę bajtów o takim rozmiarze, utworzono plik o wielkości 1 GB z losową zawartością. Następnie, plik został wczytany do tablicy. Pomiary zostały wykonane na kilku różnych komputerach.



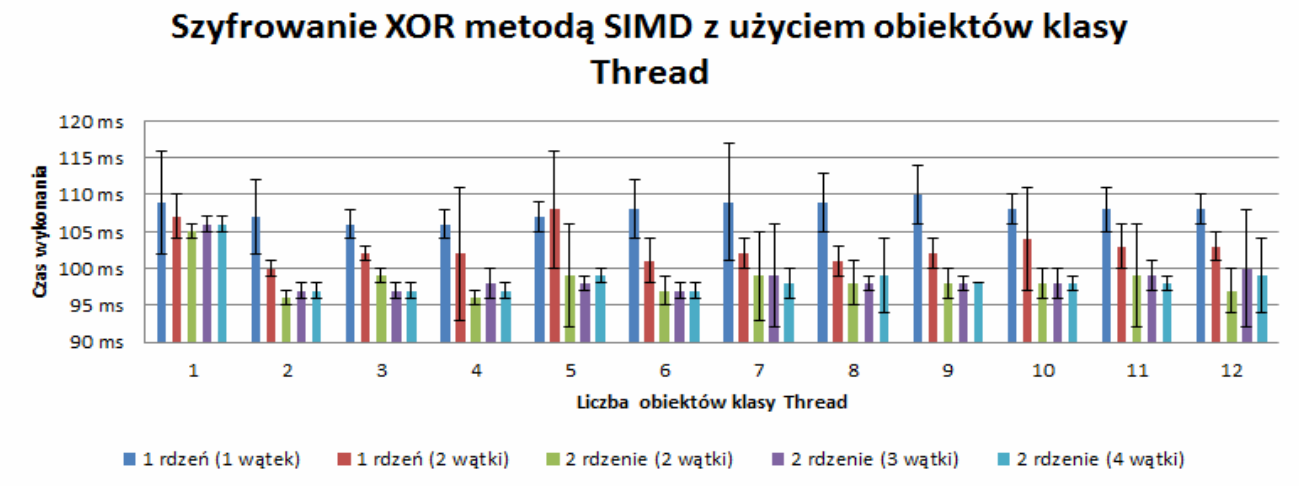
Rys. 1. Pomiar wydajności algorytmu szyforwania XOR z wykorzystaniem obiektów klas Thread na komputerze z procesorem AMD Athlon 64 X2Dual Core 4200+



Rys. 2. Pomiar wydajności algorytmu szyforwania XOR z wykorzystaniem obiektów klas Task na komputerze z procesorem AMD Athlon 64 X2 Dual Core 4200+



Rys. 3. Pomiar wydajności algorytmu szyforwania XOR z wykorzystaniem obiektów klas Task na komputerze z procesorem Intel i3-4160



Rys. 4. Pomiar wydajności algorytmu szyforwania XOR z wykorzystaniem obiektów klas Thread oraz SIMD na komputerze z procesorem Intel i3-4160

4. Znajdowanie kolizji funkcji skrótu MD5

MD5 jest popularnym algorytmem funkcji skrótu. Z ciągu danych o dowolnej (także równej zero) długości generowany jest skrót o wielkości 128 bitów. Algorytm, mimo, że jest dość stary oraz potencjalnie niebezpieczny, nadal jest często wykorzystywany. W niniejszym rozdziale przedstawiony zostanie również algorytm znajdujący kolizję metodą siłową (bruteforce). Zostaną również zbadane różnice w wydajności, gdy algorytm jest zrównoleglony.

Metoda FindCollision (Listing 7), jako parametry przyjmuje skrót, dla którego szukana będzie kolizja; losowy ciąg znaków; oraz obiekt klasy CancellationTokencancelationToken [6], który służy do przerywania obliczeń.

Listing 7. Metoda FindCollision

```
string FindCollision(string inputHash, string randomString,
CancellationTokencancelationToken) {
    string hash2 = randomString, input2;
    do{
     if (cancelationToken.IsCancellationRequested){
          return null;
     }
     input2 = hash2;
     hash2 = CalculateMD5Hash(input2);
    }
    while (inputHash.Equals(hash2));
    return input2;
}
```

Aby zrównoleglić obliczenia, napisano metodę, która w wielu wątkach uruchamia metodę CalculateInManyTasks (Listing 8).

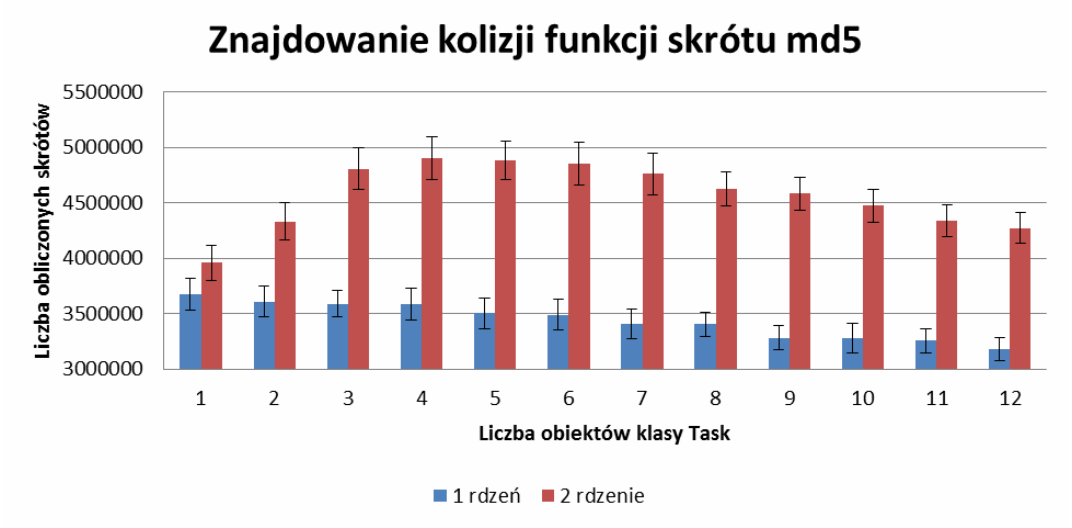
Listing 8. Metoda CalculateInManyTasks

```
static Task<string>CalculateInManyTasks(int taskCount){
    Task<string>[] tasks = new Task<string>[taskCount];
    int[] numberOfHashedCalculated = new int[taskCount];
    varcancelToken = new CancellationTokenSource();
    string input = RandomString(r);
    string hash = CalculateMD5Hash(input);
    for (inti = 0; i<taskCount; i++){
     string randomString = RandomString(r);
     int i2 = i;
     tasks[i] = new Task<string>(() =>{
          return FindCollision(hash, randomString,
     cancelToken.Token);
     });
    }
    for (inti = 0; i<taskCount; i++){
     tasks[i].Start();
    }
    TaskcompletedTask = await Task.WhenAny(tasks);
    cancelToken.Cancel();
    returncompletedTask.Result;
}
```

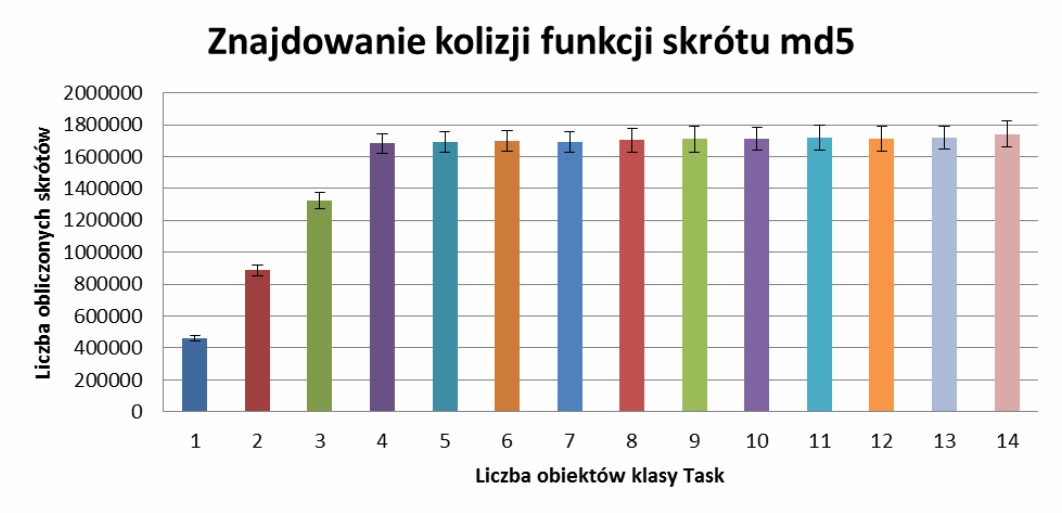
W przeciwieństwie do poprzedniego przykładu – szyfrowania danych metodą XOR, w tym doświadczeniu program nie oczekuje na zakończenie pracy wszystkich obiektów klasy Task. W momencie gdy jeden z wątków znajdzie kolizję i tym samym zakończy swoje działanie, praca pozostałych wątków nie jest już potrzebna, a więc jest kończona poprzez uruchomienie metody Cancel() obiektu cancelToken. Po zakończeniu pracy wszystkich wątków, zwracany jest wynik.

4.1. Pomiary wydajności

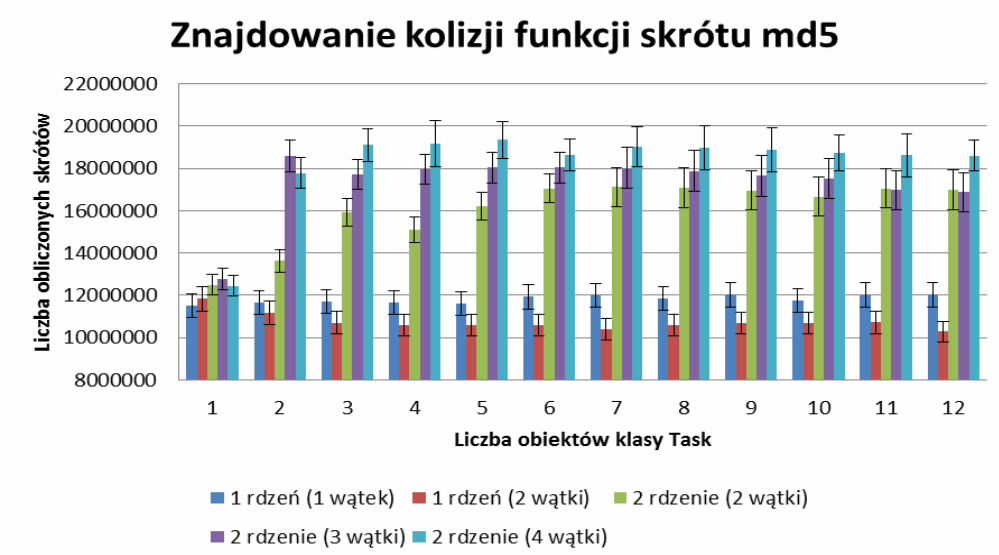
Do wykonania wszystkich pomiarów posłużono się sprzętem komputerowym opisanym w rozdziale 4. Wyniki zostały przedstawione w postaci wykresów zamieszczonych na rysunkach 5, 6 oraz 7. Na osi rzędnych przedstawiono liczbę obliczonych skrótów w ciągu 60 sekund pracy programu. Wykresy przedstawiają wartość średnią z 50 prób. Oś odciętych wskazuje liczbę obiektów klasy Task, które zostały użyte do obliczeń. Zastosowanie obiektów klasy Task zamiast Thread nie spowodowało zmiany wydajności. Na urządzeniu mobilnym VKWorld VK700 Max nie ograniczano liczby pracujących rdzeni procesora tak jak w przypadku pozostałych pomiarów. Dzięki zastosowaniu frameworka Xamarin, kod napisany w języku C# został bez istotnych zmian skompilowany na urządzenie mobilne pracujące na systmie Android 5.1. Warto zauważyć, że w przypadku tego doświadczenia, zrównoleglanie obliczeń dało największy zysk wydajności na urządzeniu mobilnym.



Rys. 5. Pomiar wydajności algorytmu wyszukiwania kolizji funkcji skrótu na komputerze z procesorem Athlon 64 X2 Dual Core 4200+



Rys. 6. Pomiar wydajności algorytmu wyszukiwania kolizji funkcji skrótu na urządzeniu mobilnym z procesorem MediaTek MT6582



Rys. 7. Pomiar wydajności algorytmu wyszukiwania kolizji funkcji skrótu na komputerze z procesorem Intel i3-4160

5. Wnioski

Wydajność programów została sprawdzona na kilku komputerach klasy PC. Uruchamiane aplikacje wykorzystywały różną liczbę wątków oraz różną liczbę rdzeni procesora. Pozwoliło to na zbadanie jaki zysk wydajności można osiągnąć dzięki wykorzystaniu kilku rdzeni procesora.

Dzięki wykorzystaniu dwóch rdzeni na komputerze z procesorem AMD Athlon 64 X2 Dual Core 4200+ udało się obliczyć 33% więcej skrótów, niż w przypadku użycia jednego rdzenia. Na komputerze z procesorem Intel i3-4160, dzięki wykorzystaniu dwóch rdzeni logicznych, na jednym rdzeniu fizycznym, udało się obliczyć 17% więcej skrótów niż w przypadku użycia tylko jednego rdzenia logicznego. Wykorzystanie dwóch rdzeni logicznych, na dwóch osobnych rdzeniach fizycznych, pozwoliło obliczyć 40% więcej funkcji skrótu, natomiast wykorzystanie wszystkich 4 rdzeni logicznych pozwoliło na obliczenie 54% więcej skrótów niż w przypadku wykorzystania jednego rdzenia logicznego.

W przypadku algorytmu szyfrowania XOR również na każdym urządzeniu, użycie większej liczby rdzeni procesora spowodowało skrócenie czasu operacji. Wykorzystanie klasy Task zamiast klasy Thread nie powoduje spadku wydajności. Na komputerze, z procesorem AMD Athlon 64 X2 Dual Core 4200+, wykonywanie obliczeń z wykorzystaniem dwóch rdzeni procesora zajmuje 50% czasu w porównaniu do obliczeń na jednym rdzeniu. Komputer z procesorem Intel i3-4160 posiada 2 rdzenie fizyczne. Na każdy z nich przypadają dwa rdzenie logiczne. Wykorzystanie dwóch rdzeni logicznych na jednym fizycznym rdzeniu procesora daje zysk wydajności około 13% w porównaniu do obliczeń na tylko jednym rdzeniu logicznym. Natomiast użycie dwóch rdzeni fizycznych daje aż 99% zysk wydajności w porównaniu do obliczeń na jednym rdzeniu. Wykorzystanie wszystkich 4 rdzeni logicznych daje 120% wzrost wydajności w porównaniu do obliczeń na jednym rdzeniu, oraz 10% wzrost wydajności w porównaniu do obliczeń na dwóch osobnych rdzeniach fizycznych. Dzięki zastosowaniu technologii SIMD, w przypadku algorytmu szyfrowania XOR, udało się osiągnąć wzrost wydajności o ponad 600% w porównaniu do metody iteracyjnej, używając tylko jednego rdzenia procesora. Użycie 4 liczby rdzeni zamiast jednego w przypadku metody SIMD, daje zysk wydajności o wartości zaledwie 10%. Warto zaznaczyć, że obliczenia metodą SIMD na jednym rdzeniu procesora wykonują się ponad 200% szybciej w porównaniu do tradycyjnej metody iteracyjnej z użyciem wszystkich 4 rdzeni logicznych procesora. Oprócz zysku wydajności, warto zaznaczyć, że dzięki zastosowaniu klasy Task, zamiast klasy Thread uzyskano krótszy, zwięźlejszy i prostszy w zrozumieniu kod.

