

Streszczenie

Spis treści

1	Wstęp i cele pracy	7
2	Stan wiedzy	9
2.1	Układ nerwowy	9
2.1.1	Komórka nerwowa (neuron)	9
2.1.2	Podział układu nerwowego	10
2.2	Zanikowe stwardnienie boczne (ALS)	11
2.2.1	Opis jednostki chorobowej	11
2.2.2	Próby terapii chorych.	12
2.3	Augmentative and Alternative Communication System	13
2.4	Technologiczne rozwiązania dla chorych na ALS	15
2.4.1	Eye-tracking	15
2.5	Podsumowanie	17
3	Analiza projektowa	19
3.1	Wybrane technologie	19
3.1.1	C++	19
3.1.2	QT	20
3.2	Architektura systemu	21
3.2.1	Architektura aplikacji	21
3.2.2	Komunikacja aplikacji	22
3.3	Wymagania funkcjonalne	23
3.4	Przypadki użycia	23
3.5	Propozycja rozwiązania	25
3.5.1	Prototyp interfejsu oprogramownia	25
3.6	Projekt testów	28
3.7	Możliwość dalszego rozwoju	28
3.8	Wymagania pozafunkcjonalne	29
3.8.1	Ograniczenia	29

3.9	Podsumowanie	29
4	Implementacja	31
4.1	Diagram klas	31
4.2	Opis klas zastosowanych	32
4.3	Zaimplementowane algorytmy	34
4.3.1	Działanie przycisków	34
4.3.2	Praca z tekstem	38
4.3.3	Trie tree	40
4.3.4	Dynamiczna zmiana czasu progowego fiksacji	42
4.3.5	Korzystanie z wyszukiwarki internetowej	43
4.4	Zaimplementowany interfejs użytkownika	45
4.5	Podsumowanie	49
5	Testy i wyniki	51
6	Podsumowanie	53

Rozdział 1

Wstęp i cele pracy

Rozdział 2

Stan wiedzy

W tej sekcji zostanie przedstawiony stan wiedzy dotyczący neurologii, choroby zanikowego stwardnienia bocznego oraz rozwiązań technologicznych - w tym eye-trackingu.

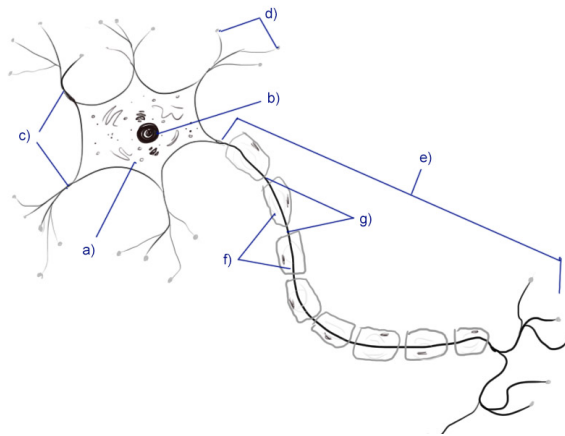
2.1 Układ nerwowy

Układ nerwowy jest to zespół narządów służących do odbierania, przetwarzania i przewodzenia bodźców z środowiska zewnętrznego oraz wewnętrznego do narządów wykonawczych - mięśni oraz gruczołów. Zapewnia on łączność organizmu ze środowiskiem zewnętrznym, a także reguluje działalność komórek organizmu poprzez współpracę z układem hormonalnym i krwionośnym. Podstawową jednostką budulcową jest neuron.

2.1.1 Komórka nerwowa (neuron)

Neuron jest jednostką morfologiczno-czynnościową układu nerwowego zbudowaną według schematu przedstawionego na rys. 2.1. Dendryty odpowiedzialne są za przewodzenie impulsów elektrycznych pobranych przez synapsy w kierunku somy - ciała komórki nerwowej. Wyprowadzeniem impulsu zajmuje się akson - wypustka osiowa. Kierunek przebiegu jest zawsze stały - mówi o tym prawo laryzacji dynamicznej. [2]

Przemieszczanie się impulsu elektrycznego zachodzi dzięki pompie sodowo-potasowej, która utrzymuje w komórce stałe napięcie powierzchniowe na błonie komórkowej. Potencjał spoczynkowy dla komórek nerwowych mieści w zakresie od -60mV do -90mV , natomiast gdy dochodzi do pobudzenia komórki błona posiada potencjał czynnościowy, który mieści się między $+20\text{mV}$ do $+50\text{mV}$ [1]. Właśnie dzięki postępującej różnicy potencjałów dochodzi do przewodzenia impulsu od dendrytu do aksonu i tym sposobem do kolejnego neuronu lub narządu wykonawczego.



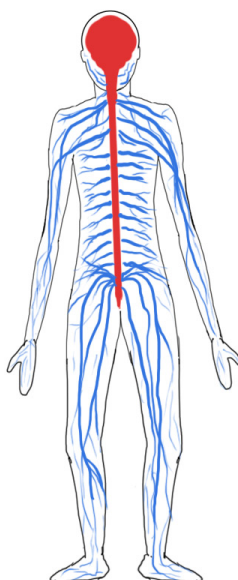
Rys. 2.1 Grafika przedstawiająca schematyczną budowę neuronu na podstawie danych z [1].

a) Soma/perikaryon - ciało neuronu zawierające SER, RER, aparaty Golgiego, neurotubule i neurofilamenty, rybosomy, mitochondria, a także b) jądro komórkowe, c) dendryty, d) synapsy, e) neuryt/axon, f) osłonka mielonowa, g) przewężenia Ranviera

2.1.2 Podział układu nerwowego

Układ nerwowy można podzielić najogólniej na układ somatyczny oraz autonomiczny. W skład pierwszego wchodzi ośrodkowy układ nerwowy oraz obwodowy układ nerwowy. Ośrodkowym układem nerwowym nazywa się mózgowie wraz z rdzeniem kręgowym, a obwodowym nerwy oraz zwoje. Schematyczny obraz układu nerwowego przedstawiono na rys. 2.2 - kolorem czerwonym centralny układ nerwowy, niebieskim obwodowy układ nerwowy. Układ autonomiczny składa się z układu współczulnego i przywspółczulnego.

Nerwy z kolei dzieli się ze względu na kierunek przewodnictwa impulsu nerwowego na: nerwy ruchowe (tzw. motoneurony) oraz czuciowe i mieszane (ruchowo-czuciowe). Zgrupowanie komórek nerwowych poza ośrodkowym układem nerwowym nazywa się zwojem.



Rys. 2.2 Grafika przedstawiająca schematyczną budowę układu nerwowego.

2.2 Zanikowe stwardnienie boczne (ALS)

2.2.1 Opis jednostki chorobowej

Zanikowe stwardnienie boczne (ang. amyotrophic lateral sclerosis ALS, łac. sclerosis lateralis amyotrophica SLA) jest rzadką chorobą neurologiczną, dotyczącą w 75% przypadków zachorowań mężczyzn między 40 a 65 rokiem życia. [1] Jednakże termin ALS jest używany do określania więcej niż jednego schorzenia związanego z degeneracją neuronów ruchowych. Najczęściej stosowany jest, gdy mowa jest o klasycznej formie ALS (choroba Charcota), ale także w przypadku: Postępującego Porażenia Opuszkowego (Progressive bulbar palsy (PBP)), Postępującego Zaniku Mięśni (Progressive muscular atrophy (PMA)), Pierwotnego Stwardnienia Bocznego (Primary lateral sclerosis (PLS)), Syndromu Ramienia Cepowatego (Flail arm syndrome (Vulpian-Bernhardt syndrome)), Syndromu Nogi Cepowatej (Flail leg syndrome (Pseudopolyneuritic form)) oraz ALS z powikłaniami wielonarządowymi (np. ALS Dementia) [3]. W celu ujednolicenia nazewnictwa Lord Russell Brain zaproponował termin Chorób Neuronu Ruchowego (Motor Neurone Disease (MND)). [3]

Istotą choroby jest zwyrodnienie komórek nerwowych odpowiedzialnych za przekazywanie sygnałów, wynikiem czego jest atrofia unerwianych przez nie mięśni. Degenerują neurony kory motorycznej (górny motoneuron), oraz neurony ruchowe rogów brzusznych rdzenia kręgowego, lub rdzenia przedłużonego (opuszki) (dolny motoneuron). Objawia się to początkowo ich osłabieniem, a ostatecznie zanikiem (przy czym same komórki mięśniowe nie obumierają, a jedynie zmniejszają swój rozmiar), co skutkuje ograniczeniem ruchów chorego, między innymi: gryzienia, chodzenia, mówienia i oddychania. ALS nie atakuje układu pokarmowego oraz krwionośnego. Swoją funkcjonalność stosunkowo długo zachowują neurony jądra

Onufa unerwiające pęcherz moczowy i jądro nerwu okoruchowego odpowiedzialnego za ruchy gałek ocznych. Drogi czuciowe i sprawność intelektualna są zachowane. [4] Powoduje to, że pacjent jest więźniem we własnym ciele - często nie mogącym się skomunikować z otoczeniem. Co więcej ciągła świadomość pogarszającego się stanu chorego znacząco wpływa na jego samopoczucie i zdrowie psychiczne, prowadząc do depresji.

Nieznane są przyczyny zachorowań, jednak autorzy publikacji [3] oraz [5] są zdania iż ALS jest wynikiem wzajemnego oddziaływania wielu czynników takich jak: czynniki genetyczne (między innymi dziedziczny defekt genu odpowiedzialnego za syntezę enzymu - desmutazy ponadtlenu Cu/Zn. [[1]- [3]]), ekscytotoksyczność (patologiczny proces, w którym neurony są uszkodzane i zabijane przez aminokwasy o charakterze kwasowym m.in. glutaminian), stres oksydacyjny (gromadzenie reaktywnych form tlenu, na skutek zakłócenia równowagi pomiędzy ich wytwarzaniem, a biologiczną zdolnością do szybkiej detoksykacji, co prowadzi do obumierania komórek), dysfunkcje mitochondrialne (nieprawidłowości zarówno w ich budowie jak i funkcji), ograniczony transport aksonalny, anormalne gromadzenie neurofilamentów (to grupa białek włóknkowych stanowiących jeden z głównych komponentów cytoszkieletu komórek nerwowych), agregacja białek w cytoplazmie w postaci wtrętów, zaburzenia układu odpornościowego, reakcje autoimmunologiczne oraz niedobór neurotransmiterów. ALS jest chorobą postępującą, co oznacza, że wraz z biegiem czasu następuje nasilenie objawów, w tym całkowita utrata kontroli nad ruchami dobrowolnymi. Na dzień dzisiejszy jest nieuleczalna i prowadzi do śmierci chorego w przeciągu 1-2 lat (25% chorych). Jednak należy pamiętać, że SLA w swoim przebiegu jest chorobą bardzo indywidualną, więc tzw. okres przeżycia może wahać się w szerokich granicach. Ponad 25% chorych przeżywa więcej niż 5 lat, w tym ok. 5% powyżej 10 lat . [6]

2.2.2 Próby terapii chorych.

Do tej pory nie został znaleziony sposób skutecznego leczenia przyczynowego ALS. Jedynym lekiem o sprawdzonej skuteczności i dopuszczonym do stosowania w terapii jest Rilutek (Riluzole, benzotiazol). Autorzy książki [7] poświęconej schorzeniu donoszą, że „Badania naukowe dowodzą, że leczenie Riluzolem w dawce 100 mg dziennie (2x50 mg) przez 18 miesięcy nieznacznie wydłuża przeżycia chorych (średnio 3 – 4 miesiące) niestety nie wpływając na poprawę ich stanu klinicznego i jakości życia.”. Terapia farmakologiczna chorych na ALS sprowadza się więc do leczenia objawowego.

Innym ważnym aspektem terapii chorego jest odpowiednia rehabilitacja. Zwiększenie zakresu ruchów chorego oraz jak najdłuższe utrzymanie jego zdolności do samodzielnego wykonywania codziennych czynności jest głównym zadaniem specjalistycznej rehabilitacji ruchowej, która powinna być prowadzona od momentu diagnozy. Przykładowo wykonywane są: ćwiczenia w sali gimnastycznej, zajęcia ruchowe w basenie wzmacniające odpowiednie grupy mięśni, ćwiczenia czynne na sali chorych (chodzenie w balkonikach, stabilizatorach), czy też ćwiczenia bierne w łóżku chorego połączone z masażem leczniczym u pacjentów bez spastycznego napięcia. [7] Korzystne efekty mogą wywoływać też takie zabiegi jak hydroterapia, kąpiele ciepłe, elektroterapia, krioterapia. [6]

Z powodu utraty kontroli nad aparatem mowy komunikacja z chorym może stać

się ograniczona, lub wręcz niemożliwa, dlatego lekarze, logopedzi i opiekunowie pacjenta mają za zadanie jak najdłuższe utrzymanie komunikacji. Mowa tu nie tylko o komunikacji werbalnej, ale o nowych metodach wypracowywanych indywidualnie przez chorego i opiekuna. Przy narastających zaburzeniach dyzartrycznych wykorzystuje się metody systemu ACC (ang. Augmentive and Alternative Communication System) czyli wspomagającego zastępczego systemu komunikacji, zwiększającego możliwości komunikacyjne. [7]

2.3 Augmentative and Alternative Communication System

Augmentative and alternative communication (AAC) jest dziedziną badań klinicznych oraz edukacyjnych. Głównym założeniem jest próba zgłębienia oraz kompensacji, gdy to konieczne, tymczasowych lub trwałych upośledzeń, czy też ograniczeń ze względu na możliwość posługiwania się oraz rozumienia komunikacji werbalnej oraz pisemnej. Najczęstszymi przyczynami sięgania do tych metod komunikacji są: upośledzenia umysłowe, mózgowie porażenie dziecięce, autyzm oraz postępująca apraksja mowy [8] (jak w przypadku ALS). Zadaniem AAC nie jest jedynie wprowadzenie możliwości komunikacji osób chorych z otoczeniem, ale także umożliwienie im czynnego udziału w życiu społecznym oraz towarzyskim, wykonywania czynnego zawodu, czy też poświęcaniu się hobby jak pisanie prozy, bądź poezji, co w sposób znaczący odbija się na ich samopoczuciu oraz jakości życia. Dąży się zatem do zapewnienia tego podstawowego prawa każdemu pacjentowi.

