

## STRESZCZENIE

Niniejsza praca opisuje proces powstawania projektu pod tytułem: "Interaktywna wizualizacja stron Wikipedii w jaskini rzeczywistości wirtualnej". Jest inspirowany Projektem Opte, którego twórcy podjęli próbę zwizualizowania całego internetu, skupia się jednak na mniejszym, bardziej osiągalnym celu: pokazywaniu związków i połączeń między artykułami zamieszczonymi w Wikipedii.

Po wstępnie znajdują się przykłady istniejących już podobnych rozwiązań. Następnie wypisane są wymagania oraz scenariusze użycia tak jak zostały one zdefiniowane przed rozpoczęciem prac nad projektem. Następnie opisany jest proces wytwarzania, poczynając od pozyskania i przetworzenia danych. Pobrane zrzuty z baz Wikipedii są poddawane ciągowi przekształceń usuwających niepotrzebne informacje oraz kompresujących pozostałe w przygotowaniu do wygodnego użycia przez główną aplikację. Cały proces jest zamknięty w prostym programie z interfejsem użytkownika pozwalającym na wybranie typu przetwarzanej Wikipedii oraz monitorowanie postępu czasochłonnego tworzenia plików danych.

Opis aplikacji obejmuje zastosowane techniki reprezentacji wizualnej, z uwzględnieniem kroków powiększonych w celu optymalizacji symulacji, szczegółowy opis jej działania oraz dostępnych funkcji. Są to między innymi dwa tryby poruszania się po wizualizacji, konsola operatora z możliwością wyszukiwania oraz automatyczne odtwarzanie przygotowanych wcześniej akcji. Wyjaśniony jest też proces dostosowania działania projektu do środowiska Laboratorium Zanurzonej Wizualizacji Przestrzennej. Dodatkowo rozwinięty zostaje koncept linii czasu, która została opisana w specyfikacji wymagań, jednak z przyczyn technicznych nie mogła zostać dodana.

Rozdział "Eksperyment" dokumentuje użycie aplikacji w docelowym środowisku dużej jaskini Laboratorium Zanurzonej Wizualizacji Przestrzennej. W podsumowaniu znajduje się przegląd całości projektu. Zamieszczone zostało tam też zestawienie pierwotnych założeń z końcowym produktem oraz wnioski autorów.

**Słowa kluczowe:** Wikipedia, artykuły, połączenia, dane, wizualizacja, jaskinia rzeczywistości wirtualnej, CAVE, Laboratorium Zanurzonej Wizualizacji Przestrzennej, trasa, konsola, integracja, wymagania

**Dziedzina nauki i techniki, zgodnie z wymogami OECD:** Nauki inżynierskie i techniczne, Elektrotechnika, elektronika, inżynieria informatyczna, Sprzęt komputerowy i architektura komputerów

## ABSTRACT

This paper describes the process of creating the project titled: "Interactive visualization of Wikipedia pages in the Immersive 3D Visualization Lab". It is inspired by the Opte Project, whose creators have attempted to visualize the entire Internet. The project however focuses on a smaller, more achievable goal: showing relations and connections between Wikipedia articles.

After the introduction, examples of similar applications are listed. Then the requirements and use cases have been specified as they had been defined before any work on the project begun. Next, the production process is described, starting with the data acquisition and processing. Downloaded dumps from Wikipedia databases are subjected to a series of transformations removing unnecessary information and compressing the data in preparation for use by the main application. The whole process is enclosed in a simple program with a user interface that allows to easily select the type of Wikipedia to be processed and to monitor the progress of this time-consuming task.

The application description includes the techniques used in visual representation as well as the steps taken to optimize the simulation and a detailed description of its inner workings and available functions. These include, among others, two modes of navigating the visualization, the operator's console with the ability to search and automatically play sets of actions prepared earlier. The process of adapting the project to the environment of the Immersive 3D Visualization Lab is also explained. In addition, the concept of a timeline, which was described in the requirements but for technical reasons could not be added is discussed in greater detail.

The "Experiment" chapter documents the application being used in its target environment: large cave in the Immersive 3D Visualization Lab. The summary contains an overview of the entire project, comparison of the original assumptions with the final product as well as the authors' conclusions.

**Keywords:** Wikipedia, articles, connections, data, visualization, virtual reality, CAVE, Immersive 3D Visualization Lab, route, console, integration

## **SPIS TREŚCI**

1. WSTĘP I CEL PRACY .....	9
1.1. Inspiracja .....	9
1.2. Opis projektu.....	9
1.3. Główne cele projektu.....	10
1.4. Interesariusze i użytkownicy .....	10
2. ANALIZA ISTNIEJĄCYCH ROZWIĄZAŃ .....	11
2.1. Aplikacja Webverse .....	11
2.2. Aplikacje WikiGalaxy i Wikiverse .....	12
2.3. Pozostałe rozwiązania.....	13
3. SPECYFIKACJA WYMAGAŃ PROJEKTU .....	14
3.1. Wymagania funkcjonalne.....	14
3.2. Wymagania jakościowe.....	17
3.3. Wymagania techniczne.....	17
3.4. Scenariusze użycia .....	18
3.5. Diagram przypadków użycia .....	19
4. PRZYGOTOWYWANIE DANYCH WEJŚCIOWYCH .....	20
4.1. Źródło danych.....	20
4.2. Pobieranie i dekompresja .....	21
4.3. Parsowanie informacji.....	22
4.4. Tworzenie struktury grafu .....	24
4.4.1. Generowanie brakujących danych .....	24
4.4.2. Opis poszczególnych plików .....	25
4.4.3. Metoda generowania plików .....	26
4.5. Narzędzie WikiGraph Parser.....	28
5. IMPLEMENTACJA GŁÓWNEJ APLIKACJI WIKIGRAPH .....	30
5.1. Model danych .....	30
5.2. Wizualna reprezentacja grafu .....	30
5.2.1. Reprezentacja węzła .....	31
5.2.2. Reprezentacja połączenia .....	33
5.3. Tryby poruszania się i widoki węzła .....	33
5.4. Elementy zwiększające użyteczność aplikacji.....	35
5.5. Historia przeglądania oraz zaprogramowane trasy .....	37
5.6. Integracja z jaskinią rzeczywistości wirtualnej.....	38
5.6.1. Moduł wejścia .....	39
5.6.2. Synchronizacja stanu.....	40
5.7. Zarządzanie pamięcią w aplikacji .....	40
5.8. Konsola operatora .....	41
5.9. Linia czasu .....	43
5.10.Ostateczny schemat interakcji .....	44

6. EKSPERYMENT .....	45
7. PODSUMOWANIE.....	51
Wykaz literatury.....	53
Spis rysunków .....	53
Spis tablic .....	54
Spis listingów .....	55

## **WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW**

**LZWP, Laboratorium** Laboratorium Zanurzonej Wizualizacji Przestrzennej, znajdujące się w budynku Wydziału ETI Politechniki Gdańskiej

**Duża Jaskinia** Główna jaskinia rzeczywistości wirtualnej, składająca się z sześciu ścian

**Średnia Jaskinia** Jaskinia rzeczywistości wirtualnej składająca się z czterech ścian - prawej, lewej, frontalnej oraz dolnej

**Mała Jaskinia** Środowisko stworzone z czterech monitorów, pomniejszona wersja średniej jaskini

# 1. WSTĘP I CEL PRACY

## 1.1. *Inspiracja*

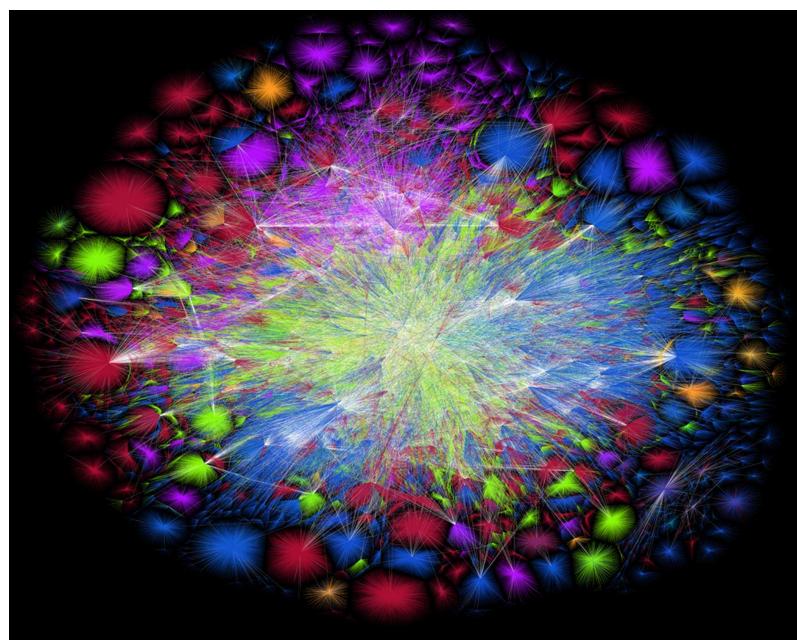
MIKOŁAJ MIRKO

W 2003 roku niejaki Barrett Lyon, informatyk oraz artysta, postawił sobie za cel stworzenie wiernego odwzorowania połączeń między komputerami w sieci Internet. Wizualizacja grafu miała być zrealizowana za pomocą kolorowych linii oraz punktów. W ten sposób w październiku 2003 roku powstał open source'owy Projekt Opte. Głównym celem tego projektu było zilustrowanie wciąż szybko rozwijającego się Internetu oraz wyróżnienie regionów, które w tamtych czasach doświadczały gwałtownego wzrostu łączności z Internetem.

Projekt szybko zyskał dużą popularność, a jeden z efektów końcowych można było zobaczyć na żywo w Muzeum Sztuki Nowoczesnej w Nowym Jorku. Przykładową wizualizację zaprezentowano na rysunku 1.1.

Twórca Opte Project w artykule dla Time [?] krótko podsumował swoją motywację:

“ The Internet is really big, very connected and extremely complex.  
It's this whole world you can't see. That's the fun part of visualizing it. ” [?]



Rysunek 1.1. Jedna z wizualizacji stworzona przez Opte Project [?]

## 1.2. *Opis projektu*

Tak jak w przypadku Opte Project, głównym zadaniem projektu jest przygotowanie grafu oraz jego wizualizacja. Nie jest to jednak graf połączeń sieci Internet, lecz sieć artykułów i kategorii internetowej encyklopedii Wikipedia. Tworzona jest ona na podstawie odnośników znajdujących się w treści artykułu, prowadzących do innych, tematycznie powiązanych, artykułów.

Nasza aplikacja oferuje nowy i innowacyjny sposób na przeglądanie Wikipedii - poprzez wizualizację połączeń pomiędzy artykułami. Dostępne są również narzędzia pozwalające na sprawne poruszanie się po grafie, jak i dodatkowe funkcjonalności służące podstawowej analizie prezentowanych danych.

W celu pogłębienia immersji aplikacja została napisana na środowisko jaskini rzeczywistości wirtualnej znajdującej się w LZWP (Laboratorium Zanurzonej Wizualizacji Przestrzennej). Mając do dyspozycji widok ze wszystkich stron, można przenieść użytkownika w dowolne miejsce skomplikowanego grafu, co pozwoli mu przyjrzeć się połączeniom z bliska i lepiej zrozumieć strukturę Wikipedii.

### **1.3. Główne cele projektu**

Głownym celem projektu jest propozycja rozwiązania mającego zainteresować oraz inspirować odbiorców nietypowym przedstawieniem znanej powszechnie Wikipedii. Nietypowość polega głównie na zmianie medium, przez które użytkownicy zwykle odbierają Wikipedię. Zamiast traktować ją jako zbiór artykułów, skupiamy się na uwidocznieniu związków między nimi, które mogłyby być trudne do uchwycenia podczas przeglądania stron encyklopedii w przeglądarce internetowej.

Aplikacja wzbogaci nieustannie powiększający się zestaw materiałów dydaktycznych przygotowywanych w środowisku Laboratorium Zanurzonej Wizualizacji Przestrzennej na Politechnice Gdańskiej. Różnorodne aplikacje pozwalają lepiej zaprezentować możliwości rozwiązań wizualizacji przestrzennej gościom odwiedzającym LZWP.

### **1.4. Interesariusze i użytkownicy**

Projekt jest realizowany jako projekt inżynierski pod opieką dr inż. Jacka Lebiedzia. Jest on głównym interesariuszem, a także jednym z użytkowników. Pozostałymi użytkownikami są osoby pracujące w laboratorium, goście zaproszeni w celu prezentacji jego możliwości i oprogramowania oraz zespół deweloperski opracowujący aplikację. Szczegółowa lista interesariuszy i użytkowników zamieszczona została w tablicy 1.1.

Interesariusz/użytkownik	Opis
dr inż. Jacek Lebiedź	Opiekun projektu inżynierskiego oraz kierownik LZWP, wykładowca na WETI PG, pracownik Katedry Inteligentnych Systemów Interaktywnych
Pracownicy laboratorium	Zespół pracowników laboratorium rozwijający oprogramowanie jaskini oraz oferujący pomoc w dostosowaniu aplikacji do środowiska i warunków LZWP
Zespół deweloperski	Osoby pracujące nad aplikacją
Goście LZWP	Zaprozone osoby, którym prezentowane są możliwości jaskini na różnego rodzaju wydarzeniach, m.in. konferencjach i dniach otwartych

Tablica 1.1. Zestawienie interesariuszy i użytkowników projektu

## 2. ANALIZA ISTNIEJĄCYCH ROZWIĄZAŃ

MIKOŁAJ MIRKO

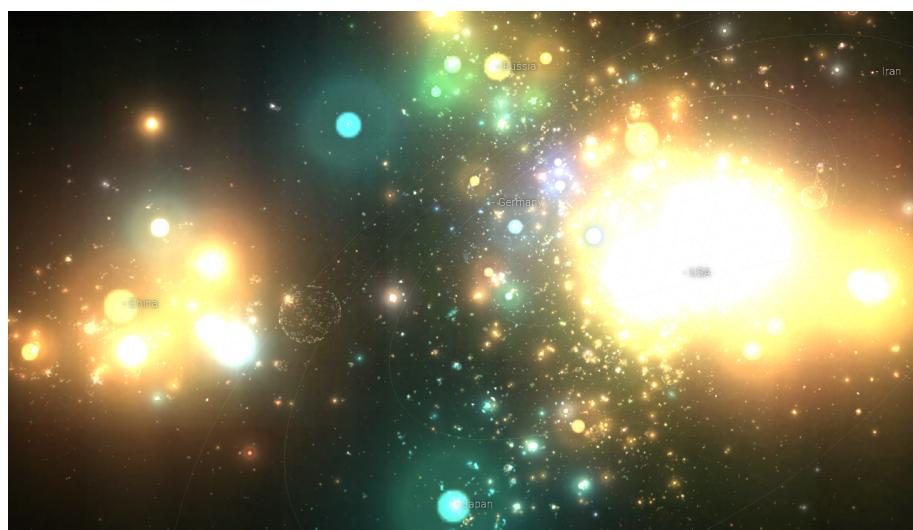
W tym rozdziale przeanalizowane zostaną różne istniejące rozwiązania przetwarzania i wizualizowania danych. Większość z nich pracuje na danych z Wikipedii, wszystkie zaś reprezentują zgromadzone informacje w postaci grafu połączeń. Każda pozycja zostanie opisana oraz oceniona.

### 2.1. *Aplikacja Webverse*

Webverse [?] to aplikacja przeglądarkowa stworzona w listopadzie 2016 r. przez Owena Corneca, amerykańskiego artystę i dewelopera. W swojej pracy skupia się na wizualizacjach różnego typu danych. W przypadku tej aplikacji, zaprezentował on trójwymiarowy graf połączeń pomiędzy 200 tysiącami stron w sieci Internet. Projekt ten powstał w ramach obchodów dwudziestolecia istnienia Internet Archive<sup>1</sup> – serwisu archiwizującego miliony stron internetowych oraz innych treści.

Aplikacja składa się z interfejsu (zawierającego wyszukiwarkę stron, ustawienia wyświetlania i listy najpopularniejszych lokalizacji) oraz przestrzeni z grafem (zrzut ekranu aplikacji na rysunku 2.1). Węzły grafu reprezentują domeny stron internetowych, a linie pojawiające się po zaznaczeniu węzła informują użytkownika o przynależności do jakiejś grupy. Autor aplikacji skupia się na wizualizacji skupisk (grup) stron, należących do tych samych lokalizacji geograficznych. Użytkownik jest więc w stanie w łatwy sposób porównać wielkość grupy stron pochodzących ze Stanów Zjednoczonych, z grupą reprezentującą Chiny.

Głównym problemem aplikacji jest uciążliwe poruszanie się w przestrzeni i nakładające się na siebie punkty grafu, utrudniające interakcję. Dużą zaletą jest możliwość wyszukiwanie stron po nazwie oraz rejestrowanie historii poruszania się.



Rysunek 2.1. Fragment grafu aplikacji Webverse [?]

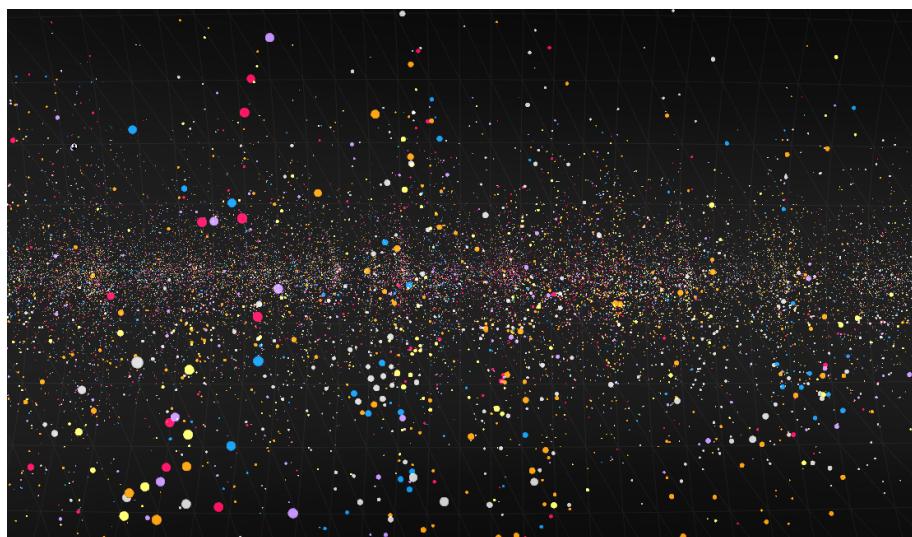
---

<sup>1</sup>Internet Archive dostępne jest pod adresem <https://archive.org/>

## 2.2. Aplikacje WikiGalaxy i Wikiverse

Owen Corne (twórca Webverse [?]) jest również autorem kolejnych dwóch przytoczonych aplikacji: WikiGalaxy [?] (z roku 2014) oraz jej nowszej wersji Wikiverse [?] (z roku 2016). Oba projekty wizualizują artykuły oraz ich połączenia oferując dodatkowe narzędzia wspomagające pracę z aplikacją.