Literatura

[1] Stallings W.: Kryptografia i bezpieczeństwo sieci komputerowych. Matematyka szyfrów i techniki kryptologii, Helion, Gliwice 2011-11-24, ISBN Książki drukowanej: 978-83-246-2986-2, 9788324629862

[2] Karbowski M.: Podstawy kryptografii. Wydanie III, Helion, Data wydania ebooka: 2015-01-08, ISBN Ebooka: 978-83-283-0958-6, 9788328309586

[3] Warczak M., Matulewski J., Pawłaszek R.: Programowanie równoległe i asynchroniczne w C# 5.0, Helion, Gliwice 2013-11-22 , ISBN Książki drukowanej: 978-83-246-6698-0, 9788324666980

[4] Lee Wei-Meng: 2008. Warsztat programisty C#, Wrox, ISBN Książki drukowanej: 978-83-246-2244-3, 9788324622443

[5] Alex Davies: Async in C#,O'Reilly Media,2012, ISBN:978-1-4493-3711-7

[6] Stephen Cleary: Concurrency in C# Cookbook, O'Reilly Media,2014, ISBN:978-1-4493-6755-8

[7] Numerics in the .NET Framework, (stan na dzień 20.04.2016r.)https://msdn.microsoft.com/en-us/library/dn879696(v=vs.110).aspx

JCSI 2 (2016) 127-132
Journal of Computer Sciences Institute

Wysłane: 2016-11-25
Przyjęte: 2016-11-28

Analiza możliwości zarządzania bazą danych przechowywaną w chmurze

Rafał Góźdź*, Maria Skublewska-Paszkowska

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

Streszczenie. W artykule przedstawiono porównawczą analizę wydajnościową wybranych systemów baz danych. Porównano czasy wykonania zapytań SQL przez przygotowane bazy danych działające w usłudze chmurowej. Przygotowana baza danych Northwind działająca na SQL Server została przeniesiona na systemy Oracle i PostgreSQL, a następnie trzy wymienione bazy zostały umieszczone w chmurze Microsoft Azure. Proces badawczy został podzielony na trzy etapy. W pierwszej kolejności analizie podlegają zapytania typu Select. W drugim etapie analiza skupia się na kwestii wydajności zapytań SQL DDL, a końcowy etap opisuje analizę zapytań SQL DML.

Słowa kluczowe: usługi chmurowe; baza danych; analiza wydajnościowa

*Autor do korespondencji.

Adres e-mail: rafal.gozdz@pollub.edu.pl

Analysis of the possibility of managing the database stored in the cloud

Rafał Góźdź*, Maria Skublewska-Paszkowska

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

Abstract. The article presents a comparative analysis of the performance of selected database systems. The time of execution of SQL statements has been compared by prepared database service which are running in the cloud. Prepared Northwind database running on SQL Server has been moved to systems Oracle and PostgreSQL, and then three of these bases have been placed in the cloud Microsoft Azure. The research process was divided into three stages. Firstly, queries with Select are analyzed. In the second stage, the analysis focuses on the performance of SQL DDL, and the final stage of the analysis describes the SQL DML.

Keywords: cloud computing; database; performance analysis

*Corresponding author.

E-mail address: rafal.gozdz @pollub.edu.pl

1. Wstęp

Popularność usług chmurowych w ciągu ostatnich lat znacząco wzrosła [1]. Obecnie rynek informatyczny oferuje szereg zróżnicowanych usług chmurowych, które dają możliwość wykonywania podstawowych operacji takich jak: przechowywanie danych, tworzenie kopii zapasowych i innych o określonej specjalizacji. Pojęcie chmury w tematyce IT można rozumieć na kilka sposobów: jako chmura obliczeniowa (ang. Cloud computing) oraz jako dysk do przechowywania danych z możliwością przeglądania za pomocą urządzeń z dostępem do sieci. Funkcjonuje ona jako platforma, na której działają aplikacje, systemy informatyczne, maszyny wirtualne oraz systemy bazodanowe. Dostęp odbywa się za pomocą komputera lub urządzenia mobilnego z dowolnego miejsca, a żądania są przetwarzane po stronie dostawcy usługi.

Rosnąca popularność rozwiązań chmurowych wynika z możliwości dostępu do plików i usług z dowolnego komputera na świecie bez konieczności posiadania dodatkowych urządzeń. Kolejną zaletą jest możliwość wykupienia zasobu obliczeniowego tylko na okres, w którym będzie on potrzebny. Jest to atrakcyjne dla firm, które nie chcą inwestować w drogi sprzęt oraz infrastrukturę, a wolą skoncentrować się na rozwoju biznesu [3].

2. Charakterystyka chmury obliczeniowej

Chmura obliczeniowa stanowi pojęcie o wielu znaczeniach, dla jednych jest to dostarczenie usług obliczeniowych przez sieć, dla innych jest to inna definicja Internetu. Według NIST (National Institute of Standards and Technology) chmura obliczeniowa jest modelem umożliwiającym powszechny i wygodny dostęp do wspólnej puli konfiguracji zasobów obliczeniowych za pośrednictwem sieci. Przykładem takich zasobów są: sieci, serwery, pamięci masowe, aplikacje oraz usługi, które mogą być uruchomione w sposób zabezpieczony i z minimalnym obciążeniem po stronie usługodawcy [1].

Model chmury posiada pięć zasadniczych cech [2]:

- Samoobsługa na żądanie – klient ma możliwość jednostronnego zarządzania parametrami usług takimi jak: czas serwera, sieciowa pamięć masowa.
- Szeroki dostęp do sieci – możliwość dostępu do sieci poprzez standardowe mechanizmy (np. telefony komórkowe, tablety, laptopy i stacje robocze).
- Łączenie zasobów – zasoby usługodawcy są łączone, aby służyć wielu konsumentom. Istnieje możliwość dynamicznego przypisywania różnych zasobów fizycznych i wirtualnych w zależności od potrzeb klienta.

- Wysoka elastyczność – szybka skalowalność, dostosowywanie możliwości zasobów w dowolnym momencie może być dla klienta wręcz nieograniczone.
- Miarowość usług – systemy chmurowe automatycznie sterują i optymalizują wykorzystanie zasobów na pewnym poziomie abstrakcji w odpowiedni sposób dla rodzaju usługi (np. przechowywanie i przetwarzanie danych, przepustowość). Wykorzystanie zasobów jest monitorowane, aby zapewnić kontrolę i przejrzystość zarówno dla dostawcy jak i klientów usług.

Usługi chmurowe można podzielić ze względu na model świadczonych usług [3]:

- Software as Service (Saas) – oprogramowanie jest świadczone jako usługa, której zarządzanie odbywa się po stronie dostawcy usługi. Jest to najbardziej popularna metoda świadczenia usług chmurowych po stronie klienta. Takie usługi przyczyniają się do zmniejszenia kosztów eksploatacji oprogramowania poprzez pominięcie procesu instalacji i potrzeby aktualizacji. Przykładami takich aplikacji jest: DropBox, Google Drive do przechowywania danych oraz aplikacje biurowe takie jak: Office online, czy Google Docs.
- Platform as a Service (PaaS) – platforma stanowi usługę, na której oprogramowanie jest tworzone i wdrażane. Dostawca platformy udostępnia serwer wraz z systemem operacyjnym oraz zapewnia wysoką jakość infrastruktury sieciowej. Przykładami usługi PaaS są: Microsoft Azure, Google Aps i Heroku.
- Infrastructure as a Service (IaaS) – ten rodzaj usługi składa się z zautomatyzowanych i wysoce skalowalnych jednostek obliczeniowych, przy użyciu których konsument jest w stanie wdrożyć i uruchomić dowolne oprogramowanie. Użytkownicy mają większą kontrolę nad serwerem niż w przypadku modelu PaaS. Poza kontrolą nad systemem operacyjnym użytkownik ma możliwość zarządzania całym serwerem i pamięcią masową. Jest to najbardziej elastyczny model chmury obliczeniowej ze względu na wysoką skalowalność, możliwość automatycznego wdrażania serwerów, dostęp do mocy obliczeniowej, danych i infrastruktury sieciowej. Przykładami modelu IaaS są dostawcy usług: HPCloud i CloudSigma.

3. Bazy danych

3.1. PostgreSQL

PostgreSQL jest obiektowo–relacyjnym systemem zarządzania bazami danych opracowanym przez Uniwersytet Kalifornijski w Berkeley [4]. Jest jednym z najpopularniejszych systemów bazodanowych ze względu na wieloplatformowość oraz darmową licencję open source. System pozwala na tworzenie obszernych baz danych, przetwarzanie żądań nadesłanych przez inne aplikacje. System może obsługiwać obciążenia pochodzące od małych aplikacji, jak i od rozbudowanych aplikacji internetowych obsługujących równocześnie wielu użytkowników. PosgreSQL działa na wielu systemach operacyjnych: Linux, Windows oraz macOS oraz wspiera najpopularniejsze architektury sprzętowe: x86, x86-64 i ARM. Baza danych Postgres implementuje standard SQL:2011, obsługuje procedury składowane w języku PL/pgSQL, indeksy, mechanizm wyzwalaczy, czy definiowanie reguł.

PostgreSQL jest oparty na architekturze klient–serwer. Sesja składa się z trzech współpracujących procesów [4]:

- procesu administratora – postmaster,
- aplikacji użytkownika – frontend,
- serweru bazy danych –backend.

Proces administratora zarządza klastrem baz danych na komputerze docelowym. Aplikacja kliencka, np. pgAdmin, otrzymuje dostęp do bazy danych w klastrze poprzez zapytania do biblioteki libpq. Biblioteka przekazuje żądania za pomocą sieci do postmastera, który tworzy nowy proces serwera i łączy go z procesem klienta. Po tym zdarzeniu komunikacja klient–serwer odbywa się bez udziału procesu administratora. Posmaster jest procesem stale działającym i oczekującym na żądania. Inaczej jest z procesami frontend i backend, które mają określone cykle życia. Pojedynczy klient ma możliwość nawiązania wielu połączeń z procesem serwera, który z postmasterem działa na tym samym serwerze, podczas gdy klient może nawiązać połącznie z innego miejsca. Dostęp do bazy danych odbywa się za pomocą logowania użytkowników. Każdy użytkownik posiada swoją unikalna nazwę, identyfikator (usserid) oraz przypisane uprawnienia pozwalające na dostęp i modyfikację obiektów bazodanowych. W PostgreSQL zarządzanie uprawnieniami jest zrealizowane poprzez role. Użytkownik uprzywilejowany ze wszystkimi uprawnieniami to postgres.

3.2. Oracle

Oracle to obiektowo–relacyjny system do zarządzania bazami danych opracowany przez firmę o takiej samej nazwie. Do obsługi bazy danych zastosowano standardowy język SQL oraz do tworzenia procedur składowanych język PL/SQL i Java (od wersji 8). Przechowywanie danych w systemie Oracle jest oparte na wykorzystaniu pamięci współdzielonej dostępnej dla wszystkich użytkowników bazy danych określanej jako Globalny Obszar Współdzielony – SGA. Instancja Oracle składa się z zablokowanego globalnego obszaru współdzielonego SGA i procesów tła, których celem jest minimalizacja ruchu informacji przy zachowaniu bezpieczeństwa utrzymania danych. Polecenia SQL trafiają do odpowiedniego bufora SGA, po przetworzeniu i przeanalizowaniu bloki danych są pobierane do obszaru SGA, a następnie zwracane użytkownikowi. Jeżeli wystosowane polecenie SQL odwoła się do danych znajdujących się już w obszarze SGA to zostaje pominięty etap pobierania danych i wynik zostanie zwrócony szybciej.

Architektura oparta na klient–serwer w systemie Oracle pozwala na równoczesny dostęp użytkowników do tej samej bazy danych, operacje odczytów danych przez wielu użytkowników nie powodują konfliktów. Podczas operacji modyfikacji mogą występować konflikty i niespójności, dlatego wprowadzono mechanizm transakcji. W przypadku konfliktu system bazodanowy szereguje operacje transakcji tak, aby nie powstawały konflikty. Użytkownik logujący się do systemu zarządzania bazy danych rozpoczyna tym samym

sesje i kończy w momencie zamknięcia systemu. W danej sesji może być realizowanych wiele transakcji, jedna po drugiej opisywanych za pomocą polecenia SQL. Użytkownik tworząc obiekty bazodanowe staje się ich właścicielem. Miejscem docelowym obiektów jest schemat użytkownika, który jest tworzony automatycznie podczas dodawania nowego użytkownika, posiadający unikalną nazwę oraz będącym logiczną przestrzenią bazy danych [5]. W systemie Oracle istnieje wiele obiektów służących do przechowywania danych oraz do wspomagania zarządzania utworzonych danych. Tworzone obiekty przez użytkowników wymagają posiadania przez nich określonych uprawnień. System Oracle obsługuje obiekty takie jak: tabele, indeksy, klastry, widoki, sekwencje, wyzwalacze oraz pakiety, procedury i funkcje.

3.3. SQL Server

Microsoft SQL Server (MS SQL) to system zarządzania bazą danych opracowany i rozwijany przez firmę Microsoft. Jest to platforma typu klient–serwer, w której jedna instancja serwera przechowuje wiele baz danych. Bazy danych można podzielić na systemowe i użytkownika [6]. Wielu użytkowników systemu może posiadać uprawnienia do jednej bazy danych, referencja do takiego obiektu zapisana będzie w postaci: BazaDanych.uzytkownik.obiekt.

Domyślnym schematem i jednocześnie użytkownikiem posiadającym pełne uprawnienia do bazy danych jest dbo. Dane znajdujące się w bazie są przechowywane w postaci plików, które są odpowiednikiem fizycznych plików na dysku. Podczas tworzenia bazy danych tworzony jest plik główny dla danych będący miejscem startowym bazy danych o rozszerzeniu .mdf oraz pozostałe pliki nazywające się seconderyfile i mające rozszerzenie .ndf. Trzecim typem plików są pliki dziennika transakcji zawierające informacje niezbędne do odtworzenia stanu bazy danych po awarii, mające rozszerzeniem .ldf i nienależące do żadnej grupy plików. Baza może składać się z wielu plików danych i dziennika transakcji, które są zarządzane w inny sposób niż pliki danych. Utworzone pliki są wykorzystywane tylko przez jedną bazę danych. Takie rozwiązanie umożliwia umieszczenie plików na wielu dyskach fizycznych, poprawiając w ten sposób wydajność serwera. Referencje do wszystkich plików w bazie są składowane w pliku głównym (.mdf). System odczytuje plik główny bazy danych w sytuacji aktualizacji lub odtwarzania stanu bazy danych, a dostęp odbywa się za pomocą odwołania do nazwy logicznej, jak i fizycznej. Nazwę logiczną używa się w instrukcjach T-SQL, przy czym nazwa i identyfikator muszą być unikalne w całej bazie. Nazwa fizyczna pliku jest to nazwa pliku w systemie operacyjnym z rozszerzeniem mdf. Pliki bazy danych mdf oraz ndf składają się z ponumerowanych stron, gdzie numeracja rozpoczyna się od 0. Pierwsza strona plików zawiera nagłówki informujące o parametrach pliku. Każdy plik ma swój unikalny identyfikator. Odwołując się do konkretnej strony w pliku należy podać identyfikator pliku oraz numer strony. Rozmiary opisywanych plików rosną samoczynnie, istnieje możliwość określenie przyrostu rozmiaru pliku po jego zapełnieniu. Jeżeli plik należy do grupy plików, przyrost rozmiaru pliku może nastąpić jedynie po zapełnieniu wszystkich plików w grupie. Pliki bazy danych należące do grupy plików tworzą tzw. przestrzeń nazw znaną z innych systemów baz danych np. Oracle. Obiekty bazodanowe rozmieszczane są w utworzonych grupach plików, które można podzielić na typy [6]:

- główny – obejmuje główny plik bazy danych i wszystkie inne pliki nieprzypisane do innej grupy plików,
- zdefiniowane przez użytkownika.