ACC polega na wymianie wiadomości, które muszą zostać sformułowane, przechowane oraz odzyskane w dowolnym momencie. Wiadomością może być pojedyncze słowo, kod lub też struktury bardziej złożone mające na celu podtrzymanie komunikacji interpersonalnej, pisanej – także tej na mediach społecznościowych, które w dzisiejszych czasach stanowią ważny filar interakcji międzyludzkich.

Istnieje kilka metod formowania pełnych zdań : jedną z nich jest wpisywanie wiadomości znak po znaku, inną jest korzystanie z gotowych bibliotek wyrazów. Możliwe jest także wykorzystanie już gotowych wzorów pełnych, lub części zdań – co zdecydowanie przyspiesza komunikację, jednak metoda ta jest ograniczona przez pamięć biblioteki przechowującej takie wzorce. Biblioteka taka tworzona i kompletowana jest zgodnie z indywidualnymi potrzebami chorego i zależy od jego płci, wieku, zawodu, stylu życia itp. Dla dzieci przygotowywane są specjalne tablice obrazkowe, które pozwalają im na jasny przekaz.

Tablice można podzielić ze względu na ułożenie ich obiektów na statyczne oraz dynamiczne. W tablicach statycznych każdy obiekt ma swoje ustalone położenie, które nie zmienia. Do komunikacji niezbędna jest więc duża ilość statycznych tablic – przykładem są tablice fizyczne – papierowe, gdzie nie można zmieniać położenia wydrukowanych na kartce elementów. Dynamiczne wyświetlacze odnoszą do się wyświetlaczy komputerowych, w których dzięki odpowiedniemu oprogramowaniu, można dynamicznie poruszać się między widokami. W celu ułatwienia korzystania z bibliotek zebranych w tablice istnieje kilka metod przedstawionych w [8].

1. Semantic-Syntactic Grid Displays (Semantyczno-syntaktyczna siatka/tablica) – tablica cechująca się podziałem wyświetlanych na niej słów na kategorie, zazwyczaj ze względu na części mowy. Możliwe jest także przedstawienie słów w logicznej kolejności występowania w zdaniu. Przykładowym algorytmem sortowania jest klucz Fitzgeralda – słowa układa się od lewej do prawej, układając je w klasy odpowiadające na pytania: „Kto?”, „Co robi?”, „Co?”, „Gdzie?”, „Jaki?”, „Kiedy?” itd. Słowa, bądź litery najczęściej wybierane układane są w pierwszym bądź ostatnim wierszu tablicy. Elementem stanowczo usprawniającym korzystanie było kolorowanie elementów zgodnie z ich przynależnością do klasy.
2. Taxonomic Grid Displays (Siatka/tablica taksonomiczna) – dzieli słownictwo na klasy: osoby, miejsca, uczucia, jedzenie, napoje, słowa opisujące czynności. Nie jest to rekomendowany typ tablic dla dzieci w wieku poniżej 6 roku życia.
3. Activity Grid Displays (Tablica/siatka czynności) – najpopularniejszy typ tablic. Kategoryzacja słownictwa następuje ze względu na rodzaj czynności, wydarzenia, czy rutyny, której ono dotyczy np. „zakupy” lub „płacenie przy kasie”, czy też „rozmowa ze sprzedawcą”. Każda kategoria jest podzielona jak wyżej na mniejsze podklasy ze względu na to, co dane słowo określa.
4. Pragmatic Organization Dynamic Display (Dynamiczny wyświetlacz struktur pragmatycznych) – metoda kładącą duży nacisk na efektywność komunikacji. W tym celu stosuje się kombinację kilku różnych strategii organizacji słownictwa.
5. Visual Scene Displays (Wyświetlacz obrazkowy) – słownictwo posortowane jest według przynależności do czynności lub rutyny, które zobrazowane są w postaci symbolu kojarzącego się z nimi. Rozmieszczenie tych grafik nie jest siatkowe, a metodyczne. Takie rozwiązanie wykazało się wyjątkowo skuteczne w pracy z małymi dziećmi, które w bardzo krótkim czasie opanowały taką formę komunikacji. Zauważono także większą chęć interakcji w przypadku korzystania z tej formy niż w przypadku korzystania z tablic.
6. Hybrid Display (Wyświetlacz hybrydowy) – połączenie metody wyświetlacza obrazkowego oraz tablic. Przedstawia się zdjęcie, na którym zaznaczone są elementy i wyświetlane związane z nimi słowa np. na obrazku widać osobę, więc wyświetlane będą np. części ciała, czynność jaka ta osoba wykonuje, emocje itp. ...

Ze szczególnym przypadkiem, który wymaga osobnego potraktowania, mamy do czynienia w przypadku osób dorosłych z nabytą niepełnosprawnością - jak ma to miejsce w wypadku chorych na ALS. 93% osób chorujących jest uzależniona od metod AAC. Dorośli potrzebują więcej czasu, żeby przestawić się i przyzwycząić do niewerbalnej komunikacji, dlatego też zalecana im jest nauka korzystania np. z tablic od razu po diagnozie, nawet jeśli chory jest jeszcze w pełni sprawny. Pogorszenie stanu zdrowia w wypadku tej jednostki może nastąpić w bardzo krótkim czasie.

2.4 Technologiczne rozwiązania dla chorych na ALS

Jak wyżej wspomniano, do rozwiązań dla chorych na ALS należą tablice statyczne oraz dynamiczne, jednak aby się nimi posługiwać należy albo opracować metodę AAC np. dwa mrugnięcia oznaczają tak, a jedno nie, lub wykorzystać sygnały płynące z organizmu ludzkiego jako wskaźniki. Do mierzonych wartości należą m.in. impulsy elektryczne pochodzące z mózgu (mierzone za pomocą EEG), mięśni (mierzone za pomocą EMG) lub detekcja mikroruchów kończyn. [9] Ze względu na to, że technologie wykorzystujące EEG wymagają bardzo stabilnych warunków elektromagnetycznych i są niezwykle czułe na szum, nie są optymalnymi rozwiązaniami dla chorych na ALS. Metodą bardzo dobrze się sprawdzającą, ze względu na długie zachowanie funkcjonalności gałek ocznych, jest EGT, czy też eye-tracking, czyli śledzenie wzroku. Jest to metoda bardziej niezawodna niż wyżej wymienione.

2.4.1 Eye-tracking

Eye-tracking, znany również jako okulorografia, ma za zadanie np. określenie ruchów gałek ocznych, a przez to punktu fiksacji wzroku, najczęściej w czasie rzeczywistym. Do metod pozwalających na śledzenie wzroku oraz ruchów oczu należy m.in. elektrookulografia, śledzenie ruchu gałek ocznych, powiek, metody działające w oparciu o szkła kontaktowe, metody wykorzystujące refleksy na rogówce lub źrenicy [9]. Do poprawnego działania wymienionych metody najczęściej niezbędny jest również pomiar ruchów głowy.

Technologia ta wykorzystywana jest często podczas badań marketingowych do określenia skuteczności reklamy – ustalenia, co przyciąga największą uwagę odbiorcy, bądź w celu orzeczenia poziomu użyteczności zaprojektowanego interfejsu. Coraz częściej implementuje się dane techniki np. w grach, w medycynie m. in. w celu umożliwienia korzystania z urządzeń ekranowych osobom z upośledzeniami fizycznymi, które ich przed tym powstrzymują oraz wszędzie tam, gdzie użytkownik nie może mieć zajętych rąk. We współpracy z urządzeniami elektronicznymi takimi jak komputer lub urządzenia mobilne osoba upośledzona, bądź chora może w sposób znaczący poprawić swoją jakość życia. W celu sprawnej współpracy niezbędne jest dokładne określenie punktu fiksacji wzroku (punktu, na którym użytkownik skupia swój wzrok przez co najmniej 0,15s) [10].

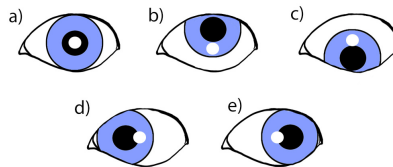
Podział urządzeń do eye-trackingu

Z przykładem bardzo dobrego podziału urządzeń można zapoznać się czytając materiały wykładowe dr inż. Bartoszka Kunki [11]. Według niego pierwszym kryterium podziału powinna być mobilność urządzenia pomiarowego na: mobilne i niemobilne, gdzie do pierwszych zaliczyć można wszystkie urządzenia nagłowne np. smartglasses, a do drugich urządzenia stacjonarne – związane na stałe np. z monitorem lub zamontowane na stojakach – mam wtedy do czynienia najczęściej z urządzeniami bezdotykowymi. Urządzenia mobilne bazują na technice wykorzystującej podczerwień, natomiast stacjonarne mogą, prócz wyżej wspomnianej, monitorować badane parametry również niezależnie od światła

IR – wykorzystując światło dzienne.

Działanie urządzeń opartych na oświetleniu w podczerwieni. [[11], [12]]

W celu obserwacji ruchów gałki ocznej i określenia punktu fiksacji należy oko naświetlać światłem punktowym w podczerwieni albo w linii zgodnej z osią kamery, bądź poza nią i w zależności od wybranej metody zauważa się inne zjawiska optyczne. Podczerwień powoduje znaczny wzrost kontrastu między tęczówką, a źrenicą oka, co pozwala na dokładniejsze określenie pozycji źrenicy, co wyznaczane jest na podstawie odbicia obserwowanego na nagraniu tworzonym przez kamerę, która przez cały czas działania aparatury skierowana jest na jedno z oczu osoby badanej [12]. Wśród obserwowanych odbić najbardziej istotnym względem badanego aspektu jest tzw. glint (błysk na rogówce, powstający, gdy dioda jest poza osią kamery). Glint, znany również jako pierwszy obraz Purkiniego [[11], [12]], nie zmienia swego położenia wraz z ruchami gałki ocznej – dlatego uważany jest jako punkt referencyjny, tak długi, jak głowa badanej osoby pozostaje nieruchoma względem kamery. Diody umieszczone na osi kamery wywołują zjawisko jasnej źrenicy [11] – część podczerwonego światła przedostaje się do źrenicy i zostaje od niej odbita, wywołując efekt przypominający np. kocie oko oświetlone w ciemności. Porównując więc pozycję glinta (lub glintów) oraz ruchomej jasnej źrenicy jesteśmy w stanie określić punkt fiksacji oka na podstawie ich względnego położenia. Określenie kierunku skierowania wzroku na podstawie względnego położenia glintu oraz źrenicy przedstawiono na rysunku 2.3. Schemat algorytmu wyznaczania punktu fiksacji przedstawiono na rysunku 2.4 opartym na materiałach wykładowych [11].

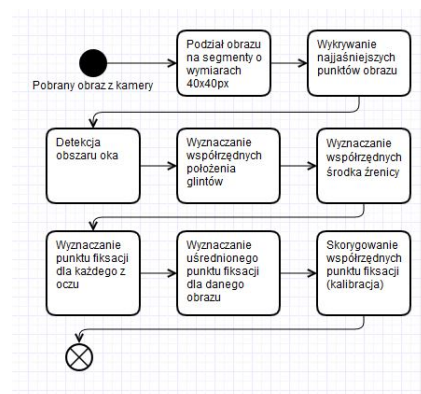


Rys. 2.3 Położenie glinta względem środka źrenicy w zależności od ruchu gałki ocznej.

a) patrzenie prosto w źródło światła b) patrzenie powyżej źródła światła c) patrzenie poniżej źródła światła d) patrzenie w lewo od źródła światła e) patrzenie w prawo od źródła światła

Odległości wyliczane są na podstawie obrazów przechwyconych przez kamerę i przetworzonych komputerowo. Tą samą technologię można również zmodyfikować, by wyznaczać punkt fiksacji z położenia 4 glintów (wywołanych 4 diodami rozmieszczonymi w narożnikach ekranu) lub przy zastosowaniu skomplikowanych przekształceń matematycznych, poprawiających dokładność wyznaczenia punktu fiksacji. [11]

Inną metodą – wykorzystującą na przemian dwa rodzaje diod (na lub poza osią kamery) jest metoda różnicowa [11]. Wykorzystuje się kolejne klatki nagrania, przy czym jedna klatka zawiera efekt jasnej źrenicy, a druga zawiera efekt ciemnej źrenicy i glinty. Punkt fiksacji po raz kolejny jest wynikiem różnicy odległości obu zjawisk i obliczany jest na podstawie wynikowego obrazu różni-



Rys. 2.4 Poglądowy schemat algorytmu wyznaczania punktu fiksacji przy wykorzystaniu światła IR.

cowego.