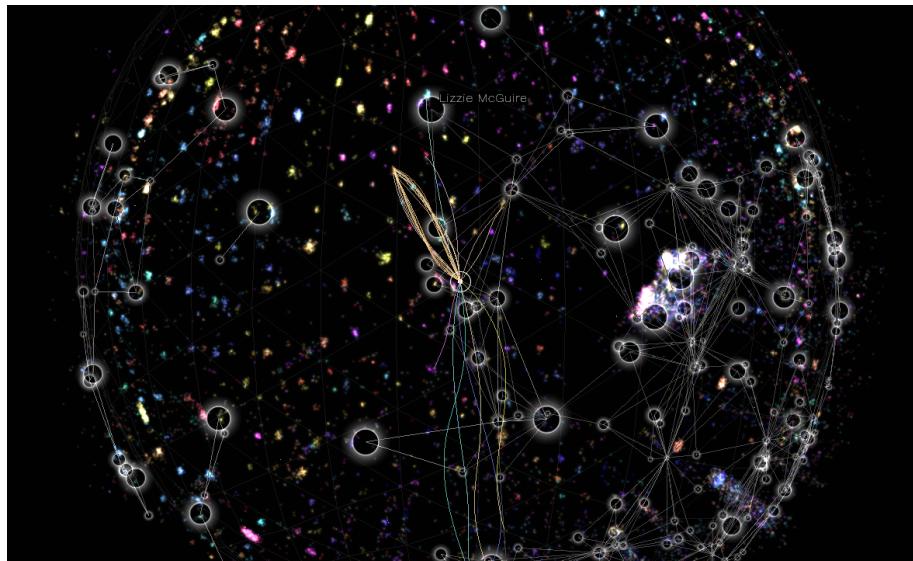
WikiGalaxy oferuje dwa widoki: mapy (widok z góry) oraz pierwszoosobowy (poruszanie się w przestrzeni – widok zilustrowany na rysunku 2.2). Każdy ze 100 tysięcy artykułów reprezentowany jest jako punkt w przestrzeni. Kolor punktu określa jego przynależność do jednej z predefiniowanych kategorii. Po zaznaczeniu artykułu użytkownik jest w stanie przeglądać połączenia wychodzące z wybranego artykułu do innych artykułów. Aplikacja dostarcza (podobnie jak w przypadku Webverse) wyszukiwarkę i historię wykonanych akcji, ułatwiając nawigację po grafie. Na dzień 29 listopada 2019 r. aplikacja zdaje się mieć problemy z wczytywaniem tytułów artykułów.



Rysunek 2.2. Widok pierwszoosobowy przestrzeni połączeń WikiGalaxy [?]

Wikiverse to sprawnie działający podgląd artykułów i ich połączeń w postaci grafu wzorowanego na galaktyce. Zawiera on 250 tysięcy artykułów rozmieszczonych w przestrzeni trójwymiarowej za pomocą specjalnego algorytmu fizycznego formułującego sferyczny rozkład punktów (rysunek 2.3). Historia odwiedzonych punktów znajduje się w panelu bocznym. Aplikacja nie posiada wyszukiwarki. Daje jednak możliwość szybkiego podglądu aktualnej treści artykułu wewnątrz aplikacji.

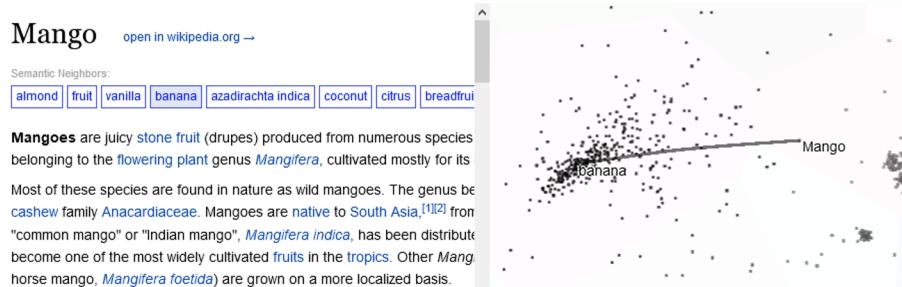
Niestety, w obu projektach wspierany jest tylko fragment anglojęzycznej Wikipedii. Grafy zostały wygenerowane odpowiednio w 2014 i 2016 roku. WikiGalaxy nie aktualizuje grafu od tego czasu, a Wikiverse doczytuje brakujące dane tylko wtedy, kiedy są potrzebne.



Rysunek 2.3. Wizualizacja połączeń w aplikacji Wikiverse [?]

### 2.3. Pozostałe rozwiązania

Wśród innych rozwiązań znaleźć można serwis Encartopedia [?], który rozkłada artykuły na przestrzeni dwuwymiarowej. Autor aplikacji otwarcie przyznaje, że jego inspiracją było WikiGalaxy. Aplikacja skupia się jednak bardziej na przeglądaniu treści w klasyczny sposób, uatrakcyjniając go prostą mapą z punktami i oznaczonymi połączonymi (po wskazaniu kursorem na link wychodzący z artykułu – rysunek 2.4). Źródłem danych, również w tym przypadku, jest pewien fragment anglojęzycznej Wikipedii.



Rysunek 2.4. Serwis Encartopedia z wybranym artykułem "Mango" i połączeniem do "Banana" [?]

Na koniec warto wspomnieć o małym skrypcie zatytułowanym wiki-graph [?], napisanym w języku Python. Generuje on rekurencyjnie dwuwymiarowy graf połączeń (wynik działania to plik obrazka). Możliwe jest zastosowanie skryptu do jakiejkolwiek Wikipedii, a dodatkowe parametry pozwalają dopracować wygląd grafu. Rozwiązanie te nie nadaje się jednak do tworzenia dużych struktur grafowych.

### 3. SPECYFIKACJA WYMAGAŃ PROJEKTU

MATEUSZ JANICKI

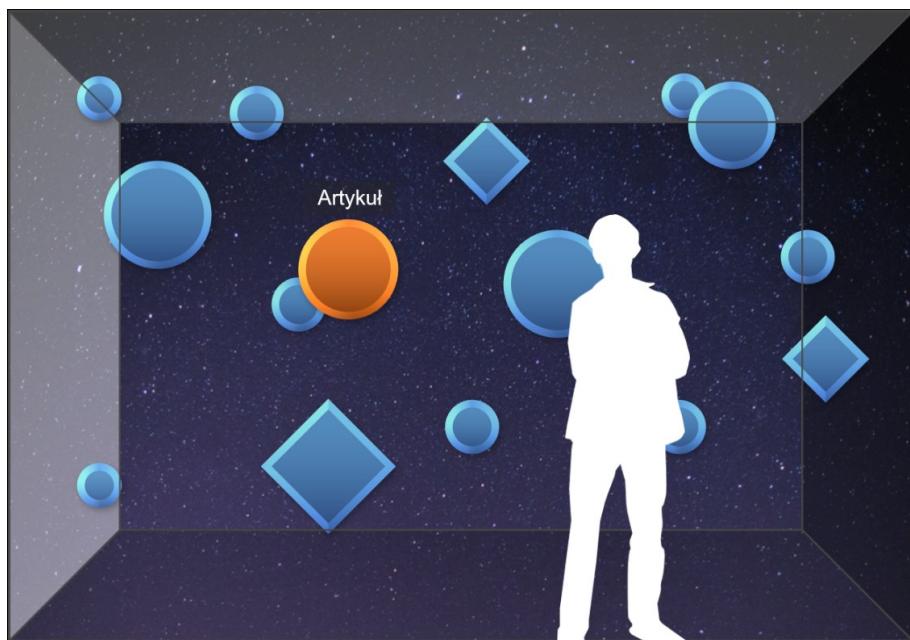
Ten rozdział obejmuje opis wymagań funkcjonalnych, jakościowych oraz technicznych projektu, a także kilka scenariuszy użycia i diagram UML przypadków użycia. Był przygotowywany przed rozpoczęciem prac nad aplikacją i zostanie zwalidowany i skomentowany w podsumowaniu pracy.

#### 3.1. *Wymagania funkcjonalne*

W treści każdego artykułu na Wikipedii znajdują się odnośniki prowadzące do innych artykułów. Na podstawie tej własności można stworzyć sieć połączeń pomiędzy wszystkimi stronami. Dodatkowo, każdy artykuł należy do kategorii określającej jego tematykę. Kategorie łączą się także pomiędzy sobą, określając swoje kategorie nadzędne i podrzędne.

Aplikacja opiera się na grafie stworzonym z węzłów reprezentujących artykuły i kategorie oraz połączeń określających związki pomiędzy nimi. Taka struktura jest przedstawiana użytkownikowi w kilku trybach, tak aby umożliwić interakcję z grafem na różne sposoby. Istnieją trzy widoki przeglądania danych: widok grafu, widok węzła oraz widok treści.

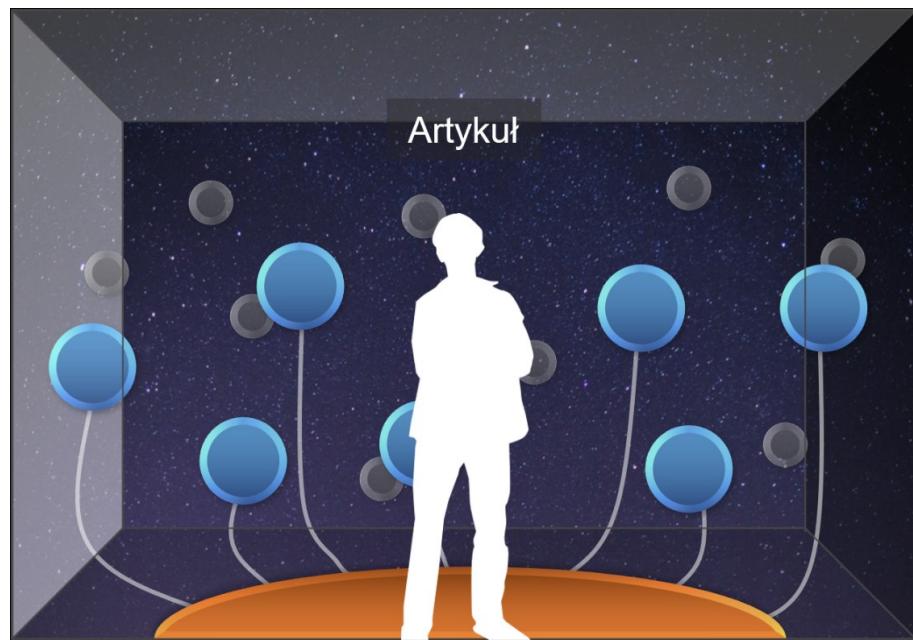
Widok grafu (rysunek 3.1) jest głównym widokiem aplikacji. W przestrzeni rozmieszczone są węzły w postaci punktów. Użytkownik może się swobodnie poruszać ("latać") we wszystkich kierunkach. Połączenia pomiędzy punktami nie są wyświetlane w celu zachowania przejrzystości wyświetlanych informacji. Punkty można wskazywać, a następnie wybierać. Po wykonaniu tej czynności otwierany jest widok węzła.



Rysunek 3.1. Prototyp widoku grafu

Widok węzła umieszcza wybrany artykuł lub kategorię na dolnym ekranie. Połączenia wychodzące z tego węzła rozłożone są na ścianach bocznych. Aby ułatwić wskazywanie połączonych węzłów, będą one pokazywane w formie otaczających użytkownika kopii punktów, zawieszonych na krzywych biegących od wybranego węzła do oryginalnego punktu. Użytkownik może zmienić typ wyświetla-

nych połączeń, z tych które prowadzą od wybranego węzła do innych, na połączenia które prowadzą do wybranego węzła od innych. Powyższy tryb został zilustrowany na rysunku 3.2.



Rysunek 3.2. Prototyp widoku węzła (przypadek widoku powiązań artykułów)

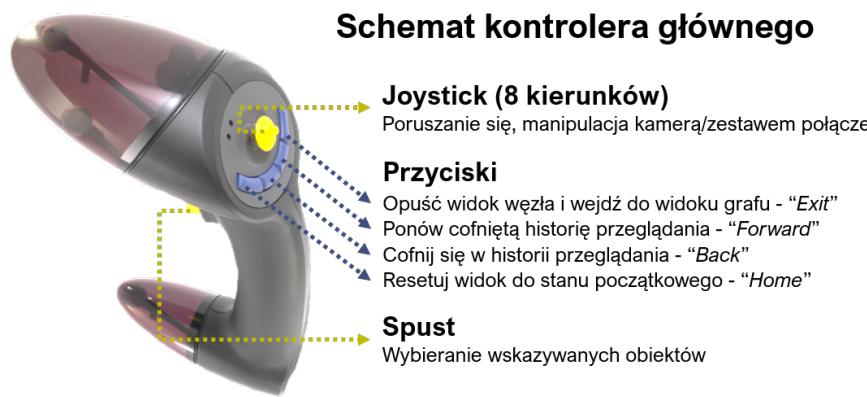
Aby nie przytłaczać użytkownika zbyt dużą ilością informacji, wyświetlana jednocześnie liczba połączeń jest ograniczona, a utworzone przedziały można przewijać. Po wybraniu połączonego węzła przenosimy się do nowo wybranego węzła. Możliwe jest cofanie się w historii przeglądania, a także ponowne wykonywanie cofniętej operacji. W tle widoczne są węzły z widoku grafu, jednak wyłączona jest możliwość ich wybierania w celu ułatwienia wskazywania powiązań. Nazwa wybranego węzła wyświetlana jest ponad poziomem głowy użytkownika zgodnie z kierunkiem w który on patrzy. Istnieje opcja automatycznego poruszania się pomiędzy węzłami, która ułatwia prezentację aplikacji. Polega ona na przechodzeniu do losowo wybranych węzłów powiązanych z aktualnie wybranym do momentu, w którym zostanie ona ręcznie zatrzymana. W każdym momencie możliwe jest opuszczenie widoku węzła i przejście do widoku grafu. Użytkownik w takim przypadku zostaje ustawiony obok opuszczonego węzła.

Do widoku treści można przejść z poziomu widoku węzła. Wyświetlane są w nim statystyki i informacje ogólne na temat aktualnie wybranej strony na Wikipedii. Dostępny jest również podgląd fragmentu treści w przypadku artykułu.

W widoku grafu i widoku węzła widoczna jest także oś czasu. Pokazuje ona datę, z której wyświetlany jest stan grafu. Zmienia ona swoje położenie na podstawie ruchów głowy użytkownika tak, aby zawsze znajdowała się nad nim. Zmiana czasu na osi powoduje pokazywanie tylko takich węzłów, które istniały w wybranym momencie (wraz z ich połączeniami).

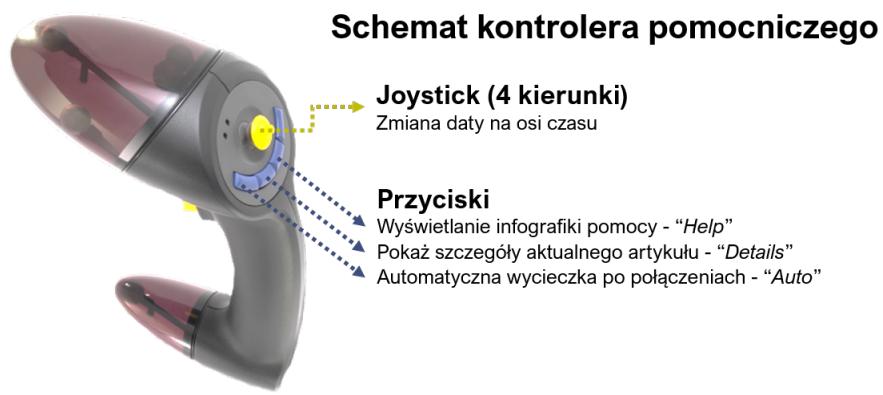
Aplikacja wymaga zastosowania dwóch kontrolerów ze względu na dużą liczbę interakcji dostępną dla użytkownika. Dla zapewnienia wyższej jakości obsługi istnieje opcja wyboru układu przycisków - praworęczny oraz leworęczny. Wybór ten możliwy jest przy starcie aplikacji. W momencie wyświetlania okna wyboru widoczna jest także infografika z informacjami o sterowaniu. Można otworzyć ją także w trakcie użytkowania aplikacji. Z pozycji infografiki można także zmienić tryb sterowania za pomocą wyświetlonego przycisku.

Za pomocą głównego kontrolera (rysunek 3.3) można wskazywać i wybierać węzły zarówno w widoku grafu, jak i w widoku węzła. Za pomocą joysticka w każdym widoku można dokonywać ruchu kamerą. W widoku grafu służy on także do swobodnego poruszania się, a w widoku węzła do przesuwania grup połączeń. Cztery pozostałe przyciski służą do resetowania widoku do stanu początkowego (przycisk *Home*), cofania się w historii przeglądania (przycisk *Back*), ponowienia cofniętej historii przeglądania (przycisk *Forward*) oraz opuszczenia widoku węzła i przejście do widoku grafu (przycisk *Exit*).



Rysunek 3.3. Schemat kontrolera głównego

Kontroler pomocniczy (rysunek 3.4) wykorzystuje joystick oraz trzy przyciski. Joystick służy do manipulowania osią czasu. Za jego pomocą można zmieniać datę. Pierwszy przycisk służy do automatycznej wycieczki po węzłach (przycisk *Auto*, ponowne jego naciśnięcie zatrzymuje wycieczkę), drugi - do wyświetlenia treści i statystyk wybranego artykułu (przycisk *Details*). Trzeci przycisk (przycisk *Help*) powoduje wyświetlenie pomocy z infografiką o sterowaniu, która jest dostępna w każdym widoku. Przyciski pierwszy i drugi są dostępne tylko w trybie widoku węzła.



Rysunek 3.4. Schemat kontrolera pomocniczego

### **3.2. Wymagania jakościowe**

**Niezawodność** Aplikacja nie może posiadać żadnych błędów, które powodowałyby zaburzenie imersji. Należą do nich wszelkie błędy graficzne, błędy w interfejsie, jak również nieprawidłowe umiejscowienie węzłów i połączeń w przestrzeni. Aplikacja musi mieć także dobrze zaprojektowaną i przemyślaną strukturę grafu, ponieważ stanowi ona rdzeń programu. Jakiekolwiek błędy z nią związane powodowałyby niezdolność do korzystania z aplikacji i niepowodzenie całego projektu. W przypadku wykrycia takich błędów powinny one uzyskać najwyższy priorytet i być naprawione w następnej wersji.

**Użyteczność** Aplikacja musi być estetyczna i wygodna w użyciu. Ważne jest zastosowanie nowoczesnych animacji i innowacyjnego interfejsu, pozwalającego zanurzyć się w wizualizacji. Użytkownik ma się poczuć, jakby naprawdę podróżował po stworzonym świecie. Interfejs musi wykorzystywać zalety jaskini rzeczywistości wirtualnej: możliwość ruchu użytkownika i śledzenie ruchów głowy. Aby zapewnić widok dla wielu osób maksymalnie niezależny od pozycji okularów wiodących, obiekty muszą znajdować się w odległości odpowiadającej pozycji ścian jaskini. Sterowanie aplikacją, mimo dużej ilości interakcji, powinno być intuicyjne, a w razie problemów musi istnieć pomoc dla użytkownika.

**Czas reakcji** Aplikacja, ze względu na swoją specyfikę, musi być przyjemna w odbiorze dla użytkownika. Czas przejścia z widoku grafu do widoku węzła lub odwrotnie oraz czas podróży pomiędzy węzłami nie powinien trwać dłużej niż 1,5 sekundy. Dopuszczalny jest dłuższy czas uruchamiania aplikacji ze względu na wymóg zbudowania grafu z pliku, jednak nie powinien on przekraczać 20 sekund, gdyż spowodowałby to obniżenie zadowolenia użytkownika i spowolniłby to prezentowanie możliwości jaskini odwiedzającym.

### **3.3. Wymagania techniczne**

Aby projekt został zaliczony, kluczowe jest jego poprawne działanie w środowisku jaskini rzeczywistości wirtualnej. Na komputerach obsługujących jaskinię działa aktualnie system Windows 7. Aplikacje wykorzystujące jaskinię zbudowane są w oparciu o środowisko Unity. Użycie najnowszych wersji platformy nie jest zalecane z powodu braku kompatybilności. Z tego powodu projekt jest realizowany w Unity 2018.1.9 oraz używa platformy .NET 4.x.

Do działania aplikacji są wymagane wcześniej przygotowane pliki danych, na podstawie których budowany jest graf artykułów i połączenia między nimi. Do realizacji funkcjonalności widoku treści wymagane połączenie z internetem - docelowo w Dużej Jaskini. Początkowo testy będą wykonywane w Małej Jaskini LZWP. Dopiero, gdy aplikacja będzie działała na niej bez problemów, możliwe będzie przetestowanie aplikacji w Średniej, a następnie w Dużej Jaskini.