Istnieje również domyślna grupa plików, do której serwer przypisuje pliki, które nie mają określonej grupy plików.

Na serwerze SQL Server znajduje się baza danych tempdb, która jest buforową bazą, przechowywującą tymczasowo dane z aktualnie przetwarzanych transakcji, w której wyniki zapytań są obrabiane pod zadanymi kryteriami przed wysłaniem odpowiedzi. Zalecane jest umieszczenie bazy tempdb na innym dysku fizycznym celem zwiększenia wydajności serwera [6].

4. Metoda badawcza

Do przeprowadzenie badań użyto relacyjną bazę danych Northwind przeznaczoną na platformę SQL Server. Baza zawiera przykładowe dane na temat sprzedaży określonych produktów dostarczonych przez odpowiednich dostawców. W celu zaimportowania bazy danych należy przenieść plik northwind.mdf do katalogu \Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA. Następnie w środowisku SQL Server Management Studio z menu kontekstowego Databases należy wybrać Attach Databases. Importowanie bazy odbywa się za pomocą konsoli z użyciem pliku o rozszerzeniu sql. Po utworzeniu pustej bazy należy wpisać przykładowe polecenie: sqlcmd -S host -d Northwind -i sciezka_do_bazy. Wykonanie komendy spowoduje utworzenie kompletnego obiektu bazodanowego, a w sekcji Object Explorer pojawi się struktura zaimportowanej bazy Northwind w postaci drzewa katalogów [6]. Kolejnym etapem jest przeniesienie bazy danych Northwind na systemy Oracle i PostgreSQL. Na początku należy utworzyć pustą strukturę danych oddzielnie na każdym z systemów. Podczas tworzenia należy wziąć pod uwagę, że nie każdy typ danych występuje na wszystkich trzech systemach baz danych. W takim wypadku należy wybrać typ i jego rozmiar, który będzie kompatybilny z typem bazy SQL Server. Następnie należy przeprowadzić proces migracji danych, który określa sposób przeniesienia danych z miejsca źródłowego do miejsca docelowego. Dla realizacji celów pracy dane zostały przeniesione z tabel bazy źródłowej w systemie SQL Server do systemów Oracle i PostgreSQL.

Do migracji danych na gotowych strukturach baz danych wykorzystano Pentaho Data Integration w wersji próbnej [7]. W pierwszej fazie migracji należy utworzyć trzy połącznia do baz danych: jedno do bazy źródłowej oraz dwa do baz, do których dane będą przenoszone W kolejnym etapie należy utworzyć indeksy, które w każdym systemie baz danych są tworzone w podobny sposób. Po utworzeniu gotowych skryptów i wykonaniu ich za pośrednictwem narzędzi PgAdmin i SQL developer połączonych z serwerami bazodanowymi, zostaną utworzone puste struktury bazy danych Northwind na systemach PostgreSQL i Oracle.

Kolejną czynnością jest zdefiniowanie zbioru wejściowego dla określonej tabeli, poprzez dodanie z zakładki Input elmentu Table Input. W ustawieniach należy wybrać połączenie z źródłową bazą danych MS SQL oraz wybrać dane z określonej tabeli za pomocą polecenia SQL. Alternatywnie można wybrać opcje Get SQL Statement, które automatycznie wprowadzi w pole tekstowe polecenia wybierające wszystkie rekordy z tabeli. Wybrane dane będą importowane do bazy docelowej, dlatego należy zwrócić uwagę, czy wszystkie typy i formaty danych są kompatybilne. Po ustawieniu Table Input dla określonej tabeli należy dołączyć do niej obiekt, do którego dane zostaną zaimportowane. W tym przypadku będzie to także tabela, ale innej bazy danych (Oracle albo PostgreSQL), dlatego zostaje dołączony Table output. W ustawieniach należy wybrać połączenie bazy, do której dane będą importowane oraz tabele. Istnieje także możliwość ustawienia mapowania metadanych kolumn. W tym przypadku mapowanie odbywa się automatycznie ze względu na spójność nazw. Po utworzeniu takiego przepływu należy powtórzyć operację dla każdej tabeli w bazie danych, a następnie uruchomić proces migracji danych. Należy pamiętać, że w czasie tego procesu tabele nie mogą być powiązane ze sobą, ponieważ może wystąpić wyjątek naruszenia unikalności klucza obcego. Po sfinalizowaniu procesu tabele obu baz danych powinny zawierać identyczną ilość danych. Następnie należy dokonać konfiguracji platformy Azure. Użyte konto jest w wersji próbnej 30 dniowej, w której użytkownik ma do wykorzystania 170 euro na zasoby obliczeniowe.

Do umieszczenia baz danych w systemach MS SQL, PostgreSQL i Oracle użyto odpowiednio:

- Azure SQL Server,
- Oracle Database 12.1.0.2 Standard Edition,
- PostgreSQL 9.5 działającym na Ubuntu 14.04.

Azure SQL Database jest wbudowaną usługą relacyjnych baz danych w chmurze Azure opartej na Microsoft SQL Server. Usługa SQL w chmurze współpracuje z narzędziami, bibliotekami i API SQL Server umożliwiając w ten sposób prostsze przenoszenie rozwiązań do chmury. Przykładem jest możliwość przeniesienia bazy danych za pomocą narzędzia Management Studio do SQL Azure przy użyciu opcji Deploy Database to SQL Azure [8].

Do utworzenia pozostałych systemów baz danych w usłudze Azure użyto maszyn wirtualnych będących rozwiązaniem typowo infrastrukturalnym opartym na modelu IaaS, w którym aplikacje, dane, środowisko jak i system są zarządzane przez użytkownika. Microsoft zapewnia różne warianty platformy sprzętowej takich jak. moc obliczeniowa CPU, ilość pamięci RAM oraz przestrzeń dyskowa, które są wybierane przez klienta podczas tworzenia maszyny wirtualnej. Płatność odbywa się za czas rezerwacji zasobów sprzętowych od momentu uruchomienia maszyn wirtualny z systemami Windows oraz Linux.

Dla potrzeb badań utworzono dwie maszyny wirtualne [9]:

- Standardowa DS12 v2 (4 rdzenie, 28 GB pamięci) z systemem operacyjnym Ubuntu oraz zainstalowanym PostgreSQL 9.5,
- Standardowa DS12 v2 (4 rdzenie, 28 GB pamięci) opartym na systemie Linux z zaimplementowanym Oracle Database 12.1.0.2 Standard Edition.

5. Badania

Badania zostały przeprowadzone dla trzech baz danych: SQL Server, Oracle i PostgreSQL, działających w ramach usługi Microsoft Azure. Dane zostały przedstawione w formie tabel oraz wykresów słupkowych, mając na celu utrzymanie przejrzystości rezultatów [9]. Kryterium badawczym jest czas wykonania zapytań przedstawionych na listingach 1, 2 i 3 mierzony w milisekundach [ms] dla poszczególnych systemów baz danych: SQL Server, PostgreSQL i Oracle. Wykonywanie zapytań zostało powtórzone 10-krotnie z wyłączoną funkcją cachowania, aby wyniki były miarodajne. Po wykonaniu zapytań, które dokonały modyfikacji struktury bazy danych, konieczne było przywrócenie jej w procesie backupu. Przykład 1 zawiera zapytania dla kategorii DML.

Przykład 1. Zapytania Select

```
--1. Skrytp tworzący widok zamówień wszystkich klientów.
Create View VCustomersOrders AS
Select c.CustomerID , c.CompanyName, c.ContactName,
c.Address, c.City, c.PostalCode, c.Country, o.OrderID,
o.OrderDate, o.ShippedDate, o.ShipAddress,o.ShipCity,
o.ShipCountry
FROM [Customers]  c Left JOIN [Orders] o ON
c.CustomerID=o.CustomerID;

--2. Wybranie danych o kliencie oraz ilości jego zamówień
Select c.CustomerID, c.CompanyName ,c.Address, c.City,
c.Country, Count(o.OrderID) AS "Order quantity"FROM
[Customers]  c Left JOIN [Orders] o ON
c.CustomerID=o.CustomerID GROUP BY c.CustomerID,
c.CompanyName ,c.Address, c.City, c.Country;


--3. Wybranie dane o produkcie i ilości jego zamówień w
poszczególnych transakcjach
Select p.ProductID, p.ProductName, o.OrderID, o.Quantity
FROM [Products] p LEFT JOIN [Order Details] o ON
p.ProductID=o.ProductID Order by p.ProductID, Quantity;

--4. Wybranie danych o dostawcach i ich klientach
Select c.CustomerID, c.CompanyName , s.SupplierID,
s.CompanyName From Customers c Left Join Orders o  ON
c.CustomerID=o.CustomerID Left Join [Order Details] od ON
o.OrderID=od.OrderID Left Join Products p ON od.ProductID
=p.ProductID Left Join Suppliers s ON
p.SupplierID=s.SupplierID ORDER BY 1,3;


--5. Zapytanie wybierające dane o ilości zamówień
zrealizowanych w poszczególnych regionach przez
pracowników.
Create View vOrderEmployees AS
Select e.EmployeeID,e.LastName, e.FirstName,
Count(o.OrderID) AS 'Orders quantity',
t.TerritoryDescription,r.RegionDescription FROM Orders o
Left Join Employees e ON o.EmployeeID=e.EmployeeID
Left Join EmployeeTerritories et on
e.EmployeeID=et.EmployeeID
Left Join Territories t On et.TerritoryID=t.TerritoryID
Left Join Region r ON t.RegionID=r.RegionID
```

```
Group BY e.EmployeeID, e.LastName, e.FirstName,
t.TerritoryDescription, r.RegionDescription;
```

W przykładzie 2 zebrano zapytania z kategorii SQL DDL.

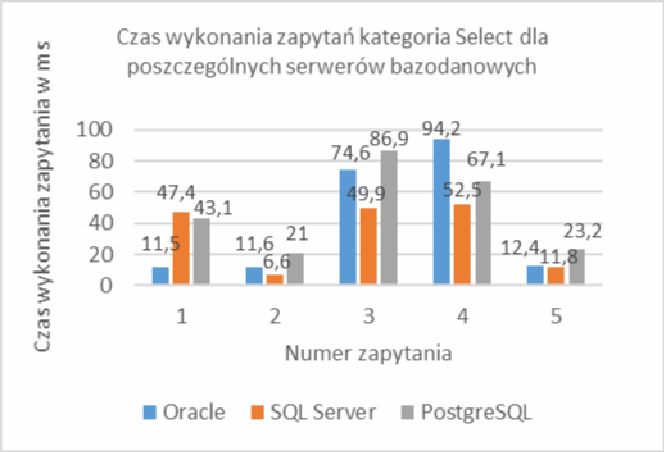
Przykład 2. Zapytania SQL DDL

```
--6. Skrypt dodający do tabeli Employee kolumny idRating
jako powiązanie z tabelą EmployeeRating
Alter Table Employees add RatingID INT;

-- 7. Skrypt dodający tabele Rating
Create Table Rating (
                        RatingID INt PRIMARY KEY NOT
NULL,
                        RatingNumber samllint,
                        ratingDescribtion Varchar(255),
                        superEmployee boolean
                              );


Alter Table Employees add Constraint RatingIDFK FOREIGN
KEY(RatingID) REFERENCES Rating(RatingID);

--8. Skrypt usuwający kolumnę discount z tabeli OrderDetails
Alter table [Order Details] DROP Constraint
DF_Order_Details_Discount
Alter table [Order Details] DROP Constraint CK_Discount
Alter Table [Order Details] DROP column Discount

-- 9.Skrypt usuwający tabele Contacts
Drop table  [Contacts]
```

W Listingu 3. zebrano zapytania z zapytania łączone z poprzednich listingów, które przedstawiono w kolejnej części artykułu.

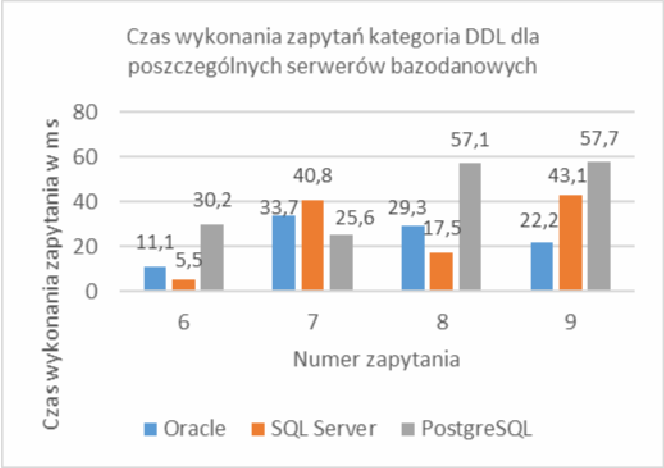
Przykład 3. Zapytania SQL DML

```
--10.  Skrypt usuwający rekordy o szczegółach zamówień
wybranego produktu Delete FROM [northwind].[dbo].[Order
Details] od
inner join [northwind].[dbo].Products p ON
od.ProductID=p.ProductID --Order by ProductName
 where p.ProductName='Gnocchi di nonna Alice'

--11.  Skrypt aktualizujący dane o zniżkach produktów
Update Products set Discontinued=1 where UnitPrice>25;
Update [Order Details] set Discount =0.50 where UnitPrice >
25;
```



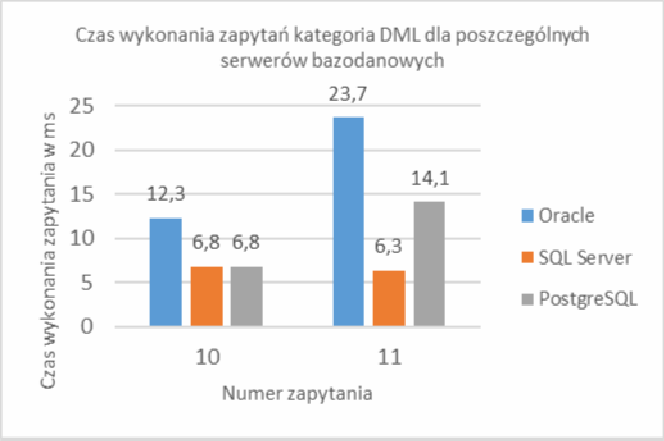
Rys. 1. Wykres czasów wykonania zapytań Select na wybranych serwerach baz danych

Rysunek 1 przedstawia wykres z rezultatami czasów wykonania zapytań Select. Zapytanie numer 1 zostało wykonane w podobnym czasie przez SQL Server i PostgreSQL(ponad 43 ms), jednak najkrótszy czas odnotowuje się dla bazy Oracle i wynosi 11,5 ms będący wynikiem 4 razy mniejszym od pozostałych baz danych. Drugie zapytanie z tej kategorii zostało najszybciej wykonane przez bazę Microsoftu w czasie 6,6 ms, prawie 2 krotnie szybciej niż Oracle i 3 krotnie szybciej niż baza PostgreSQL. Zapytanie 3 i 4 SQL Server ponownie wykonał najszybciej w czasie ok. 50 ms, przy czym w zapytaniu 3 PostgreSQL okazał się najwolniejszy (o 74 %), natomiast w zapytaniu 4 Oracle miał znacznie odbiegający czas wykonania wynoszący 94,2 ms stanowiący wynik o 81 % wolniejszy niż SQL Server i o 41 % wolniejszy niż baza PostgreSQL. Ostatnie zapytanie z tej kategorii zostało wykonane najszybciej przez SQL Server (11,8 ms) i Oracle (12,4 ms) co stanowi wynik o połowę mniejszy niż w przypadku PostgreSQL.