Przykładem technologii, która do tej pory wykorzystwała powyższy algorytm jest np. Erica [12].

Działanie urządzeń opartych na świetle dziennym.

W przypadku metod wykorzystujących światło dzienne i kamerę rejestrującą ruchy gałek ocznych mamy do czynienia z bardzo złożonymi algorytmami komputerowymi, które pozwalają na przetwarzania obrazu powodującego określenie punktu fiksacji wzroku. Wraz z zaletą, jaką jest brak zapotrzebowanie na dodatkowy sprzęt (algorytmy mogą współpracować z wbudowanymi kamerkami komputerowymi), technologie te, niestety, często charakteryzują się gorszą dokładnością.

Przykładem technologii korzystającej ze światła dziennego jest Eye Mouse – projekt powstały na terenie Politechniki Gdańskiej [9], którego głównym założeniem jest wykorzystanie wzroku jako zastępstwo myszy komputerowej oraz klawiatury. W tym celu wykorzystuje się dwie kamery: jedną śledzącą wzrok, a drugą rejestrującą położenie głowy osoby badanej względem ekranu – tak, że na otrzymywany wynik nanoszona jest korekta. W rogach ekranu umieszczone są 4 diody światła podczerwonego jako znaczniki. W procesie przetwarzania obrazów usuwany jest szum oraz wyznaczana jest źrenica oraz jej środek. Kolejnym krokiem jest oznaczenie wzajemnej zależności pozycji ekranu względem głowy osoby badanej – w tym celu wykorzystuje się znaczniki LED. Proces ten niezbędny jest w celu kalibracji.

2.5 Podsumowanie

Zapoznawszy się z problematyką choroby zanikowego stwardnienia bocznego oraz z możliwościami technologicznymi oferowanymi pacjentom można przejść do stworzenia aplikacji komputerowej, która ma na celu ułatwienie komunikacji chorych z komputerem. Zastosuje się do tego metodę eye-trackingu korzystającej ze światła dziennego - zwiększając w ten sposób dostępność oraz uniwer-

salność aplikacji.

Rozdział 3

Analiza projektowa

Rozdział poświęcony jest analizie wybranych technologii oraz wymagań projektu. Przedstawiono również wstępną analizę dotyczącą przypadków użycia oraz metod testowania produktu.

3.1 Wybrane technologie

Krótki opis teoretyczny wybranych technologii - C++ oraz bibliotek QT.

3.1.1 C++

C++ jest powszechnie stosowanym językiem programowania, będącym potomkiem języka C. Wprowadzono w nim szereg udogodnień. C++, w porównaniu z C, zapewnia dokładniejsze sprawdzanie typów danych, wspiera abstrakcje, programowanie obiektowe (z tego względu mówi się o nim jako o języku pseudo-obiektowym), programowanie uogólnione i proponuje więcej stylów programistycznych. Do wspieranych założeń programowania obiektowego należą polimorfizm, enkapsulacja i dziedziczenie. Nowszą wersją C - C++, posiada również bardzo dużą ilość bibliotek, których wykorzystanie znacznie ułatwia jego implementację. W skład klasy tego języka wchodzi plik wykonywalny (*.cpp) oraz nagłówkowy (*.h). W plikach nagłówkowych klas definiuje się wszystkie metody oraz zmienne. Głównymi powodami do tworzenia oddzielnych plików nagłówkowych są [13]:

- przyspieszenie czasu kompilacji programu. Pliki nagłówkowe nie wymagają re-kompilacji, o ile nie nastąpiła w nich zmiana, co skraca czas kompilacji w czasie rozwijania dużych projektów,
- organizacja struktury kodu,
- rozdzielenie interfejsów od implementacji.

Przykładowy plik nagłówkowy klasy ButtonOperator:

```

#ifndef BUTTONOPERATOR_H
#define BUTTONOPERATOR_H
    using namespace std;
#include <QtWidgets>
#include <QStringList>

    class ButtonOperator : public QPushButton
    {
        Q_OBJECT

    public:
        explicit ButtonOperator(QWidget *parent=0);
        void setDisplayList(QStringList sDisplayList);
        QStringList getDisplayList();
        void setSpecial(bool sIsSpecial);
        bool getSpecial();
        bool getHover();

    protected:
        void hoverEnter(QHoverEvent *event);
        void hoverLeave(QHoverEvent *event);
        void hoverMove(QHoverEvent *event);
        bool event(QEvent *event);

    private:
        bool isSpecial;
        bool isHovered;

        QStringList displayList;

    };

#endif // BUTTONOPERATOR_H

```

Widać w nim jasno wydzielone sekcje modyfikatorów, typy zmiennych i metod oraz ich argumenty.

3.1.2 QT

QT jest zespołem przenośnych bibliotek oraz narzędzi programistycznych napisanych w C++ do tworzenia aplikacji desktopowych, zagnieżdżonych, a także mobilnych. Wspiera on takie systemy jak: Linux, OS X, Windows, xWorks, QNX, Android, iOS, BlackBerry, Sailfish OS i dzięki czemu jest nadzwyczaj uniwersalnym narzędziem kompatybilnym z następującymi językami programowania: C++,

QML (QT Modeling Language), Python, Ring, Go, Rust, PHP i Java. [14] [15] QT zapewnia listę dodatkowych możliwości rozszerzających C++. Są to między innymi:

- mechanizmy pozwalające na komunikację między obiektami, zwane sygnałami i otworami (slots);
- wyjątkowa możliwość edycji wyglądu i responsywności obiektów ;
- swobodna możliwość edycji zachowań w przypadku różnego rodzaju zdarzeń ;
- kontekstowe tłumaczenie stringów do internacjonalizacji;
- hierarchiczne i responsywne drzewa obiektowe, organizujące strukturę obiektów;
- automatyczna zmiana wartości wskaźników na 0 w przypadku zniszczenia obiektu, w przeciwieństwie do wskaźników w C++, które stając się zawieszonymi (dangling pointers) [15];

3.2 Architektura systemu

System składa się z warstwy aplikacji oraz warstwy transportowej. Komunikacja zrealizowana jest za pomocą broadcastu przy pomocy protokołu UDP (user data protocol).

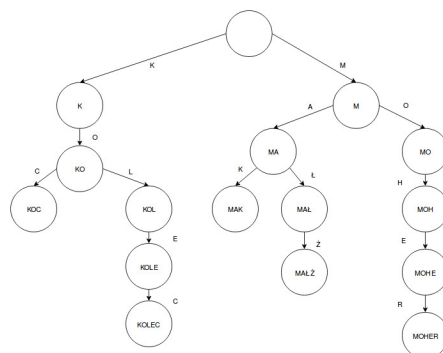
3.2.1 Architektura aplikacji

Aplikację zrealizowano w wyżej wymienionym języku C++ z wykorzystaniem wielu bibliotek QT np. QWidget, QString itd.. Do funkcji autouzupełniania tekstu posłużono się algorytmem wykorzystującym skompresowane drzewo trie.

Zkompresowane Drzewo Trie

Drzewo Trie jest drzewem służącym docelowo do sortowania wartości danych tekstowych, w którym każdemu węzłowi przypisany jest wspólny prefix(fragment klucza). Wartości tekstowe zapisywane są w liściach drzewa. [16] Do drzewa wczytywane zostają wszystkie słowa z danego mu słownika, a następnie każde rozkłada się na litery i rozmieszcza w drzewie, tak by czytając znaki od korzenia do liścia tworzyły pożądaną ciąg. Biorąc na przykład słownik składający się z następujących słów: mak, róża, kolec, małż, koc, moher otrzymujemy następujące drzewo 3.1. W ten sposób poszukując słów zaczynających się na "ko" w bardzo prosty sposób możemy określić, że zaliczają się do nich koc oraz kolec. Drzewo to w znaczący sposób skraca czas przeszukiwania dużych słowników (w wypadku tego projektu 3639970 wyrazów) i umożliwia np. autouzupełnianie albo korektę słów bieżąco wpisywanych przez użytkownika.

Zastosowany algorytm wykorzystuje fakt występowania wspólnych węzłów i zapamiętuje jedynie numer ostatniego węzła związanego z wyszukiwanym słowem



Rys. 3.1 Przykładowe drzewo typu Trie.

np. dla frazy "ko" - węzeł nr. 2, a następnie pobiera wartości wszystkich dzieci tego węzła, zespala je i tworzy wszystkie możliwe końcówki (tu "c" oraz "lec"), które w połączeniu z szukaną frazą dają "autouzupełniane" słowa.

3.2.2 Komunikacja aplikacji

System umożliwia komunikację poprzez broadcast z innymi użytkownikami oraz poprzez specjalne API Google z przeglądarką internetową.

Broadcast

Broadcast jest metodą komunikacji (rozsyłania danych) między jednym nadawcą, a wieloma odbiorcami w tym samym czasie. Jest to dyfuzyjny, jednokierunkowy tryb transmisji, charakterystyczny dla sieci LAN. Protokołem regulującym dany ruch sieciowy jest w tym wypadku UDP (User Datagram Protocol). Jest to bezpołączeniowy protokół, który pozwala aplikacjom dobudować własne - niezbędne do poprawnego działania, protokoły. Zapewnia on aplikacji możliwość wysyłania enkapsulowanych datagramów IP bez potrzeby wytworzenia połączenia, co jest jednocześnie dużo szybszą metodą komunikacji, niż tryb synchroniczny. Nie dba on o to, czy wysłane ramki dotrą do odbiorcy w całości, co jest typowe dla komunikacji asynchronicznej. UDP transmituje segmenty złożone z 8-bajowego nagłówka oraz fragment przesyłanych danych, będący formą wiadomości. UDP zapewnia informację o portach źródłowych i docelowych. Port źródłowy jest przydatny w momencie, gdy urządzenie odbiorcy zechce odesłać odpowiedź na otrzymany segment. Rozmiar ramki wynosi od 8-65 515 bajtów. [17]

Google Custom Search

Google Custom Search umożliwia stworzenie własnej wyszukiwarki pozwalającej na przeszukiwanie zarówno stron internetowych jak i obrazów. Możliwe jest zawężenie i personalizacja wyników wyszukiwania np. do wyników pochodzących z konkretnej strony, lub zawierających sprecyzowaną frazę. Google Search Engine występuje w dwóch wersjach: darmowej - Custom Search Engine oraz płat-

nej - Google Site Search. Do potrzeb projektowych wystraczająca jest wersja darmowa, która pozwala na korzystanie z dodatkowych API umożliwiających zwrócenie wyników wyszukiwania w postaci pliku XML czy też JSON. API te upraszczają komunikację aplikacji z przeglądarką do zapytań RESTowych typu get w celu otrzymania uporządkowanej struktury danych. [18]

3.3 Wymagania funkcjonalne

Po przeanalizowaniu problemu klawiatury ekranowej oraz potrzeb osób chorych na ALS stwierdzono następujące wymagania funkcjonalne:

- Klawiatura ekranowa reagująca na sygnał wejściowy będący współrzędnymi punktu fiksacji, które określone są z dokładnością jednego stopnia. Zakładając, że odległość użytkownika od ekranu to 60 cm, wyliczono iż dokładność wyznaczania pozycji to obszar ok. 3,2 cm powierzchni ekranowej. Dla komputera o ekranie 14 cal (monitor testowy), gdzie rozmiar pojedynczego piksela to ok. 0,2269 mm - to, w przybliżeniu, 141px dla każdego przycisku.
- Możliwość autouzupełniania słów za pomocą słownika języka polskiego.
- Możliwość wprowadzenia przycisków w stan nieaktywny.
- Wykorzystywanie polskich znaków w tekście.
- Wysyłanie wiadomości za pomocą broadcastu.
- Wyszukiwanie stron w przeglądarce Google.
- Możliwość konfigurowania wyglądu aplikacji - różne stopnie kontrastu oraz kolorystyki, a także rozmiaru czcionki tekstu wpisywanego.

Szczegółowe przypadki użycia klawiatury zobrazowano w rozdziale "Przypadki użycia" 3.4.

3.4 Przypadki użycia

Na zamieszczonym rysunku 3.2 przedstawiono diagram przypadków użycia przedstawiający możliwe interakcje użytkownika z aplikacją.