Budowa aplikacji musi być zgodna z wymaganiami jaskini rzeczywistości wirtualnej. Aplikacja będzie uruchamiać się jednocześnie na wielu komputerach, dlatego dane muszą się aktualizować dostatecznie szybko, aby zachować płynność i zgodność wyświetlanych obrazów.

### **3.4. Scenariusze użycia**

#### **1. Przejście z węzła “Politechnika Gdańsk” do węzła “Gdańsk”**

W celu wędrówki po powiązanych artykułach należy najpierw znaleźć interesujący nasz węzeł - “Politechnika Gdańsk”. Do ruchu w widoku grafu użytkownik wykorzystuje joystick głównego kontrolera. Po odszukaniu węzła użytkownik wskazuje na niego kontrolerem i wybiera go za pomocą przycisku spustu. Po wybraniu wokół użytkownika pokażą się pierwsze alfabetycznie artykuły powiązane z aktualnie wybranym. W celu znalezienia artykułu “Gdańsk” należy przesunąć przedziały wyświetlanych powiązań, naciskając przycisk na kontrolerze, aż go znajdziemy. Po jego wskazaniu i wybraniu przeszliśmy do interesującego nas artykułu.

#### **2. Zmiana daty wyświetlanego grafu na 15-ty tydzień 2015 roku z daty 2-gi tydzień 2019 roku**

Aby zmienić datę, należy początkowo wybrać interesującą nas jednostkę czasu. W tym wypadku najlepiej zacząć od zmiany roku - wybieramy jednostkę za pomocą joysticka (przechylając go na lewo lub prawo) na kontrolerze pomocniczym. Również za pomocą joysticka (poruszając go w przód i w tył) cofamy się cztery lata w czasie. Po cofnięciu się w latach, możemy wybrać dokładniejszą jednostkę, jaką jest tydzień. Czas przewijamy trzykrotnie razy do przodu. Po zaakceptowaniu zmian następuje przebudowanie grafu. Po ukończeniu operacji widoczny jest graf zawierający artykuły istniejące w wybranym momencie czasowym.

#### **3. Wyświetlanie statystyk o artykule “Netflix”**

Do przeprowadzenia tej operacji wymagane jest połączenie z Internetem. Proces wyświetlania statystyk rozpoczynamy od znalezienia i wybrania węzła. Użytkownik porusza się w widoku grafu za pomocą joysticka głównego kontrolera, wskazuje na węzeł i wybiera go używając przycisku spustu. Następnie wystarczy przejść do widoku szczegółowego za pomocą przycisku *Details* na kontrolerze. Wokół użytkownika pojawi się okno podglądu szczegółowych informacji na temat artykułu, wraz z jego krótką treścią i obrazem.

#### **4. Przejście do artykułu należącego do tej samej kategorii co artykuł “Jan Matejko”**

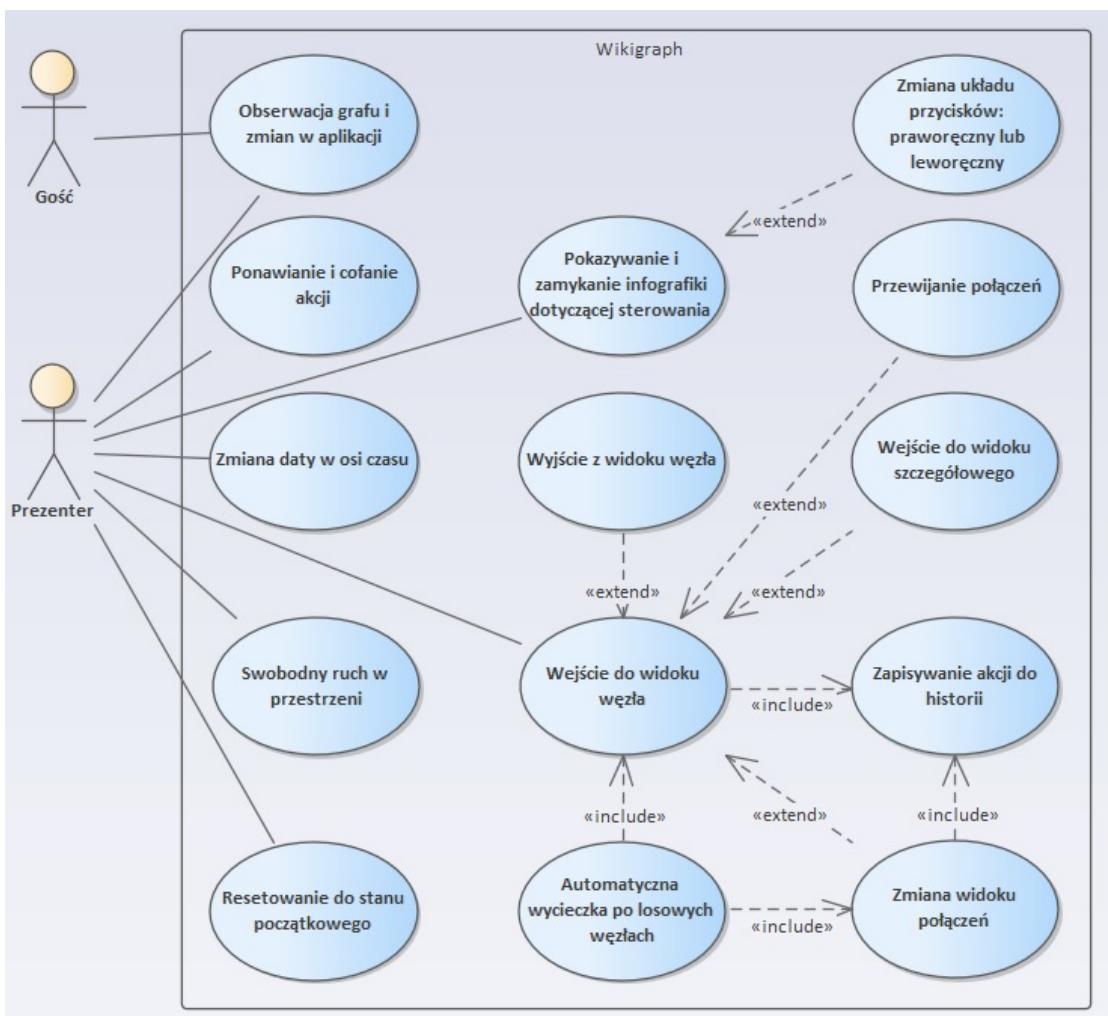
Po znalezieniu artykułu “Jan Matejko” w przestrzeni, należy go wskazać i wybrać za pomocą przycisku spustu. Wokół użytkownika pokażą się połączenia z innymi artykułami. W celu znalezienia kategorii do której należy wybrany artykuł, należy najpierw zmienić typ wyświetlanych połączeń na połączenia prowadzące do niego od innych węzłów, celując joystickiem w węzeł na którym stoi użytkownik i naciskając spust. Połączenia dookoła użytkownika zmienią się i należy zlokalizować dowolną kategorię, która będzie przedstawiona za pomocą innego kształtu geometrycznego. Jeżeli nie wyświetlają się żadne kategorie, należy przewinąć wyświetlane połączenia joystickiem w górę i w dół, aż jakaś kategoria się wyświetli. Należy w nią wycelować kontrolerem i nacisnąć spust, a stanie się ona aktualnie wybranym węzłem. Spomiędzy aktualnie wyświetlanych połączeń należy wybrać artykuł o tytule innym niż “Jan Matejko” i tak samo jak poprzednio wycelować w niego i nacisnąć spust. Jest to artykuł, który należy do tej samej kategorii co artykuł “Jan Matejko”.

## 5. Wyświetlenie pomocy przy sterowaniu i zmiana układu sterowania

Aby wyświetlić pomoc dotyczącą sterowania aplikacją, należy wcisnąć przycisk *Help* na kontrolerze pomocniczym. Ukaże się wtedy szczegółowa infografika z możliwościami sterowania, jakie oferują kontrolery. Jeśli nie odpowiada nam układ sterowania z powodu lewo- lub praworęczności, można go zmienić wciskając przycisk spustu podczas wyświetlania pomocy, określając tym samym nowy główny kontroler. Po opuszczeniu pomocy (klikając ponownie przycisk *Help*) zmiana układu kontrolerów zostanie zastosowana.

### 3.5. Diagram przypadków użycia

Na rysunku 3.5 znajduje się diagram UML przypadków użycia aplikacji opisujący jej podstawowe funkcjonalności. Zawiera on jeden, główny system, jakim jest aplikacja wizualizująca graf w jaskini, nazwana WikiGraph.



Rysunek 3.5. Diagram UML przypadków użycia

## **4. PRZYGOTOWYWANIE DANYCH WEJŚCIOWYCH**

Bardzo ważną częścią naszego projektu są dane. Zanim omówiona zostanie wizualizacja i integracja ze środowiskiem jaskini, warto określić podstawowe źródło danych, a także skupić się na procesie przygotowywania informacji do głównej aplikacji. Kolejny rozdział, opisujący jej implementację, będzie wykorzystywał stworzone na tym etapie dane wejściowe.

### **4.1. Źródło danych**

*MIKOŁAJ MIRKO*

Internetowa encyklopedia Wikipedia to jeden z projektów organizacji Wikimedia Foundation. Projekt ten ma na celu gromadzenie, porządkowanie i weryfikowanie otwartych danych tworzonych przez społeczność wolontariuszy (często nazywanych „Wikipedystami”). Według Podstawowego Rankingu Międzynarodowego [?] liczba artykułów w angielskiej wersji językowej na dzień 1 listopada 2019 r. wynosi prawie 6 milionów, a co miesiąc przybywa blisko 20 tysięcy nowych pojęć. Przechowywanie takiej ilości danych (uwzględniając całą treść i media artykułu, znajdujące się w nim powiązania wewnętrzne i zewnętrzne oraz historię jego edycji) jest nie lada wyzwaniem.

Wikimedia Foundation prowadzi również inne inicjatywy, takie jak m.in. Wikibooks (zbiór książek i podręczników), Wikinews (dziennik wydarzeń) i Wikiquote (kolekcja rozmaitych cytatów) – każda z nich posiadająca wiele wersji językowych. Wszystkie oferowane przez fundację serwisy opierają się na oprogramowaniu MediaWiki. Odpowiada ono za ogólną strukturę strony typu wiki, oferując jednocześnie wiele dodatkowych mechanizmów ułatwiających pracę z dużą ilością danych. Skonfigurowane zewnętrzne API pozwala na dostęp do danych innym oprogramowaniem, użycie szablonów stron ułatwia oddzielenie warstwy wizualnej od samych danych, a moduł archiwizacji odpowiada za tworzenie kopii zapasowych baz danych.

Ostatnia z przytoczonych funkcjonalności odgrywa dużą rolę w sposobie zdobycia danych do aplikacji. Kolejny projekt fundacji, o którym trzeba wspomnieć, to Meta-Wiki. Odpowiedzialny jest on za koordynację wszystkich pozostałych projektów. Zawiera bogatą dokumentację, historię zmian i aktualizacji oraz raporty aktywności. Udostępnia również publiczne zrzuty baz danych, zawierające część informacji z każdej dostępnej wersji językowej każdego projektu. Wykonywane są one z częstotliwością około 1 raz na 2 tygodnie i używają wspomnianego modułu archiwizacji (choć nie są typową kopią zapasową całego serwisu).

Wśród oferowanych danych w zrzutach można znaleźć przede wszystkim informacje o stronach (czyli artykułach, kategoriach, szablonach, przestrzeniach nazw i kilku innych), historii ich edycji oraz wewnętrznych połączeniach między stronami. Oprócz tego dostępne są różnego rodzaju listy, statystyki i metadane. Niektóre z nich dostępne są w formatach takich jak .txt, .json i .xml, ale większość z nich zapisana jest w postaci plików .sql, zawierających definicję tabeli bazy danych oraz ciąg wpisów z danymi. Wszystkie udostępniane pliki są dodatkowo zarchiwizowane za pomocą programu GZIP.

## 4.2. Pobieranie i dekompresja

MIKOŁAJ MIRKO

Serwis Wikimedia Downloads, oprócz skompresowanych plików zrzutów, posiada również plik o nazwie `index.json`. Zawiera on spis wszystkich ostatnio wykonanych operacji archiwizacyjnych na bazie danych. Na jego podstawie łatwo uzyskać bezpośrednie adresy URL do interesujących nas plików, a także dodatkowe informacje o ich rozmiarze i dacie stworzenia. Zawiera on również status przetwarzania każdej porcji danych – wszystkie pobierane pliki muszą być zakończone w ramach tego samego zrzutu, inaczej nie będą ze sobą zgodne, co uniemożliwi generowanie plików wejściowych aplikacji.

Listing 4.1 przedstawia przykładowy fragment pliku `index.json`.

```
1  "plwiki": {
2      "jobs": {
3          //...
4          "pagetable": {
5              "files": {
6                  "plwiki-20191101-page.sql.gz": {
7                      "size": 112182889,
8                      "md5": "d59ca88559792b2520f50368b4c3815a",
9                      "sha1": "49d16053f695cb27134cae278b1269c6e250445a",
10                     "url": "/plwiki/20191101/plwiki-20191101-page.sql.gz"
11                 }
12             },
13             "updated": "2019-11-02 09:03:12",
14             "status": "done"
15         },
16         //...
17     },
18     "version": "0.8"
19 }
```

Listing 4.1: Fragment informacji o ostatnim zrzucie bazy danych polskiej Wikipedii

Na potrzeby naszej aplikacji potrzebujemy danych z następujących trzech plików:

- `page.sql.gz` – posiada identyfikatory stron artykułów i kategorii, ich przestrzenie nazw oraz tytuły (reszta informacji nie jest wykorzystywana),
- `pagelinks.sql.gz` – zawiera spis wewnętrznych połączeń między artykułami - są to odnośniki znajdujące się w treści artykułu, prowadzące do powiązanych tematycznie innych artykułów,
- `categorylinks.sql.gz` – zawiera, analogiczny do poprzedniego pliku, spis połączeń między kategoriami oraz między artykułami a kategoriami.

Rozmiary wymienionych plików zależą od wielkości Wikipedii, z której pochodzą. Suma rozmiarów tych trzech plików dla angielskiej Wikipedii wynosi około 10GB, zaś dla Polskiej około 1GB. Istnieje także wiele innych, drobniejszych encyklopedii (w mniej popularnych językach oraz zawierających dane z innych projektów). Ich rozmiary mogą mieścić się w kilku megabajtach. Wielkość danych po dekompresji z formatu `.gz` jest w stanie wzrosnąć nawet dziesięciokrotnie.

### 4.3. Parsowanie informacji

MIKOŁAJ MIRKO

Po pobraniu i rozpakowaniu rozpoczyna się proces parsowania danych. Każdy z plików .sql składa się z definicji struktury tabeli bazy danych oraz listy wpisów w postaci ukazanej na listingu 4.2. Parsowanie informacji polega na przejściu po każdej linijce danego pliku, wydostaniu kolejnych wartości następujących po wyrażeniu VALUES i zapisaniu tylko tych, które są istotne dla dalszego przetwarzania.

```
1 INSERT INTO `page` `VALUES
2   (10,0,'AccessibleComputing','','1,0,0.33167112649574004,'20191003224230',
3    '20190105021557',854851586.94,'wikitext',NULL),
4   (12,0,'Anarchism','','0,0,0.786172332974311,'20191101063615','20191031183024'
5    ,923631615,104479,'wikitext',NULL),
6   (13,0,'AfghanistanHistory','','1,0,0.0621502865684687,'20191029091312',
7    '20190618192734',783865149.90,'wikitext',NULL),
8   ...
9 
```

Listing 4.2: Fragment pliku enwiki-20191101-page.sql zawierający dane o stronach

Plik page.sql zawiera nieposortowane informacje o różnych stronach Wikipedii. W celu odróżnienia strony artykułu od strony kategorii (wraz z informacjami o ich identyfikatorach i tytułach) używana jest druga wartość w ciągu pojedynczego wpisu, oznaczająca przestrzeń nazw strony. Według dokumentacji MediaWiki, liczba równa 0 oznacza typową stronę artykułu, a liczba 14 stronę typu kategoria. Na podstawie tego rozróżnienia tworzone są dwa nowe pliki zawierające 2-elementowe krotki, których pierwszym elementem jest identyfikator strony, a drugim jej tytuł.

Pliki pagelinks.sql i categorylinks.sql są parsowane w podobny sposób. Z każdego wpisu pobierany jest identyfikator artykułu lub kategorii, z którego połączenie wychodzi, oraz tytuł artykułu lub kategorii, do którego połączenie te kieruje. Zanim jednak informacje o połączeniach zostaną zapisane do osobnych plików, potrzebne jest przekształcenie tytułu (drugiego pobieranego parametru) do odpowiadającego identyfikatora strony, tak aby z postaci ID strony → Tytuł strony otrzymać postać ID strony → ID strony. To pozwoli na zmniejszenie wielkości pliku wynikowego i łatwiejszą do dalszego przetwarzania strukturę. Dodatkowo, dzięki informacji o pochodzeniu odnośnika w pliku categorylinks.sql, następuje podział zawieranych połączeń na te określające związek między dwiema kategoriami (tworzące strukturę hierarchiczną stron kategorii) oraz związek między artykułem a kategorią (przypisanie artykułu do kategorii).

Po wykonaniu wymienionych przekształceń otrzymywanych jest 5 nowych plików (oznaczonych rozszerzeniem .map), zawierających dane potrzebne do stworzenia struktury właściwego grafu. Wszystkie te pliki są dodatkowo sortowane po identyfikatorach w celu przyspieszenia kolejnego etapu ich przetwarzania. Wytworzone zostały:

- **page.map** – identyfikatory i tytuły artykułów,
- **category.map** – identyfikatory i tytuły kategorii,
- **pagelinks.map** – identyfikatory artykułów i odpowiadające im listy identyfikatorów artykułów, do których prowadzą odnośniki znajdujące się w ich treści,
- **categorylinksfromcategory.map** – analogicznie wyglądający spis połączeń między kategoriami,
- **categorylinksfrompage.map** – analogicznie wyglądający spis połączeń między stronami artykułów a stronami kategorii.

Przykład zastosowanych struktur w plikach zawierających tytuły stron ilustruje rysunek 4.1, a w plikach połączeń rysunek 4.2. Widoczne dane zostały stworzone na podstawie Wikipedii "simplewiki" w dniu 20 października 2019 r. Kolorem niebieskim oznaczony został artykuł o ID 48 zatytułowany "Astronomy". Wśród jego połączeń do innych artykułów znajduje się oznaczony na pomarańczowo artykuł o ID 51. Jest to strona o nazwie "Asteroid". Rysunek 4.3 to wycinek ekranu prezentujący artykuł "Astronomy" w Wikipedii "simplewiki". Data wykonania tego zrzutu ekranu to 24 października 2019 r. Można zauważyć, że jednym z jego odnośników to faktycznie artykuł "Asteroid". Jest to również siódmy link, licząc od początku treści artykułu, zarówno w pliku pagelinks.map, jak i na stronie internetowej Wikipedii (zachowana jest ich kolejność).