Rys. 2. Wykres czasów wykonania zapytań DDL na wybranych serwerach baz danych

Z wykresu na rysunku 2 można zaobserwować rozbieżność w czasach wykonaniu poszczególnych zapytań DDL dla różnych systemów baz danych. W przypadku zapytania 6 SQL Server wykonał je w czasie 5,5 ms, tj. ponad 5 krotnie szybciej niż PostgreSQL i 2 krotnie szybciej niż Oracle. Czasy wykonania zapytania numer 7 są już miej rozbieżne od poprzedniego. Baza danych PostgeSQL wykonała zapytanie najszybciej, w czasie 25 ms, czyli o 40 % krótszym niż SQL Server i o 24% krótszym niż Oracle. W kolejnym zapytaniu można znowu zaobserwować dużą rozbieżność w czasie wykonania zapytania. Baza SQL Server wykonała je najszybciej, w czasie 17,5 ms – ponad 3 krotnie szybciej niż PostgreSQL i o 41 % szybciej niż Oracle. Zapytanie 9 zostało najszybciej zrealizowane przez bazę Oracle z czasem 22 ms – 2 krotnie szybciej niż SQL Server i 2,5 razy szybciej niż PostgreSQL.



Rys. 3. Wykres czasów wykonania zapytań DML na wybranych serwerach baz danych

Rysunek 3 przedstawia wykres z czasami wykonania zapytań dla obu kategorii DDL i DML. Zapytanie numer 10 zostało wykonane najszybciej przez dwie bazy danych: SQL Server i PostgreSQL w jednakowym czasie wynoszącym 6,8 ms, który jest o 44 % krótszy niż wynik pochodzący z bazy Oracle. W drugim zapytaniu z kategorii DML odnotowuje się kolejne rozbieżności w wynikach wykonania zapytania. Baza Microsoftu ponownie wykonała zapytanie najszybciej z czasem 6,3 ms, baza PostgreSQL wykonała je w czasie ponad 2 krotnie dłuższym, a baza Oracle w czasie ponad 3 krotnie dłuższym.

Analizując wyniki zgromadzone na wykresach można zaobserwować otrzymywanie mało–odbiegających wartości czasów dla bazy danych PostgreSQL. W przypadku bazy Oracle różnice pomiędzy czasami są nieznacznie większe podczas, gdy w bazie SQL Server te wahania są dużo większe.

6. Wnioski

Usługi chmurowe pozwalają w łatwy i szybki sposób konfigurować i uruchamiać serwery baz danych. Dostępny wybór zasobów obliczeniowych jest dostosowany do indywidulanych klientów jak i dużych firm informatycznych, które coraz częściej potrzebują ogromnej mocy obliczeniowej. Duże zainteresowanie wykorzystaniem chmury obliczeniowej powoduje ciągły jej rozwój i zwiększenie zapotrzebowania ze strony firm informatycznych, które coraz częściej inwestują w tę technologię. Na podstawie przeprowadzonych badań można stwierdzić, że baza danych SQL Server wspierana technologią Azure, pochodząca od tego samego producenta, osiągnęła najlepsze rezultaty. Czasy wykonywania wszystkich zapytań SQL DML i części DDL na bazie danych Microsoftu były nawet kilkukrotnie krótsze niż konkurencyjne bazy danych. We wszystkich badanych kategoriach zapytań występują rozbieżności w czasach ich wykonania. Największą rozbieżność odnotowano w zapytaniu nr 6, w którym najkrótszy czas wynosi 5,5 ms dla SQL Server, a najdłuższy 30,2 ms. dla PostgreSQL. Najmniejszą rozbieżność wyników odnotowano w zapytaniu nr 7 i waha się ona w granicach od 25,6 ms dla PostgreSQL do 40,8 ms dla SQL Server. Powodem wyżej przedstawionych wyników jest wsparcie technologii Microsoftu oraz lepsza optymalizacja usługi niż w przypadku tradycyjnej konfiguracji serwera bazodanowego na systemie Linux. Jednakże w zapytaniu dotyczącym tworzenia nowej tabeli oraz zapytaniu dotyczącym usuwania wybranych rekordów baza danych PostgreSQL uzyskuje najlepsze rezultaty w stosunku do pozostałych opisywanych powyżej baz danych. Bazy danych Oracle i PostgreSQL mimo, że zostały skonfigurowane na maszynach wirtualnych o identycznych parametrach osiągały różne czasy wykonania tych samych zapytań. Uzyskane wyniki mogą ułatwić wybór systemu bazodanowego w zależności od struktury bazy oraz od tego jakie operacje będą na niej wykonywane.

Literatura

[1] Peter Mell, Timothy Grance, The NIST Definition of Cloud Computing, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, [11.09.2016]

[2] Daniel J. Abadi, Data Management in the Cloud: Limitations and Opportunitie, strony 2-5 , Yale University 2009.

[3] Ben Kepes, Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS, https://support.rackspace.com/white-paper/understanding-the-cloud-computing-stack-saas-paas-iaas/, [11.09.2016]

[4] Scott Zhang, Install and configure PostgreSQL on Azure, https://azure.microsoft.com/pl-pl/documentation/articles/virtual-machines-linux-postgresql-install/, [11.09.2016]

[5] Andrzej Klusiewicz, Kurs Oracle, http://andrzejklusiewicz.blogspot.com/2010/11/kurs-oracle-sql-podstawowe-definicje.html, [11.09.2016]

[6] Lech Banchowski, Realizacja SZBD w SQL Server, http://edu.pjwstk.edu.pl/wyklady/szb/scb/wyklad15/w15.htm, [11.09.2016]

[7] Pentaho Data Integration manual, http://www.pentaho.pl/kettle-etl.html. [11.09.2016]

[8] T. Redkar, T. Guidici, Platforma Windows Azure, Helion 2013.

[9] Rob Boucher, Introducing Microsoft Azure, https://azure .microsoft.com/en-us/documentation/articles/ fundamentals-introduction-to-azure, [11.09.2016]

JCSI 2 (2016) 133-139
Journal of Computer Sciences Institute

Wysłane: 2016-11-21
Przyjęte: 2016-11-30

Porównanie możliwości wykorzystania oraz analiza wydajności baz danych na systemach mobilnych

Mateusz Grudzień*, Konrad Korgol*, Dariusz Gutek
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia wybrane formy składowania danych tj. Local Storage, Shared Preferences, pliki płaskie oraz SQLite w kontekście dwóch systemów mobilnych – Android oraz Windows Mobile. Opisuje on również sposoby, poprzez które możliwe jest połączenie z zewnętrznymi systemami bazodanowymi takimi jak Microsoft SQL Server, PostgreSQL czy MySQL i stara się odpowiedzieć na pytanie, użycie której z tych opcji ma największy sens w poszczególnych przypadkach. Całość argumentuje badaniami wydajności, którym poddane zostały wszystkie z wymienionych opcji.

Słowa kluczowe: Android; Windows Mobile; baza danych; urządzenie mobilne

*Autorzy do korespondencji.

Adresy e-mail: mateusz.grudzien@protonmail.com, konrad.korgol@gmail.com

Comparison of the possible uses and performance analysis of databases on mobile operating systems

Mateusz Grudzień*, Konrad Korgol*, Dariusz Gutek
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Abstract. This publication presents chosen forms of data persistence such as: Local Storage, Shared Preferences, flat files and SQLite in the context of two widely used mobile operating systems – Android and Windows Mobile. It also describes ways to connect to external database engines such as Microsoft SQL Server, PostgreSQL or MySQL and tries to answer the question which one of these data persistence forms makes the most sense and when. The arguments are based on performance tests that all of the described solutions were participants of.

Keywords: Android; Windows Mobile; database; mobile device

*Corresponding authors.

E-mail addresses: mateusz.grudzien@protonmail.com, konrad.korgol@gmail.com

1. Wstęp

Obecnie obserwowane tempo rozwoju technologii niejednego świadomego obserwatora przyprawić może o zawrót głowy. Istnieje wiele przykładów gdzie to, co w połowie ubiegłego stulecia składało się na dobry film science-fiction, dziś dla większości ludzi jest częścią ich codziennego życia. Tak szybki postęp dokonuje się praktycznie we wszystkich gałęziach świata technologii a niejednokrotnie rozwój w jednej dziedzinie jest zalążkiem dla powstania całkowicie nowej gałęzi. Sytuacja taka miała miejsce w przypadku postępującej miniaturyzacji, która najpierw dała narodziny przenośnym telefonom komórkowym, które kilkanaście lat później zostały z kolei zastąpione przez tzw. smartfony. Urządzenie, które prawie każdy z nas nosi dziś w kieszeni, za pomocą internetu i szybkiej transmisji danych, daje swojemu posiadaczowi możliwość kontaktu i wymiany wiedzy oraz poglądów z osobami znajdującymi się na drugim końcu globu, wykonanie i przesłanie wysokiej jakości zdjęcia czy nagrania wideo. Przede wszystkich jednak daje ono dostęp do ogromnych zasobów informacji, które trzeba nie tylko składować, ale również odpowiednio przetworzyć.

Jednym ze skutków spowodowanych rozwojem cyfrowego świata jest właśnie rosnąca ilość wytwarzanych danych. W 2010 roku, zarządzający wówczas Google, Eric Schmidt ogłosił że wg szacunków prowadzonych przez kierowaną przez niego firmę, od zarania dziejów do roku 2003, ludzkość wytworzyła 5 eksabajtów danych. W momencie, w którym Schmidt podzielił się tym wynikiem – w roku 2010 – czas potrzebny naszej cywilizacji na wytworzenie tej samej ilości danych wynosił już jedynie 2 dni. Chociaż od tamtego czasu nikt nie pokusił się o odnowienie tych szacunków, spodziewać się można, że ilość danych jakie przetwarzamy jedynie wzrosła, a trend wydaje się wcale nie wykazywać oznak spowolnienia.

Zestawiając ze sobą oba te fakty, tj. rozwój technologii mobilnych oraz gwałtownie rosnącą ilość wytwarzanych danych, pytanie jakie automatycznie nasuwa się na myśl brzmi: jak dobrze dostosowane do przetwarzania takiej ilości danych są dostępne obecnie urządzenia mobilne. Czy sposoby składowania danych dostępne na nowoczesnych mobilnych systemach operacyjnych oferują przystępne czasy dostępu do danych?

2. Opis obiektów badań

Systemy mobilne oferują mnogą liczbę sposobów przechowywania danych. SQLite, pliki płaskie czy zewnętrzne bazy danych są to formy występujące na wielu systemach mobilnych. Istnieją też takie formy składowania danych, które są dostępne tylko i wyłącznie na konkretnym systemie. Mowa tutaj o Shared Preferences, które obecne jest na Androidzie,

a także Local Settings istniejące na platformie Windows Mobile. Co prawda docelowy system obu form się różni, jednakże sposób, w jaki przechowują one dane, jest jednakowy i jest to para klucz-wartość[4]. Przechowywanie oraz dostęp do samych danych za pomocą tych form jest niewątpliwie najprostszy. Takie rozwiązanie świetnie sprawdzi się w przypadku przechowywania ustawień aplikacji czy kont użytkownika.

W przypadku, gdy aplikacja pobiera dużo danych z internetu, te w postaci zdjęć czy filmów warto umieszczać jako pliki płaskie w lokalnej pamięci urządzenia[3]. Dodatkowo w systemie Android takie pliki mogą być przechowywane w tzw. Internal Storage[14] oraz External Storage[13]. Rozwiązanie to zapewnia oszczędność transferu danych użytkownika, gdyż aplikacja, z której korzysta użytkownik, nie pobiera danych ponownie, lecz odwołuje się do tych wcześniej zapisanych. O ile przechowywanie takich danych w postaci plików płaskich nosi ze sobą wiele zalet, o tyle przechowywanie danych użytkowników w postaci rekordów danych nie ma większego sensu. Niesie to ze sobą wysokie koszty czasowe dostępu do danych oraz mało efektywne zarządzanie samymi danymi.

W tej sytuacji najlepszym rozwiązaniem jest lokalna baza danych SQLite. Przechowuje ona w postaci binarnej ustrukturyzowane dane aplikacji, które umieszczone są w tabelach i mogą posiadać między sobą relacje[2]. Do bazy dostęp ma tylko i wyłącznie aplikacja dla której baza została stworzona, zaś sam użytkownik może dotrzeć do bazy SQLite tylko i wyłącznie posiadając prawa administratora systemu[6].

W dobie wszechobecnego szybkiego dostępu do internetu należy zwrócić uwagę na bazy danych dostępne na zewnętrznych serwerach. MySQL, PostgreSQL czy Microsoft SQL Server to tylko kilka przykładów baz danych, które mogą zostać wdrożone na takim serwerze. Aplikacje korzystające z takiej formy składowania danych oferują użytkownikom dostęp do danych nawet w sytuacji, gdy urządzenie użytkownika zostanie zniszczone lub zostanie wykonana operacja czyszczenia danych w całym systemie. Wystarczy, że użytkownik zaloguje się na jego indywidualne konto w aplikacji i będzie on posiadał już wszystkie wprowadzone dane. Należy tutaj zwrócić uwagę na fakt, że o ile takie rozwiązanie niesie ze sobą tę nieocenioną zaletę, jak nieulotność danych, tak też posiada ono bardzo ważną wadę. W przypadku braku dostępu do internetu o korzystaniu z aplikacji można zapomnieć, a samo utrzymanie serwera ponosi ze sobą rozmaite koszty.