Aby rozpocząć pracę z klawiaturą należy odblokować pisanie znaków wybierając przycisk *start*. Główną funkcjonalnością, a zarazem najczęściej zachodzącym przypadkiem użycia będzie wybór przycisku z literą w celu wypisania tekstu na ekranie. Użytkownik ma do wyboru klawiaturę podstawową (z 28 znakami oraz cyframi z zakresu 0-9), 2 tablice przycisków ze znakami specjalnymi oraz jedną poświęconą jedynie polskiemu znakom. Każdy z przycisków reaguje na zmianę położenia punktu fiksacji - w momencie, w którym wykryte zostanie położenie nad przyciskiem naliczany zostaje czas, którego narastanie wizualizuje się za pomocą paska postępu. Gdy 100% postępu zostanie osiągnięte dana litera pojawia się na ekranie w przeznaczonym obszarze i w określonej, za pomocą kursora, pozycji w tekście. W przypadku, jeśli użytkownik zechce zmienić wielkość

3.5 Propozycja rozwiązania

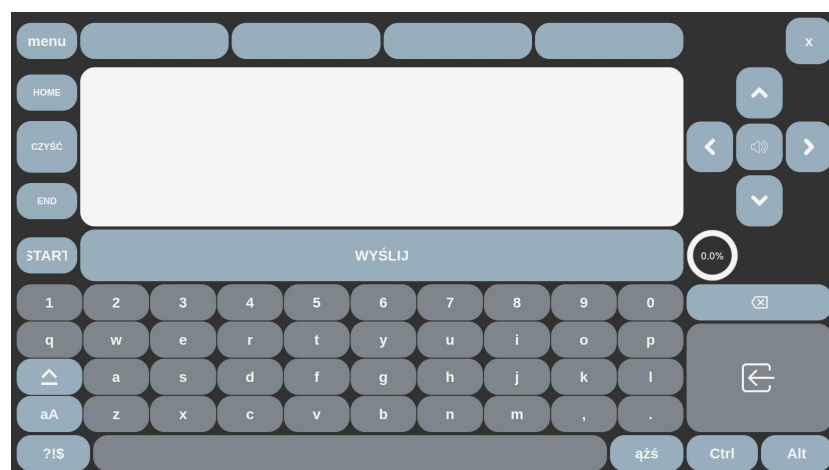
Jedną z możliwych propozycji rozwiązania, którą postanowiono zrealizować jest oprogramowanie komputerowe, które wyglądem przypominać powinno klawiaturę ekranową, czułą na pozycję kursora, z wbudowanym edytorem tekstu. Klawiatura ta powinna realizować żądanie użytkownika w zakresie wyżej wspomnianych wymagań funkcjonalnych tj. pisanie tekstu, wyszukiwanie treści w internecie oraz komunikacją z innymi użytkownikami poprzez broadcast. Co więcej, interfejs użytkownika powinien pozwolić na dostosowanie wyglądu do indywidualnych potrzeb. W wyniku pierwszej analizy powstał przedstawiony w podrozdziale 3.5.1 prototyp oprogramowania.

3.5.1 Prototyp interfejsu oprogramowania

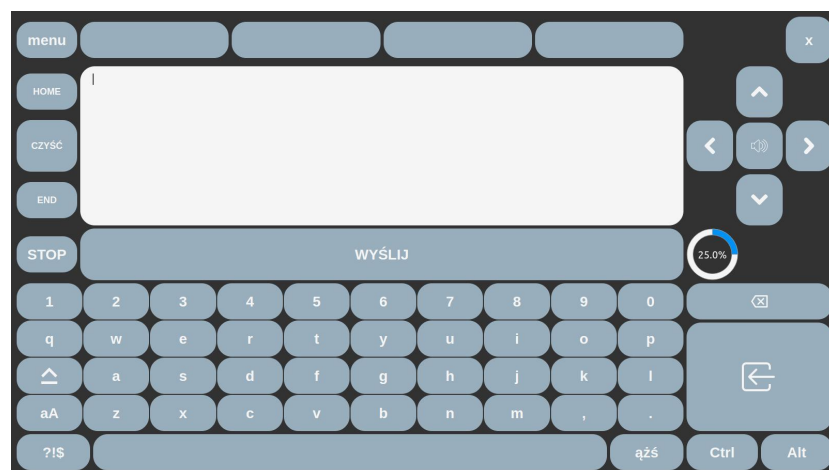
Tworząc prototy oprogramowania zwrócono szczególną uwagę na funkcjonalność interfejsu. Aplikacja otwiera się w trybie pełnoekranowym, tak by otrzymać maksymalny dostępny rozmiar elementów zawartych w interfejsie. Zamieszczono w nim opisane powyżej przyciski niezbędne do poprawnego działania programu. Użytkownik informowany jest o pozostałym czasie do wyboru przycisku poprzez zmieniający się pasek postępu.

Kolorystyka aplikacji została dobrana tak, by chory na stwardnienie boczne rozsiane mógł korzystać z niej nie męcząc oczu podczas długiego korzystania. Połączenie kolorów białego i niebieskiego daje wystarczający kontrast dla ludzkiego oka, by przy niewielkim wysiłku odczytać zapisany tekst. Nie można było zastosować interfejsów czarno-białych o najwyższym poziomie kontrastu ze względu na to, iż obłe kształty przycisków w tych kolorach powodowały powstawanie złudzenia optycznego, które uniemożliwiało pracę na klawiaturze.

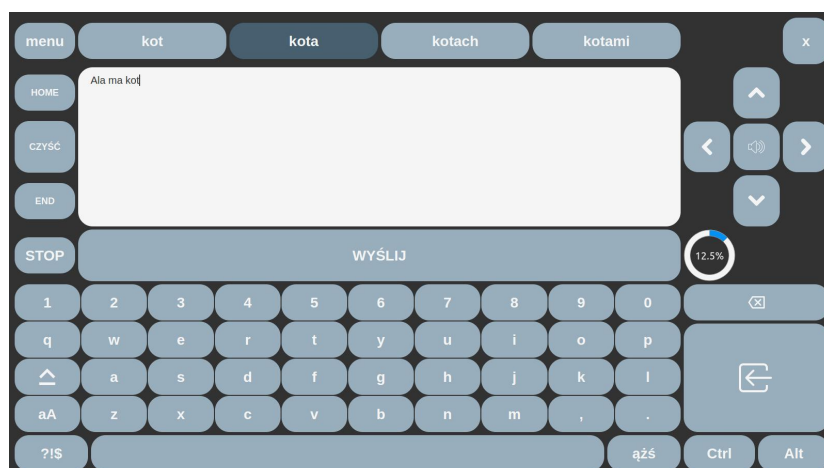
Pierwszym widokiem prezentowanym użytkownikowi jest widok z zablokowaną klawiaturą przedstawiony na rysunku 3.3. Zadbano o to, by przyciski w trybie "wstrzymania" przedstawione były w innym kolorze od przycisków aktywnych. Po użyciu przycisku *START* klawiatura zmienia swój wygląd na jednolity, taki jak na rysunku 3.4. Dalsze dwie grafiki 3.5 oraz 3.6 prezentują wpisywanie tekstu oraz wyświetlanie się możliwych opcji autouzupełnienia tekstu wpisywanego.



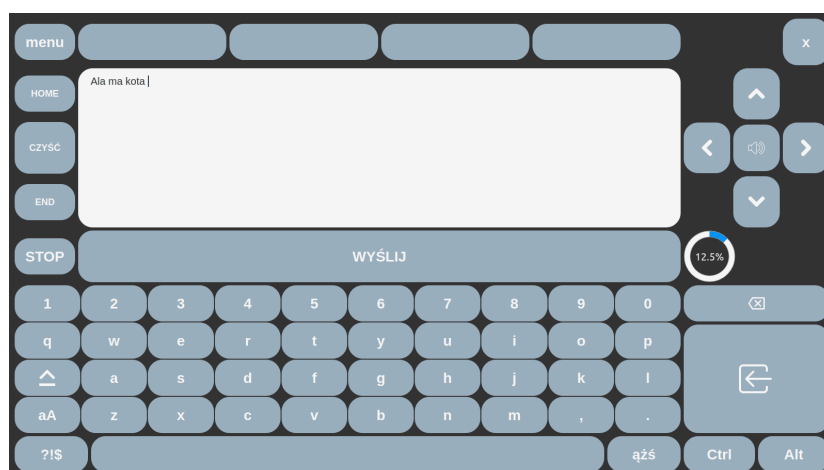
Rys. 3.3 Początkowy wygląd aplikacji z nieaktywnymi przyciskami.



Rys. 3.4 Aktywny wygląd aplikacji.



Rys. 3.5 Przykład wyświetlanych opcji autouzupełniania.



Rys. 3.6 Tekst i podpowiedzi po wybraniu sugerowanego tekstu.

3.6 Projekt testów

Podczas testowania aplikacji szczególną uwagę należy zwrócić na czas, jaki zajmuje użytkownikowi wprowadzenie tekstu do edytora za pomocą klawiatury ekranowej, przydatność udogodnień typu autouzupełnianie słów oraz subiektywne odczucia osób badanych. W związku z tym zaplanowano następujący przebieg testów:

1. Pomiar czasu wprowadzania tekstu numer 1 do edytora tekstu za pomocą klawiatury komputerowej.
2. Pomiar czasu wprowadzania tekstu numer 2 do edytora tekstu za pomocą klawiatury ekranowej.
3. Pomiar czasu wprowadzania tekstu numer 1 do edytora tekstu za pomocą klawiatury ekranowej.
4. Pomiar czasu wprowadzania tekstu numer 2 do edytora tekstu za pomocą klawiatury komputerowej.
5. Pomiar czasu wprowadzania dowolnego tekstu o określonej tematyce do edytora tekstu za pomocą klawiatury ekranowej.

W celu poprawnego przebiegu testów należy przyjąć pewne założenia:

- osoby testowane nie będą miały możliwości wcześniejszego zapoznania się z treścią wpisywanych tekstów, ani nie będą znały tematu ostatniego z punktów testu,
- osoby testowane piszą obcym sobie komputerze, by nie znały układu klawiszy komputerowych,
- testujący do wprowadzania danych na klawiaturze ekranowej wykorzystywać będą myszy komputerowej (odrzucaamy czynnik braku umiejętności pracy z nowym sprzętem),
- każda z osób testowanych, przejdzie krótkie szkolenie z zakresu działania klawiatury ekranowej,
- w czasie pomiaru czasu zapisywana będzie również ilość wykorzystywanych podpowiedzi tekstu w celu określenia ich skuteczności,
- pod koniec testów z każdym z badanych przeprowadzona będzie rozmowa na temat jego subiektywnych wrażeń podczas pracy z klawiaturą.

3.7 Możliwość dalszego rozwoju

Przewidziano możliwość dalszego rozwoju aplikacji wprowadzając do prototypu przyciski, których działania nie obsłużono. Są nimi *Ctrl*, *Alt* oraz *Tekst to Speech*. Dwa pierwsze przewidziane zostały w celu współpracy z programami zewnętrznymi oraz wprowadzenia skrótów klawiszowych (np. *Ctrl+A*, *Ctrl+V*,

Ctrl+C, Ctrl+X). Należy pamiętać, że użytkownikami klawiatury są często osoby starsze, które przed zachorowaniem miały już styczność z pracą na komputerze. Wprowadzenie znanych im skrótów, czy funkcjonalności może usprawnić ich pracę z komputerem. Bardzo ważnym aspektem oferowanym przez innych producentów oprogramowania dla osób spracizowanych, bez możliwości komunikacji werbalnej jest TextToSpeech, czyli synteza mowy na podstawie tekstu. Tak zsyntetyzowana ścieżka dźwiękowa z wprowadzoną wypowiedzią mogłaby zostać odtworzona poprzez głośniki urządzenia i umożliwić werbalną komunikację z innymi osobami.

3.8 Wymagania pozafunkcjonalne

Podstawowymi wymaganiami pozafunkcjonalnymi są:

- komputer lub urządzenie mobilne z systemem operacyjnym Linux, bądź Windows,
- mysz komputerowa lub specjalne okulary do eye-trackingu,
- dwa monitory do pracy z internetem,
- 4GB pamięci RAM,
- min. 3,5 GB wolnej pamięci,
- monitor o rozdzielczości nie mniejszej niż 14 cal.

3.8.1 Ograniczenia

Ze względu na ograniczoną powierzchnię ekranową nie udało się zrealizować przycisków o minimalnych wymiarach 140px x 140px.

3.9 Podsumowanie

Rozdział ten stanowi analizę techniczną zaplanowanego oprogramowania. Tłumaczy jego działanie oraz opisuje przewidziane technologie. Przedstawiono przewidywane wymagania funkcjonalne oraz wynikające z analizy sprzętowej wymagania pozafunkcjonalne. Dokładnie omówiono przypadki użycia aplikacji oraz oferowane funkcjonalności. Poświęcono także podrozdział, by omówić ewentualne możliwości rozwoju aplikacji w celu jej udoskonalenia, które nie są celem tego projektu.