47	Atom
48	Astronomy
49	Architecture
50	Anatomy
51	Asteroid
52	Afghanistan
53	Angola
54	Argentina

Rysunek 4.1. Fragment pliku page.map zawierający tytuły artykułów

47	2138, 216276, 177079, 84829, 215781, 2126, 238508, 6478, 21424
48	80978, 4492, 15034, 38378, 249671, 361313, 51, 29179, 9235, 501
49	21739, 375627, 1942, 15034, 16174, 24298, 13221, 13217, 6, 1111
50	123820, 15034, 40647, 5771, 2950, 653450, 15304, 226028, 13451
51	82184, 75821, 167688, 167555, 287644, 170029, 289045, 67404, 1
52	184396, 338677, 445698, 338656, 338463, 339588, 301977, 50608
53	341047, 184402, 1942, 28093, 5404, 649501, 340769, 298518, 171
54	342254, 184405, 227444, 7726, 299314, 172928, 342181, 342195,

Rysunek 4.2. Fragment pliku pagelinks.map z artykułami i ich połączniami

## Astronomy

From Wikipedia, the free encyclopedia

**Astronomy** (from the Greek *astron* (ἀστρον) meaning "star" and *nomos* (νόμος) meaning "law") is the scientific study of celestial bodies.

The objects studied include stars, galaxies, planets, moons, asteroids, comets and nebulae. Phenomena outside the Earth's atmosphere are also studied. That includes supernovae explosions, gamma ray bursts, and cosmic microwave background radiation). Astronomy concerns the development, physics, chemistry, meteorology and movement of celestial bodies, as well as the structure and development of the Universe.

Rysunek 4.3. Zrzut ekranu artykułu zatytułowanego "Astronomy"

## **4.4. Tworzenie struktury grafu**

JAN KRUCZYŃSKI

Celem tego etapu jest wytworzenie plików, na których operuje aplikacja. Podczas projektowania ich struktury wzięto pod uwagę, że muszą one:

1. Zawierać w sobie pełną informację na temat struktury skierowanego grafu połączeń pomiędzy kolejnymi węzłami.
2. Rozróżniać, czy dany węzeł reprezentuje artykuł czy kategorię.
3. Przechowywać tytuły z Wikipedii każdego węzła.
4. Przechowywać ID strony na Wikipedii.
5. Umożliwiać szybkie odnajdywanie interesujących nas danych o konkretnym węźle bez przeszukiwania wszystkich plików za każdym razem, gdy potrzebujemy wydobyć jakąś informację.
6. Przechowywać dane w skompresowanej formie - bez zbędnych bajtów.

Struktura owych plików powinna dać możliwość funkcjonowania aplikacji bez trzymania wszystkich danych w pamięci tymczasowej. Odczytywanie danych z plików powinno być możliwie najszybsze.

### **4.4.1. Generowanie brakujących danych**

Dane linków zawierają wyłącznie połączenia typu "z węzła - do innego węzła", ale nie mają połączeń odwrotnych "do węzła z innych węzłów". Aby aplikacja była w stanie zaprezentować pełny skierowany graf połączeń potrzebne jest odwzorowanie odwrotne. Do finalnego, poprawnego generowania plików potrzebne są dodatkowe pliki:

- Artykuły, które prowadzą do konkretnego artykułu (odwrotność pagelinks.map)
- Kategorie, do których należy dany artykuł (odwrotność categorylinksfrompage.map)
- Kategorie, do których należy dana kategoria (odwrotność categorylinksfromcategory.map)

Dane, które są potrzebne, nie znajdują się w plikach SQL, lecz dzięki opisany w sekcji 4.3 plikom .map, istnieje możliwość generacji na ich podstawie brakujących informacji. Aby to zrobić, należy odczytać interesujący plik .map i linia po linii wypełnić słownik o następującej formie:

```
1 Dictionary<int , List<string>> reverseMap = new Dictionary<int , List<string>>();
```

Listing 4.3: Słownik przechowujący odwzorowanie odwrotne

Jako że pliki .map są uporządkowane według ID, aby osiągnąć ten sam efekt, słownik zdefiniowany w listingu 4.3 został umieszczony w strukturze SortedDictionary, a następnie zapisany w pliku o odpowiedniej nazwie.

Klucz w słowniku to ID Wikipedii danego artykułu lub kategorii (w zależności od pliku .map, który przetwarzamy), typu int, aby umożliwić łatwe sortowanie. Wartość danego klucza to lista powiązanych połączeń, analogiczna do pliku źródłowego .map. Jako że nie ma potrzeby rzutowania na typ numeryczny - odczyt i zapis operuje na typie string - lista przechowuje właśnie takie wartości. Przykład odwzorowania odwrotnego widać na listingach 4.4 i 4.5.

Po zakończeniu tego etapu zostały wygenerowane następujące pliki:

- R\_pagelinks.map
- R\_categorylinksfrompage.map
- R\_categorylinksfromcategory.map

1	12,18,20
2	4,11,12,13
3	11,12,13,17,20
4	19,11
5	33,13,20
6	41,12,13

Listing 4.4. Przykładowy fragment pliku pagelinks.map

1	11,4,19
2	12,1,4,11,41
3	13,4,11,33,41
4	17,11
5	18,1
6	20,1,11,33

Listing 4.5. Odwzorowanie odwrotne z listingu 4.4 (fragment R\_pagelinks.map)

#### 4.4.2. Opis poszczególnych plików

Aby osiągnąć postawione wymagania, utworzono strukturę rozbitą na 5 plików. Wszystkie posiadają tą samą nazwę. Jest nią nazwa wersji Wikipedii, którą opisują (np. "simplewiki" lub "plwiki"). Różnią się rozszerzeniami, gdyż każdy plik posiada inną strukturę.

**Plik mapy .wgm** Jest to plik instruujący aplikację, na którym miejscu w innych plikach odnajdzie interesujące dane. Każdy węzeł zawiera swój wpis w pliku .map zajmując dokładnie 12 bajtów o strukturze opisanej w tablicy 4.1. Offset informuje, od którego bajtu w danym pliku możemy odczytywać informację o danym węźle.

4 bajty	4 bajty	4 bajty
Offset w pliku .wgg	Offset w pliku .wgt	ID Wikipedii

Tablica 4.1. Reprezentacja pojedynczego węzła w pliku .wgm

W tym miejscu następuje swoiste przekonwertowanie ID Wikipedii danej strony na nowy ID, którym jest numer w kolejności danego węzła w pliku .map. W aplikacji oraz w pozostałych plikach, gdy znajduje się odniesienie do jakiegoś węzła, użyto nie jego faktycznego ID Wikipedii, ale nowo utworzony identyfikator, rozpoczynający się od zera. Znając go można dokonać mnożenia przez rozmiar każdego wpisu (12) i otrzymać offset w pliku .wgm.

Podczas konstrukcji plików informacje o danym węźle są jednocześnie umieszczane we wszystkich plikach. Dzięki temu nie ma potrzeby przechowywać danej informującej ile bajtów należy odczytać. Odczytu należy dokonać aż do pozycji offset, który znajduje się w następnym węźle z pliku .wgm, czyli 12 bajtów dalej.

**Plik struktury grafu .wgg** Jest to plik zawierający dane połączeń skierowanego grafu. Rozróżniane są połączenia: #1 od których węzłów można dojść do aktualnego węzła oraz #2 do których węzłów prowadzi aktualny węzeł. Struktura użyta w tym pliku jest opisana w tablicy 4.2.

3 bajty	#1: 3 bajty * N	#2: 3 bajty * X
N: Ilość połączeń typu #1	a <sub>1</sub> , b <sub>2</sub> , c <sub>3</sub> , ... z <sub>N</sub>	A <sub>1</sub> , B <sub>2</sub> , C <sub>3</sub> , ... Z <sub>X</sub>

Tablica 4.2. Reprezentacja pojedynczego węzła w pliku .wgg

A<sub>1</sub>, B<sub>2</sub>, C<sub>3</sub> oraz a<sub>1</sub>, b<sub>2</sub>, c<sub>3</sub> to nie są ID artykułów Wikipedii, lecz informacje, który z kolei artykułu z pliku .wgm mamy na myśli. Jako że ilość węzłów nigdy nie przekracza 2<sup>24</sup>, 3 bajty wystarczają na przekazanie tej informacji.

**Plik informacyjny .wgi** Zawiera wyłącznie jedną, 4-bajtową liczbę typu `int` - jest to ilość artykułów, które znajdują się w plikach. Jest to jedyne rozróżnienie dla aplikacji, który węzeł traktować jako artykuł, a który jako kategorię - w pozostałych plikach nie ma pomiędzy nimi rozróżnienia. Do plików najpierw są zapisywane wszystkie artykuły, dzięki czemu przy pobieraniu danych o węźle, aplikacja oznacza ów węzeł jako kategorię, gdy jego numer w pliku `.wgm` jest większy niż wartość w pliku `.wgi`.

**Plik tytułów .wgt** Zawiera w sobie zapisane w formacie UTF-8 tytuły wszystkich artykułów i kategorii.

**Plik odwzorowań posortowanych tytułów .wgs** Aby umożliwić szybkie działanie wyszukiwarki, utworzono oddzielnego pliku o prostej do przeszukiwania strukturze (tablica 4.3). Zawiera on posortowane alfabetycznie wszystkie tytuły - zarówno kategorie jak i artykuły, zakodowane w formacie UTF-8. Aby ułatwić przeszukiwanie, każdy węzeł jest reprezentowany przez oddzielnego wiersza o formacie:

Tytuł węzła	";"	Numer węzła (od 0) w pliku <code>.wgm</code>	"\n"
-------------	-----	--	------

Tablica 4.3. Reprezentacja pojedynczego węzła w pliku `.wgs`

#### 4.4.3. Metoda generowania plików

Program generujący pliki `.wgX` został napisany w języku C#. Program zawiera kilka pomocniczych struktur ułatwiających konstrukcję wynikowych plików:

```
1 Dictionary<int, int> pageMap;
2 Dictionary<int, int> categoryMap;
3 List<string, int> sortedTitles;
4
5 public class GraphObject {
6     public bool isArticle;
7     public int id;
8     public string title;
9     public int offsetTitle;
10    public int offsetGraph;
11    public int order;
12 }
```

Listing 4.6: Pomocnicze struktury dla programu generującego pliki dla aplikacji

Na listingu 4.6 linijki 1 i 2 to mapy przechowujące odwzorowanie ID z Wikipedii (7) (który reprezentuje dany węzeł w plikach `.map`) na numer w kolejności węzła w pliku `.wgm` (nowo utworzony ID (11)). Jako że kategorie oraz artykuły mogą posiadać taki sam ID Wikipedii, potrzebne na to są dwie oddzielne struktury. Konstrukcja w wierszu 3 stanowi listę przechowującą krótki tytuł węzła i odwzorowania ID do późniejszej generacji posortowanych tytułów (wartości z linii 8 i 11)

Każdy węzeł podczas przetwarzania jest traktowany jako `GraphObject` i zawiera w sobie informacje o tytule na Wikipedii (8), liczbie informującej od którego bajtu, w pliku tytułów `.wgt` (9) oraz strukturze grafu `.wgg` (10) zostanie zapisana ta informacja oraz o tym, czy jest artykułem czy kategorią (6).

**Operacje przygotowujące** Zanim program rozpoczęte generację właściwych plików `.wgX`, potrzebuje dokonać generacji. W pierwszej kolejności program wywołuje metodę generującą odwzorowania odwrotne dla każdego z wymagających tego trzech plików, zgodnie z metodą opisaną w rozdziale 4.4.1.

Następnie program odczytuje tytuły wszystkich artykułów w linia po linii zapełnia mapę `pageMap` (1) oraz `sortedTitles` (3). Po tym etapie odczytuje tytuły kategorii i wypełnia `categoryMap` oraz dalej

uzupełnia listę sortedTitles w analogiczny sposób. Iteracja jest jednorazowa po wszystkich tytułach, więc złożoność obliczeniowa tego etapu to  $O(n)$ .

Dzięki zapisaniu mapy odwzorowań tytułów (3) w krotkach, sortowanie możliwe jest do realizacji metodą `Array.Sort()` z własną implementacją porównania elementów `IComparer<T>`. Metoda sortująca dostarczona jest przez samą technologię .NET, a algorytm sortujący to implementacja algorytmu QuickSort, który ma średnią złożoność obliczeniową  $O(n \log n)$ .

Następnie program odczytuje wszystkie elementy z posortowanej listy i zapisuje je pojedynczo do pliku `.wgs` dla tytułów artykułów oraz `sortedCategoryTitles.map` dla kategorii. Następuje iteracja po każdym elemencie listy, co oznacza złożoność obliczeniową  $O(n)$ . Po dokonaniu zapisu mapy tytułów nie są już potrzebne w dalszej części programu, więc następuje ich usunięcie z pamięci.

Uznając ilość artykułów za  $n$ , a ilość kategorii za  $m$ , łączna złożoność obliczeniowa algorytmów etapu operacji, przygotowujących generację plików, została przedstawiona we wzorze 4.1.

$$O(2n) + O(n \log n) + O(2m) + O(m \log m) = O(n \log n + m \log m) \quad (4.1)$$

**Generacja plików Wikigraphu** Kolejnym etapem jest już faktyczne zapisanie danych o węźle do wszystkich plików. Liczba artykułów jest już znana - wystarczy policzyć elementy w mapie `pageMap` (linia 1 w listingu 4.6) - i zapisać ją do pliku `.wgi`.

Aby zachować taką samą kolejność węzłów, ponownie dokonany jest odczyt pliku tytułów dla artykułów i dla każdego węzła parsowane są informacje do struktury reprezentującej dany element w programie (listing 4.6). Program jest napisany w postaci klasy zawierającej wszystkie metody zapisu i odczytu danych, co umożliwia łatwe przekazywanie zmiennych pomiędzy metodami, bez konieczności przekazywania wszystkiego w argumentach.

Plik z tytułami zawiera wszystkie interesujące nas elementy. Nie istnieje element w innych plikach, który nie posiada swojej reprezentacji w pliku z tytułami (element z pliku z tytułami może natomiast nie zawierać wpisu w plikach połączeń). Program przechowuje w głównej klasie numery linii dla każdego pliku z połączaniami, które powinniśmy aktualnie odczytać.

1. Podczas przetwarzania artykułu odczytujemy pliki

- `pagelinks.map`
- `R_pagelinks.map`
- `categorylinksfrompage.map`

2. Podczas przetwarzania kategorii odczytujemy pliki

- `R_categorylinksfromcategory.map`
- `categorylinksfromcategory.map`
- `R_categorylinksfrompage.map`

Przy odczytaniu wartości ID w pliku tytułów, odczytywane są także pojedyncze linie w powyższych plikach `.map`. Jeżeli ID w pliku z połączaniami odpowiada ID w pliku tytułów, te połączenia są uwzględniane do aktualnego węzła, a licznik odczytu linii dla danego pliku jest zwiększany. Jeżeli w pliku połączzeń nie znaleziono wartości ID z pliku tytułów, dany artykuł bądź kategoria nie zawiera w tym pliku `.map` żadnych połączeń, a licznik dla tego pliku nie jest zwiększany.

Posiadając wszystkie informacje o danym węźle, używając klasy `BinaryWriter` dane zostają dopisane do plików `.wgg` i `.wgt` zgodnie z ich strukturą opisaną w rozdziale 4.4.2. Ilość dopisanych bajtów do każdego z tych plików jest dodawana do odpowiednich liczników, a ich wartości zostają wraz

z ID z Wikipedii dopisane do pliku `.wgm`. Całość jest powtarzana dla każdego z artykułów, a następnie dla każdej z kategorii, zgodnie z plikiem z tytułem.

Ponieważ kategorie znajdują się na końcu pliku `.wgm`, w momencie odczytywania ich kolejności w pliku `.wgm` z mapy odwzorowań (listing 4.6 linia 2) do tej wartości dodawana jest ilość artykułów, czyli liczba zapisana w pliku `.wgi`. Przykładowo pierwsza kategoria będzie po ostatnim artykule, więc jeżeli jej ID oryginalnie wynosiło 0 (jako pierwsza w kolejności), jej nowy ID z mapy odwzorowań będzie równe ilości artykułów.

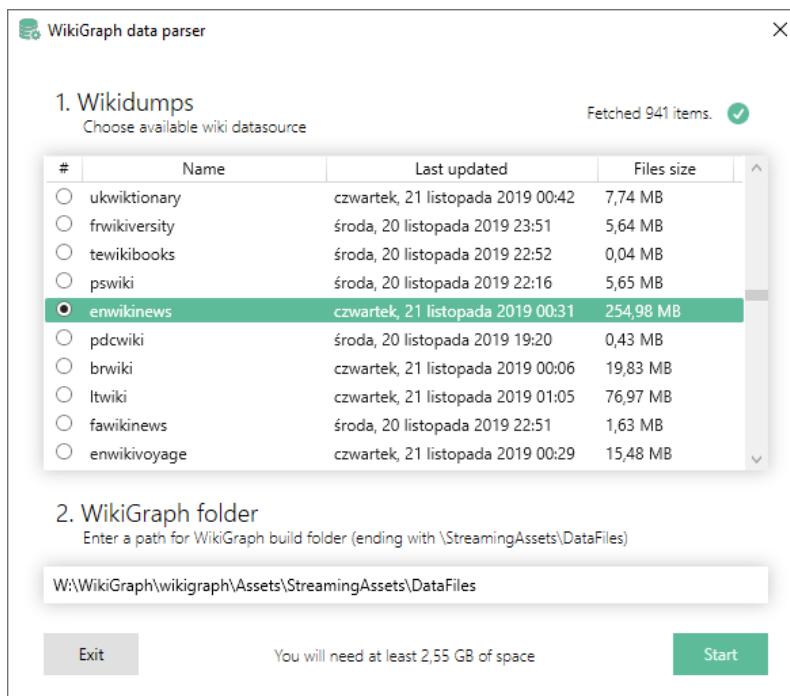
Złożoność obliczeniowa tego etapu, ponieważ następuje tutaj wyłącznie pojedyncza iteracja po każdym artykule i kategorii, wynosi  $O(n + m)$ , gdzie  $n$  to ilość artykułów, a  $m$  to ilość kategorii. Główna czasochłonność pochodzi z czasu otwierania i zamykania strumieni dostępu do plików oraz odczytywania danych.

#### 4.5. Narzędzie *WikiGraph Parser*

MIKOŁAJ MIRKO

Proces pozyskiwania danych opisany w podrozdziałach 4.1 - 4.4 został zautomatyzowany i zaimplementowany w postaci dodatkowego, pomocniczego programu. Aplikacja WikiGraph Parser ma na celu usprawnienie pracy z pozyskiwanymi danymi oraz zmniejszenie ryzyka wystąpienia nieprawidłowości. Aplikacja została napisana w języku C# (z platformą docelową .NET Framework 4.6.1) i posiada interfejs graficzny stworzony za pomocą framework'a UI WPF.

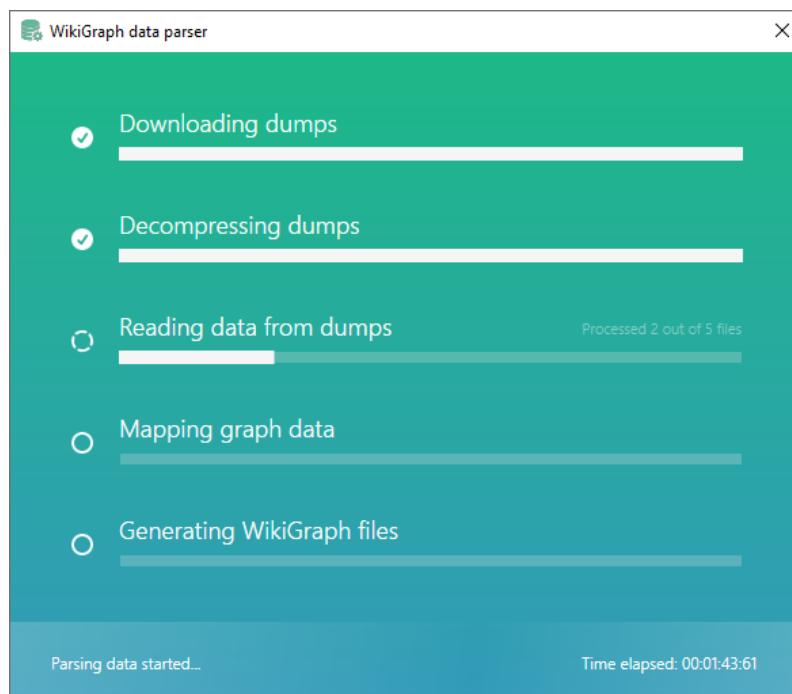
Interfejs programu został zaprojektowany z myślą o 10 heurystykach Nielsena [?]. Zastosowano estetyczny i minimalistyczny wygląd wykorzystujący typowe standardy aplikacji okienkowych. W wielu miejscach użytkownik jest informowany o stanie programu za pomocą etykiet oraz symboli graficznych. Walidacja pozwala uniknąć przewidywalnych błędów, a w razie wystąpienia nieoczekiwanych problemów pojawiają się opisowe okna dialogowe. Pomoc zawarta jest w postaci tytułów sekcji i ich podpowiedzi.