Każda przedstawiona forma przechowywania danych może być łatwo i skutecznie zaimplementowana w aplikacji, dzięki bogatej bibliotece odpowiednich klas w pakiecie SDK dla danego systemu mobilnego. W przypadku systemu Android i Shared Preferences jest to klasa SharedPreferences[10], dla SQLite są to klasy SQLiteOpenHelper[12], SQLiteDatabase[11], ContentValues oraz klasa Cursor[7]. Operacje na plikach mogą zostać łatwo wykonane poprzez klasy File, FileOutputStream[8], a także FileInputStream[9]. Zewnętrzne bazy danych najlepiej obsługiwać poprzez REST API znajdujące się na serwerze. Powinny być one również obsługiwane asynchroniczne dzięki czemu operacja pobierania i zapisu danych odbywa się w wątku działającym w tle, co eliminuje negatywny efekt braku odpowiedzi aplikacji na komendy użytkownika przez określoną ilość czasu potrzebną na wykonanie operacji. Komunikaty przychodzące oraz zwrotne warto parsować do popularnego formatu JSON, który zapewnia standaryzację komunikacji oraz łatwość manipulacji danymi[1]. Nie należy również zapominać o pisaniu optymalnego kodu aplikacji oraz optymalizacji zapytań do bazy danych[5].

3. Metoda badań

Badaniu poddane zostały wybrane formy przechowywania danych na dwóch mobilnych systemach operacyjnych. Opcja unikalna dla systemu Windows Mobile to Local Storage, które udostępnia API umożliwiające dostęp do pamięci wewnętrznej urządzenia oraz do kontenera Local Settings przechowującego pary klucz-wartość. Opcje unikalne dla systemu Android to odpowiednik Local Settings z systemu Windows Mobile czyli Shared Preferences oraz Internal i External Storage. Opcja poniekąd wspólna, poniekąd, bo na obu systemach różniąca się implementacją, to wewnętrzna baza danych SQLite. Wreszcie, na obu systemach, badaniu poddane zostały również zewnętrzne bazy danych takie jak Microsoft SQL Server, PostgreSQL oraz MySQL. Dostęp do tych silników bazodanowych zrealizowany został poprzez przygotowany w tym celu serwer PHP, który działał jako pośrednik wystawiając przygotowane na potrzeby badań REST API. Urządzenia mobilne łączyły się z serwerem poprzez lokalną sieć WiFi.

Wersje wykorzystanego oprogramowania to:

- Android 5.0.1,
- Windows Mobile 10.0.14393.448,
- SQLite dla WM10: v. 3.15.1,
- SQLite dla Androida: 3.8.4.3,
- Microsoft SQL Server 2016,
- PostgreSQL 9.3,
- MySQL 5.7.14.

Badania dla systemu Windows Mobile wykonywane były na urządzeniu Microsoft Lumia 640XL o specyfikacji:

- CPU: Qualcomm MSM8226 Snapdragon 400, 4x1.2 GHz Cortex-A7,
- pamięć: 8GB (brak informacji o szybkości pamięci),
- WiFi 802.11 b/g/n.

Badania dla systemu Android przeprowadzane były na telefonie Samsung Galaxy I9505. Specyfikacja urządzenia:

- CPU: Qualcomm Snapdragon 600, 4x1,9GHz,
- pamięć: 16GB, zapis: 22.95MB/s, odczyt 92.05MB/s,
- WiFi v802.11 a/b/g/n/ac 2,4GHz i 5GHz.

Serwer, na którym uruchomione były baz danych oraz który oferował dostęp do nich poprzez REST API:

- CPU: Intel Core I7-4800MQ 2.70GHz,
- pamięć: Kingston HyperX 16GB DDR3 PC1300,
- dysk twardy: SSD Kingston V300 R/W 450MB/s.

Zarówno w przypadku systemu Windows Mobile jak i Android przygotowane zostały specjalne aplikacje, które automatyzowały proces badawczy. Na podstawie danych wprowadzonych przez prowadzącego badania, takich jak: rozmiar danych, liczba rekordów, liczba powtórzeń badania

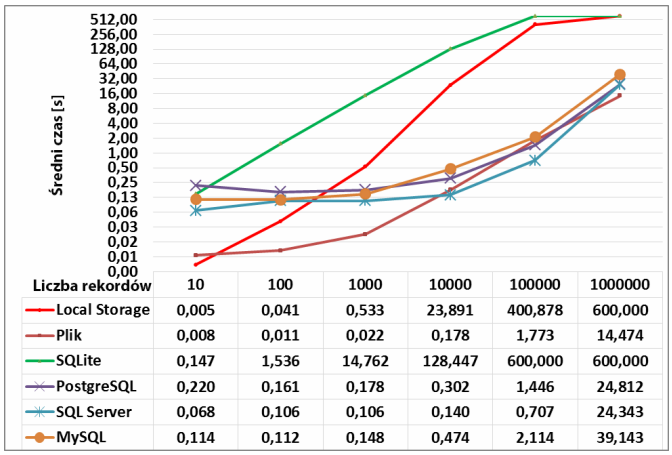
oraz wybrana opcja składowania danych do badania, aplikacja, w przypadku badania czasów zapisu, generowała zadaną ilość losowych danych, a następnie wykonywała żądanie zapisu. Czas w tym przypadku mierzony był od momentu wysłania żądania zapisu wygenerowanych już danych do momentu otrzymania odpowiedzi o sukcesie operacji. W przypadku żądania odczytu danych aplikacja liczyła czas operacji od momentu wysłania żądania o odczyt danych do momentu, gdy otrzymane dane zostały przetworzone do takiej samej formy w jakiej zostały one odesłane do zapisu. Jest to o tyle ważne, że np. w przypadku odczytu rekordów z danymi tekstowymi z pliku płaskiego, zawartość pliku musiała zostać poddana odpowiedniemu parsowaniu. Pomiar czasu w systemie Windows Mobile stosowany był poprzez klasę Stopwatch z pakietu System.Diagnostics, w systemie Android natomiast za pośrednictwem klasy Date i jej metody getTime().

Typy danych, które poddane zostały badaniu to: pary klucz-wartość, rekordy z danymi tekstowymi o długości 255 znaków, 100 znaków oraz 10 znaków, rekordy ze zmiennoprzecinkowymi danymi numerycznymi zapisywanymi na 32-bitach, rekordy z danymi binarnymi o rozmiarze 10kB, 1kB i 100B oraz rekordy zawierające różne typy danych. Rekord z typami mieszanymi składał się z liczby całkowitej zapisanej na 32 bitach, liczby zmiennoprzecinkowej o takim samym rozmiarze, daty, losowego ciągu o długości 255 znaków oraz danych binarnych o rozmiarze 100 bajtów.

4. Wyniki badań

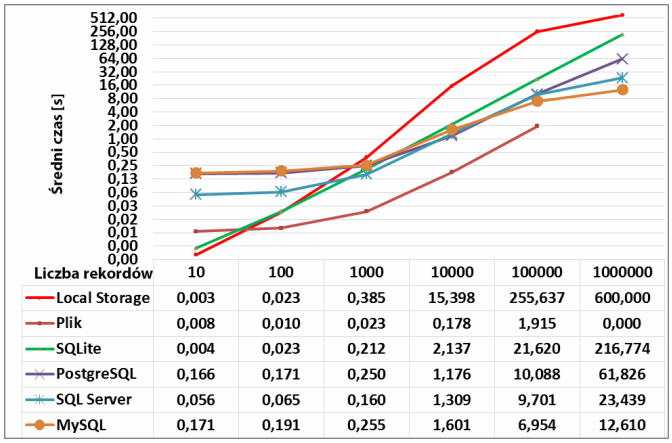
Poniżej przedstawione zostaną wyniki części przeprowadzonych badań. Wykresy widoczne na rys. 1 do rys. 20 przedstawiają wyniki kolejno badań zapisu i odczytu par klucz-wartość, rekordów zawierających dane tekstowe, rekordów zawierających dane będące liczbami zmiennoprzecinkowymi, rekordów zawierających dane binarne oraz rekordów, na które składały się dane o różnych typach. Dla każdej z tych opcji przedstawione zostały wyniki najpierw dla systemu Windows Mobile, a następnie Android. Do każdego wykresu dołączona została legenda, na której widoczne są osiągnięte średnie czasy dla każdej uczestniczącej w tym konkretnym badaniu opcji składowania danych. Dla przykładu: na rys. 1, na przecięciu kolumny oznaczonej wartością 1000 oraz wiersza oznaczonego wartością SQLite, odczytać można liczbę 14,762. Oznacza to, że zapis 1000 par klucz-wartość do bazy danych SQLite na systemie Windows Mobile zajął średnio 14,762 sekundy. Wartości te naniesione zostały na znajdujące się wyżej wykresy, gdzie wartości na pionowej osi oznaczają średni czas w sekundach, jaki upłynął do ukończenia badania. Na osi poziomej z kolei widoczna jest liczba rekordów, na których operowało to badanie lub w przypadku wykresów z rys.1 do rys. 4 jest to liczba zapisywanych bądź odczytywanych par klucz-wartość. W przypadku, gdy z jakiegoś powodu w danym scenariuszu badanie nie zostało ukończone (np. urządzenie, na którym przeprowadzano badanie posiadało zbyt małą ilość pamięci), wynik dla takiego przypadku oznaczany był jako 0 a linia wyników nie uwzględnia go na wykresie.

Rysunek 1 oraz rysunek 2 przedstawiają wyniki badań zapisu oraz odczytu par klucz-wartość na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Local Storage	0,005	0,041	0,533	23,891	400,878	600,000
Plik	0,008	0,011	0,022	0,178	1,773	14,474
SQLite	0,147	1,536	14,762	128,447	600,000	600,000
PostgreSQL	0,220	0,161	0,178	0,302	1,446	24,812
SQL Server	0,068	0,106	0,106	0,140	0,707	24,343
MySQL	0,114	0,112	0,148	0,474	2,114	39,143

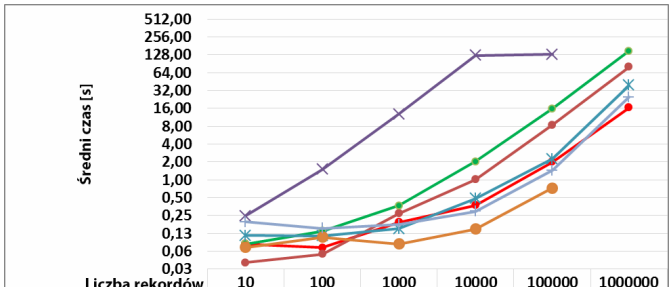
Rys. 1. Wyniki badań czasów zapisu par klucz-wartość na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Local Storage	0,003	0,023	0,385	15,398	255,637	600,000
Plik	0,008	0,010	0,023	0,178	1,915	0,000
SQLite	0,004	0,023	0,212	2,137	21,620	216,774
PostgreSQL	0,166	0,171	0,250	1,176	10,088	61,826
SQL Server	0,056	0,065	0,160	1,309	9,701	23,439
MySQL	0,171	0,191	0,255	1,601	6,954	12,610

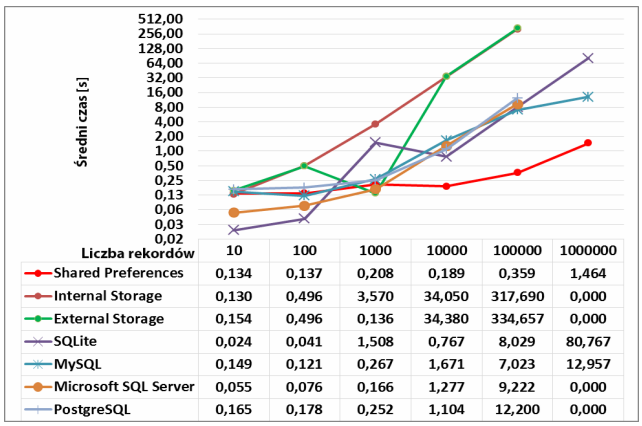
Rys. 2. Wyniki badań czasów odczytu par klucz-wartość na systemie Windows Mobile

Rysunek 3 i rysunek 4 przedstawiają wyniki dla tego samego badania, ale przeprowadzonego na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Shared Preferences	0,083	0,072	0,192	0,372	1,998	16,564
Internal Storage	0,040	0,056	0,274	1,011	8,426	80,325
External Storage	0,083	0,136	0,369	2,030	15,853	148,840
SQLite	0,248	1,508	12,893	125,018	129,736	0,000
MySQL	0,115	0,115	0,151	0,484	2,253	39,428
Microsoft SQL Server	0,073	0,109	0,084	0,148	0,728	0,000
PostgreSQL	0,196	0,152	0,175	0,296	1,444	24,990

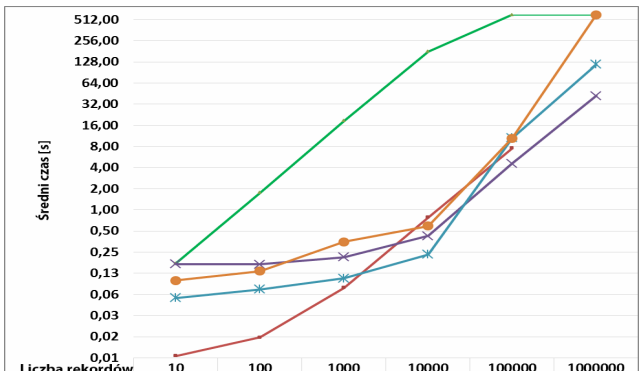
Rys. 3. Wyniki badań czasów zapisu par klucz-wartość na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Shared Preferences	0,134	0,137	0,208	0,189	0,359	1,464
Internal Storage	0,130	0,496	3,570	34,050	317,690	0,000
External Storage	0,154	0,496	0,136	34,380	334,657	0,000
SQLite	0,024	0,041	1,508	0,767	8,029	80,767
MySQL	0,149	0,121	0,267	1,671	7,023	12,957
Microsoft SQL Server	0,055	0,076	0,166	1,277	9,222	0,000
PostgreSQL	0,165	0,178	0,252	1,104	12,200	0,000

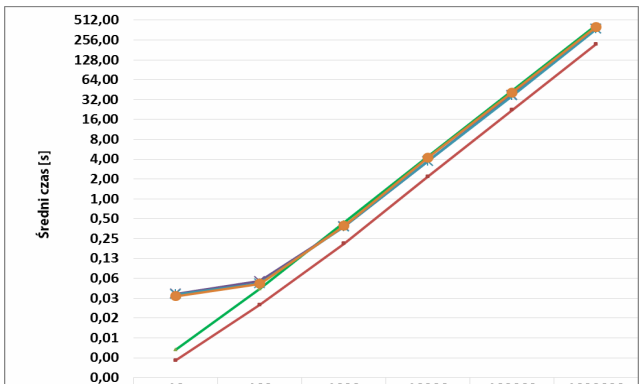
Rys. 4. Wyniki badań czasów odczytu par klucz-wartość na systemie Android

Rysunek 5 oraz rysunek 6 widoczne poniżej przedstawiają czasy jakie osiągnęły poszczególne obiekty badawcze w przypadku rekordów z danymi tekstowymi na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik	0,008	0,015	0,078	0,780	7,551	0,000
SQLite	0,173	1,757	18,510	180,523	600,000	600,000
PostgreSQL	0,169	0,167	0,213	0,427	4,581	42,029
SQL Server	0,056	0,074	0,106	0,233	10,524	117,903
MySQL	0,099	0,134	0,350	0,591	10,490	600,000