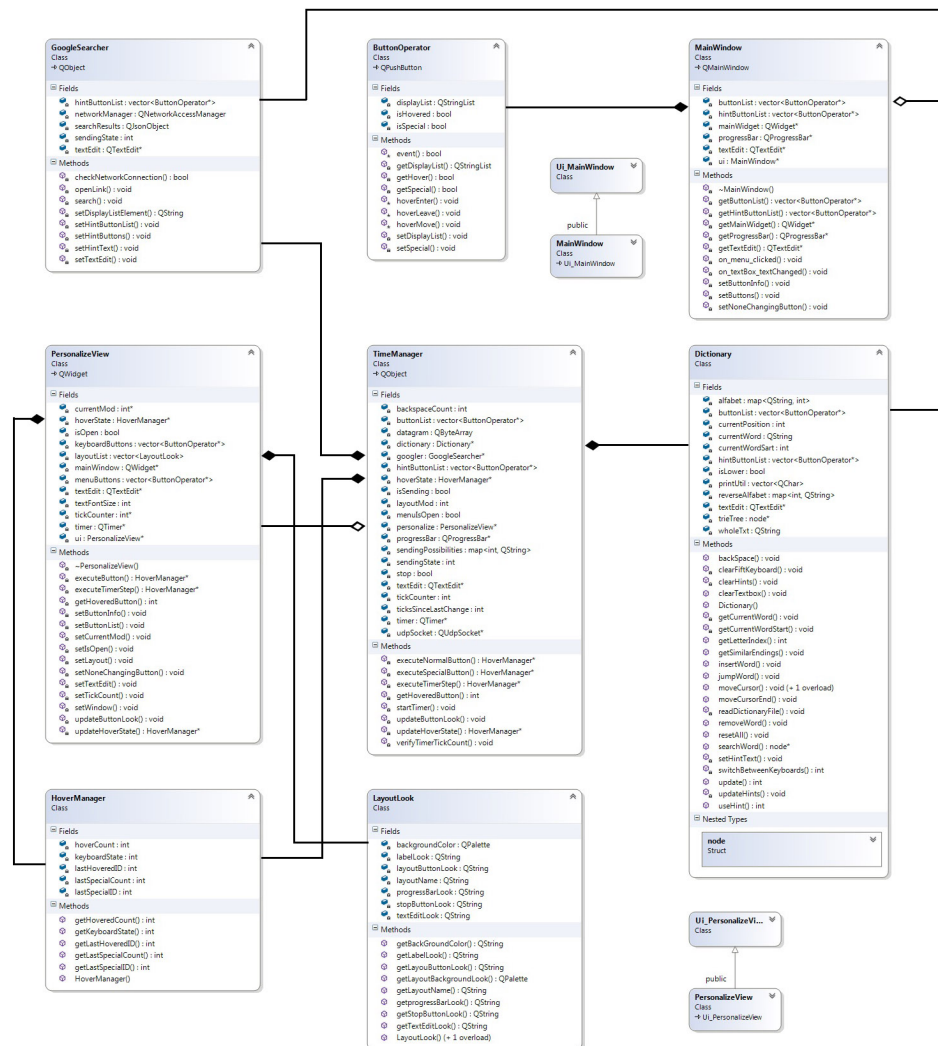
Rozdział 4

Implementacja

W aplikacji, wykorzystującej wyżej wymienione technologie, stworzono 8 klas w języku C++ wraz z odpowiadającym im plikiem nagłówkowym, dodatkowy plik zawierający stałe programowe oraz 2 widoki.

4.1 Diagram klas

Na rysunku 4.1 przedstawiono poglądowy diagram klas wraz z ich zależnościami. Opis klas oraz zastosowanych algorytmów zawarto w podrozdziałach 4.2 oraz 4.3.



Rys. 4.1 Poglądowy diagram klas zaimplementowanych w projekcie.

4.2 Opis klas zastosowanych

Klasami użytymi w programie są: ButtonOperator, Dictionary, GoogleSearcher, HoverManager, LayoutLook, MainWindow, PersonalizeView, TimeManager. Dla każdej z nich można określić główną funkcję. Zestawienie to przedstawiono w tabeli 4.1.

Nazwa klasy	Funkcja w programie
ButtonOperator	Klasa dziedzicząca po QPushButton (wewnętrzna klasa biblioteki QT) odpowiedzialna za działanie przycisków w projekcie.
Dictionary	Klasa zajmująca się pracą ze słownikiem, a także zarządzaniem wprowadzonym tekstem. Do konstruktora przyjmuje wskaźnik na pole tekstowe, listę wskaźników na wszystkie działające przyciski oraz listę wskaźników na przyciski odpowiedzialne za podpowiedzi. Klasa posiada również strukturę "node", która stanowi podstawowy element budulcowy dla słownika. Więcej na ten temat w podrozdziale 4.3.3.
GoogleSearcher	Klasa zajmująca się komunikacją programu z internetem, obsługą zapytania GET oraz przetworzeniem odpowiedzi. Więcej na ten temat w podrozdziale 4.3.5.
HoverManager	Klasa nadzorująca działanie przycisków. Obiekty przechowują informację o: ID przycisku ostatnio fiksovanego, ID ostatnio wybranego specjalnego przycisku, czasie fiksacji nad jednym przyciskiem, czasem działania specjalnego przycisku. Więcej o przyciskach i ich działaniu w podrozdziale 4.3.1.
LayoutLook	Klasa przewidziana z myślą o dynamicznej zmianie wyglądu przez użytkownika w oknie menu. Obiekty klasy zebrane w listę w klasie PersonalizeView przechowują komplet danych dotyczących jednego wyglądu. W skład tego wchodzi nazwa wyglądu, wygląd przycisków zapisany przy pomocy kaskadowego arkusza styli, paletę zmieniającą kolor tła okien, arkusz styli dla paska postępu, podpisów oraz wyglądu przycisków w trybie zablokowanym.
MainWindow	Główna klasa projektu odpowiedzialna za komunikację interfejsu z resztą klas.
PersonalizeView	Klasa odpowiedzialna za obsługę wszystkich zdarzeń związanych z oknem menu tzn. zmianą wyglądu aplikacji, zmianą wielkości czcionki, zmianą czasu progowego fiksacji.
TimeManager	Rozbudowana klasa będąca głównym zarządcą procesów zachodzących w aplikacji dzięki umiejętności pracy z regulatorem czasowym (timer).

Tab. 4.1 Tabela przyporządkowania głównych funkcji klasom projektu.

4.3 Zaimplementowane algorytmy

W poniższych podrozdziałach zaprezentowano zasadę działania zaimplementowanych w oprogramowaniu algorytmów.

4.3.1 Działanie przycisków

W aplikacji każdy przycisk jest obiektem klasy `ButtonOperator`, która dziedziczy po klasie `QPushButton`. Dzięki czemu obiekty przycisków mogą korzystać zarówno z metod klasy nadrzędnej np. `isChecked()`, jak i klasy dziedziczącej. Właśnie dzięki nadpisaniu metod domyślnych klasy `QPushButton` - `hoverEnter()` i `hoverLeave()` stworzono możliwość detekcji, nad którym przyciskiem aktualnie znajduje się punkt fiksacji wzroku. Podczas wywołania obu metod zmieniana jest wartość logiczna zmiennej `isHovered` aktualnie obserwowanego przycisku. Prócz informacji o tym, czy dany przycisk jest aktualnie używany przechowuje się również dane o tym, czy przycisk zalicza się do tzw. "specjalnych", czy też nie, jak i listę dostępnych dla niego tekstów do wyświetlania – wyglądu przycisku dla 6 stanów klawiatury (małe litery, duże litery, znaki specjalne karta pierwsza, znaki specjalne karta druga, polskie litery, menu kontekstowe). Wszystkie te dane wprowadza się podczas uruchamiania klawiatury i wtedy także przypisuje się przyciski kolejno do specjalnej listy. Kolejność jest znacząca w przypadku przycisków "specjalnych", gdyż ich obsługa zależna jest od wartości ich ID zapisanego w pliku ze stałymi. Określenie przycisku działającego odbywa się poprzez sprawdzenie jego ID, którego zmienna `isHovered` jest prawdziwa w klasie `TimeManager`. Do zarządzania stanem klawiatury, ostatecznie wybranym przyciskiem specjalnym oraz ostatnio obserwowanym przyciskiem powstała klasa `HoverManager`. Zbiera ona na bieżąco informację o ID przycisku ostatnio najechanego, o tym przez jaki czas dany przycisk jest już pod punktem fiksacji, o aktualnym stanie klawiatury, o ostatnio wybranym specjalnym przycisku (np. `CapsLock`) oraz przez jaki czas działanie tego przycisku się utrzymuje. Większość z tych danych odświeżana jest co 200ms (stała zdefiniowana w pliku ze stałymi) podczas każdego wykonywania się metody `TimerStep()`. Jej działanie przedstawiono za pomocą pseudokodu 1.

Zadaniem wyżej wspomnianej funkcji `executeTimerStep()` jest sprawdzenie, czy dany przycisk był fiksowany przez odpowiednią ilość czasu (zmienna, której wartość zależy od ilości pomyłek popełnionych przez użytkownika, jest dynamicznie zmieniana podczas korzystania z klawiatury – analizowane to jest w funkcji `verifyTimerTickCount()`). Jeżeli warunek ten został spełniony to dalszy przebieg działań zależy od tego, czy przycisk był specjalny, czy też nie oraz czy klawiatura nie znajduje się w trybie wstrzymania.

Działanie przycisków specjalnych

W tabeli 4.2 wymieniono wszystkie specjalne przyciski z widoku klawiatury oraz opisano po krótku algorytm ich działania. W tabeli 4.3 przedstawiono przyciski widoku menu oraz ich zachowanie.

Jako wciśnięcie rozumie się tu czas fiksacji nad przyciskiem przekraczający progową wartość. Stany klawiatury to odpowiednio 0-małe litery, 1-wielkie

Nazwa przycisku	Działanie
Backspace	Wywołuję metodą <code>backspace()</code> klasy <code>Dictionary</code> , której działanie opisano w podrozdziale 4.3.2.
CapsLock	W przypadku pierwszego wciśnięcia zmienia stan klawiatury z 0 na 1 i utrzymuje ją dopóki nie zostanie wybrany po raz wtóry lub wykonana zostanie czynność innego przycisku zmieniającego stan klawiatury.
Czyść	Korzysta z metody <code>resetAll()</code> klasy <code>Dictionary</code> i powoduje czyszczenie edytora tekstowego oraz z nim związanych zmiennych jak <code>currentPosition</code> , <code>currentWord</code> , <code>currentWordStart</code> , <code>wholeText</code> (więcej na ten temat w rozdziale 4.3.2).
Menu	Otwiera okno klasy <code>Personalize</code> .
Przesuń się o jedno słów w prawo lub w lewo	Wywołuje funkcje <code>jumpWord()</code> klasy <code>Dictionary</code> , która powoduje przesunięcie się kursora na koniec poprzedniego słowa, lub na początek kolejnego. Każdorazowo zmieniana jest wartość zmiennej <code>currentWord</code> i <code>wholeText</code> . 4.3.2
Przesuń się w kierunku początku tekstu (home) lub na jego koniec (end)	Wybranie przycisku powoduje przesunięcie się kursora w jednym z dwóch kierunków oraz zmianę wartości zmiennej <code>wholeText</code> i <code>currentWord</code> (4.3.2).
Przyciski podpowiedzi	Wyświetlane są na nich podpowiedzi, których tworzenie opisane jest w rozdziale 4.3.3. Ich użycie powoduje zastąpienie aktualnie wpisywanej frazy autouzupełnionym słowem pobranym ze słownika.
Shift	W przypadku wciśnięcia zmieniony zostaje stan klawiatury z 0 na 1, a po użyciu przycisku ze znakiem stan klawiatury wraca do stanu 0.
Stop i start	Wprowadza klawiaturę w stan wstrzymania lub w stan pracy. W zależności od wartości zmiennej <code>isStop</code> jest możliwe, lub nie, korzystanie z przycisków ze znakami.
Strzałka w lewo i prawo	Korzystając z metody <code>moveCursor</code> klasy <code>Dictionary</code> przemieszcza kursor o jeden znak w danym kierunku. Może zmienić wartość <code>currentWord</code> (4.3.2) korzystając z metod <code>getCurrentWord()</code> oraz <code>getCurrentWordStart()</code> (2) opisanych w późniejszych rozdziałach.
Wyjdź	Powoduje opuszczenie aplikacji.
Wyszukaj na podanej platformie	Przyciśnięcie przycisku powoduje sprawdzenie połączenia internetowego, a następnie wysłanie wpisanej treści pola tekstowego do funkcji <code>search()</code> klasy <code>GoogleSearcher</code> . 4.3.5
Wyślij	Wysyła dotychczas wpisany tekst na broadcast na port 45454 za pomocą metody <code>writeDatagram()</code> klasy <code>QUdpSocket</code> .
Zmiana trybu wyszukiwania	Strzałki powodują na zmianę trybu wyszukiwania. Do wyboru "Google", "YouTube", "Filmweb". Zmieniają wartość zmiennej <code>sendingState</code> niezbędnej do poprawnego działania funkcji <code>search()</code> klasy <code>GoogleSearcher</code> . 4.3.5
Znaki specjalne	Zmienia stan klawiatury na odpowiednio 2 i 3 przy kolejnych kliknięciach.

Tab. 4.2 Lista specjalnych przycisków wraz z ich działaniem.