Rysunek 4.4. Ekran konfiguracji parametrów WikiGraph Parser-a

Użytkownik obsługujący WikiGraph Parser ma możliwość wyboru interesującego go źródła informacji oraz określenie ścieżki aplikacji, wykorzystującej stworzone zasoby (rysunek 4.4). Ograniczając go do wyboru tylko tych dwóch parametrów, tworzony jest prosty interfejs, który zachowuje jednocześnie swobodę i kontrolę użytkowania. Giles Colborne w swojej książce "Prostota i użyteczność" [?] opisuje zasadę zachowania złożoności, według której zabieg usuwania i ukrywania funkcjonalności przed użytkownikiem sprowadza ją do jego minimalnego poziomu złożoności.

" Cała sztuka projektowania prostych rozwiązań polega na przenoszeniu złożoności w odpowiednie miejsce, tak aby korzystanie z samego narzędzia było łatwe. " [?]



Rysunek 4.5. Ekran statusu postępu przetwarzania danych

Przetwarzanie danych jest rozpoczęte po naciśnięciu przycisku *Start*. Na ekranie z informacją o aktualnym postępie (rysunek 4.5) można wyróżnić pięć kroków. Są to:

- **Downloading dumps** Na podstawie wybranego zrzutu bazy pobierany jest odpowiedni zestaw plików. Do tego zadania wykorzystana jest klasa `WebClient`.
- **Decompressing dumps** Dekompresja pobranych plików odbywa się za pomocą klasy `GzipStream`. Po tym kroku pliki `.sql` gotowe są do przetwarzania.
- **Reading data from dumps** Podczas tego kroku tworzone są pliki z rozszerzeniem `.map`, zawierające tylko te dane, które są wykorzystywane do dalszego przetwarzania. Na tym etapie odbywa się również sortowanie danych – użyty jest pakiet `Sortiously`<sup>1</sup>.
- **Mapping graph data** Następnie przetwarzane są przerobione i uporządkowane dane uzupełniające je o brakujące odwrotne połączenia. Problem ten został opisany w sekcji 4.4.1.
- **Generating WikiGraph files** Na koniec tworzone są pliki grafu o strukturze binarnej, czytane przez aplikację główną. Wykorzystane są m.in. klasy takie jak `BinaryWriter` oraz `BinaryConverter`. Po zakończeniu tymczasowe pliki zostają usunięte.

<sup>1</sup>Pakiet użytkownika fredgdaley2 dostępny jest w serwisie GitHub: <https://github.com/fredgdaley2/Sortiously>

## 5. IMPLEMENTACJA GŁÓWNEJ APLIKACJI WIKIGRAPH

W tym rozdziale opisana zostanie implementacja aplikacji wizualizującej graf połączeń. Zostanie omówiony przyjęty model danych oraz sposób ich wczytywania. Następnie rozwinięta zostanie kwestia reprezentacji węzłów i połączeń oraz integracji aplikacji ze środowiskiem jaskini. Na koniec zostaną zestawione opisy pozostałych elementów interfejsu i funkcjonalności.

### 5.1. Model danych

STANISŁAW GÓRA

Model danych aplikacji tworzony jest na podstawie plików opisanych w sekcji 4.4. Są one otwierane jako strumienie używając klasy `FileStream`, co pozwala na jednakowy dostęp do każdej części pliku.

Listing 5.1 zawiera używany przez aplikację model węzła. Przy żądaniu załadowania do pamięci węzła o konkretnym numerze (wiersz 5 na listingu 5.1) najpierw odczytywane są informacje z pliku mapy, a następnie na ich podstawie wczytywany jest tytuł (8) oraz połączenia węzła (2 i 3). Określany jest też typ węzła (10) oraz jego identyfikator przypisany przez Wikipedię (6). Na koniec nowo wczytanemu węzlowi zostaje przypisany stan (11) aktywny.

```
1 public class Node {
2     public uint[] Children;
3     public uint[] Parents;
4
5     public readonly uint ID;
6     public uint WikiID;
7
8     public string Title;
9
10    public NodeType Type;
11    public NodeState State;
12    ...
13 }
```

Listing 5.1: Model węzła grafu

Załadowane węzły są dodawane jako wartości w mapie `ID → Node`, co pozwala na późniejsze szybkie uzyskanie modelu na podstawie jego identyfikatora. Przechowywanie grafu w pamięci zostanie szczegółowo opisane w sekcji 5.2.

W wyniku interakcji użytkownika z węzłem pokazywane są połączenia. Zostały one zamodelowane jako zbiór par węzłów. Przy tym przyjmujemy, że połączenie zaczyna się w węźle z którym nastąpiła interakcja.

### 5.2. Wizualna reprezentacja grafu

STANISŁAW GÓRA

Graf jest przechowywany w pamięci aplikacji jako zbiór map:

- **Identyfikator węzła → Model węzła**

Opisana już mapa warstwy modelu

- **Model węzła → Obiekt wizualny węzła**

Przechowuje widoczne aktualnie w aplikacji węzły jako wartości przypisane do modelu reprezentowanego węzła. Obiekt wizualny przechowuje numer ID węzła, co zamyka pętlę powiązań

i w efekcie pozwala na uzyskanie informacji o węźle (z jego modelu), mając na wejściu jego obiekt wizualny na podstawie dwóch map.

- **Model połączenia → Obiekt wizualny połączenia**

Mapa analogiczna do poprzedniej, przechowująca informacje o połączeniach między węzłami. Podobnie obiekt wizualny połączenia przechowuje dwa identyfikatory węzłów które łączy.

- **Model połączenia → Obiekt wizualny reprezentacji węzła końcowego połączenia**

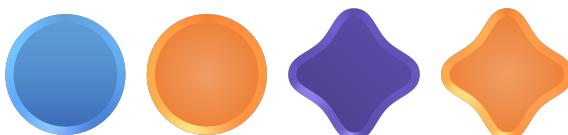
Mapa przechowująca pomocnicze reprezentacje węzłów końcowych połączenia opisane szczegółowo w sekcji 5.3.

#### 5.2.1. *Reprezentacja węzła*

Z uwagi na dużą ilość jednocześnie załadowanych węzłów (kilka do kilkunastu tysięcy na raz) konieczne okazały się pewne optymalizacje, poprawiające szybkość działania aplikacji.

Obiekty węzłów nie są w większości przypadków tworzone dynamicznie. Zamiast tego przy starcie aplikacji tworzona jest pula nieużywanych obiektów. W momencie ładowania węzła nowy obiekt wizualny tworzony jest tylko w przypadku, kiedy pula została wyczerpana. Przy usuwaniu węzła jego reprezentacja nie jest niszczona, tylko zwracana do puli w celu przyszłego wykorzystania.

W celu odciążenia karty graficznej zrezygnowaliśmy z wyświetlania każdego węzła jako trójwymiarowego modelu (jak na przykład kuli). Reprezentacją węzła jest prosta grafika rastrowa (rysunek 5.1). Dzięki temu siatka przetwarzana przez komputer jest dużo prostsza, ponieważ zawiera tylko cztery wierzchołki (dwa trójkąty) - jest to kwadratowa płaszczyzna z nałożoną tekturem. Dzięki temu aplikacja jest w stanie wyświetlić więcej węzłów na raz bez utraty płynności działania.

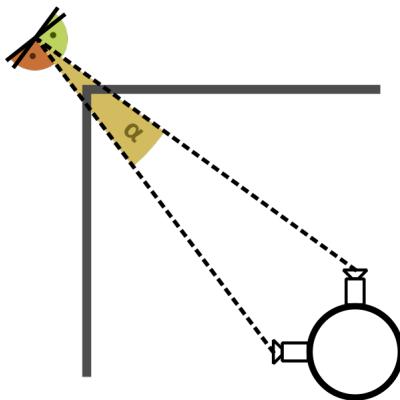


Rysunek 5.1. Grafiki węzłów (od lewej): artykuł aktywny i wskazany, kategoria aktywna i wskazana

Takie rozwiązanie powoduje jednak pewną komplikację. Płaszczyzna z grafiką, w przeciwieństwie do przestrzennego modelu, jest dobrze widoczna tylko kiedy kamera jest ustawiona bezpośrednio przed nią i zwrócona w jej stronę. Pojawia się więc potrzeba obracania obiektu węzła tak, aby był on zawsze zwrócony przodem do użytkownika zwiedzającego graf. Najprostszym rozwiązaniem byłoby dodanie krótkiego skryptu do każdego obiektu, który przy każdym odświeżeniu ekranu obracałby reprezentację węzła w stronę kamery. Nie jest to jednak dobre rozwiązanie z powodu narzutu czasowego. W każdej klatce ta sama operacja byłaby wykonywana potencjalnie kilkanaście tysięcy razy, co mogłoby mieć widoczny efekt na szybkości działania aplikacji.

Znacznie lepszym rozwiązaniem jest użycie shadera. W przeciwieństwie do standardowych skrypcji aplikacji jego działanie jest zrównoleglone przez kartę graficzną dla potencjalnie każdego przetwarzanego wierzchołka. W środowisku Unity do pisania Shaderów wykorzystuje się język ShaderLab oraz HLSL/Cg [?]. Podczas pierwszych testów tego rozwiązania pojawił się jednak kolejny problem. Powszechnie stosowane są tak zwane "Billboard Shadery", które służą dokładnie do interesującego nas celu, jednak w tym szczególnym przypadku aplikacji budowanej na środowisko LZWP są one niewystarczające.

**Problem** Każda ściana jaskini rzeczywistości wirtualnej jest kontrolowana przez osobny komputer z własną instancją aplikacji, posiadającą swoją parę kamer (dla każdego oka) skierowaną w kierunku danej ściany w przestrzeni wizualizacji. Ze względu na wymagania stereoskopii oraz spójności obrazów na każdej ze ścian, kamery te nie znajdują się w dokładnie tym samym punkcie w przestrzeni symulacji. Są od siebie lekko oddalone (rozstaw oczu dla każdej ściany oraz obrót głowy dla każdej pary przylegających ścian). Przykład został zilustrowany na rysunku 5.2. Billboard Shader odwraca obiekty węzłów tak, aby były zwrócone przodem do punktu, w którym znajduje się kamera na lokalnej ścianie. Powoduje to, że jeśli dany węzeł znajdzie się w pozycji wyświetlanej na przecięciu ekranów (na przykład połowa węzła na jednej a połowa na drugiej ścianie jaskini), na każdym z odpowiadających za ściany komputerów zostanie obrócony w nieco innym kierunku. Różnica kątów obrotu  $\alpha$  będzie tu odpowiadała odległości kątowej pomiędzy położeniem kamer obu ścian z perspektywy węzła. W efekcie części obiektu węzła mogą nie być spójne z punktu widzenia obserwatora znajdującego się w jaskini.



Rysunek 5.2. Poglądowy schemat ilustrujący problem, widok z góry  
(prawidłowa skala nie została przedstawiona)

**Rozwiązanie** Aby rozwiązać ten problem, należało wykonywać obrót grafik węzłów względem jednego punktu znajdującego się w dokładnie tej samej pozycji dla wszystkich ścian (użyta została centralna pozycja głównych okularów śledzonych w jaskini). To przekształcenie wymagało ręcznego przepisania shadera z powodu braku gotowego i dostępnego rozwiązania. Problem sprowadził się do stworzenia macierzy  $R$  (wzór 5.1 oraz 5.2) przekształcającej wektor trójwymiarowy  $a$  w  $b$  - w tym przypadku domyślny zwrot grafiki węzła w kierunek wskazujący na pozycję nowego wybranego obiektu wspólnego. Implementacja jest wzorowana na wpisie z forum matematycznego [?].

$$R = I + [v]_{\times} + [v]_{\times}^2 \frac{1}{1+c} \quad (5.1)$$

$$[v]_{\times} \stackrel{\text{def}}{=} \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}, \quad (5.2)$$

gdzie:

- $v = a \times b$
- $c = a \cdot b$

### **5.2.2. Reprezentacja połączenia**

Połączenia są w aplikacji reprezentowane jako krzywe (rysunek 5.3). Ich kształt został najpierw zawinięty w góre tak, aby przechodzić przez stworzone dla każdego połączenia węzły pomocnicze opisane w sekcji 5.3, a następnie zmierza łukiem w kierunku faktycznego węzła końcowego. Na początku do tworzenia tras połączeń używane były krzywe Béziera, jednak obecnie, z powodu ich skomplikowanego kształtu, zastąpione zostały krzywymi B-Spline zaimplementowanymi według algorytmu De Boora.

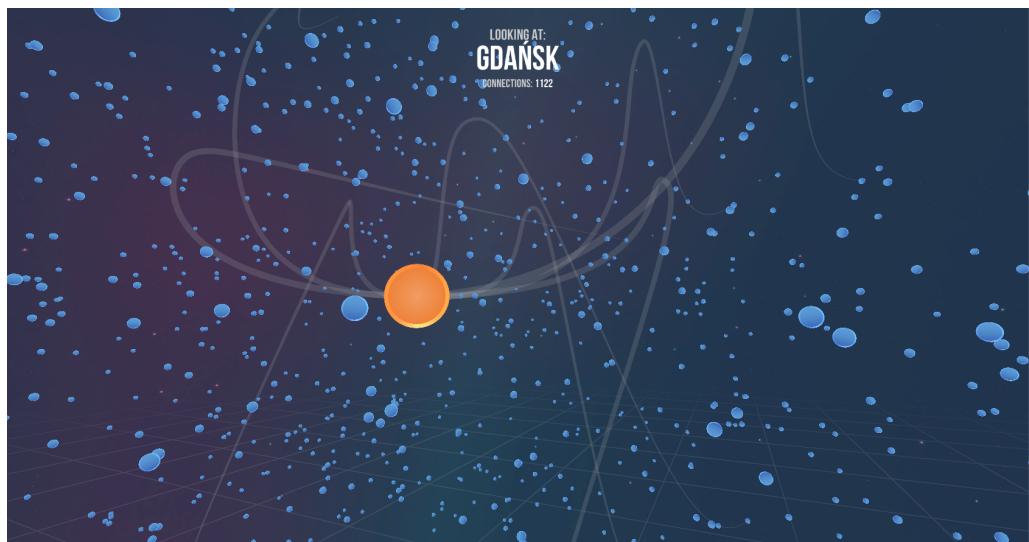


Rysunek 5.3. Wygląd węzłów i połączeń w aplikacji

### **5.3. Tryby poruszania się i widoki węzła**

MIKOŁAJ MIRKO

Aplikacja posiada dwa tryby poruszania się: swobodne latawanie oraz podróżowanie po węzłach. Pierwszy z nich charakteryzuje się pełną swobodą przemieszczania się. Po wskazaniu kierunku kontrolerem i pochyleniu joysticka do przodu, kamera zaczyna przemieszczać się do przodu, przy odchyleniu joysticka do tyłu w przeciwną stronę. Otaczająca użytkownika przestrzeń jest wizualnie wzorowana na przestrzeni kosmicznej (skybox posiada przygotowaną teksturę imitującą gwiaździste niebo). Rozmieszczone w tej przestrzeni węzły (artykuły i kategorie) można wskazywać i wybierać. W przypadku wskazania jednego z nich wyświetlany jest podgląd jego pierwszych połączeń (jeżeli jakiekolwiek posiada). Rysunek 5.4 prezentuje ten scenariusz na przykładzie artykułu "Gdańsk".



Rysunek 5.4. Widok grafu ze wskazanym węzłem i podglądem jego połączeń

Po wybraniu węzła (naciśnięciu przycisku spustu na kontrolerze) użytkownik umieszczany jest w widoku węzła i wykorzystuje tryb poruszania się po węzłach. W tym momencie kamera umieszczana jest nad wybranym węzłem, a linie połączeń wypełniają się kolorem. Nieaktywne węzły są wygaszane, aby zwiększyć przejrzystość widoku węzła. W każdej chwili można powrócić do widoku grafu i swobodnego latania klikając przycisk oznaczony jako *Exit* na kontrolerze. Na rysunku 5.5 znajduje się przykładowygląd wychodzących połączeń z artykułu "Politechnika Gdańsk" ze wskazanym artykułem "Technologia" i podglądem jego połączeń. Przemieszczanie w tym trybie polega głównie na przechodzeniu z węzła na węzeł poprzez dostępne połączenia. Cofanie się po przebytej drodze to inny sposób trawersowania w tym trybie – został on szczegółowo opisany w sekcji 5.5.



Rysunek 5.5. Widok węzła z połączonymi i zaznaczonym powiązanym węzłem

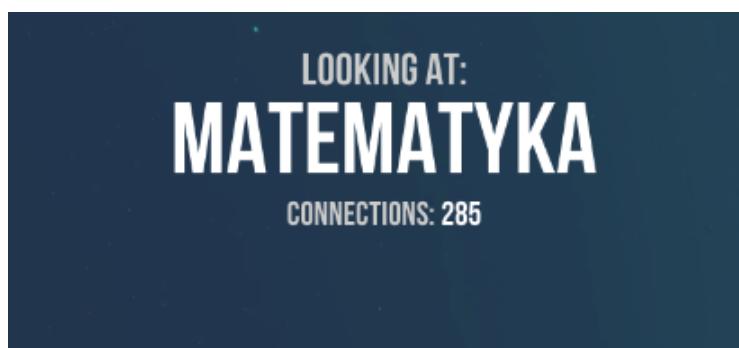
Widok węzła składa się z dwóch części. Pierwsza z nich definiuje wyświetlane połączenia jako wychodzące z wybranego węzła (jest to domyślne rozwiązanie). Drugą częścią są połączenia, które docierają z innych węzłów do aktualnie wybranego. Aby przełączyć się między widokami, należy nacisnąć spust wskazując kontrolerem w aktualnie wybrany element, znajdujący się na dolnym ekranie. Różnice pomiędzy tymi rodzajami połączeń (oraz sposoby ich pozyskiwania) zostały opisane w sekcji 4.4.1.

## **5.4. Elementy zwiększające użyteczność aplikacji**

MIKOŁAJ MIRKO

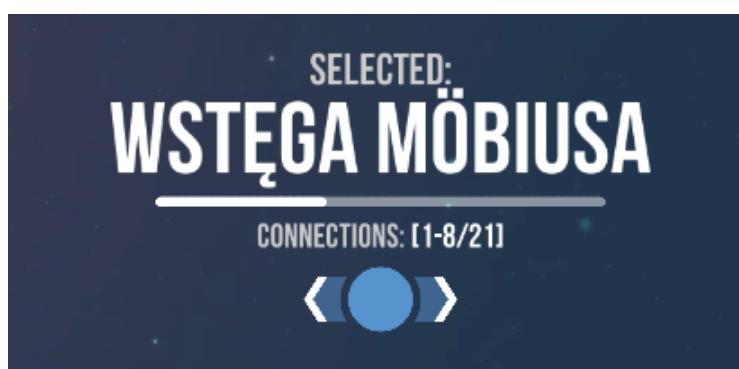
Jednym z ważniejszych elementów znajdujących się w aplikacji jest podążający za wzrokiem nagłówek z informacjami o stanie grafu. Umiejscowiony jest on powyżej linii wzroku tak, aby nie przeszkadzał w interakcji z węzłami, a zarazem był wyraźnie widoczny z każdego miejsca oraz kąta patrzenia. Jego pozycja pionowa jest zaczepiona w stałym punkcie, a pozycja pozioma uzależniona od kierunku wyznaczanego przez główne okulary jaskini.