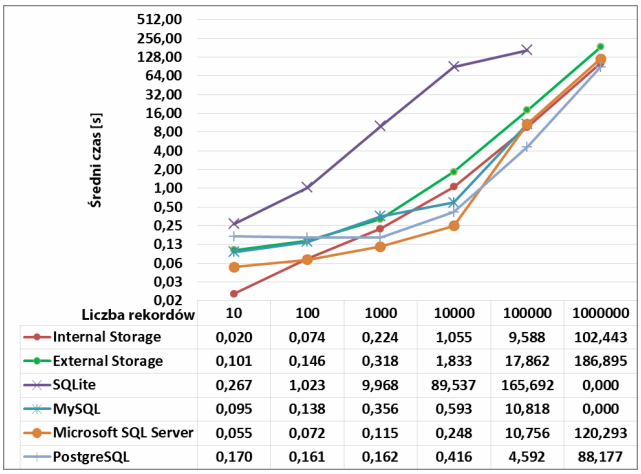
Rys. 5. Wyniki badań czasów zapisu rekordów z danymi tekstowymi na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik	0,004	0,025	0,211	2,187	22,378	223,467
SQLite	0,005	0,044	0,442	4,413	43,730	436,345
PostgreSQL	0,037	0,057	0,382	3,747	37,885	380,039
SQL Server	0,036	0,053	0,381	3,741	36,672	376,749
MySQL	0,034	0,052	0,390	4,224	41,030	405,539

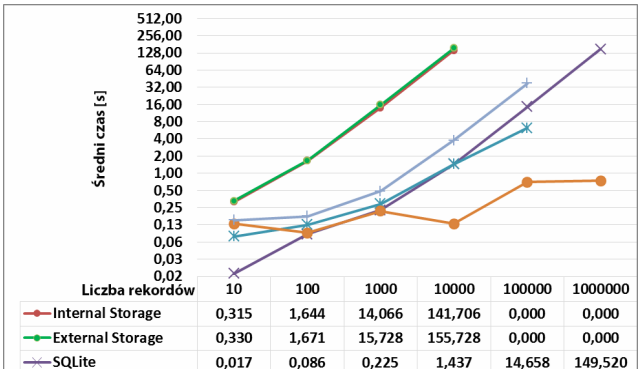
Rys. 6. Wyniki badań czasów odczytu rekordów z danymi tekstowymi na systemie Windows Mobile

Rysunek 7 i rysunek 8 przedstawia wyniki tego badania na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,020	0,074	0,224	1,055	9,588	102,443
External Storage	0,101	0,146	0,318	1,833	17,862	186,895
SQLite	0,267	1,023	9,968	89,537	165,692	0,000
MySQL	0,095	0,138	0,356	0,593	10,818	0,000
Microsoft SQL Server	0,055	0,072	0,115	0,248	10,756	120,293
PostgreSQL	0,170	0,161	0,162	0,416	4,592	88,177

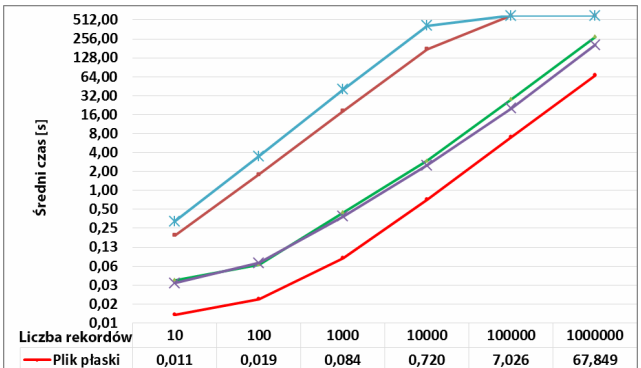
Rys. 7. Wyniki badań czasów zapisu rekordów z danymi tekstowymi na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,315	1,644	14,066	141,706	0,000	0,000
External Storage	0,330	1,671	15,728	155,728	0,000	0,000
SQLite	0,017	0,086	0,225	1,437	14,658	149,520
MySQL	0,079	0,126	0,292	1,453	6,239	0,000
Microsoft SQL Server	0,131	0,090	0,219	0,131	0,706	0,735
PostgreSQL	0,150	0,176	0,485	3,786	37,702	0,000

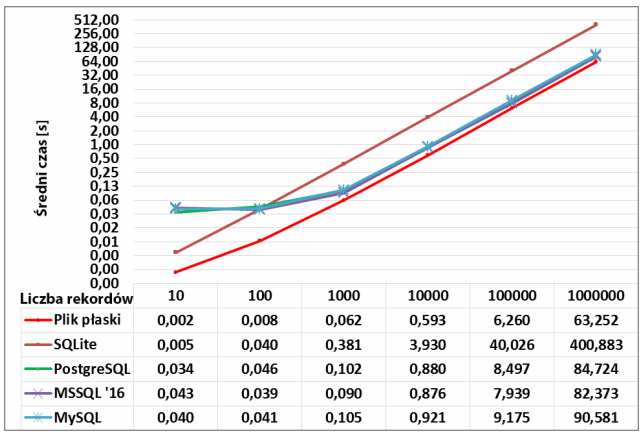
Rys. 8. Wyniki badań czasów odczytu rekordów z danymi tekstowymi na systemie Android

Rysunek 9 oraz rysunek 10 prezentują wyniki zapisu oraz odczytu rekordów z danymi numerycznymi na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik płaski	0,011	0,019	0,084	0,720	7,026	67,849
SQLite	0,193	1,802	18,321	176,785	600,000	600,000
PostgreSQL	0,038	0,066	0,446	2,975	27,975	276,017
MSSQL '16	0,034	0,070	0,386	2,492	20,207	204,524
MySQL	0,322	3,482	40,253	414,799	600,000	600,000

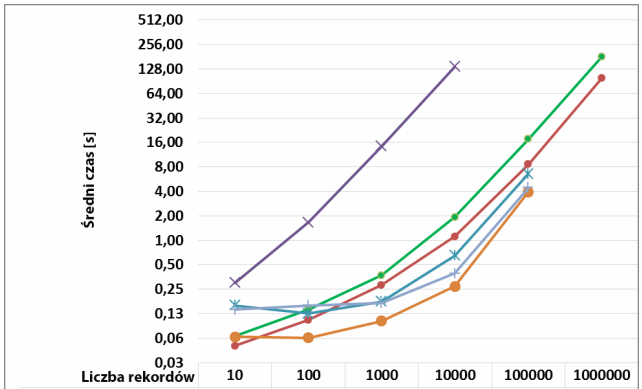
Rys. 9. Wyniki badań czasów zapisu rekordów z danymi numerycznymi na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik płaski	0,002	0,008	0,062	0,593	6,260	63,252
SQLite	0,005	0,040	0,381	3,930	40,026	400,883
PostgreSQL	0,034	0,046	0,102	0,880	8,497	84,724
MSSQL '16	0,043	0,039	0,090	0,876	7,939	82,373
MySQL	0,040	0,041	0,105	0,921	9,175	90,581

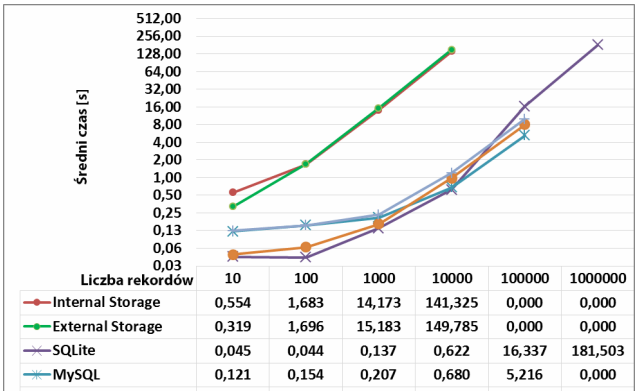
Rys. 10. Wyniki badań czasów odczytu rekordów z danymi numerycznymi na systemie Windows Mobile

Dwa kolejne wykresy widoczne na rysunku 11 i rysunku 12 to wyniki dla kolejno zapisu oraz odczytu rekordów z danymi numerycznymi na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,050	0,104	0,281	1,118	8,644	98,586
External Storage	0,066	0,141	0,373	1,935	17,632	180,796
SQLite	0,302	1,675	14,427	137,300	0,000	0,000
MySQL	0,158	0,126	0,177	0,648	6,629	0,000
Microsoft SQL Server	0,065	0,063	0,102	0,271	3,946	0,000
PostgreSQL	0,141	0,159	0,168	0,393	4,464	0,000

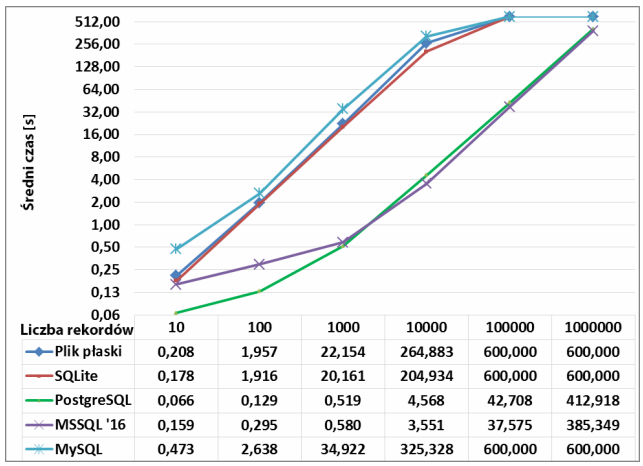
Rys. 11. Wyniki badań czasów zapisu rekordów z danymi numerycznymi na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,554	1,683	14,173	141,325	0,000	0,000
External Storage	0,319	1,696	15,183	149,785	0,000	0,000
SQLite	0,045	0,044	0,137	0,622	16,337	181,503
MySQL	0,121	0,154	0,207	0,680	5,216	0,000
Microsoft SQL Server	0,049	0,065	0,161	0,977	8,038	0,000
PostgreSQL	0,124	0,154	0,234	1,200	9,914	0,000

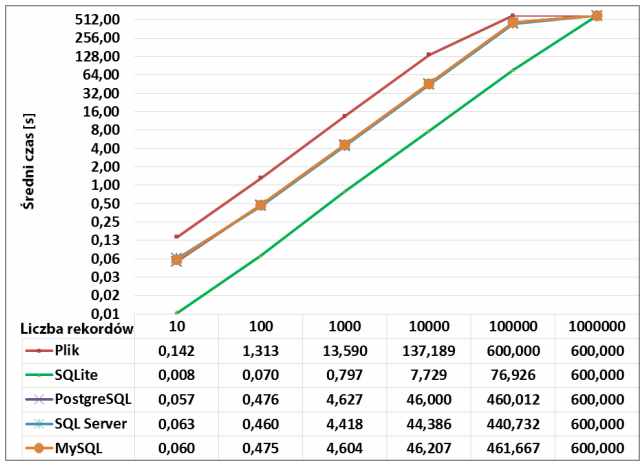
Rys. 12. Wyniki badań czasów odczytu rekordów z danymi numerycznymi na systemie Android

Wykresy z rysunku 13 do rysunku 15 włącznie dotyczą pracy na rekordach zawierających dane binarne. Rysunek 13 oraz rysunek 14 przedstawiają czasy uzyskiwane na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik płaski	0,208	1,957	22,154	264,883	600,000	600,000
SQLite	0,178	1,916	20,161	204,934	600,000	600,000
PostgreSQL	0,066	0,129	0,519	4,568	42,708	412,918
MSSQL '16	0,159	0,295	0,580	3,551	37,575	385,349
MySQL	0,473	2,638	34,922	325,328	600,000	600,000

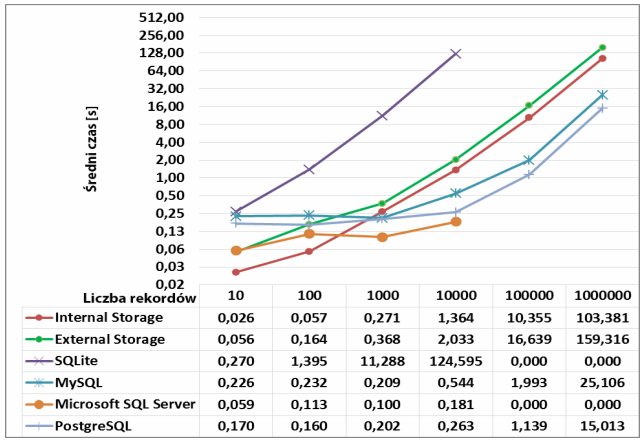
Rys. 13. Wyniki badań czasów zapisu rekordów z danymi binarnymi na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik	0,142	1,313	13,590	137,189	600,000	600,000
SQLite	0,008	0,070	0,797	7,729	76,926	600,000
PostgreSQL	0,057	0,476	4,627	46,000	460,012	600,000
SQL Server	0,063	0,460	4,418	44,386	440,732	600,000
MySQL	0,060	0,475	4,604	46,207	461,667	600,000

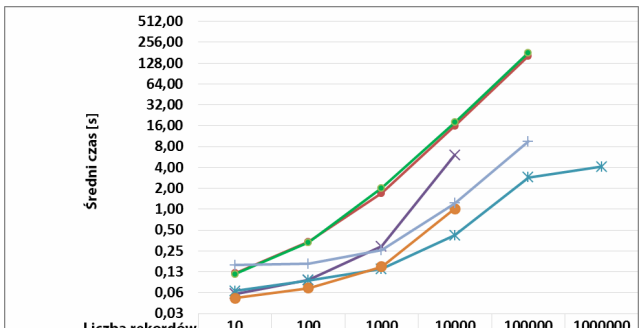
Rys. 14. Wyniki badań czasów odczytu rekordów z danymi binarnymi na systemie Windows Mobile

Rysunek 15 oraz 16 to wyniki tego samego badania na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,026	0,057	0,271	1,364	10,355	103,381
External Storage	0,056	0,164	0,368	2,033	16,639	159,316
SQLite	0,270	1,395	11,288	124,595	0,000	0,000
MySQL	0,226	0,232	0,209	0,544	1,993	25,106
Microsoft SQL Server	0,059	0,113	0,100	0,181	0,000	0,000
PostgreSQL	0,170	0,160	0,202	0,263	1,139	15,013

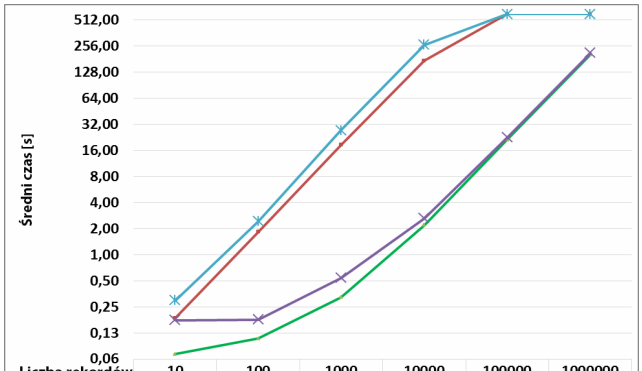
Rys. 15. Wyniki badań czasów zapisu rekordów z danymi binarnymi na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,120	0,339	1,703	15,786	161,564	0,000
External Storage	0,115	0,331	2,018	17,821	180,128	0,000
SQLite	0,060	0,095	0,290	6,041	0,000	0,000
MySQL	0,066	0,094	0,137	0,420	2,872	4,110
Microsoft SQL Server	0,052	0,072	0,148	1,009	0,000	0,000
PostgreSQL	0,156	0,162	0,256	1,218	9,473	0,000

Rys. 16. Wyniki badań czasów odczytu rekordów z danymi binarnymi na systemie Android

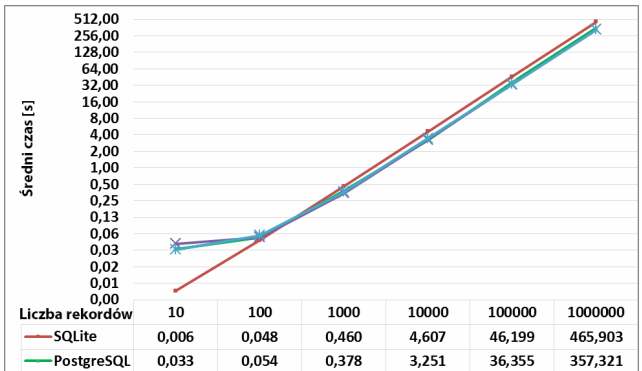
Ostatnią kategorię wyników stanowią wyniki badań czasów zapisu i odczytu rekordów z danymi o różnych typach. Na rysunku 17 widoczne są wyniki na systemie Windows Mobile uzyskane podczas zapisu danych.