Algorithm 1 Działanie funkcji TimerStep()

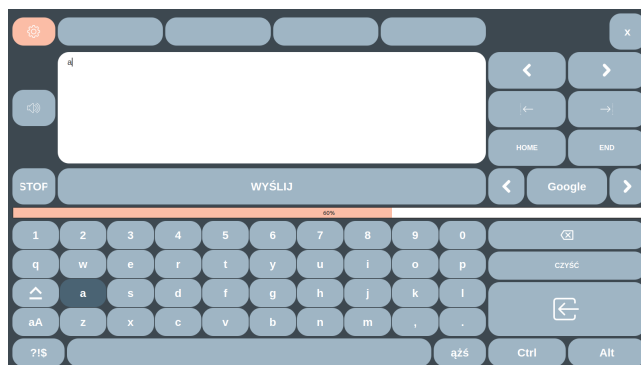
```

pobierz ID aktualnie fiksowanego przycisku
if pobrane ID różne jest od -1 then
  if zwykle przyciski są w trybie wstrzymania then
    if jeśli aktualnie fiksowany przycisk jest specjalny then
      hoverState (obiekt klasy HoverManager) ustaw aktualnie aktywny
      przycisk na pobrane ID
      Wykonaj krok timera (funkcja executeTimerStep())
      Pokaż czas przez jaki przycisk jest fiksowany na pasku postępu dla
      informacji użytkownika.
    end if
  else
    Wykonaj powyższe funkcje dla wszystkich przycisków - niezależnie od
    tego, czy są specjalne, czy nie.
  end if
end if
Wywołaj funkcję verifyTimerTickCount().

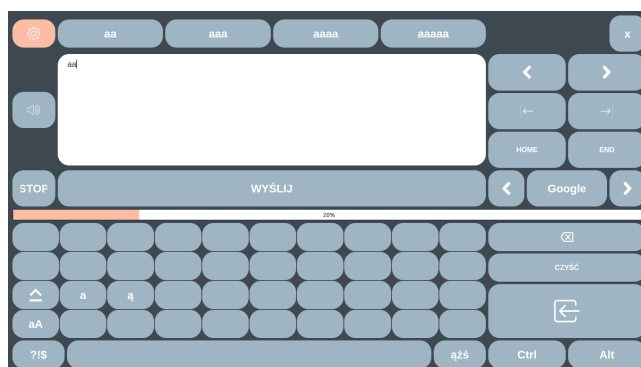
```

Nazwa przycisku	Działanie
Strzałki zmieniające aktualną wartość progową czasu fiksacji	Strzałki powodują zmniejszenie lub zwiększenie wartości zmiennej tickCounter przechowującej ilość 200ms interwałów, po upływie których (w przypadku braku zmiany fiksowanego punktu) przycisk można uznać za wciśnięty. Początkowo czas ten wynosi 3s, jednak w wyniku dynamicznej korekty następuje zmniejszenie lub zwiększenie liczby tych interwałów. Korekta uruchamia się, gdy użytkownik przekroczy dopuszczalną ilość błędów w ciągu minuty. Więcej na temat dynamicznej zmiany czasu w podrozdziale 4.3.4.
Strzałki zmieniające wielkość czcionki w edytorze tekstu	Strzałki powodują zamianę właściwości edytora tekstu.
Strzałki zmieniające wygląd aplikacji	Użytkownik może wybrać między jednym z 5 wersji kolorystycznych poprzez dynamiczną zmianę wyglądu na podstawie danych z listy i aktualnego indeksu.
Wyjście	Powoduje zamknięcie okna menu.

Tab. 4.3 Lista specjalnych przycisków z menu w raz z ich działaniem.



Rys. 4.2 Widok aplikacji z klawiatura w stanie zerowym po wpisaniu litery 'a'.



Rys. 4.3 Widok aplikacji z klawiaturą w stanie piątym po wpisaniu drugiej litery 'a'.

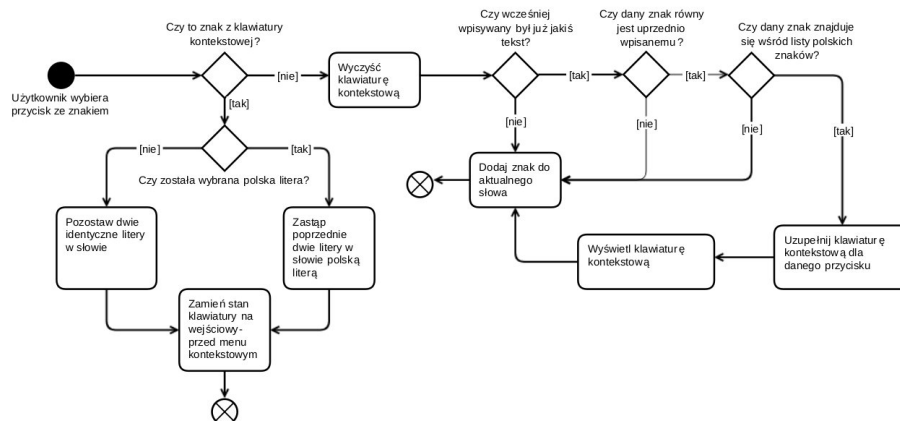
litery, 2-znaki specjalne strona pierwsza, 3-znaki specjalne strona druga, 4-polskie znaki, 5-menu kontekstowe dla polskich znaków.

Działanie przycisków nominalnych

Za wpisywanie znaków z klawiatury do edytora odpowiedzialna jest funkcja - `update()` klasy `Dictionary`. Jest ona głównym zarządcą jeśli chodzi o wprowadzanie znaków w odpowiedniej pozycji. W pierwszej kolejności sprawdzane jest, czy stan klawiatury różny jest od stanu piątego (menu kontekstowe). Gdy spełniony jest dany warunek to porównywana jest aktualnie wprowadzana wartość z uprzednio wprowadzoną. Wyjątek stanowi pierwsze wprowadzenie znaku do edytora. W przypadku dwóch identycznych liter czyszczona jest zawartość piątego stanu klawiatury i przechodzi się do wywołania funkcji `switchBetweenKeyboards()`, która zwraca informację o tym, czy stan klawiatury się zmienił. Tam sprawdzane jest, czy wprowadzony znak jest jedną z liter posiadających polskie odpowiedniki tj. *a-q, c-ć, e-ę, l-ł, n-ń, o-ó, s-ś, z-ż-ź*. Gdy znajdzie się on wśród wymienionej listy, to dla odpowiadającego przycisku ustawiany jest test dla klawiatury stanu piątego i następuje jej wyświetlenie. Działanie takiego zachowania widać na rysunkach 4.2, 4.3.

W innym wypadku aktualna litera dopisywana jest do aktualnie tworzonego

słowa (`currentWord`), kursor przesuwany jest o jedną pozycję w prawo. Gdy dodana jest spacja aktualnie przetwarzane słowo jest uznawane za skończone i zapamiętywany jest nowy początek kolejnego słowa. Odświeżony zostaje również widok podpowiedzi, których powstawanie omówiono w późniejszym rozdziale 4.3.3. W sytuacji, gdy wybrana zostaje litera z menu kontekstowego (klawiatury w stanie piątym), to albo wpisany tekst zostaje podmieniony na polską literę, lub nie ulega zmianie. Przykładowo po wpisaniu "aa" użytkownik po raz kolejny wybiera literę "a" - tzn. planował wpisanie frazy "aa". Jeśli jednak decyduje się na literę "ą", to w miejsce widniejącego napisu "aa" pojawia się "ą". Poglądowy diagram przepływu przedstawiono na rysunku 4.4.



Rys. 4.4 Widok aplikacji z klawiaturą w stanie piątym po wpisaniu drugiej litery 'a'.

4.3.2 Praca z tekstem

Wpisywanie tekstu w aplikacji odbywa się poprzez specjalny algorytm kontrolujący zawartość aktualnie pisanego słowa oraz całego tekstu. W czasie działania programu, w pamięci przechowywany jest `currentWord`, czyli aktualne słowo tj. ciąg znaków, liter lub cyfr, które zaczynają się na początku wpisywanego tekstu lub po spacji, a kończą się w pozycji kursora. Dodanie spacji po ciągu znaków kończy słowo i usuwa je ze zmiennej `currentWord`, a dodaje do zmiennej zwanej `wholeText`, która przechowuje dotychczas wpisany tekst. Przykładowo jeśli mamy tekst jak na rysunku 4.5 to w zmiennej `wholeText` przechowujemy "Wymagajcie od siebie choćby inni od was nie wymagali." Jan Paweł II", a w `currentWord` "sie". W ten sposób podpowiedzi generowane będą jedynie dla części "sie", a tekst wpisywany będzie w pozycji kursora, która również monitorowana jest przez zmienną `currentPosition`. Scalenie `wholeText` i `currentWord` następuje, gdy zmieniamy pozycję kursora strzałkami, lub jeśli do tekstu dodana zostanie spacja, a za kursorem nie znajduje się żaden znak. Przykład przedstawia rysunek 4.6.

Zmienna `currentWord` została zespolona z `wholeText` poprzez wklejenie jej na

```
„Wymagajcie od siebie choćby inni od was nie wymagali.”
~Jan Paweł II
```

Rys. 4.5 Przykładowe zapisywanie tekstu do zmiennych w zależności od pozycji kursora.

```
„Wymagajcie od siebie |choćby inni od was nie wymagali.”
~Jan Paweł II
```

Rys. 4.6 Przykładowe zapisywanie tekstu do zmiennych w zależności od pozycji kursora.

pozycji zapisanej jako `currentWordStart`. W celu dynamicznego ustalania pozycji `currentWordStart` oraz zawartości `currentWord` powstały funkcje: `getCurrentWordStart()` oraz `getCurrentWord()`. Działanie pierwszej zademonstrowano za pomocą pseudokodu 2.

Algorithm 2 Działanie funkcji `getCurrentWordStart()`

```
if currentWord nie jest pusty && currentPosition nie jest na początku tekstu
then
  for każda pozycja aż do początku tekstu do
    pobierz literę z wholeText w danej pozycji
    if pobrana wartość nie jest liczbą ani literą then
      currentWordStart = currentPosition + 1
    end if
  end for
end if
```

Działanie drugiej sprowadza się do wycięcia fragmentu tekstu między `currentWordStart`, zaimplementowanym w wyniku działania poprzedniej funkcji, a `currentPosition`.

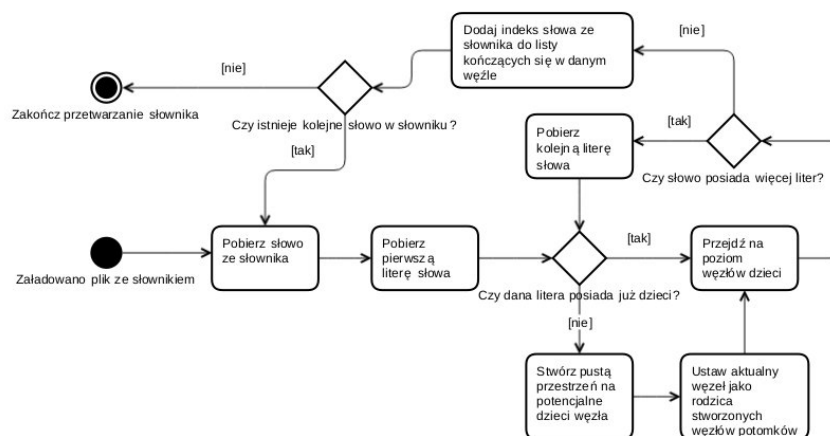
Usuwanie znaków

Proces usuwania znaków staje się trudniejszy ze względu na ułatwiające kontrolę nad tekstem `currentWord` i `wholeText`. Pierwszym napotkanym problemem była sytuacja, w której `currentWord` jest puste. W takim wypadku należy usunąć znak znajdujący się w `currentPosition` ze zmiennej `wholeText`, a następnie za pomocą wcześniej opisanych funkcji `getCurrentWordStart()` oraz `getCurrentWord()` otrzymać informację o nowej wartości `currentWord`. Kolejnym krokiem jest wycięcie wartości zmiennej `currentWord` z `wholeText`, przesunięcie kursora dla informacji użytkownika w odpowiednią pozycję oraz odświeżenie wyglądu odpowiedzi. Gdy `currentWord` ma wartość, sytuacja upraszcza się do usunięcia ostatniego znaku z `currentWord`. W celu reprezentacji tekstu dla użytkownika należy scalić `wholeText` z `currentWord`, ale by nie utracić wartości `wholeText` stworzono tymczasową zmienną `currentText`, który zawiera całą treść wpisanego

tekstu.

4.3.3 Trie tree

Jak wyżej 3.2.1 wspomniano, w pracy wykorzystano drzewo typu Trie w celu pracy z rozległym słownikiem. Słowa z pliku w formacie TXT wczytywane są do programu podczas uruchamiania klawiatury- każda z linii dostarczanego pliku powinna stanowić pojedynczy wyraz. Słowa te, za pomocą sztucznie wygenerowanych list kodujących oraz dekodujących polski alfabet, wprowadzane są do drzewa typu Trie. Drzewo takie powstaje poprzez stworzenie pustego węzła typu node - struktury zadeklarowanej w pliku nagłówkowym klasy obsługującej współpracę ze słownikiem – Dictionary. Struktura node przechowuje informacje o rodzicu bieżącego węzła, o jego potomkach – czyli węzłach następujących oraz wektor zawierający informację o przynależności danego węzła literowego do słowa. Po zainicjowaniu węzła zerowego, po kolei analizowane są słowa ze słownika w funkcji insertWord(). Każde jest rozpatrywane jako tablica liter (char) i iteracyjnie następuje najpierw kodowanie litery na odpowiadający jej indeks (za pomocą mapy alfabet), potem, sprawdzane jest, czy w drzewie nie istnieje już węzeł odpowiadający danej wartości. Gdy nie znaleziono gałęzi drzewa pasujących do poszukiwanej wartości tworzy się za pomocą funkcji calloc przestrzeń w pamięci na przyszłe dzieci tego węzła, a jako ich rodzica podaje się aktualnie przeglądany węzeł z literą. Kolejno, niezależnie od wyniku uprzednio rozpatrywanego warunku, o ile słowo wciąż posiada litery do przeglądu, przenosi się o poziom niżej w drzewie (na poziom dziecka uprzednio rozpatrywanej litery) i proces zachodzi od początku. Gdy przetworzono wszystkie litery słowa do listy wystąpień ostatniego odwiedzanego węzła dopisany zostaje indeks słowa w słowniku – tym samym oznaczając jego koniec. Na rysunku 4.7 przedstawiono, w sposób graficzny, działanie danej funkcji.