Na samej górze znajduje się informacja o tym, czy węzeł jest tylko wskazywany kontrolerem, czy jesteśmy w widoku grafu z aktualnie wybranym węzłem. Zaraz pod spodem widoczny jest tytuł artykułu lub kategorii. W trybie swobodnego latania nagłówek wyświetlany jest tylko w przypadku wskazywania węzłów. Rysunek 5.6 nakreśla układ nagłówka po wskazaniu artykułu "Matematyka". Jeśli węzeł posiada jakieś połączenia, ich liczba wyświetlana jest pod tytułem.



Rysunek 5.6. Nagłówek po wskazaniu węzła

Dla widoku węzła dostępna jest większa ilość informacji. Pod tytułem pojawia się podłużny wskaźnik przypominający poziomy pasek przewijania. Wraz z informacją tekową, znajdująca się tuż pod nim, określa aktualnie wyświetlany wycinek listy połączeń. Łatwo zauważać, które połączenia są wyświetlane wokół użytkownika oraz jaką część stanowią. Na rysunku 5.7 wybrany został artykuł zatytułowany "Wstęp Möbiusa" z 21 połączonymi wychodzącymi (aktualnie wyświetlane są połączenia od pierwszego do ósmego włącznie).



Rysunek 5.7. Nagłówek w widoku węzła z informacją o połączeniach

Ostatnią częścią nagłówka jest ikona stanu. Zawiera ona 3 istotne informacje: rodzaj węzła (artykuł lub kategoria), aktualny typ połączeń (wychodzące z węzła lub do niego trafiające) oraz wariant poruszania się. Rodzaj węzła jest ilustrowany poprzez kolor oraz kształt (rysunek 5.1). O typie połą-

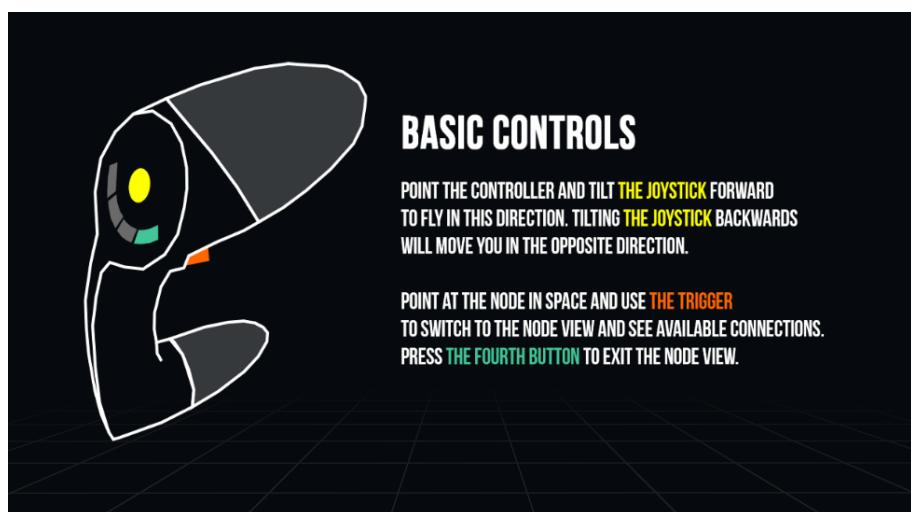
ceń świadczy również kolor, a także skierowanie strzałek w stosunku do symbolu węzła. (Na rysunku 5.7 ikona oznacza artykuł z połączonymi wychodzącymi). Domyślnym wariantem poruszania się jest ręczne przeskakiwanie z węzła na węzeł. Przy uruchomionej zaprogramowanej trasie (czyli automatycznym przechodzeniu po zdefiniowanej ścieżce - więcej o tym w sekcji 5.5) na symbolu węzła pojawia się fraza *AUTO* (rysunek 5.8).



Rysunek 5.8. Nagłówek z ikoną stanu sygnalizującą trwające automatyczne przemierzanie trasy

Aplikacja została dodatkowo wyposażona w elementy wspomagające pracę z grafem. Na dolnym ekranie widoczne są półprzecroczyste linie pomocnicze określające płaszczyznę podłoża. Przemieszczają się one z kamerą i mają na celu zapewnienie punktu odniesienia do innych obiektów otaczających użytkownika. Opisana siatka widoczna jest na rysunku 5.5.

Kolejnym elementem jest przestrzeń pomocy. Jest to specjalna przestrzeń wydzielona od struktury grafu, do której można wejść w dowolnej chwili korzystania z aplikacji. Użytkownik jest w niej umieszczany również na starcie programu. Zawiera ona podstawowe informacje o programie, a także zestaw infografik zapoznających użytkownika ze sposobami sterowania i możliwościami aplikacji (rysunek 5.9).



Rysunek 5.9. Jedna z infografik dostępnych w przestrzeni pomocy

Dodatkowo na potrzeby aplikacji została stworzona muzyka. Przez cały czas działania aplikacji w tle odtwarzany jest klimatyczna ścieżka dźwiękowa w gatunku ambient wzmacniająca immersję. Ponadto każda akcja użytkownika ma przypisany inny efekt. Sygnalizuje on użytkownikowi, że aplikacja poprawnie odebrała wciśnięcie przycisku, co zwiększa intuicyjność sterowania. Co więcej, sygnalizuje specjalne akcje, takie jak zmiana trybu przeglądania, czy włączenie lub wyłączenie pomocy.

## 5.5. Historia przeglądania oraz zaprogramowane trasy

MATEUSZ JANICKI

Specyfika działania aplikacji, związaną z wykonywaniem przez użytkownika akcji nawigacyjnych, oraz środowisko działania skłoniły do zaimplementowania funkcjonalności związanych z historią. Znacznie ułatwia i przyspiesza one poruszanie się po węzłach grafu. Bez możliwości skorzystania z historii użytkownik byłby zmuszony do każdorazowego szukania poprzedniego węzła w grafie, bądź w odpowiednim widoku wybranego węzła. Co więcej, aby zapewnić powtarzalność podczas prezentacji, a także, aby wyeksponować ciekawe przypadki połączeń pomiędzy węzłami, będzie możliwe zapisanie tras przeglądania. Odtwarzanie tras umożliwia dokładne planowanie prezentacji i eliminuje problemy wynikające z losowości wyświetlanego węzłów.

Do historii zapisywane są tylko akcje kluczowe dla nawigacji: zmiana widoku wyświetlania oraz wybieranie węzłów. Inne działania użytkownika nie powodują znaczących zmian w nawigacji po grafie i ich przechowywanie jest bezcelowe. Po naciśnięciu *Przycisku 2* na kontrolerze (ilustracja interakcji na rysunku 5.10), uprzednio przytrzymując przycisk spustu, wywoływana jest poprzednia akcja wykonana przez użytkownika. Odpowiednio, aby ponowić cofniętą akcję, należy nacisnąć *Przycisk 3*. W przypadku braku akcji do ponowienia lub cofnięcia nie jest podejmowane żadne działanie. Wywoływanie tras dostępne jest z poziomu konsoli operatora opisanej w następnym punkcie.



Rysunek 5.10. Schemat interakcji przewijania historii

Akcje użytkownika są przechowywane jako odpowiednie klasy dziedziczące po interfejsie `UserAction` (listing 5.2). Posiada on wirtualne funkcje `Execute()` oraz `UnExecute()`, służące odpowiednio ponawianiu i cofaniu akcji. Każda akcja posiada także swojego własnego delegata zależnego od typu akcji.

```
1 public interface UserAction {  
2     void Execute();  
3     void UnExecute();  
4 }
```

Listing 5.2: Interfejs `UserAction`

Bezpośrednie operacje na historii wykonywane są przez `HistoryService`. Przechowuje on akcje użytkownika we dwu stosach: pierwszy służy zapisywaniu akcji do cofnięcia, drugi akcji do ponowienia. W momencie wciśnięcia *Przycisku 2* wyzwalane jest zdarzenie wywołujące funkcję `UndoAction()`, która poza wykonaniem kolejnego zdarzenia wywołującego funkcję akcji odpowiedzialnej za cofanie, zdejmuje tę akcję ze stosu akcji do cofania i wkłada na stos akcji do ponawiania.

Wszystkie delegowane funkcje służące historii oraz trasom są przypisywane w kontrolerze `HistoryController`. Warunkiem jest, aby aplikacja była serwerem.

Zaznaczanie węzłów z historii oraz zmiana widoku odbywa się w identyczny sposób, jak zwykłe wywoływanie tych akcji przy użyciu kontrolera, jednak nie są one ponownie zapisywane do historii, jak przy zwykłym wywołaniu.

Analogicznie do modułu historii zaprojektowane zostało odtwarzanie tras. Pliki zawierające trasy posiadają rozszerzenie `.wgr`. Struktura pliku jest stosunkowo prosta - pierwsza cyfra służy rozróżnieniu akcji użytkownika: 0 oznacza zmianę widoku, a 1 wybranie węzła. W przypadku zmiany widoku pośredniku można znaleźć cyfrę 0 lub 1. 0 odpowiada zmianie widoku na widok dzieci, a 1 zmianie widoku na widok rodziców. Jeśli akcję było wybranie węzła, po średniku znajduje się ID artykułu do wybrania. Przykładowy plik widać na rysunku 5.11.

```
1 1;712
2 0;0
3 1;895
4 0;1
5 1;1512
6 0;1
7 1;113
8 1;6539
9 0;0
10 1;1877
11 1;3509
12 1;99
13 1;12455
14 1;121
```

Rysunek 5.11. Fragment przykładowego pliku `.wgr` zawierającego trasę

Po wybraniu trasy rozpoczyna się proces jej wczytywania. Są za to odpowiedzialne dwie klasy: `RoutesReader` oraz `RoutesLoader`. Pierwsza ma za zadanie pobrać dostępne trasy, ich nazwy i długość oraz wczytywać kolejne linijki plików. Druga przekształca wczytane dane linijką po linijce na odpowiednie akcje i dopisuje do stosu.

W momencie włączenia trasy inicjowane jest wczytywanie trasy z plików, a po tej operacji tworzony jest współprogram (ang. coroutine), który ma za zadanie wykonywanie kolejnych akcji trasy co daną liczbę sekund. Odtwarzanie trasy sygnalizowane jest przez ikonę *AUTO* w nagłówku. W każdym momencie możliwe jest wyłączenie odtwarzania trasy za pomocą *Przycisku 4* lub jakiekolwiek akcji wykonanej przez użytkownika, co powodujące zatrzymanie współprogramu. W przypadku włączenia innej trasy w trakcie odtwarzania, współprogram poprzedniej trasy jest także wyłączany, a następnie wczytywana i inicjalizowana jest nowa trasa.

## 5.6. Integracja z jaskinią rzeczywistości wirtualnej

STANISŁAW GÓRA

Laboratorium Zanurzonej Wizualizacji Przestrzennej [?] posiada szereg innowacyjnych rozwiązań z zakresu rzeczywistości wirtualnej. Największą instalacją jest wykorzystywana w tym projekcie duża jaskinia będąca sześciennym pomieszczeniem o boku 3,4m. Na każdą ze ścian wyświetlany jest z zewnątrz obraz z dwóch projektorów zdolnych stworzyć projekcję stereoskopową odbieraną przy pomocy okularów migawkowych [?]. Sterowanie symulacją odbywa się poprzez użycie kontrolera z joystickiem oraz pięcioma przyciskami. Dodatkowo zainstalowany jest system śledzenia ruchu pozwalający na wykrycie położenia kontrolera oraz okularów osoby sterującej symulacją.

Przystosowanie aplikacji do działania w środowisku LZWP dzieli się na dwa podzadania: obsługa wejścia użytkownika oraz synchronizacja stanu aplikacji pomiędzy komputerami obsługującymi ściany jaskini. Reszta integracji jest zapewniana przez stworzoną przez pracowników LZWP bibliotekę.

### 5.6.1. Moduł wejścia

W założeniu aplikacja ma wspierać dwie metody interakcji użytkownika, nazywane później środowiskami:

- **PC** - gdzie użytkownik posługuje się klawiaturą i myszą, używane głównie podczas procesu wytwarzania aplikacji
- **Jaskini** - gdzie użytkownik nosi okulary i trzyma specjalny kontroler nazywany flystickiem

Aby ułatwić sobie wytwarzanie głównej logiki aplikacji, wejście zostało odseparowane w osobny moduł, który może być przed uruchomieniem ustawiony w jeden z dwóch trybów, odpowiadającym każdemu środowisku. Tryb ten definiuje jakiego rodzaju wejścia ma nasłuchiwać aplikacja. Moduł wejścia składa się z trzech mniejszych części:

**Definicje akcji użytkownika** Ustala wspólny interfejs dla każdej akcji użytkownika dostępnej w aplikacji. Są to między innymi:

- Użycie przycisku klawiatury i myszy (PC) lub flystica (jaskinia)
- Przesunięcie myszy (PC) lub flystica (jaskinia)

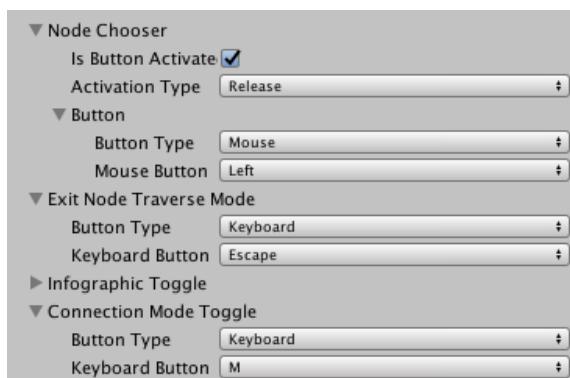
Oprócz tego jest odpowiedzialna za nasłuchiwanie wykonania przez użytkownika swojej akcji i odpowiednie powiadomienie aplikacji.

**Definicje akcji aplikacji** Spis akcji wykonywanych w aplikacji. Są to między innymi:

- Wybranie lub podświetlenie węzła
- Poruszanie się po grafie
- Zmiana zakresu wyświetlanych połączeń

Każda akcja ma swój typ (opisany w poprzednim punkcie). Dostępny jest też interfejs, dzięki któremu możliwe jest łatwe wybranie konkretnego przycisku (w każdym ze środowisk), który ma aktywować daną akcję (rysunek 5.12).

**Procesor akcji użytkownika** Odpowiada za połączenie modułu wejścia z pozostałą częścią aplikacji poprzez powiadamianie jej o każdej wykonanej przez użytkownika akcji po ewentualnym wstępnym przetworzeniu sygnału wejściowego.

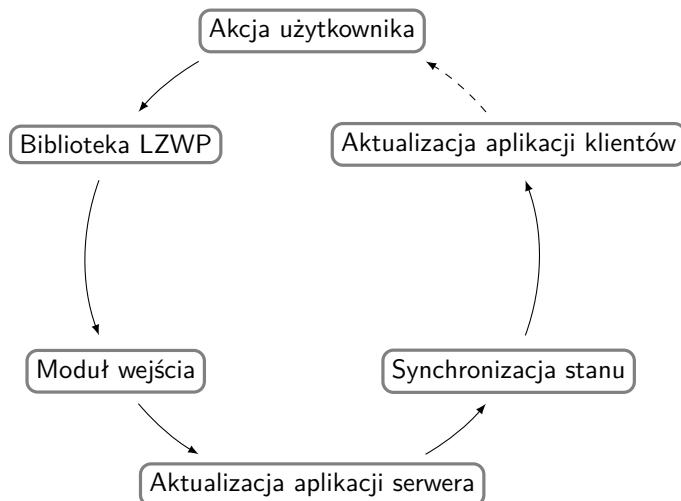


Rysunek 5.12. Fragment interfejsu wyboru przycisków dla środowiska PC

### 5.6.2. Synchronizacja stanu

Jak już zostało wspomniane, w LZWP każda ściana jaskini sterowana jest przez osobny komputer. Oznacza to, że środowisko aplikacji podobne jest do wieloosobowej gry w sieci LAN i wymusza synchronizację stanu z głównym komputerem, odpowiedzialnym za przetwarzanie akcji użytkownika. W tym celu powstał w aplikacji moduł wysyłający zmiany w grafie komputerom do aktualizacji ekranów. Cały proces zaprezentowany został na diagramie na rysunku 5.13.

Na opisanie zasługuje tu synchronizacja załadowanych lub usuniętych węzłów grafu. Mechanizm przesyłający wiadomości w sieci jest dosyć prosty i pozwala na dołączanie tylko najprostszych typów danych jak ciąg znaków, liczby i pozycja. Wysyłanie informacji o załadowaniu każdego węzła z osobna mogłoby zapełnić kanał komunikacyjny zwłaszcza przy starcie aplikacji, kiedy ładowane są tysiące węzłów na raz. Został więc wprowadzony mechanizm, którego celem jest zbieranie tych komunikatów i łączenie ich w większe paczki, które po przesłaniu są z powrotem rozpakowywane i przetwarzane. Dzięki temu aplikacja zachowuje optymalne wykorzystanie łącza w jaskini.



Rysunek 5.13. Diagram kolejności wykonywania modułów przy akcji użytkownika

## 5.7. Zarządzanie pamięcią w aplikacji

JAN KRUCZYŃSKI

Ze względu na to, że aplikacja doładowywuje węzły wraz z kolejnymi akcjami użytkownika, gdy przemieszcza się on po grafie lub wskazuje na węzły w przestrzeni, musiał zostać opracowany sposób ich usuwania, by utrzymać płynność działania. Aby to osiągnąć powstał oddzielnny kontroler zarządzania pamięcią.

```
1 private List<uint> lowPriorityNodes;
2 private List<uint> highPriorityNodes;
3 public int maxAmountOfNodes;
```

Listing 5.3: Pomocnicze struktury i zmienne kontrolera zarządzania pamięcią

Węzły które są załadowane w aplikacji zostały podzielone na dwie kategorie - węzły niskiego priorytetu (wiersz 1 na listingu 5.3), które są potencjalnymi kandydatami na węzły do usunięcia, oraz wysokiego priorytetu (wiersz 2 na listingu 5.3), których usunąć aktualnie nie można.

Początkowo wszystkie załadowane węzły trafiają do listy niskiego priorytetu. Jeżeli użytkownik dokonuje interakcji z węzłem, to zarówno ten węzeł, jak i wszyscy wyświetlani jego sąsiedzi trafiają

do listy wysokiego priorytetu. Jeżeli któryś z tych węzłów znajdował się w liście niskiego priorytetu, jest z niej usuwany i przenoszony na początek listy wysokiego priorytetu.

Funkcjonuje to w taki sposób, aby aplikacja zwalniała w pierwszej kolejności węzły z którymi użytkownik nigdy nie miał interakcji - i dzięki temu nie dostrzegł, że czegoś brakuje. Jednocześnie realizowany jest dodatkowy cel, by ilość wyświetlanych (i przez to przechowywanych w pamięci) węzłów w dowolnej chwili była stała z dokładnością do kilkunastu sztuk.

Kontroler nasłuchuje zdarzenia załadowania węzła przez aplikację. Gdy takie zdarzenie zostanie wykryte, jeżeli liczba wszystkich węzłów, które są wyświetlane w aplikacji, nie przekracza maksymalnej dopuszczonej ilości, nie dzieje się nic. Jeżeli ta liczba zostanie przekroczona, zwalnia kilkanaście węzłów z listy niskiego priorytetu.