Liczba rekordów	10	100	1000	10000	100000	1000000
SQLite	0,185	1,826	18,456	173,163	600,000	600,000
PostgreSQL	0,071	0,108	0,323	2,191	21,284	205,225
MSSQL '16	0,175	0,178	0,540	2,637	22,552	212,789
MySQL	0,297	2,417	27,305	265,329	600,000	600,000

Rys. 17. Wyniki badań czasów zapisu rekordów z danymi o różnych typach na systemie Windows Mobile

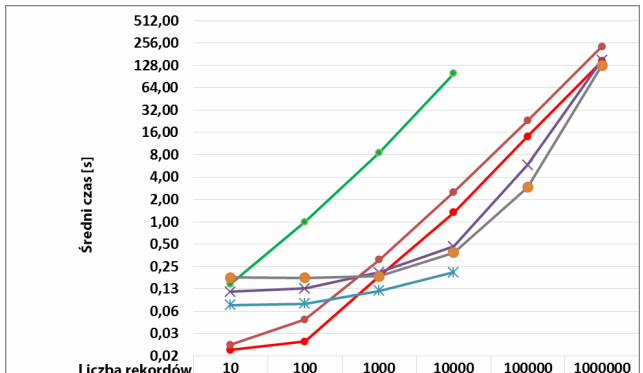
Rysunek 18 przedstawia wyniki odczytu tych danych.



Liczba rekordów	10	100	1000	10000	100000	1000000
SQLite	0,006	0,048	0,460	4,607	46,199	465,903
PostgreSQL	0,033	0,054	0,378	3,251	36,355	357,321
MSSQL '16	0,041	0,054	0,346	3,276	33,359	333,855
MySQL	0,032	0,059	0,366	3,447	33,801	336,017

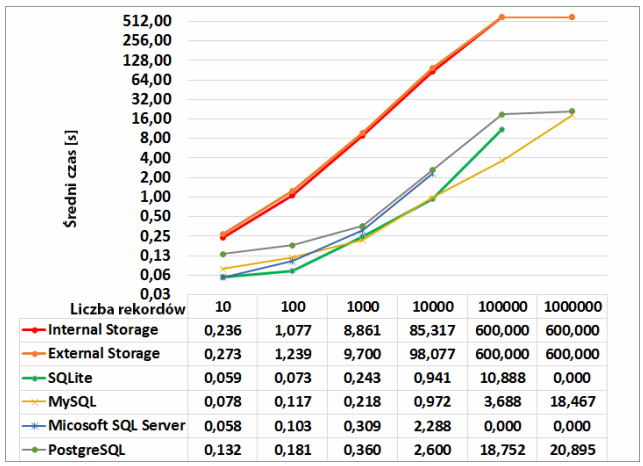
Rys. 18. Wyniki badań czasów odczytu rekordów z danymi o różnych typach na systemie Windows Mobile

Rysunek 19 i rysunek 20 to wyniki zapisu i odczytu rekordów z danymi o różnych typach na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,019	0,024	0,185	1,329	14,139	146,243
External Storage	0,022	0,048	0,307	2,515	23,021	228,600
SQLite	0,146	0,999	8,528	99,633	0,000	0,000
MySQL	0,114	0,127	0,209	0,467	5,831	150,341
Micosoft SQL Server	0,076	0,079	0,117	0,209	0,000	0,000
PostgreSQL	0,177	0,176	0,186	0,385	2,907	127,819

Rys. 19. Wyniki badań czasów zapisu rekordów z danymi o różnych typach na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,236	1,077	8,861	85,317	600,000	600,000
External Storage	0,273	1,239	9,700	98,077	600,000	600,000
SQLite	0,059	0,073	0,243	0,941	10,888	0,000
MySQL	0,078	0,117	0,218	0,972	3,688	18,467
Micosoft SQL Server	0,058	0,103	0,309	2,288	0,000	0,000
PostgreSQL	0,132	0,181	0,360	2,600	18,752	20,895

Rys. 20. Wyniki badań czasów odczytu rekordów z danymi o różnych typach na systemie Android

5. Wnioski

Należy podkreślić, że nie można wyznaczyć jednoznacznie najlepszej i uniwersalnej formy przechowywania danych na systemach mobilnych. Jak podczas każdego projektu programistycznego, całość sprowadza się do określenia indywidualnych potrzeb stawianych przez wymagania aplikacji oraz wyboru pod tym względem najbardziej odpowiedniego i optymalnego rozwiązania zapewniającego spełnienie określonych przez programistę celów, a także zapewniającego osobie korzystającej z aplikacji odpowiednio wysokie doświadczenie użytkownika.

Istnieją cechy, które są wspólne dla każdej badanej formy przechowywania danych. Mowa tutaj o liczbie rekordów, która wpływa na zwiększenie czasu operacji dodawania oraz wybierania w przypadku każdej badanej formy tj. Local Storage, Shared Preferences, Internal Storage, External Storage, SQLite oraz zewnętrznych baz danych. Najwolniejszą formą przechowywania danych na obu

systemach jest lokalna baza danych SQLite. Kolejną wspólną cechą jest to, że wszystkie formy przechowywania oraz wybierania danych są efektywne do ok. 10 000 rekordów i czas operacji nie przekracza w takich przypadkach kilkunastu sekund. Powyżej tej ilości występują problemy z alokacją pamięci urządzenia, przerwanie połączenia przez zdalny serwer bądź czas operacji wydłuża się do tego stopnia, że może być nieakceptowalny przez użytkownika urządzenia.

W przypadku par klucz-wartość stwierdzono następujące cechy:

- najszybszą formą przechowywania oraz wybierania lokalnych danych formie klucz-wartość jest Shared Preferences na systemie Android oraz LocalSettings na systemie Windows Mobile. Różnica w szybkości wybierania danych potrafi być kilkukrotnie większa na korzyść Shared Preferences w stosunku do External Storage oraz Internal Storage, podobnie sprawa ma się w przypadku LocalSettings w stosunku do pozostałych form przechowywania danych na systemie Windows Mobile;
- najszybszą formą przechowywania oraz wybierania danych na zewnętrznym serwerze jest Microsoft SQL Server, który uzyskał najlepsze czasy do 100 000 tyś rekordów. Powyżej tej ilości wystąpiły problemy z alokacją pamięci na serwerze. W tym przypadku świetnie sprawdził się MySQL, który pomimo najwolniejszych czasów tych operacji w stosunku do pozostałych zdalnych form przechowywania danych pozwolił na bezproblemowe wykonanie badań;
- najwolniejszą formą wybierania danych jest Internal oraz External Storage, w których w każdym z przypadków przy milionie rekordów operacje trwają powyżej 10 minut.

W przypadku rozpatrywania rekordów danych można wyróżnić następujące właściwości:

- najszybszą formą przechowywania lokalnych danych jest Internal Storage;
- najszybszą formą wybierania lokalnych danych typu VARCHAR, NUMERIC oraz rekordów wielotypowych jest SQLite. Dla typu BLOB jest to Internal Storage;
- najwolniejszą formą wybierania lokalnych danych typu VARCHAR, NUMERIC oraz rekordów wielotypowych w formie rekordów jest zarówno Internal Storage jak i External Storage, gdzie operacja wybierania począwszy od 100 000 rekordów trwa powyżej 10 minut. Dla typu BLOB jest to SQLite, który powoduje wyłączenie aplikacji wyrzucając wyjątek już dla 10 000 rekordów przy wielkości danych 10 kB.

Literatura

[1] Hengming F., Jia Ch., Bin X.; The Interaction Mechanism based on JSON for Android Database Application, Academic Journal, 2013 – JSON.

[2] Lee S.; Creating and Using Databases for Android Applications, International Journal of Database Theory and Application Vol. 5 No. 2, 2012.

[3] H.V. Leong and A. Si, Database Caching Over the Air-Storage, The Computer Journal 40(7) , 1997.

[4] Nurseitow N., Paulson M., Reynolds R., Izurieta C.; Comparison of JSON and XML Data Interchange Formats: A Case Study; Montana State University – Bozeman.

[5] Si A., Leong H. L., The Hung Kong Polytechnic University Query optimization for broadcast database, 1998.

[6] Wei J.; Android Database Programming, Packt Publishing Ltd., 2012.

[7] Klasa Cursor, Android Developers, https://developer.android.com/reference/android/database/Cursor.html, dostęp: październik 2016r.

[8] Klasa FileOutputStream, Oracle Help Center, https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html, dostęp: wrzesień 2016r.

[9] Klasa FileInputStream, Oracle Help Center, https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html, dostęp: wrzesień 2016r.

[10] Klasa SharedPreferences, Android Developers, https://developer.android.com/reference/android/content/SharedPreferences.html, dostęp: wrzesień 2016.

[11] Klasa SQLiteDatabase, Android Developers, https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html, dostęp: październik 2016r.

[12] Klasa SQLiteOpenHelper, Android Developers, https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html, dostęp: październik 2016r.

[13] Save a File on External Storage, Android Developers, https://developer.android.com/training/basics/data-storage/files.html#WriteExternalStorage, dostęp: wrzesień 2016r.

[14] Save a File on Internal Storage, Android Developers, https://developer.android.com/training/basics/data-storage/files.html#WriteInternalStorage, dostęp: wrzesień 2016.

JCSI 2 (2016) 140-143
Journal of Computer Sciences Institute

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

Analiza porównawcza algorytmów rozwiązujących Sudoku

Emil Wnuk*, Edyta Łukasik
Instytut Informatyki, Politechnika Lubelska, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia analizę porównawczą wybranych algorytmów, służących do rozwiązywania Sudoku. Porównywane są puzzle różnej wielkości, posiadające różny poziom trudności. Sprawdzany jest wpływ tych czynników na czas działania algorytmów. Na potrzeby badań stworzono specjalną aplikację, która zawiera wszystkie potrzebne algorytmy.

Słowa kluczowe: Sudoku; algorytmy; czas działania

*Autor do korespondencji.

Adres e-mail: emil.wnuk@gmail.com

Comparative analysis of algorithms for solving Sudoku

Emil Wnuk*, Edyta Łukasik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents a comparative analysis of selected algorithms for solving Sudoku. Puzzles of different sizes and having different levels of difficulty are compared. The impact of these factors on the duration of the algorithms is examined. For the study, a special application that contains all the necessary algorithms has been created.

Keywords: Sudoku; algorithms; solving time

*Corresponding author.

E-mail address: emil.wnuk@gmail.com

1. Wstęp

Wbrew powszechnej opinii, Sudoku nie zostało wynalezione w Japonii. Pierwsza publikacja łamigłówki miała miejsce w Stanach Zjednoczonych w 1979 roku [1]. Do kraju samurajów puzzle dotarły siedem lat później, jednak międzynarodową popularność zyskały dopiero w roku 2005 [2]. Nazwa układanki pochodzi od japońskiego wyrażenia oznaczającego: „cyfry muszą być pojedyncze" [3].

Klasyczna wersja Sudoku ma kształt diagramu 9 × 9, który dzieli się na dziewięć kwadratów 3 × 3 [4]. Celem łamigłówki jest wypełnienie schematu tak, aby w każdym wierszu, każdej kolumnie oraz każdym kwadracie 3 × 3 znalazło się dokładnie po jednej cyfrze z zakresu od 1 do 9 [5]. Cyfry mają tu znaczenie umowne, ponieważ inne symbole (litery, znaki, kolory) również mogą być stosowane [6].

Każde Sudoku zaprojektowane jest tak, aby posiadało dokładnie jedno rozwiązanie [7]. Podczas wyszukiwania brakujących cyfr nie można zmieniać już istniejących w danym diagramie. Aby poprawnie rozwiązać Sudoku trzeba dokładnie, krok po kroku, wykonywać pewne żmudne algorytmy. Pośpiech i brak umiejętności logicznego myślenia sprzyjają popełnianiu błędów, które często zauważane są dopiero w końcowym etapie rozwiązywania. Taka sytuacja sprzyja frustracji uczestnika gry, dlatego bardzo pomocny okazuje się komputer – idealne narzędzie do rozwiązywania tego typu zagadek.

W przeprowadzonych testach analizowano czas rozwiązywania Sudoku za pomocą algorytmów komputerowych. Zbadano puzzle różnej wielkości:

- *s2* (4 × 4);
- *s3* (9 × 9);
- *s4* (16 × 16);
- *s5* (25 × 25);

posiadające różny poziom trudności:

- *low* (niski);
- *med* (średni);
- *top* (wysoki);
- *xxx* (ekstremalny).

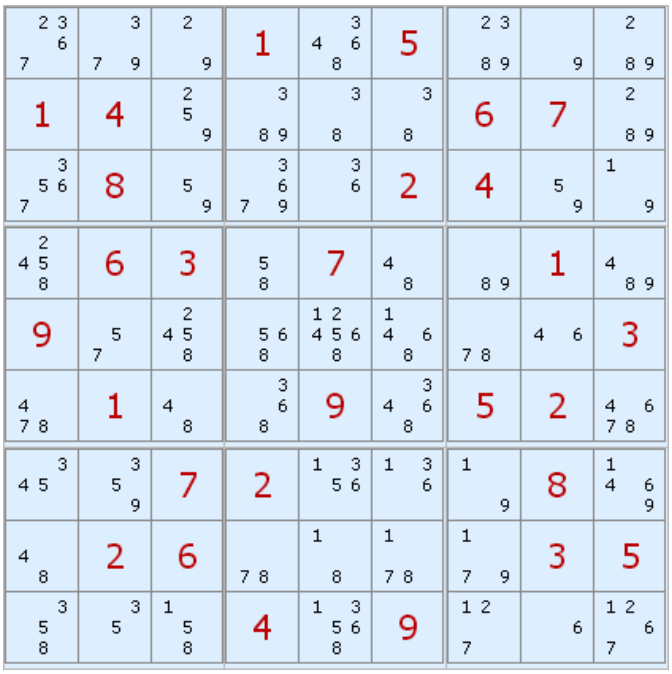
Na potrzeby badań stworzono specjalną aplikację, która zawiera wszystkie potrzebne algorytmy:

1) Brute Force (BF)
2) Brute Force Fast (BFF)
3) Backtracking (BT)
4) Backtracking Fast (BTF)
5) Naked Single (NS)
6) Naked Single Backtracking (NSBT)
7) Hidden Single (HS)
8) Hidden Single Backtracking (HSBT)
9) Multiple Choice (MC)

W przypadku, gdy czas działania algorytmu przekroczył 60 sekund, przerywano jego wykonywanie, a do wyniku przypisywano symbol *inf*. W przypadku, gdy rozwiązanie okazało się niepoprawne, wynikiem stawał się symbol *err*.