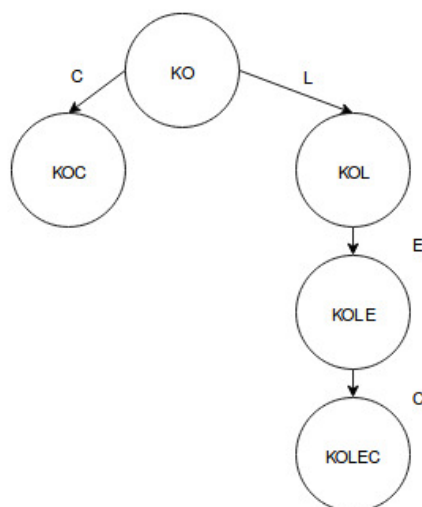


Rys. 4.7 Diagram przepływu dla funkcji insertWord() wprowadzającej słowa do drzewa typu Trie.

Po uzupełnieniu słownika nastąpić może autouzupełnianie wpisywanego tek-

stu. Każde wpisanie litery powoduje wywołanie metody `updateHints()`, która jest odpowiedzialna za tworzenie oraz wyświetlanie podpowiedzi, zgodnie z wprowadzonym do tej pory słowem. Jako poszukiwaną frazę traktujemy ciąg znaków, które użytkownik wpisał do pozycji kursora od ostatniej spacji, bądź początku tekstu. W wypadku, gdy ten ciąg znaków jest dłuższy niż dwa, wykorzystywana jest funkcja komunikująca się ze stworzonym słownikiem – `searchWord()`. Przekazywane jest drzewo Trie słownika oraz aktualnie poszukiwana fraza (bez formatowania). Wpisywana fraza, również traktowana jest jako zbiór liter, które przeglądane są jedna po drugiej. Dla każdej następuje zmiana dzięki mapie kodującej alfabet na odpowiadający indeks, który umożliwia przeszukiwanie słownika. Sprawdzane jest, czy istnieje potomek drzewa Trie o danym indeksie – jeśli tak, to następuje zmiana węzła na węzeł dziecka, tak, że przy przeszukiwaniu drzewa będą brane pod uwagę jedynie węzły z rodzicem będącym pierwszą literą poszukiwanej frazy. Gdy przeszuka się wszystkie litery, bądź w trakcie tego procesu zabraknie węzłów potomków dla danej kombinacji liter to zwracany jest odpowiednio ostatni węzeł wspólny dla danej frazy lub też węzeł pusty.

W celu lepszego zrozumienia działania algorytmu rozpatrzmy go na przykładzie. Załóżmy, że mamy drzewo takie jak na rysunku 3.1. Jeśli wyszukamy frazę "ko" to funkcja zwróci nam węzeł i jego dzieci. Możliwe autouzupełnienia wyglądają wtedy tak jak na rysunku 4.8. Złożenie końcówek wyrazów zachodzi w

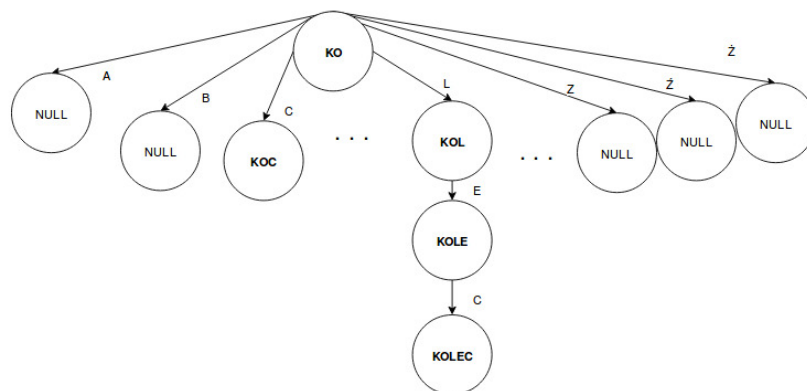


Rys. 4.8 Przykładowy węzeł do autouzupełniania słów po wpisaniu frazy "ko".

funkcji `getSimilarEndings()`, której przekazywany jest znaleziony węzeł, pusty wektor, do którego mają zostać zapisane wynikowe wyrazy oraz węzeł znaków niezbędny do dekodowania. W pierwszym kroku sprawdzane jest, czy w danym węźle kończą się jakieś słowa, jeśli tak (wektor wystąpień węzła jest różny od zera),

to każdą literę zapisaną w wyżej wymienionym wektorze znaków zapisuje się do jednego słowa (tworząc końcówkę do autouzupełniania). Gotowy ciąg znaków zapisywany jest do wektora z końcówkami. W wypadku pierwszego wyko-

niania się funkcji mamy do czynienia z pustym wektorem znaków- toteż nie zostanie stworzona żadna końcówka. Nawet jeśli "ko" było pełną formą słowa nie powinna się ona wyświetlać w proponowanych opcjach użytkownika. Aby uzupełnić wektor znaków należy przejrzeć przesłany węzeł (ten z rysunku 4.8) – w tym celu sprawdza się, czy dzieci węzła odpowiadającej każdej literze alfabetu nie są puste, gdyż programowe drzewo ma, prócz wcześniej przedstawionych gałęzi, jeszcze 33 (zakładając, że zaimplementowany alfabet posiada 35 liter) nieobdarzone wartością gałęzie przedstawione na rysunku 4.9. Gdy znaleziono element o niezerowej wartości pobierana jest za pomocą mapy symetrycznej (reverseAlfabet) wartość literowa węzła i wprowadzana jest do wektora znaków. Węzeł z "ko" zmieniany jest w węzeł pierwszego pierwszego dziecka – w tym wypadku "koc". Następnie przez rekurencję ponownie rozpoczyna się sprawdzanie, czy dane słowo kończy się w tym węźle. Jeśli tak, to proces zachodzi według powyżej opisanych kroków, jeśli nie, to znów poszukiwany jest węzeł potomny z kolejną literą końcówki słowa do autouzupełniania. Algorytm przedstawiono w sposób graficzny na rysunku 4.10

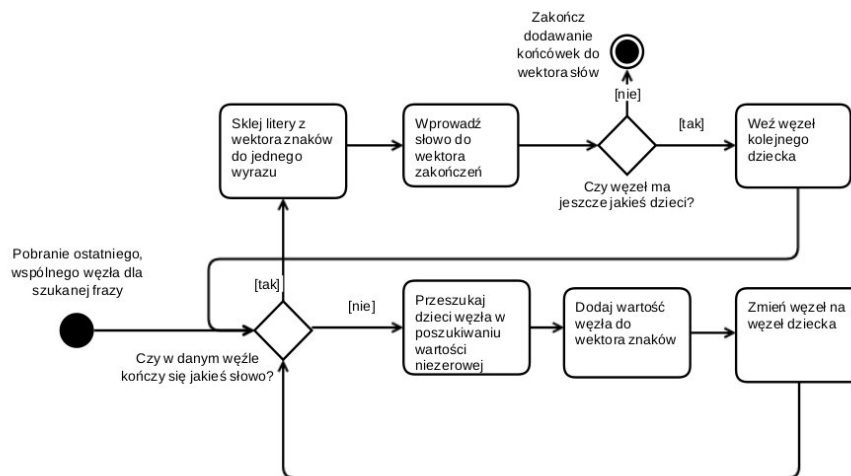


Rys. 4.9 Reprezentacja przykładowego węzła widzianego w programie.

Ostatnim krokiem w stworzeniu podpowiedzi jest skrócenie listy końcówek do liczby przycisków przeznaczonych na podpowiedzi oraz zespolenie ich z dotychczas wpisanym słowem. Taki ciąg znaków można przedstawić użytkownikowi jako napis na przycisku, który po wybraniu wpisuje reprezentowany tekst do pola tekstowego zamieniając dotychczasową frazę na wybraną oraz dodając znak spacji na końcu nowowybranego słowa.

4.3.4 Dynamiczna zmiana czasu progowego fiksacji

Jak wspomniano wcześniej klawiatura umożliwia dynamiczną zmianę czasu czasu progowego fiksacji (decydującego o tym kiedy dany przycisk zostanie wywołany). Zmiana następuje w oparciu o ilość błędów wykonanych przez użytkownika w określonym oknie czasowym. Weryfikacja następuje co minutę. Jeśli w tym czasie użytkownik popełni 5 lub więcej błędów (użyje przycisku Backspace), to czas fiksacji ulegnie wydłużeniu o jedną sekundę. Pięć błędów stanowi, dla ustawień początkowych, czyli progowego czasu fiksacji ustawionego na 3s, 25% znaków wpisanych w tym czasie. Dla ilości błędów mniejszej niż dwa czas fiksacji zostaje



Rys. 4.10 Diagram przepływu dla funkcji `getSmliarEndings()`.

skrócony. Jeśli użytkownik użyje przycisku Backspace 3-4 razy w ciągu minut czas progowy fiksacji nie ulegnie zmianie. Czas fiksacji został ograniczony obustronnie poprzez stałe z pliku `const.h`. Założono, że nie może on być krótszy niż 1s i dłuższy niż 6s. Użytkownik ma również możliwość manualnego ustawienia czasu progowego, poprzez zmianę wartości w oknie menu.

4.3.5 Korzystanie z wyszukiwarki internetowej

W celu pracy z przeglądarką niezbędne jest podłączenie drugiego ekranu, na którym może się otwierać okno przeglądarki. W innym wypadku okno klawiatury ulegnie minimalizacji i nie ma możliwości powrotu do okna.

W przypadku skorzystania z jednego z trybów wysłania ("Google", "YouTube", "Filmweb") wywoływana jest metoda, `GoogleSearcher, search()`. W zależności od przesłanej wartości `sendingState`, informującego o tym, gdzie wysłane ma być zapytanie, do wyszukiwanego tekstu (pobranego z pola tekstowego) dołączana jest informacja, w którym serwisie szukać wyników. Taka informacja załączana jest jako argument uzupełniający URL powstały przez stworzenie spersonalizowanej wyszukiwarki, dzięki specjalnemu Google API. Następnie obiekt klasy `QNetworkAccessManager` wywołuje metodę RESTową GET na obiekcie klasy `QNetworkRequest`, który za argument konstruktora przyjmuje powstałe URL z parametrem.

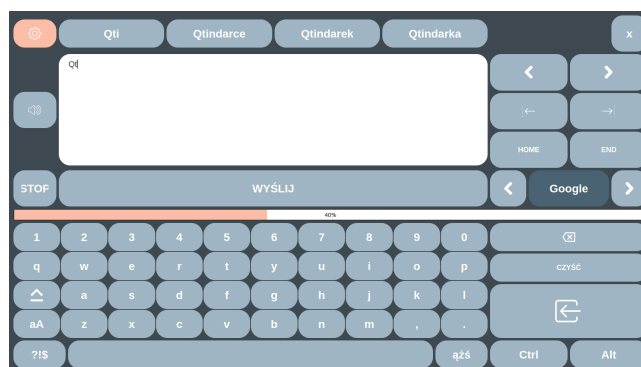
Google Api

Jak już wcześniej wspomniano 3.2.2 do projektu wykorzystano specjalne API Google - Google Custom Search Engine. Dzięki temu powstał specjalny link URL umożliwiający przyjmowanie różnych parametrów jako zapytanie do wyszukiwarki. Co więcej specjalne API na rządanie GET zwraca informację w postaci `QNetworkReply`, gdzie funkcja `handleNetworkData()` umożliwia jego przetworze-

nie w ustrukturyzowaną postać JSON. Obiekt JSON zawiera 10 pierwszych wyników wyszukiwania w specjalnej przeglądarce. Każdy z obiektów typu JSON posiada informacje o wyniku wyszukiwania. W skład takiego obiektu wchodzi [19]:

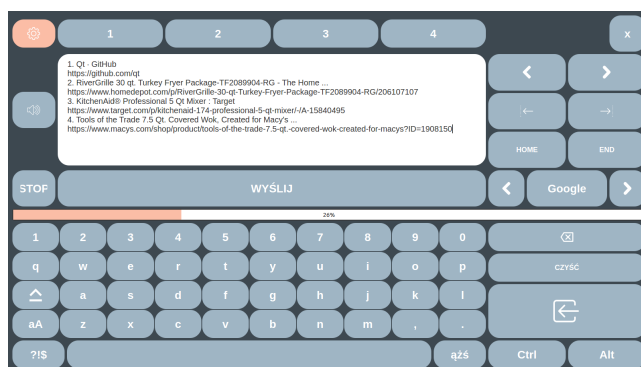
- "kind": "customsearch#result"
- "title": string
- "htmlTitle": string
- "link": string
- "displayLink": string
- "snippet": string
- "htmlSnippet": string
- "cacheId": string
- "mime": string,
- "fileFormat": string
- "formattedUrl": string
- "htmlFormattedUrl": string
- "pagemap"

Wykorzystując dane z "title" oraz "link" przedstawiane są użytkownikowi cztery pierwsze wyniki wyszukiwania w polu tekstowym, a ich wywołanie (otworzenie strony) odbywa się przez wybranie przystosowanych przycisków podpowiedzi z numerem wyniku wyszukiwania w przypadku wyszukiwania danych w Google. Przykład działania przedstawiono na rysunkach 4.11 oraz 4.12.



Rys. 4.11 Przykładowy tekst wpisany do pola tekstowego, wysyłany jako argument wyszukiwania Google.

W wypadku wyszukiwań w Youtube oraz Filmweb użytkownik nie ma możliwości wyboru wyniku wyszukiwania - otwierany jest pierwszy wynik, który zawiera odpowiednie słowa kluczowe.