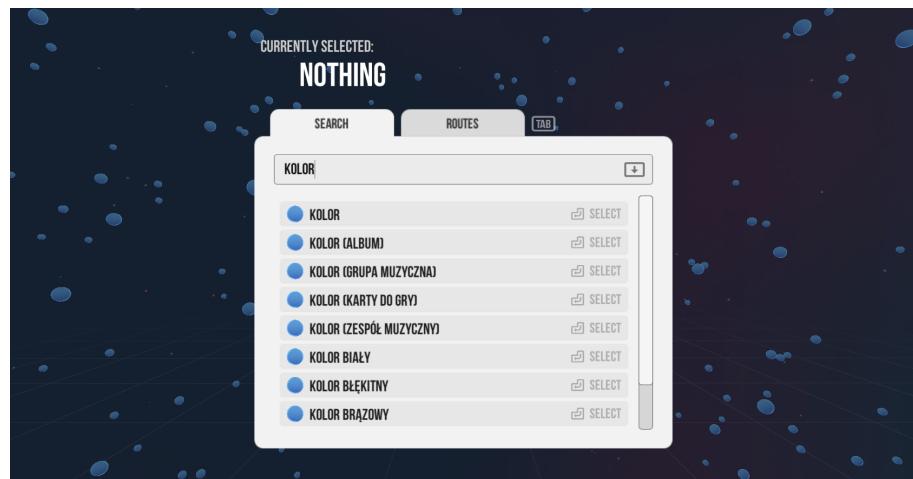
Gdy lista niskiego priorytetu jest pusta, kontroler przerzuca najstarsze węzły z listy wysokiego priorytetu do listy niskiego priorytetu (są to węzły, z którymi użytkownik prowadził interakcję najdawniej). Gdy użytkownik dokonuje interakcji z węzłem, jest on przerzucany na sam początek listy wysokiego priorytetu.

## 5.8. Konsola operatora

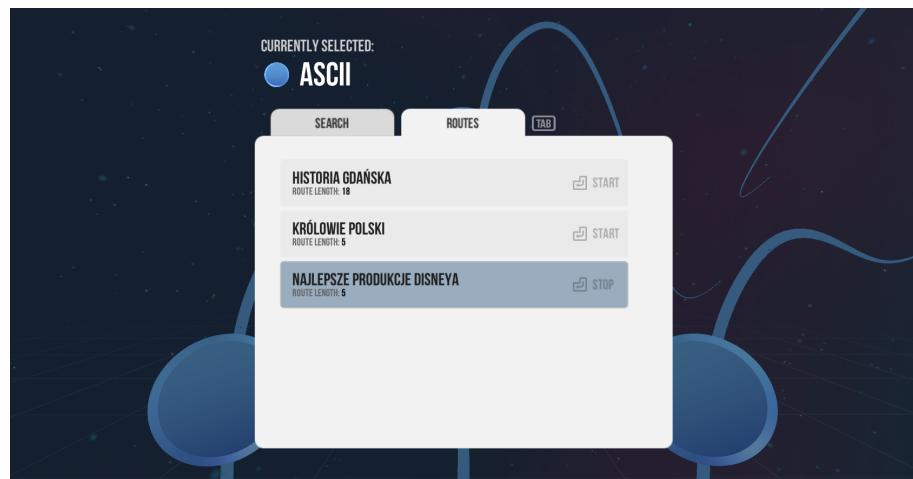
MATEUSZ JANICKI

Aplikacja, poza wykorzystaniem flysticków lub klawiatury, może być sterowana za pomocą specjalnej konsoli. Każdorazowe skorzystanie z jaskini wymaga obecności operatora, który uruchamia aplikację w trakcie prezentacji. Będzie on posiadał dostęp do konsoli, a jego zadaniem będzie przenoszenie użytkownika na konkretne węzły przy użyciu wyszukiwarki, a także włączanie zaprogramowanych tras przeglądania grafu.

Konsola operatora domyślnie wywoływana jest po naciśnięciu klawisza *F1* na klawiaturze serwera. Interfejs posiada dwie zakładki: *Search* (rysunek 5.14) oraz *Routes* (rysunek 5.15). Pierwsza jest odpowiedzialna za wyszukiwanie węzłów, na które ma zostać przeniesiony użytkownik, druga służy włączaniu tras. Obydwa elementy posiadają skrypt o nazwie *Toggle*, służący odpowiedniemu przełączaniu aktywnych elementów w konsoli. W obu zakładkach znajduje się obiekt przechowujący wyniki wyszukiwania, umieszczane w obiekcie zawierającym komponent *Scroll Rect*. Umożliwia on automatyczne utworzenie paska przewijania, co daje możliwość umieszczenia bardzo dużej ilości elementów. Dodatkowo w zakładce *Search* znajdują się także obiekty o nazwie *SearchInput* odpowiedzialny za wpisywanie wyszukiwanego węzła. Posiada on komponent skryptu *InputField*. Ponadto w górnej części konsoli znajduje się również informacja o tym, jaki węzeł jest aktualnie wybrany. Poza korzystaniem z konsoli za pomocą myszki istnieje możliwość użycia wyłącznie klawiatury, a odpowiednie klawisze wykonujące akcje przedstawione zostały w interfejsie konsoli. Każdorazowe wybranie węzła lub jakakolwiek akcja na trasach powoduje automatyczne zamknięcie konsoli. Gdy konsola jest otwarta, użytkownika informuje o tym specjalna ikona. Blokowane są również wszystkie akcje użytkownika, aby nie spowodować wyścigów i niezgodnych stanów przy synchronizacji.



Rysunek 5.14. Zrzut ekranu konsoli z otwartą zakładką Search



Rysunek 5.15. Zrzut ekranu konsoli z otwartą zakładką Routes i włączoną trasą

Wyszukiwanie węzłów odbywa się na przygotowanym uprzednio pliku, zawierającym posortowane alfabetycznie nazwy artykułów i kategorii oraz ich ID. Przeszukiwanie listy odbywa się za pomocą algorytmu wyszukiwania binarnego. Lista elementów aktualizuje się z każdym wystąpieniem wydarzenia `OnValueChanged` obiektu `SearchInput`. Instancjonowane są kolejne prefaby wyników wyszukiwania, w których zmieniana jest ich pozycja w pionie nazwa i ikona symbolizująca artykuł lub kategorię. W momencie wybrania węzła wywoływane jest zdarzenie `OnClick`, do którego podpięta jest funkcja `HistoryController` odpowiedzialna za wybranie węzła. Tło wybranego przycisku jest ściemniane na moment kliknięcia myszką.

Lista tras jest pobierana na podstawie aktualnie wybranej wersji Wikipedii. Prefab wyświetlający trasę posiada nazwę, jej długość oraz przycisk służący jej włączaniu lub wyłączaniu. W celu utworzenia kolejnych elementów listy wykorzystywana jest funkcja `Instantiate()`. W kolejnych instancjach zmieniana jest wysokość obiektu, nazwa przycisku, nazwa trasy oraz długość trasy. W momencie włączenia trasy wywoływane jest zdarzenie `OnClick`, do którego podpięta jest funkcja `HistoryController` wczytująca trasę o numerze indeksu pod wybranym przyciskiem. Ponadto tło wybranego obiektu zmieniane jest na niebieskie.

## **5.9. Linia czasu**

MATEUSZ JANICKI

Specyfikacja wymagań projektu inżynierskiego zawierała funkcjonalność dotyczącą linii czasu. Wybranie daty miało spowodować wyświetlanie tylko takich węzłów, których data utworzenia była starsza niż wybrana data. Pomogłoby to w przybliżonej wizualizacji, w jak szybki sposób rozrastała się konkretna Wikipedia. Szereg komplikacji spowodował jednak odrzucenie przez nas tej funkcjonalności.

Sterowanie linią czasu miało odbywać się za pomocą drugiego kontrolera. Po wcisnięciu przycisku miał pokazywać się specjalny interfejs ukazujący aktualnie wybraną datę. Można w niej było wybrać, przesuwając joystick w lewo i prawo, miesiąc lub rok, który następnie dałoby się zmieniać przesuwając joystick do góry lub w dół. Zatwierdzenie daty przyciskiem spustu wywoływałoby zmiany w grafie.

Funkcjonalność nie została zaimplementowana głównie z powodu braku prostej informacji o stworzeniu artykułu w głównie wykorzystywanym przez nas źródle danych, czyli zrzutach baz danych Wikipedii. Istnieją archiwa zawierające kompletną historię edycji artykułów, jednak wielkość tych plików jest nieporównywalnie większa w porównaniu ze zwykłym wykazem stron czy nawet połączonymi pomiędzy stronami. Przetworzenie tych plików tylko po to, aby wydobyć z nich datę pierwszej publikacji strony, jest niewymierna do korzyści wynikających z tej funkcjonalności.

Innym sposobem wydobycia danych mogłoby być pozyskiwanie tych informacji przy użyciu Wikipedia API. Pomyśl ten jednak nie jest wykonalny z powodu ograniczeń, które posiada API. Aby zdobyć informacje dotyczące każdego artykułu, musielibyśmy wywołać zapytania, których liczba znacznie przewyższa maksymalną dopuszczalną liczbę zapytań dla jednego użytkownika. Przekroczenie tej liczby powoduje zablokowanie adresu IP, z którego pochodziły zapytania i utrudnia dalsze pozyskiwanie danych.

Co więcej, nasza idea linii czasu nie byłaby dokładnym odwzorowaniem stanu Wikipedii w wybranym przez użytkownika czasie. Artykuły i kategorie Wikipedii są bardzo często zmieniane, co implikuje możliwość zmiany połączeń pomiędzy artykułami. Ponadto artykuły i kategorie mogą być także usuwane. Przechowywanie całej historii stron lub stworzenie plików zawierających dokładną historię wszystkich węzłów, jakie kiedykolwiek pojawiły się w Wikipedii, jest niemożliwe na zwykłych komputerach osobistych z powodu zbyt dużej wymaganej przestrzeni na dysku. Nawet jeśli udałoby się je zapisać, przetwarzanie i wczytywanie takich plików zajęłoby bardzo dużo czasu i spowodowałoby niską responsywność aplikacji. Nasza wersja linii czasu mogłaby być myląca dla nowych użytkowników, którzy mogliby pomyśleć, że przedstawiana jest im dokładna wersja Wikipedii w danym czasie.

## 5.10. Ostateczny schemat interakcji

JAN KRUCZYŃSKI

Po odrzuceniu funkcjonalności, takich jak widok treści i linia czasu, możliwa stała się obsługa aplikacji za pomocą tylko jednego kontrolera. Podział na kontroler główny (rysunek 3.3) i pomocniczy (rysunek 3.4) został odrzucony, a wszystkie interakcje zostały umieszczone na jednym kontrolerze. Ten zabieg znacznie ułatwi użytkownikowi sterowanie aplikacją w jaskini. Nowy schemat sterowania został umieszczony na rysunku 5.16.



Rysunek 5.16. Aktualny schemat sterowania kontrolerem

Konsola operatora opisana w sekcji 5.8 uruchamiana jest na komputerze głównym klawiszem *F1*. Możliwa jest obsługa za pomocą myszki i klawiatury lub wyłącznie klawiatury. Przełączanie pomiędzy zakładkami *Search* i *Routes* następuje po wcisnięciu klawisza *Tab*. Poruszanie się po liście wyników wyszukiwania i liście dostępnych tras odbywa się za pomocą klawiszy strzałek, a potwierdzenie wyboru zaznaczonej pozycji to klawisz *Enter*.

## 6. EKSPERYMENT

MATEUSZ JANICKI

Aplikacja została przetestowana na dużej jaskini, po pomylnym przejściu testów na małej jaskini. Początkowo występuły drobne problemy z synchronizacją, jednak zostały one przez nas rozwiązane. Dopiero na środowisku docelowym istnieje możliwość sprawdzenia poprawności działania efektu 3D przy użyciu okularów zamieszczonych na rysunku 6.1 oraz intensywniejszej synchronizacji między komputerami.



Rysunek 6.1. Okulary umożliwiające widzenie w 3D

Do testów aplikacji wykorzystane zostały scenariusze użycia opisane w sekcji 3.4. Ostatecznie tylko trzy z pięciu opisanych scenariuszy mogły zostać zrealizowane, ponieważ pozostałe korzystają z niezaimplementowanych i odrzuconych funkcjonalności. Ostatni może zostać zrealizowany tylko częściowo, ponieważ ostatecznie wykorzystywany jest tylko jeden kontroler, więc implementacja systemu zmiany układu sterowania była bezcelowa.

- **Przejście z węzła “Politechnika Gdańsk” do węzła “Gdańsk”**

Aplikacja została uruchomiona na dużej jaskini. Po włączeniu ukazała się przestrzeń informująca o podstawach sterowania. W celu przejścia do widoku grafu naciśnięty został przycisk 4 na kontrolerze. Spowodowało to odtworzenie krótkiej melodii i pojawiienie się węzłów w przestrzeni. Z powodu ograniczonej liczby wyświetlanych węzłów w jednym momencie, znalezienie artykułu “Politechnika Gdańsk” jest bardzo utrudnione. Znaleziony został on za pomocą konsoli operatora, co znacznie przyspieszyło cały proces. Użytkownik został przeniesiony w momencie wybrania nad węzłem, a na ścianach wokół niego pojawiły się połączenia do innych węzłów. Pojawił się nagłówek z informacją o aktualnie wybranym węźle, ilości połączeń i aktualnie wyświetlonym prze dziale ze wszystkich połączeń. Jednocześnie wyświetlane ich było dwanaście, a węzeł “Gdańsk” nie znajdował się w pierwszym zestawie. Zmusiło to użytkownika do przewinięcia dalszych połączeń i szukania docelowego artykułu. Artykuł “Gdańsk” znaleziony został w siódmym zestawie połączeń. Użytkownik wskazał go kontrolerem, co spowodowało wyświetlenie się nazwy “Gdańsk” i liczby połączeń na nagłówku z informacjami. Odtworzona została także animacja wyświetlająca połączenia wskazywanego artykułu z innymi artykułami, a kolor węzła zmienił się na pomarań-

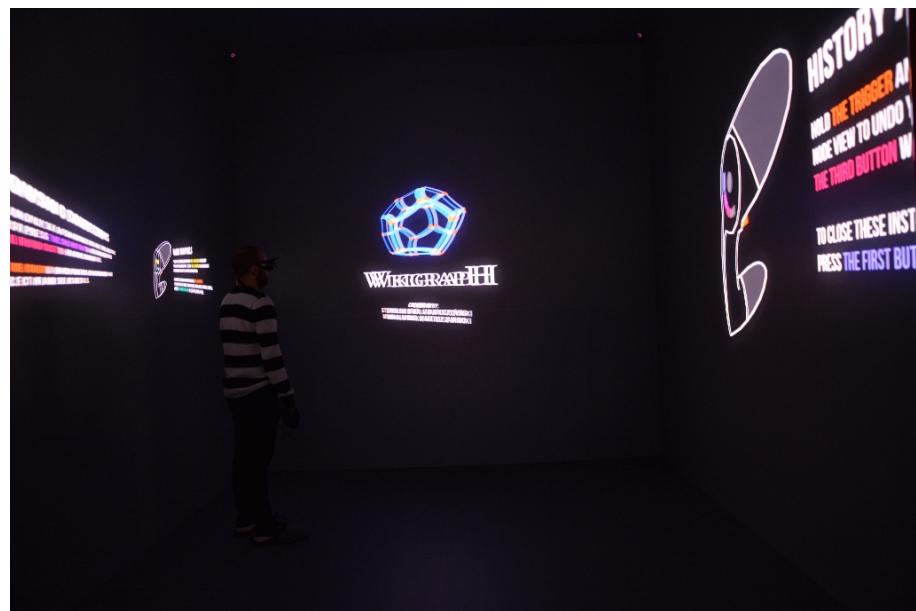
czowy. Użytkownik wcisnął przycisk spustu, co skutkowało zaznaczeniem węzła i wyświetlenie jego połączeń.

- **Przejście do artykułu należącego do tej samej kategorii co artykuł "Jan Matejko"**

Po przejściu do widoku grafu ponownie otaczają nas jedynie losowe węzły, a prawdopodobieństwo pojawienia się wśród nich poszukiwanego artykułu jest bardzo niskie. Przed przeprowadzeniem została jednak przygotowana trasa zawierająca najbardziej znanych malarzy w Polsce, a więc wykorzystanie jej jest szybkim sposobem na znalezienie artykułu. Po otwarciu konsoli i przejęciu do zakładki *Routes* został wcisnięty klawisz *Enter* na zaznaczonej trasie o nazwie "Polscy malarze", co spowodowało przeniesienie użytkownika na artykuł o nazwie "Józef Chełmoński". Ponadto zmianie uległa ikona stanu znajdująca się pod nagłówkiem - dopisany do niej został napis *AUTO*, który świadczy o tym, że jest aktualnie uruchomiona trasa. Po odczekaniu wyznaczonego w konfiguracji czasu użytkownik jest przenoszony na następny artykuł o nazwie "Aleksander Gierymski", a po następnie na poszukiwany przez nas artykuł "Jan Matejko". W tym momencie naciśnięty został przycisk 4 powodujący przerwanie trasy oraz zniknięcie napisu *AUTO* z ikony stanu. Aby wyświetlić kategorie, należy skierować kontroler na wybrany węzeł i nacisnąć spust. Spowoduje to wyświetlenie zarówno kategorii, do których należy artykuł "Jan Matejko", jak i wyświetlenie artykułów, które mają odniesienie do naszego artykułu. Ponadto zmieni się ikona stanu symbolizująca skierowanie w grafie. Poszukiwane kategorie przedstawione są w formie fioletowych figur. Połączenia można przewijać za pomocą przycisków 2 i 3 na kontrolerze. Użytkownik odnalazł kategorię "Polscy malarze", wskazał ją i nacisnął spust, zostając przeniesionym na miejsce w grafie gdzie ta kategoria się znajdowała. Ponownie skierował kontroler na węzeł pod sobą i nacisnął spust by powrócić do typu wyświetlanych połączeń prowadzących do artykułów w tej kategorii. Następnie wskazał na artykuł "Julian Cegielski" i nacisnął spust.

- **Wyświetlenie pomocy przy sterowaniu i zmiana układu sterowania**

Inicjalnie pomoc dotycząca sterowania wyświetlana jest przy starcie aplikacji. Po przejściu do trybu grafu w każdym momencie możliwe jest jednak wyświetlenie pomocy za pomocą przycisku 1 na kontrolerze. Wychodzeniu i wchodzeniu do pomocy za każdym razem towarzyszy efekt dźwiękowy.



Rysunek 6.2. Ekran pomocy zawierający informacje o sterowaniu



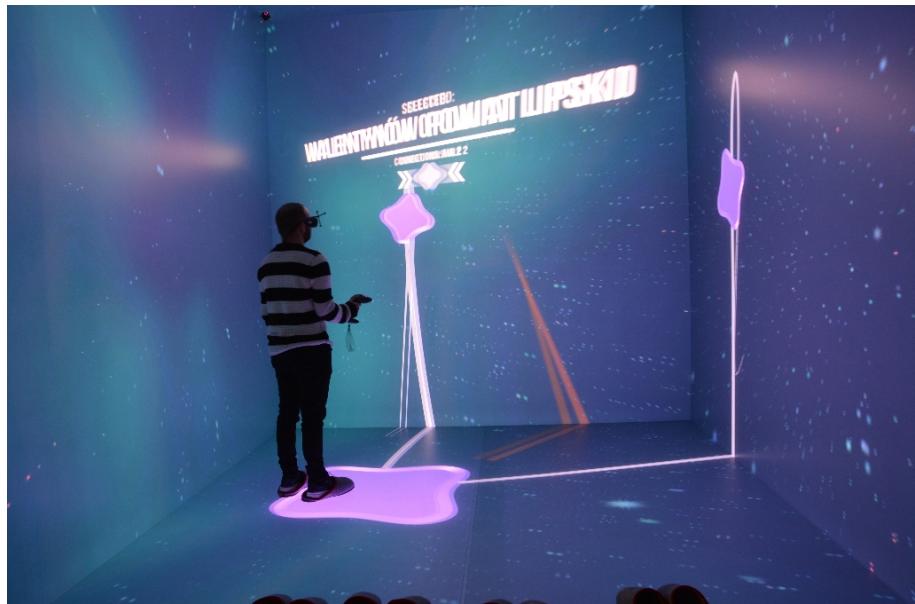
Rysunek 6.3. Wskazywanie artykułu w trybie swobodnego lotu

Poza wymienionymi wyżej scenariuszami w trakcie eksperymentu sprawdzono również poprawność innych funkcji. Wielokrotnie wykonywano podstawowe akcje dostępne w aplikacji, takie jak pokazywanie pomocy przedstawione na rysunku 6.2, czy wskazywanie artykułów i kategorii w trybie swobodnego lotu po grafie (rysunek 6.3).