2. Brute Force

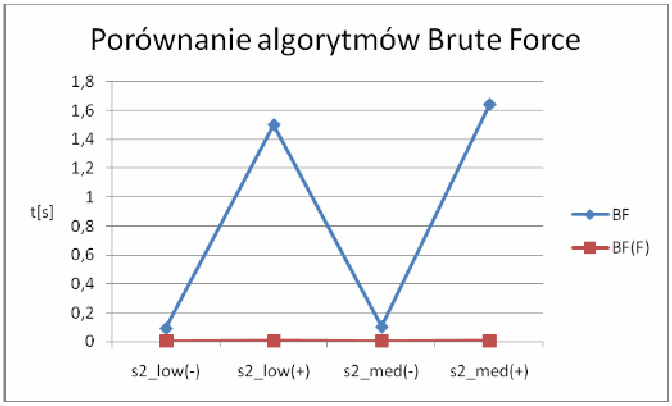
Algorytm Brute Force (BF) to prosty algorytm naiwny, który wpisuje w puste pola wszystkie kombinacje cyfr od 1 do 9 i za każdym razem sprawdza, czy rozwiązanie jest poprawne. W przykładowym Sudoku (rysunek 1) znajduje się 28 cyfr początkowych. Do wstawienia pozostało 53 cyfry, zatem algorytm musi sprawdzić aż 9^{53} przypadków. BF jest najwolniejszym z zaimplementowanych algorytmów. Jego zaletami są prostota i pewność, że wszystkie prawidłowe rozwiązania zostaną znalezione [8].



Rys. 1. Przykładowe Sudoku

Brute Force Fast (BFF) jest ulepszoną wersją algorytmu BF, która jest dużo szybsza dzięki temu, że w puste pola wstawiane są tylko te wartości, które nie kolidują z cyframi początkowymi. Wykaz tych wartości przedstawia Rysunek 1.

Na rysunku 2 przedstawiono wykres z czasami działania algorytmów BF i BFF. Testy przeprowadzono na Sudoku drugiego stopnia (4 × 4).



Rys. 2. Porównanie algorytmów Brute Force

3. Backtracking

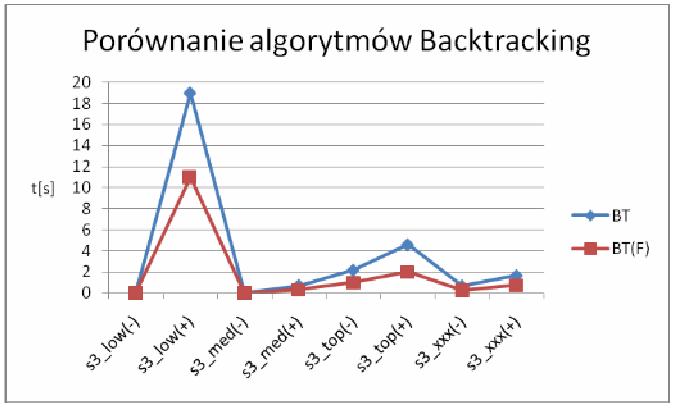
Algorytm Backtracking (BT), podobnie jak BF, także korzysta ze wszystkich cyfr od 1 do 9, ale po każdym wstawieniu cyfry sprawdza, czy nie koliduje ona z pozostałymi wartościami znajdującymi się na planszy [9].

Jeśli wystąpi sprzeczność, to dana wartość jest usuwana, a na jej miejsce zostaje wstawiona kolejna. Pokazane jest to na rysunku 3, gdzie cyfra znajdująca się w polu (8,2) musi zostać zastąpiona, ponieważ koliduje z polem (2,2) oraz (7,7).

Rys. 3. Działanie algorytmu Backtracking

Backtracking Fast (BTF) jest ulepszoną wersją algorytmu BT, która (podobnie jak algorytm BFF) korzysta z wartości, które nie kolidują z cyframi początkowymi. Dzięki temu jest najszybszym spośród zaimplementowanych algorytmów naiwnych (BF, BFF, BT, BTF). Złożoność obliczeniowa algorytmów naiwnych jest postaci potęgowej.

Rysunek 4 przedstawia porównanie algorytmów BT i BTT, przeprowadzone na Sudoku trzeciego stopnia (9 × 9).

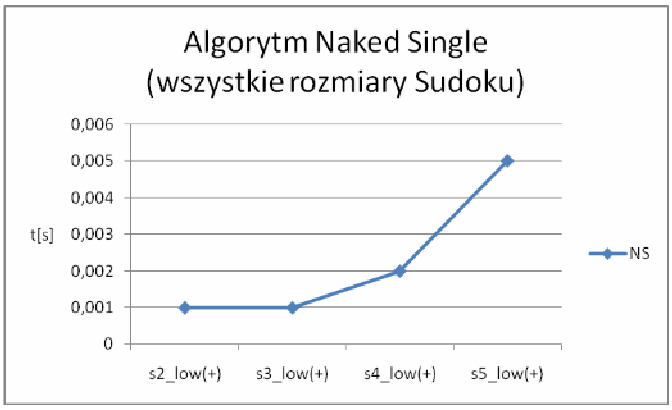


Rys. 4. Porównanie algorytmów Backtracking

4. Algorytmy eliminacyjne

Naked Single (NS) to algorytm, który stosuje metodę eliminacji. Jeżeli w danym polu znajduje się tylko jeden kandydat, wartość ta zostaje wstawiona na stałe. W przykładzie na rysunku 1 takimi polami są (8,1) i (8,9). Nowo powstałe wartości wykorzystywane są do eliminacji następnych. Jeżeli przynajmniej jedna cyfra zostanie wstawiona, to algorytm wykonuje się jeszcze raz, w przeciwnym wypadku działanie zostaje przerwane [10].

Na rysunku 5 znajduje się wykres przedstawiający wyniki algorytmu NS ze wszystkich rozmiarów Sudoku.

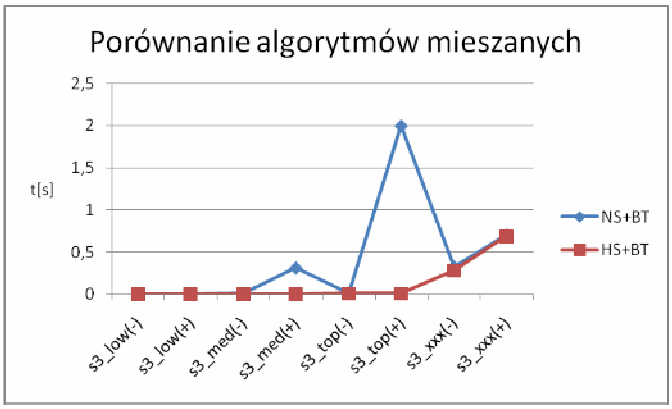


Rys. 5. Algorytm Naked Single

Algorytm Hidden Single (HS) jest zbliżony do NS, który wyszukiwał komórkę z jednym kandydatem. HS przeszukuje wiersze, kolumny i kwadraty w celu znalezienia takiego ciągu, gdzie dany kandydat występuje tylko raz [11].

5. Algorytmy mieszane

Na rysunku 6 znajdują się wykresy algorytmów Naked Single Backtracking (NSBT) oraz Hidden Single Backtracking (HSBT). NSBT jest połączeniem dwóch algorytmów: NS i BT z tym, że BT uruchamiany jest na wartościach, które nie zostały wyeliminowane, lecz tylko i wyłącznie wtedy, gdy NS nie uzyskał rozwiązania. W przypadku HSBT sposób działania jest analogiczny do NSBT. Najpierw uruchomiony jest HS, a następnie BT.

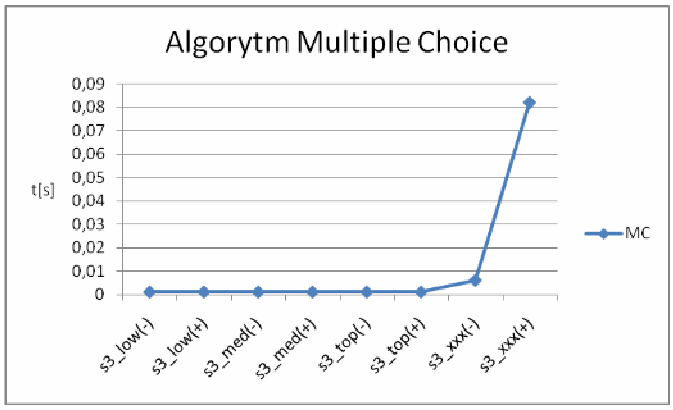


Rys. 6. Porównanie algorytmów mieszanych

Algorytmy mieszane to połączenie algorytmów eliminacyjnych (NS i HS) z naiwnym BT. Na wykresie (rysunek 6) widać, że na łatwych poziomach trudności czas wykonania jest zbliżony do 0. Na trudniejszych poziomach występuje tendencja rosnąca z tym, że HSBT ma bardziej przewidywalny przebieg w porównaniu do NSBT.

Multiple Choice (MC) to połączenie trzech algorytmów: NS, HS oraz specjalnego typu Backtrackingu, który polega na tym, że po wykonaniu NS i HS, algorytm wybiera pole posiadające dwóch kandydatów i dzieli się na dwie rekurencyjne części. Każda część uznaje swojego kandydata za prawidłowego i działa w oparciu na tej tezie. Po wykryciu sprzeczności następuje powrót to wyższego poziomu rekursji.

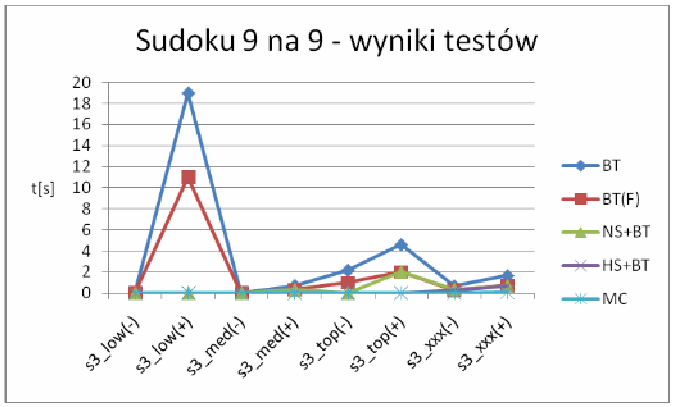
Na rysunku 7 znajduje się wykres przedstawiający wyniki algorytmu MC.



Rys. 7. Algorytm Multiple Choice

Sudoku na wykresie na rys. 7 są uporządkowane w kolejności od najłatwiejszego do najtrudniejszego i w tym wypadku czasy działania algorytmów potwierdziły tę tezę.

Podsumowanie testów dla Sudoku trzeciego stopnia przedstawia wykres na rysunku 8.



Rys. 8. Sudoku 9 na 9 – wyniki testów

Zbiorcze zestawienie wszystkich wyników przedstawia tabela 1.

Tabela 1. Zbiorcze zestawienie wszystkich wyników

Sudoku	BF	BF(F)	BT	BT(F)	NS	NS+BT	HS	HS+BT	MC
s2_low(-)	0,094	0,004	0,005	0,004	0,001	0,001	0,001	0,001	0,001
s2_low(+)	1,500	0,006	0,005	0,004	0,001	0,001	0,001	0,001	0,001
s2_med(-)	0,104	0,004	0,004	0,004	0,001	0,001	0,001	0,001	0,001
s2_med(+)	1,640	0,006	0,004	0,004	0,001	0,001	0,001	0,001	0,001
s3_low(-)	inf	inf	0,008	0,008	0,001	0,001	0,001	0,001	0,001
s3_low(+)	inf	inf	19,00	11,00	0,001	0,001	0,001	0,001	0,001
s3_med(-)	inf	inf	0,014	0,008	err	0,008	0,001	0,001	0,001
s3_med(+)	inf	inf	0,719	0,315	err	0,314	0,001	0,001	0,001
s3_top(-)	inf	inf	2,172	0,984	err	0,007	err	0,007	0,001
s3_top(+)	inf	inf	4,600	1,984	err	2,000	err	0,007	0,001
s3_xxx(-)	inf	inf	0,672	0,297	err	0,328	err	0,281	0,006
s3_xxx(+)	inf	inf	1,656	0,734	err	0,703	err	0,688	0,082
s4_low(-)	inf	inf	inf	inf	0,002	0,002	0,002	0,002	0,002
s4_low(+)	inf	inf	inf	inf	0,002	0,002	0,002	0,002	0,002
s4_med(-)	inf	inf	inf	inf	err	0,015	0,002	0,002	0,002
s4_med(+)	inf	inf	inf	inf	err	inf	err	inf	0,828
s4_top(-)	inf	inf	inf	inf	err	inf	err	0,254	0,010
s4_top(+)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_low(-)	inf	inf	inf	inf	0,004	0,004	0,004	0,004	0,004
s5_low(+)	inf	inf	inf	inf	0,005	0,005	0,005	0,005	0,005
s5_med(-)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_med(+)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_top(-)	inf	inf	inf	inf	err	inf	err	inf	inf
s5_top(+)	inf	inf	inf	inf	err	inf	err	inf	inf
inf - czas działania algorytmu przekroczył 60 sekund									
err - algorytm zakończył działanie (wynik niepoprawny)									

6. Wnioski

Algorytmy Brute Force (BF i BFF) uzyskały czasy poniżej dwóch sekund dla wszystkich Sudoku drugiego stopnia.

Algorytmy Backtracking (BT i BTF) uzyskały czasy poniżej dwudziestu sekund dla wszystkich Sudoku 9 × 9.

Algorytmy eliminacyjne (NS i HS) wykonały się poprawnie dla wszystkich Sudoku z poziomu *low*. Były to również jedyne algorytmy, które zwróciły zły wynik.

Algorytm MC uzyskał najlepszy wynik w każdym z rozpatrywanych przykładów – oznacza to, że jest najbardziej optymalnym algorytmem. Jest to również jedyny algorytm, który dał poprawne rozwiązanie w przykładzie *s4_med(+)*.

Hierarchia algorytmów pod względem czasów wykonania przedstawia się następująco: BF > BT > BFF > BTF > NS > NSBT > HS > HSBT > MC.

Literatura

[1] Wilson R.J., Jak rozwiązywać sudoku, Dom Wydawniczy Rebis, Poznań 2005

[2] H. Intelm, How to Solve Every Sudoku Puzzle, Vol.2, Geostar Publishing LLC. 2005.

[3] N. Jussien, A-Z of Sudoku, ISTE Ltd., USA, 2007.

[4] W.-M. Lee, Programming Sudoku, Apress, USA, 2006.

[5] Arnoldy, Ben. "Sudoku Strategies". *The Home Forum*. The Christian Science Monitor.

[6] Gordon Royle, The University of Western Australia. Minimum Sudoku.

[7] M.H.Alsuwaiyel, Algorithms Design Techniques and Analysis. London: World Scientific Publishing Company. 2008.

[8] Daniel J. Bernstein, Understanding brute force. 2007.

[9] Peter van Beek, Backtracking Search Algorithms, 2006 Elsevier

[10] George T. Heineman, Algorithms in a nutshell, Apress, 2008

[11] H. Intelm, How to Solve Every Sudoku Puzzle, Vol.4, Geostar Publishing LLC. 2007.