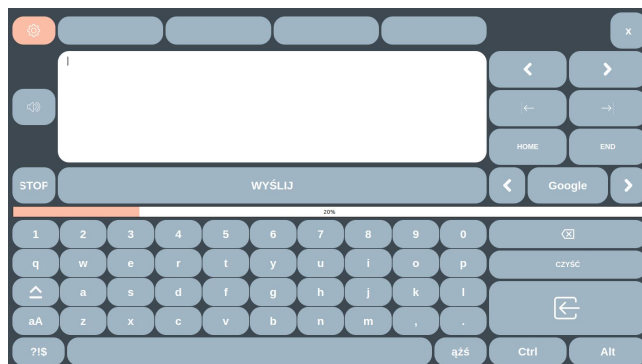
Rys. 4.12 Wyniki wyszukiwania tekstu wpisanego na rysunku 4.11.

4.4 Zaimplementowany interfejs użytkownika

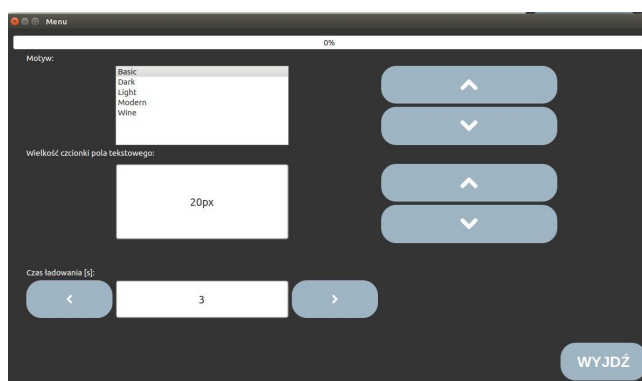
Zaimplementowany interfejs użytkownika uległ przemodelowaniu w porównaniu z przedstawionym wcześniej projektem. 3.5.1 Wygląd nowego interfejsu można zaobserwować na rysunkach 4.13, 4.14, 4.15. Analizując widok klawiatury od górnego lewego rogu, pierwszą zmianą, którą da się zauważyć to zmiana napisu "menu" na intuicyjną ikonę, która w sposób bardziej spójny pasuje do całościowego wyglądu. Wszystkie ikony projektu pobrane zostały ze strony [20] i udostępniane są na podstawie licencji CC 3.0. Autorami są Smashicons, D. Grandy, C. Fertu oraz G. Cresnar. Kolejną zmianą jest rozmieszczenie przycisków "home", "end" oraz "czyść". Zostały posortowane tematycznie, tak by użytkownikowi było łatwiej je odszukać. W ich miejsce przedstawiony został przycisk TextToSpeech (aktualnie nieobsługiwany). W miejsce strzałek służących do poruszania się w kierunku góra-dół po tekście stworzono przyciski, których zadaniem jest przemieszczanie się po tekście o jedno słowo. Podczas testów zauważono, że przy niedługich fragmentach tekstu, do których prawdopodobnie najczęściej wykorzystywana będzie klawiatura, są znacząco bardziej wykorzystywane niż strzałki góra-dół. Dużym ułatwieniem dla użytkownika była również zmiana dotychczas niewielkiego, kolistego paska postępu na pasek o szerokości aplikacji. Łatwiej w ten sposób obserwować zmieniający się stan bez



Rys. 4.13 Widok interfejsu, gdy przyciski są w trybie wstrzymania.



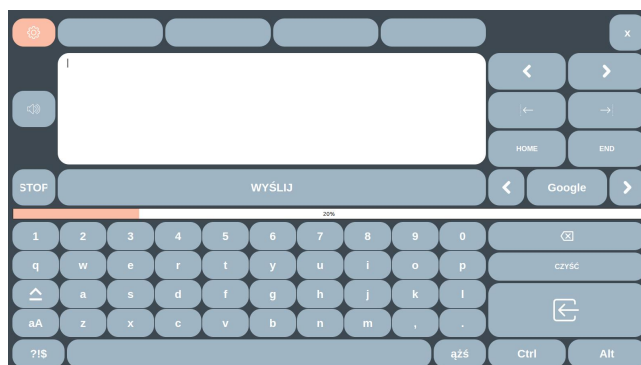
Rys. 4.14 Widok interfejsu, gdy przyciski są w trybie aktywnym.



Rys. 4.15 Widok okna menu.

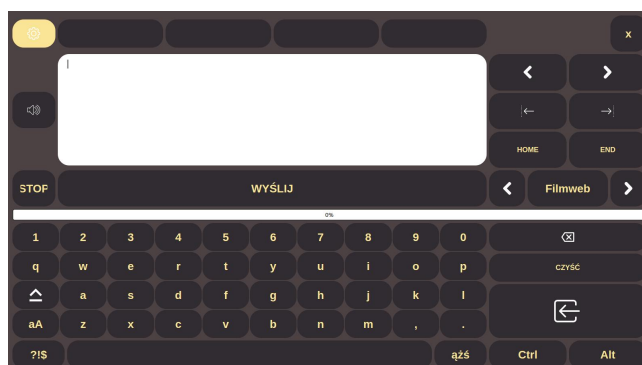
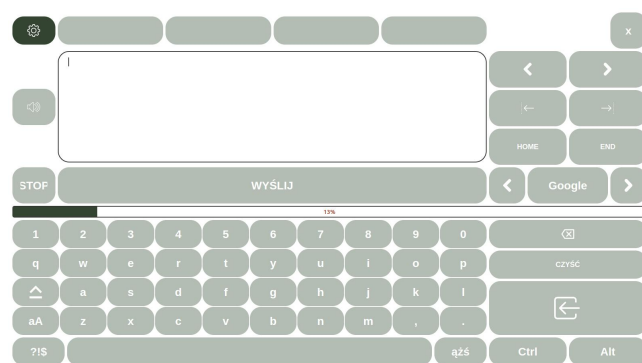


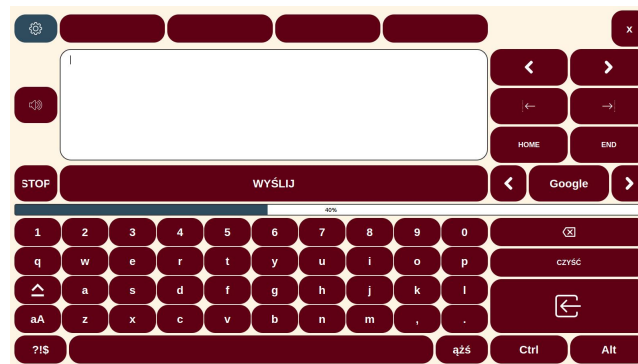
Rys. 4.16 Możliwe widoki trybów wyszukiwania.

Rys. 4.17 Widok interfejsu w trybie *Basic*.

odrywania wzroku od fikowanego przycisku. Pasek dodatkowo jest w kolorze kontrastującym. Obok przycisku "wyślij", który celowo został zaprojektowany jako największy w klawiaturze (by był łatwy do trafienia i szybki w dostępie podczas konwersacji przez broadcast), pojawiły się przyciski do poruszania się po trybach wyszukiwania oraz przycisk zmieniający swój napis adekwatnie do wybranego trybu. Zmiana wygląda tak jak na rysunku 4.16.

Jak widać na rysunku 4.15 użytkownikowi do wyboru przedstawiono 5 wersji kolorystycznych interfejsu 4.17, 4.18, 4.19, 4.20, 4.21. Kolory dobrano tak, by stanowiły przyjemną do oka, spójną wizję aplikacji, a jednocześnie dawały wystarczający kontrast do wydajnej pracy. Paletę *Dark* zaprojektowano z myślą o pracy przy słabym oświetleniu, tak by jasne kolory ekranu nie raziły użytkownika w oczy. Paleta *Light* przystosowana jest dla użytkowników o potrzebie mniejszego kontrastu. Zarówno kolor zielony jak i niebieski są uważane według autorów publikacji [21] za barwy o charakterze relaksującym użytkownika. Palety *Modern* oraz *Wine* powstały w celu zwiększenia możliwości wyboru użytkownika, gdyż ostateczna ocena wyglądu interfejsu zależy od subiektywnych odczuć osoby korzystającej. Wszystkie wyżej wymienione palety powstały w oparciu o palety stworzone na stronie [22].

Rys. 4.18 Widok interfejsu w trybie *Dark*.Rys. 4.19 Widok interfejsu w trybie *Light*.Rys. 4.20 Widok interfejsu w trybie *Modern*.

Rys. 4.21 Widok interfejsu w trybie *Wine*.

4.5 Podsumowanie

W ramach realizacji projektu udało się zaimplementować wszystkie zaplanowane w rozdziale 3 funkcjonalności. Niektóre zachowania zostały zmienione (np. wybór trybu wysyłania/wyszukiwania) względem zaplanowanych na diagramie przypadków użycia 3.4. Wynikło to z analizy sposobu korzystania z przycisków klawiatury i uznano nowy rozkład za bardziej ergonomiczny. Działanie oprogramowania omówiono w powyższym rozdziale ze szczegółowym opisem zastosowanych algorytmów bazowych, dzięki którym możliwe jest poprawne działanie aplikacji. Wymieniono i wytłumaczono rolę najważniejszych zmiennych. Omówiono nowy wygląd interfejsu użytkownika oraz powody, dla których wprowadzono różnice w porównaniu z prototypem.

Rozdział 5

Testy i wyniki

W poniższym rozdziale przedstawiono opis testów dokonanych w celu weryfikacji jakości wykonanej aplikacji, wyniki oraz wnioski z nich wynikające.

Rozdział 6

Podsumowanie

Bibliografia

- [1] Taub E. Mumenthaler M., Mattle H. *Fundamentals of Neurology: An Illustrated Guide*. Georg Thieme Verlag, 2006.
- [2] Krechowicki A., Kubik W., Sokołowska-Pituchowa J. Łasiński W., Narkiewicz O., Sylwanowicz W., and Szostakiewicz-Sawicka H. *Anatomia Człowieka Podręcznik dla Studentów Medycyny*. Państwowy Zakład Wydawnictw Lekarskich, 1983. Wydanie IV poprawione i uzupełnione.
- [3] Leigh P. N. Wijesekera L.C. Amyotrophic lateral sclerosis. Institute of Psychiatry, Kings College London, February 2009.
- [4] Ossowska K. Leczenie choroby parkinsona i stwardnienia zanikowego bocznego.
- [5] Ryszard Podemski. *Kompendium Neurologii Wydanie III*. VIA Medica, 2014.
- [6] Bała P., Karolczyk E., Zychowska E., Jóźwiak B., Janicki R. and Śliwka A., Żuławiński M., and Szkulmowski Z. Poradnik dla chorych na śl/mnd, 2009.
- [7] Tomik B. Adamek D. *Stwardnienie boczne zanikowe*. ZOZ Ośrodek UMEA Shinoda-Kuracejo, 2005.
- [8] Pat M. Beukelman D. *Augmentative and Alternative Communication, Supporting Children and Adults with Complex Communication Needs Fourth Edition*. Paul H. Brookes Publishing Co., 2013.
- [9] J. Wtorek T. Kocejko, A. Bujnowski. Eye mouse for disabled. Conference on Human System Interactions, 2008.
- [10] B. Kunka. System monitorujący stopień koncentracji uwagi uczniów. In *Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 26 XIX Seminarium ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICIE' 2009 Oddział Gdański PTETiS Referat nr 16*. Katedra Systemów Multimedialnych, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 2009.
- [11] B. Kunka. System śledzenia punktu fiksacji wzroku na monitorze komputera. Katedra Systemów Multimedialnych, Politechnika Gdańska, 2013. Wykład.

- [12] T. E. Hutchinson, K. P. White, W.N. Martin, K. C. Reichert, and L. A. Frey. Human- computer interaction using eye-gaze input. In *IEEE Transactions On Systems, Man, And Cybernetics*, Vol. 19, No. 6, November/December 1989.
- [13] C++ pliki nagłówkowe. <http://www.cplusplus.com/forum/articles/10627/>. Data dostępu: 2017-11-28.
- [14] About qt. http://wiki.qt.io/About_Qt. Data dostępu: 2017-11-11.
- [15] Language bindings. http://wiki.qt.io/Language_Bindings. Data dostępu: 2017-11-11.
- [16] Trie tree - definicja national institute of standards and technology. <https://xlinux.nist.gov/dads/HTML/trie.html>. Data dostępu: 2017-11-17.
- [17] D. J. Wetherall. A. S. Tanenbaum. Computer networks. Pearson, 2003. Edycja 5.
- [18] Google custom searchengine. <https://developers.google.com/custom-search/docs/overview>. Data dostępu: 2017-11-20.
- [19] Google custom searchengine json resposne. <https://developers.google.com/custom-search/json-api/v1/reference/cse/list#response>. Data dostępu: 2017-11-27.
- [20] Strona źródłowa dla pobranych ikon wykorzystanych w pracy. <https://www.flaticon.com/>. Data dostępu: 2017-11-06.
- [21] P. Fenrich. *Practical Principles of Instructional Design, Media Selection, and Interface Design with a Focus on Computer-based Training/Educational Software*. British Columbia Institute of Technology, 2014. str. 281.
- [22] Strona źródłowa dla kombinacji kolorystycznych dobranych w interfejsie użytkownika. <http://colorpalettes.net/category/contrasting-color/>. Data dostępu: 2017-11-14.