Ponadto poprawnie działa przechodzenie do trybu chodzenia po węzłach. Tryb ten został przedstawiony na rysunku 6.4 dla zaznaczonego artykułu oraz na rysunku 6.5 dla zaznaczonej kategorii.

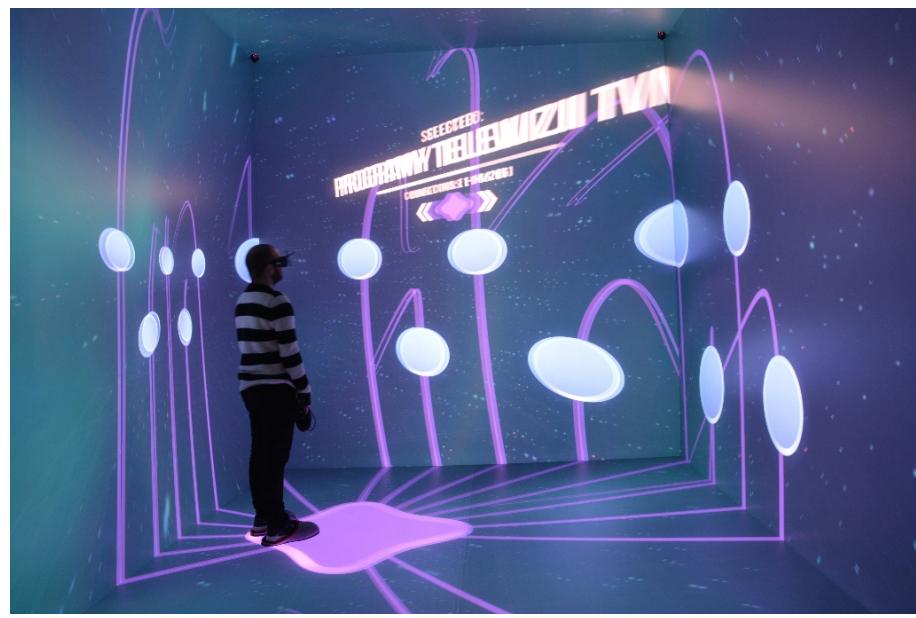


Rysunek 6.4. Widok węzła dla wybranego artykułu



Rysunek 6.5. Widok węzła dla wybranej kategorii

Problematyczne okazało się wyświetlanie węzłów na krawędziach ścian jaskini. Ewentualne niescisłości najbardziej widoczne są w trybie chodzenia po węzłach. Gdy przesuwamy kamerę za pomocą joysticka w lewo i prawo można zauważać płynne przechodzenie grafik połączonych węzłów ze ściany na ścianę na rysunku 6.6, co świadczy o poprawnym działaniu usuwania niewidocznych powierzchni (ang. frustum culling). Przetestowane zostało również wskazywanie (rysunek 6.7) i wybieranie połączeń oraz zmiana trybu wyświetlania. Za każdym razem poprawnie wyświetlana jest ikona informująca o stanie połączeń. Aplikacja zachowuje się poprawnie przy cofaniu i ponawianiu akcji wykonanych przez użytkownika.

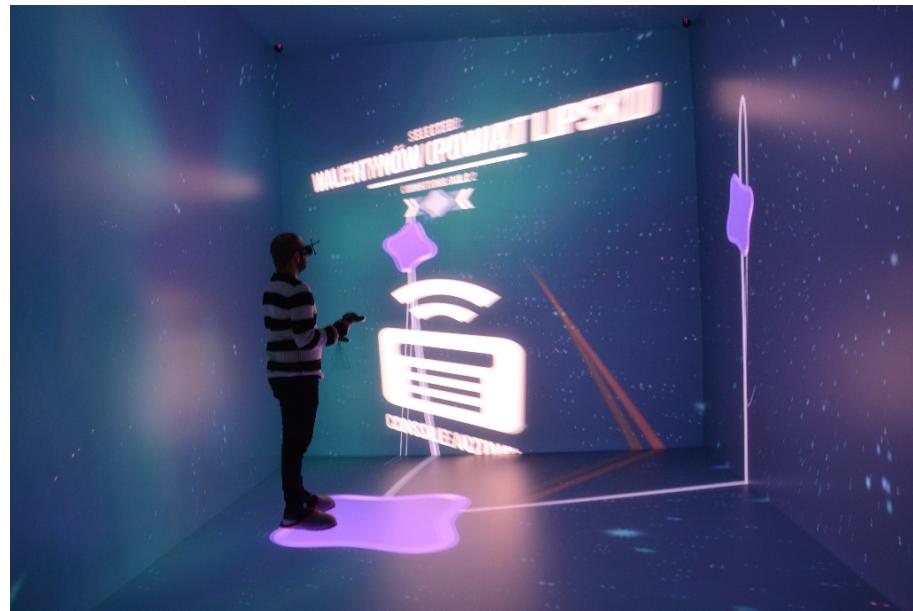


Rysunek 6.6. Wyświetlanie prawidłowego kształtu dla węzła znajdującego się na dwóch ścianach jaskini

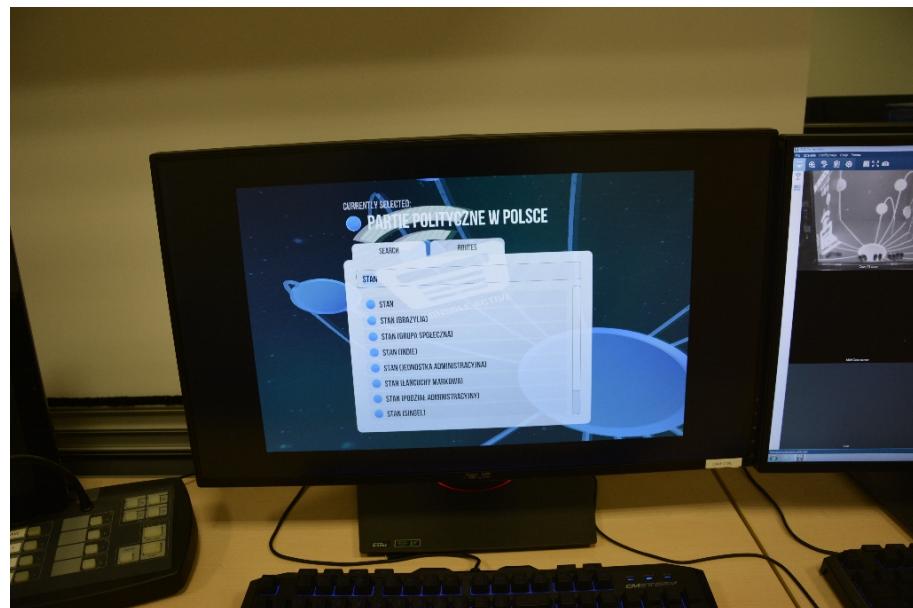


Rysunek 6.7. Wskazywanie kategorii w widoku węzła

W momencie włączenia konsoli operatora, pojawia się ikona informująca użytkownika o otwartej konsoli (rysunek 6.8). Operator może wyszukać dowolny węzeł, a wyniki wyszukiwania aktualizują się w momencie zmiany pola tekstowego (rysunek 6.9). Poprawnie działa również funkcjonalność związana z trasami - możliwe jest odtwarzanie i wyłączanie tras. Wszystkim akcjom użytkownika towarzyszy efekt dźwiękowy, co dodatkowo informuje użytkownika o odebraniu sygnału przez aplikację.



Rysunek 6.8. Ikona informująca o otwartej konsoli operatora



Rysunek 6.9. Konsola operatora na głównym komputerze w dużej jaskini

Po przeprowadzeniu eksperymentu można stwierdzić, że aplikacja, mimo braku kilku funkcjonalności zawartych w specyfikacji projektu, jest kompletna i użyteczna. Wybrane rozwiązania ułatwiające sterowanie idealnie spełniają swoje zadanie - konsola na głównym komputerze umożliwia przenoszenie na dowolne węzły oraz włączanie predefiniowanych tras. Immersja w aplikacji jest wysoka ze względu na dobrze działający efekt 3D, intuicyjny interfejs i klimatyczną muzykę. Całość powoduje przyjemny odbiór aplikacji i pokazuje nowy pomysł na wykorzystanie jaskini do wizualizacji tworów nie mających reprezentacji w rzeczywistości.

## 7. PODSUMOWANIE

Główym celem pracy inżynierskiej było zaproponowanie rozwiązania nietypowego przedstawienia treści dobrze znanego serwisu Wikipedia i zainteresowania nim potencjalnego użytkownika. Uwaga odbiorcy skierowana jest na strukturę Wikipedii i połączenia między artykułami i kategoriami, a nie ich treść.

Powstała aplikacja spełnia postawione na początku założenie. Wizualizacja grafu połączeń w przestrzeni pozwala zauważać istniejące związki między stronami, które trudno zauważać przeglądając serwis w przeglądarce internetowej. Aplikacja dołączy również do biblioteki materiałów dostępnych do uruchomienia w LZWP i pozwoli zaprezentować różne możliwości jaskini odwiedzającym ją gościom. Większość dostępnych aplikacji odzwierciedla świat rzeczywisty, nasza aplikacja pozwala wprowadzić użytkownika w zupełnie nowy, wygenerowany dynamicznie świat.

Podział zadań w projekcie oraz przy pisaniu pracy:

- **Stanisław Góra:**  
Podstawa aplikacji, integracja ze środowiskiem jaskini, nadzór nad całością
- **Mikołaj Mirko:**  
Interfejs użytkownika oraz wygląd aplikacji i programu do tworzenia danych
- **Jan Kruczyński:**  
Przetwarzanie danych Wikipedii na potrzeby aplikacji, ładowanie węzłów
- **Mateusz Janicki:**  
Dodatkowe funkcje aplikacji (historia, trasy oraz wyszukiwanie), kompozycja muzyki

Projektowanie aplikacji na środowisko jaskini rzeczywistości wirtualnej wiąże się z wieloma trudnościami. Projekt, od rozpoczęcia pracy nad nim, uległ wielu zmianom, co łatwo zauważyc walidując specyfikację napisaną na samym początku z uzyskanym produktem końcowym. Bazowe założenia pozostały jednak bez zmian.

Wśród zmian w wymaganiach funkcjonalnych (sekcja 3.1) znajduje się widok treści (podgląd części treści w odrębnym widoku aplikacji). Zrezygnowano z tej funkcjonalności ze względu na aktualny brak dostępu do Internetu w średniej i dużej jaskini. Widoki grafu (swobodnego poruszania się) oraz węzła (podgląd połączeń) zostały zaimplementowane bez większych zmian (sekcja 5.3). Funkcjonalność automatycznego i losowego poruszania się po węzłach grafu została zamieniona na możliwość odtwarzania wcześniej przygotowanych tras (sekcja 5.5). Celem tej opcji było ułatwienie prezentowania związków między artykułami, lecz w przypadku losowego podrózowania trudno odróżnić interesujące połączenia od wszystkich pozostałych.

Dużą zmianą wprowadzoną w specyfikacji funkcjonalności była rezygnacja z linii czasu. Ze względu na niewystarczające rozeznanie oraz błędne założenia dotyczące oferowanych przez wybrane źródło danych informacji niemożliwe było zgromadzenie potrzebnych do tej funkcjonalności danych. Szczegółowy opis tej problemowej funkcjonalności został opisany w specjalnej sekcji 5.9. Względem spisanych wymagań, zmianom uległ również sposób wyświetlania informacji o węzłach, infografiki instruktażowe oraz dodatkowe elementy (sekcja 5.4). Nagłówek został uszczegółowiony, a pozostałe elementy (takie jak na przykład siatka wspomagająca) dodane w celu zwiększenia użyteczności aplikacji.

W związku z opisanymi modyfikacjami schemat kontrolera uległ zmianie. Liczba kontrolerów została zredukowana do jednego, a jego przyciski zostały rozplanowane trochę inaczej - tak, aby korzystanie z kontrolera było wygodniejsze. Nowy schemat znajduje się na rysunku 5.16. To, jak i inne parametry aplikacji zostały, w ramach testów, dostosowane do potrzeb opisanych w wymaganiach jakościowych (sekcja 3.2). Ograniczenie ilości funkcjonalności pozwoliło na skupienie się na pozostałych elementach aplikacji oraz dopracowanie ich.

Nieprzewidzianym przez nas aspektem pracy stało się wydobycie potrzebnych danych. W pierwotnych założeniach sprowadzało się do pobrania i prostego przetworzenia informacji. Z czasem pojawiały się jednak kolejne problemy (dostępność danych, ich format i rozmiar, oferowane informacje o pojedynczym artykule itp.). Odpowiedzią na skomplikowany proces pozyskiwania i przetwarzania danych jest dodatkowe narzędzie WikiGraph Parser (opisane w rozdziale 4). Zostało stworzone w celu ułatwienia generowania plików wejściowych do głównej aplikacji oraz zebraniu algorytmów przetwarzających w jednym miejscu.

Podczas wykonywania projektu rozwiązane zostało wiele napotkanych problemów. Generowanie plików wykorzystywanych przez aplikację zostało przyspieszone (pomimo, często bardzo dużych, rozmiarów plików), a ich analizowanie przez program uproszczone, wykorzystując maksymalnie skompresowane dane w plikach binarnych. Rozmieszczenie i wizualizacja grafu, a także zastosowane elementy interfejsu sprawnie przekazują niezbędne informacje użytkownikowi, zachowując przy tym prosty, ale zrozumiały wygląd. Integracja elementów jaskini, takich jak okulary i kontrolery, została uproszczona poprzez własny system zarządzania różnymi środowiskami oraz przypisanymi do przycisków i joysticków akcjami. Rozróżnienie środowiska PC do testów na lokalnych komputerach przy pomocy klawiatury i myszki oraz środowiska CAVE do wyświetlania aplikacji w jaskini i sterowania za pomocą kontrolerów niebyvale zoptimizowało naszą pracę nad aplikacją. Problem braku klawiatury wewnętrz jaskini został częściowo rozwiązany poprzez stworzoną konsolę operatora.

Aplikacja z pewnością może być rozwijana w przyszłości o dodatkowe funkcjonalności, zwiększając jej wartość jako narzędzie analityczne. Uwzględniając w wizualizacji dodatkowe dane, takie jak popularność lub rozmiar treści artykułu, można skupić się na innych zależnościach wewnętrz struktur Wikipedii. Rozwinięcie możliwości sterowania aplikacją o system gestów lub komendy głosowe powinno podwyższyć stopień imersji i ułatwić użytkownikowi odnalezienie się w przestrzeni grafu.

## SPIS RYSUNKÓW

1.1	Jedna z wizualizacji stworzona przez Opte Project [?]	9
2.1	Fragment grafu aplikacji Webverse [?]	11
2.2	Widok pierwszoosobowy przestrzeni połączeń WikiGalaxy [?]	12
2.3	Wizualizacja połączeń w aplikacji Wikiverse [?]	13
2.4	Serwis Encartopedia z wybranym artykułem “Mango” i połączeniem do “Banana” [?]	13
3.1	Prototyp widoku grafu . . . . .	14
3.2	Prototyp widoku węzła (przypadek widoku powiązań artykułów)	15
3.3	Schemat kontrolera głównego . . . . .	16
3.4	Schemat kontrolera pomocniczego . . . . .	16
3.5	Diagram UML przypadków użycia . . . . .	19
4.1	Fragment pliku page.map zawierający tytuły artykułów . . . . .	23
4.2	Fragment pliku pagelinks.map z artykułami i ich połączonymi . . . . .	23
4.3	Zrzut ekranu artykułu zatytułowanego “Astronomy” . . . . .	23
4.4	Ekran konfiguracji parametrów WikiGraph Parser-a . . . . .	28
4.5	Ekran statusu postępu przetwarzania danych . . . . .	29
5.1	Grafiki węzłów (od lewej): artykuł aktywny i wskazany, kategoria aktywna i wskazana .	31
5.2	Poglądowy schemat ilustrujący problem, widok z góry (prawidłowa skala nie została przedstawiona) . . . . .	32
5.3	Wygląd węzłów i połączeń w aplikacji . . . . .	33
5.4	Widok grafu ze wskazanym węzłem i podglądem jego połączzeń . . . . .	34
5.5	Widok węzła z połączonymi i zaznaczonym powiązanym węzłem . . . . .	34
5.6	Nagłówek po wskazaniu węzła . . . . .	35
5.7	Nagłówek w widoku węzła z informacją o połączeniach . . . . .	35
5.8	Nagłówek z ikoną stanu sygnalizującą trwające automatyczne przemierzanie trasy . . . . .	36
5.9	Jedna z infografik dostępnych w przestrzeni pomocy . . . . .	36
5.10	Schemat interakcji przewijania historii . . . . .	37
5.11	Fragment przykładowego pliku .wgr zawierającego trasę . . . . .	38
5.12	Fragment interfejsu wyboru przycisków dla środowiska PC . . . . .	39
5.13	Diagram kolejności wykonywania modułów przy akcji użytkownika . . . . .	40
5.14	Zrzut ekranu konsoli z otwartą zakładką Search . . . . .	42
5.15	Zrzut ekranu konsoli z otwartą zakładką Routes i włączoną trasą . . . . .	42
5.16	Aktualny schemat sterowania kontrolerem . . . . .	44
6.1	Okulary umożliwiające widzenie w 3D . . . . .	45
6.2	Ekran pomocy zawierający informacje o sterowaniu . . . . .	47
6.3	Wskazywanie artykułu w trybie swobodnego lotu . . . . .	47
6.4	Widok węzła dla wybranego artykułu . . . . .	48
6.5	Widok węzła dla wybranej kategorii . . . . .	48
6.6	Wyświetlanie prawidłowego kształtu dla węzła znajdującego się na dwóch ścianach jaskini	49
6.7	Wskazywanie kategorii w widoku węzła . . . . .	49
6.8	Ikona informująca o otwartej konsoli operatora . . . . .	50
6.9	Konsola operatora na głównym komputerze w dużej jaskini . . . . .	50

## **SPIS TABLIC**

1.1	Zestawienie interesariuszy i użytkowników projektu . . . . .	10
4.1	Reprezentacja pojedynczego węzła w pliku .wgm . . . . .	25
4.2	Reprezentacja pojedynczego węzła w pliku .wgg . . . . .	25
4.3	Reprezentacja pojedynczego węzła w pliku .wgs . . . . .	26

## **SPIS LISTINGÓW**

4.1	Fragment informacji o ostatnim zrzucie bazy danych polskiej Wikipedii . . . . .	21
4.2	Fragment pliku enwiki-20191101-page.sql zawierający dane o stronach . . . . .	22
4.3	Słownik przechowujący odwzorowanie odwrotne . . . . .	24
4.4	Przykładowy fragment pliku pagelinks.map . . . . .	25
4.5	Odwzorowanie odwrotne z listingu 4.4 (fragment R_pagelinks.map) . . . . .	25
4.6	Pomocnicze struktury dla programu generującego pliki dla aplikacji . . . . .	26
5.1	Model węzła grafu . . . . .	30
5.2	Interfejs UserAction . . . . .	37
5.3	Pomocnicze struktury i zmienne kontrolera zarządzania pamięcią . . . . .	40